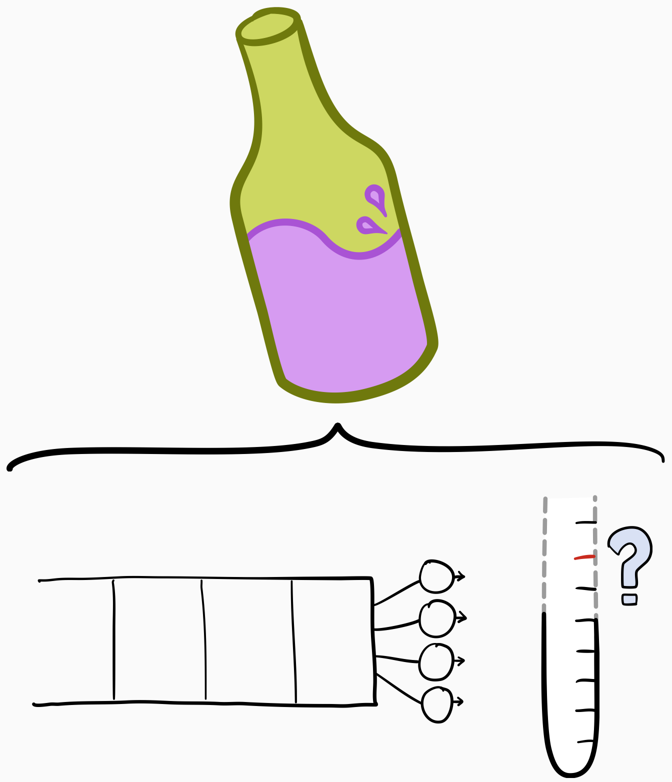
How Robust Is *the* Gittins **Policy** *for* Queue Scheduling?

Ziv Scully
MIT/Harvard (now) → Cornell (Fall 2023)

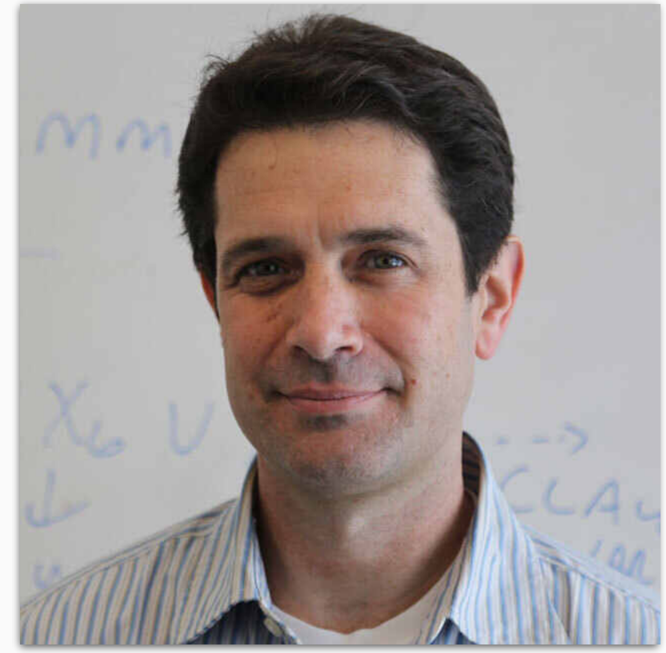
zivscully@cornell.edu
https://ziv.codes



Collaborators

Mor Harchol-Balter
CMU

Isaac Grosof
CMU

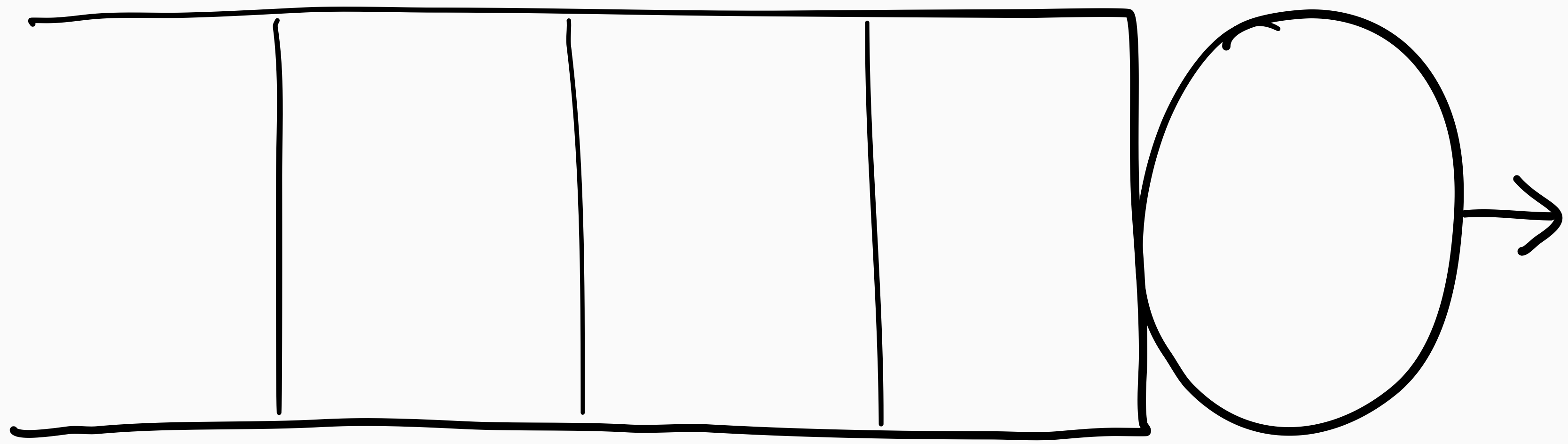


Michael Mitzenmacher
Harvard

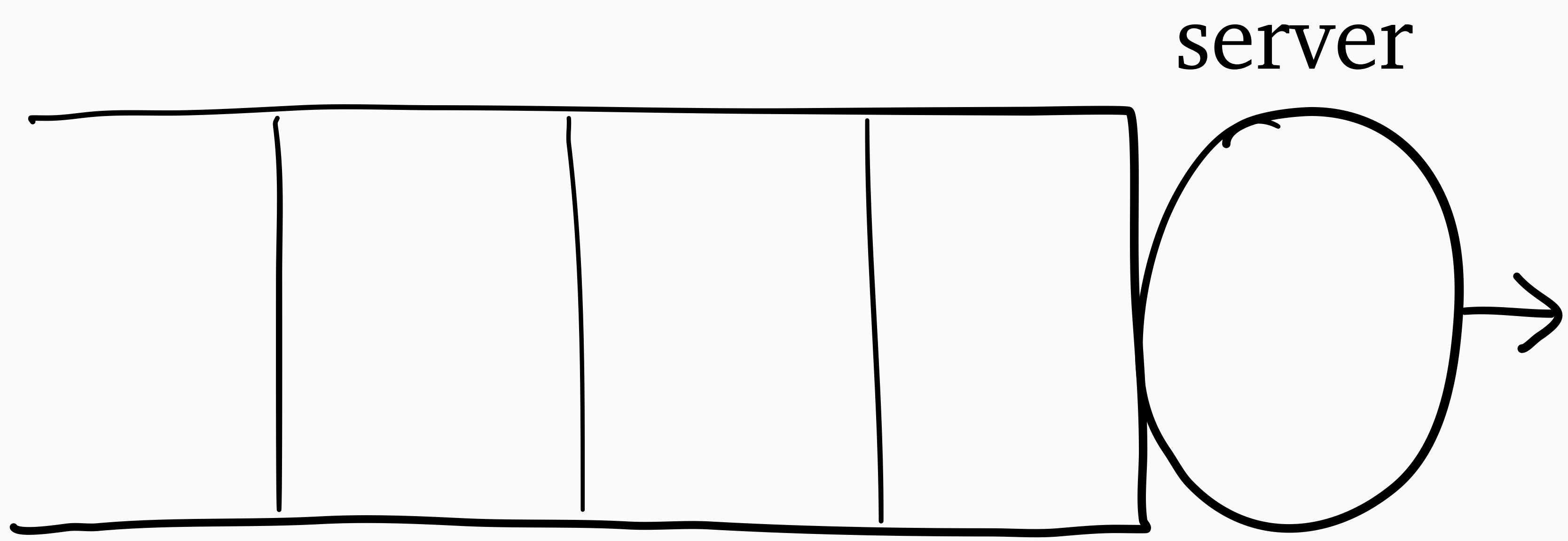
References:

- Scully, Grosof, & Harchol-Balter, POMACS 2020 / SIGMETRICS 2021
- Scully & Harchol-Balter, WiOpt 2021
- Scully, Grosof, & Mitzenmacher, ITCS 2022
- Scully, PhD thesis 2022

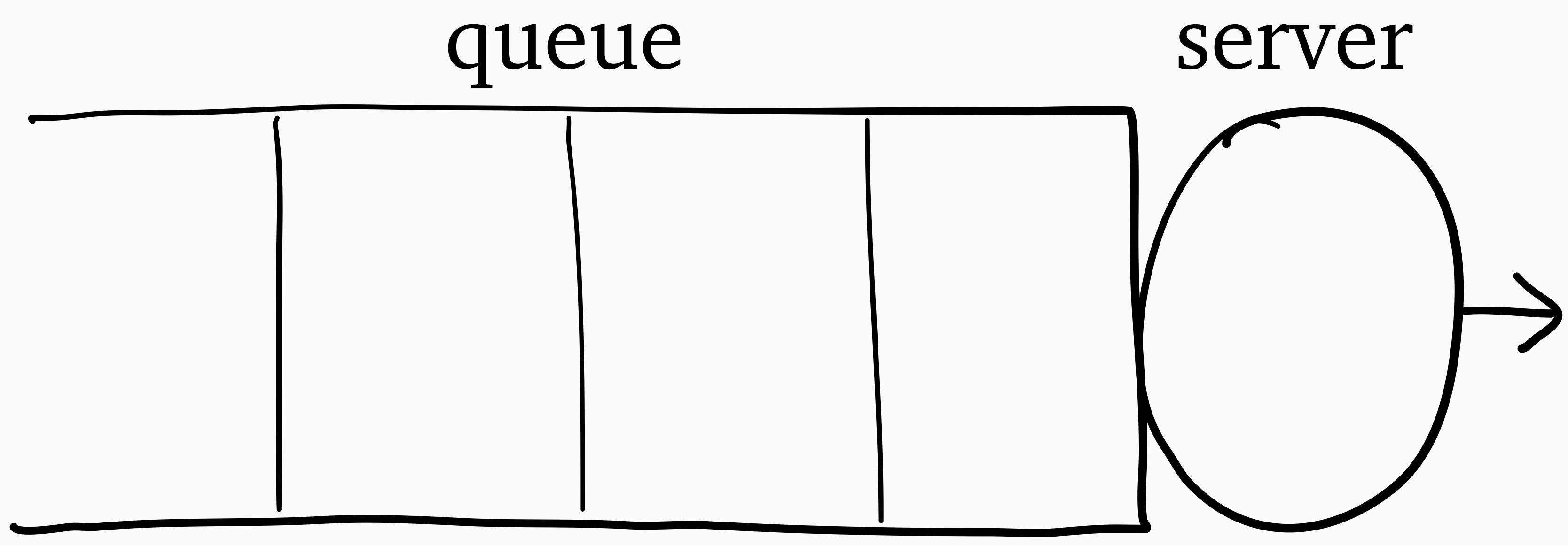
Queue scheduling (M/G/1)



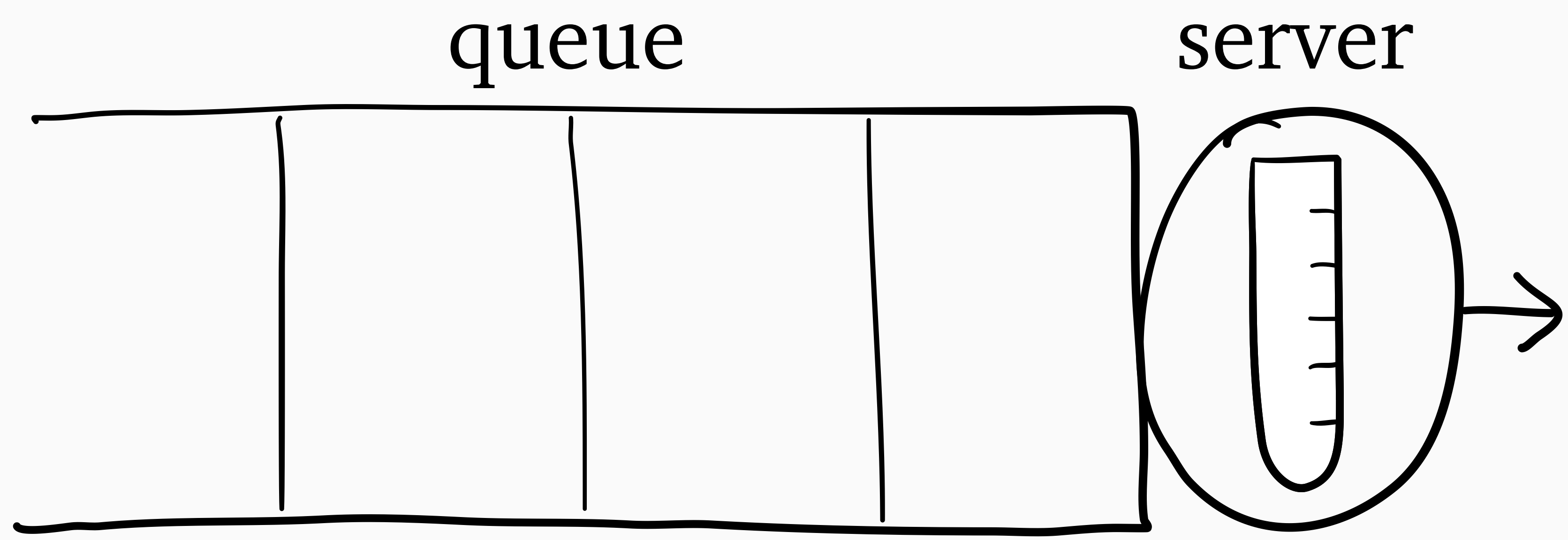
Queue scheduling (M/G/1)

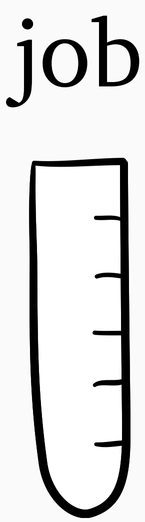


Queue scheduling (M/G/1)

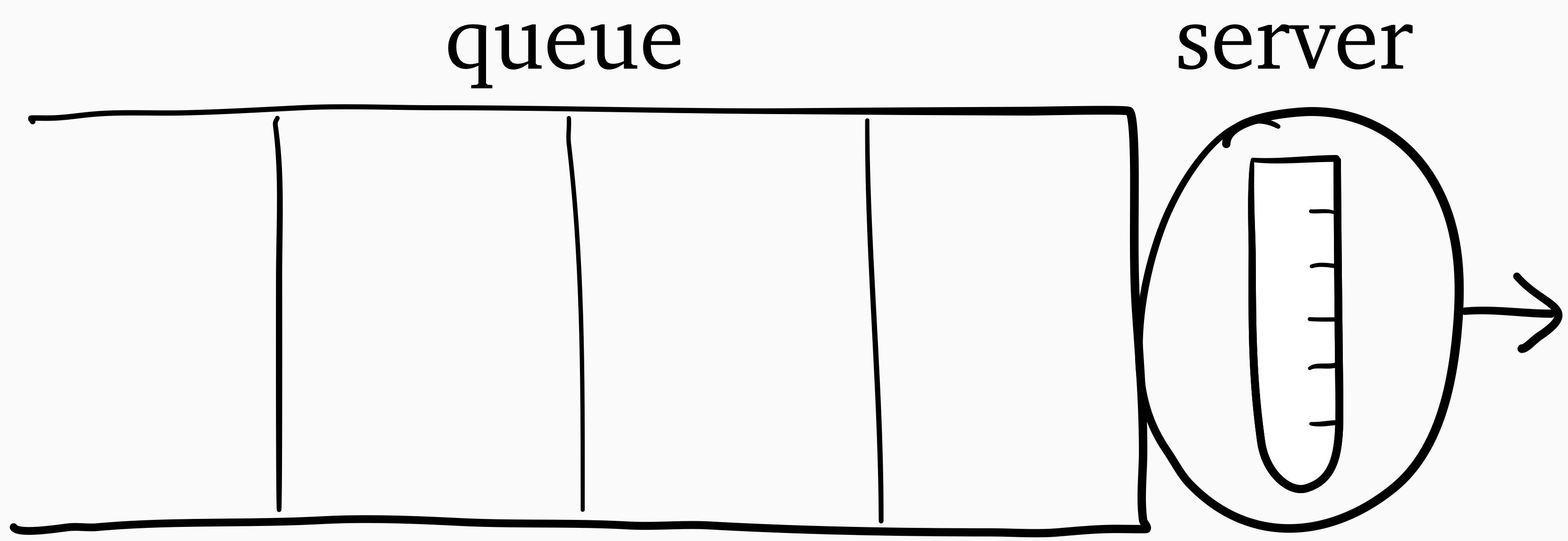


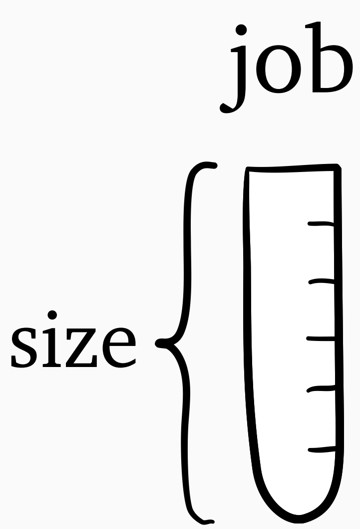
Queue scheduling (M/G/1)



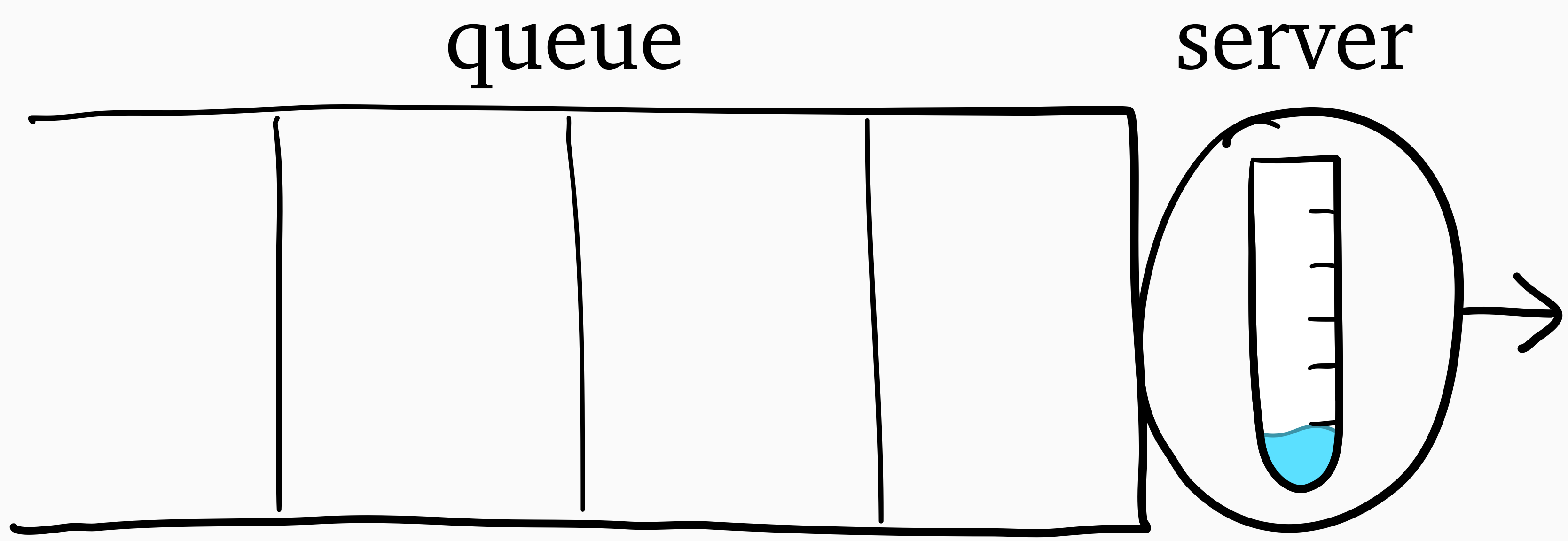


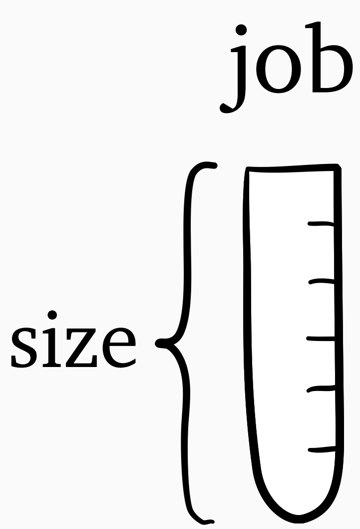
Queue scheduling (M/G/1)



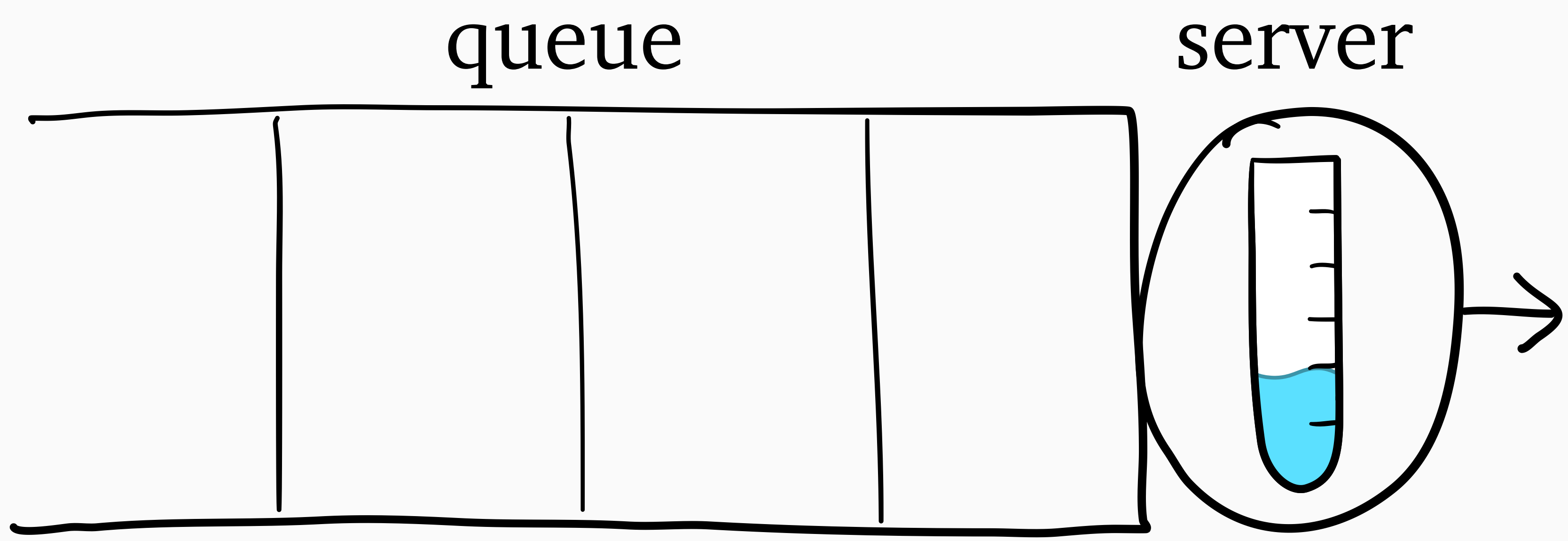


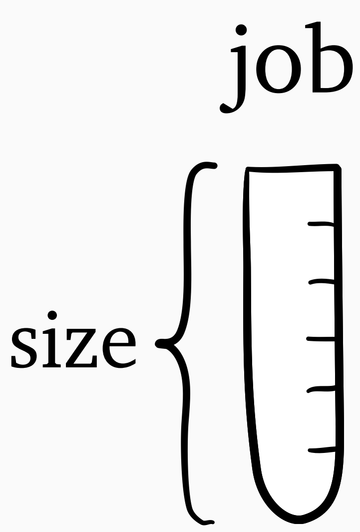
Queue scheduling (M/G/1)



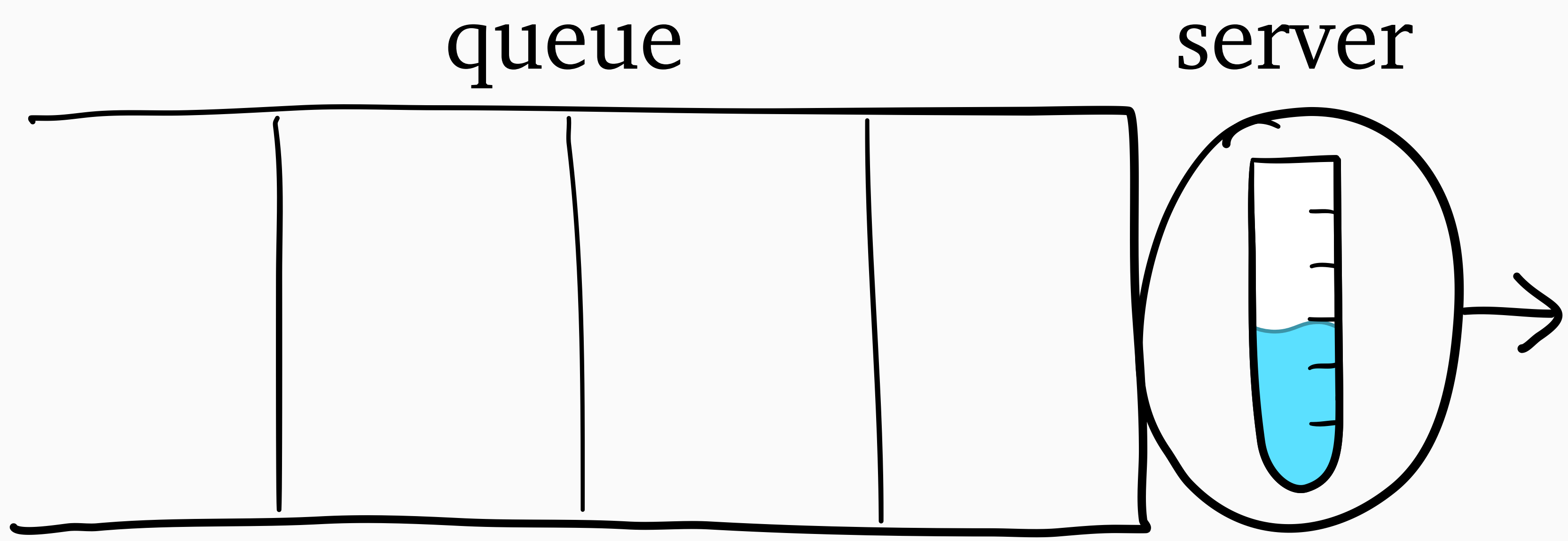


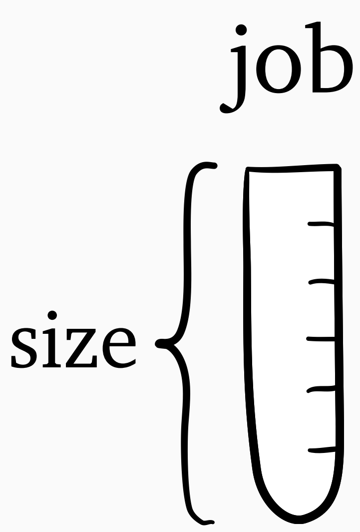
Queue scheduling (M/G/1)



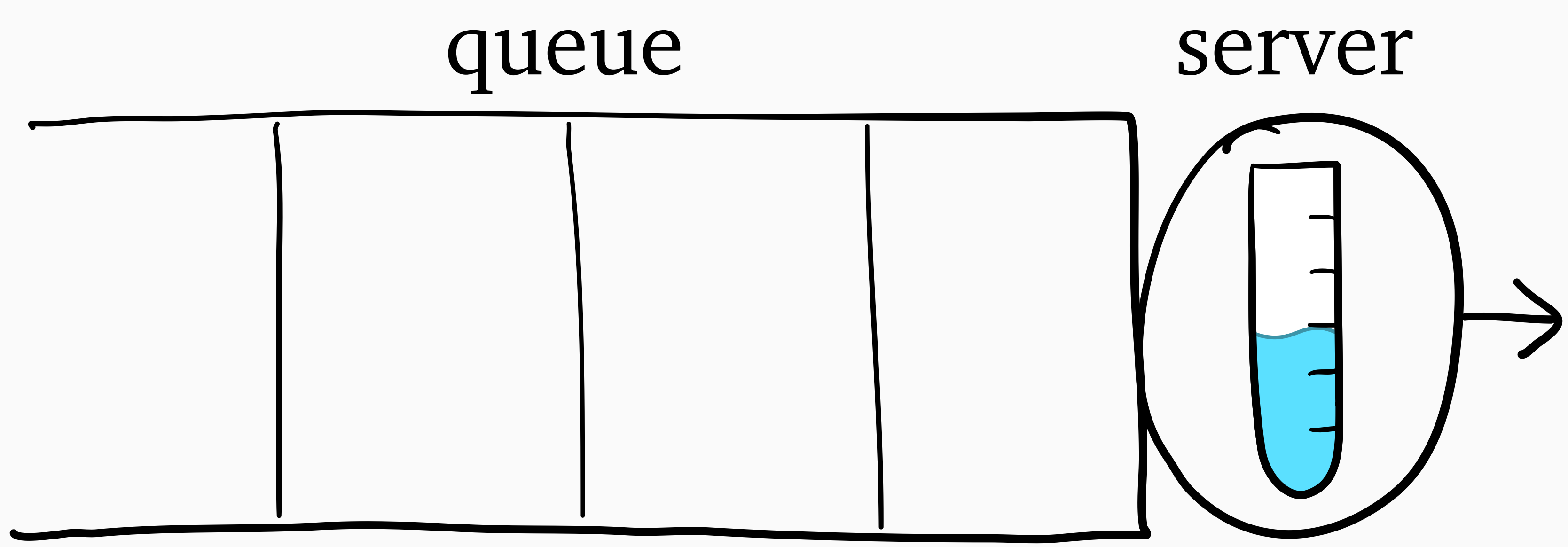


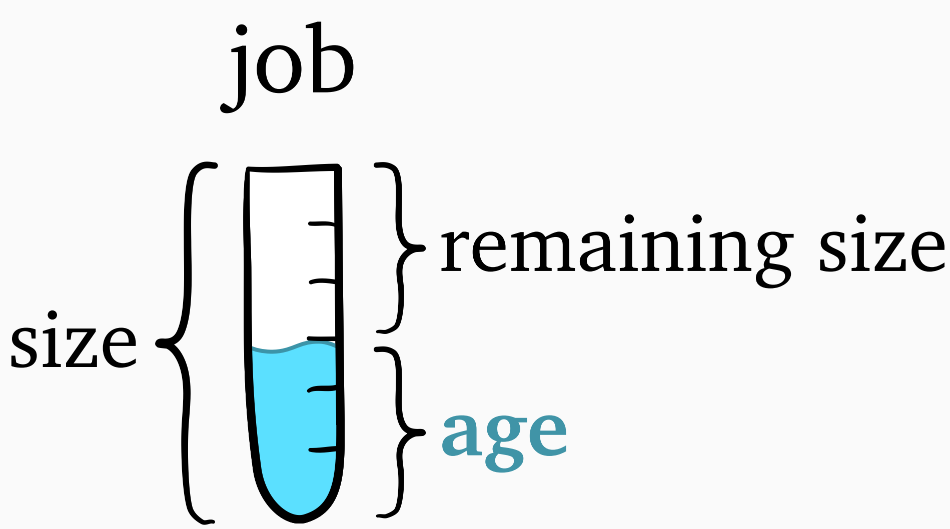
Queue scheduling (M/G/1)





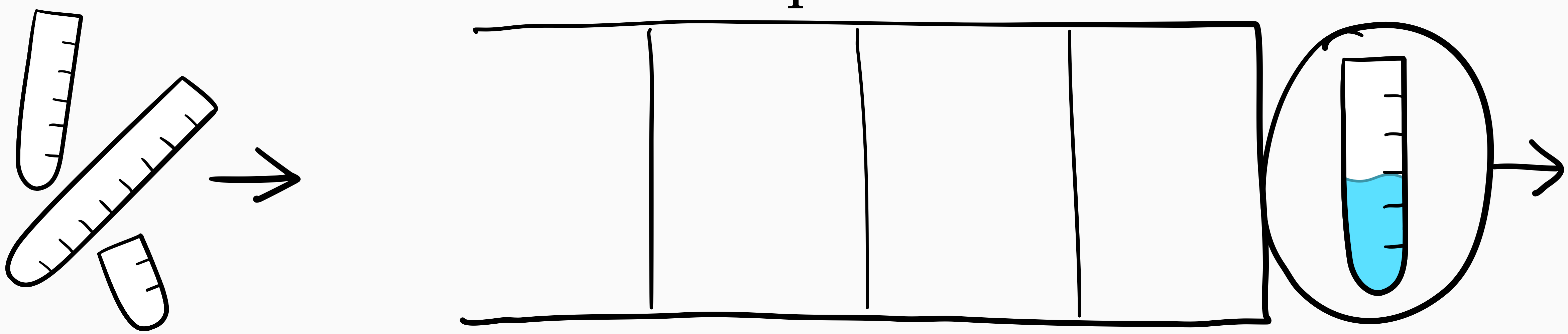
Queue scheduling (M/G/1)

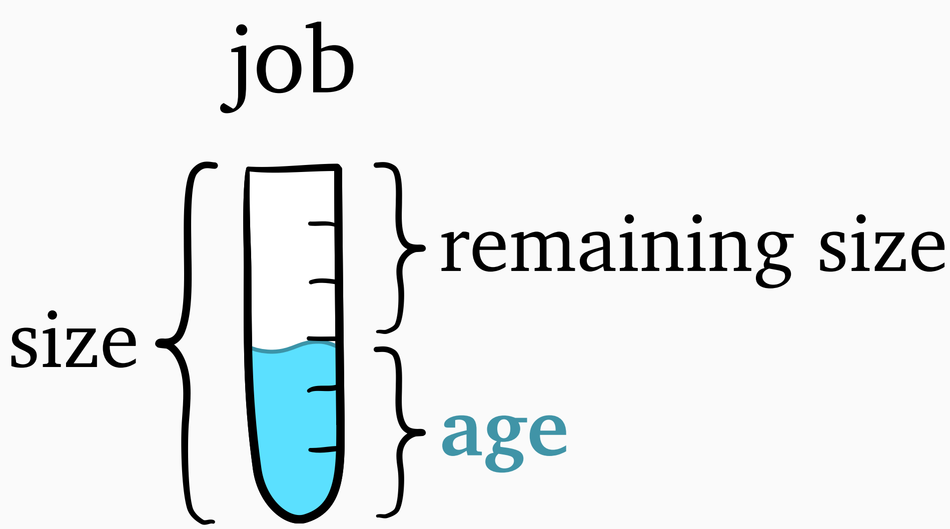




Queue scheduling (M/G/1)

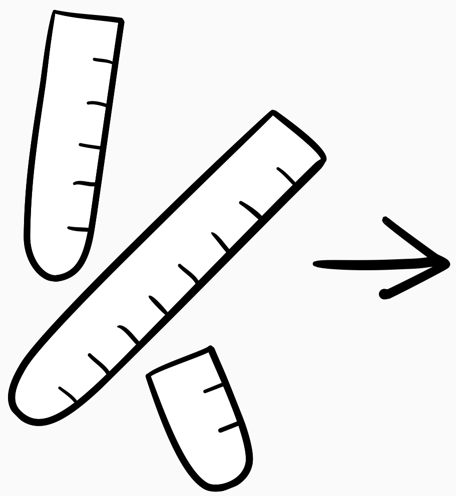
S = size distribution

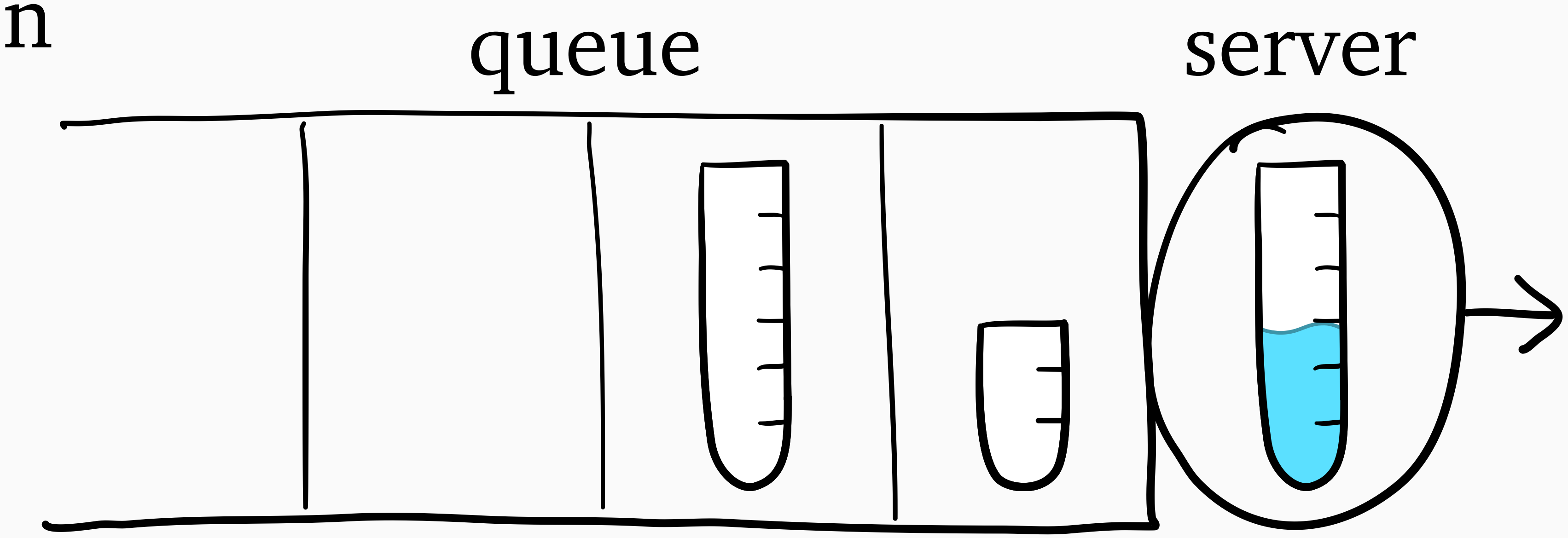


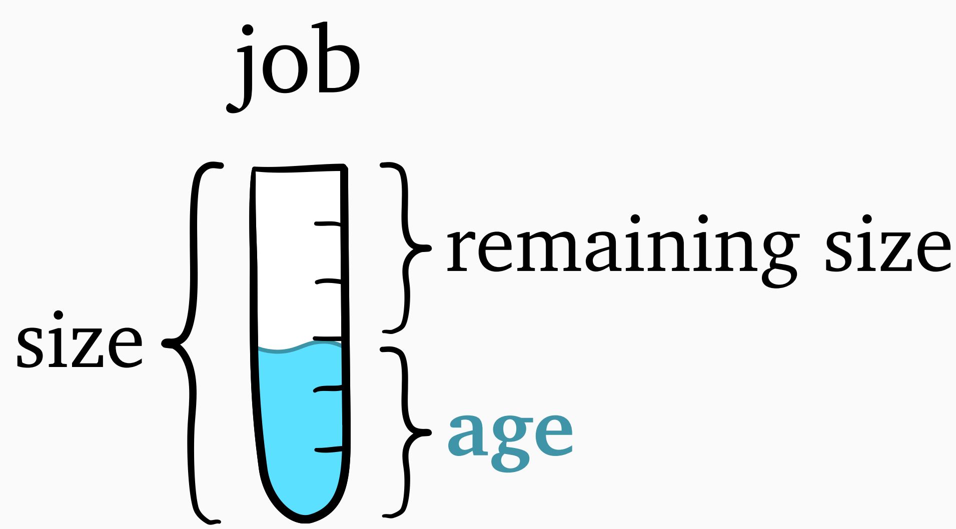


Queue scheduling (M/G/1)

S = size distribution

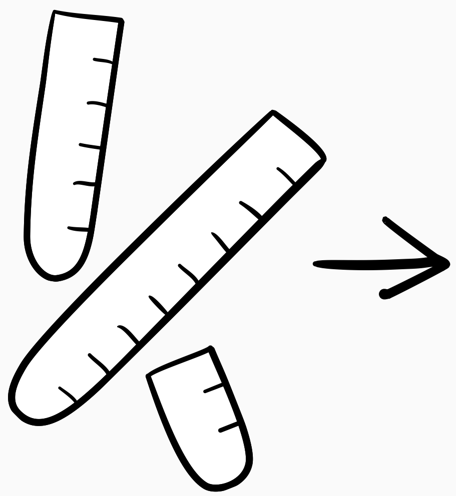


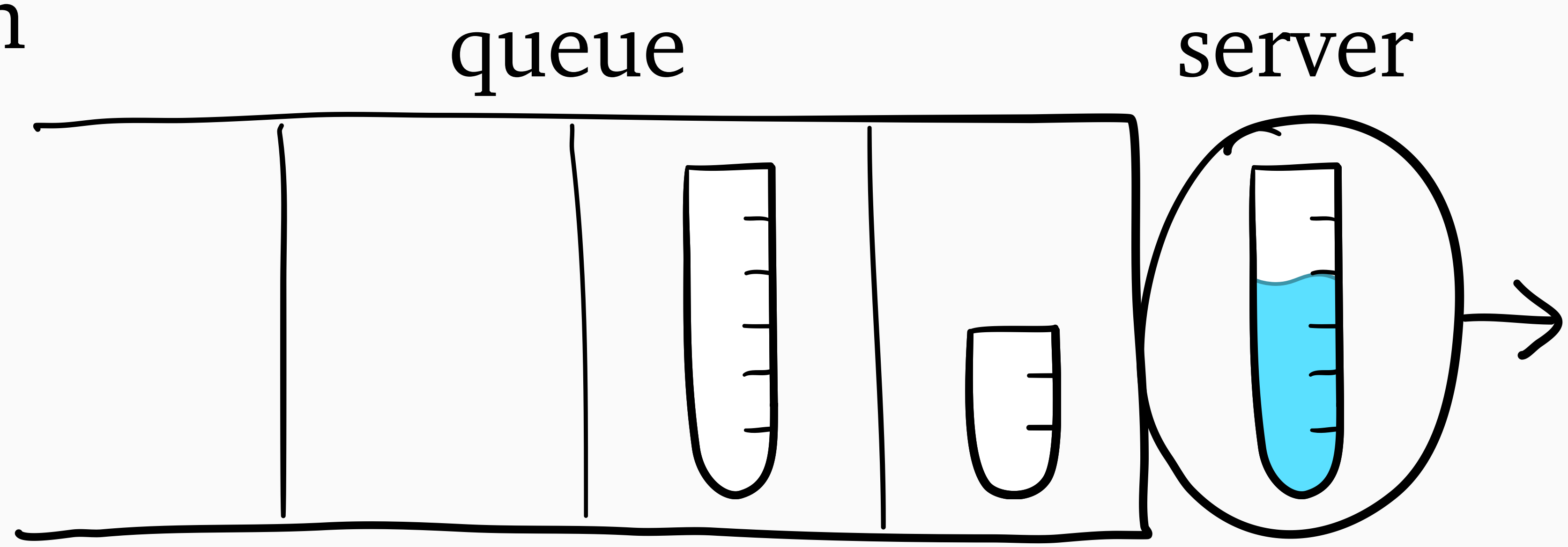


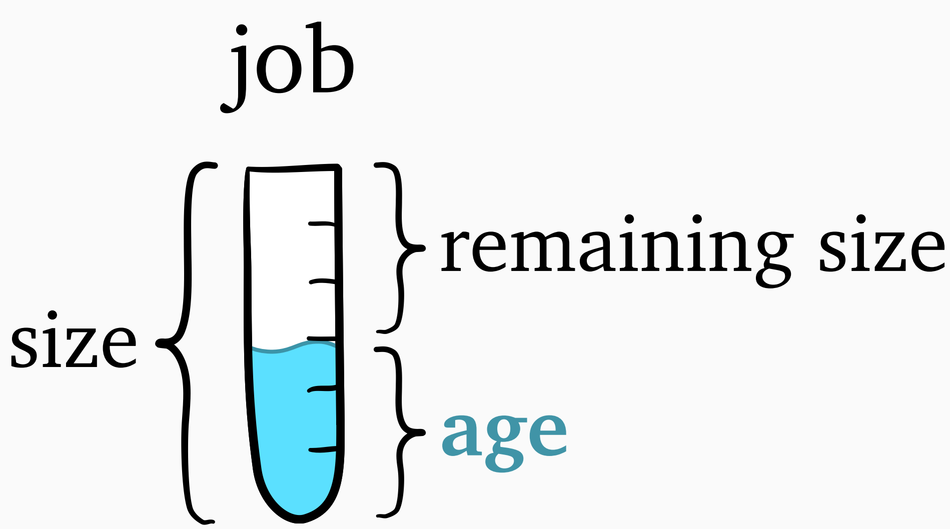


Queue scheduling (M/G/1)

S = size distribution

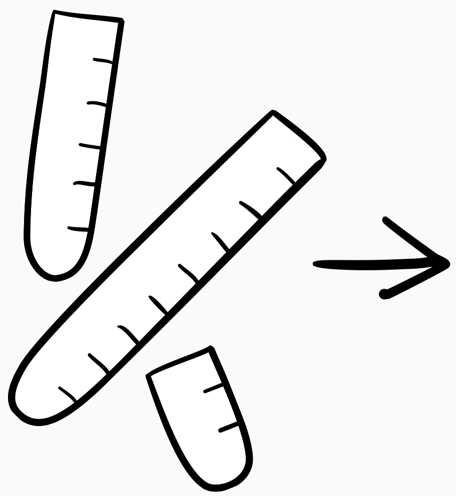


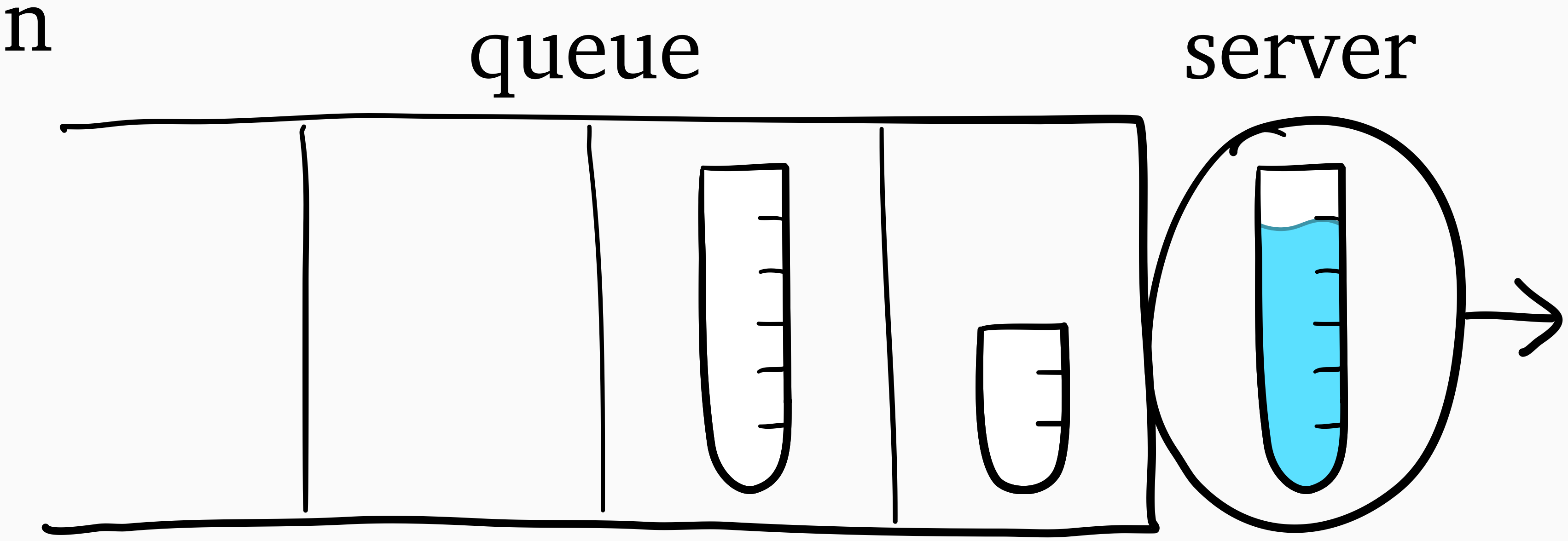


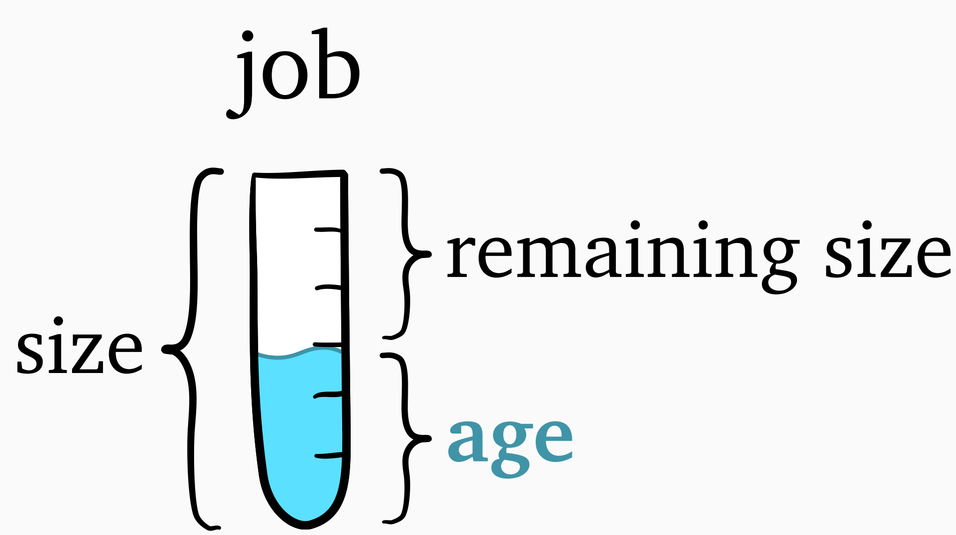


Queue scheduling (M/G/1)

S = size distribution

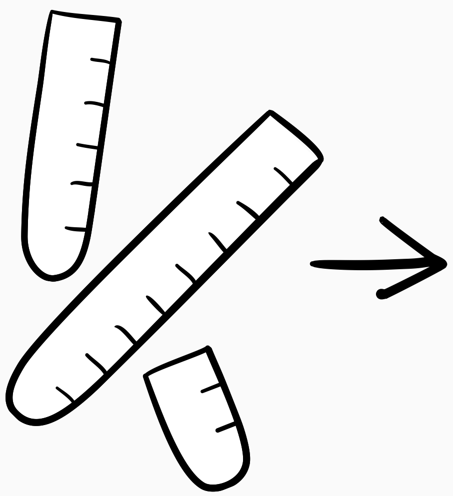


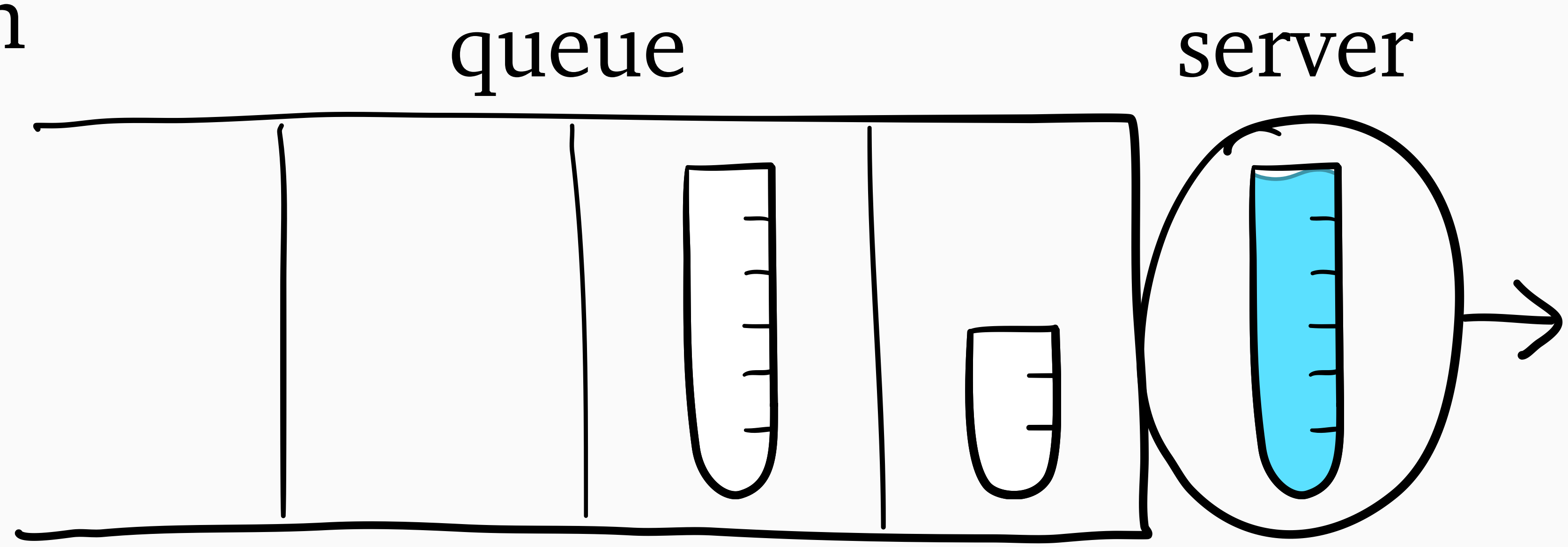


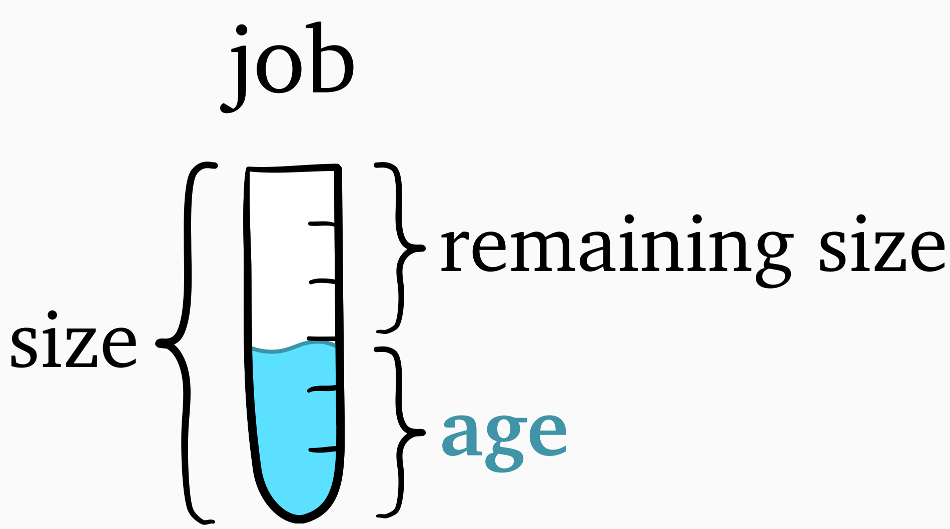


Queue scheduling (M/G/1)

S = size distribution

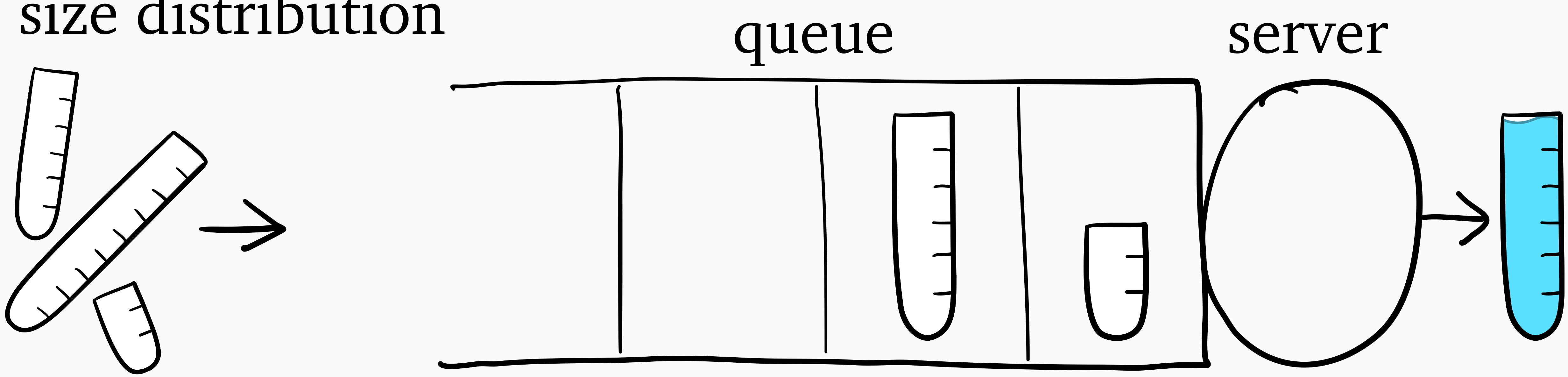


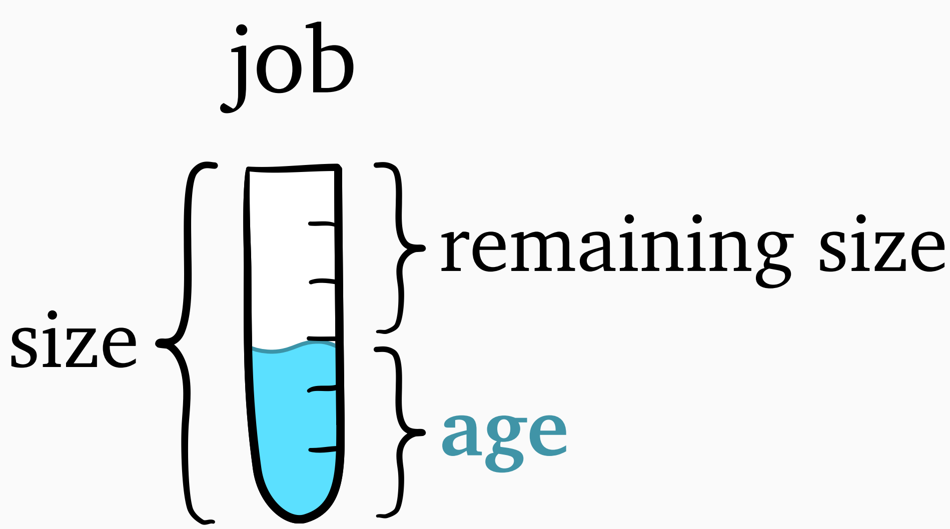




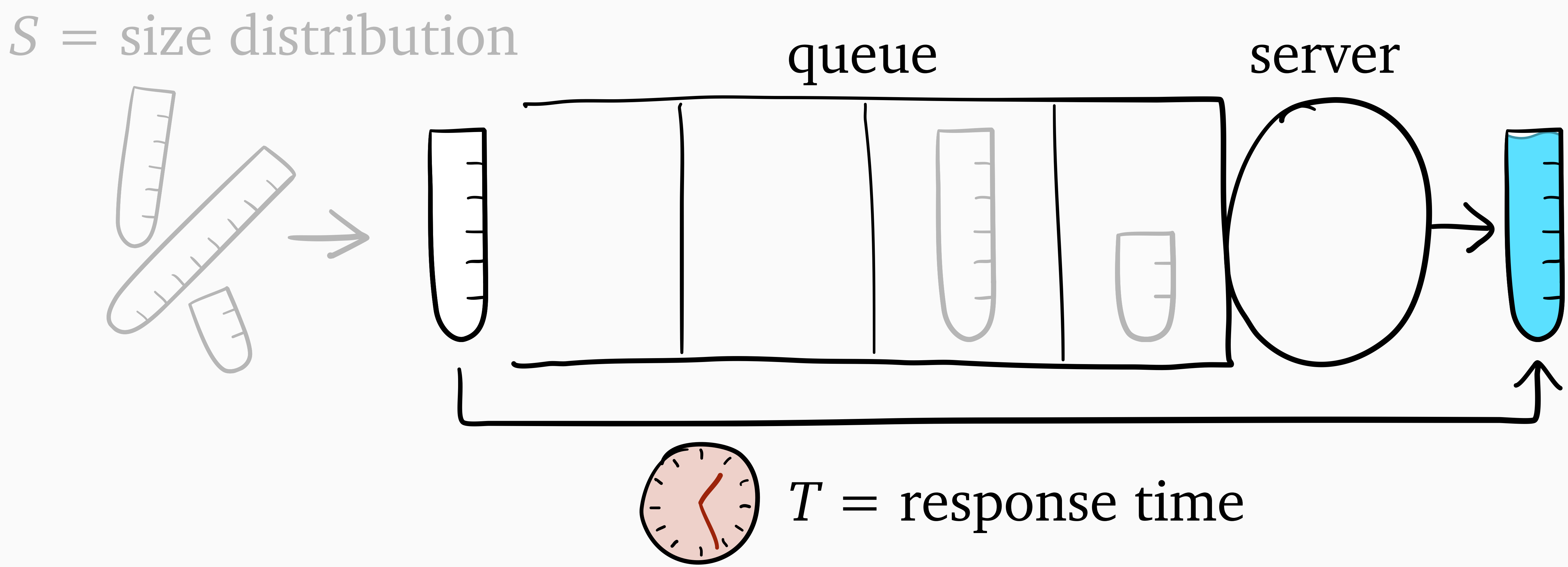
Queue scheduling (M/G/1)

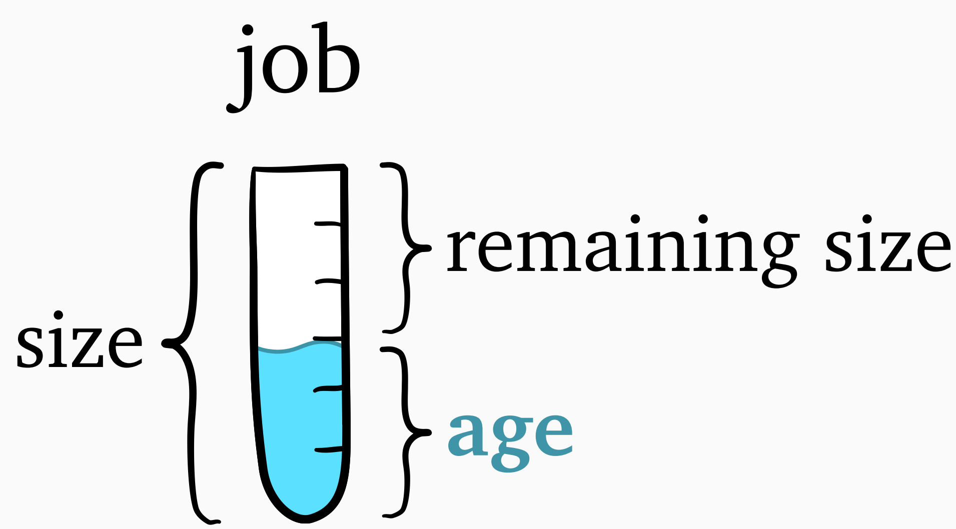
S = size distribution



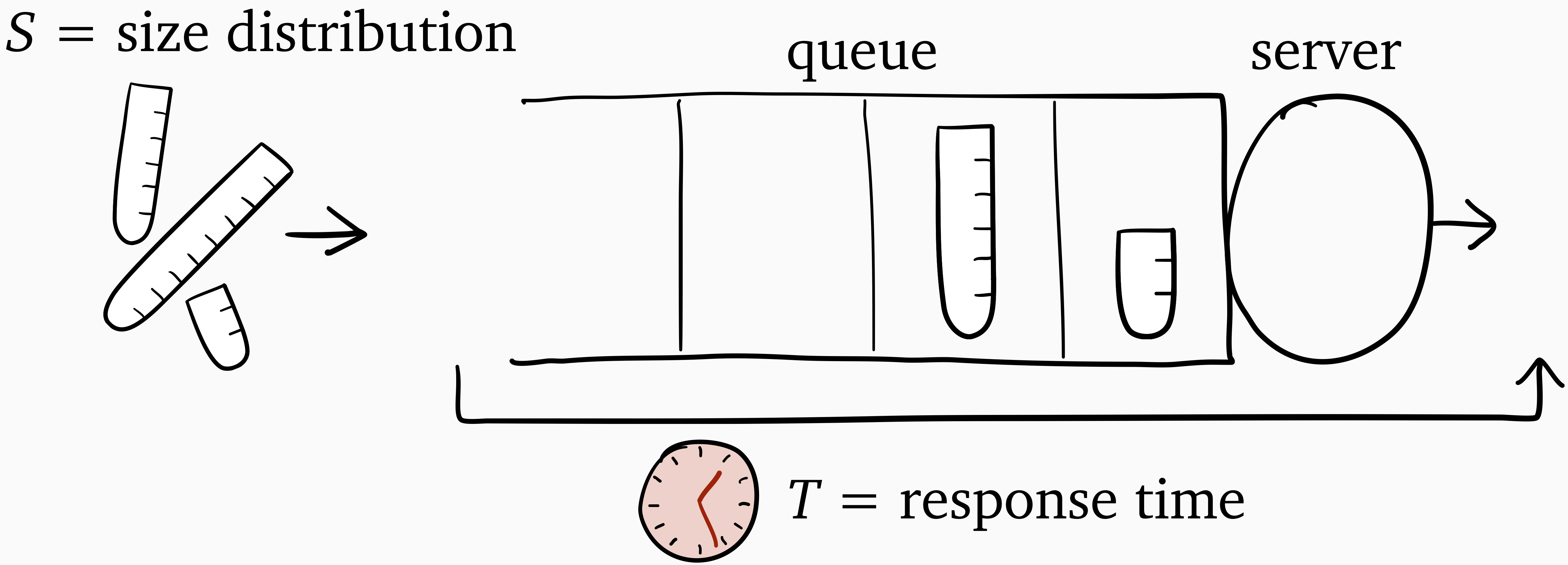


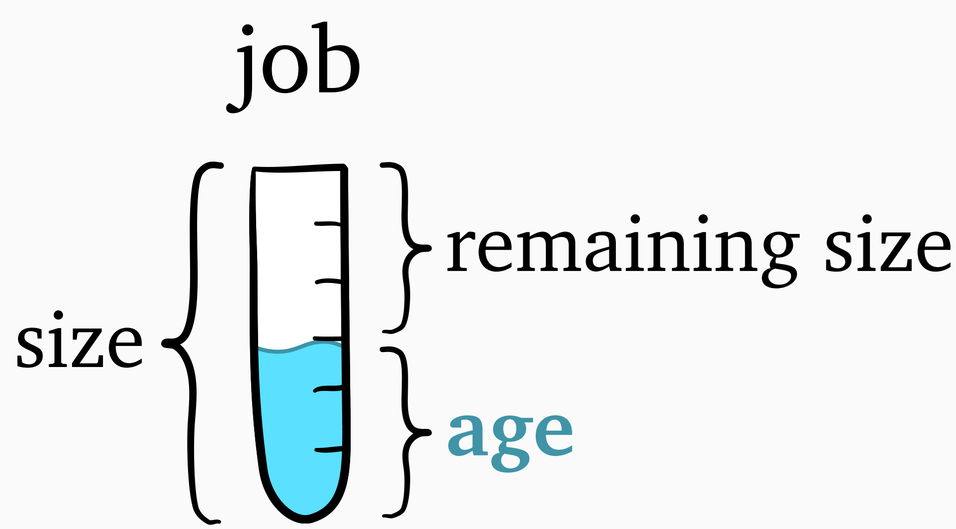
Queue scheduling (M/G/1)





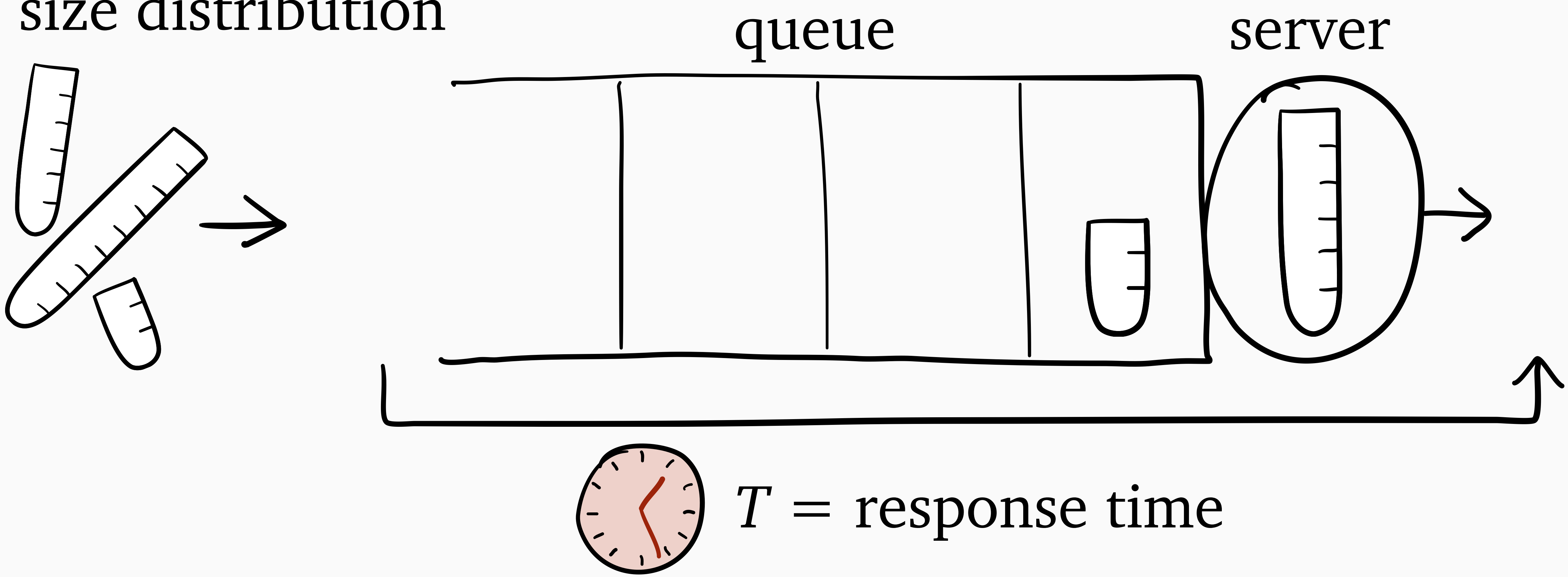
Queue scheduling (M/G/1)





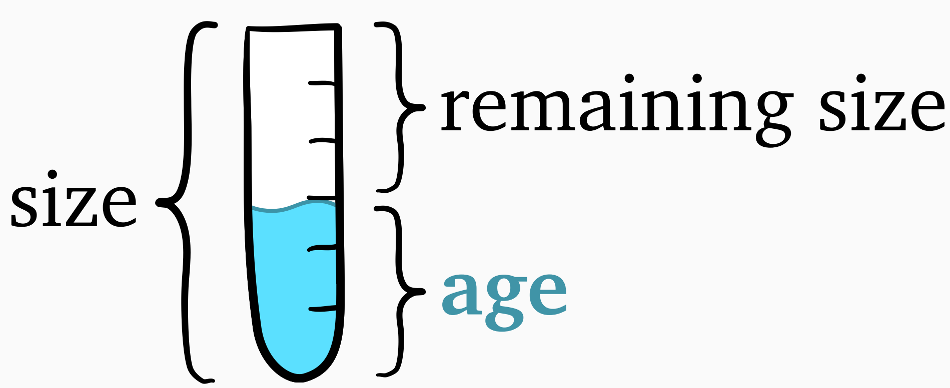
Queue scheduling (M/G/1)

S = size distribution

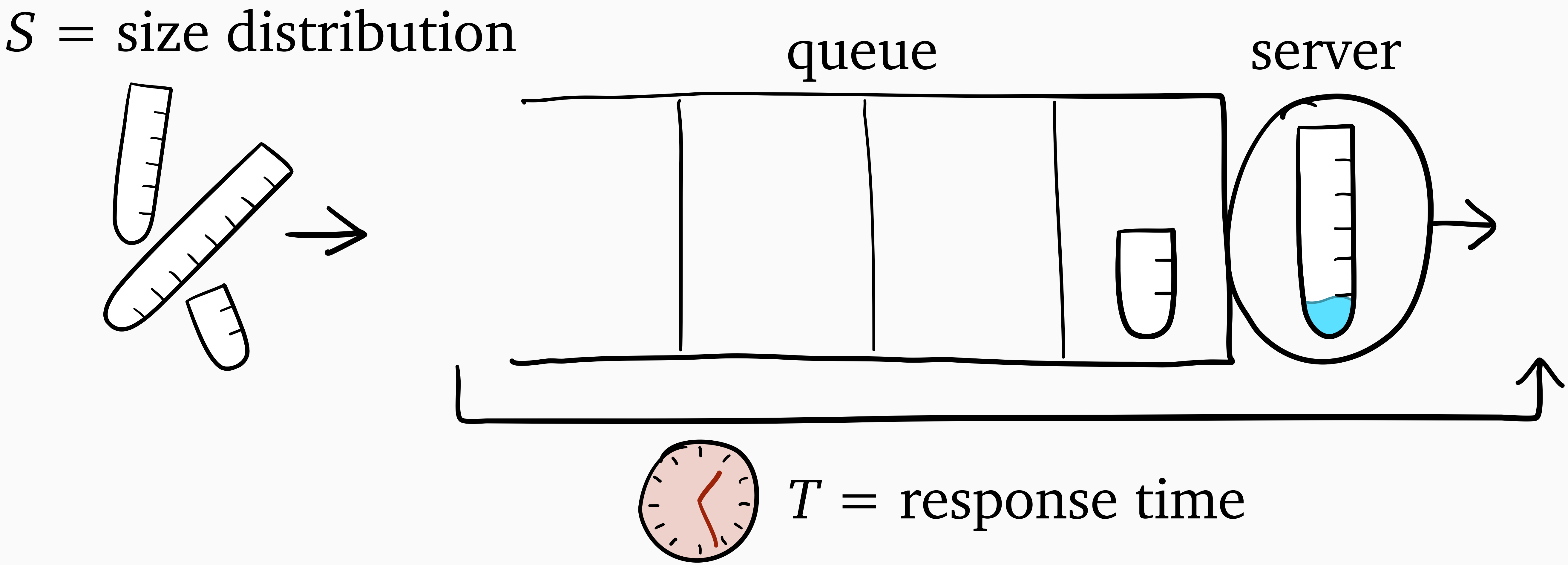


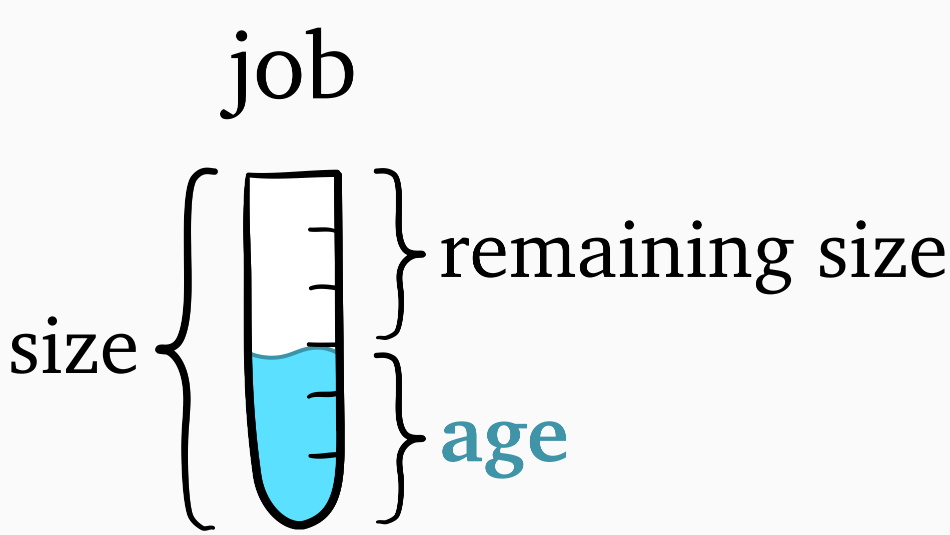
T = response time

job



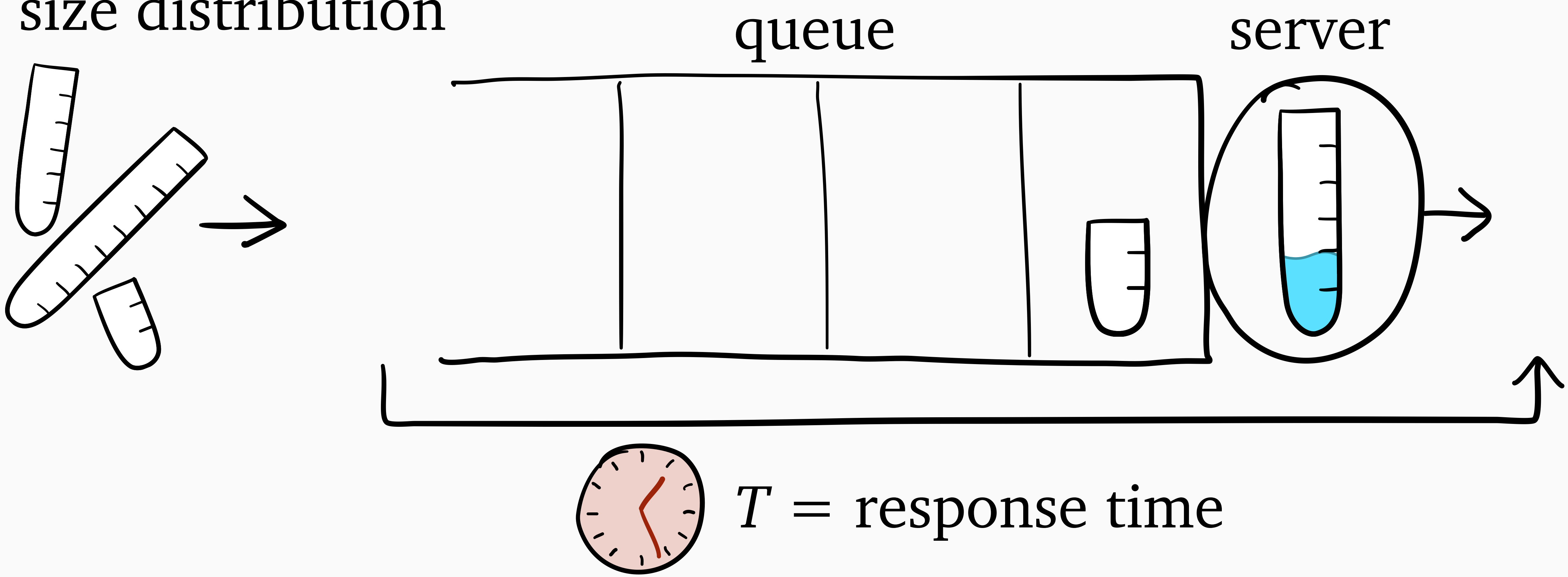
Queue scheduling (M/G/1)



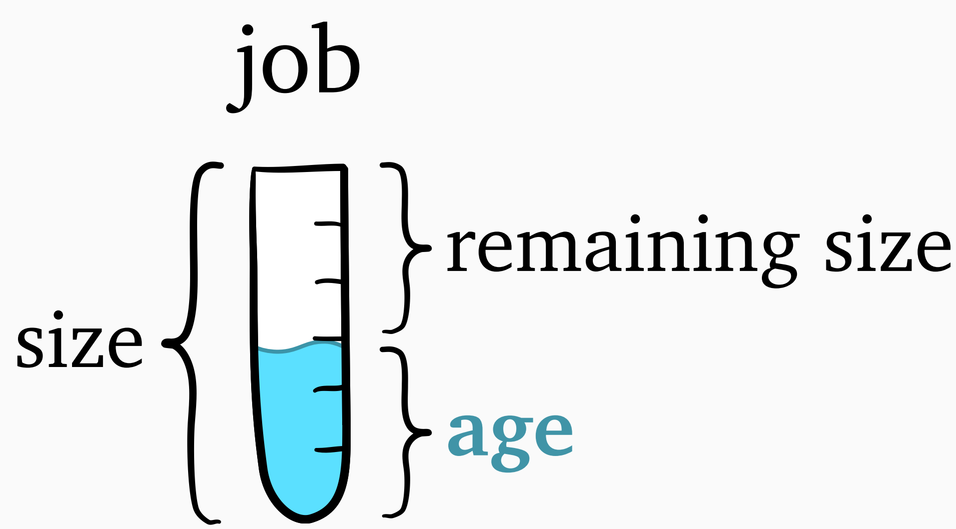


Queue scheduling (M/G/1)

S = size distribution

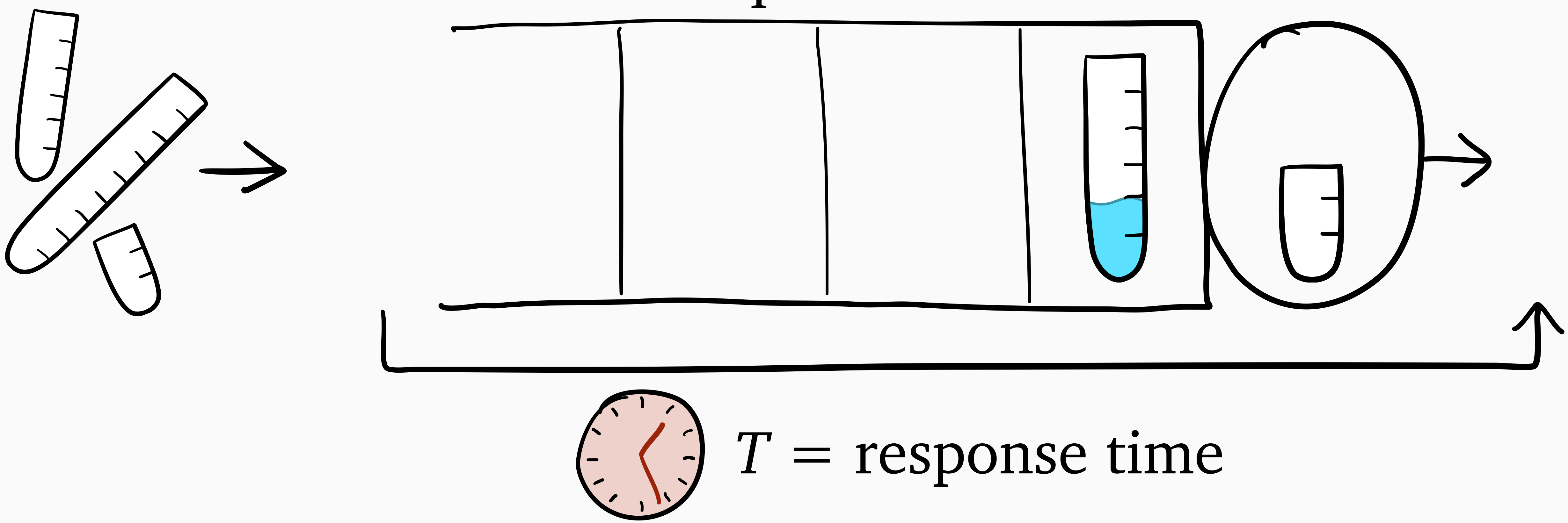


T = response time



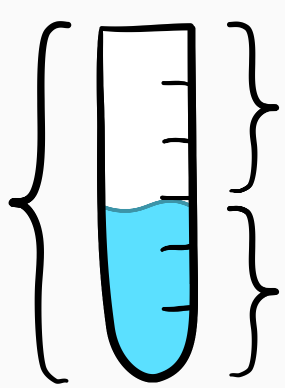
Queue scheduling (M/G/1)

S = size distribution



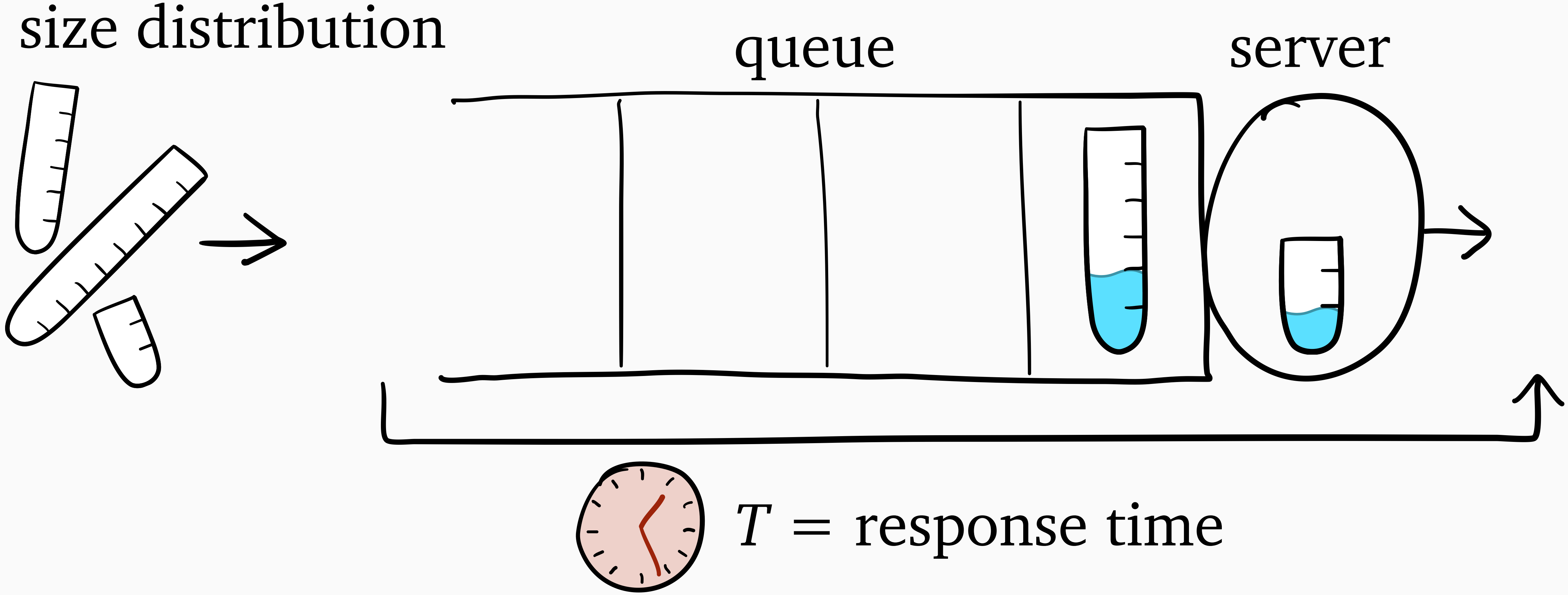
T = response time

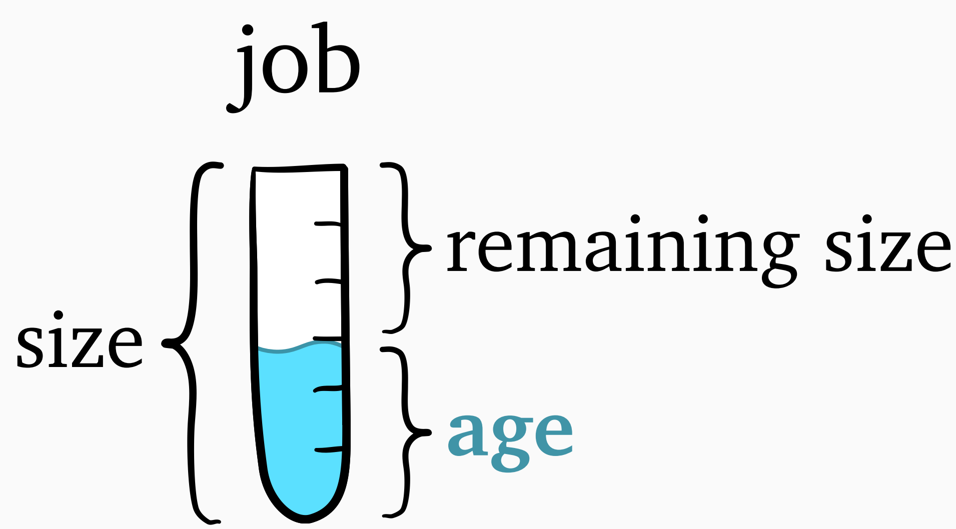
job



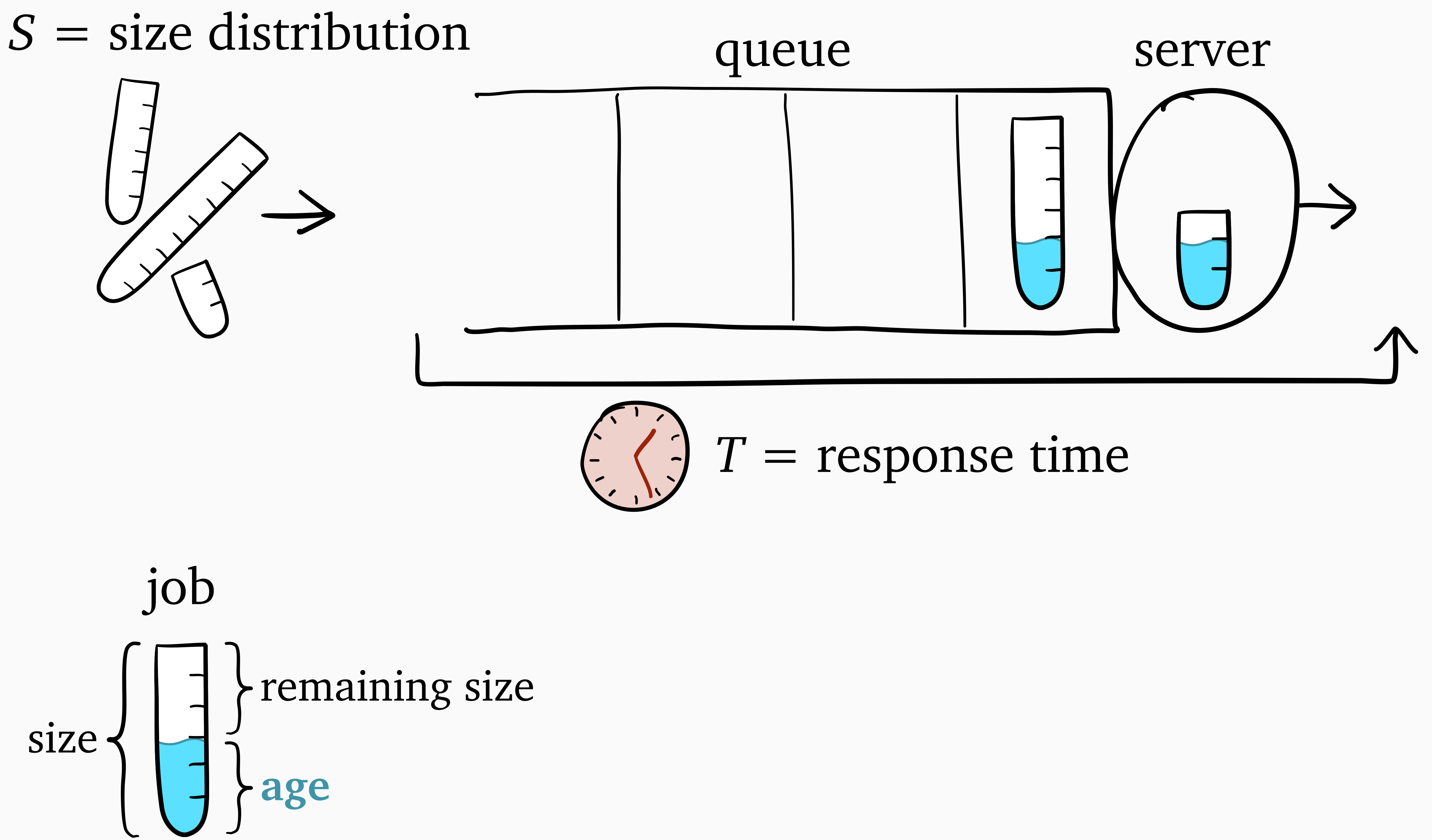
Queue scheduling (M/G/1)

S = size distribution



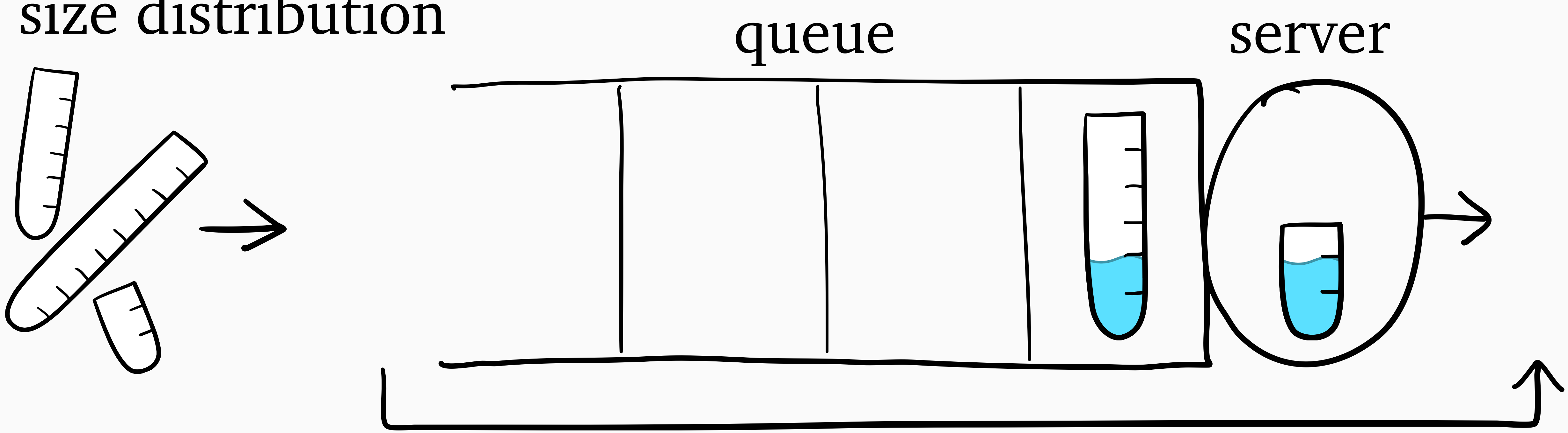


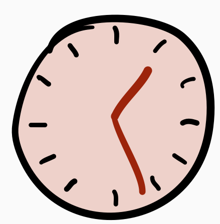
Queue scheduling (M/G/1)



Queue scheduling (M/G/1)

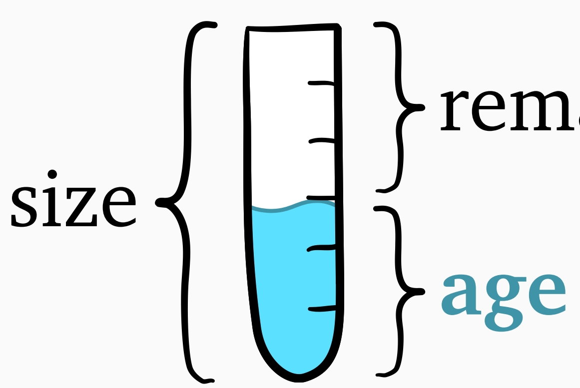
S = size distribution





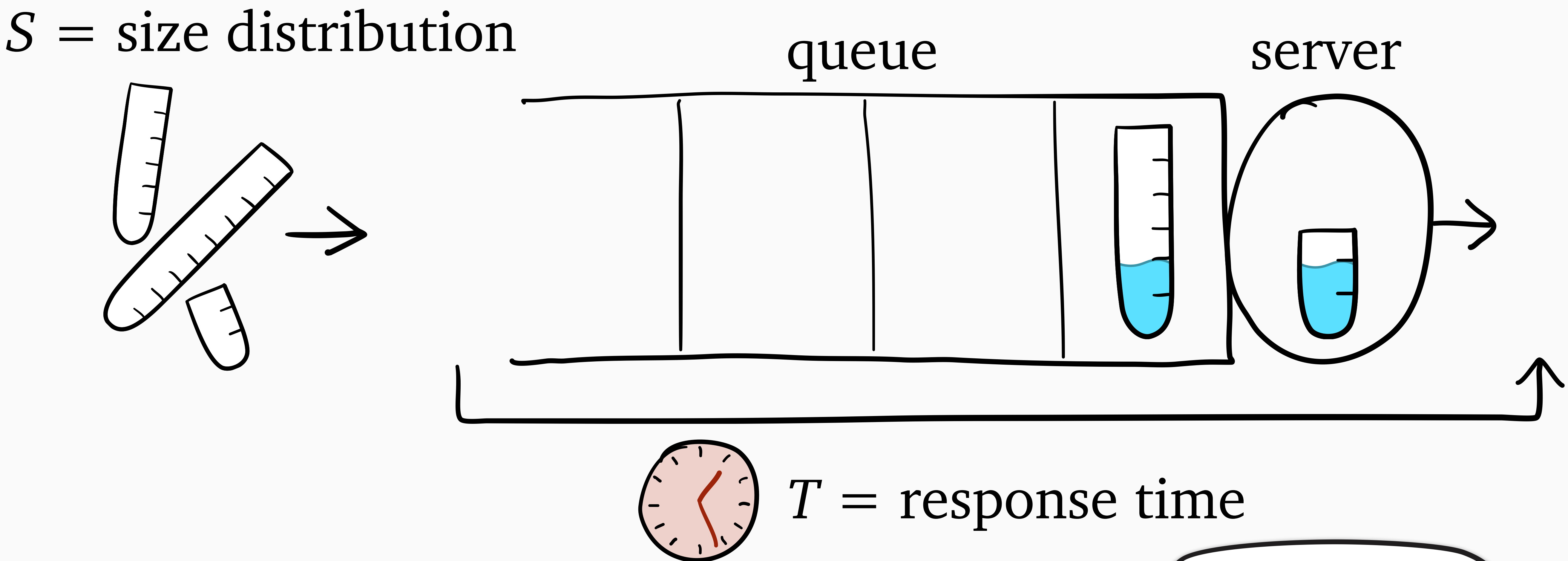
T = response time

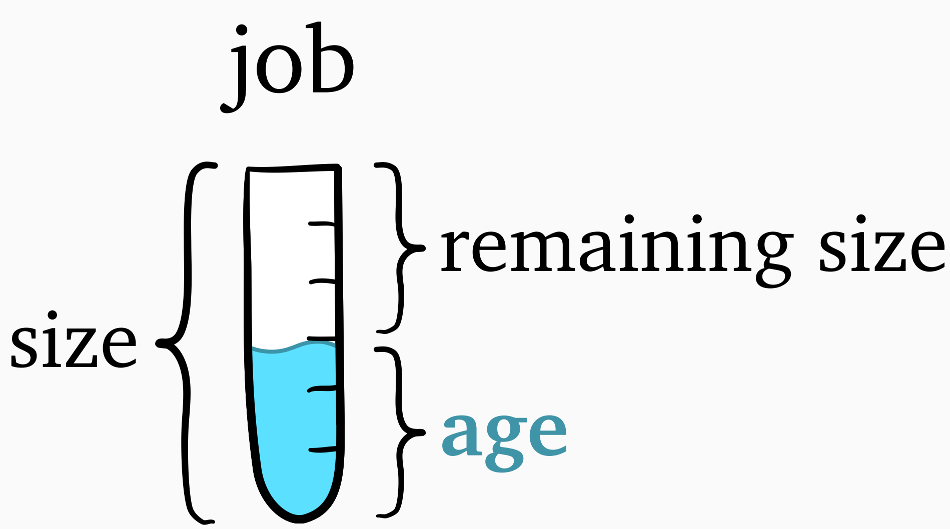
job



Question: schedule to minimize $\mathbf{E}[T]$?

Queue scheduling (M/G/1)

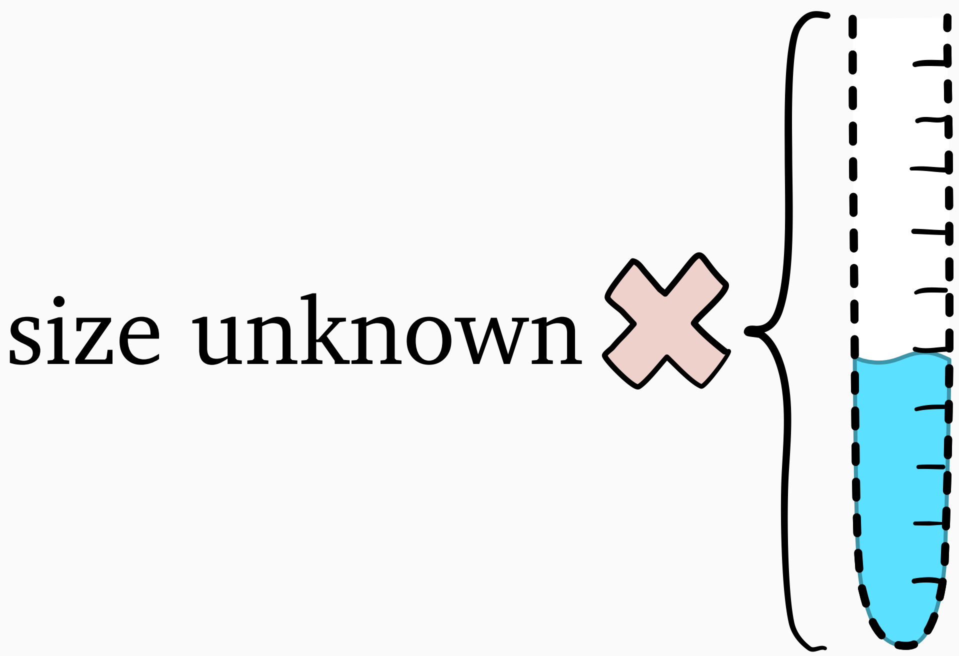




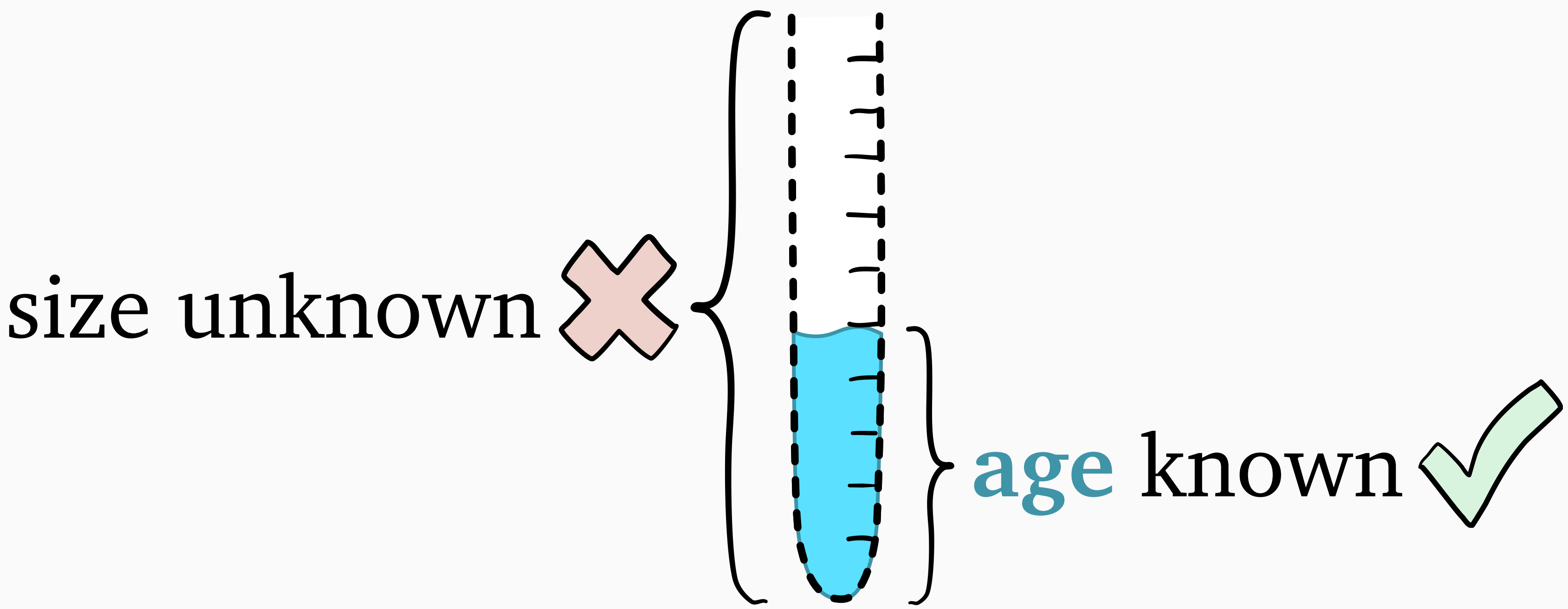
Question: schedule to minimize $\mathbf{E}[T]$?

Scheduling with unknown sizes

Scheduling with unknown sizes

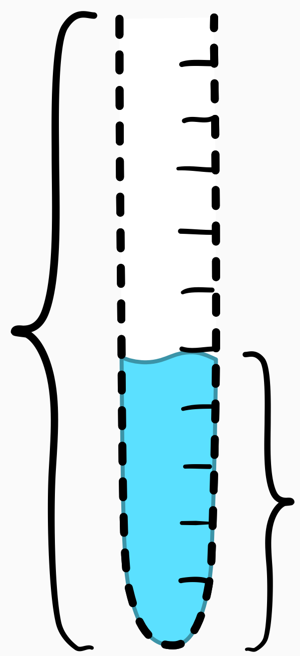


Scheduling with unknown sizes

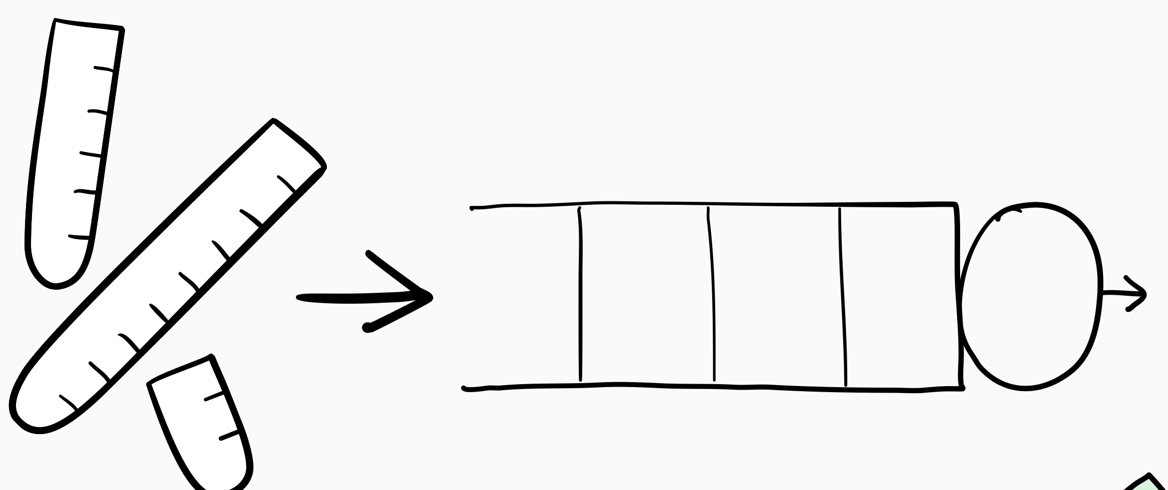


Scheduling with unknown sizes

size unknown ✗

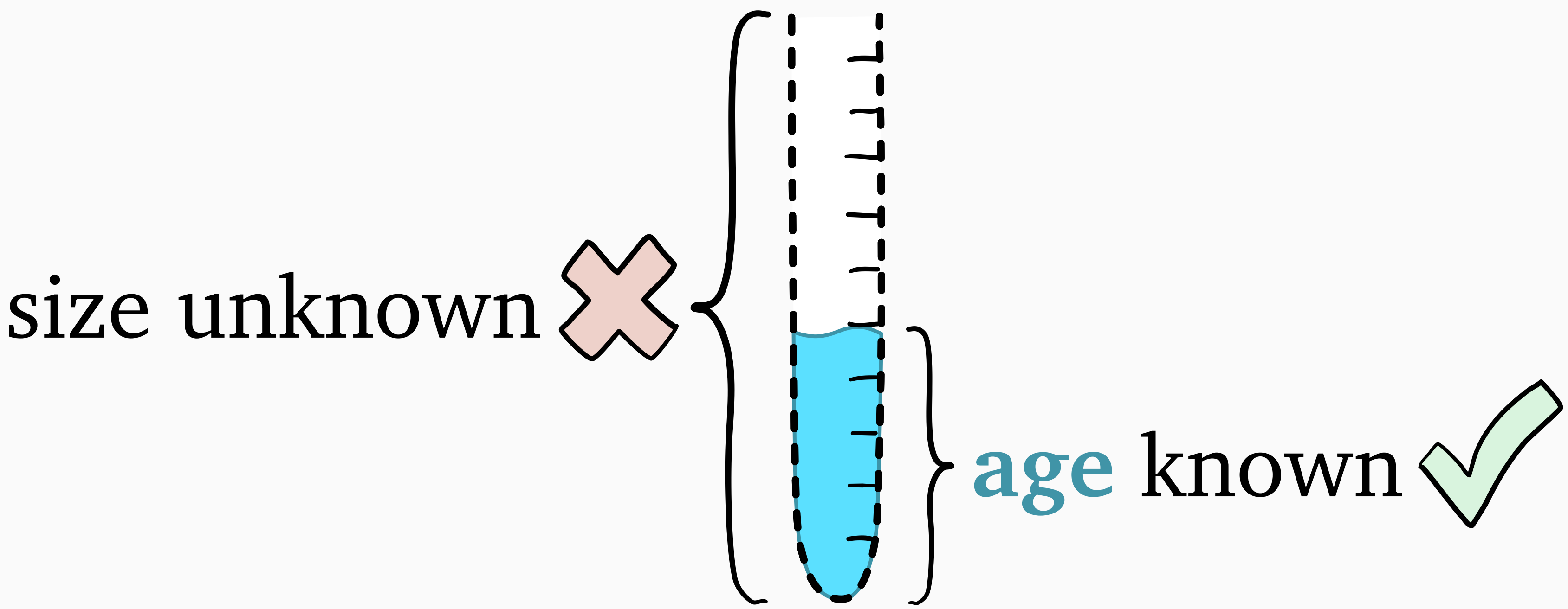


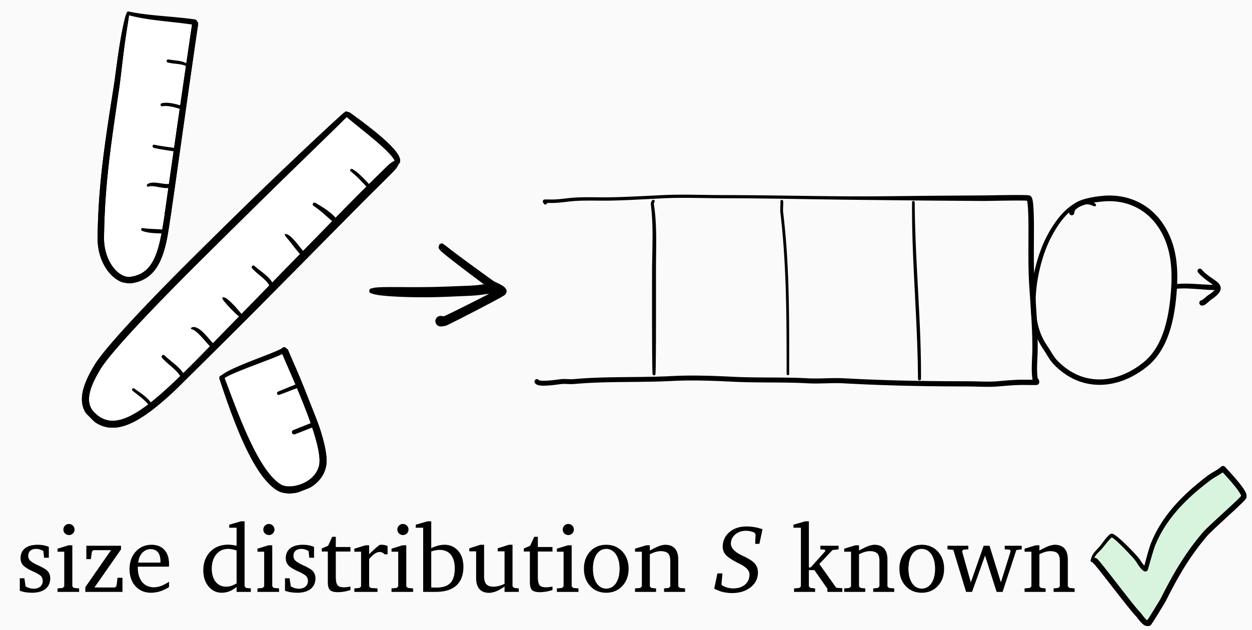
age known ✓



size distribution S known ✓

Scheduling with unknown sizes

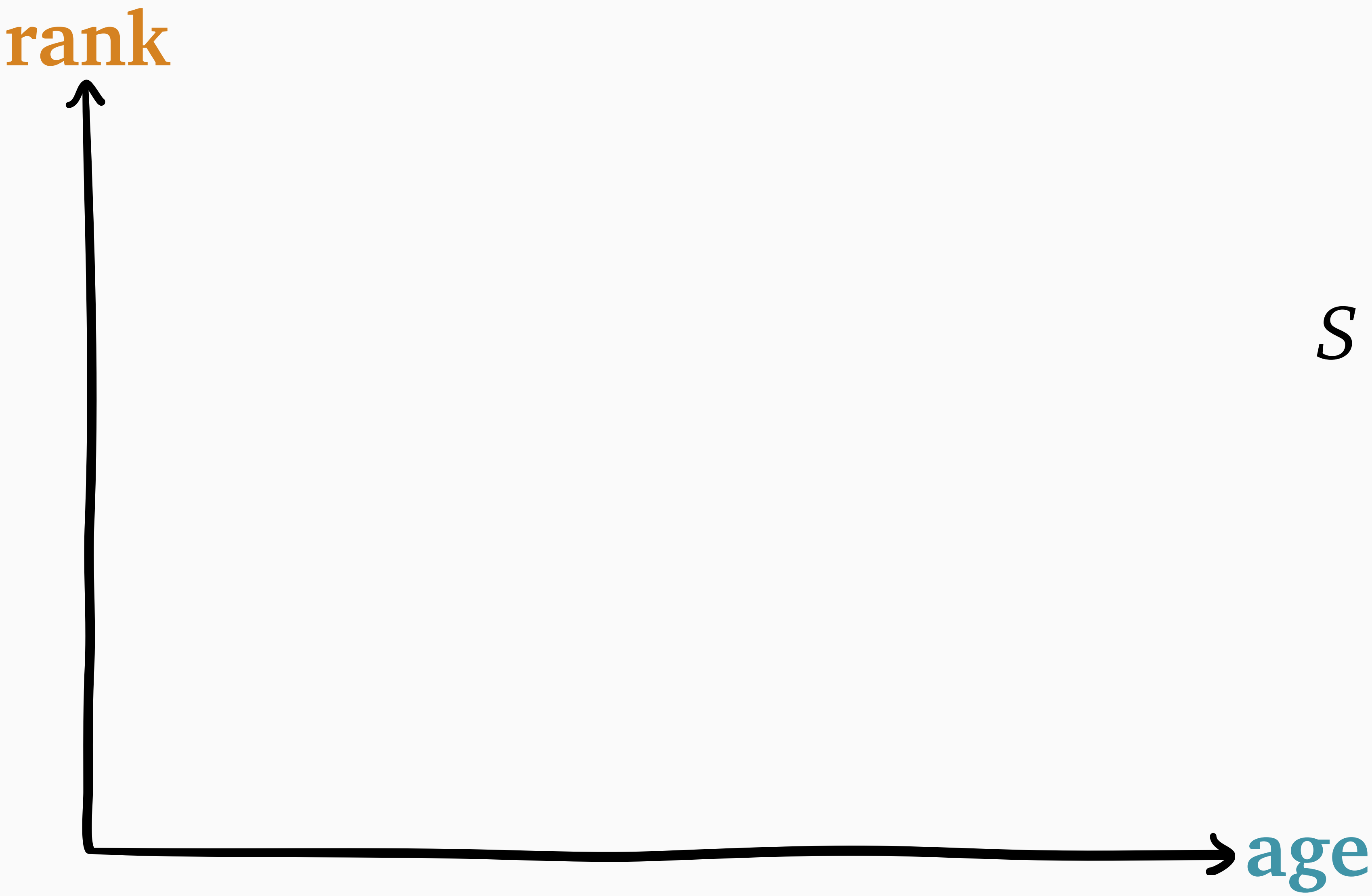




Example:

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

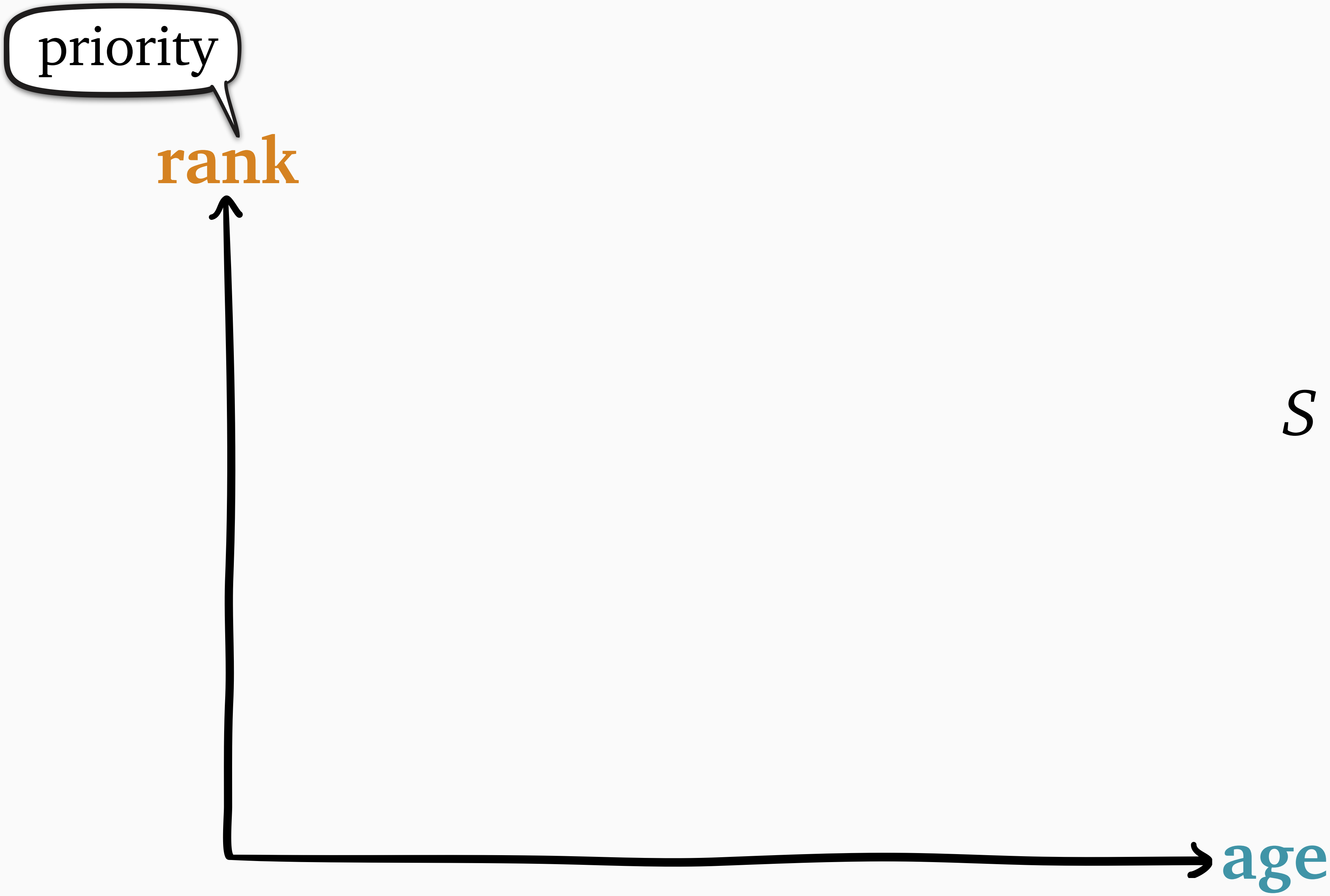
Scheduling with unknown sizes



Example:

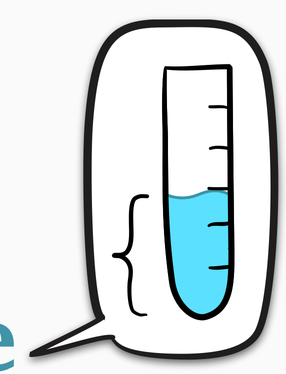
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

Scheduling with unknown sizes

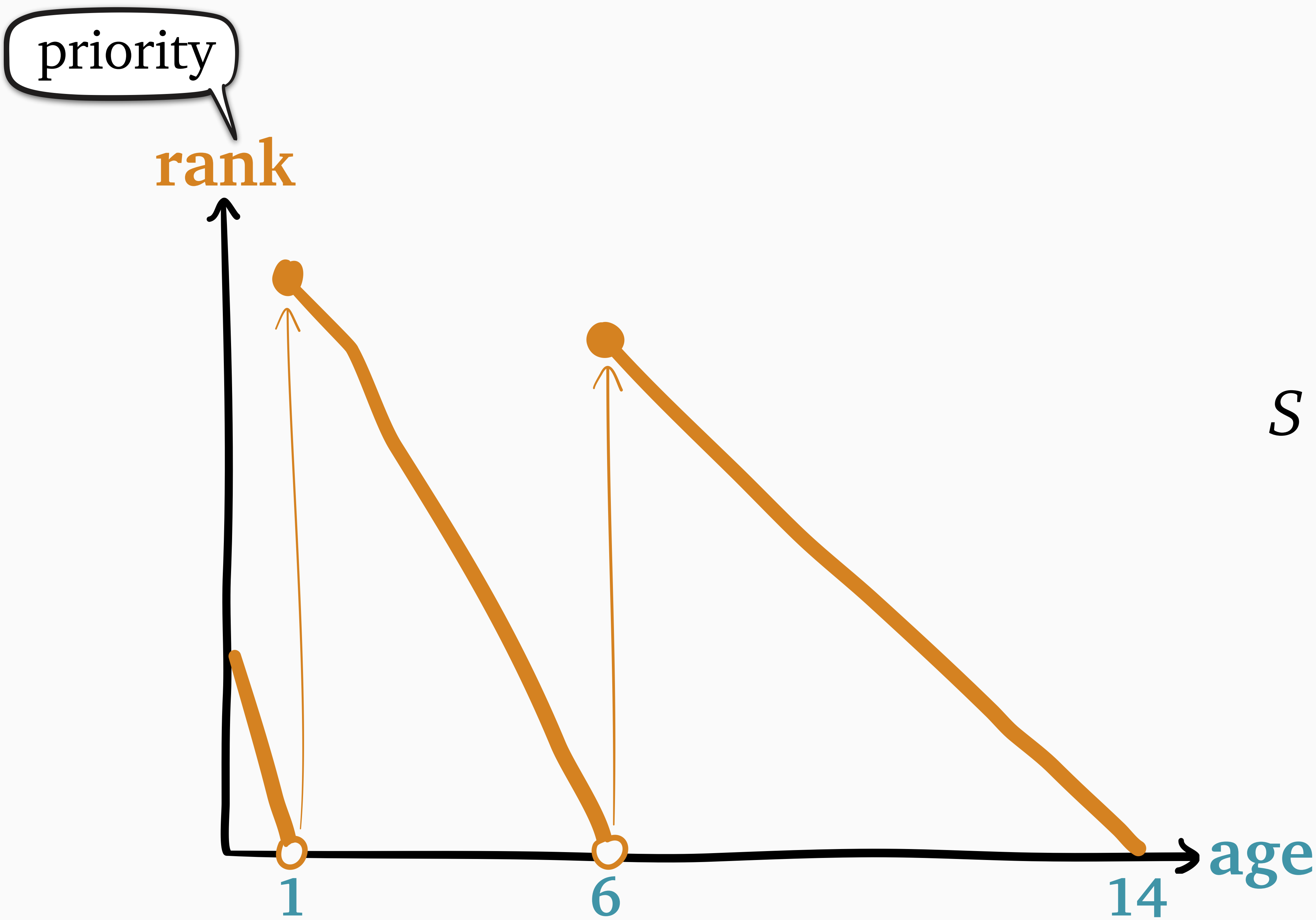


Example:

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

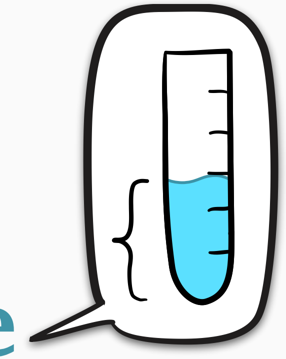


Scheduling with unknown sizes



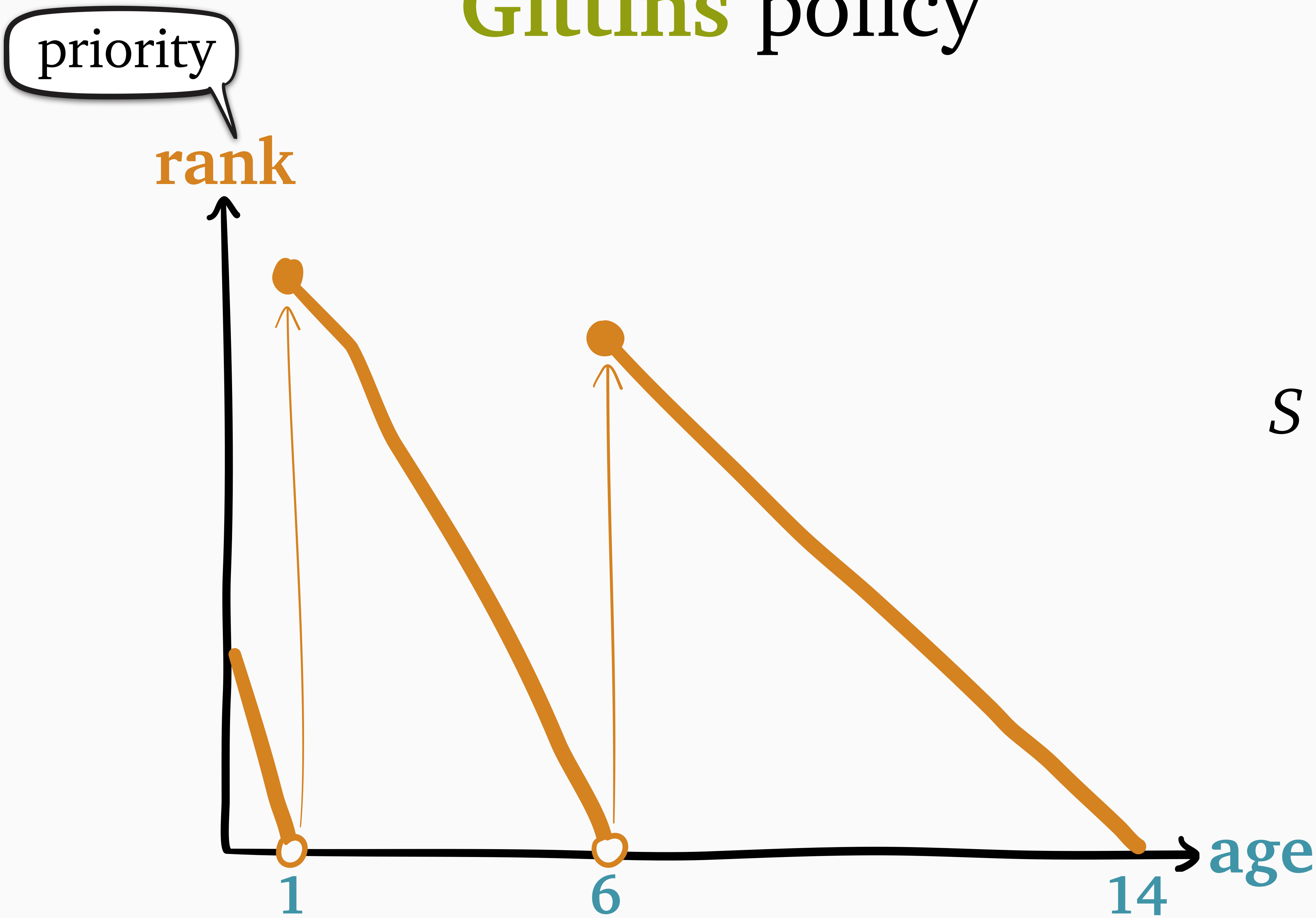
Example:

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



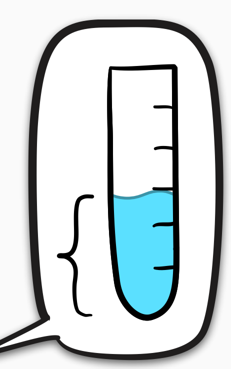
Scheduling with unknown sizes

Gittins policy



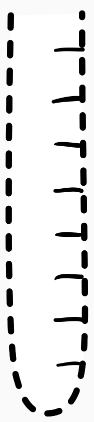
Example:

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

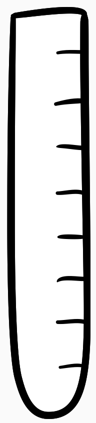


Gittins is general yet limited

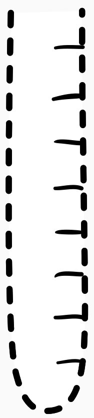
Gittins is general yet limited



Gittins is general yet limited

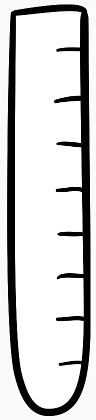


known
sizes

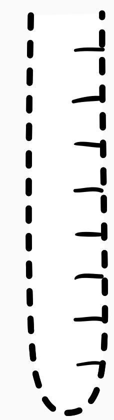


unknown
sizes

Gittins is general yet limited



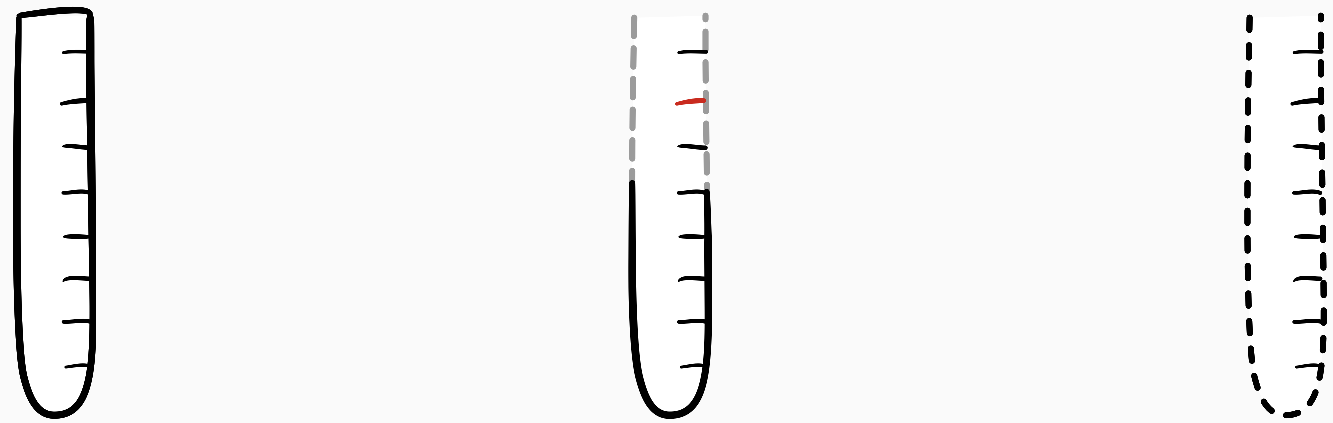
known sizes



unknown sizes

Gittins is optimal in M/G/1

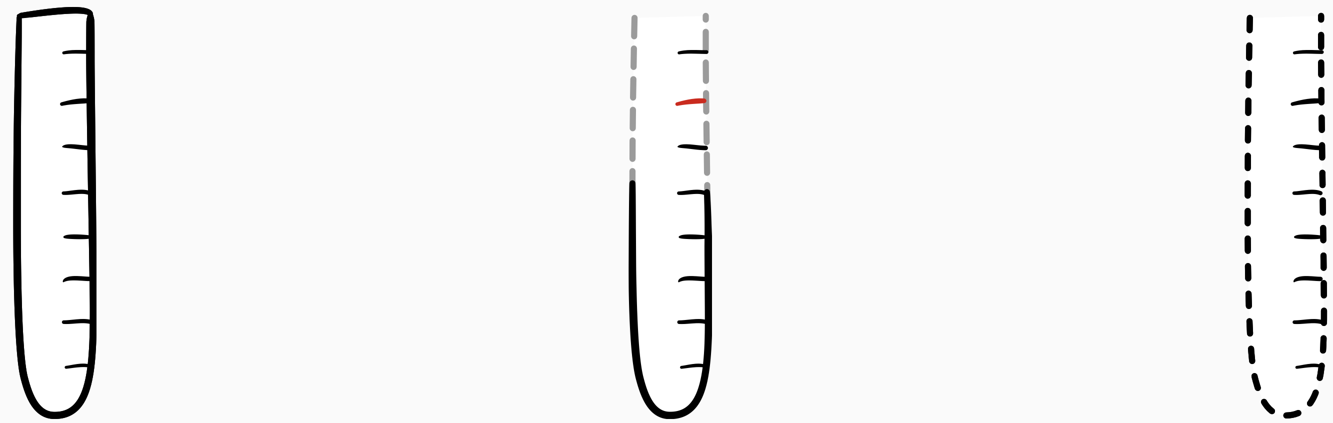
Gittins is general yet limited



known sizes | partially known sizes | unknown sizes

✓ Gittins is optimal in M/G/1

Gittins is general yet limited

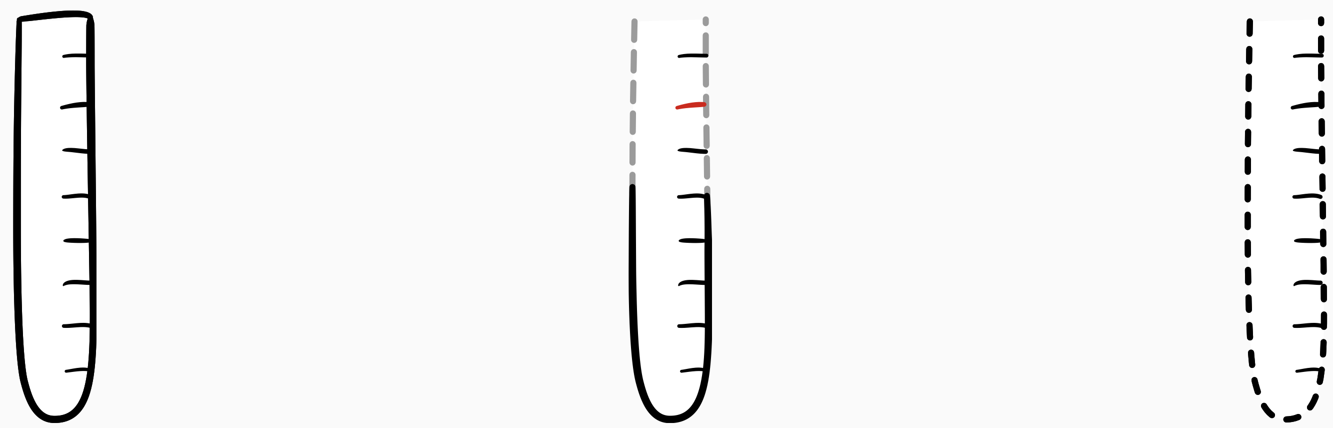


known sizes | partially known sizes | unknown sizes

✓ Gittins is optimal in M/G/1

? non-M/G/1 queues

Gittins is general yet limited



known sizes

partially known sizes

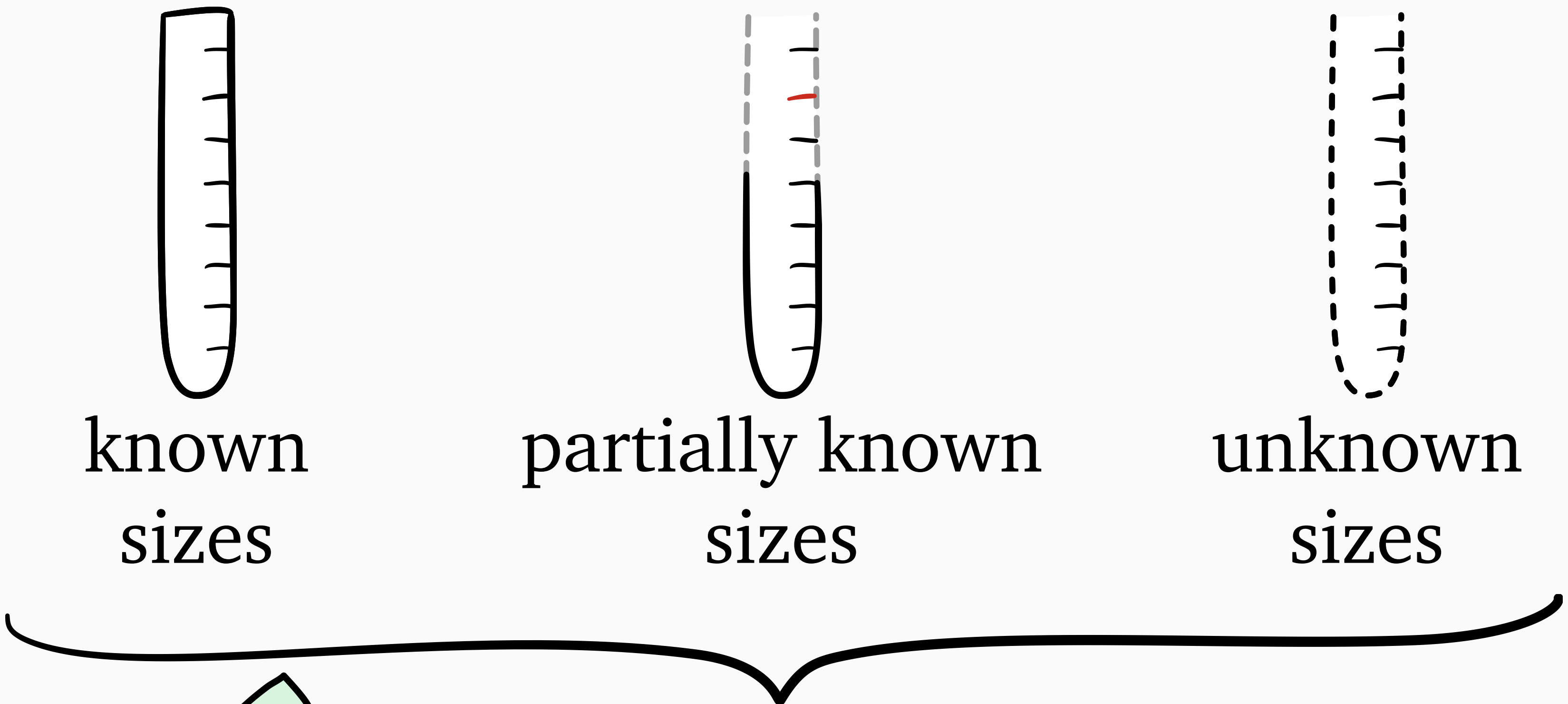
unknown sizes

✓ Gittins is optimal in M/G/1

? non-M/G/1 queues

? imperfect implementation

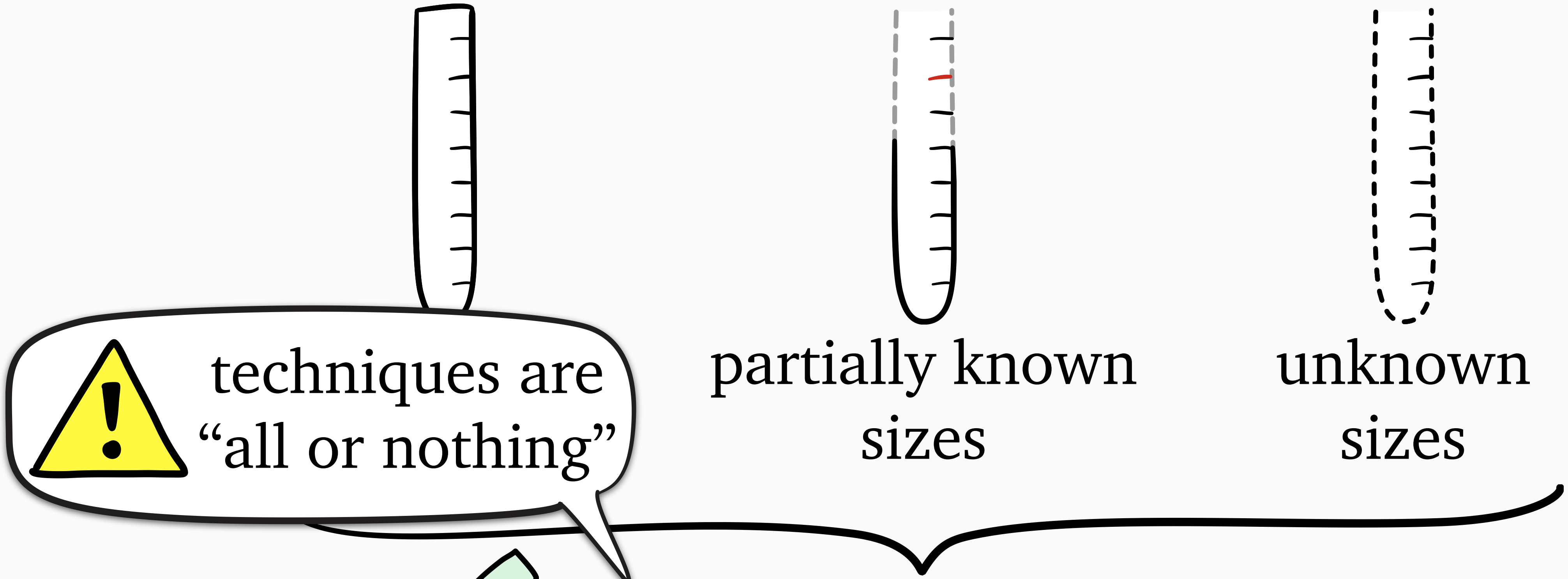
Gittins is general yet limited



Gittins is optimal in M/G/1

- non-M/G/1 queues
- imperfect implementation
- unknown job size distribution/model

Gittins is general yet limited



Gittins is optimal in M/G/1

- non-M/G/1 queues
- imperfect implementation
- unknown job size distribution/model

Contributions

Contributions

NEW!

WINE

queueing identity for understanding **Gittins**

Contributions

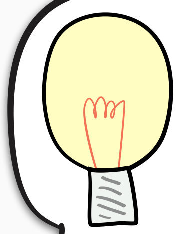
WINE

queueing identity for
understanding **Gittins**

applicable even in
suboptimal settings

Contributions

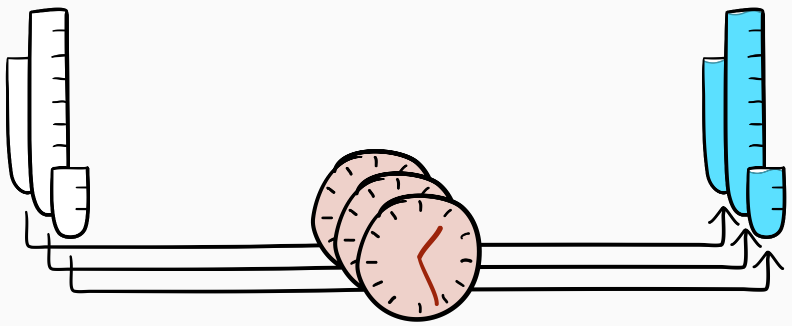
WINE



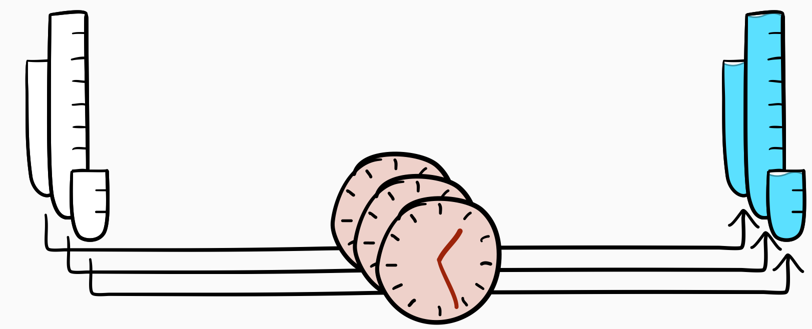
applicable even in suboptimal settings

queueing identity for understanding **Gittins**

- non-M/G/1 queues
- imperfect implementation
- unknown job size distribution/model



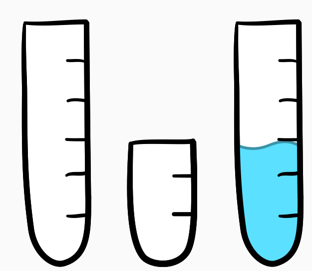
response time T



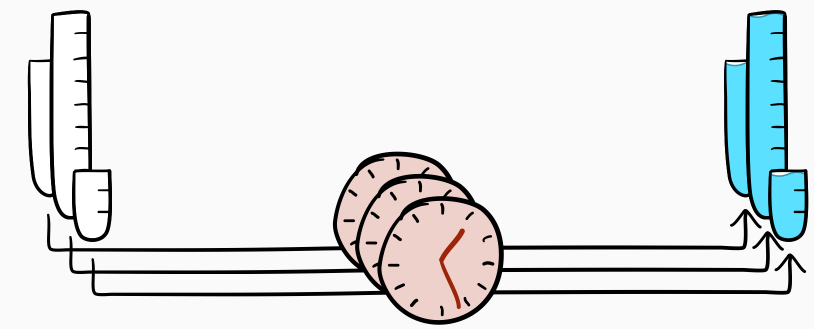
response time T

Little's law

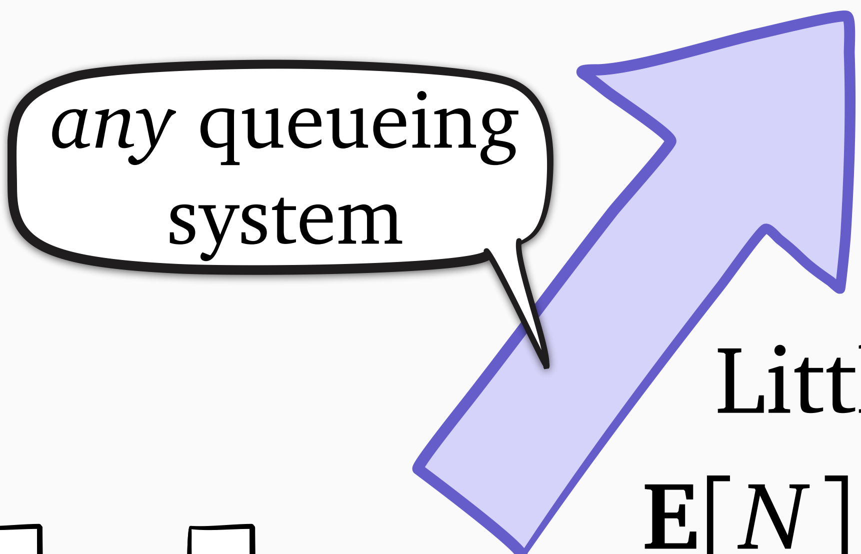
$$\mathbf{E}[N] = \lambda \mathbf{E}[T]$$



number of jobs N

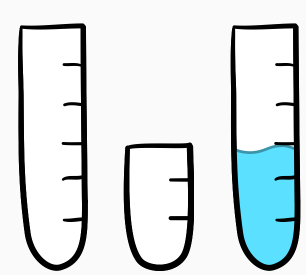


response time T

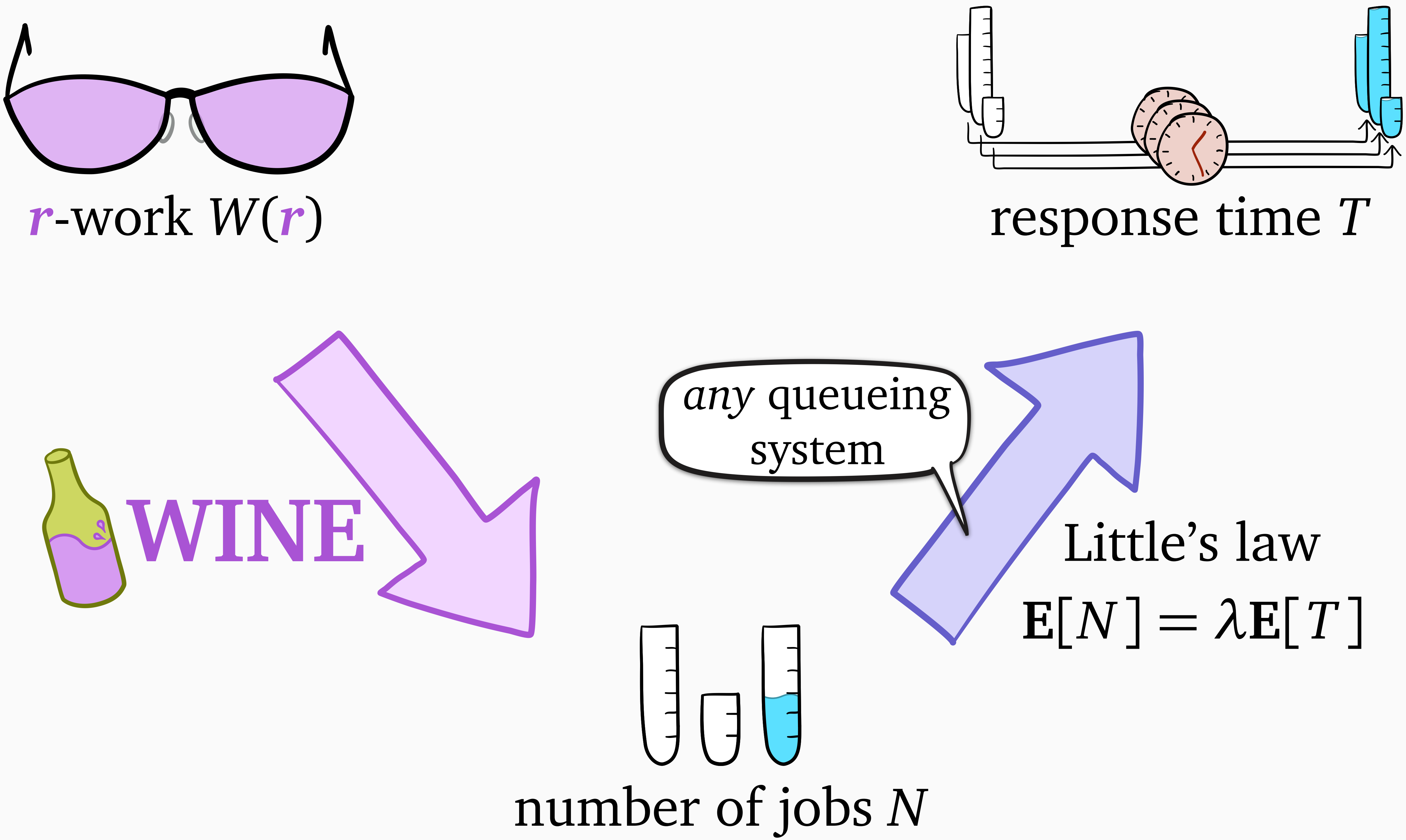


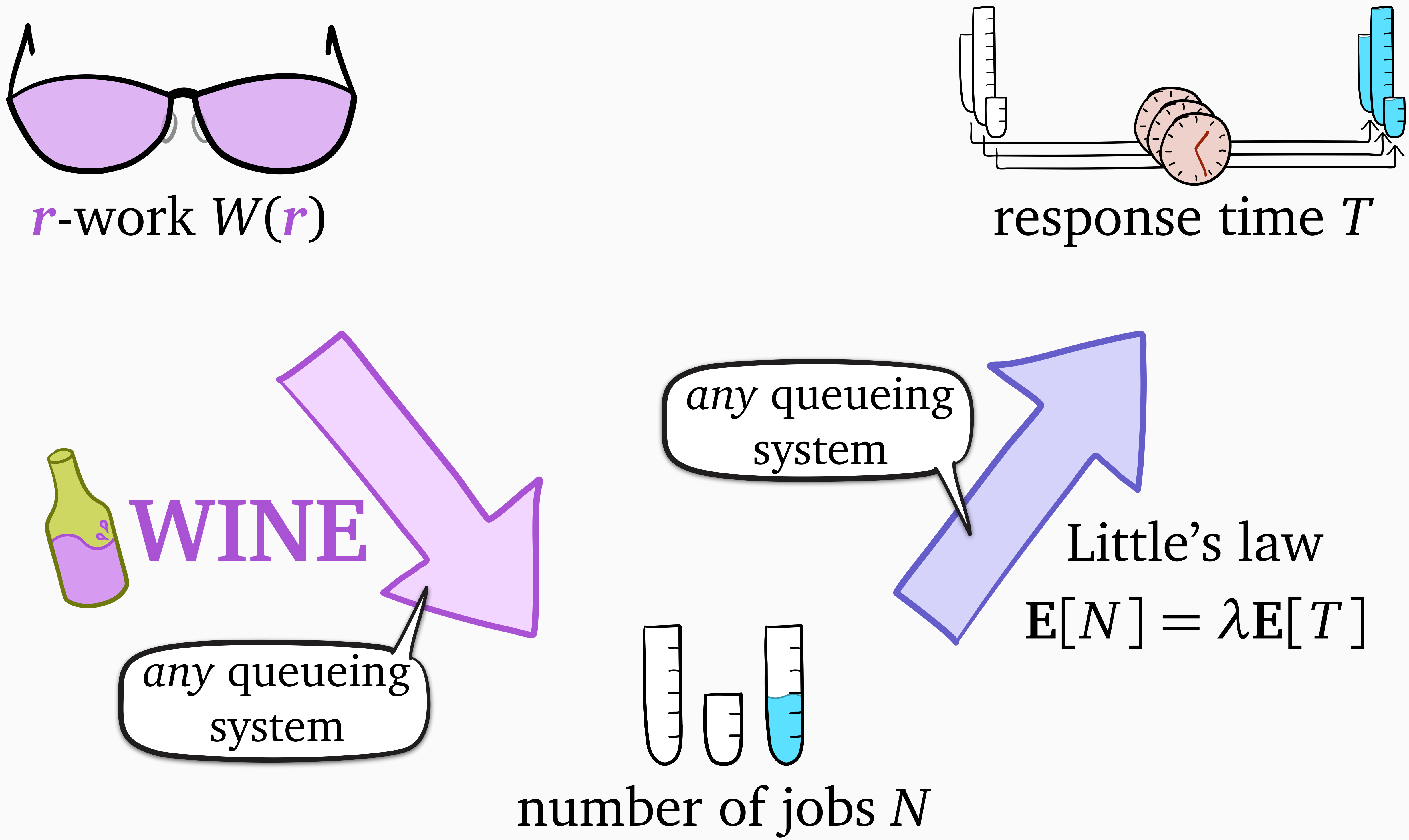
Little's law

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



number of jobs N

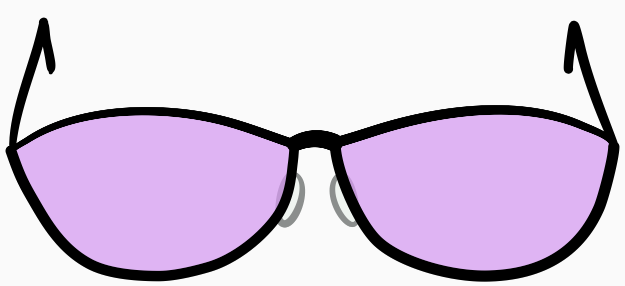

r-work $W(r)$
response time T
WINE
any queueing system
Little's law
$\mathbf{E}[N] = \lambda \mathbf{E}[T]$
number of jobs N



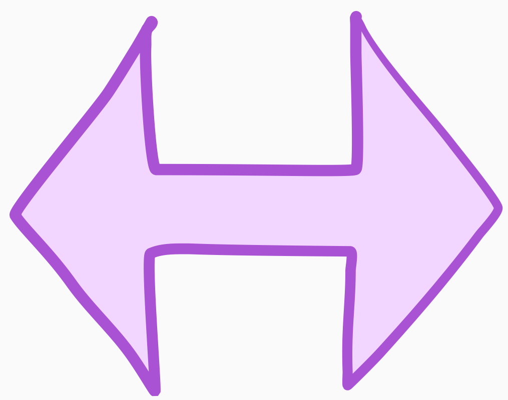
r-work $W(r)$
response time T
WINE
any queueing system
any queueing system
number of jobs N
Little's law
$\mathbf{E}[N] = \lambda \mathbf{E}[T]$

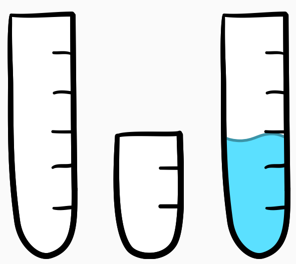
WINE

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



r-work $W(r)$

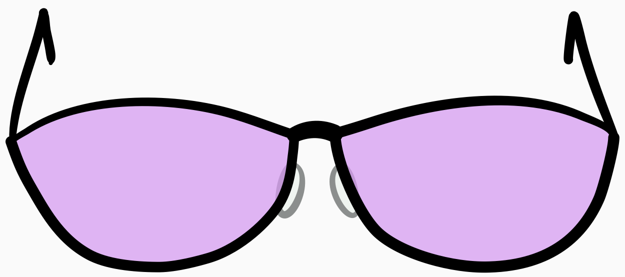




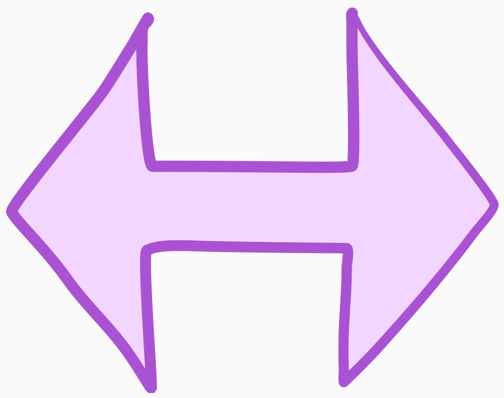
number of jobs N

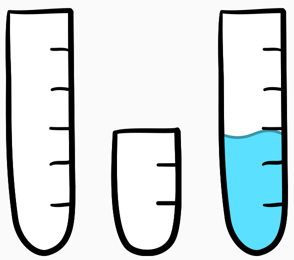
WINE

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



r-work $W(r)$



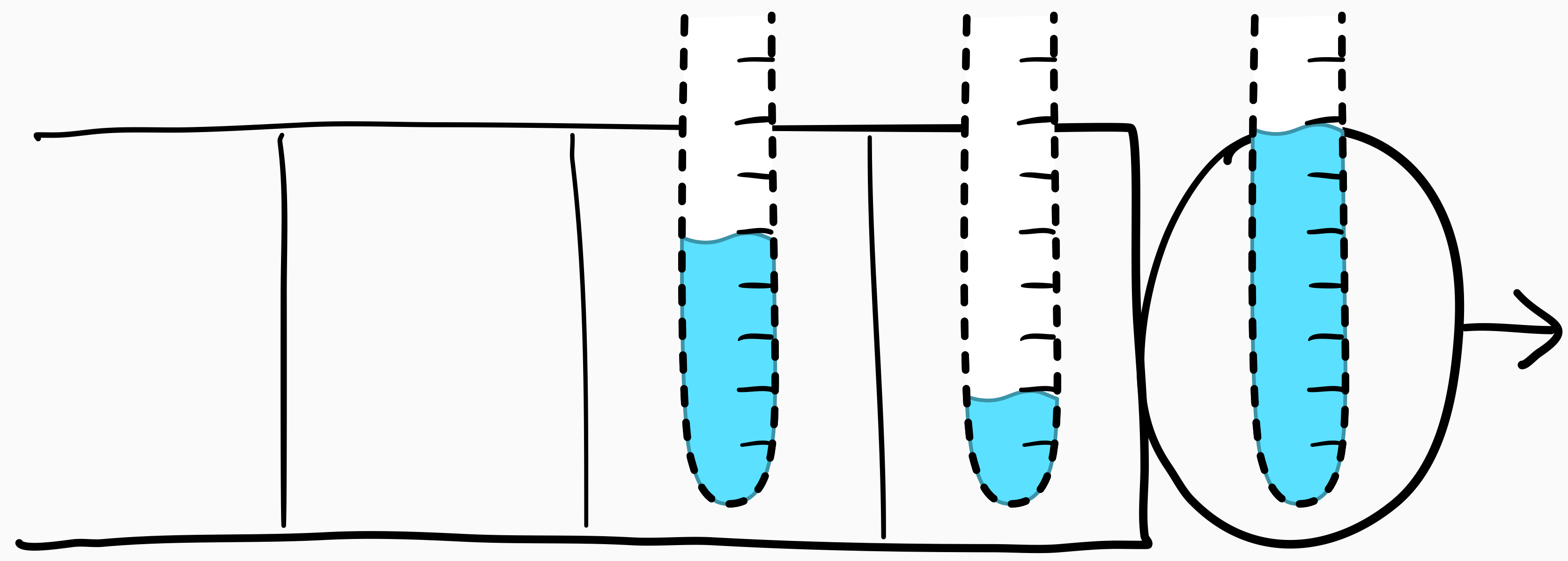


number of jobs N

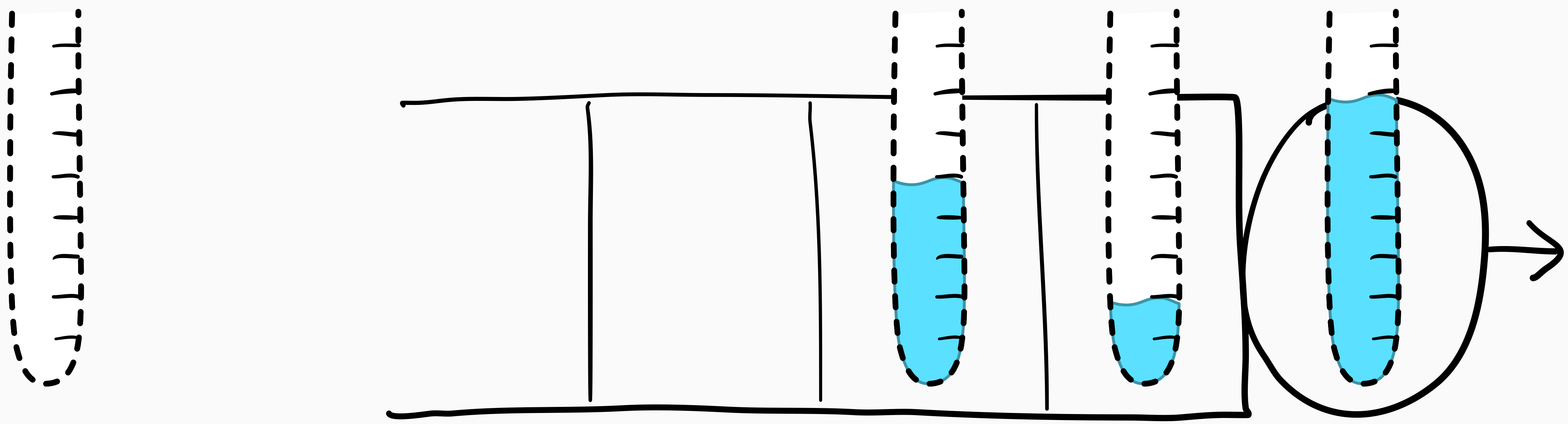
What is r-work?

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

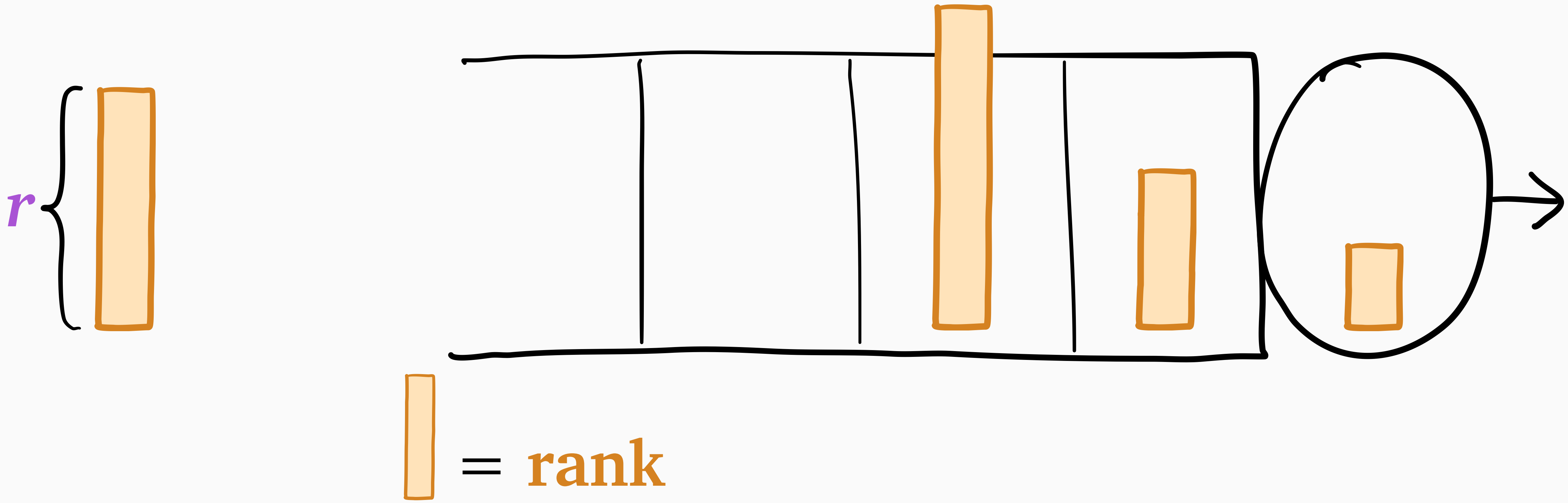
What is r-work $W(r)$?



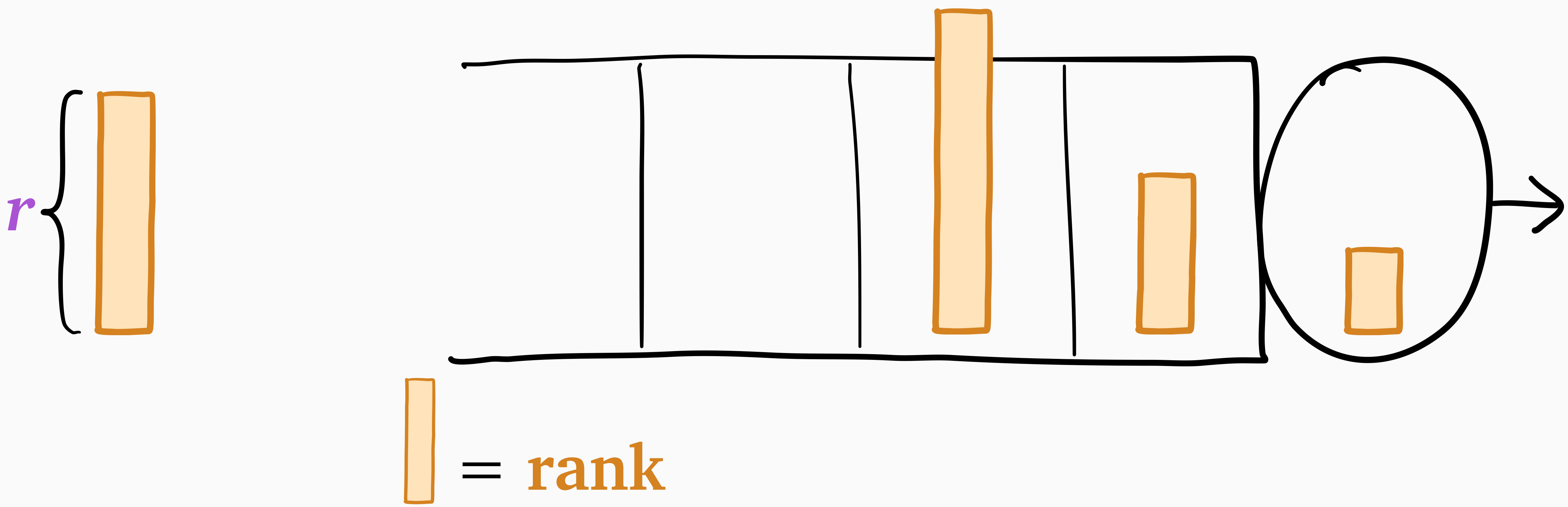
What is r-work $W(r)$?



What is r-work $W(r)$?

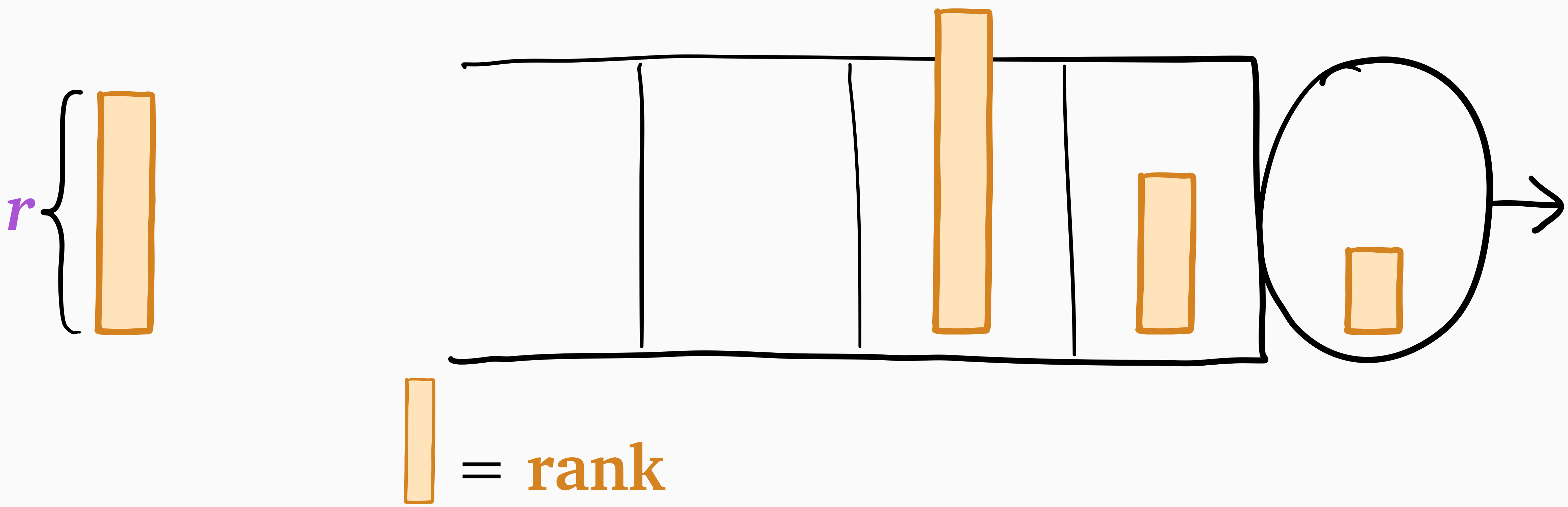


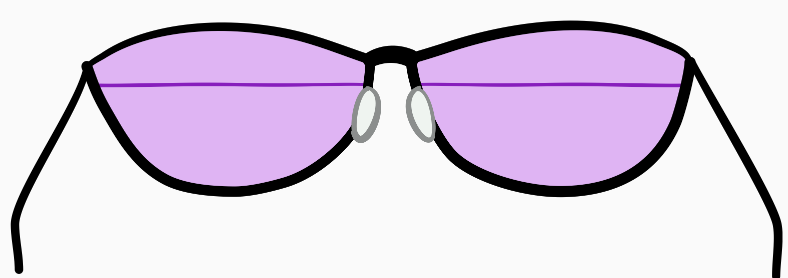
What is r-work $W(r)$?



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

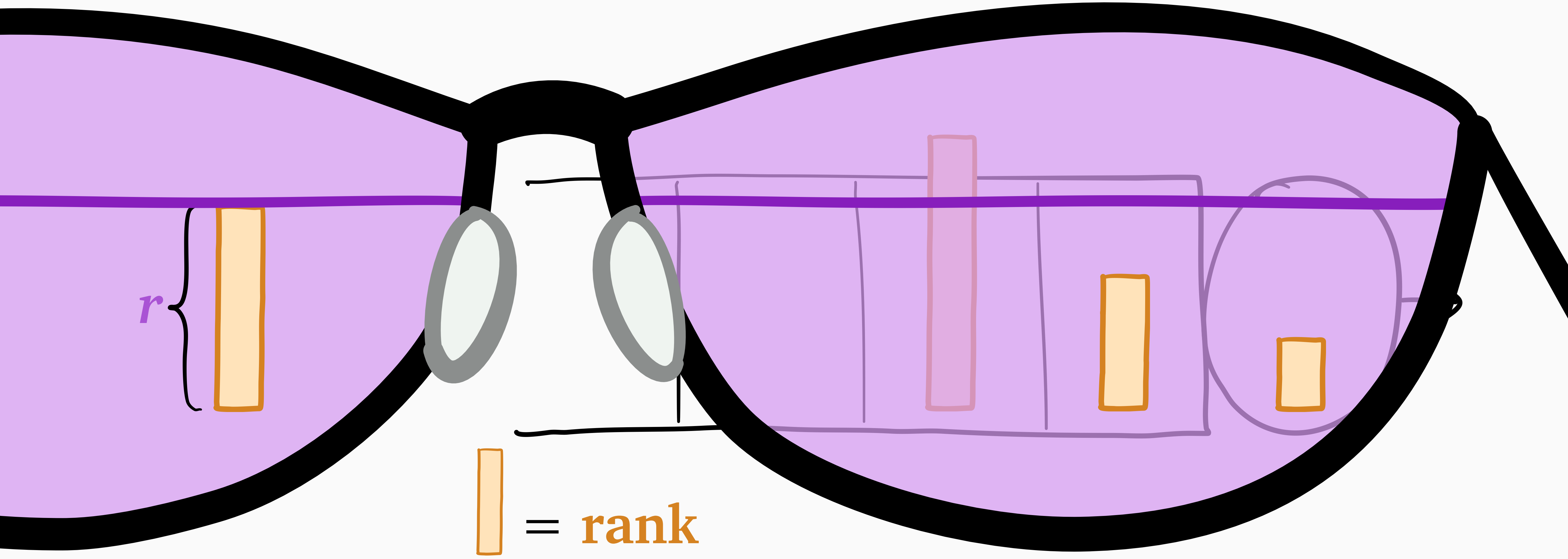
What is r-work $W(r)$?





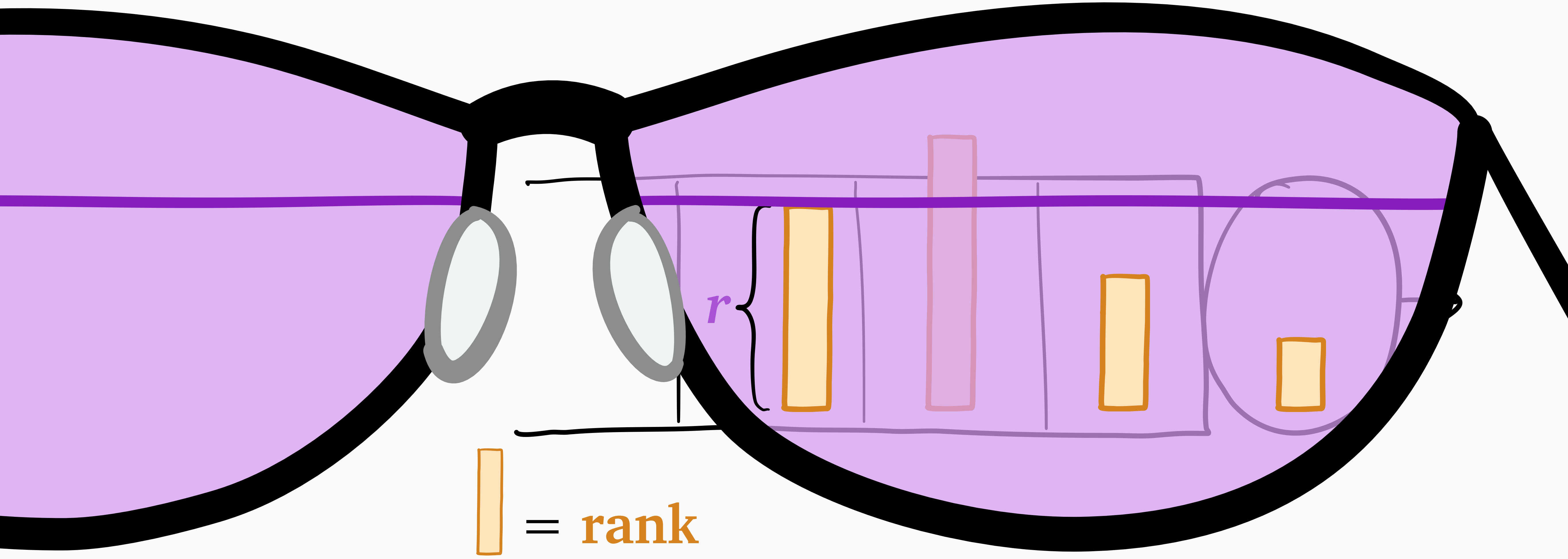
$W(r)$ = work relevant to job of **rank** r

What is r-work $W(r)$?



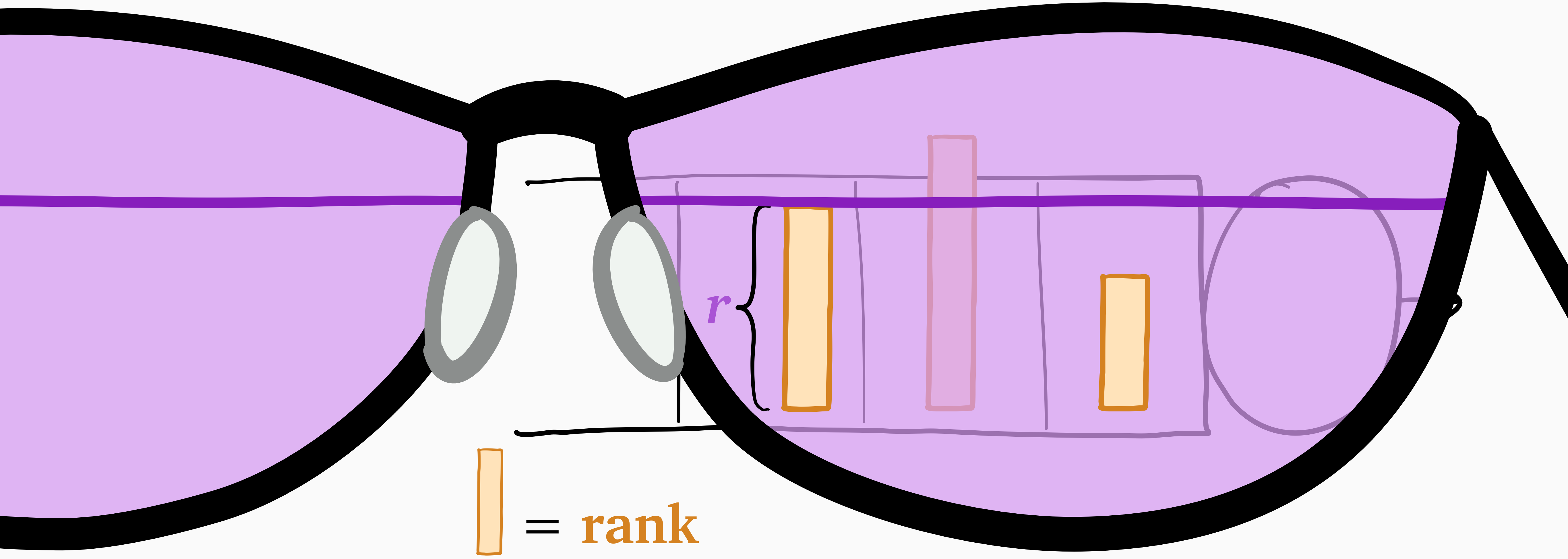
$W(r)$ = work relevant to job of **rank** r

What is r-work $W(r)$?



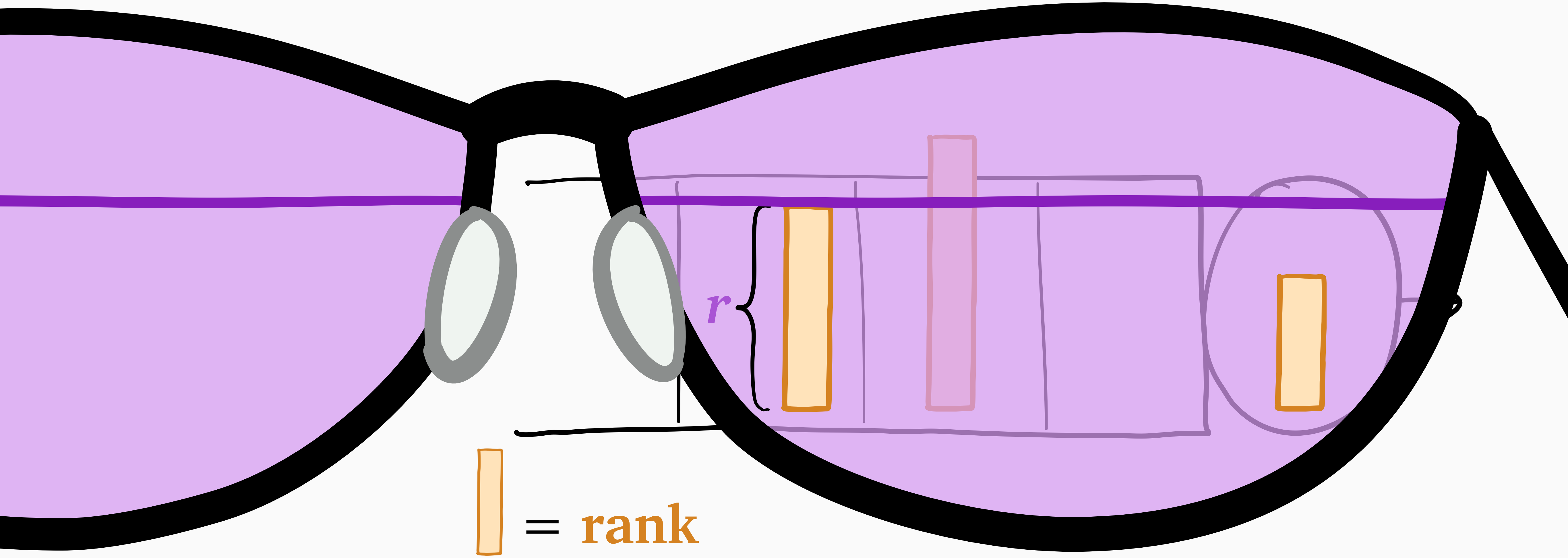
$W(r)$ = work relevant to job of **rank** r

What is r-work $W(r)$?



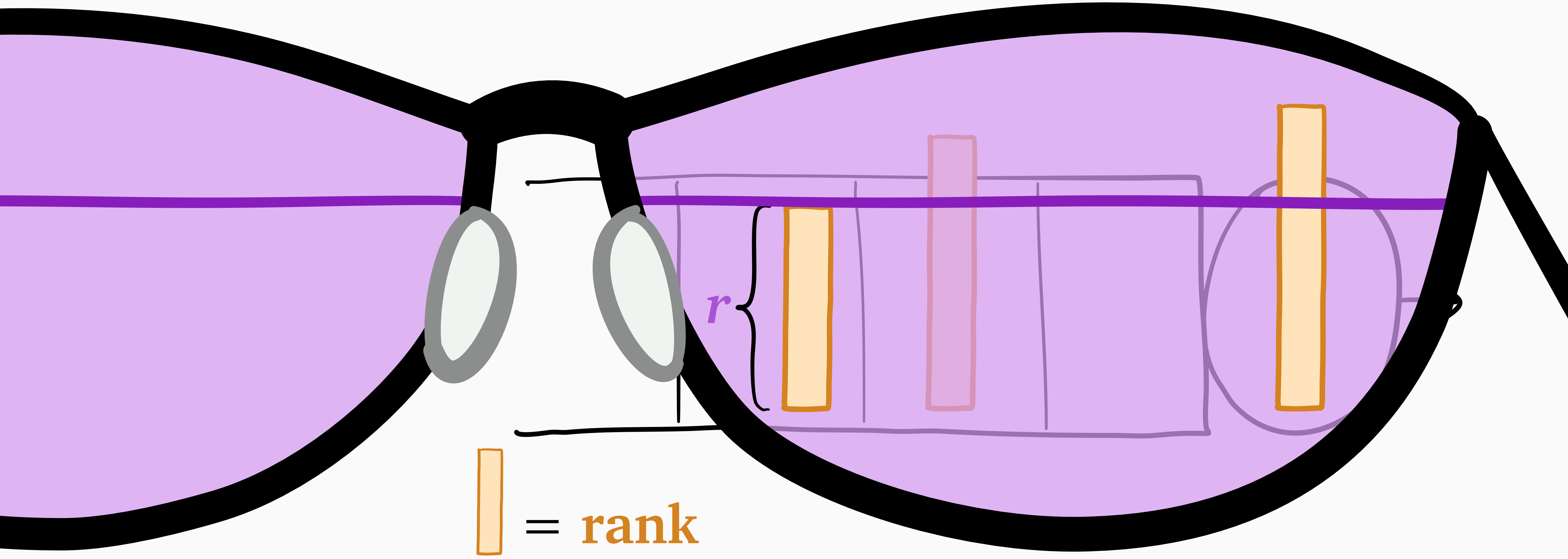
$W(r)$ = work relevant to job of **rank** r

What is r-work $W(r)$?



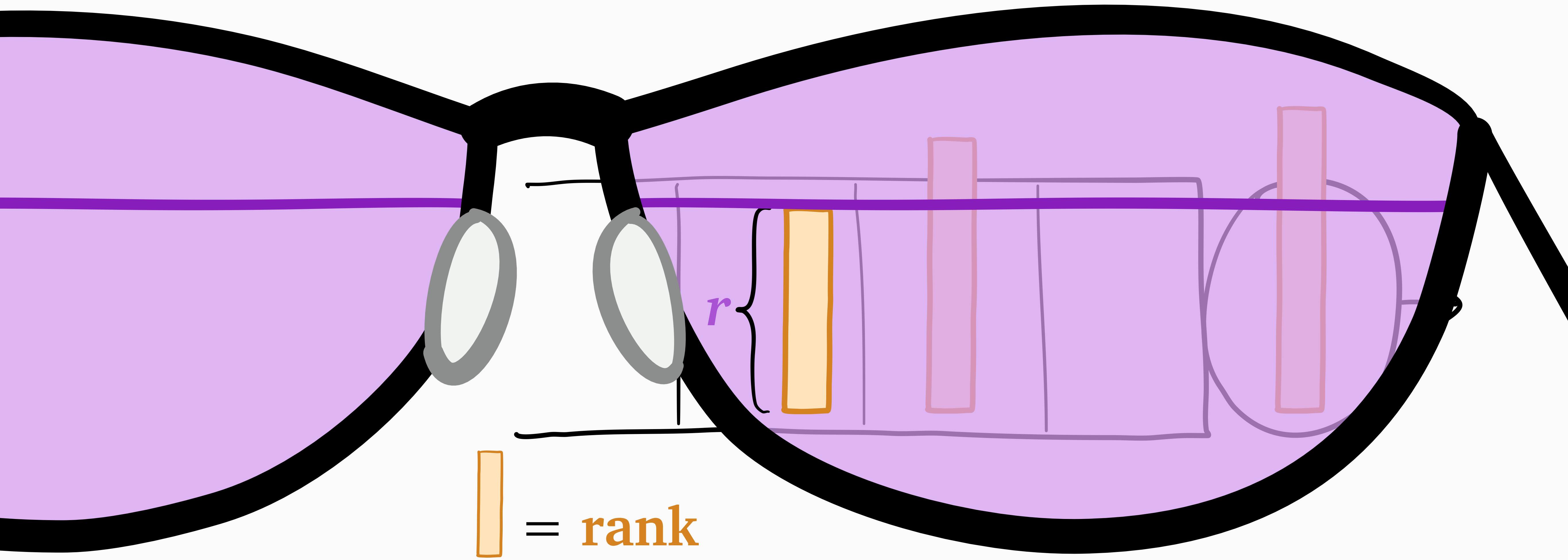
$W(r)$ = work relevant to job of **rank** r

What is r-work $W(r)$?



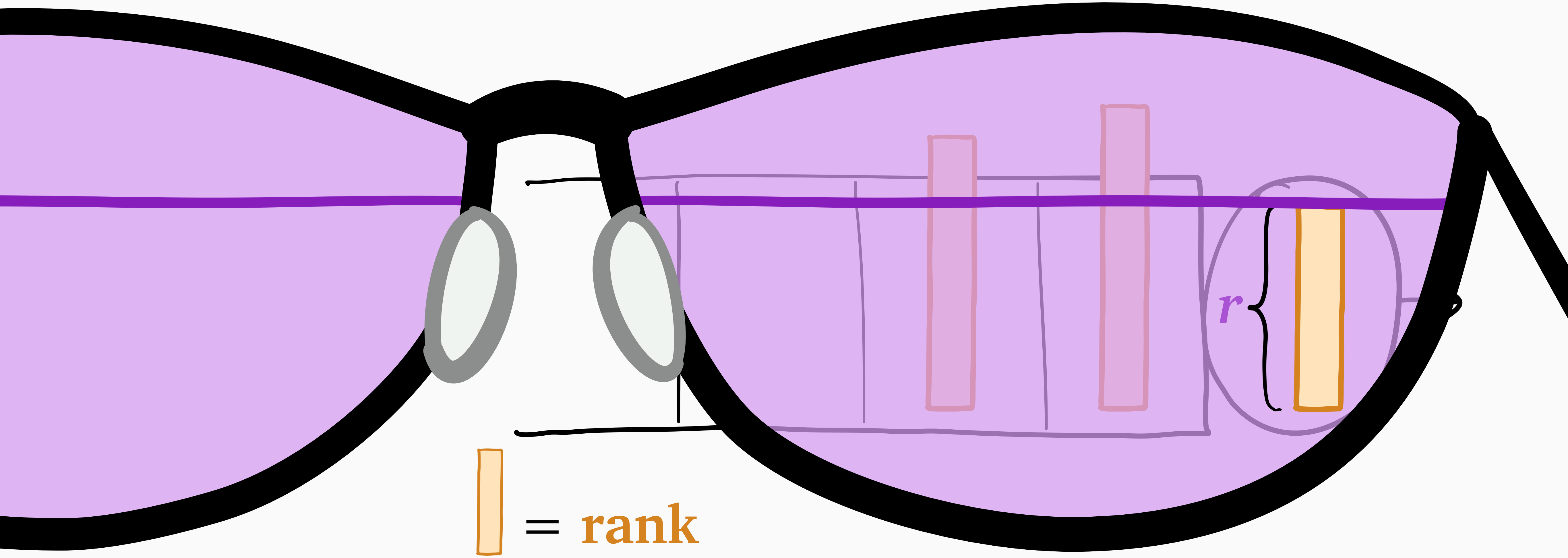
$W(r)$ = work relevant to job of **rank** r

What is r-work $W(r)$?



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

What is r-work $W(r)$?



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

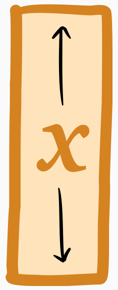
Defining r-work for SRPT

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

Defining r-work for SRPT

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

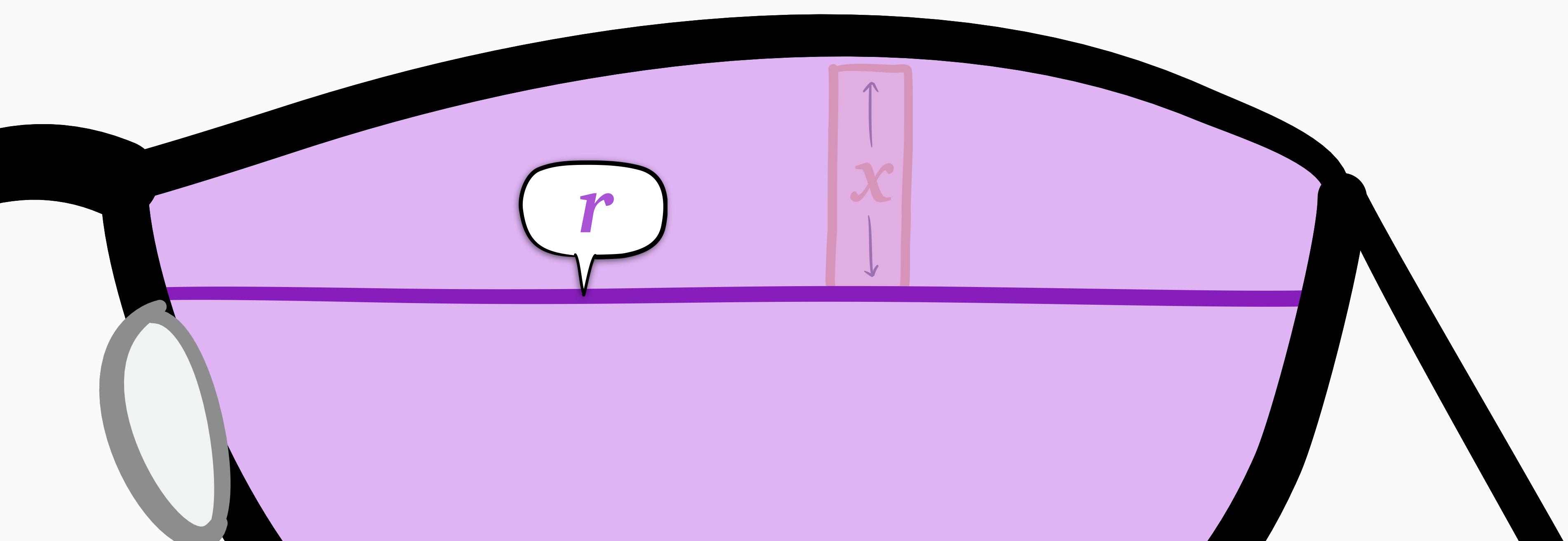
$w_x(r)$ = r-work of *single job* of rem. size x = $\left\{ \right.$



Defining r-work for SRPT

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

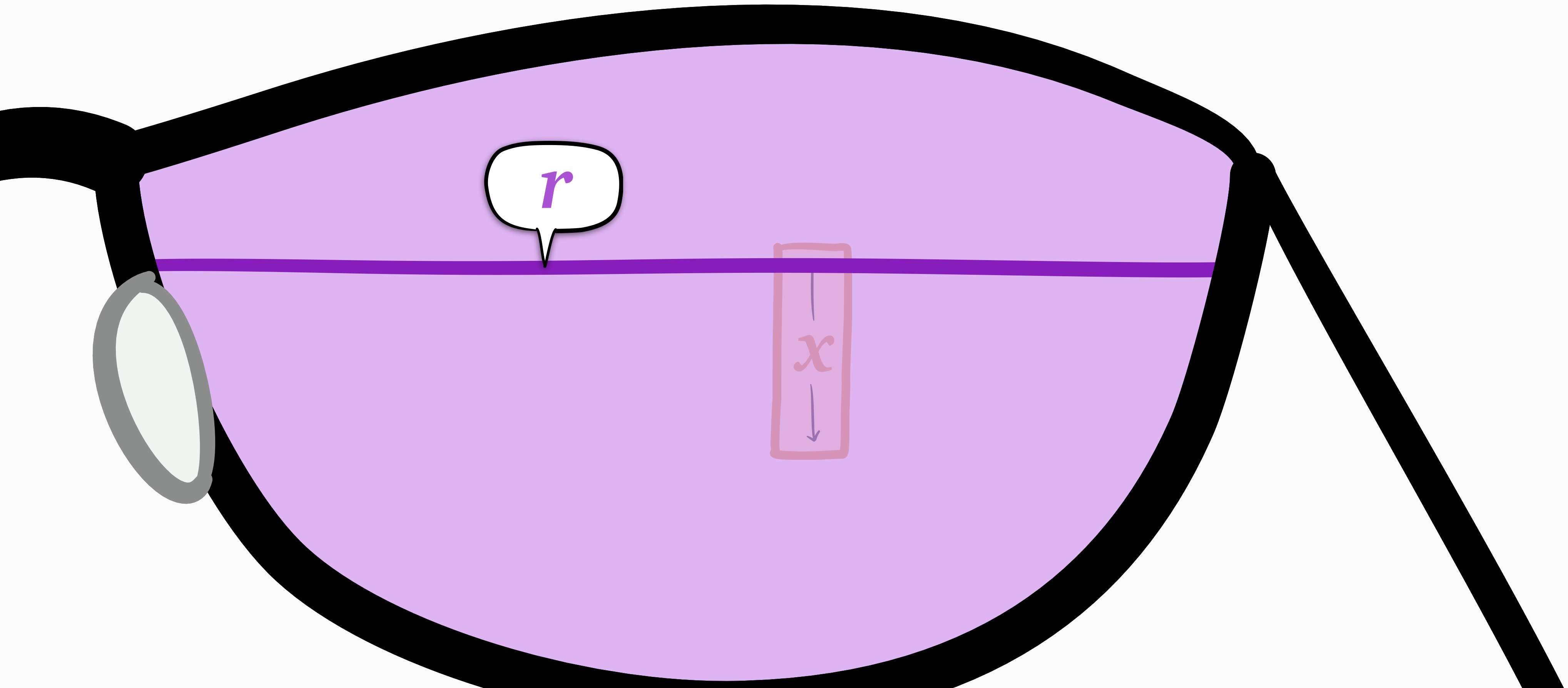
$w_x(r)$ = r-work of *single job* of rem. size x = $\{$



Defining r-work for SRPT

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

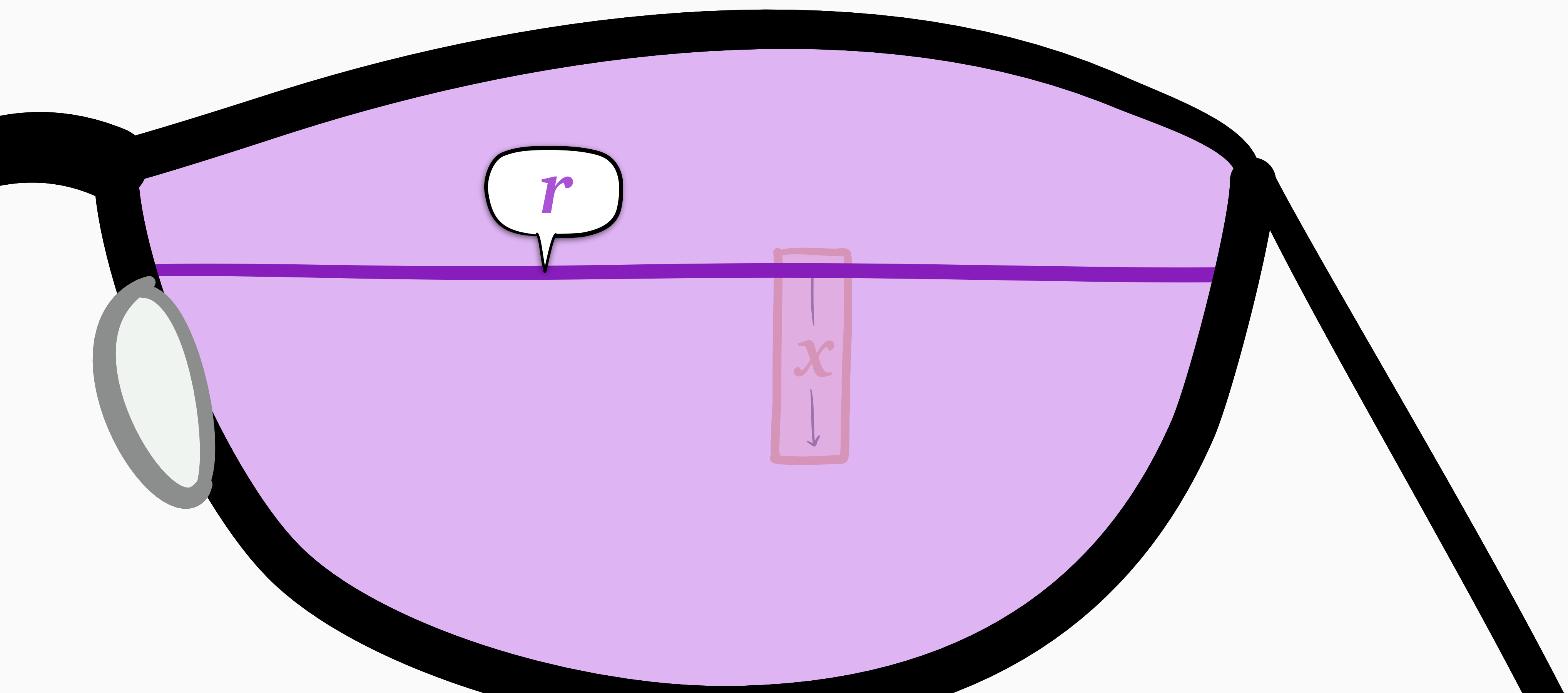
$w_x(r)$ = r-work of *single job* of rem. size x = $\{$



Defining r-work for SRPT

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

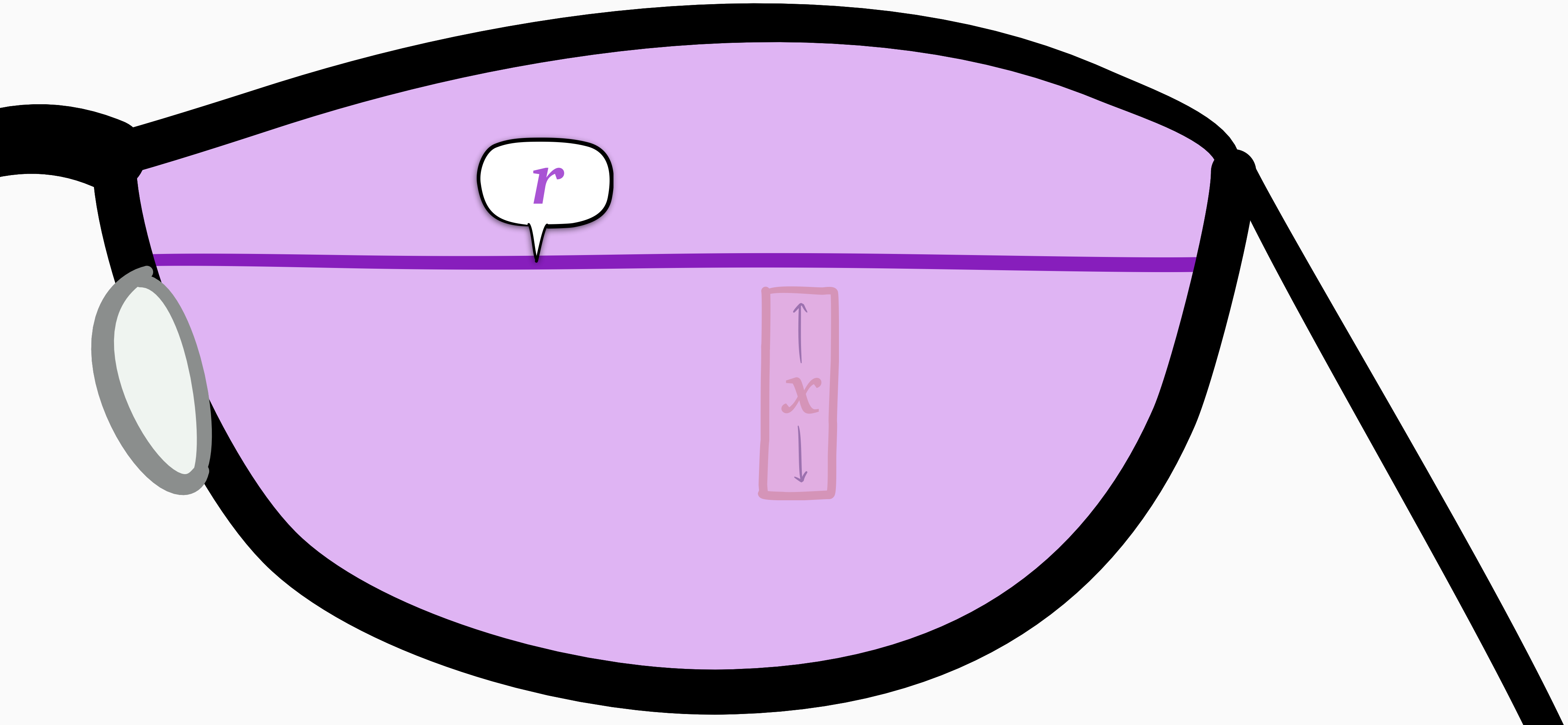
$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ & \end{cases}$



Defining r-work for SRPT

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

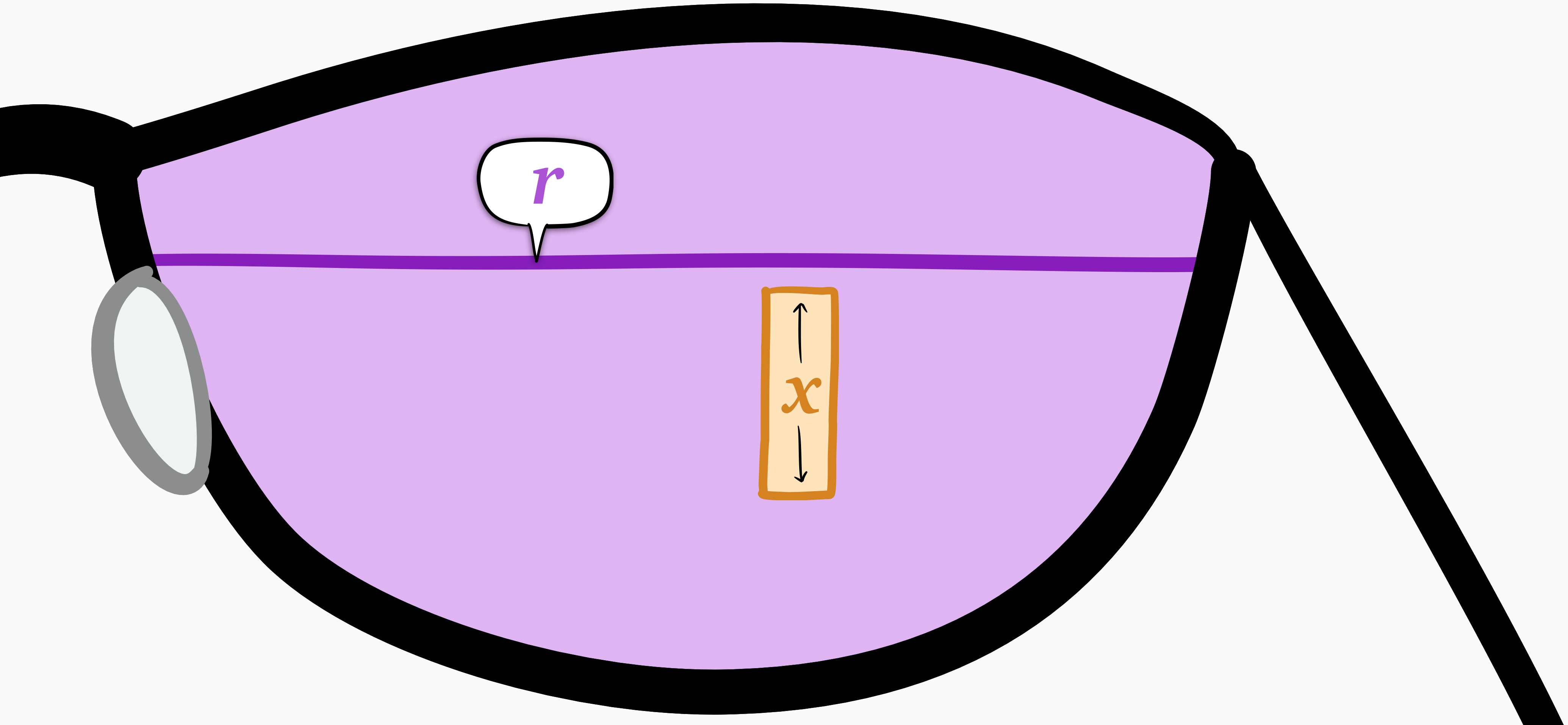
$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ & \end{cases}$



Defining r-work for SRPT

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

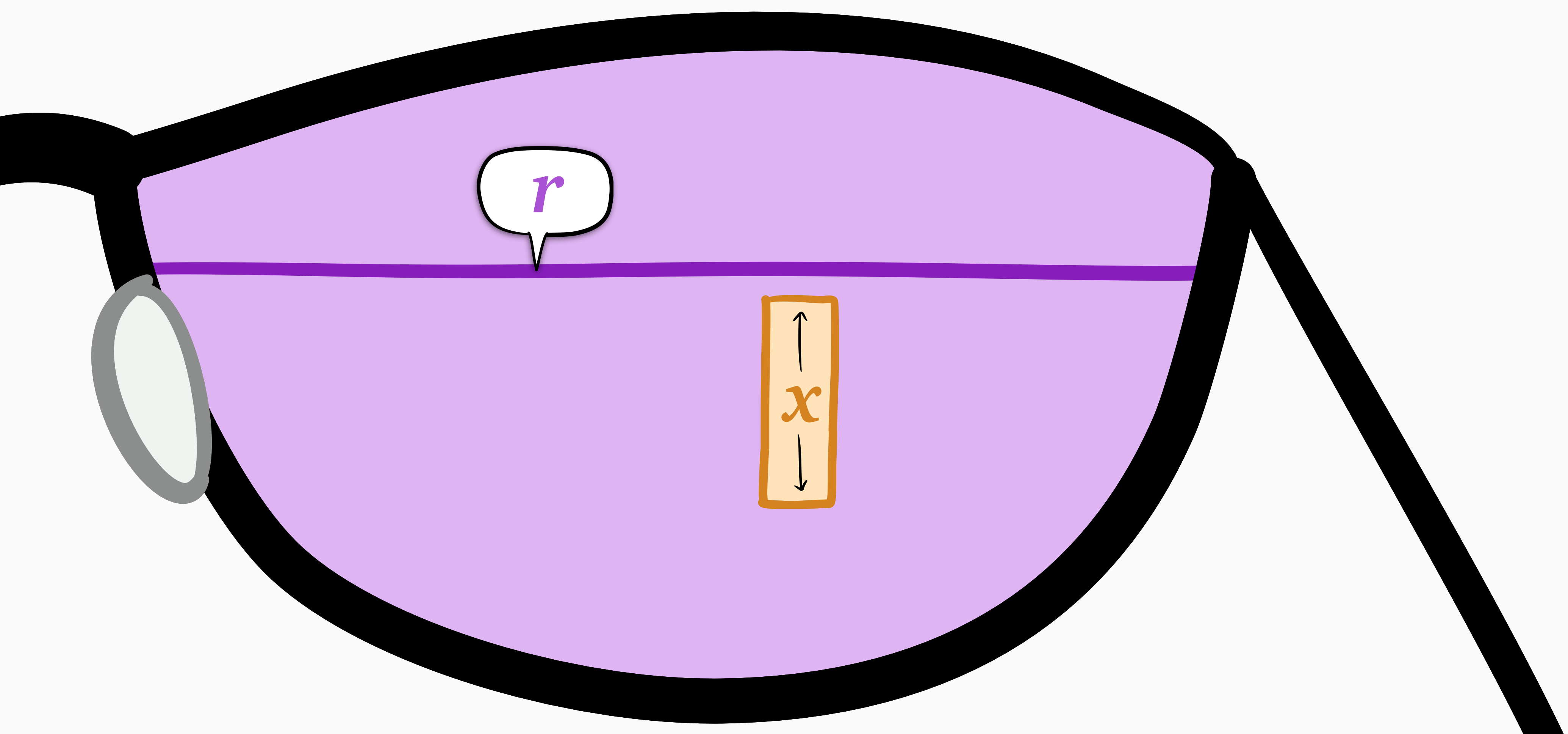
$$w_x(r) = r\text{-work of } \textit{single job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ & \end{cases}$$



Defining r-work for SRPT

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

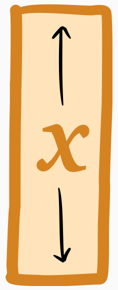
$$w_x(r) = r\text{-work of } \textit{single job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$



Defining r-work for SRPT

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

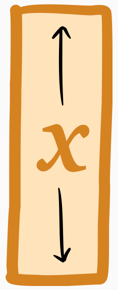
$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$



Defining r-work for SRPT

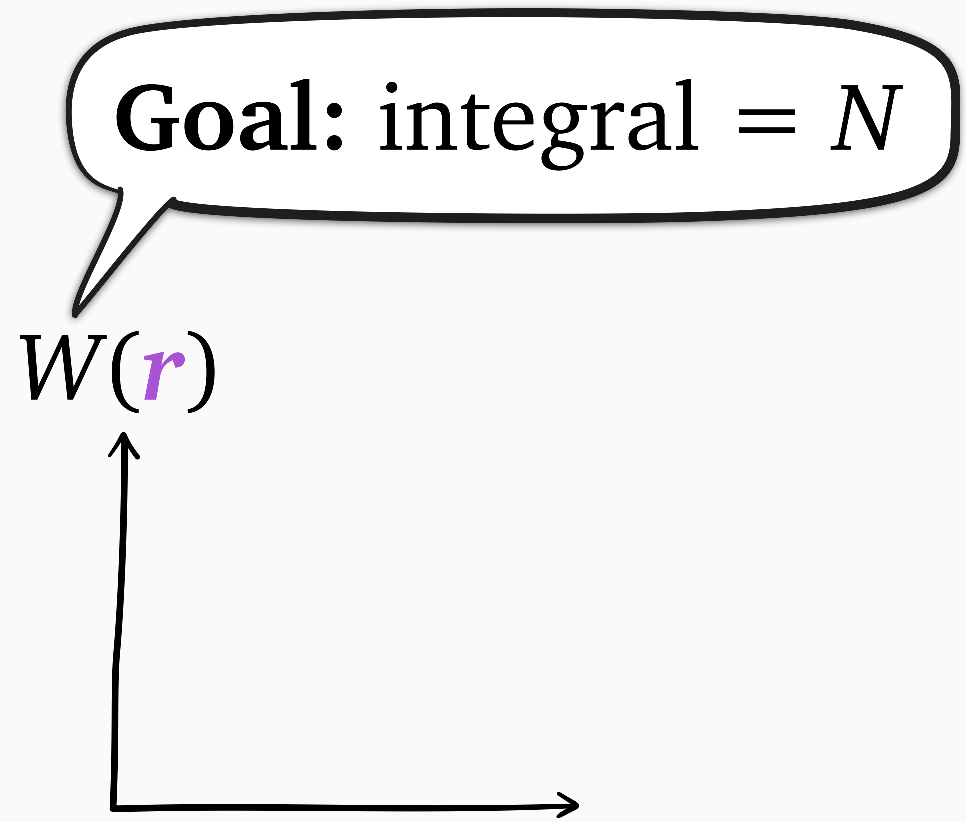
$$W(r) = \text{work relevant to } \textbf{rank } r$$
$$= \text{total } r\text{-work of all jobs}$$

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

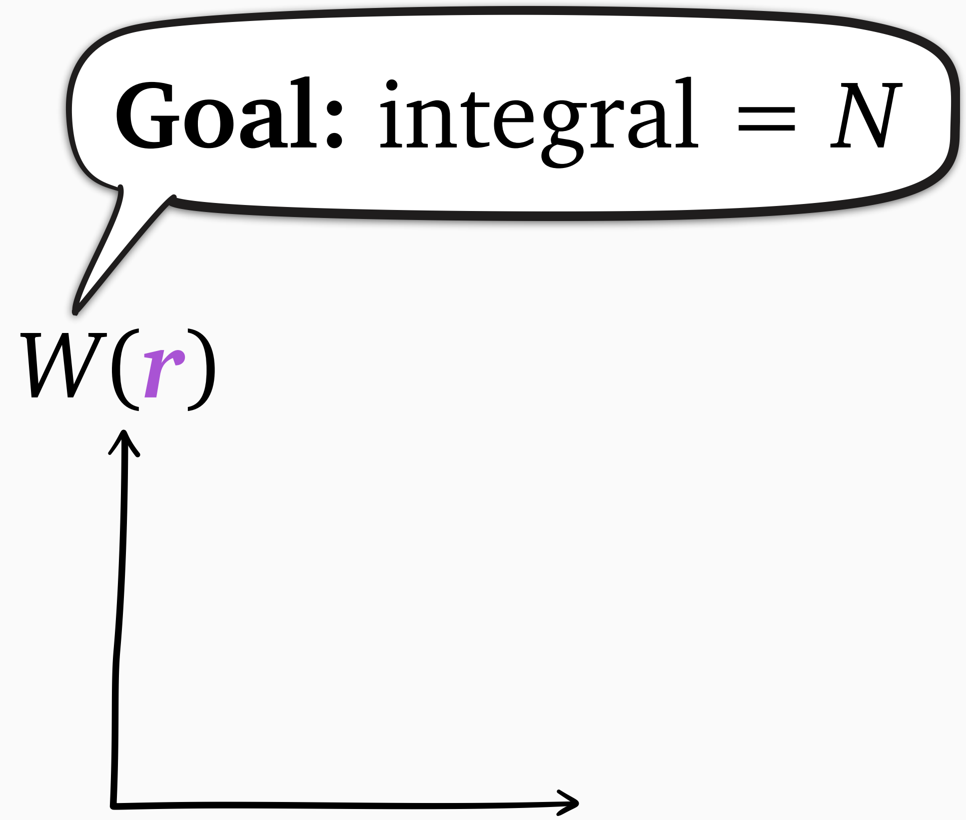


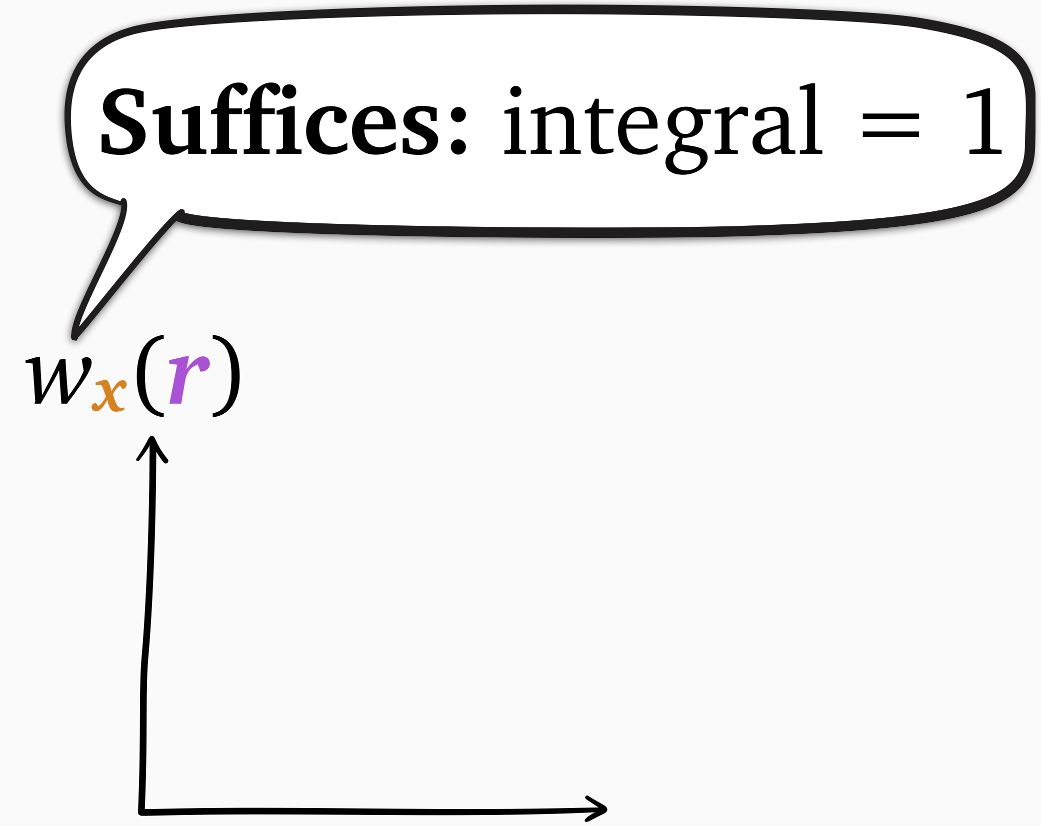
How do we get N from $W(\mathbf{r})$?

How do we get N from $W(r)$?



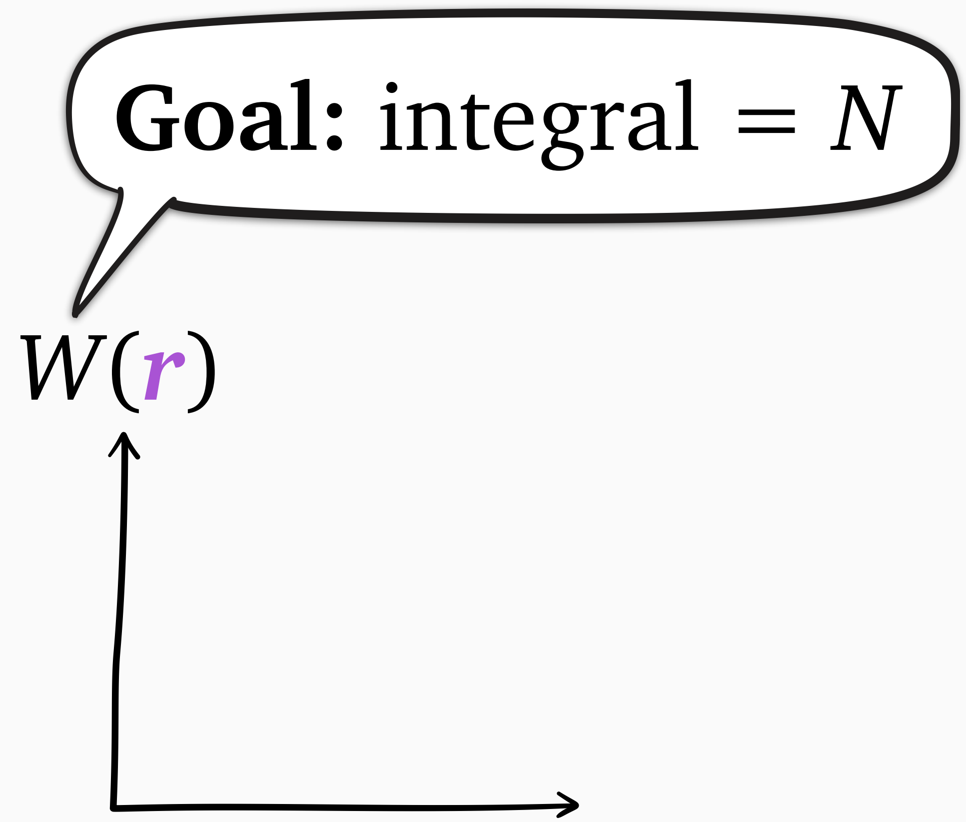
How do we get N from $W(r)$?

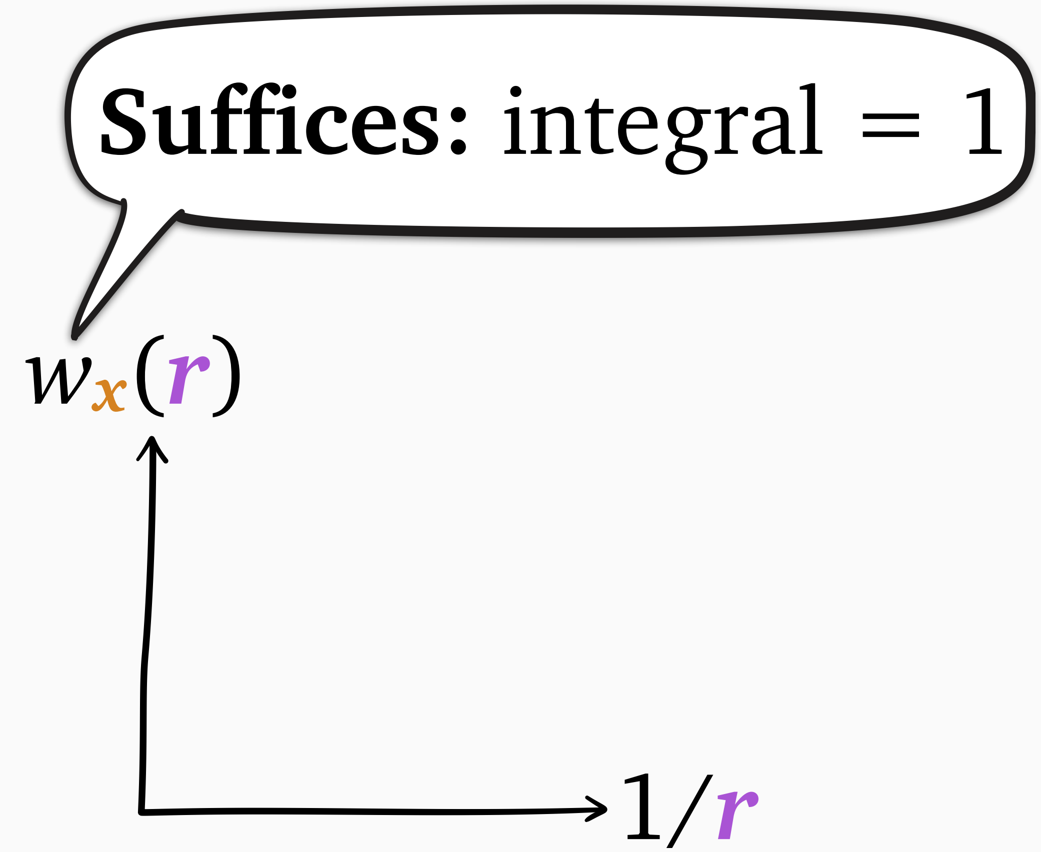




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

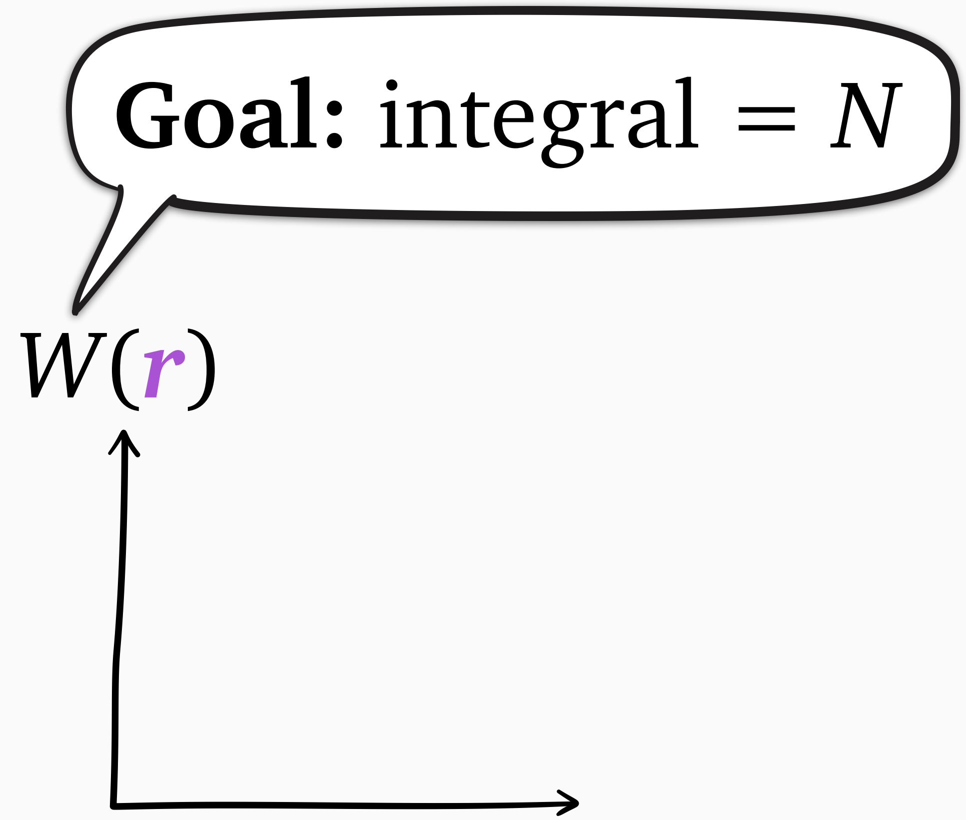
How do we get N from $W(r)$?

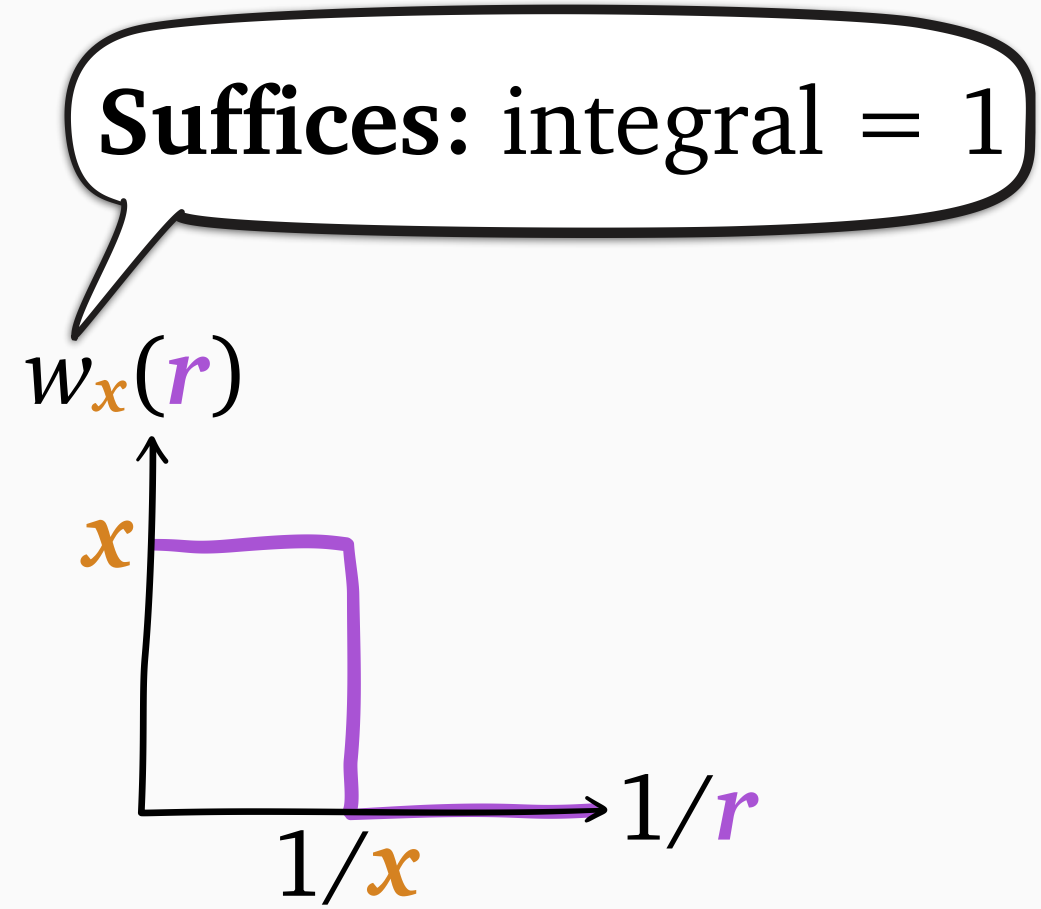




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

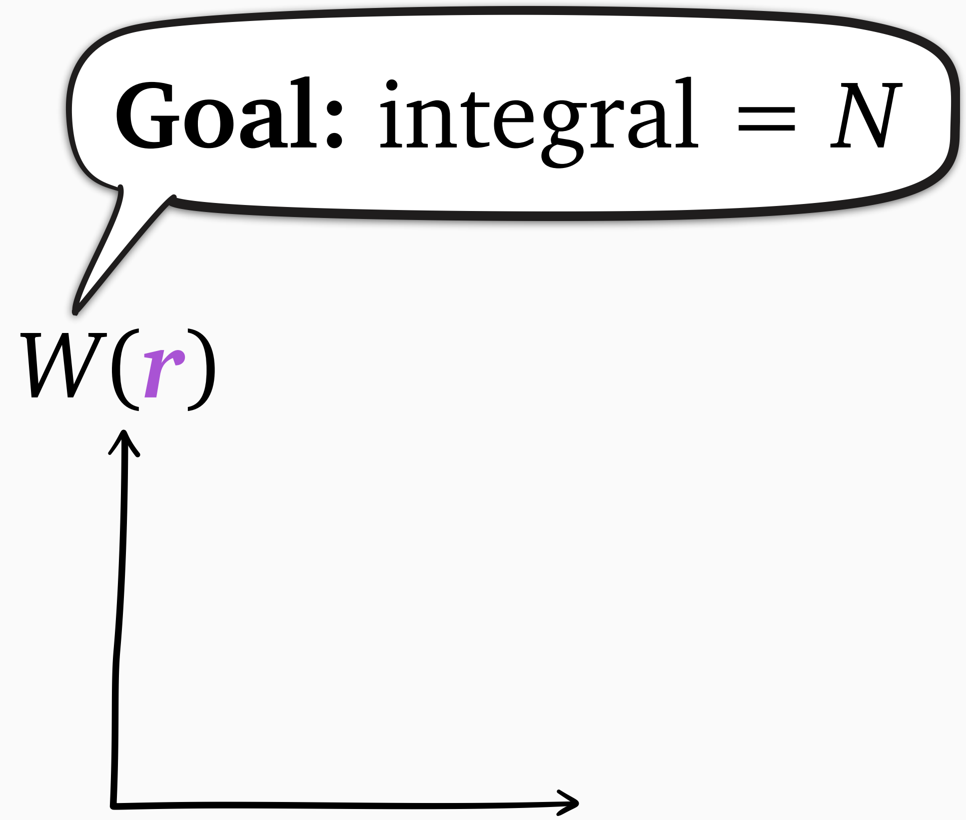
How do we get N from $W(r)$?

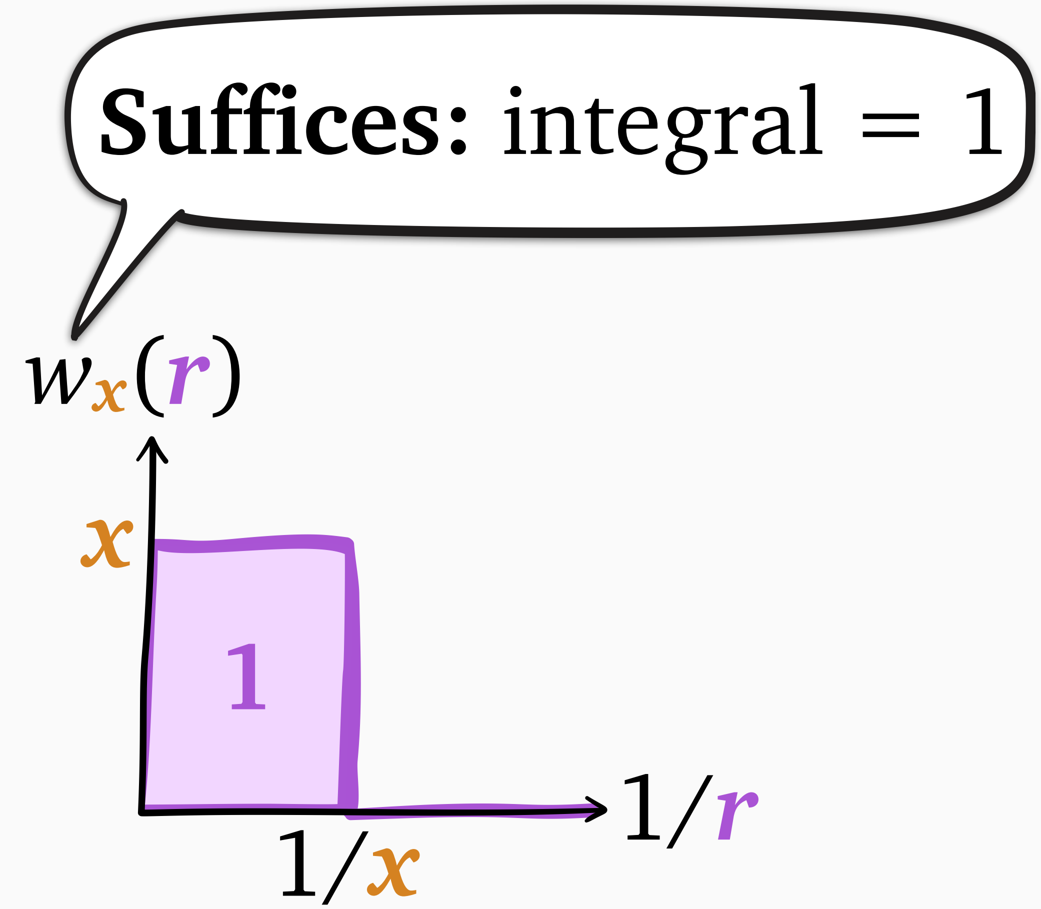




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

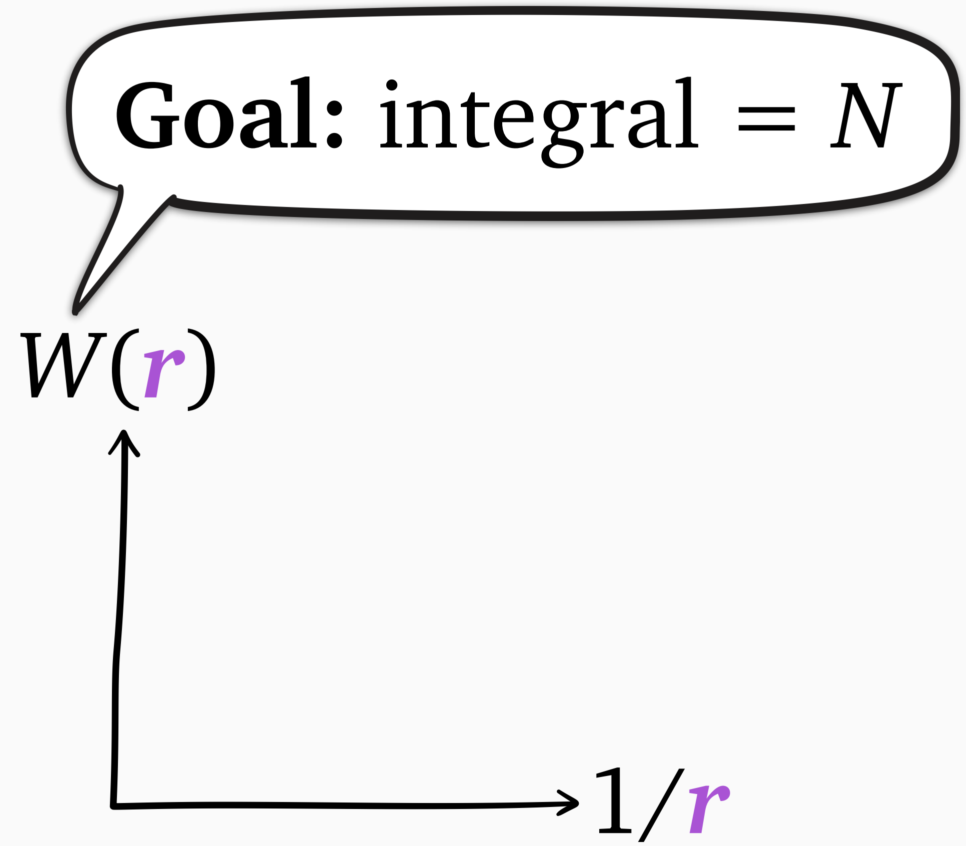
How do we get N from $W(r)$?

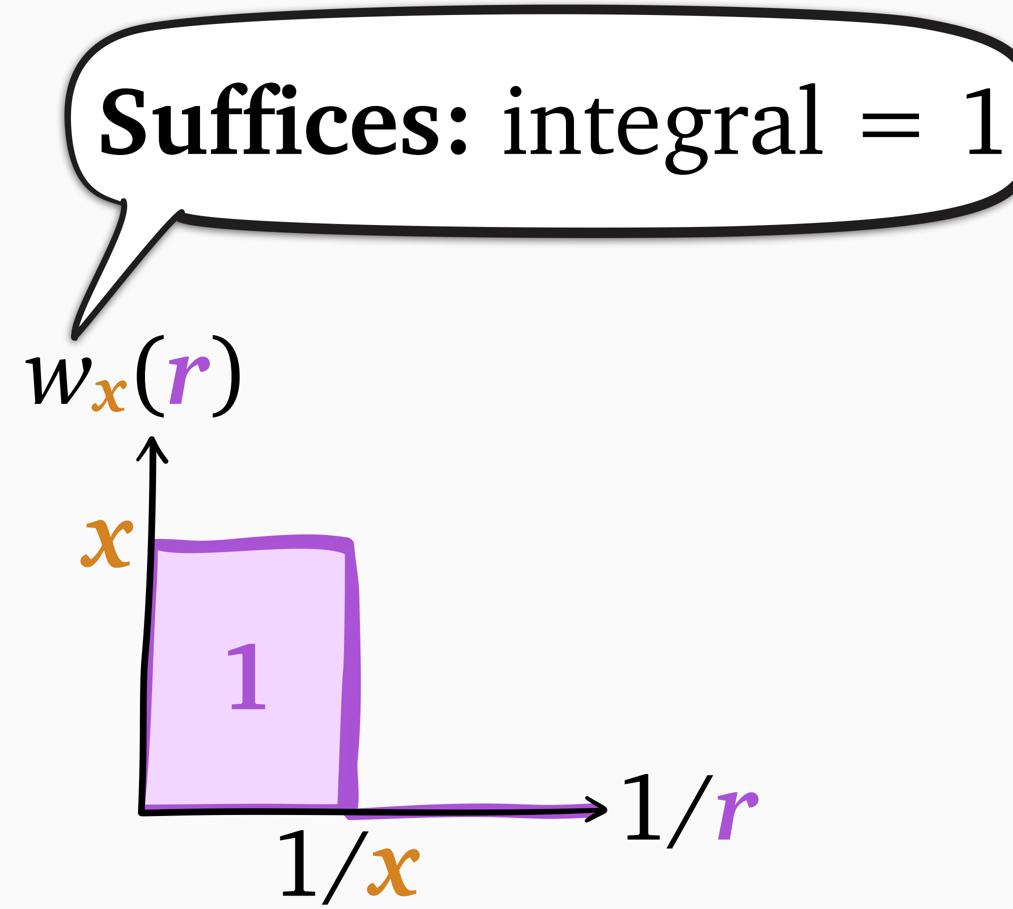




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

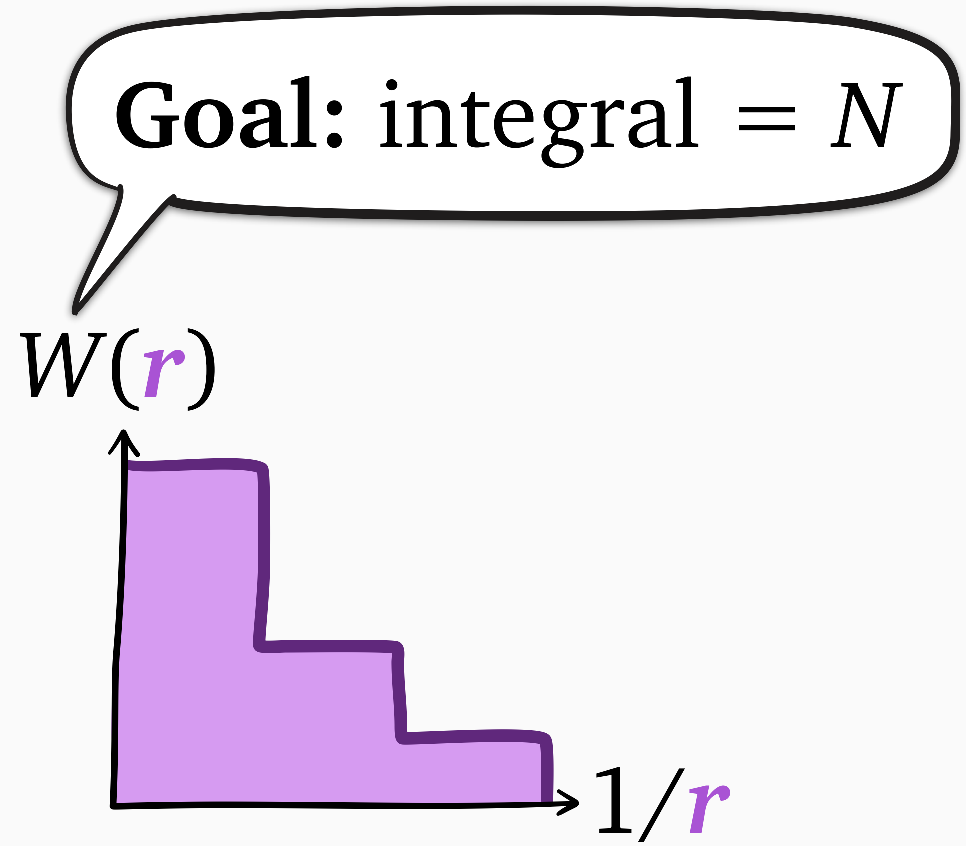
How do we get N from $W(r)$?

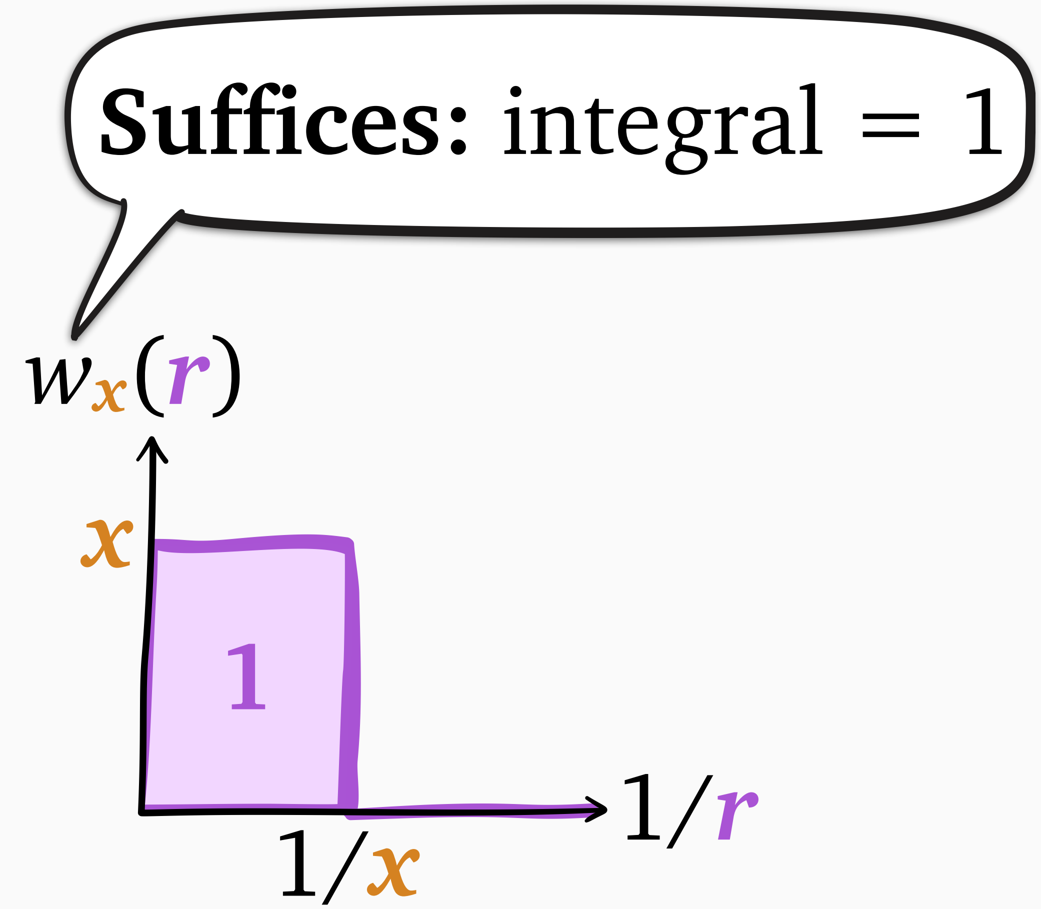




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

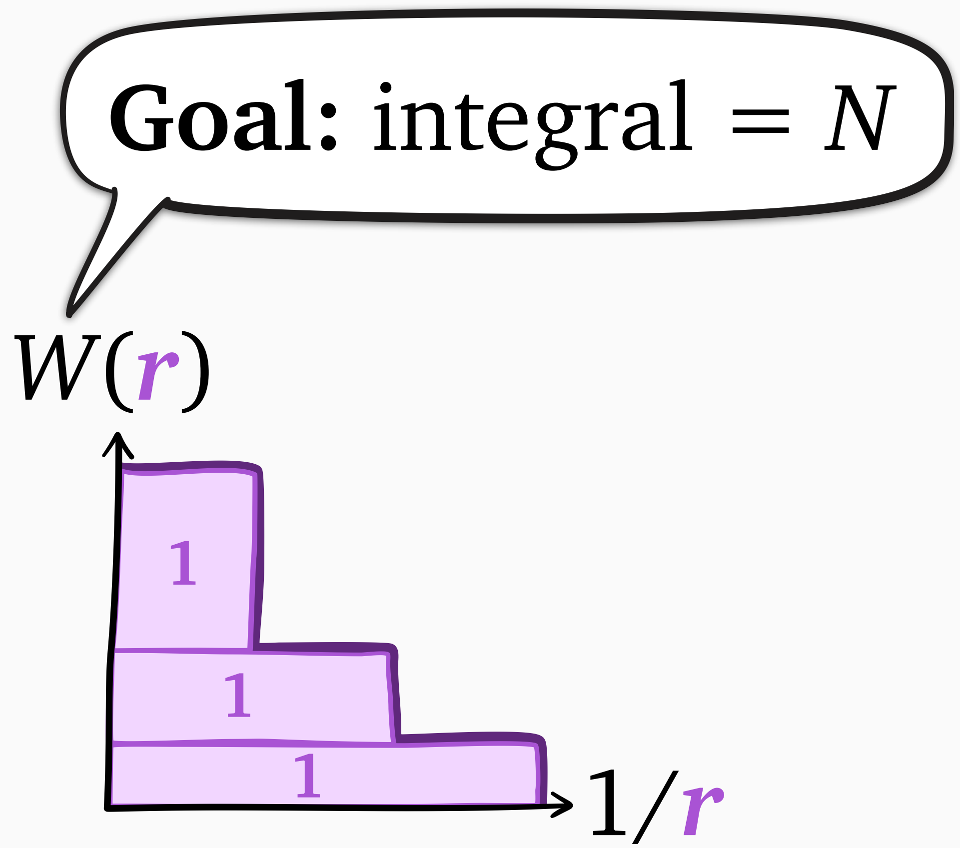
How do we get N from $W(r)$?

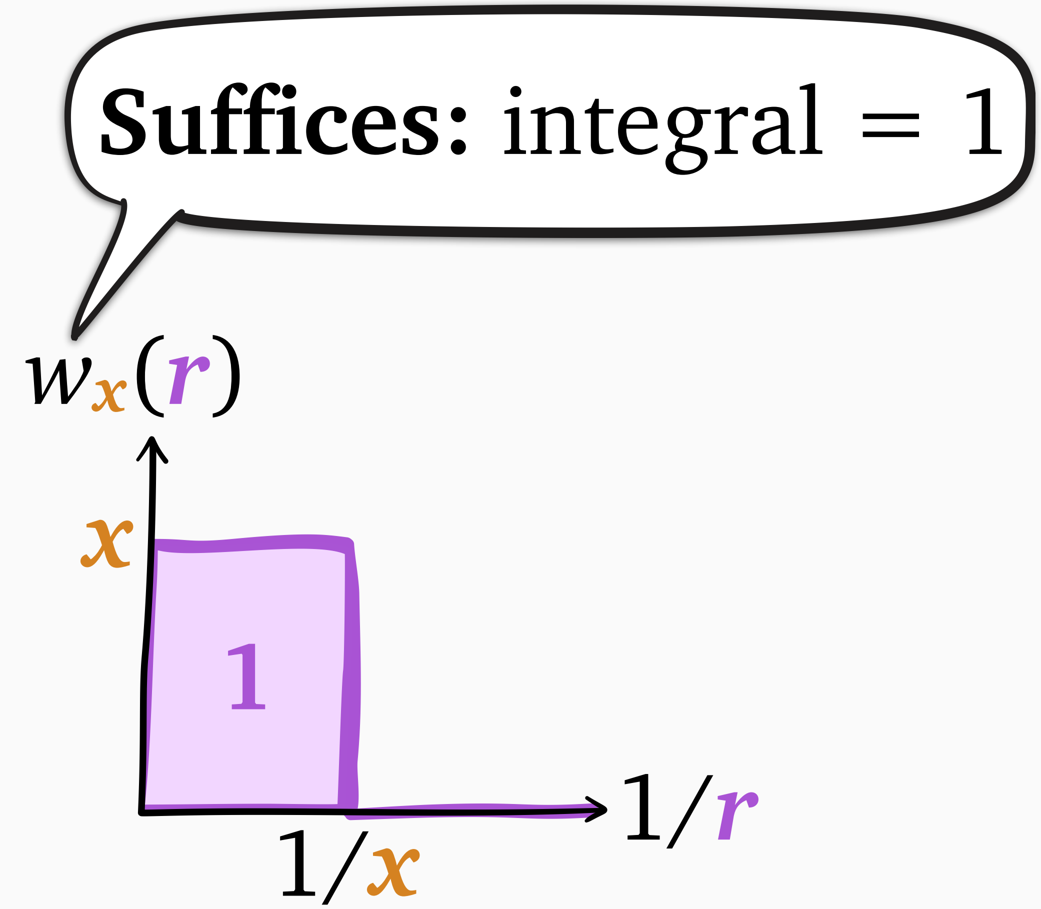




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

How do we get N from $W(r)$?

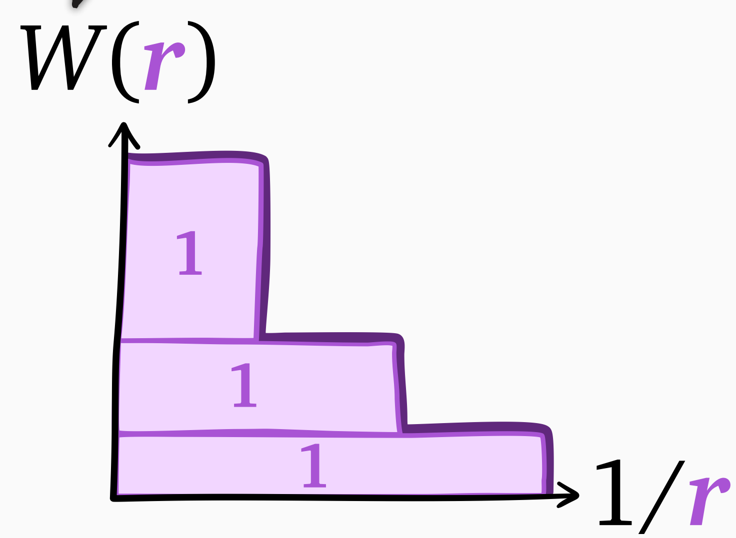




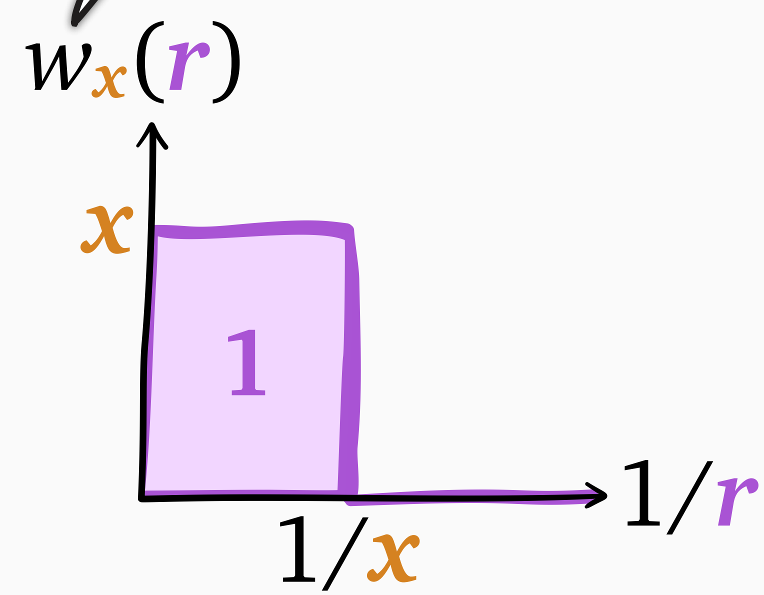
$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

How do we get N from $W(r)$?

Goal: integral $= N$



Suffices: integral $= 1$



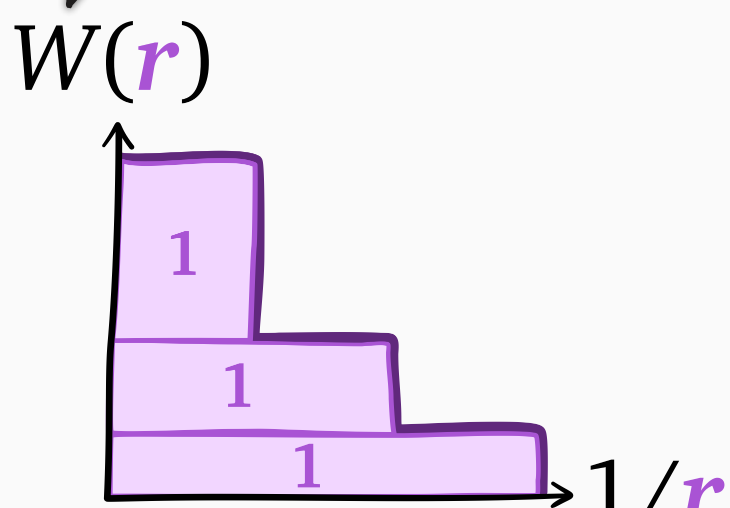
NEW!

Theorem:

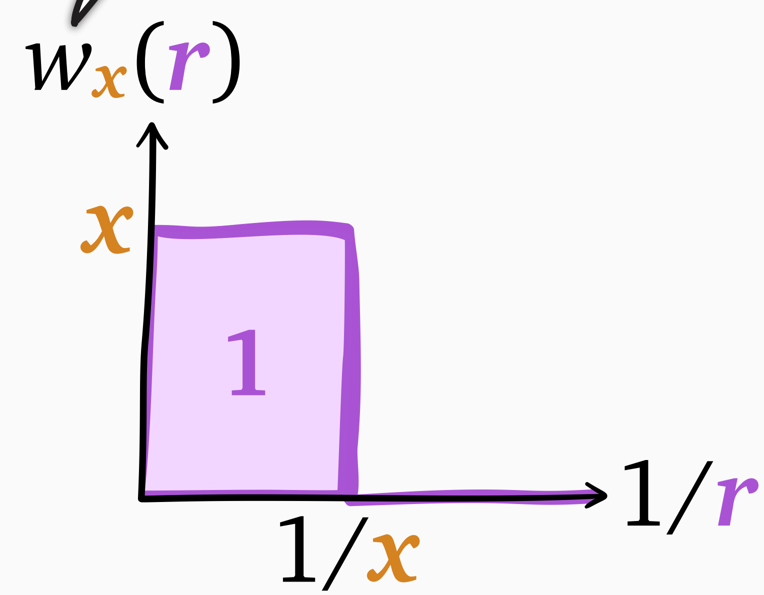
$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

How do we get N from $W(r)$?

Goal: integral $= N$



Suffices: integral $= 1$

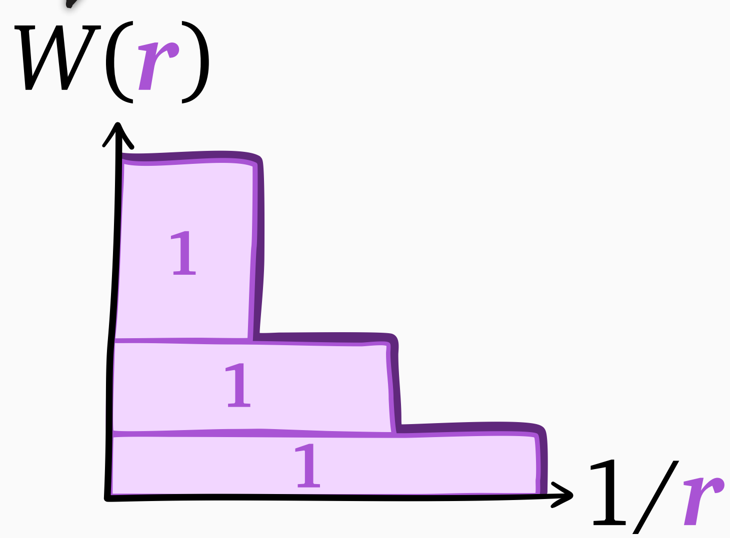


Theorem:

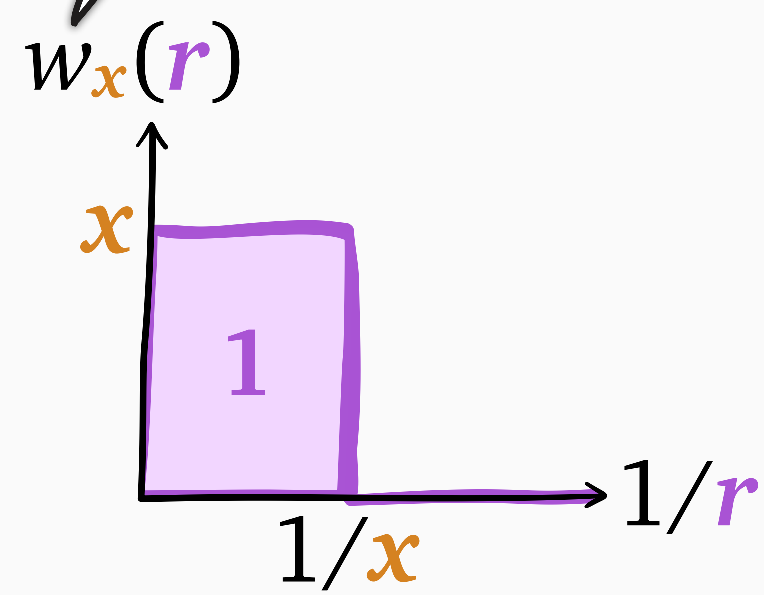
$$N = \int_0^\infty \frac{W(r)}{r^2}\, \mathrm{d}r$$

How do we get N from $W(r)$?

Goal: integral = N



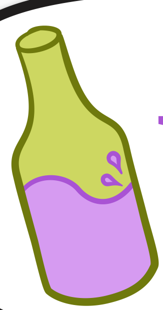
Suffices: integral = 1



Theorem:

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

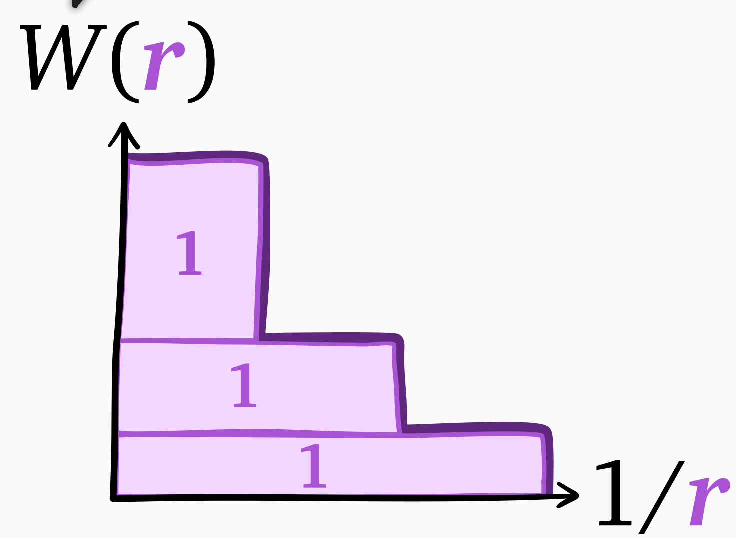
seen by **SRPT** glasses



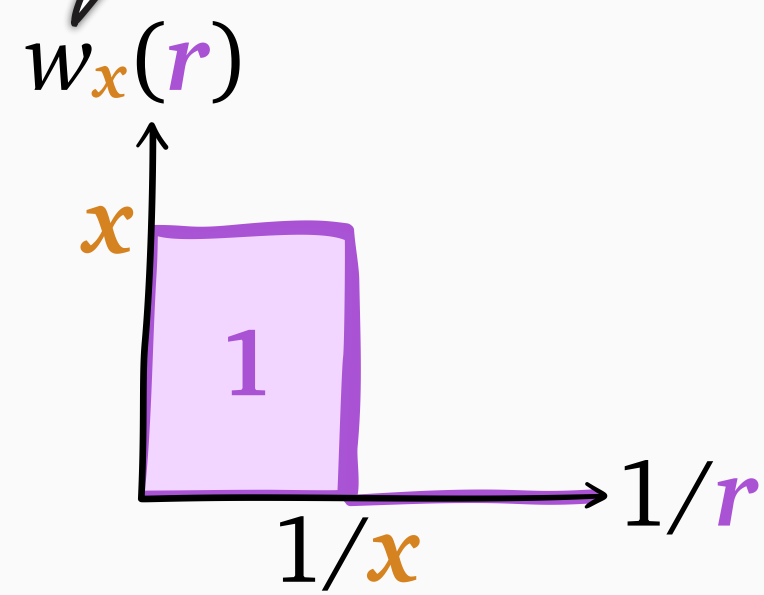
WINE

How do we get N from $W(r)$?

Goal: integral $= N$



Suffices: integral $= 1$

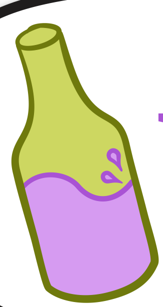


NEW!

Theorem: in *any* system,

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

seen by **SRPT** glasses



WINE

WINE

Work Integral Number Equality

WINE

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

$$N = \int_0^\infty \frac{\mathbf{E}[W(\boldsymbol{r}) \mid \text{known info}]}{\boldsymbol{r}^2} \, \mathrm{d}\boldsymbol{r}$$

WINE

Work Integral Number Equality

seen by Gittins glasses

$$N = \int_0^\infty \frac{\mathbf{E}[W(r) \mid \text{known info}]}{r^2} \, \mathrm{d}r$$

WINE

Work Integral Number Equality

seen by Gittins glasses

$$N = \int_0^\infty \frac{\mathbf{E}[W(r) \mid \text{known info}]}{r^2}\, \mathrm{d}r$$

“Definition”: a job’s **rank** under **Gittins** is whatever makes **WINE** true

WINE implies Gittins's optimality

$$N = \int_0^\infty \frac{\mathbf{E}[W(r) \mid \text{known info}]}{r^2} \, \mathrm{d}r$$

WINE implies Gittins's optimality

$$N = \int_0^\infty \frac{\mathbf{E}[W(r) \mid \text{known info}]}{r^2} \, \mathrm{d}r$$

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

WINE implies Gittins's optimality

$$N = \int_0^\infty \frac{\mathbf{E}[W(r) \mid \text{known info}]}{r^2} \, \mathrm{d}r$$

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

How to minimize $\mathbf{E}[W(r)]$?

WINE implies Gittins's optimality

$$N = \int_0^\infty \frac{\mathbf{E}[W(r) \mid \text{known info}]}{r^2} \, \mathrm{d}r$$

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

How to minimize $\mathbf{E}[W(r)]$?

Prioritize **rank** $\leq r$ before **rank** $> r$

WINE implies Gittins's optimality

$$N = \int_0^\infty \frac{\mathbf{E}[W(r) \mid \text{known info}]}{r^2}\, \mathrm{d}r$$

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

… in the M/G/1

How to minimize $\mathbf{E}[W(r)]$?

Prioritize **rank** $\leq r$ before **rank** $> r$

Contributions

WINE

queueing identity for
understanding **Gittins**

- non-M/G/1 queues
- imperfect implementation
- unknown job size distribution/model

Contributions

WINE

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

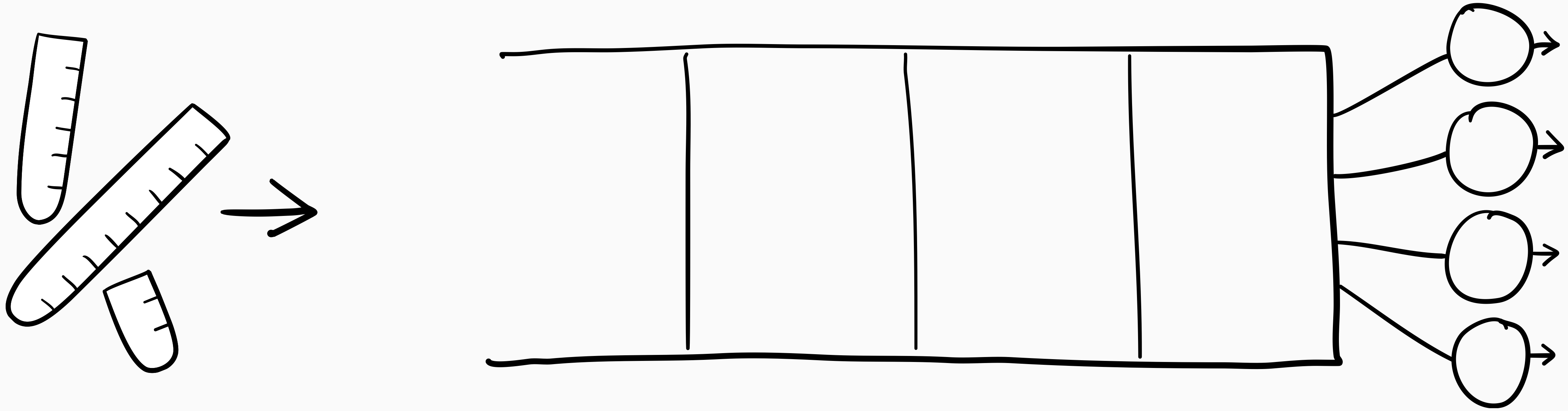
queueing identity for
understanding **Gittins**

? non-M/G/1 queues

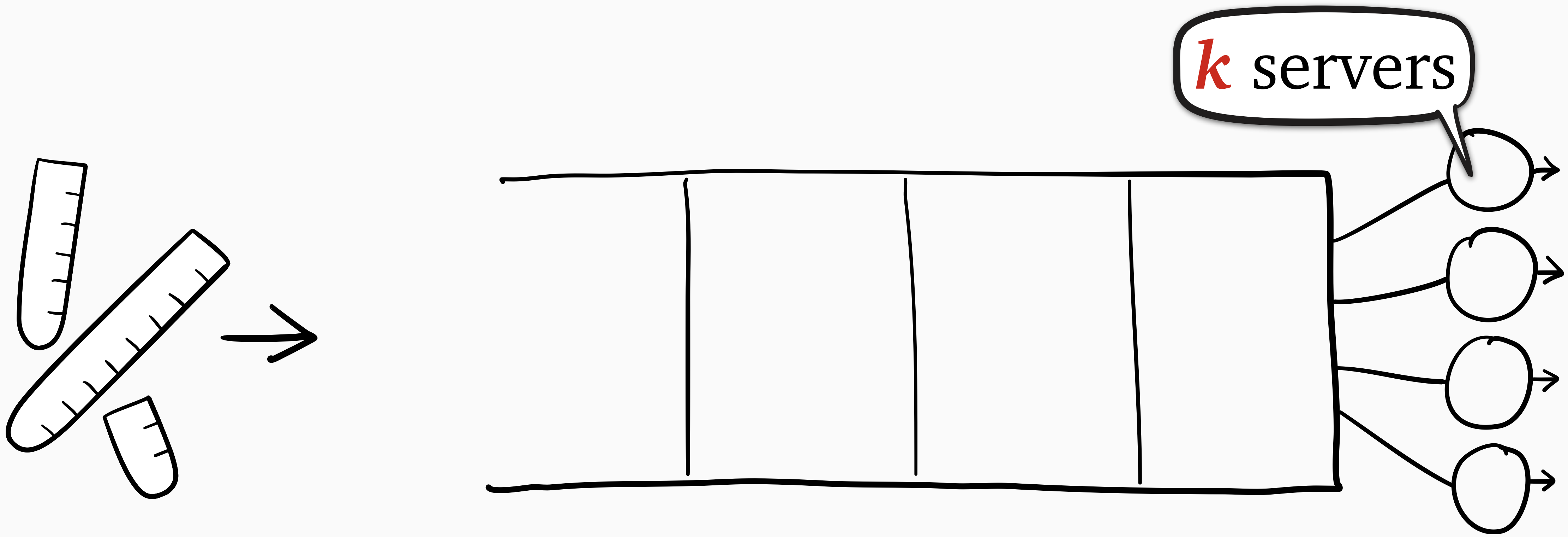
? imperfect implementation

? unknown job size distribution/model

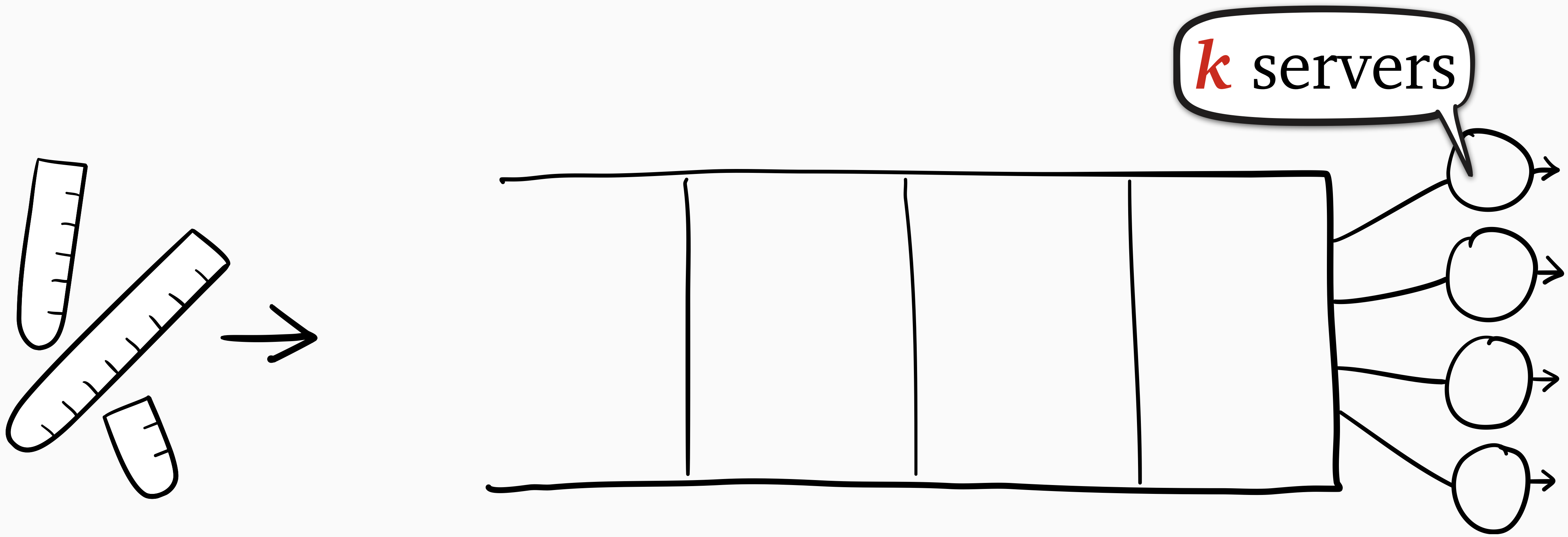
Multiserver scheduling (M/G/*k*)



Multiserver scheduling (M/G/*k*)

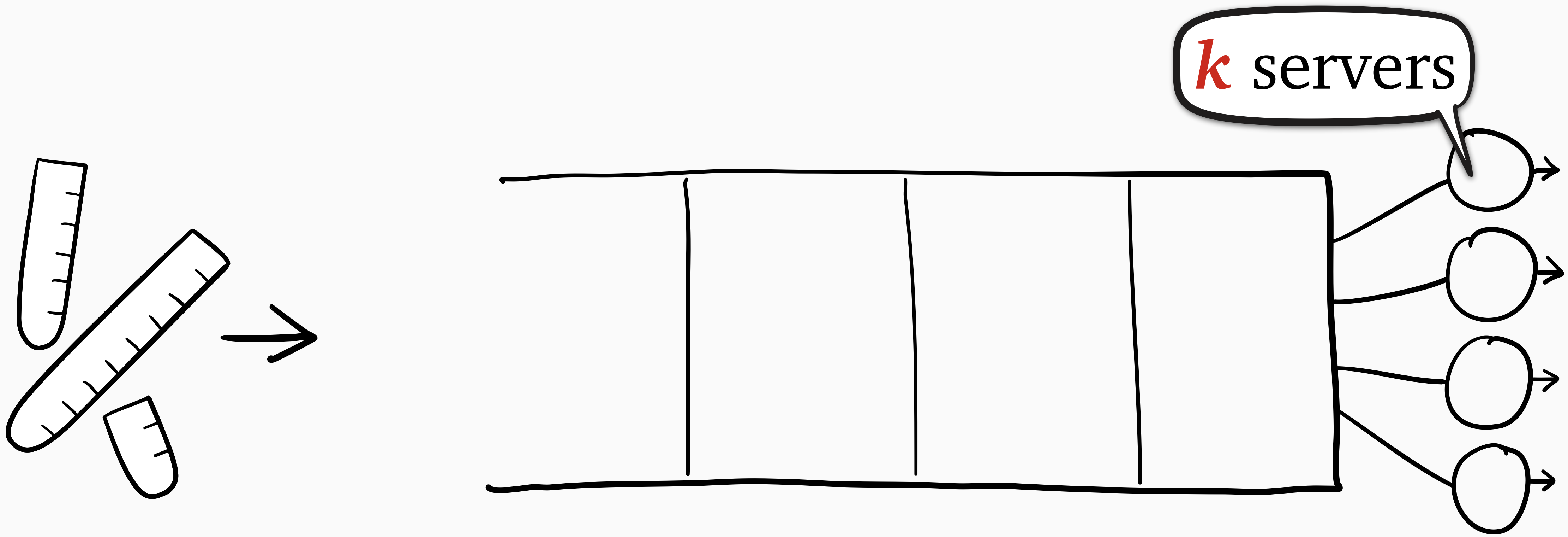


Multiserver scheduling (M/G/*k*)



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

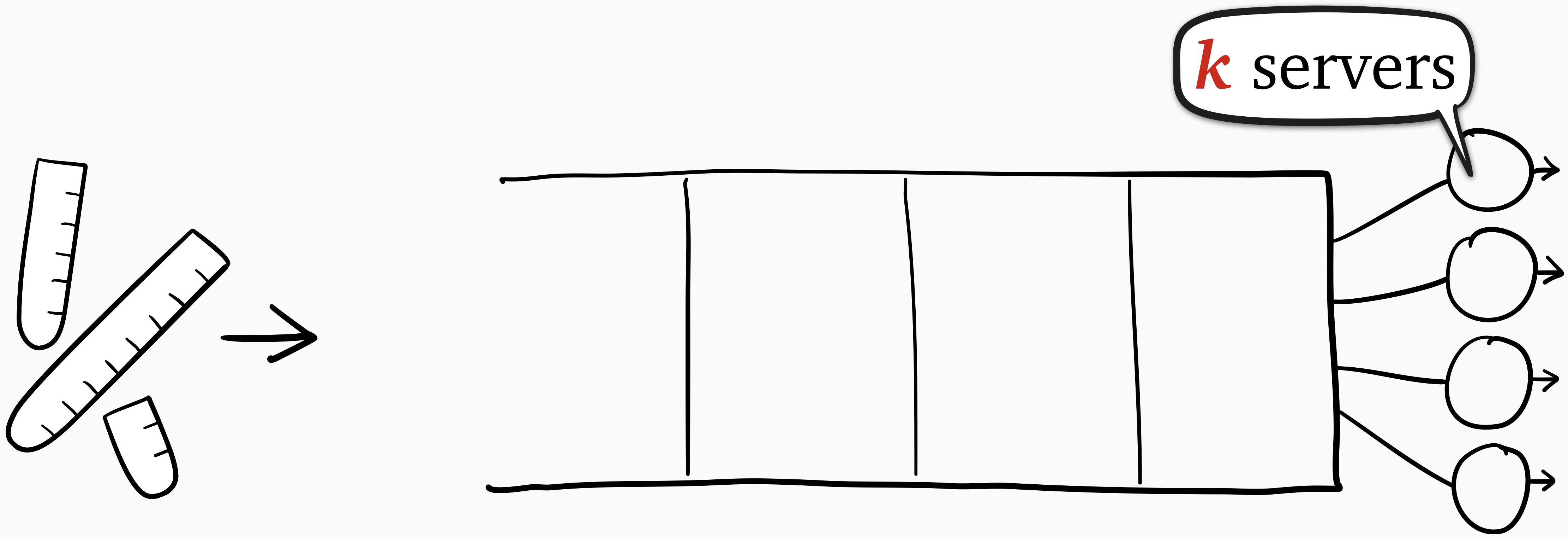
Multiserver scheduling (M/G/*k*)



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

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

Multiserver scheduling (M/G/k)

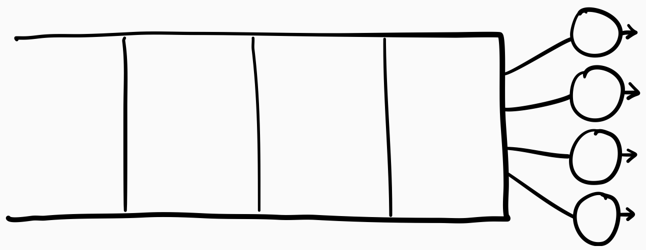


Gittins-1 (single-server): serves the 1 job of least **rank**

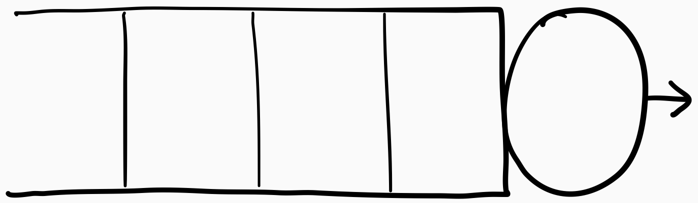
Gittins-k (multiserver): serves the k jobs of least **rank**

Is **Gittins**-k near-optimal in the M/G/k?

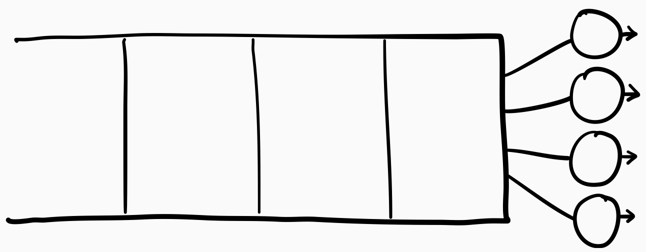
Gittins-*k*



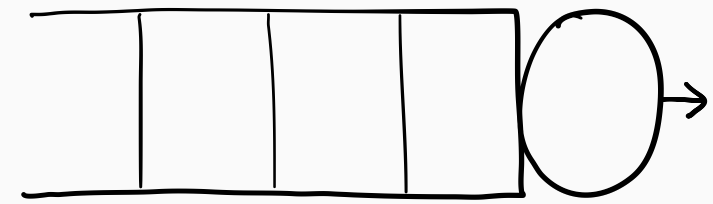
Gittins-1

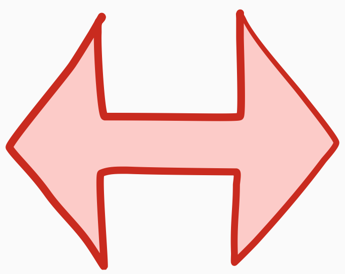


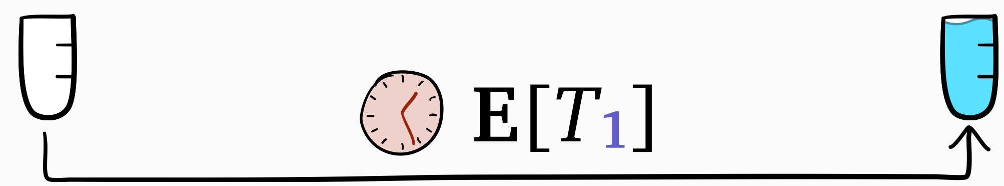
Gittins-k

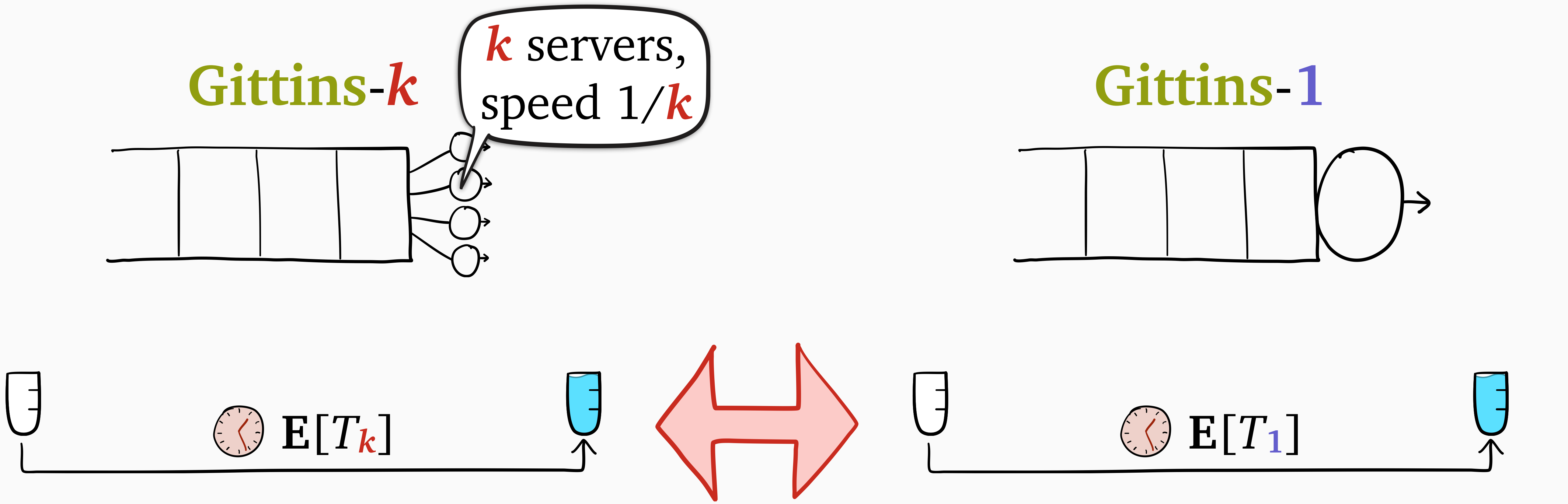


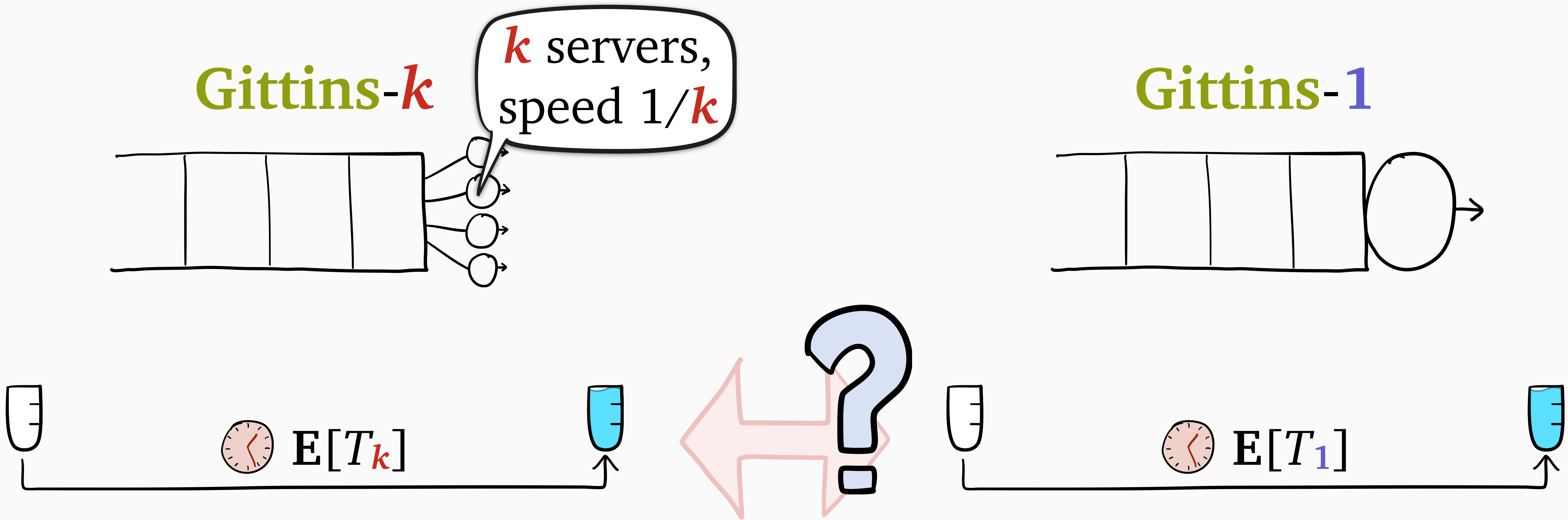
Gittins-1

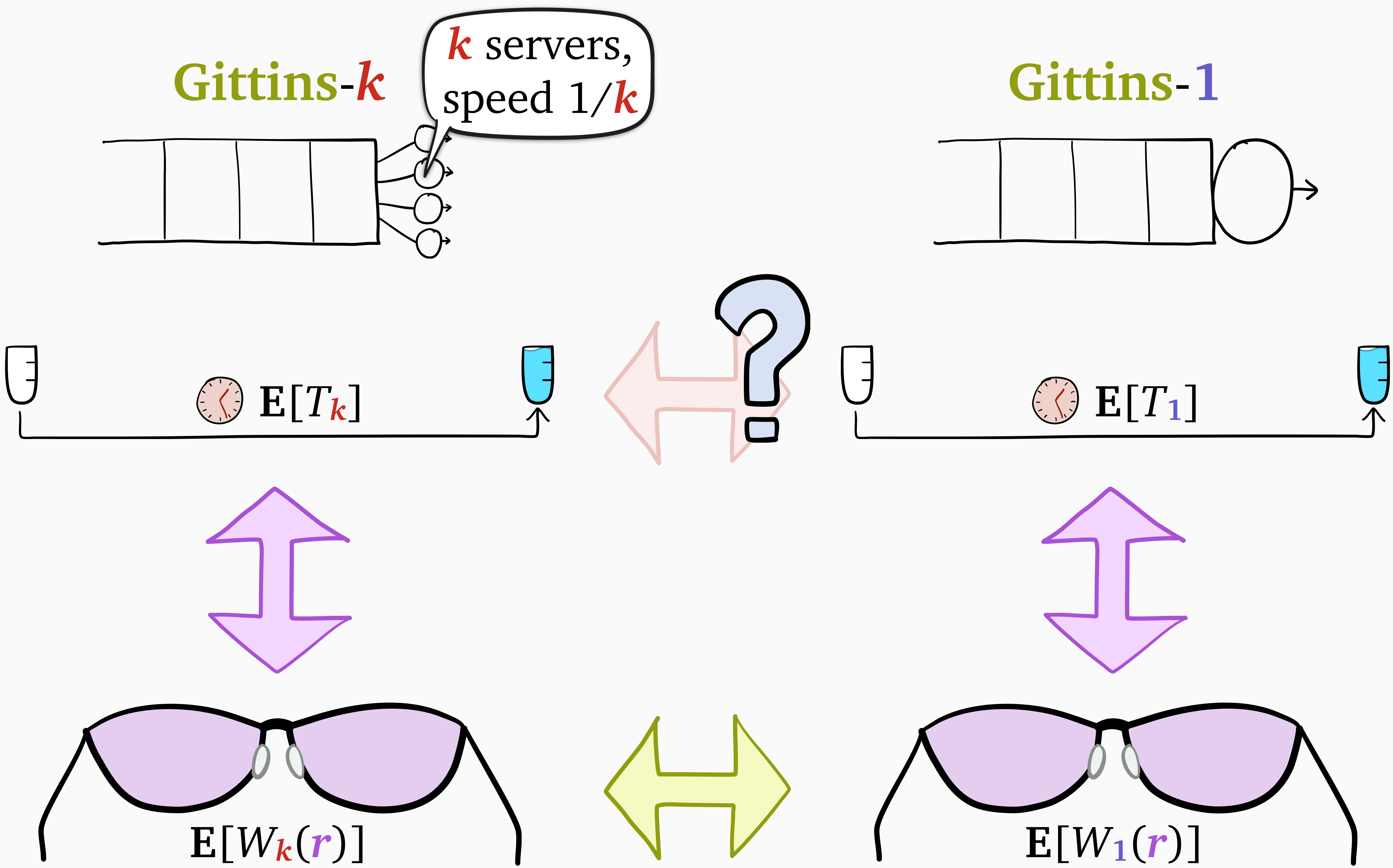




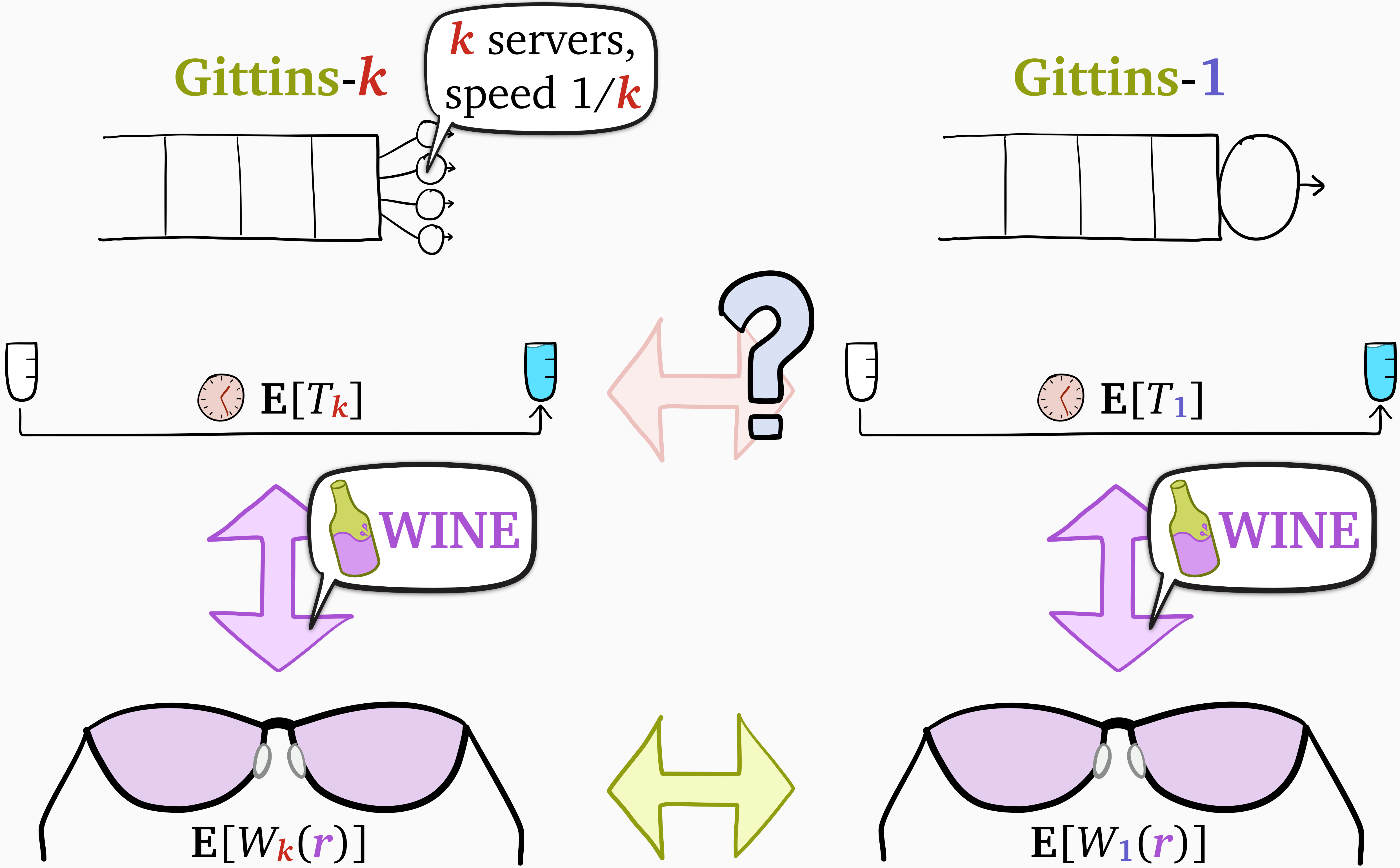


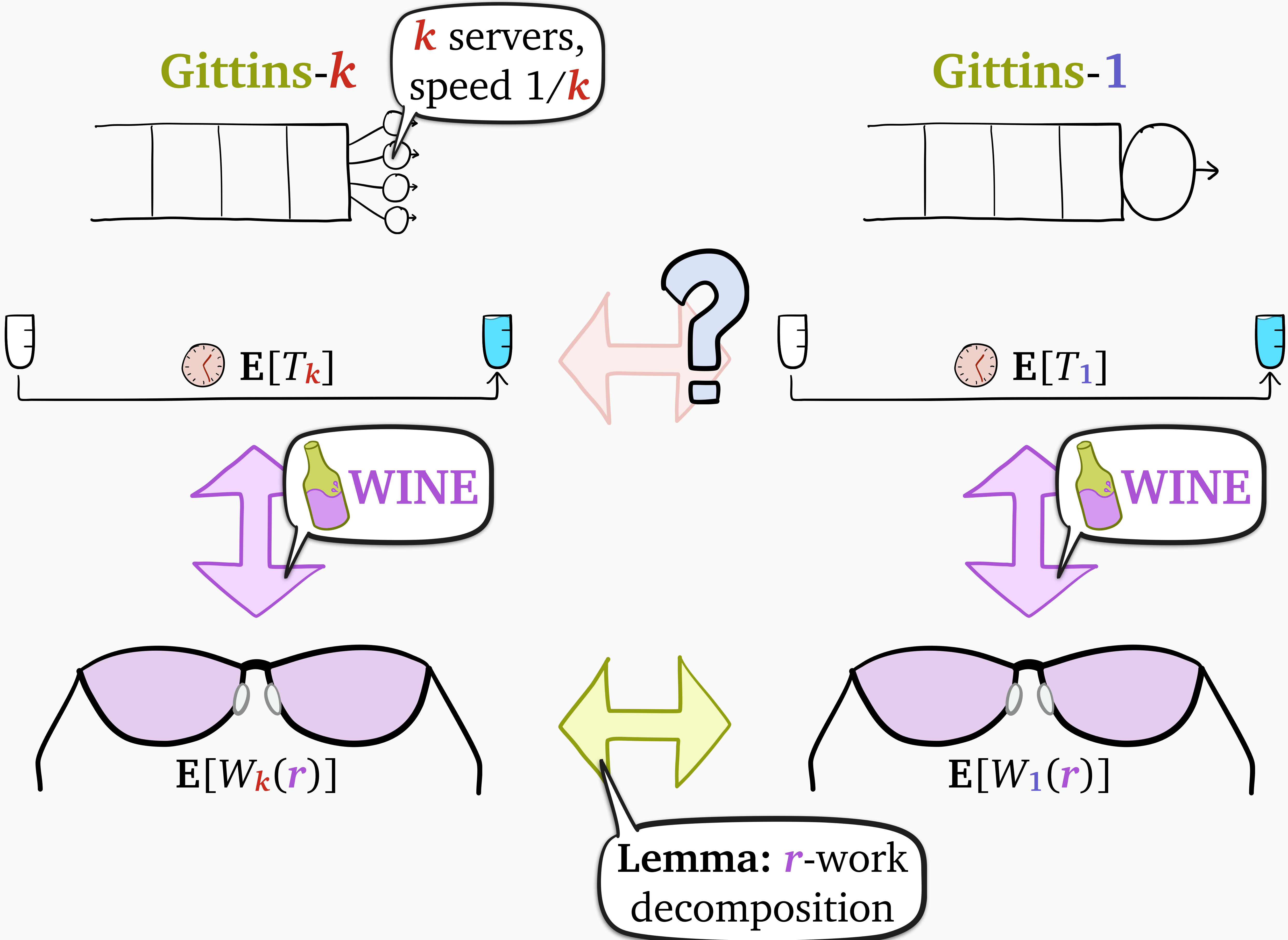

Gittins-k
k servers,
speed 1/k
Gittins-1
E[T_k]
E[T_1]

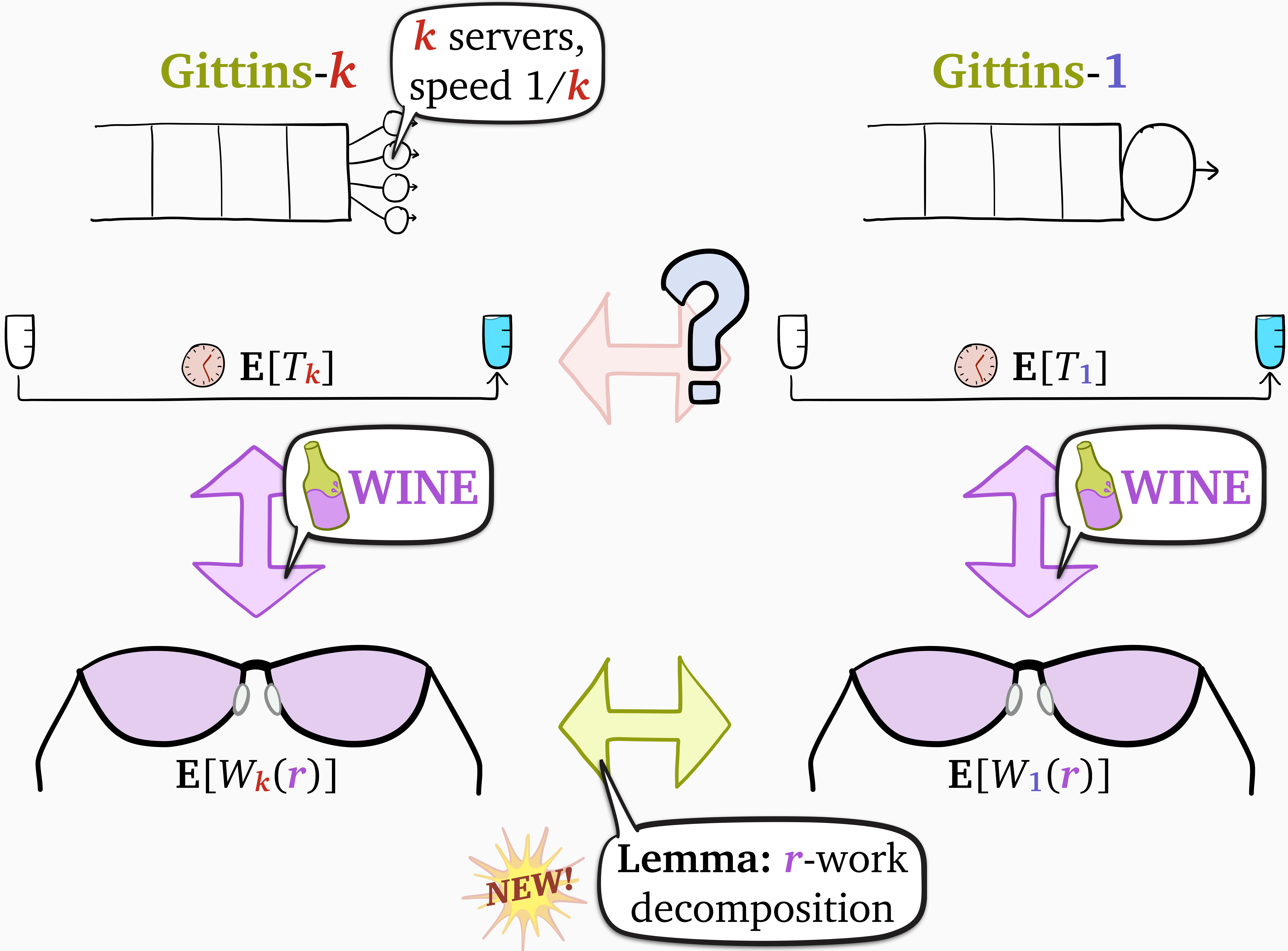

Gittins-k
k servers, speed $1/k$
Gittins-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$

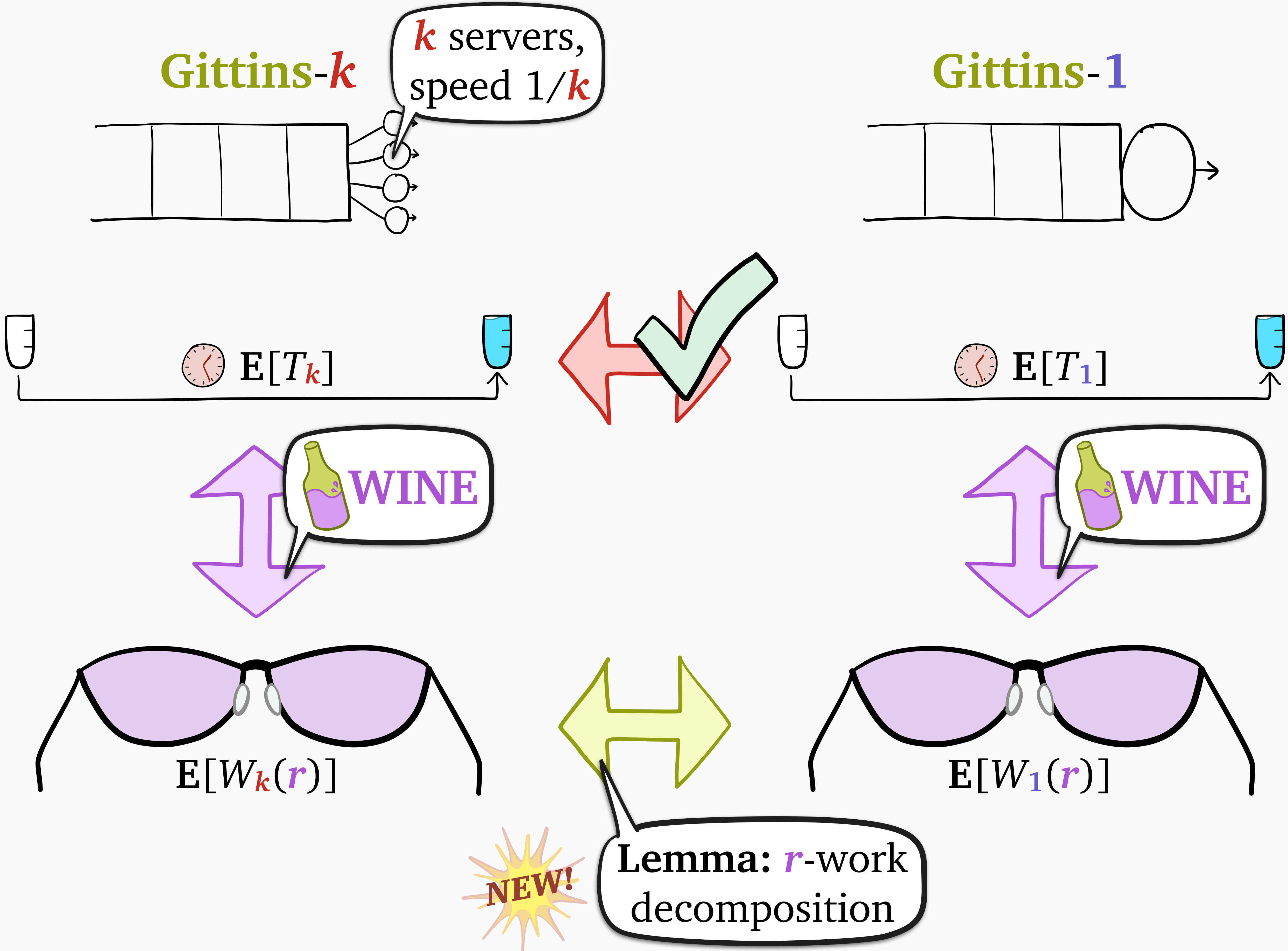


Gittins-k
k servers, speed $1/k$
Gittins-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$
$\mathbf{E}[W_k(r)]$
$\mathbf{E}[W_1(r)]$


Gittins-k
k servers, speed 1/k
Gittins-1
E[T_k]
E[T_1]
WINE
WINE
E[W_k(r)]
E[W_1(r)]


Gittins-k
k servers, speed 1/k
Gittins-1
E[T_k]
E[T_1]
WINE
WINE
E[W_k(r)]
E[W_1(r)]
Lemma: r-work decomposition


Gittins-k
k servers, speed 1/k
Gittins-1
E[T_k]
E[T_1]
WINE
WINE
E[W_k(r)]
E[W_1(r)]
NEW!
Lemma: r-work decomposition


Gittins-k
k servers, speed $1/k$
Gittins-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

M/G/k suboptimality gap

Lemma: under **Gittins**,

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

M/G/k suboptimality gap

Lemma: under **Gittins**,

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

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

M/G/k suboptimality gap

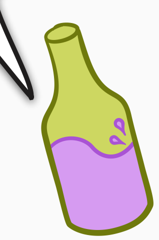
Lemma: under Gittins,

$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \mathbf{E}[\text{``}{\leq}\ k-1 \text{ jobs' } r\text{-work''}]$$

Theorem: under Gittins,

$$\mathbf{E}[N_k] \leq \mathbf{E}[N_1] + (k-1)$$

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



M/G/k suboptimality gap

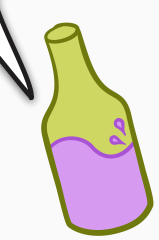
Lemma: under **Gittins**,

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

Theorem: under **Gittins**,

$$\mathbf{E}[N_k] \leq \mathbf{E}[N_1] + (k-1) \cdot 3.8 \log \frac{1}{1-\rho}$$

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



M/G/k suboptimality gap

Lemma: under Gittins,

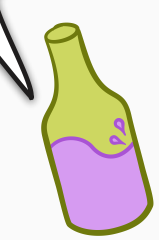
$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \mathbf{E}[\text{“}\leq k-1 \text{ jobs' } r\text{-work”}]$$

Theorem: under Gittins,

$$\mathbf{E}[N_k] \leq \mathbf{E}[N_1] + (k-1) \cdot 3.8 \log \frac{1}{1-\rho}$$

load, a.k.a. utilization

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



M/G/k suboptimality gap

Lemma: under Gittins,

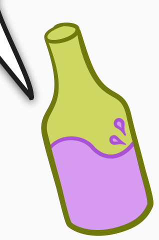
$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \mathbf{E}[\text{“}\le k-1 \text{ jobs’ } r\text{-work”}]$$

Theorem: under Gittins,

$$\mathbf{E}[N_k] \le \mathbf{E}[N_1] + \underbrace{(k-1)\cdot 3.8\log\frac{1}{1-\rho}}_{o(\mathbf{E}[N_1])}$$

load, a.k.a. utilization

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



M/G/k suboptimality gap

Lemma: under Gittins,

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

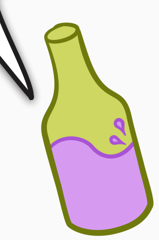
Theorem: under Gittins,

$$\mathbf{E}[N_k] \leq \mathbf{E}[N_1] + \underbrace{(k-1)\cdot 3.8 \log \frac{1}{1-\rho}}_{o(\mathbf{E}[N_1])}$$

load, a.k.a. utilization

Corollary: Gittins-k minimizes $\mathbf{E}[T]$ in M/G/k as $\rho \to 1$

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



Contributions

WINE

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

queueing identity for understanding **Gittins**

First bound for M/G/k
(G/G/k coming soon!)

- non-M/G/1 queues
- imperfect implementation
- unknown job size distribution/model

Multiplicative **rank** errors

Definition: an **approximate Gittins** policy satisfies

$$\boldsymbol{\beta}\,\mathbf{rank}_{\mathbf{Gittins}}(\boldsymbol{x}) \le \mathbf{rank}_{\mathbf{approx}}(\boldsymbol{x}) \le \boldsymbol{\alpha}\,\mathbf{rank}_{\mathbf{Gittins}}(\boldsymbol{x})$$

Multiplicative rank errors

Definition: an approximate Gittins policy satisfies

$$\beta\,\mathrm{rank}_{\mathrm{Gittins}}(x) \le \mathrm{rank}_{\mathrm{approx}}(x) \le \alpha\,\mathrm{rank}_{\mathrm{Gittins}}(x)$$

Lemma:

$$\mathbf{E}[W_{\mathrm{approx}}(r)] \le \mathbf{E}\left[W_{\mathrm{Gittins}}\left(\frac{\alpha}{\beta}r\right)\right]$$

Multiplicative **rank** errors

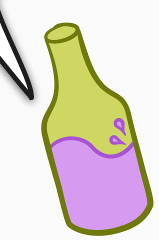
Definition: an approximate Gittins policy satisfies

$$\beta\,\mathbf{rank}_{\mathrm{Gittins}}(x) \le \mathbf{rank}_{\mathrm{approx}}(x) \le \alpha\,\mathbf{rank}_{\mathrm{Gittins}}(x)$$

Lemma:

$$\mathbf{E}[W_{\mathrm{approx}}(r)] \le \mathbf{E}\left[W_{\mathrm{Gittins}}\left(\frac{\alpha}{\beta}r\right)\right]$$

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



Multiplicative rank errors

Definition: an approximate Gittins policy satisfies

$$\beta\,\mathrm{rank}_{\mathrm{Gittins}}(x) \le \mathrm{rank}_{\mathrm{approx}}(x) \le \alpha\,\mathrm{rank}_{\mathrm{Gittins}}(x)$$

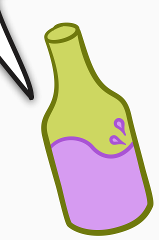
Lemma:

$$\mathbf{E}[W_{\mathrm{approx}}(r)] \le \mathbf{E}\left[W_{\mathrm{Gittins}}\left(\frac{\alpha}{\beta} r\right)\right]$$

Theorem:

$$\mathbf{E}[N_{\mathrm{approx}}] \le \frac{\alpha}{\beta}\mathbf{E}[N_{\mathrm{Gittins}}]$$

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



Contributions

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

WINE

queueing identity for
understanding **Gittins**

First bound for M/G/k
(G/G/k coming soon!)

? non-M/G/1 queues

? imperfect implementation

? unknown job size distribution/model

Contributions

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

WINE

queueing identity for understanding **Gittins**

First bound for M/G/k (G/G/k coming soon!)

? non-M/G/1 queues

multiplicative error → approximation ratio

? imperfect implementation

? unknown job size distribution/model

Contributions

$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\, dr$$

WINE

queueing identity for
understanding **Gittins**

First bound for M/G/k
(G/G/k coming soon!)

? non-M/G/1 queues

multiplicative error
→ approximation ratio

? imperfect implementation

? unknown job size distribution/model

using size estimates
without distribution

Contributions

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

WINE

queueing identity for
understanding **Gittins**

First bound for M/G/k
(G/G/k coming soon!)

? non-M/G/1 queues

multiplicative error
⇒ approximation ratio

? imperfect implementation

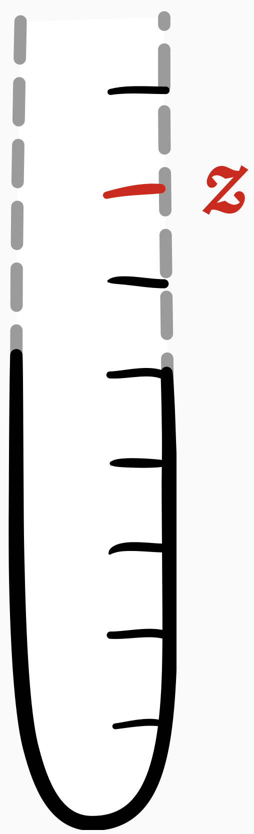
? unknown job size distribution/model

using size estimates
without distribution

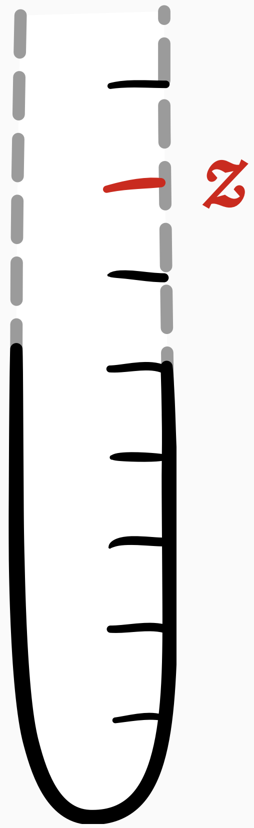
zivscully@cornell.edu
https://ziv.codes

Bonus slides

Noisy size estimates



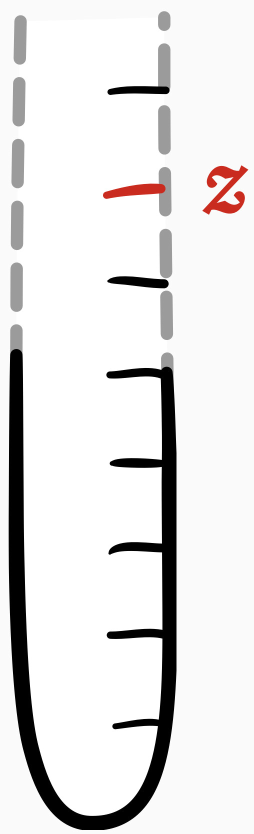
Noisy size estimates



Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $\boldsymbol{z} \in [\beta s, \alpha s]$

Noisy size estimates

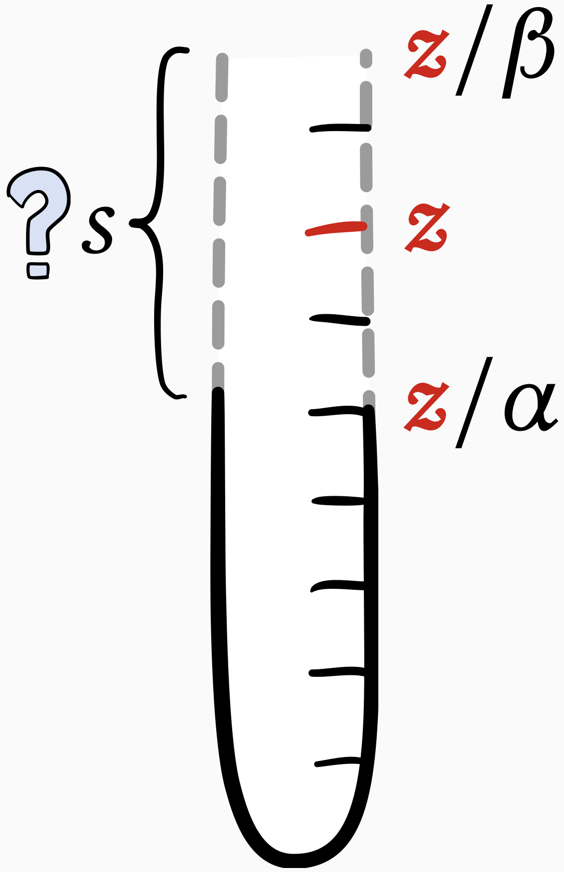


below above

Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

Noisy size estimates

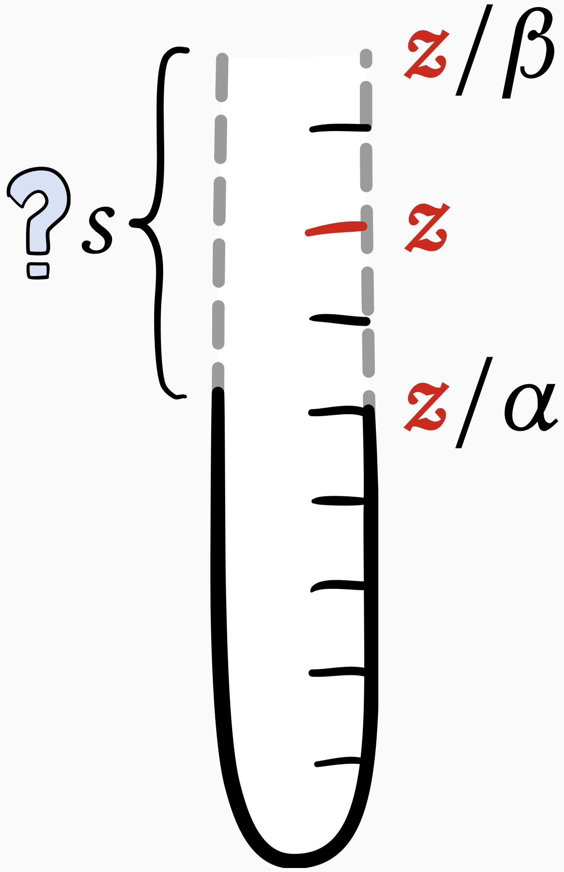


below above

Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

Noisy size estimates



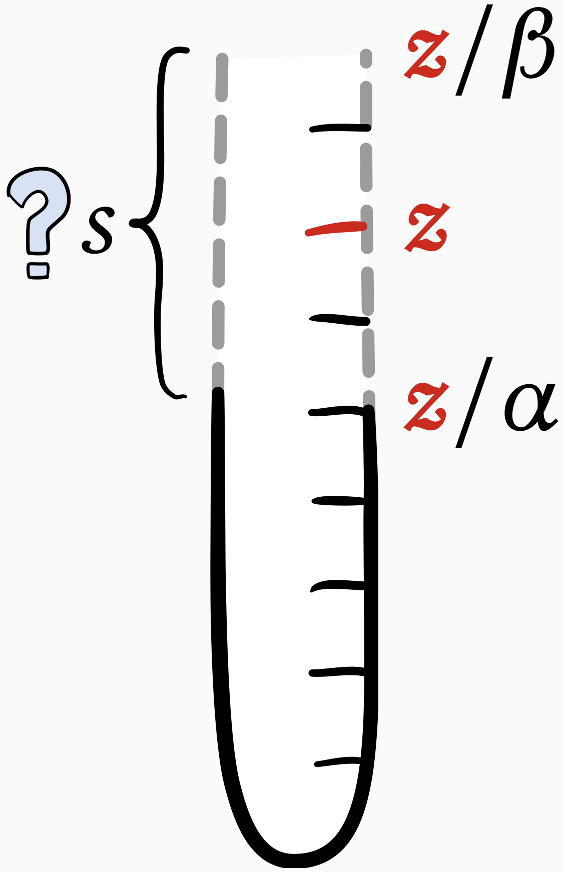
below above

Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

Generated by *joint distribution* (S, Z)

Noisy size estimates



below above

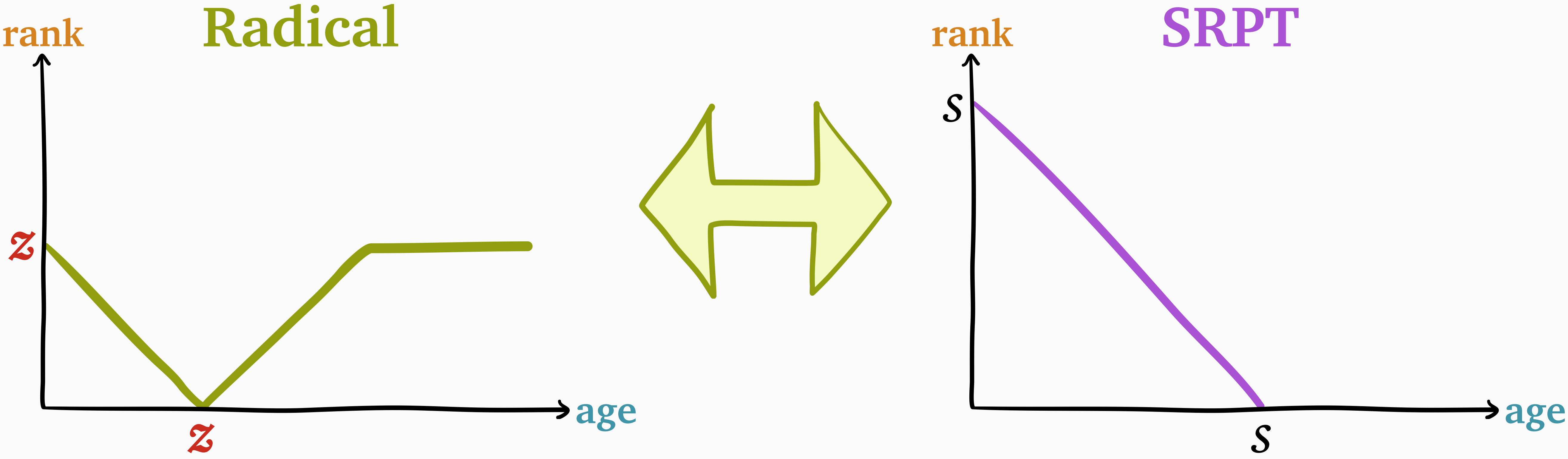
Model: (β, α)-bounded noise

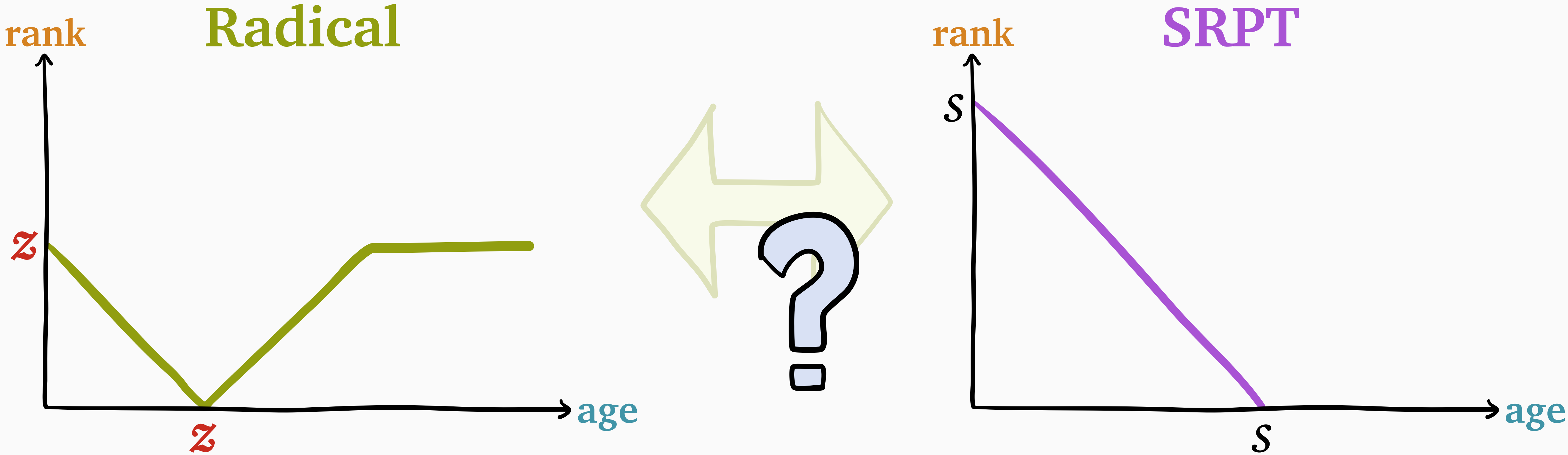
true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

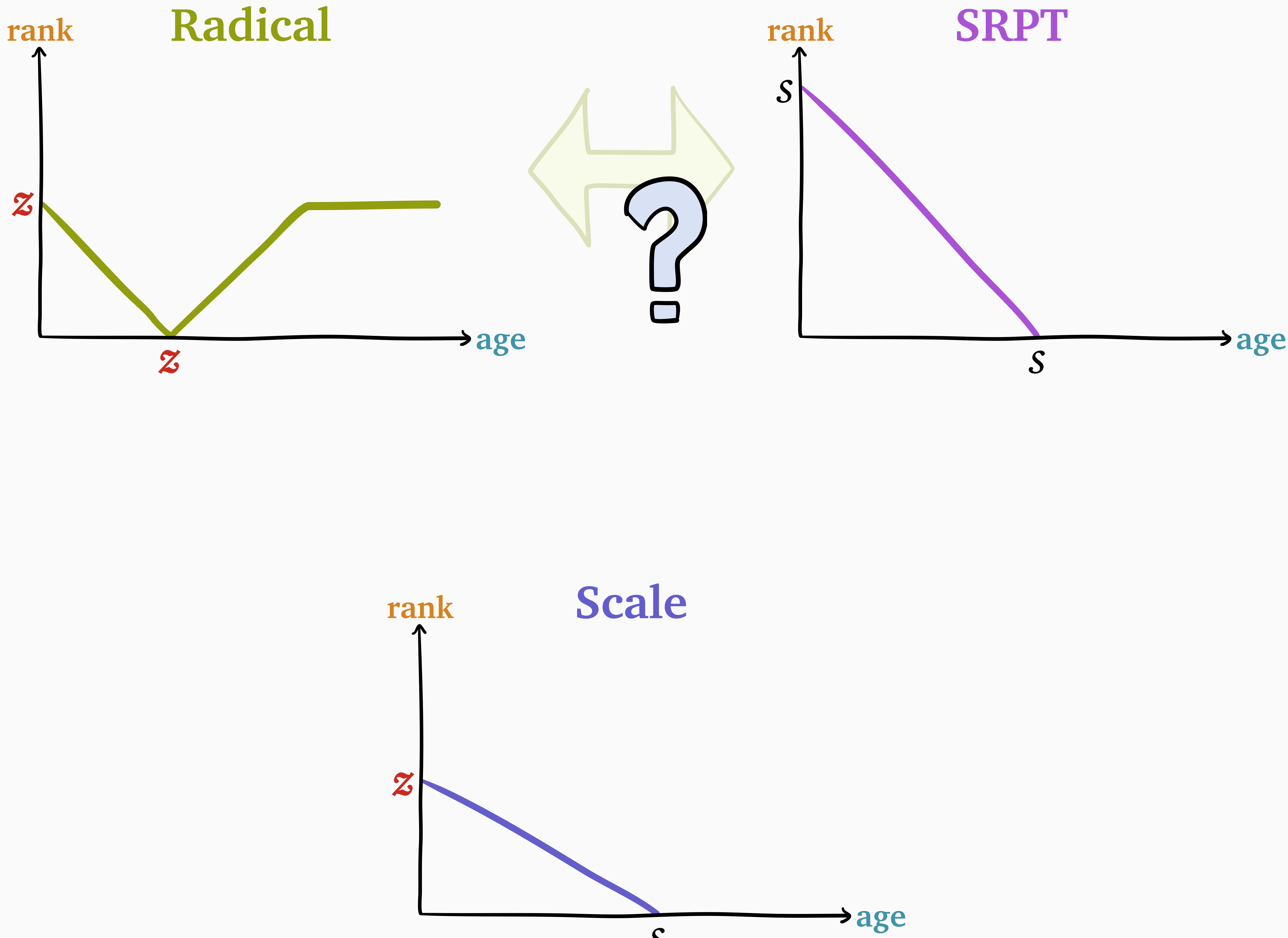
Generated by *joint distribution* (S, Z)

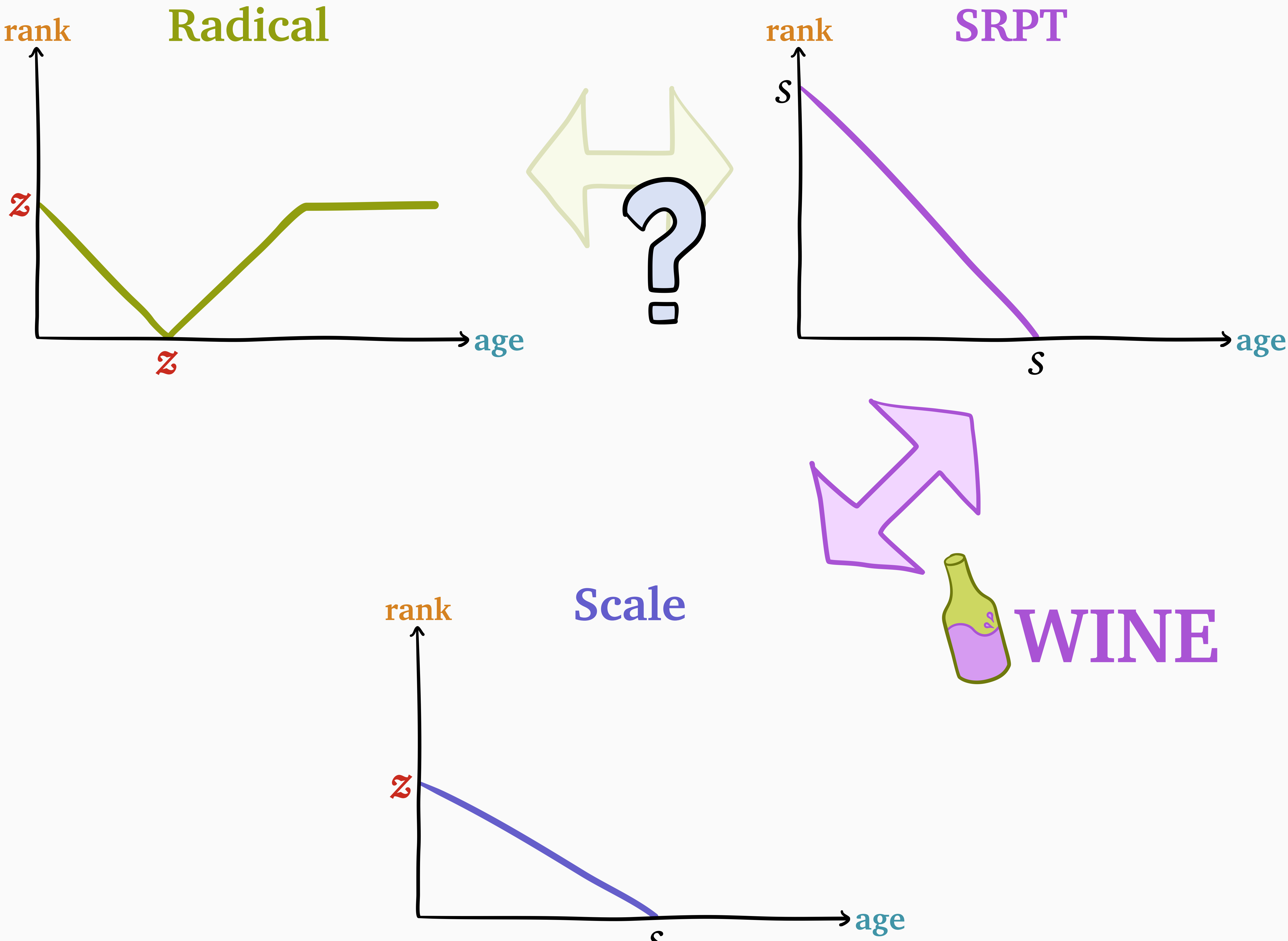
Goal: design a policy with "good" $\mathbf{E}[T]$ for

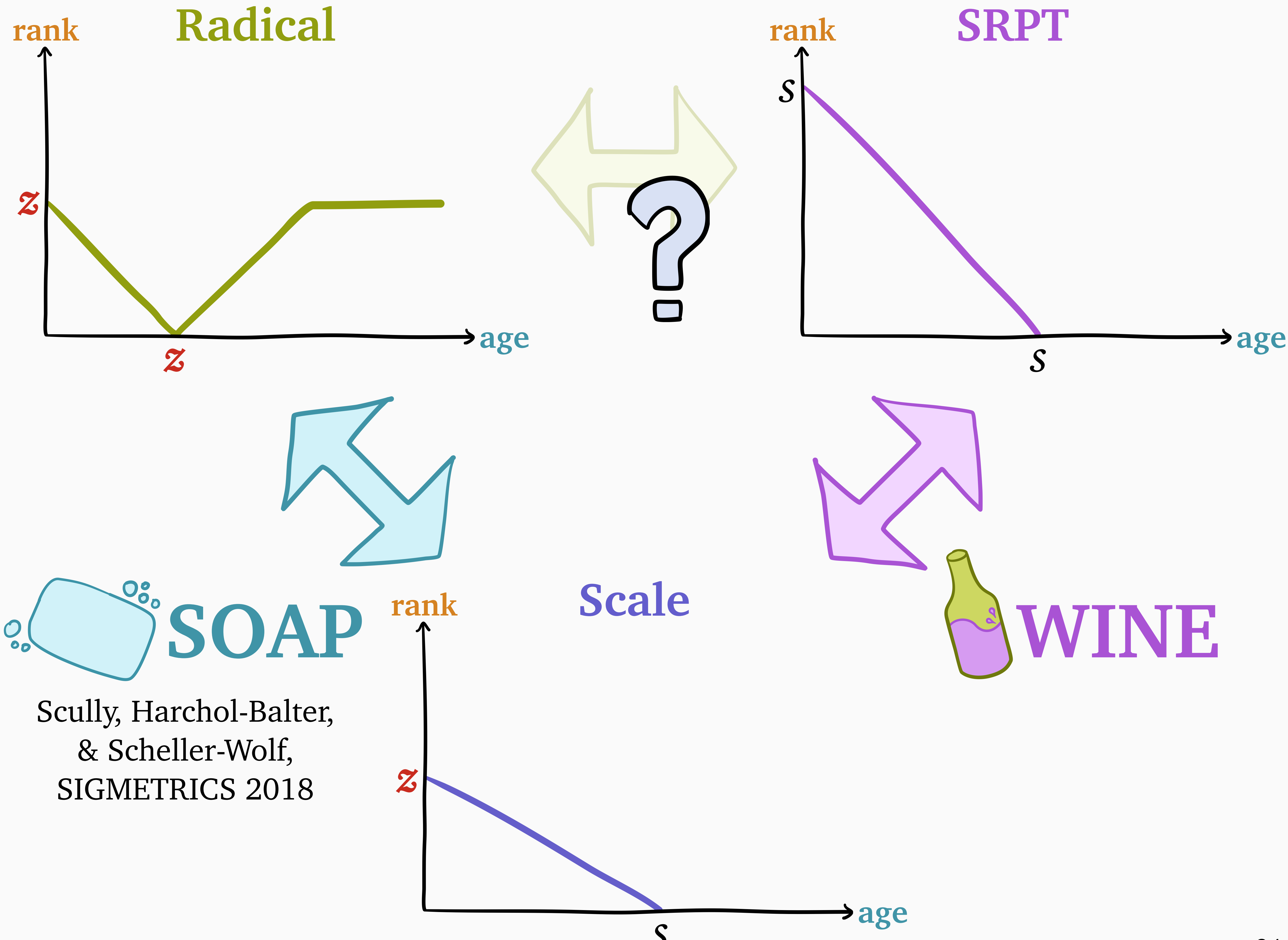
- *any* joint distribution (S, Z)
- *any* values of α, β

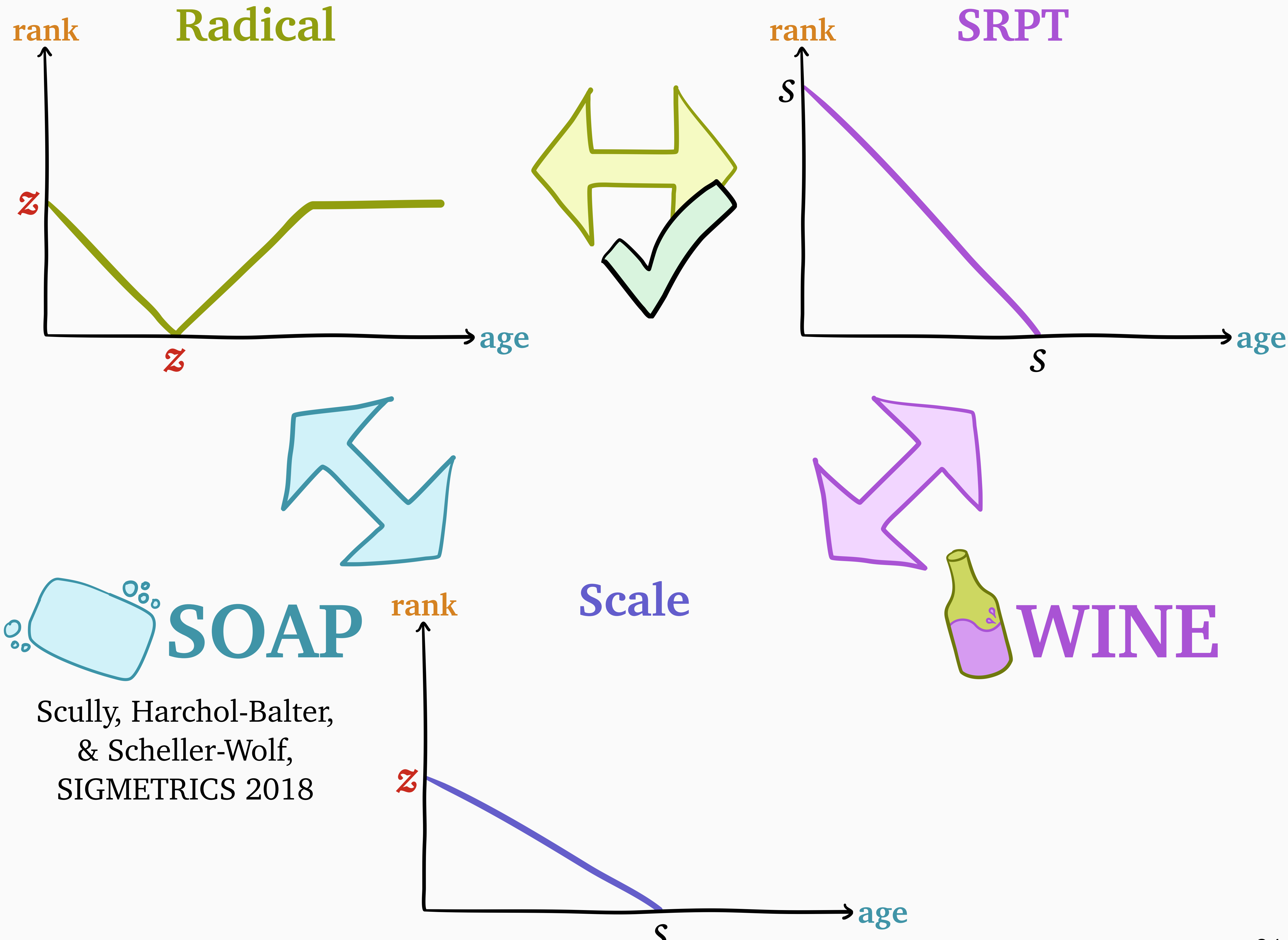

Radical
rank
z
z
age
SRPT
rank
s
s
age


Radical
rank
z
z
age
SRPT
rank
s
s
age


Radical
rank
z
z
age
SRPT
rank
s
s
age
Scale
rank
z
s
age


Radical
rank
z
z
age
SRPT
rank
s
s
age
Scale
rank
z
s
age
WINE


Radical
rank
z
z
age
SRPT
rank
s
s
age
SOAP
Scully, Harchol-Balter,
& Scheller-Wolf,
SIGMETRICS 2018
Scale
rank
z
s
age
WINE


Radical
rank
z
z
age
SRPT
rank
s
s
age
SOAP
Scully, Harchol-Balter,
& Scheller-Wolf,
SIGMETRICS 2018
Scale
rank
z
s
age
WINE