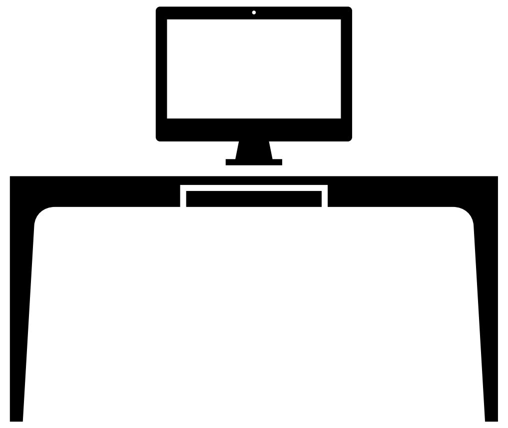
The Public University Secretary Problem

Heather Newman (Carnegie Mellon)

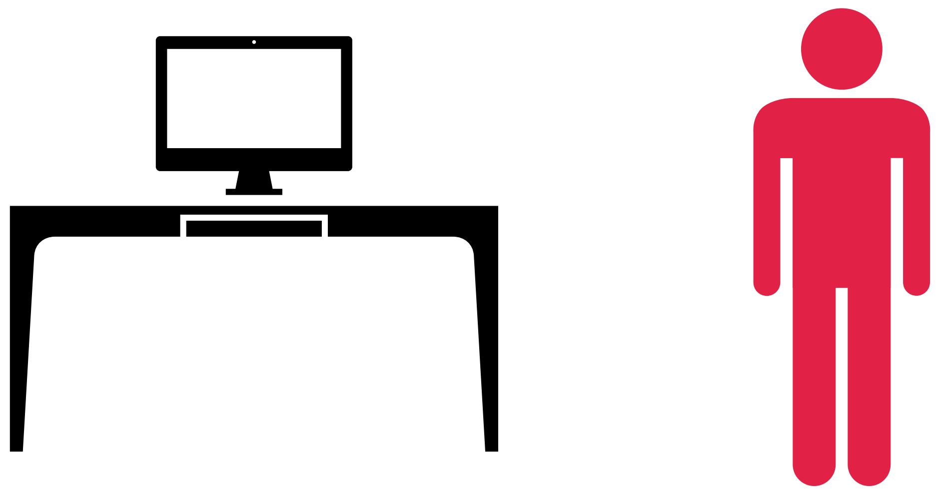
SOSA 2024

Joint work with: Benjamin Moseley (Carnegie Mellon) and Kirk Pruhs (U. of Pittsburgh)

“RAND” (Classic) Secretary Problem

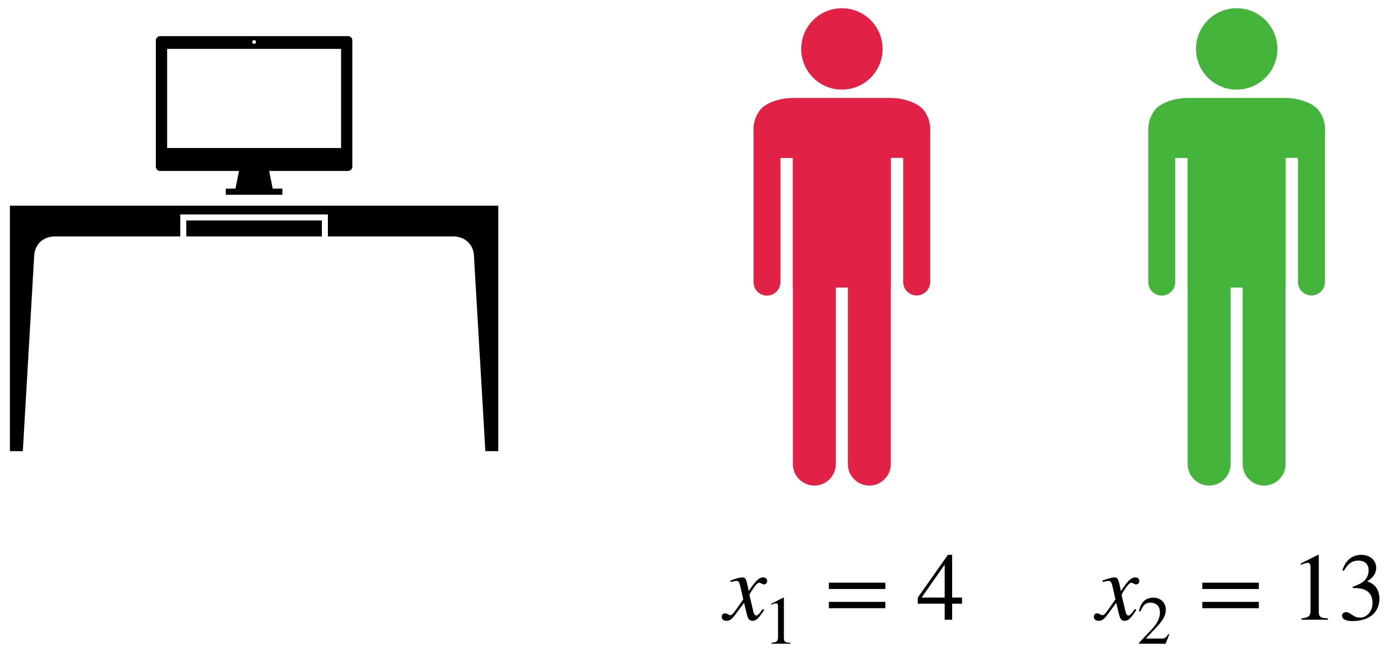


"RAND" (Classic) Secretary Problem

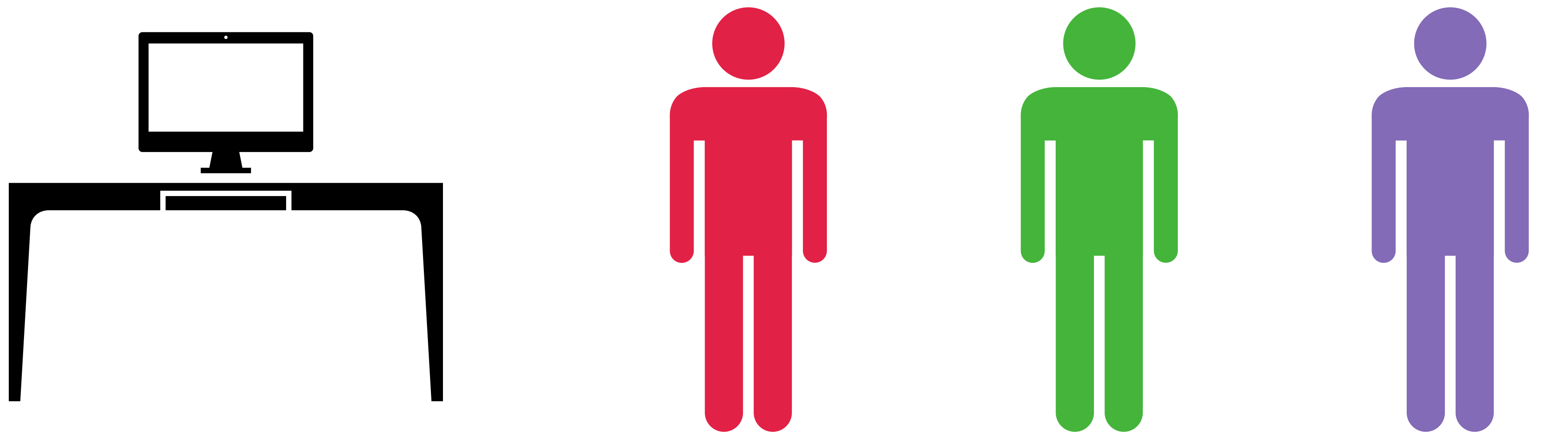


$x_1 = 4$

“RAND” (Classic) Secretary Problem

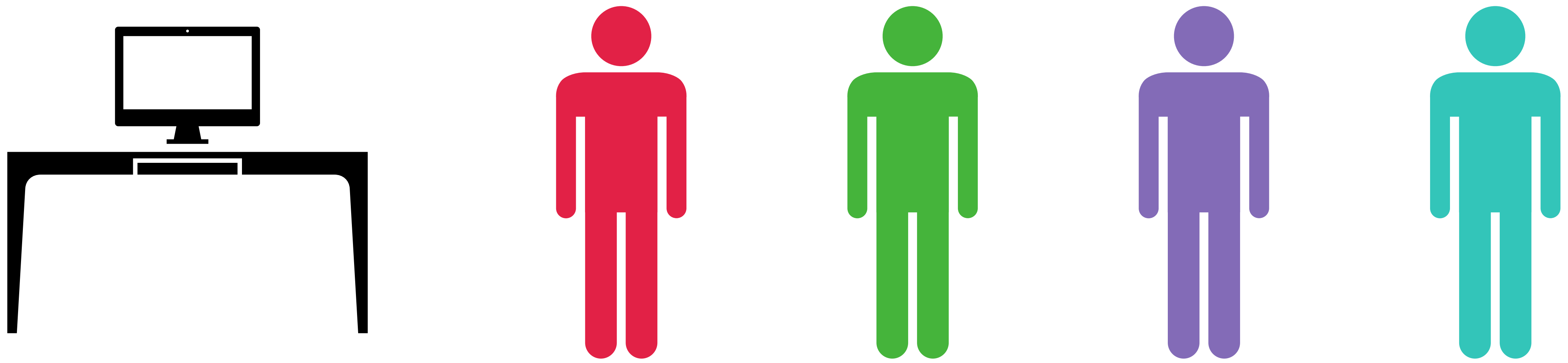


"RAND" (Classic) Secretary Problem



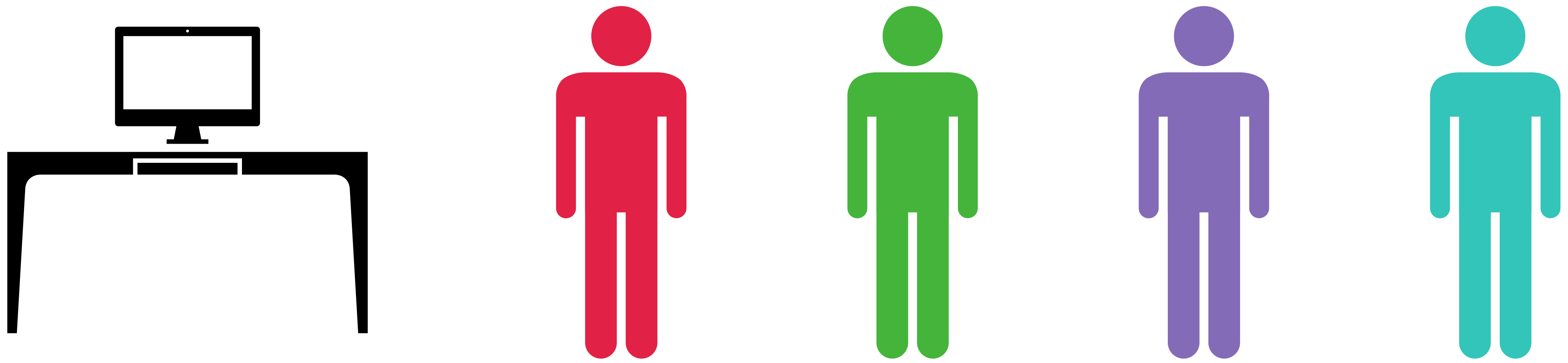
$x_1 = 4$ $x_2 = 13$ $x_3 = 2$

“RAND” (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

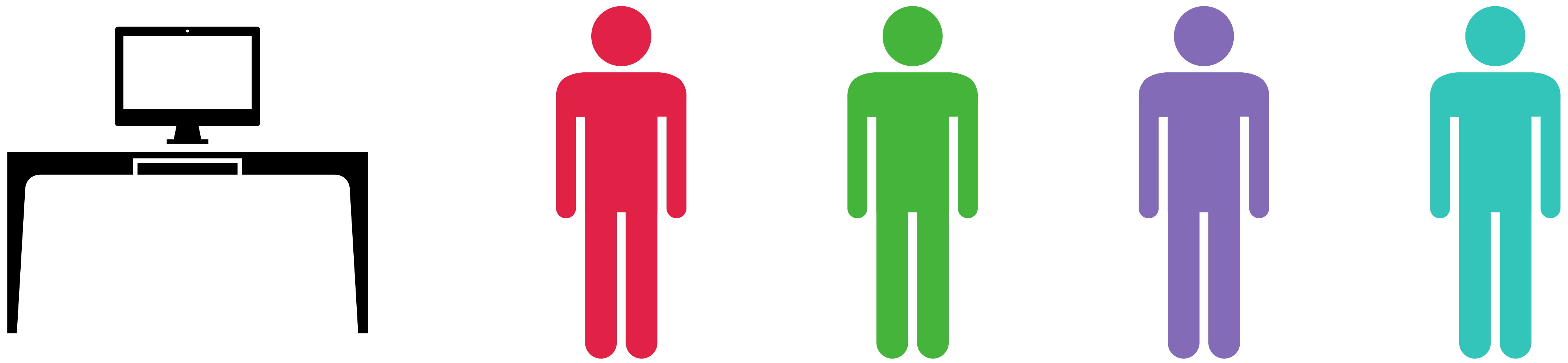
"RAND" (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

- Candidates arrive **online**

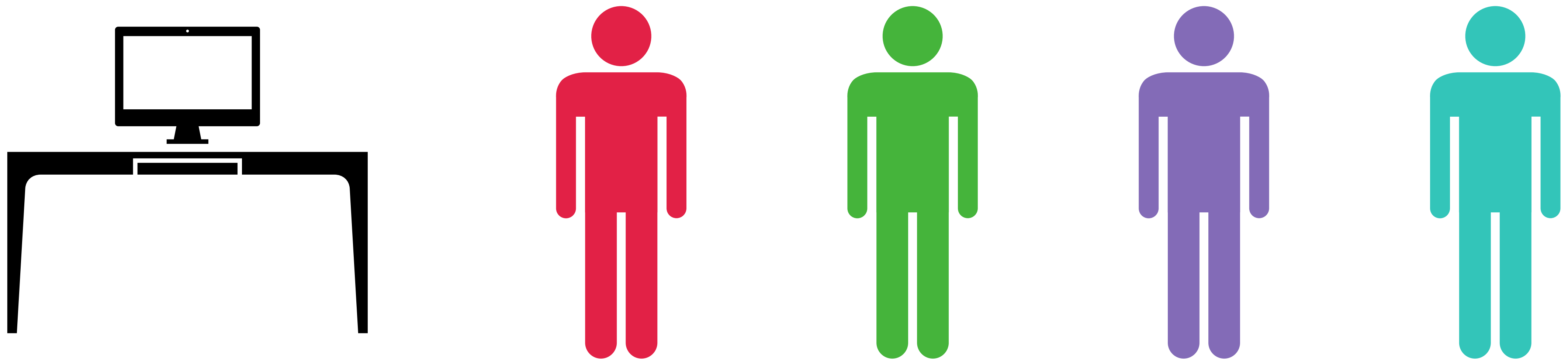
“RAND” (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival

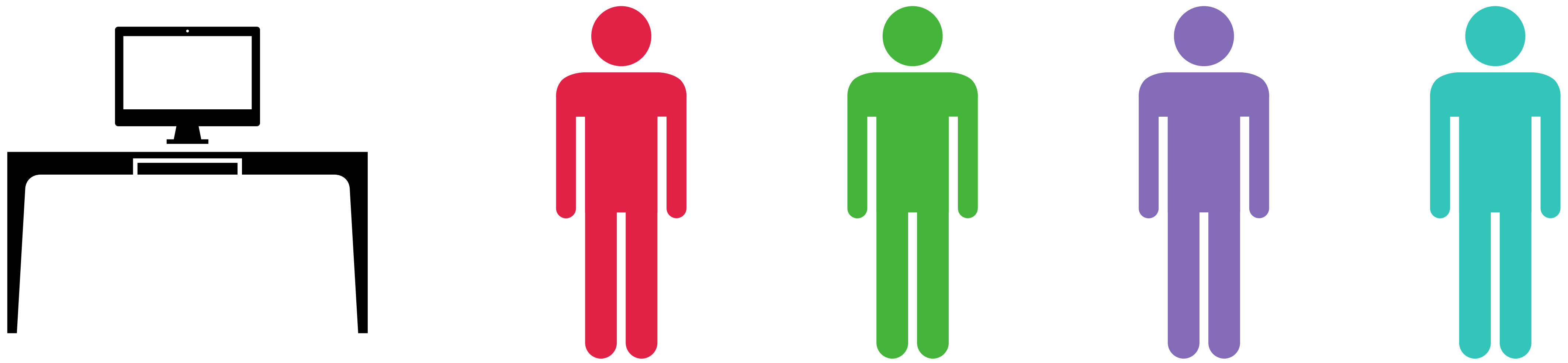
"RAND" (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **at most** k candidates

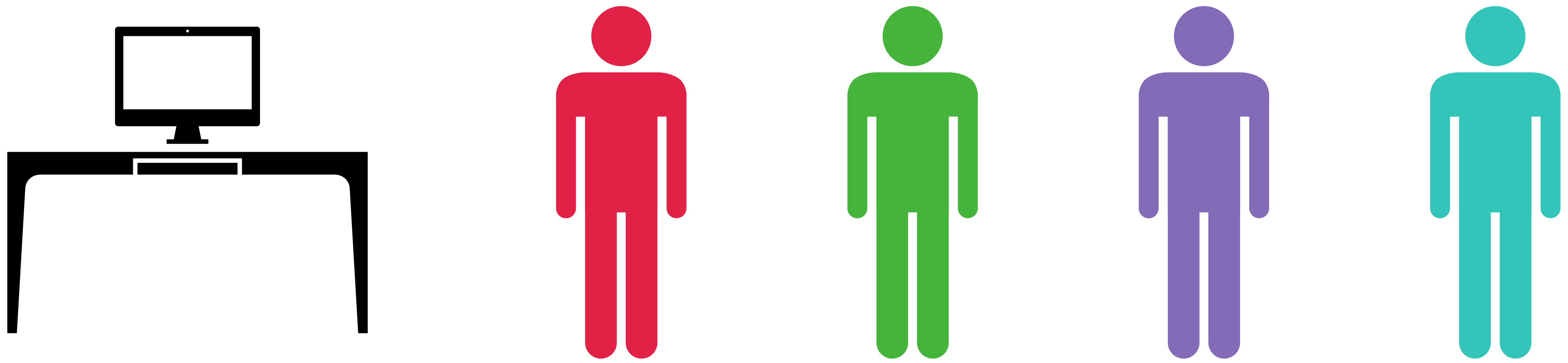
"RAND" (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **at most k** candidates
- x_i = **quality** of candidate i

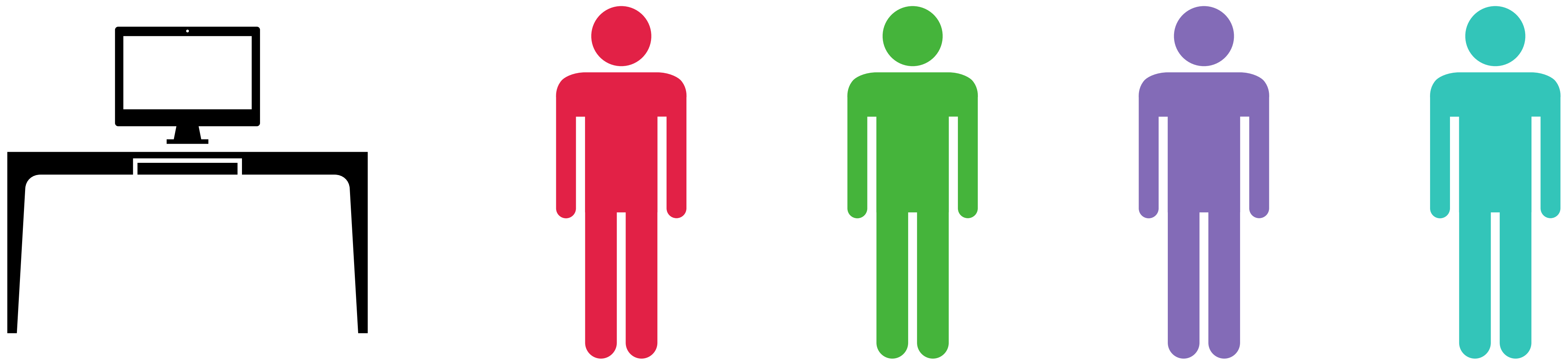
“RAND” (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **at most k** candidates
- x_i = **quality** of candidate i
- Objective: maximize aggregate quality

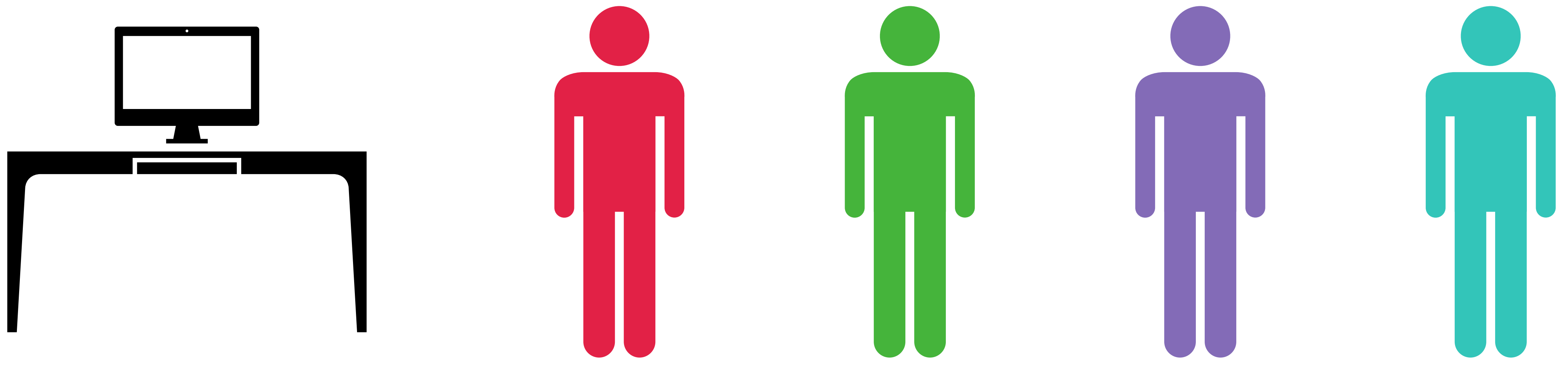
"RAND" (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

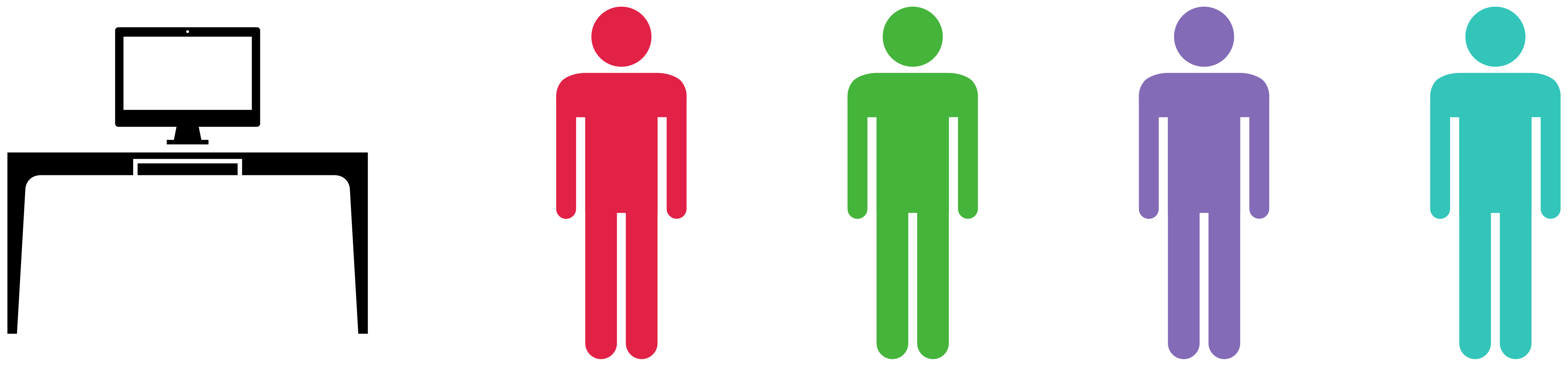
- Choose **at most** k candidates
- x_i = **quality** of candidate i
- Objective: maximize aggregate quality

“RAND” (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

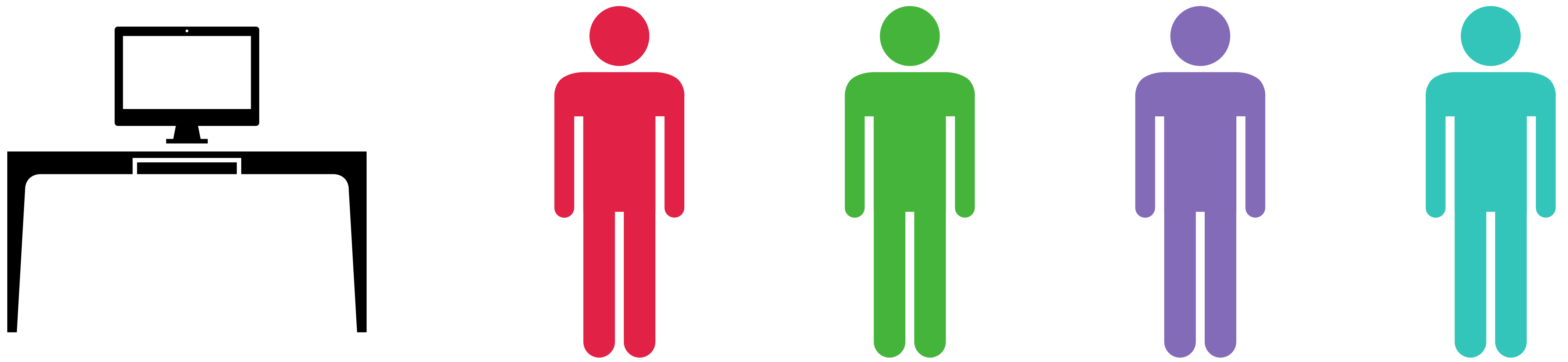
Public University Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

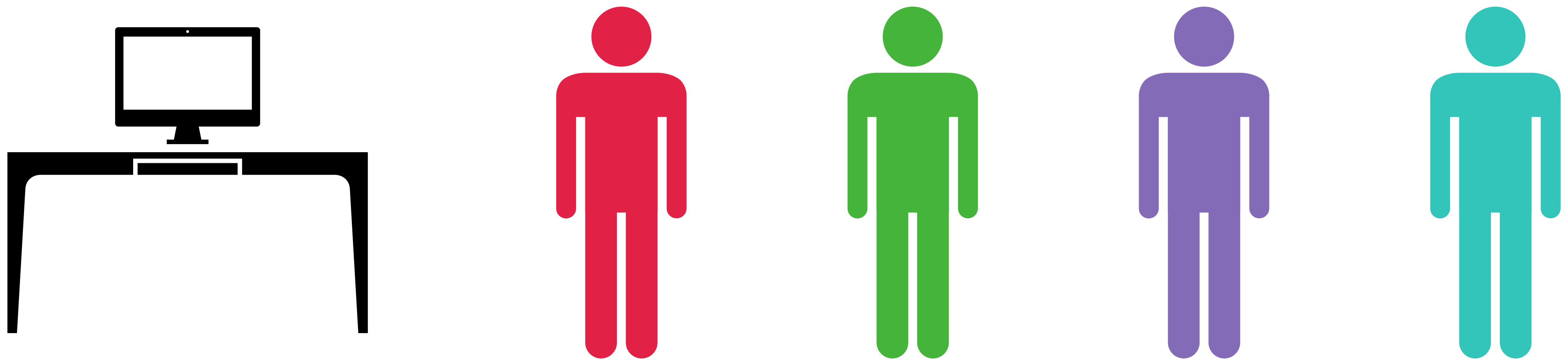
- Choose **at most** k candidates
- x_i = **quality** of candidate i
- Objective: maximize aggregate quality

"RAND" (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

Public University Secretary Problem



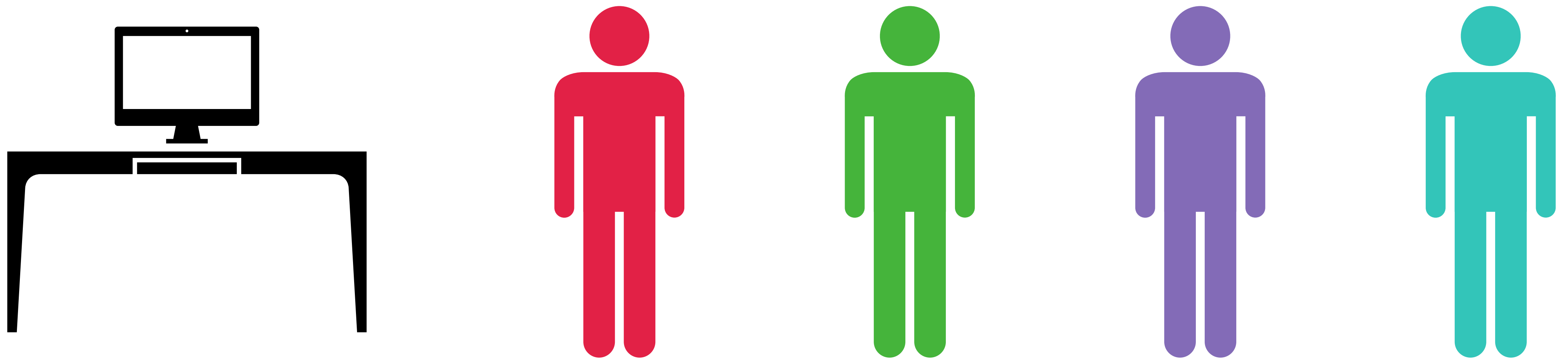
$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

Candidates arrive **online**

Irrevocably accept or reject upon arrival

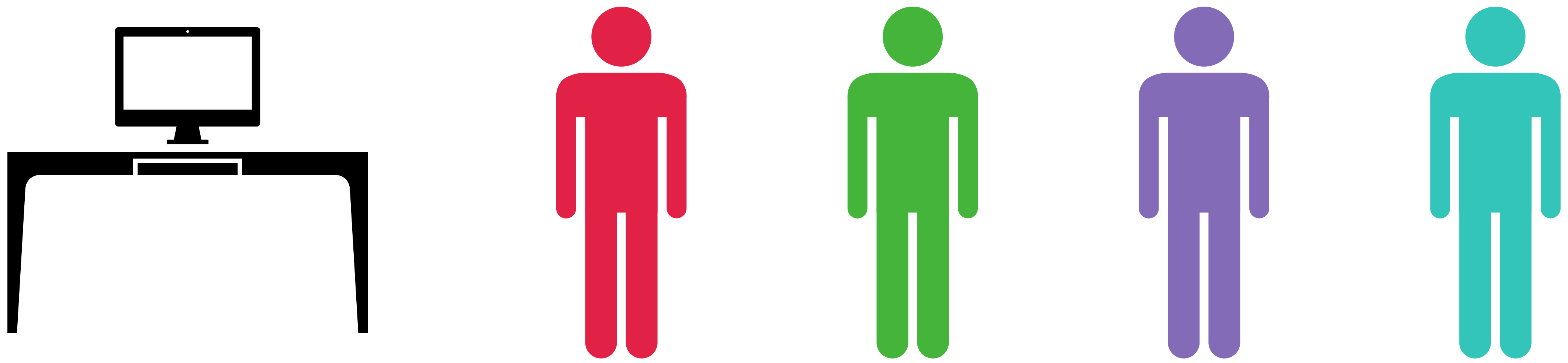
- Choose **at most k** candidates
- x_i = **quality** of candidate i
- Objective: maximize aggregate quality

“RAND” (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

Public University Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

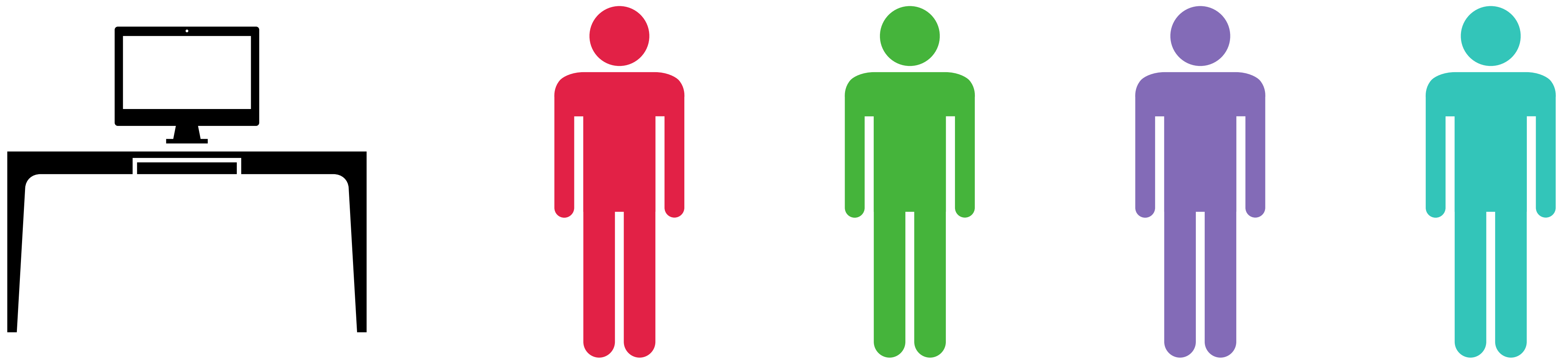
Candidates arrive **online**

Irrevocably accept or reject upon arrival

- Choose **at most** k candidates
- x_i = **quality** of candidate i
- Objective: maximize aggregate quality

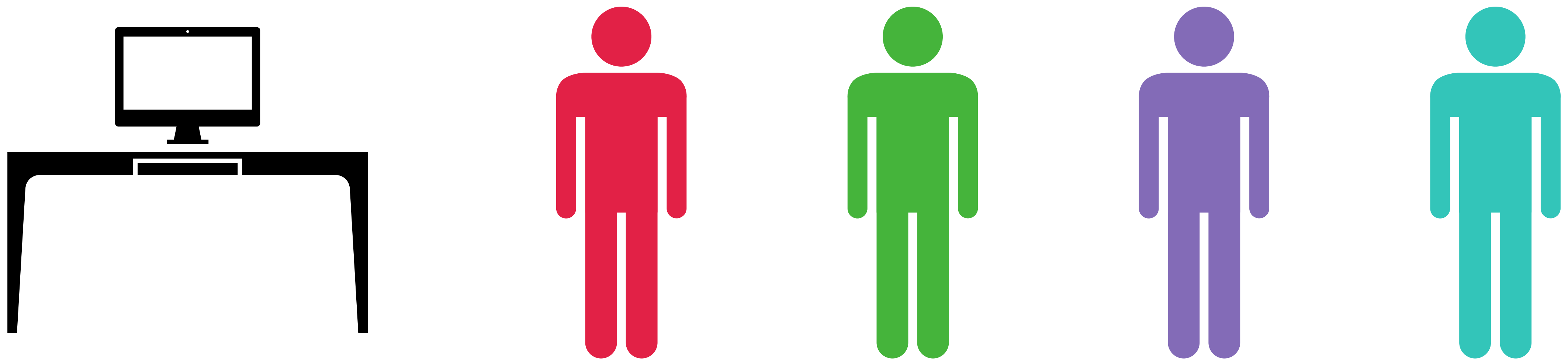
- Choose **no less than** k candidates

"RAND" (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

Public University Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

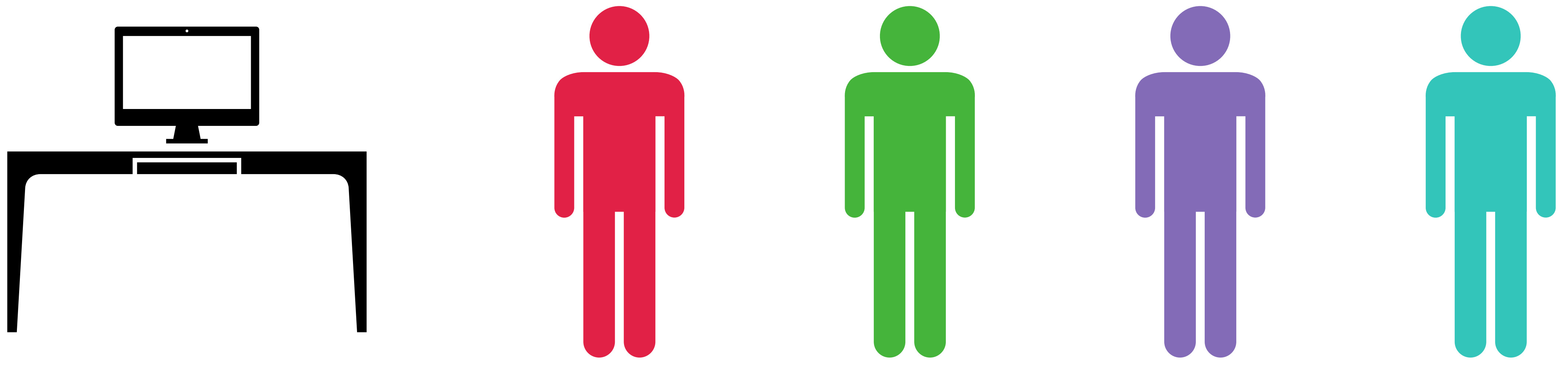
Candidates arrive **online**

Irrevocably accept or reject upon arrival

- Choose **at most** k candidates
- x_i = **quality** of candidate i
- Objective: maximize aggregate quality

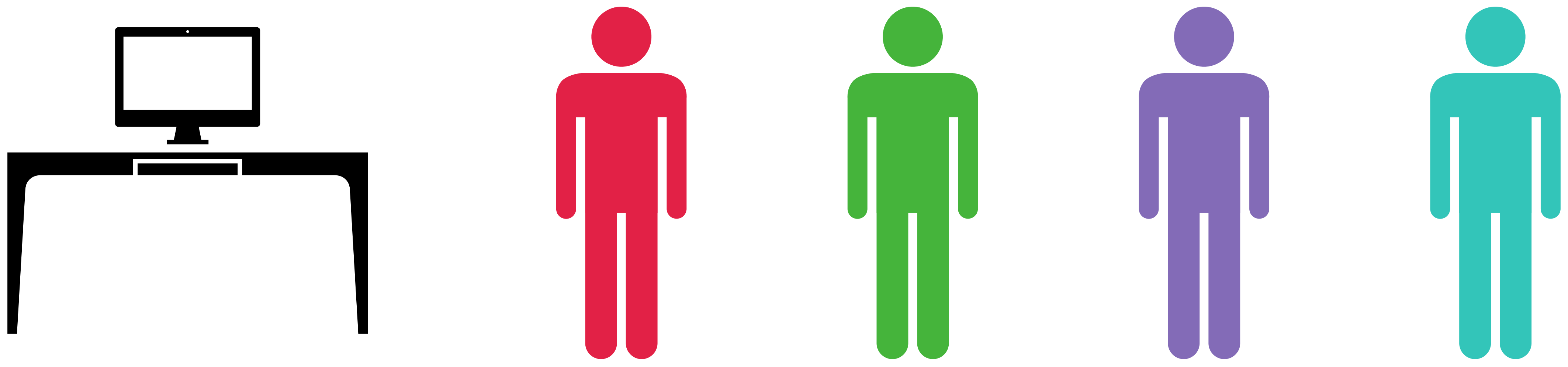
- Choose **no less than** k candidates
- x_i = **cost** of candidate i

“RAND” (Classic) Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

Public University Secretary Problem



$x_1 = 4$ $x_2 = 13$ $x_3 = 2$ $x_4 = 5$

Candidates arrive **online**

Irrevocably accept or reject upon arrival

- Choose **at most** k candidates
- x_i = **quality** of candidate i
- Objective: maximize aggregate quality

- Choose **no less than** k candidates
- x_i = **cost** of candidate i
- Objective: minimize aggregate cost

RAND (Maximization) Secretary Problems

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems
- Prophet inequality problems

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems
- Prophet inequality problems
- Secretary with advice

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems
- Prophet inequality problems
- Secretary with advice
  - Prediction on secretary quality

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems
- Prophet inequality problems
- Secretary with advice
  - Prediction on secretary quality
  - Alternatives to random order: e.g., sample of secretaries

RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems
- Prophet inequality problems
- Secretary with advice
  - Prediction on secretary quality
  - Alternatives to random order: e.g., sample of secretaries

All based on maximization objectives

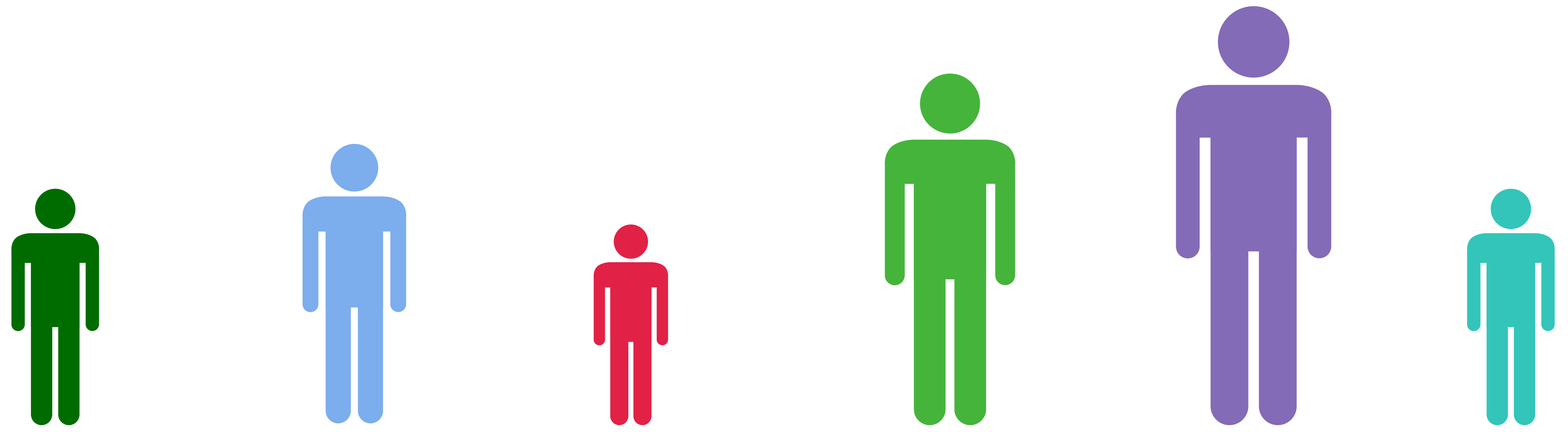
RAND (Maximization) Secretary Problems

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems
- Prophet inequality problems
- Secretary with advice
  - Prediction on secretary quality
  - Alternatives to random order: e.g., sample of secretaries

All based on maximization objectives

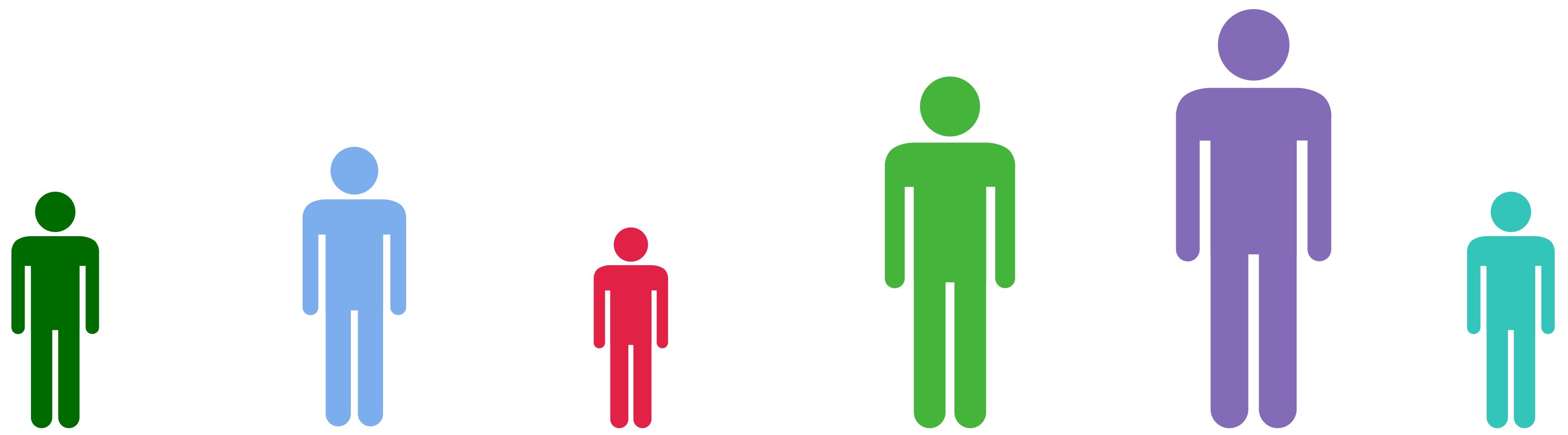
All require beyond-worst-case approach to learn scale of secretary quality

Sample-and-price for RAND



- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **at most k** candidates
- x_i = **quality** of candidate i
- Objective: maximize expected aggregate quality

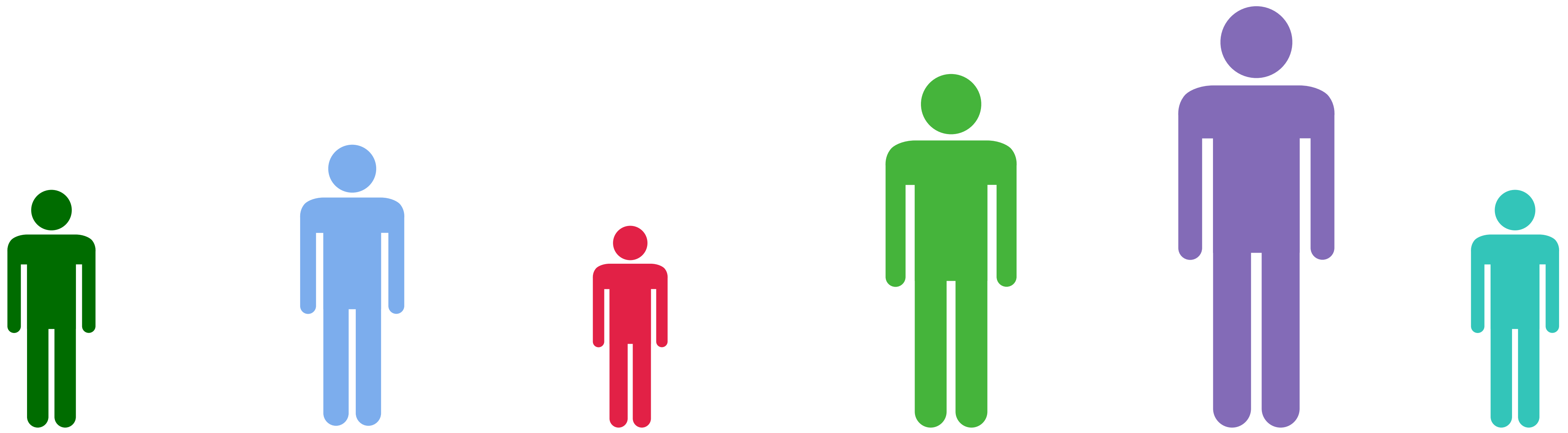
Sample-and-price for RAND



Random Order Arrivals

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **at most k** candidates
- x_i = **quality** of candidate i
- Objective: maximize expected aggregate quality

Sample-and-price for RAND

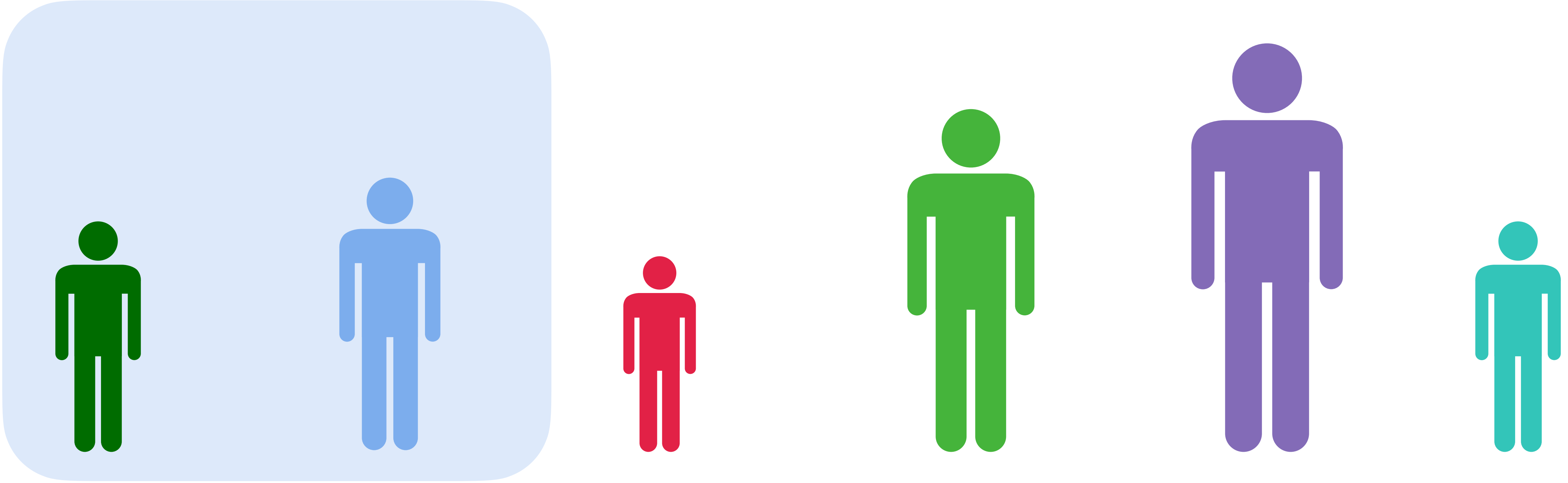


Random Order Arrivals

Know n

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **at most k** candidates
- x_i = **quality** of candidate i
- Objective: maximize expected aggregate quality

Sample-and-price for RAND

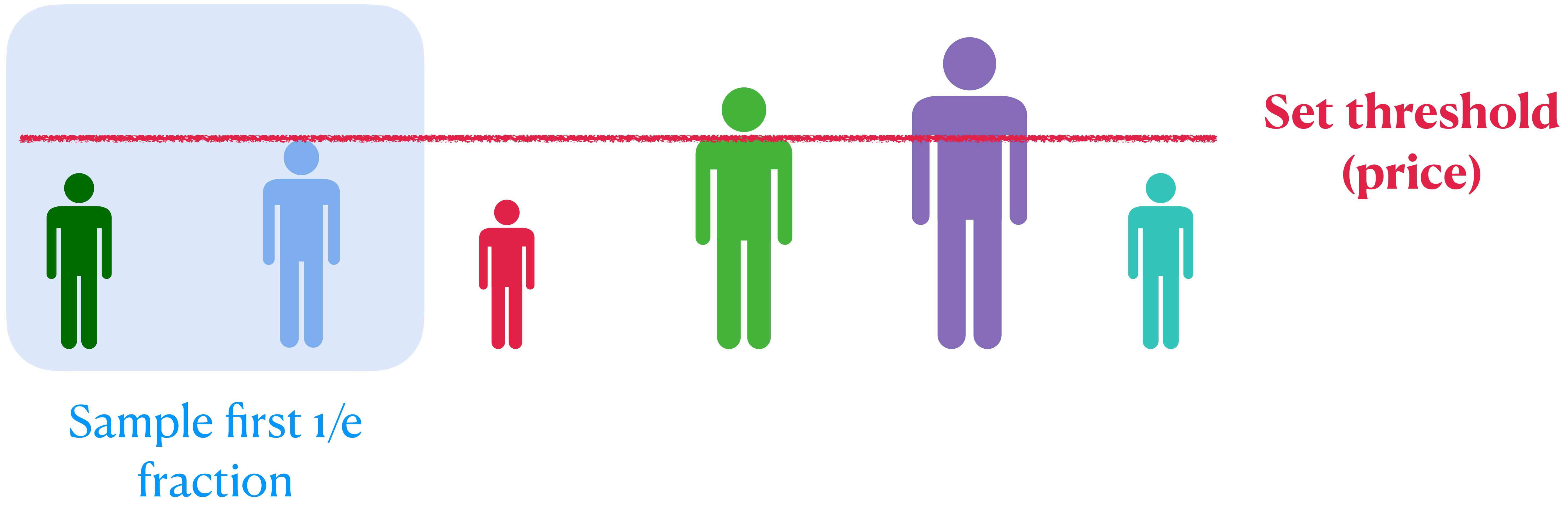


Random Order Arrivals

Know n

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **at most k** candidates
- x_i = **quality** of candidate i
- Objective: maximize expected aggregate quality

Sample-and-price for RAND

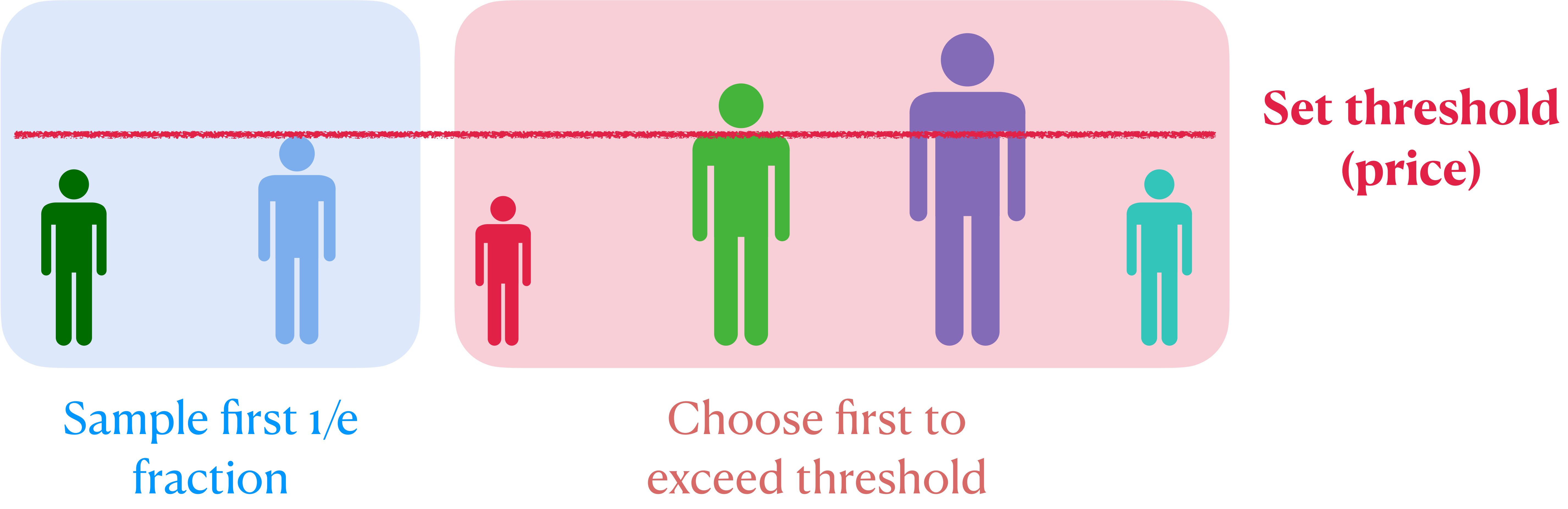


Random Order Arrivals

Know n

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **at most** k candidates
- x_i = **quality** of candidate i
- Objective: maximize expected aggregate quality

Sample-and-price for RAND

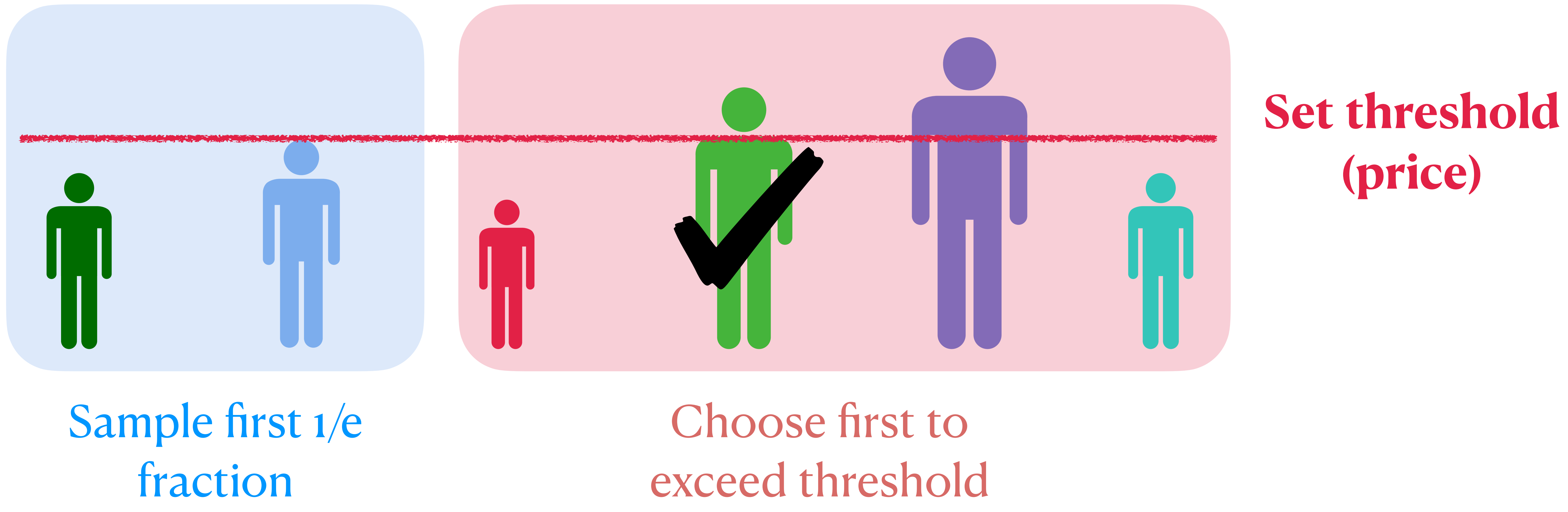


Random Order Arrivals

Know n

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **at most** k candidates
- x_i = **quality** of candidate i
- Objective: maximize expected aggregate quality

Sample-and-price for RAND

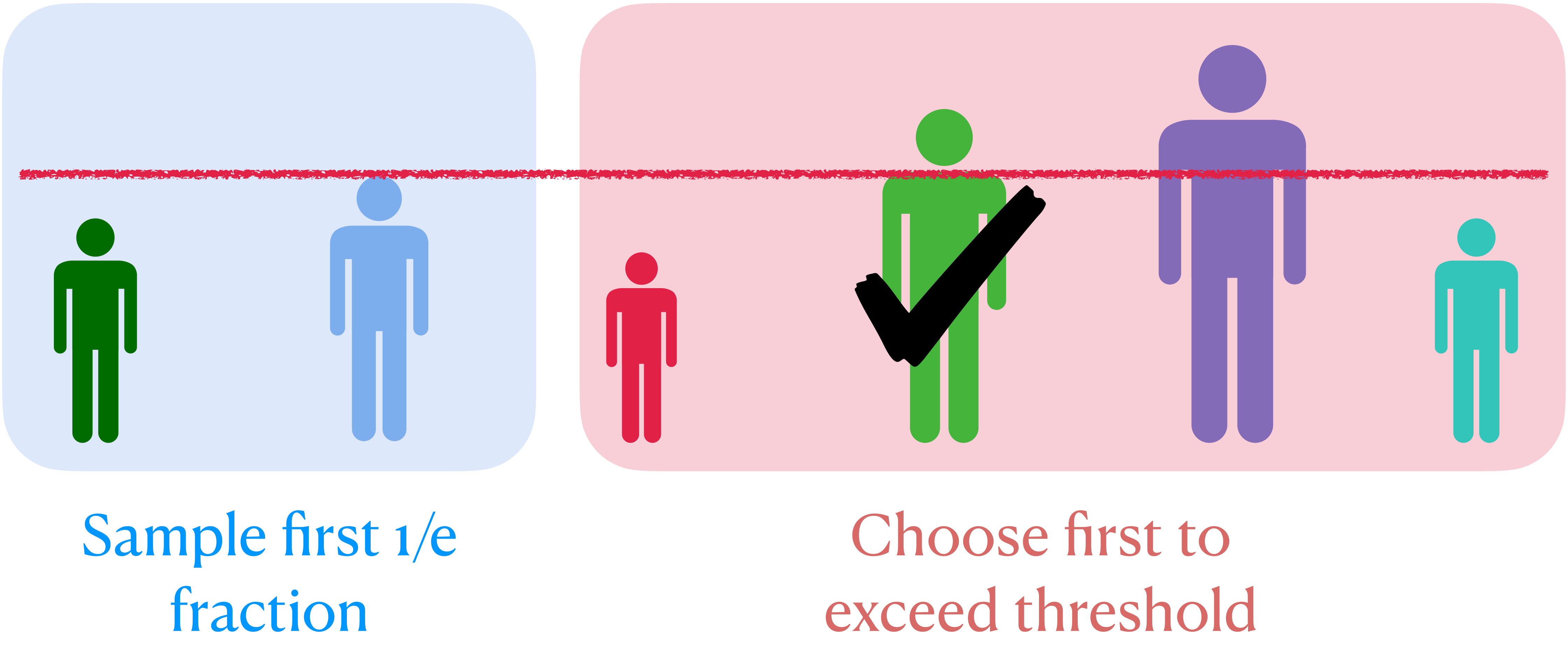


Random Order Arrivals

Know n

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **at most** k candidates
- x_i = **quality** of candidate i
- Objective: maximize expected aggregate quality

Sample-and-price for RAND



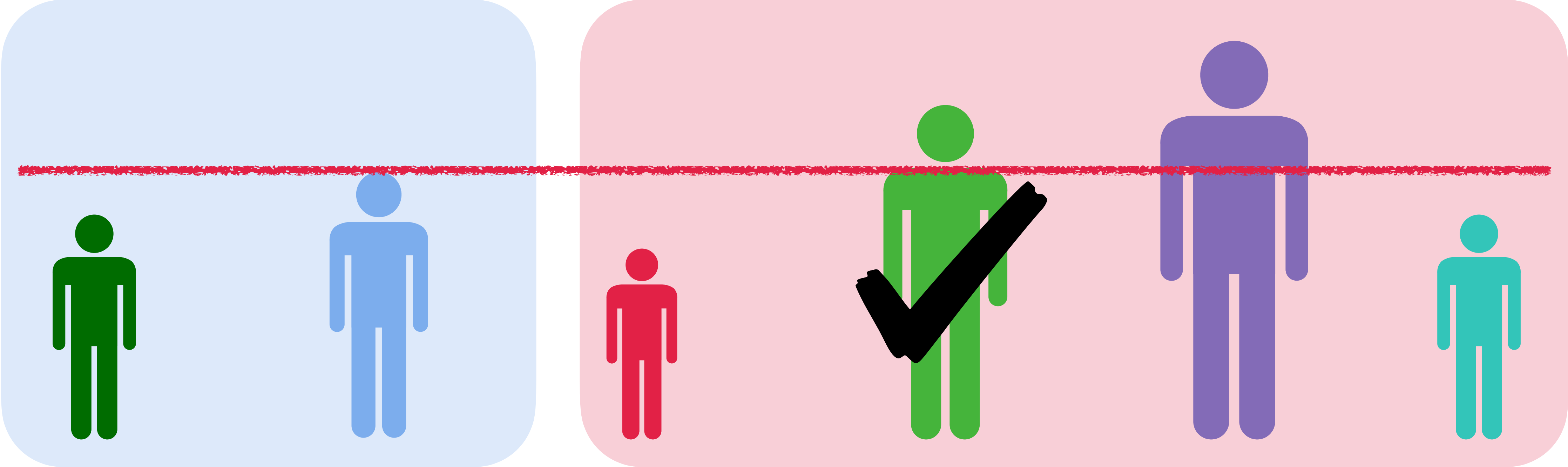
Set threshold (price)

Random Order Arrivals

Know n

Upshot: only need to select most valuable candidate **with *some* (constant) probability**

Sample-and-price for RAND



Sample first 1/e fraction

Choose first to exceed threshold

Set threshold (price)

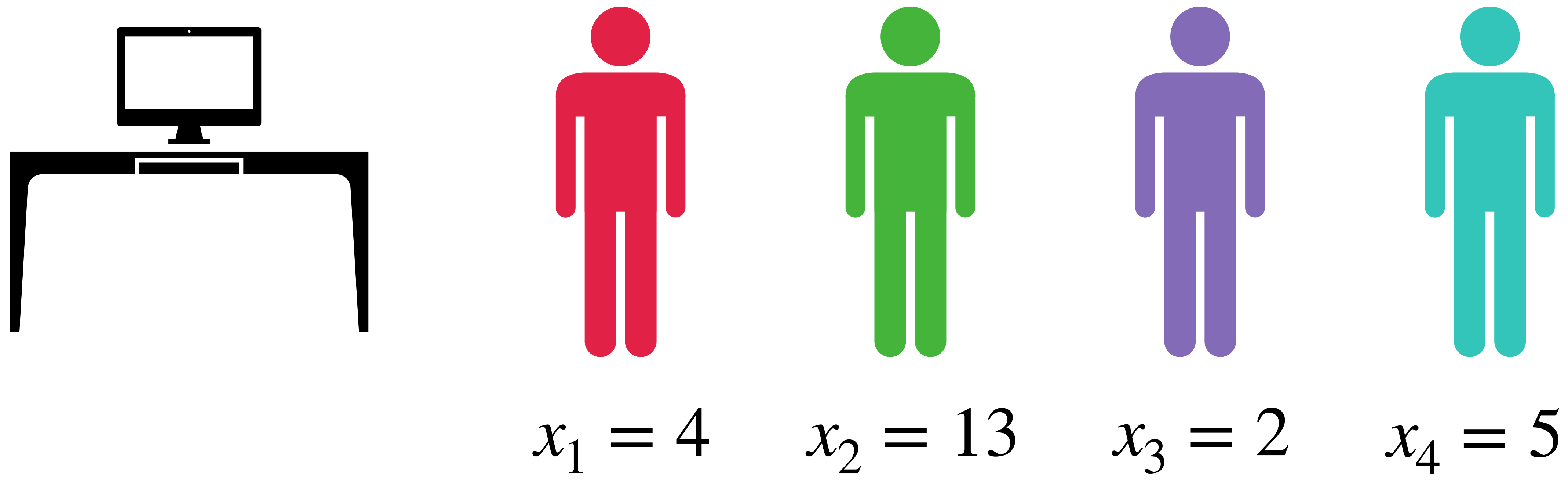
Random Order Arrivals

Know n

Upshot: only need to select most valuable candidate **with *some* (constant) probability**

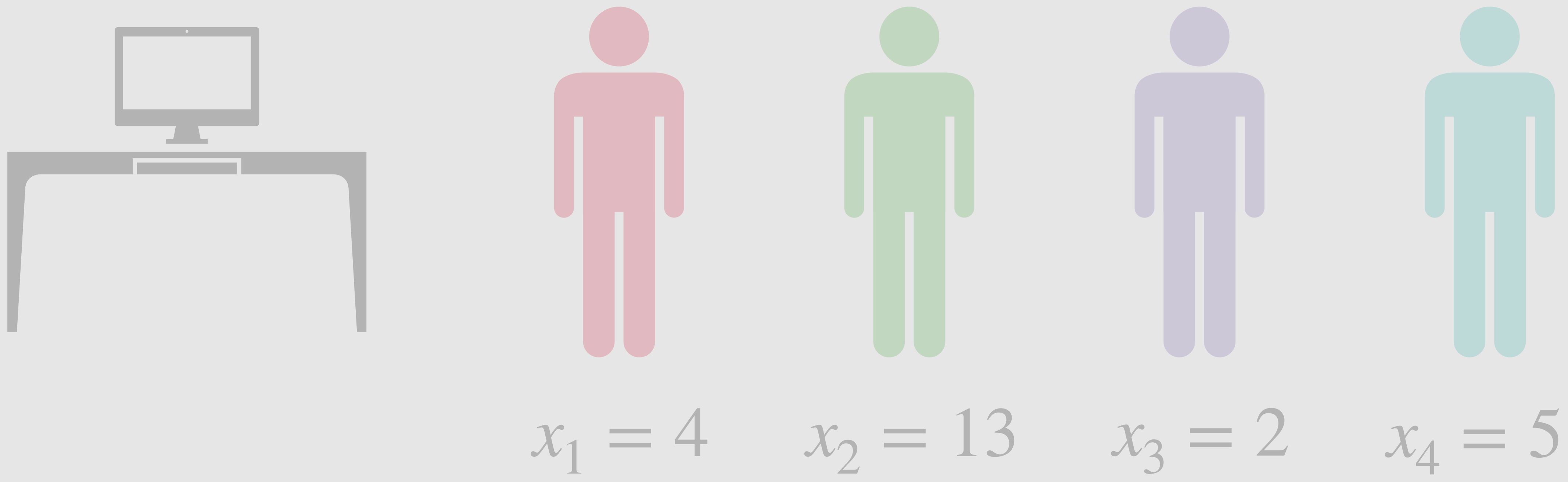
Random Order + Sample → Learn **scale** of quality in applicant pool

Public University Secretary Problem



- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **no less than** k candidates
- x_i = **cost** of candidate i
- Objective: minimize aggregate cost

Public University Secretary Problem



- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **no less than k** candidates
- x_i = **cost** of candidate i
- Objective: minimize aggregate cost

Public University Secretary Problem

Random Order

+

Knowing n

INSUFFICIENT

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **no less than** k candidates
- x_i = **cost** of candidate i
- Objective: minimize aggregate cost

Public University **Lower Bound**

Public University Secretary Problem

Random Order

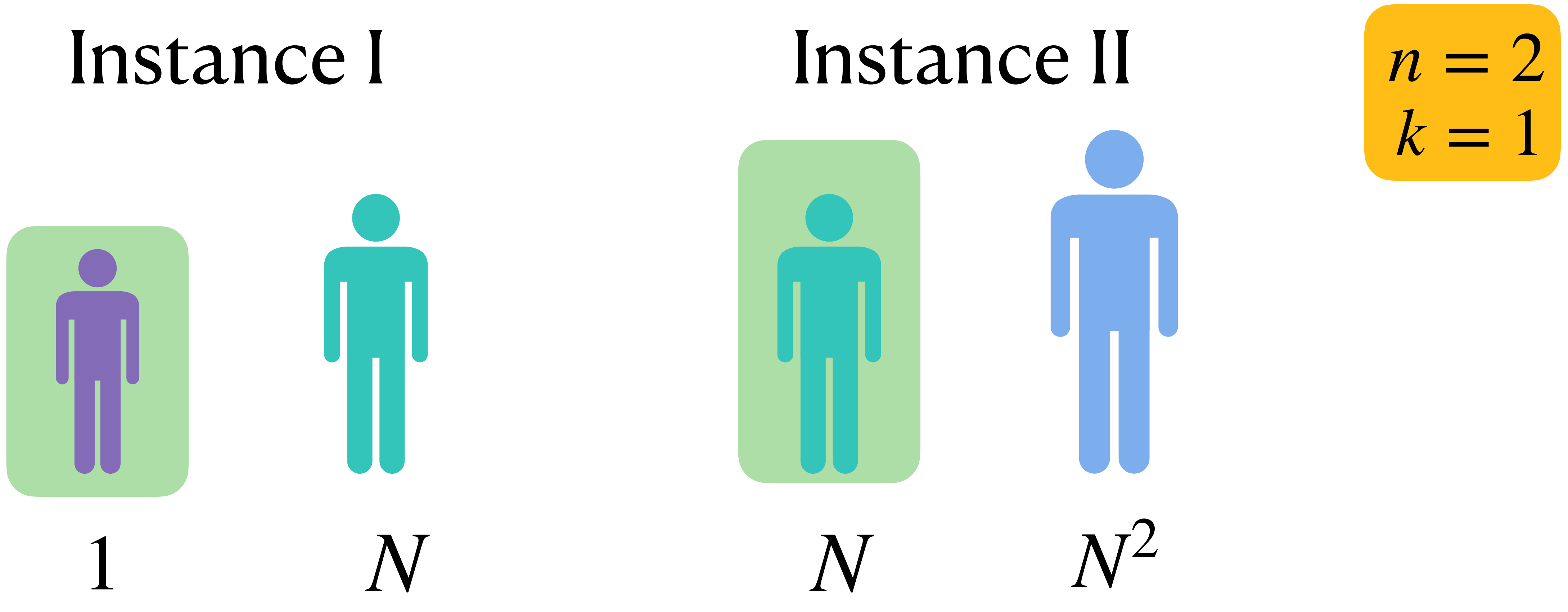
+

Knowing n

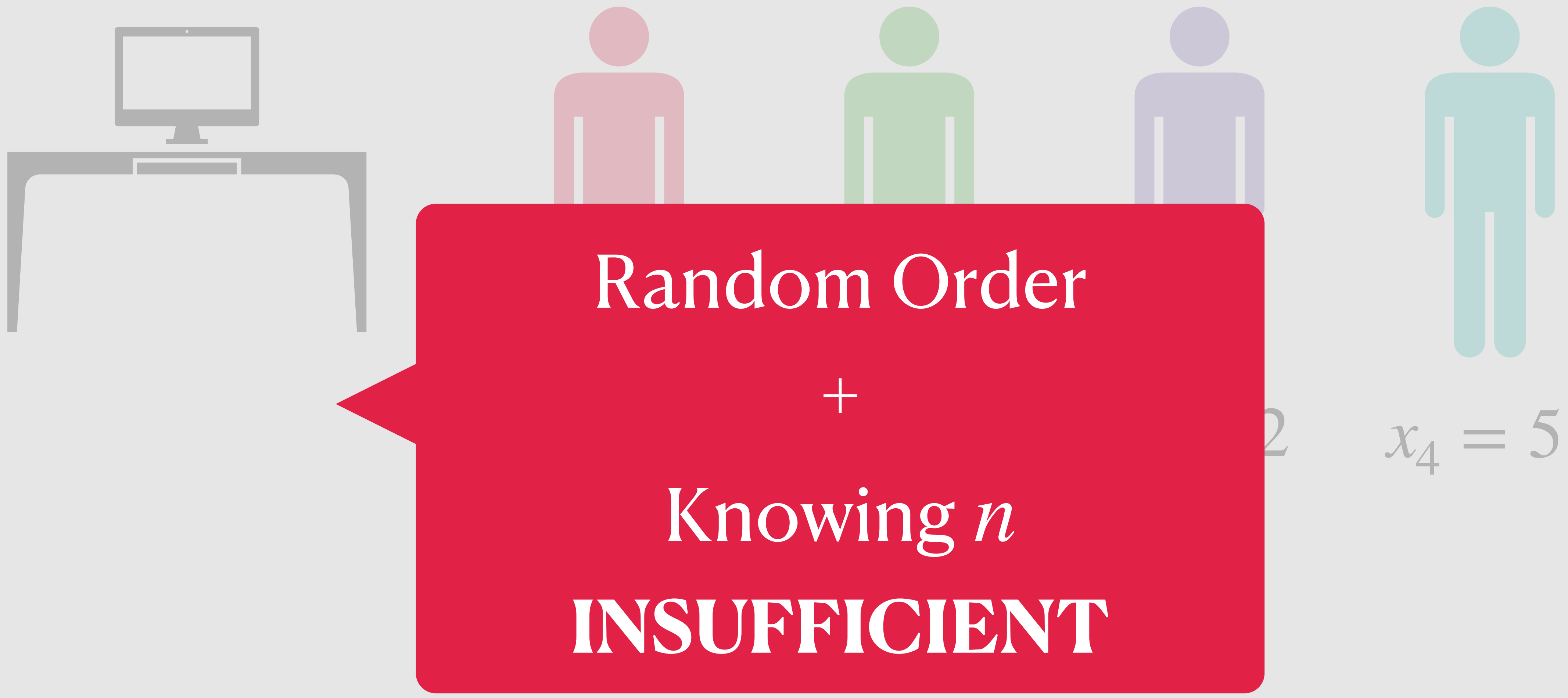
INSUFFICIENT

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **no less than** k candidates
- x_i = **cost** of candidate i
- Objective: minimize aggregate cost

Public University **Lower Bound**



Public University Secretary Problem



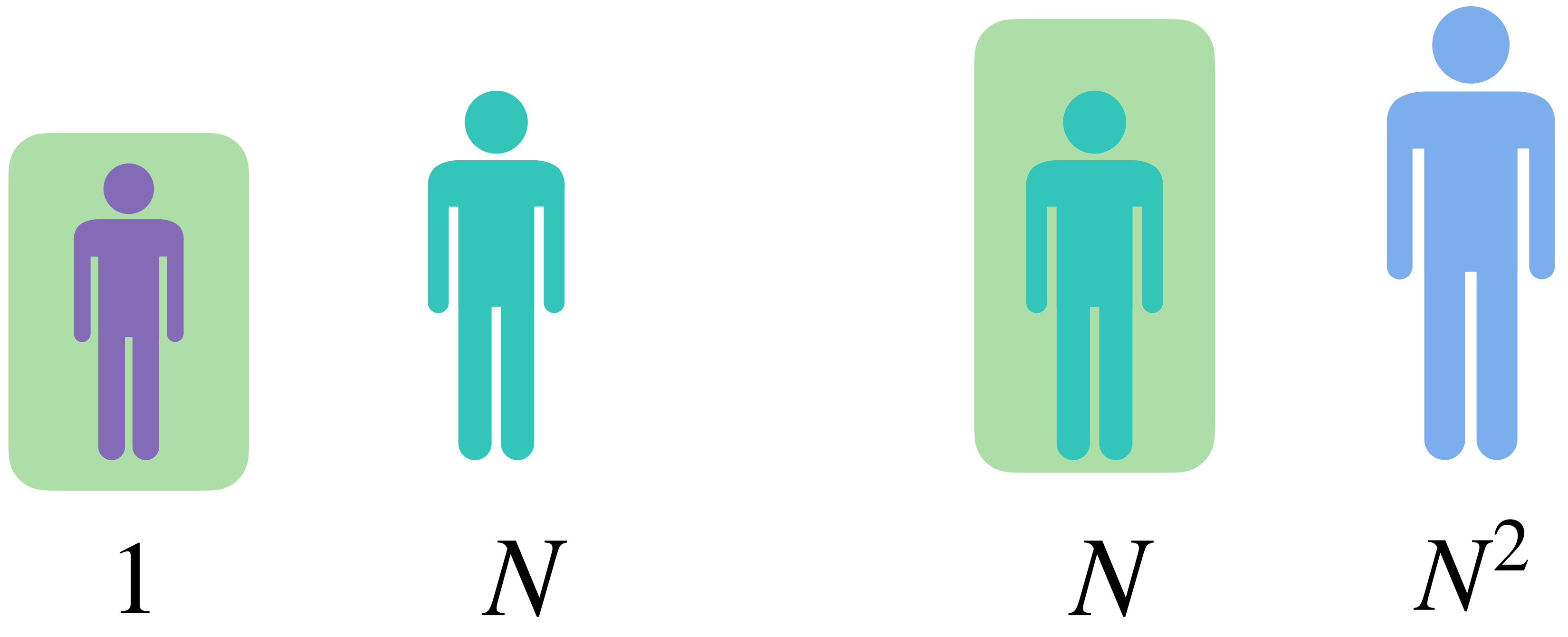
- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **no less than** k candidates
- x_i = **cost** of candidate i
- Objective: minimize aggregate cost

Public University **Lower Bound**

Instance I

Instance II

$n = 2$
$k = 1$



$N \gg 1$

unbounded competitiveness!

Public University Secretary Problem

Random Order

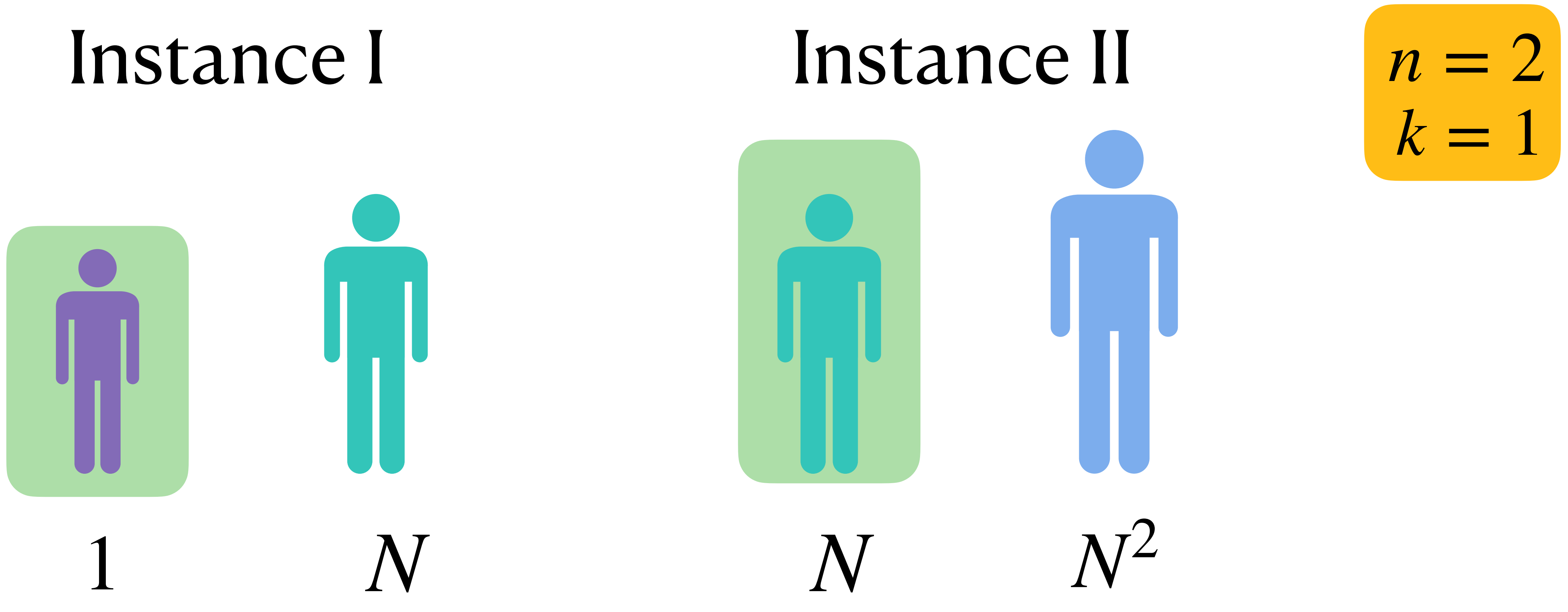
+

Knowing n

INSUFFICIENT

- Candidates arrive **online**
- **Irrevocably** accept or reject upon arrival
- Choose **no less than** k candidates
- x_i = **cost** of candidate i
- Objective: minimize aggregate cost

Public University **Lower Bound**



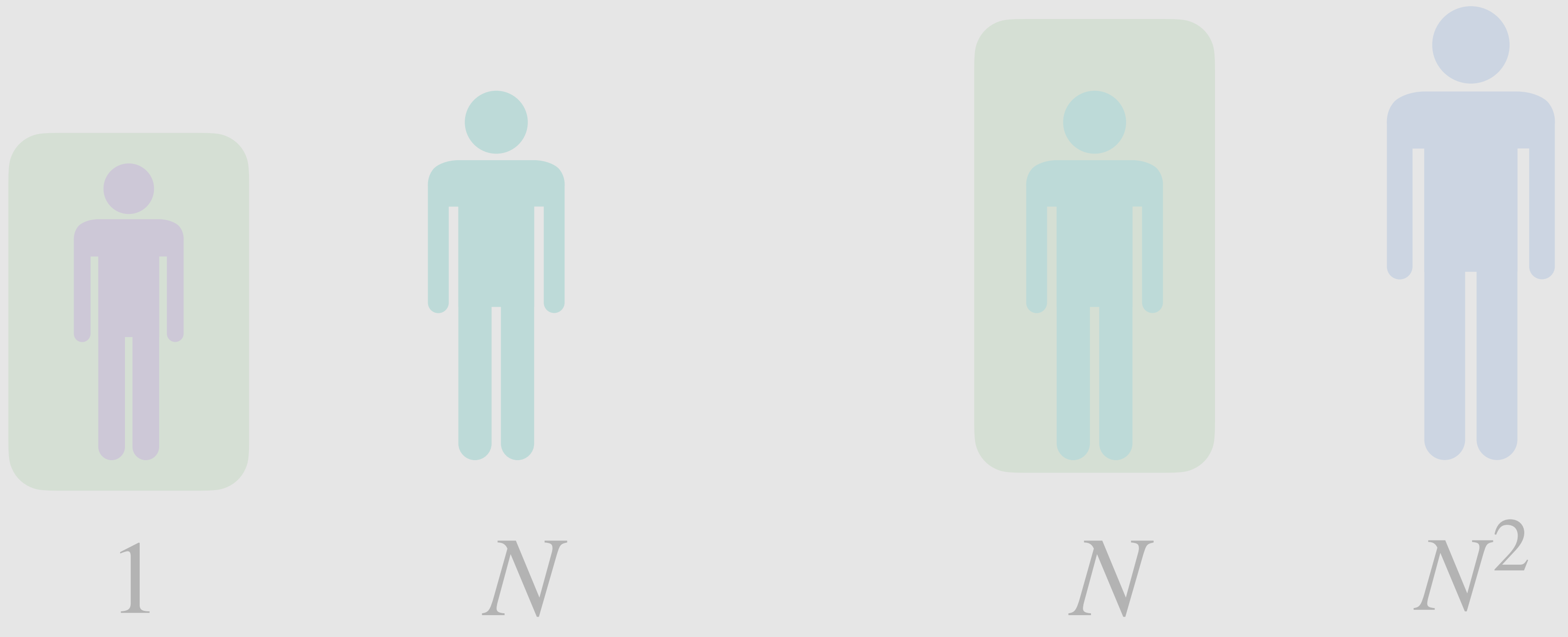
$N \gg 1$

unbounded competitiveness!

Public University **Lower Bound**

Instance I | Instance II

$n = 2$
$k = 1$



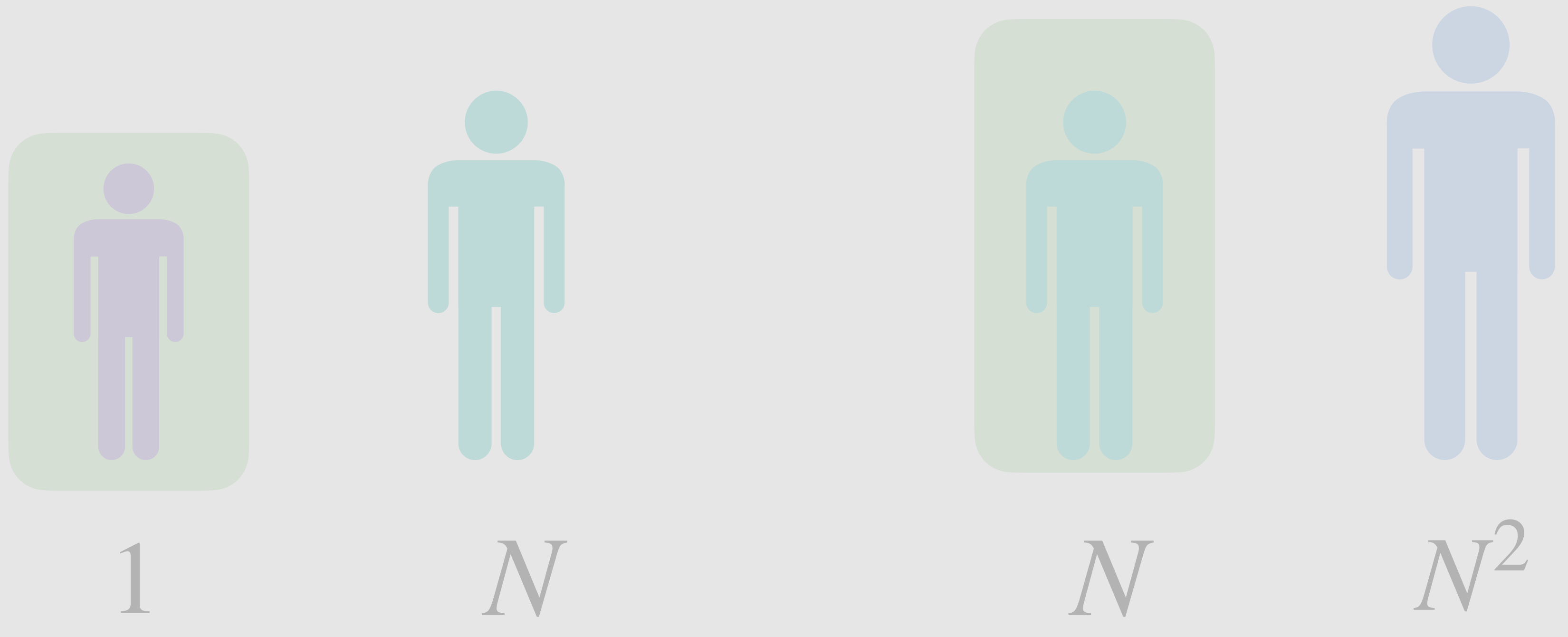
$N \gg 1$

unbounded competitiveness!

Public University **Lower Bound**

Instance I　　Instance II

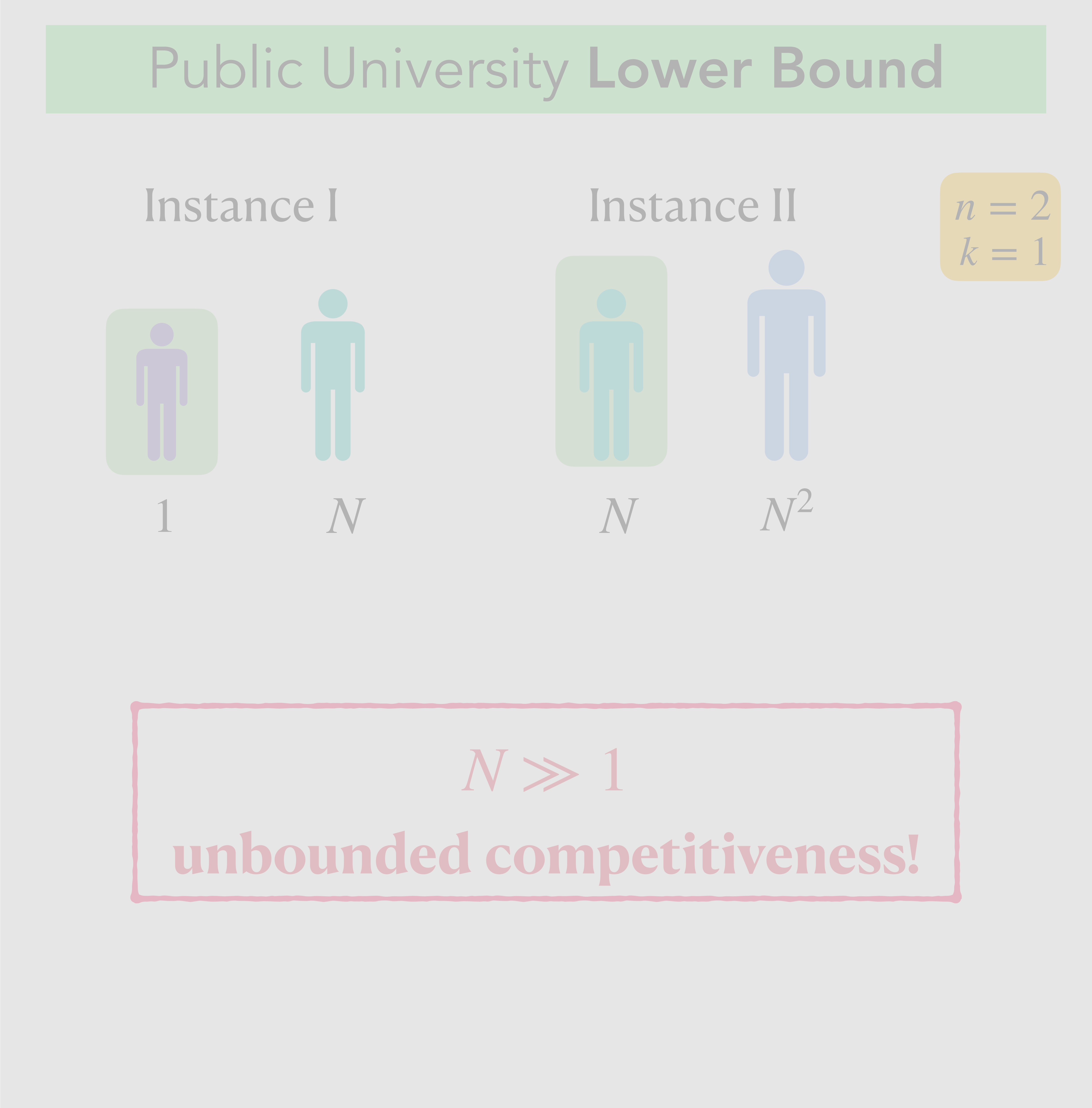
$n = 2$
$k = 1$



$N \gg 1$

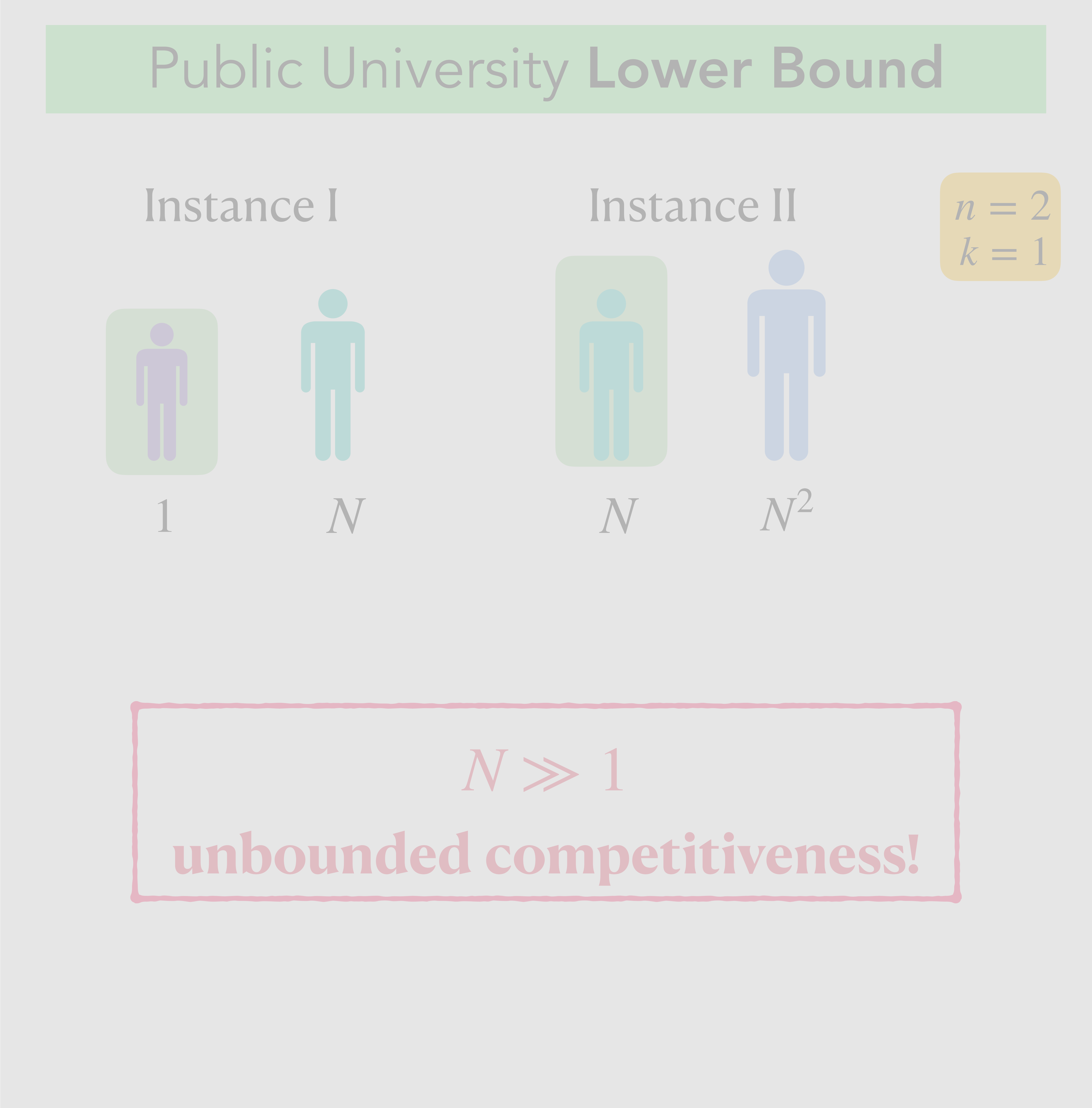
unbounded competitiveness!

Why is random order not sufficient?



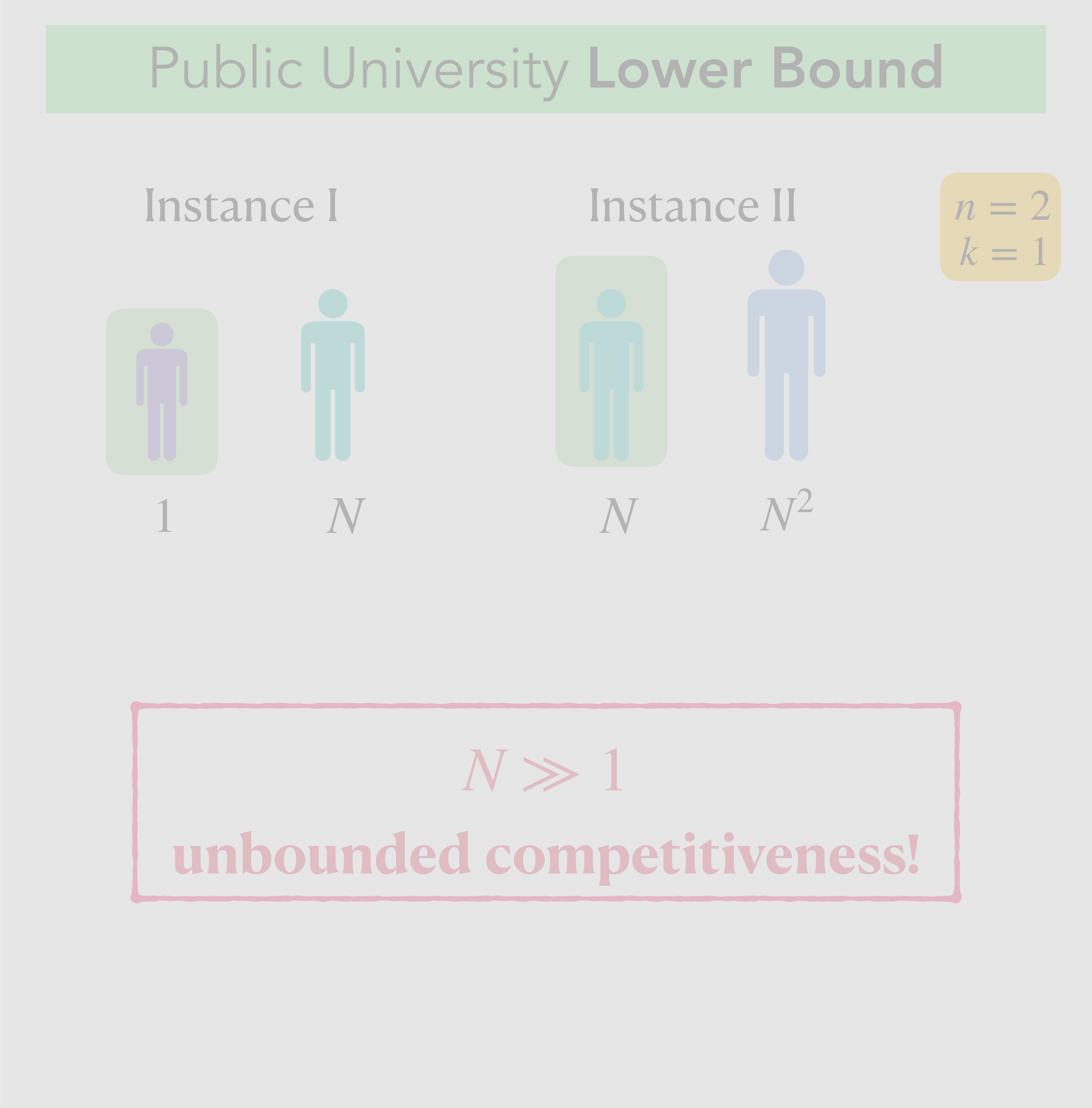
Why is random order not sufficient?

- Public University is a **minimization problem**



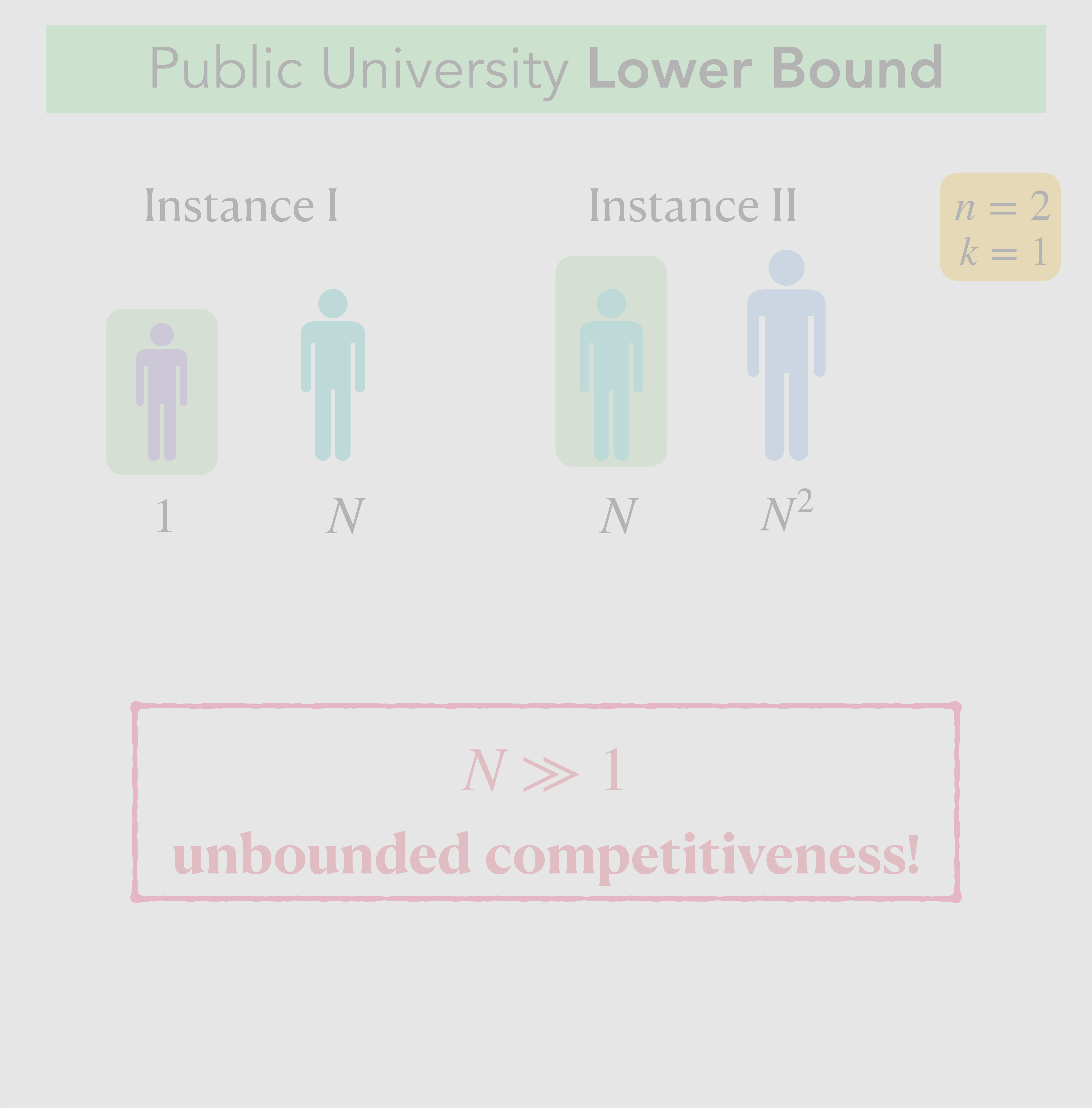
Why is random order not sufficient?

- Public University is a **minimization problem**
- Cannot simply reject secretaries



Why is random order not sufficient?

- Public University is a **minimization problem**
- Cannot simply reject secretaries
  - **Must** hire **at least** k secretaries



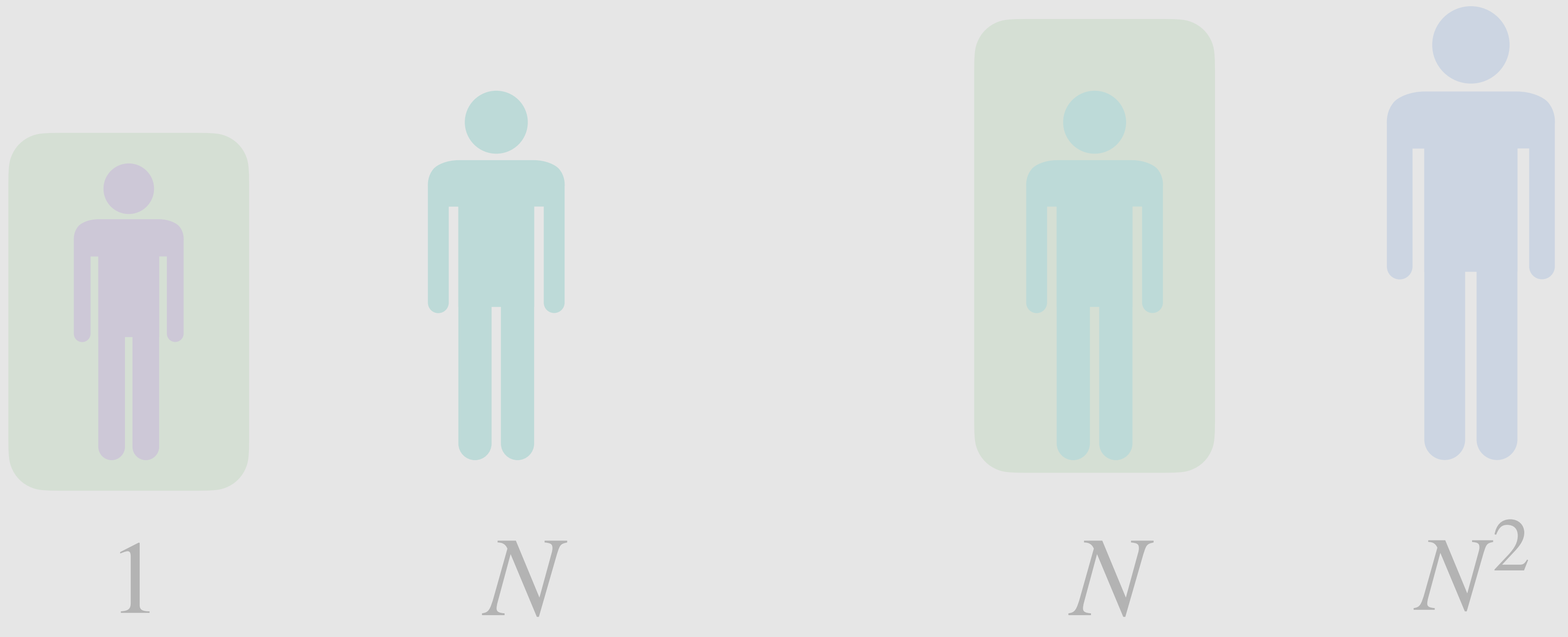
Why is random order not sufficient?

- Public University is a **minimization problem**
- Cannot simply reject secretaries
  - **Must** hire **at least** k secretaries
  - Could be **forced to incur enormous cost**

Public University **Lower Bound**

Instance I | Instance II

$n = 2$
$k = 1$



$N \gg 1$

unbounded competitiveness!

Why is random order not sufficient?

- Public University is a **minimization problem**
- Cannot simply reject secretaries
  - **Must** hire **at least** k secretaries
  - Could be **forced to incur enormous cost**
- RAND / maximization: ignore cases where low-value secretaries hired

How do we break through the strong lower bound?

How do we break through the strong lower bound?

Learning-augmented approach:

How do we break through the strong lower bound?

Learning-augmented approach:

Online algorithm given "budget" B **a priori**

B **upper bound on OPT** (cost of k cheapest secretaries)

Results

Results

Best possible online algorithm for Public University Secretary

$\Theta(\log k)$-competitive against B

in both **adversarial and random** arrival orders

Results

Best possible online algorithm for Public University Secretary

$\Theta(\log k)$-competitive against B

in both **adversarial <u>and</u> random** arrival orders

Upper Bound of $O(B \cdot \log k)$:

Results

Best possible online algorithm for Public University Secretary

$\Theta(\log k)$-competitive against B

in both **adversarial <u>and</u> random** arrival orders

Upper Bound of $O(B \cdot \log k)$:

✓ Even with **adversarial** ordering

Results

Best possible online algorithm for Public University Secretary

$\Theta(\log k)$-competitive against B

in both **adversarial <u>and</u> random** arrival orders

Upper Bound of $O(B \cdot \log k)$:

✓ Even with **adversarial** ordering

✓ Simple **deterministic** algorithm

Results

Best possible online algorithm for Public University Secretary

$\Theta(\log k)$-competitive against B

in both **adversarial and random** arrival orders

Upper Bound of $O(B \cdot \log k)$:

✓ Even with **adversarial** ordering

✓ Simple **deterministic** algorithm

Lower Bound of $\Omega(B \cdot \log k)$:

Results

Best possible online algorithm for Public University Secretary

$\Theta(\log k)$-competitive against B

in both **adversarial and random** arrival orders

Upper Bound of $O(B \cdot \log k)$:

✓ Even with **adversarial** ordering

✓ Simple **deterministic** algorithm

Lower Bound of $\Omega(B \cdot \log k)$:

✓ Even with **random** ordering

Results

Best possible online algorithm for Public University Secretary

$\Theta(\log k)$-competitive against B

in both **adversarial and random** arrival orders

Upper Bound of $O(B \cdot \log k)$:

✓ Even with **adversarial** ordering

✓ Simple **deterministic** algorithm

Lower Bound of $\Omega(B \cdot \log k)$:

✓ Even with **random** ordering

✓ Against **randomized** algorithms

Results

Best possible online algorithm for Public University Secretary

$\Theta(\log k)$-competitive against B

in both **adversarial and random** arrival orders

Upper Bound of $O(B \cdot \log k)$:

✓ Even with **adversarial** ordering

✓ Simple **deterministic** algorithm

Lower Bound of $\Omega(B \cdot \log k)$:

✓ Even with **random** ordering

✓ Against **randomized** algorithms

Key Takeaway: randomization of negligible benefit!

Results

Upper Bound of $O(B \cdot \log k)$:

✓ Even with **adversarial** ordering

✓ Simple **deterministic** algorithm

Lower Bound of $\Omega(B \cdot \log k)$:

✓ Even with **random** ordering

✓ Against **randomized** algorithms

Results

Upper Bound of $O(B \cdot \log k)$:

✓ Even with **adversarial** ordering

✓ Simple **deterministic** algorithm

↓

"Cautious" Algorithm

Lower Bound of $\Omega(B \cdot \log k)$:

✓ Even with **random** ordering

✓ Against **randomized** algorithms

Results

Upper Bound of $O(B \cdot \log k)$:

✓ Even with **adversarial** ordering

✓ Simple **deterministic** algorithm

↓

"Cautious" Algorithm

(Roughly) *hire candidate i iff they are in "best" solution up until now*

Lower Bound of $\Omega(B \cdot \log k)$:

✓ Even with **random** ordering

✓ Against **randomized** algorithms

Results

Upper Bound of $O(B \cdot \log k)$:

✓ Even with **adversarial** ordering

✓ Simple **deterministic** algorithm

↓

"Cautious" Algorithm

(Roughly) *hire candidate i iff they are in "best" solution up until now*

Lower Bound of $\Omega(B \cdot \log k)$:

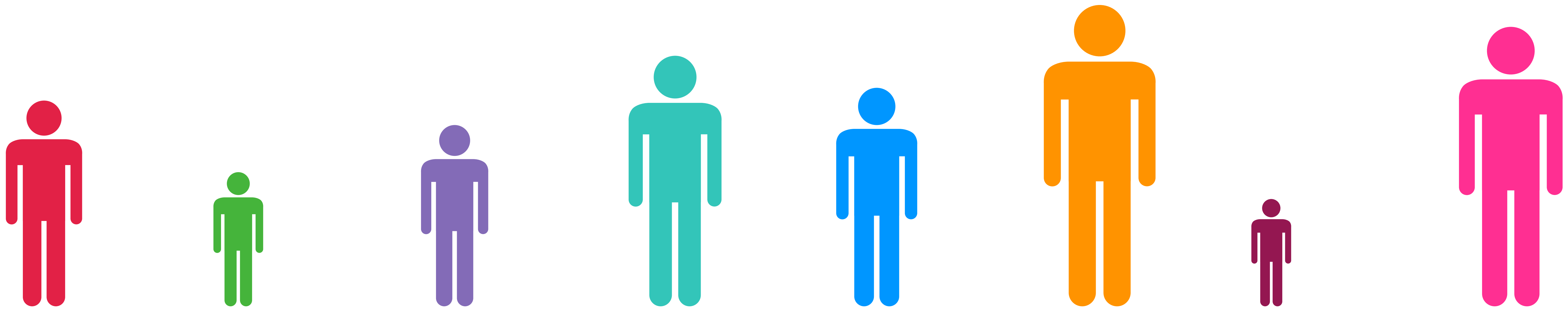
✓ Even with **random** ordering

✓ Against **randomized** algorithms

Key step: (roughly) any competitive algorithm must hire each candidate hired by the cautious algorithm

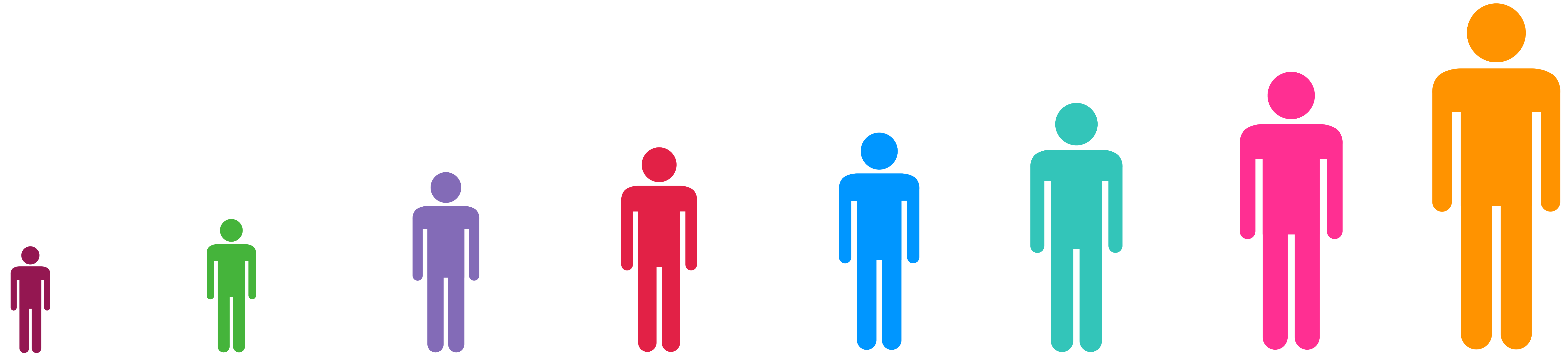
Lower Bound: Adversarial Order

First i candidates: ***(adversarial) arrival order***



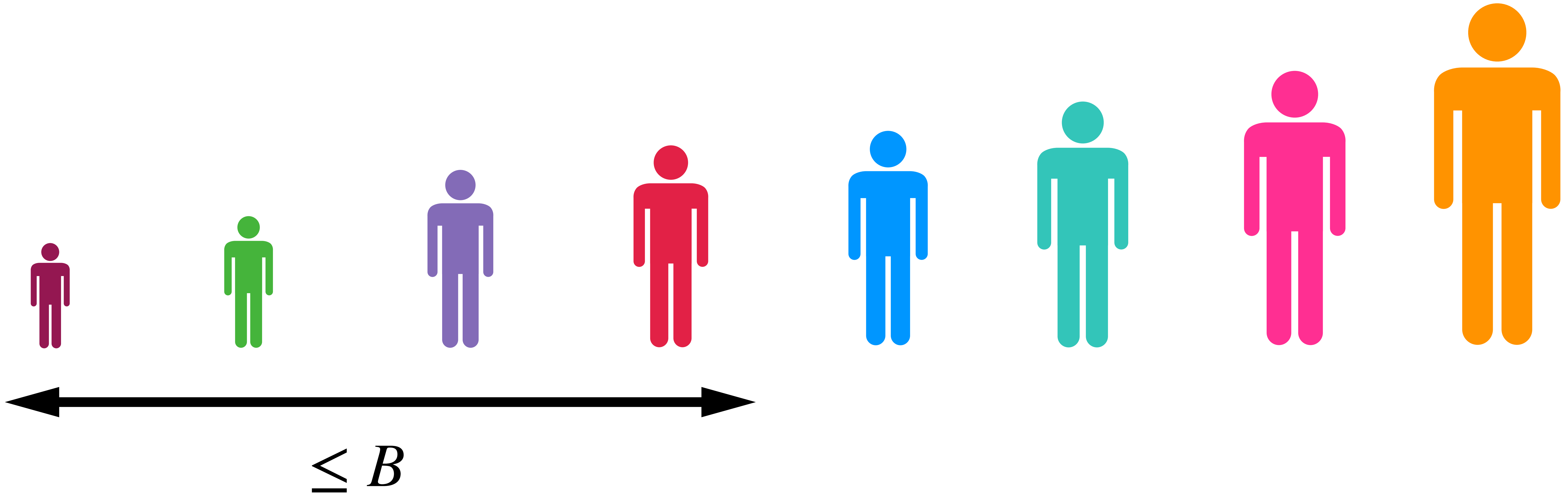
Lower Bound: Adversarial Order

First i candidates: ***sorted***



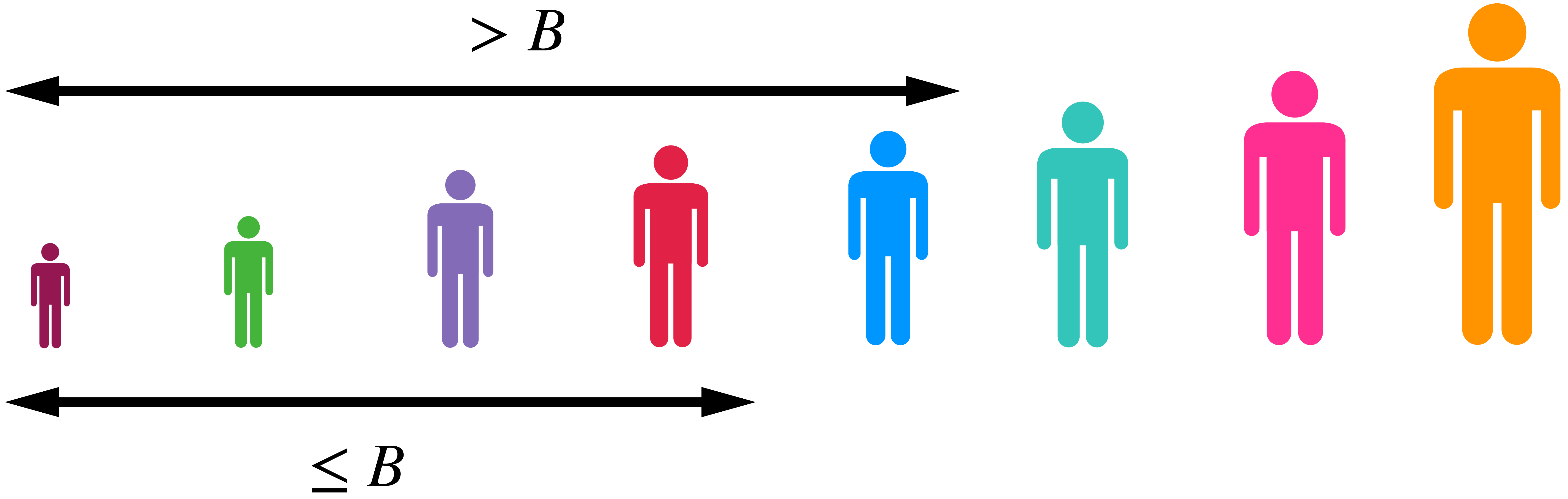
Lower Bound: Adversarial Order

First i candidates: ***sorted***

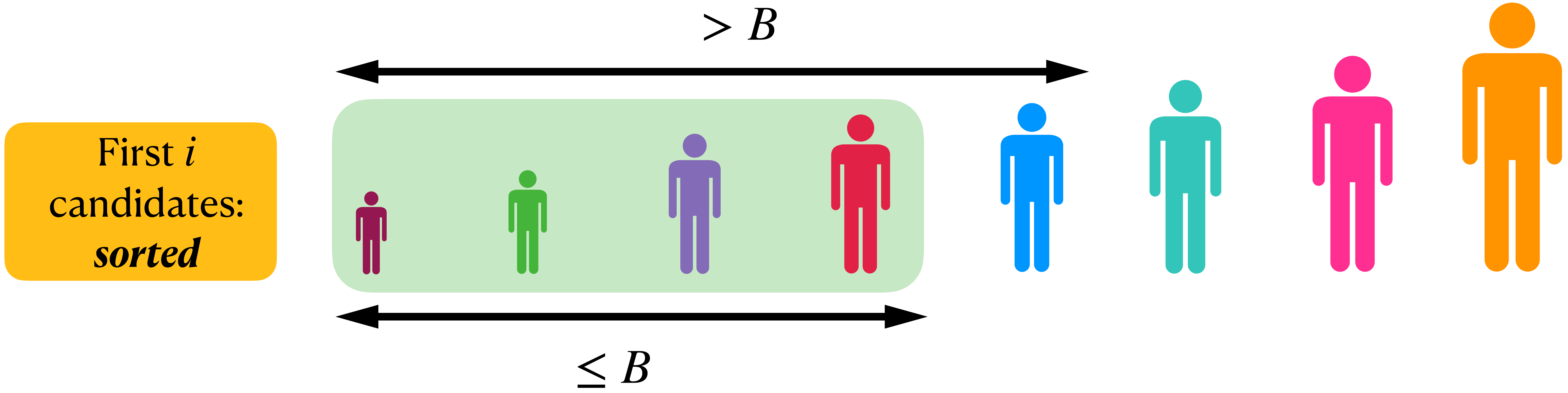


Lower Bound: Adversarial Order

First i candidates: ***sorted***

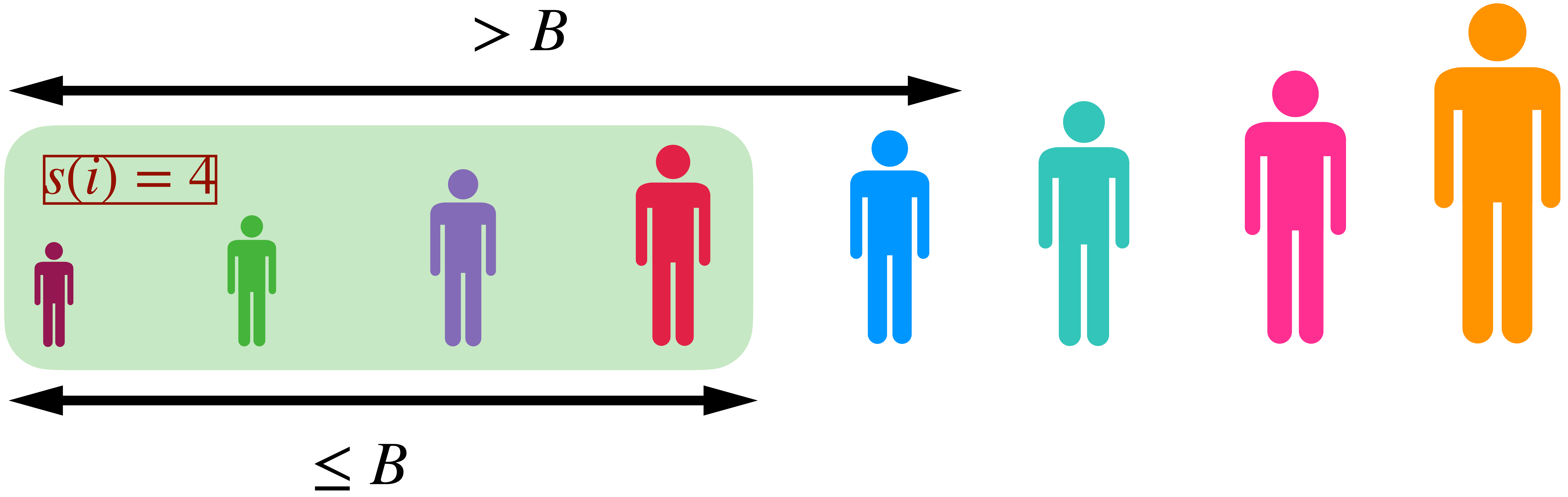


Lower Bound: Adversarial Order



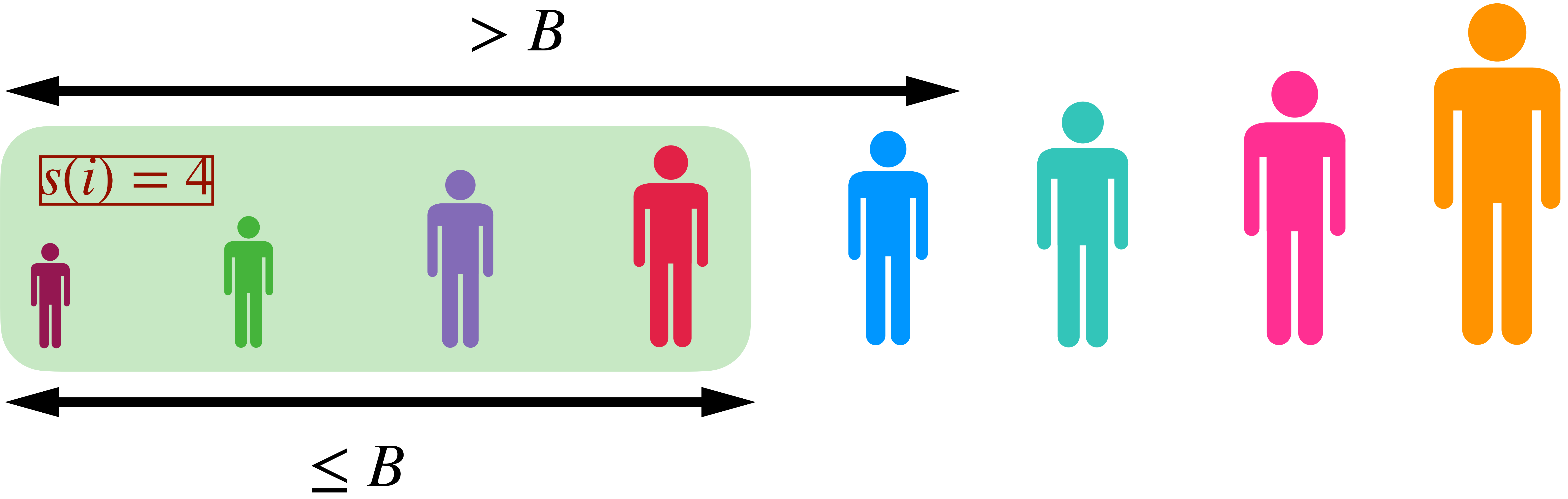
Lower Bound: Adversarial Order

First i candidates: ***sorted***



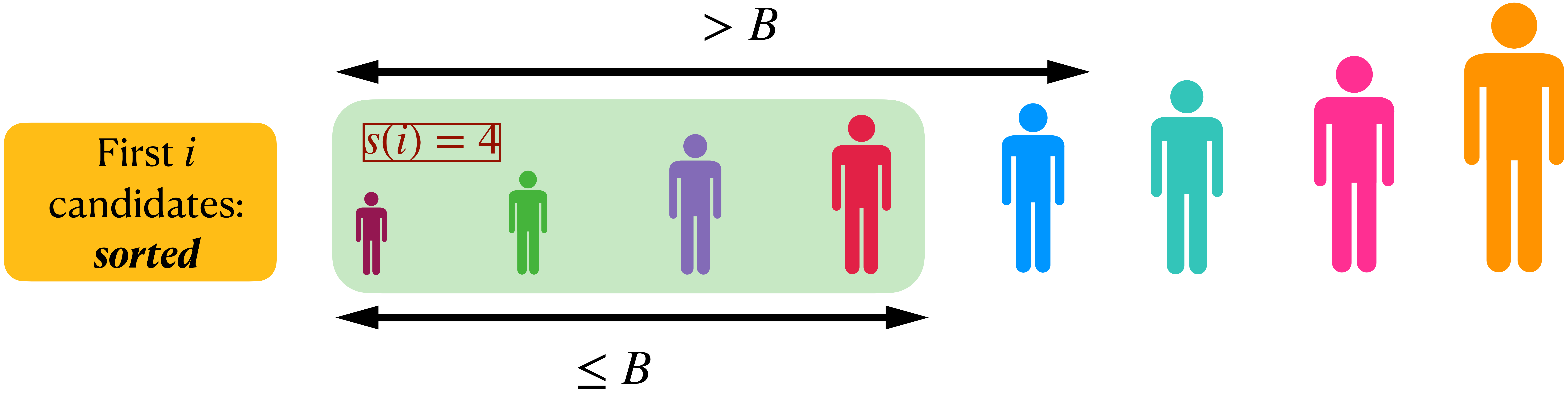
Lower Bound: Adversarial Order

First i candidates: ***sorted***



Acceptance Property: with probability 1, hire $s(i)$ candidates among first i candidates, for all i

Lower Bound: Adversarial Order



Acceptance Property: with probability 1, hire $s(i)$ candidates among first i candidates, for all i

Lemma: Every randomized algorithm that is competitive in adversarial order model has the **acceptance property.**

Lower Bound: Adversarial Order

Lower Bound: Adversarial Order

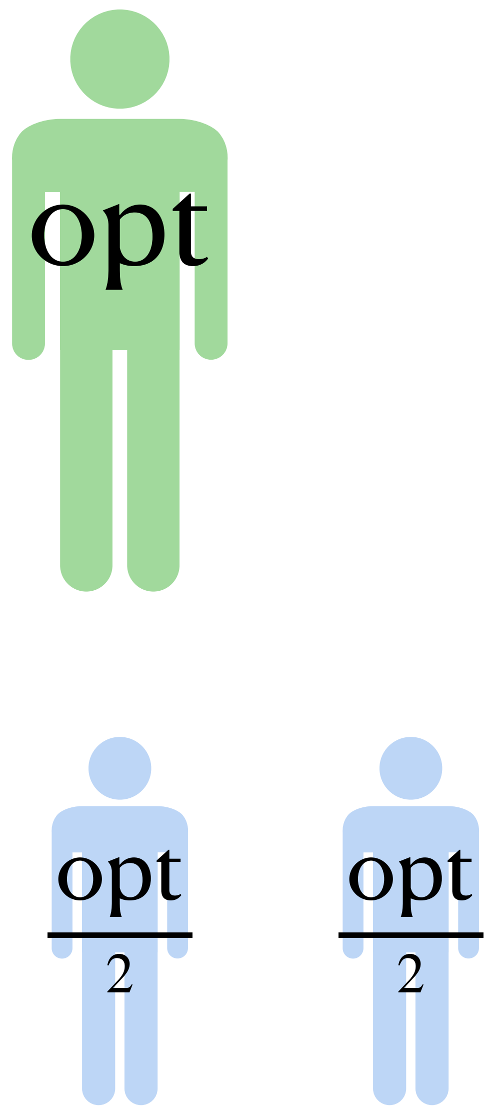
Idea: high cost secretaries arrive first

Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first

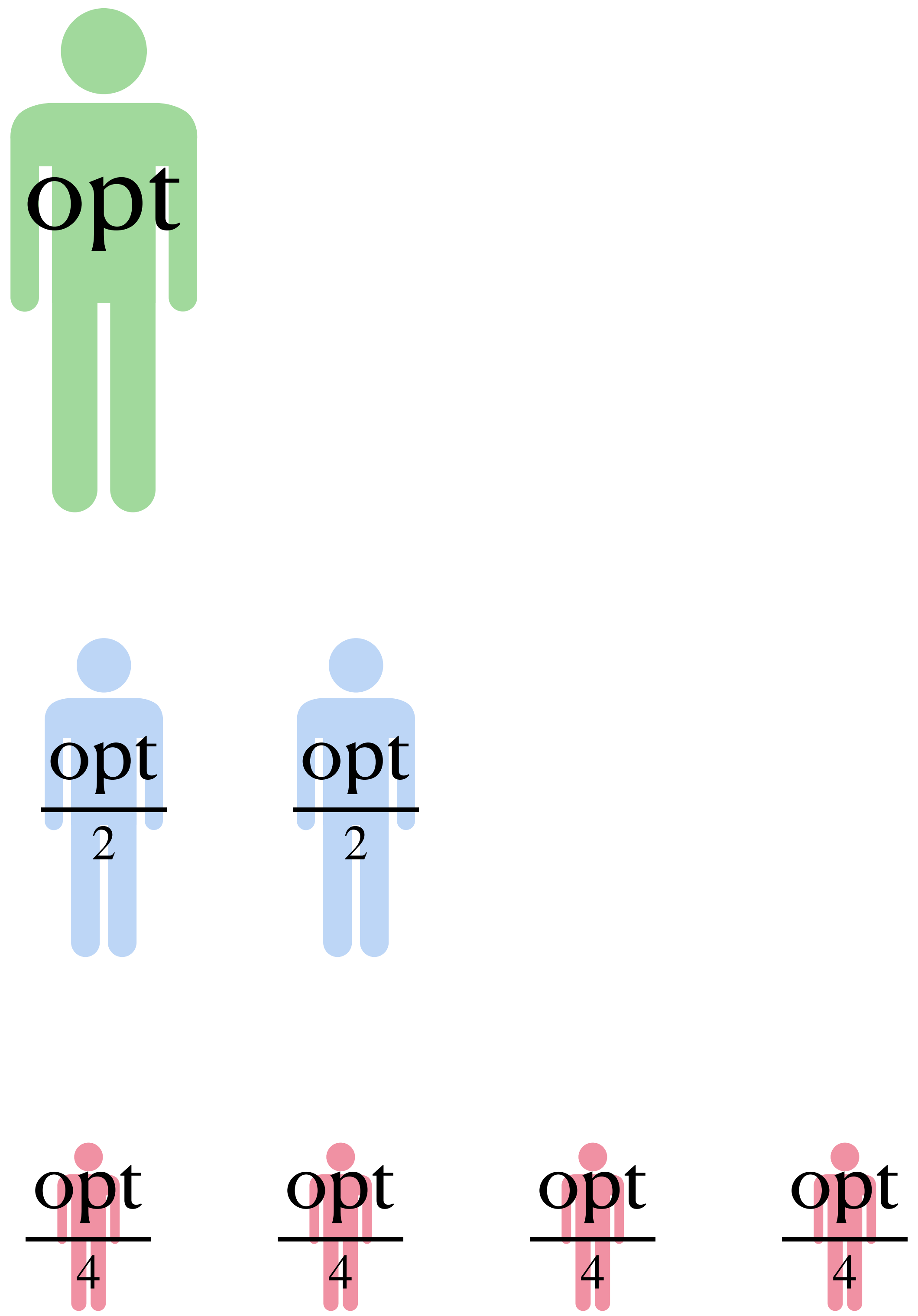
Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first



Lower Bound: Adversarial Order

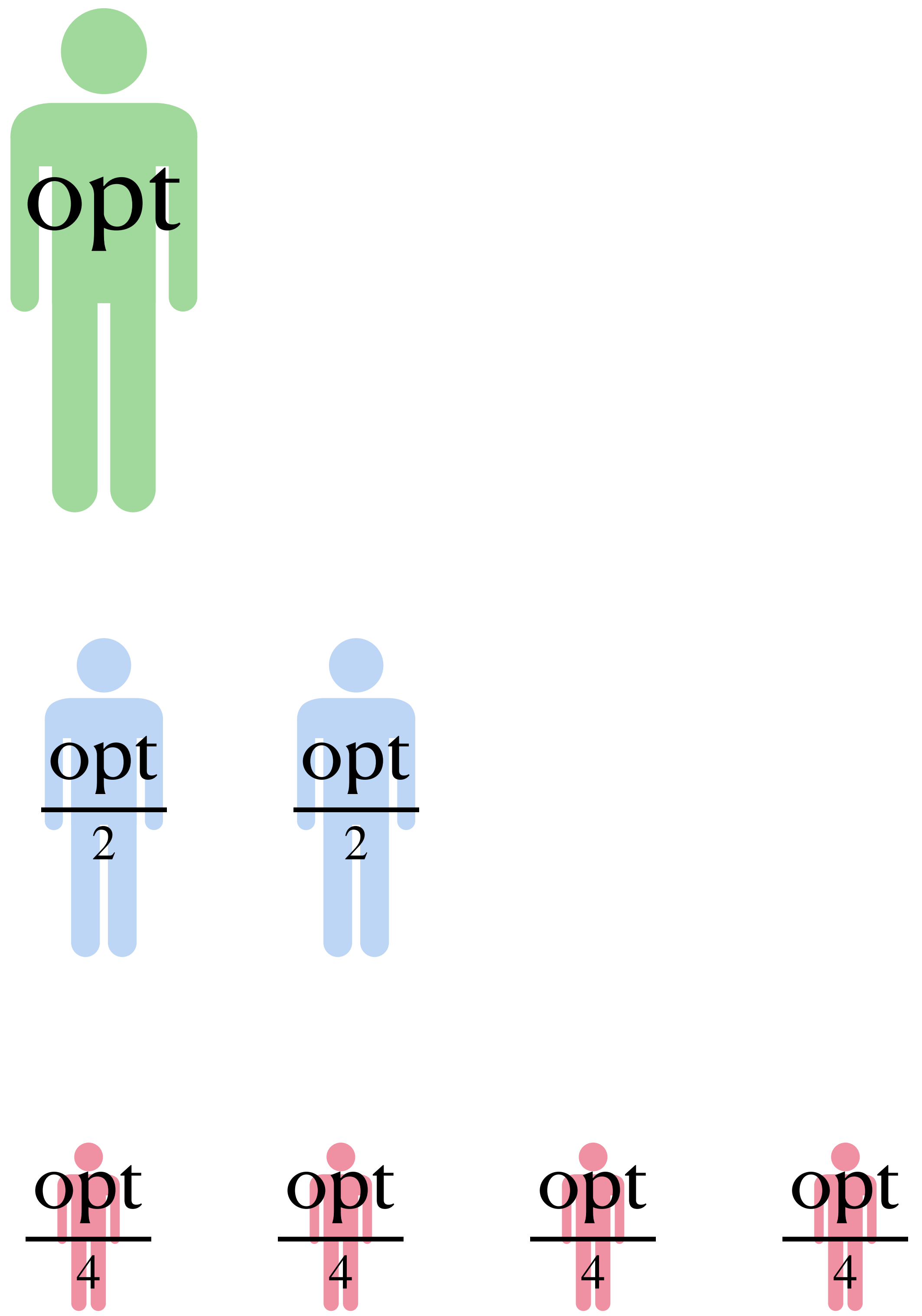
Idea: high cost secretaries arrive first



Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first

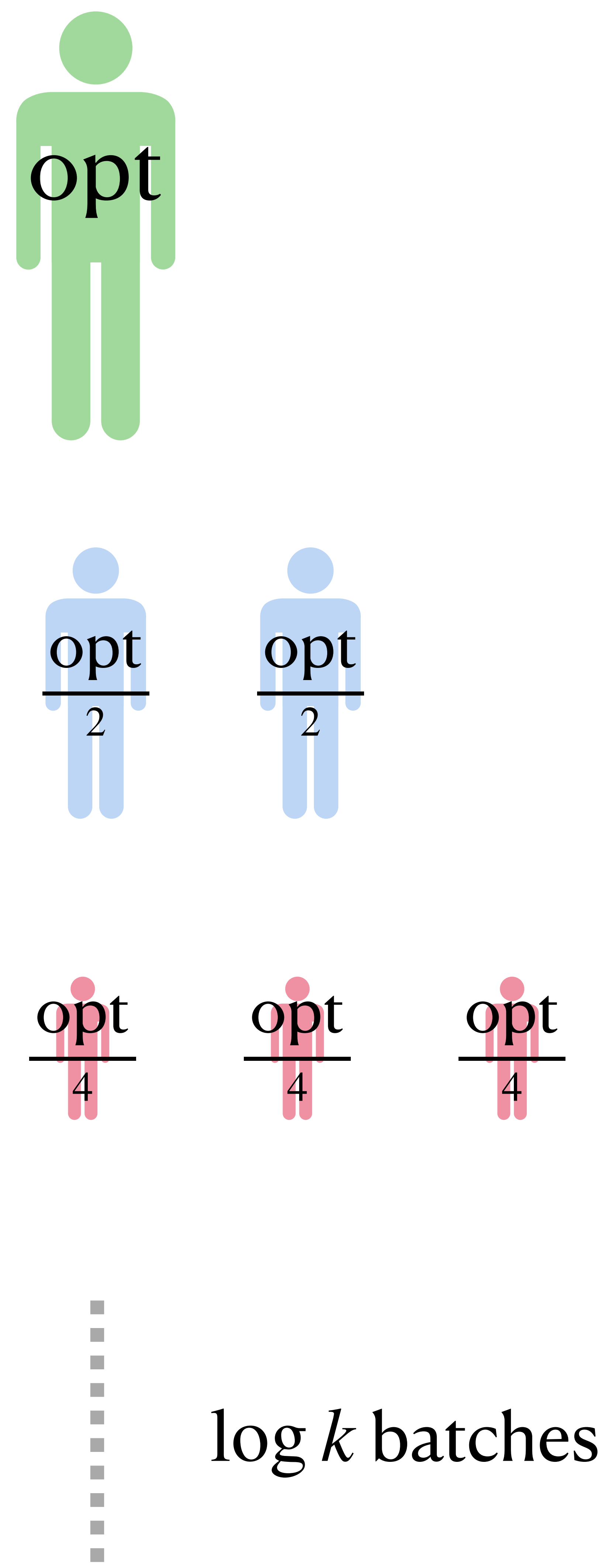
Batch i:
2^i candidates,
each w/ cost $\frac{\text{opt}}{2^i}$



Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first

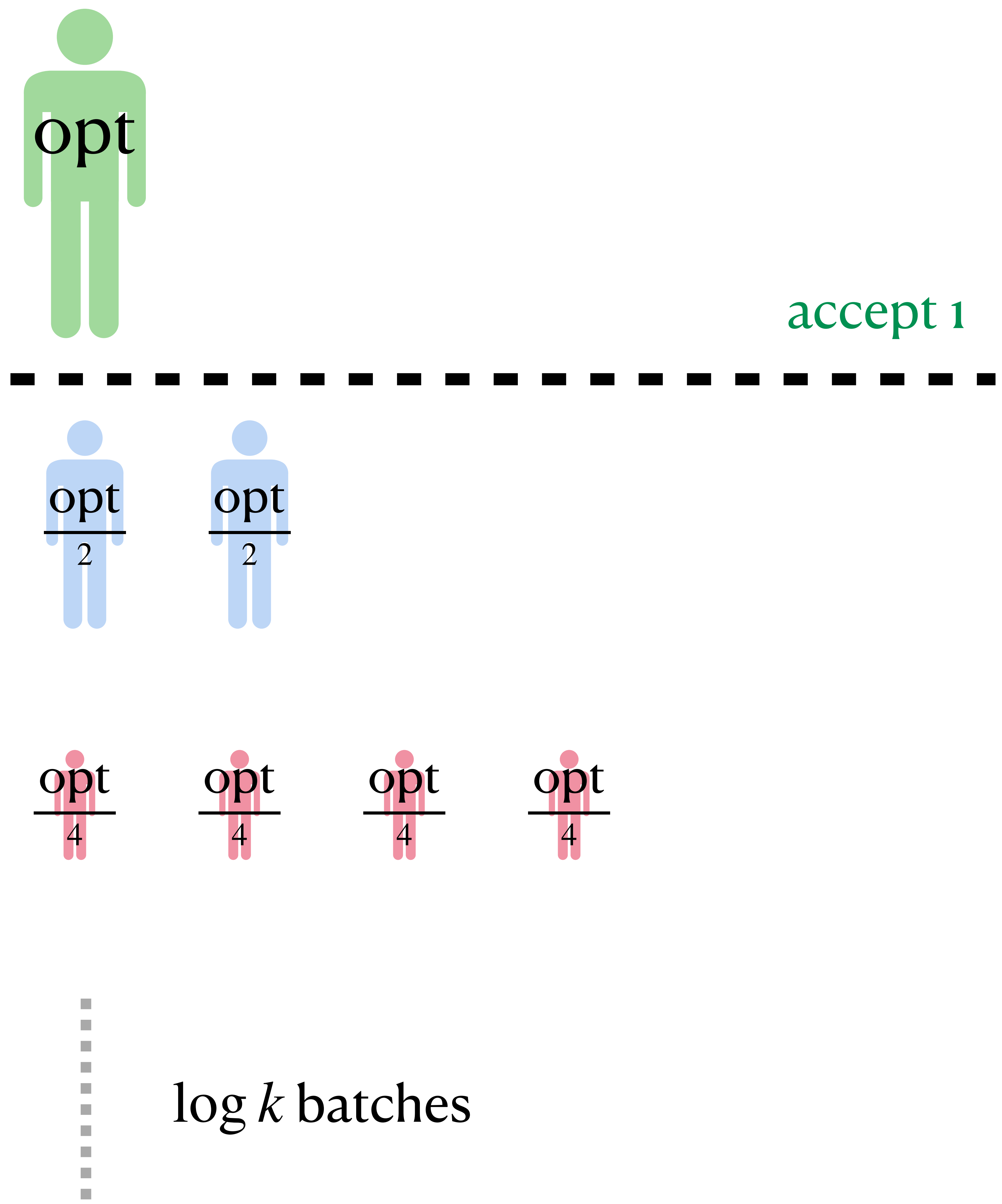
Batch i:
2^i candidates,
each w/ cost $\frac{opt}{2^i}$



Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first

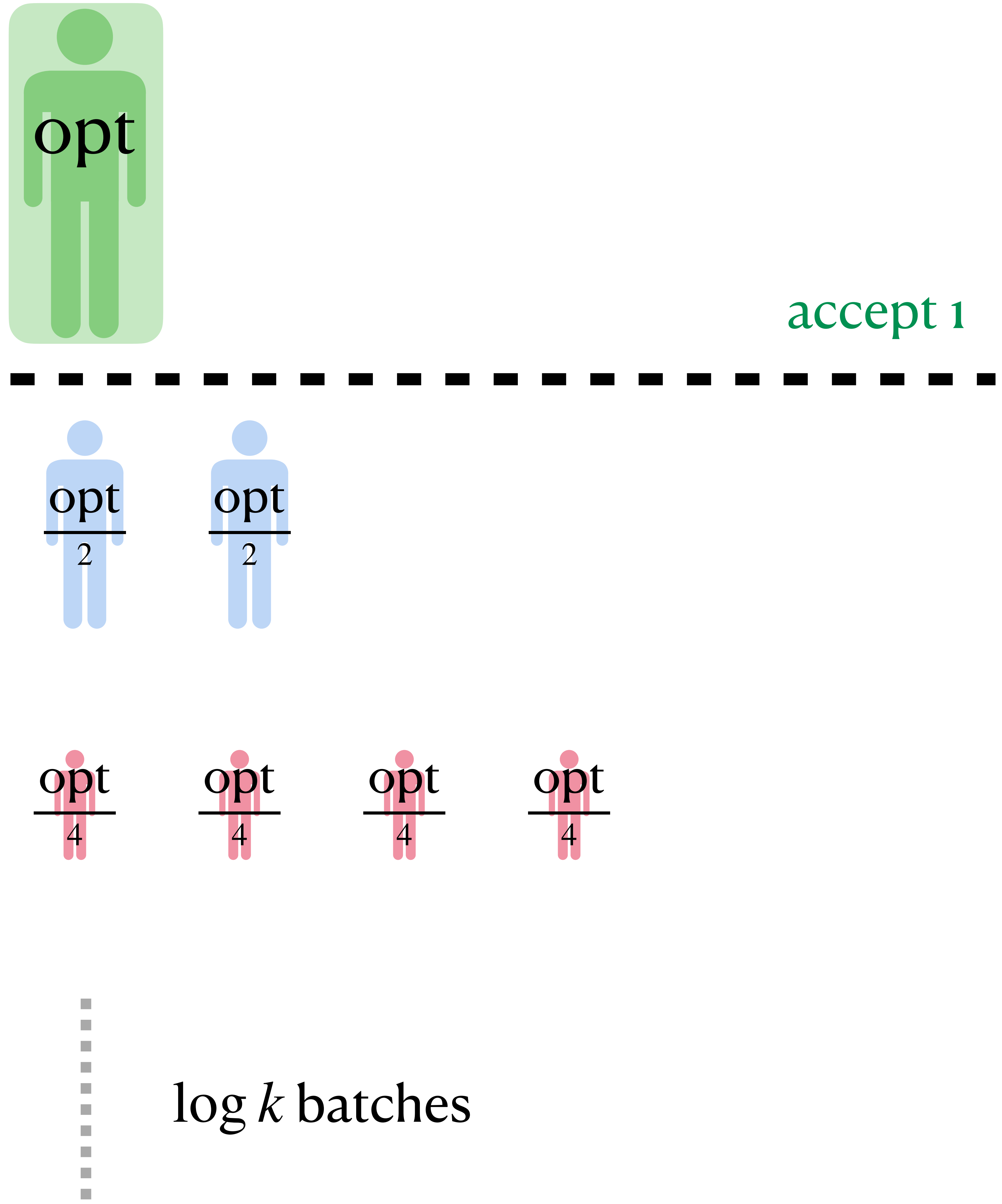
Batch i:
2^i candidates,
each w/ cost $\frac{\text{opt}}{2^i}$



Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first

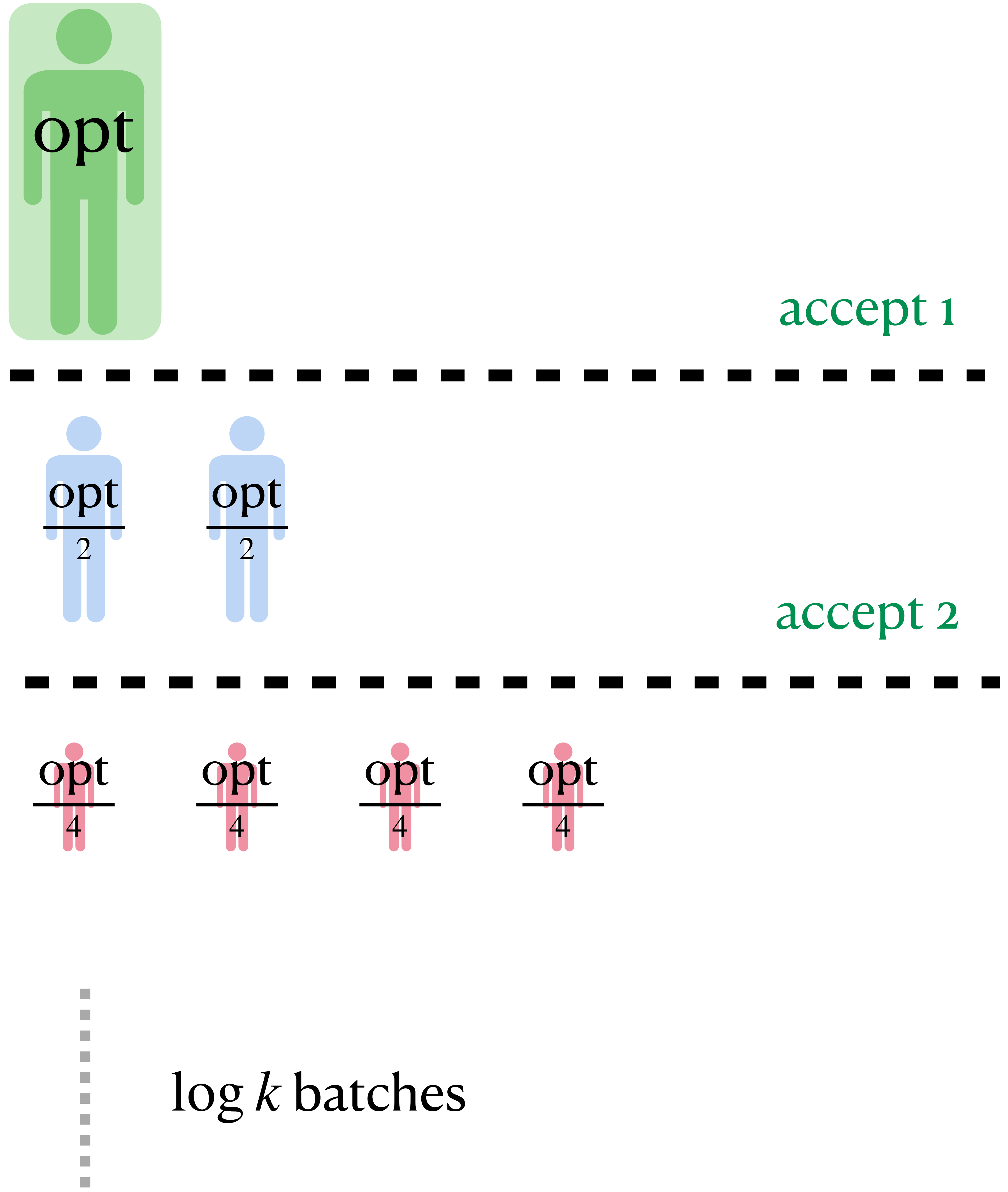
Batch i:
2^i candidates,
each w/ cost $\frac{opt}{2^i}$



Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first

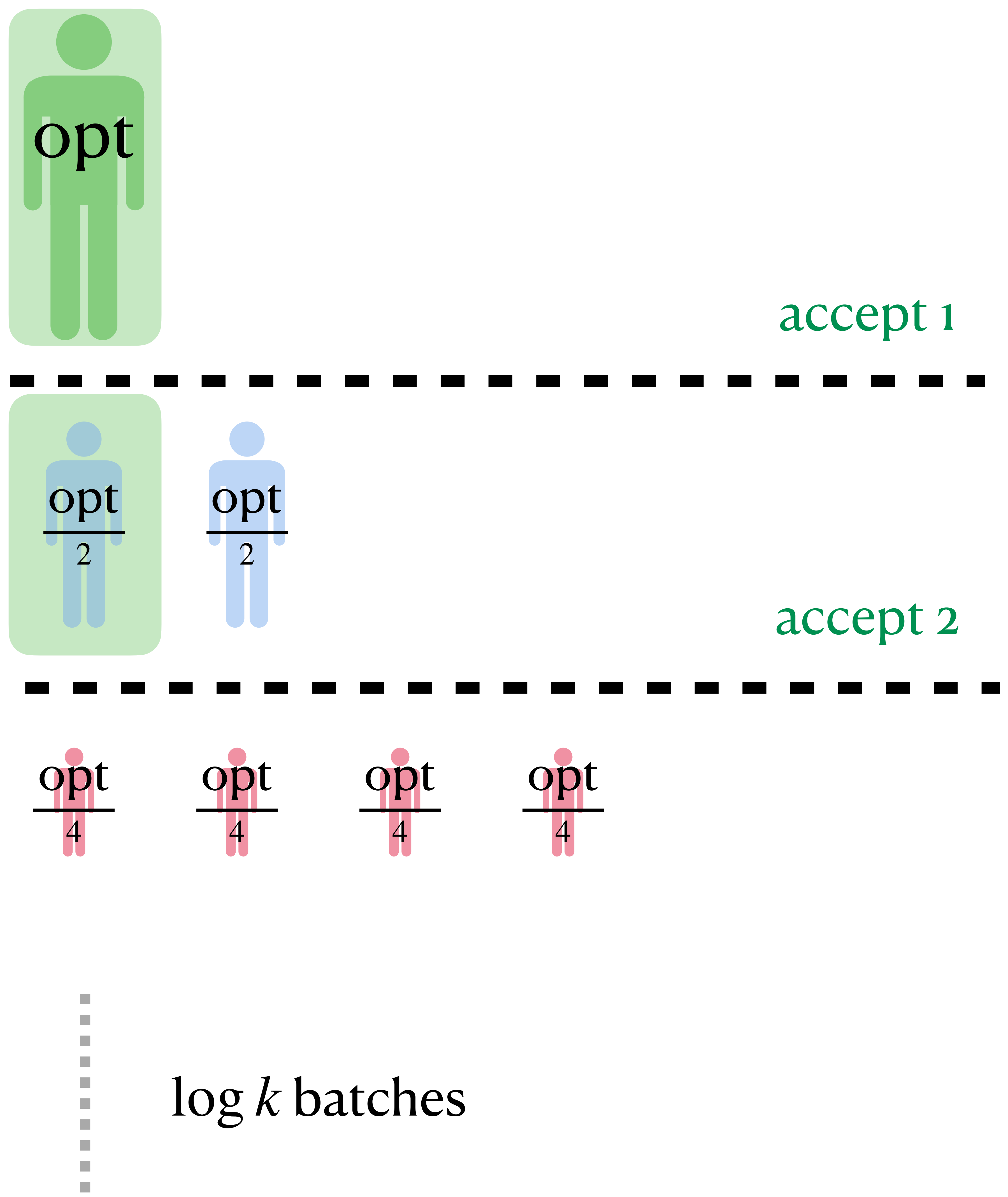
Batch i:
2^i candidates,
each w/ cost $\frac{\text{opt}}{2^i}$



Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first

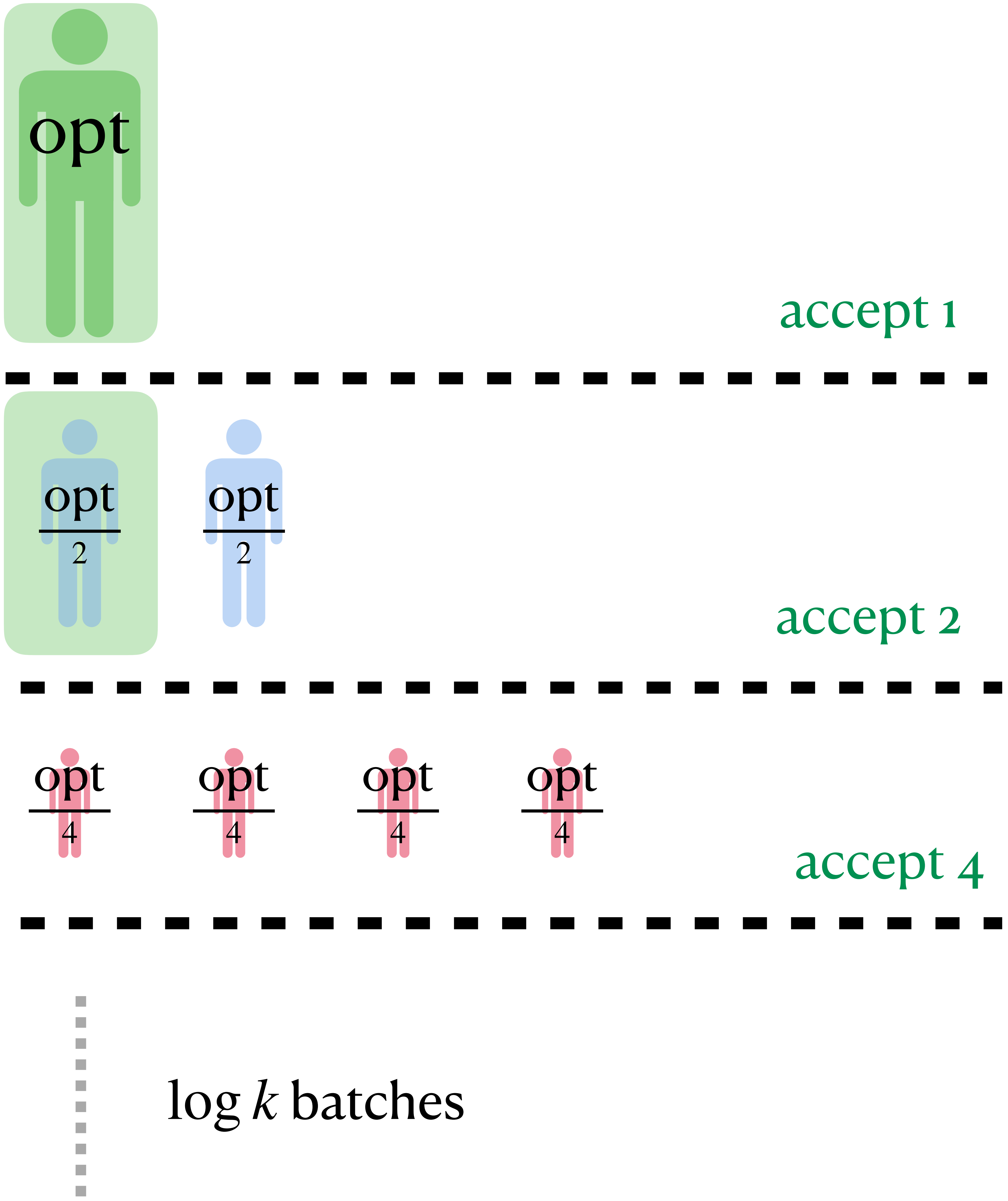
Batch i:
2^i candidates,
each w/ cost $\frac{\text{opt}}{2^i}$



Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first

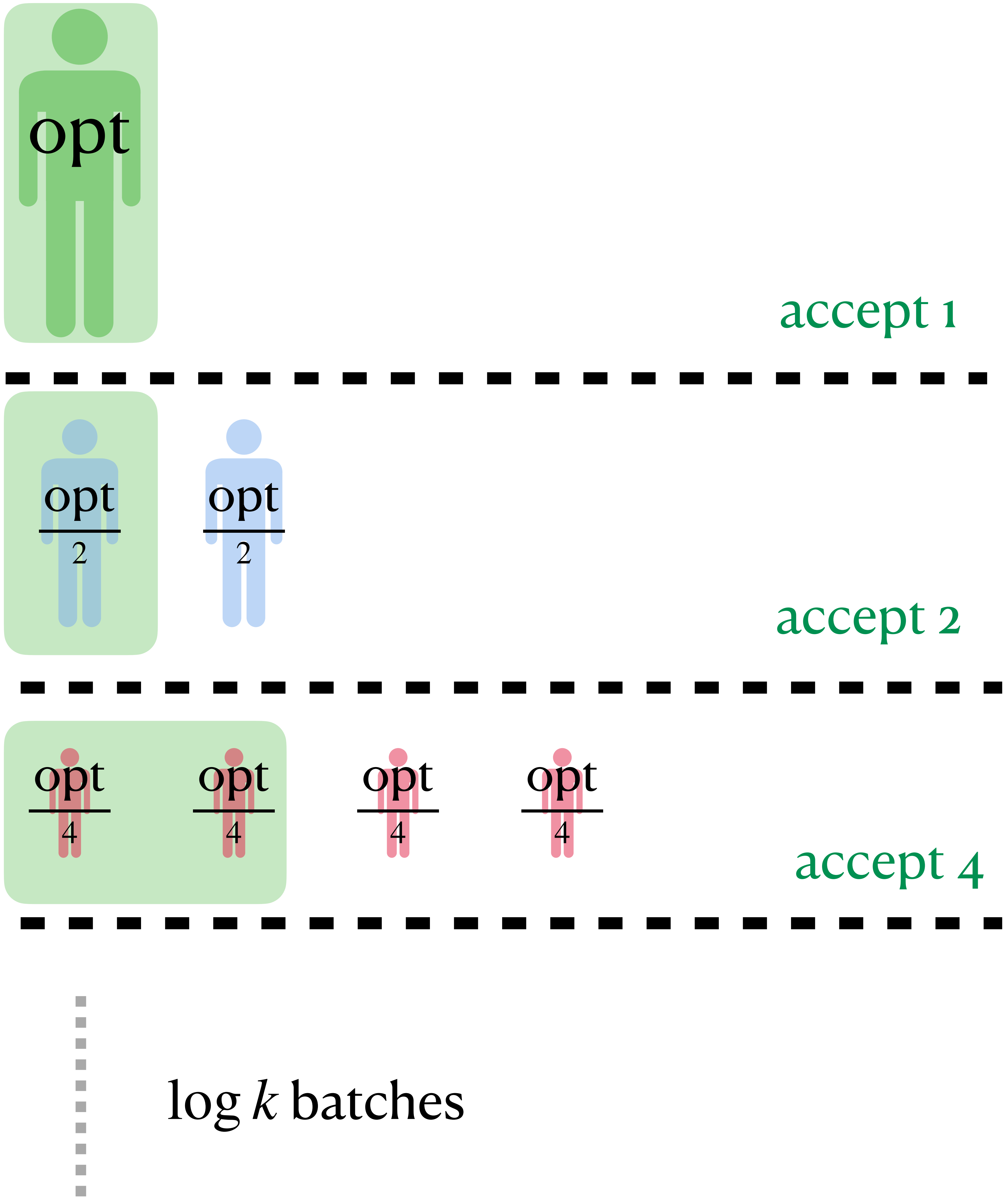
Batch i:
2^i candidates,
each w/ cost $\frac{\text{opt}}{2^i}$



Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first

Batch i:
2^i candidates,
each w/ cost $\frac{\text{opt}}{2^i}$

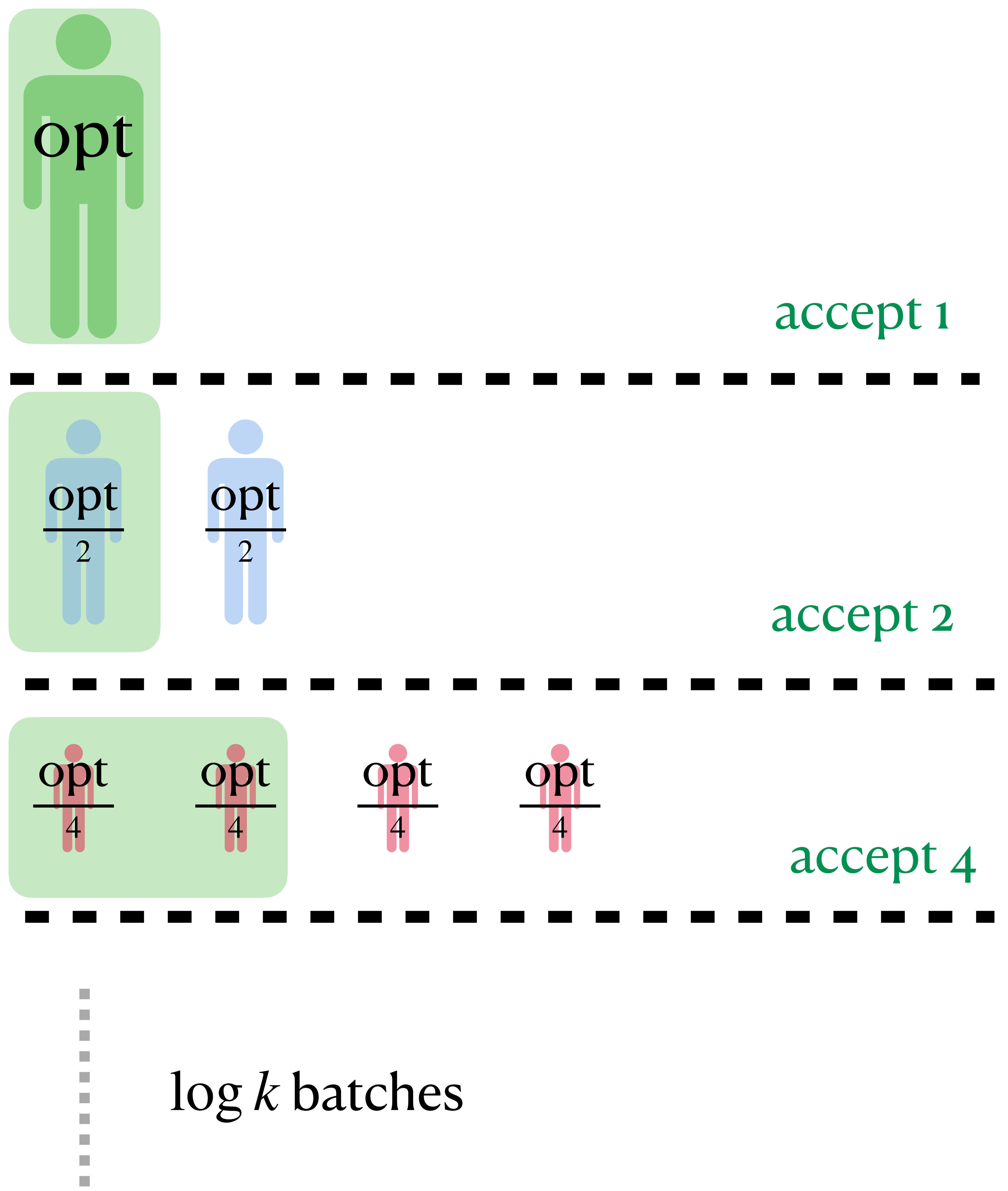


Lower Bound: Adversarial Order

Idea: high cost secretaries arrive first

Batch i:
2^i candidates,
each w/ cost $\frac{opt}{2^i}$

So we accrue $\Omega(B \cdot \log k)$ total cost!



Lower Bound: Random Order

Lower Bound: Random Order

Acceptance Property: with probability 1, hire $s(i)$ candidates among first i candidates, for all i

Lower Bound: Random Order

Acceptance Property: with probability 1, hire $s(i)$ candidates among first i candidates, for all i

Lemma: Every randomized algorithm that is competitive in **random** order model has the acceptance property.

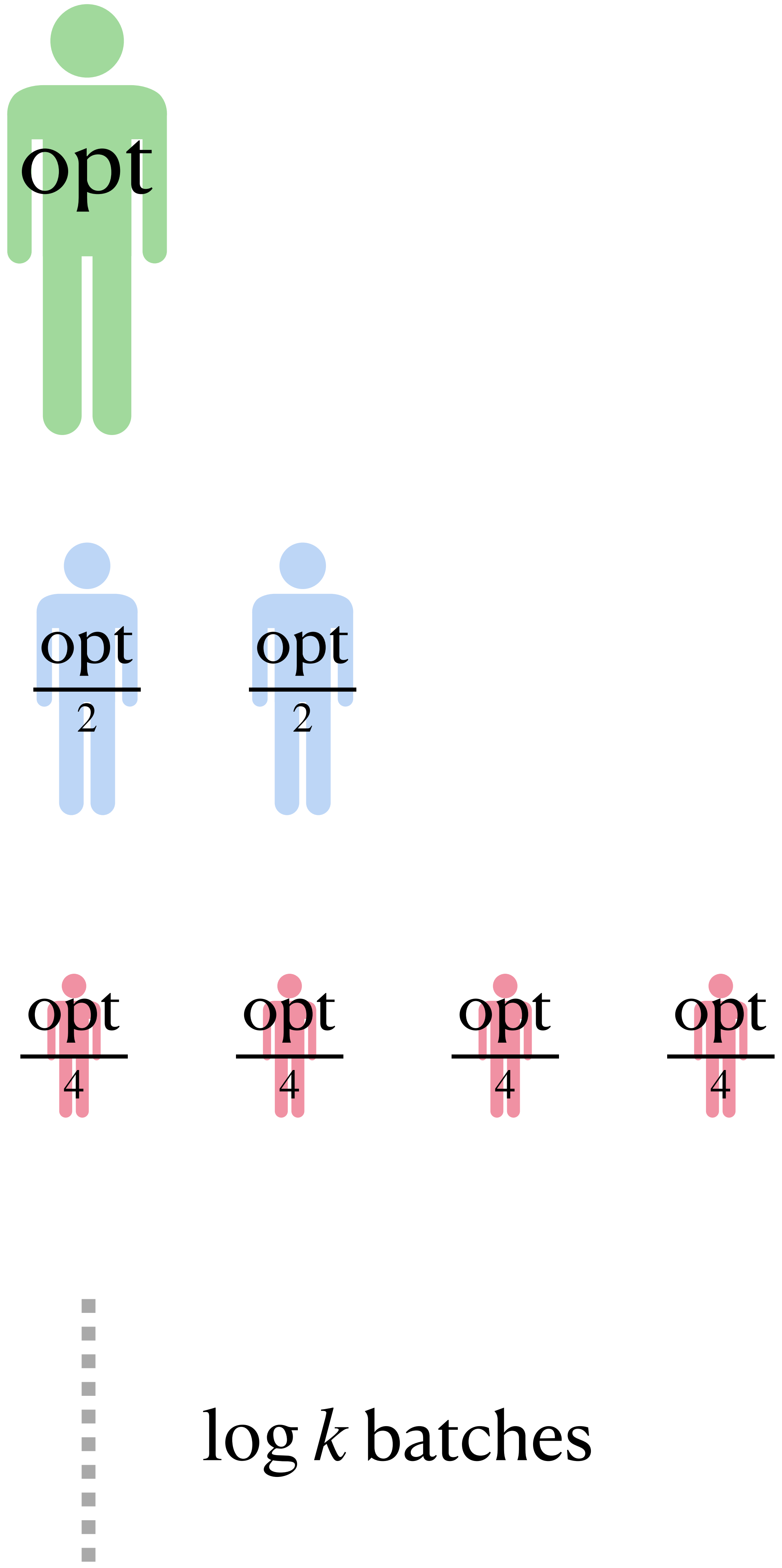
Lower Bound: Random Order

Acceptance Property: with probability 1, hire $s(i)$ candidates among first i candidates, for all i

Lemma: Every randomized algorithm that is competitive in **random** order model has the acceptance property.

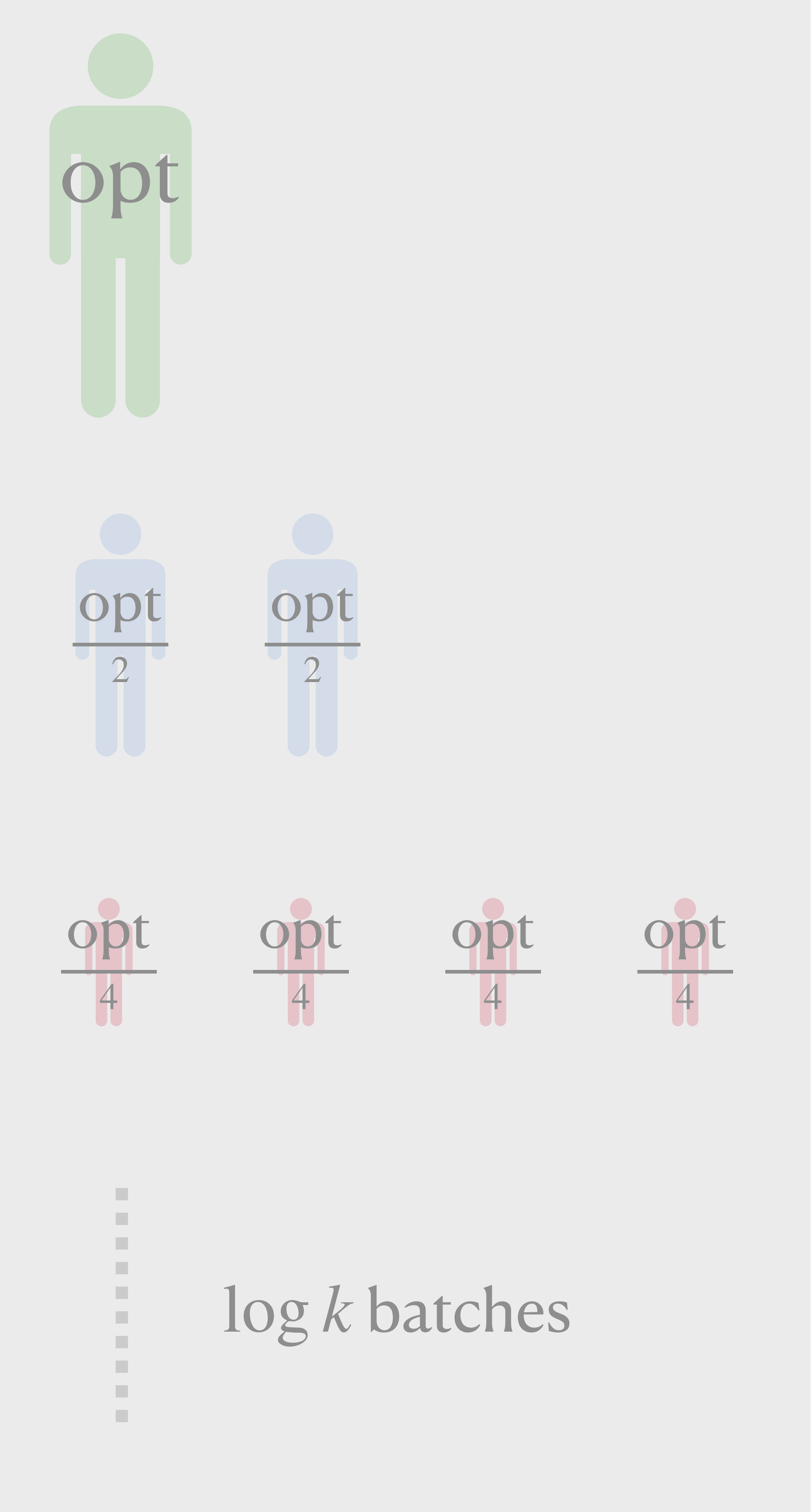
Idea: construct instance such that a random permutation "looks like" adversarial instance with constant probability

Lower Bound: Random Order



Idea: construct instance such that a random permutation "looks like" adversarial instance with constant probability

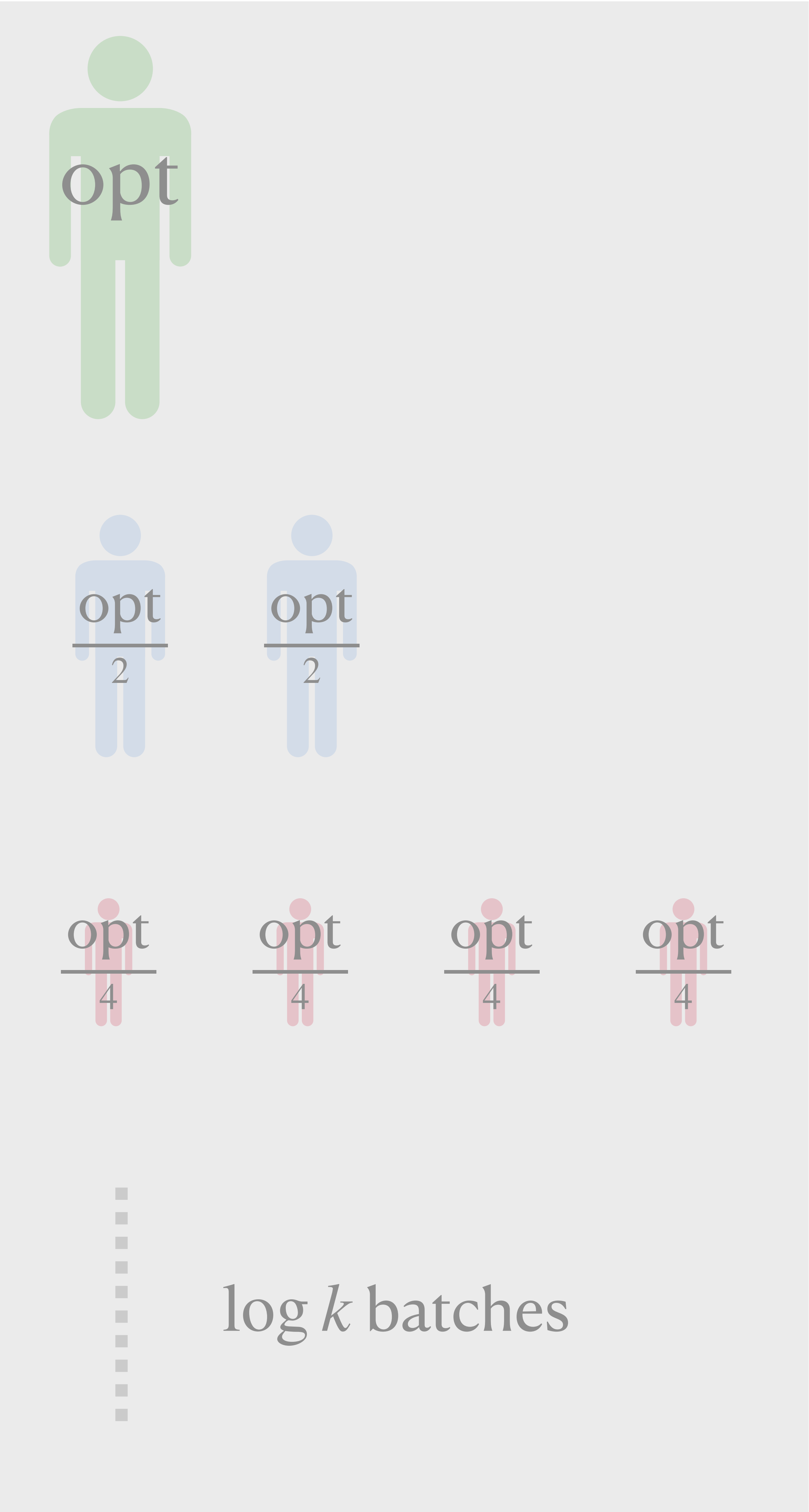
Lower Bound: Random Order



Idea: construct instance such that a random permutation "looks like" adversarial instance with constant probability

Lower Bound: Random Order

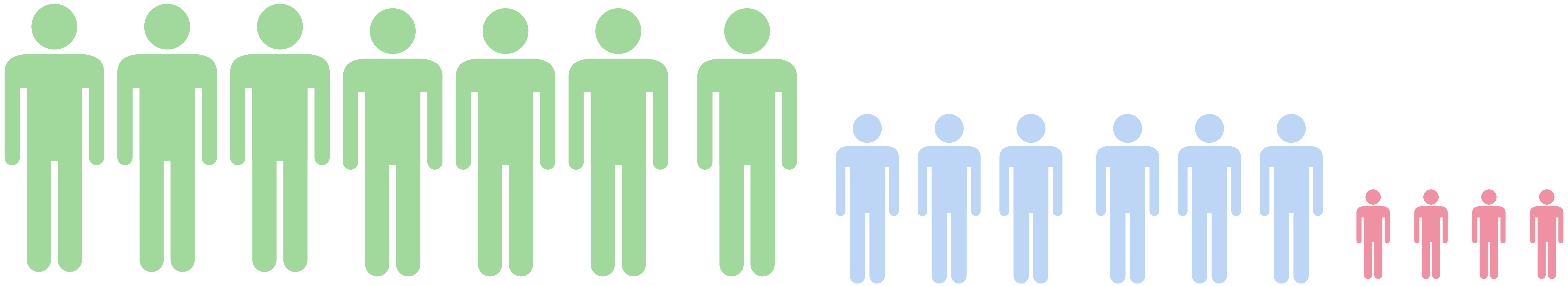
Make many copies of each batch, with earlier batches duplicated more

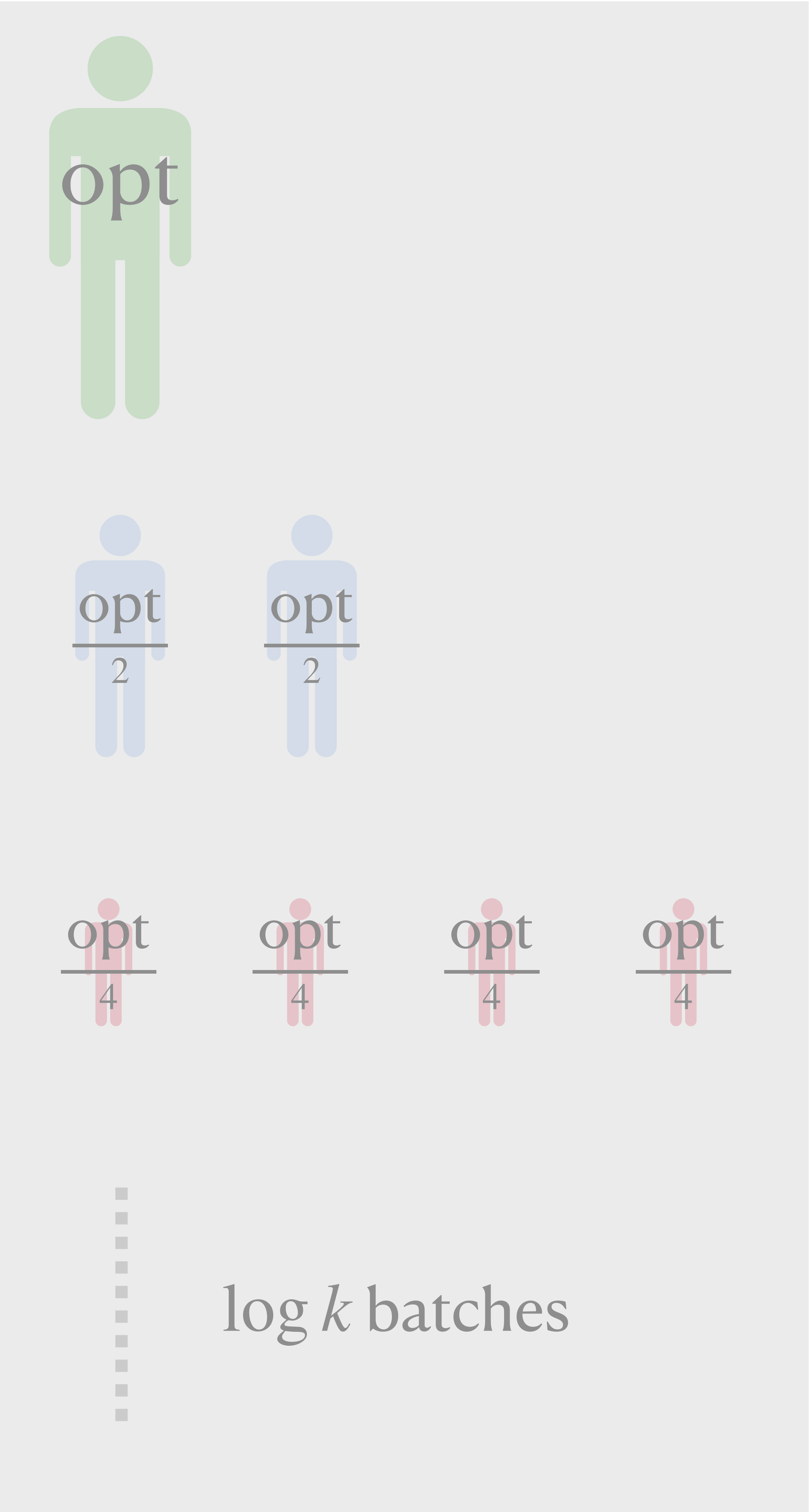


Idea: construct instance such that a random permutation "looks like" adversarial instance with constant probability

Lower Bound: Random Order

Make many copies of each batch, with earlier batches duplicated more

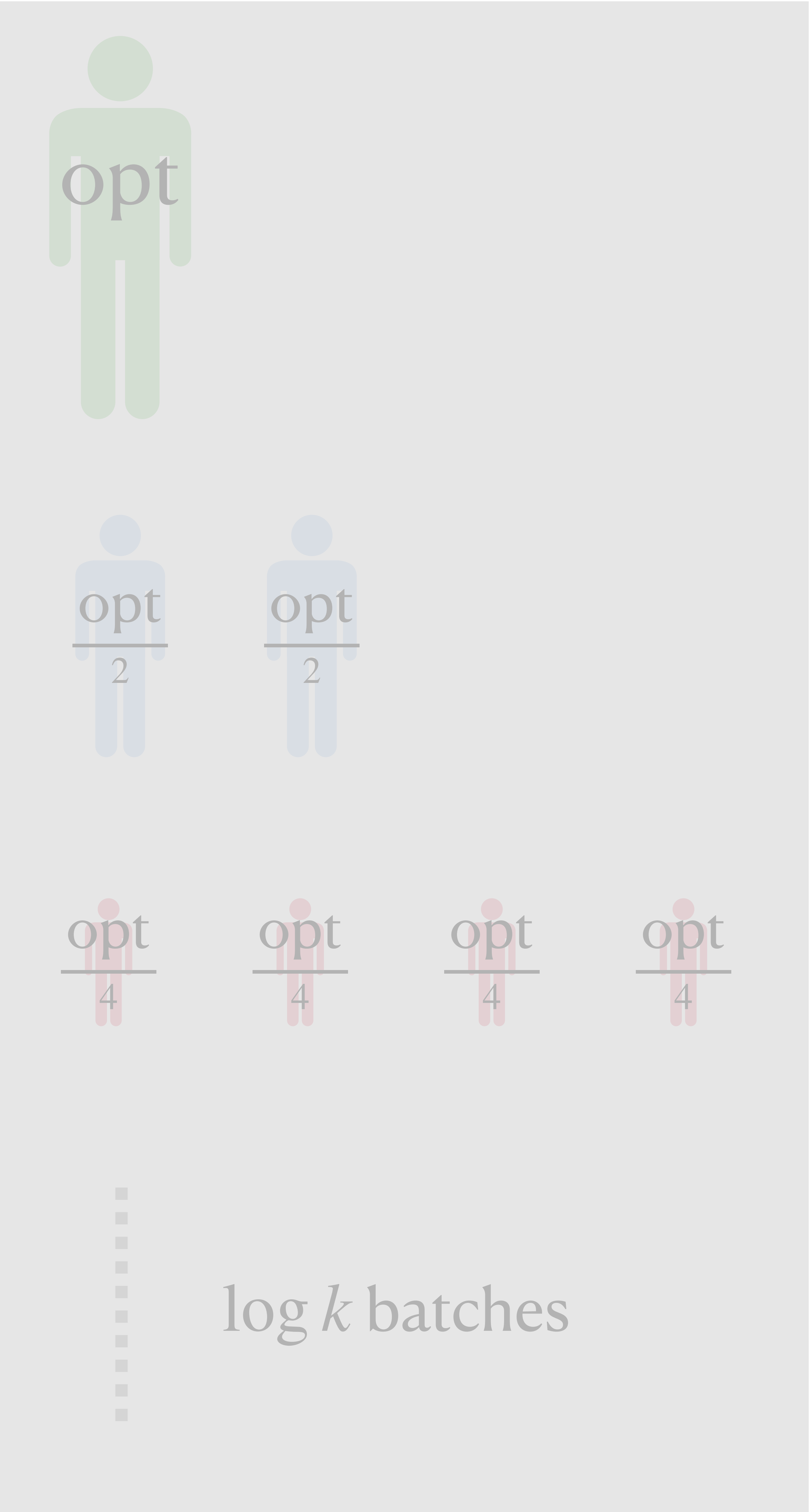




Idea: construct instance such that a random permutation "looks like" adversarial instance with constant probability

Lower Bound: Random Order

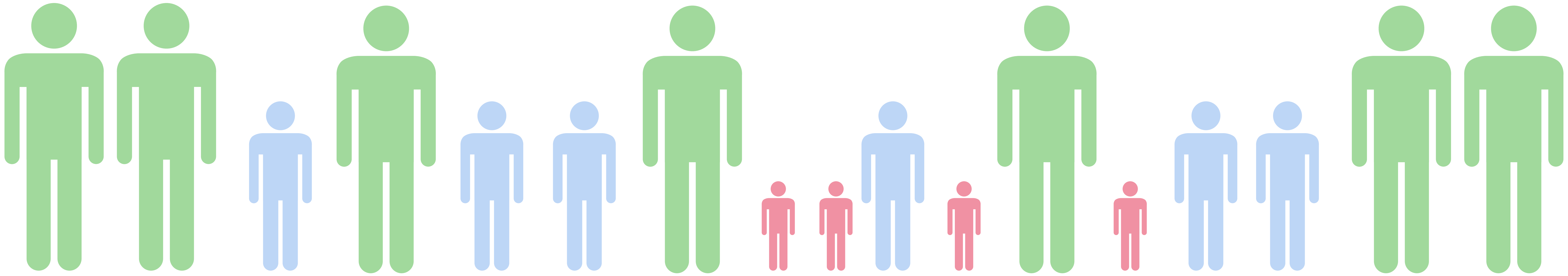
Make many copies of each batch, with earlier batches duplicated more

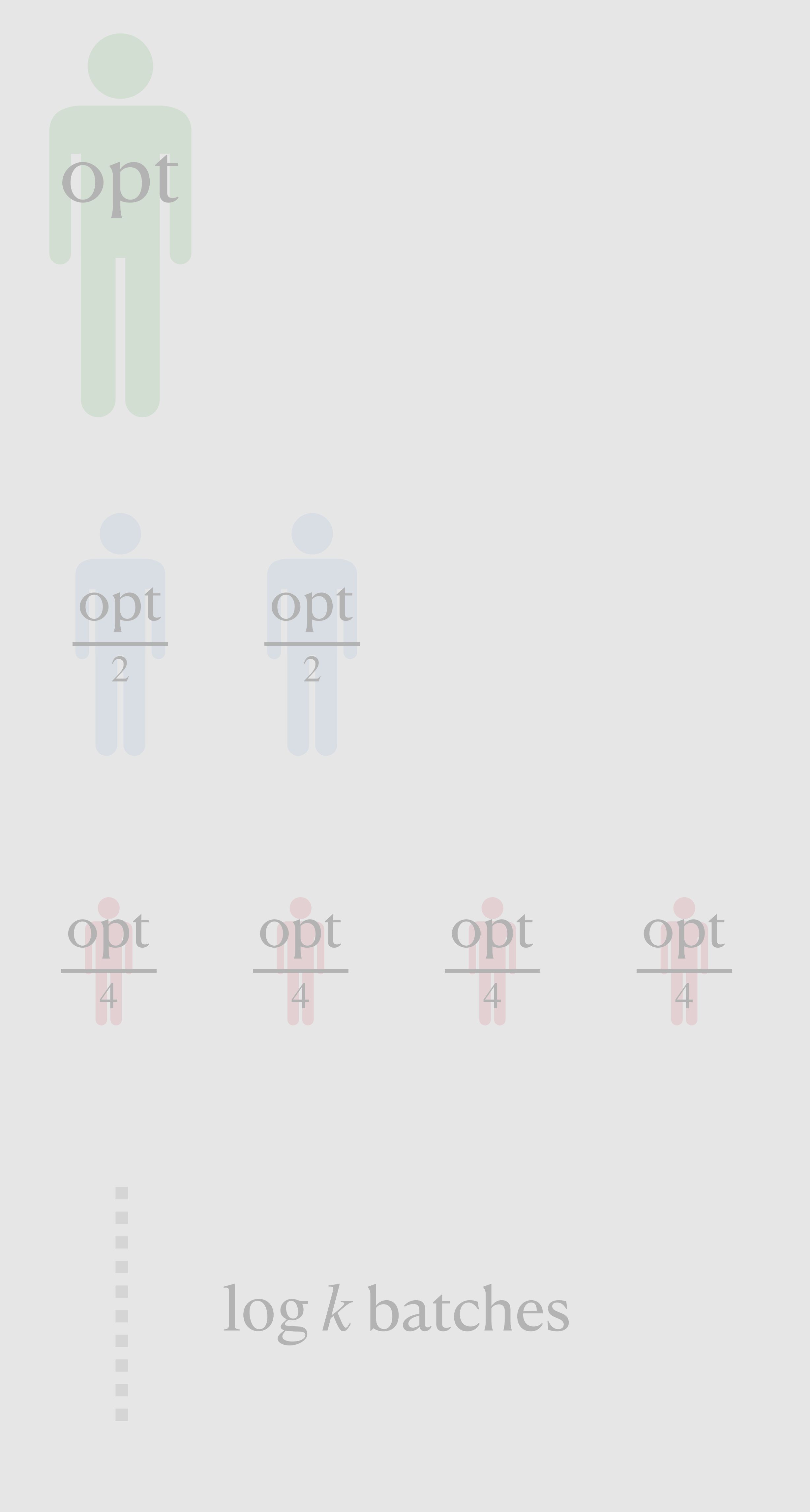


Idea: construct instance such that a random permutation "looks like" adversarial instance with constant probability

Lower Bound: Random Order

Make many copies of each batch, with earlier batches duplicated more

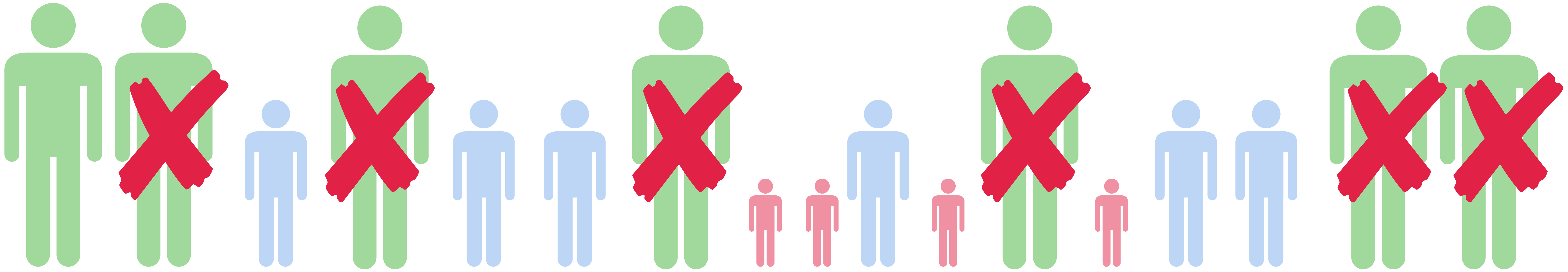


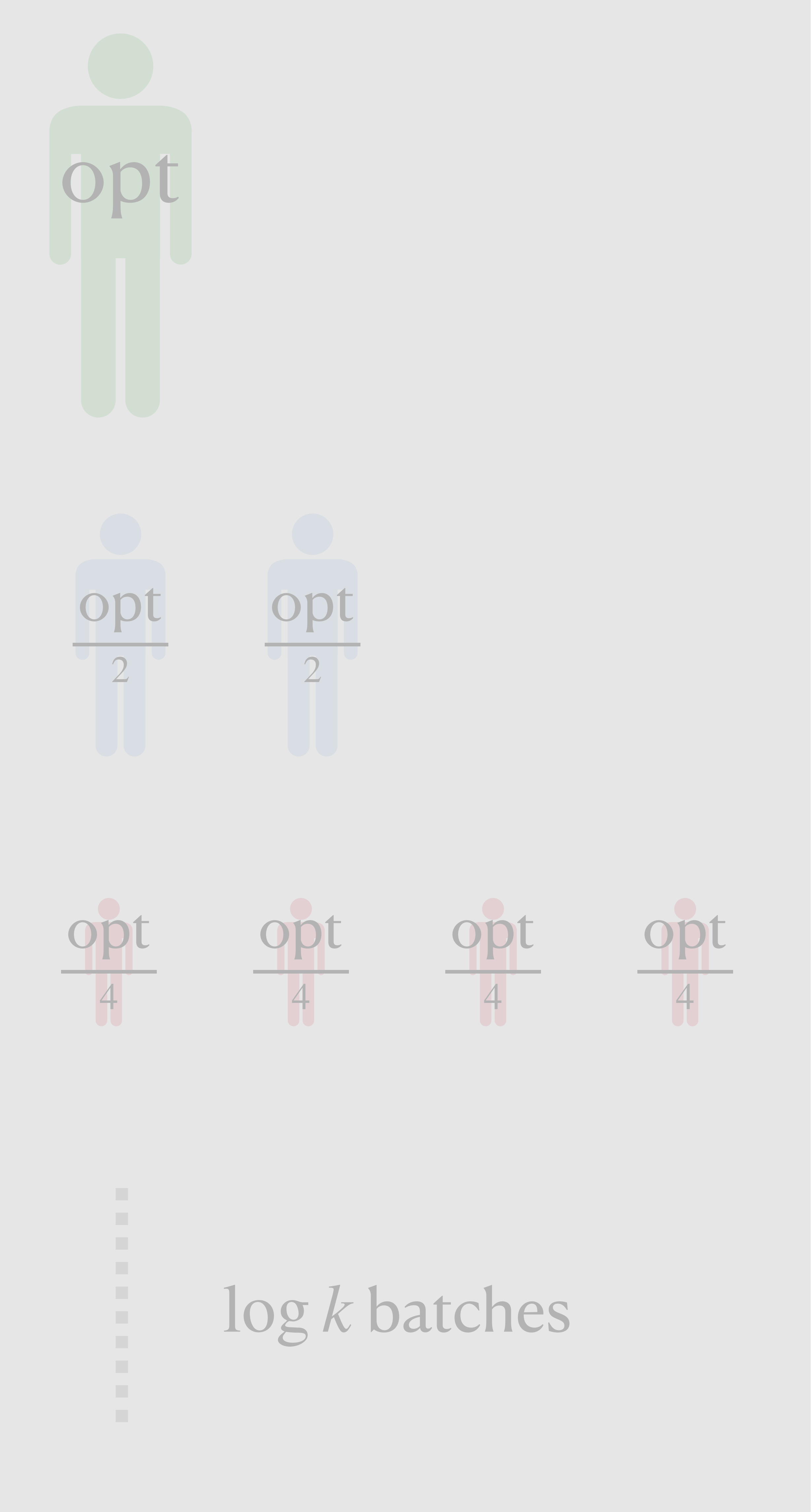


Idea: construct instance such that a random permutation "looks like" adversarial instance with constant probability

Lower Bound: Random Order

Make many copies of each batch, with earlier batches duplicated more

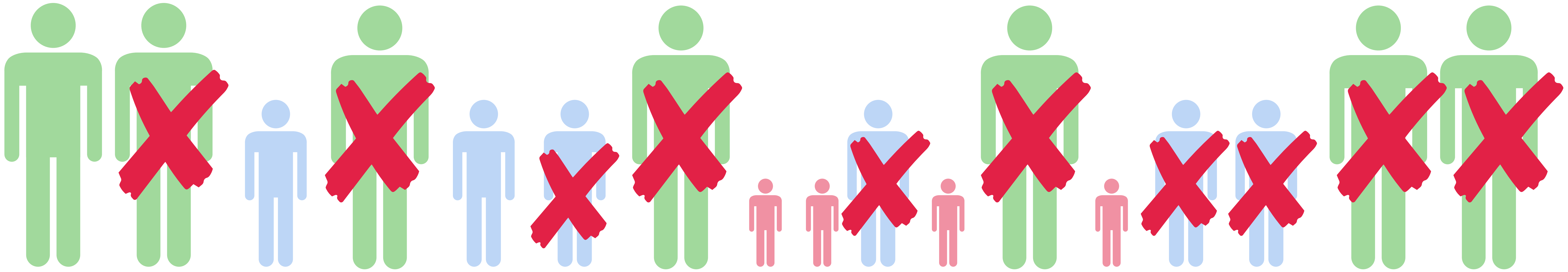


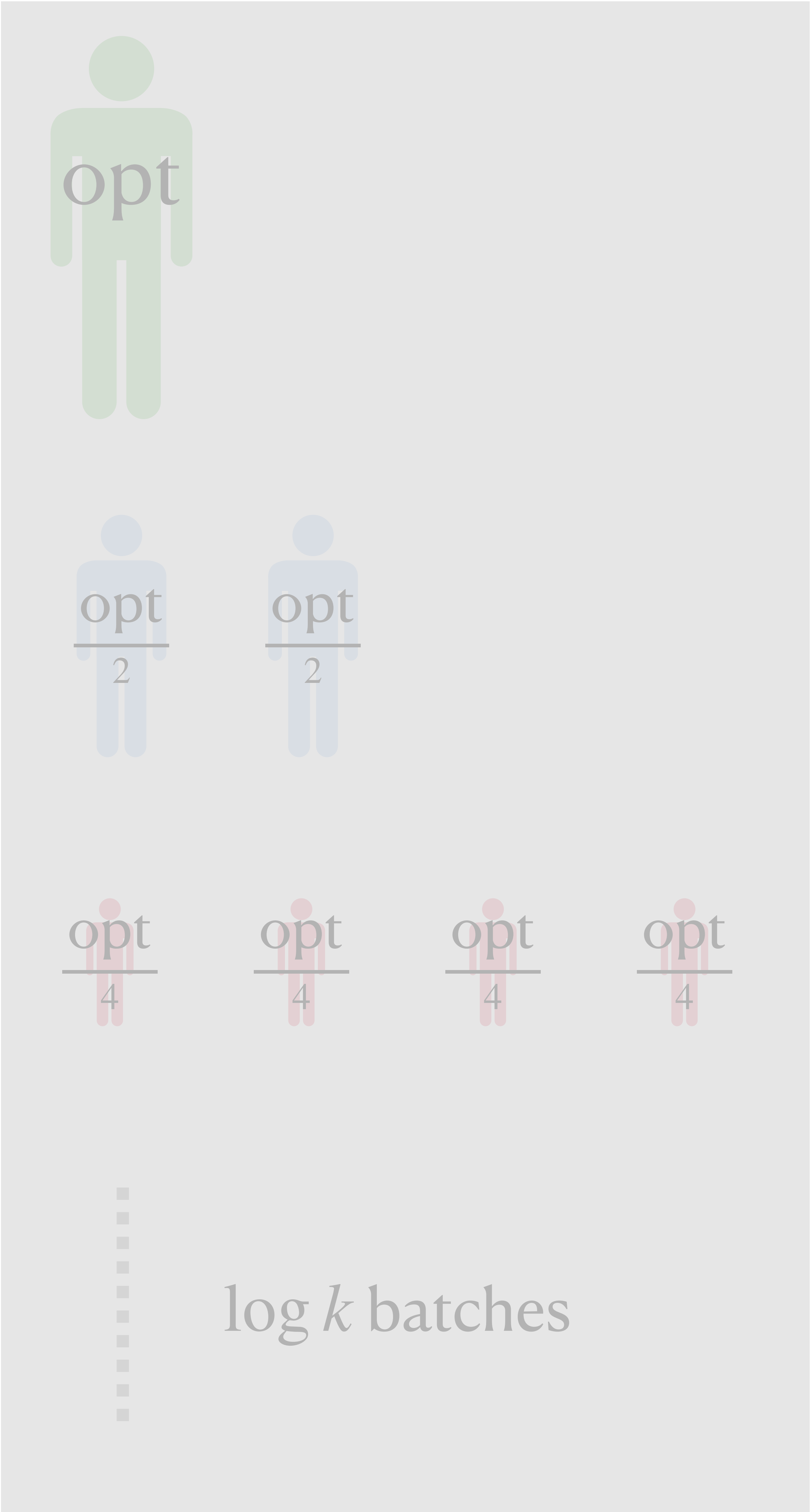


Idea: construct instance such that a random permutation "looks like" adversarial instance with constant probability

Lower Bound: Random Order

Make many copies of each batch, with earlier batches duplicated more

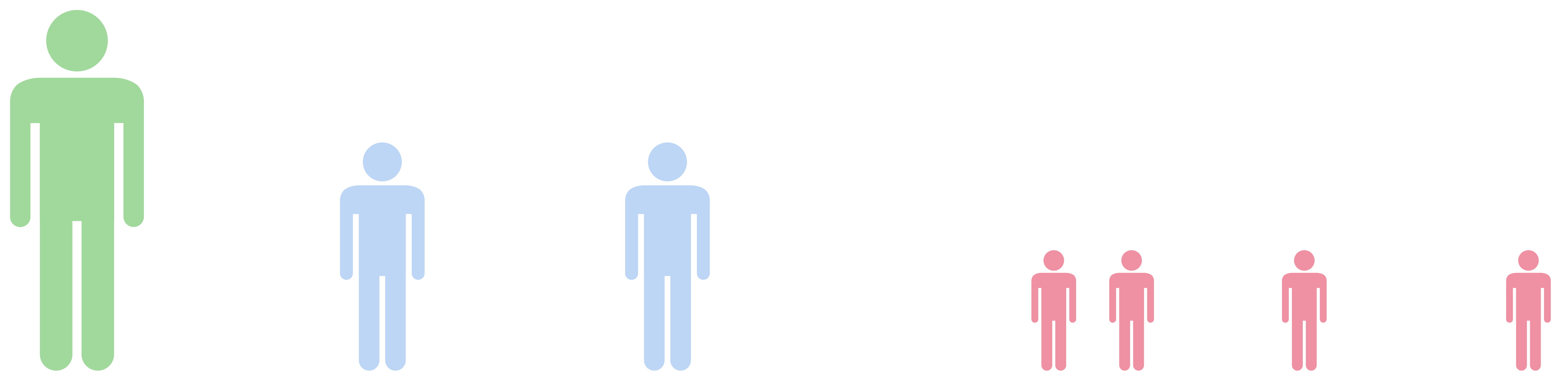


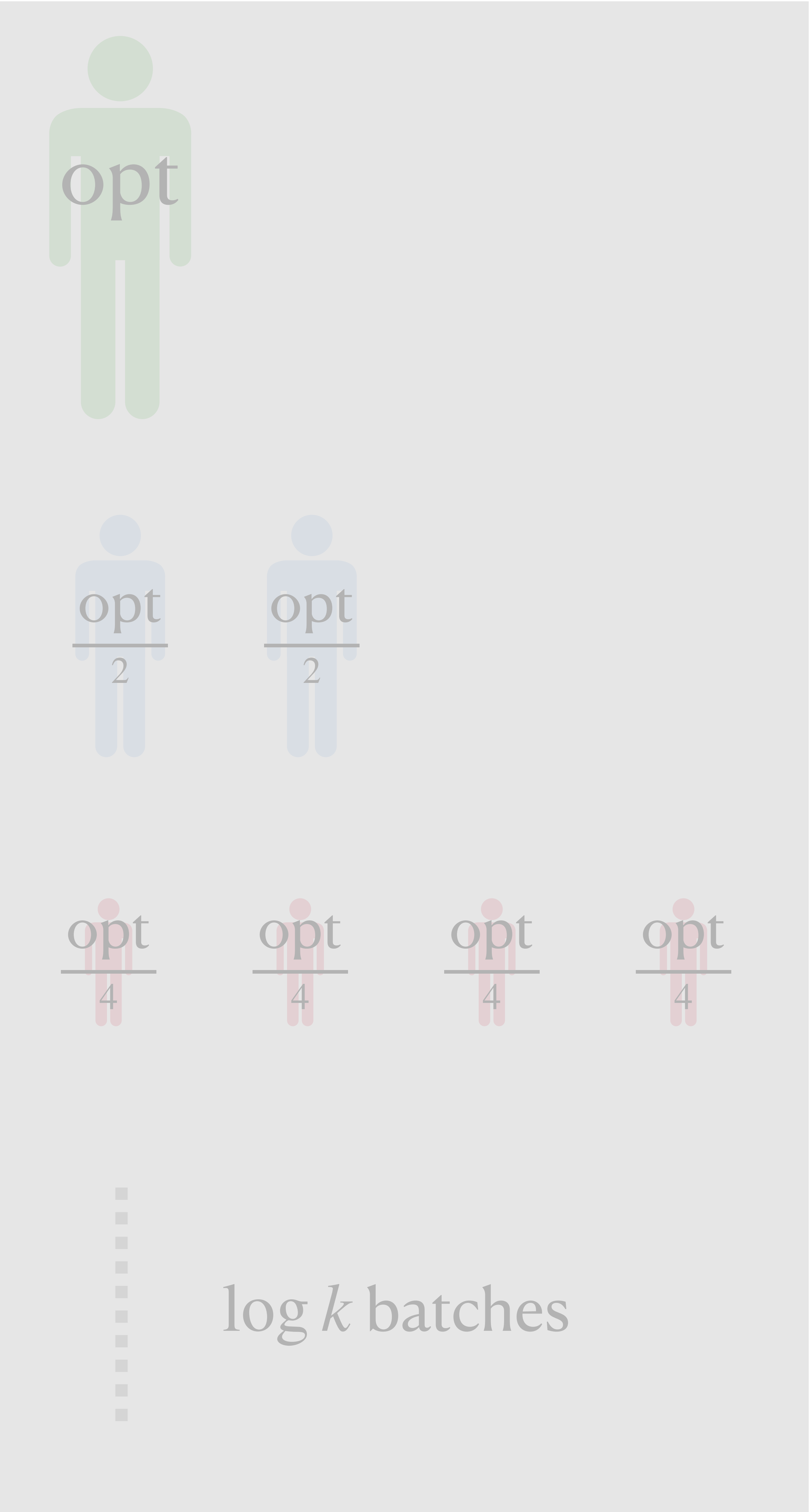


Idea: construct instance such that a random permutation "looks like" adversarial instance with constant probability

Lower Bound: Random Order

Make many copies of each batch, with earlier batches duplicated more



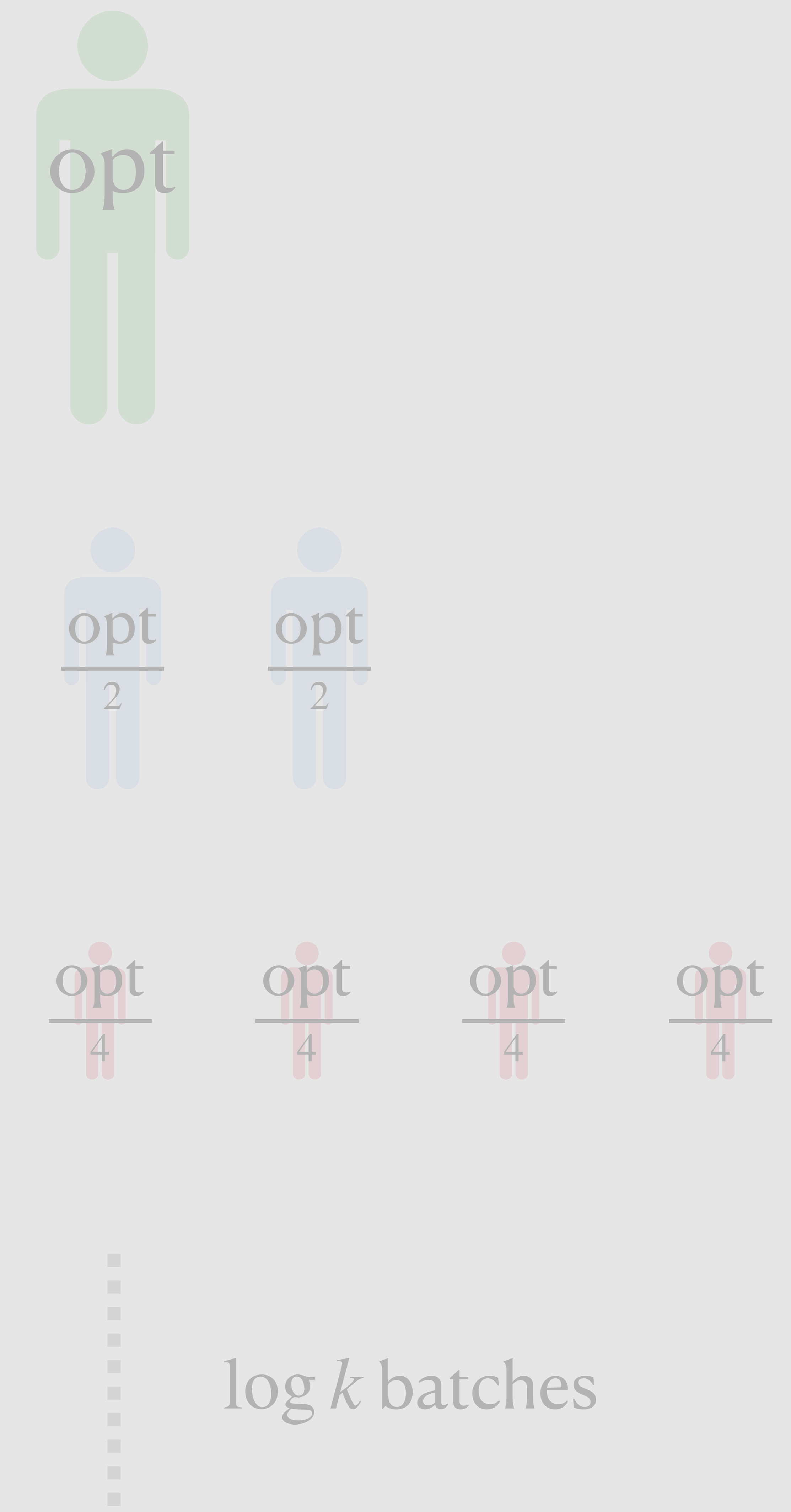


Idea: construct instance such that a random permutation "looks like" adversarial instance with

Lower Bound: Random Order

Make many copies of each batch, with earlier batches duplicated more

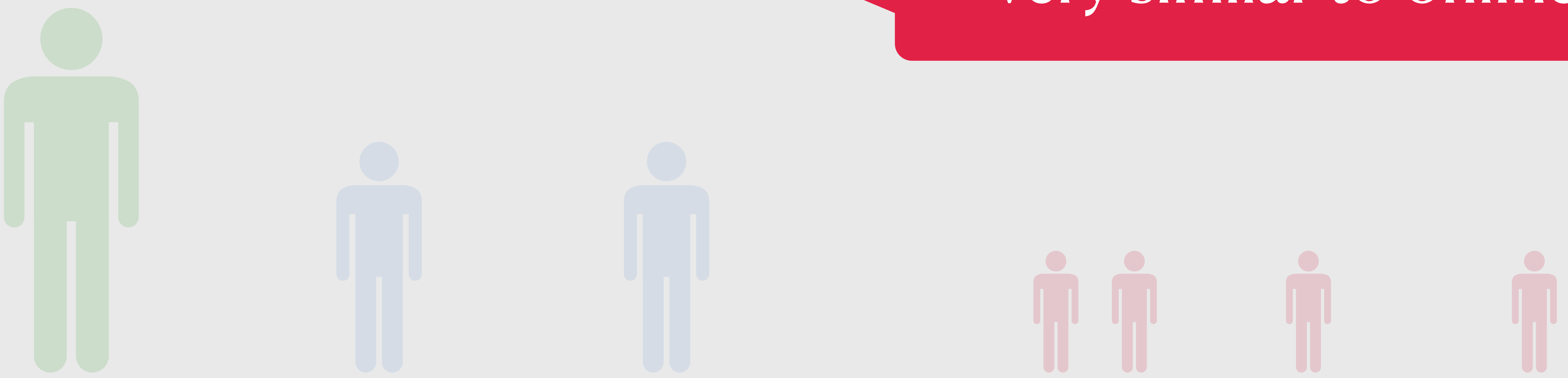
Idea: construct instance such that a random permutation "looks like" adversarial instance with



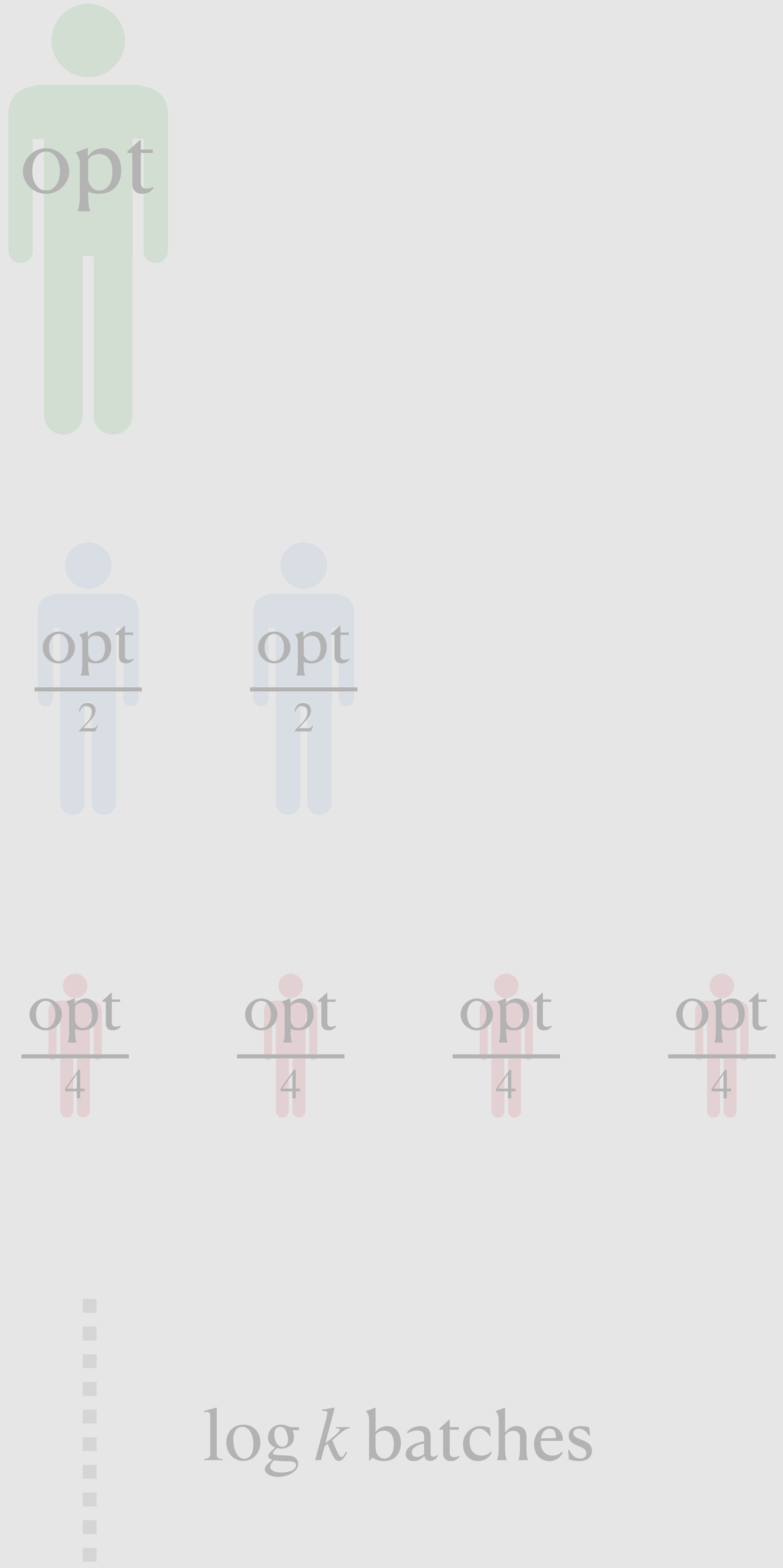
Lower Bound: Random Order

Make many copies of each batch, with earlier batches duplicated more

Very similar to online Steiner tree

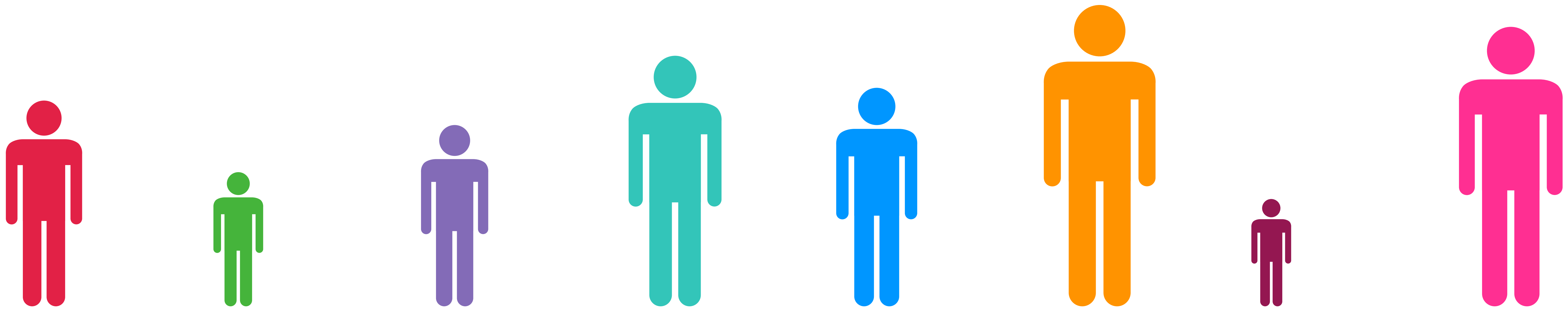


Idea: construct instance such that a random permutation "looks like" adversarial instance with



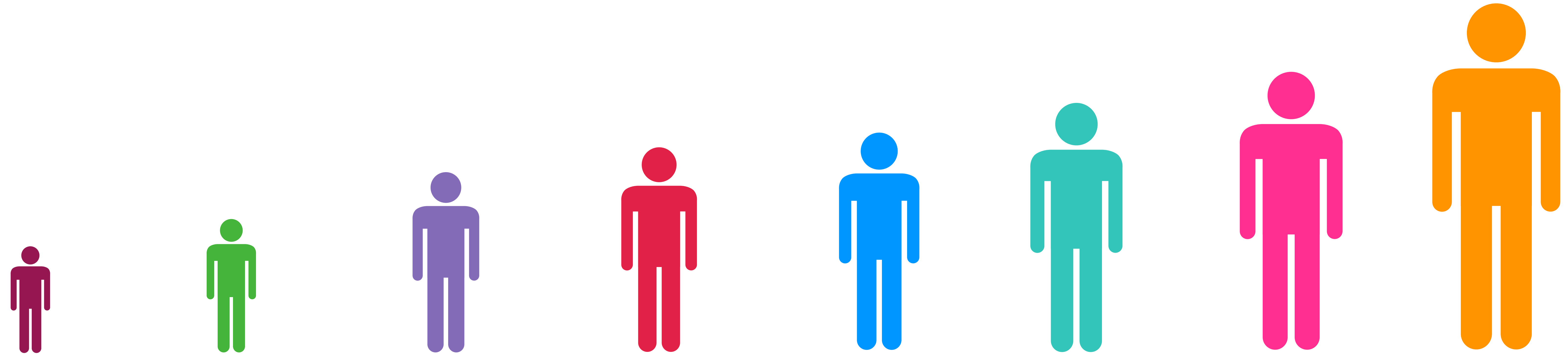
Upper Bound: Adversarial Order

First i candidates: ***(adversarial) arrival order***

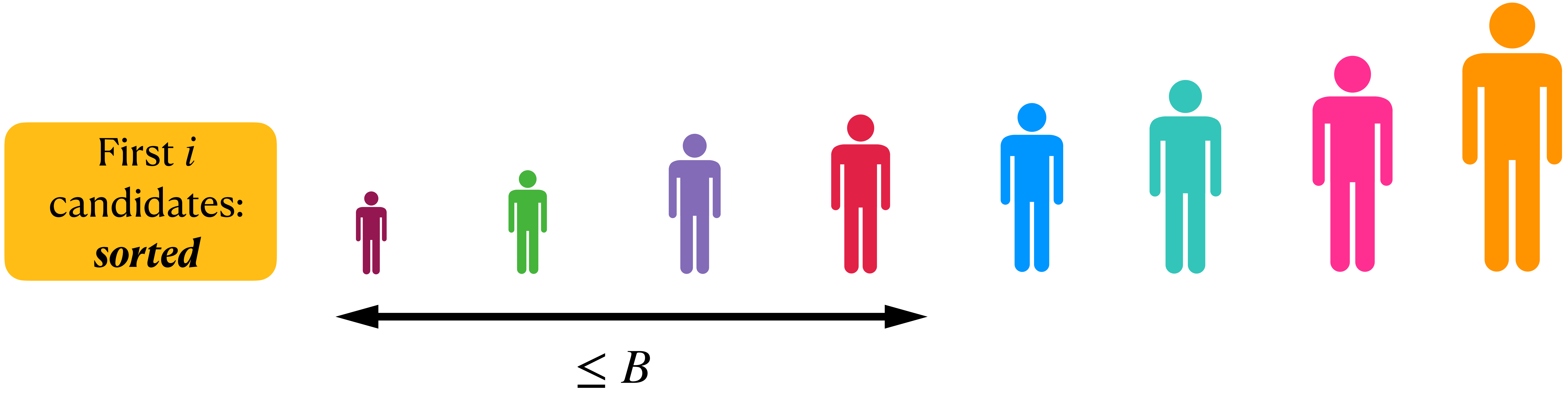


Upper Bound: Adversarial Order

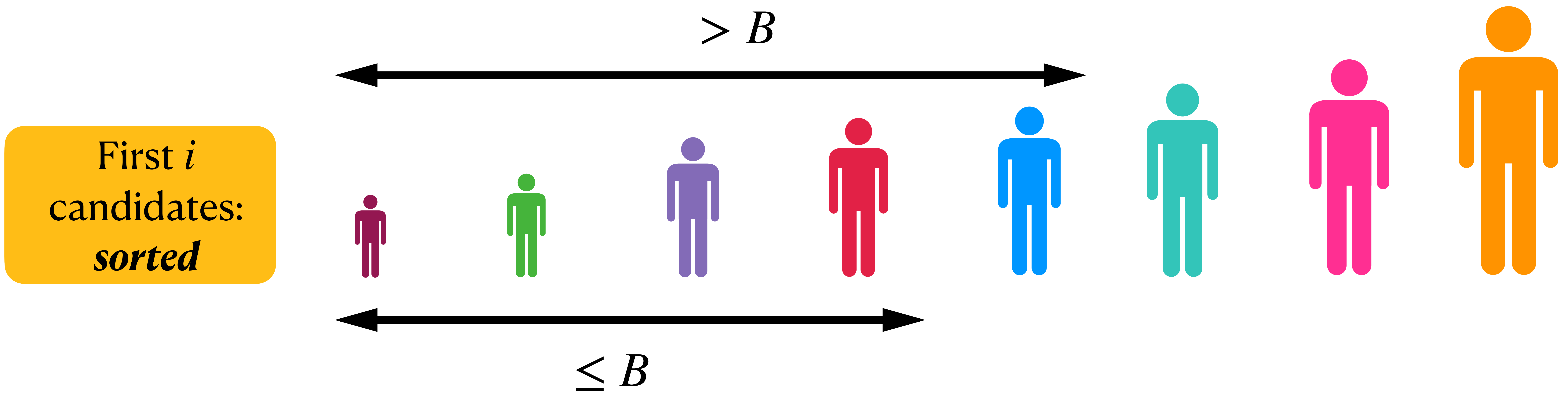
First i candidates: ***sorted***



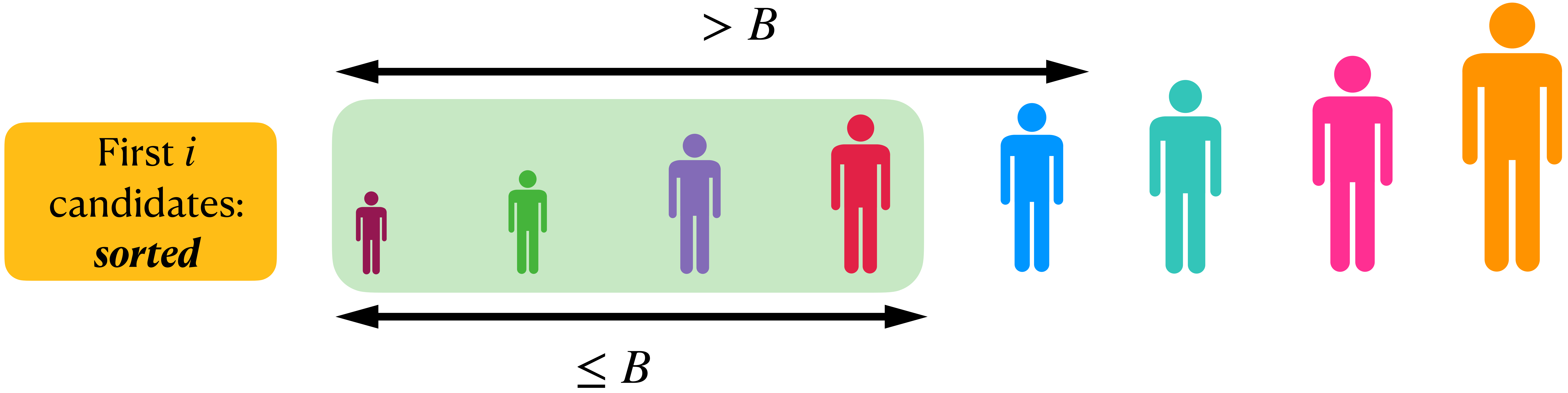
Upper Bound: Adversarial Order



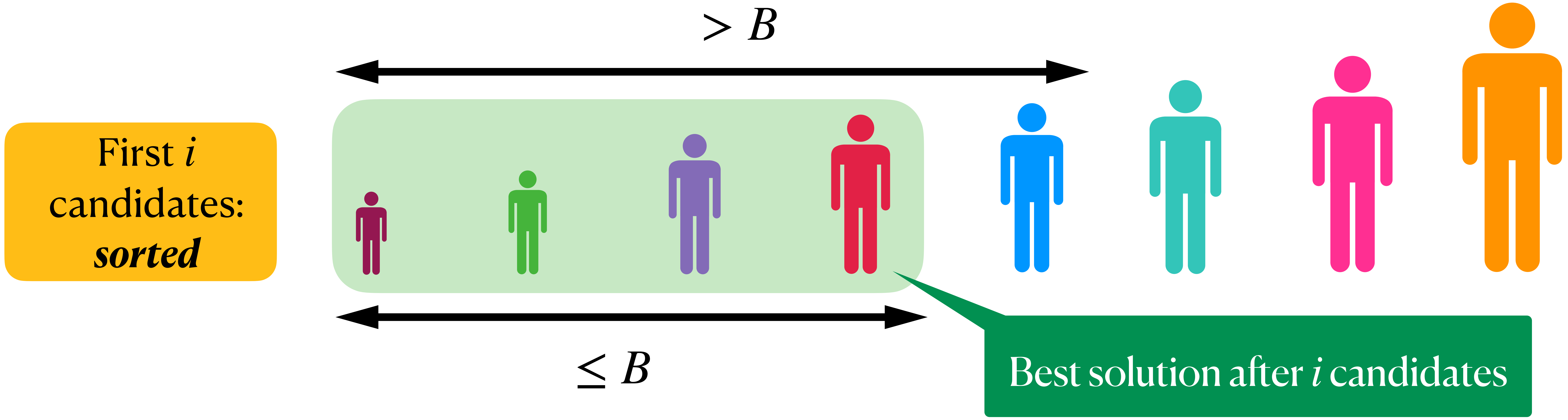
Upper Bound: Adversarial Order



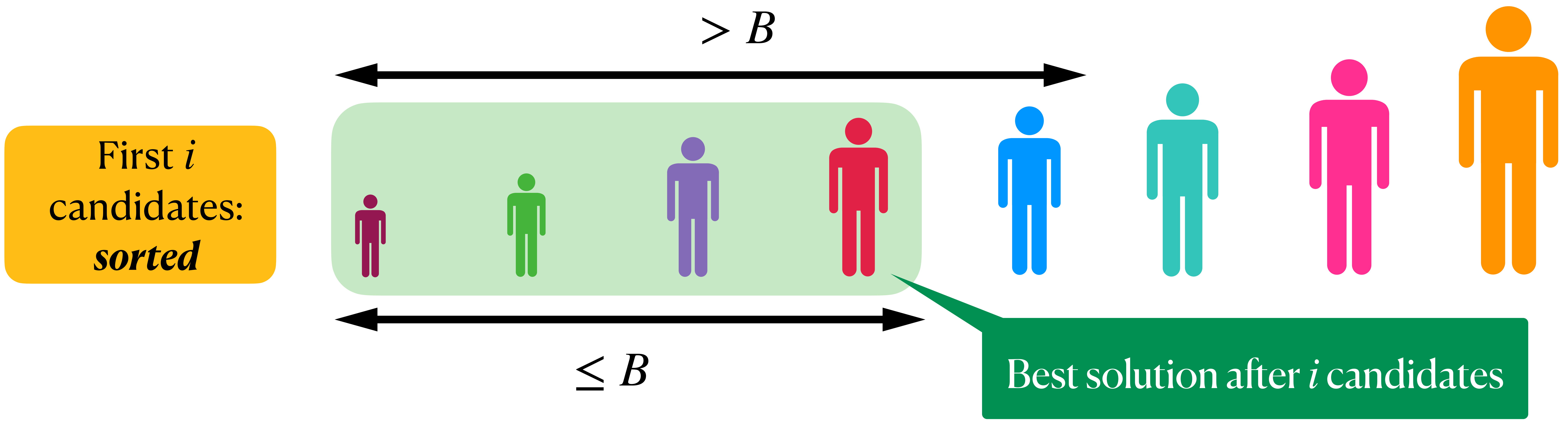
Upper Bound: Adversarial Order



Upper Bound: Adversarial Order

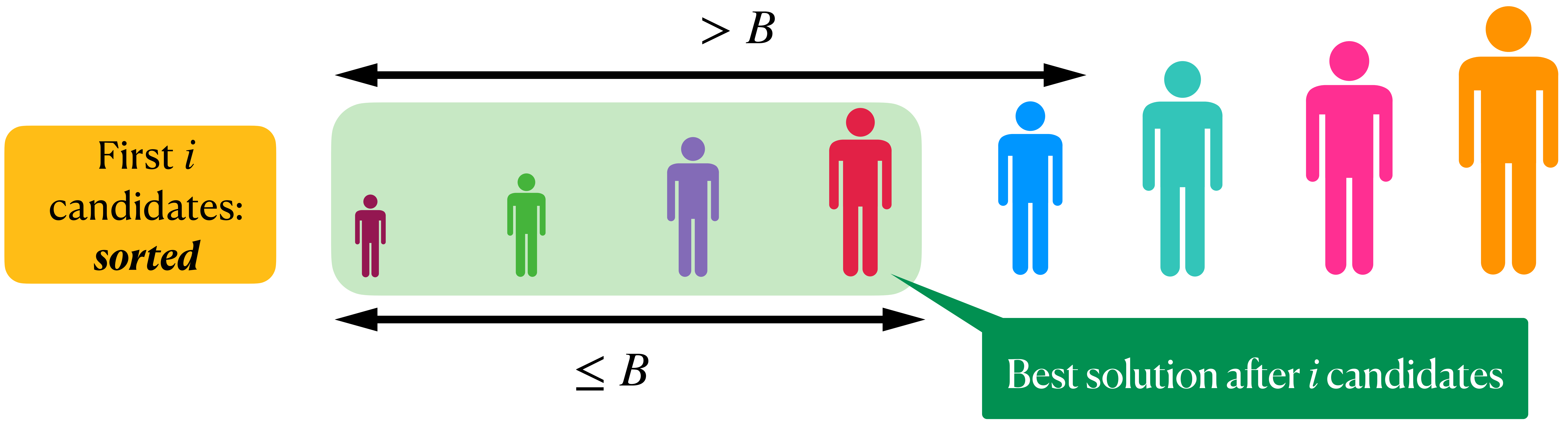


Upper Bound: Adversarial Order



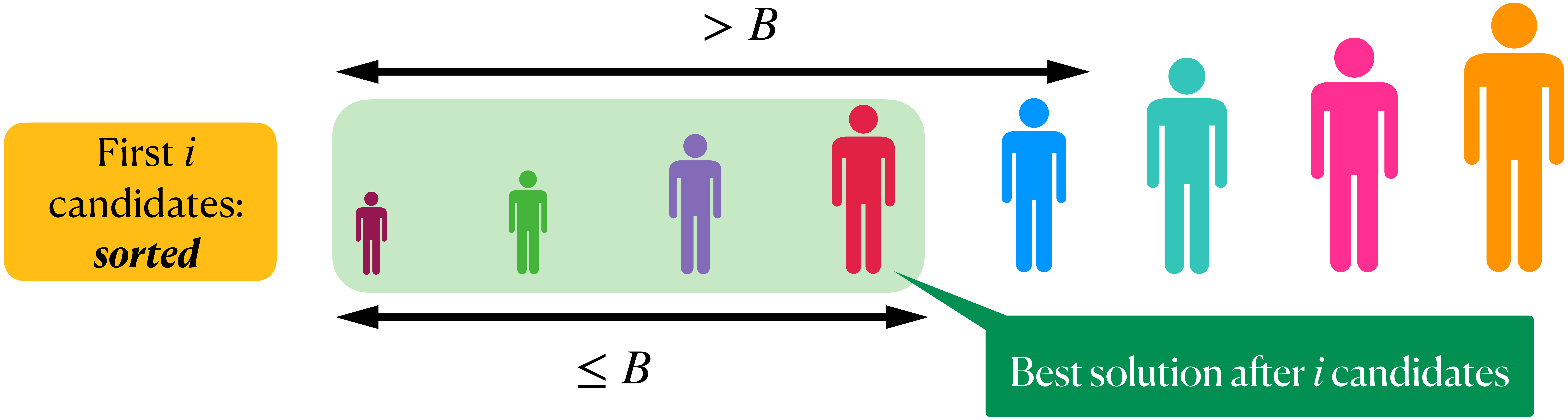
Simplified Cautious Algorithm: Candidate i hired iff candidate i in best solution after i candidates.

Upper Bound: Adversarial Order



Simplified Cautious Algorithm: Candidate i hired iff candidate i in best solution after i candidates.

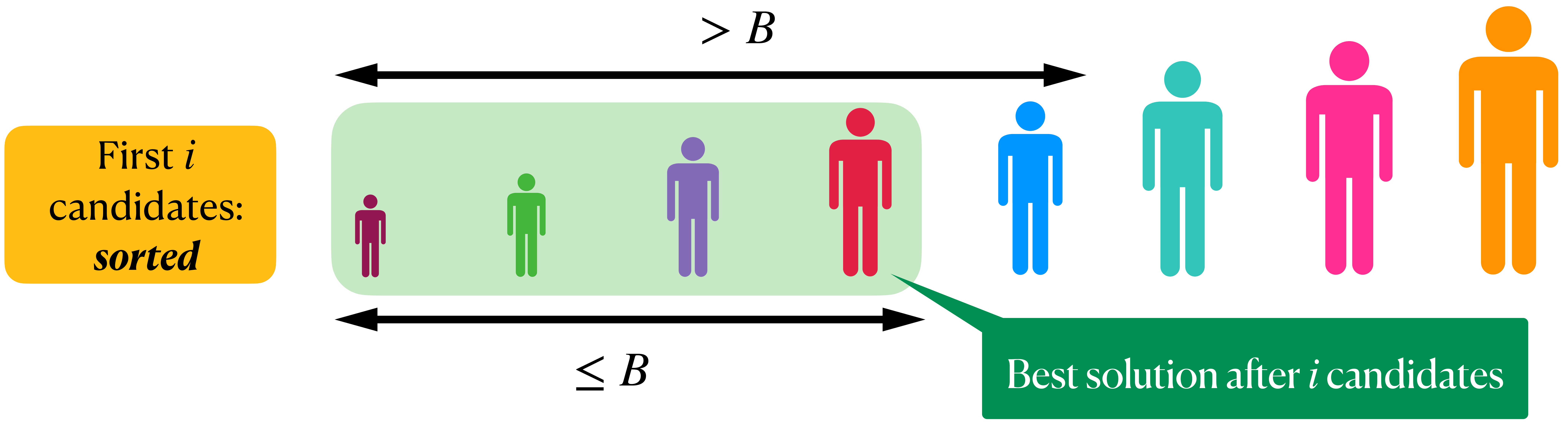
Upper Bound: Adversarial Order



~~**Simplified Cautious Algorithm:** Candidate i hired iff candidate i in best solution after i candidates.~~

Unbounded competitiveness!

Upper Bound: Adversarial Order



Simplified Cautious Algorithm: Candidate i hired iff candidate i in best solution after i candidates.

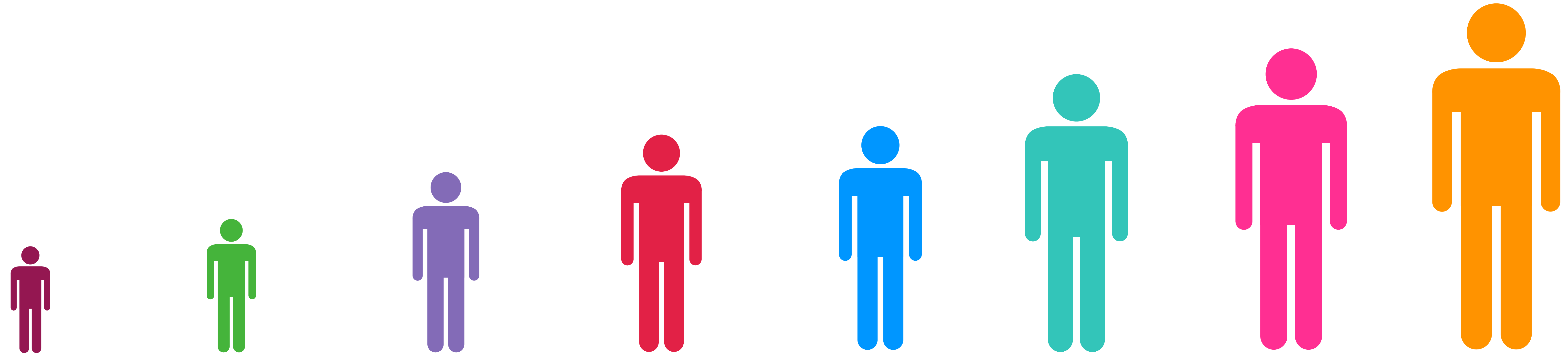
Unbounded competitiveness!

Why?

Treats candidates of similar cost differently.

Upper Bound: Adversarial Order

First i candidates: ***sorted***



~~**Simplified Cautious Algorithm:** Candidate i hired iff candidate i in best solution after i candidates.~~

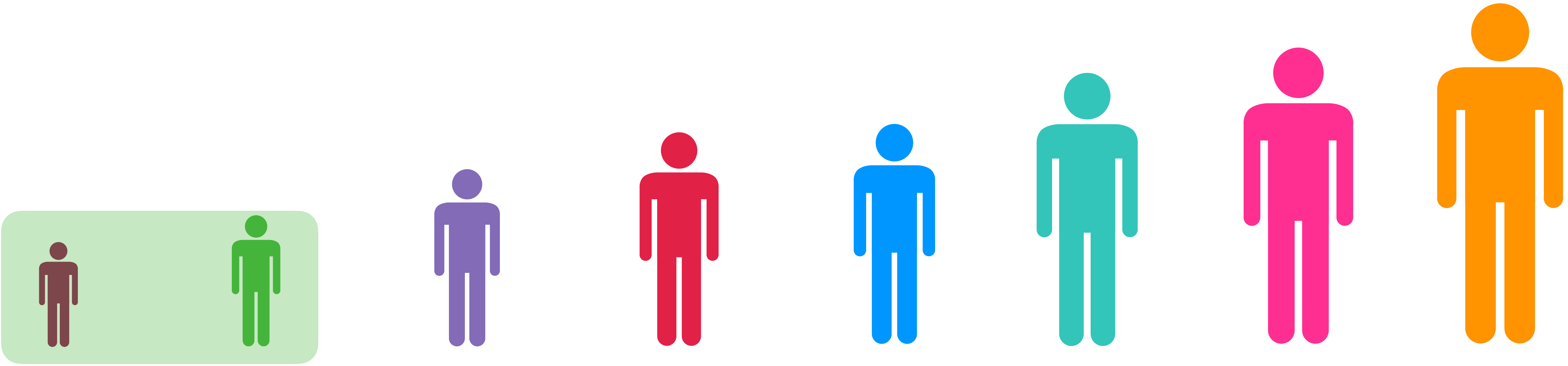
Unbounded competitiveness!

Why?

Treats candidates of similar cost differently.

Upper Bound: Adversarial Order

First i candidates: ***sorted***



Simplified Cautious Algorithm: Candidate i hired iff candidate i in best solution after i candidates.

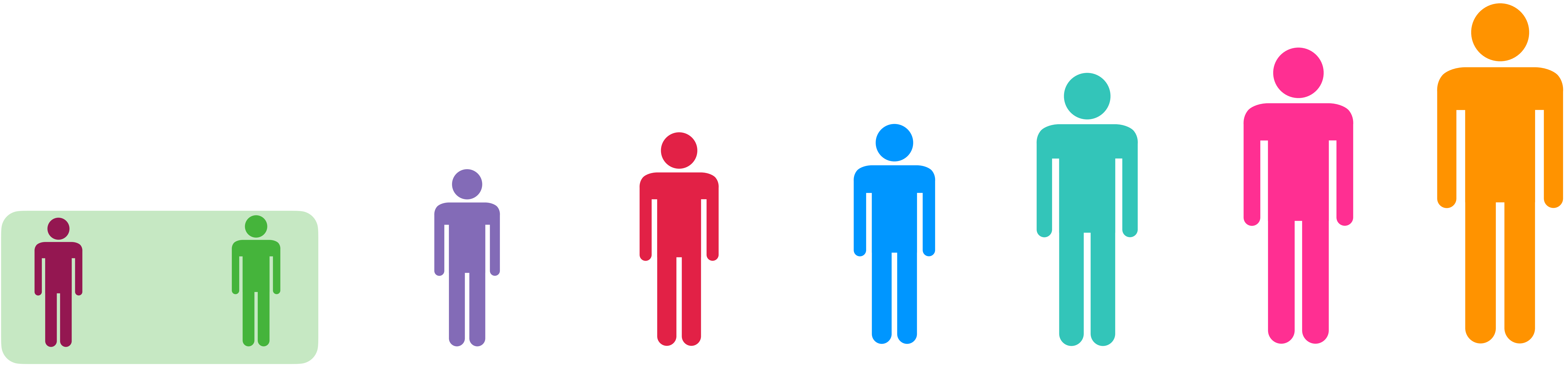
Unbounded competitiveness!

Why?

Treats candidates of similar cost differently.

Upper Bound: Adversarial Order

First i candidates: ***sorted***



Round up small costs

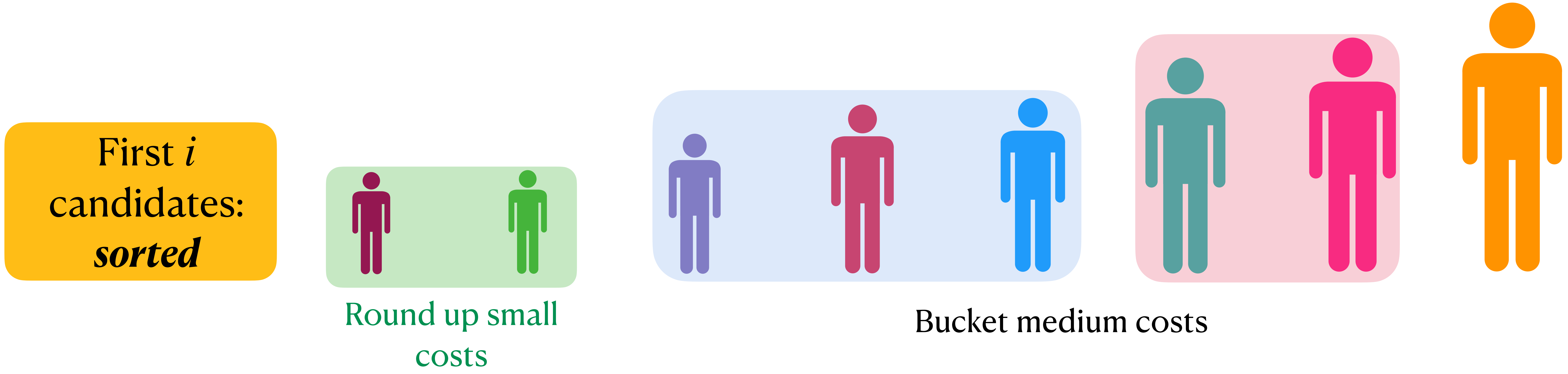
Simplified Cautious Algorithm: Candidate i hired iff candidate i in best solution after i candidates.

Unbounded competitiveness!

<u>Why?</u>

Treats candidates of similar cost differently.

Upper Bound: Adversarial Order



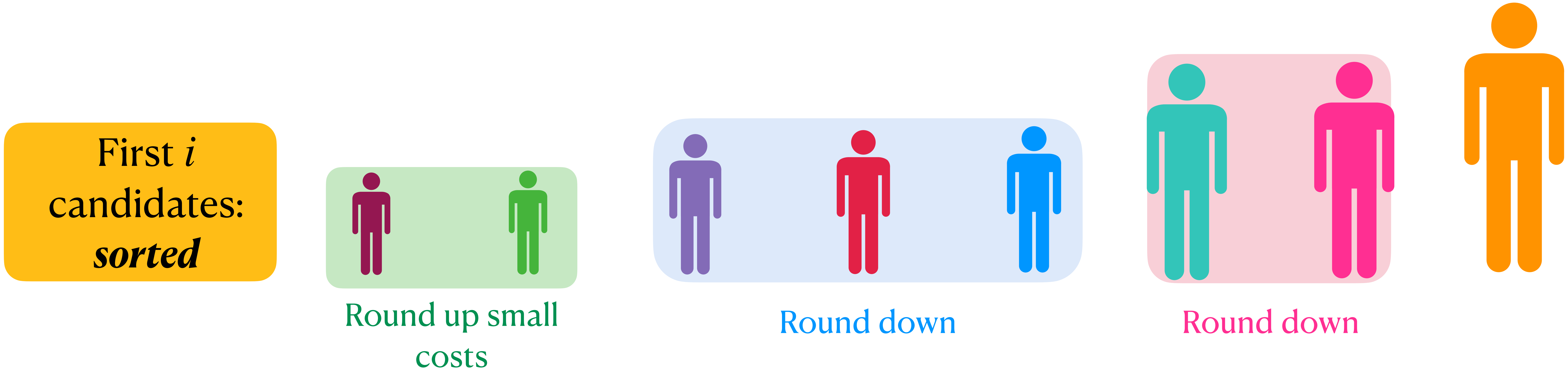
~~**Simplified Cautious Algorithm:** Candidate i hired iff candidate i in best solution after i candidates.~~

Unbounded competitiveness!

Why?

Treats candidates of similar cost differently.

Upper Bound: Adversarial Order



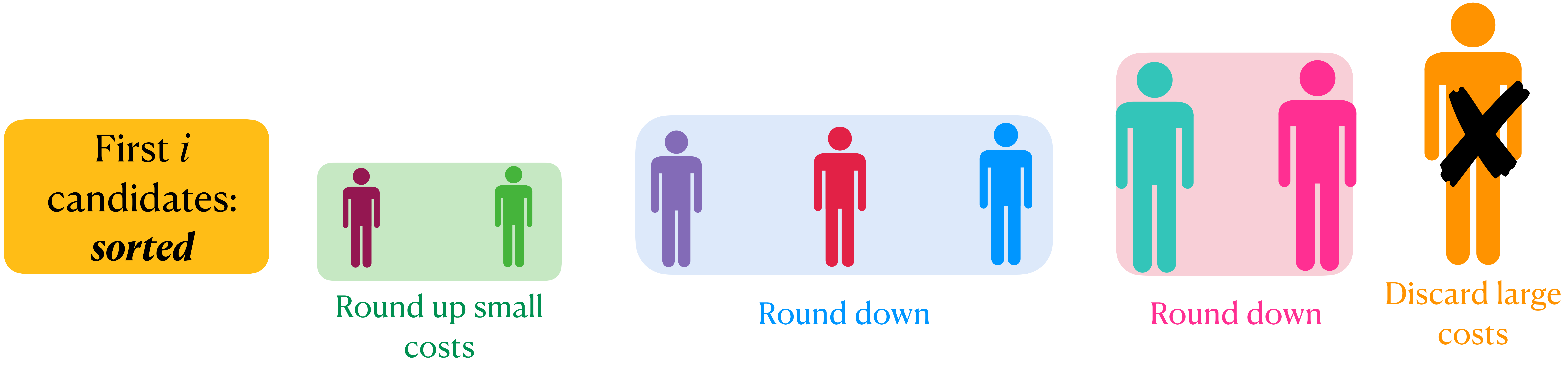
Simplified Cautious Algorithm: Candidate i hired iff candidate i in best solution after i candidates.

Unbounded competitiveness!

Why?

Treats candidates of similar cost differently.

Upper Bound: Adversarial Order



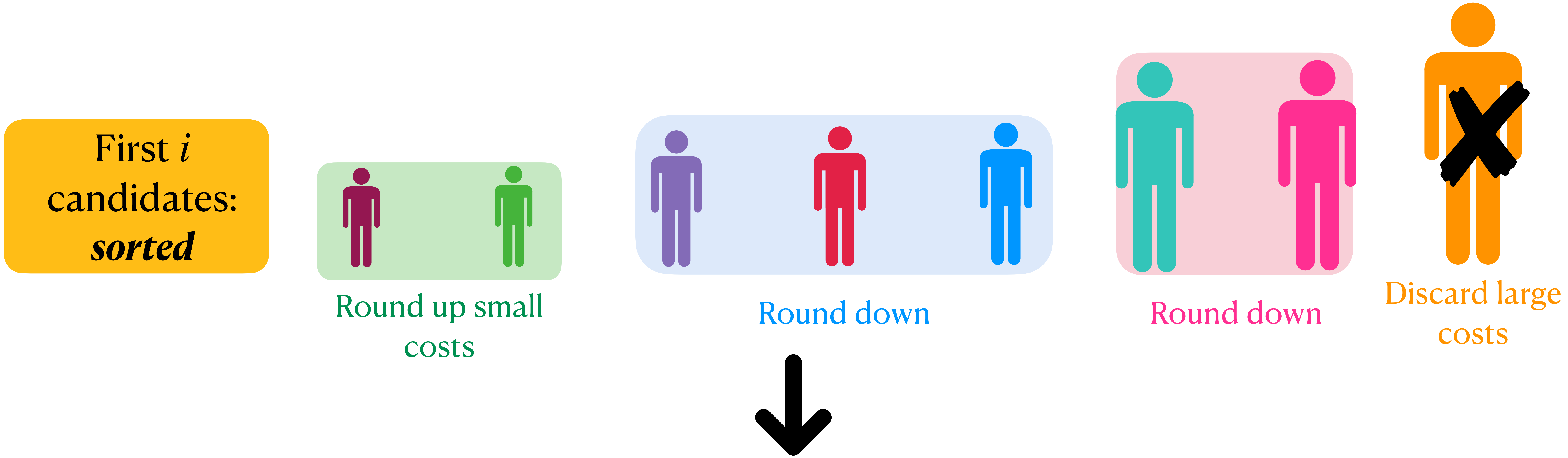
~~**Simplified Cautious Algorithm:** Candidate i hired iff candidate i in best solution after i candidates.~~

Unbounded competitiveness!

<u>Why?</u>

Treats candidates of similar cost differently.

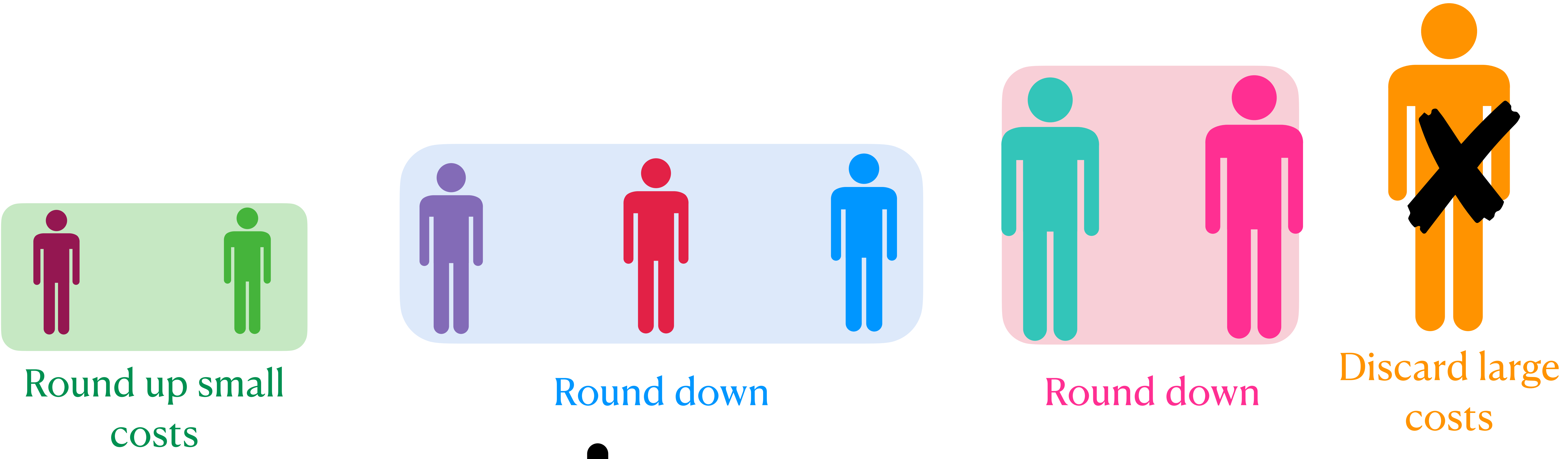
Upper Bound: Adversarial Order



General **Cautious Algorithm:** Candidate i hired iff candidate i in **"best"** solution after i candidates.

Upper Bound: Adversarial Order

First i candidates: ***sorted***



General Cautious Algorithm: Candidate i hired iff candidate i in **“best”** solution after i candidates.

$O(\log k)$**-competitive algorithm**

Future Directions

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems
- Prophet inequality problems
- Secretary with advice
  - Prediction on secretary quality
  - Alternatives to random order: e.g., sample of secretaries

Future Directions

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems
- Prophet inequality problems
- Secretary with advice
  - Prediction on secretary quality
  - Alternatives to random order: e.g., sample of secretaries

Future Directions

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems
- Prophet inequality problems
- Secretary with advice
  - Prediction on secretary quality
  - Alternatives to random order: e.g., sample of secretaries

Minimization variants

Future Directions

- Most classic setting: n candidates, maximize expected aggregate value
  - Assume n **is known** AND **random order arrivals**
  - Optimal $1/e$-competitive threshold algorithm for $k = 1$
  - $\left(1 - O\left(1/\sqrt{k}\right)\right)$- competitive algorithm for general k
- Matroid secretary (above: k-uniform matroid)
- Knapsack secretary problems
- Prophet inequality problems
- Secretary with advice
  - Prediction on secretary quality
  - Alternatives to random order: e.g., sample of secretaries

Minimization variants

Other learning-augmented approaches

Thank You