A New Toolbox *for* Scheduling Theory

Ziv Scully

Harvard & MIT → Cornell

zivscully@cornell.edu
https://ziv.codes

Collaborators

Mor Harchol-Balter

CMU, thesis advisor

Guy Blelloch

CMU, thesis advisor

Alan Scheller-Wolf

CMU

Isaac Grosof

CMU

Adam Wierman
Caltech

Onno Boxma
TU/e (Eindhoven)

Jan-Pieter Dorsman
UvA (Amsterdam)

Lucas van Kreveld
UvA (Amsterdam)

Michael Mitzenmacher
Harvard

Sid Banerjee
Cornell

Anupam Gupta
CMU

Sahil Singla
Georgia Tech

Haotian Jiang
University of Washington

Kunhe Yang
UC Berkeley

Performance

Performance

efficiency *throughput* *delay* *reliability* *safety* *value*

Performance

efficiency *throughput* *delay* *reliability* *safety* *value*

Contention

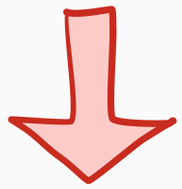
Queueing

Delay

healthcare

Contention

Queueing



Delay

supply chains

healthcare

Contention

Queueing

Delay

your local supermarket

supply chains

healthcare

Contention

Queueing

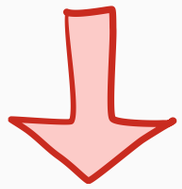
Delay

your local supermarket

supply chains **Contention** call centers

healthcare

Queueing



Delay

your local supermarket

supply chains **Contention** call centers

healthcare **Queueing** transportation

Delay

your local supermarket

supply chains

Contention

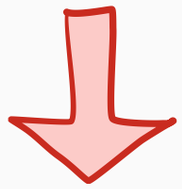
call centers

healthcare

transportation

Queueing

databases



Delay

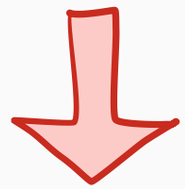
your local supermarket

supply chains

Contention

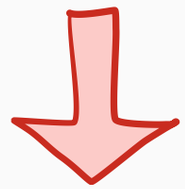
call centers

healthcare



transportation

Queueing



databases

Delay

networks

your local supermarket

supply chains **Contention** call centers

healthcare transportation

Queueing

databases

Delay networks

operating systems

your local supermarket

supply chains **Contention** call centers

healthcare **Queueing** transportation

databases

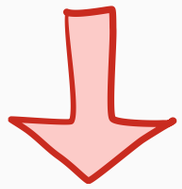
computer architecture **Delay** networks

operating systems

your local supermarket

supply chains **Contention** call centers

healthcare **Queueing** transportation



supercomputing databases

computer architecture **Delay** networks

operating systems

your local supermarket

supply chains **Contention** call centers

healthcare **Queueing** transportation

supercomputing databases

computer architecture **Delay** networks

operating systems

How to reduce delays?

your local supermarket

supply chains **Contention** call centers

healthcare **Queueing** transportation

supercomputing databases

computer architecture **Delay** networks

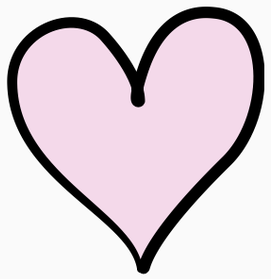
operating systems

How to reduce delays?

Scheduling

Good news:

scheduling can reduce delay



Good news:

scheduling can reduce delay

Bad news:

limited understanding of scheduling

Good news:

scheduling can reduce delay

Bad news:

limited understanding of scheduling

evaluation

Good news:

scheduling can reduce delay

Bad news:

limited understanding of scheduling

design

evaluation

Good news:

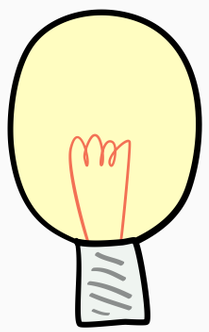
scheduling can reduce delay

Bad news:

limited understanding of scheduling

design

evaluation



We need:

rigorous theory of scheduling

Good news:

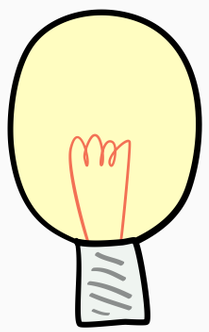
scheduling can reduce delay

Bad news:

limited understanding of scheduling

design

evaluation



We need:

rigorous theory of scheduling

CS, EE, OR, OM

Good news:

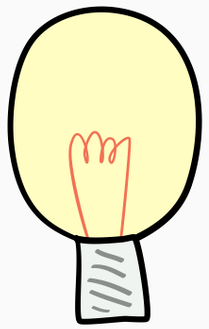
scheduling can reduce delay

Bad news:

limited understanding of scheduling

design

evaluation



We need:

rigorous theory of scheduling

applied math, probability, control, learning theory

CS, EE, OR, OM

Theory lags behind practice

Theory lags behind practice

Uncertainty:
unknown job sizes, noisy estimates

Theory lags behind practice

Uncertainty:
unknown job sizes, noisy estimates

Multiple servers:
even FCFS hard to understand

Theory lags behind practice

Uncertainty:
unknown job sizes, noisy estimates

Multiple servers:
even FCFS hard to understand

Preemption practicalities:
limitations, overhead

Theory lags behind practice

This talk:
new theoretical tools

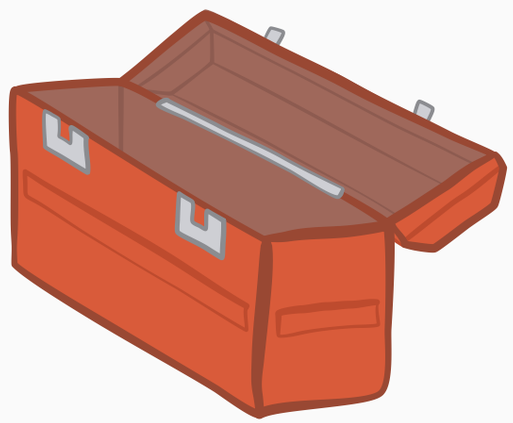
Uncertainty:
unknown job sizes, noisy estimates

Multiple servers:
even FCFS hard to understand

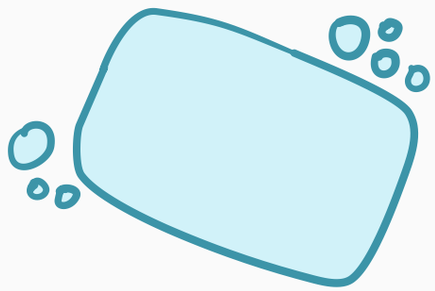
Preemption practicalities:
limitations, overhead

Outline: two new tools

Outline: two new tools

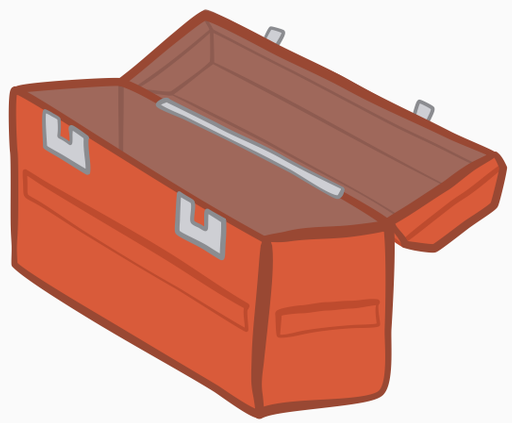


Outline: two new tools

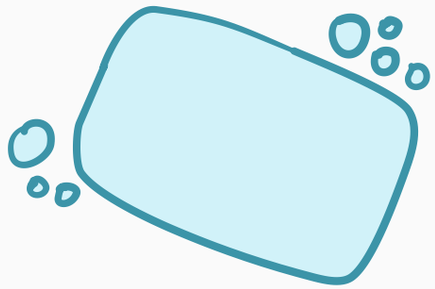


SOAP

new unifying theory of
single-server scheduling



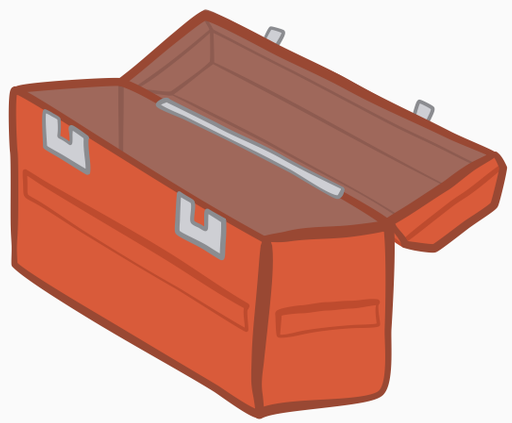
Outline: two new tools



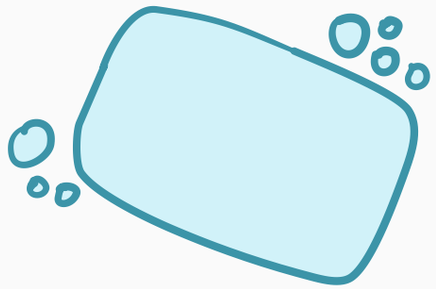
SOAP

greatly increases **number of policies** we can analyze

new unifying theory of single-server scheduling



Outline: two new tools

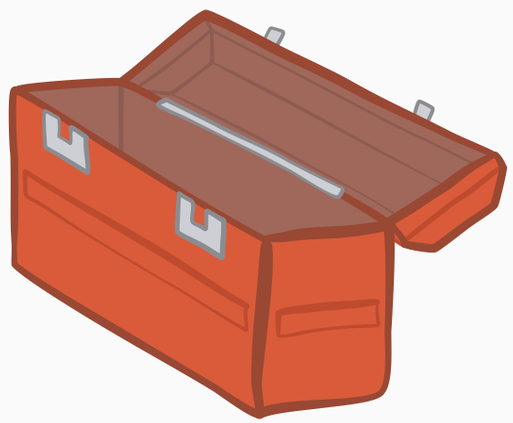


SOAP

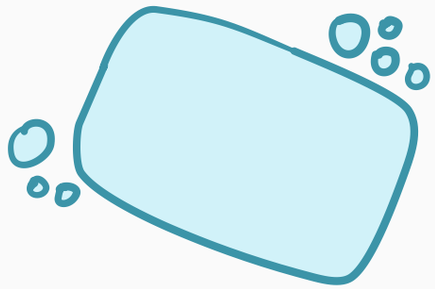
new unifying theory of single-server scheduling

greatly increases **number of policies** we can analyze

- unknown sizes
- preemption limitations



Outline: two new tools

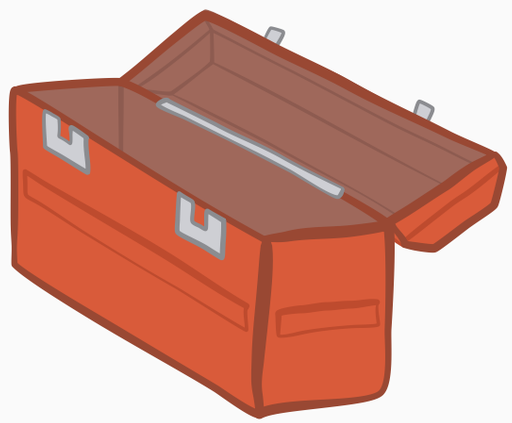


SOAP

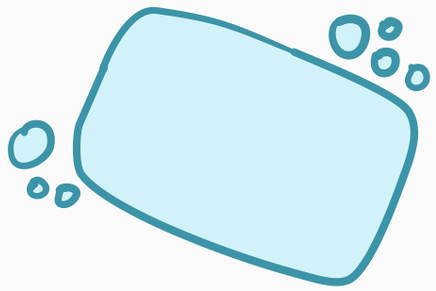
new unifying theory of single-server scheduling

unknown sizes

preemption limitations



Outline: two new tools



SOAP

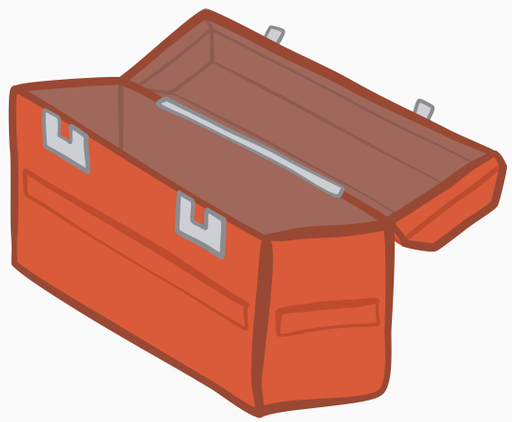
new unifying theory of single-server scheduling

→ unknown sizes

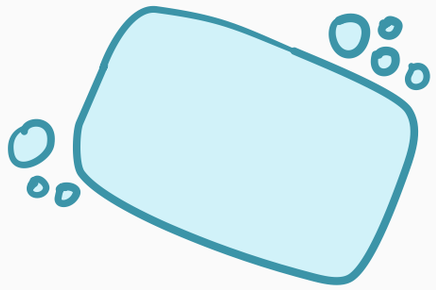
→ preemption limitations

WINE

new queueing identity to complement Little's Law



Outline: two new tools



SOAP

new unifying theory of single-server scheduling

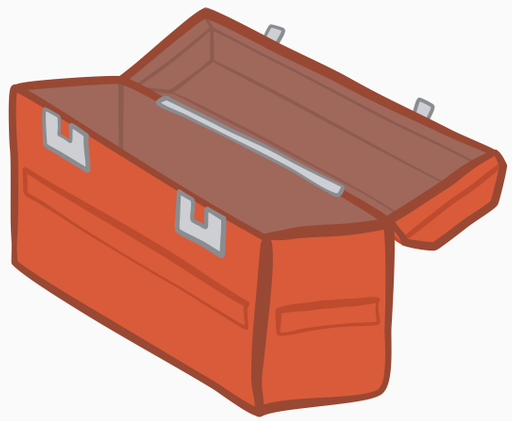
→ unknown sizes

→ preemption limitations

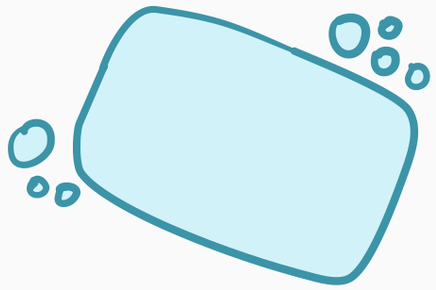
WINE

new queueing identity to complement Little's Law

reduces **complex systems** to related **simple systems**



Outline: two new tools



SOAP

new unifying theory of single-server scheduling

⇒ unknown sizes

⇒ preemption limitations

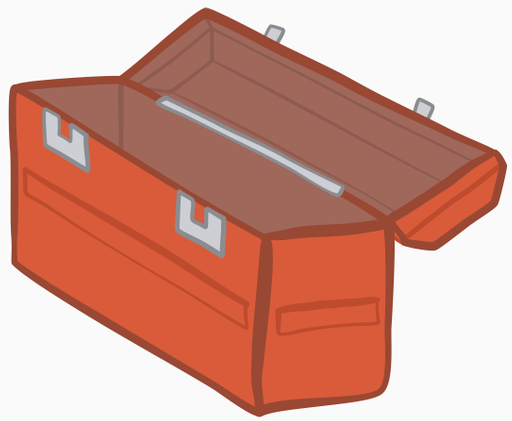
WINE

new queueing identity to complement Little's Law

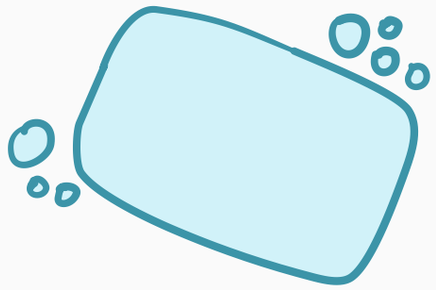
reduces **complex systems** to related **simple systems**

⇒ multiserver systems

⇒ noisy size estimates



Outline: two new tools



SOAP

new unifying theory of single-server scheduling

→ unknown sizes

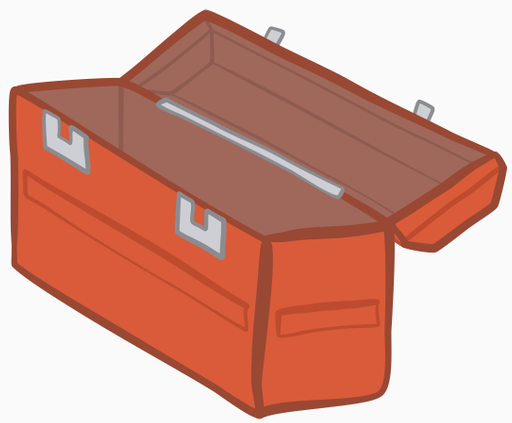
→ preemption limitations

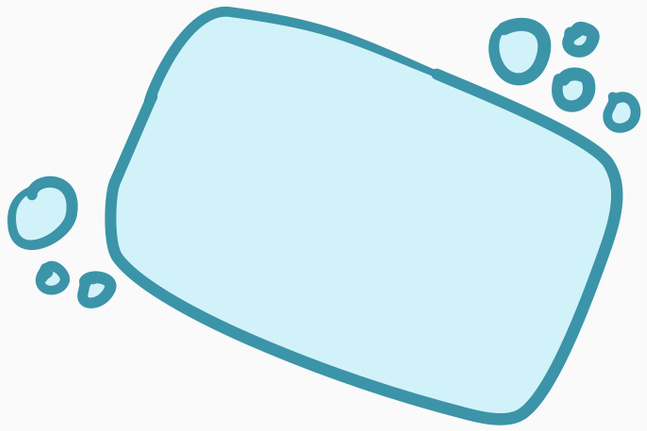
WINE

new queueing identity to complement Little's Law

→ multiserver systems

→ noisy size estimates

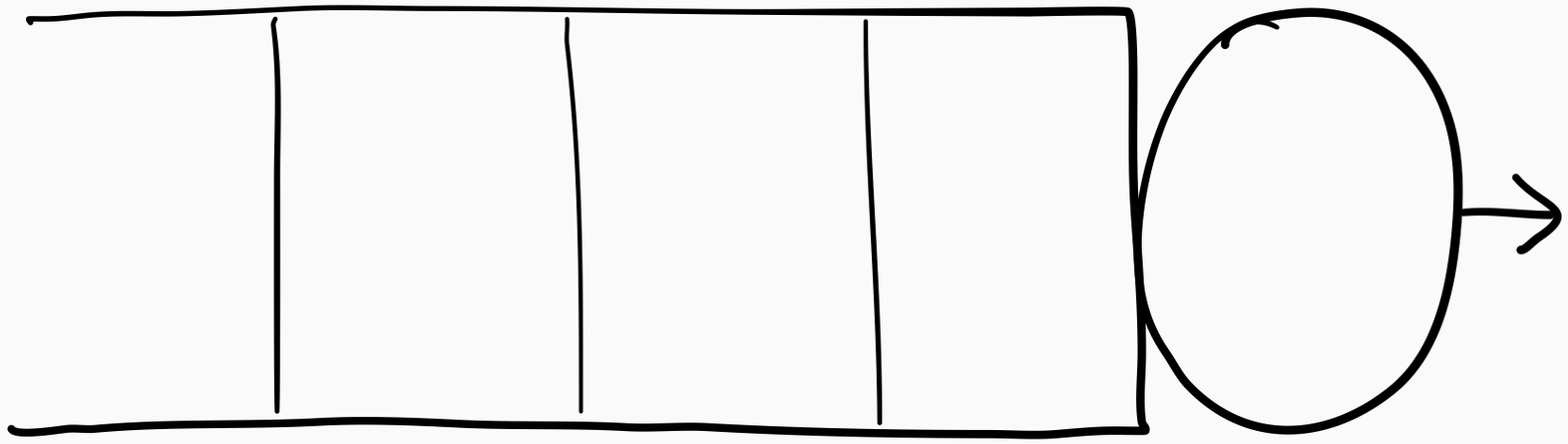




Part 1

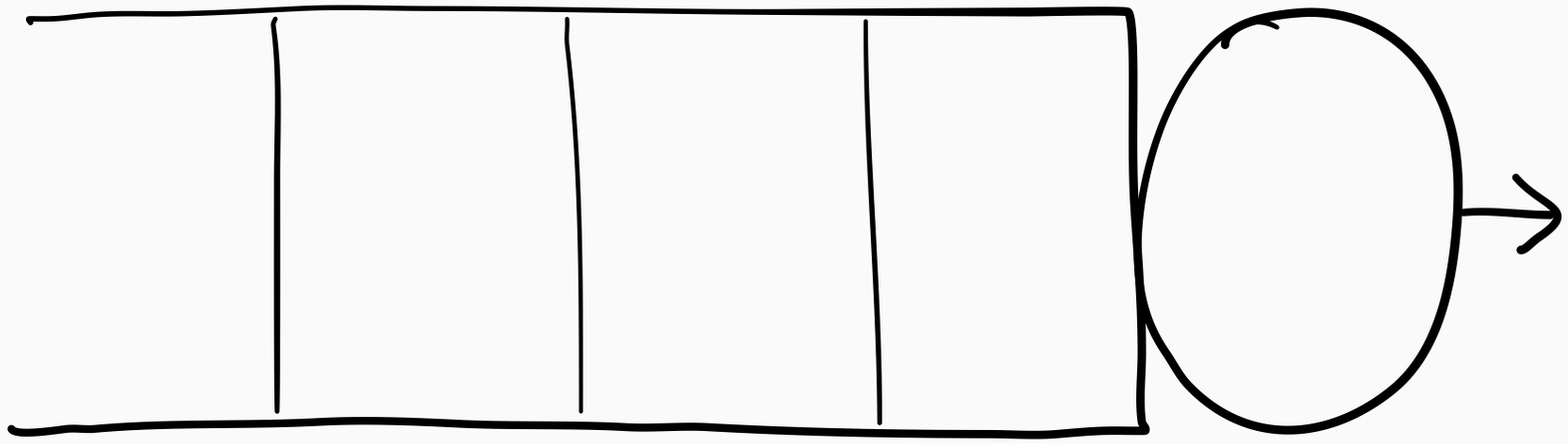
SOAP

Single-server queueing system



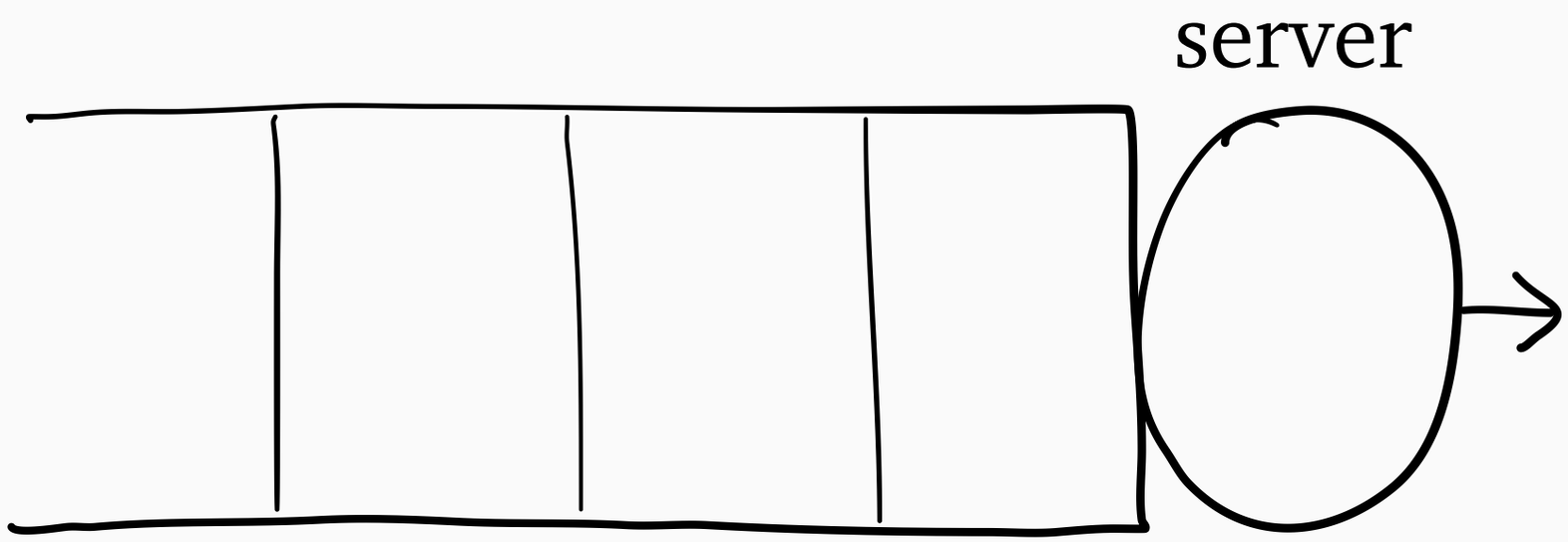
Single-server queueing system

M/G/1



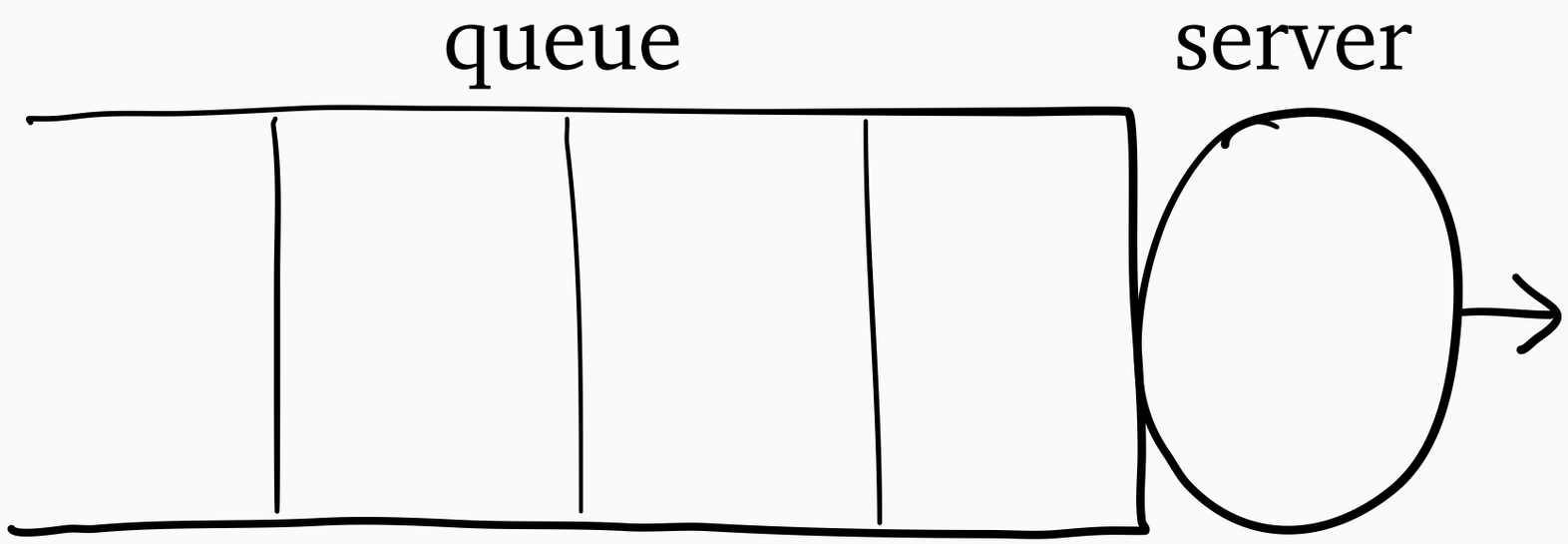
Single-server queueing system

M/G/1



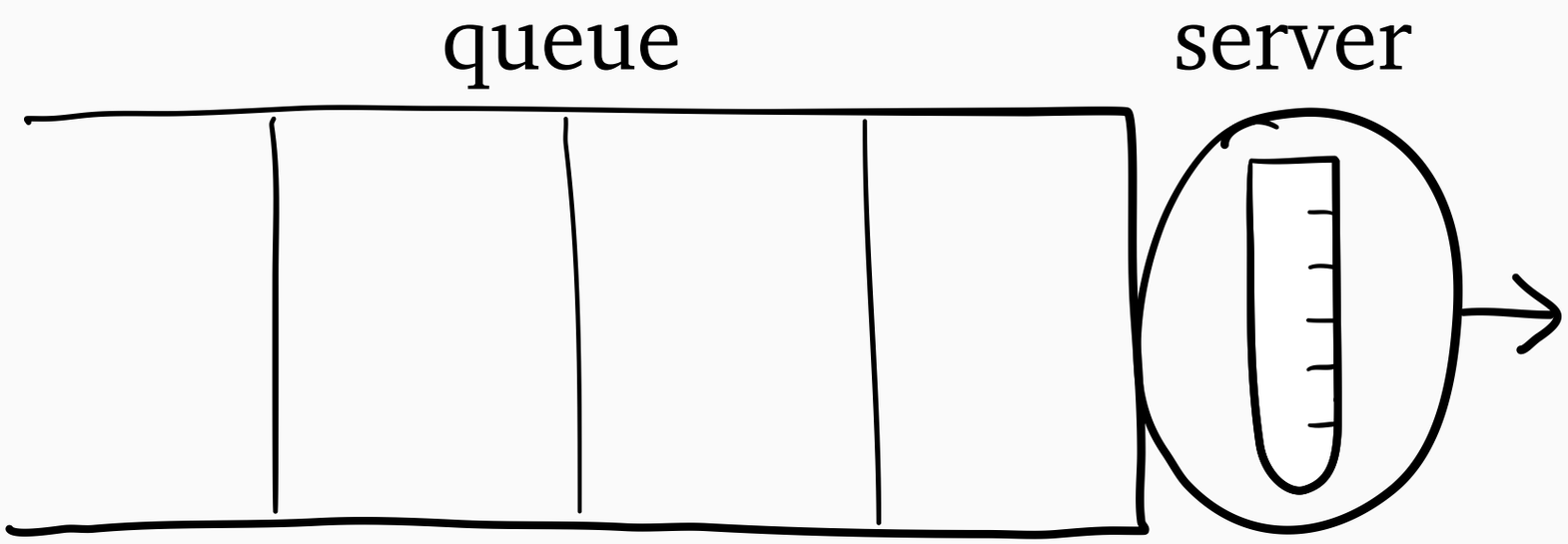
Single-server queueing system

M/G/1

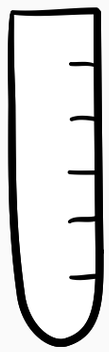


Single-server queueing system

M/G/1

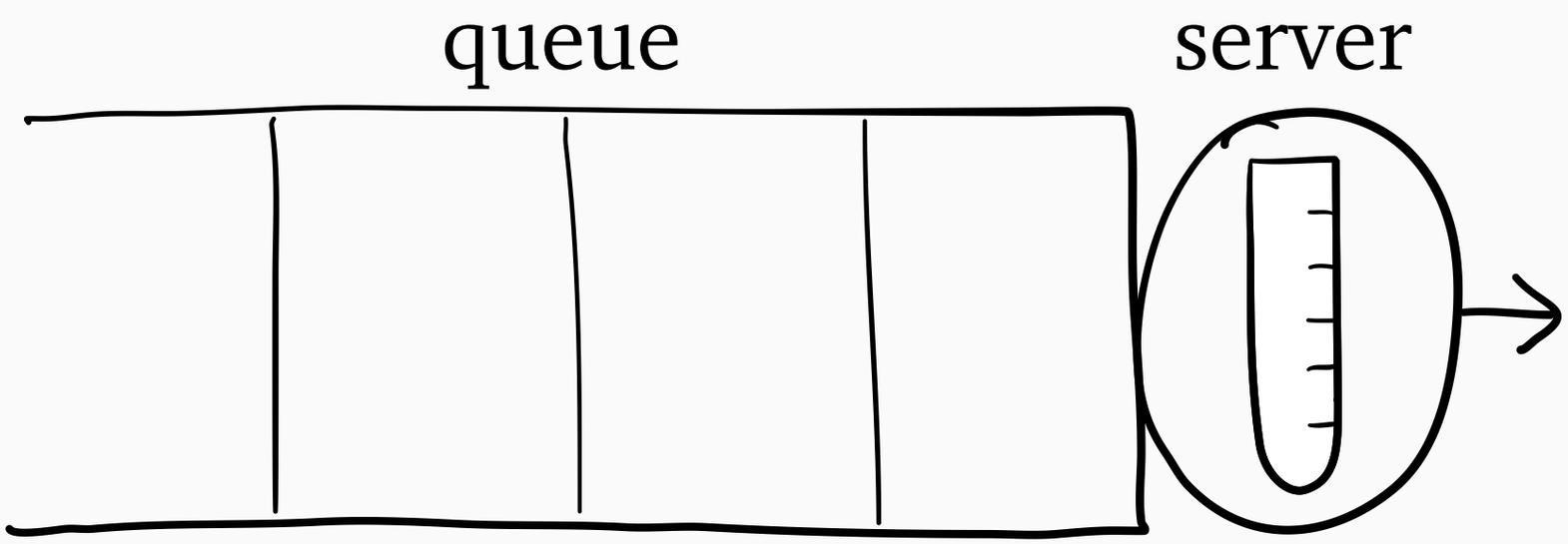


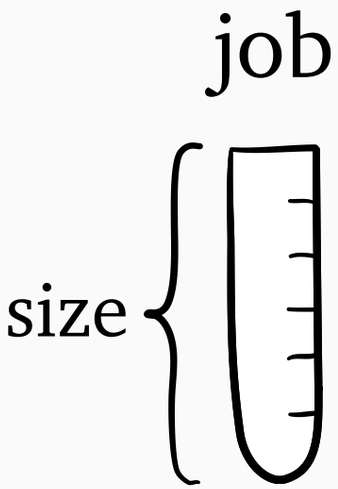
job



Single-server queueing system

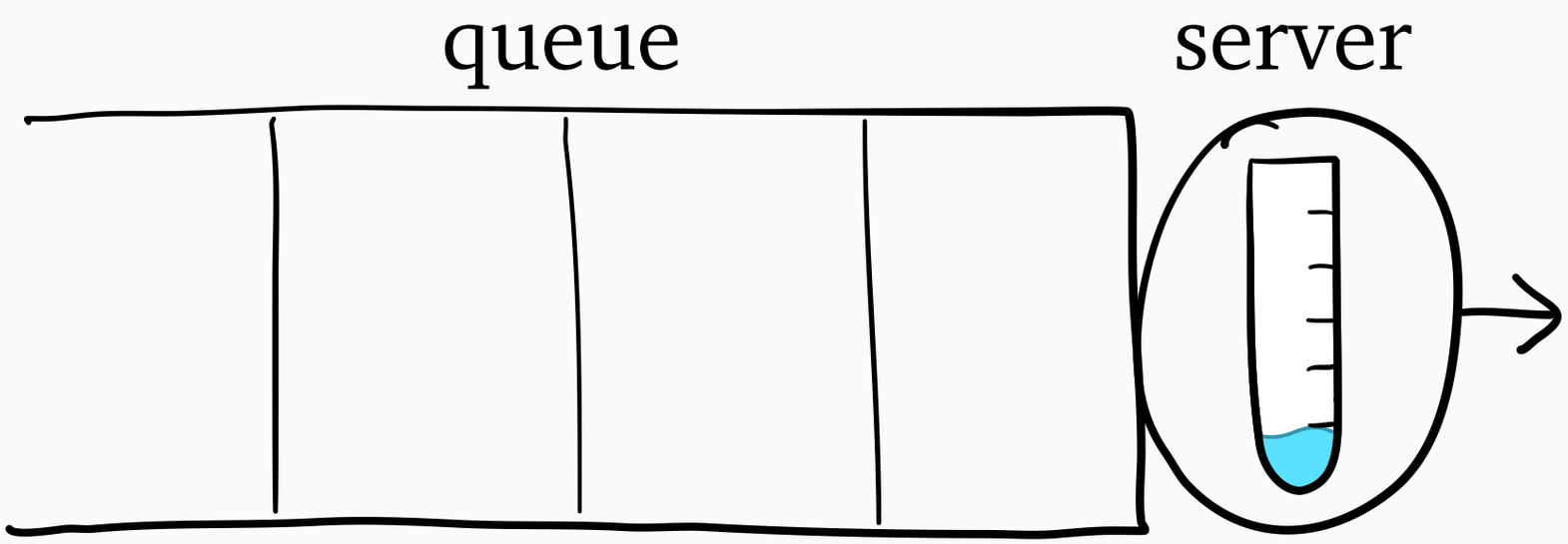
M/G/1

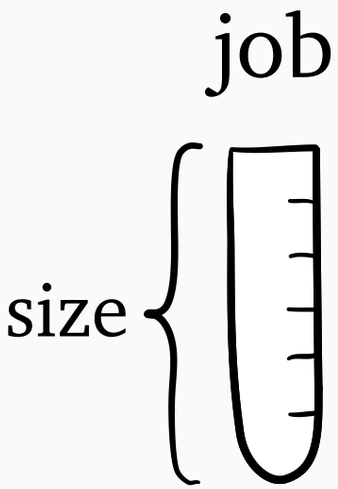




Single-server queueing system

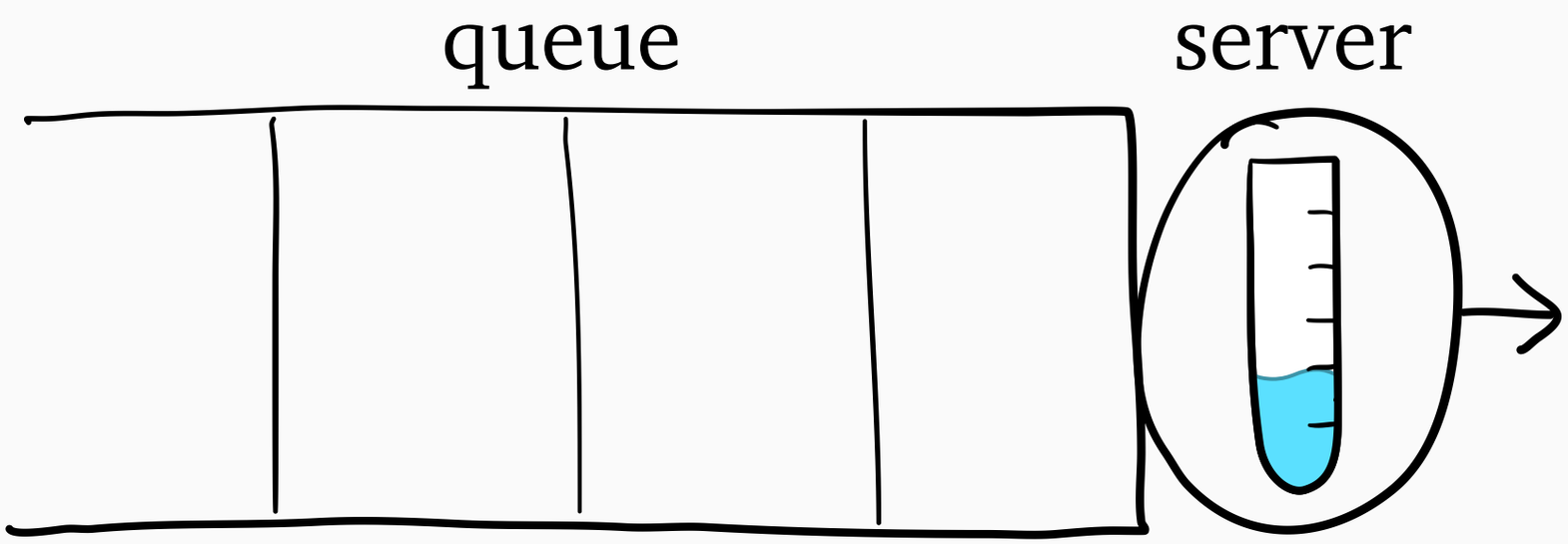
M/G/1

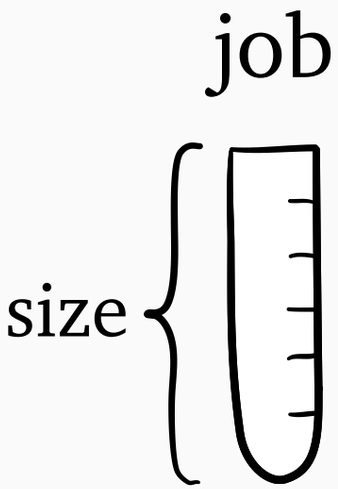




Single-server queueing system

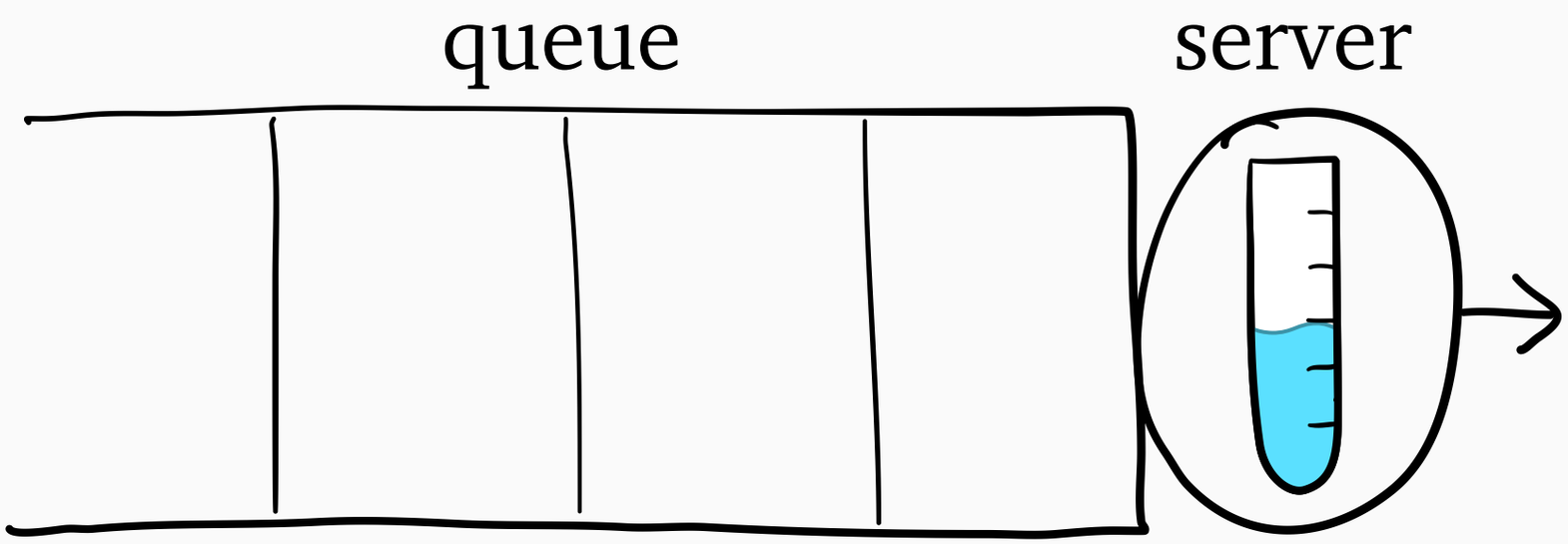
M/G/1

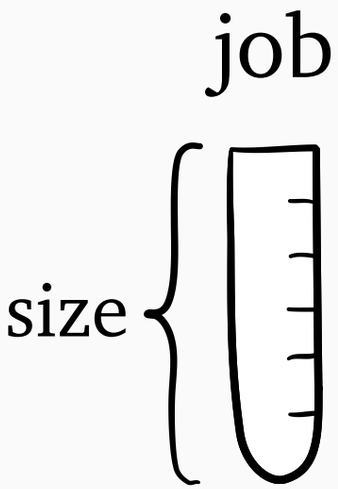




Single-server queueing system

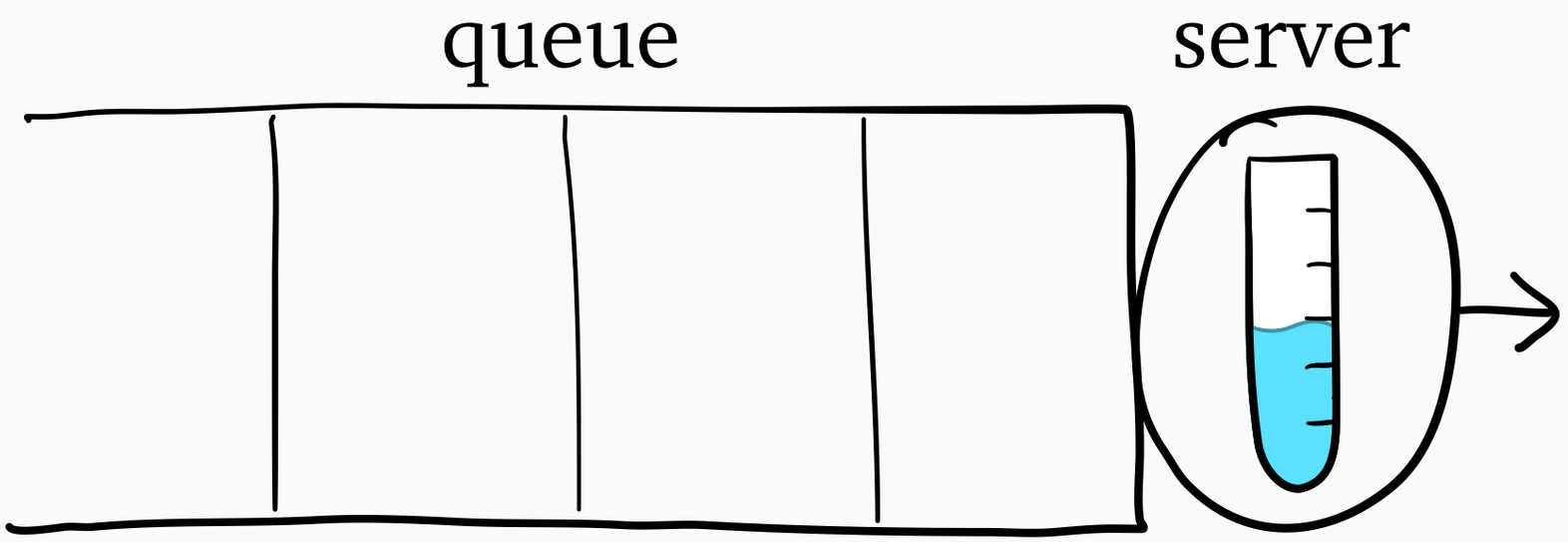
M/G/1

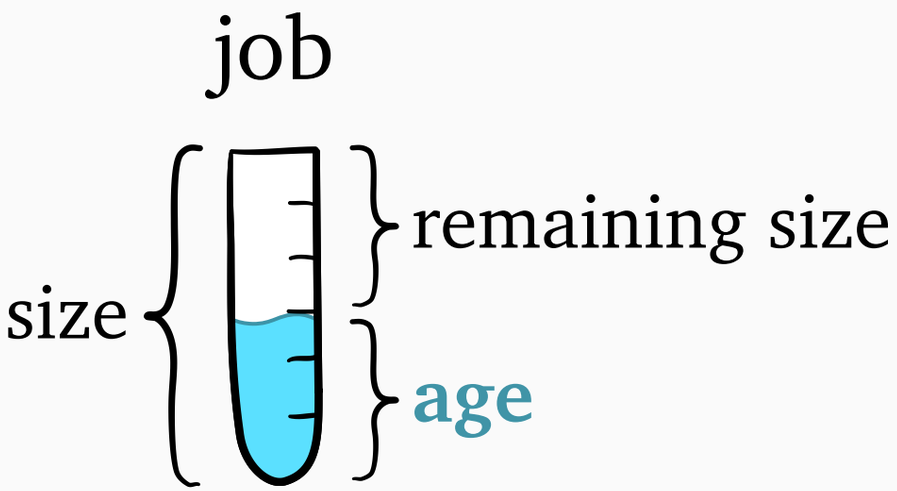




Single-server queueing system

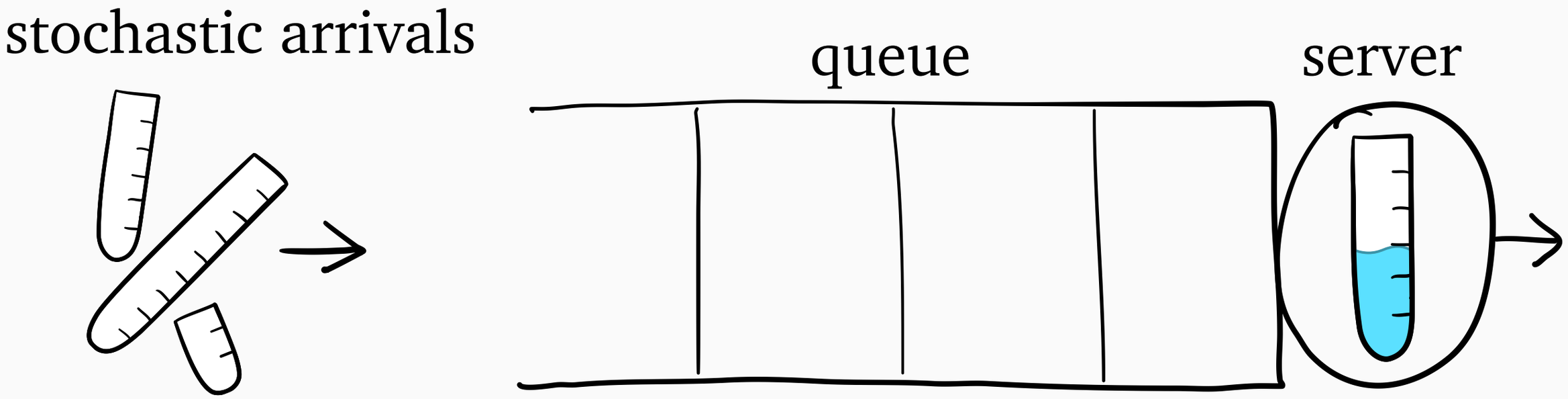
M/G/1

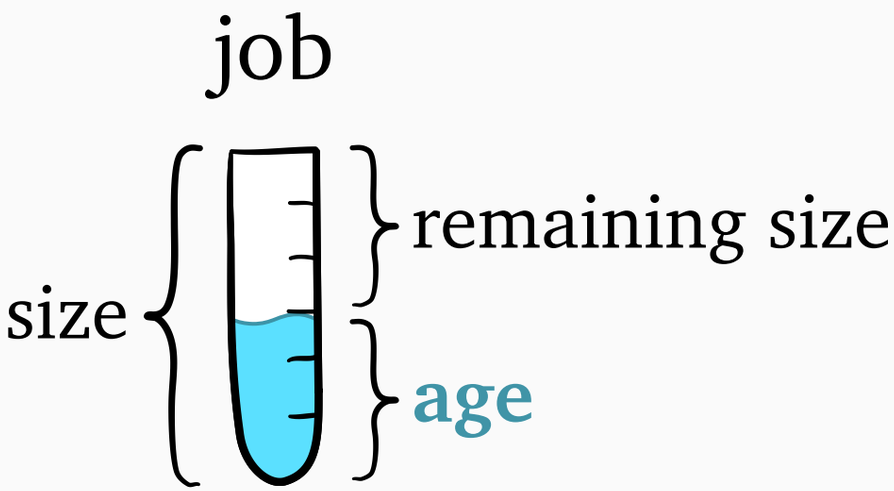




Single-server queueing system

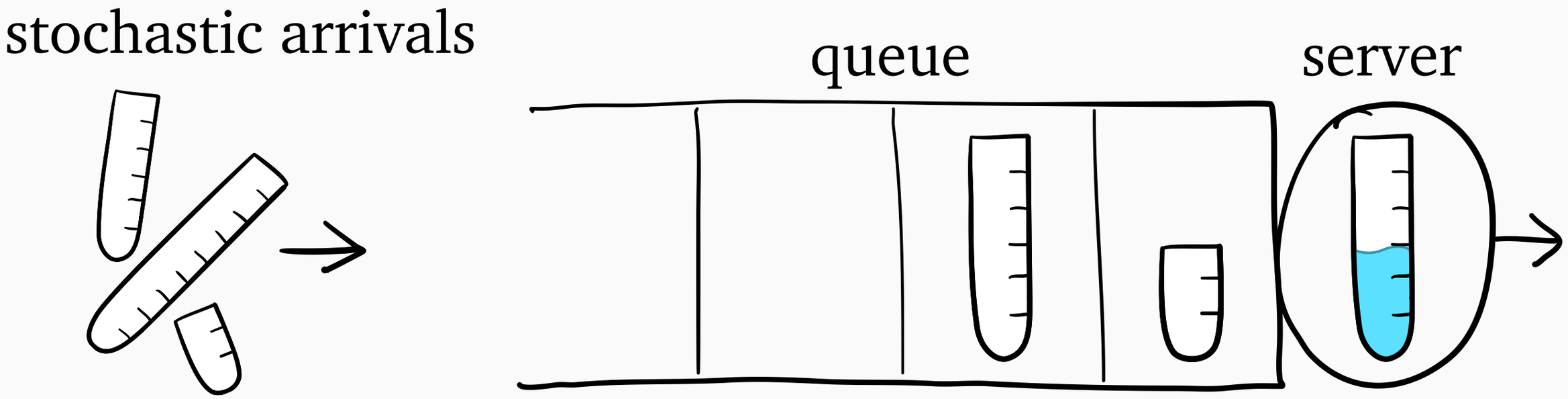
M/G/1

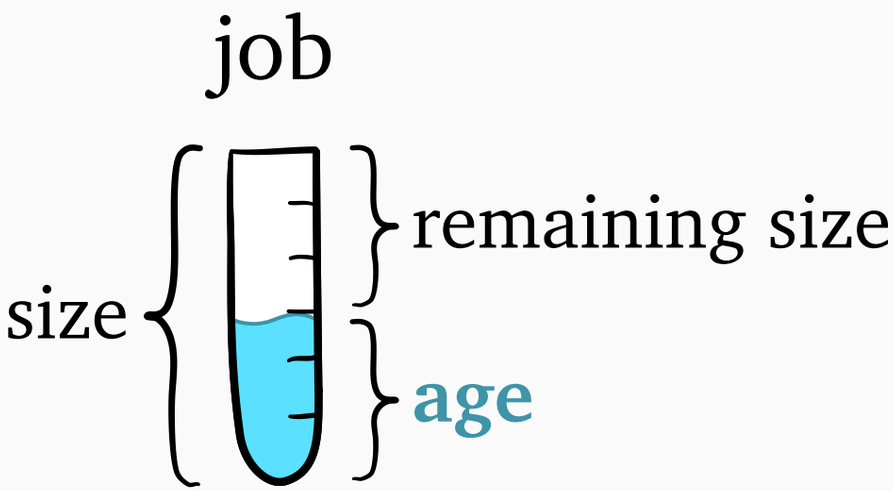




Single-server queueing system

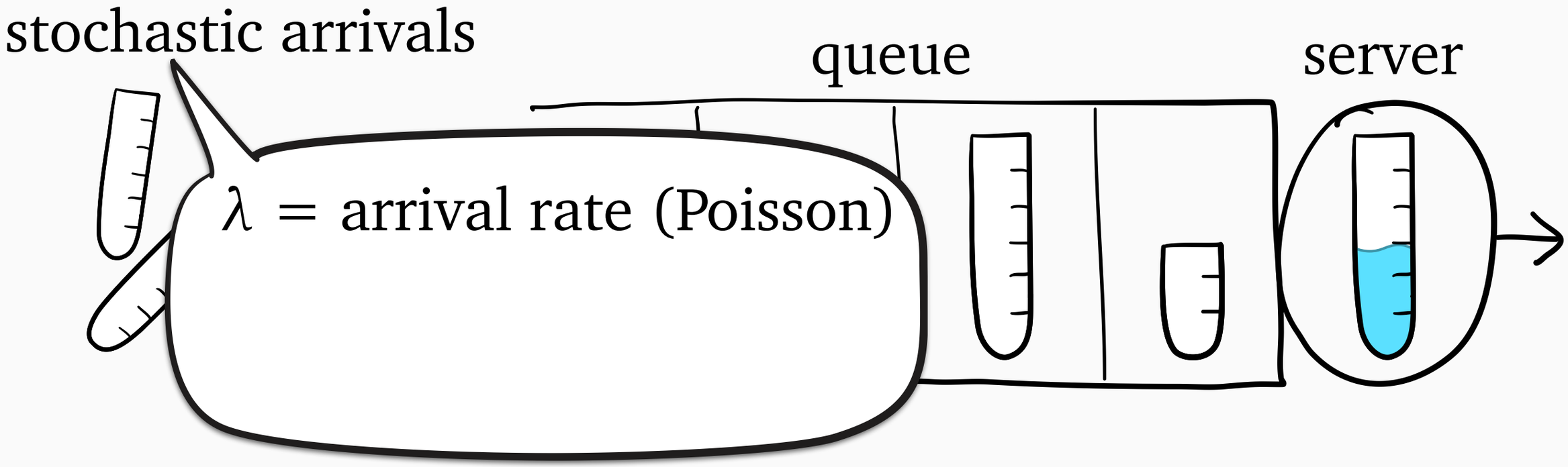
M/G/1

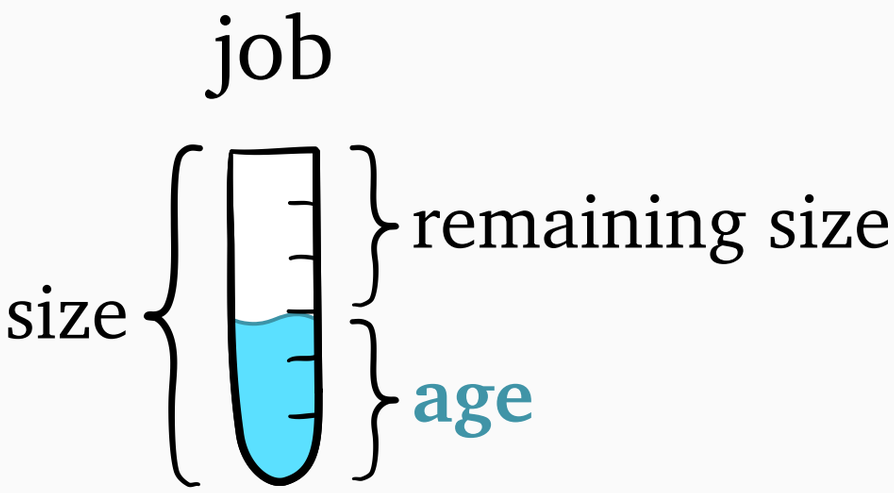




Single-server queueing system

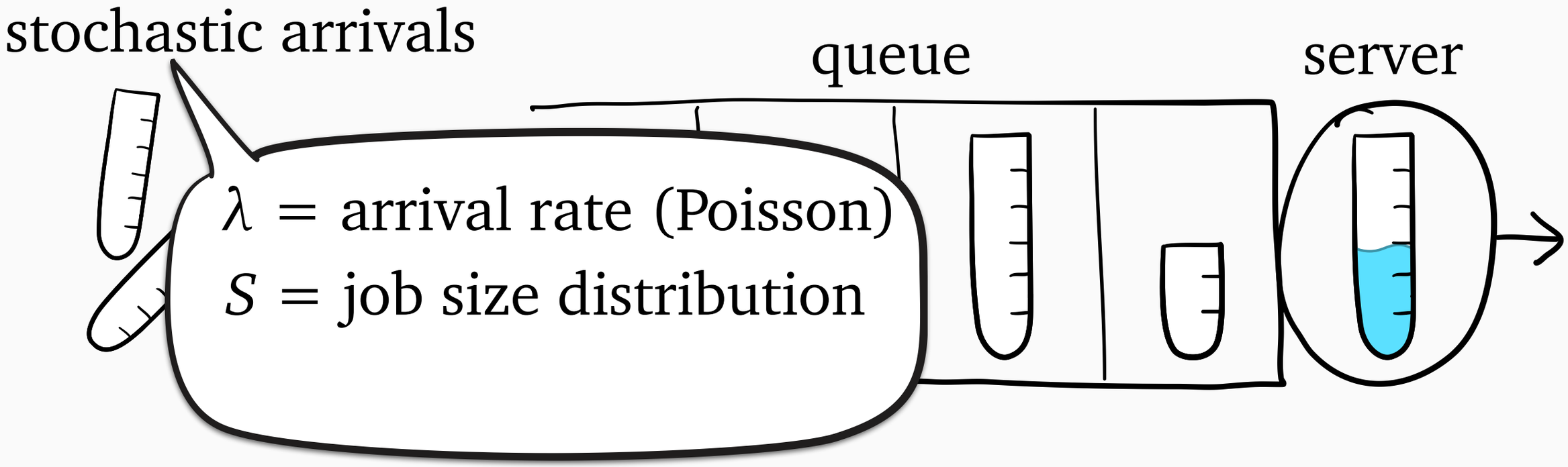
M/G/1

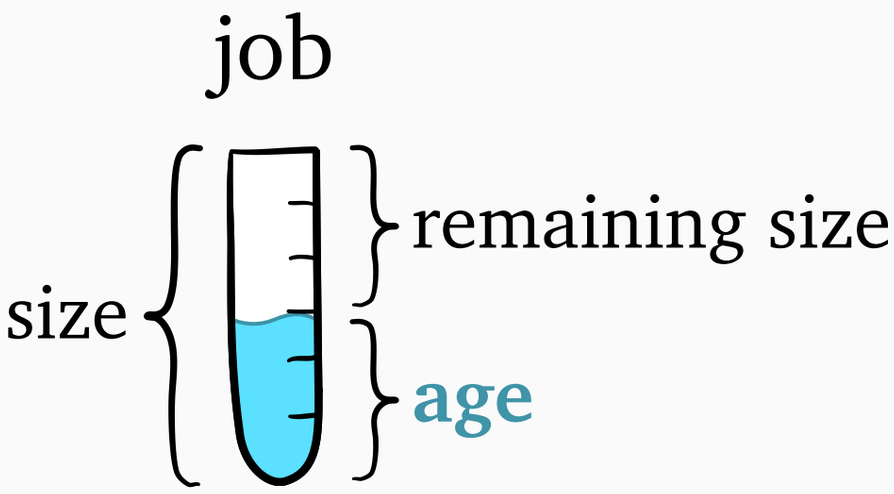




Single-server queueing system

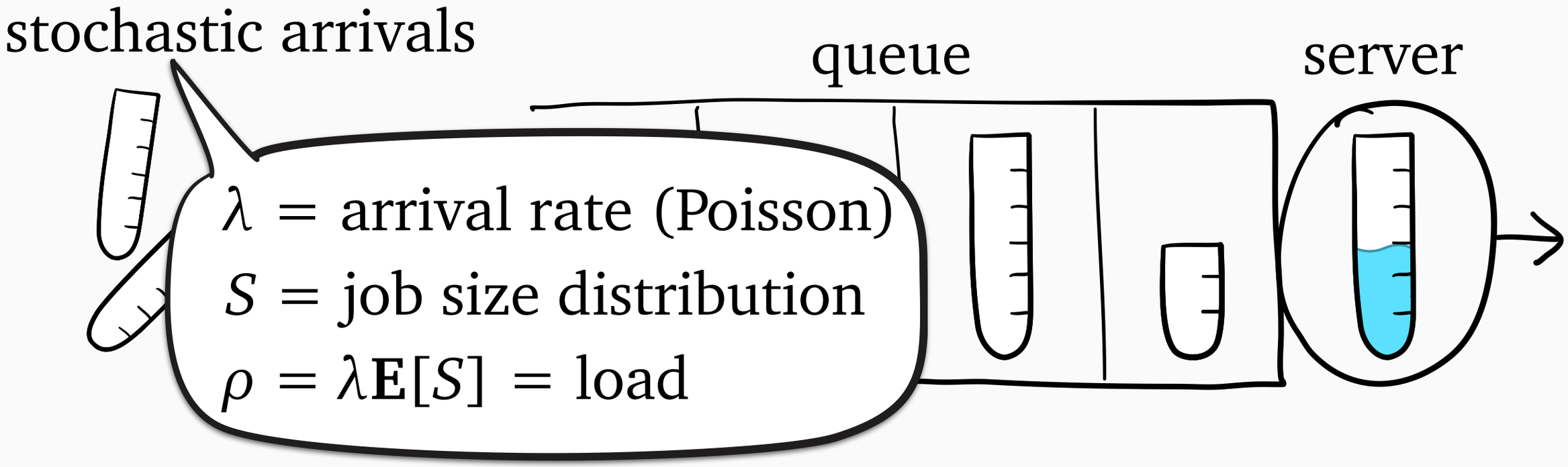
M/G/1

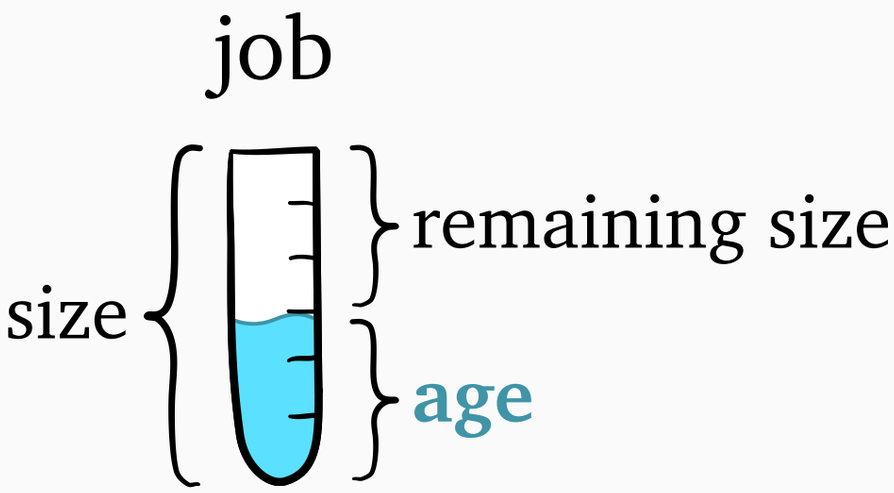




Single-server queueing system

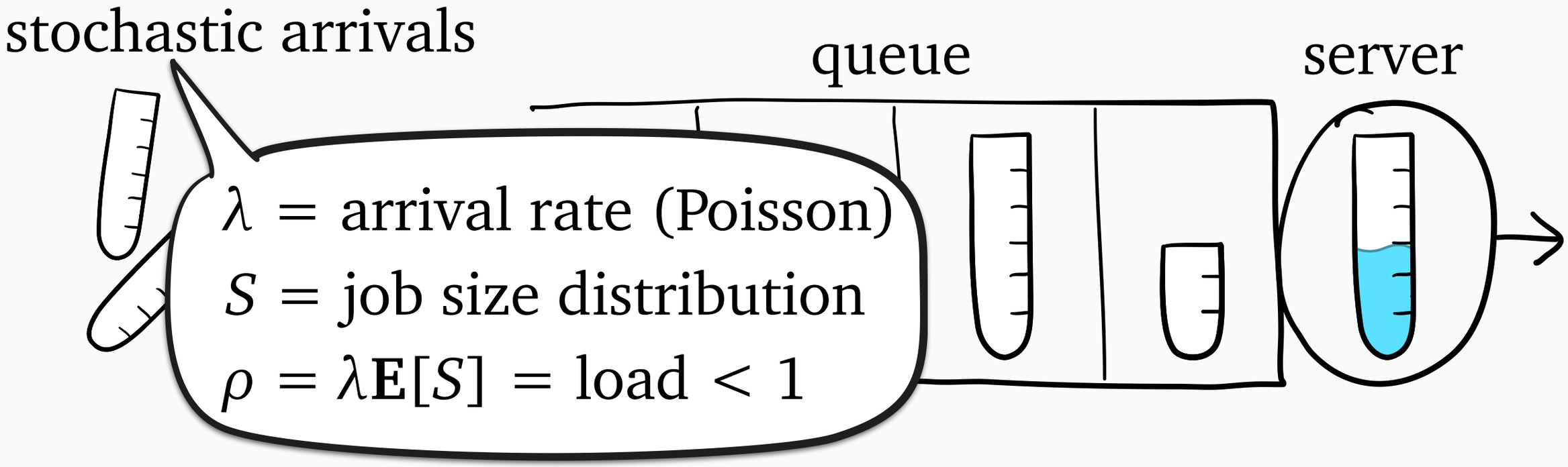
M/G/1

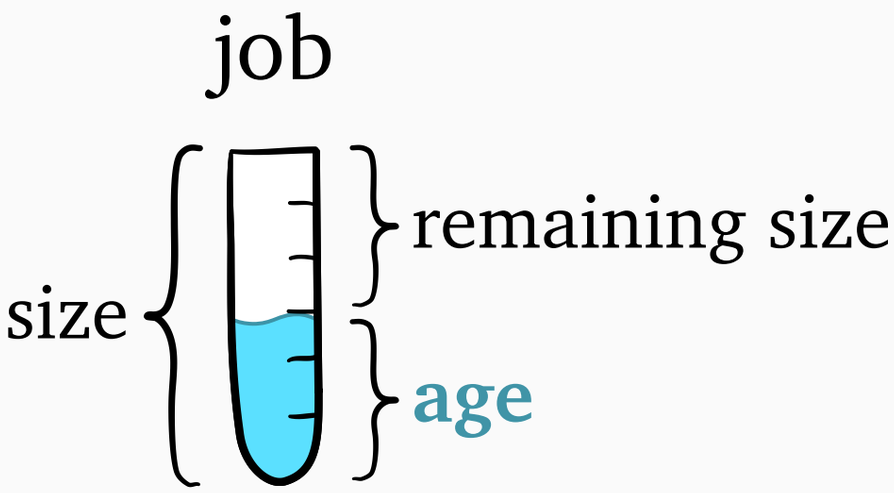




Single-server queueing system

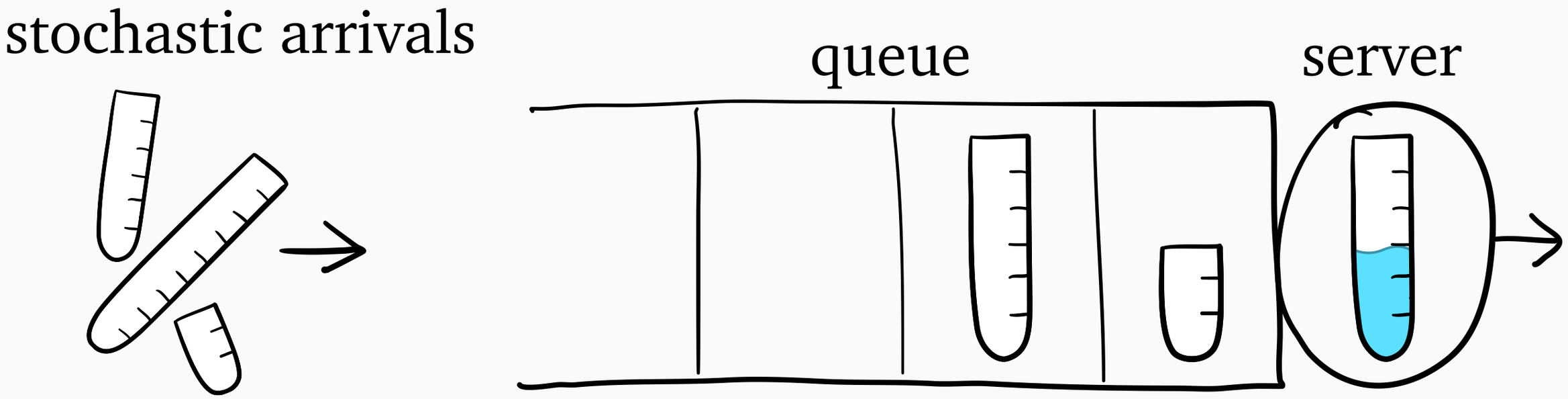
M/G/1

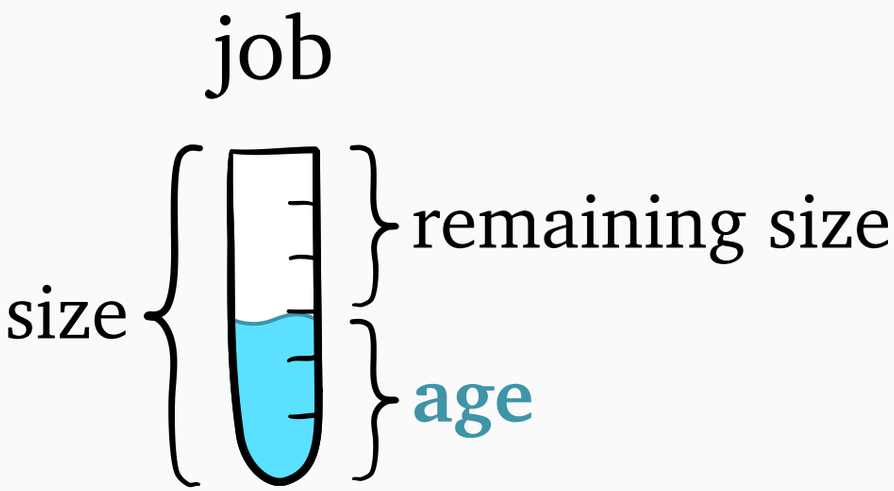




Single-server queueing system

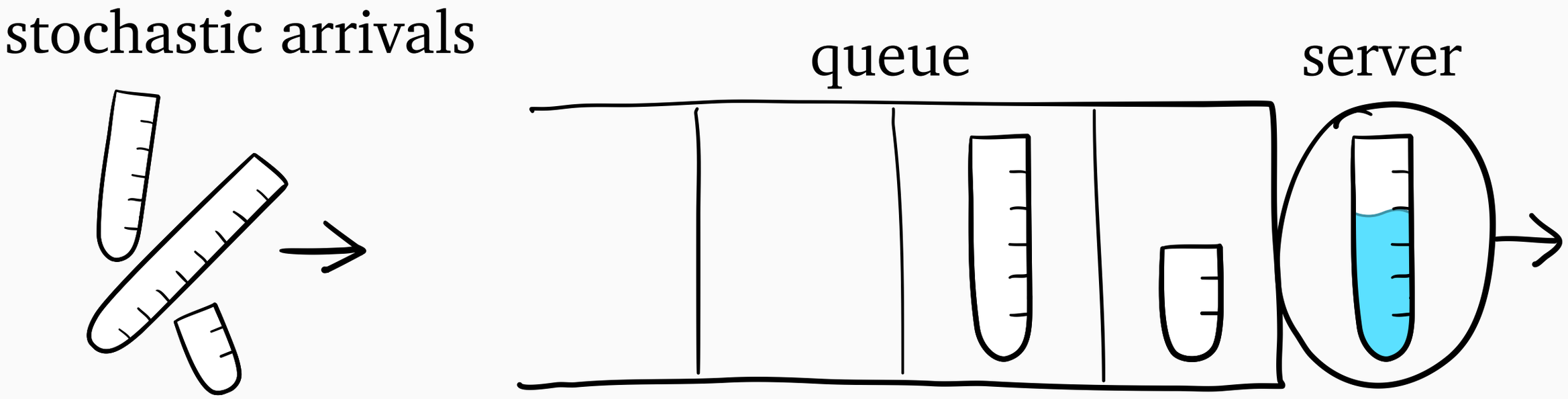
M/G/1

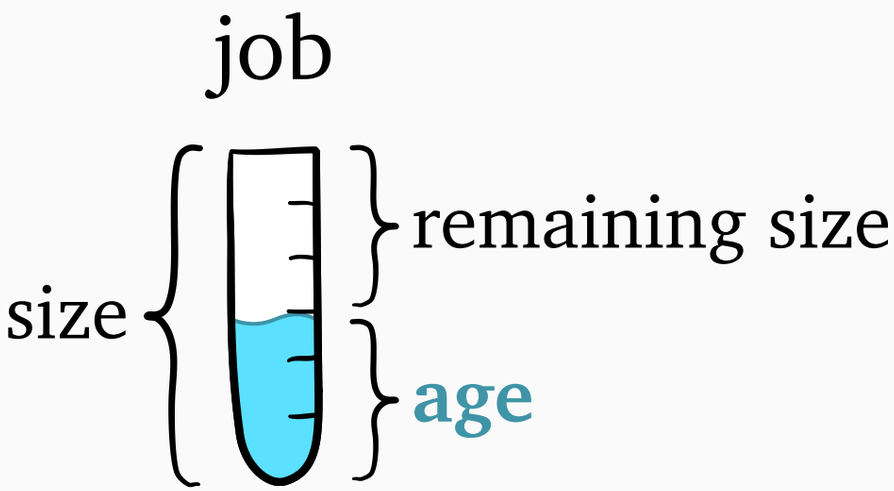




Single-server queueing system

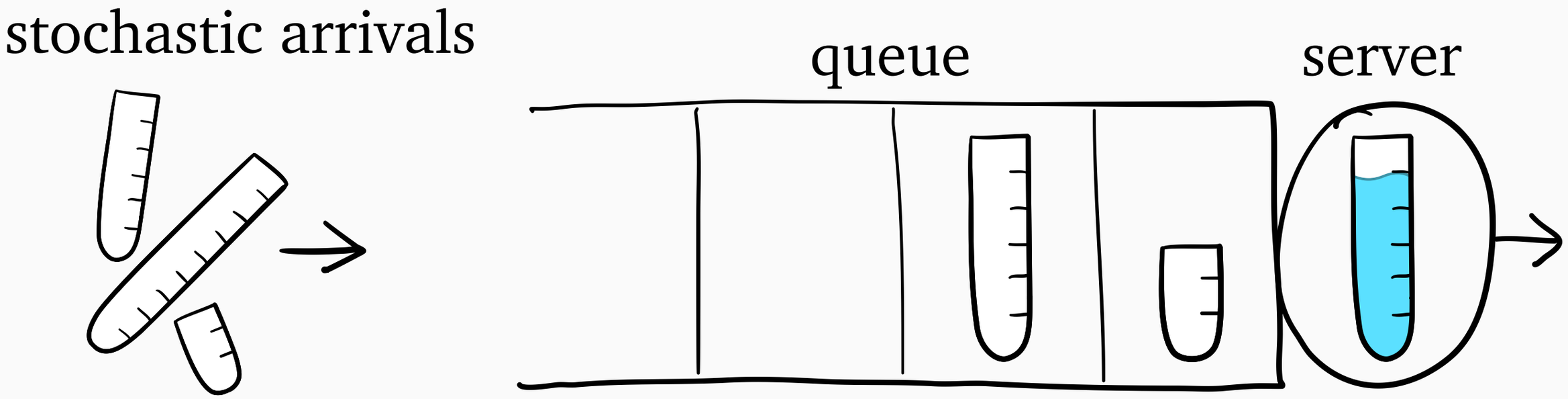
M/G/1

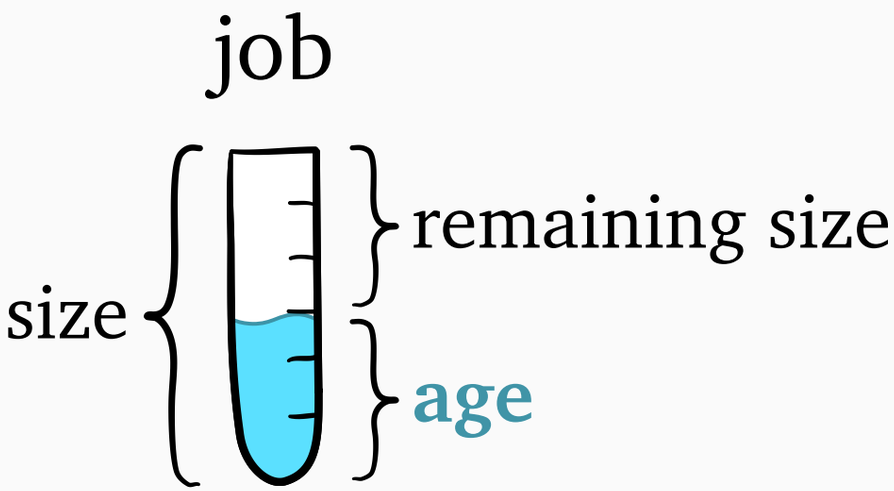




Single-server queueing system

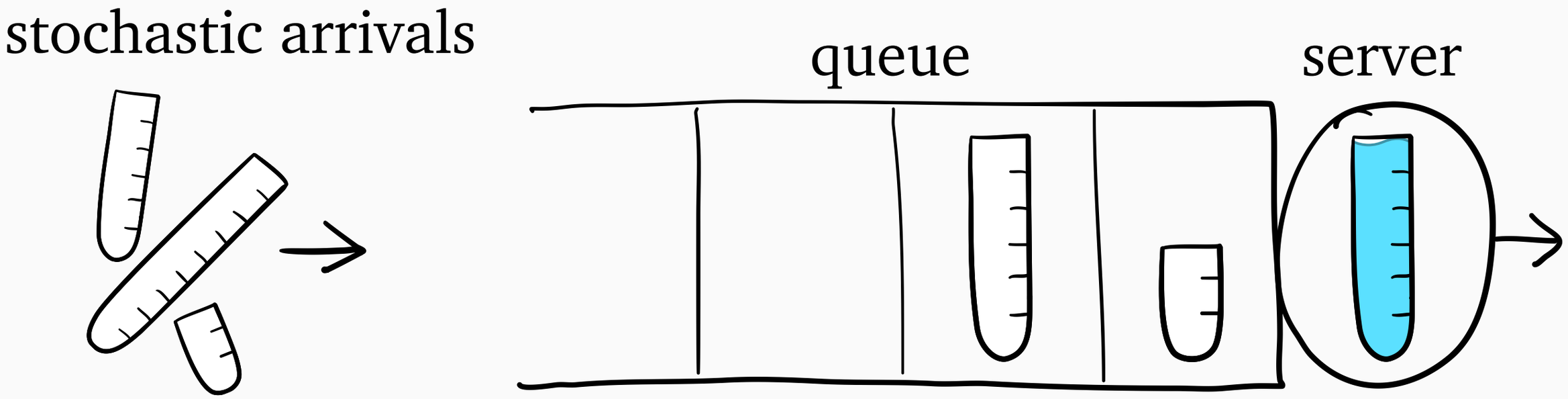
M/G/1

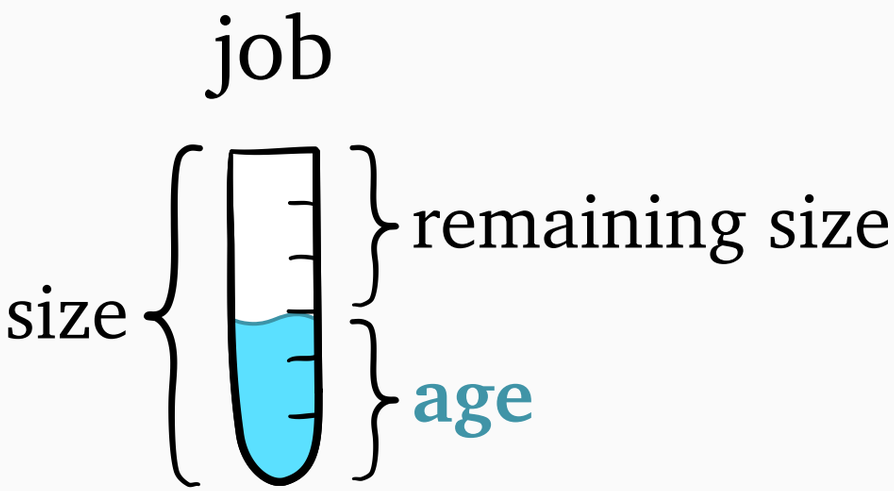




Single-server queueing system

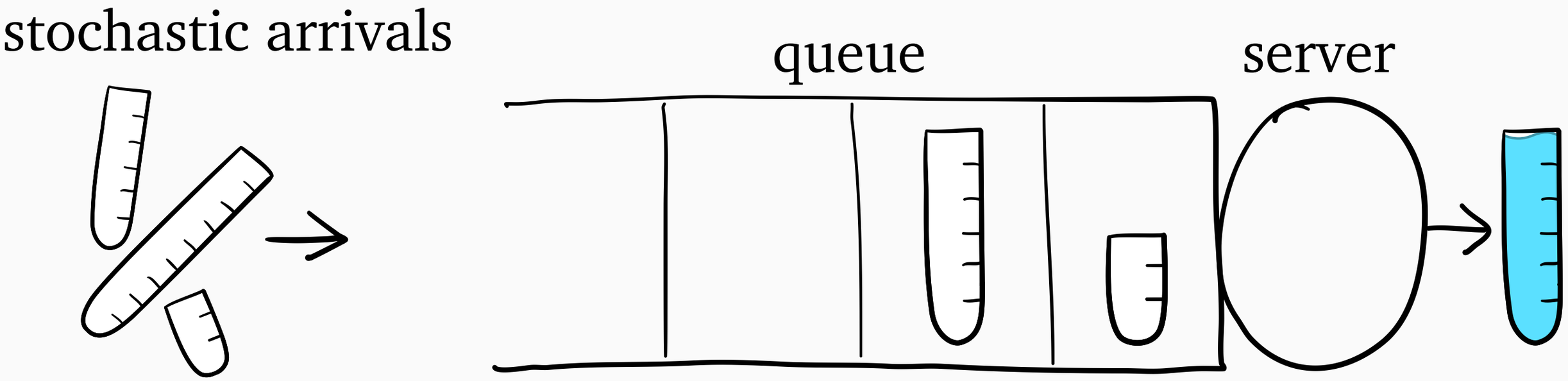
M/G/1

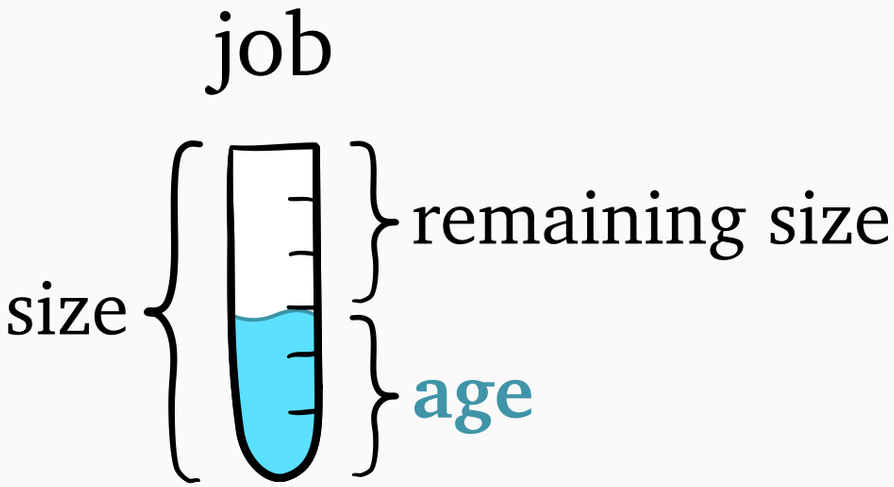




Single-server queueing system

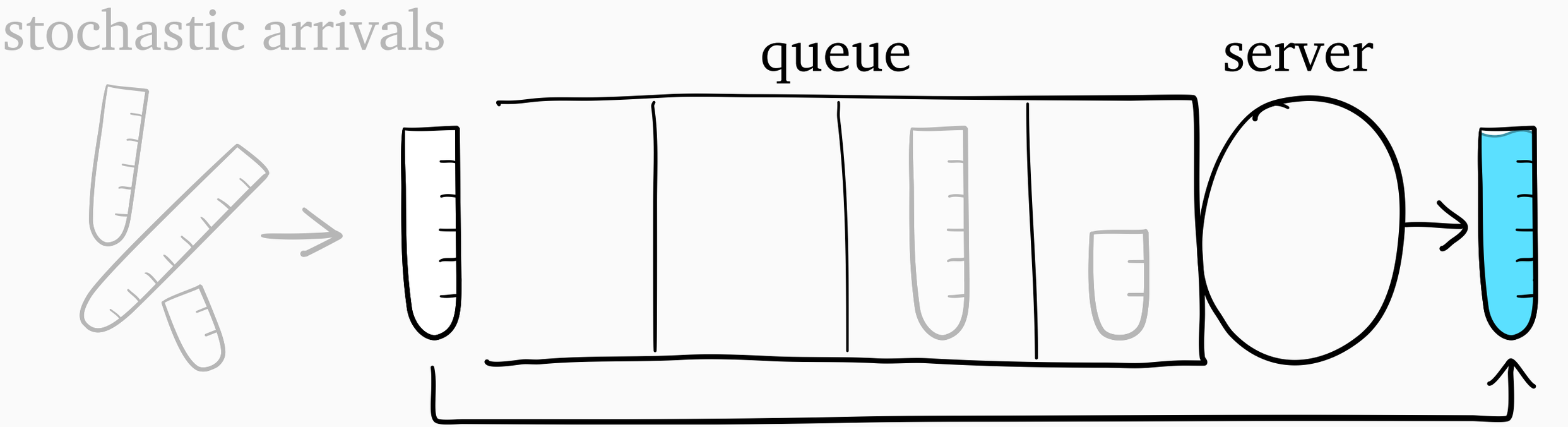
M/G/1



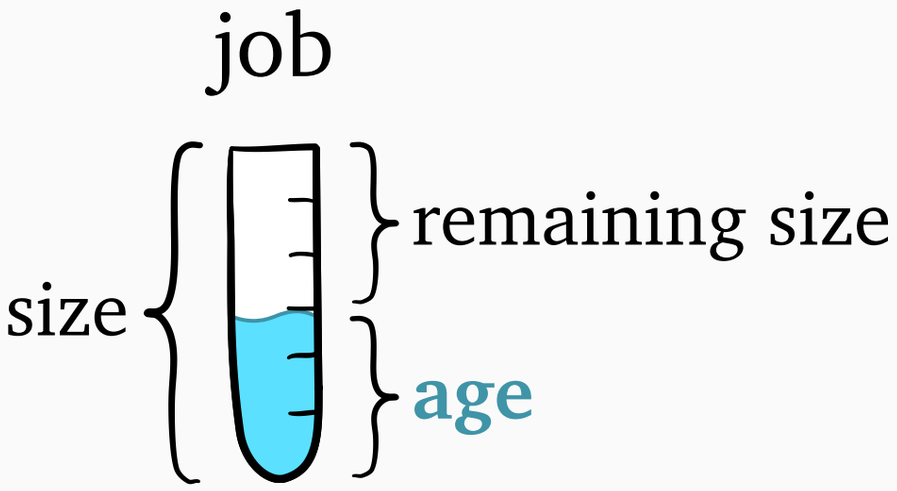


Single-server queueing system

M/G/1

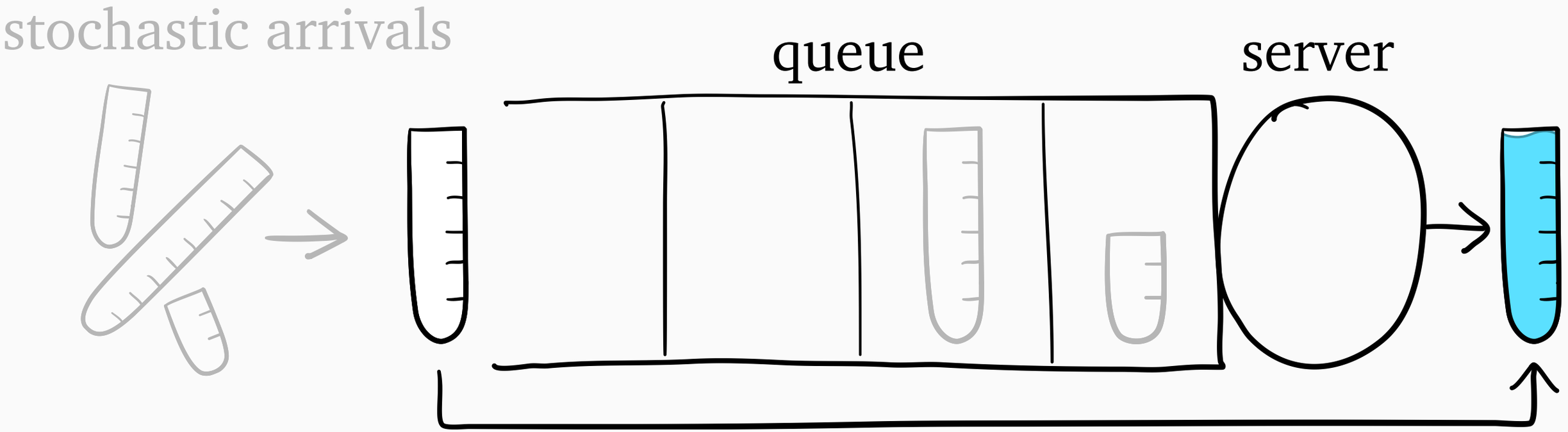


response time



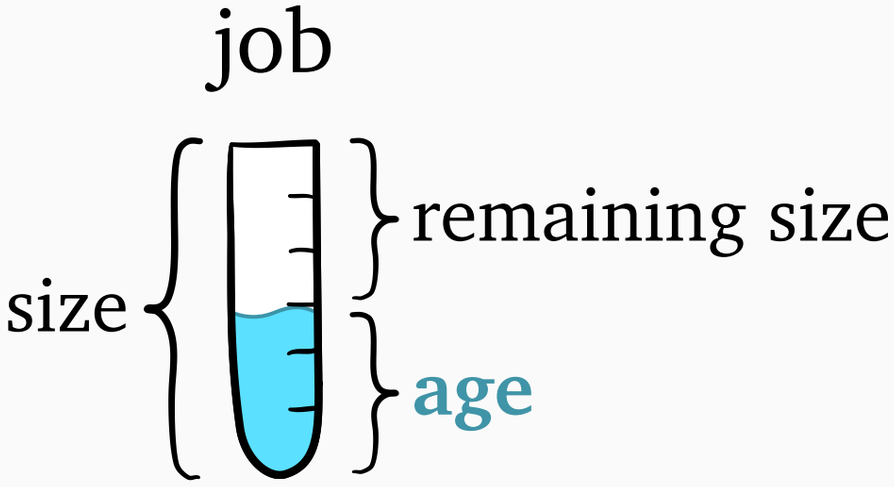
Single-server queueing system

M/G/1



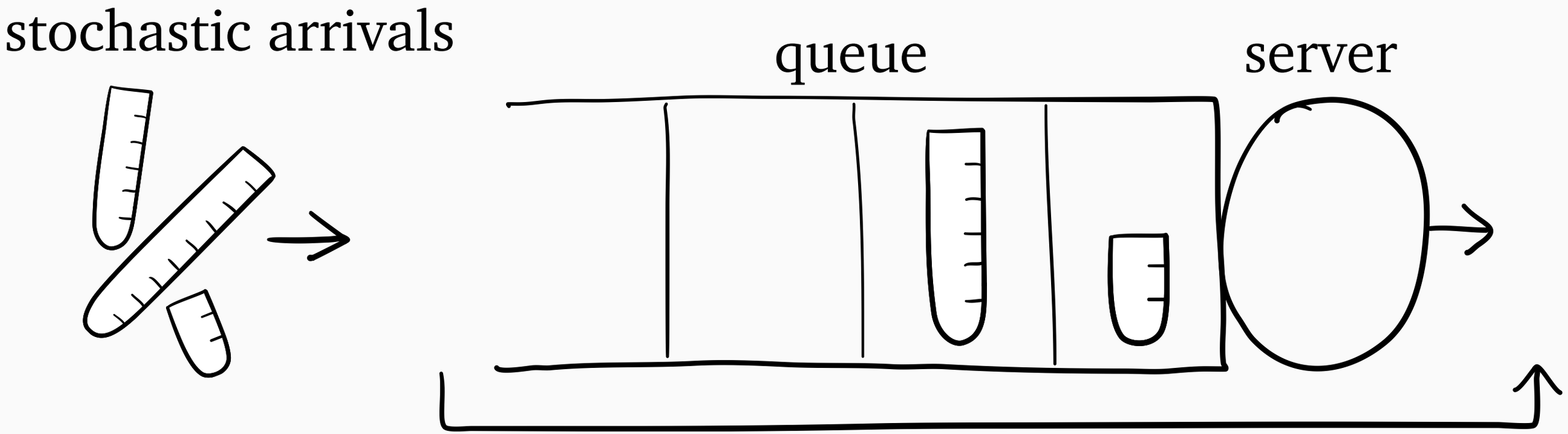
response time

a.k.a. delay, latency

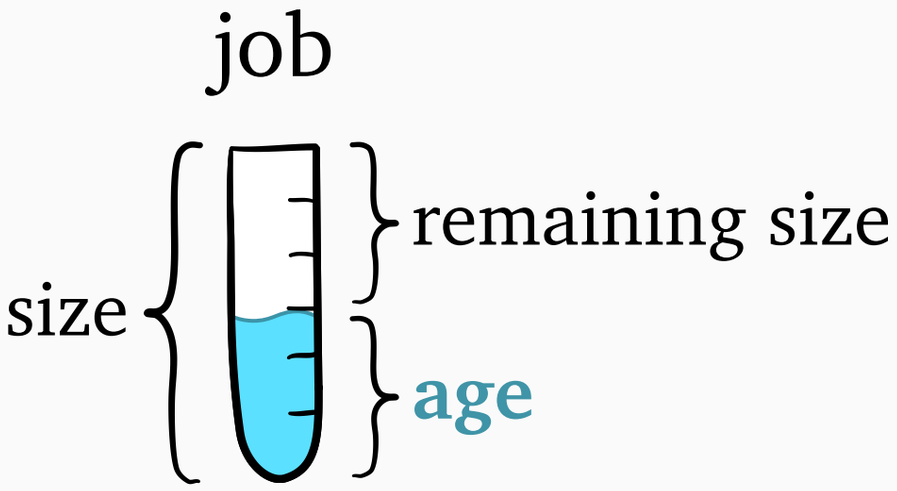


Single-server queueing system

M/G/1

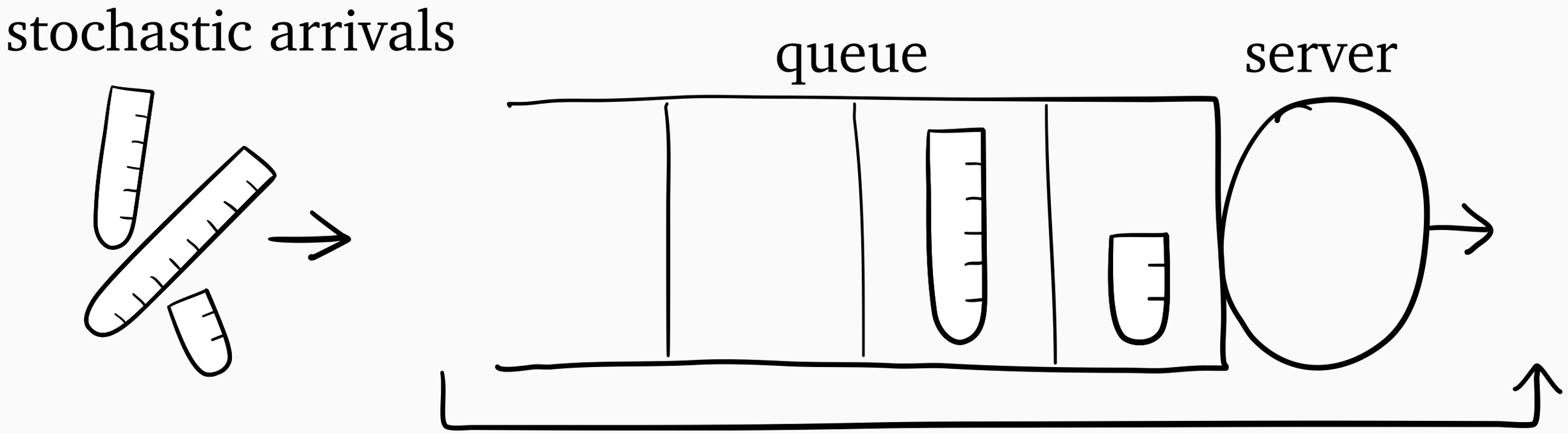


response time

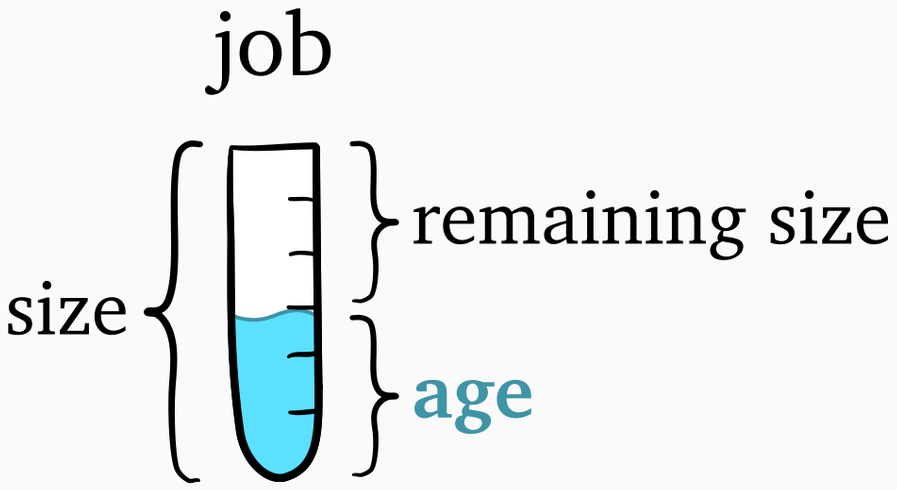


Single-server queueing system

M/G/1



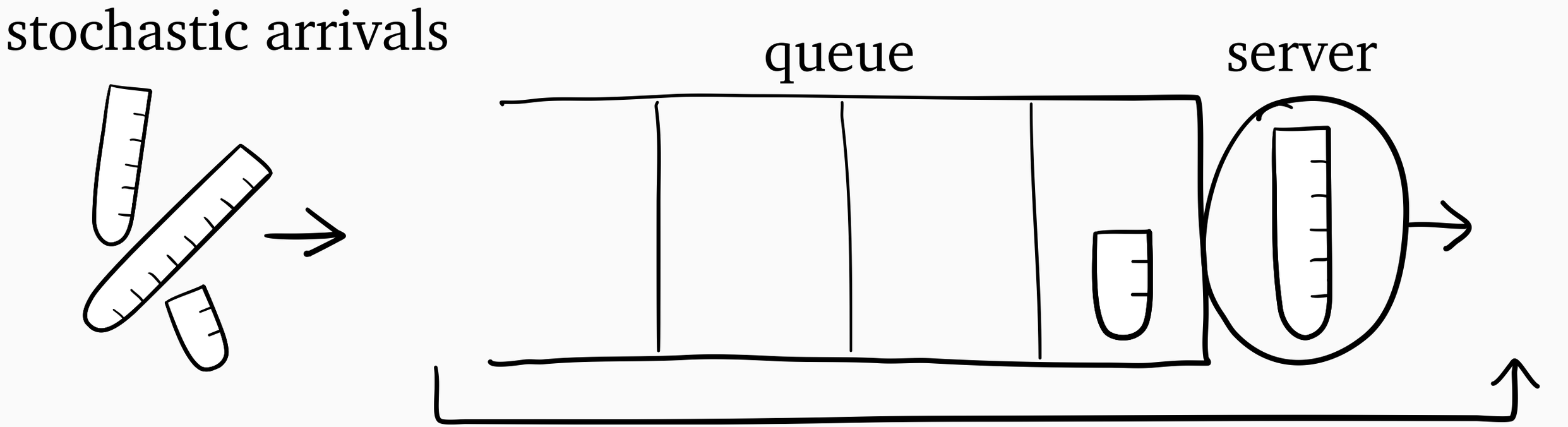
response time



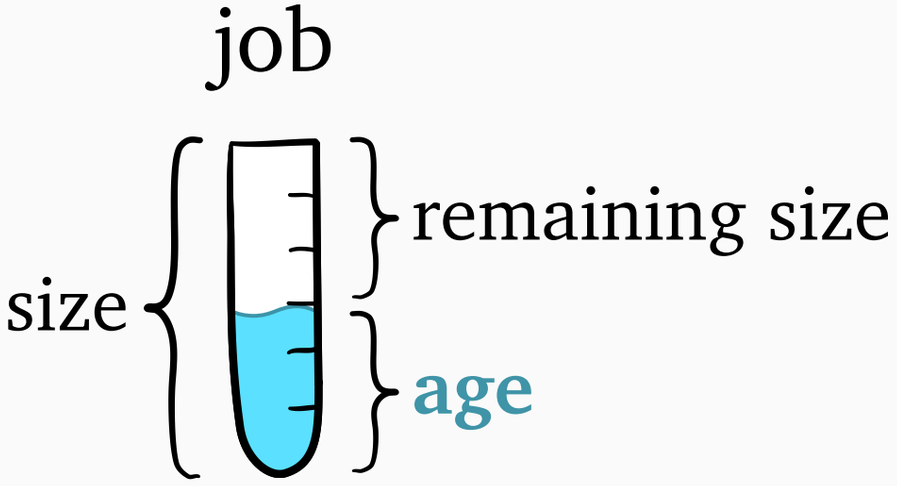
Scheduling policy:
decides which job to serve

Single-server queueing system

M/G/1



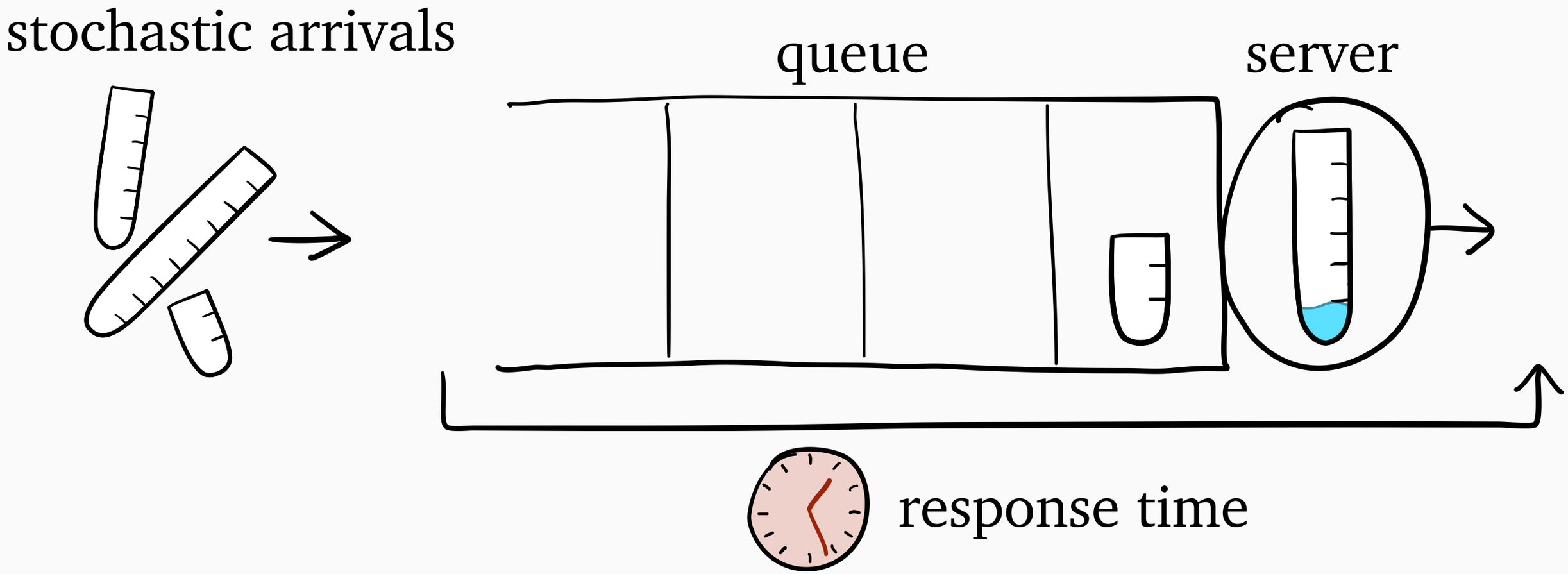
response time



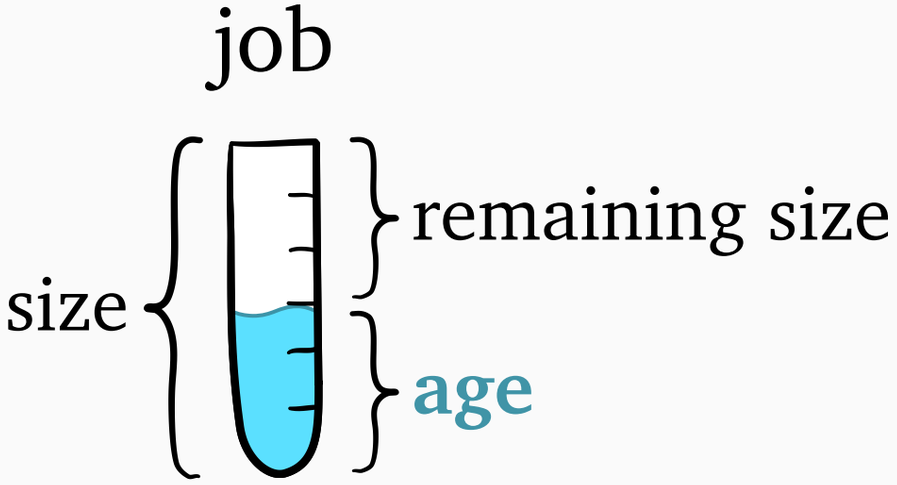
Scheduling policy:
decides which job to serve

Single-server queueing system

M/G/1



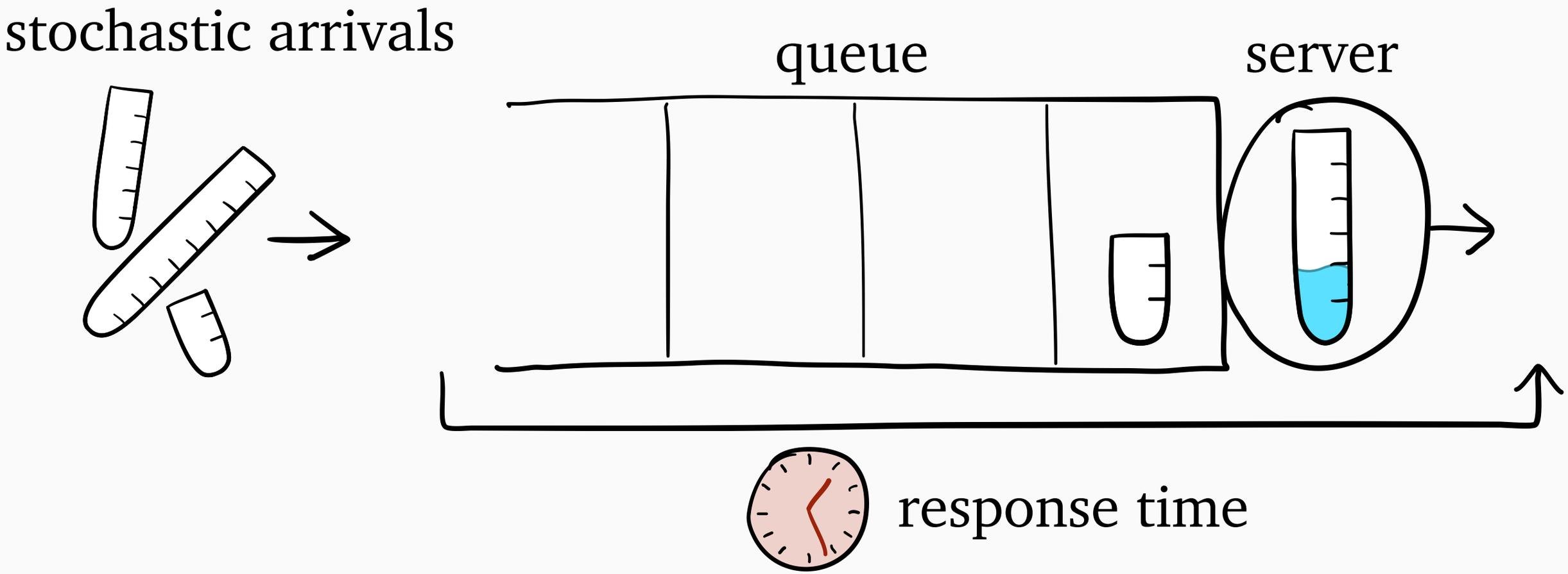
response time



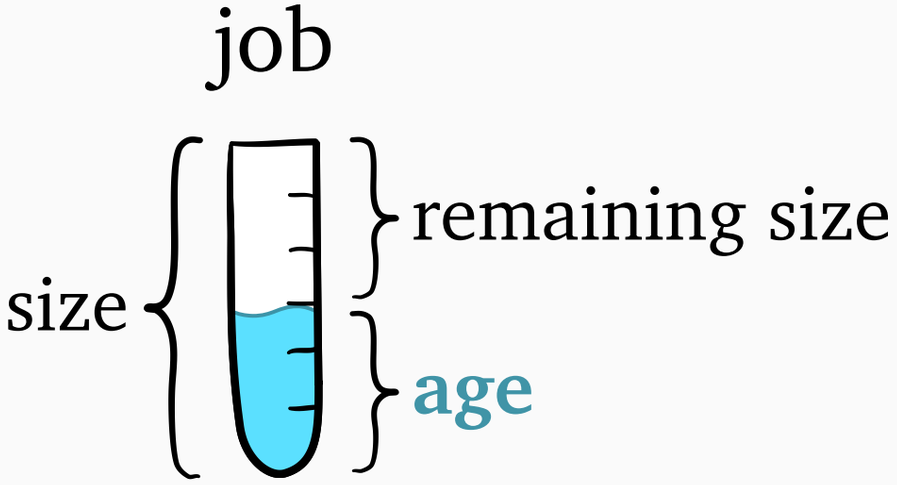
Scheduling policy:
decides which job to serve

Single-server queueing system

M/G/1



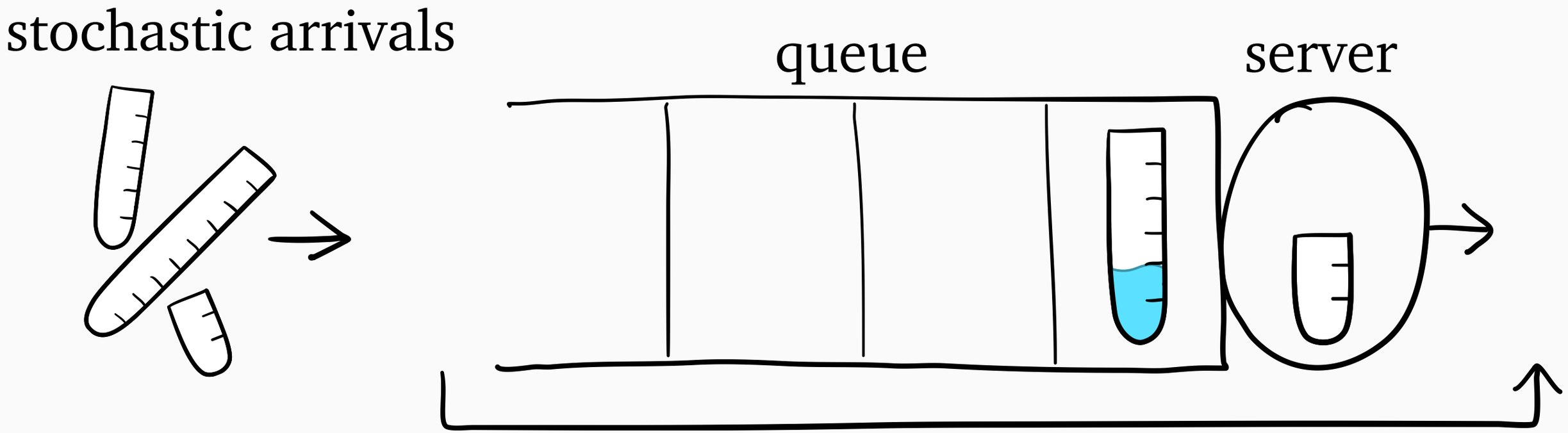
response time



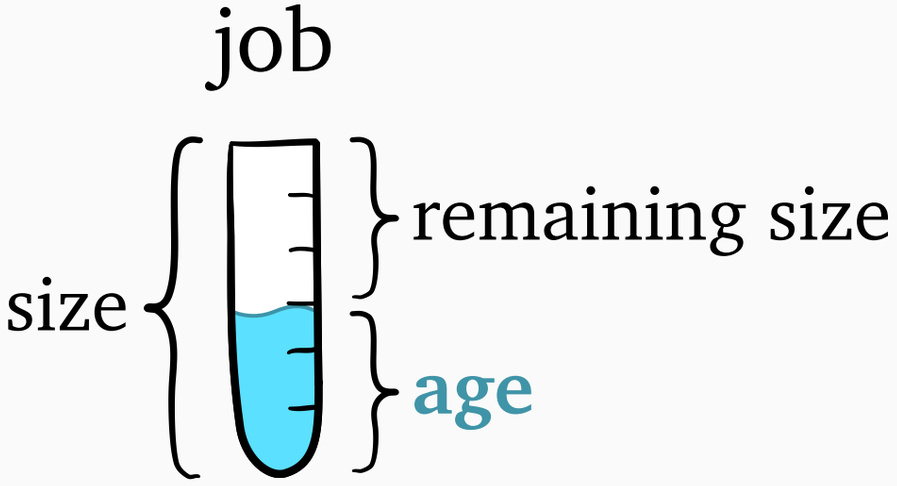
Scheduling policy:
decides which job to serve

Single-server queueing system

M/G/1



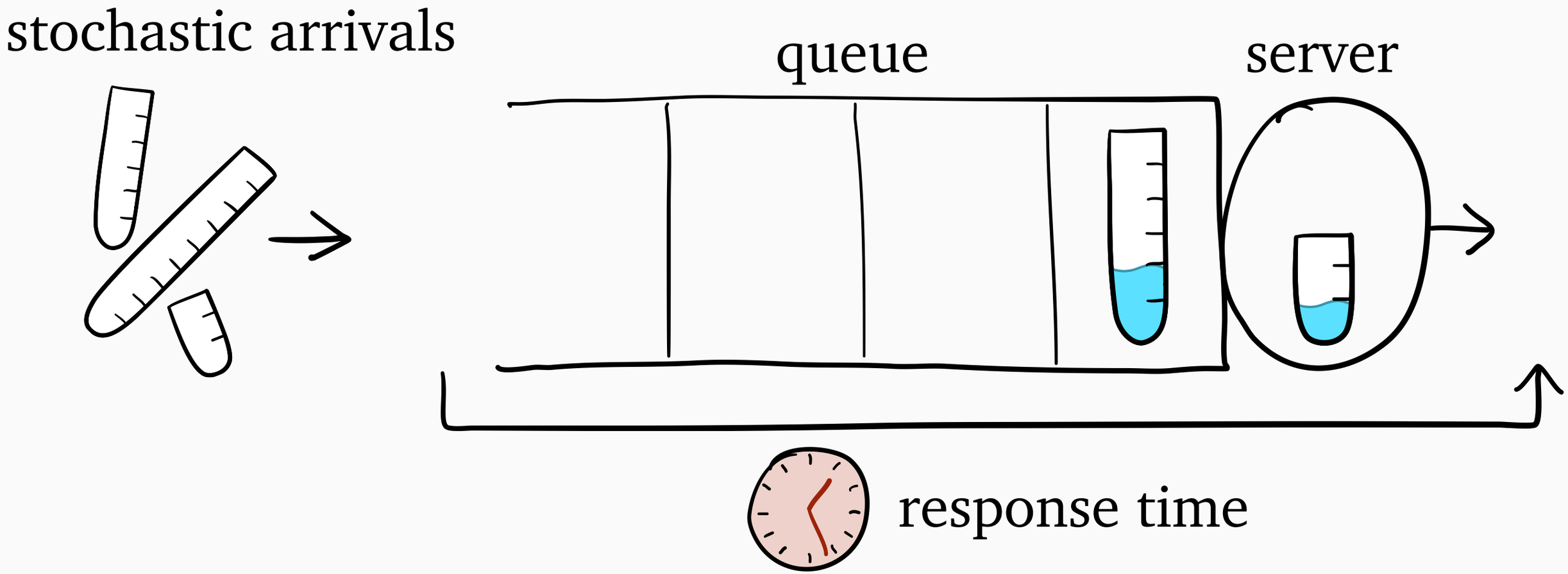
response time



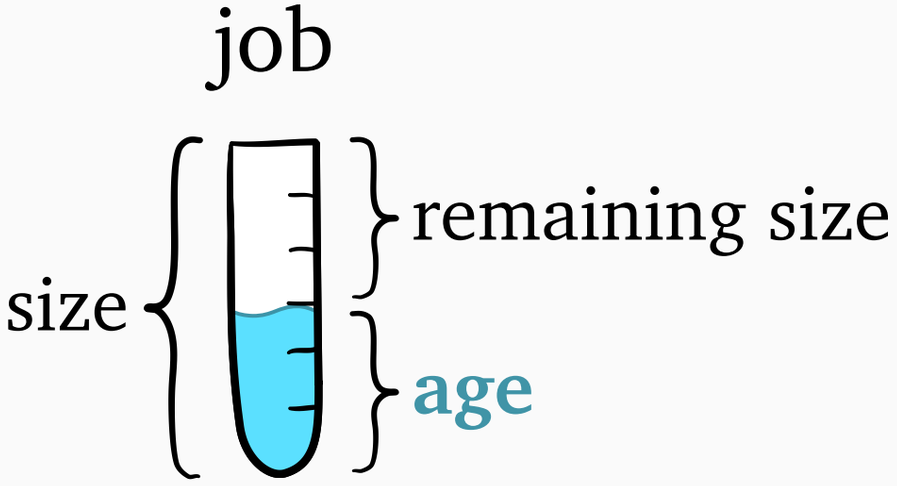
Scheduling policy:
decides which job to serve

Single-server queueing system

M/G/1



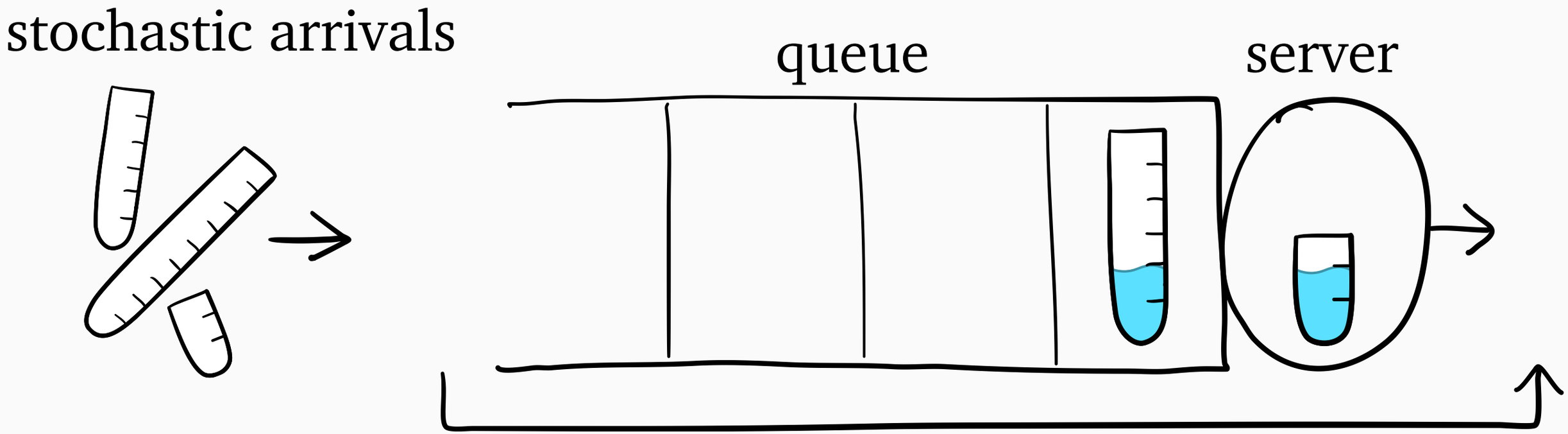
response time



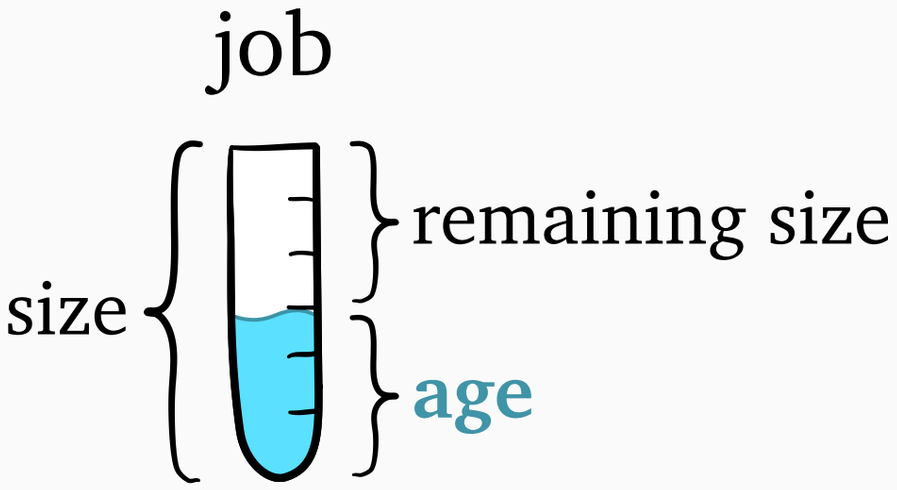
Scheduling policy:
decides which job to serve

Single-server queueing system

M/G/1



response time

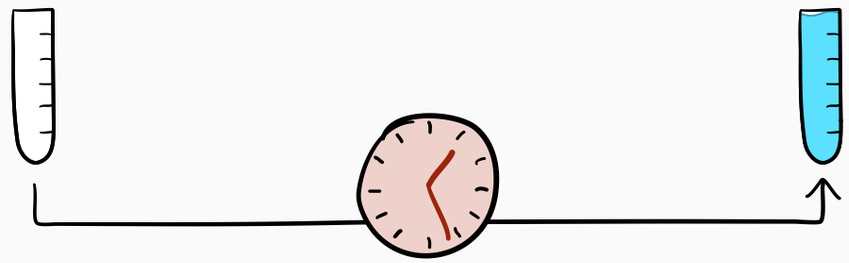


Scheduling policy:
decides which job to serve

Evaluating performance

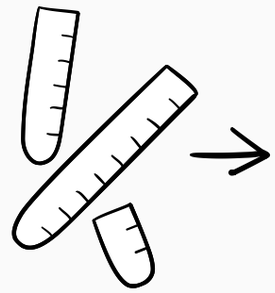
Evaluating performance

response time

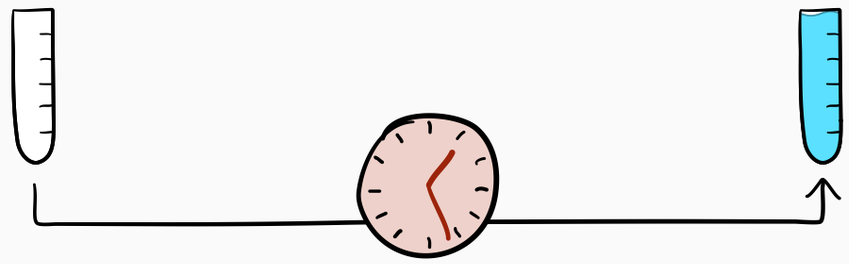


Evaluating performance

stochastic arrival process λ, S

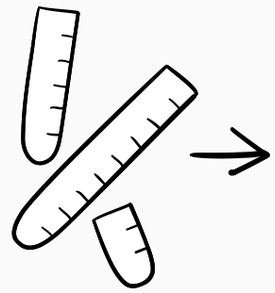


response time

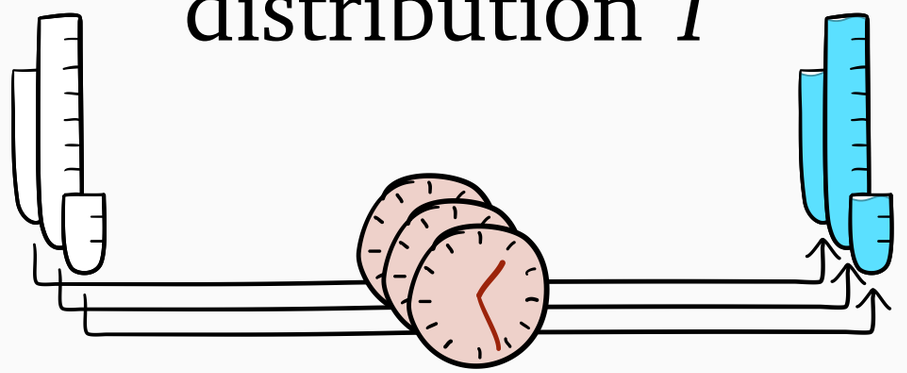


Evaluating performance

stochastic arrival process λ, S

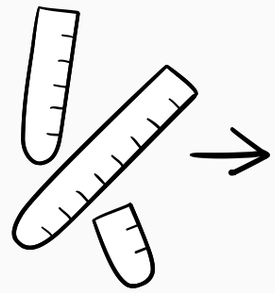


response time distribution T



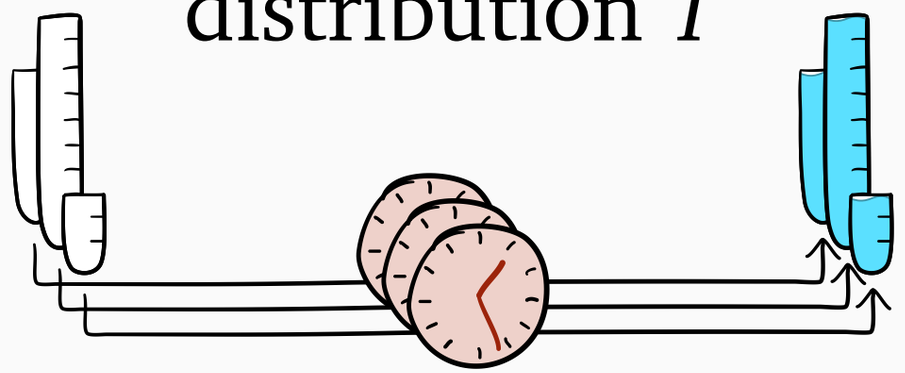
Evaluating performance

stochastic arrival process λ, S



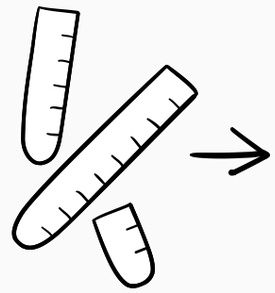
scheduling policy

response time distribution T



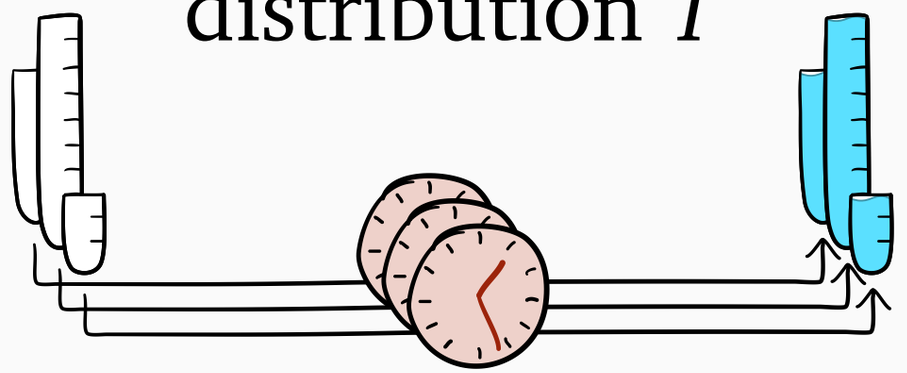
Evaluating performance

stochastic arrival process λ, S



scheduling policy

response time distribution T

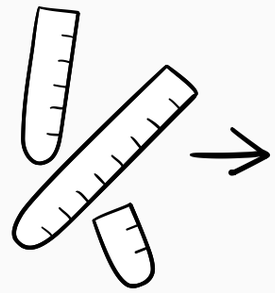


Metrics:

- mean $\mathbf{E}[T]$
- tail probability $\mathbf{P}[T > t]$
- percentiles, e.g. T_{99}

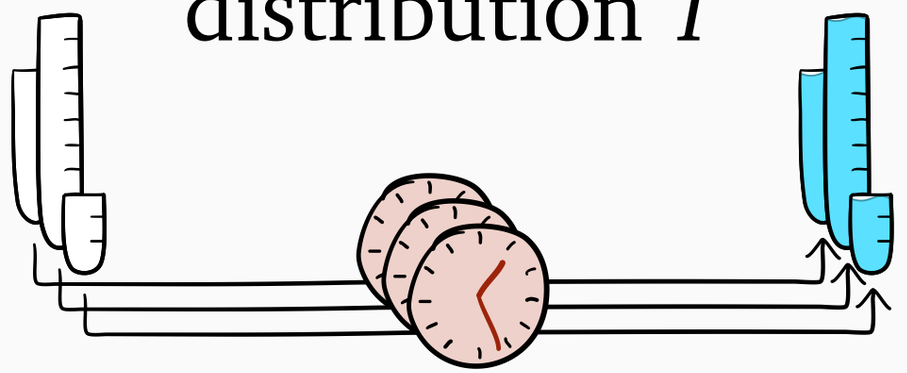
Evaluating performance

stochastic arrival process λ, S



scheduling policy

response time distribution T



Goals:

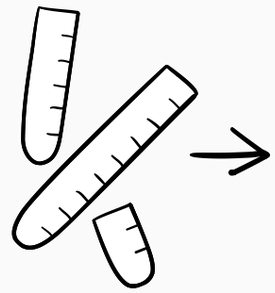
- *analyze* **policy**

Metrics:

- mean $\mathbf{E}[T]$
- tail probability $\mathbf{P}[T > t]$
- percentiles, e.g. T_{99}

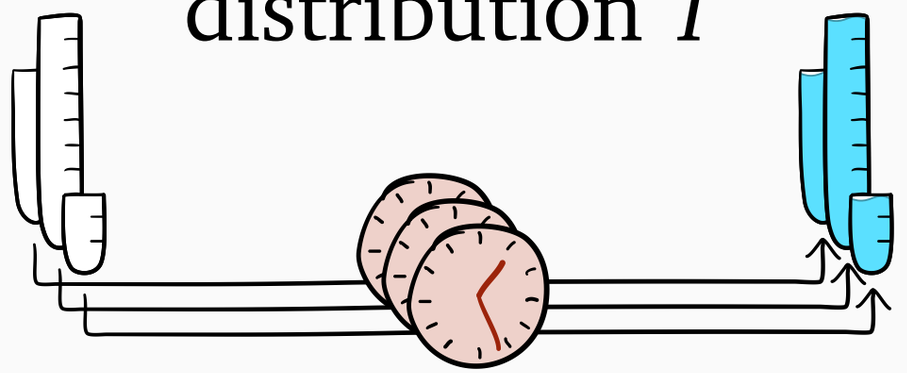
Evaluating performance

stochastic arrival process λ, S



scheduling policy

response time distribution T



Goals:

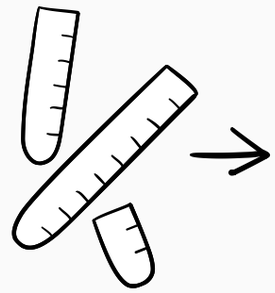
- *analyze* **policy**
- *optimize* **policy**

Metrics:

- mean $\mathbf{E}[T]$
- tail probability $\mathbf{P}[T > t]$
- percentiles, e.g. T_{99}

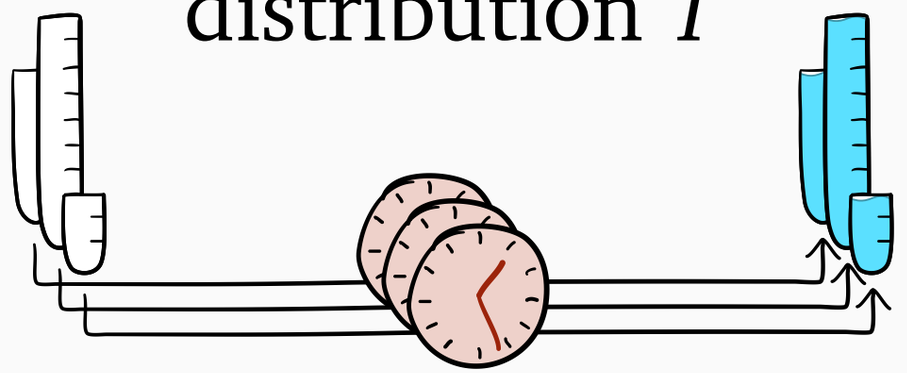
Evaluating performance

stochastic arrival process λ, S



scheduling policy

response time distribution T



Goals:

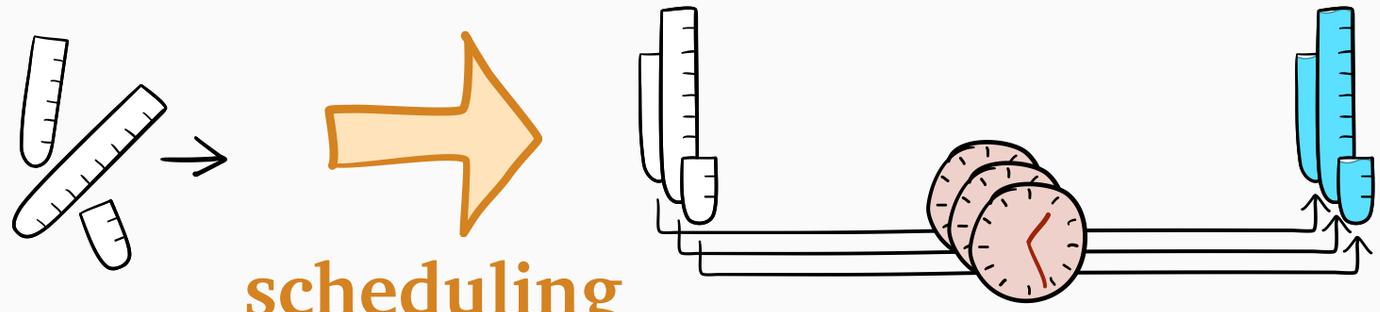
- *analyze* **policy**
- *optimize* **policy**

our focus

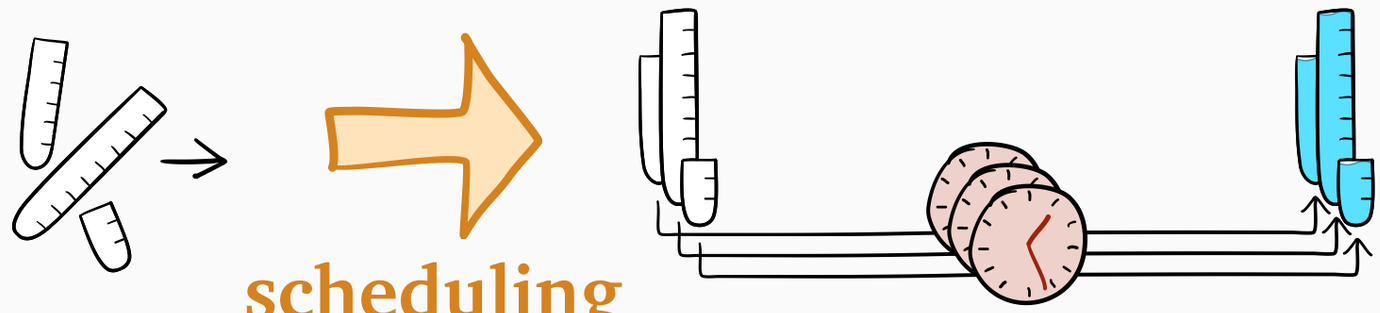
Metrics:

- mean $\mathbf{E}[T]$
- tail probability $\mathbf{P}[T > t]$
- percentiles, e.g. T_{99}

Goal: analyze impact of **policy**



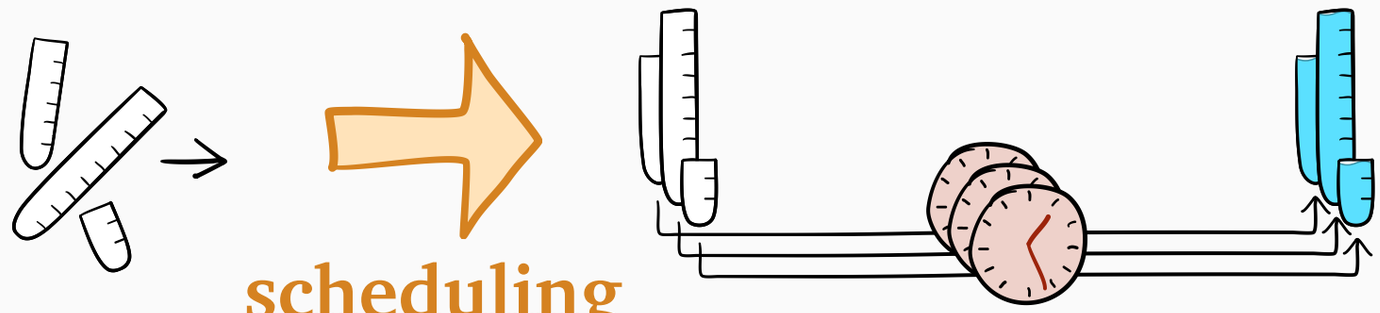
Goal: analyze impact of **policy**



Problem:

- Theory limited to “simple” **policies**

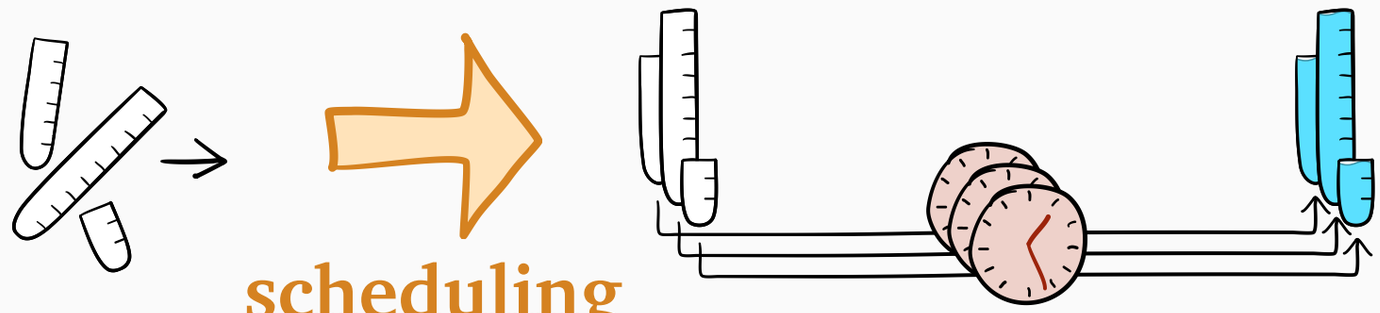
Goal: analyze impact of **policy**



Problem:

- Theory limited to “simple” **policies**
- In practice, “complex” **policies** matter

Goal: analyze impact of **policy**

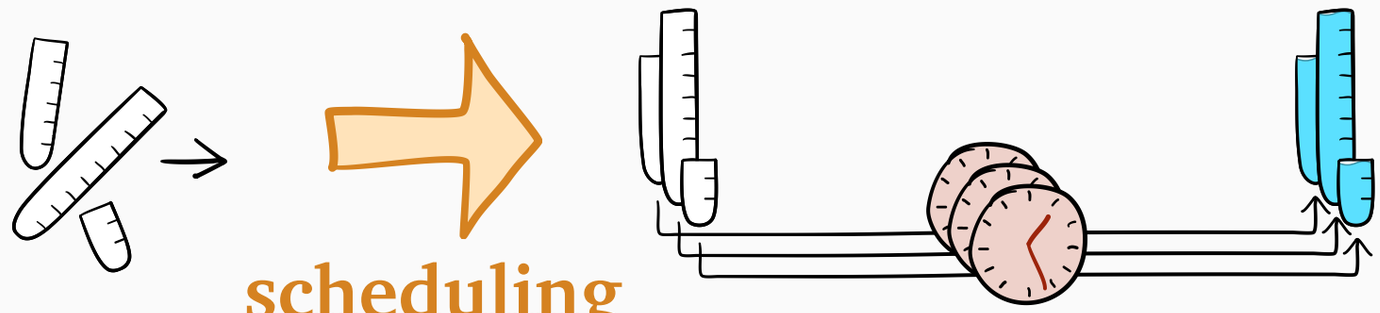


Problem:

- Theory limited to “simple” **policies**
- In practice, “complex” **policies** matter

What makes a **policy** “simple” or “complex”?

Goal: analyze impact of **policy**



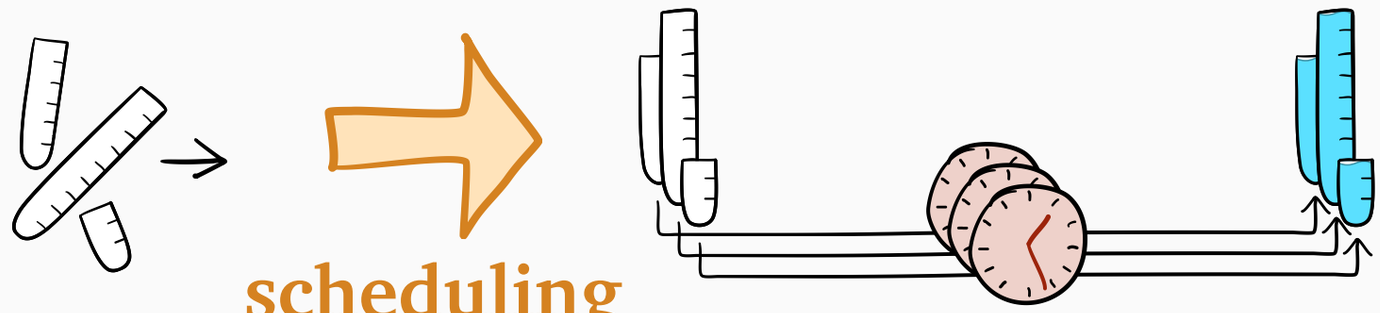
Problem:

- Theory limited to “simple” **policies**
- In practice, “complex” **policies** matter

What makes a **policy** “simple” or “complex”?

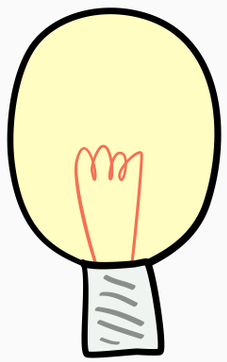
What makes “complex” **policies** hard to analyze?

Goal: analyze impact of **policy**



Problem:

- Theory limited to “simple” **policies**
- In practice, “complex” **policies** matter

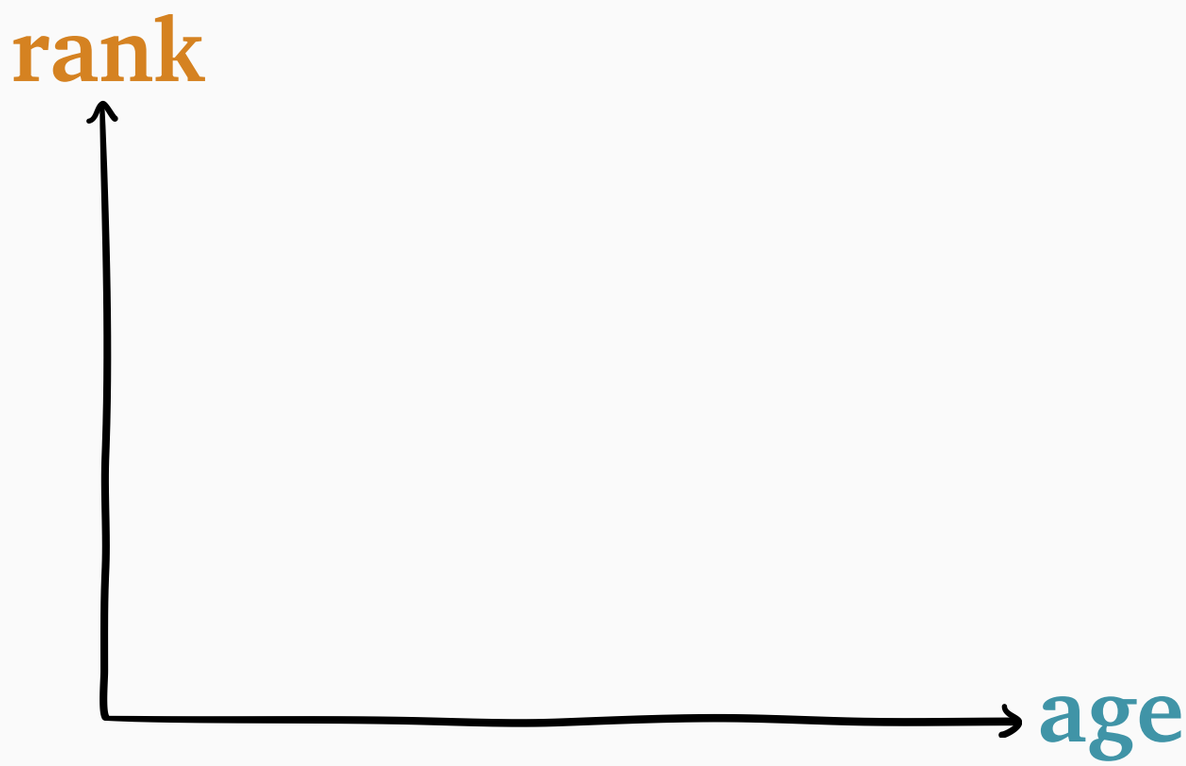


SOAP mindset:
unifying *language* to describe **policies**

What makes a **policy** “simple” or “complex”?

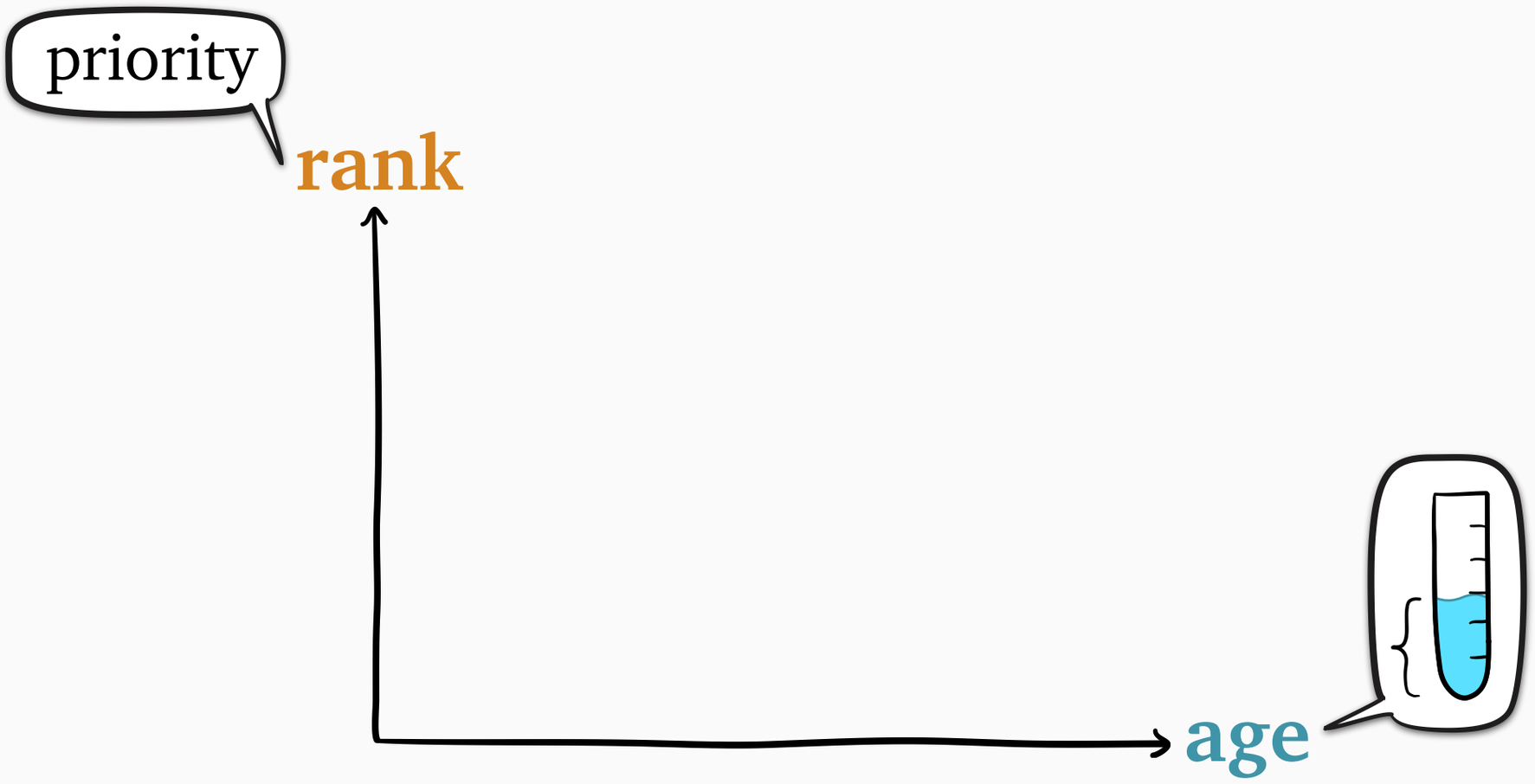
What makes “complex” **policies** hard to analyze?

Unifying language: **rank** functions

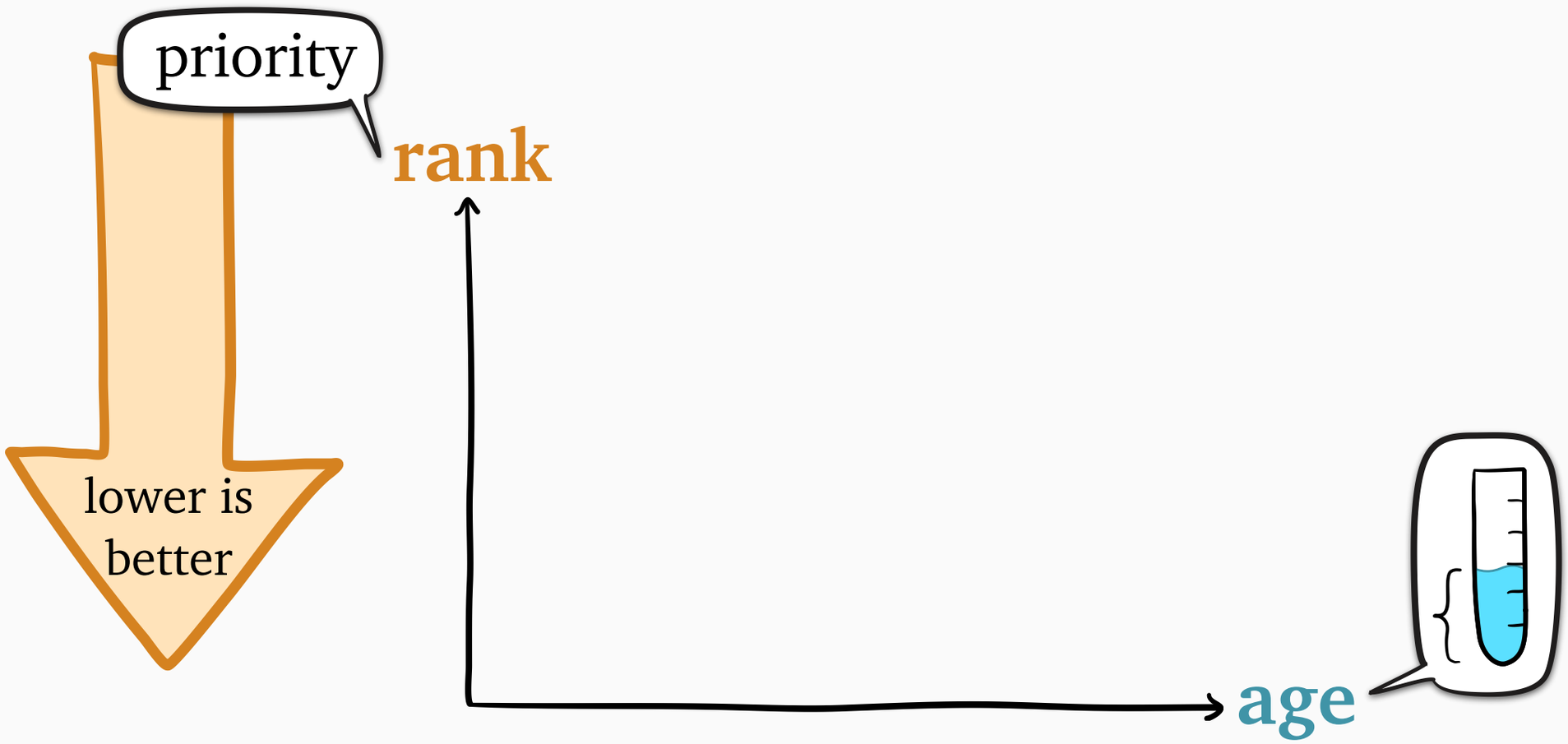


Unifying language: **rank** functions

Unifying language: **rank** functions



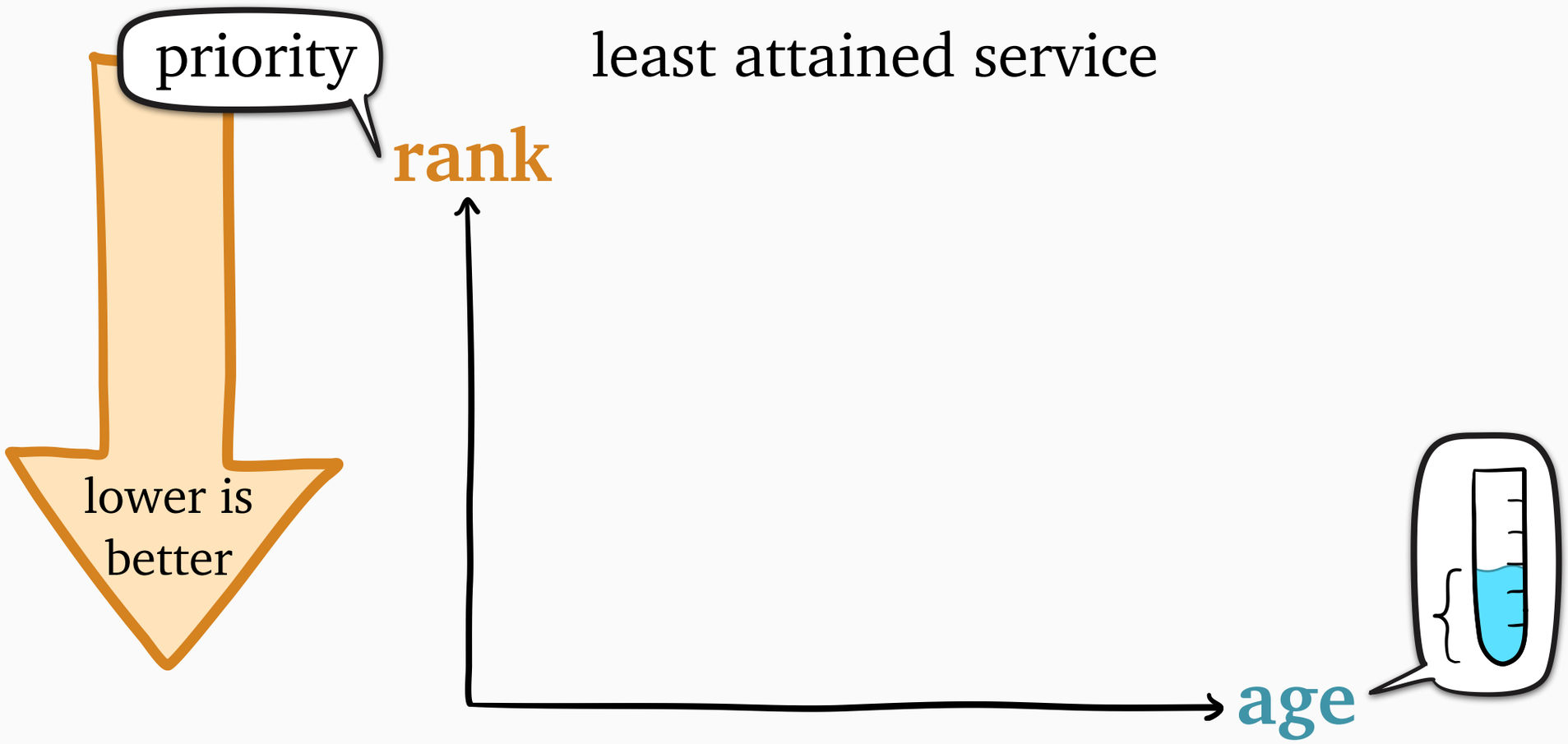
Unifying language: **rank** functions



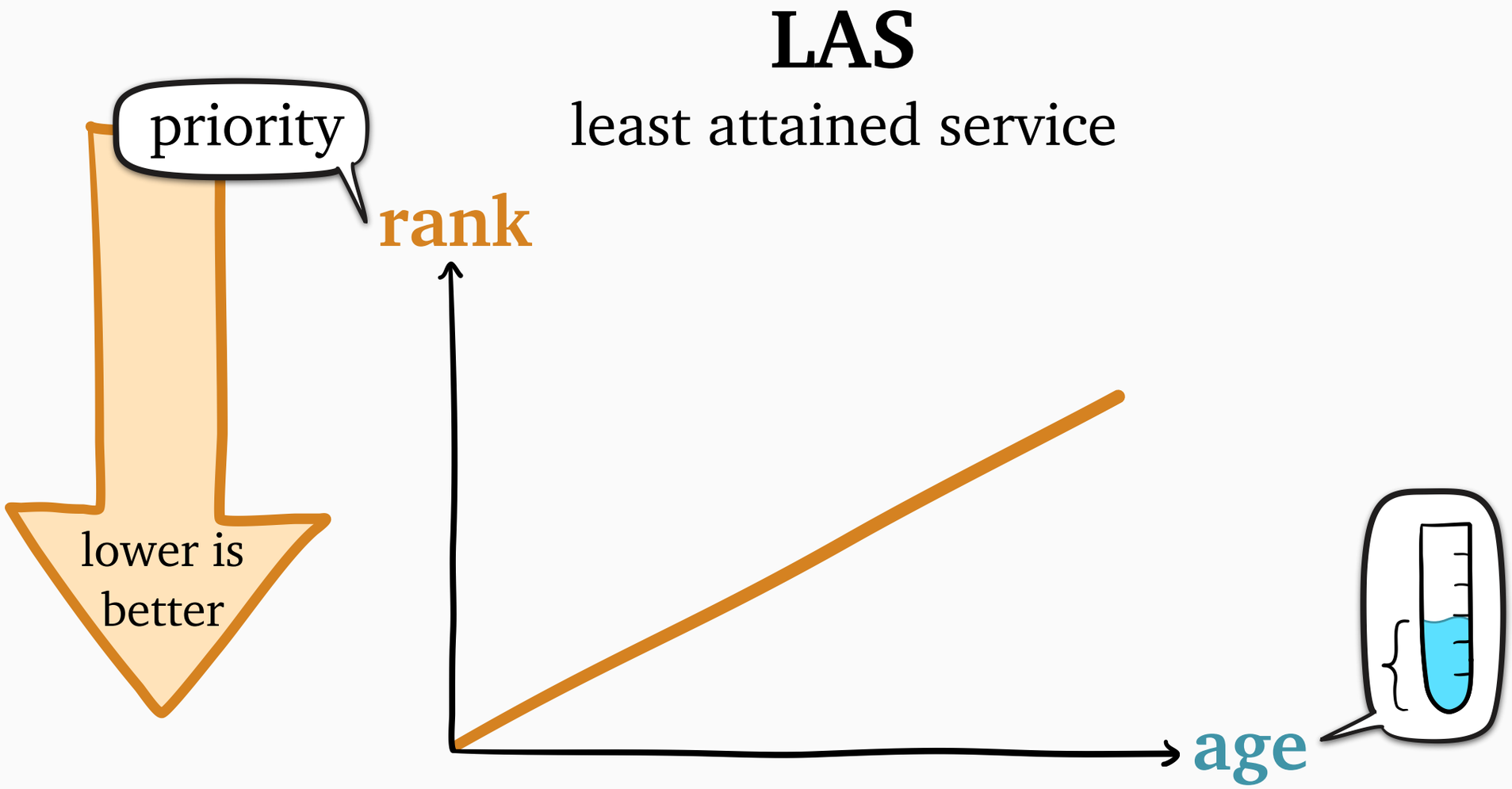
Unifying language: **rank** functions

LAS

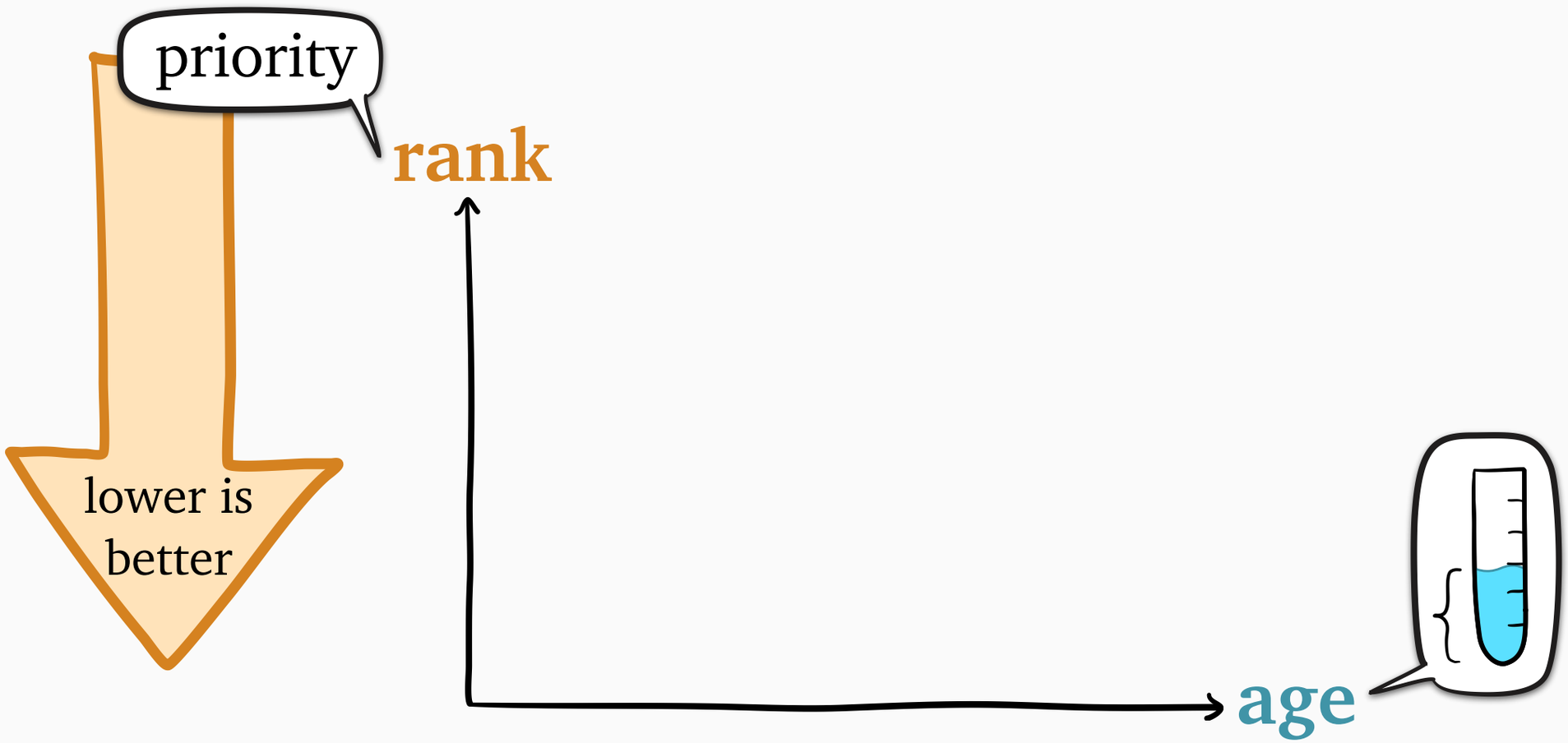
least attained service

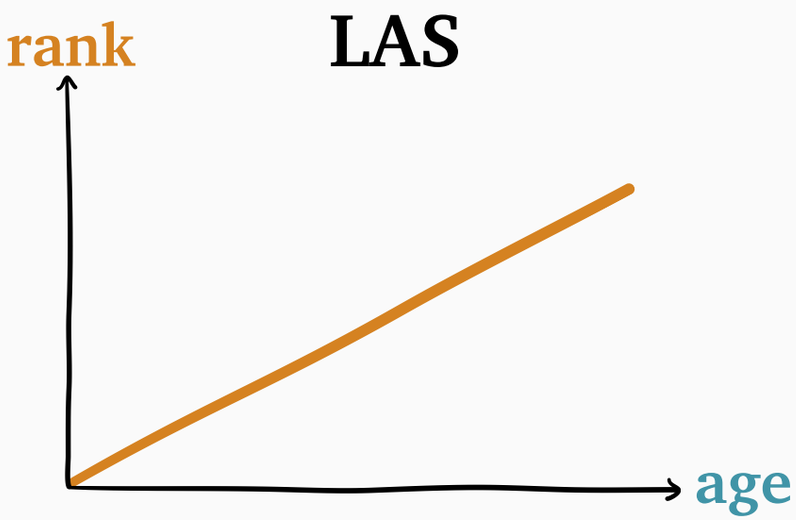


Unifying language: rank functions



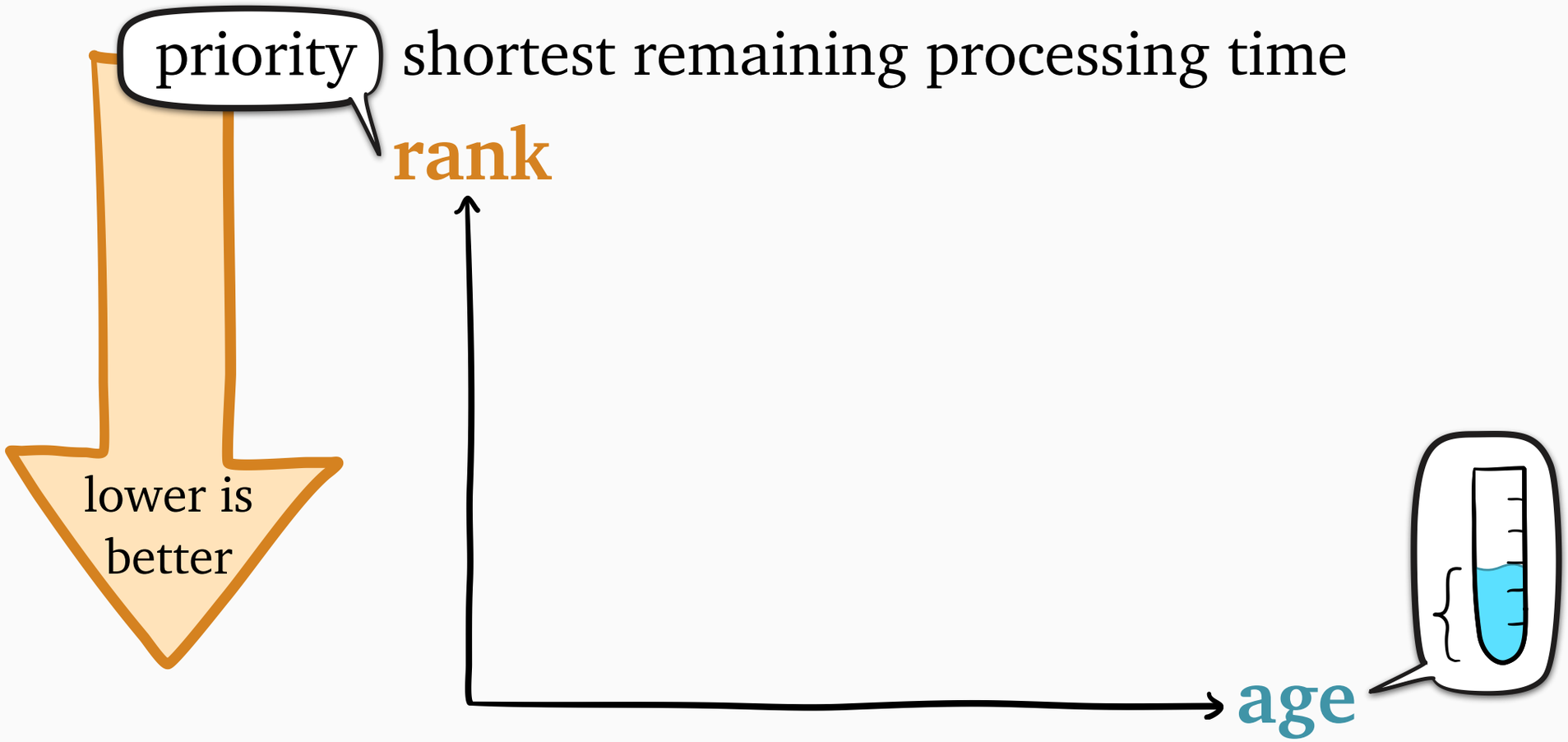
Unifying language: **rank** functions



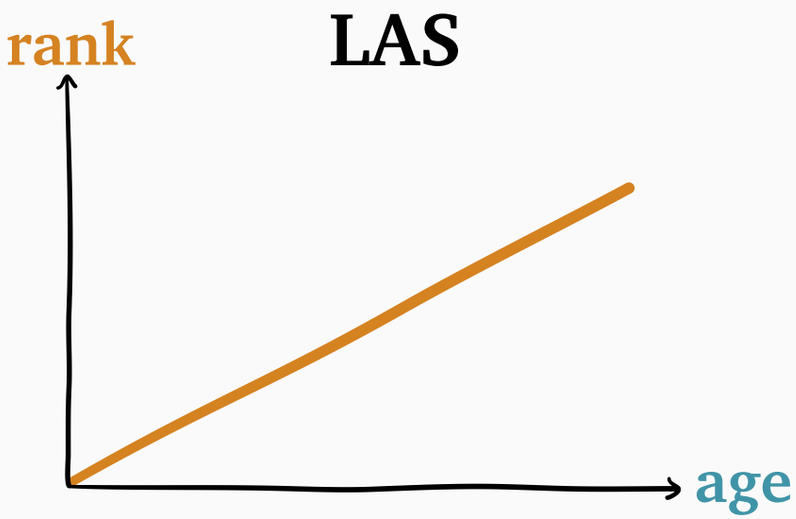


Unifying language: **rank** functions

SRPT

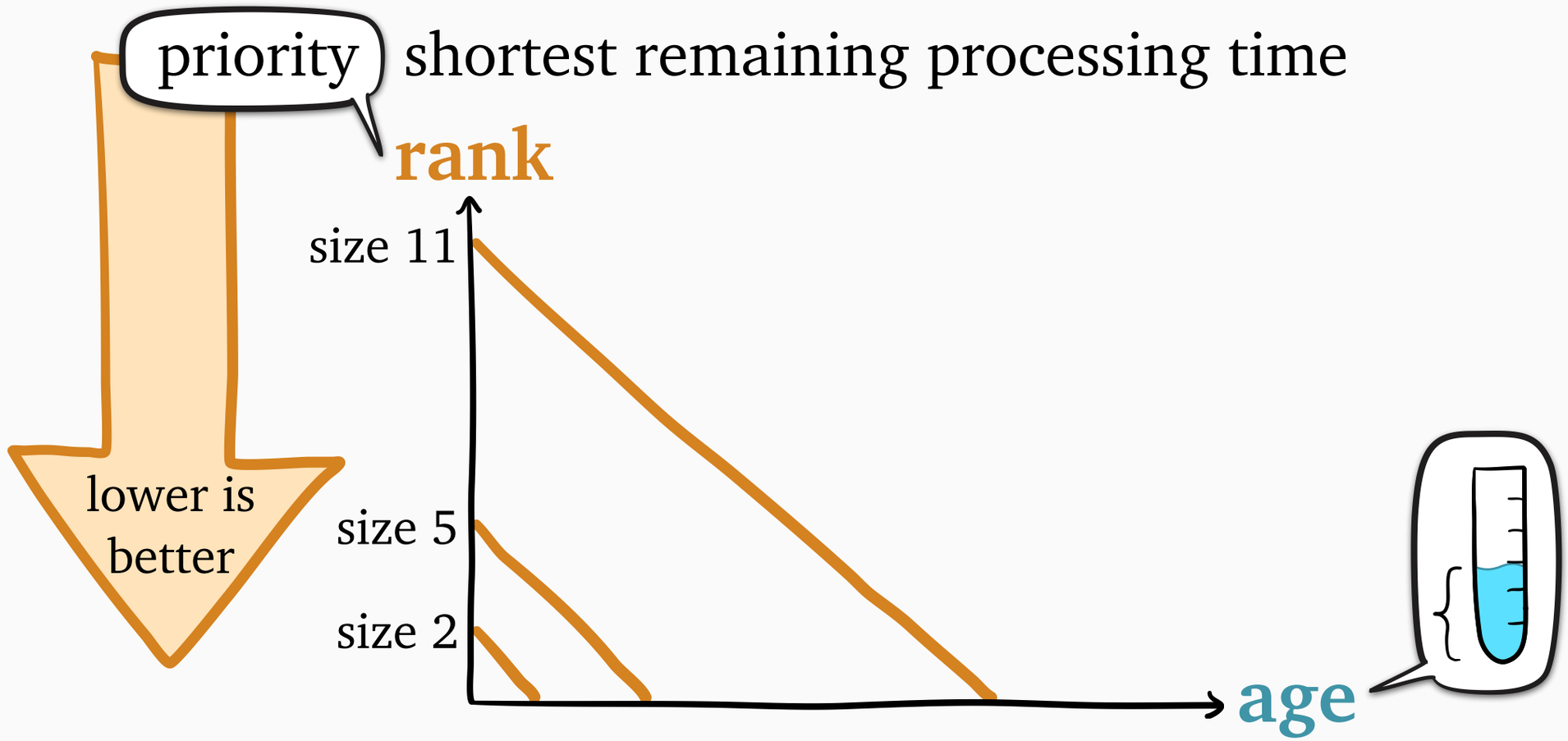


LAS



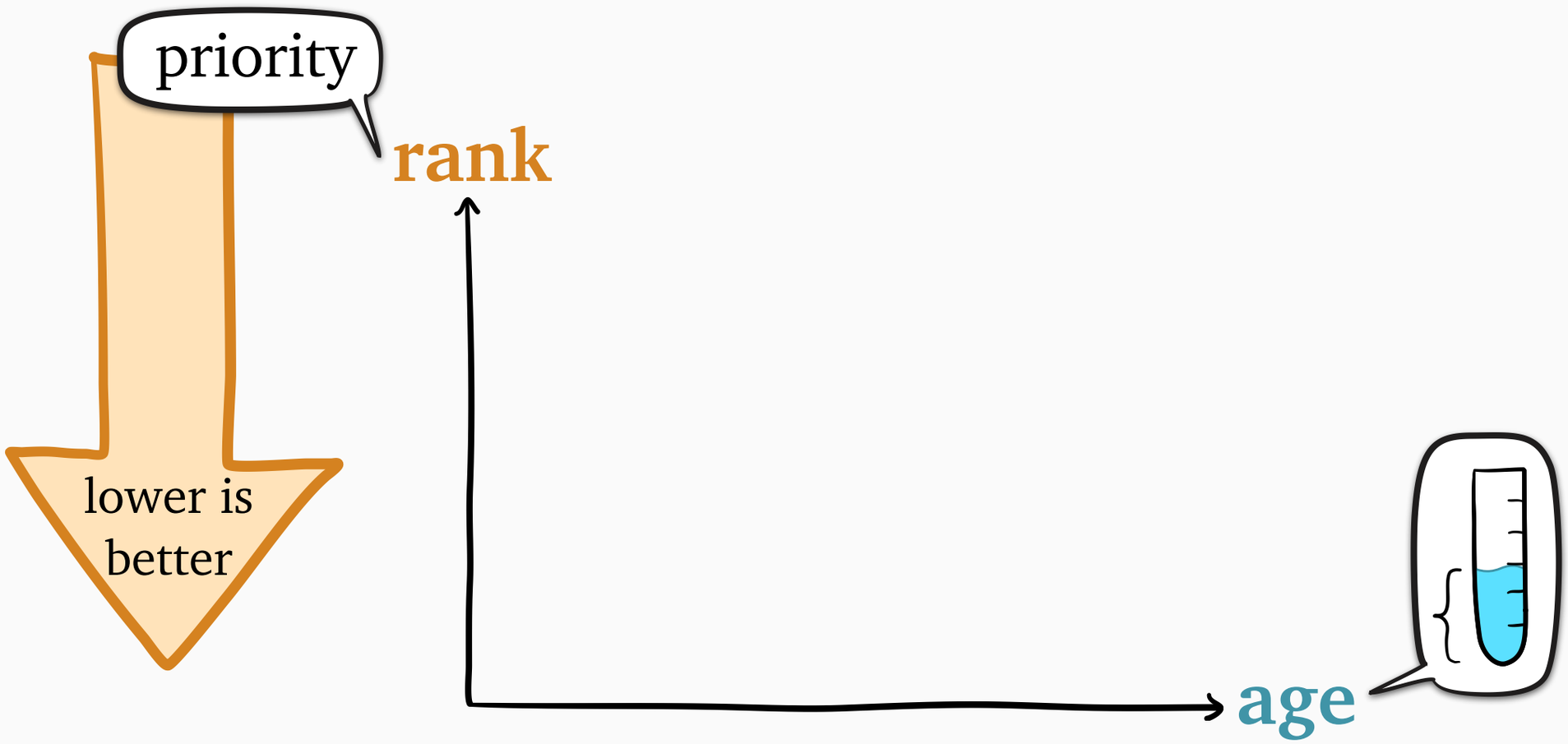
Unifying language: **rank** functions

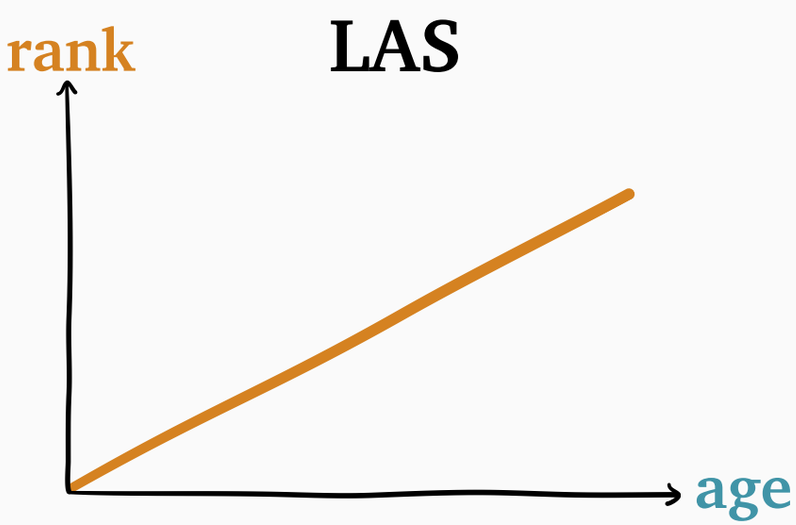
SRPT

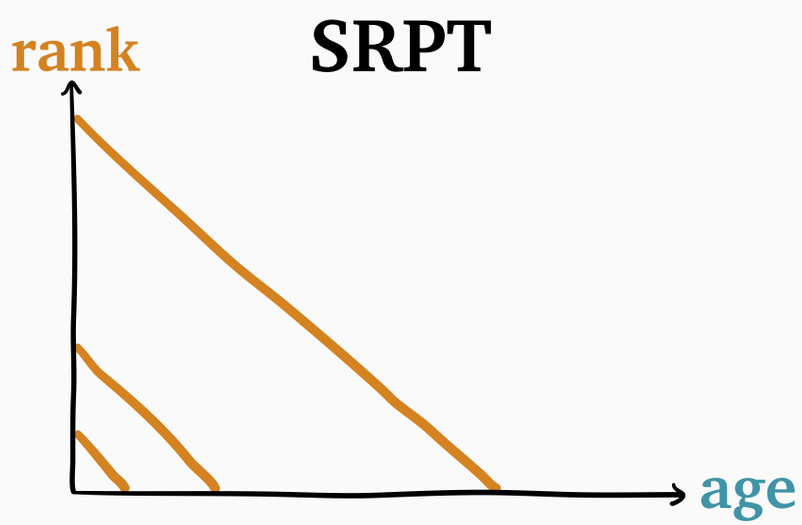


LAS

Unifying language: **rank** functions



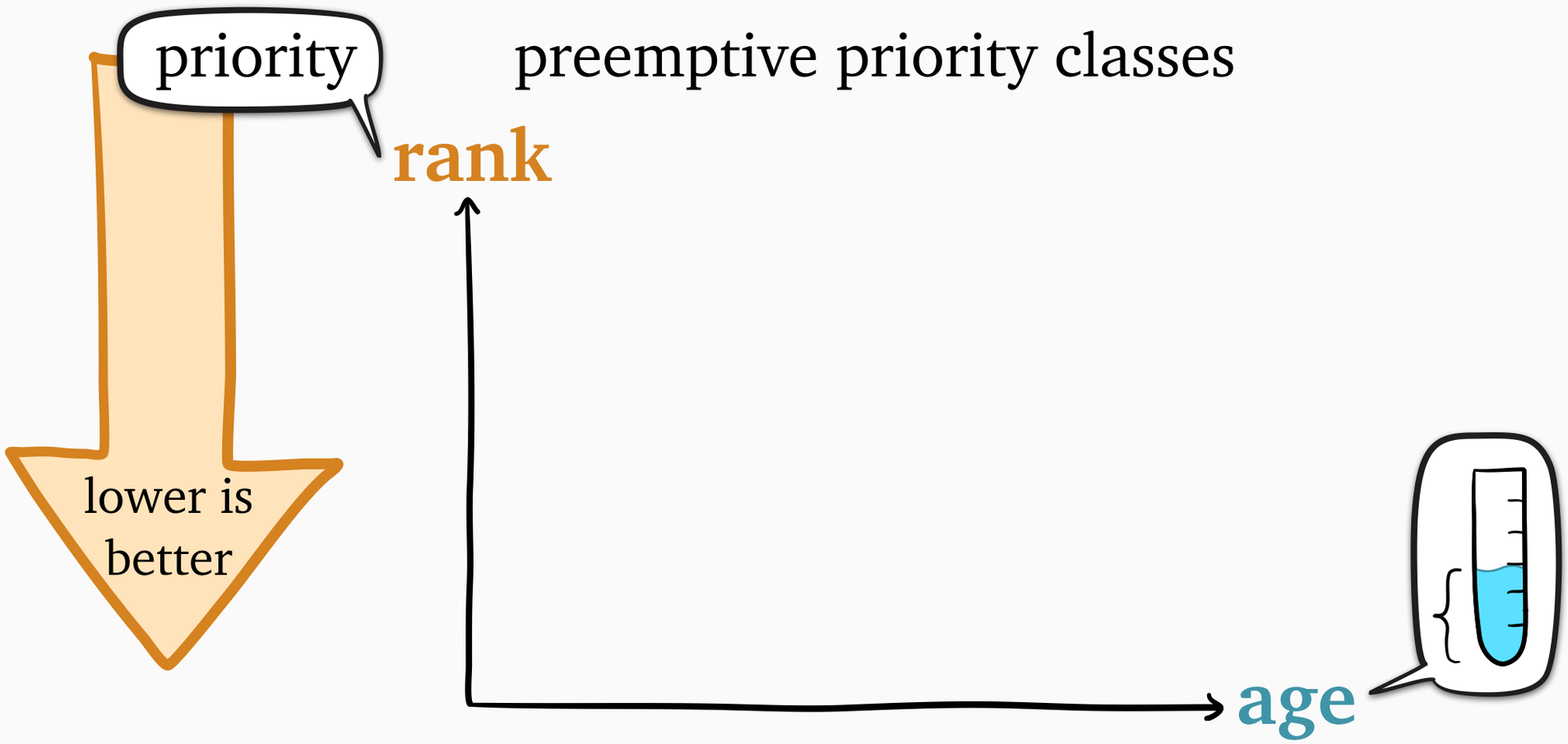




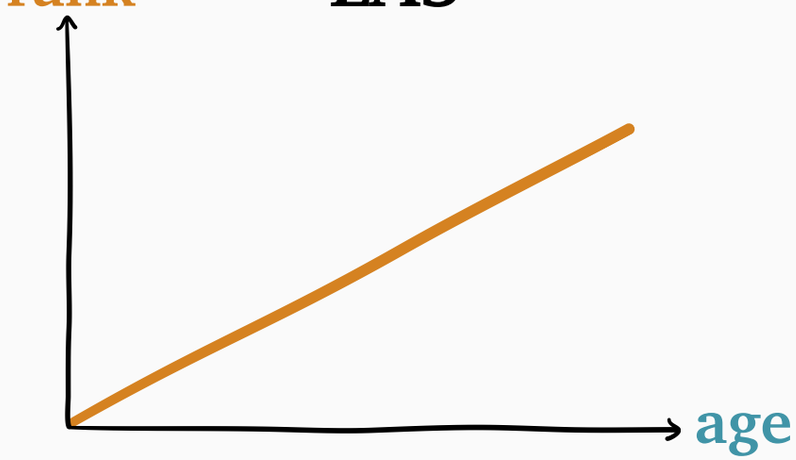
Unifying language: rank functions

P-Prio

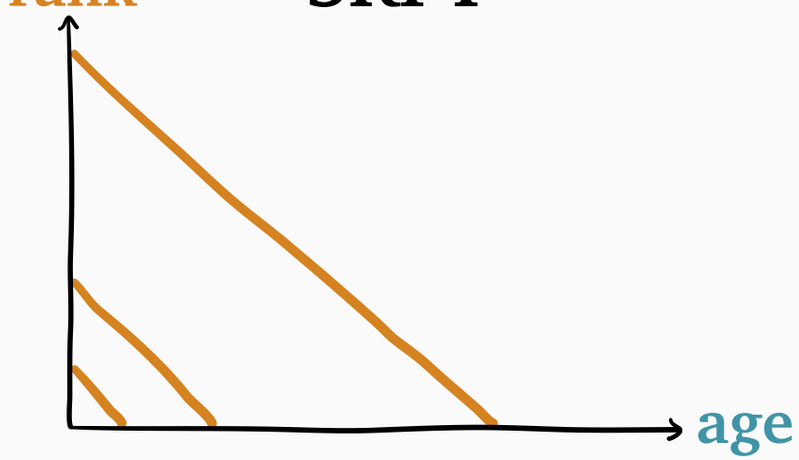
preemptive priority classes



LAS



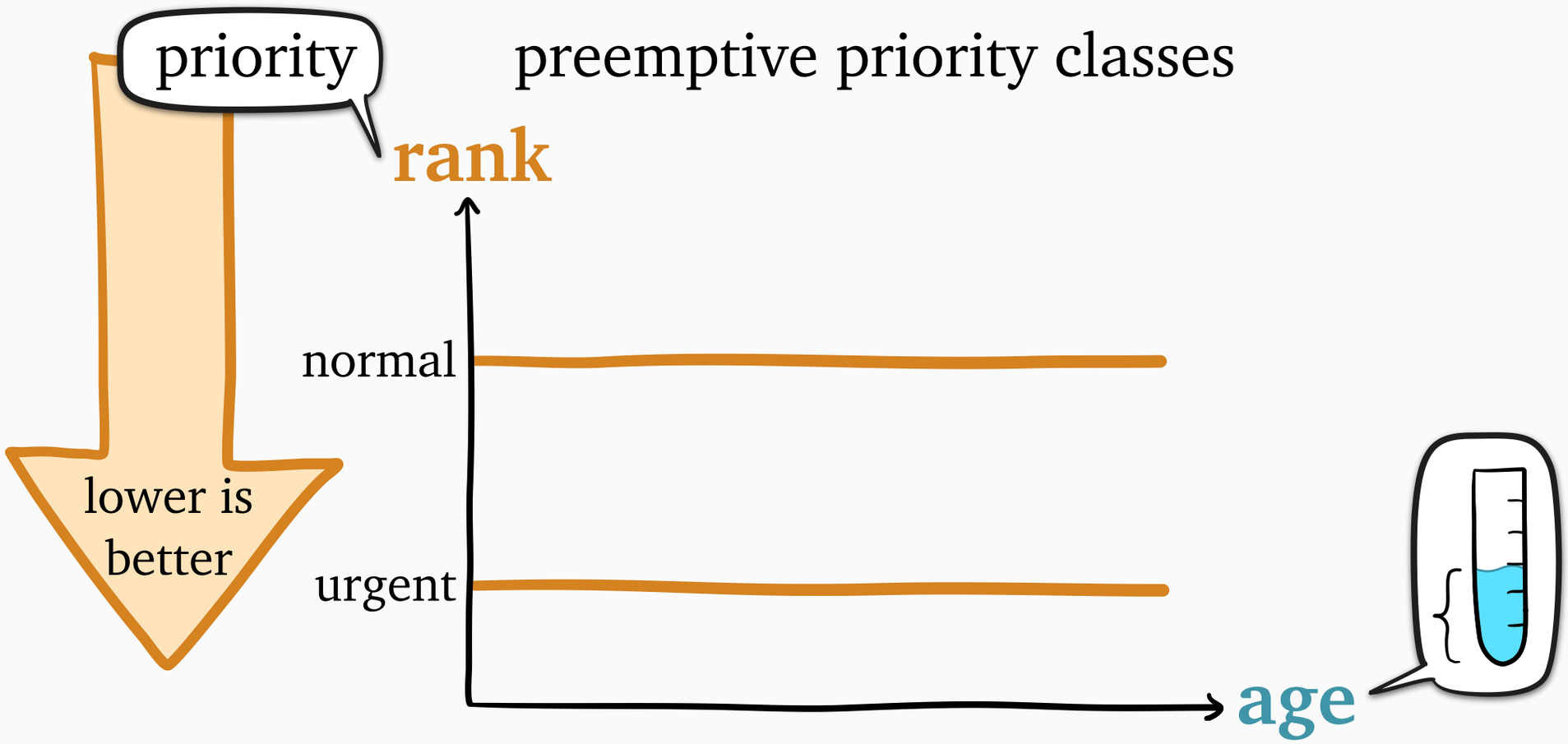
SRPT



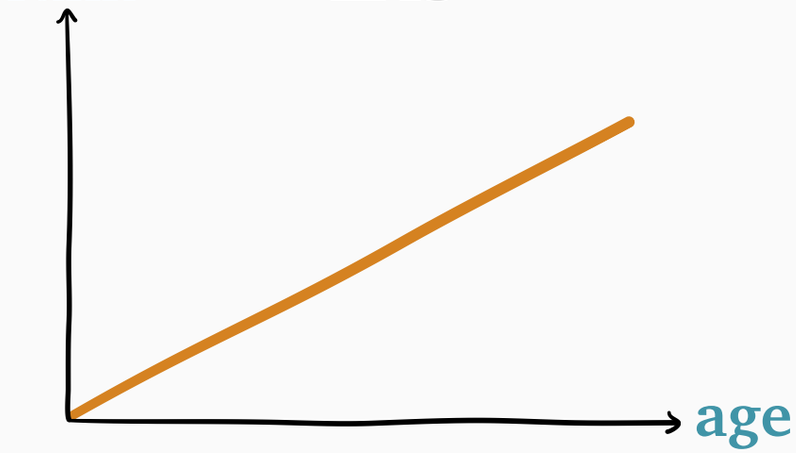
Unifying language: rank functions

P-Prio

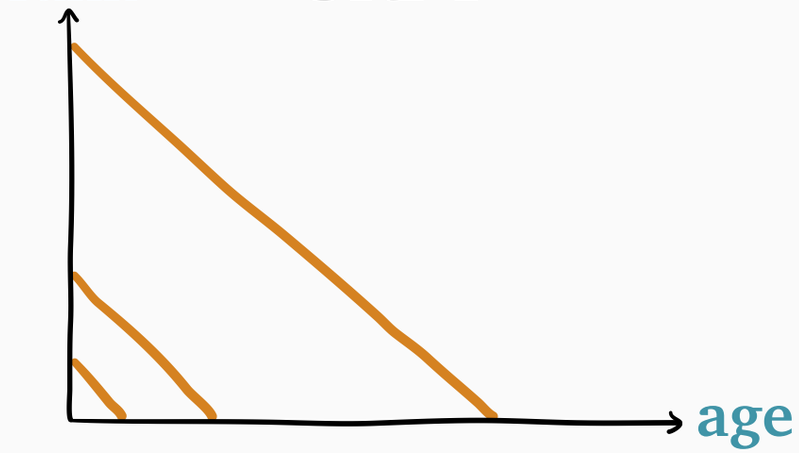
preemptive priority classes



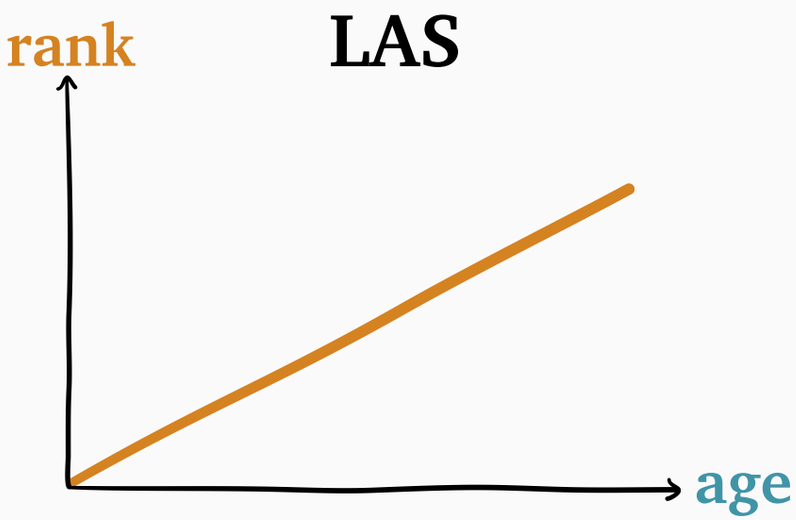
LAS

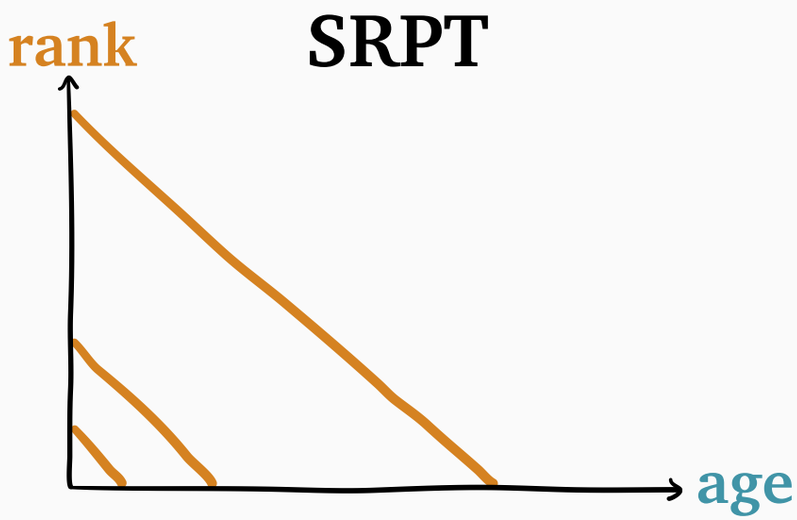


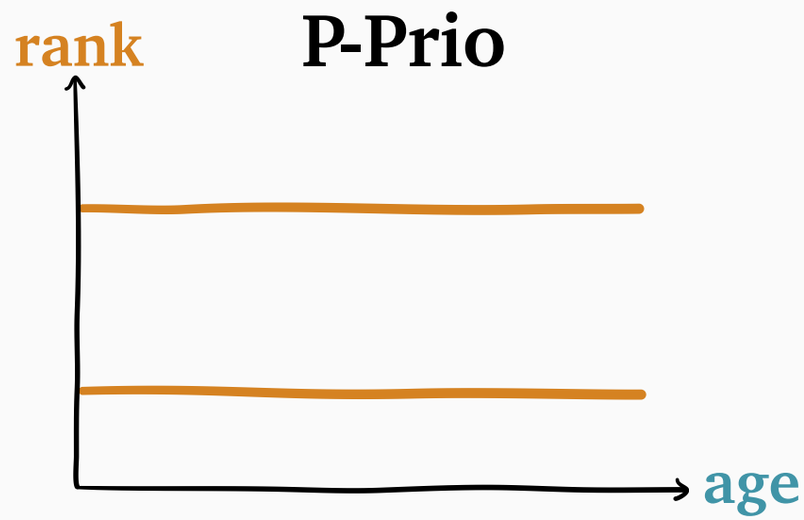
SRPT



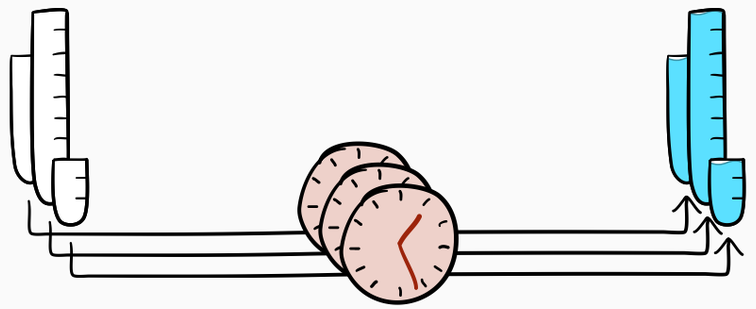
Unifying language: **rank** functions



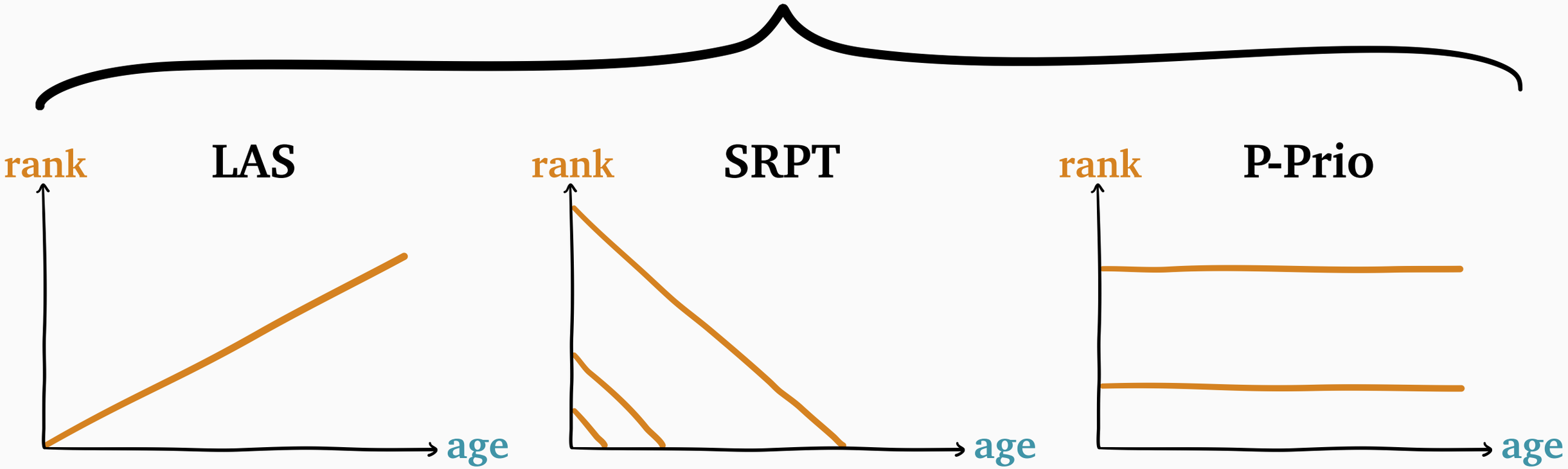




Unifying language: **rank** functions

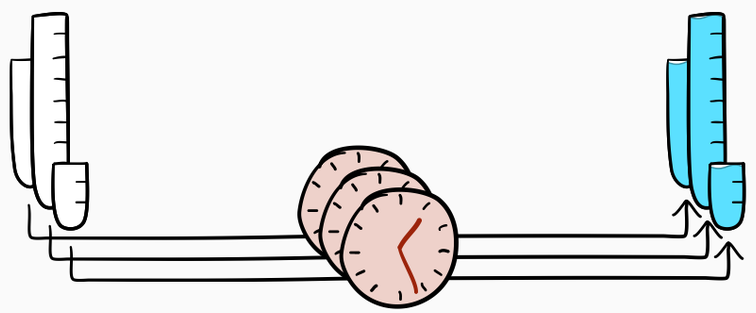


response time T known

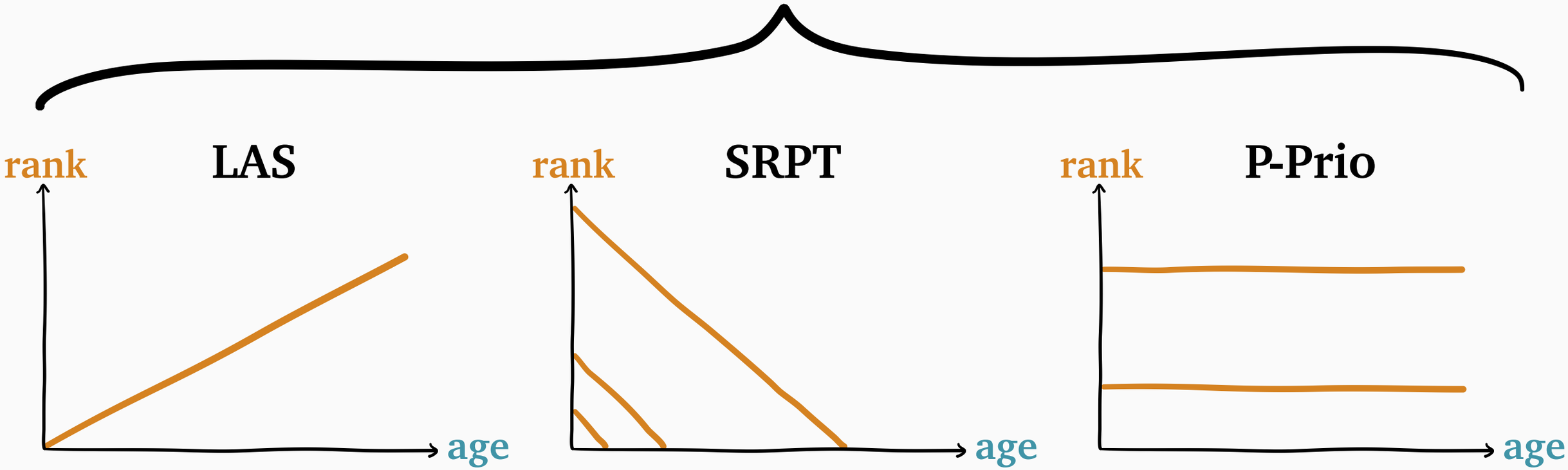


Unifying language: **rank** functions

"Simple" **policy** = ? **rank** function

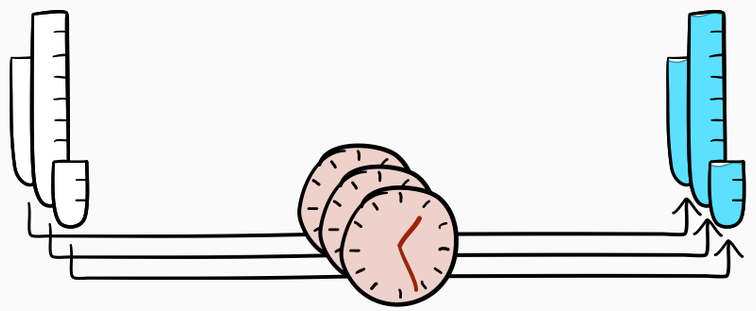


response time T known

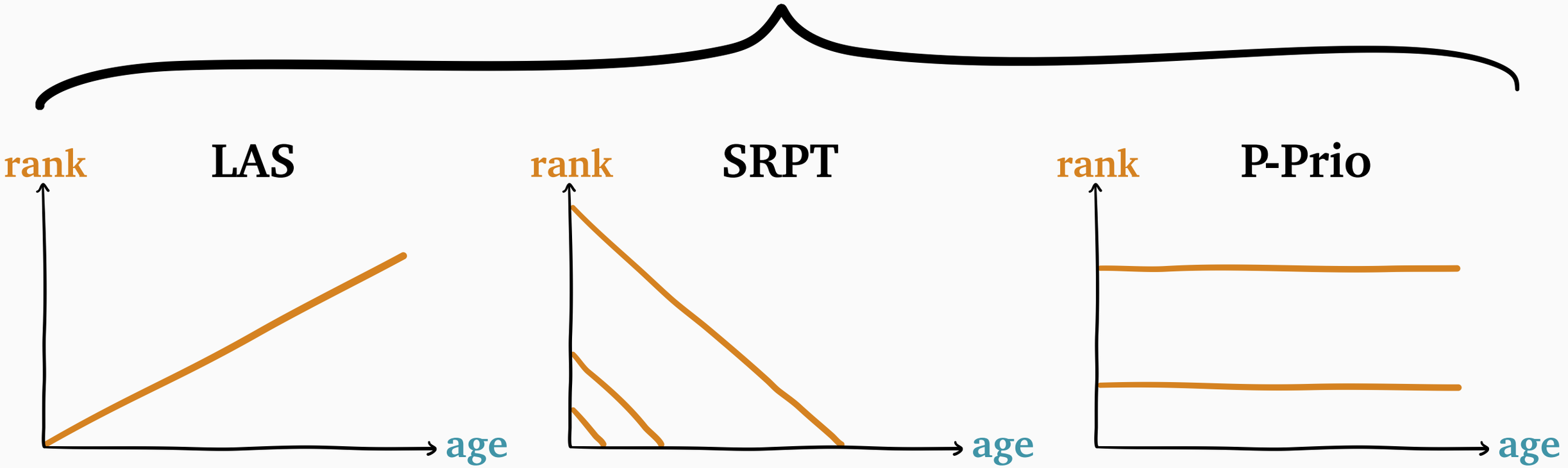


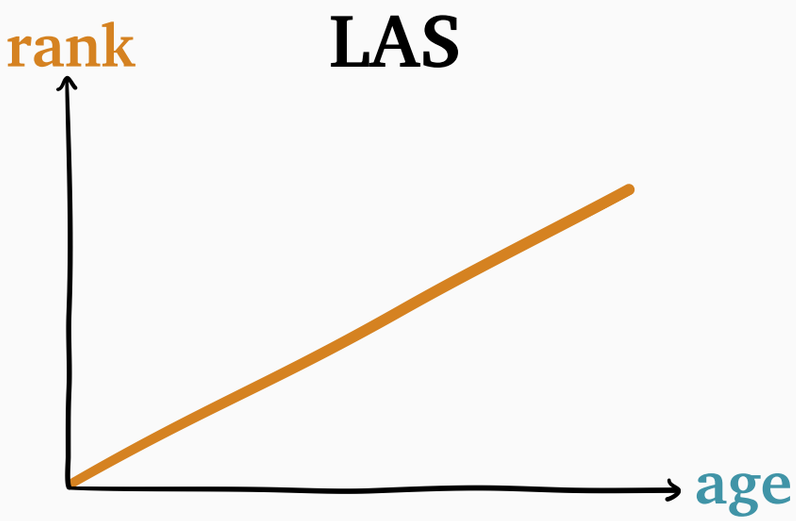
Unifying language: **rank** functions

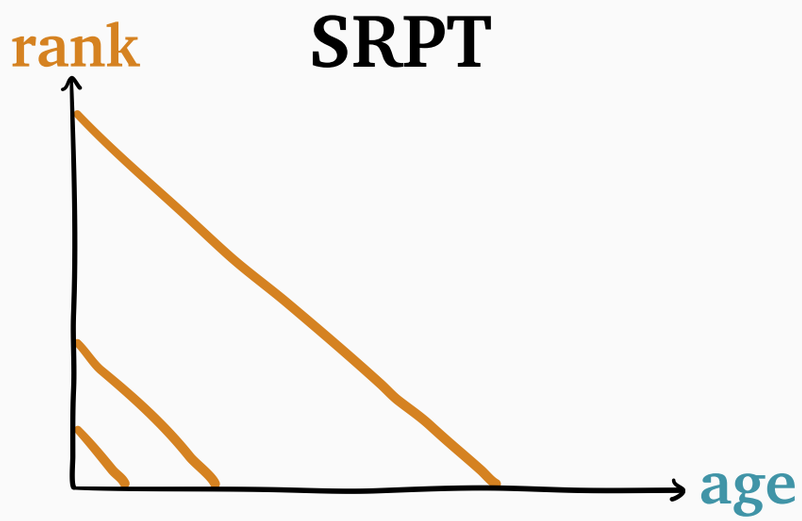
"Simple" **policy** = *monotonic* **rank** function

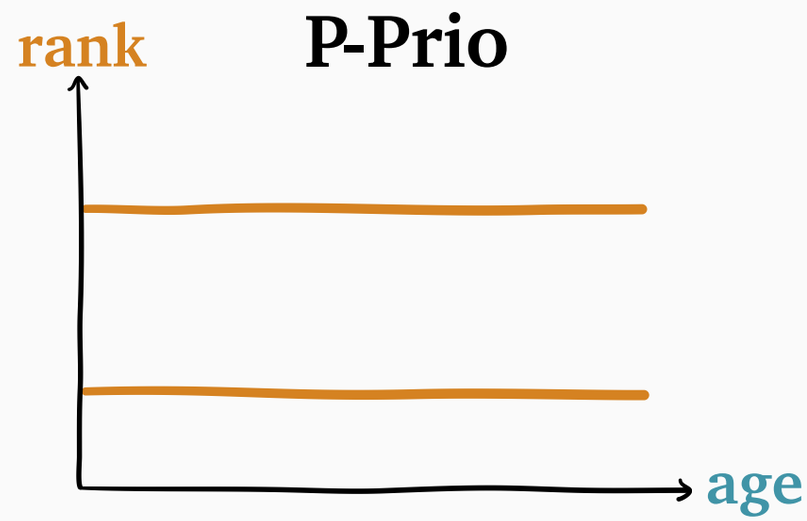


response time T known

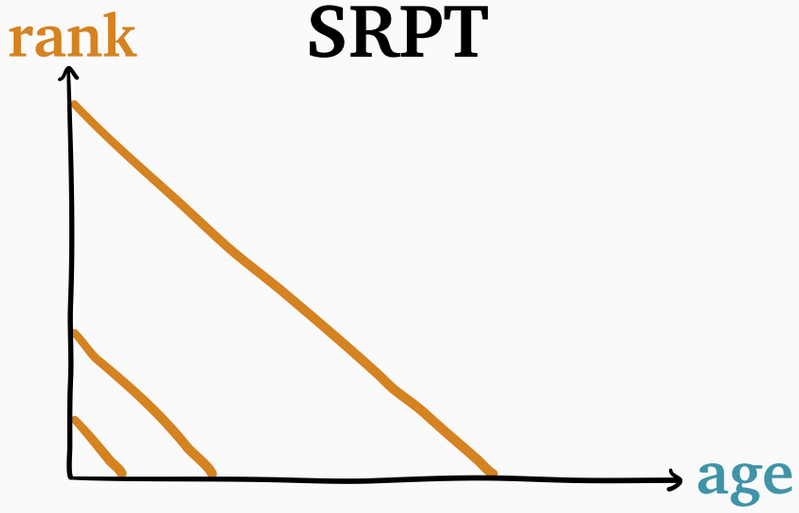


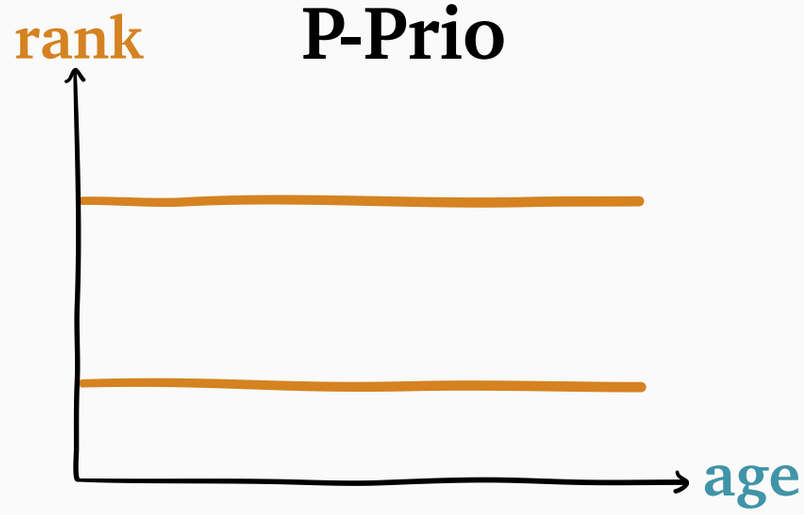






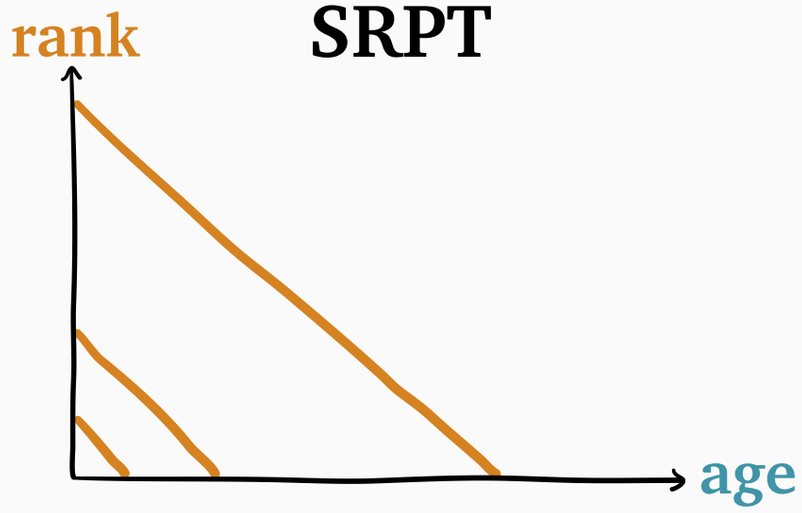
"Simple" = *monotonic* **rank** function

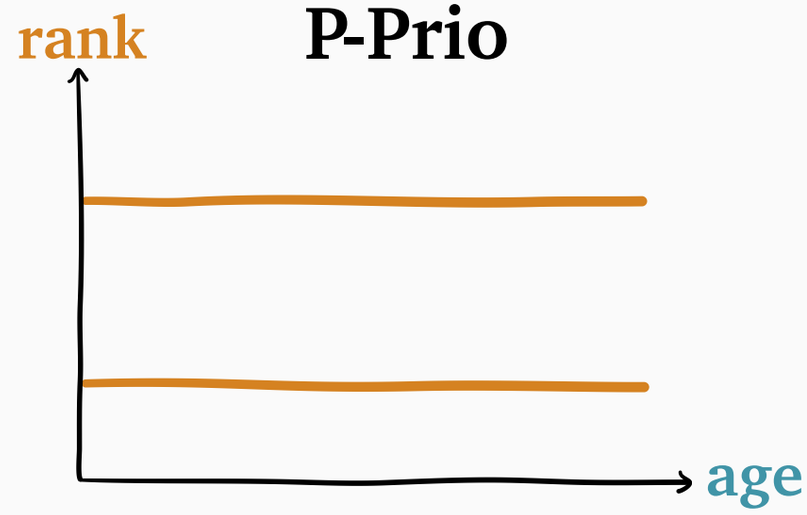




"Simple" = *monotonic* **rank** function

tractable before **SOAP**

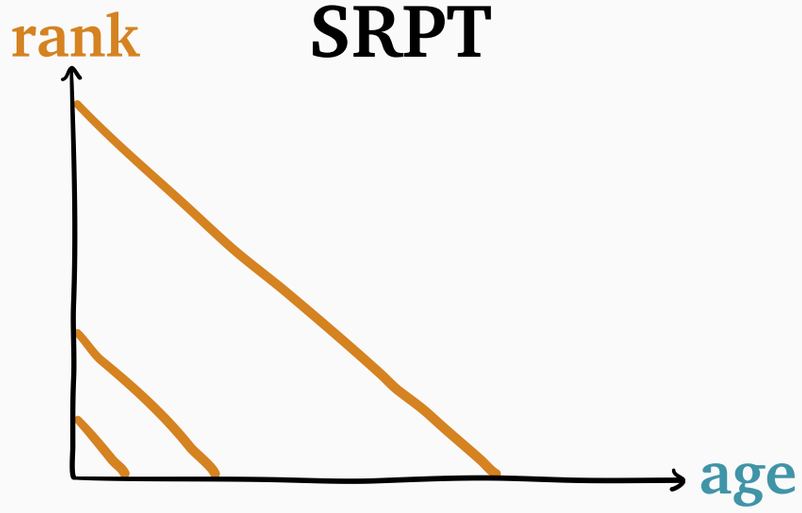


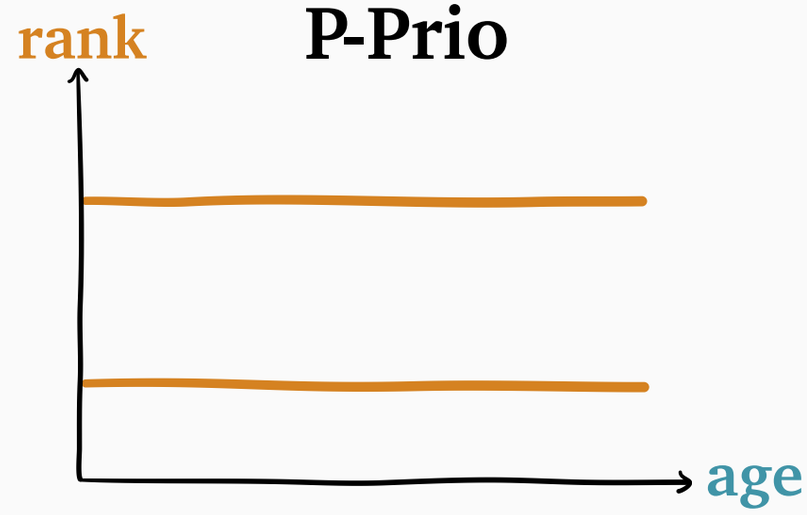


"Simple" = *monotonic* **rank** function

tractable before **SOAP**

"Complex" = *nonmonotonic* **rank** function





"Simple" = *monotonic* **rank** function

tractable before **SOAP**

"Complex" = *nonmonotonic* **rank** function

intractable before **SOAP**

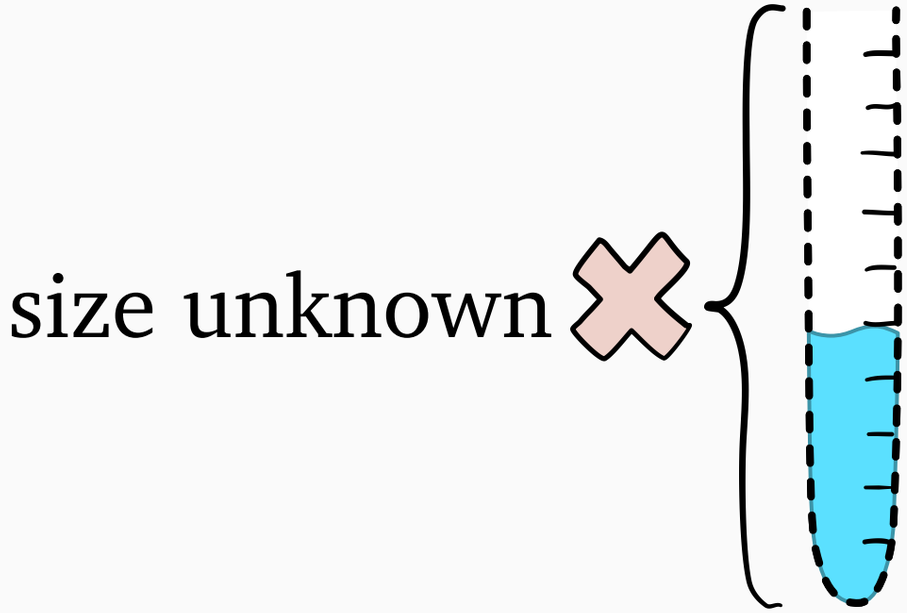
“Complex” = *nonmonotonic* **rank**

“Complex” = *nonmonotonic* **rank**

Motivation:
unknown job sizes

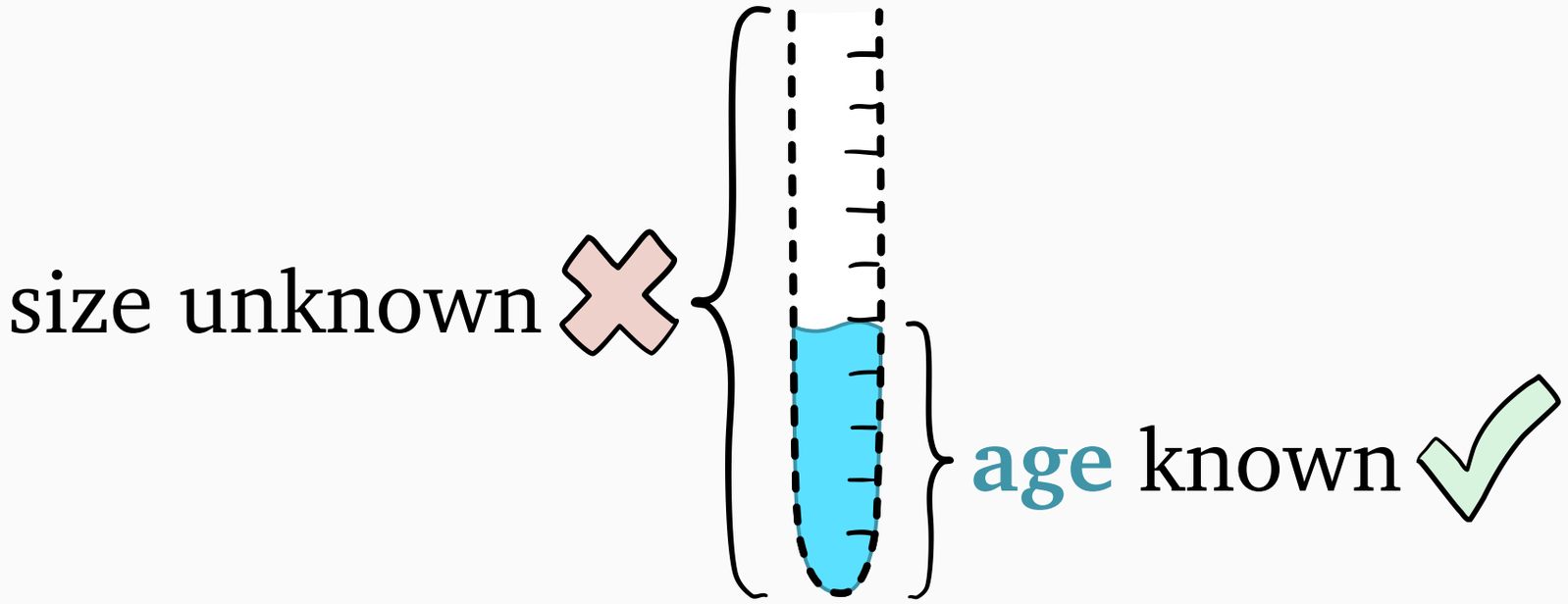
“Complex” = *nonmonotonic* **rank**

Motivation:
unknown job sizes



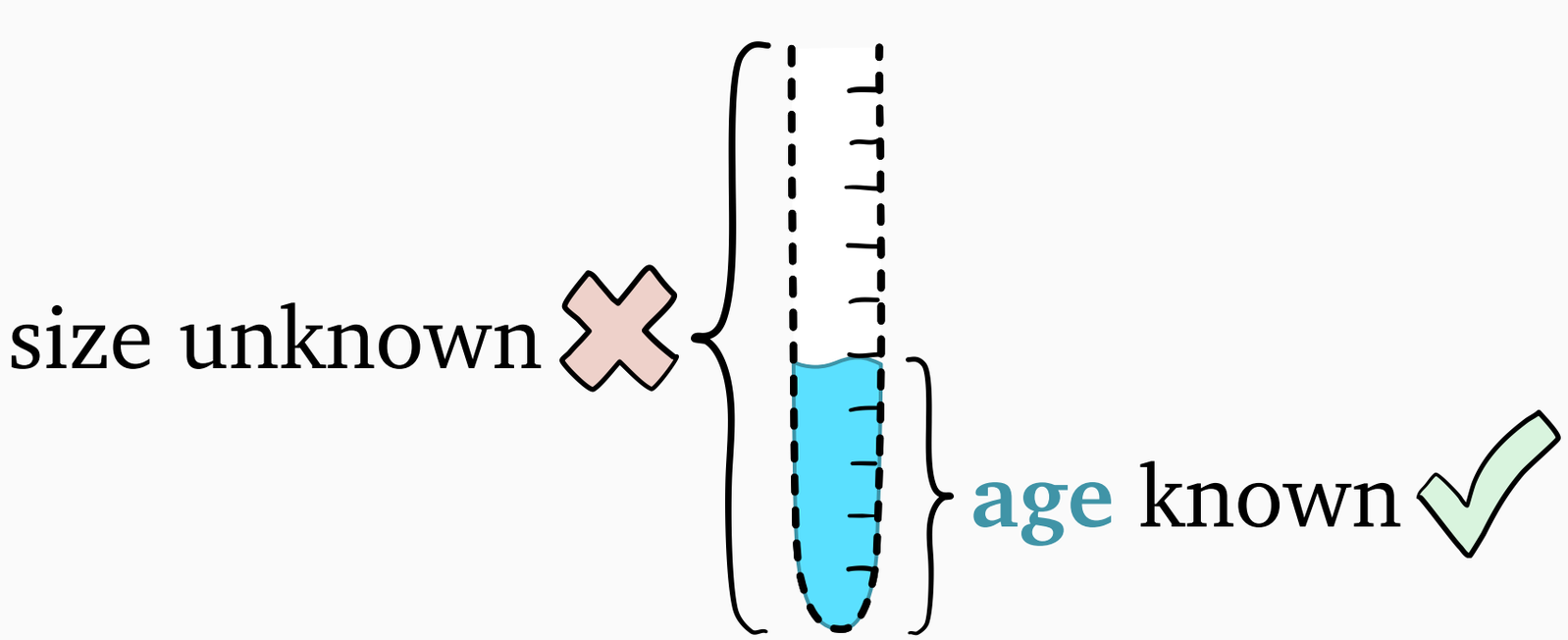
“Complex” = *nonmonotonic* **rank**

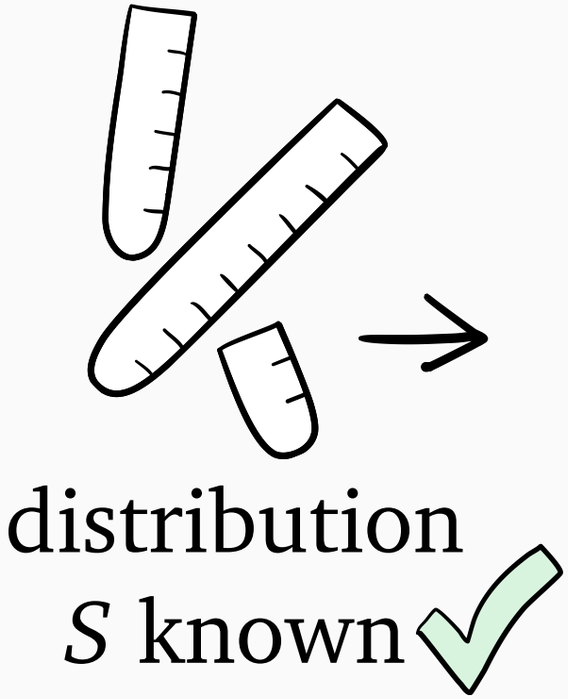
Motivation:
unknown job sizes



“Complex” = *nonmonotonic* **rank**

Motivation:
unknown job sizes



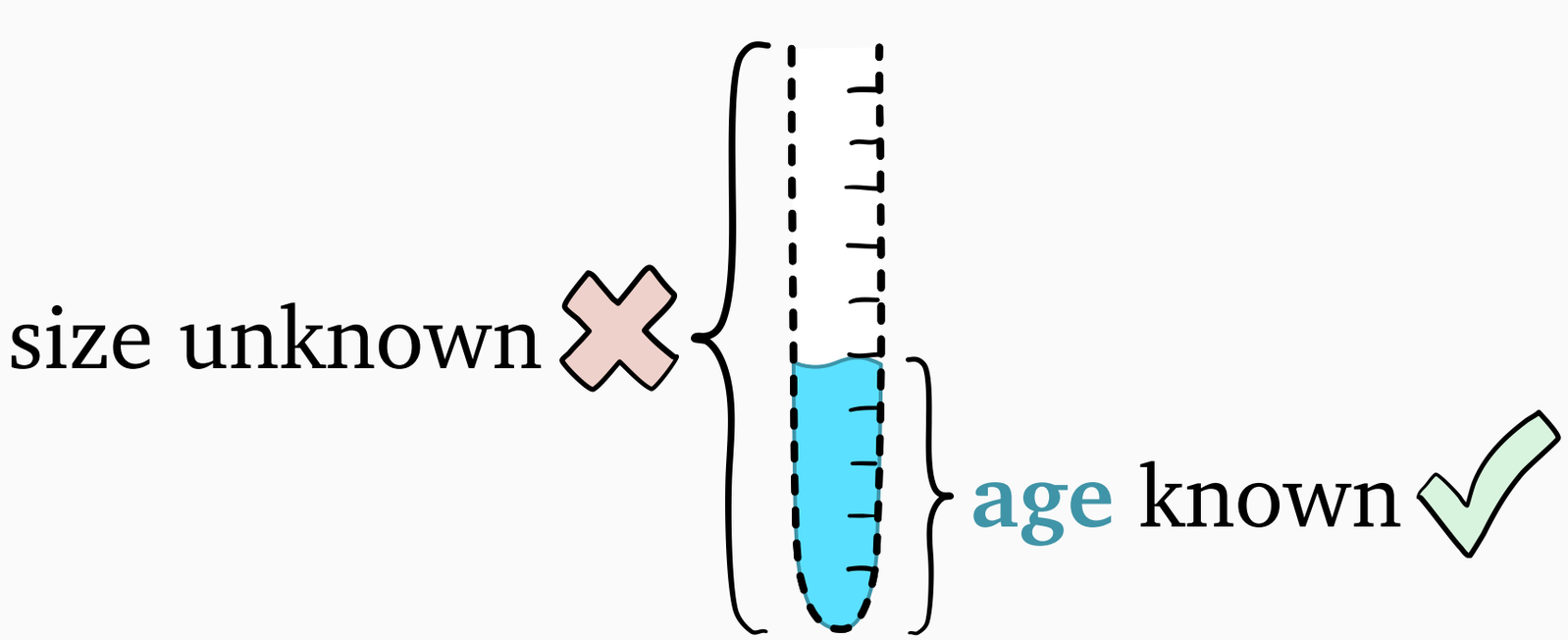


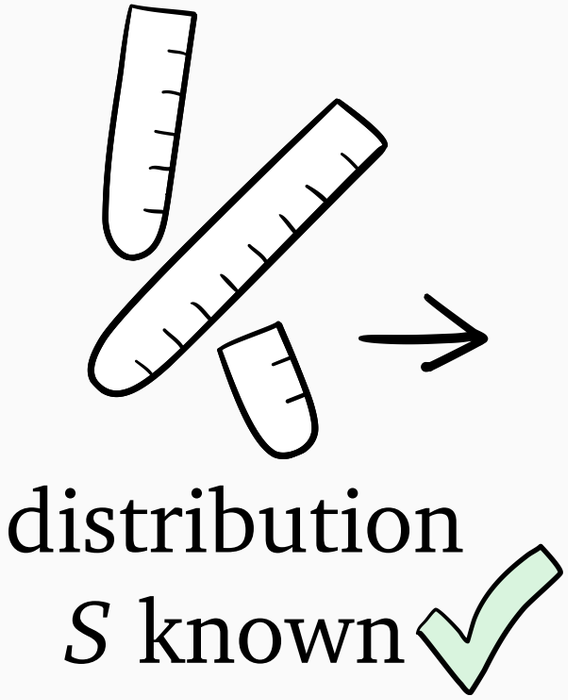
“Complex” = *nonmonotonic* **rank**

Motivation:
unknown job sizes

SERPT

shortest *expected* remaining processing time



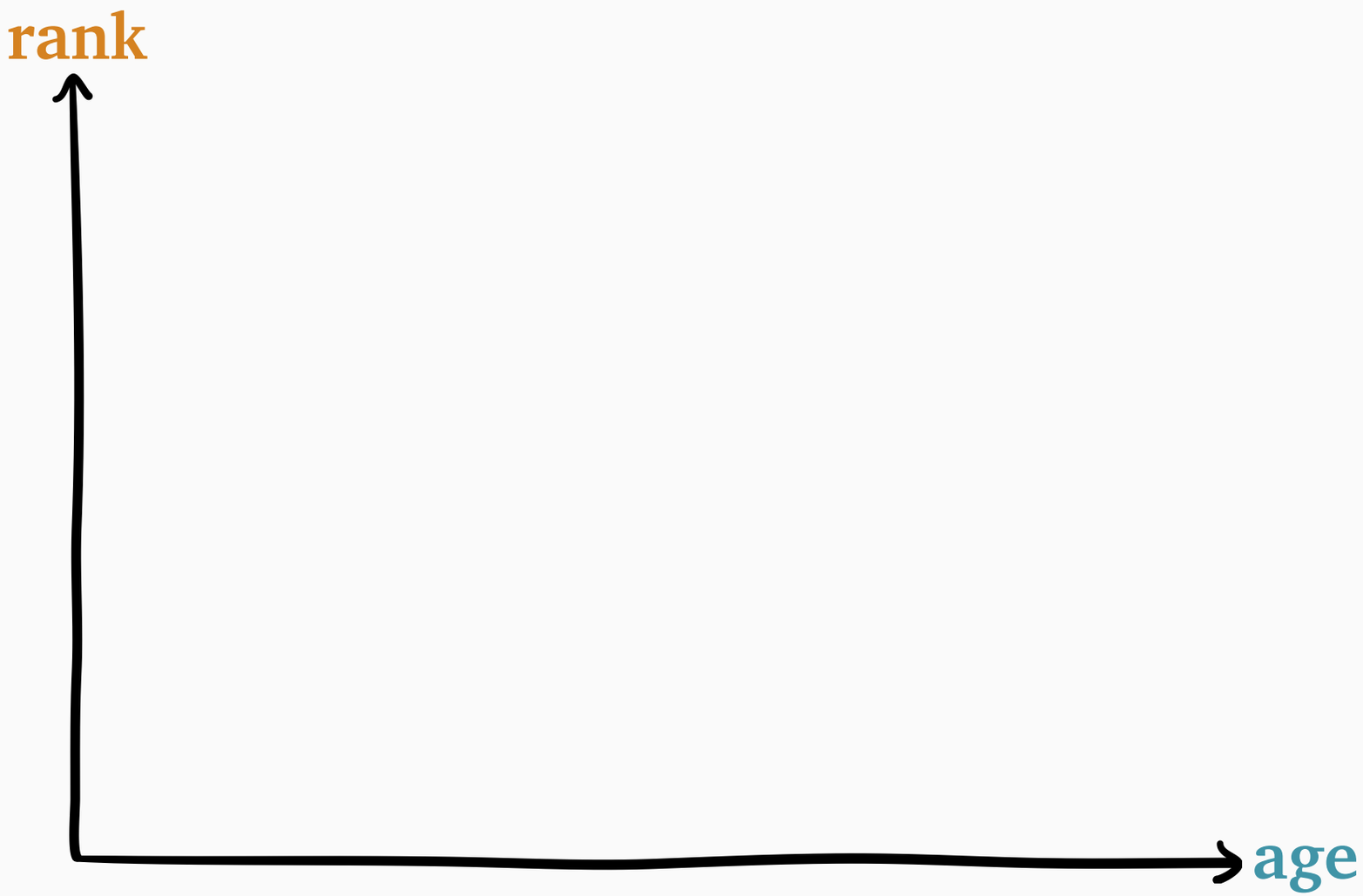


“Complex” = *nonmonotonic* **rank**

Motivation:
unknown job sizes

SERPT

shortest *expected* remaining processing time



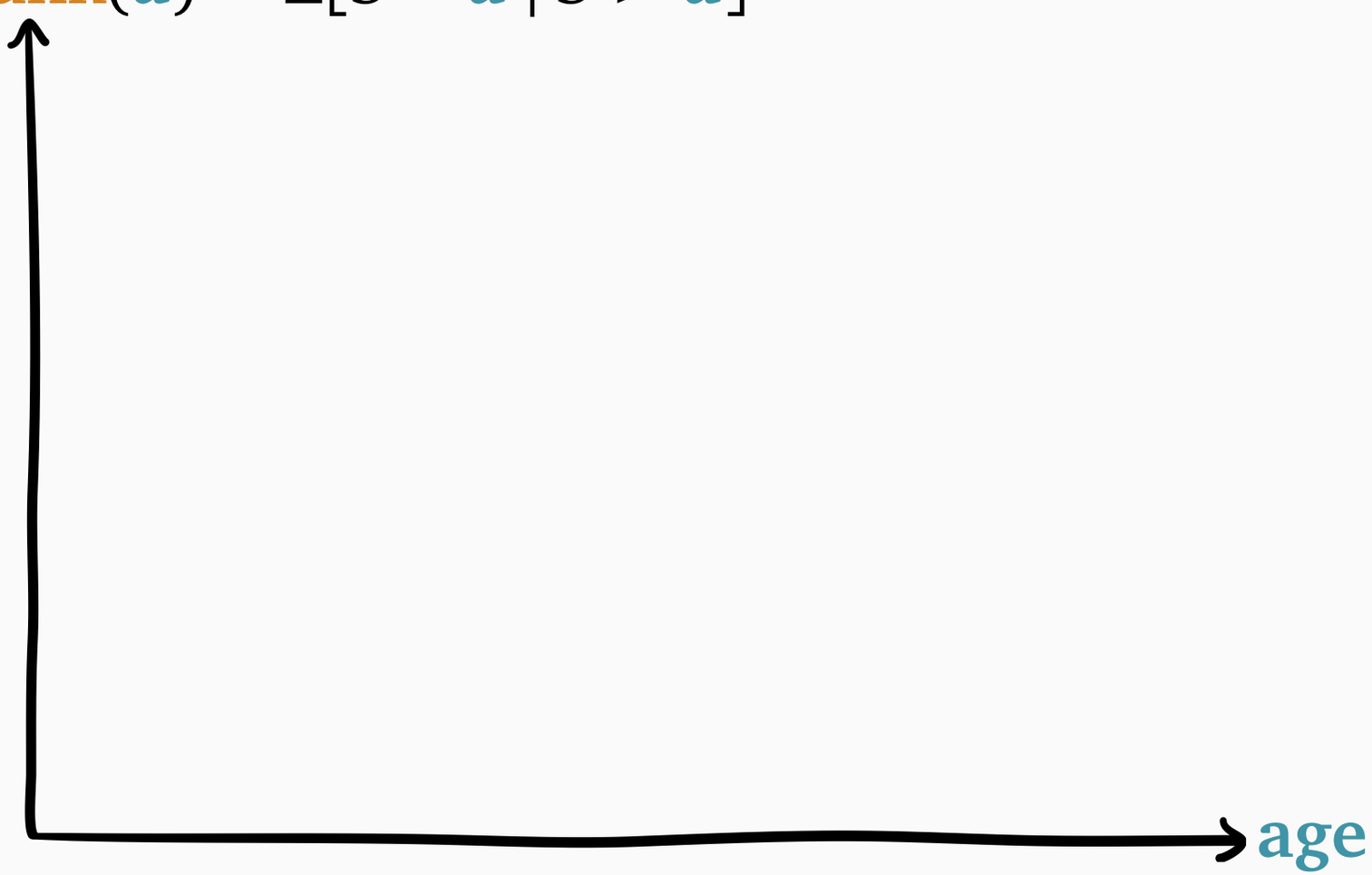
"Complex" = *nonmonotonic* **rank**

Motivation:
unknown job sizes

SERPT

shortest *expected* remaining processing time

$$\textbf{rank}(a) = \mathbf{E}[S - a \mid S > a]$$



"Complex" = *nonmonotonic* **rank**

Motivation:
unknown job sizes

SERPT

shortest *expected* remaining processing time

$$\textbf{rank}(a) = \mathbf{E}[S - a \mid S > a]$$

Job size distribution:

$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

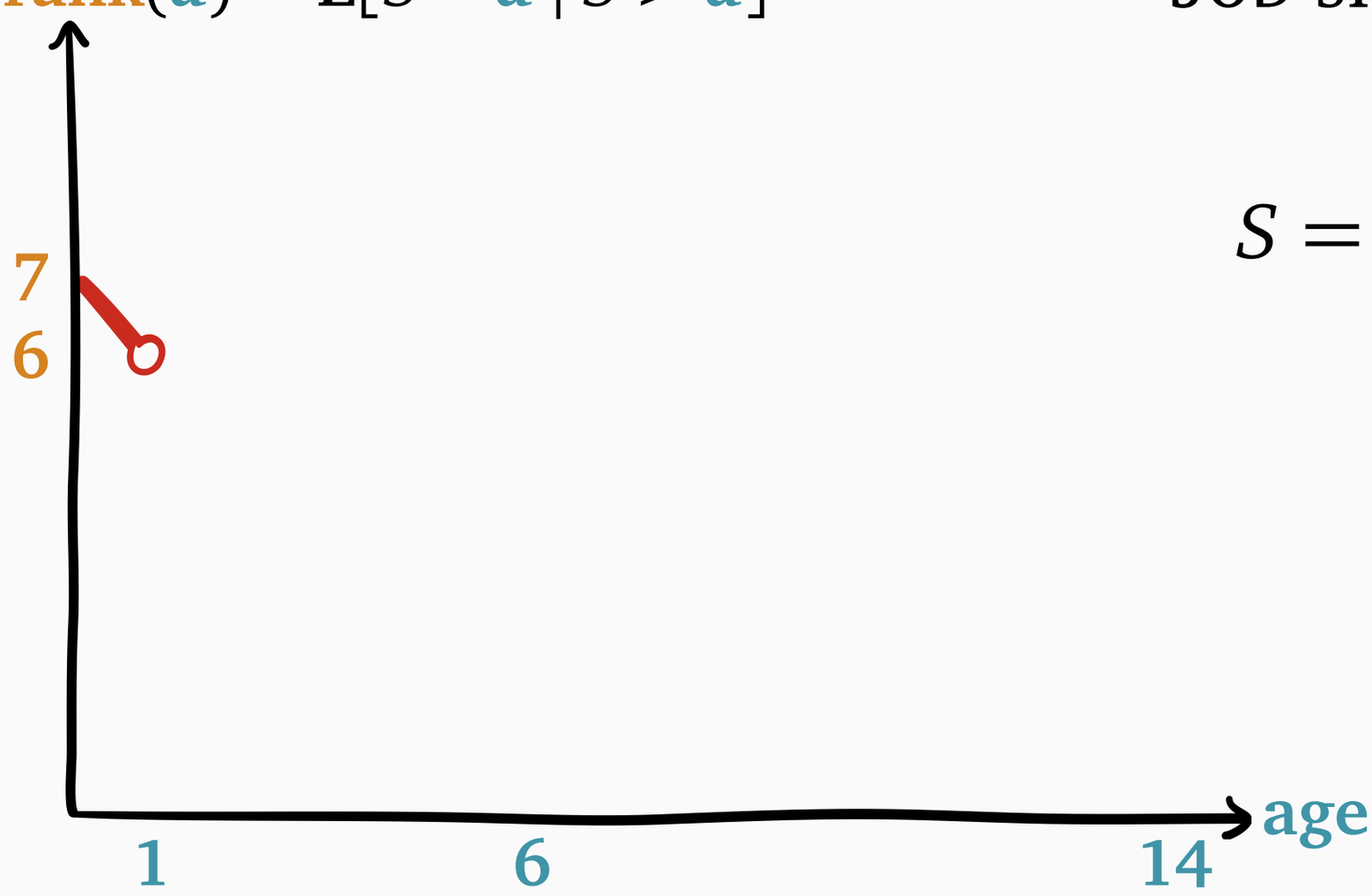
"Complex" = *nonmonotonic* **rank**

Motivation:
unknown job sizes

SERPT

shortest *expected* remaining processing time

$$\textbf{rank}(a) = \mathbf{E}[S - a \mid S > a]$$



Job size distribution:

$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

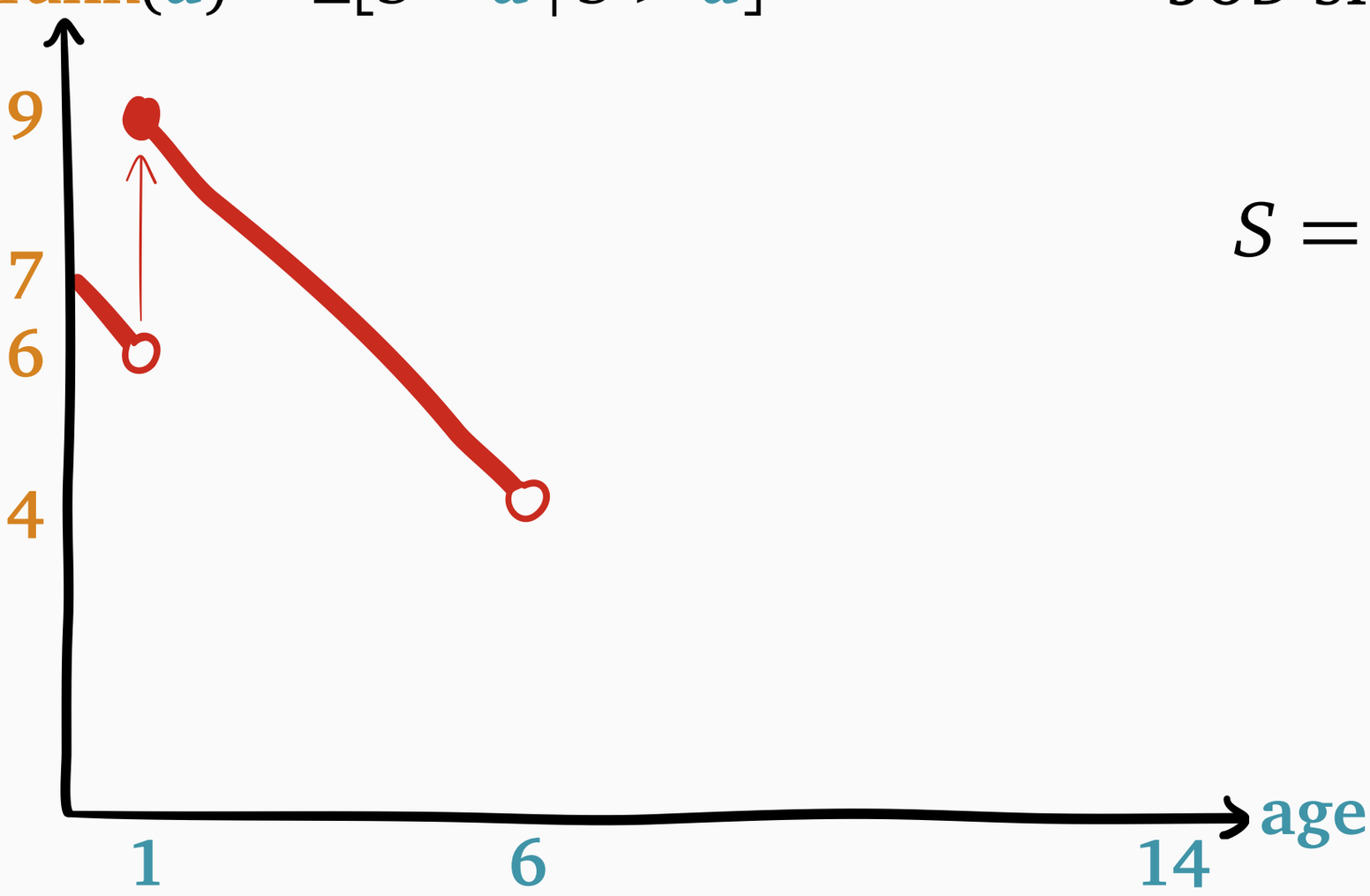
"Complex" = *nonmonotonic* **rank**

Motivation:
unknown job sizes

SERPT

shortest *expected* remaining processing time

$$\textbf{rank}(a) = \mathbf{E}[S - a \mid S > a]$$



Job size distribution:

$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

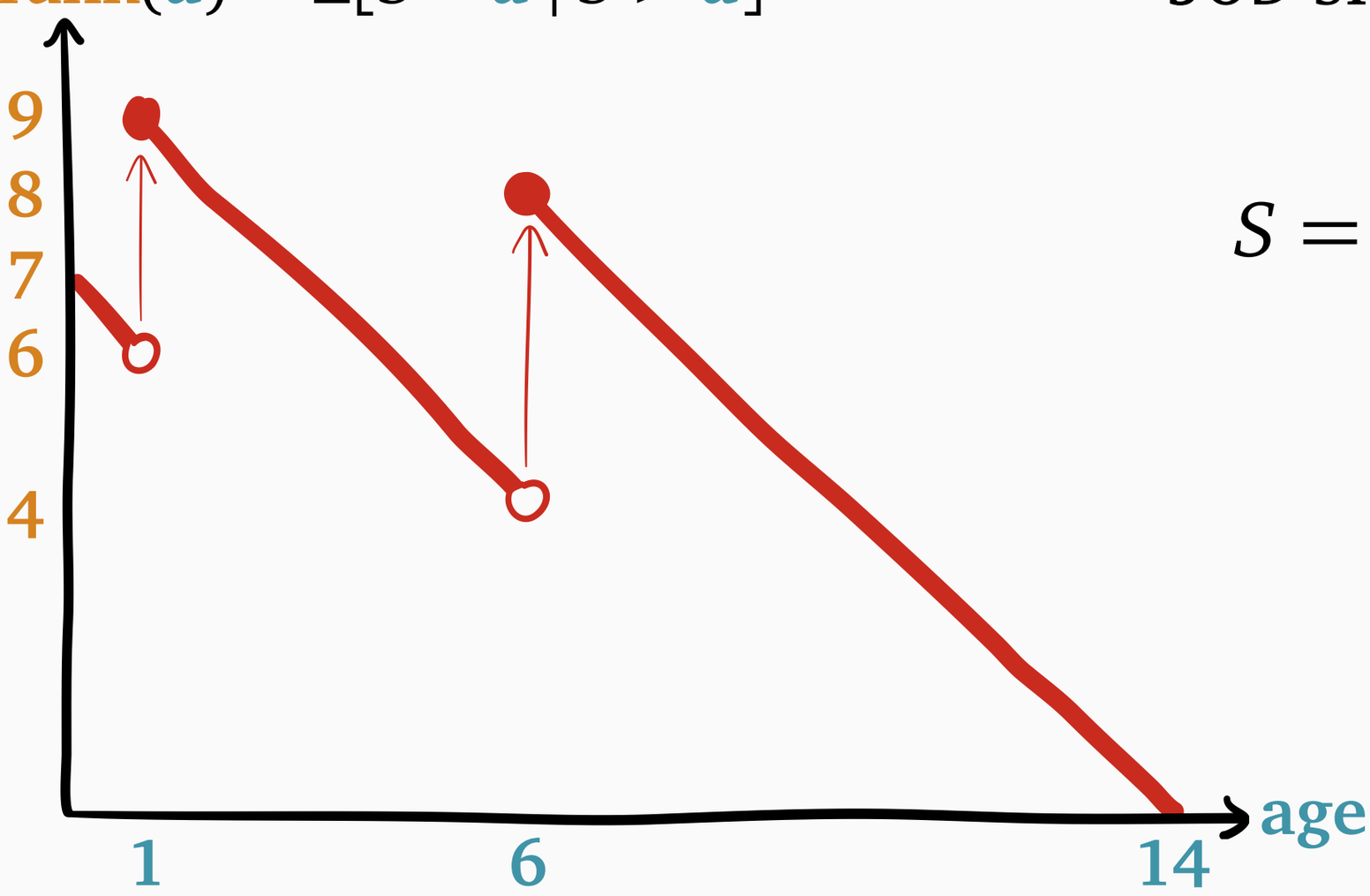
"Complex" = *nonmonotonic* **rank**

Motivation:
unknown job sizes

SERPT

shortest *expected* remaining processing time

$$\textbf{rank}(a) = \mathbf{E}[S - a \mid S > a]$$



Job size distribution:

$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

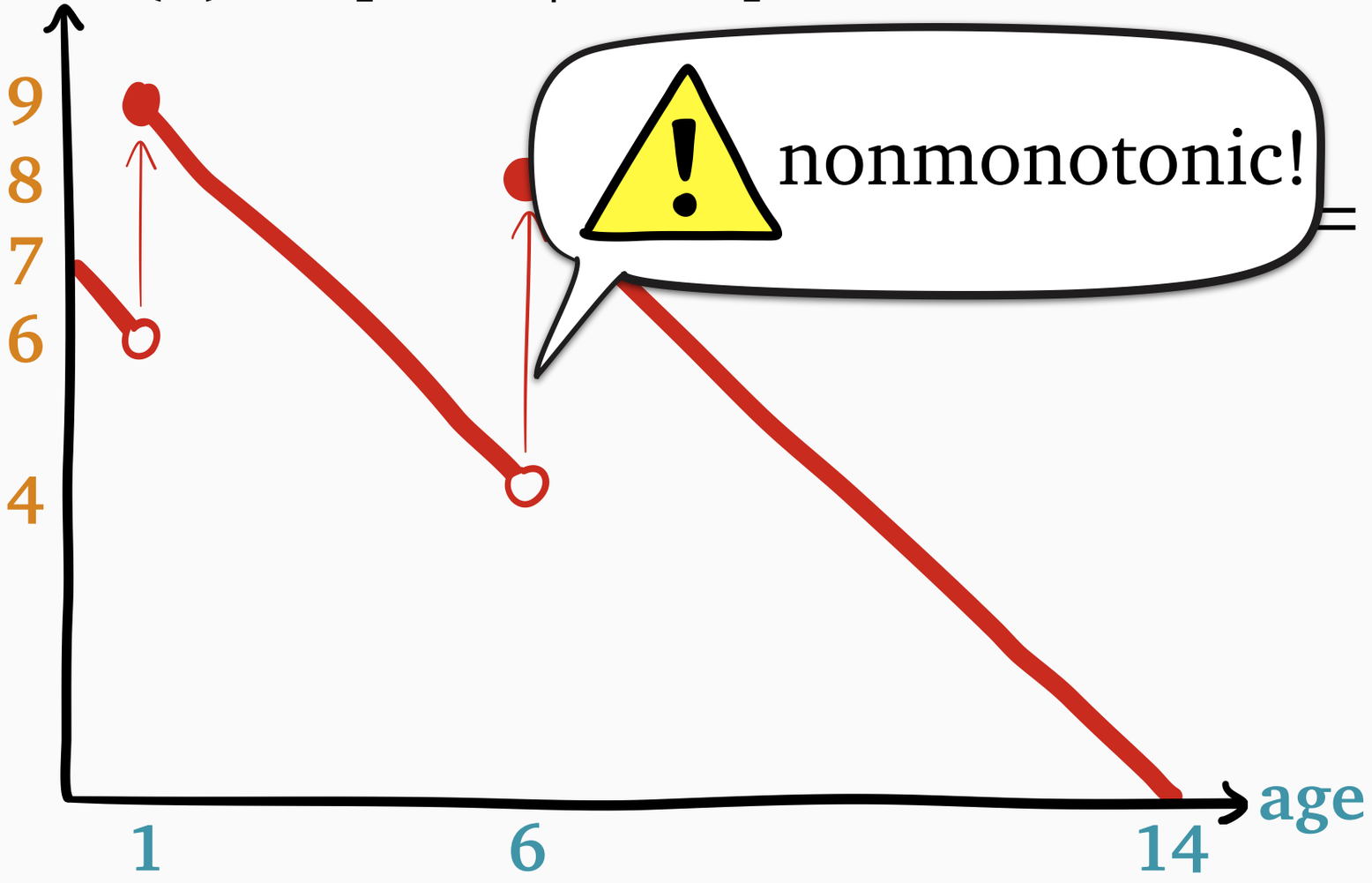
"Complex" = *nonmonotonic* **rank**

Motivation:
unknown job sizes

SERPT

shortest *expected* remaining processing time

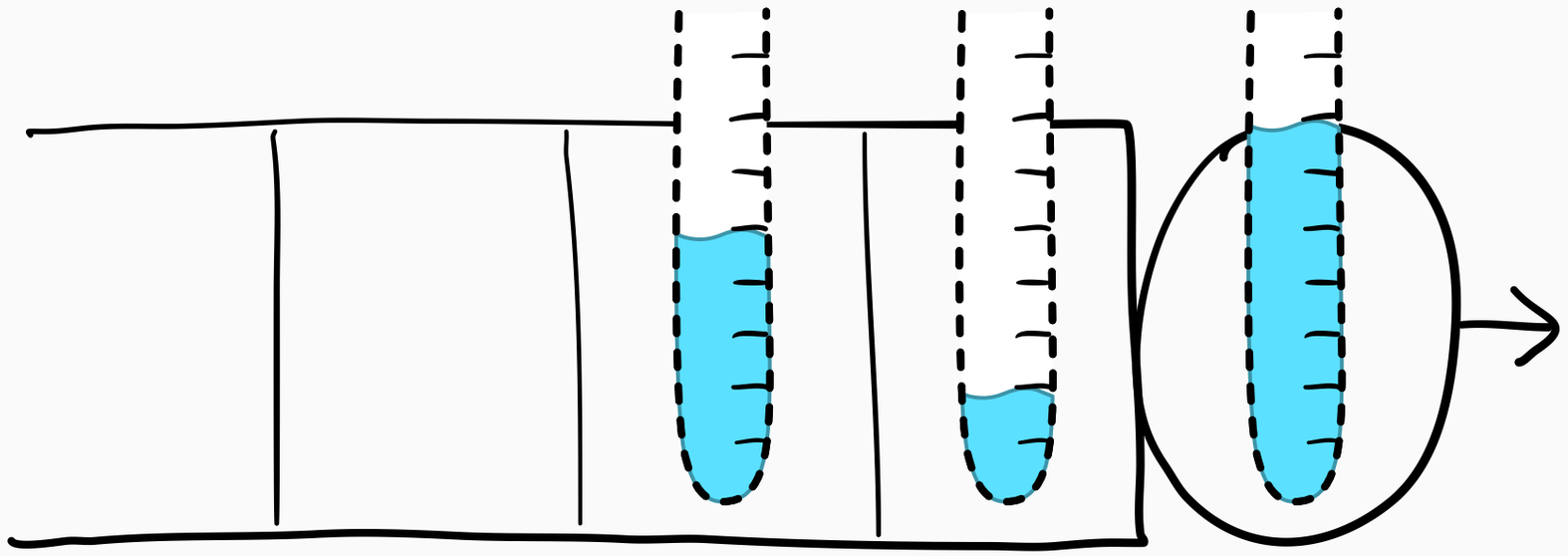
$$\textbf{rank}(a) = \mathbf{E}[S - a \mid S > a]$$



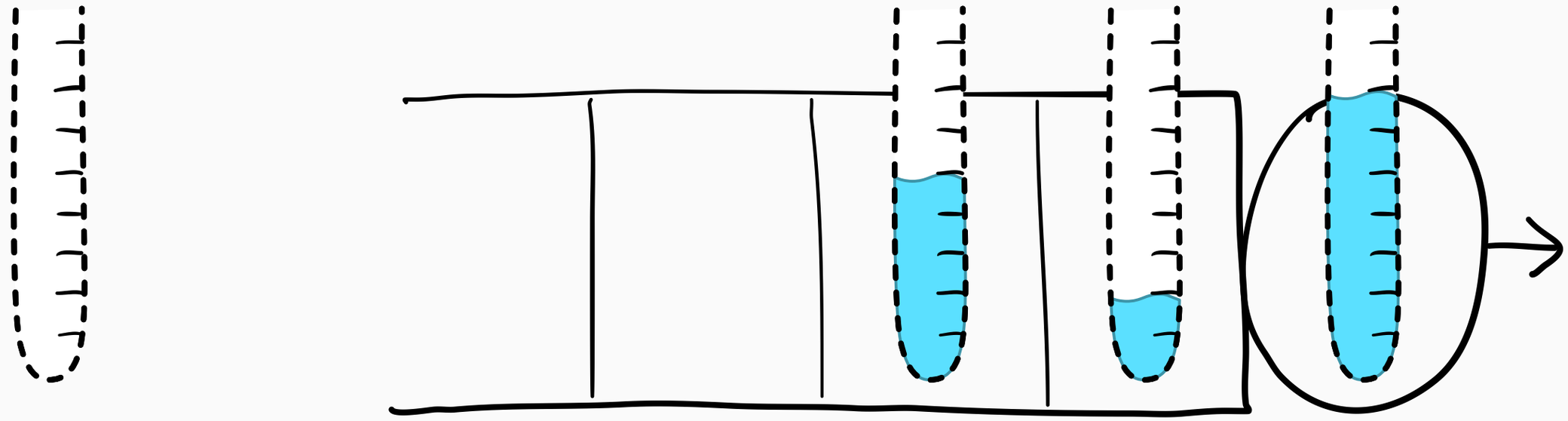
Job size distribution:

$$= \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

Nonmonotonic **rank** is hard

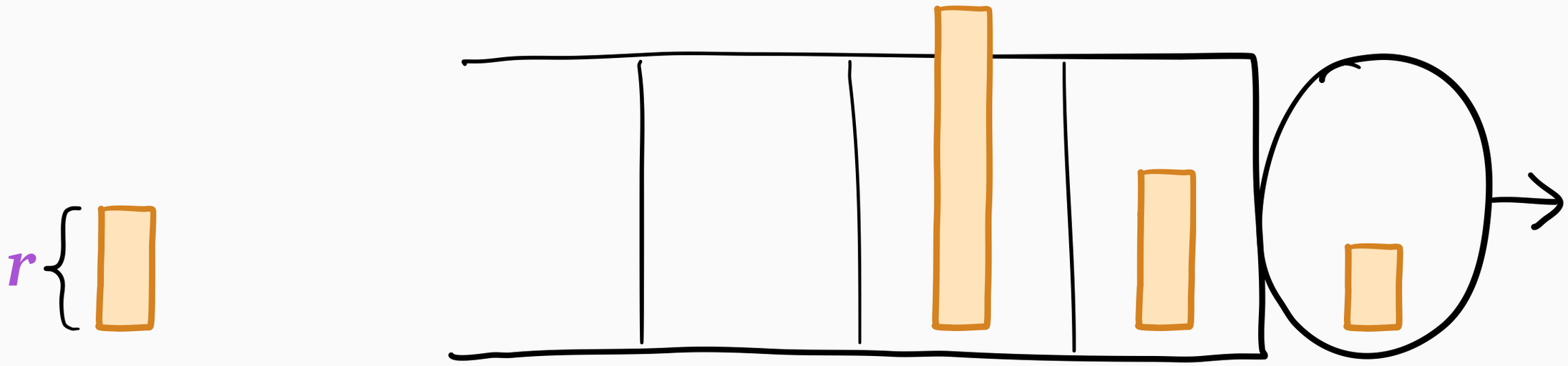


Nonmonotonic **rank** is hard



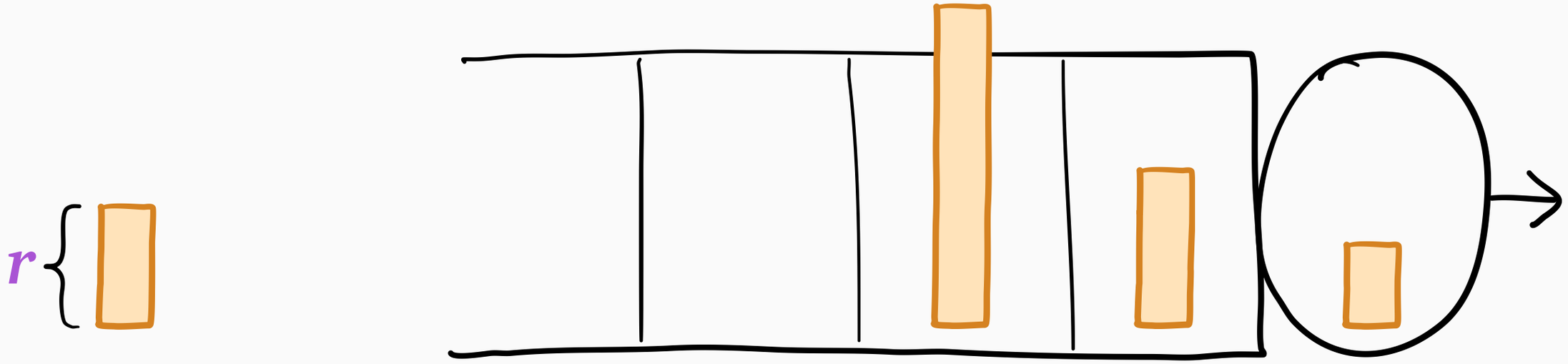
“me”

Nonmonotonic **rank** is hard



“me” = **rank**

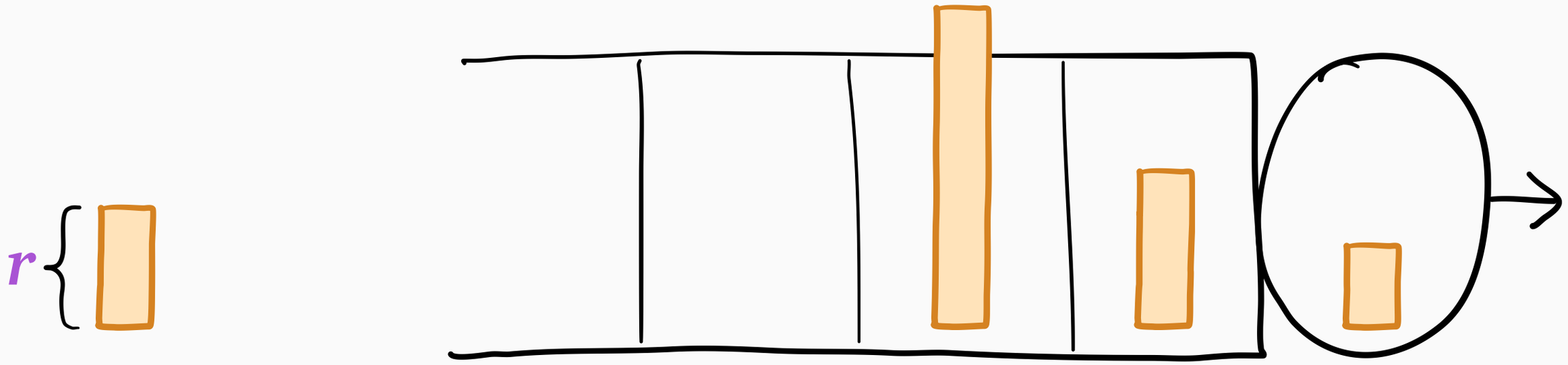
Nonmonotonic **rank** is hard



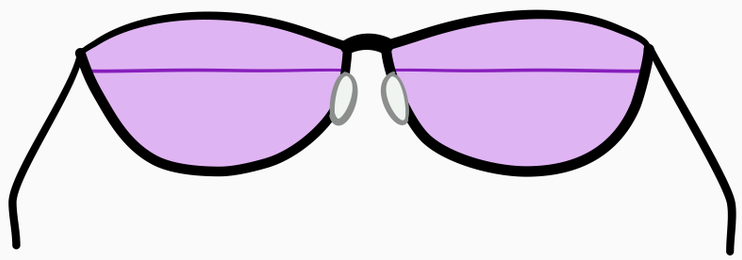
Key quantity: observed ***r*-*work*** $W(r)$

$W(r)$ = work relevant to job of **rank** r

Nonmonotonic **rank** is hard



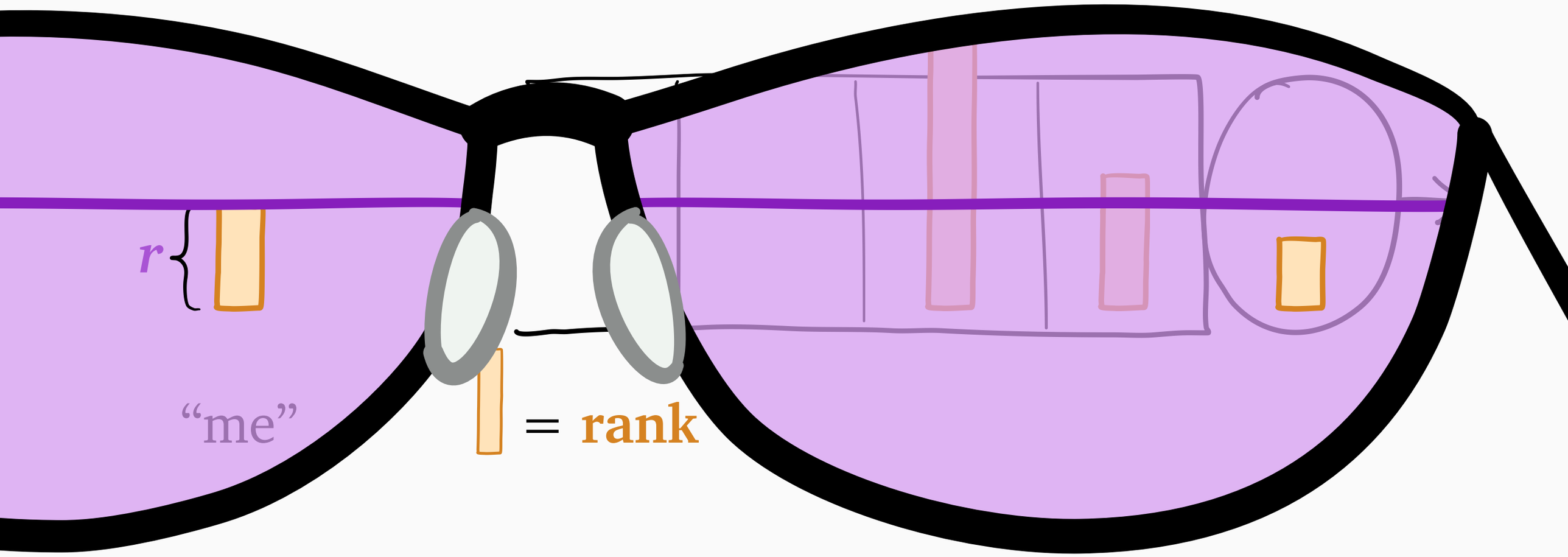
“me” = **rank**



Key quantity: observed ***r*-*work*** $W(r)$

$W(r)$ = work relevant to job of **rank *r***

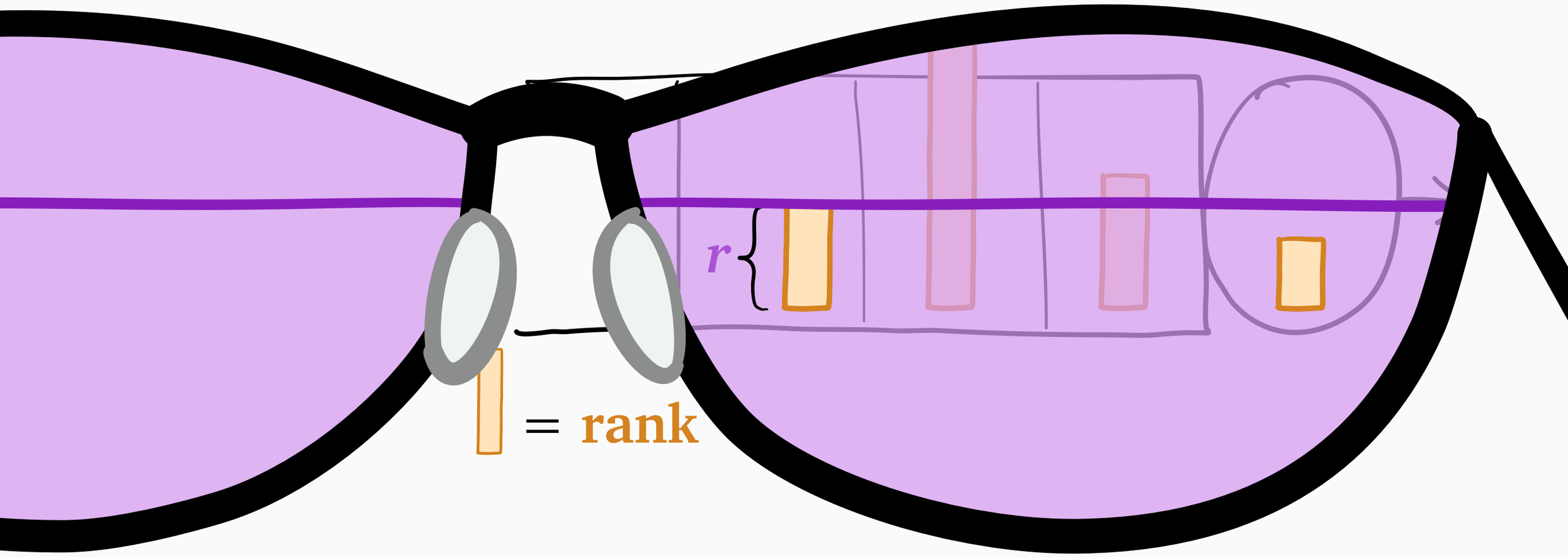
Nonmonotonic **rank** is hard



Key quantity: observed ***r*-work** $W(r)$

$W(r)$ = work relevant to job of **rank** r

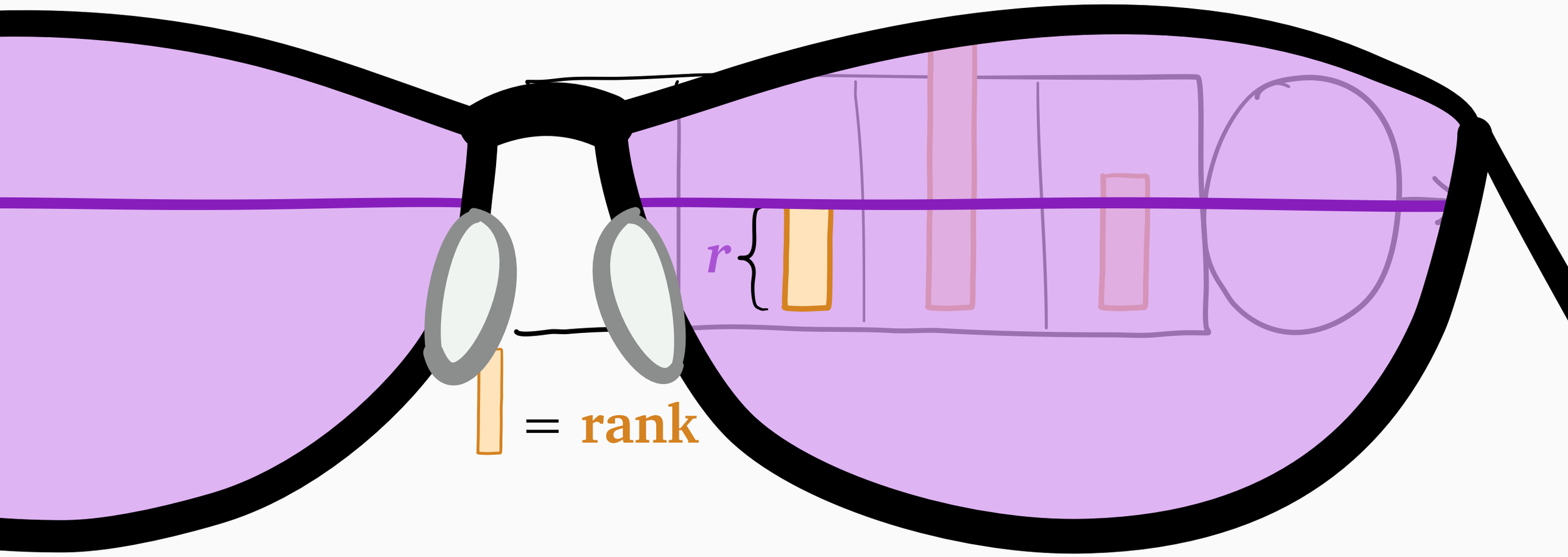
Nonmonotonic **rank** is hard



Key quantity: observed ***r*-work** $W(r)$

$W(r)$ = work relevant to job of **rank** r

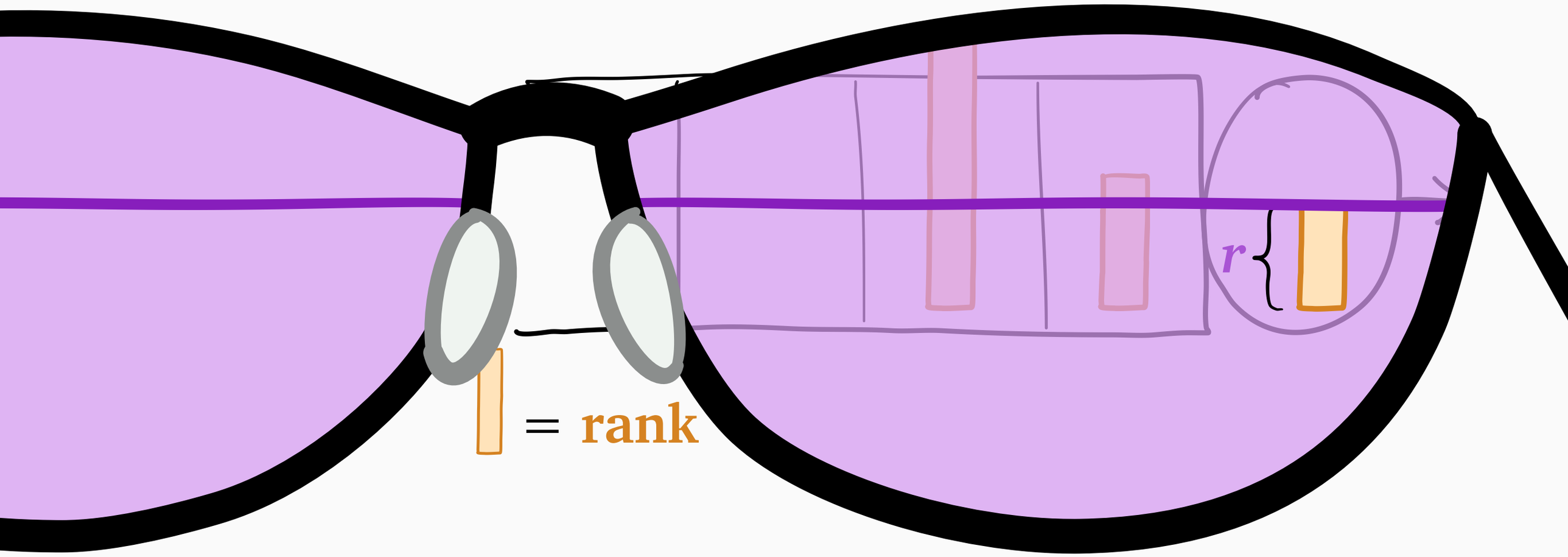
Nonmonotonic **rank** is hard



Key quantity: observed ***r*-work** $W(r)$

$W(r)$ = work relevant to job of **rank** r

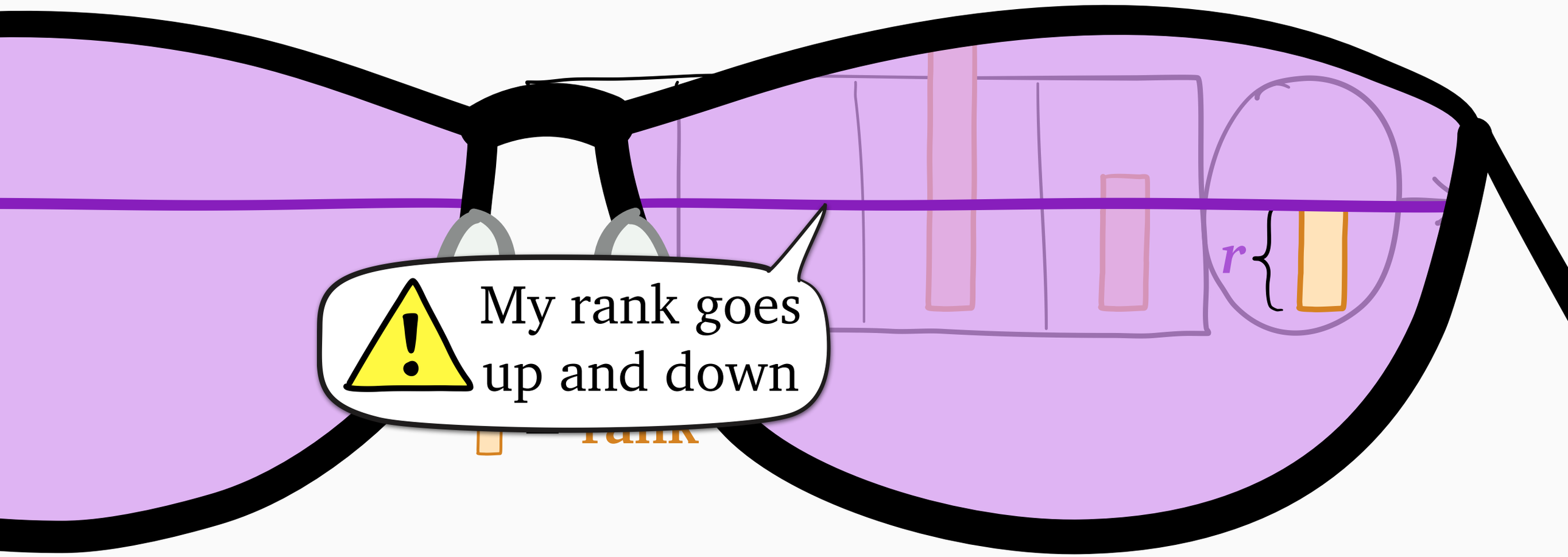
Nonmonotonic **rank** is hard



Key quantity: observed ***r*-work** $W(r)$

$W(r)$ = work relevant to job of **rank** r

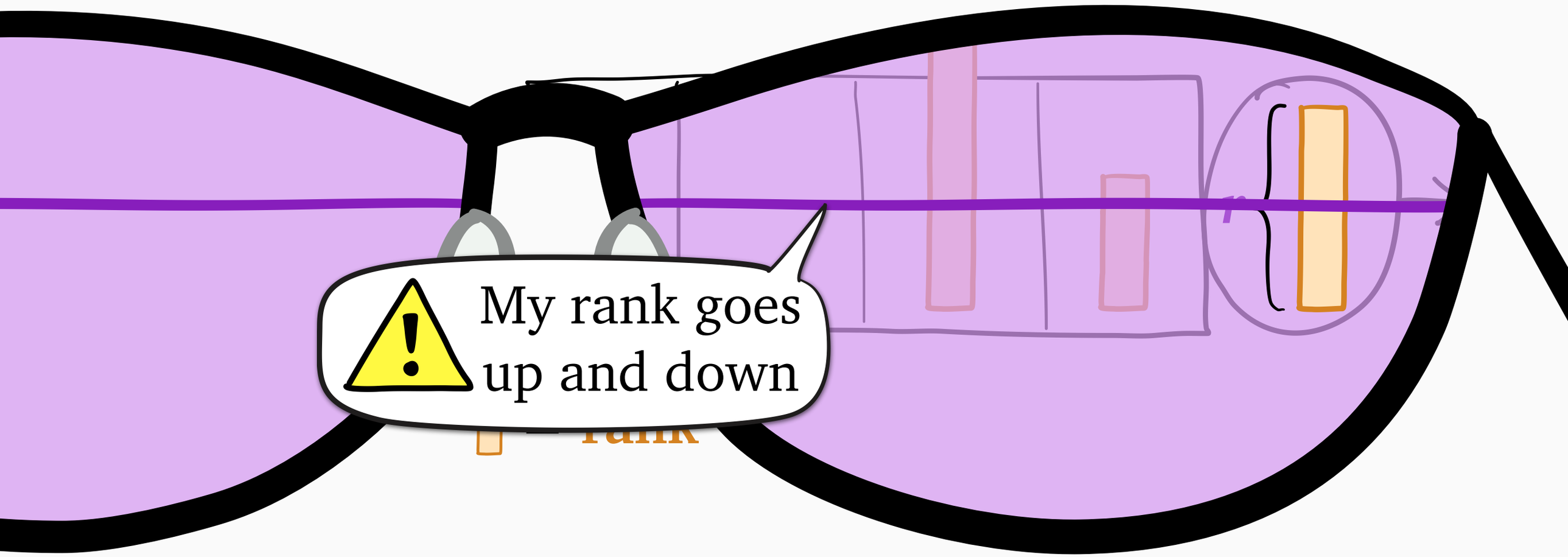
Nonmonotonic **rank** is hard



Key quantity: observed ***r*-work** $W(r)$

$W(r)$ = work relevant to job of **rank** r

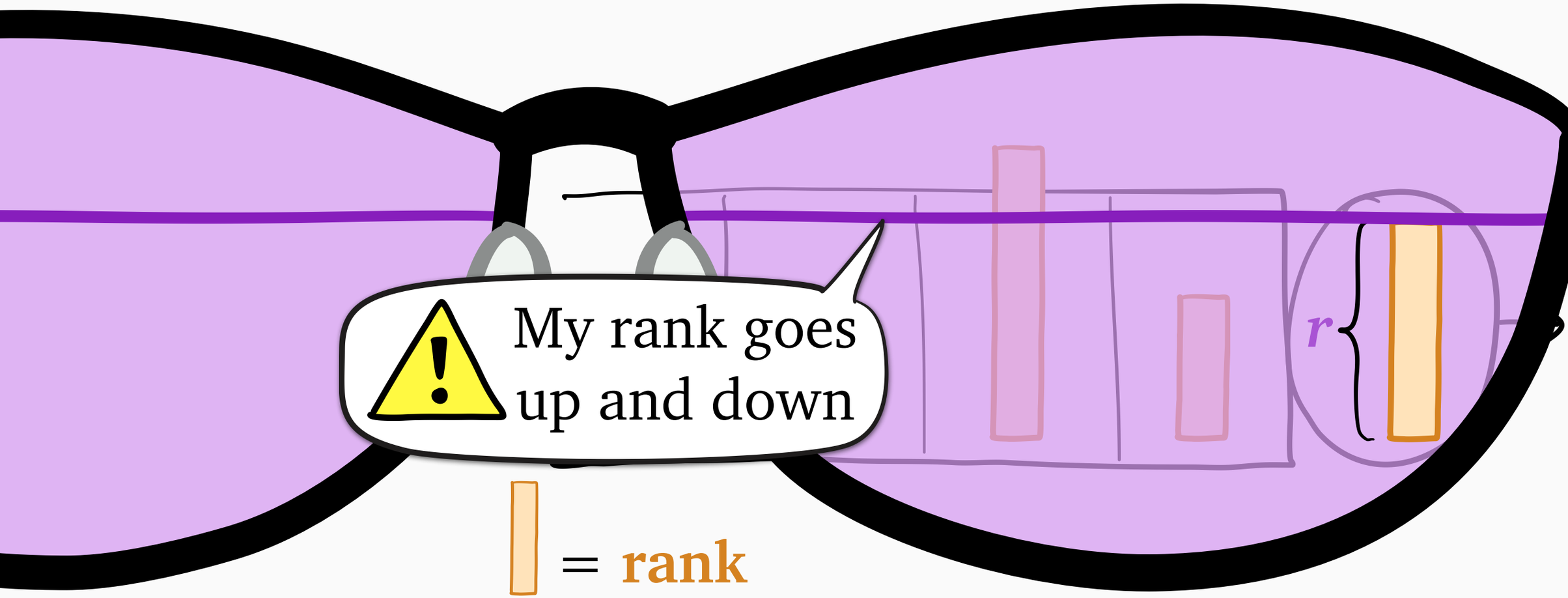
Nonmonotonic **rank** is hard



Key quantity: observed ***r*-work** $W(r)$

$W(r)$ = work relevant to job of **rank** r

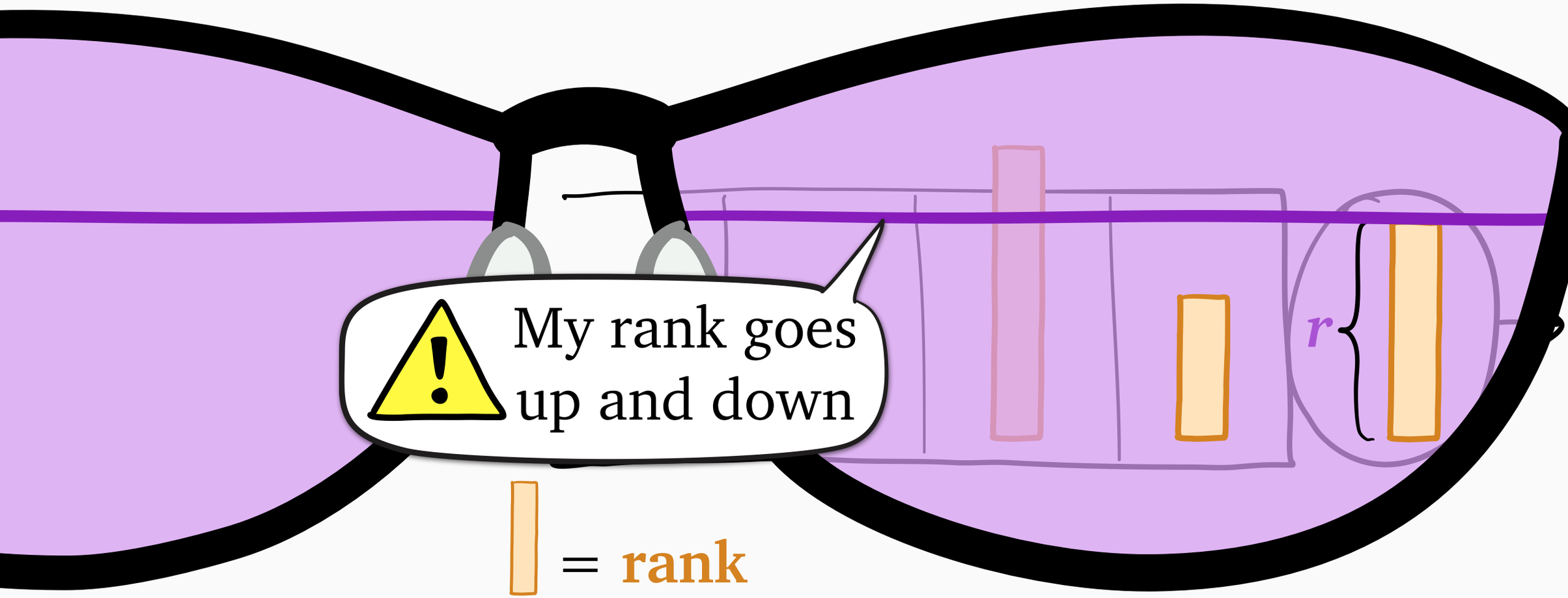
Nonmonotonic **rank** is hard



Key quantity: observed r-*work* $W(r)$

$W(r)$ = work relevant to job of **rank** r

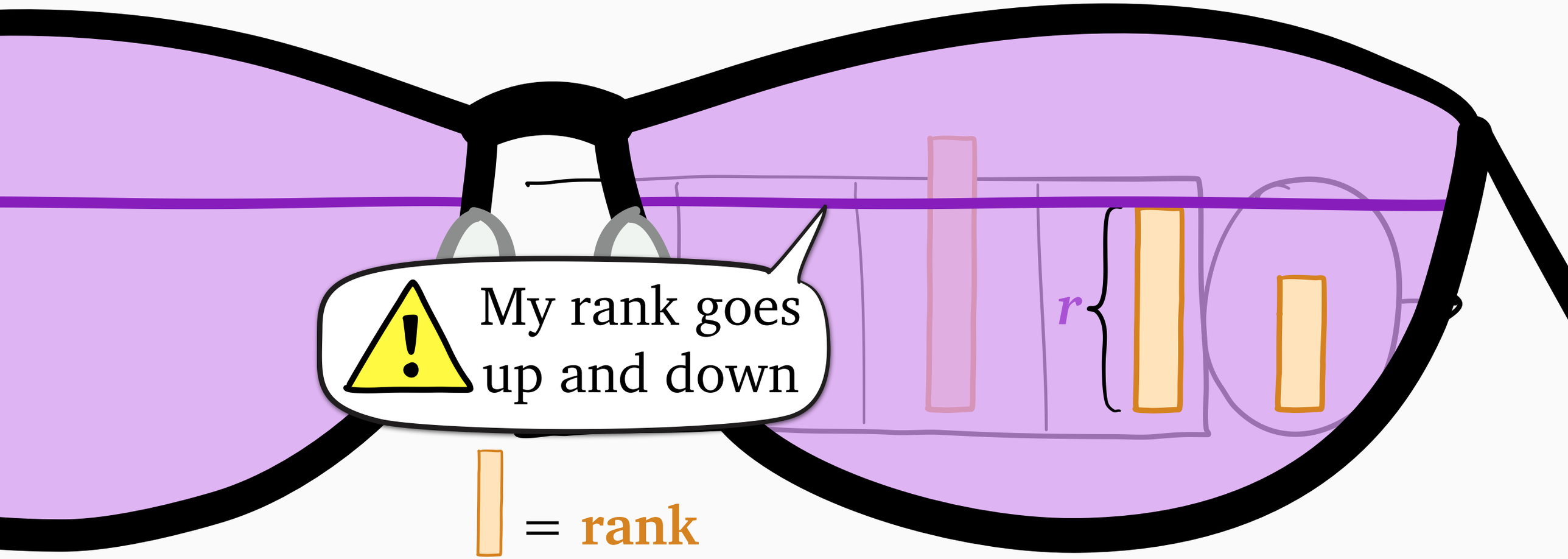
Nonmonotonic **rank** is hard



Key quantity: observed r-*work* $W(r)$

$W(r)$ = work relevant to job of **rank** r

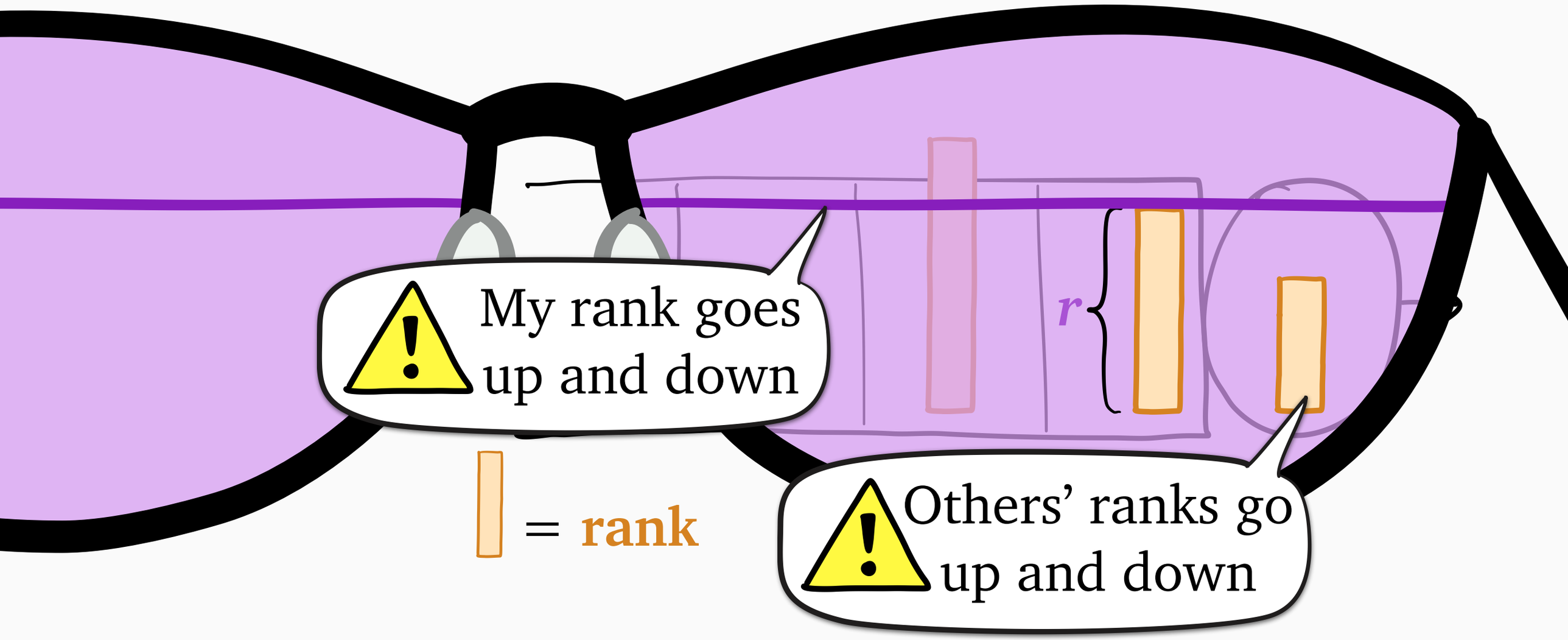
Nonmonotonic **rank** is hard



Key quantity: observed *r-work* $W(r)$

$W(r)$ = work relevant to job of **rank** r

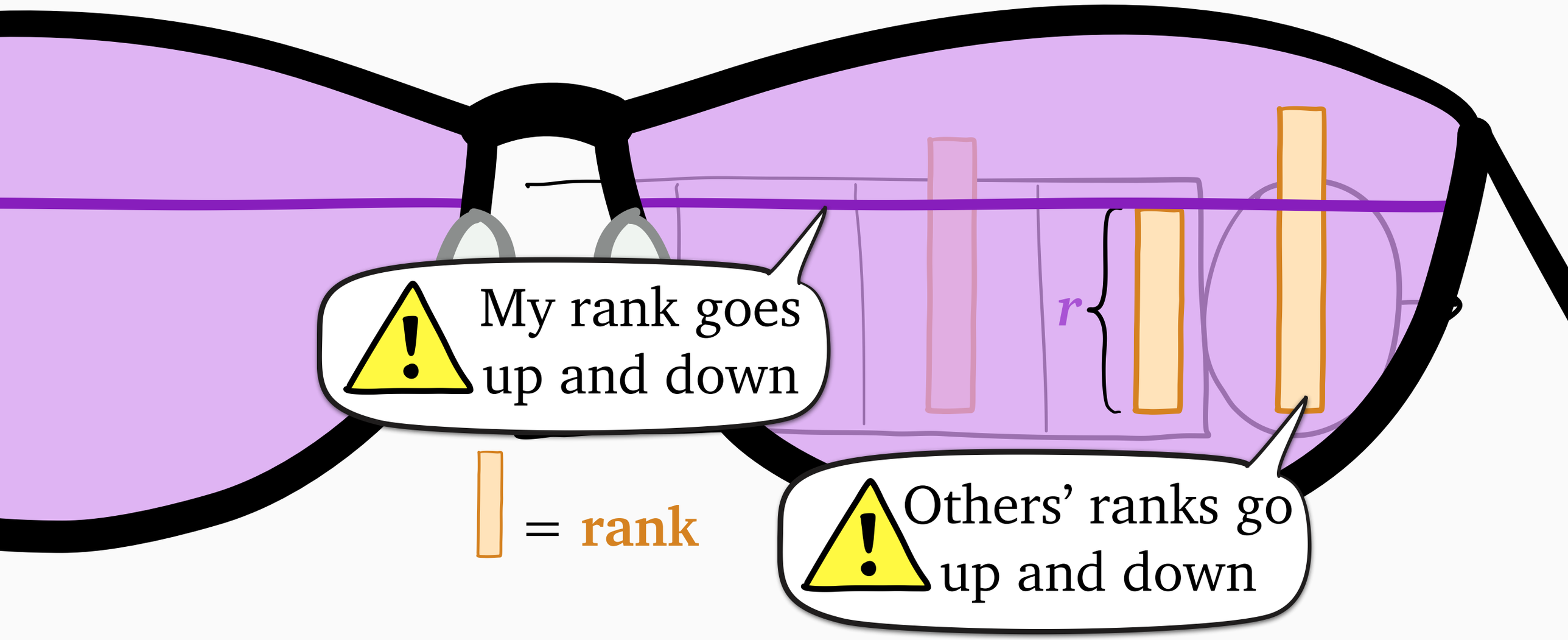
Nonmonotonic **rank** is hard



Key quantity: observed *r-work* $W(r)$

$W(r)$ = work relevant to job of **rank** r

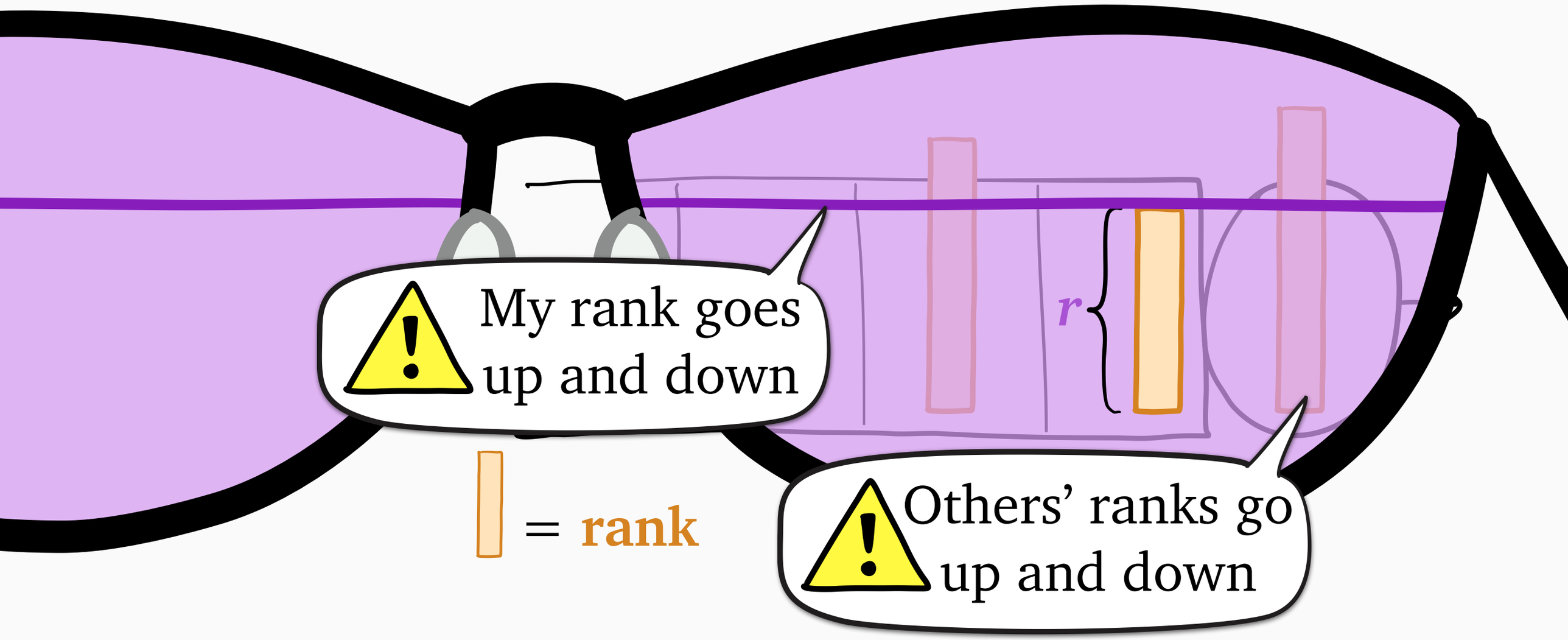
Nonmonotonic **rank** is hard



Key quantity: observed *r-work* $W(r)$

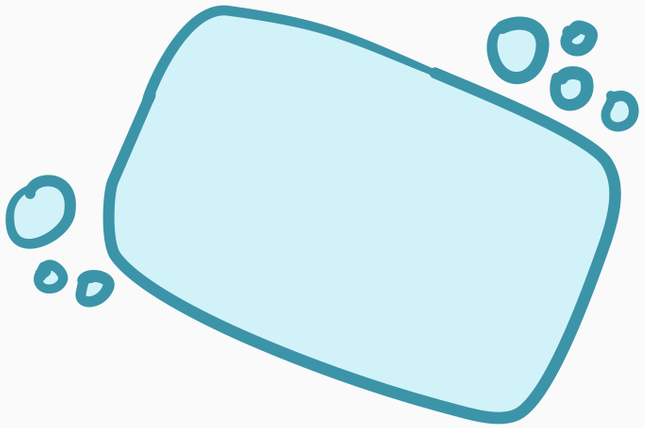
$W(r)$ = work relevant to job of **rank** r

Nonmonotonic **rank** is hard



Key quantity: observed *r-work* $W(r)$

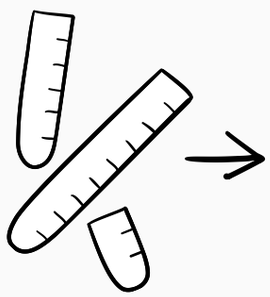
$W(r)$ = work relevant to job of **rank** r



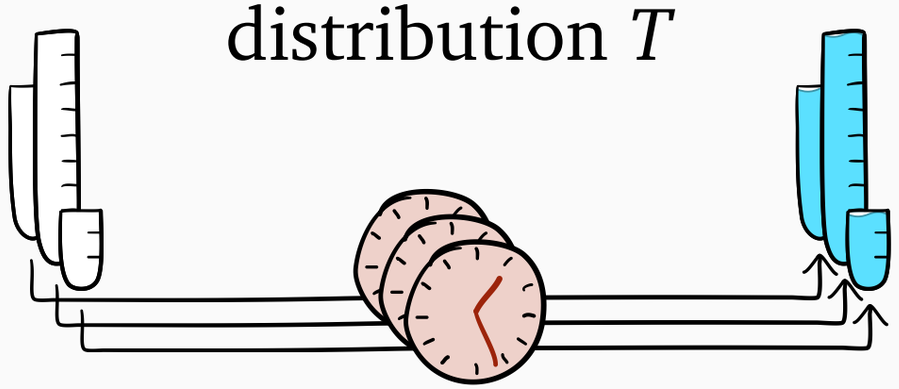
SOAP

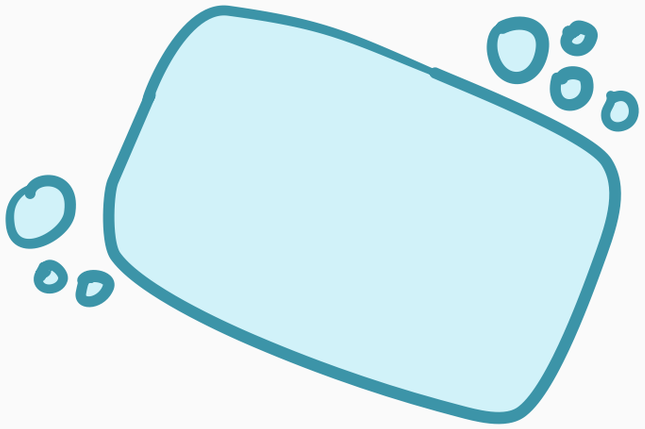
Schedule **O**rdered by **A**ge-based **P**riority

stochastic arrival process λ, S



response time distribution T

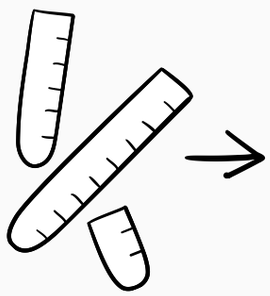




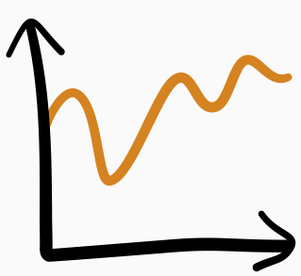
SOAP

Schedule **O**rdered by **A**ge-based **P**riority

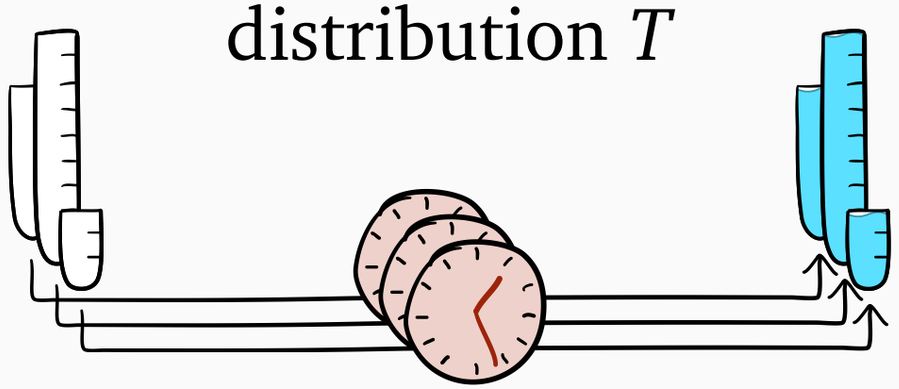
stochastic arrival process λ, S

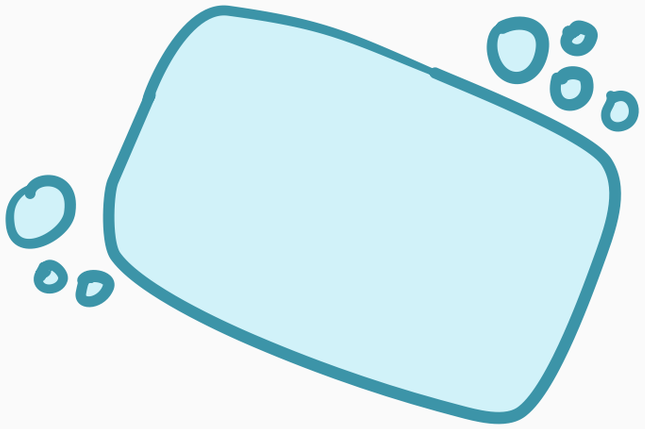


rank function



response time distribution T

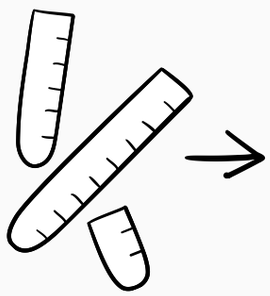




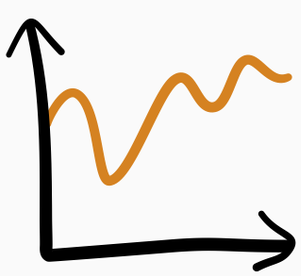
SOAP

Schedule **O**rdered by **A**ge-based **P**riority

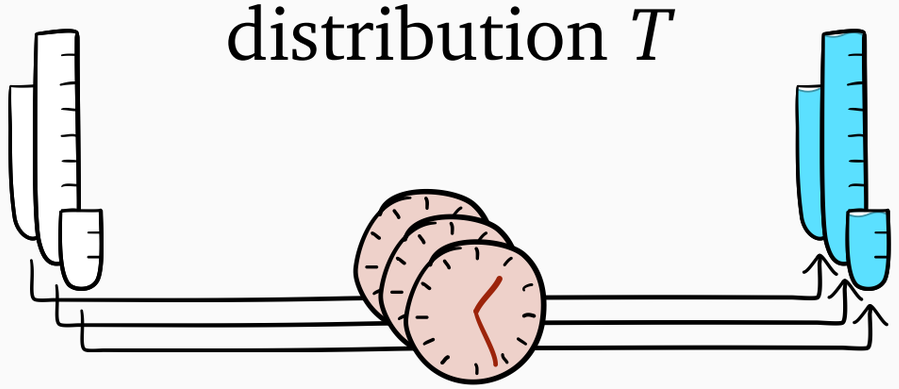
stochastic arrival process λ, S

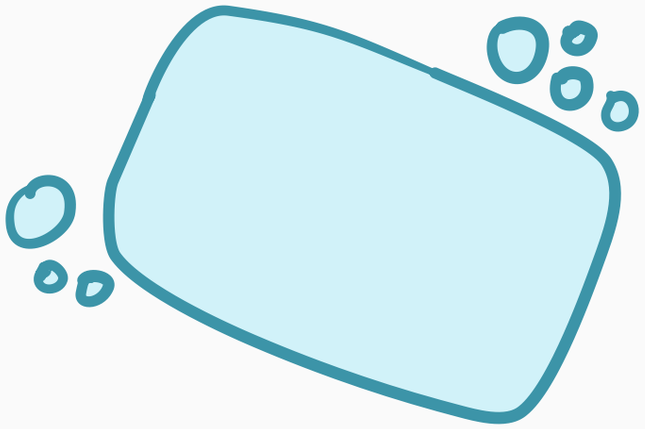


rank function



response time distribution T

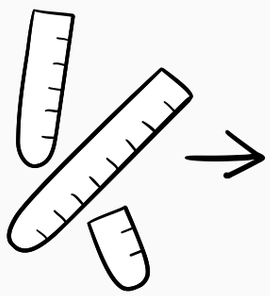




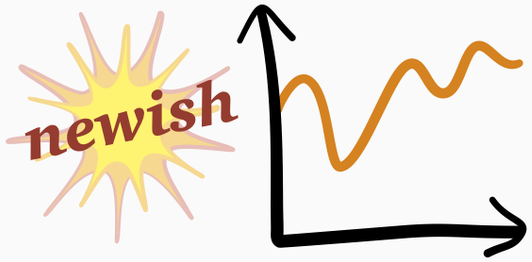
SOAP

Schedule **O**rdered by **A**ge-based **P**riority

stochastic arrival process λ, S

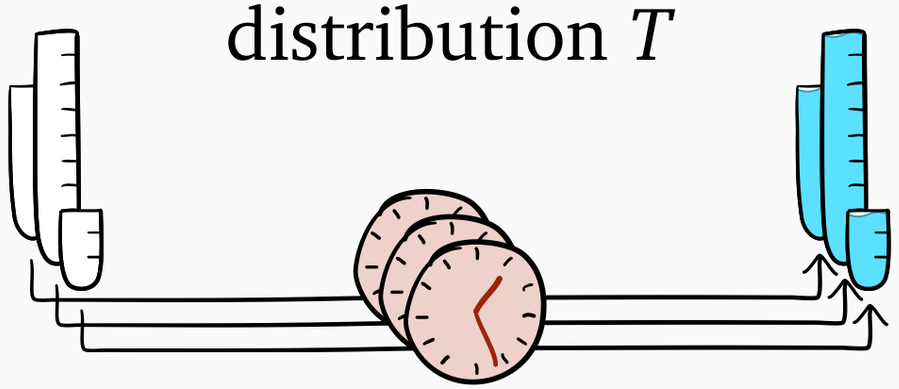


rank function

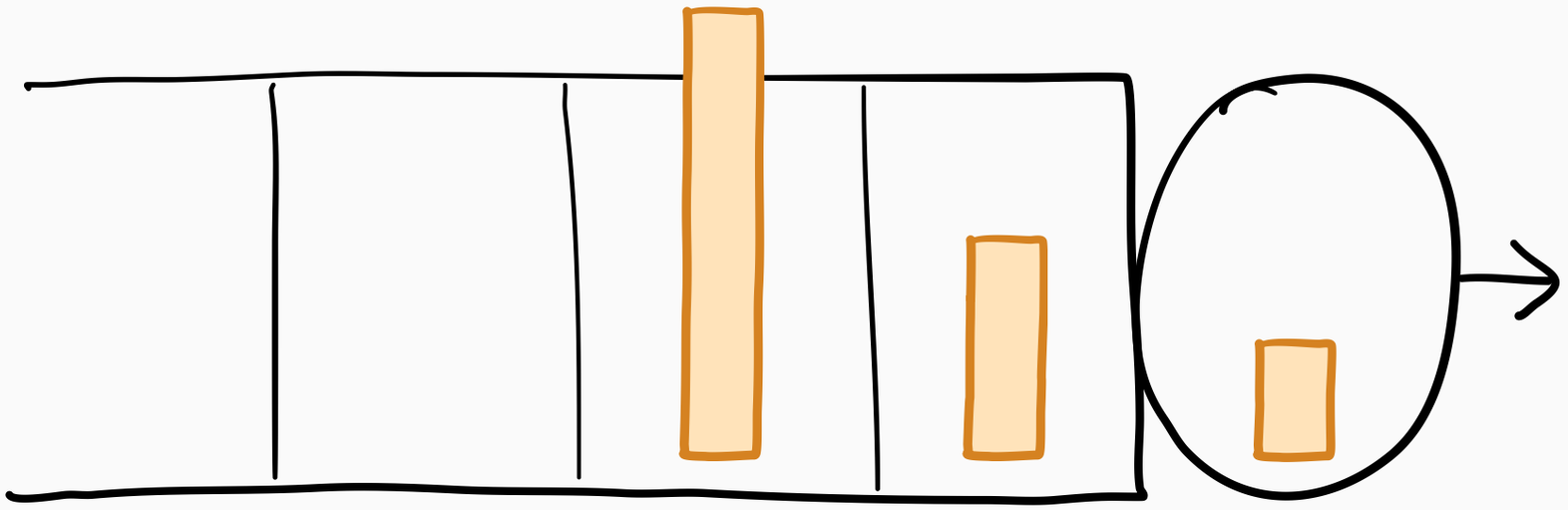


NEW!

response time distribution T

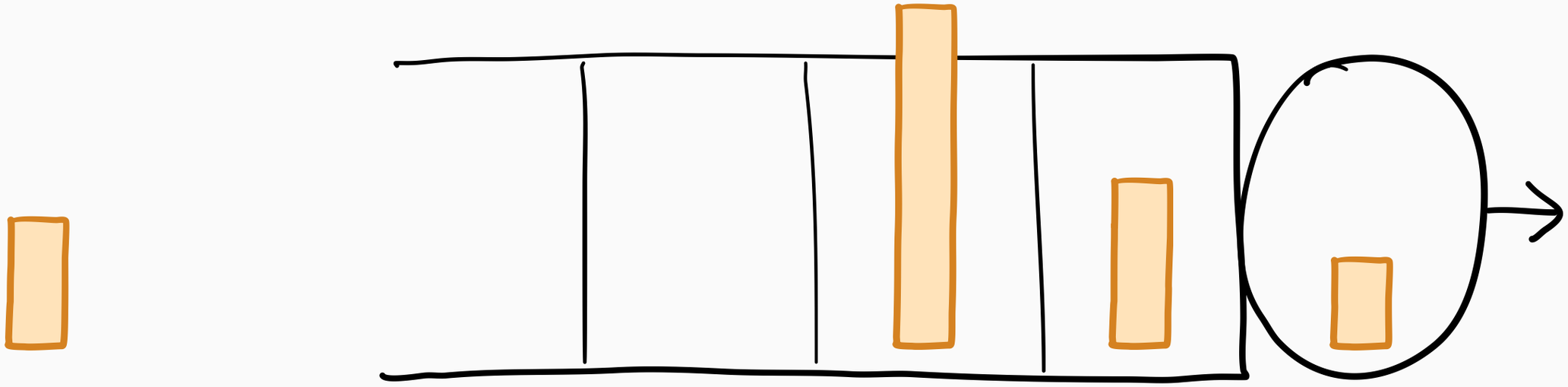


Key **SOAP** insight



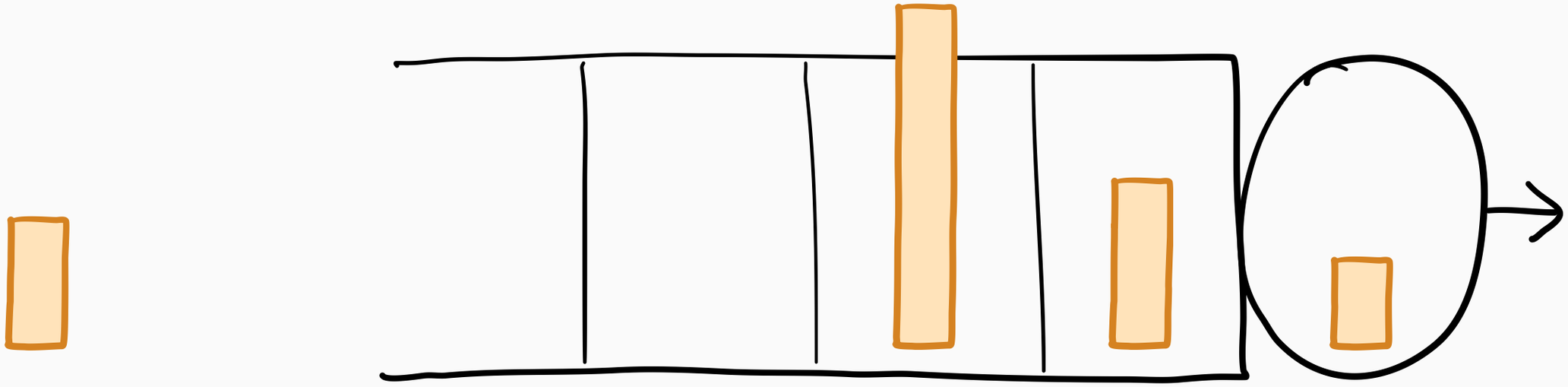
= **rank**

Key **SOAP** insight



“me” = **rank**

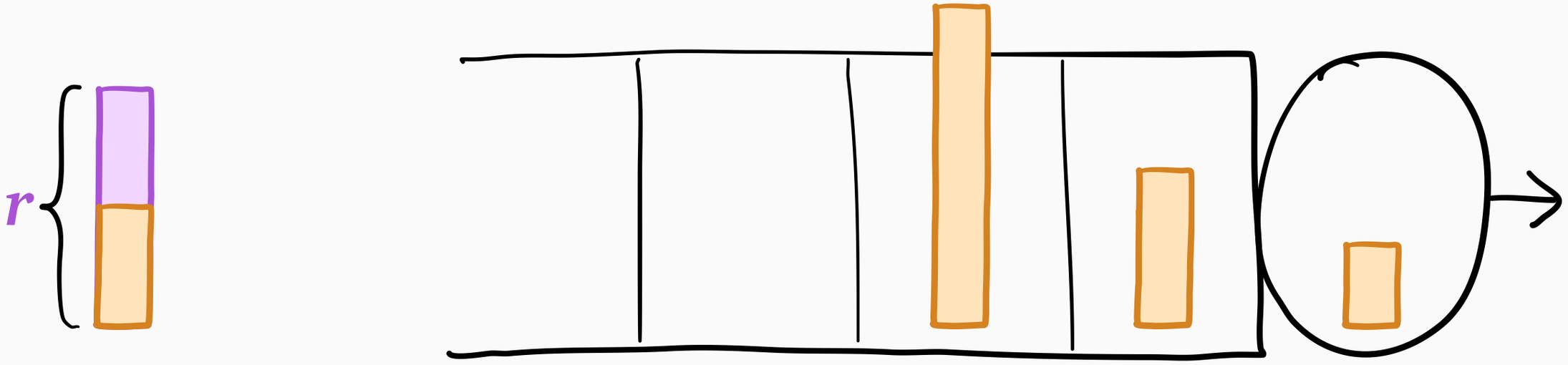
Key **SOAP** insight



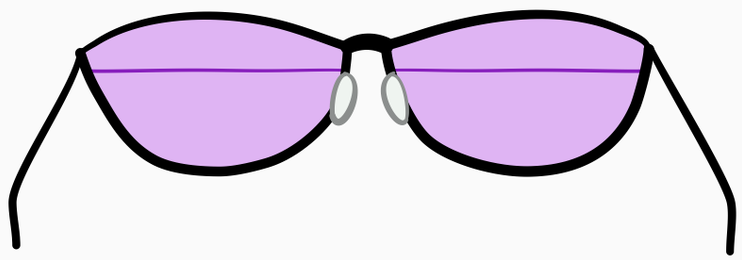
“me” = **rank**

Pessimism Priniciple: compute $W(r)$
with r = my *worst future* ***rank***

Key **SOAP** insight

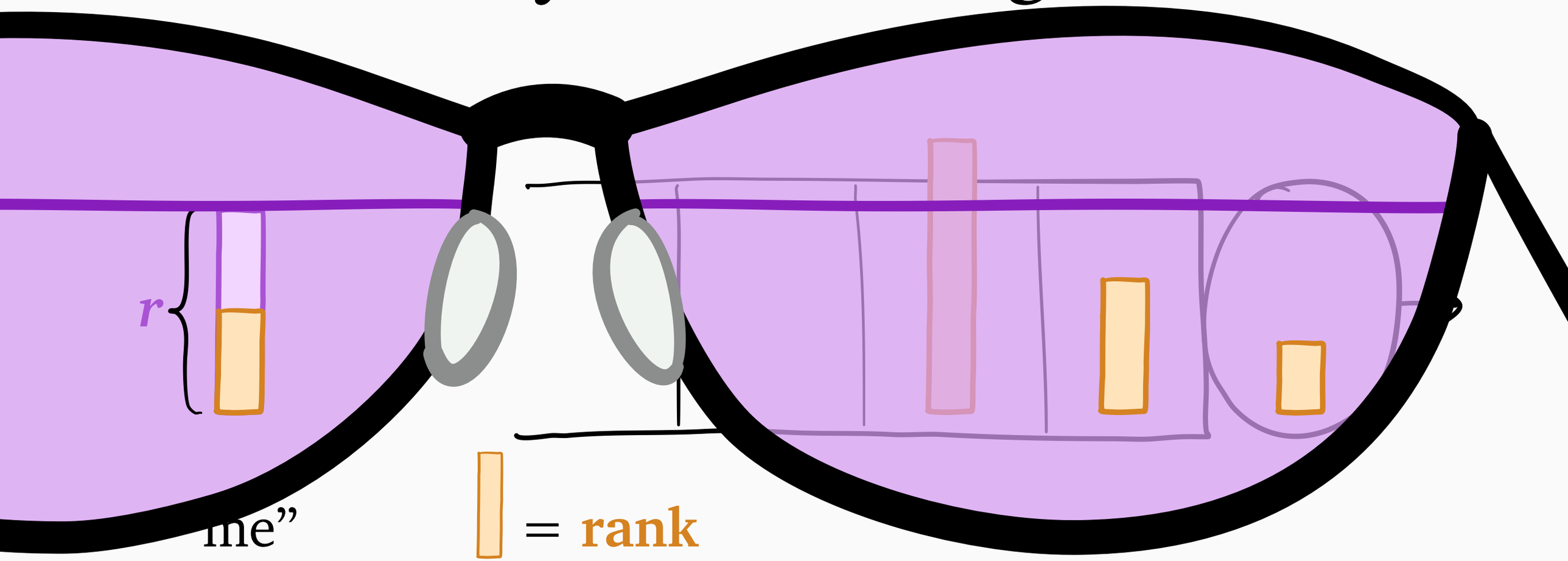


“me” = **rank**



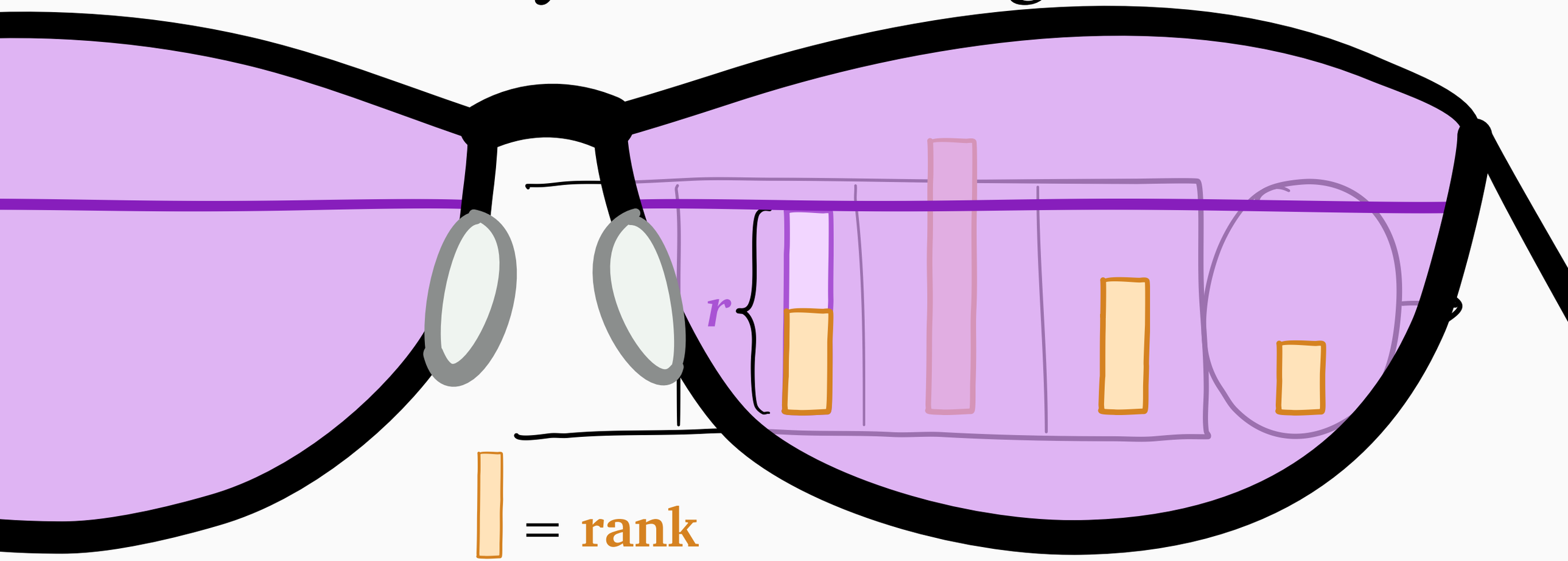
Pessimism Priniciple: compute $W(r)$
with r = my *worst future* ***rank***

Key **SOAP** insight



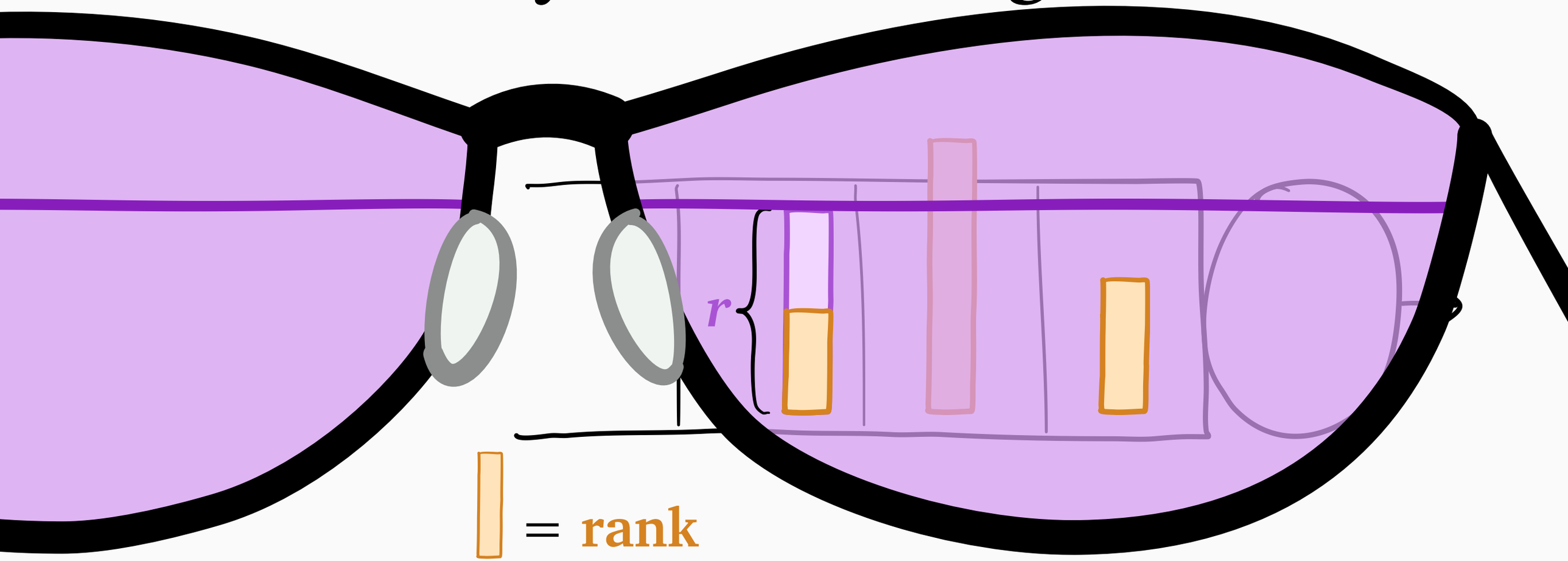
Pessimism Priniciple: compute $W(r)$
with r = my *worst future* ***rank***

Key **SOAP** insight



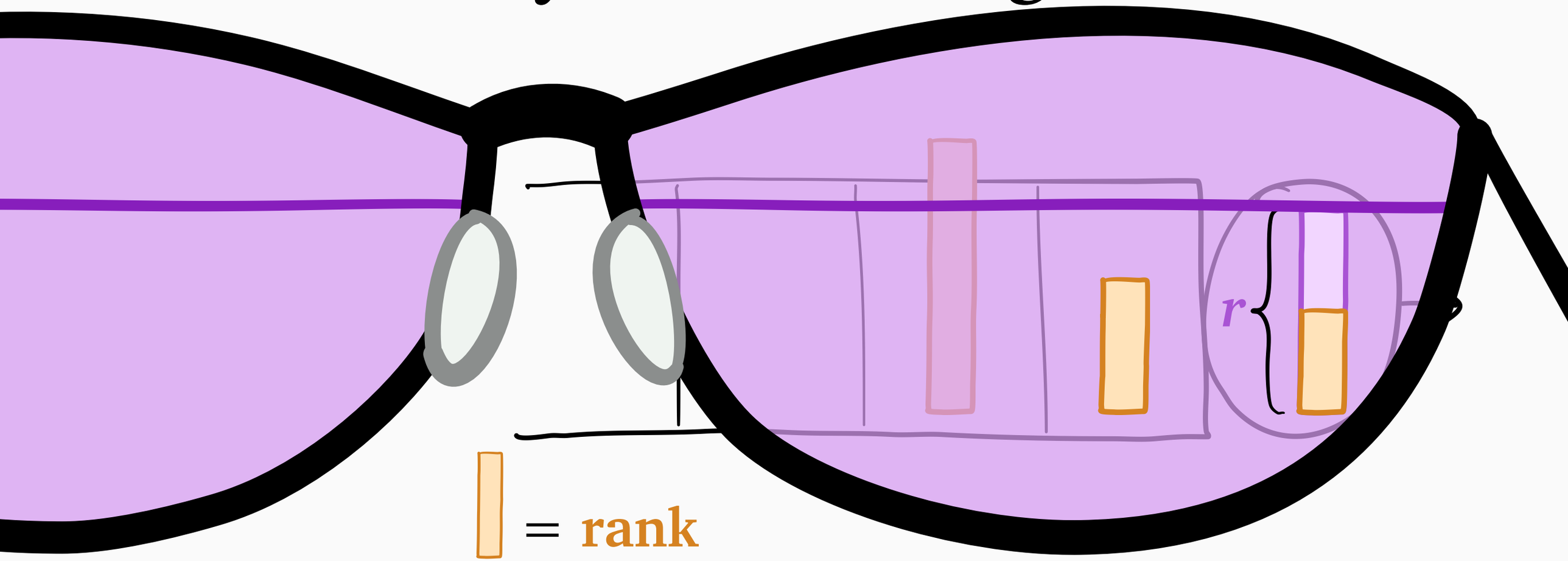
Pessimism Priniciple: compute $W(r)$
with r = my *worst future* ***rank***

Key **SOAP** insight



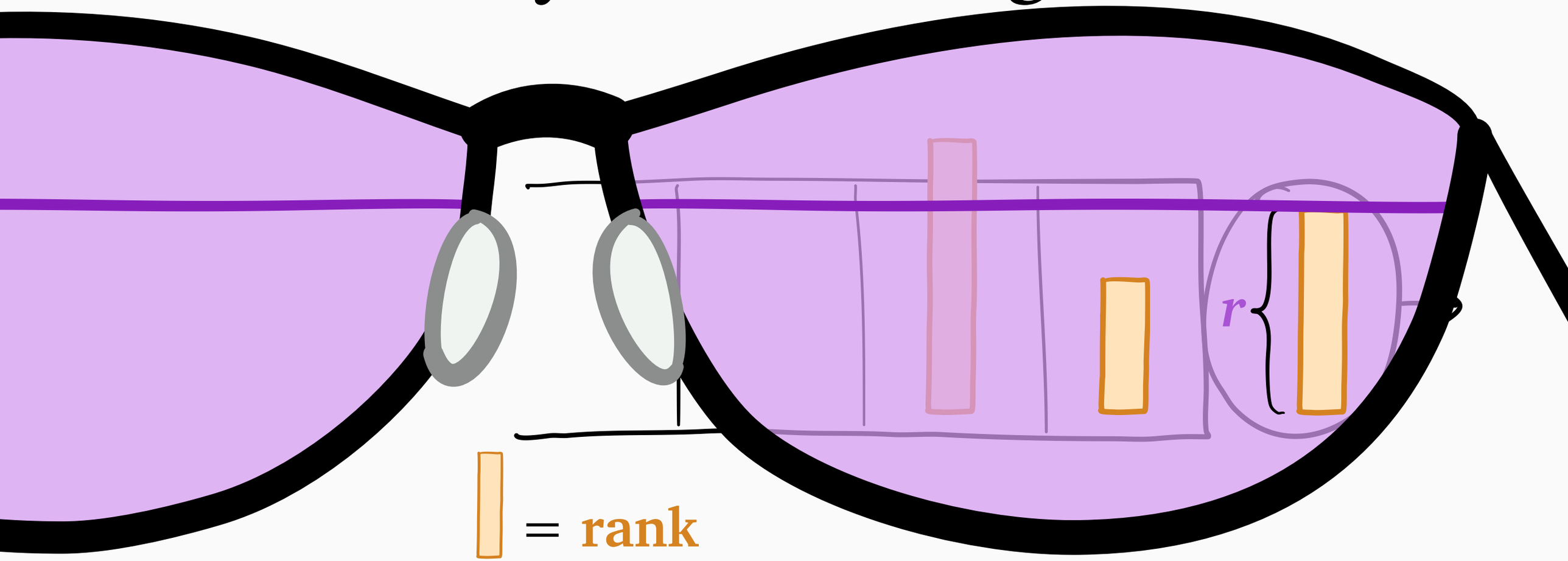
Pessimism Priniciple: compute $W(r)$
with r = my *worst future* ***rank***

Key **SOAP** insight



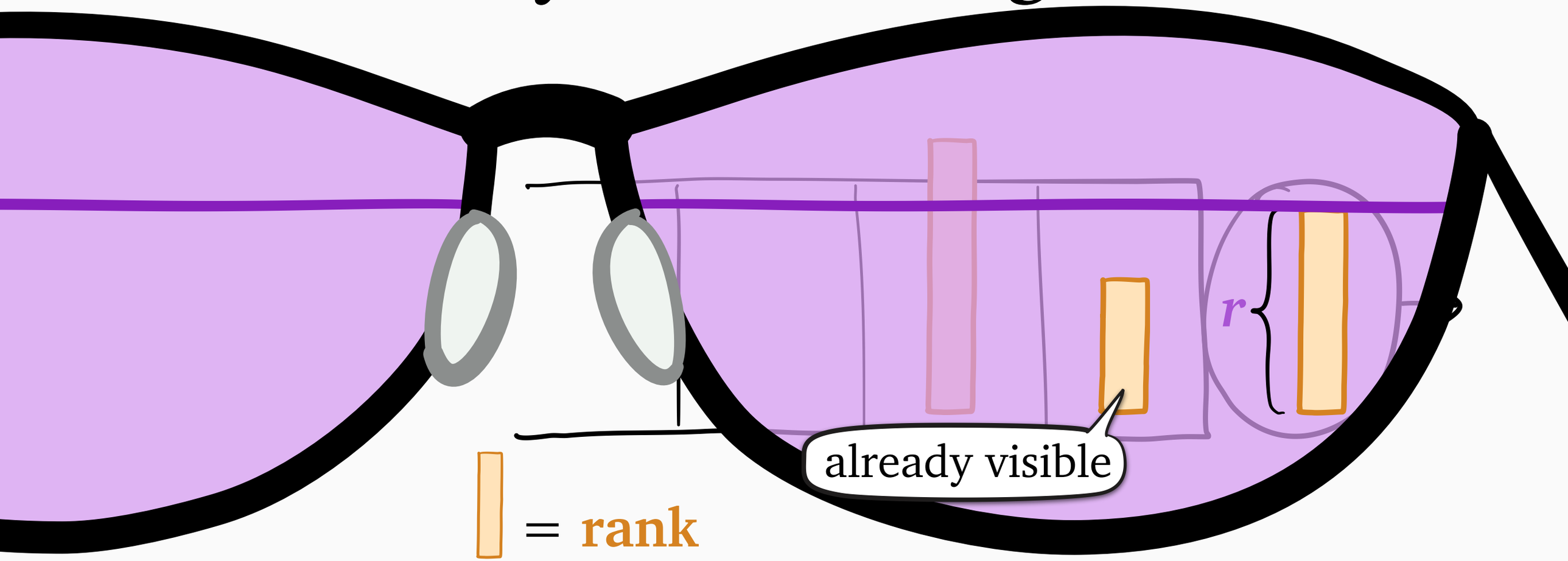
Pessimism Priniciple: compute $W(r)$
with r = my *worst future* ***rank***

Key **SOAP** insight



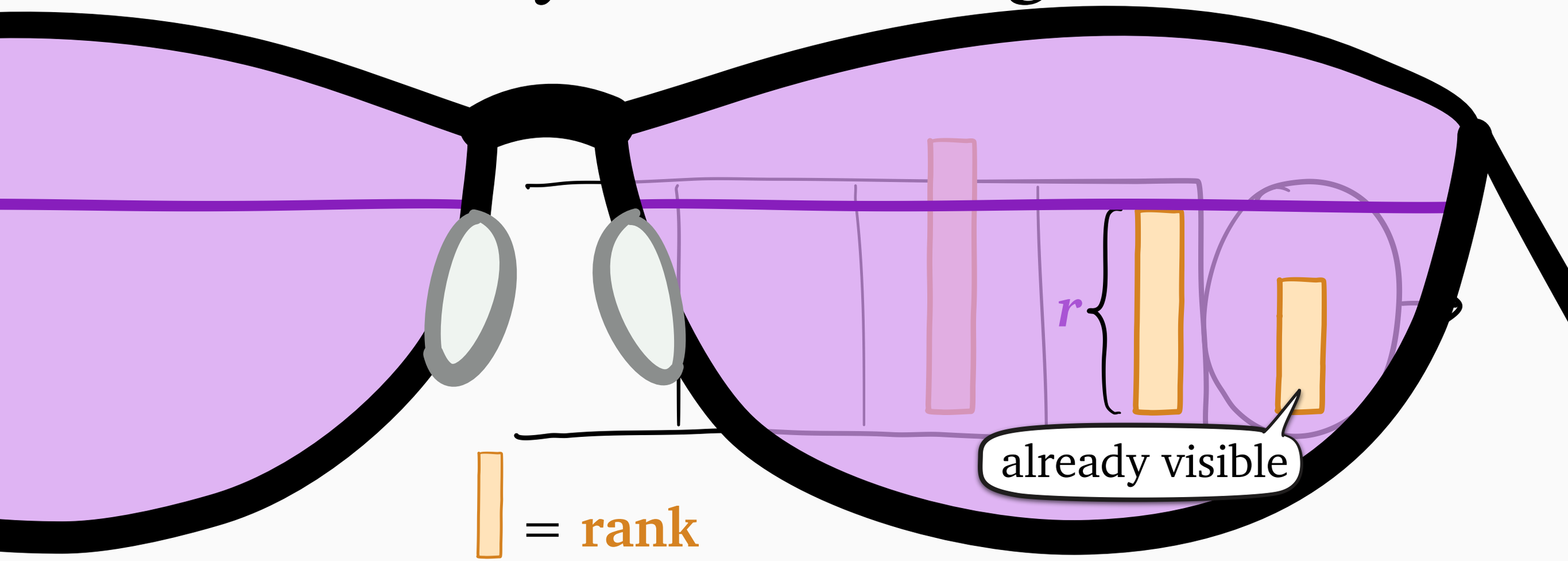
Pessimism Priniciple: compute $W(r)$
with r = my *worst future* ***rank***

Key **SOAP** insight



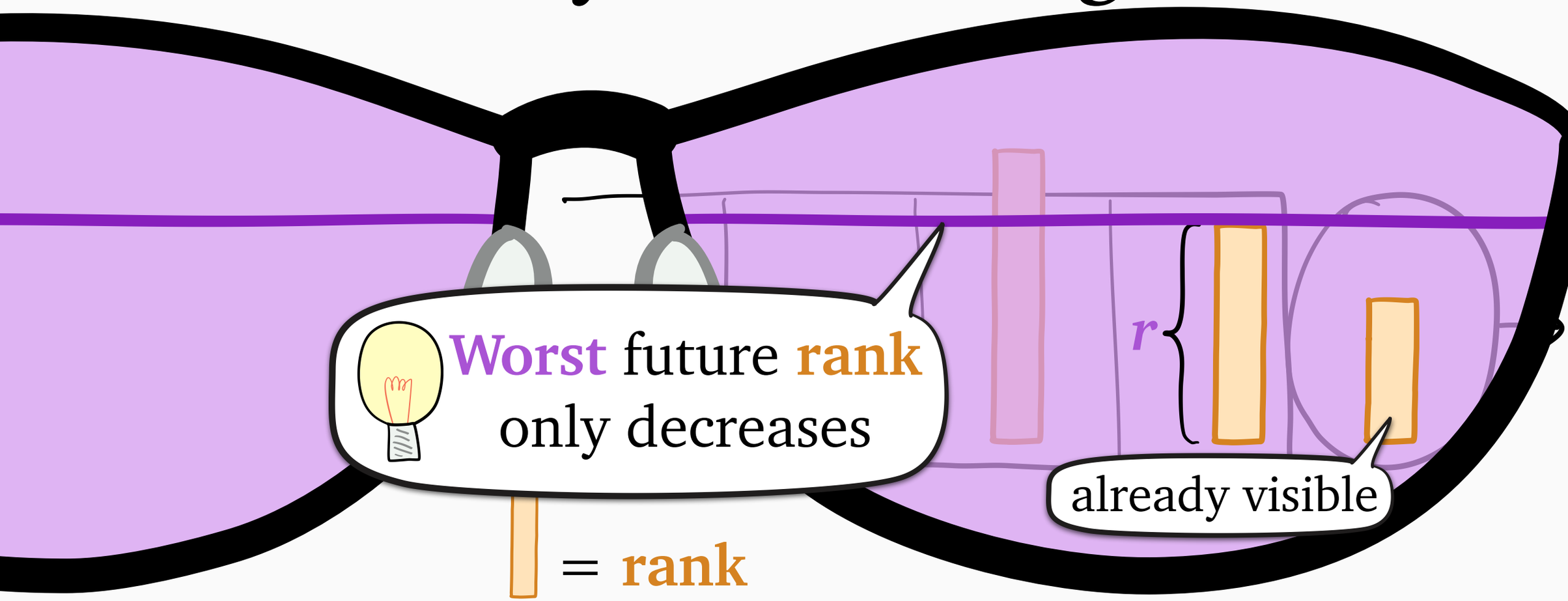
Pessimism Priniciple: compute $W(r)$
with r = my ***worst*** *future* ***rank***

Key **SOAP** insight



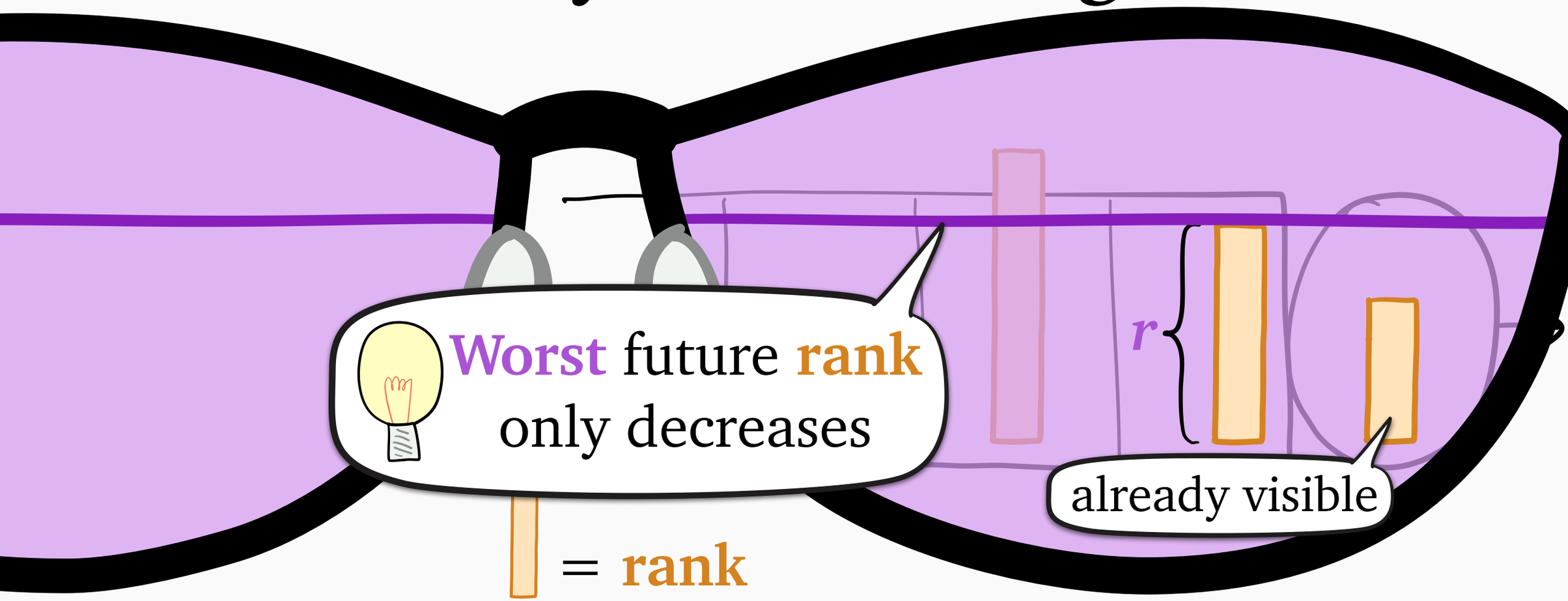
Pessimism Priniciple: compute $W(r)$
with r = my *worst future rank*

Key **SOAP** insight



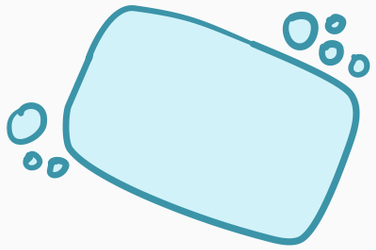
Pessimism Priniciple: compute $W(r)$
with r = my *worst future rank*

Key **SOAP** insight

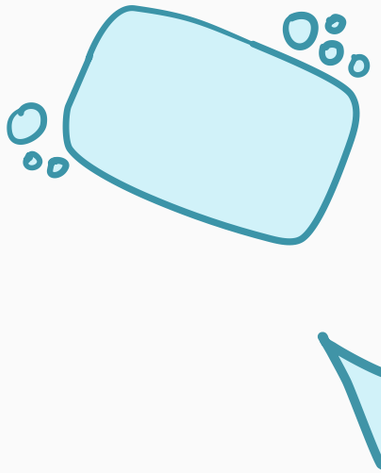


exact!

Pessimism Priniciple: compute $W(r)$ with r = my *worst future rank*



Impact of **SOAP**



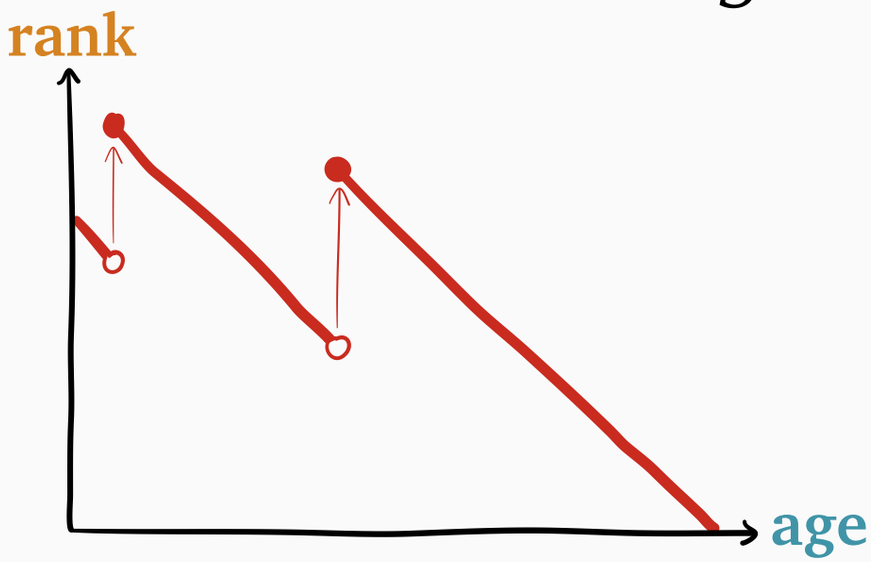
Impact of **SOAP**

unknown sizes

First analyses via **SOAP**

SERPT

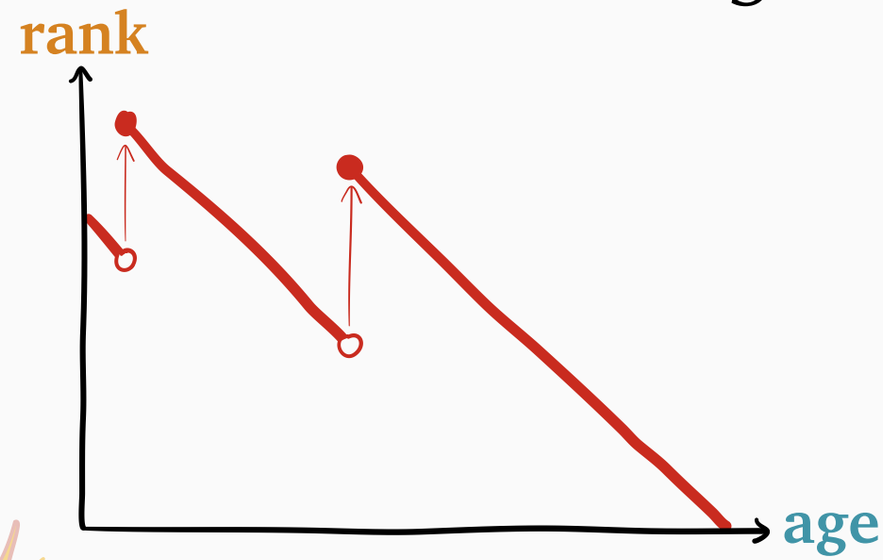
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$



First analyses via **SOAP**

SERPT

$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

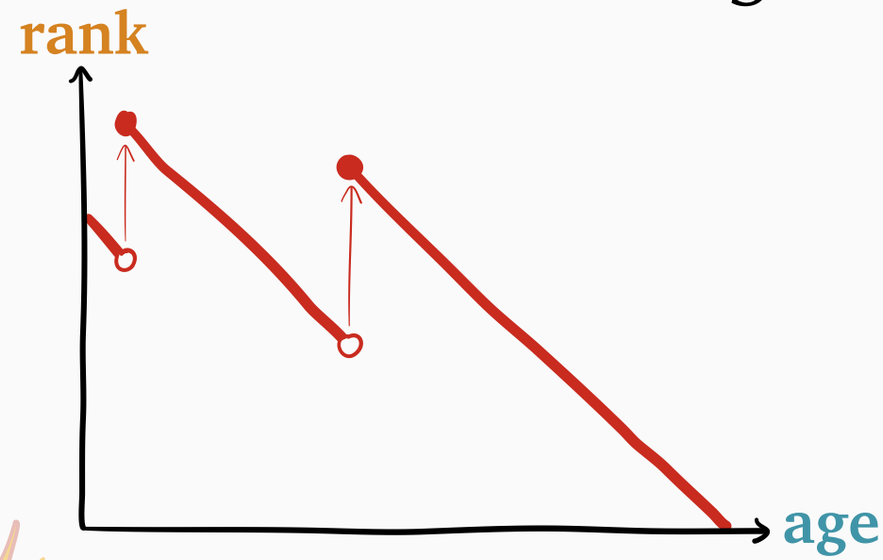


NEW! *first analysis* of **SERPT**

First analyses via **SOAP**

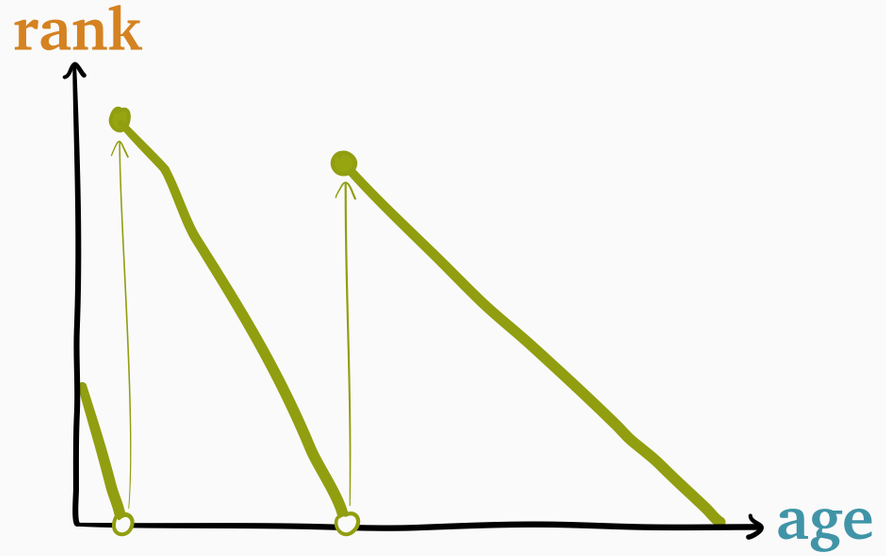
SERPT

$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$



Gittins

minimizes $\mathbf{E}[T]$

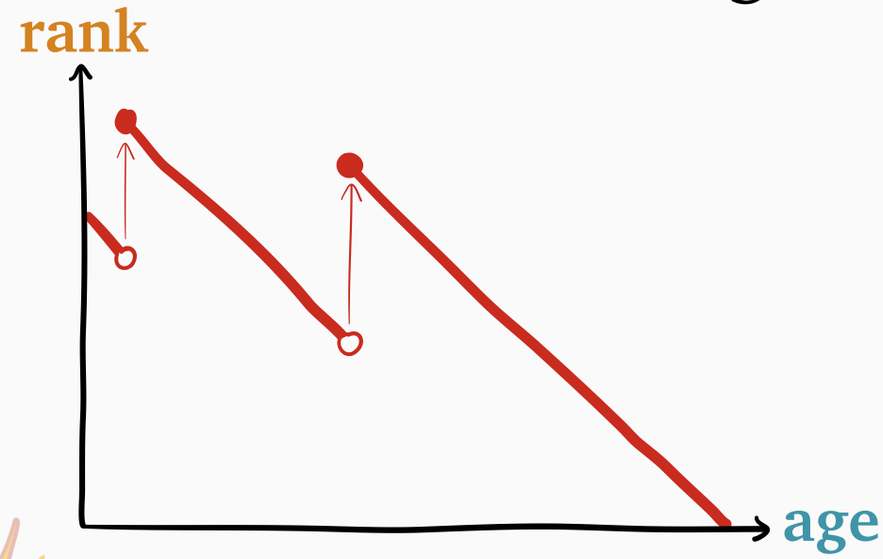


NEW! *first analysis* of **SERPT**

First analyses via **SOAP**

SERPT

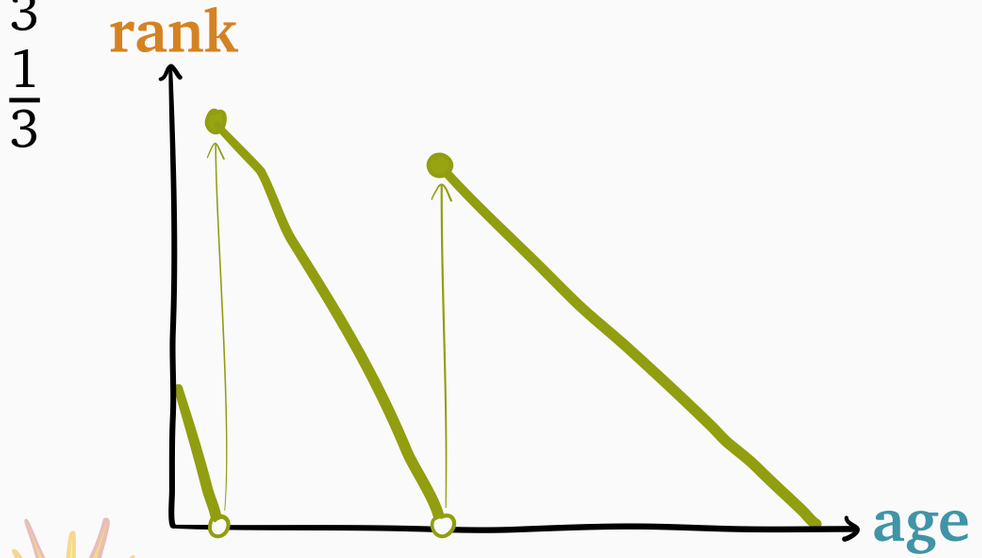
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$



NEW! *first analysis* of **SERPT**

Gittins

minimizes $\mathbf{E}[T]$

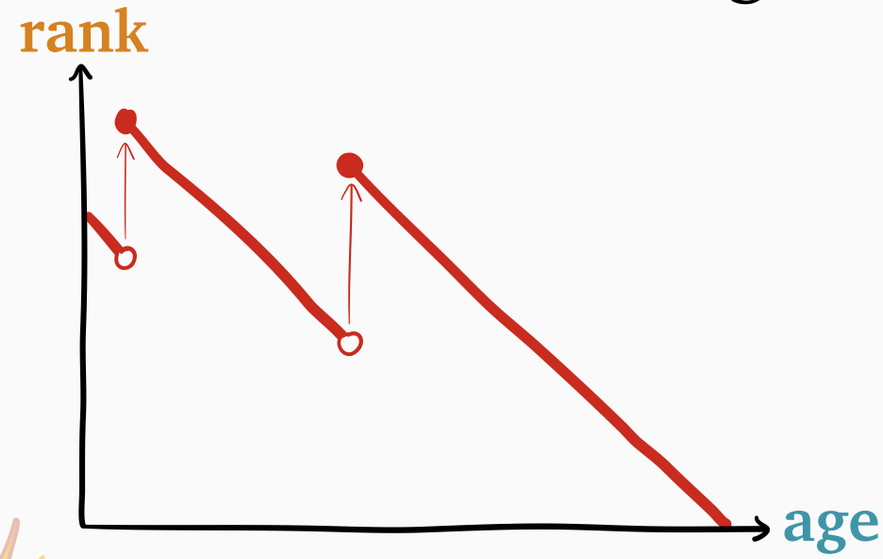


NEW! *first analysis* of **Gittins**

First analyses via **SOAP**

SERPT

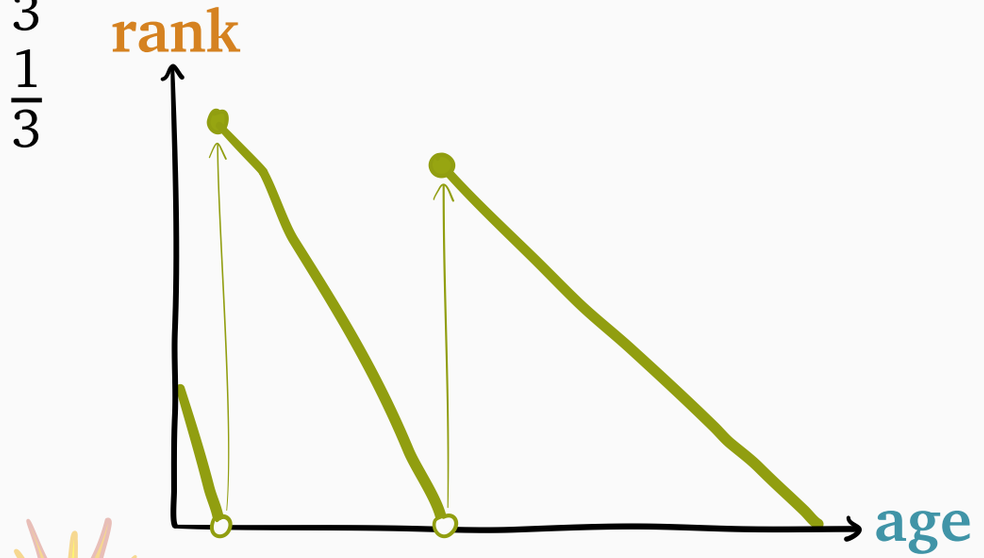
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$



NEW! *first analysis* of **SERPT**

Gittins

minimizes $\mathbf{E}[T]$



NEW! *first analysis* of **Gittins**

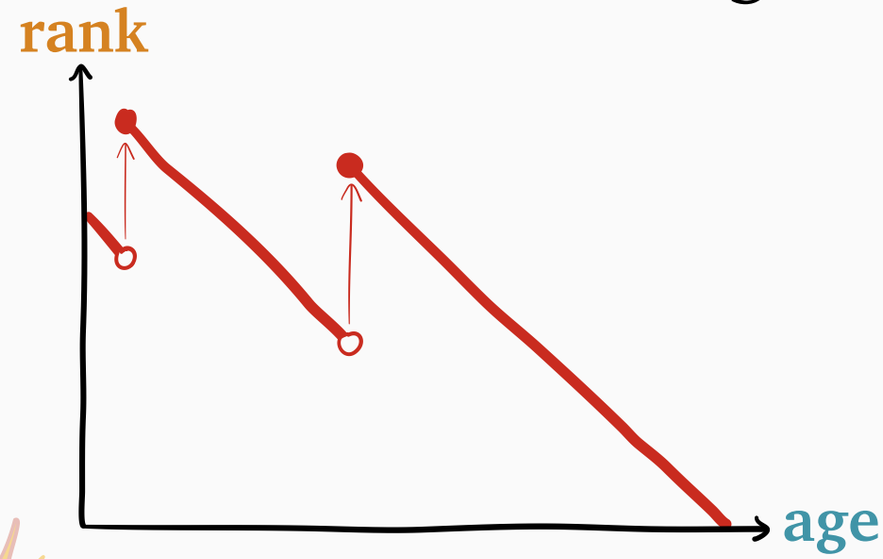
NEW! **Sample result:** for heavy-tailed S, **SERPT** and **Gittins** have *optimal tail decay* in large-t limit:

$$\mathbf{P}[T > t] \sim \mathbf{P}[S > (1-\rho)t]$$

First analyses via **SOAP**

SERPT

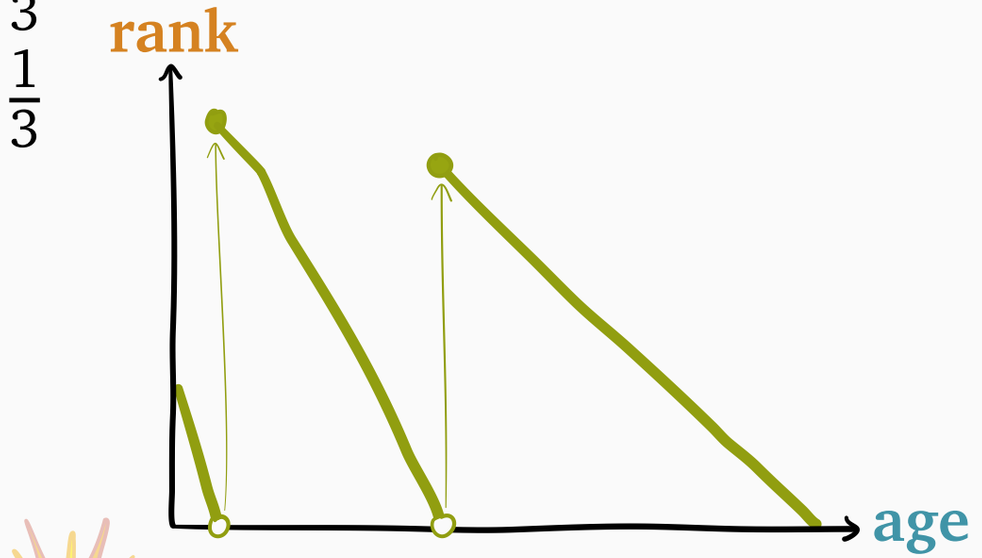
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$



NEW! *first analysis* of **SERPT**

Gittins

minimizes $\mathbf{E}[T]$

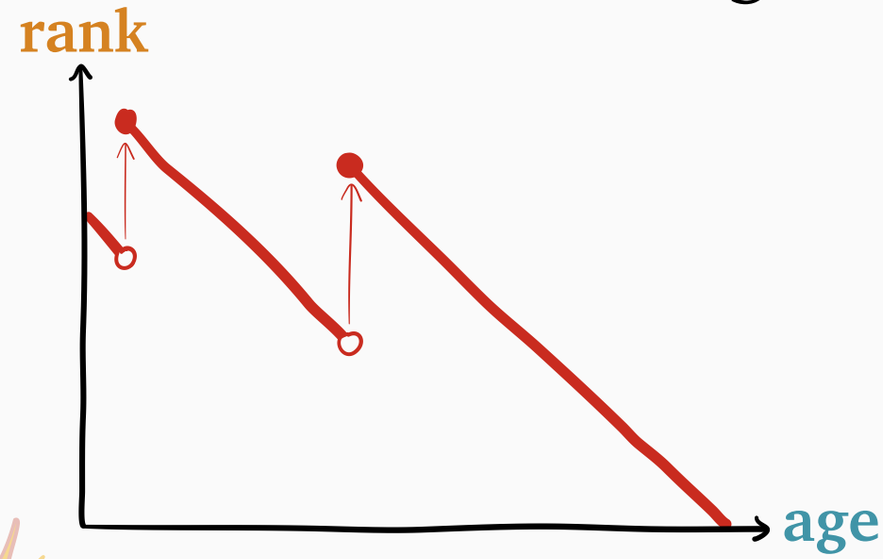


NEW! *first analysis* of **Gittins**

First analyses via **SOAP**

SERPT

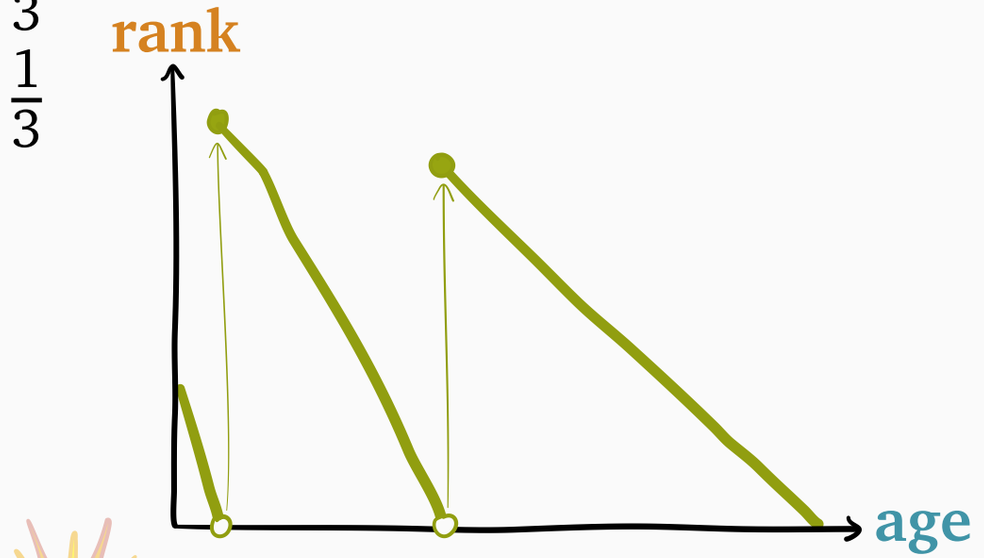
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$



NEW! *first analysis* of **SERPT**

Gittins

minimizes $\mathbf{E}[T]$



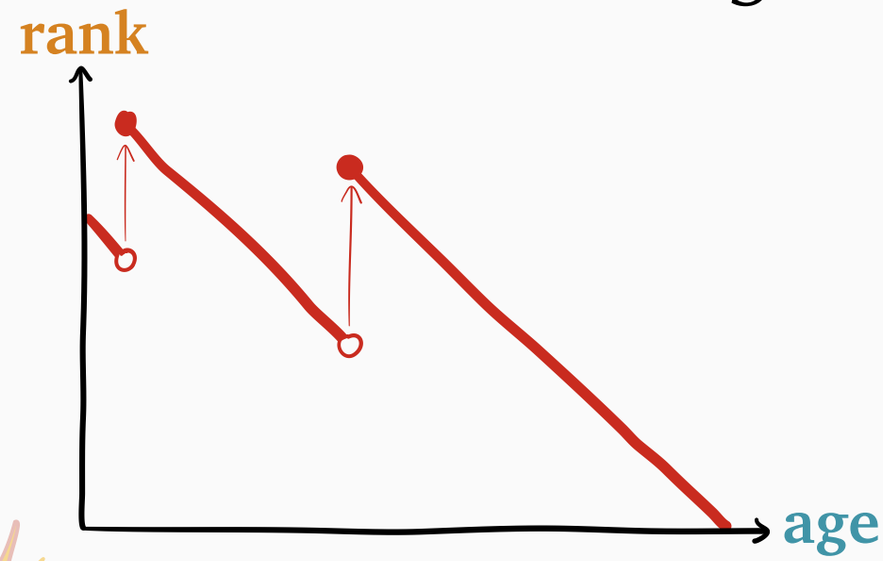
NEW! *first analysis* of **Gittins**

Observation: **SERPT**'s $\mathbf{E}[T]$ "usually" within 10% of **Gittins**'s

First analyses via **SOAP**

SERPT

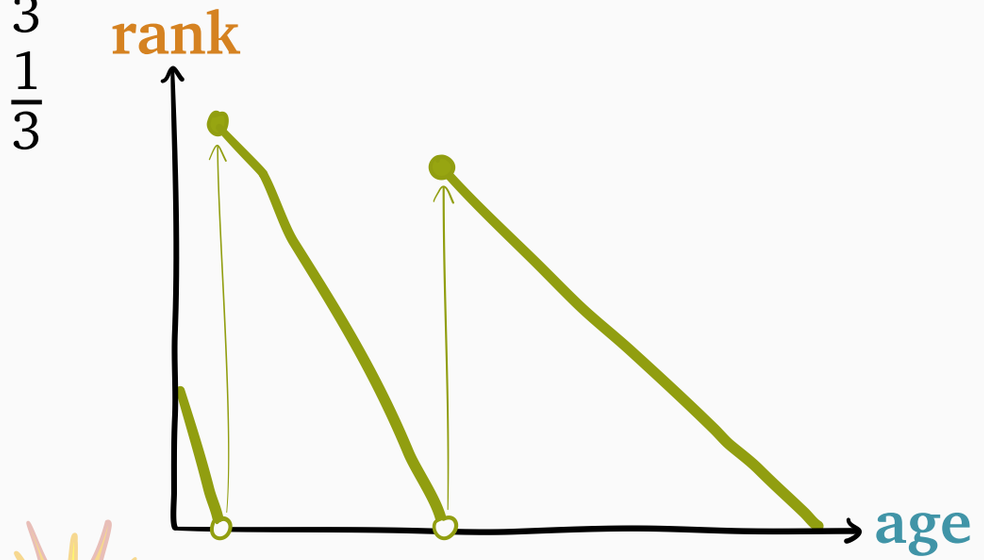
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$



NEW! *first analysis* of **SERPT**

Gittins

minimizes $\mathbf{E}[T]$



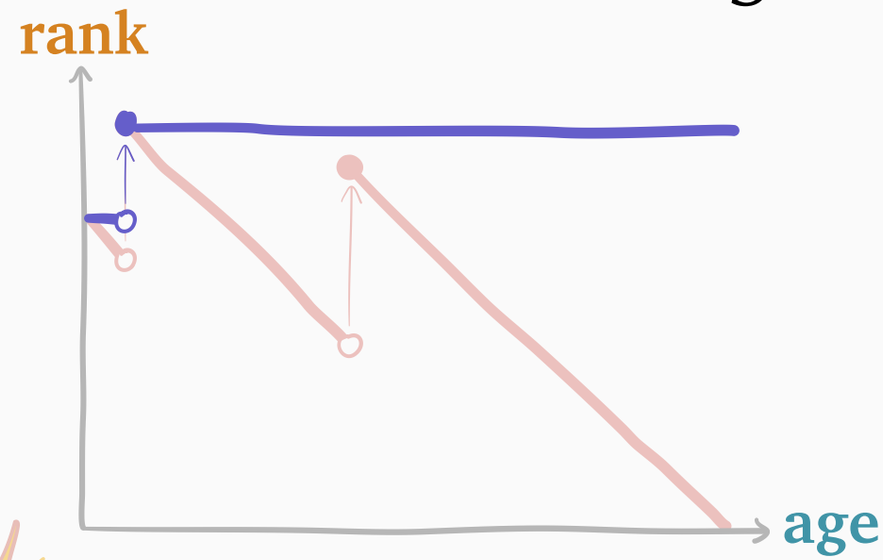
NEW! *first analysis* of **Gittins**

Observation: **SERPT**'s $\mathbf{E}[T]$ "usually" within 10% of **Gittins**'s

First analyses via SOAP

M-SERPT

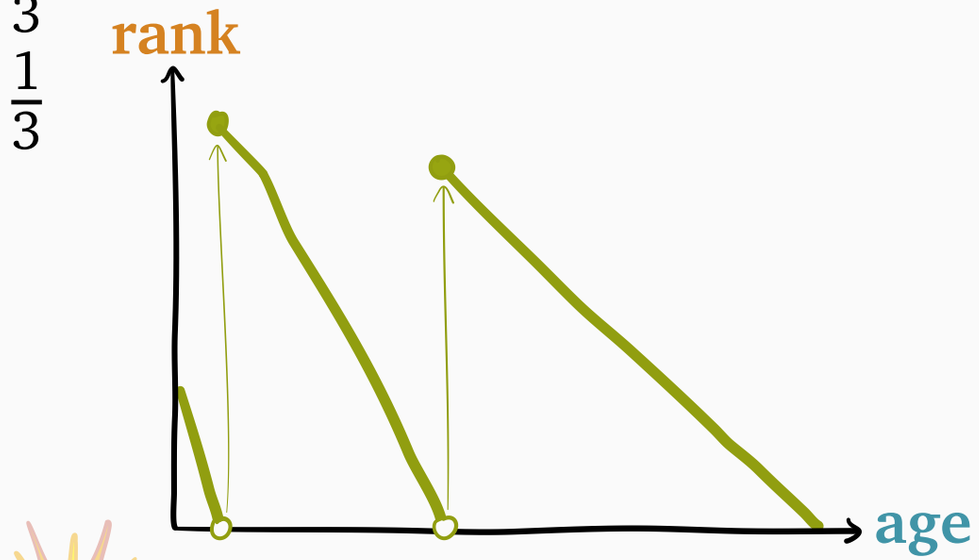
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$



NEW! *first analysis* of **SERPT**

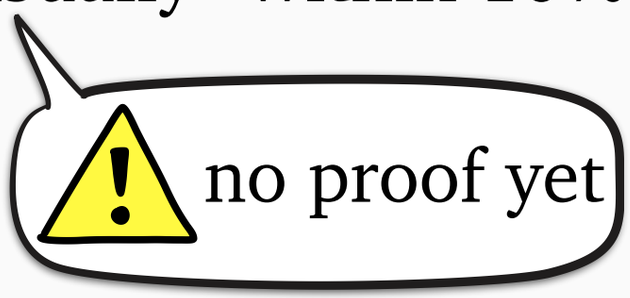
Gittins

minimizes $\mathbf{E}[T]$



NEW! *first analysis* of **Gittins**

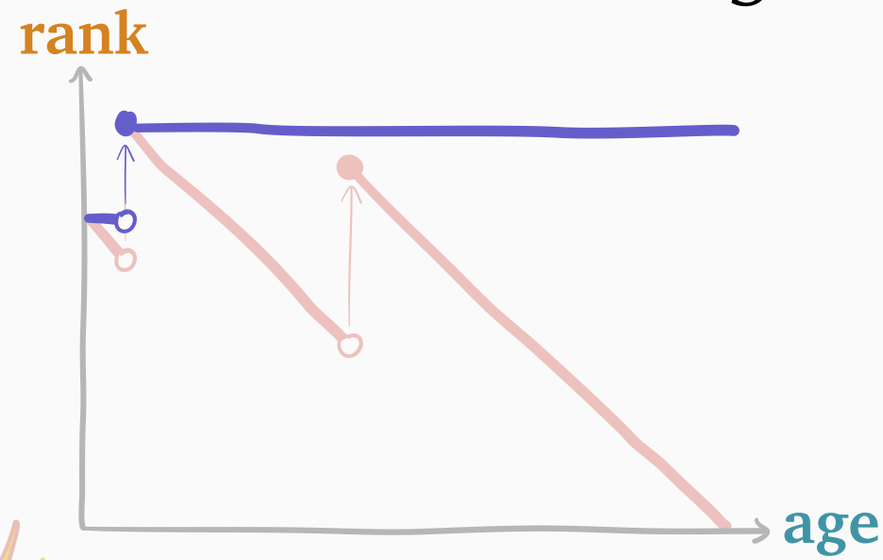
Observation: **SERPT**'s $\mathbf{E}[T]$ "usually" within 10% of **Gittins**'s



First analyses via SOAP

M-SERPT

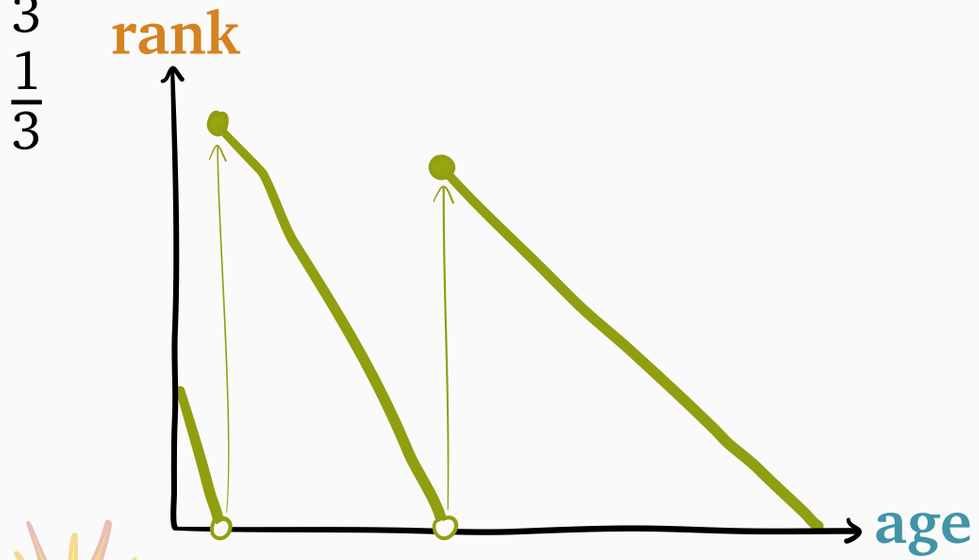
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$



NEW! *first analysis* of **SERPT**

Gittins

minimizes $\mathbf{E}[T]$



NEW! *first analysis* of **Gittins**

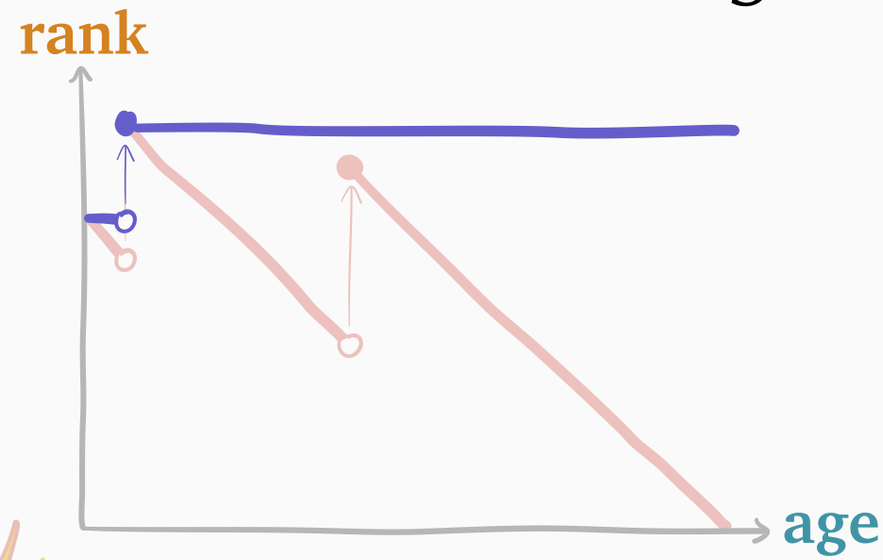
Observation: **SERPT**'s $\mathbf{E}[T]$ "usually" within 10% of **Gittins**'s

Theorem: **M-SERPT** is a 5-approximation for $\mathbf{E}[T]$

First analyses via SOAP

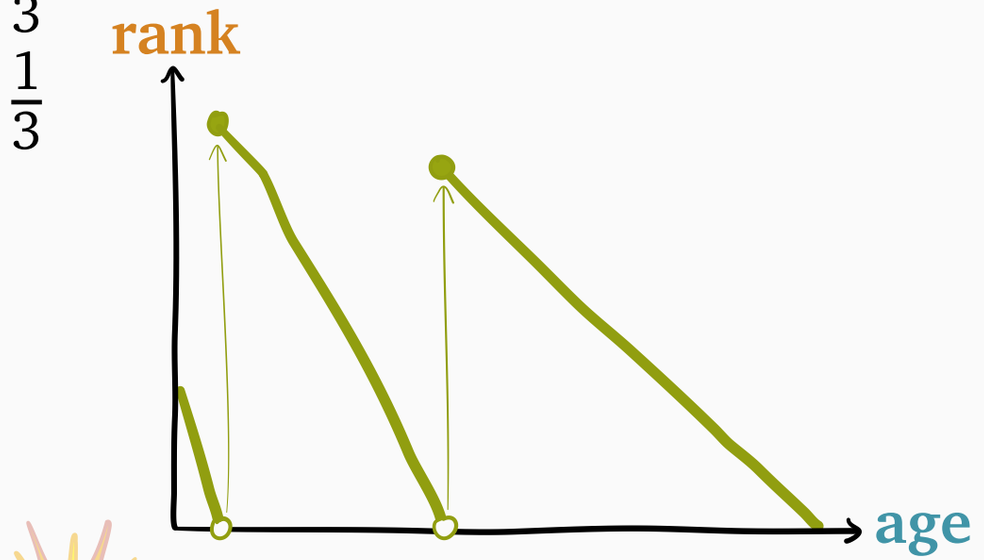
M-SERPT

$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$



Gittins

minimizes $\mathbf{E}[T]$



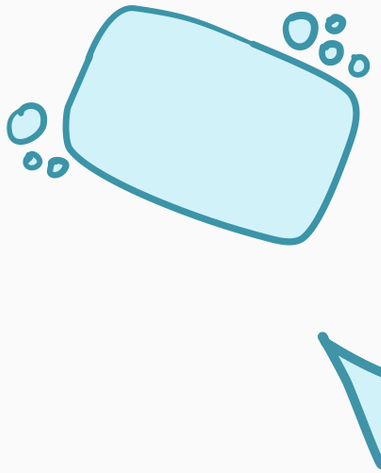
NEW! *first analysis* of **SERPT**

NEW! *first analysis* of **Gittins**

Lesson: carefully plan how **rank** gets worse

Obs [1] "usually" within 10% of **Gittins**'s

Theorem: **M-SERPT** is a 5-approximation for $\mathbf{E}[T]$



Impact of **SOAP**

unknown sizes

Impact of **SOAP**

unknown sizes

NEW! *First analysis* of

- mean
- tail decay

Impact of **SOAP**

unknown sizes

NEW!

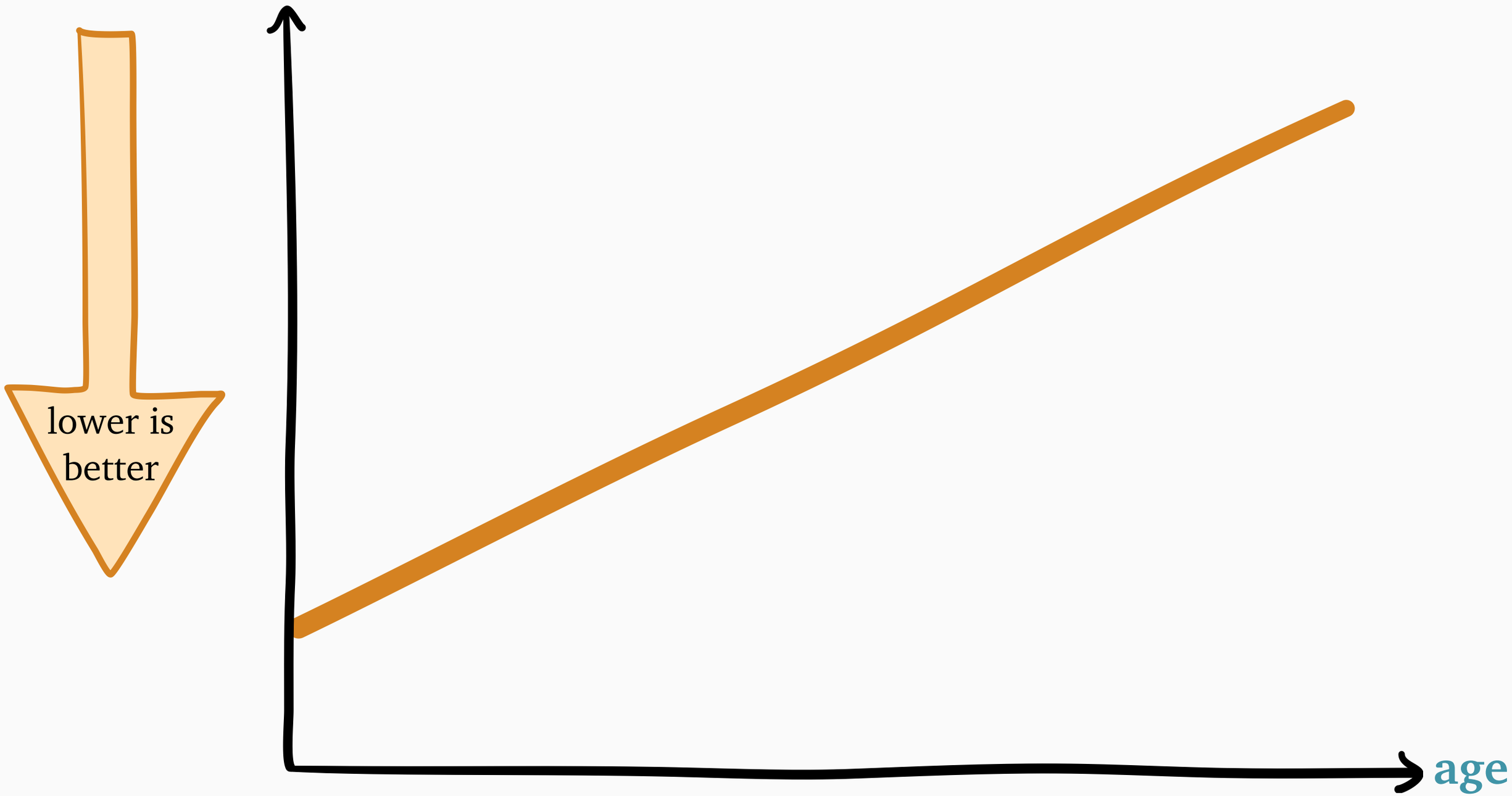
First analysis of

- mean
- tail decay

preemption limitations

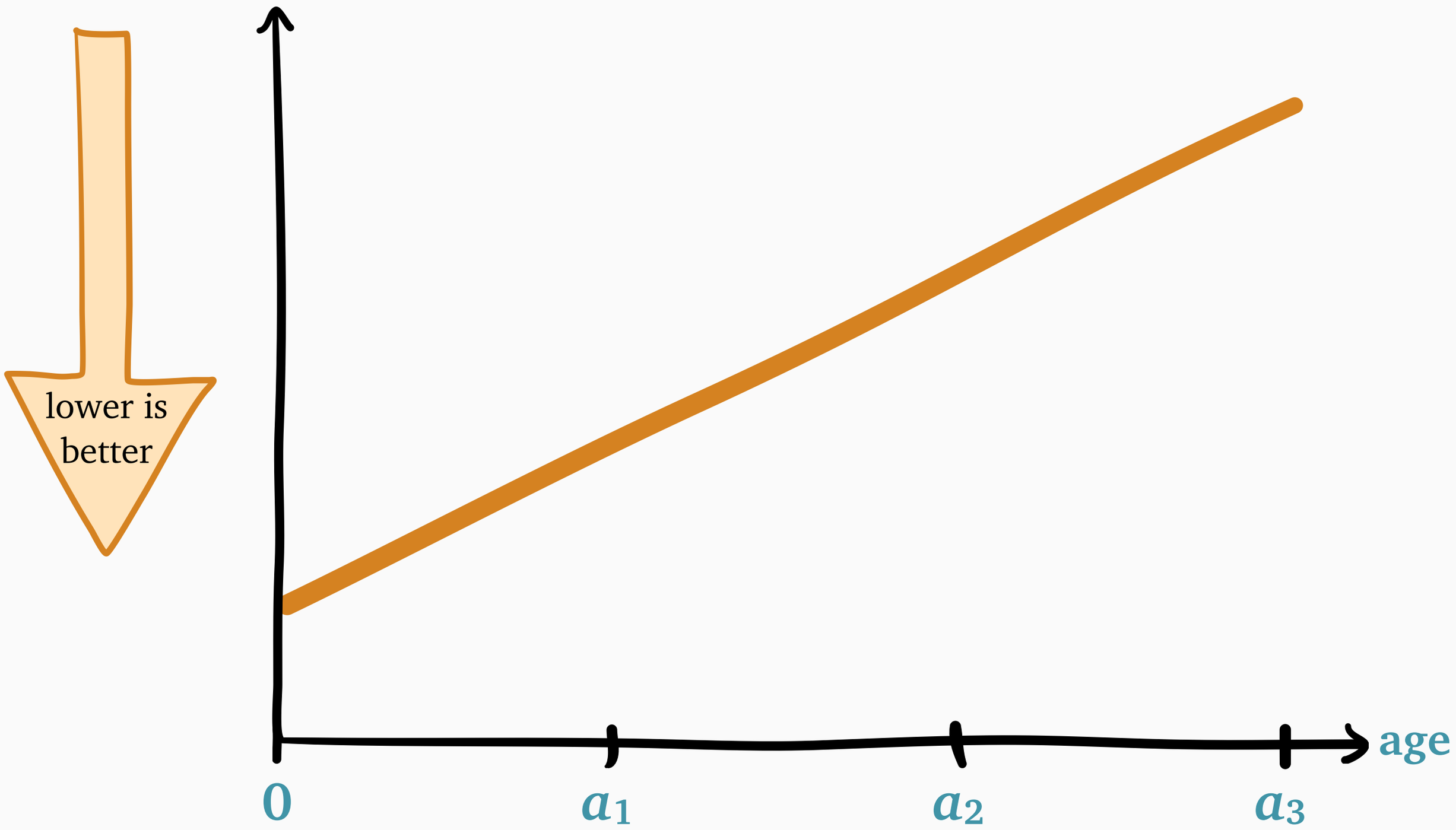
Preemption checkpoints

Can only preempt only at *checkpoint* ***ages*** a_i



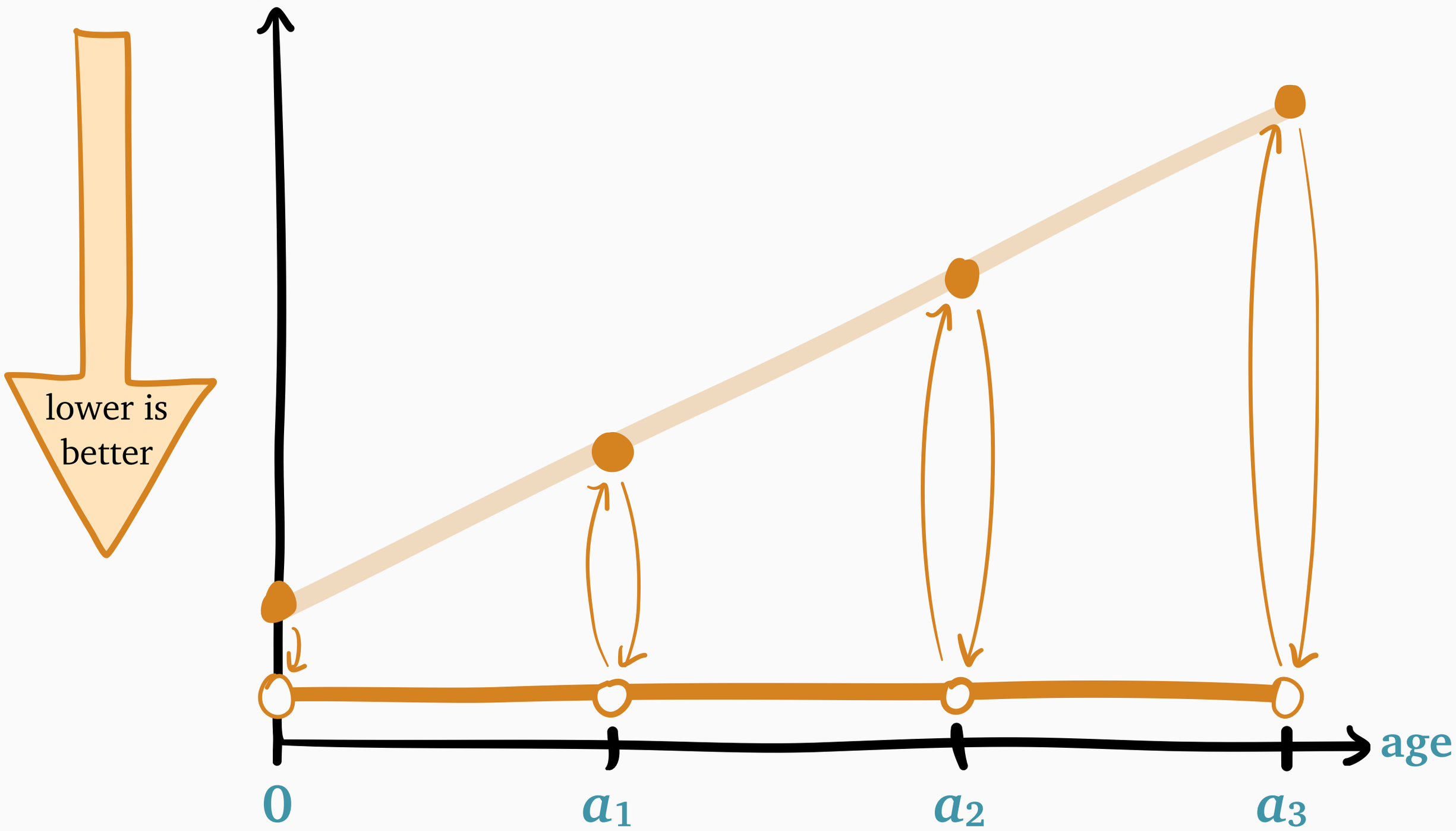
Preemption checkpoints

Can only preempt only at *checkpoint* ***ages*** a_i



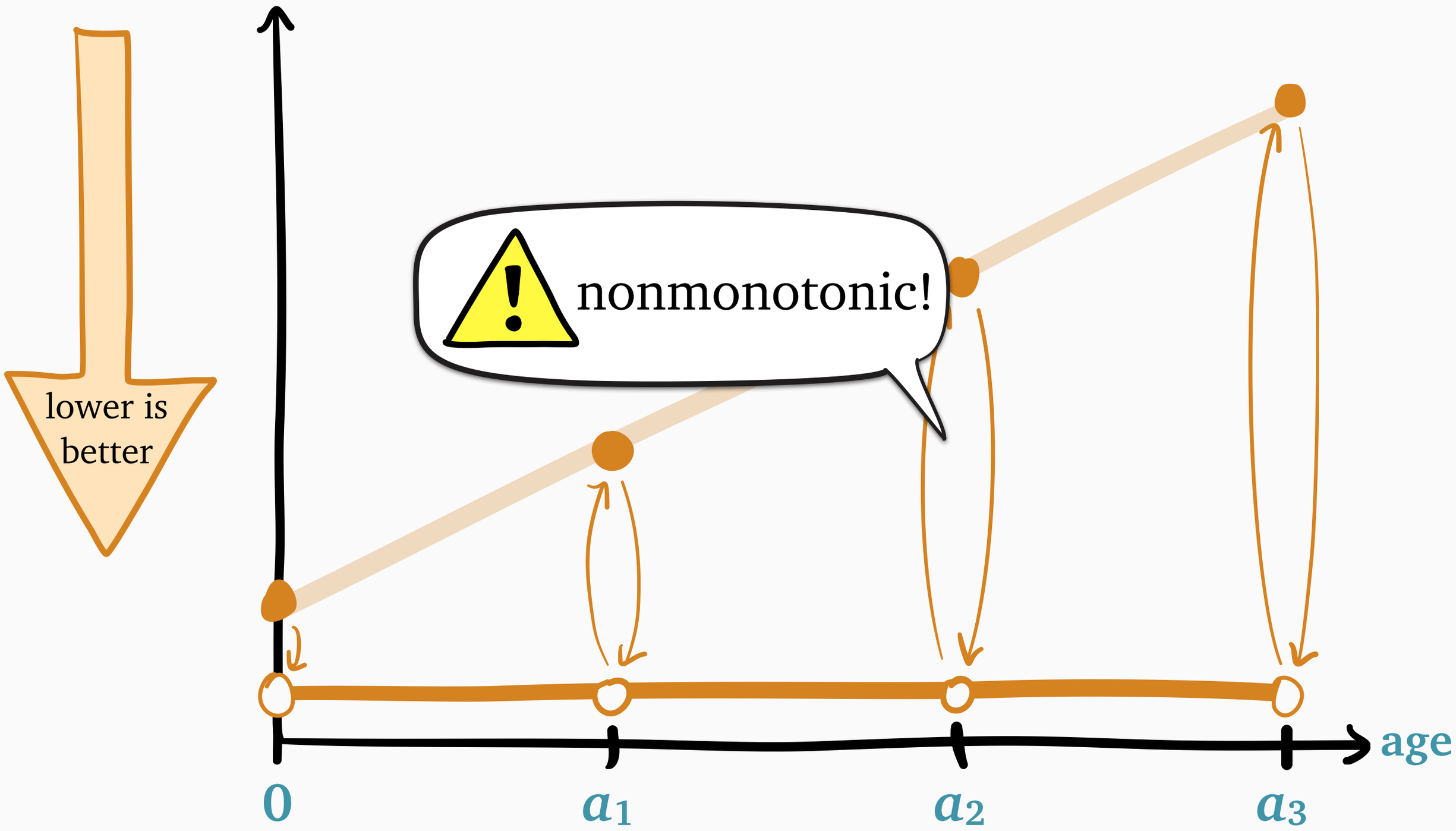
Preemption checkpoints

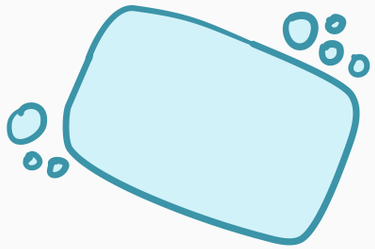
Can only preempt only at *checkpoint* ***ages*** a_i



Preemption checkpoints

Can only preempt only at *checkpoint* **ages a_i**





Impact of **SOAP**

unknown sizes

preemption limitations

First analysis of

- mean
- tail decay

Impact of **SOAP**

unknown sizes

First analysis of

- mean
- tail decay

preemption limitations

First analysis of

- mean
- tail decay

SOAP References

[1] Scully, Harchol-Balter, and Scheller-Wolf (2018). "SOAP: One Clean Analysis of All Age-Based Scheduling Policies." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2018). **Finalist: 2019 INFORMS APS Best Student Paper Prize.**

[2] Scully and Harchol-Balter (2018). "SOAP Bubbles: Robust Scheduling Under Adversarial Noise." *56th Annual Allerton Conference on Communication, Control, and Computing.*

[3] Scully, Harchol-Balter, and Scheller-Wolf (2020). "Simple Near-Optimal Scheduling for the M/G/1." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2018). **Winner: SIGMETRICS 2020 Best Video Award.**

[4] Scully, Keveld, Boxma, Dorsman, and Wierman (2020). "Characterizing policies with optimal response time tails under heavy-tailed job sizes." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2018).

[5] Scully and Kreveld (2021). "When Does the Gittins Policy Have Asymptotically Optimal Response Time Tail?" *SIGMETRICS Perform. Eval. Rev.* (MAMA 2021), *arXiv.*

[6] Scully and Harchol-Balter (2021). "How to Schedule Near-Optimally under Real-World Constraints." *arXiv.*

SOAP References

[1] Scully, Harchol-Balter, and Scheller-Wolf (2018). "SOAP: One Clean Analysis of All Age-Based Scheduling Policies." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2018). **Finalist: 2019 INFORMS APS Best Student Paper Prize.**

[2] Scully and Harchol-Balter (2018). "SOAP Bubbles: Robust Scheduling Under Adversarial Noise." *56th Annual Allerton Conference on Communication, Control, and Computing.*

[3] Scully, Harchol-Balter, and Scheller-Wolf (2020). "Simple Near-Optimal Scheduling for the M/G/1." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2018). **Winner: SIGMETRICS 2020 Best Video Award.**

[4] Scully, Keveld, Boxma, Dorsman, and Wierman (2020). "Characterizing policies with optimal response time tails under heavy-tailed job sizes." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2018).

[5] Scully and Kreveld (2021). "When Does the Gittins Policy Have Asymptotically Optimal Response Time Tail?" *SIGMETRICS Perform. Eval. Rev.* (MAMA 2021), *arXiv.*

[6] Scully and Harchol-Balter (2021). "How to Schedule Near-Optimally under Real-World Constraints." *arXiv.*

SOAP References

[1] Scully, Harchol-Balter, and Scheller-Wolf (2018). "SOAP: One Clean Analysis of All Age-Based Scheduling Policies." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2018). **Finalist: 2019 INFORMS APS Best Student Paper Prize.**

[2] Scully and Harchol-Balter (2018). "SOAP Bubbles: Robust Scheduling Under Adversarial Noise." *56th Annual Allerton Conference on Communication, Control, and Computing.*

[3] Scully, Harchol-Balter, and Scheller-Wolf (2020). "Simple Near-Optimal Scheduling for the M/G/1." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2018). **Winner: SIGMETRICS 2020 Best Video Award.**

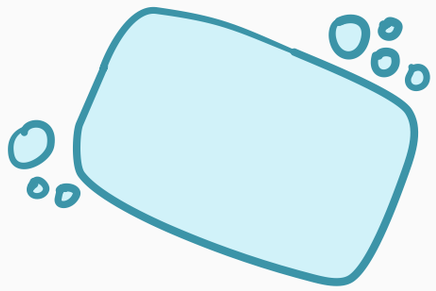
[4] Scully, Keveld, Boxma, Dorsman, and Wierman (2020). "Characterizing policies with optimal response time tails under heavy-tailed job sizes." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2018).

[5] Scully and Kreveld (2021). "When Does the Gittins Policy Have Asymptotically Optimal Response Time Tail?" *SIGMETRICS Perform. Eval. Rev.* (MAMA 2021), *arXiv.*

[6] Scully and Harchol-Balter (2021). "How to Schedule Near-Optimally under Real-World Constraints." *arXiv.*

case studies

Outline: two new tools



SOAP

new unifying theory of single-server scheduling

→ unknown sizes

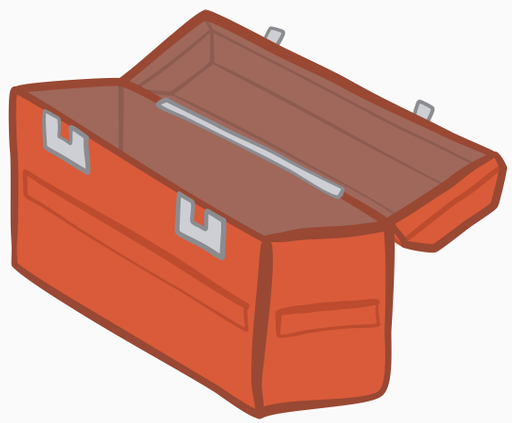
→ preemption limitations

WINE

new queueing identity to complement Little's Law

→ multiserver systems

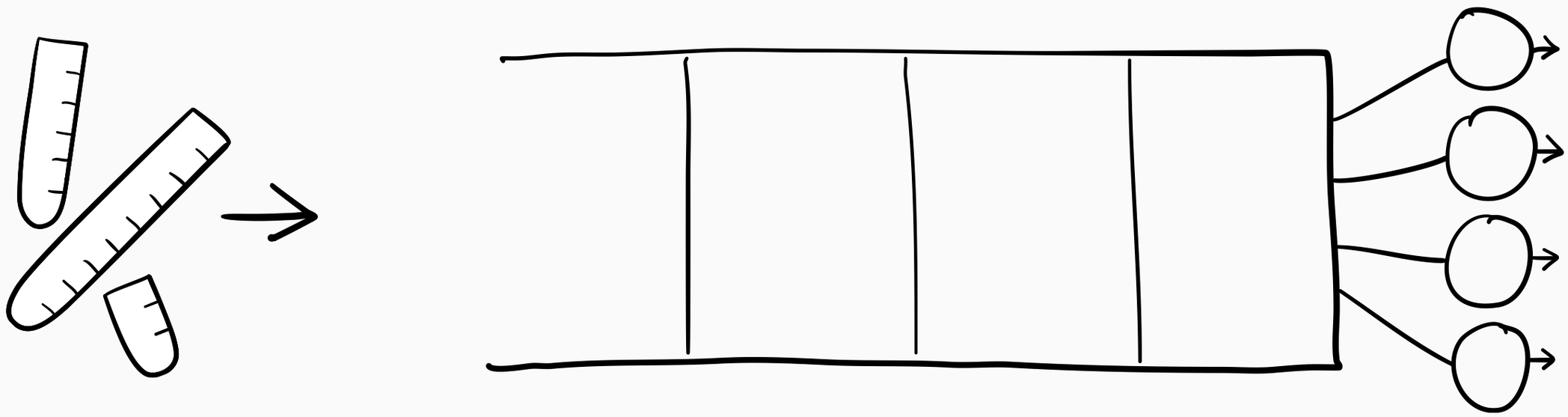
→ noisy size estimates



Part 2

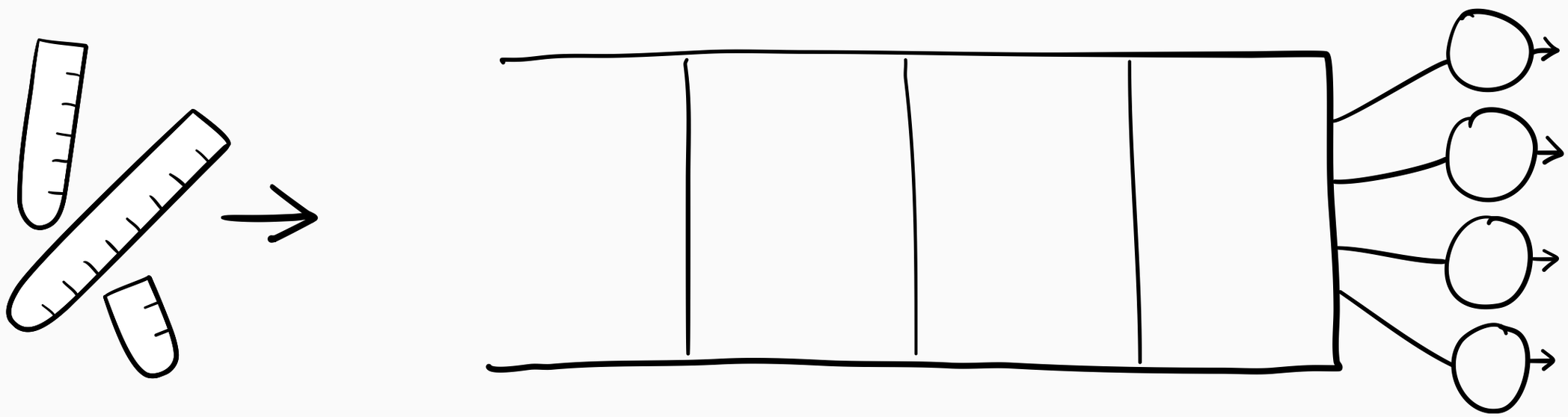
WINE

Multiserver queueing system



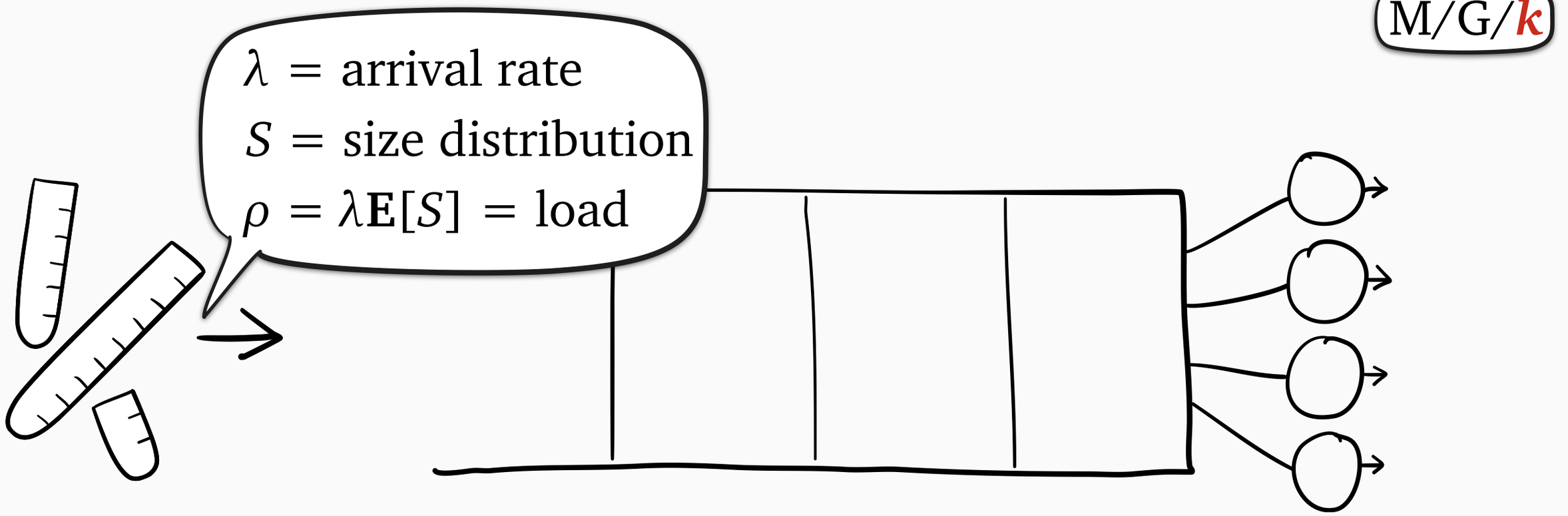
Multiserver queueing system

M/G/*k*



Multiserver queueing system

M/G/***k***



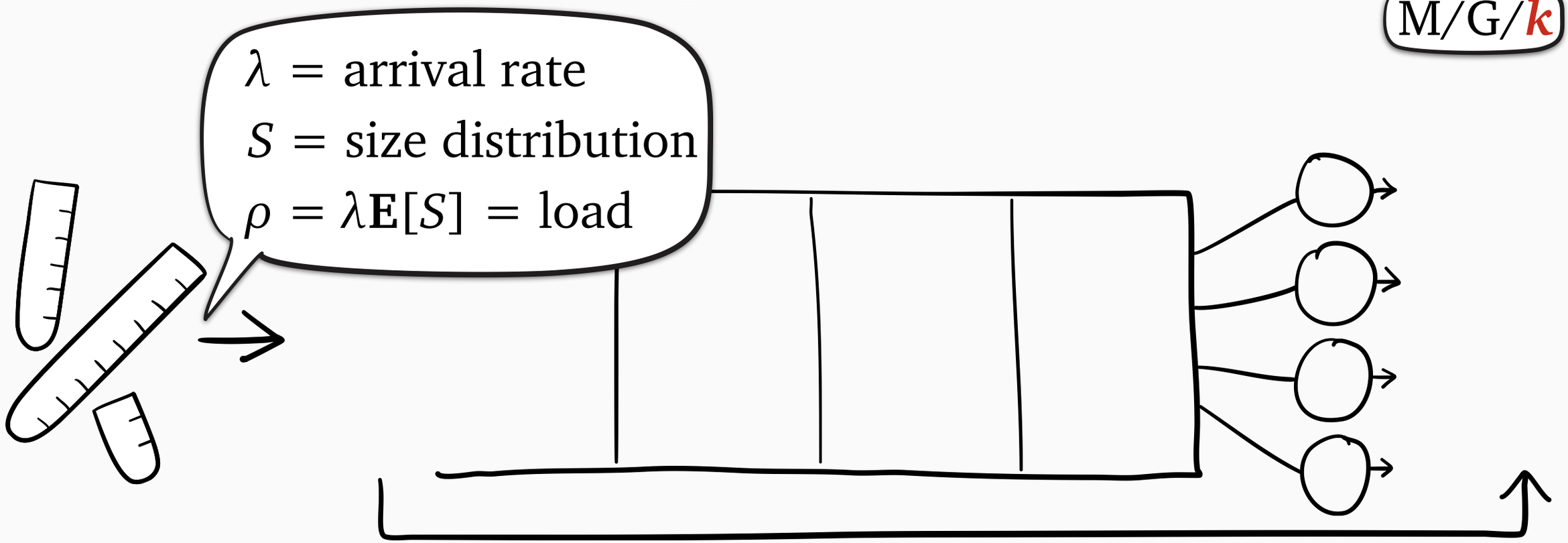
Multiserver queueing system

M/G/k

λ = arrival rate

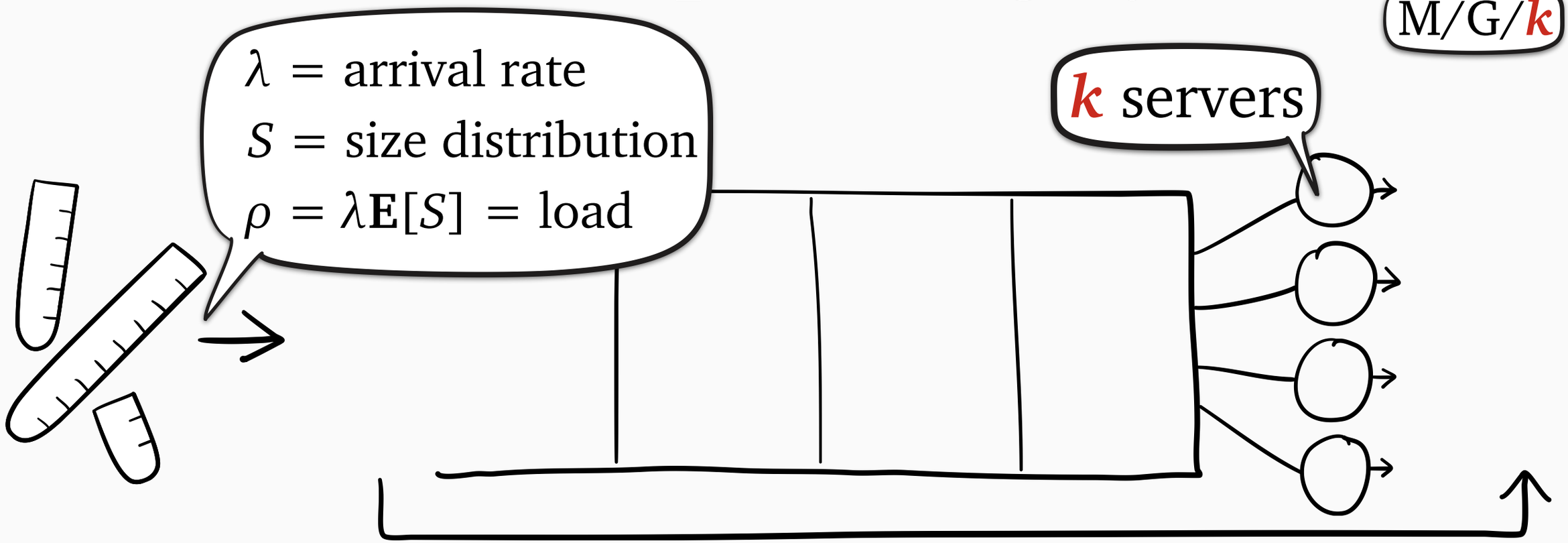
S = size distribution

$\rho = \lambda \mathbf{E}[S]$ = load



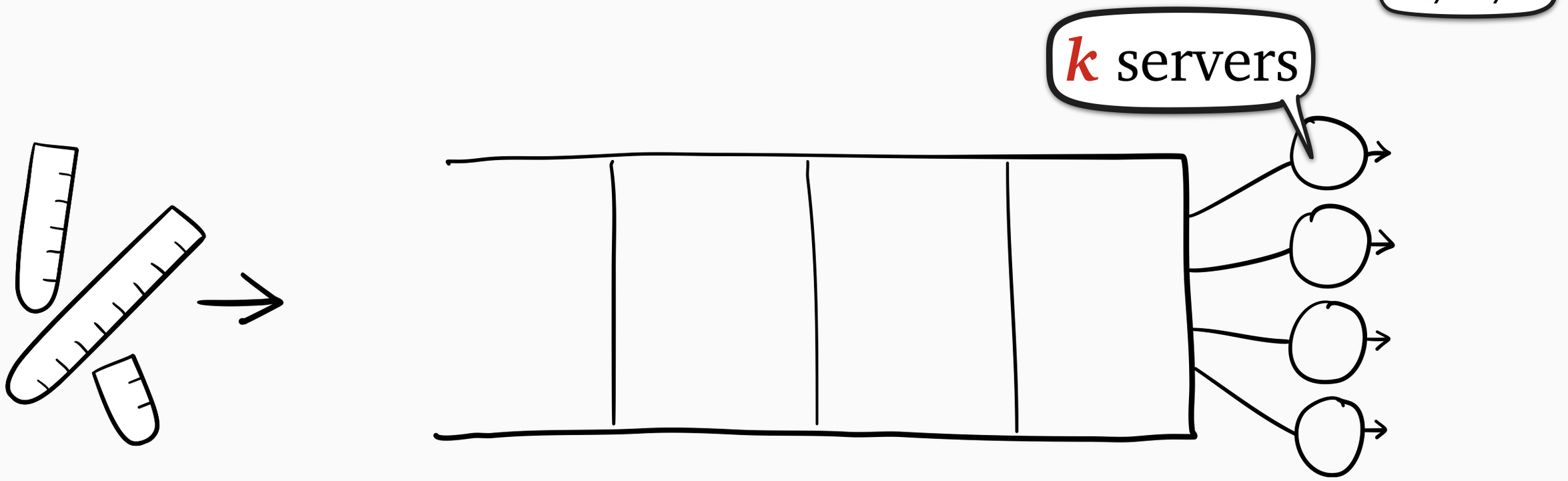
response time distribution T

Multiserver queueing system

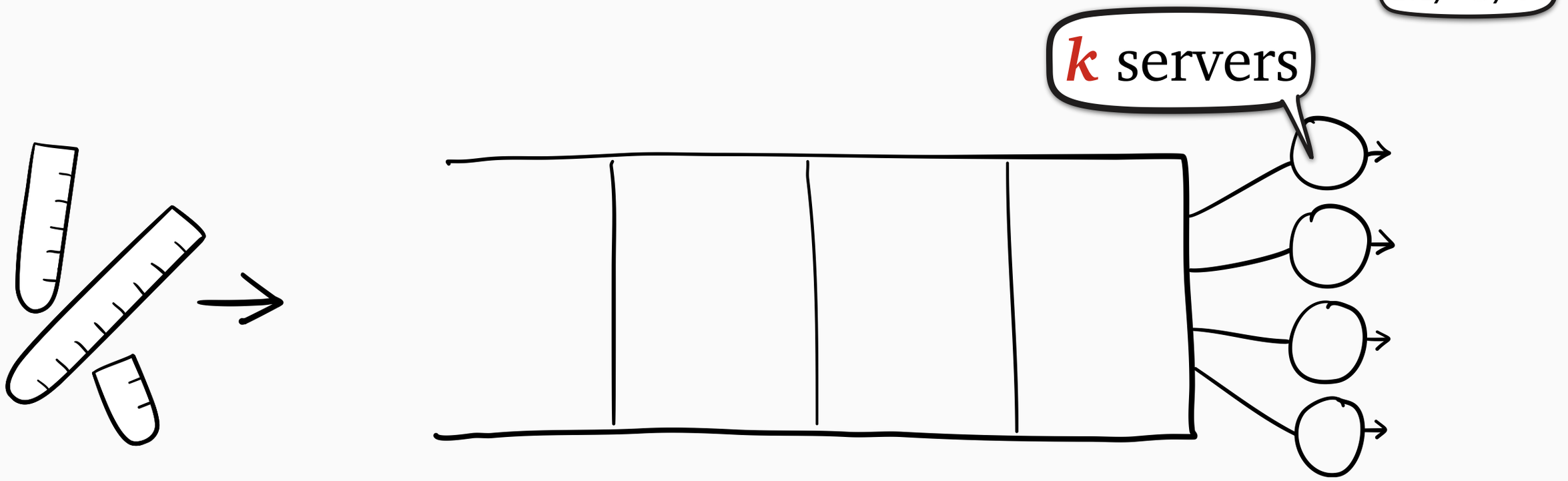


response time distribution T

Multiserver queueing system



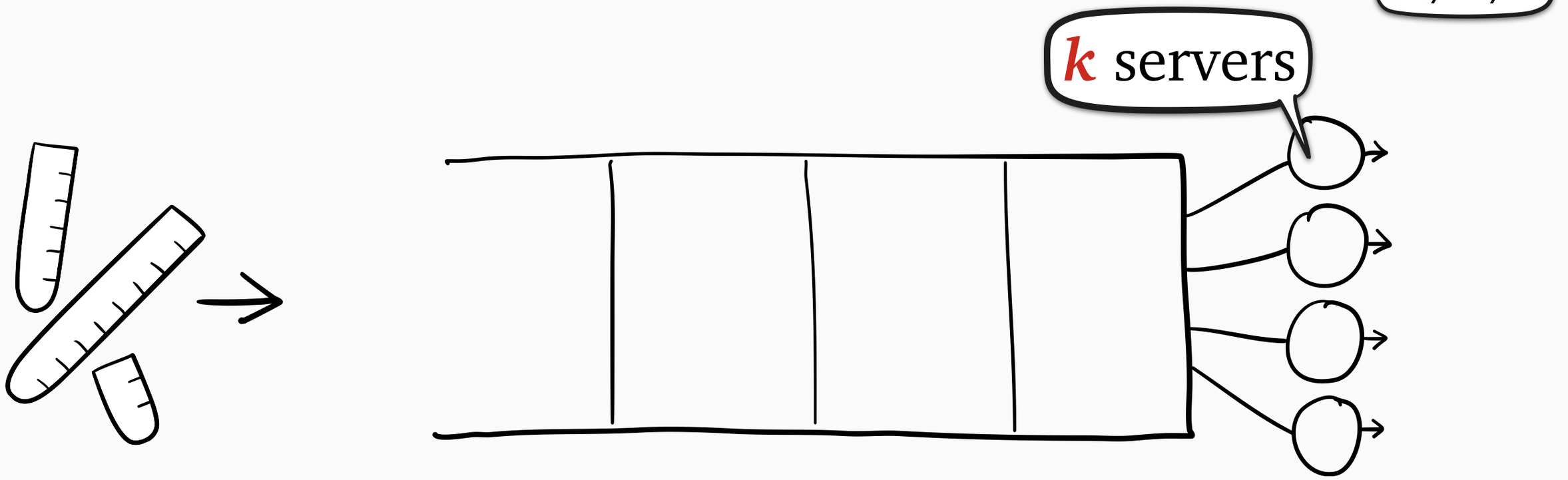
Multiserver queueing system



SRPT: **rank** = remaining size

Multiserver queueing system

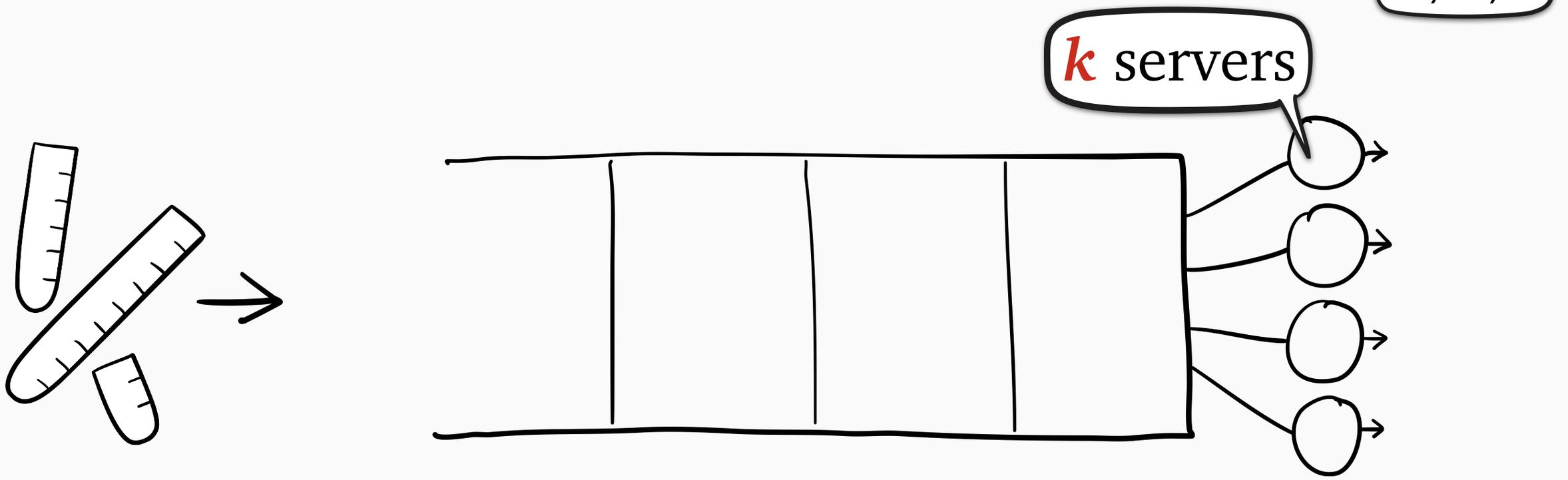
M/G/k



SRPT-1 (single-server): serves the job of least **rank**

SRPT: **rank** = remaining size

Multiserver queueing system

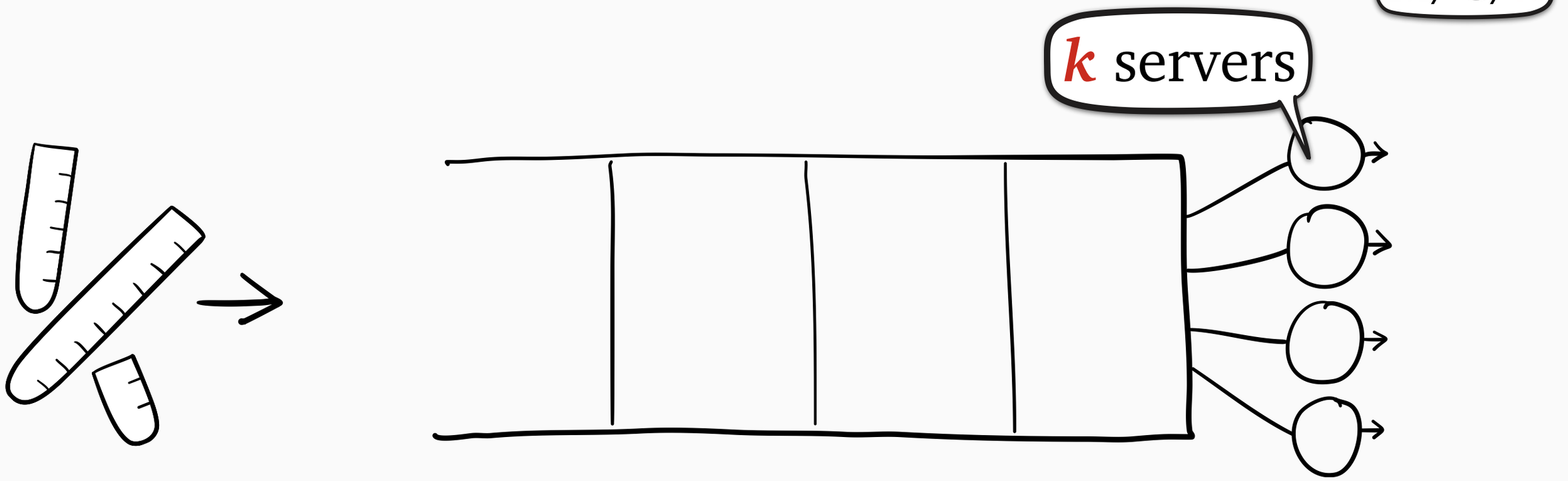


SRPT-1 (single-server): serves the job of least **rank**

SRPT: **rank** = remaining size

SRPT-*k* (multiserver): serves the ***k*** jobs of least **rank**

Multiserver queueing system



SRPT-1 (single-server): serves the job of least **rank**

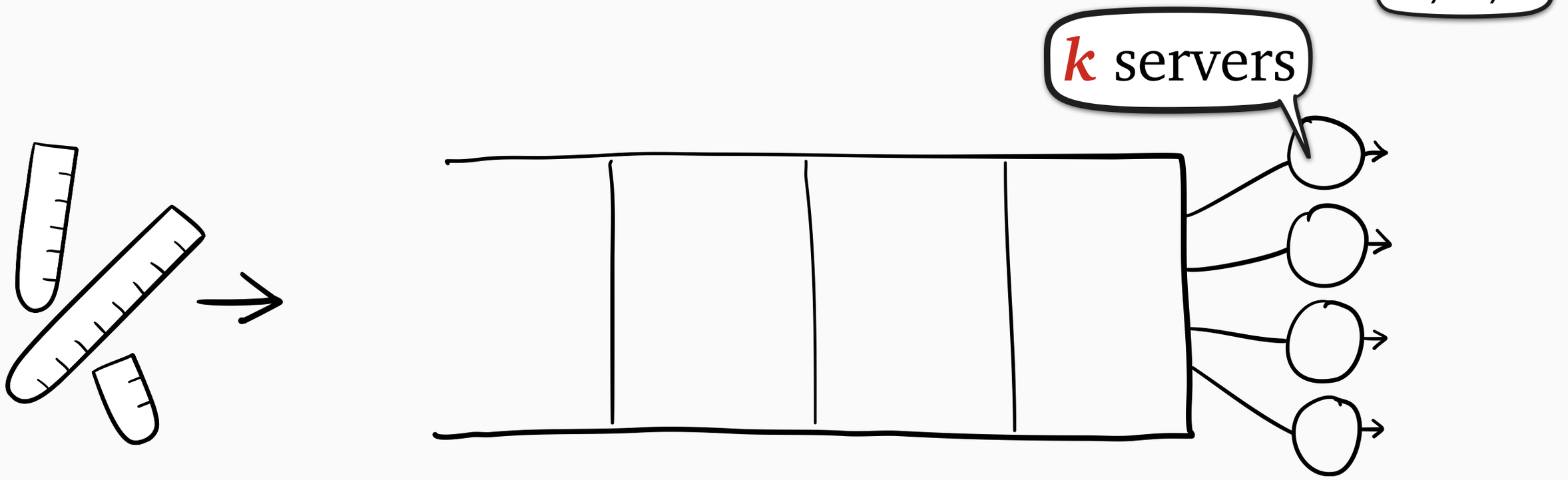
similarly for **Gittins-*k***, etc.

PT: **rank** = remaining size

SRPT-*k* (multiserver): serves the ***k*** jobs of least **rank**

Multiserver queueing system

M/G/k



analyzed

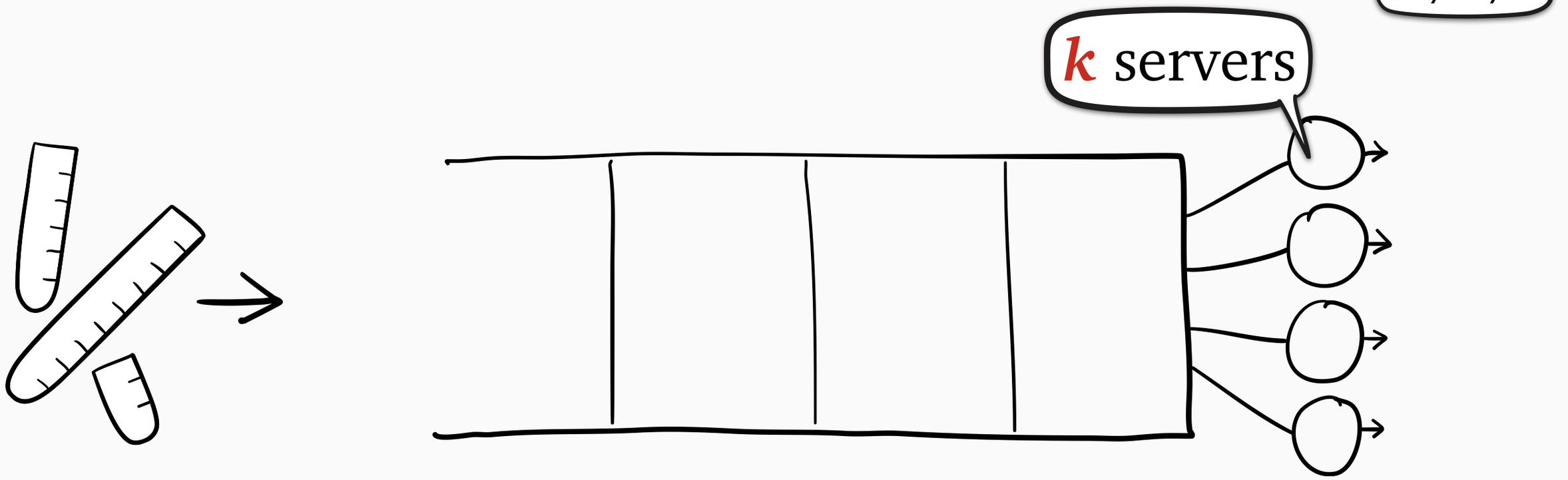
SRPT-1 (single-server): serves the job of least **rank**

SRPT: **rank** = remaining size

SRPT-k (multiserver): serves the k jobs of least **rank**

Multiserver queueing system

M/G/k



analyzed

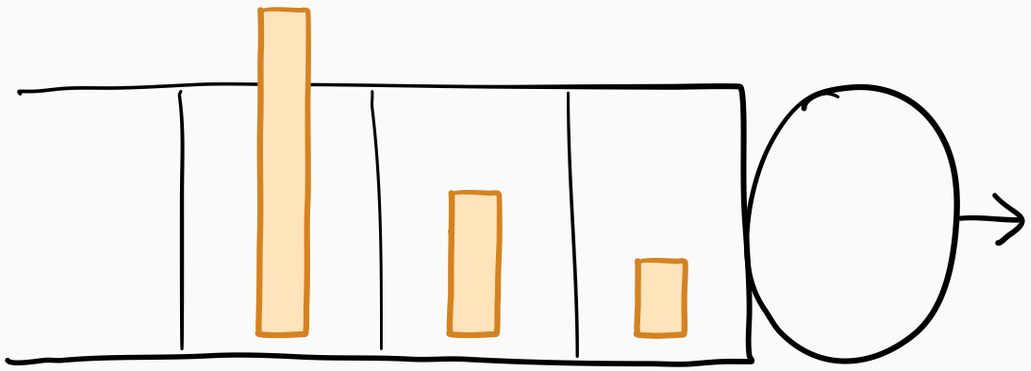
SRPT-1 (single-server): serves the job of least **rank**

analysis open

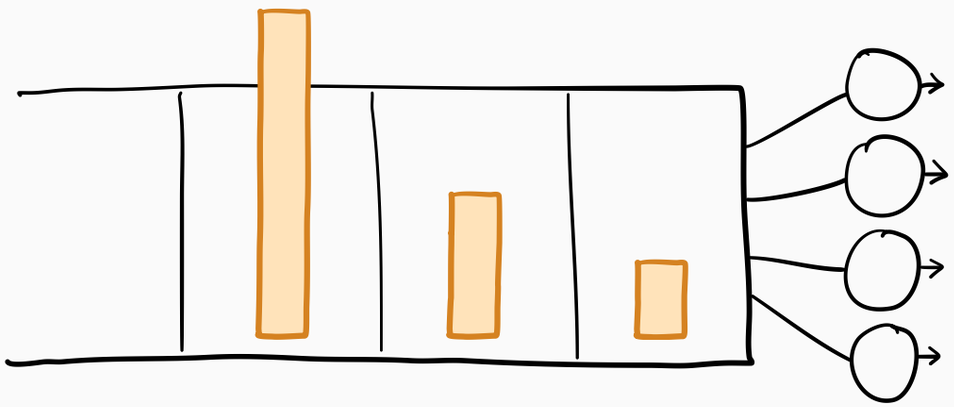
rank = remaining size

SRPT-k (multiserver): serves the k jobs of least **rank**

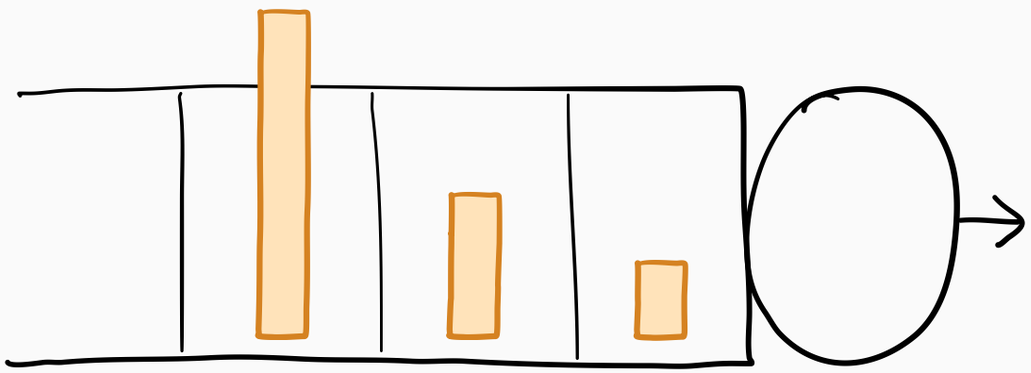
Single-server system



Multiserver system

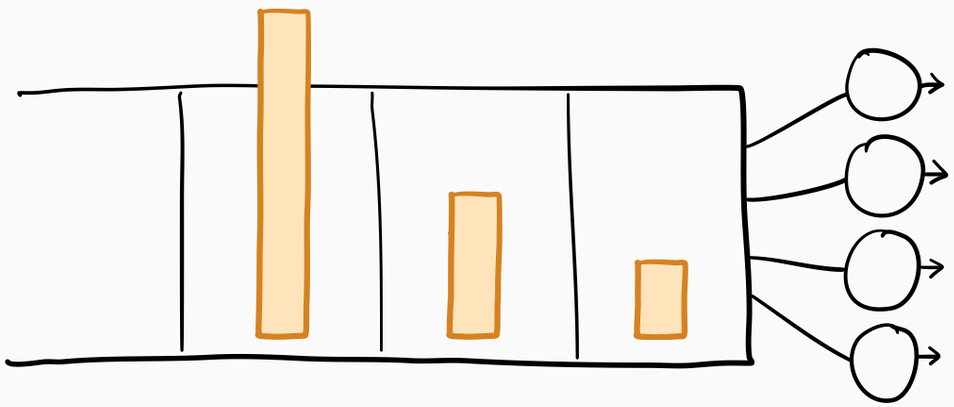


Single-server system

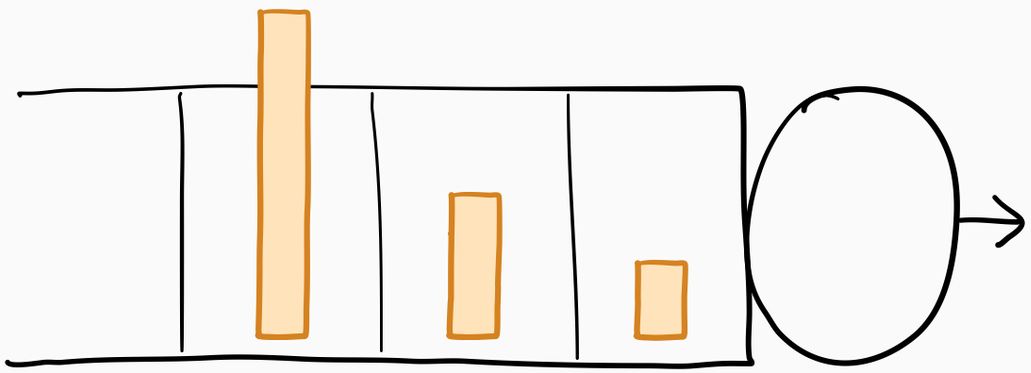


server is “choke point”

Multiserver system



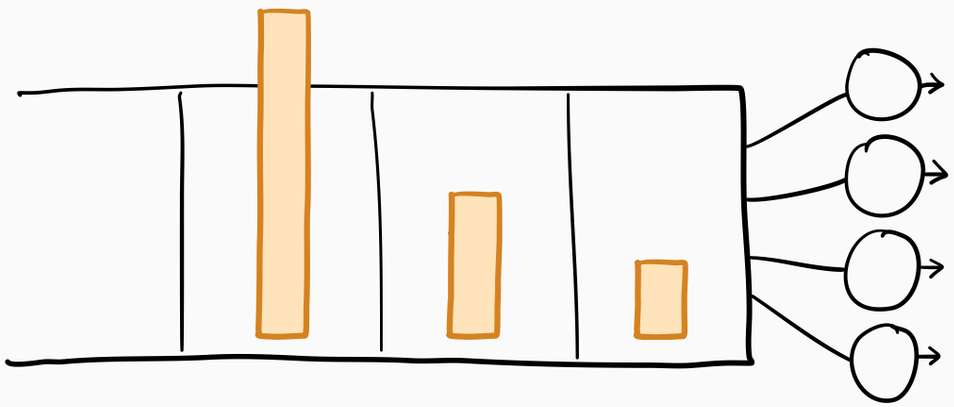
Single-server system



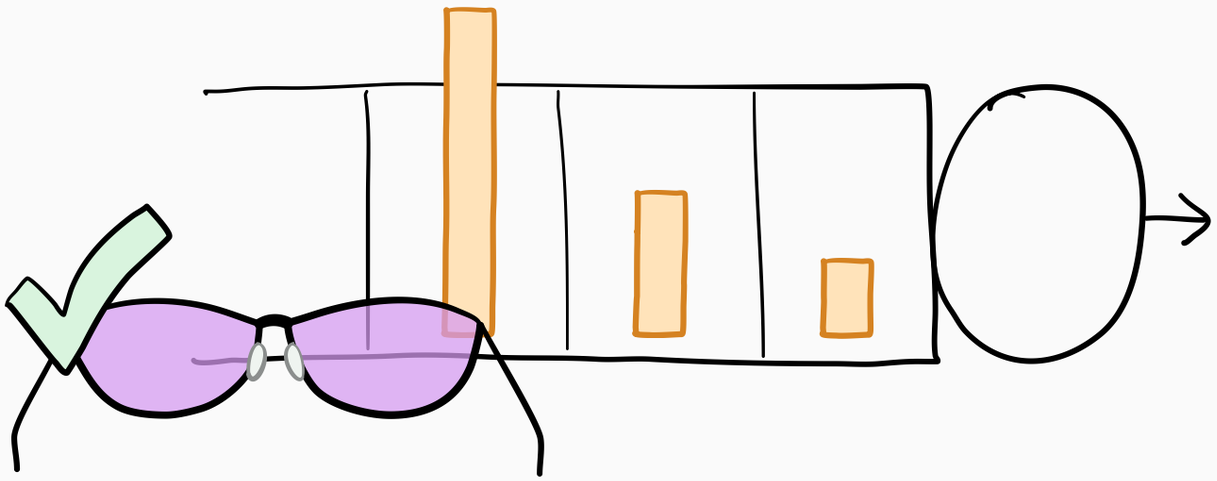
server is “choke point”

rank ordering absolute

Multiserver system



Single-server system

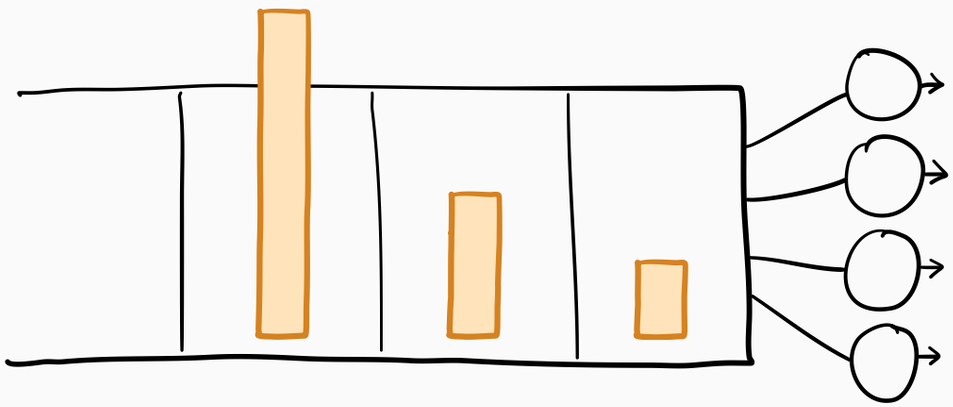


server is “choke point”

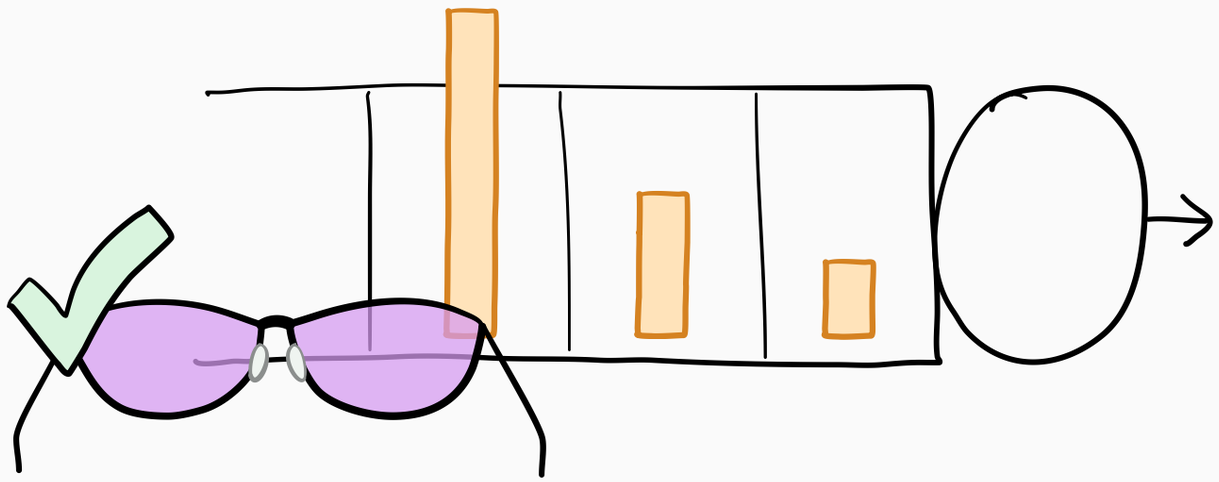
rank ordering absolute

observed ***r***-work determines T

Multiserver system



Single-server system



server is “choke point”

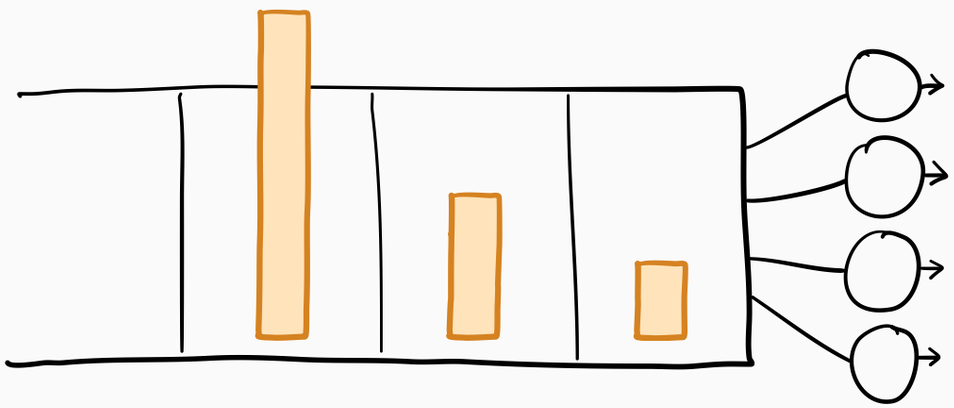
⇓

rank ordering absolute

⇓

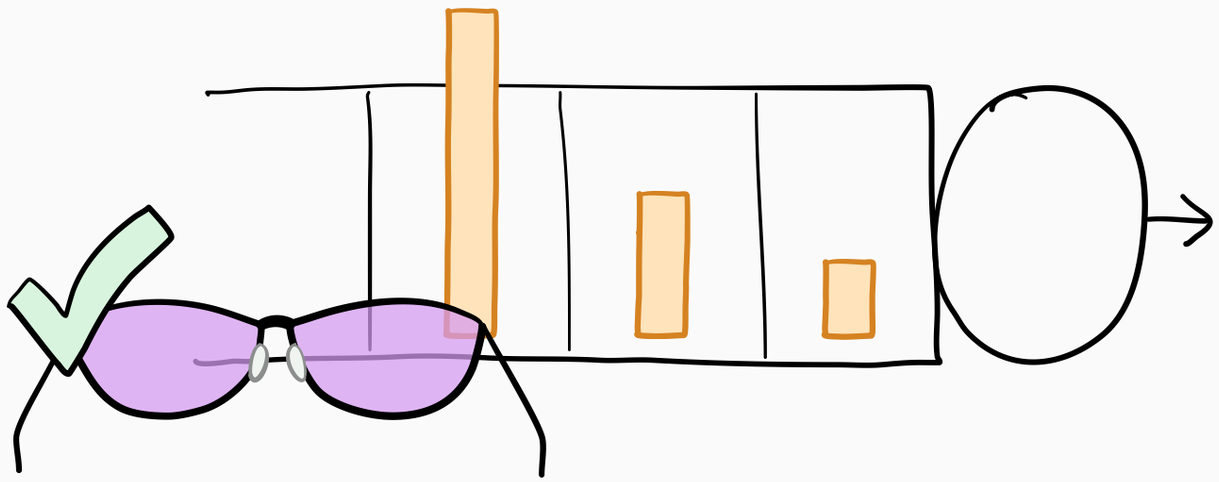
observed r-work determines T

Multiserver system



no single “choke point”

Single-server system



server is “choke point”

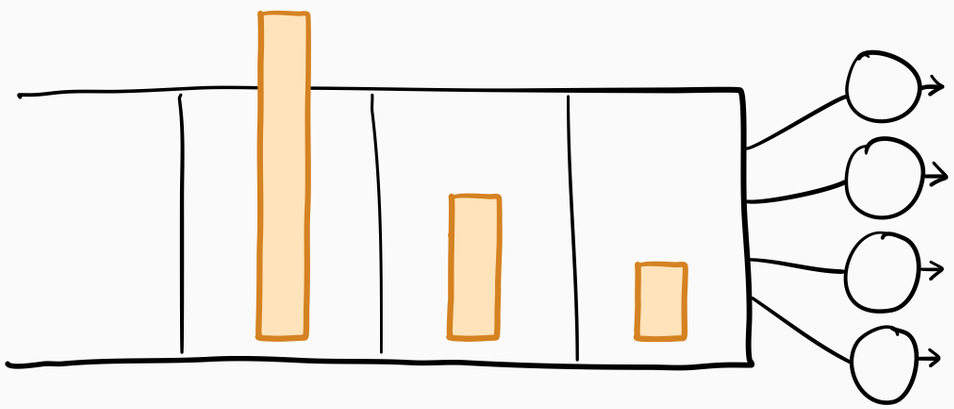
⇓

rank ordering absolute

⇓

observed r-work determines T

Multiserver system

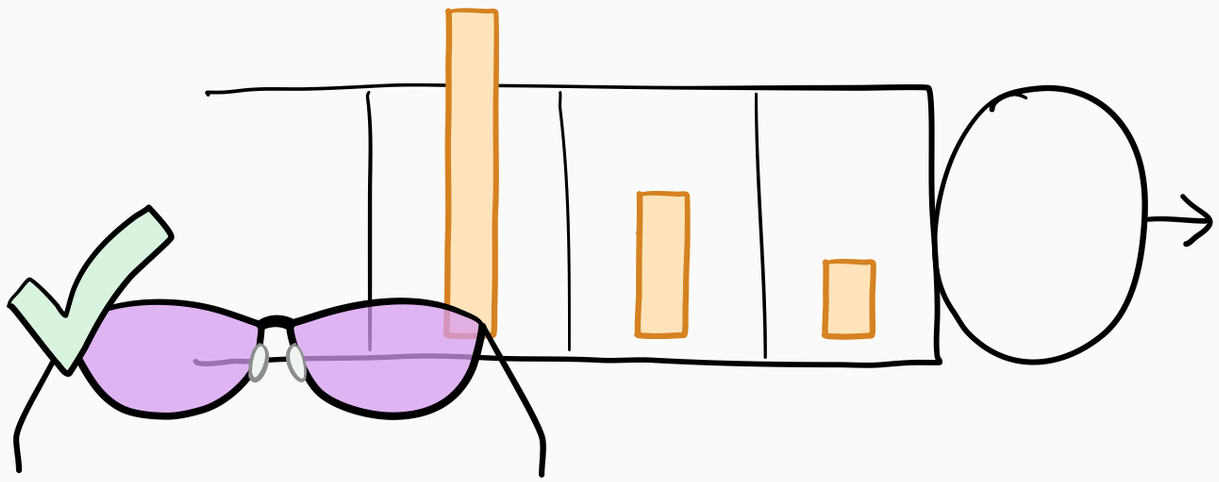


no single “choke point”

⇓

rank ordering *not* absolute

Single-server system



server is “choke point”

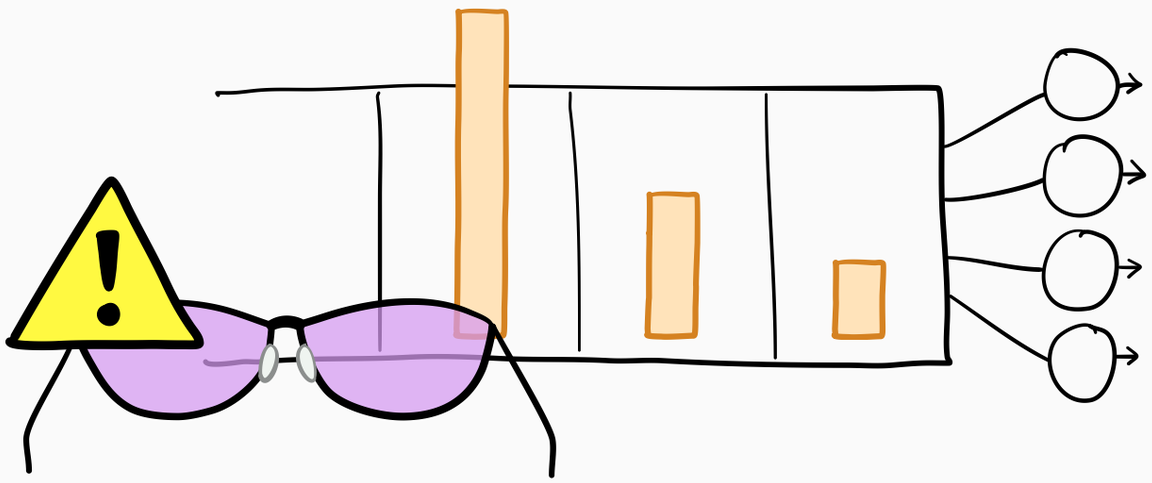
⇓

rank ordering absolute

⇓

observed ***r***-work determines T

Multiserver system



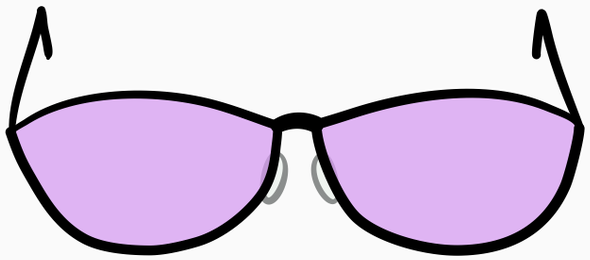
no single “choke point”

⇓

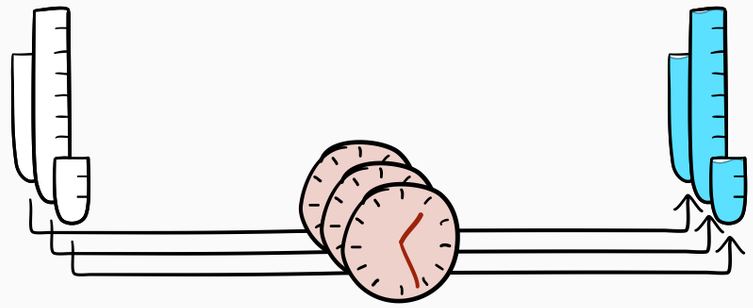
rank ordering *not* absolute

⇓

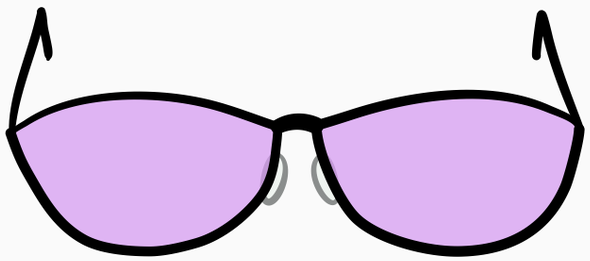
observed ***r***-work *not enough!*



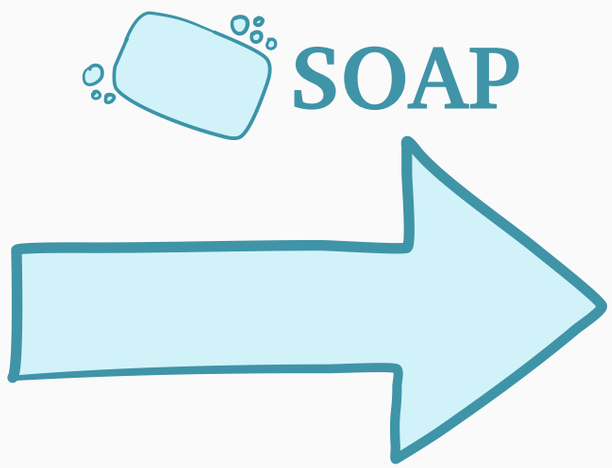
r-work $W(r)$

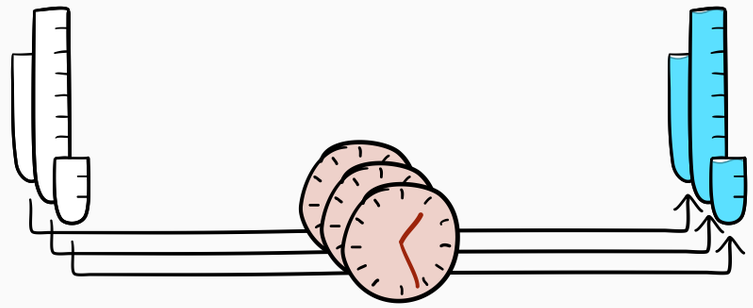


response time T

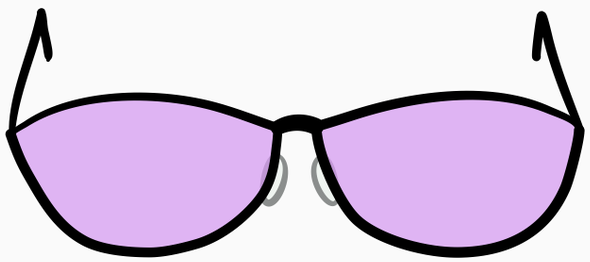


r-work $W(r)$

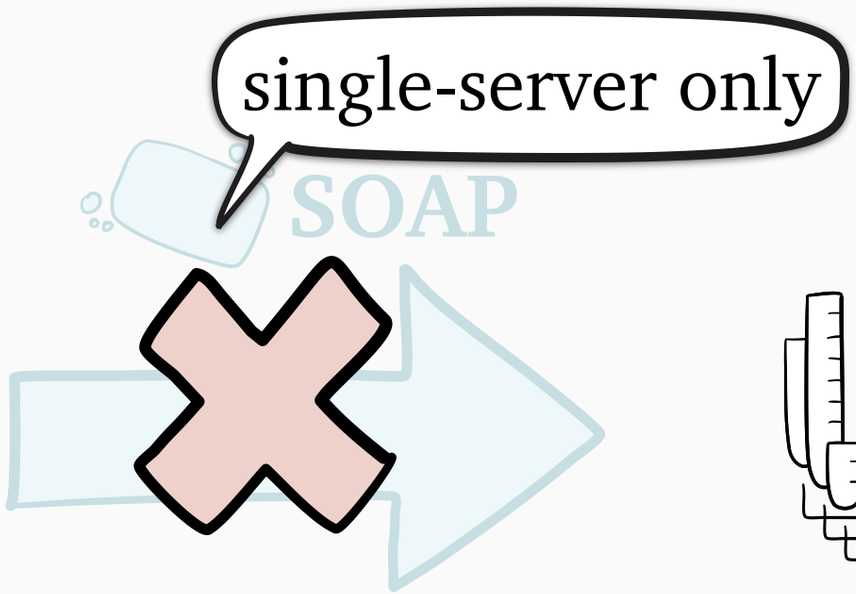


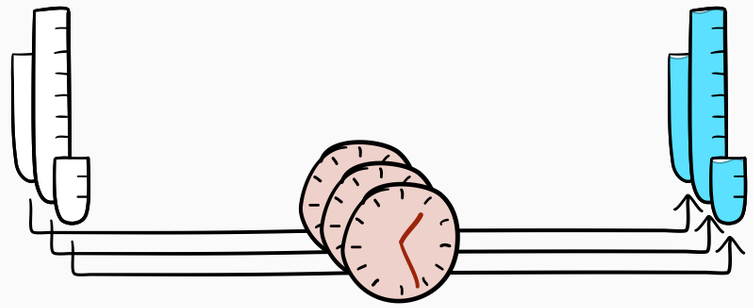


response time T



r-work $W(r)$

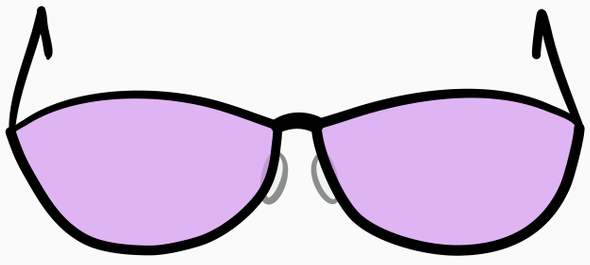




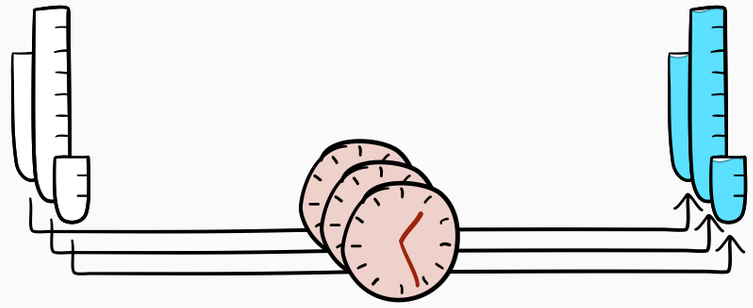
response time T

single-server only

SOAP

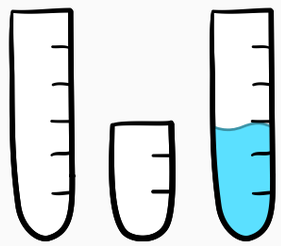


r-work $W(r)$

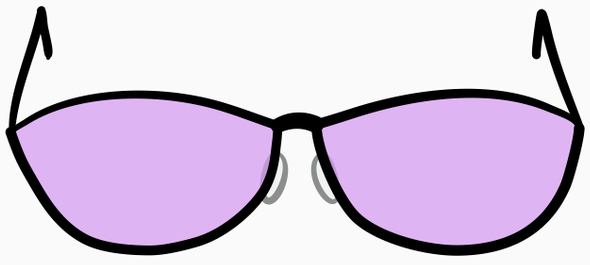


response time T

Little's law



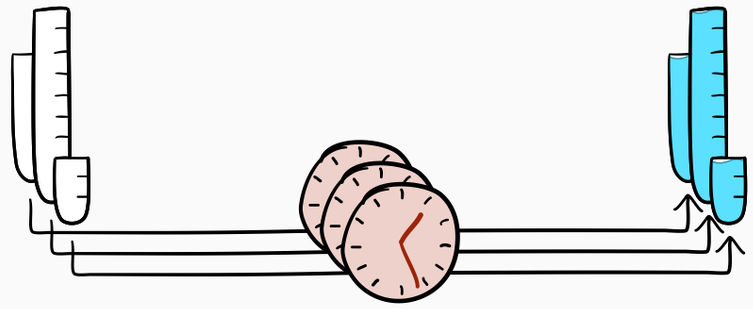
number of jobs N



r-work $W(r)$

single-server only

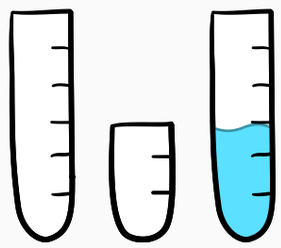
SOAP



response time T

$$\mathbf{E}[N] = \lambda \mathbf{E}[T]$$

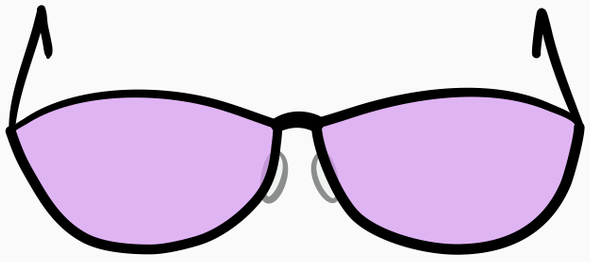
Little's law



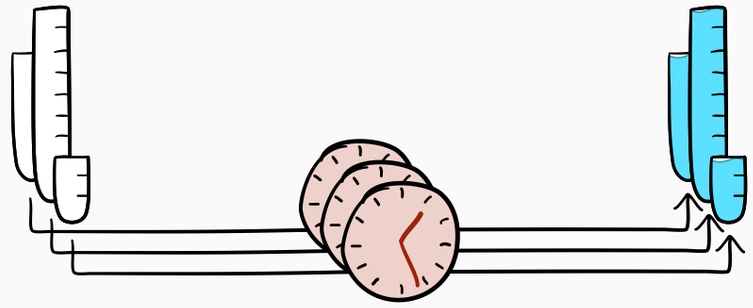
number of jobs N

single-server only

SOAP



r-work $W(r)$

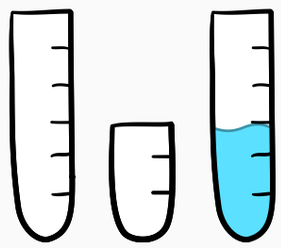


response time T

$\mathbf{E}[N] = \lambda \mathbf{E}[T]$

Little's law

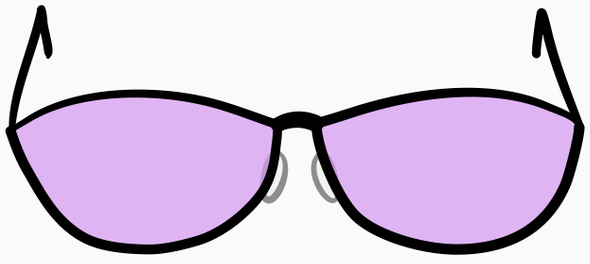
any number of servers



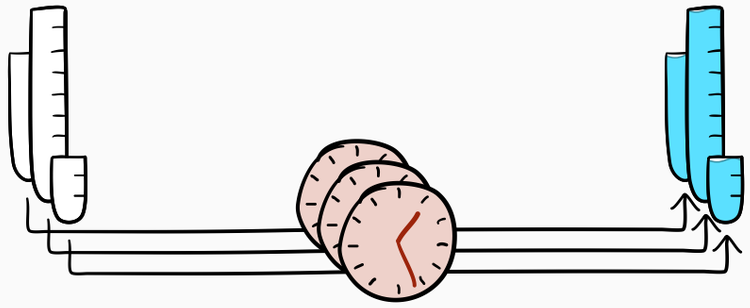
number of jobs N

single-server only

SOAP



r-work $W(r)$



response time T

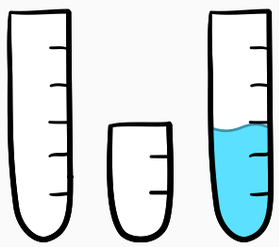
NEW!

WINE

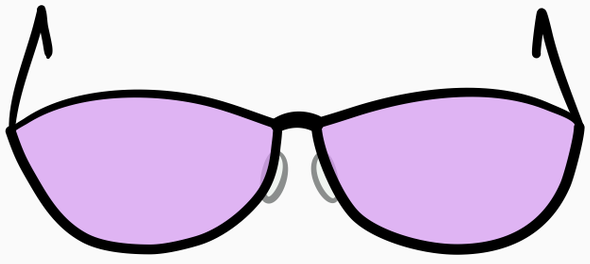
$$\mathbf{E}[N] = \lambda \mathbf{E}[T]$$

Little's law

any number of servers



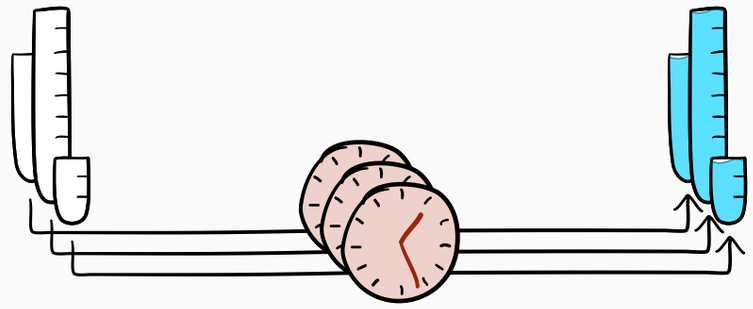
number of jobs N



r-work $W(r)$

single-server only

SOAP



response time T

NEW!

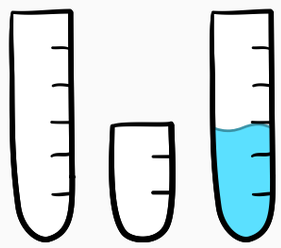
WINE

any number of servers

$\mathbf{E}[N] = \lambda \mathbf{E}[T]$

Little's law

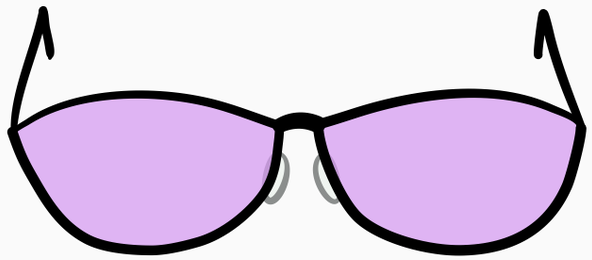
any number of servers



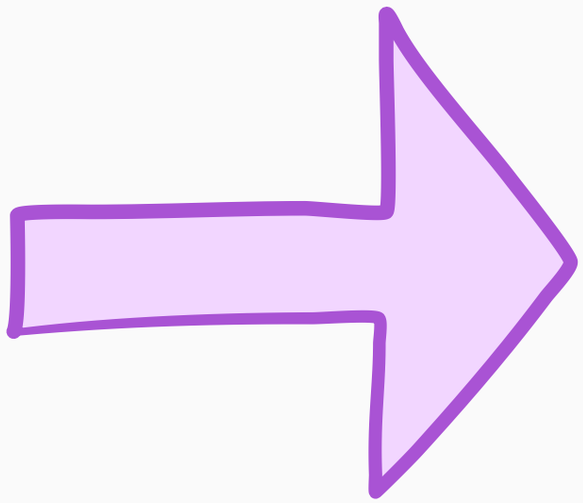
number of jobs N

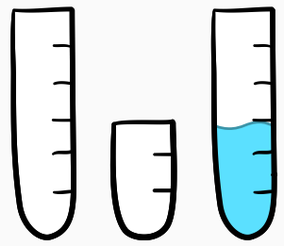
WINE

Work **I**ntegral **N**umber **E**quality



r-work $W(r)$

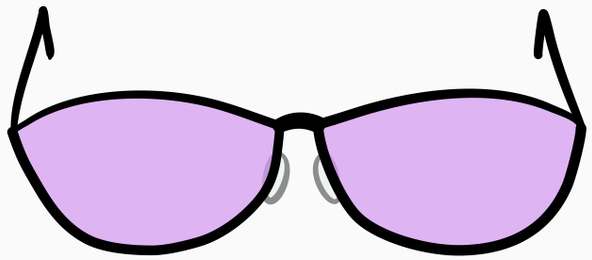




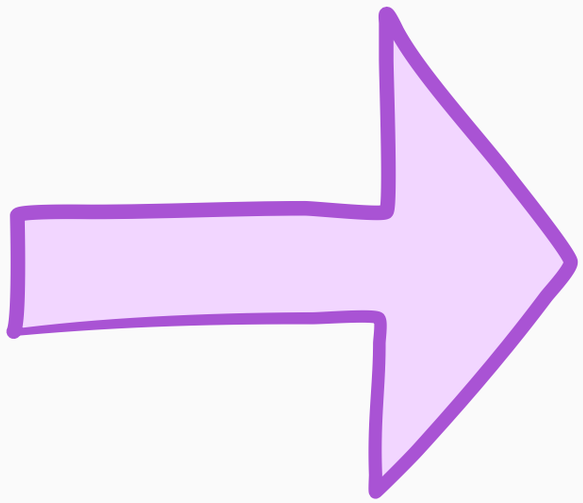
number of jobs N

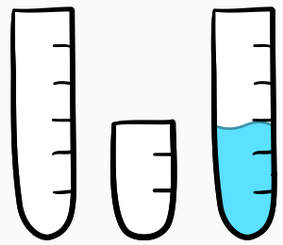
WINE

Work **I**ntegral **N**umber **E**quality

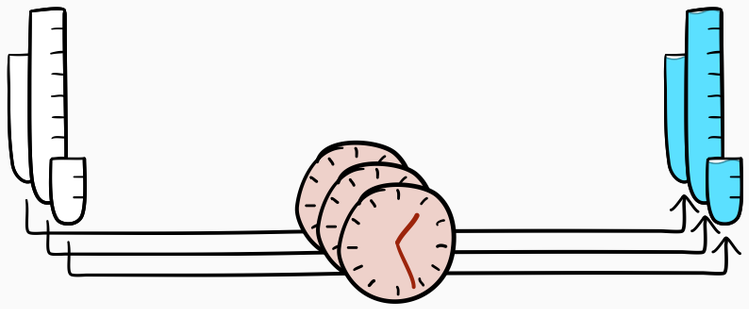


r-work $W(r)$





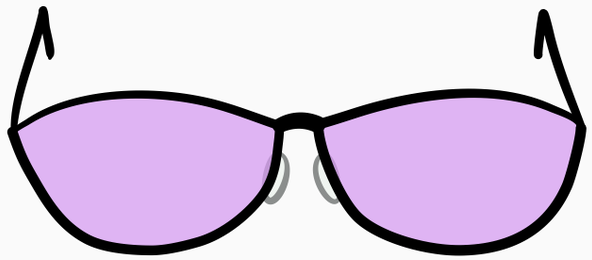
number of jobs N



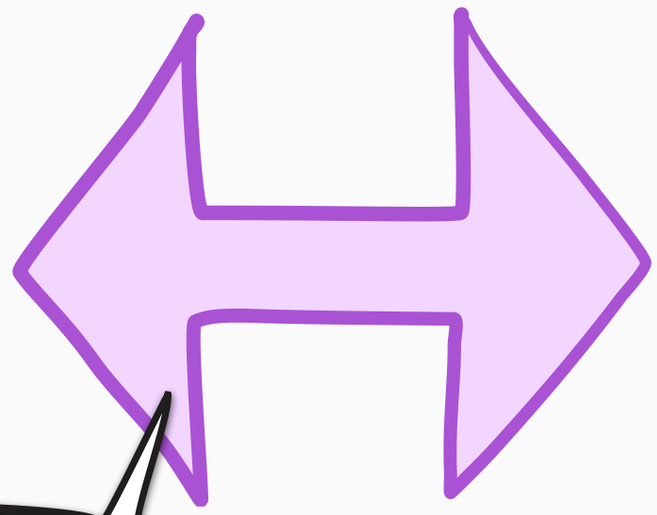
mean response time $\mathbf{E}[T]$

WINE

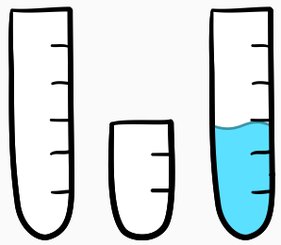
Work **I**ntegral **N**umber **E**quality



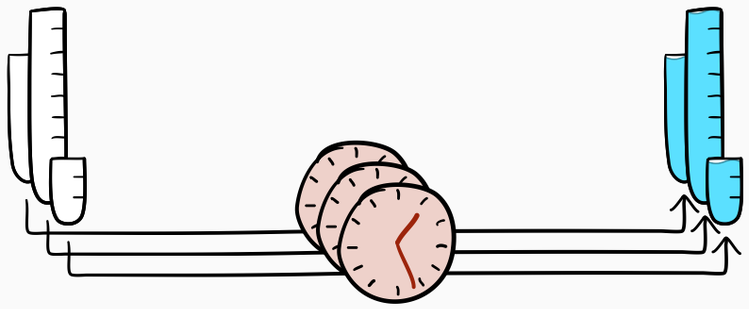
r-work $W(r)$



reduces $\mathbf{E}[T]$ to r-work



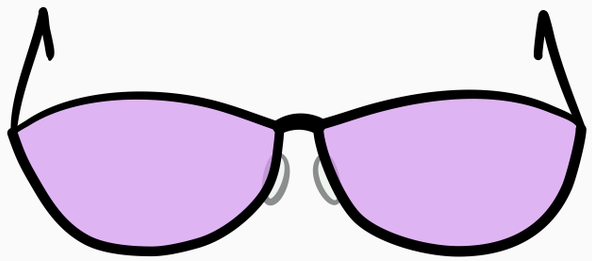
number of jobs N



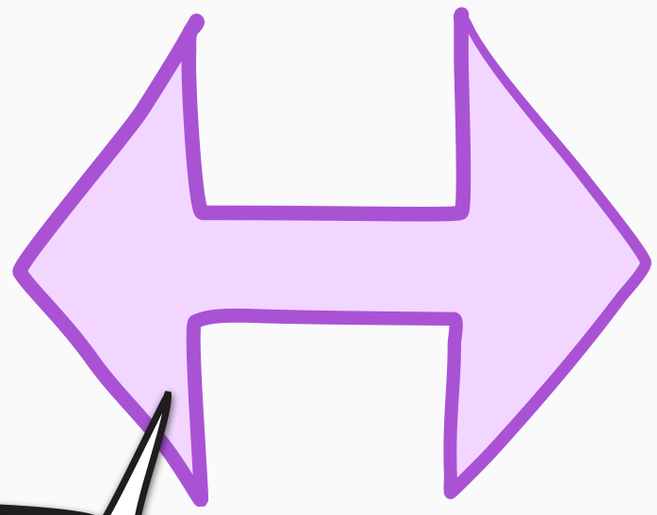
mean response time $\mathbf{E}[T]$

WINE

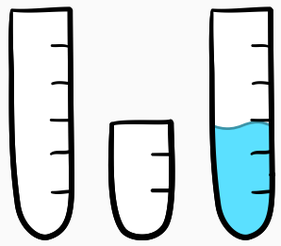
Work **I**ntegral **N**umber **E**quality



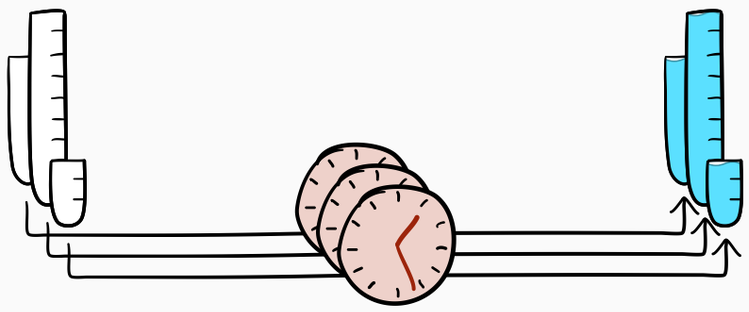
r-work $W(r)$



reduces $\mathbf{E}[T]$ to r-work

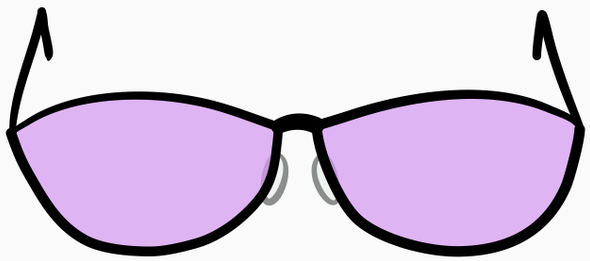


number of jobs N

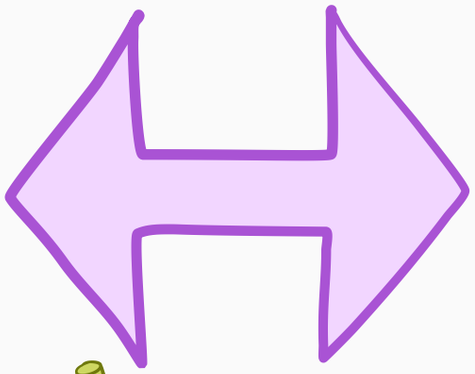


mean response time $\mathbf{E}[T]$

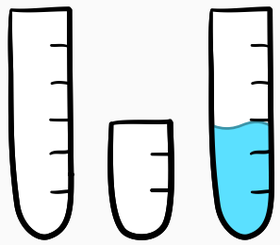
First analysis of SRPT-k, **Gittins**-k, **noisy** size estimates



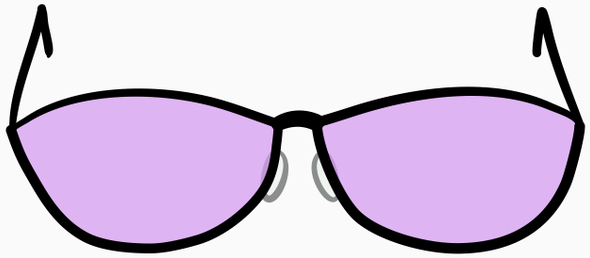
***r*-work $W(r)$**



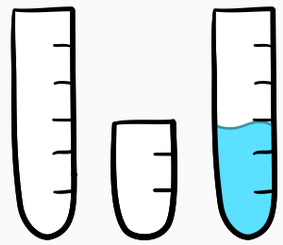
WINE



number of jobs N

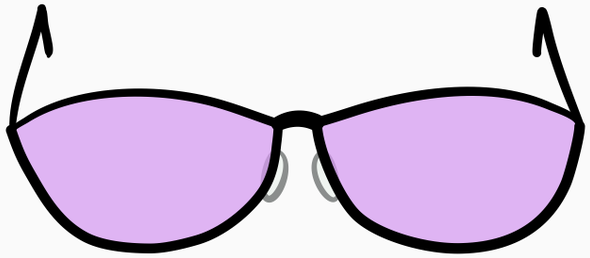


r-work $W(r)$

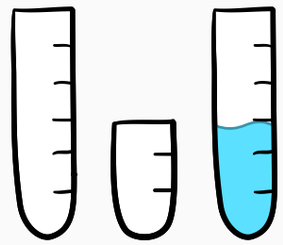


number of jobs N

What is *r*-work?



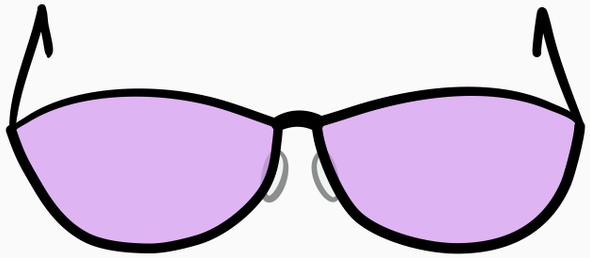
***r*-work $W(r)$**



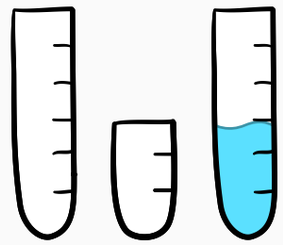
number of jobs N

What is ***r*-work?**

How do we get number of jobs from ***r*-work?**



***r*-work $W(r)$**

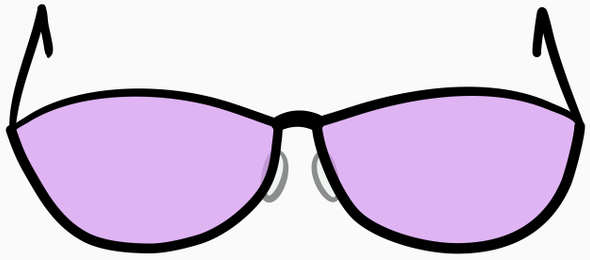


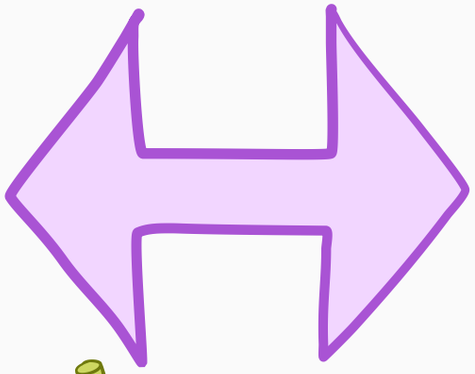
number of jobs N

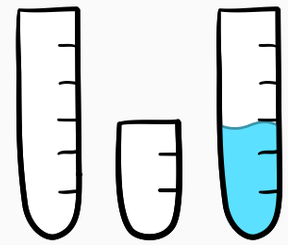
What is ***r*-work?**

How do we get number of jobs from ***r*-work?**

How do we analyze ***r*-work?**







***r*-work $W(r)$**

WINE

number of jobs N

This talk:
SRPT-***k***

What is ***r***-work?

How do we get number of jobs from ***r***-work?

How do we analyze ***r***-work?

Defining r-work

for SRPT

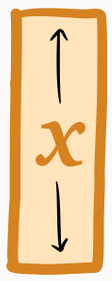
$W(r)$ = work relevant to **rank r**

Defining r-work

for SRPT

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases}$

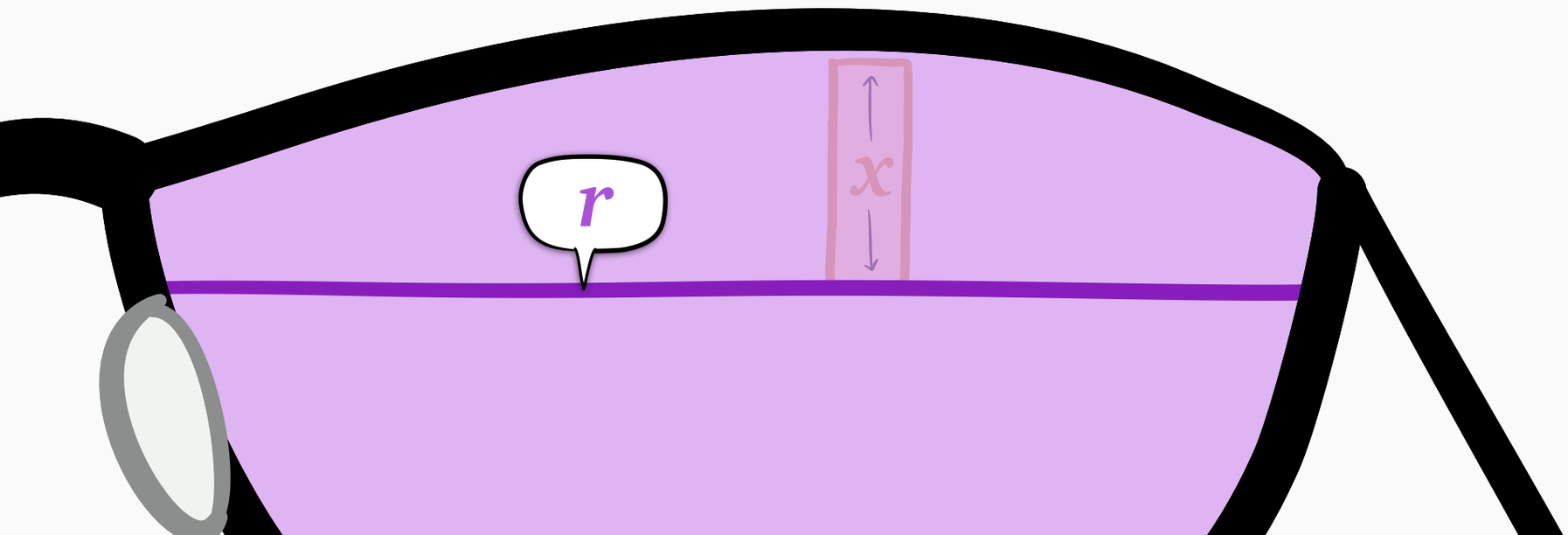


Defining r-work

for SRPT

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases}$

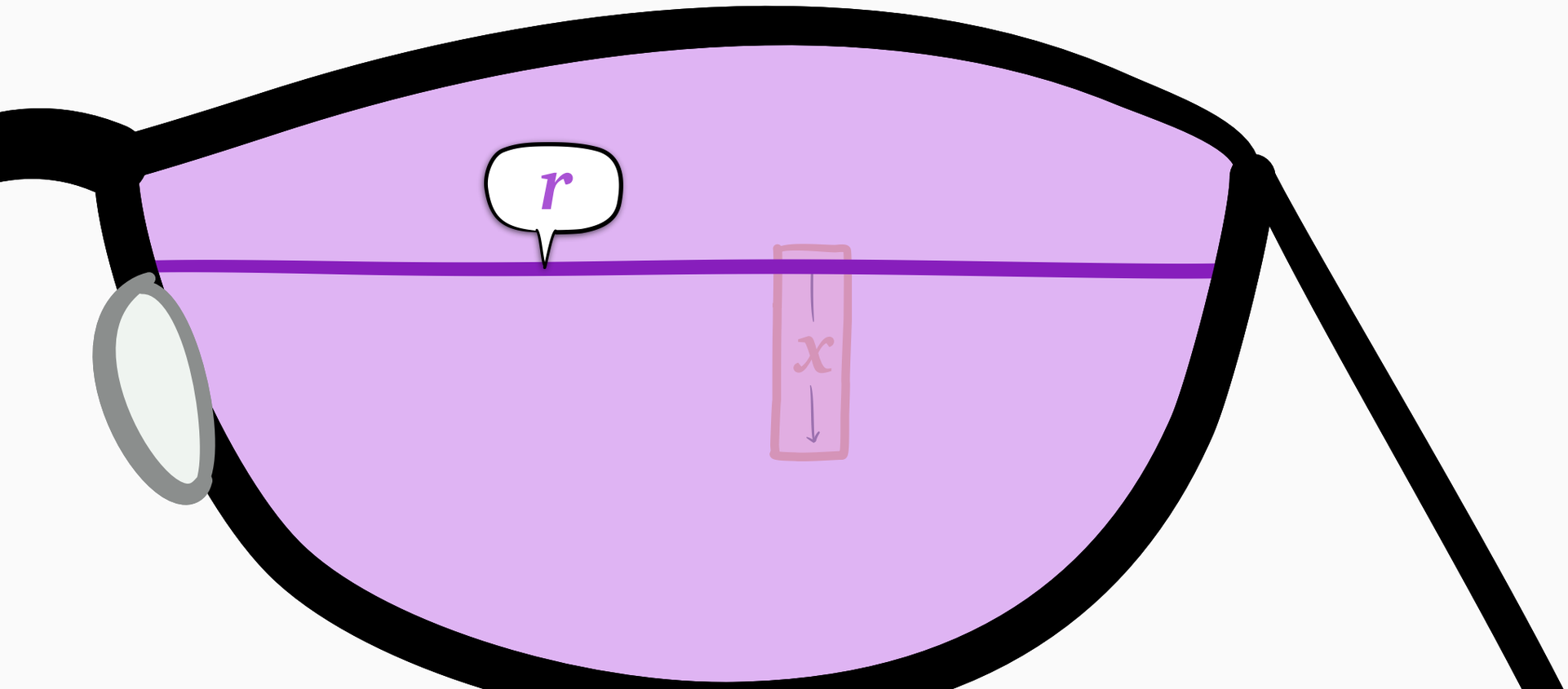


Defining r-work

for SRPT

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases}$

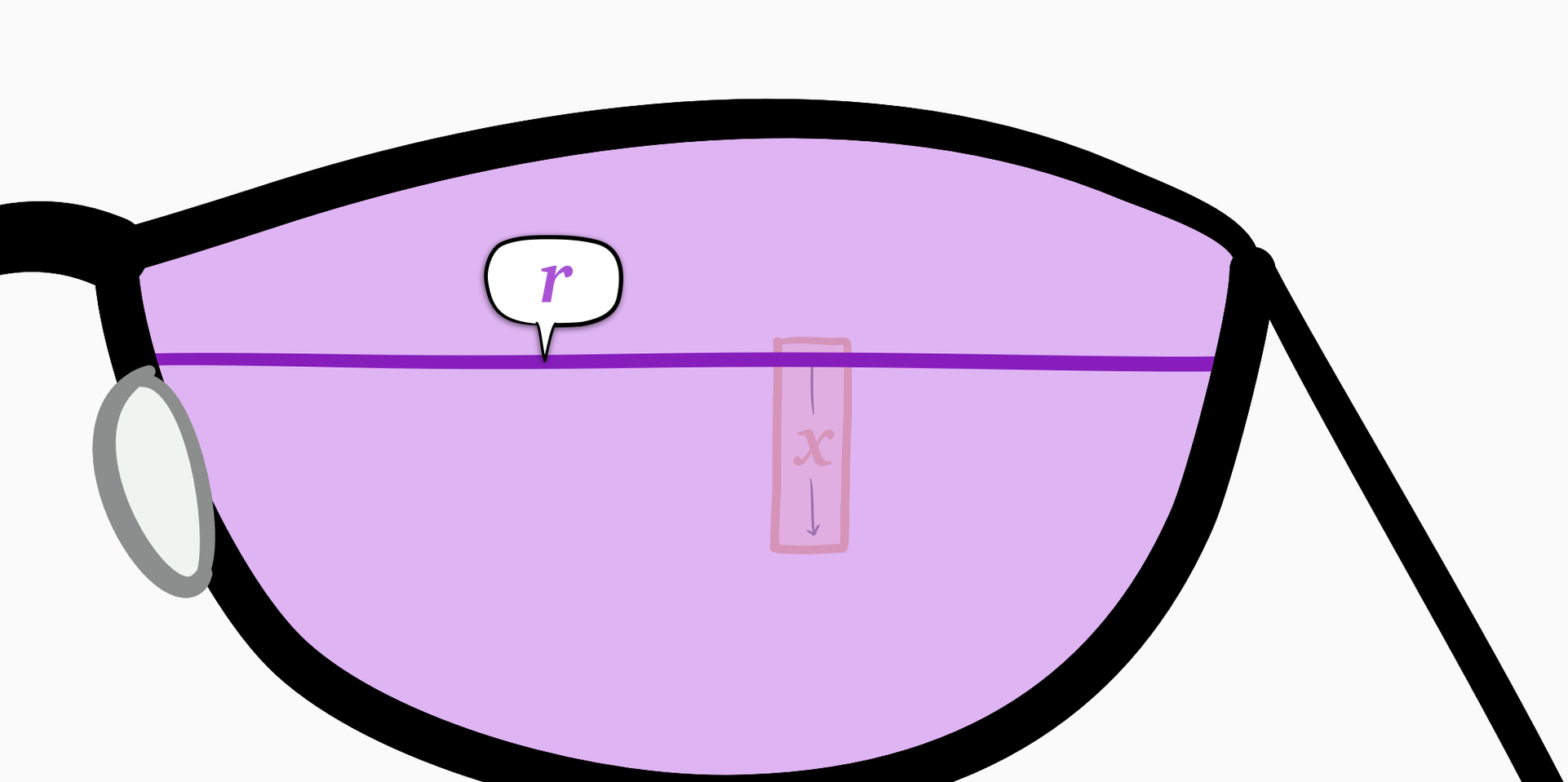


Defining r-work

for SRPT

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \end{cases}$

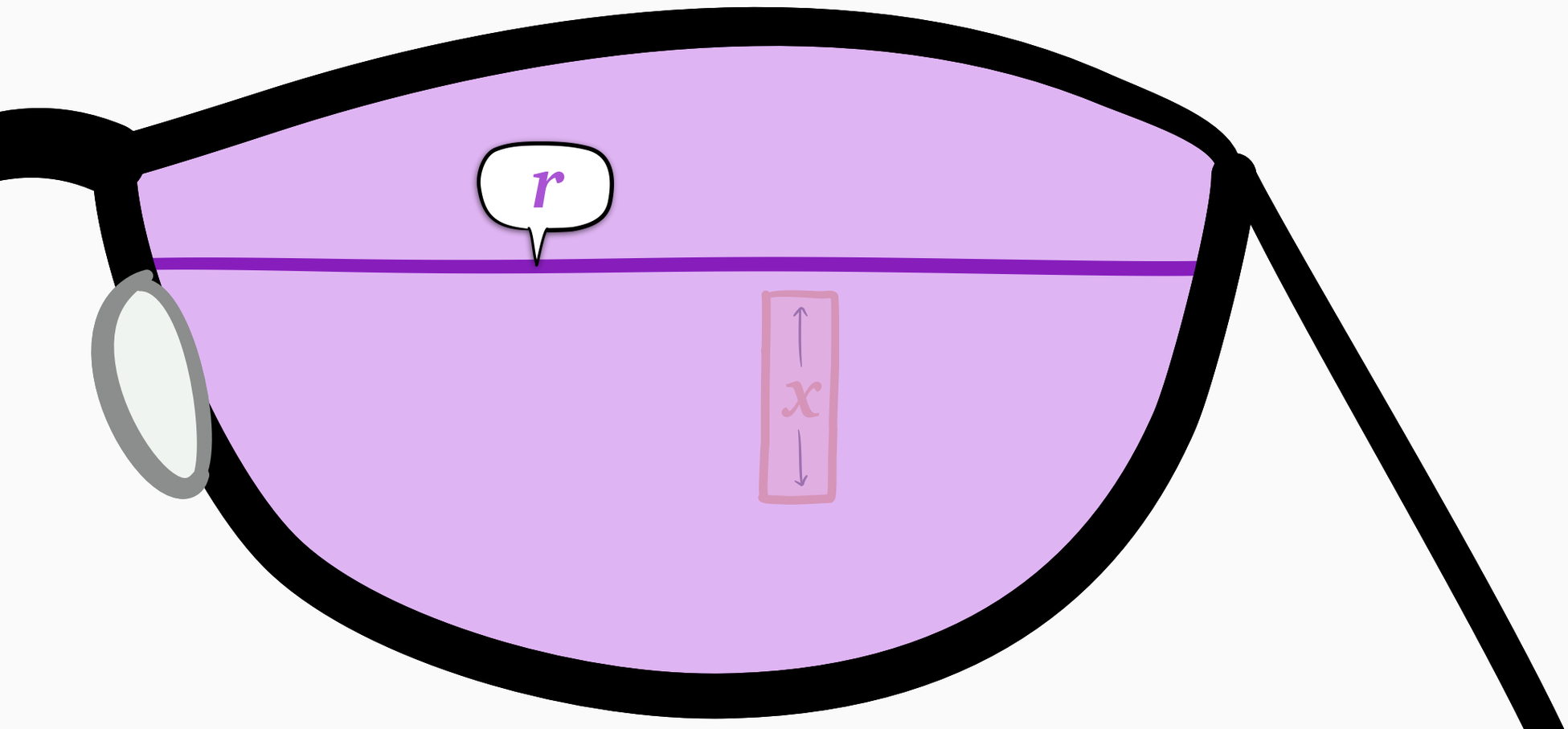


Defining r-work

for SRPT

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \end{cases}$

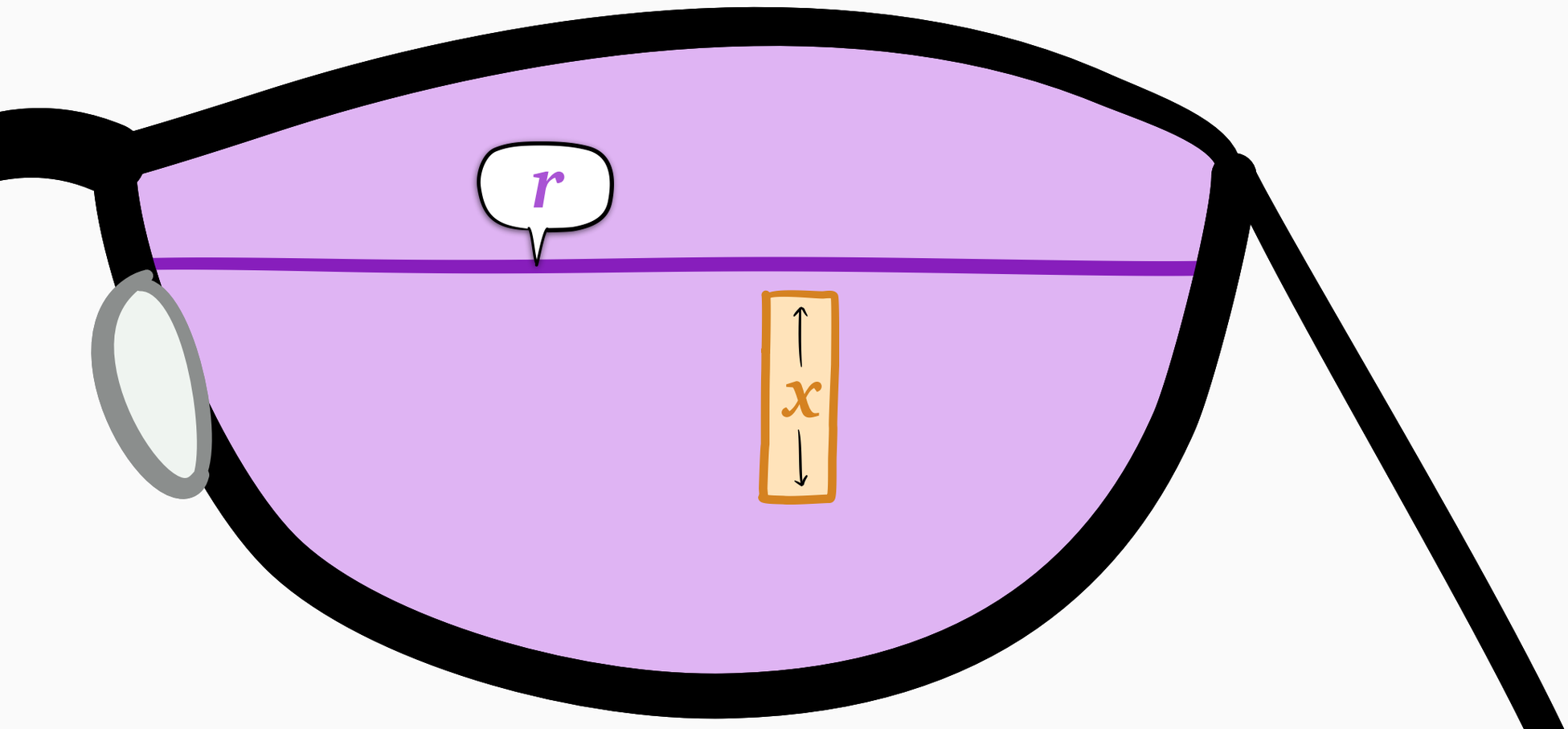


Defining r-work

for SRPT

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \end{cases}$

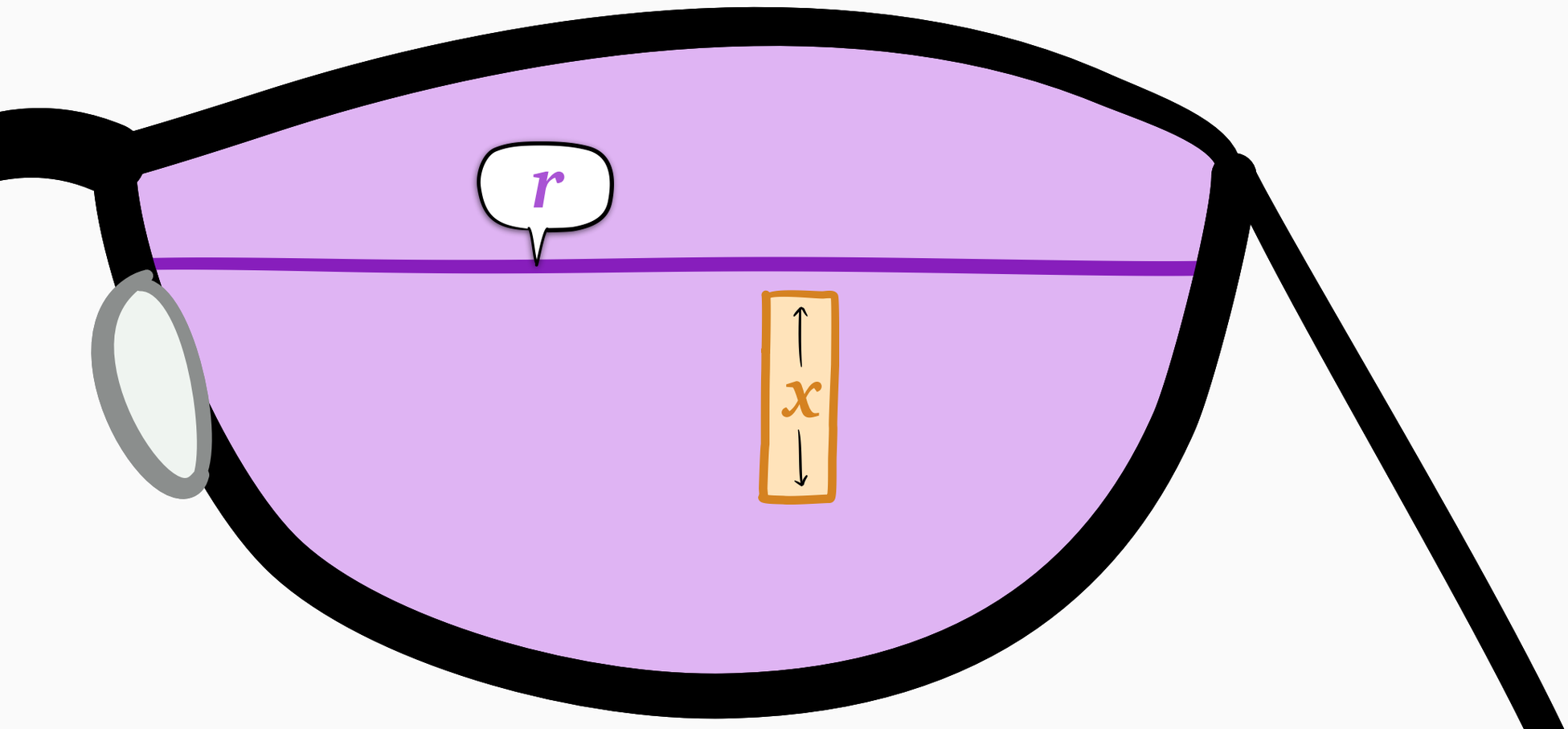


Defining r-work

for SRPT

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$

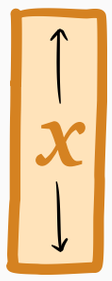


Defining r-work

for SRPT

$W(r)$ = work relevant to **rank r**

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$

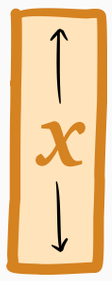


Defining r-work

for SRPT

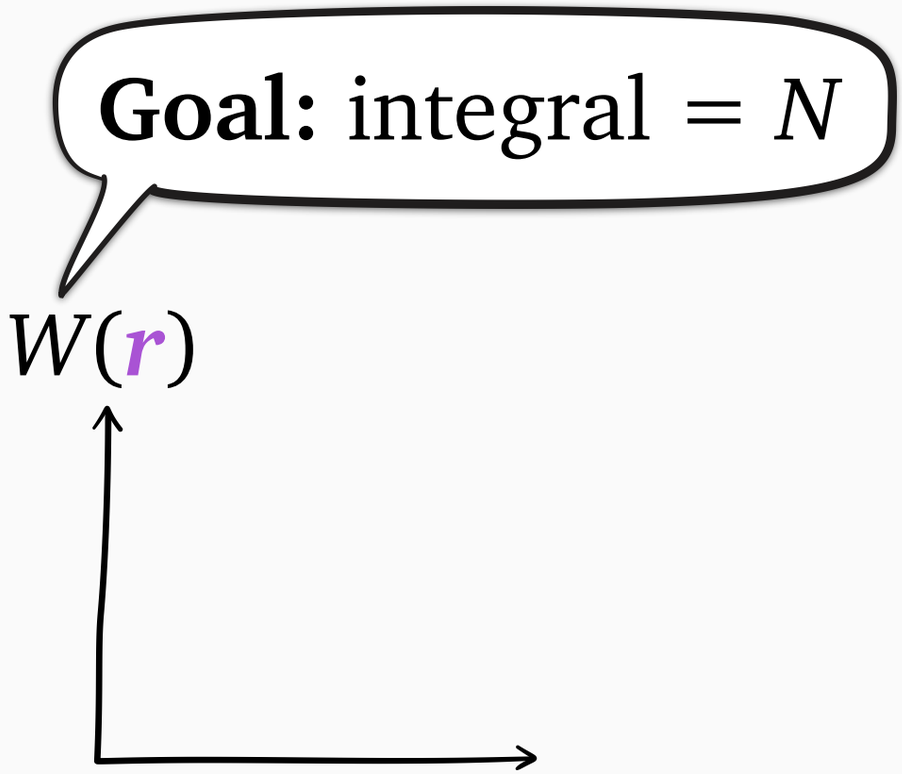
$W(r)$ = work relevant to **rank r**
= total r-work of all jobs

$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$

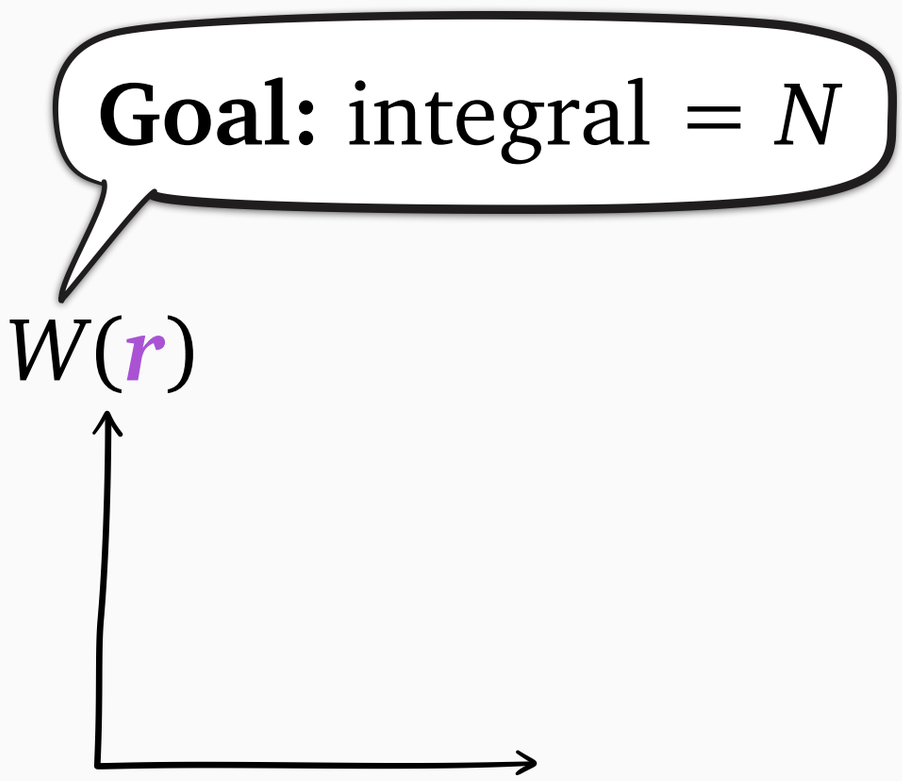


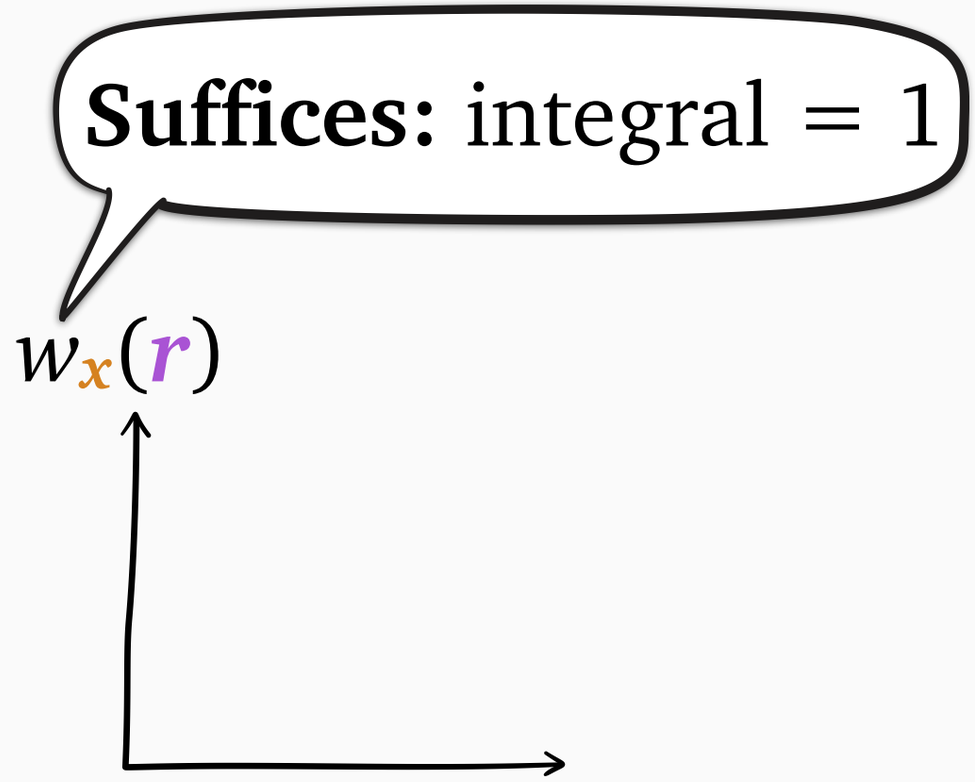
From r-work to number of jobs N

From r-work to number of jobs N



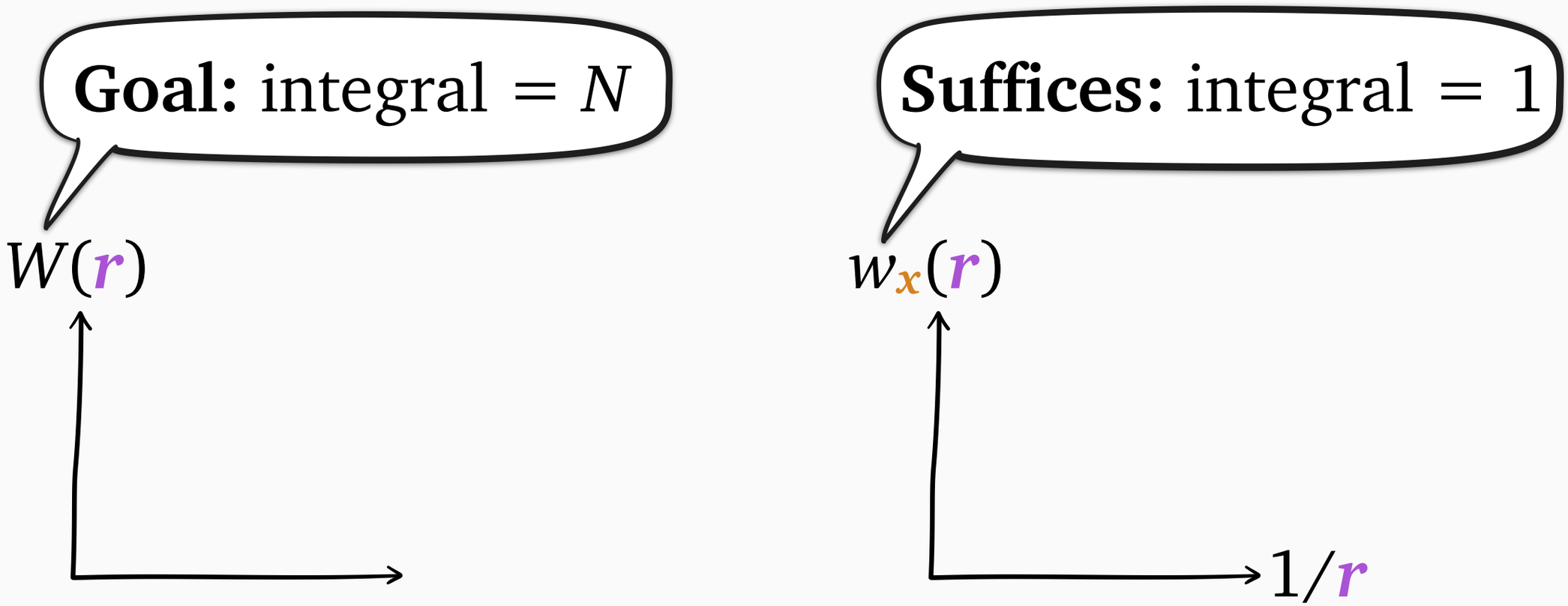
From r-work to number of jobs N





$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

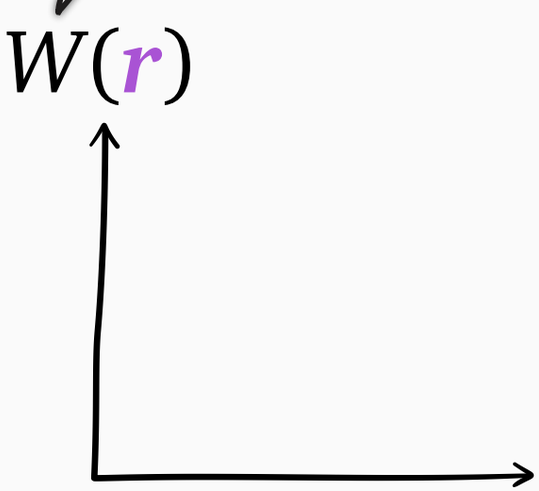
From r-work to number of jobs N



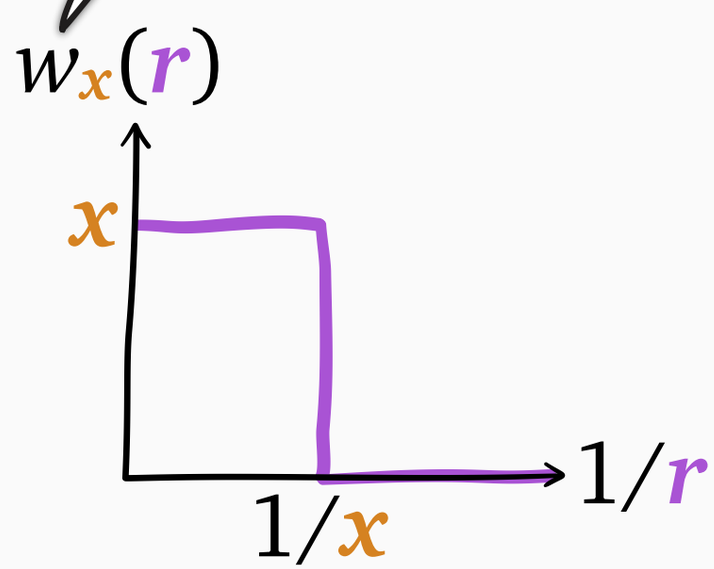
$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

From r-work to number of jobs N

Goal: integral $= N$



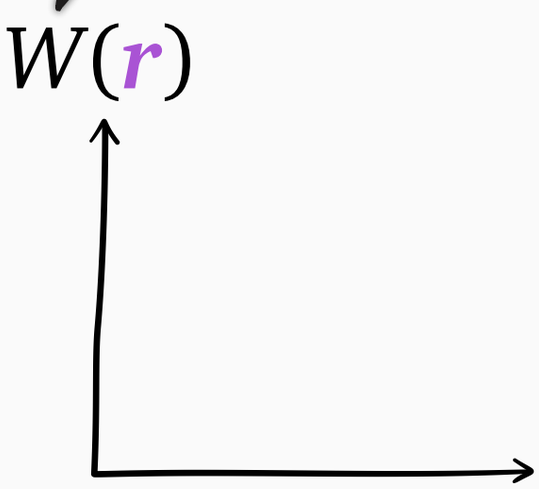
Suffices: integral $= 1$



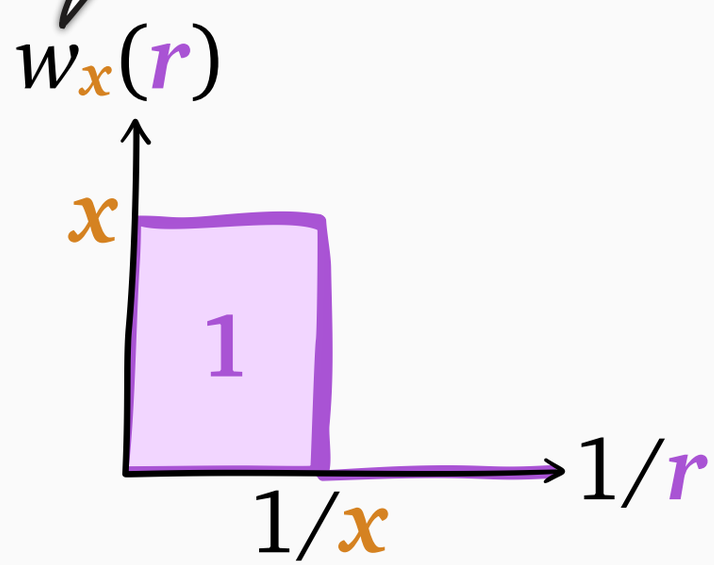
$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

From r-work to number of jobs N

Goal: integral $= N$

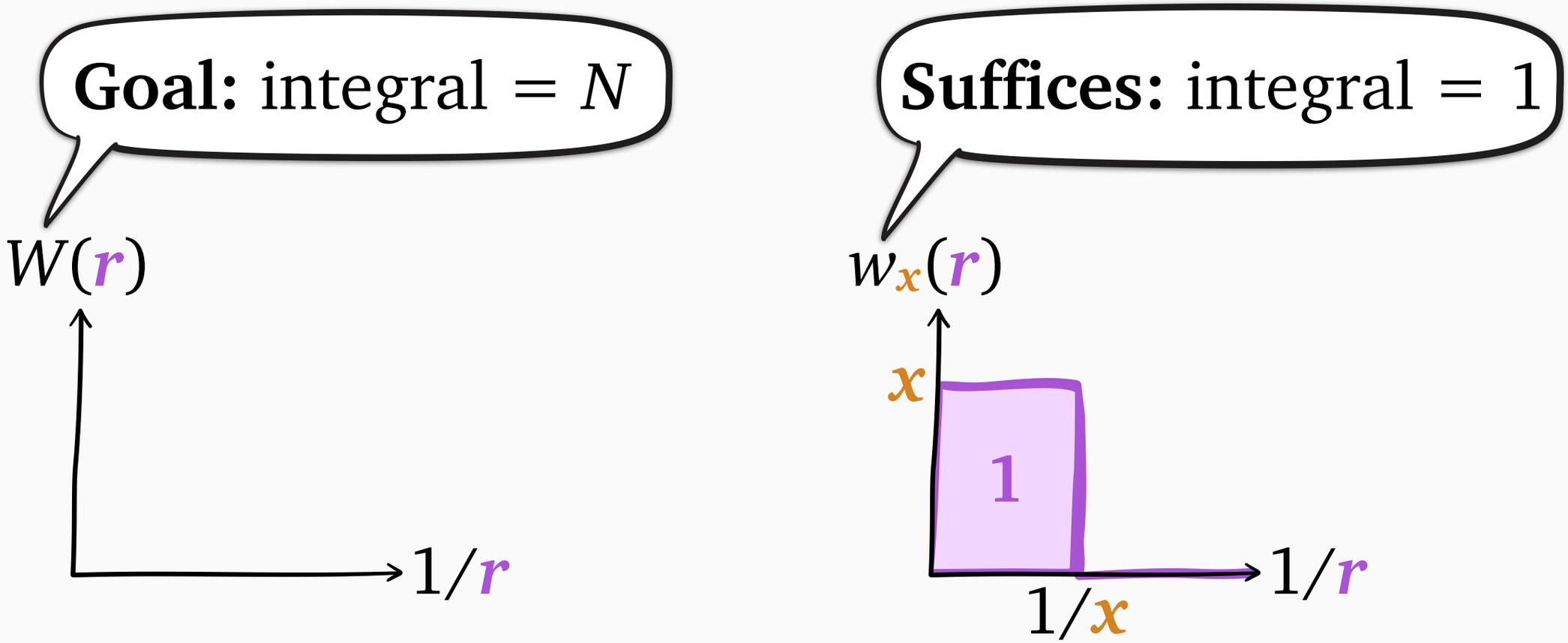


Suffices: integral $= 1$



$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

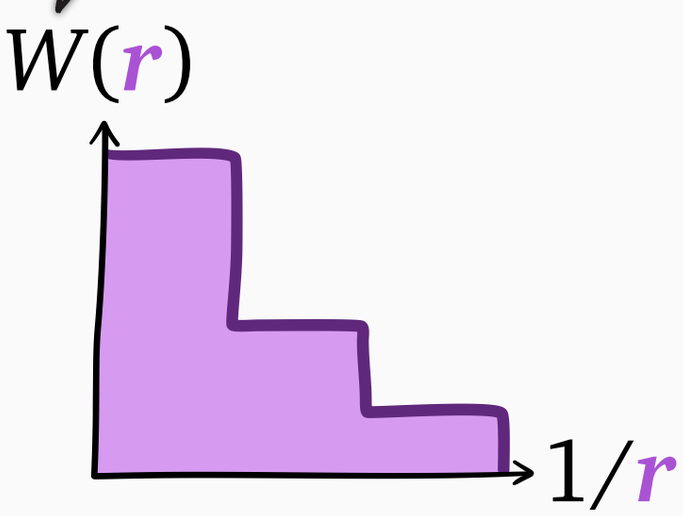
From r-work to number of jobs N



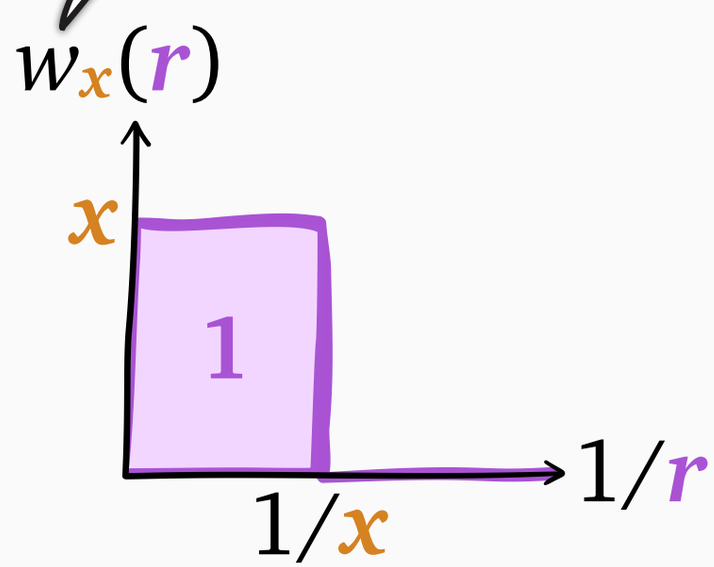
$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

From r-work to number of jobs N

Goal: integral $= N$



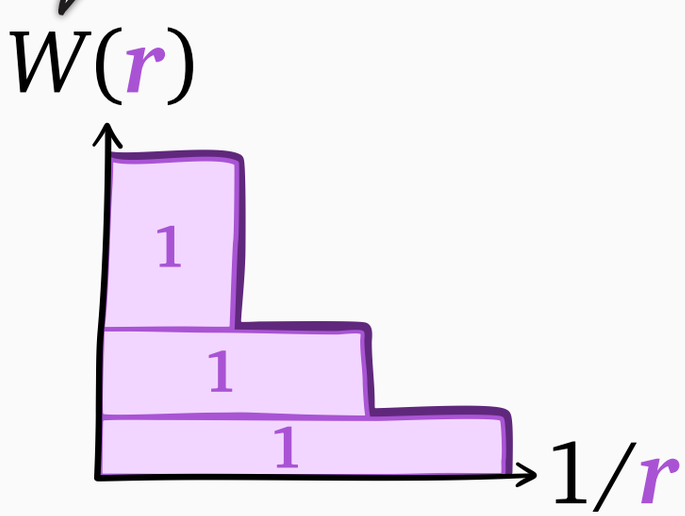
Suffices: integral $= 1$



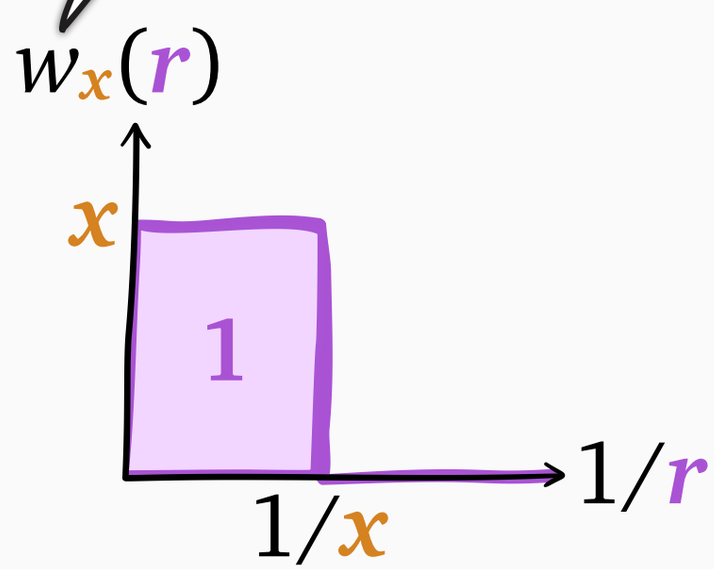
$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

From r-work to number of jobs N

Goal: integral $= N$



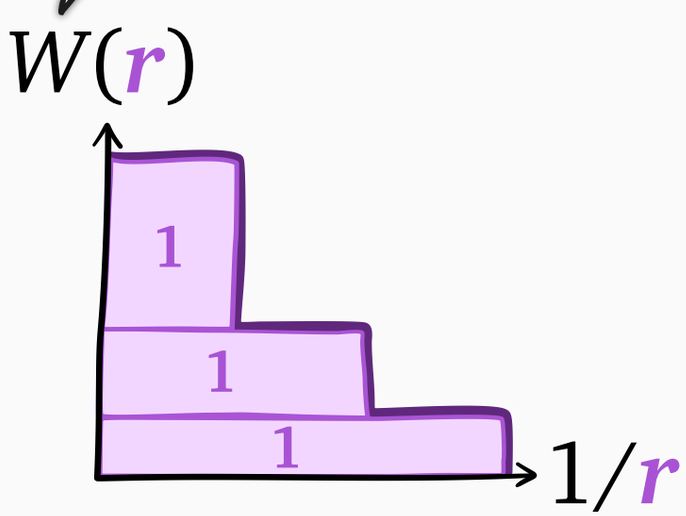
Suffices: integral $= 1$



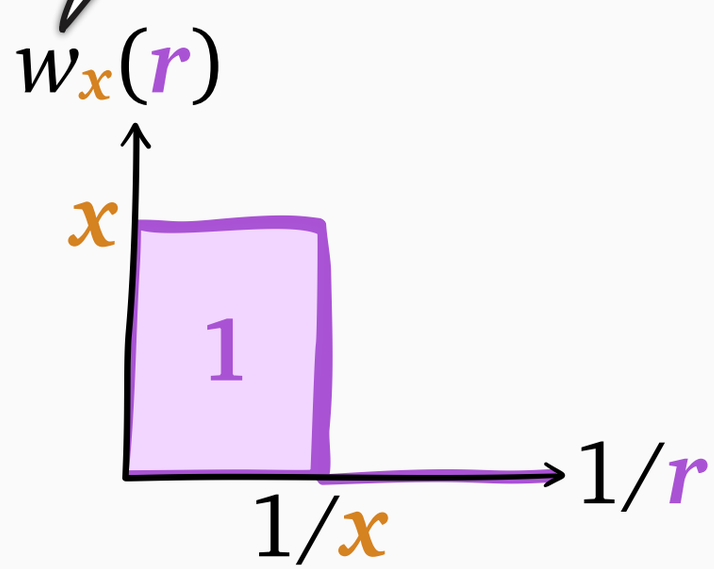
$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

From r-work to number of jobs N

Goal: integral $= N$



Suffices: integral $= 1$



NEW!

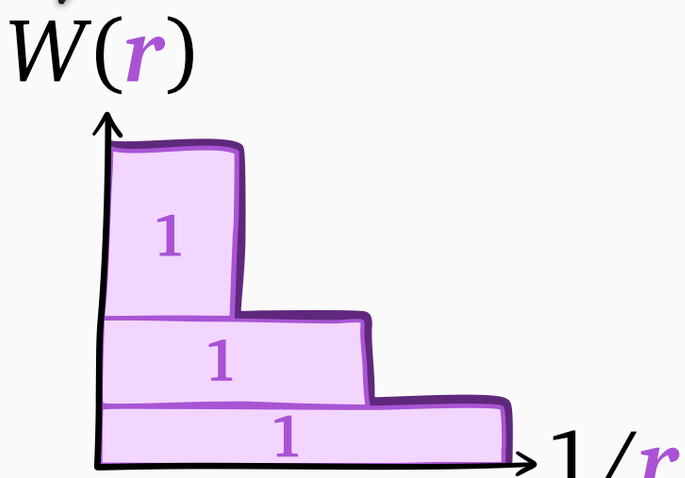
Theorem:

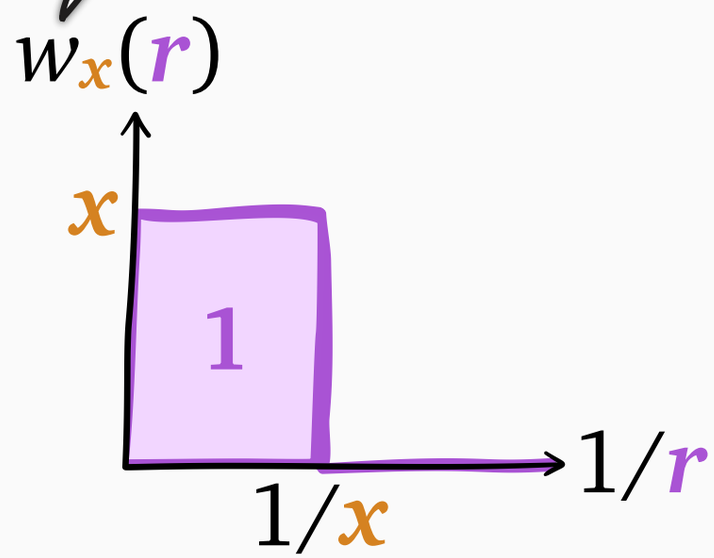
$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

From r-work to number of jobs N

Goal: integral $= N$

Suffices: integral $= 1$





NEW!

WINE

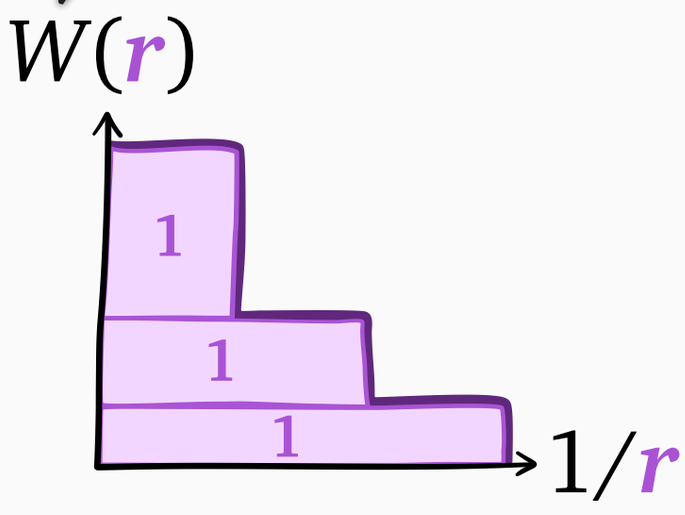
Theorem:

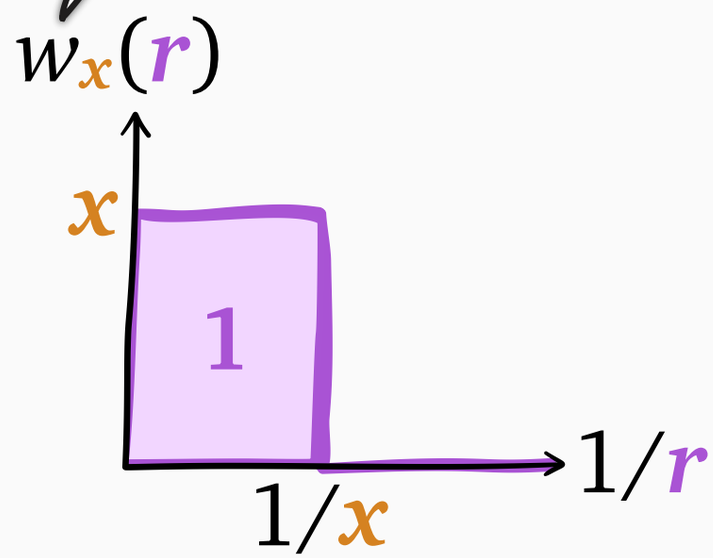
$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

From r-work to number of jobs N

Goal: integral $= N$

Suffices: integral $= 1$





NEW!

Theorem:

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

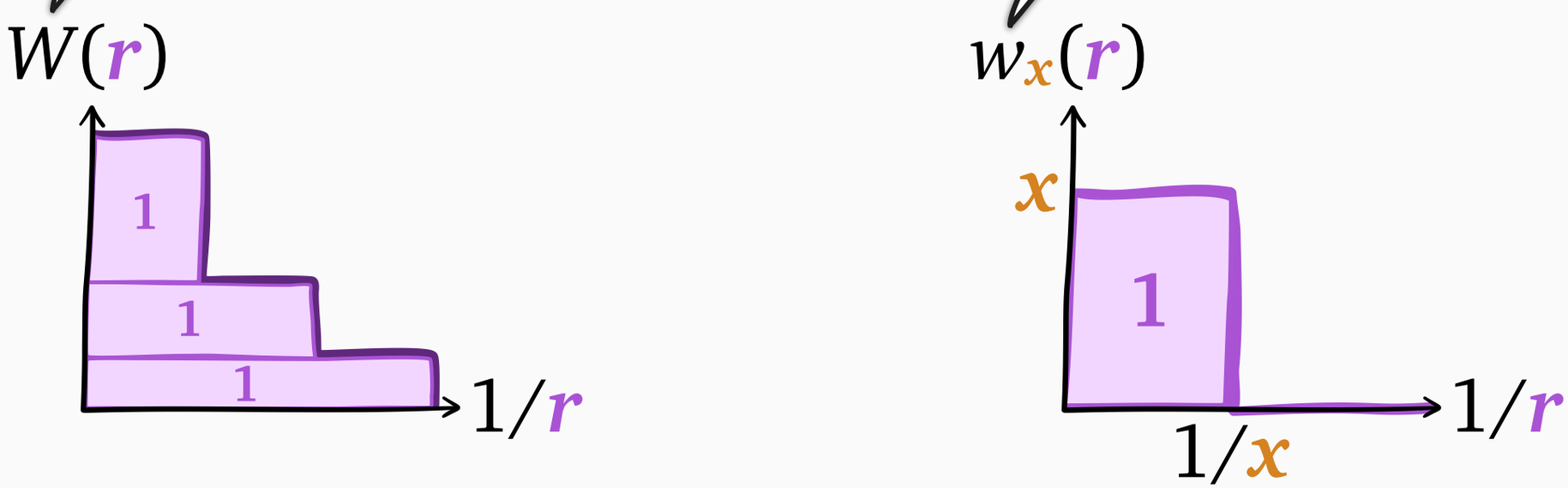
uses **rank** = rem. size

WINE

From r-work to number of jobs N

Goal: integral $= N$

Suffices: integral $= 1$



NEW!

Theorem: under *any* policy,

$$N = \int_0^\infty \frac{W(r)}{r^2} \mathrm{d}r$$

uses **rank** = rem. size

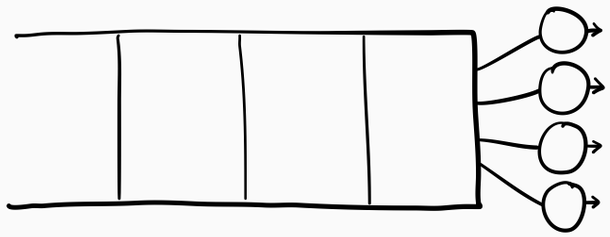
WINE

Impact of WINE

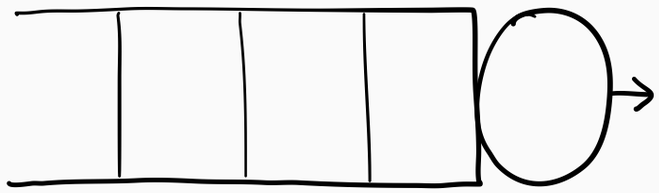
Impact of WINE

multiserver systems

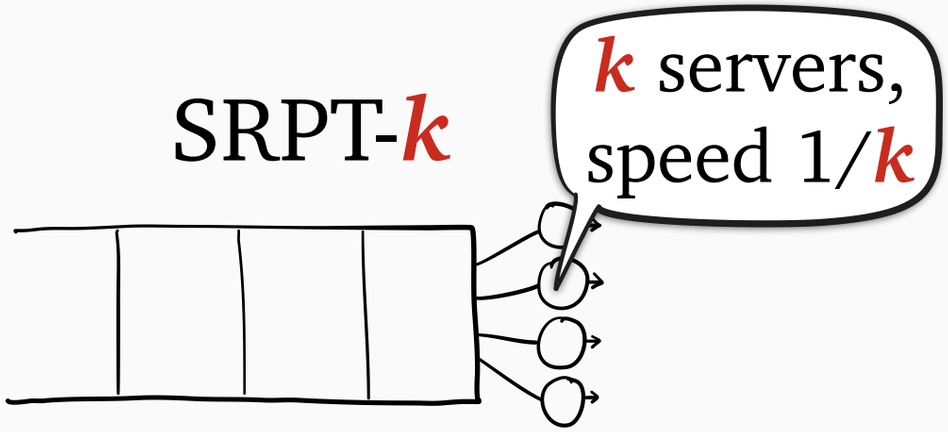
SRPT-k



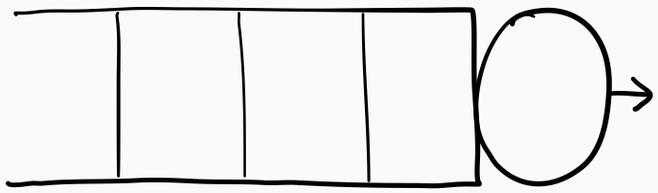
SRPT-1

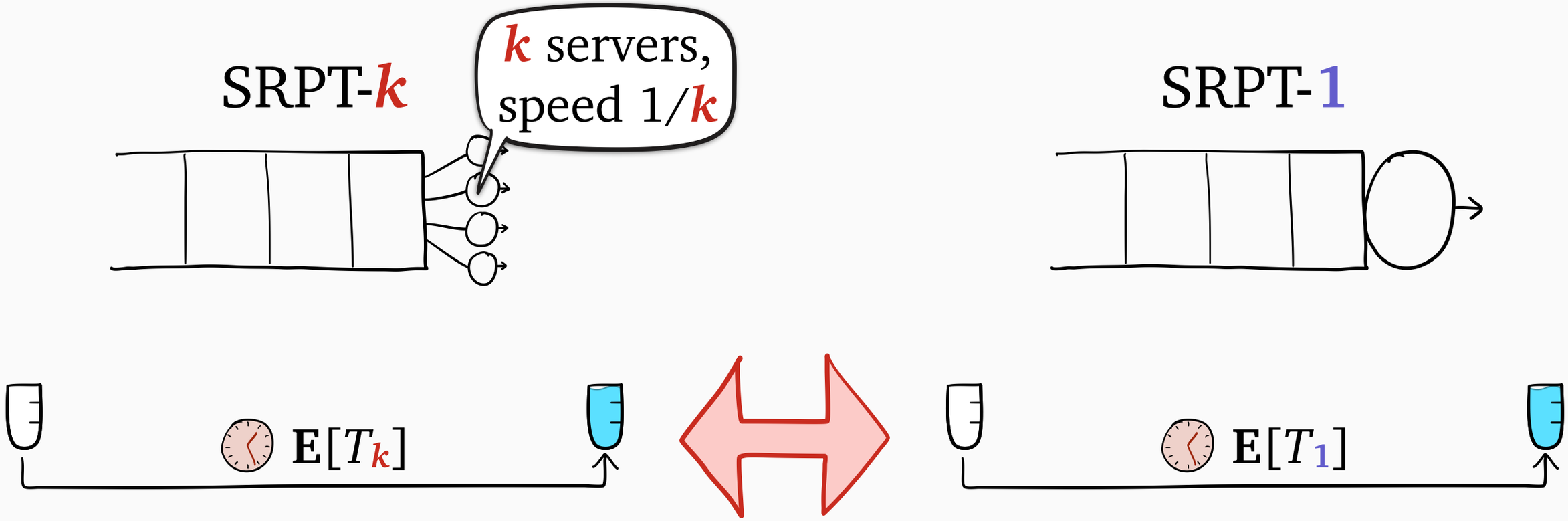


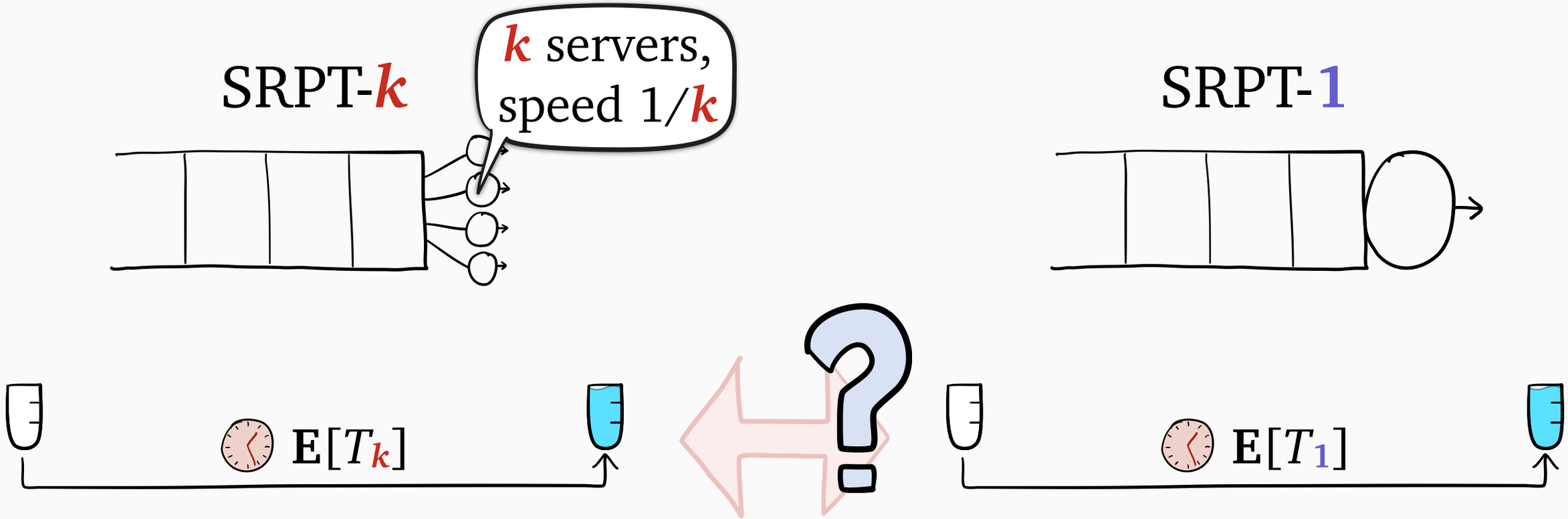
SRPT-k



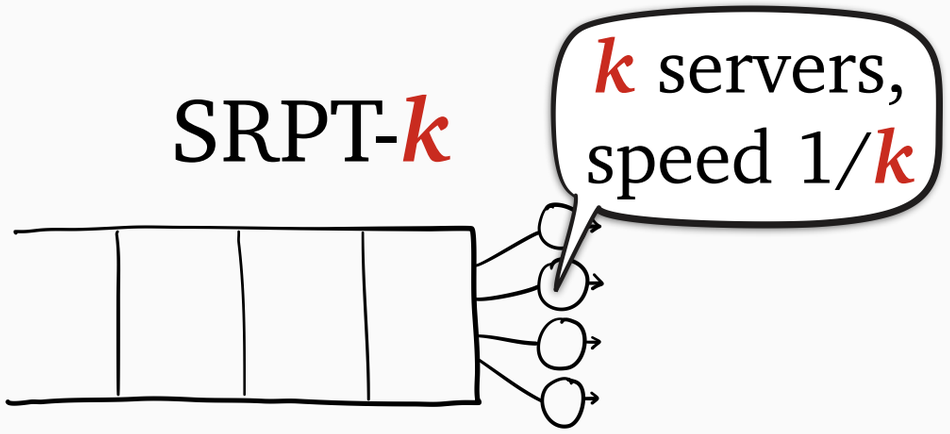
SRPT-1



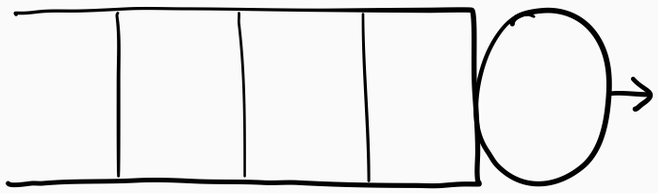

SRPT-k
k servers, speed $1/k$
SRPT-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$


SRPT-k
k servers, speed $1/k$
SRPT-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$

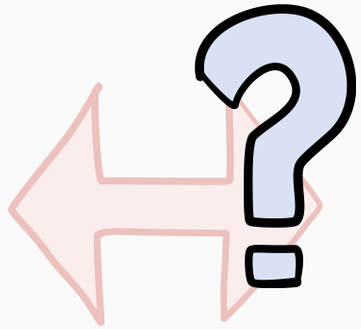
SRPT-k


k servers,
speed $1/k$

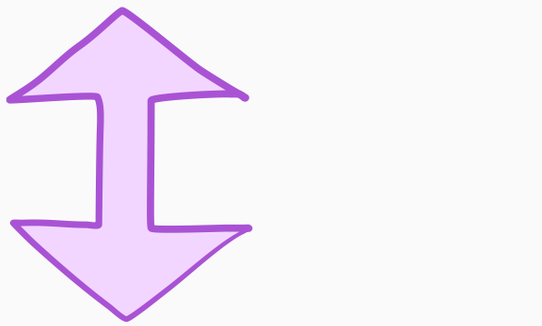
SRPT-1

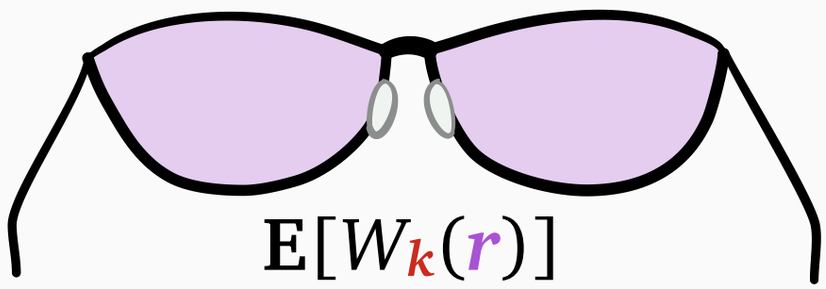


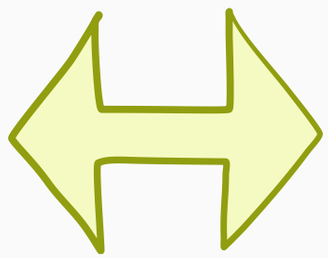

$\mathbf{E}[T_k]$

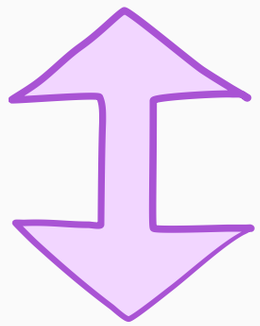


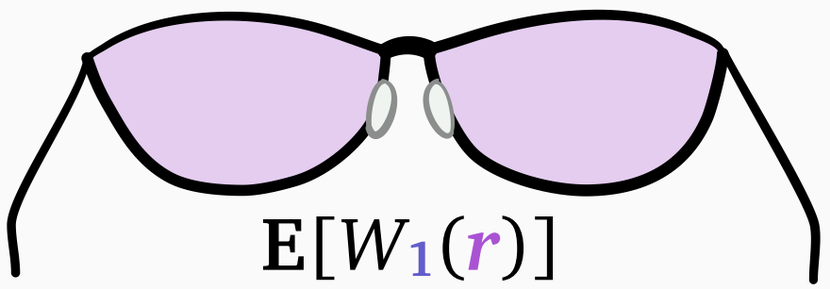

$\mathbf{E}[T_1]$



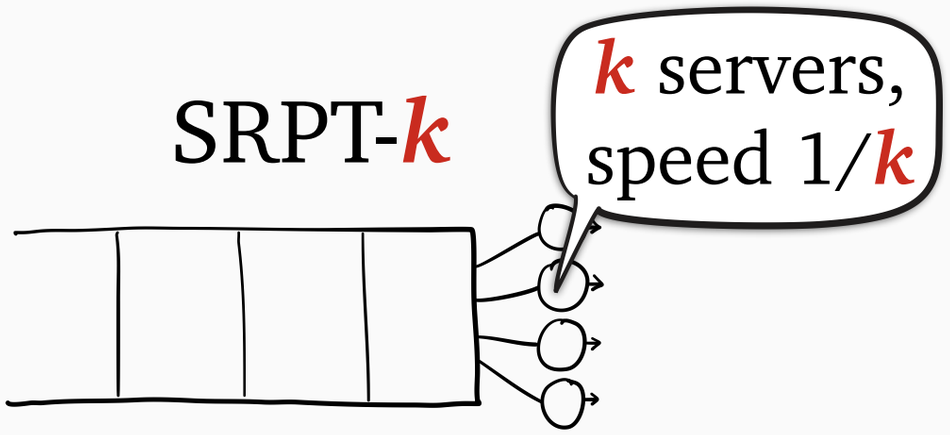

$\mathbf{E}[W_k(r)]$



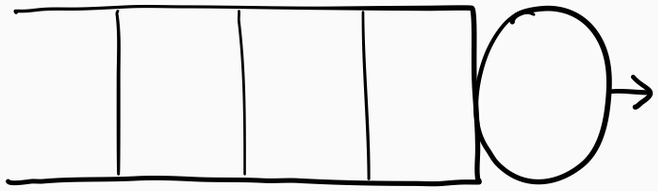



$\mathbf{E}[W_1(r)]$

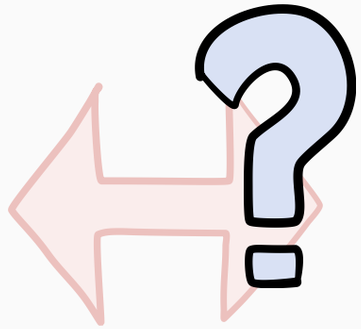
SRPT-k


k servers,
speed $1/k$

SRPT-1

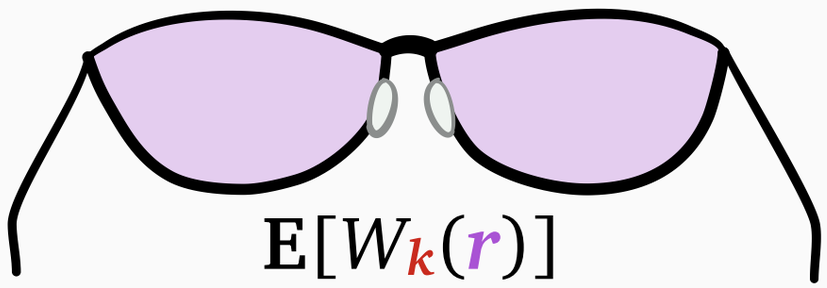


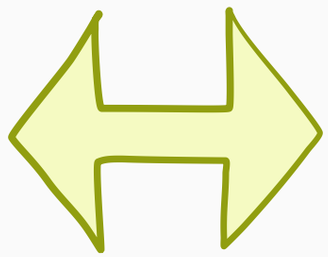

$\mathbf{E}[T_k]$



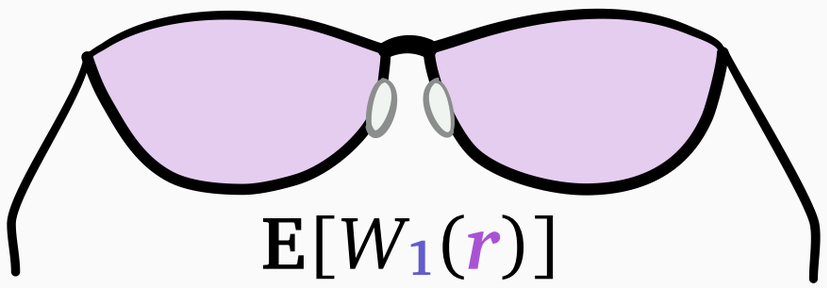

$\mathbf{E}[T_1]$


WINE


$\mathbf{E}[W_k(r)]$




WINE


$\mathbf{E}[W_1(r)]$

SRPT-k

k servers, speed $1/k$

SRPT-1

$\mathbf{E}[T_k]$

$\mathbf{E}[T_1]$

WINE

WINE

$\mathbf{E}[W_k(r)]$

$\mathbf{E}[W_1(r)]$

Lemma: r-work decomposition

SRPT-k

k servers, speed $1/k$

SRPT-1

$\mathbf{E}[T_k]$

$\mathbf{E}[T_1]$

WINE

WINE

$\mathbf{E}[W_k(r)]$

$\mathbf{E}[W_1(r)]$

NEW!

Lemma: r-work decomposition

SRPT-k

k servers, speed $1/k$

SRPT-1

$\mathbf{E}[T_k]$

$\mathbf{E}[T_1]$

WINE

WINE

$\mathbf{E}[W_k(r)]$

$\mathbf{E}[W_1(r)]$

NEW!

Lemma: r-work decomposition

Impact of WINE

multiserver systems

Impact of WINE

multiserver systems

First analysis of

- SRPT-***k***
- **Gittins-*k***

Impact of **WINE**

multiserver systems

noisy size estimates

First analysis of

- SRPT-***k***
- **Gittins-*k***

Impact of **WINE**

multiserver systems

- *First analysis* of
  - SRPT-***k***
  - **Gittins**-***k***

noisy size estimates

compare **noisy-info** ***r***-work
to **perfect-info** ***r***-work

WINE References

[7] Grosof, Scully, and Harchol-Balter (2018). "SRPT for Multiserver Systems." *Perform. Eval.* (PERFORMANCE 2018). **Winner: PERFORMANCE 2018 Best Student Paper Award.**

[8] Grosof, Scully, and Harchol-Balter (2019). "Load Balancing Guardrails: Keeping Your Heavy Traffic on the Road to Low Response Times." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2019). **Winner: SIGMETRICS 2018 Outstanding Student Paper Award.**

[9] Scully, Grosof, and Harchol-Balter (2021). "Optimal Multiserver Scheduling with Unknown Job Sizes in Heavy Traffic." *Perform. Eval.* (PERFORMANCE 2020 issue).

[10] Scully, Grosof, and Harchol-Balter (2020). "The Gittins Policy is Nearly Optimal in the M/G/k under Extremely General Conditions." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2021). **Winner: 2022 INFORMS George Nicholson Student Paper Competition.**

[11] Scully (2021). "Bounding Mean Slowdown in Multiserver Systems." *SIGMETRICS Perform. Eval. Rev.* (MAMA 2021).

[12] Scully, Grosof, and Mitzenmacher (2022). "Uniform Bounds for Scheduling with Job Size Estimates." *13th Innovations in Theoretical Computer Science Conference* (ITCS 2022).

WINE References

[7] Grosof, Scully, and Harchol-Balter (2018). "SRPT for Multiserver Systems." *Perform. Eval.* (PERFORMANCE 2018). **Winner: PERFORMANCE 2018 Best Student Paper Award.**

[8] Grosof, Scully, and Harchol-Balter (2019). "Load Balancing Guardrails: Keeping Your Heavy Traffic on the Road to Low Response Times." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2019). **Winner: SIGMETRICS 2018 Outstanding Student Paper Award.**

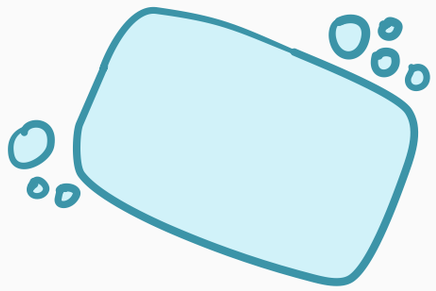
[9] Scully, Grosof, and Harchol-Balter (2021). "Optimal Multiserver Scheduling with Unknown Job Sizes in Heavy Traffic." *Perform. Eval.* (PERFORMANCE 2020 issue).

[10] Scully, Grosof, and Harchol-Balter (2020). "The Gittins Policy is Nearly Optimal in the M/G/k under Extremely General Conditions." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2021). **Winner: 2022 INFORMS George Nicholson Student Paper Competition.**

[11] Scully (2021). "Bounding Mean Slowdown in Multiserver Systems." *SIGMETRICS Perform. Eval. Rev.* (MAMA 2021).

[12] Scully, Grosof, and Mitzenmacher (2022). "Uniform Bounds for Scheduling with Job Size Estimates." *13th Innovations in Theoretical Computer Science Conference* (ITCS 2022).

Outline: two new tools



SOAP

new unifying theory of single-server scheduling

→ unknown sizes

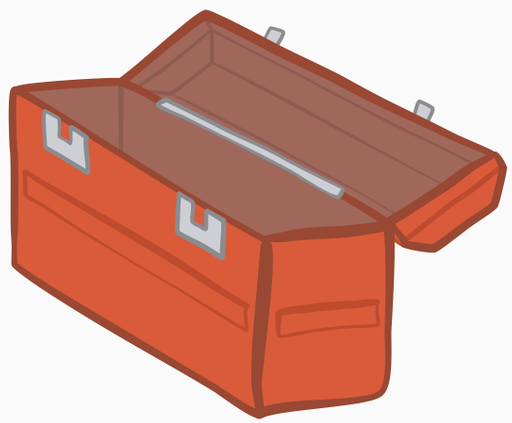
→ preemption limitations

WINE

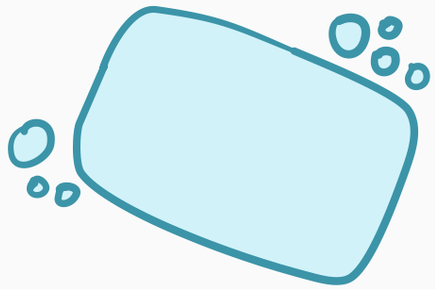
new queueing identity to complement Little's Law

→ multiserver systems

→ noisy size estimates



Outline: two new tools



SOAP

new unifying theory of single-server scheduling

→ unknown sizes

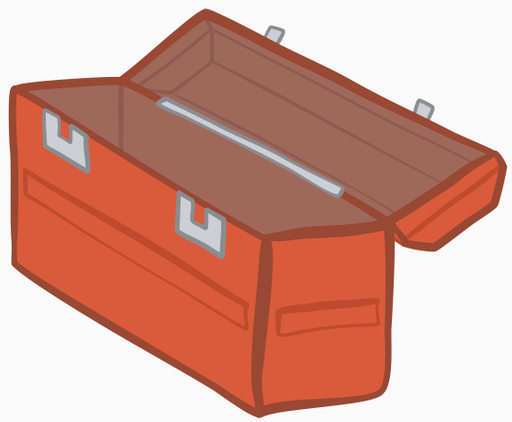
→ preemption limitations

WINE

new queueing identity to complement Little's Law

→ multiserver systems

→ noisy size estimates



SOAP &WINE

SOAP &WINE

Thesis statement?

Thesis statement?

Yes, queueing theory can!

Past:
Analyze policies one-by-one

Past:

Analyze policies one-by-one

Present:

Analyze all **rank**-function-based policies at once

MLPS, SMART

Past:

Analyze policies one-by-one

Present:

Analyze all **rank**-function-based policies at once

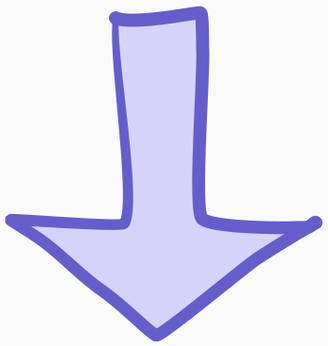
MLPS, SMART

Past:

Analyze policies one-by-one

Present:

Analyze all **rank**-function-based policies at once



Future:

Other policy classes?

Past:

Scheduling in M/G/1, only FCFS in M/G/k

Past:

Scheduling in M/G/1, only FCFS in M/G/k

Present:

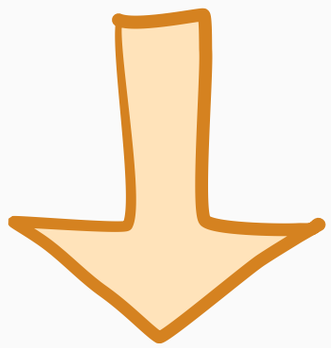
Scheduling in M/G/k

Past:

Scheduling in M/G/1, only FCFS in M/G/k

Present:

Scheduling in M/G/k



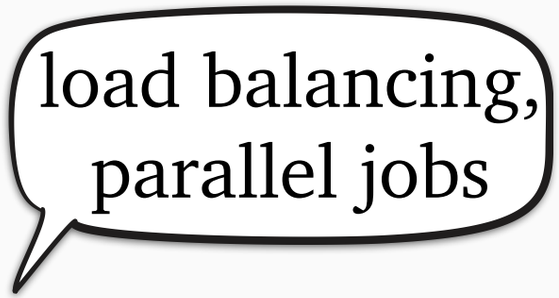
Future:

Scheduling in complex multiserver architectures?

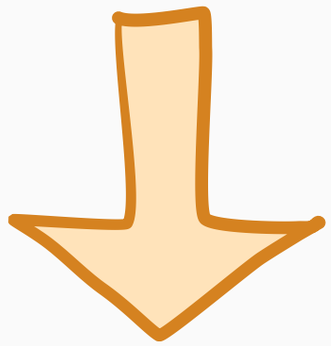
Past:

Scheduling in M/G/1, only FCFS in M/G/k

Present:



Scheduling in M/G/k



Future:

Scheduling in complex multiserver architectures?

Past:

Gittins for specific uncertainty models, one-by-one

Past:

Gittins for specific uncertainty models, one-by-one

Present:

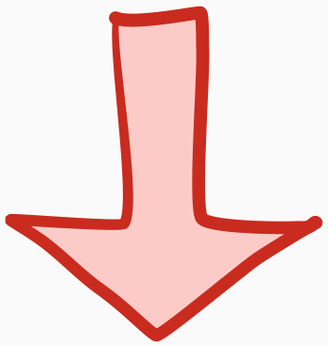
Gittins for any service-based uncertainty model

Past:

Gittins for specific uncertainty models, one-by-one

Present:

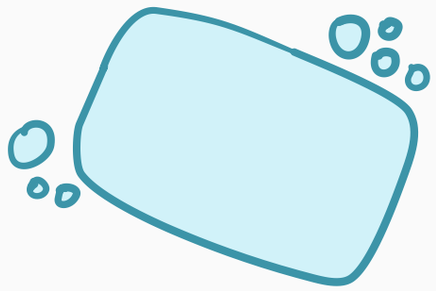
Gittins for any service-based uncertainty model



Future:

Underspecified models? “Restless” models?

Yes, queueing theory can!



SOAP

new unifying theory of single-server scheduling

→ unknown sizes

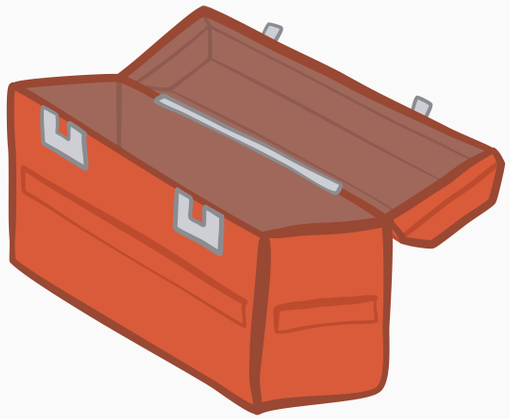
→ preemption limitations

WINE

new queueing identity to complement Little's Law

→ multiserver systems

→ noisy size estimates



SOAP def. and non-examples

Preemption checkpoints

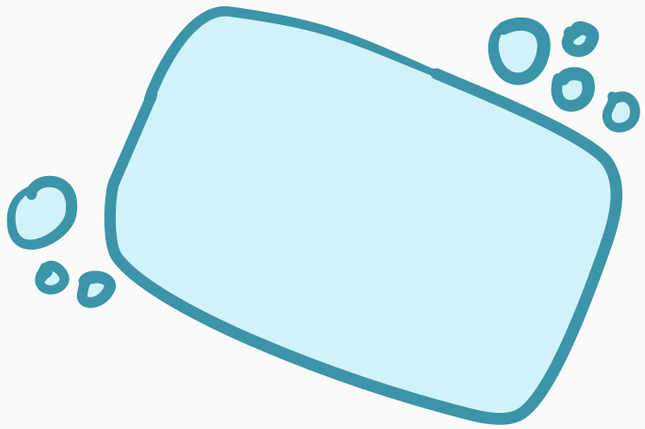
Limited priority levels

Tail decay: heavy vs. light

***r*-work decomposition**

SRPT-***k*** and **Gittins-*k*** $\mathbf{E}[T]$

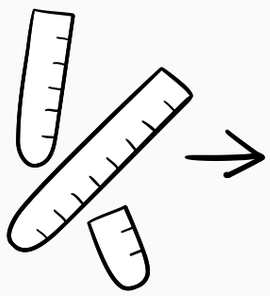
back



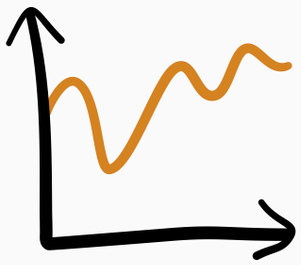
SOAP

Schedule **O**rdered by **A**ge-based **P**riority

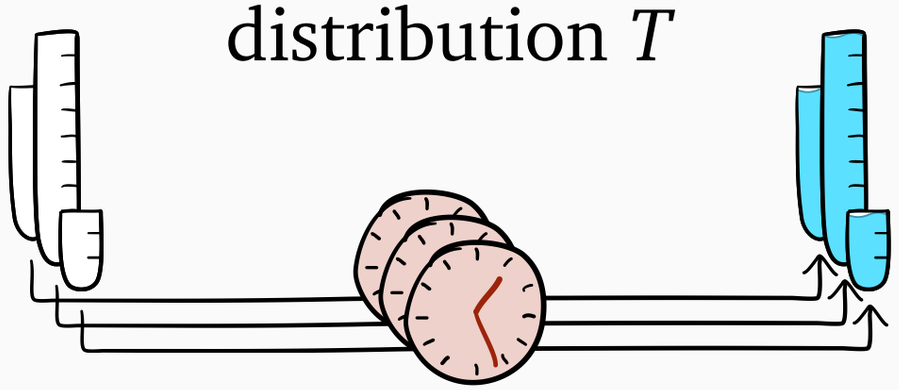
stochastic arrival process λ, S



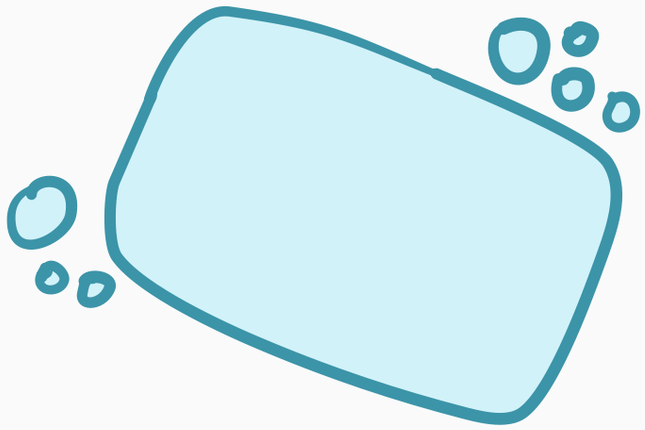
rank function



response time distribution T



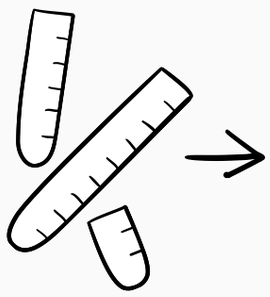
back



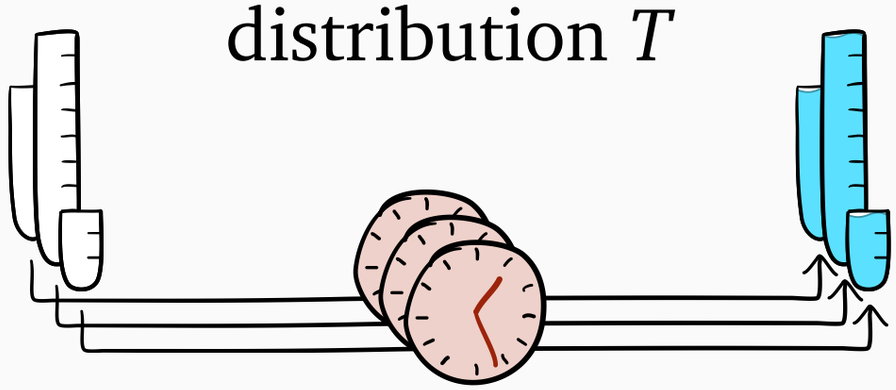
SOAP

Schedule **O**rdered by **A**ge-based **P**riority

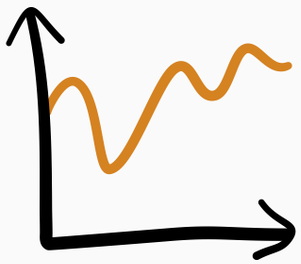
stochastic arrival process λ, S



response time distribution T



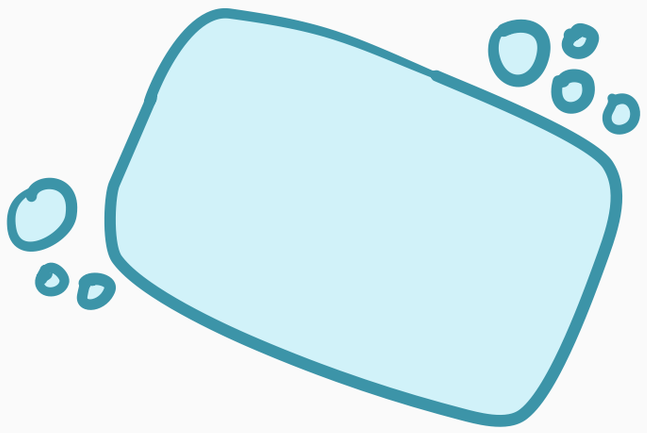
rank function



(label, **age**) ↦ **rank**

where labels are static

back

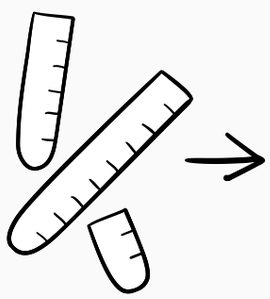


SOAP

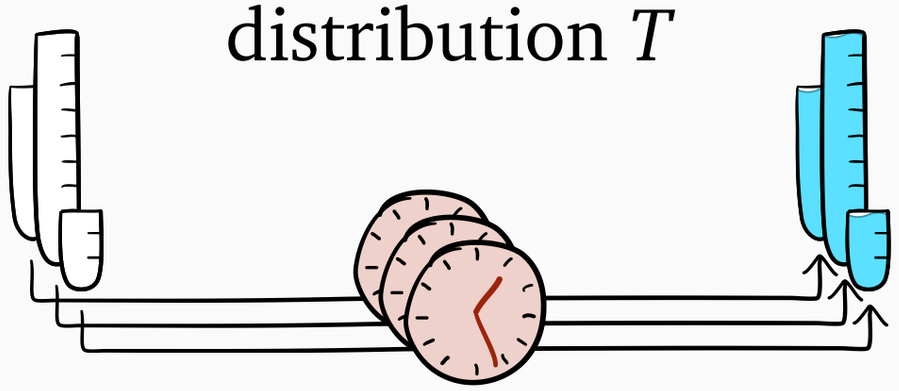
Schedule **O**rdered by **A**ge-based **P**riority

label dist. L,
(*L*, *S*) pairs i.i.d.

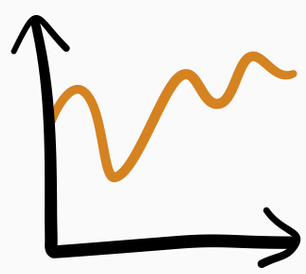
stochastic arrival process λ, S



response time distribution T



rank function



(label, **age**) ↦ **rank**

where labels are static

back

What's *not* a **SOAP** policy?

back

What's *not* a **SOAP** policy?

Scheduling not based on each job having a **rank**

back

What's *not* a **SOAP** policy?

Scheduling not based on each job having a **rank**

Nudge: can't use arrival sequence adjacency

back

What's *not* a **SOAP** policy?

Scheduling not based on each job having a **rank**

Nudge: can't use arrival sequence adjacency

Job's **rank** changes while it's not in service

back

What's *not* a **SOAP** policy?

Scheduling not based on each job having a **rank**

Nudge: can't use arrival sequence adjacency

Job's **rank** changes while it's not in service

*EDF: can't have **rank** = time until deadline*

back

What's *not* a **SOAP** policy?

Scheduling not based on each job having a **rank**

Nudge: can't use arrival sequence adjacency

Job's **rank** changes while it's not in service

*EDF: can't have **rank** = time until deadline*

Job's **rank** depends on a non-i.i.d. label

back

What's *not* a **SOAP** policy?

Scheduling not based on each job having a **rank**

Nudge: can't use arrival sequence adjacency

Job's **rank** changes while it's not in service

*EDF: can't have **rank** = time until deadline*

Job's **rank** depends on a non-i.i.d. label

*EDF: can't have **rank** = deadline*

back

What's *not* a **SOAP** policy?

Scheduling not based on each job having a **rank**

Nudge: can't use arrival sequence adjacency

Job's **rank** changes while it's not in service

*EDF: can't have **rank** = time until deadline*

Job's **rank** depends on a non-i.i.d. label

*EDF: can't have **rank** = deadline*

Tiebreaking not FCFS (or LCFS)

back

What's *not* a **SOAP** policy?

Scheduling not based on each job having a **rank**

Nudge: can't use arrival sequence adjacency

Job's **rank** changes while it's not in service

*EDF: can't have **rank** = time until deadline*

Job's **rank** depends on a non-i.i.d. label

*EDF: can't have **rank** = deadline*

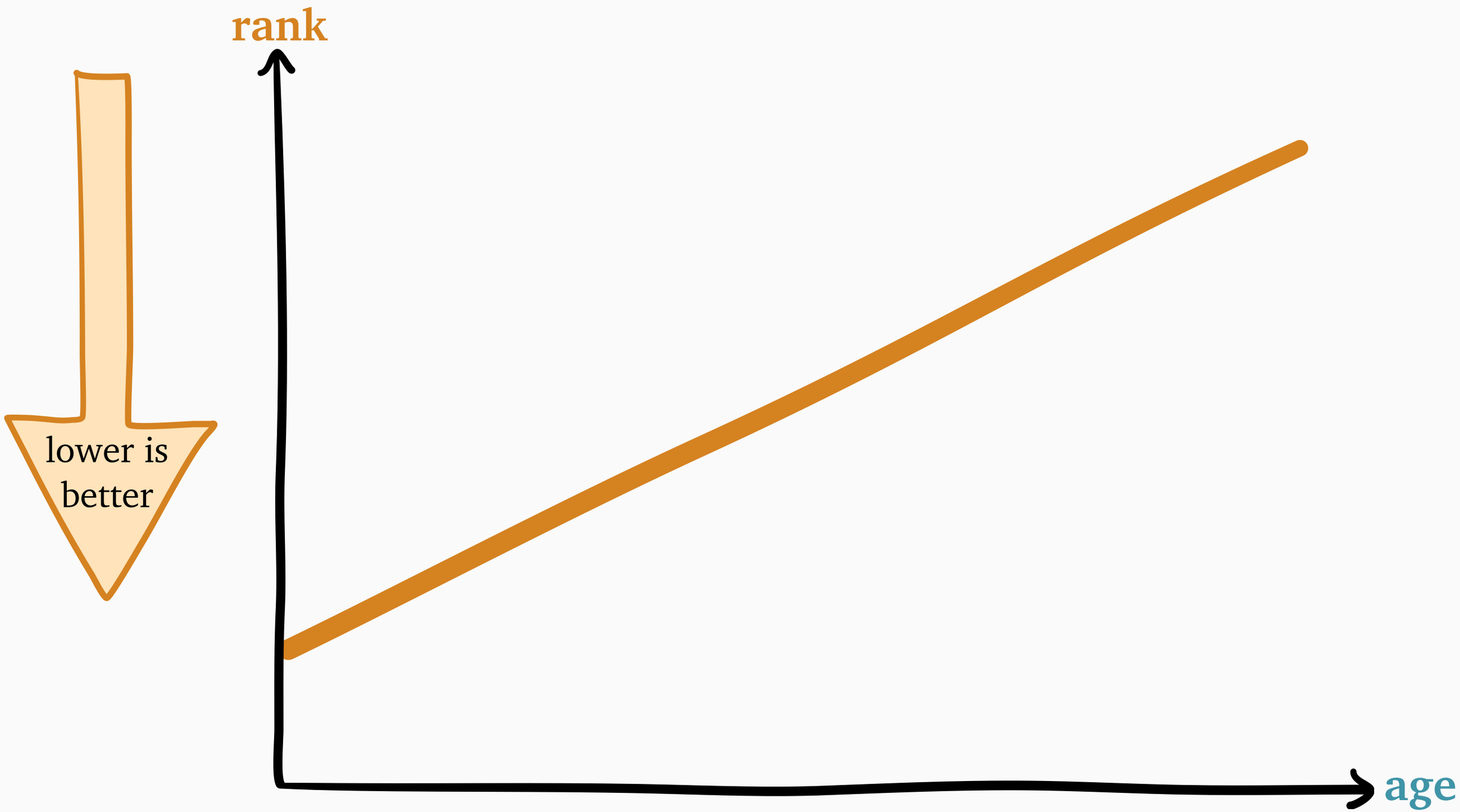
Tiebreaking not FCFS (or LCFS)

PS: can't break ties by sharing (or randomizing)

back

Preemption checkpoints

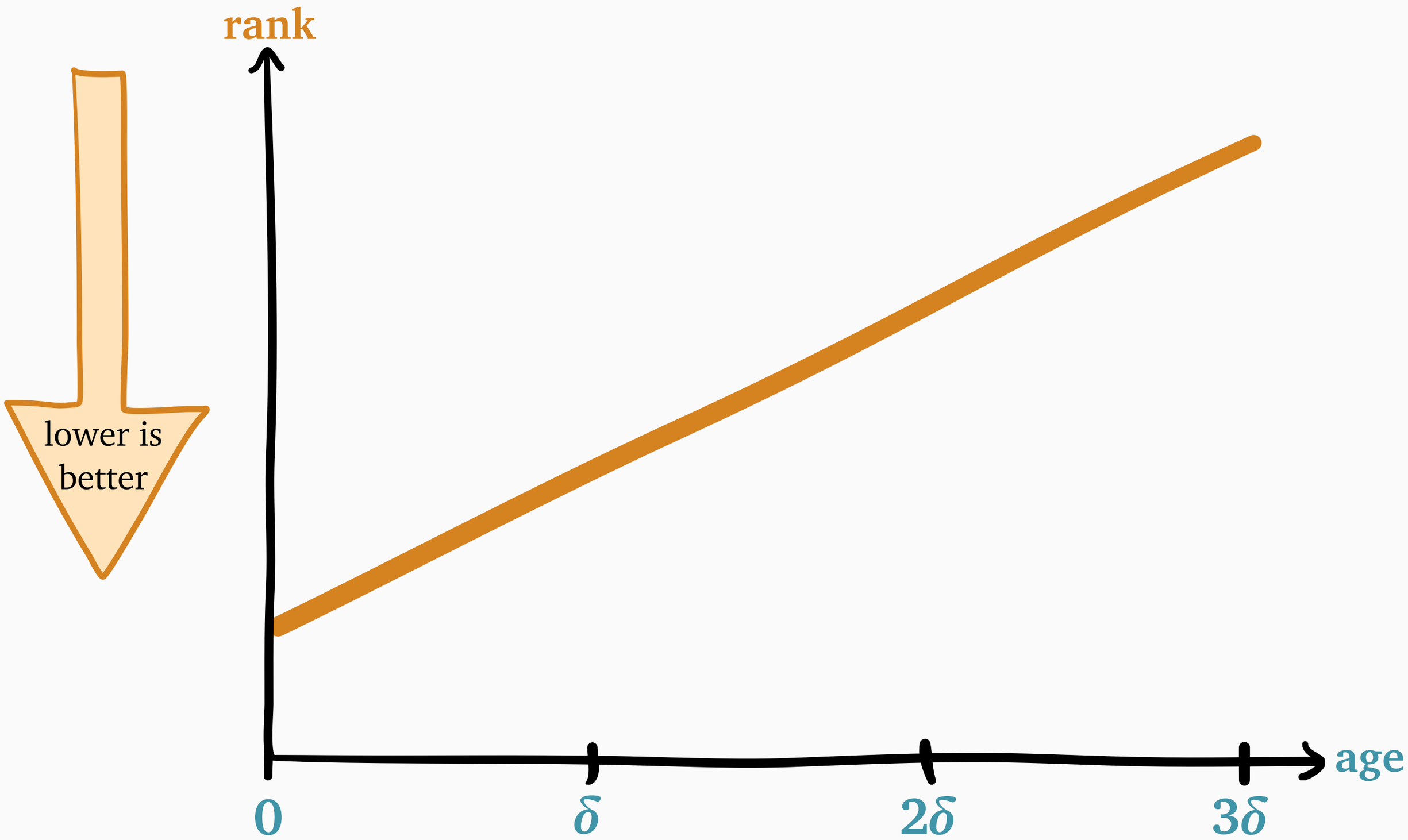
Can only preempt only at *checkpoint* ***ages***



back

Preemption checkpoints

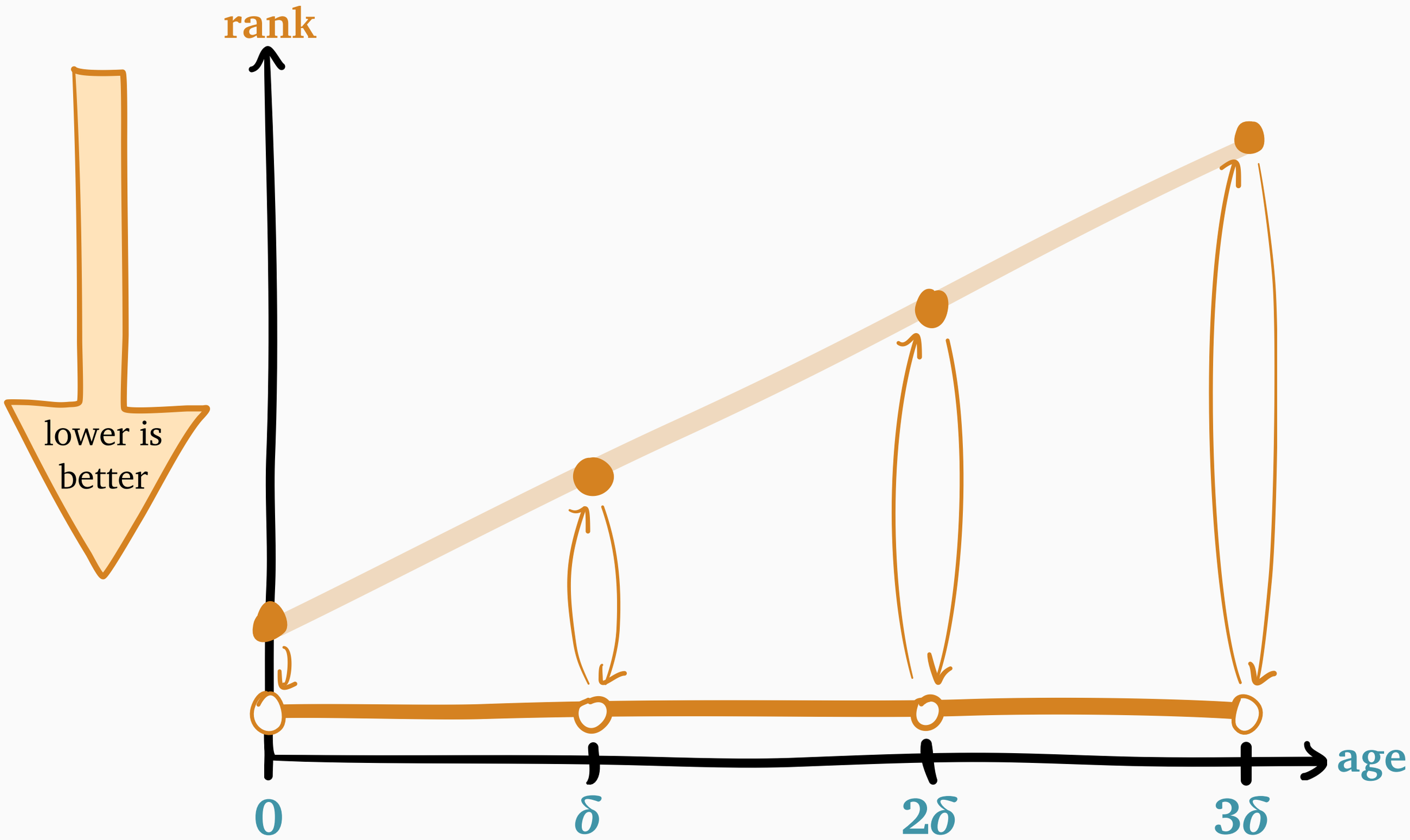
Can only preempt only at *checkpoint* ***ages***



back

Preemption checkpoints

Can only preempt only at *checkpoint* ***ages***



back

Checkpoint frequency?

Suppose each checkpoint incurs an overhead γ

back

Checkpoint frequency?

Suppose each checkpoint incurs an overhead γ

What is the optimal gap δ between checkpoints?

back

Checkpoint frequency?

Suppose each checkpoint incurs an overhead γ

What is the optimal gap δ between checkpoints?

optimal packet size?

back

Checkpoint frequency?

Suppose each checkpoint incurs an overhead γ

What is the optimal gap δ between checkpoints?

- large δ: less overhead

optimal packet size?

back

Checkpoint frequency?

Suppose each checkpoint incurs an overhead γ

What is the optimal gap δ between checkpoints?

- large δ: less overhead
- small δ: more precise scheduling

optimal packet size?

back

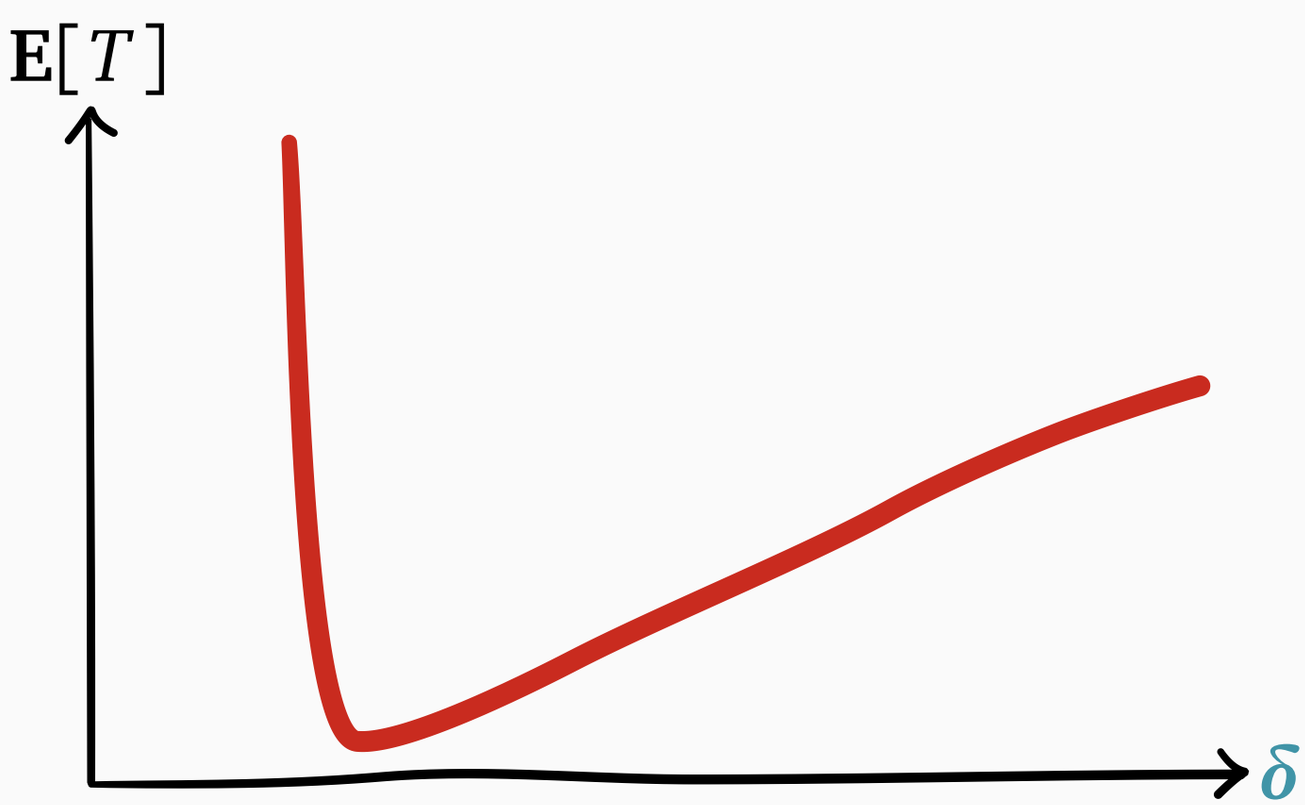
Checkpoint frequency?

Suppose each checkpoint incurs an overhead γ

What is the optimal gap δ between checkpoints?

- large δ: less overhead
- small δ: more precise scheduling

optimal packet size?



back

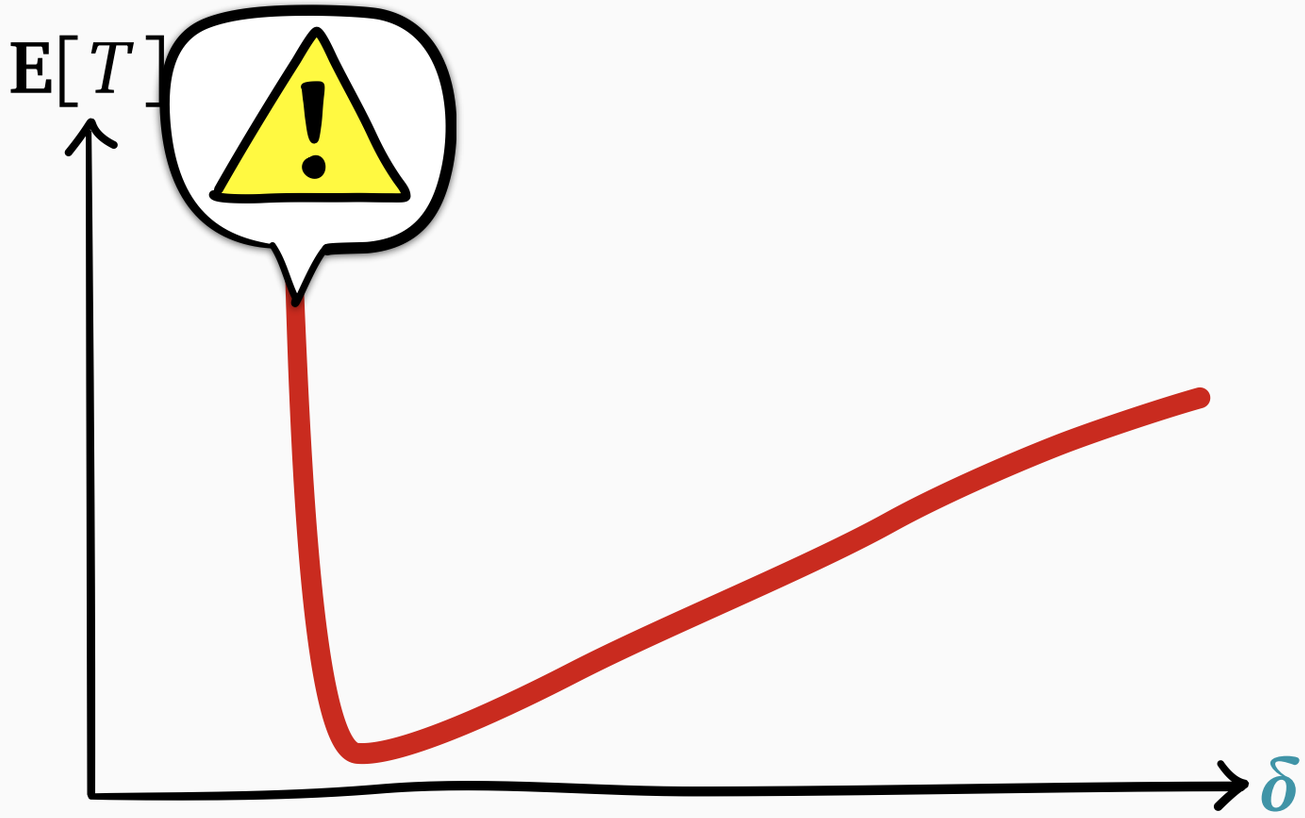
Checkpoint frequency?

Suppose each checkpoint incurs an overhead γ

What is the optimal gap δ between checkpoints?

- large δ: less overhead
- small δ: more precise scheduling

optimal packet size?



back

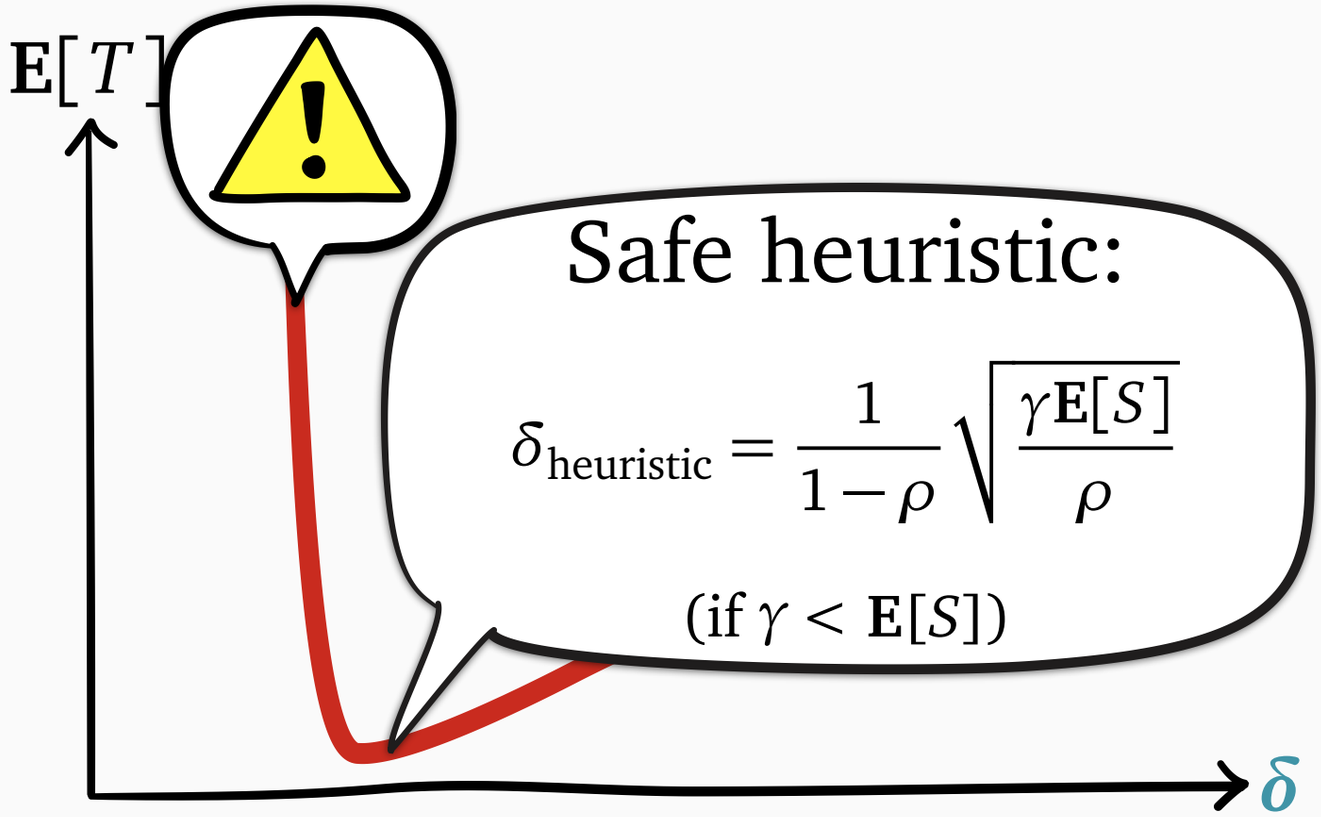
Checkpoint frequency?

Suppose each checkpoint incurs an overhead γ

What is the optimal gap δ between checkpoints?

- large δ: less overhead
- small δ: more precise scheduling

optimal packet size?

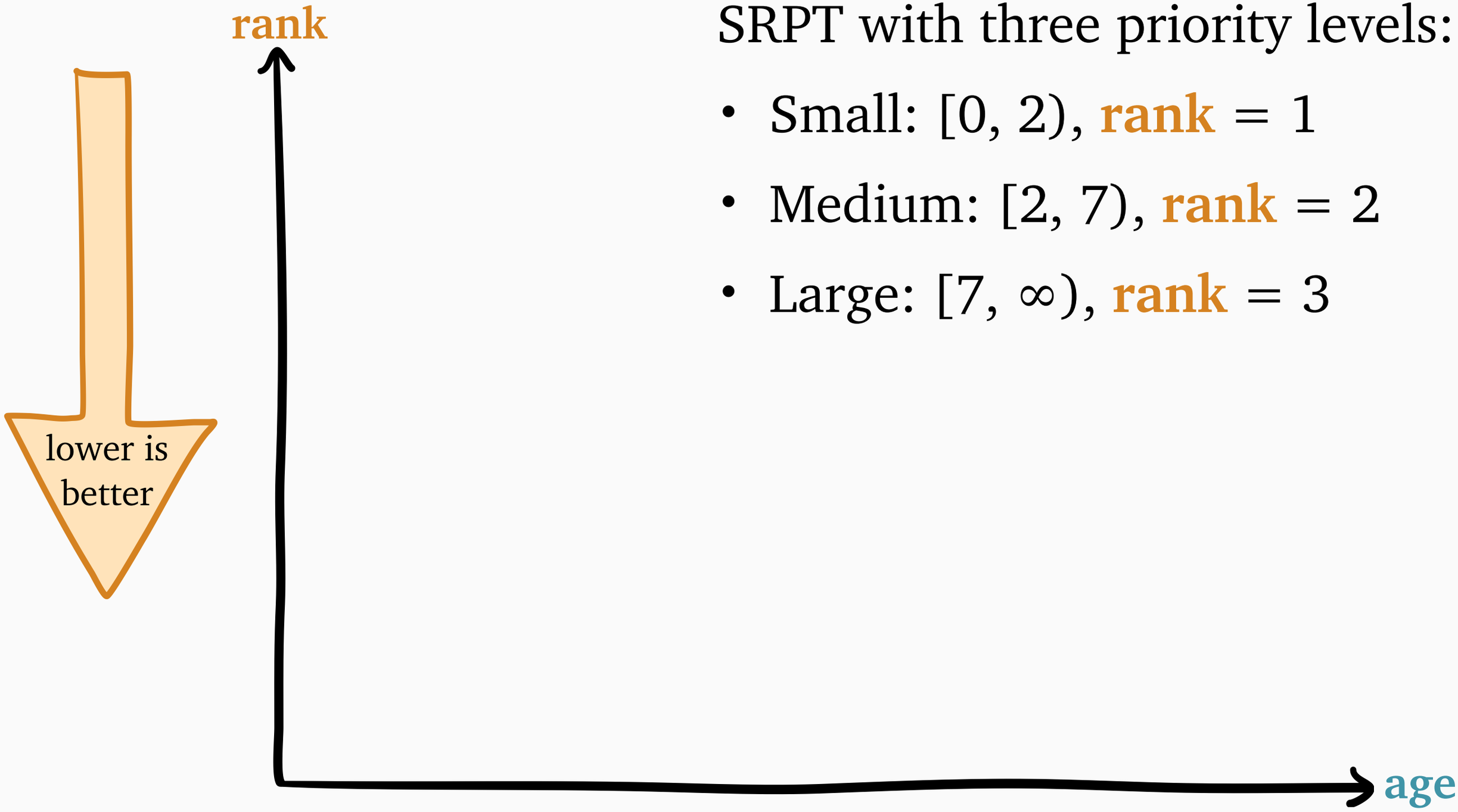


back

Limited priority levels

back

Limited priority levels

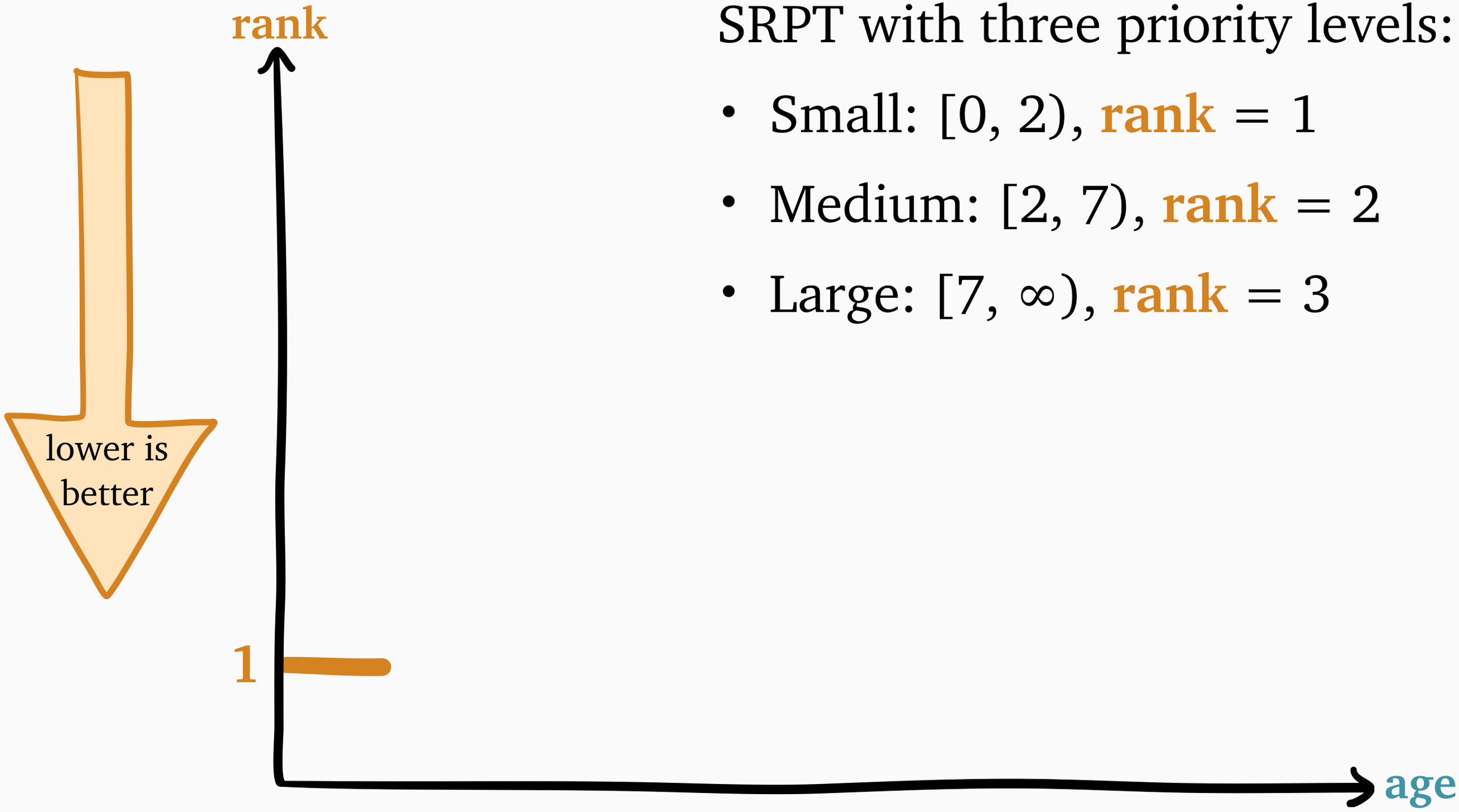


SRPT with three priority levels:

- Small: $[0, 2)$, **rank** $= 1$
- Medium: $[2, 7)$, **rank** $= 2$
- Large: $[7, \infty)$, **rank** $= 3$

back

Limited priority levels

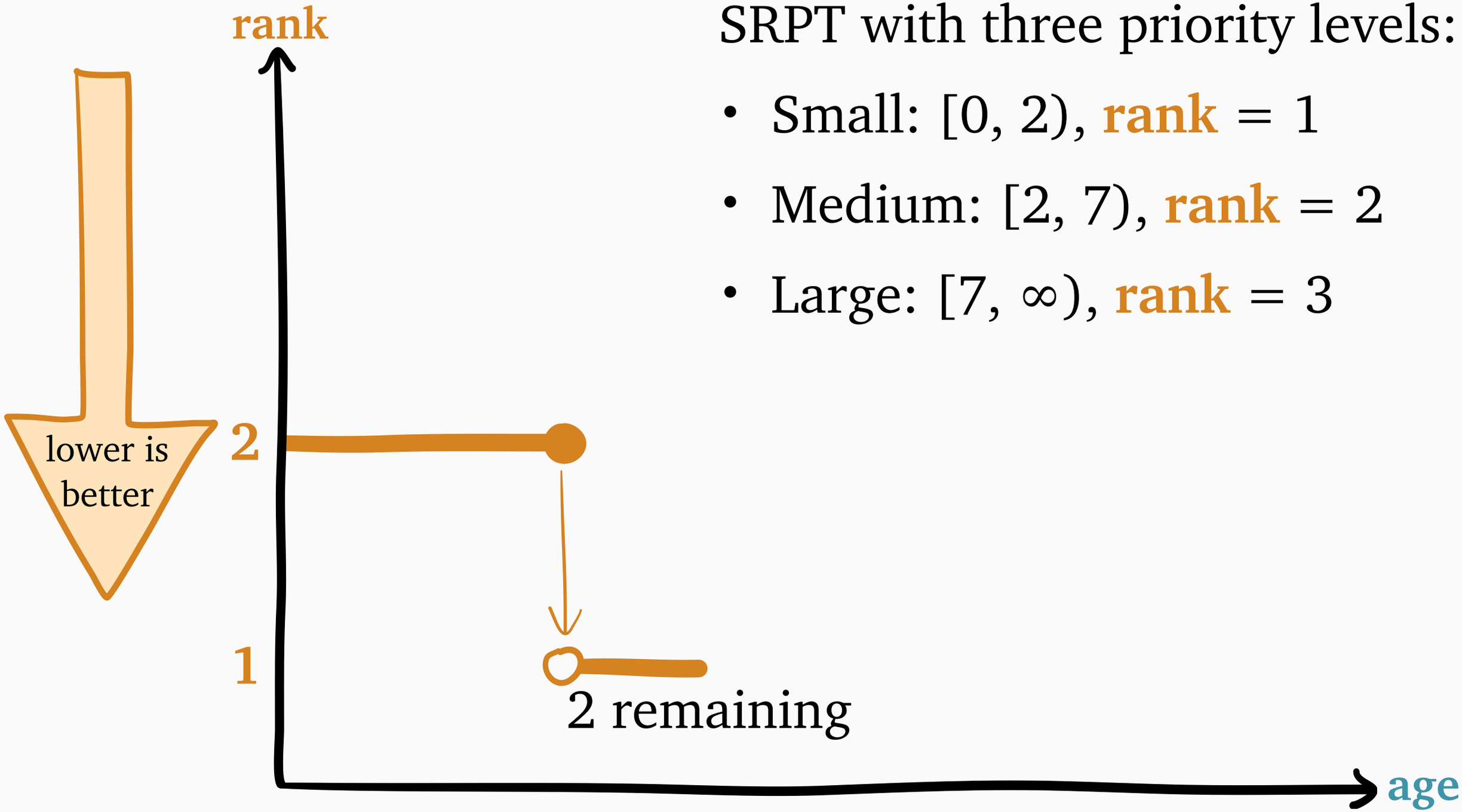


SRPT with three priority levels:

- Small: $[0, 2)$, **rank** $= 1$
- Medium: $[2, 7)$, **rank** $= 2$
- Large: $[7, \infty)$, **rank** $= 3$

back

Limited priority levels

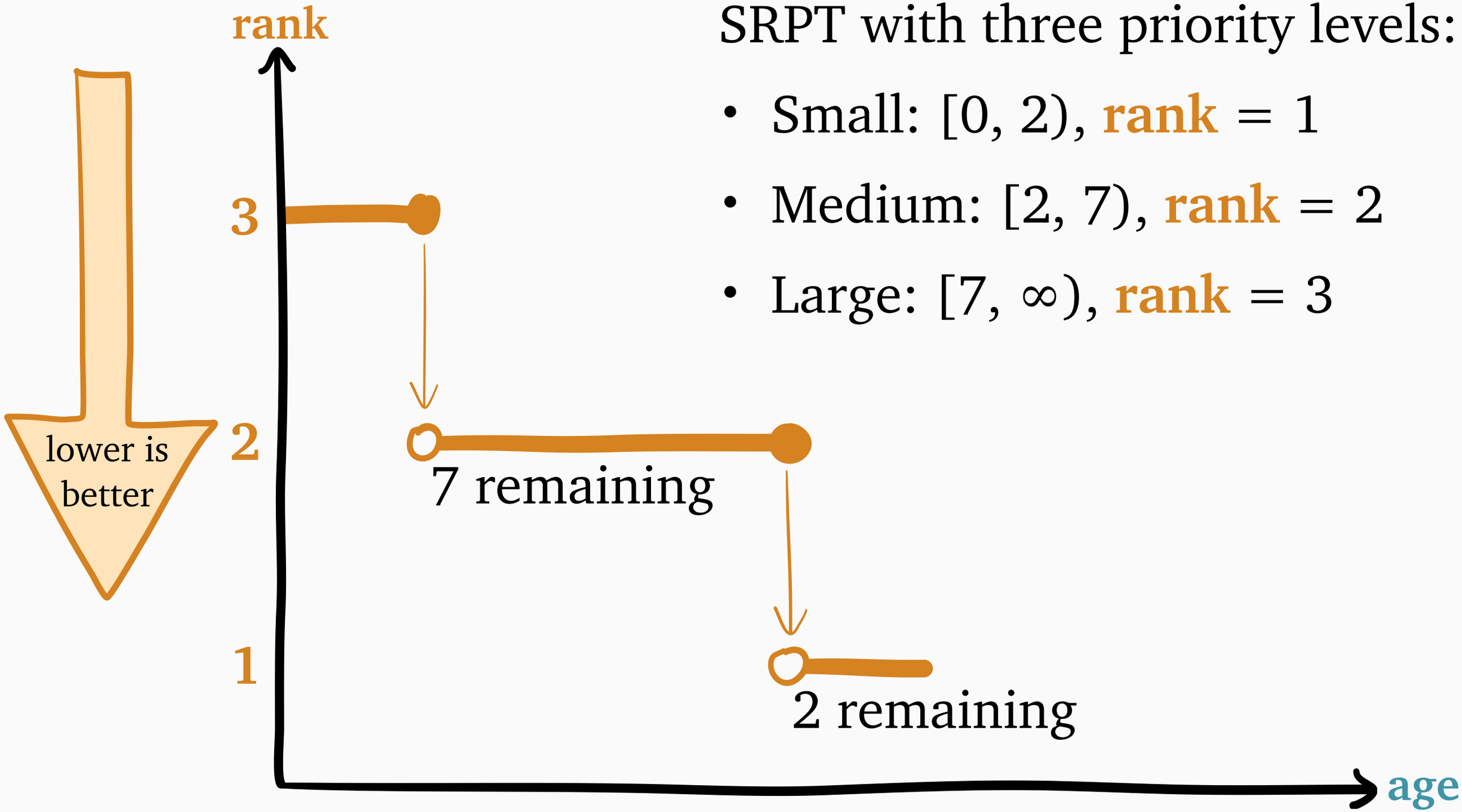


SRPT with three priority levels:

- Small: $[0, 2)$, **rank** $= 1$
- Medium: $[2, 7)$, **rank** $= 2$
- Large: $[7, \infty)$, **rank** $= 3$

back

Limited priority levels

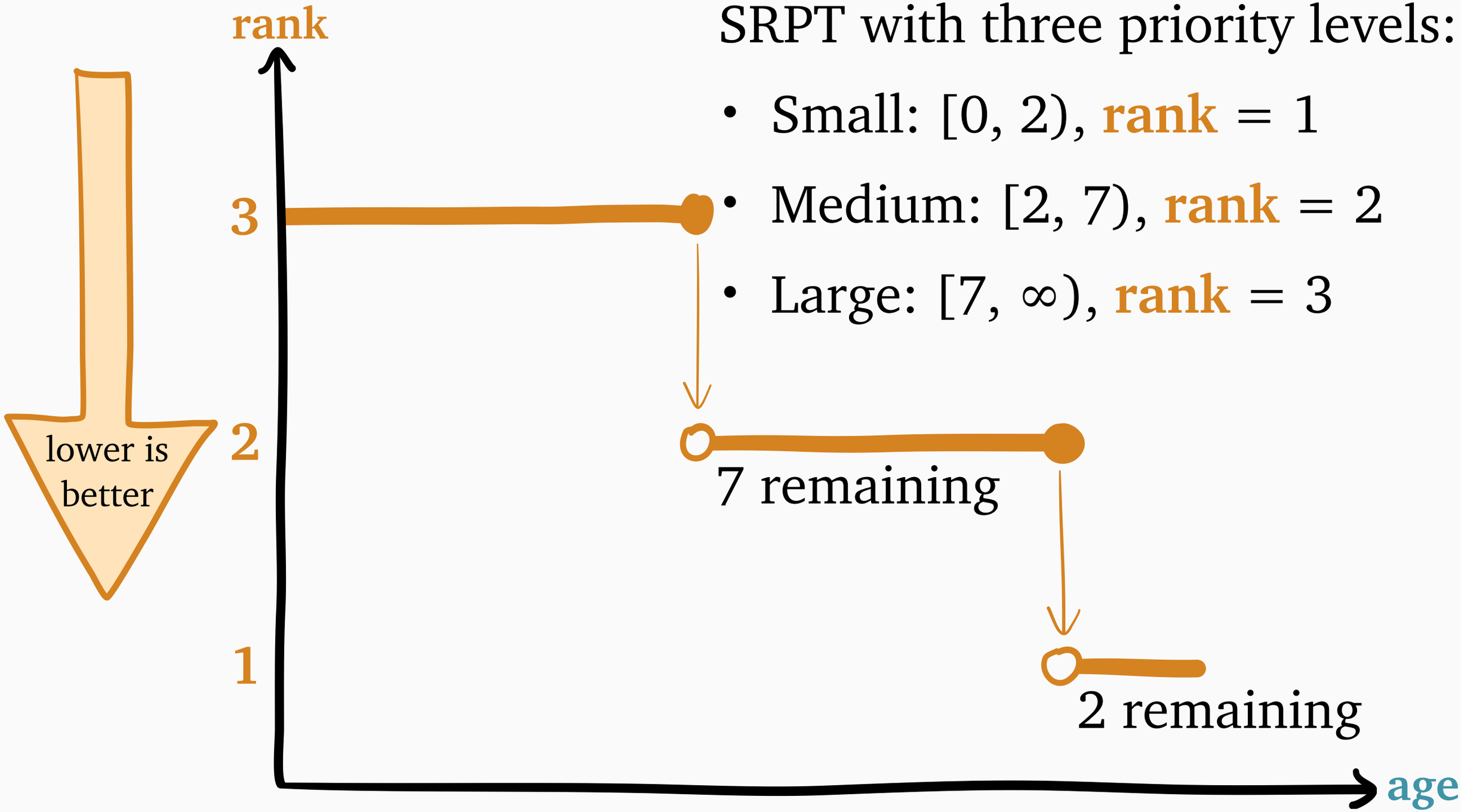


SRPT with three priority levels:

- Small: [0, 2), **rank** = 1
- Medium: [2, 7), **rank** = 2
- Large: [7, ∞), **rank** = 3

back

Limited priority levels

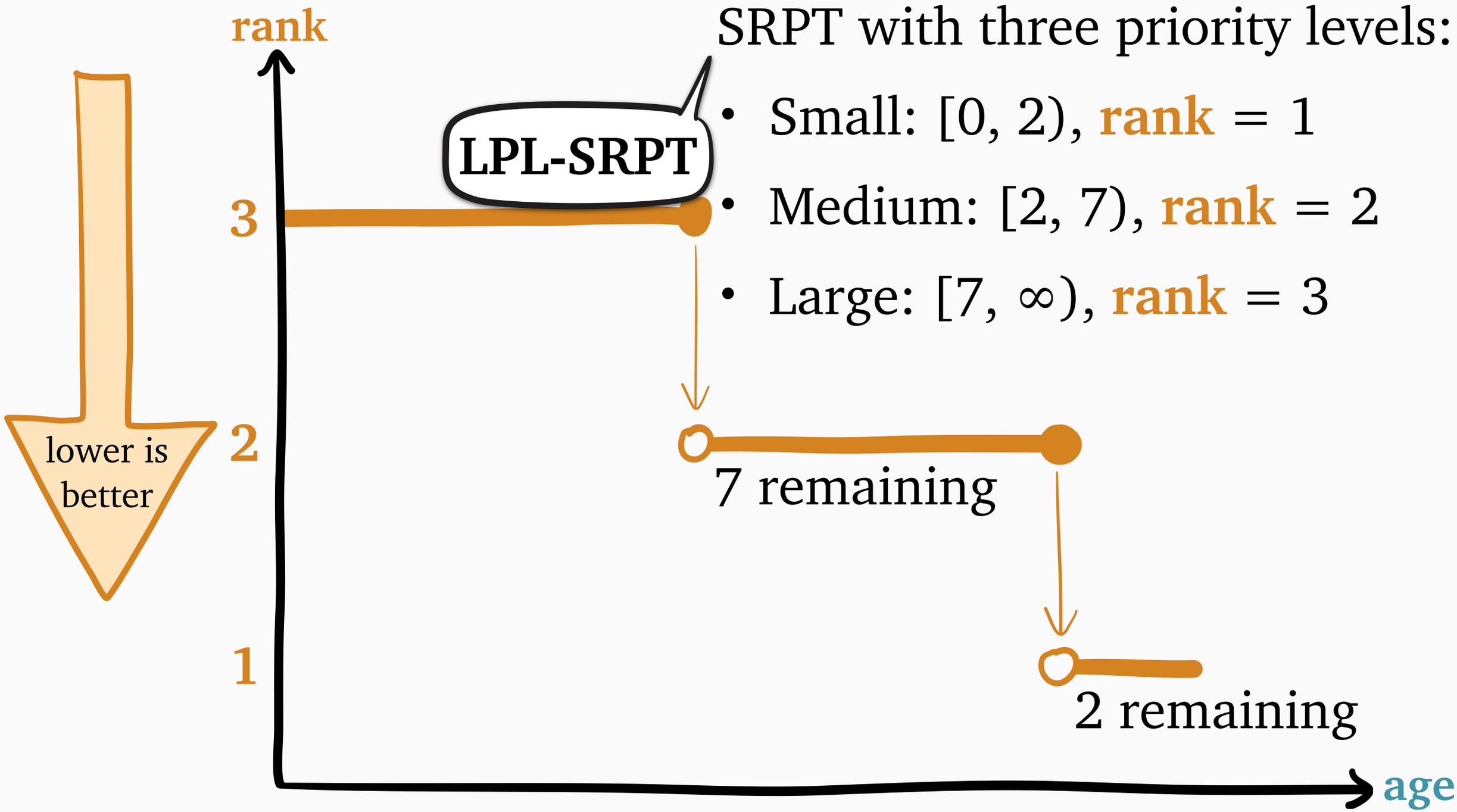


SRPT with three priority levels:

- Small: [0, 2), **rank** = 1
- Medium: [2, 7), **rank** = 2
- Large: [7, ∞), **rank** = 3

back

Limited priority levels

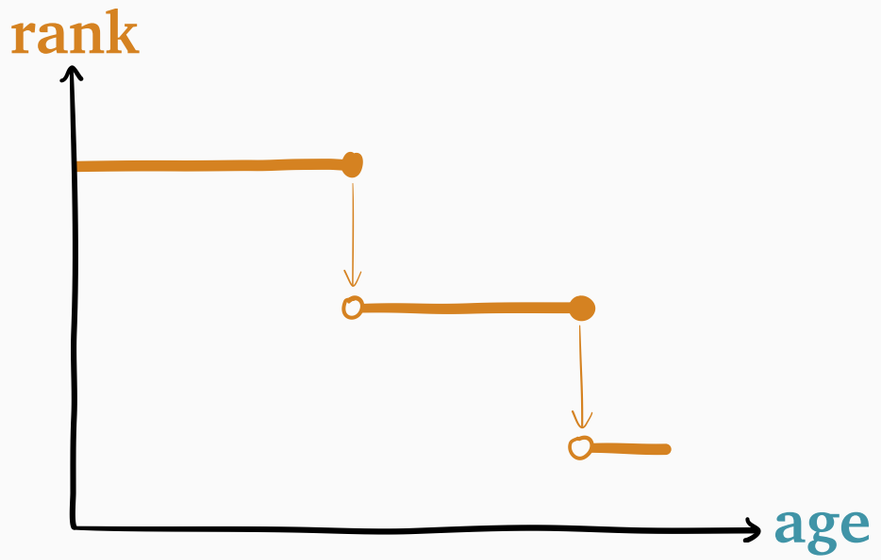


SRPT with three priority levels:

- Small: [0, 2), **rank** = 1
- Medium: [2, 7), **rank** = 2
- Large: [7, ∞), **rank** = 3

back

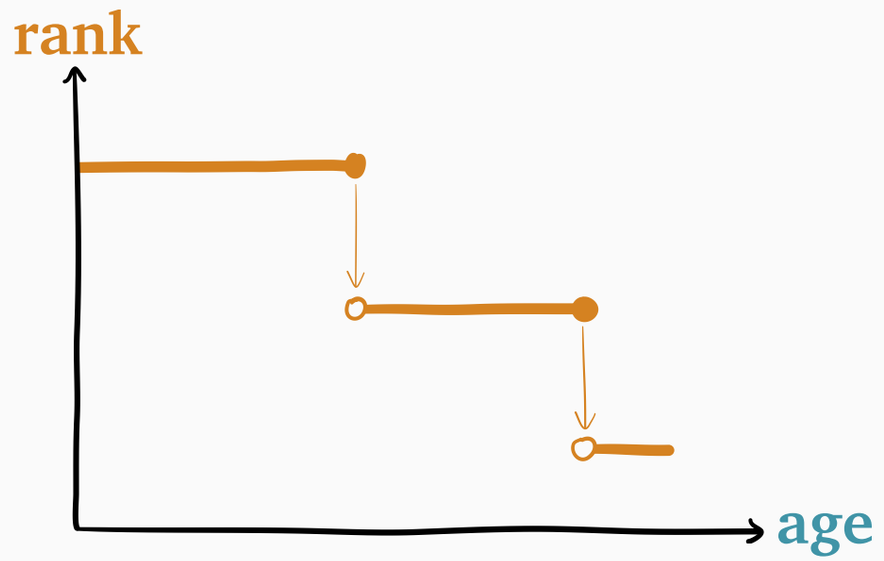
LPL-SRPT questions



- How many levels do we need?
- How do we choose size cutoffs?
- Can we do better than LPL-SRPT?

back

LPL-SRPT questions



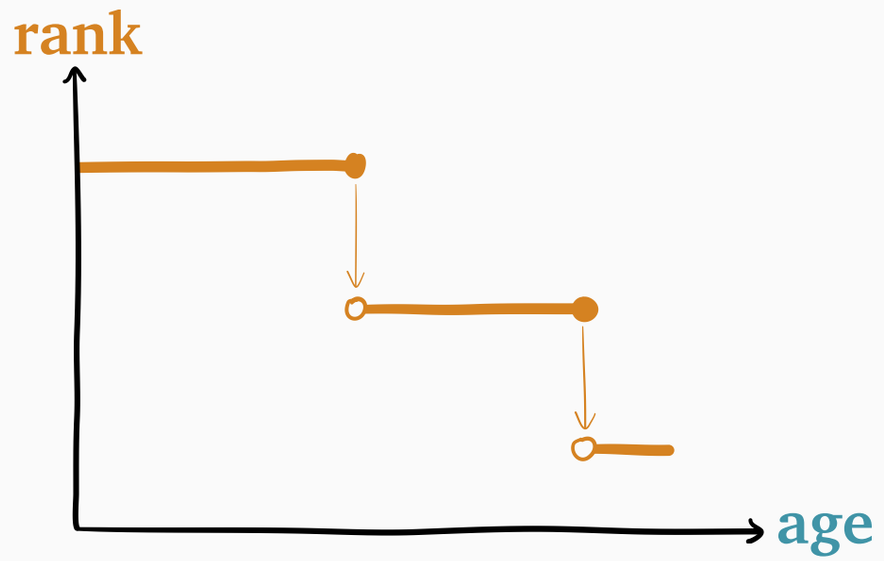
High var: 5-ish

Low var: 2-ish

- How many levels do we need?
- How do we choose size cutoffs?
- Can we do better than LPL-SRPT?

back

LPL-SRPT questions



High var: 5-ish

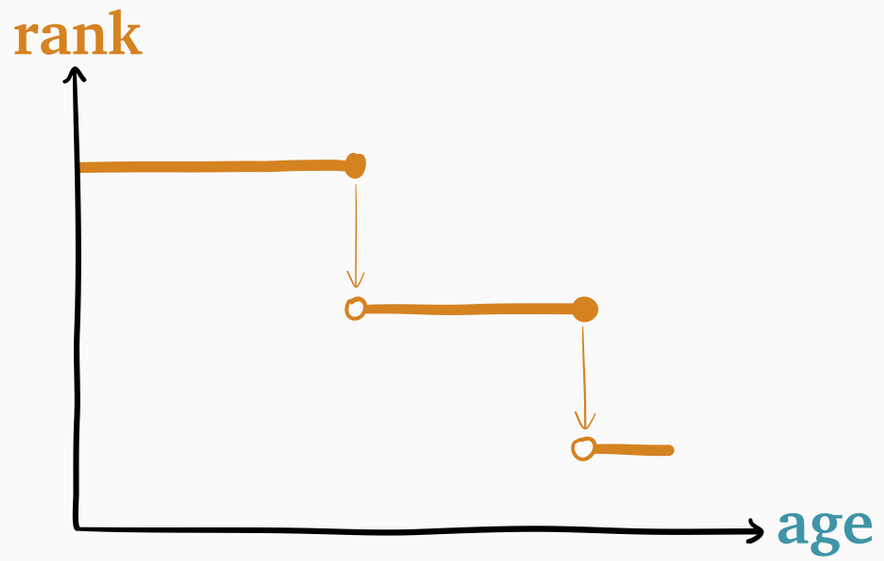
Low var: 2-ish

Balancing load is a good heuristic

- How many levels do we need?
- How do we choose size cutoffs?
- Can we do better than LPL-SRPT?

back

LPL-SRPT questions



High var: 5-ish

Low var: 2-ish

- How many levels do we need?
- How do we choose size cutoffs?
- Can we do better than LPL-SRPT?

Balancing load is a good heuristic

Yes! Constant rank better

back

	Heavy-tailed	*Light-tailed*
Prior work		
New results		

✓ = optimal
? = intermediate
✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	
	✓ SRPT	
	✓ LAS	
New results		

✓ = optimal
? = intermediate
✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results		

✓ = optimal
? = intermediate
✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results	✓ LAS with checkpoints, constant gap	

✓ = optimal
? = intermediate
✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results	✓ LAS with checkpoints, constant gap	

not always optimal with growing gap

✓ = optimal
? = intermediate
✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results	✓ LAS with checkpoints, constant gap	
	✓ **Gittins**, (**M-**)**SERPT**	

not always optimal with growing gap

✓ = optimal
? = intermediate
✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results	✓ LAS with checkpoints, constant gap	?✗ Non-FCFS **SOAP**
	✓ **Gittins**, (**M-**)**SERPT**	

✓ = optimal
? = intermediate
✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results	✓ LAS with checkpoints, constant gap	?✗ Non-FCFS **SOAP**
	✓ **Gittins**, (**M-**)**SERPT**	✓?✗ **Gittins**, (**M-**)**SERPT**

✓ = optimal
? = intermediate
✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results	✓ LAS with checkpoints, constant gap	?✗ Non-FCFS **SOAP**
	✓ **Gittins**, (**M-**)**SERPT**	✓?✗ **Gittins**, (**M-**)**SERPT**

for some S, **Gittins** = FCFS

✓ = optimal

? = intermediate

✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results	✓ LAS with checkpoints, constant gap	?✗ Non-FCFS **SOAP**
	✓ **Gittins**, (**M-**)**SERPT**	✓?✗ **Gittins**, (**M-**)**SERPT**

for some S, **Gittins** = FCFS

for some S, **Gittins** = LAS

✓ = optimal

? = intermediate

✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results	✓ LAS with checkpoints, constant gap	?✗ Non-FCFS **SOAP**
	✓ **Gittins**, (**M-**)**SERPT**	✓?✗ **Gittins**, (**M-**)**SERPT**
		✓? Modified **Gittins**

✓ = optimal
? = intermediate
✗ = pessimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results	✓ LAS with checkpoints, constant gap	?✗ Non-FCFS **SOAP**
	✓ **Gittins**, (**M-**)**SERPT**	✓?✗ **Gittins**, (**M-**)**SERPT**
		✓? Modified **Gittins**

✓ = optimal
? = intermediate
✗ = pessimal

$\mathbf{E}[T]$ within $1+\varepsilon$ factor of optimal

back

	Heavy-tailed	*Light-tailed*
Prior work	✗ FCFS	✓ FCFS
	✓ SRPT	✗ SRPT
	✓ LAS	✗ LAS
New results	✓ LAS with checkpoints, constant gap	?✗ Non-FCFS **SOAP**
	✓ **Gittins**, (**M-**)**SERPT**	✓?✗ **Gittins**, (**M-**)**SERPT**
		✓? Modified **Gittins**

✓ = optimal
? = intermediate
✗ = pessimal

$\mathbf{E}[T]$ within $1+\varepsilon$ factor of optimal

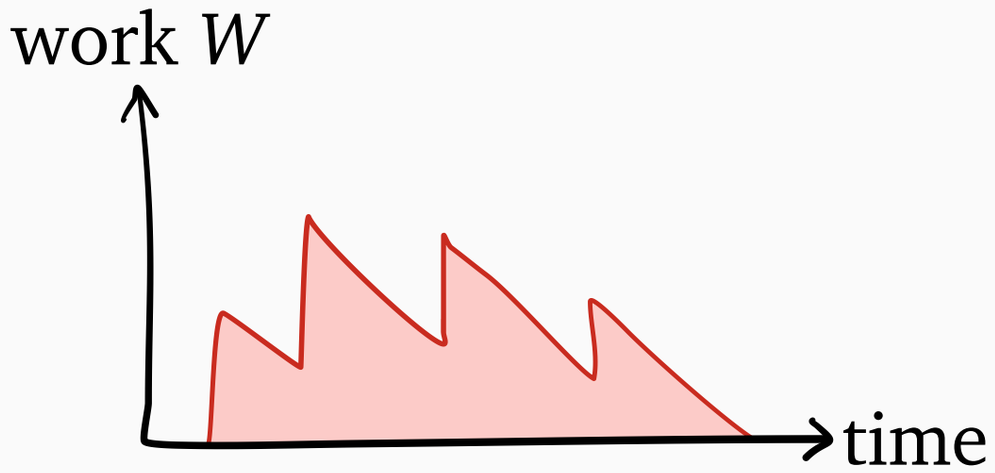
[Scully, Kreveld, Boxma, Dorsman, and Wierman, SIGMETRICS 2020]
[Scully and Kreveld, arXiv 2021]

back

Work decomposition

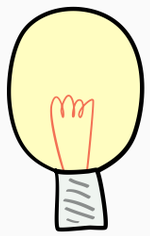
back

Work decomposition

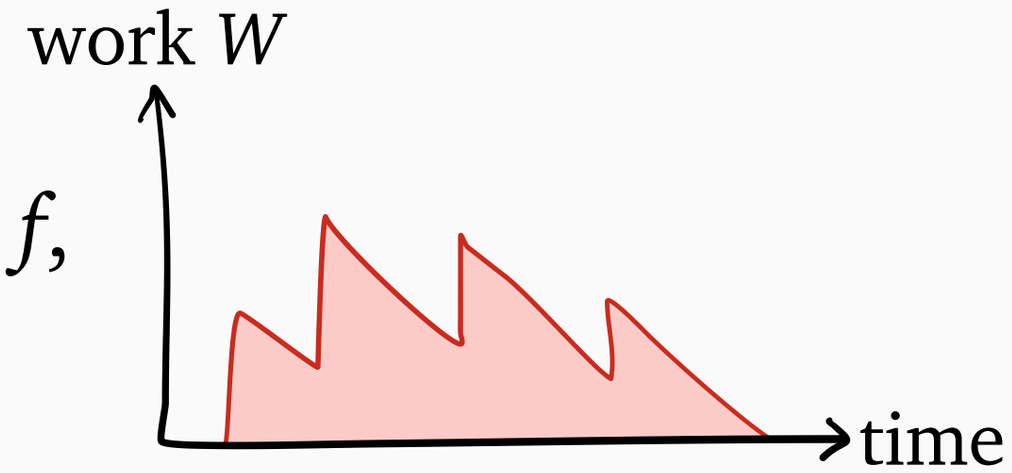


back

Work decomposition

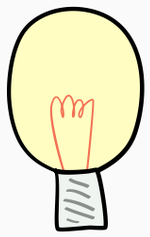


In steady-state system, for any f,
$\mathbf{E}[\, f(W)]$ constant w.r.t. time



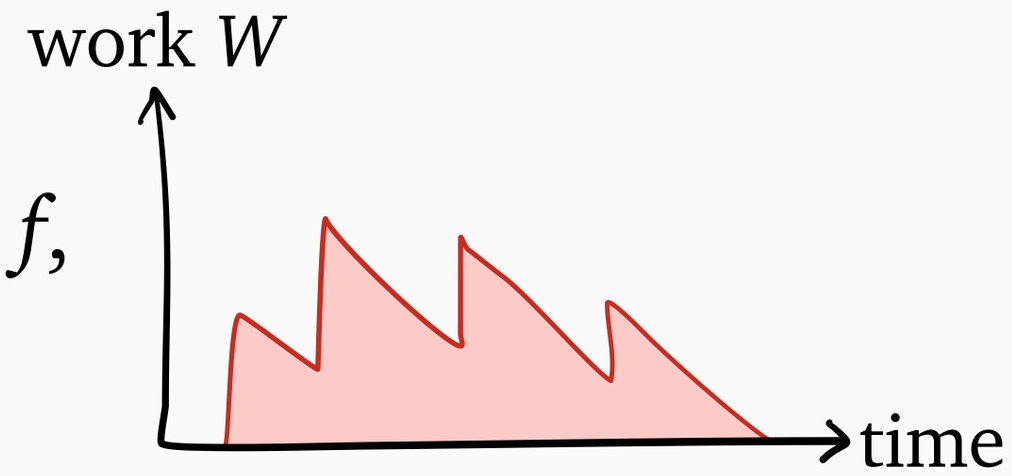
back

Work decomposition



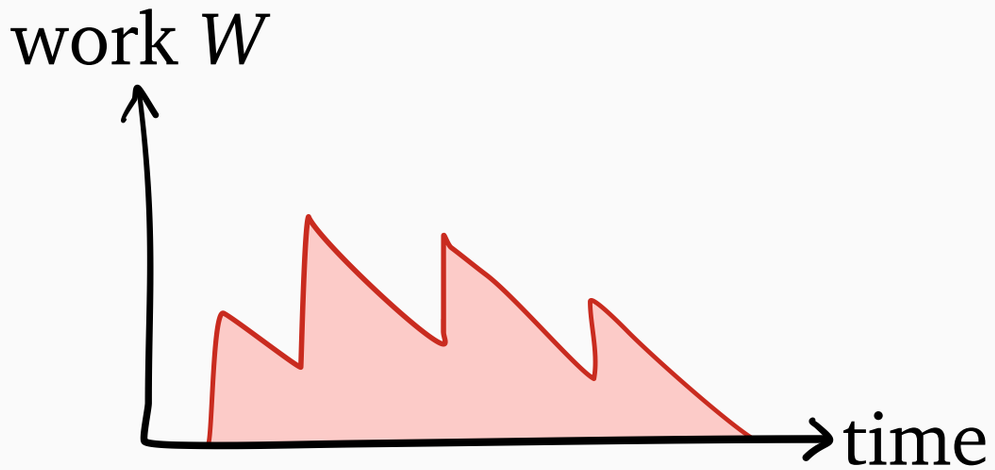
In steady-state system, for any f,
$\mathbf{E}[\, f(W)]$ constant w.r.t. time

we use
$f(w) = w^2$



back

Work decomposition



$$\mathbf{E}[W^2 \text{ decrease rate}] = 2\mathbf{E}[BW]$$

$$\mathbf{E}[W^2 \text{ increase rate}] = \lambda\mathbf{E}[(W+S)^2 - W^2]$$

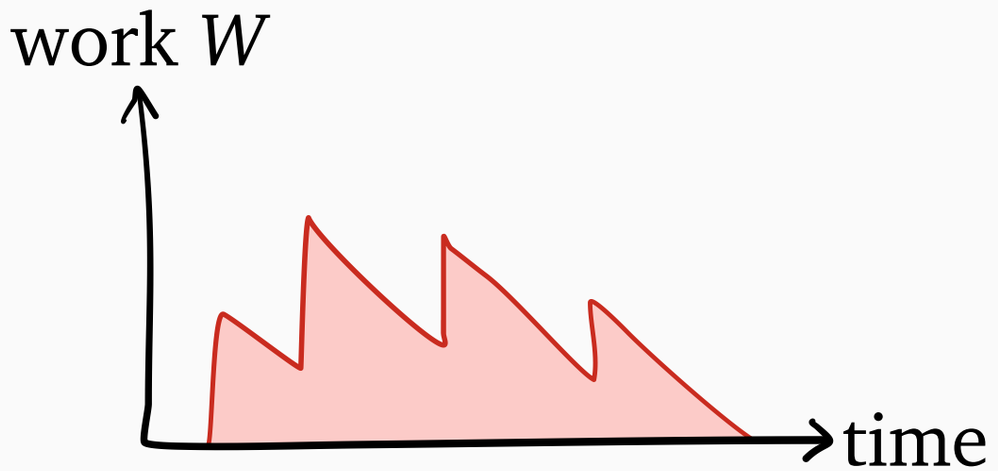
back

Work decomposition

B = service rate, a.k.a. fraction of servers busy

$$\mathbf{E}[W^2 \text{ decrease rate}] = 2\mathbf{E}[BW]$$

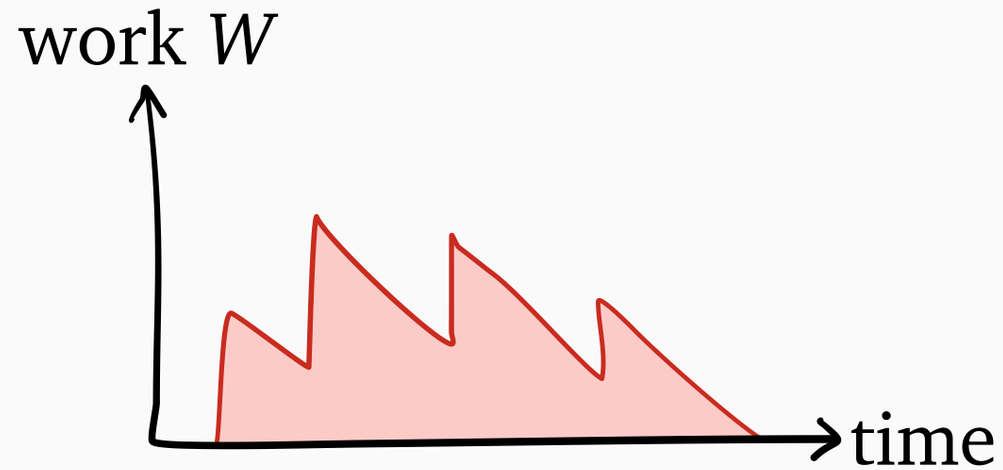
$$\mathbf{E}[W^2 \text{ increase rate}] = \lambda\mathbf{E}[(W+S)^2 - W^2]$$



back

Work decomposition

B = service rate, a.k.a. fraction of servers busy



$$\mathbf{E}[W^2 \text{ decrease rate}] = 2\mathbf{E}[BW]$$

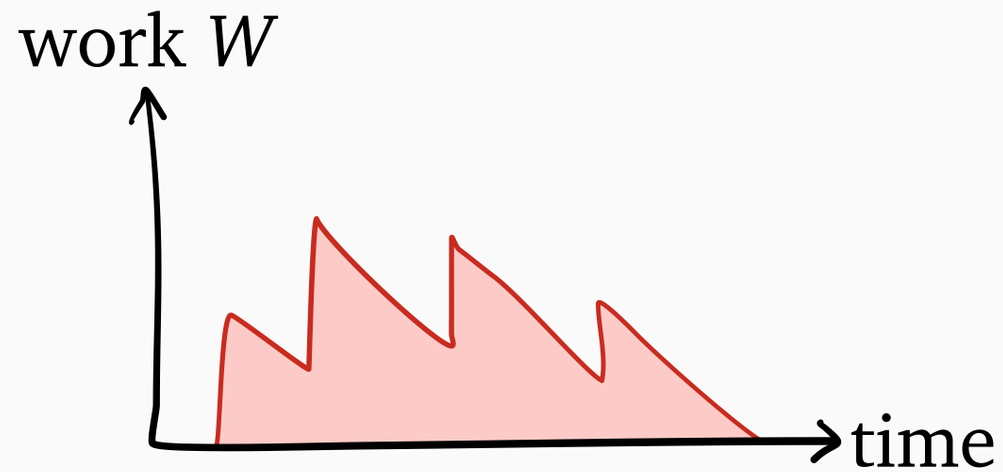
$$\mathbf{E}[W^2 \text{ increase rate}] = \lambda\mathbf{E}[(W+S)^2 - W^2]$$

$$\mathbf{E}[W] = \frac{\frac{\lambda}{2}\mathbf{E}[S^2]}{1-\rho} + \frac{\mathbf{E}[(1-B)W]}{1-\rho}$$

back

Work decomposition

B = service rate, a.k.a. fraction of servers busy



$$\mathbf{E}[W^2 \text{ decrease rate}] = 2\mathbf{E}[BW]$$

$$\mathbf{E}[W^2 \text{ increase rate}] = \lambda\mathbf{E}[(W+S)^2 - W^2]$$

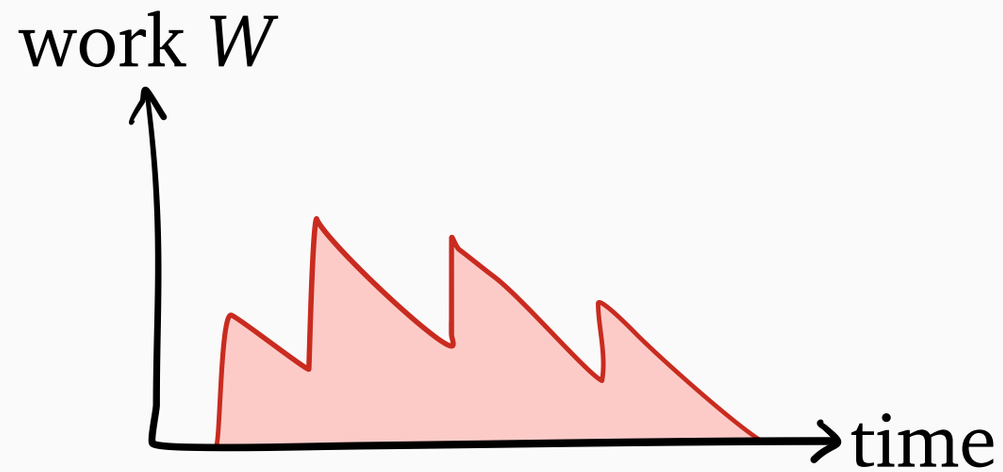
if **single server**: $(1-B)W = 0$

$$\mathbf{E}[W] = \frac{\frac{\lambda}{2}\mathbf{E}[S^2]}{1-\rho} + \frac{\mathbf{E}[(1-B)W]}{1-\rho}$$

back

Work decomposition

B = service rate, a.k.a. fraction of servers busy



$$\mathbf{E}[W^2 \text{ decrease rate}] = 2\mathbf{E}[BW]$$

$$\mathbf{E}[W^2 \text{ increase rate}] = \lambda\mathbf{E}[(W+S)^2 - W^2]$$

if **single server**: $(1-B)W = 0$

$$\mathbf{E}[W] = \frac{\frac{\lambda}{2}\mathbf{E}[S^2]}{1-\rho} + \frac{\mathbf{E}[(1-B)W]}{1-\rho}$$

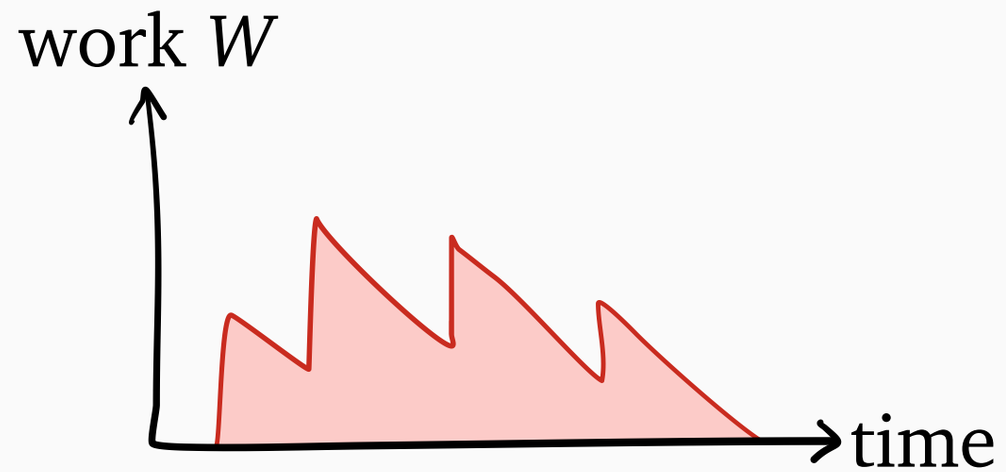
Lemma:

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1-B_k)W_k]}{1-\rho}$$

back

Work decomposition

B = service rate, a.k.a. fraction of servers busy



$$\mathbf{E}[W^2 \text{ decrease rate}] = 2\mathbf{E}[BW]$$

$$\mathbf{E}[W^2 \text{ increase rate}] = \lambda\mathbf{E}[(W+S)^2 - W^2]$$

if **single server**: $(1-B)W = 0$

newish

$$\mathbf{E}[W] = \frac{\frac{\lambda}{2}\mathbf{E}[S^2]}{1-\rho} + \frac{\mathbf{E}[(1-B)W]}{1-\rho}$$

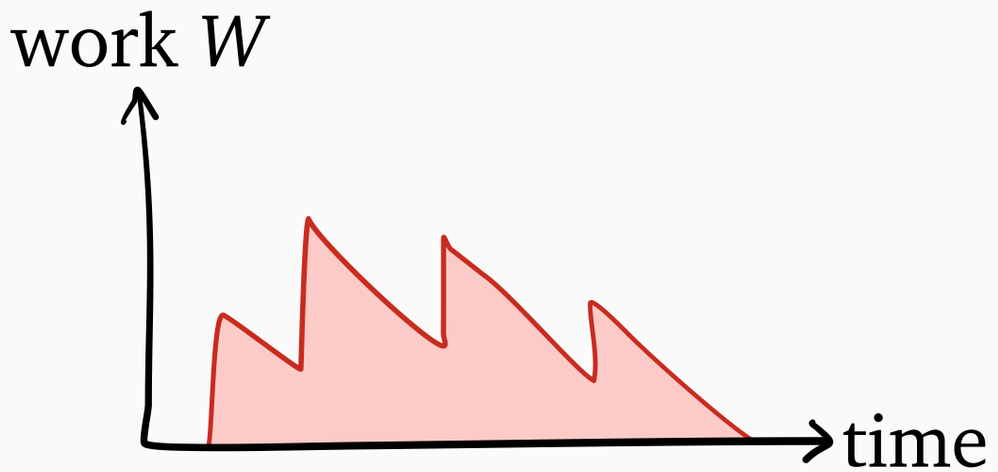
Lemma:

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1-B_k)W_k]}{1-\rho}$$

back

Work decomposition

B = service rate, a.k.a. fraction of servers busy



$$\mathbf{E}[W^2 \text{ decrease rate}] = 2\mathbf{E}[BW]$$

$$\mathbf{E}[W^2 \text{ increase rate}] = \lambda\mathbf{E}[(W+S)^2 - W^2]$$

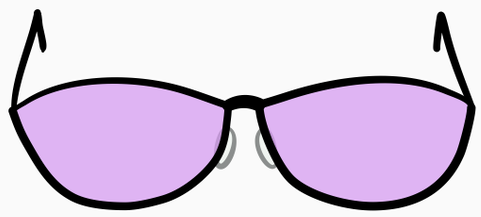
if **single server**: $(1-B)W = 0$

$$\mathbf{E}[W] = \frac{\frac{\lambda}{2}\mathbf{E}[S^2]}{1-\rho} + \frac{\mathbf{E}[(1-B)W]}{1-\rho}$$

newish

Lemma:

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1-B_k)W_k]}{1-\rho}$$



Similar story with r-work

NEW!

back

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1 - B_k)W_k]}{1 - \rho}$$

back

Suppose $S \leq s_{\max}$ with probability 1

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1 - B_k)W_k]}{1 - \rho}$$

back

Suppose $S \leq s_{\max}$ with probability 1

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1 - B_k)W_k]}{1 - \rho}$$

$\leq (k-1)s_{\max}$

back

Suppose $S \leq s_{\max}$ with probability 1

$\mathbf{E}[B] = \rho$ $\leq (k-1)s_{\max}$

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1-B_k)W_k]}{1-\rho}$$

back

Suppose $S \leq s_{\max}$ with probability 1

$\mathbf{E}[B] = \rho$ $\quad$ $\leq (k-1)s_{\max}$

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1-B_k)W_k]}{1-\rho}$$

$$\leq \mathbf{E}[W_1] + (k-1)s_{\max}$$

back

Suppose $S \leq s_{\max}$ with probability 1

$\mathbf{E}[B] = \rho$

$\leq (k-1)s_{\max}$

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1-B_k)W_k]}{1-\rho}$$

$$\leq \mathbf{E}[W_1] + (k-1)s_{\max}$$

"work of $\leq k-1$ jobs"

back

Suppose $S \leq s_{\max}$ with probability 1

$\mathbf{E}[B] = \rho$

$\leq (k-1)s_{\max}$

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1-B_k)W_k]}{1-\rho}$$

$$\leq \mathbf{E}[W_1] + (k-1)s_{\max}$$

"work of $\leq k-1$ jobs"

$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \text{“}r\text{-work of } k-1 \text{ jobs”}$$

back

Suppose $S \le s_{\max}$ with probability 1

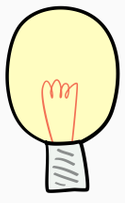
$\mathbf{E}[B] = \rho$

$\le (k-1)s_{\max}$

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1-B_k)W_k]}{1-\rho}$$

$$\le \mathbf{E}[W_1] + (k-1)s_{\max}$$

"work of $\le k-1$ jobs"



Single job's r-work is at most r

$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \text{“}r\text{-work of } k-1 \text{ jobs”}$$

back

Suppose $S \le s_{\max}$ with probability 1

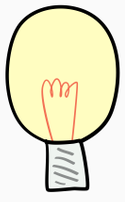
$\mathbf{E}[B] = \rho$

$\le (k-1)s_{\max}$

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1-B_k)W_k]}{1-\rho}$$

$$\le \mathbf{E}[W_1] + (k-1)s_{\max}$$

"work of $\le k-1$ jobs"



Single job's r-work is at most r

$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \text{“}r\text{-work of } k-1 \text{ jobs”}$$

$$\le \mathbf{E}[W_1] + (k-1)r$$

back

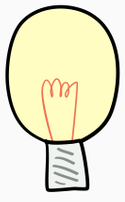
Suppose $S \le s_{\max}$ with probability 1

$\mathbf{E}[B] = \rho$ $\quad$ $\le (k-1)s_{\max}$

$$\mathbf{E}[W_k] = \mathbf{E}[W_1] + \frac{\mathbf{E}[(1-B_k)W_k]}{1-\rho}$$

$$\le \mathbf{E}[W_1] + (k-1)s_{\max}$$

"work of $\le k-1$ jobs"



Single job's r-work is at most r

$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \text{“}r\text{-work of } k-1 \text{ jobs”}$$

$$\le \mathbf{E}[W_1] + (k-1)r$$

can improve

back

SRPT-k and Gittins-k $\mathbf{E}[T]$

WINE gives *first analysis* of

- SRPT-k
- Gittins-k

back

SRPT-k and Gittins-k $\mathbf{E}[T]$

WINE gives *first analysis* of

- SRPT-k
- Gittins-k

Theorem: for SRPT and Gittins,

$$\mathbf{E}[T_k] \leq \mathbf{E}[T_1] + (k-1) \cdot O\left(\log \frac{1}{1-\rho}\right)$$

back

SRPT-k and Gittins-k $\mathbf{E}[T]$

WINE gives *first analysis* of

- SRPT-k
- Gittins-k

Theorem: for SRPT and Gittins,

$$\mathbf{E}[T_k] \leq \mathbf{E}[T_1] + \underbrace{(k-1) \cdot O\left(\log \frac{1}{1-\rho}\right)}_{o(\mathbf{E}[T_1])}$$

back

SRPT-k and Gittins-k $\mathbf{E}[T]$

WINE gives *first analysis* of

- SRPT-k
- Gittins-k

Theorem: for SRPT and Gittins,

$$\mathbf{E}[T_k] \leq \mathbf{E}[T_1] + \underbrace{(k-1) \cdot O\left(\log \frac{1}{1-\rho}\right)}_{o(\mathbf{E}[T_1])}$$

Corollary: SRPT and Gittins minimize $\mathbf{E}[T]$ in heavy traffic (in their respective settings)