University of BRISTOL

Advanced Algorithms – COMS31900

Approximation algorithms part four

Asymptotic Polynomial Time Approximation Schemes

Raphaël Clifford

Slides by Benjamin Sach

Approximation Algorithms Recap

A polynomial time algorithm A is an α-approximation for problem P if,

it always outputs a solution s with

$$\frac{\mathrm{Opt}}{\alpha} \leqslant s \leqslant \mathrm{Opt} \text{ (for a maximisation problem)}$$

$$\mathrm{Opt} \leqslant s \leqslant \alpha \cdot \mathrm{Opt} \text{ (for a minimisation problem)}$$

A poly-time approximation scheme (PTAS) is a family of algorithms:

For any constant $\epsilon > 0$, A_ϵ is a $(1+\epsilon)$-approximation for P

A PTAS is allowed to have A_ϵ which takes $O(n^{1/\epsilon})$ time

- which is polynomial in n (for any constant ϵ)

It is a (fully) FPTAS if A_ϵ takes time polynomial in both n and $1/\epsilon$ i.e. $O((n/\epsilon)^c)$

Approximation Algorithms Recap

A polynomial time algorithm A is an α-approximation for problem P if,

it always outputs a solution s with

$$\frac{\mathrm{Opt}}{\alpha} \leqslant s \leqslant \mathrm{Opt} \text{ (for a maximisation problem)}$$

$$\mathrm{Opt} \leqslant s \leqslant \alpha \cdot \mathrm{Opt} \text{ (for a minimisation problem)}$$

We have seen various c-approximations with constant c

Last lecture we saw an FPTAS for SUBSETSUM

A poly-time approximation scheme (PTAS) is a family of algorithms:

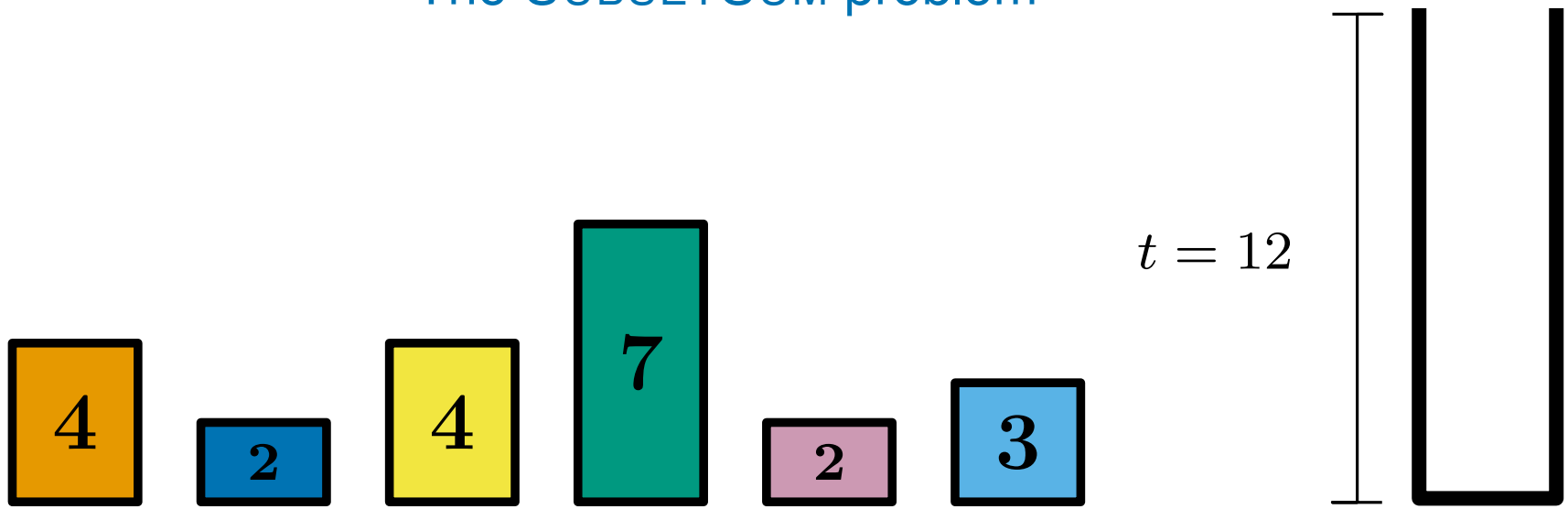
For any constant $\epsilon > 0$, A_ϵ is a $(1+\epsilon)$-approximation for P

A PTAS is allowed to have A_ϵ which takes $O(n^{1/\epsilon})$ time

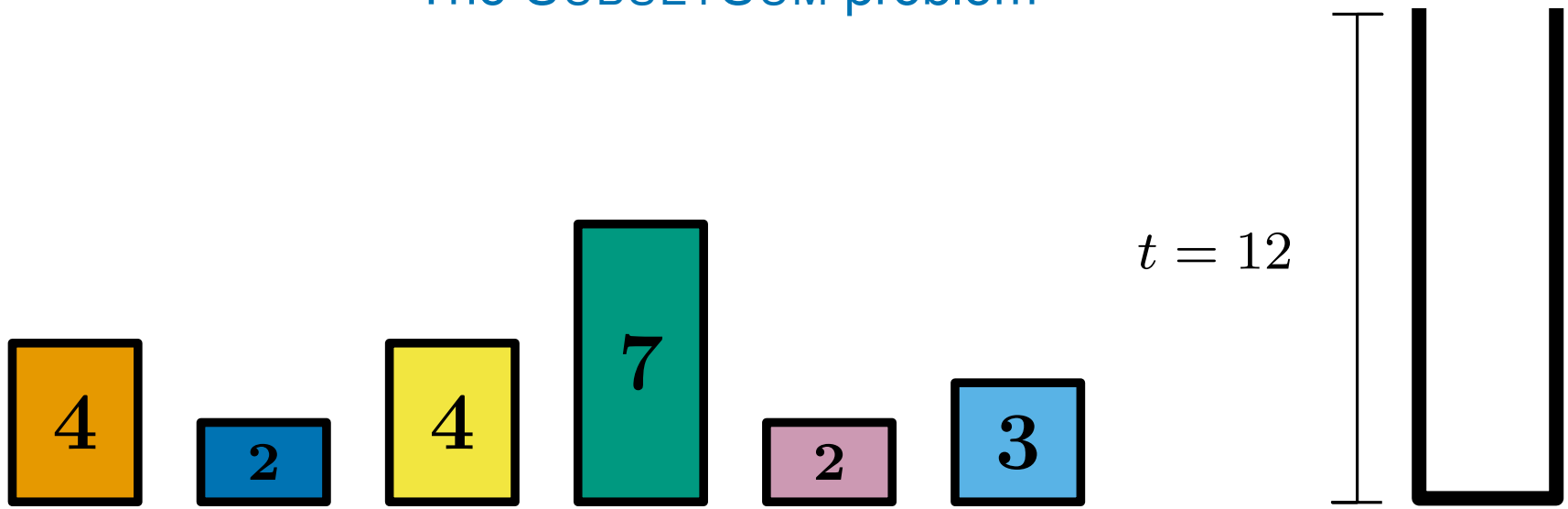
- which is polynomial in n (for any constant ϵ)

It is a (fully) FPTAS if A_ϵ takes time polynomial in both n and $1/\epsilon$ i.e. $O((n/\epsilon)^c)$

The SUBSETSUM problem



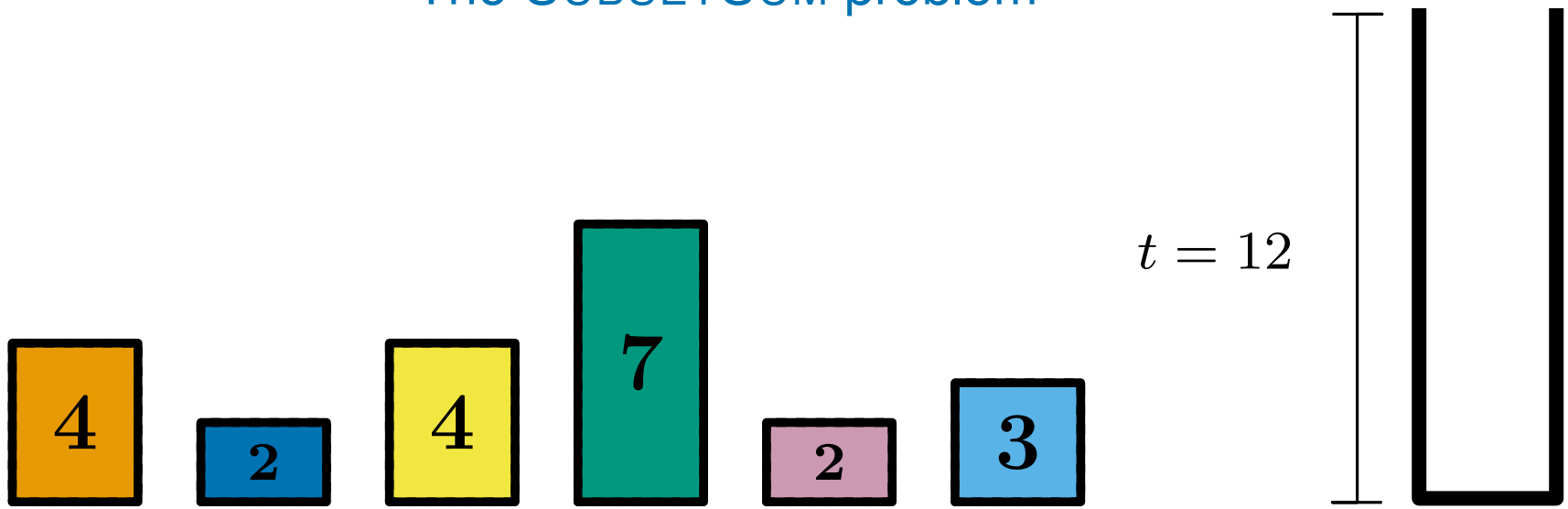
The SUBSETSUM problem



- Let S be a (multi) set of integers and t be a positive integer

here $S = \{4, 2, 4, 7, 2, 3\}$ *and* $t = 12$

The SUBSETSUM problem

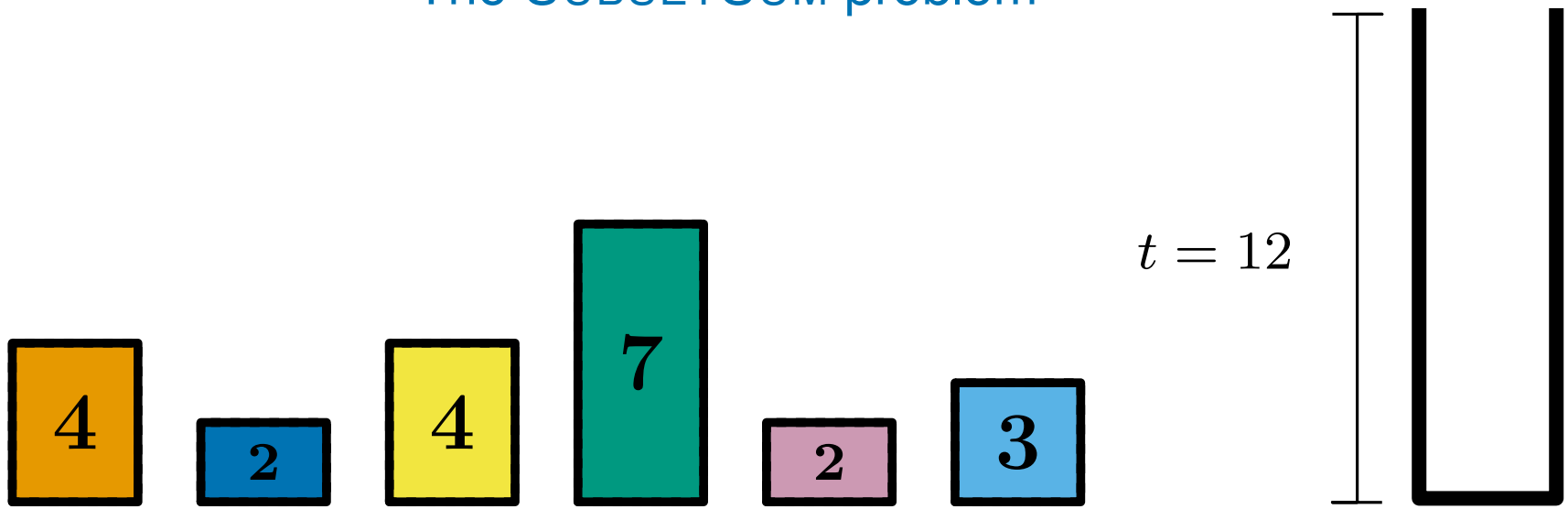


- Let S be a (multi) set of integers and t be a positive integer

here $S = \{4, 2, 4, 7, 2, 3\}$ *and* $t = 12$

Decision Problem Is there a subset, $S' \subseteq S$ with SIZE$(S') = t$?

The SUBSETSUM problem



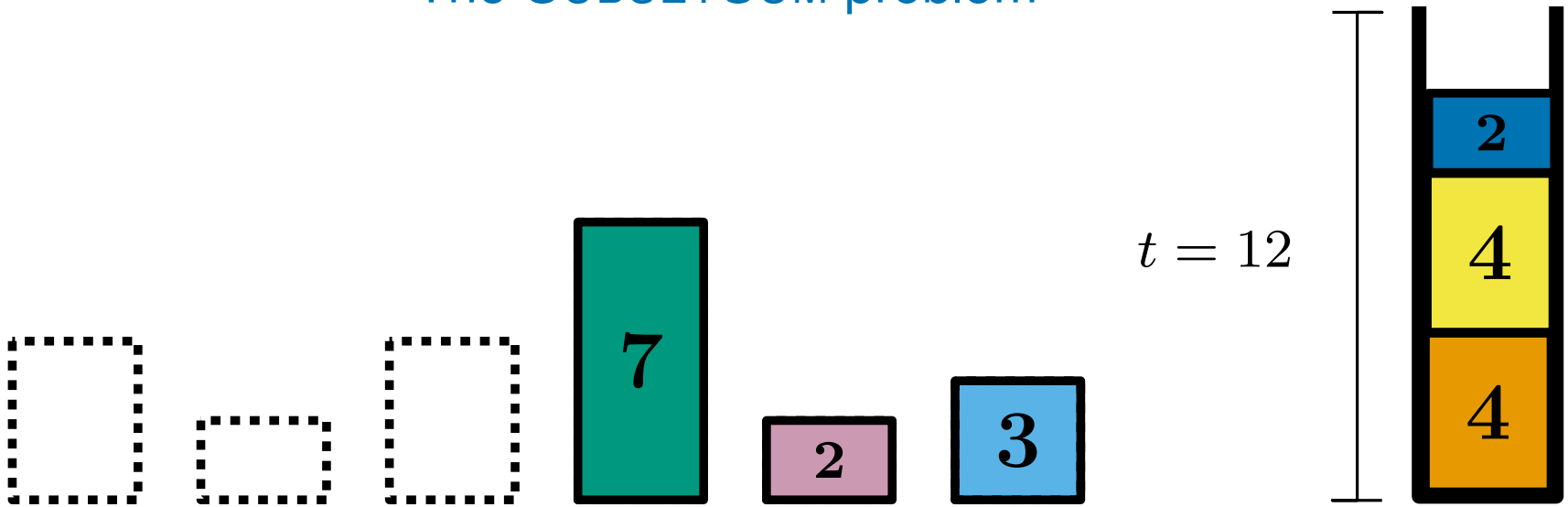
- Let S be a (multi) set of integers and t be a positive integer

here $S = \{4, 2, 4, 7, 2, 3\}$ *and* $t = 12$

Decision Problem Is there a subset, $S' \subseteq S$ with SIZE$(S') = t$?

where *SIZE*$(S') = \sum_{a \in S'} a$

The SUBSETSUM problem



- Let S be a (multi) set of integers and t be a positive integer

here $S = \{4, 2, 4, 7, 2, 3\}$ *and* $t = 12$

Decision Problem Is there a subset, $S' \subseteq S$ with SIZE$(S') = t$?

where *SIZE*$(S') = \sum_{a \in S'} a$

The SUBSETSUM problem

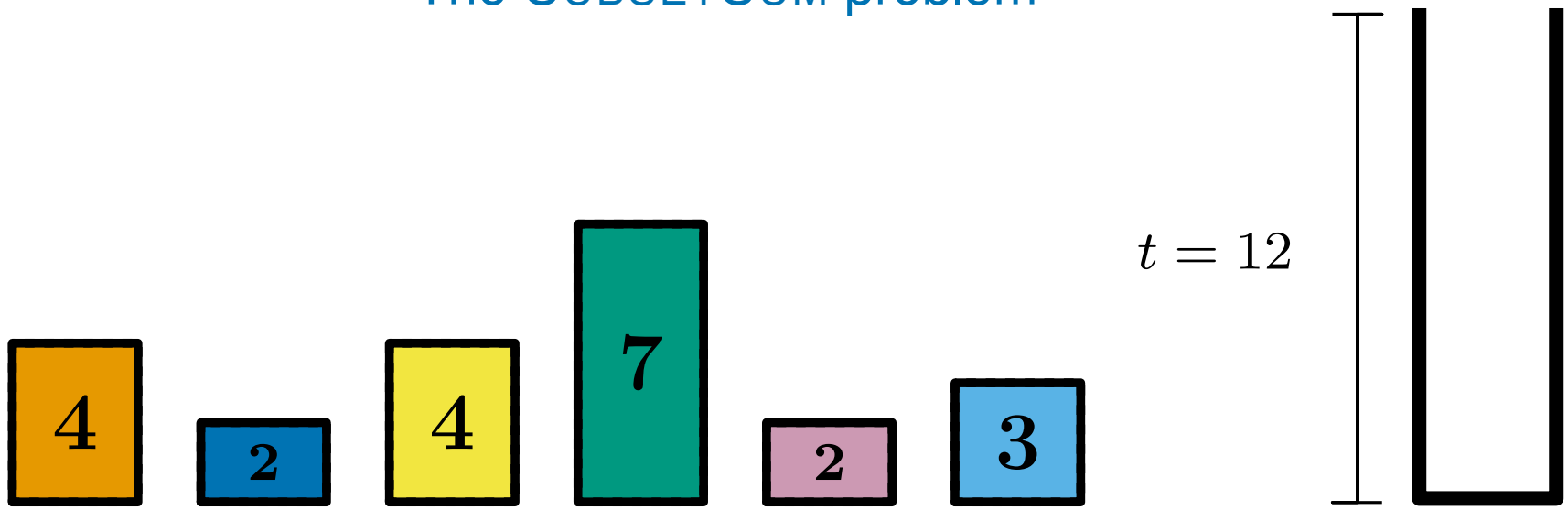
- Let S be a (multi) set of integers and t be a positive integer

here $S = \{4, 2, 4, 7, 2, 3\}$ *and* $t = 12$

Decision Problem Is there a subset, $S' \subseteq S$ with SIZE$(S') = t$?

where *SIZE*$(S') = \sum_{a \in S'} a$

The SUBSETSUM problem



- Let S be a (multi) set of integers and t be a positive integer

here $S = \{4, 2, 4, 7, 2, 3\}$ *and* $t = 12$

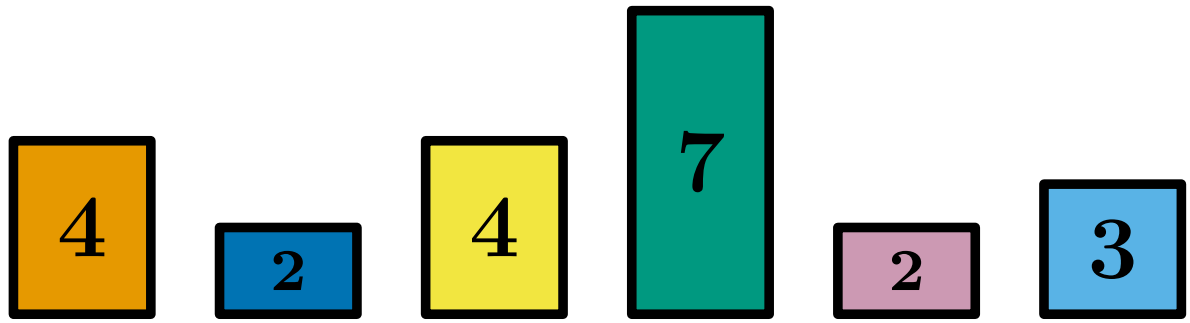
Decision Problem Is there a subset, $S' \subseteq S$ with SIZE$(S') = t$?

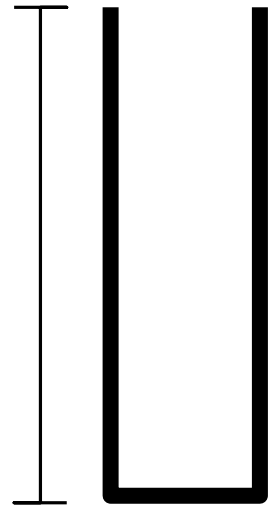
where SIZE$(S') = \sum_{a \in S'} a$

The decision version is NP-complete

(last lecture we discussed the NP-hard optimisation version)

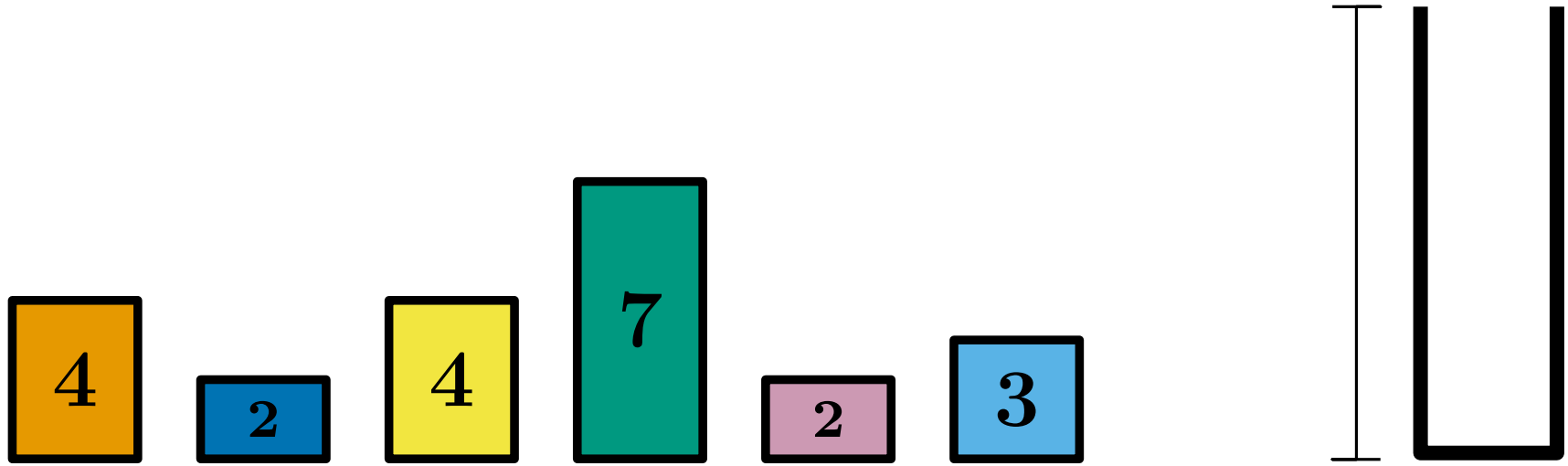
The PARTITION problem





The PARTITION problem is a special case of SUBSETSUM

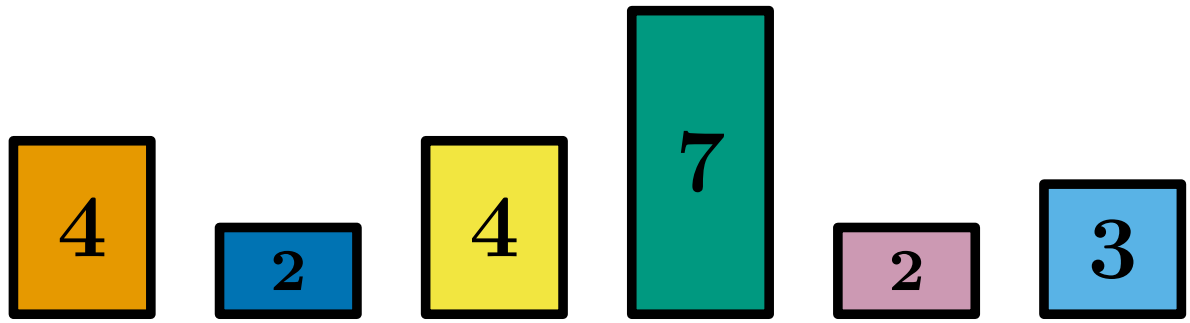
The PARTITION problem

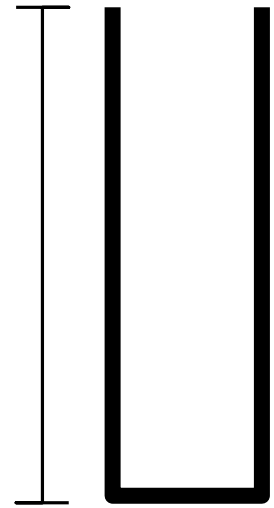


The PARTITION problem is a special case of SUBSETSUM

The input S is a multi-set of integers

The PARTITION problem



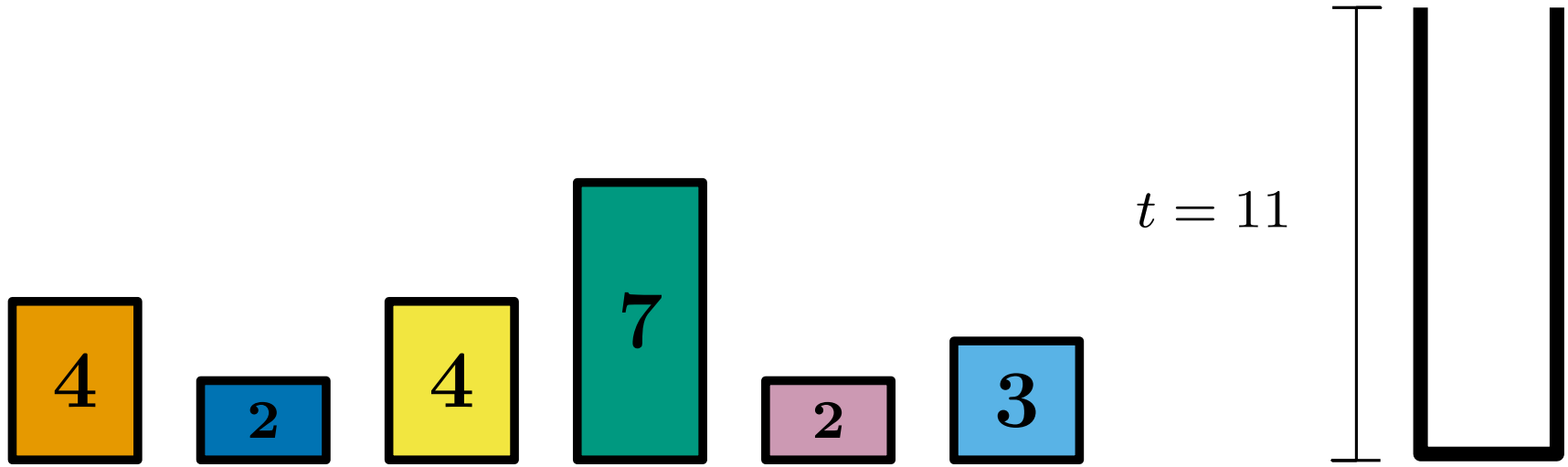


The PARTITION problem is a special case of SUBSETSUM

The input S is a multi-set of integers

but t is always half the sum of item sizes

The PARTITION problem

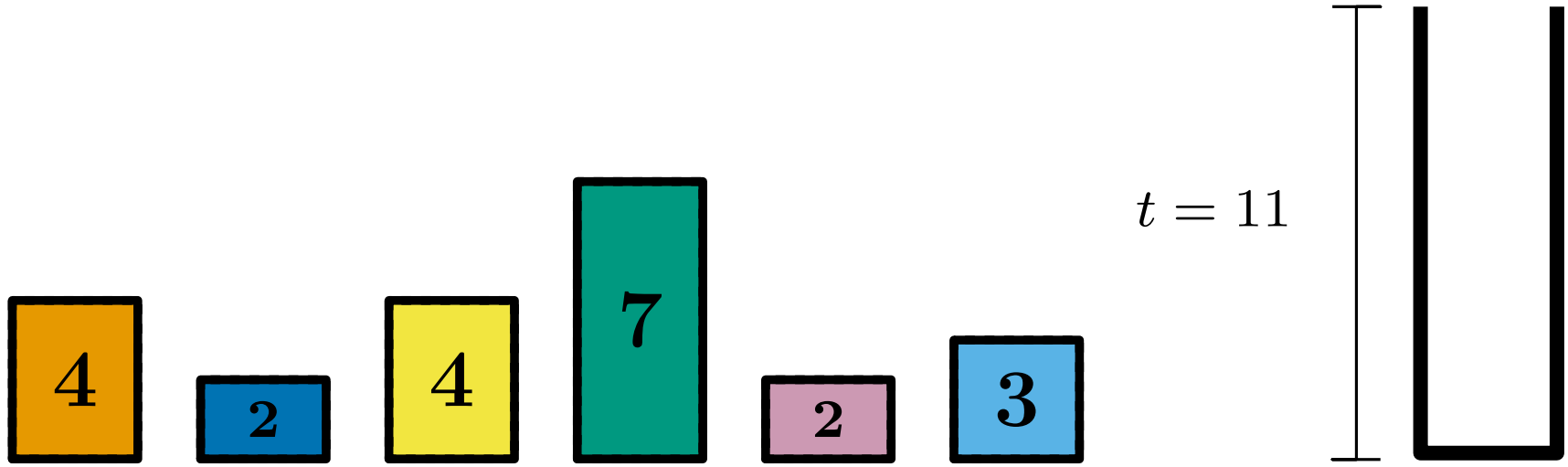


The PARTITION problem is a special case of SUBSETSUM

The input S is a multi-set of integers

but t is always half the sum of item sizes

The PARTITION problem



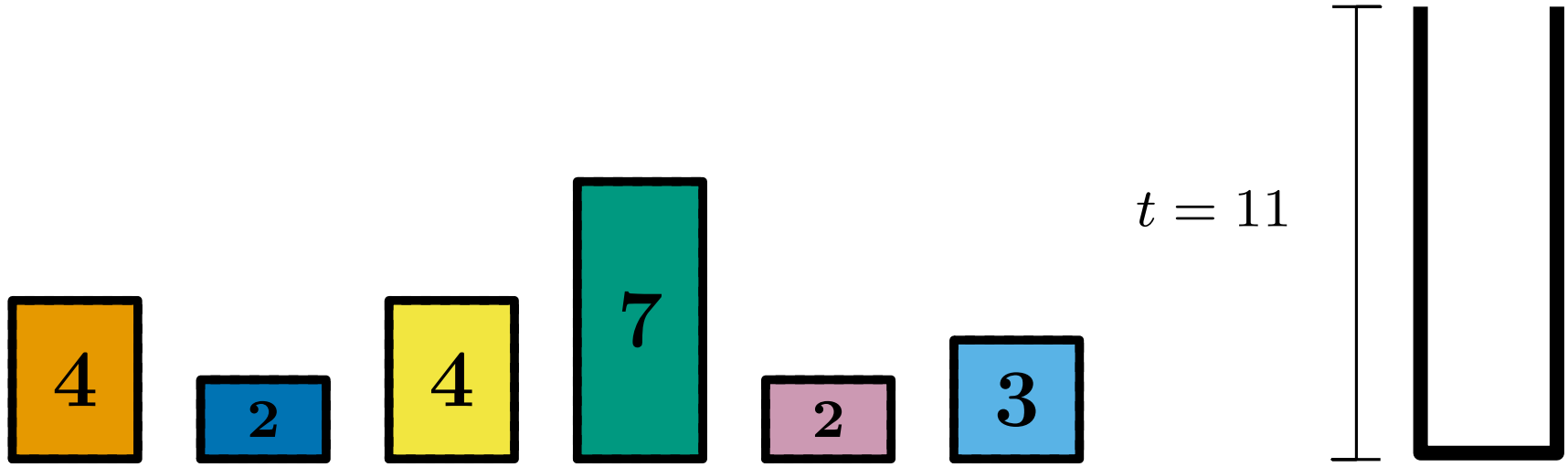
The PARTITION problem is a special case of SUBSETSUM

The input S is a multi-set of integers

but t is always half the sum of item sizes

$$t = \mathit{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

The PARTITION problem



The PARTITION problem is a special case of SUBSETSUM

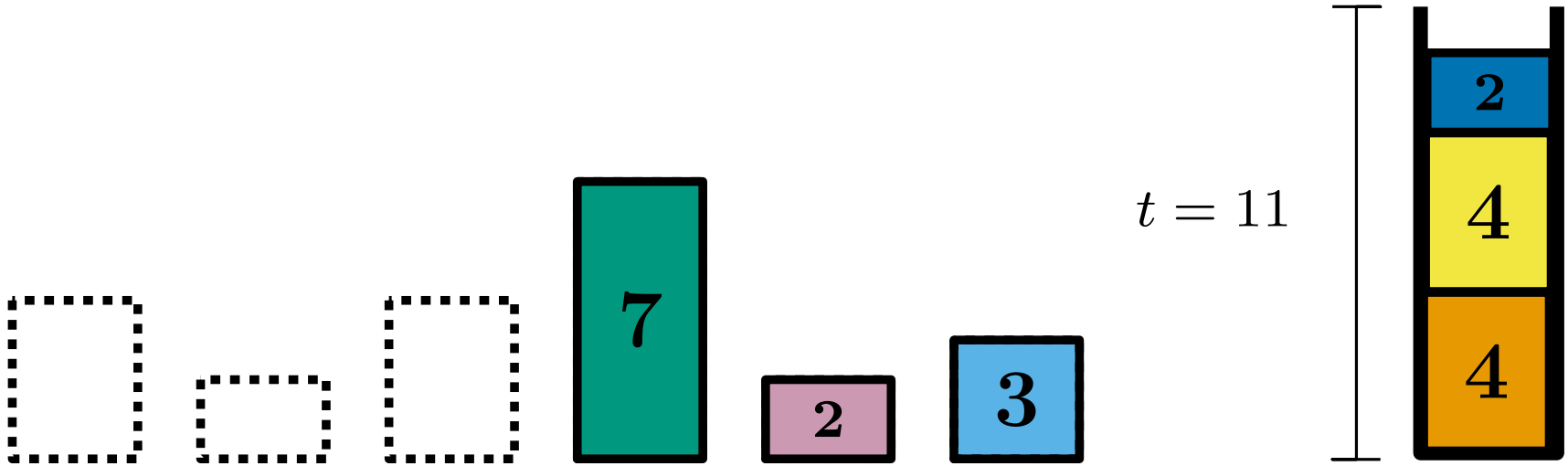
The input S is a multi-set of integers

but t is always half the sum of item sizes

$$t = \mathit{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

Decision Problem Is there a subset, $S' \subseteq S$ with $\mathrm{SIZE}(S') = \mathrm{SIZE}(S)/2$?

The PARTITION problem



The PARTITION problem is a special case of SUBSETSUM

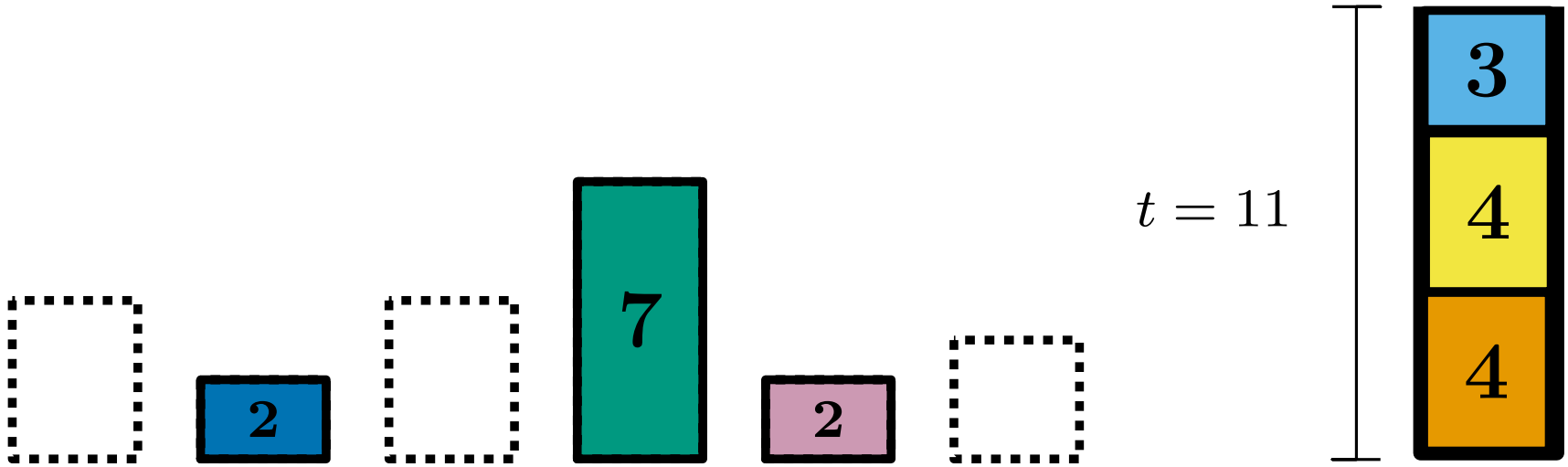
The input S is a multi-set of integers

but t is always half the sum of item sizes

$$t = \mathit{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

Decision Problem Is there a subset, $S' \subseteq S$ with $\mathrm{SIZE}(S') = \mathrm{SIZE}(S)/2$?

The PARTITION problem



The PARTITION problem is a special case of SUBSETSUM

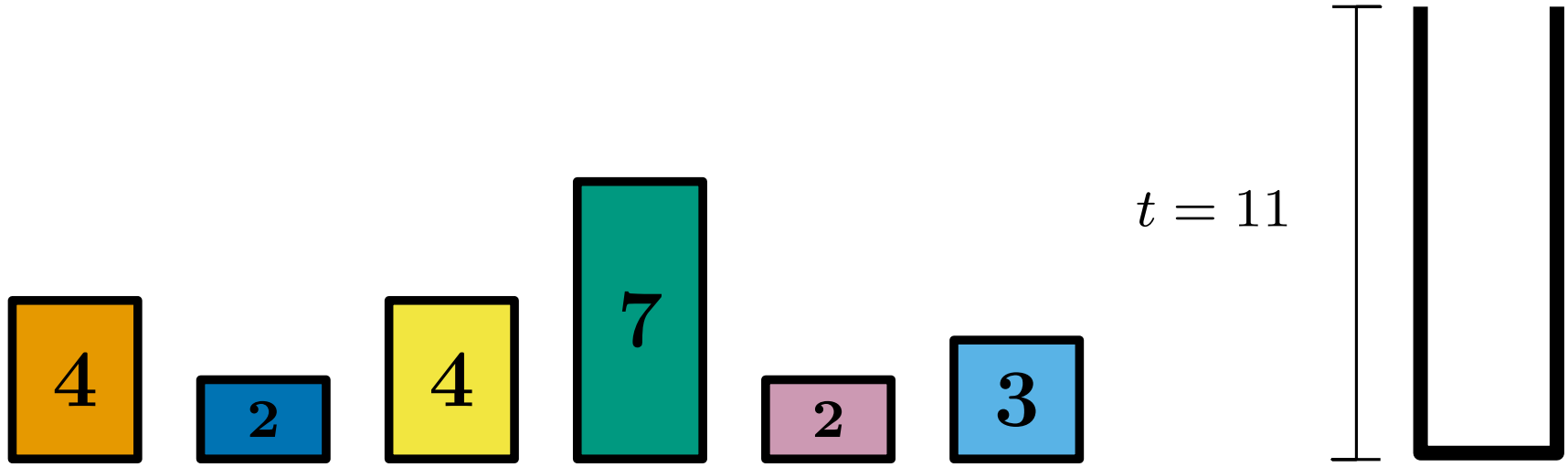
The input S is a multi-set of integers

but t is always half the sum of item sizes

$$t = \mathit{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

Decision Problem Is there a subset, $S' \subseteq S$ with $\mathrm{SIZE}(S') = \mathrm{SIZE}(S)/2$?

The PARTITION problem



The PARTITION problem is a special case of SUBSETSUM

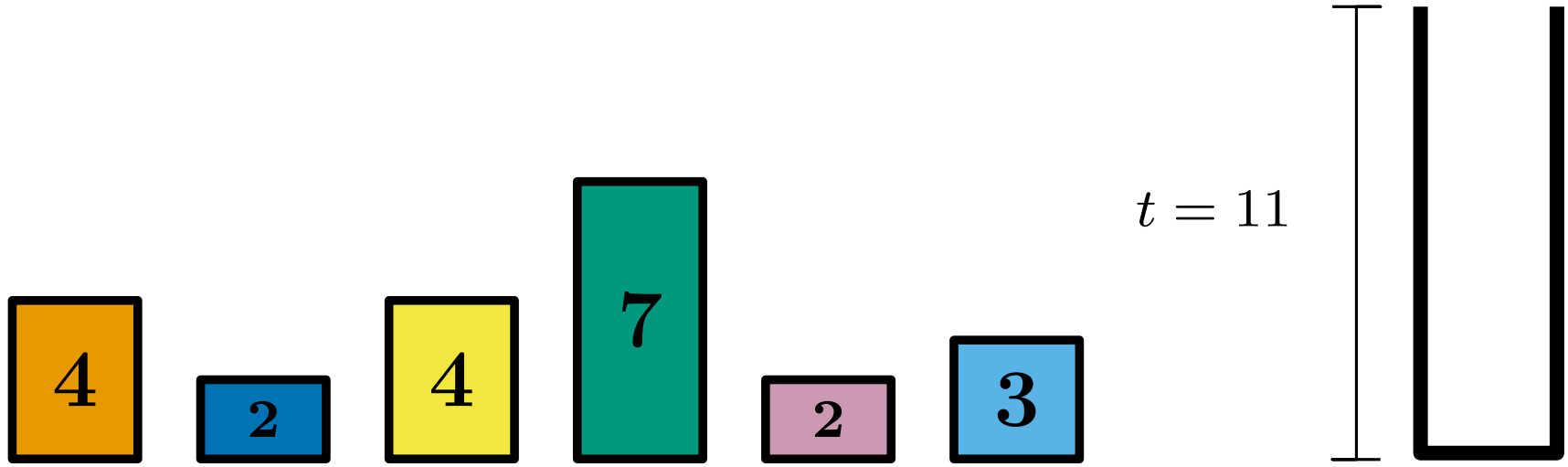
The input S is a multi-set of integers

but t is always half the sum of item sizes

$$t = \mathit{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

Decision Problem Is there a subset, $S' \subseteq S$ with $\mathrm{SIZE}(S') = \mathrm{SIZE}(S)/2$?

The PARTITION problem



The PARTITION problem is a special case of SUBSETSUM

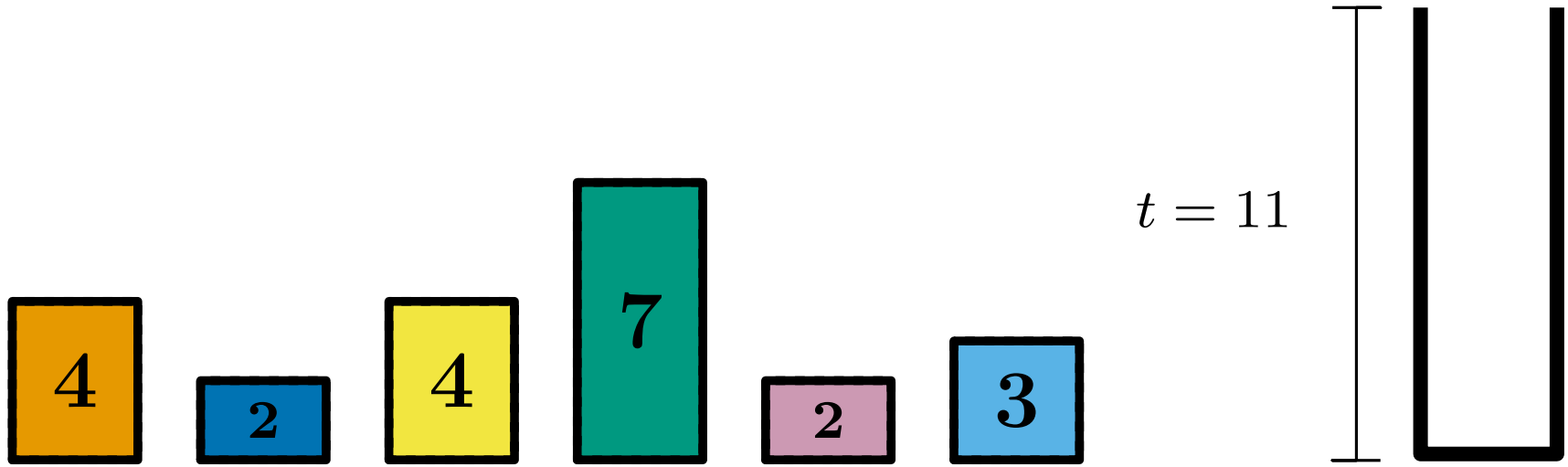
The input S is a multi-set of integers

but t is always half the sum of item sizes

$$t = \mathit{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

Decision Problem Is there a subset, $S' \subseteq S$ with $\mathrm{SIZE}(S') = \mathrm{SIZE}(S)/2$?

The PARTITION problem



The PARTITION problem is a special case of SUBSETSUM

The input S is a multi-set of integers

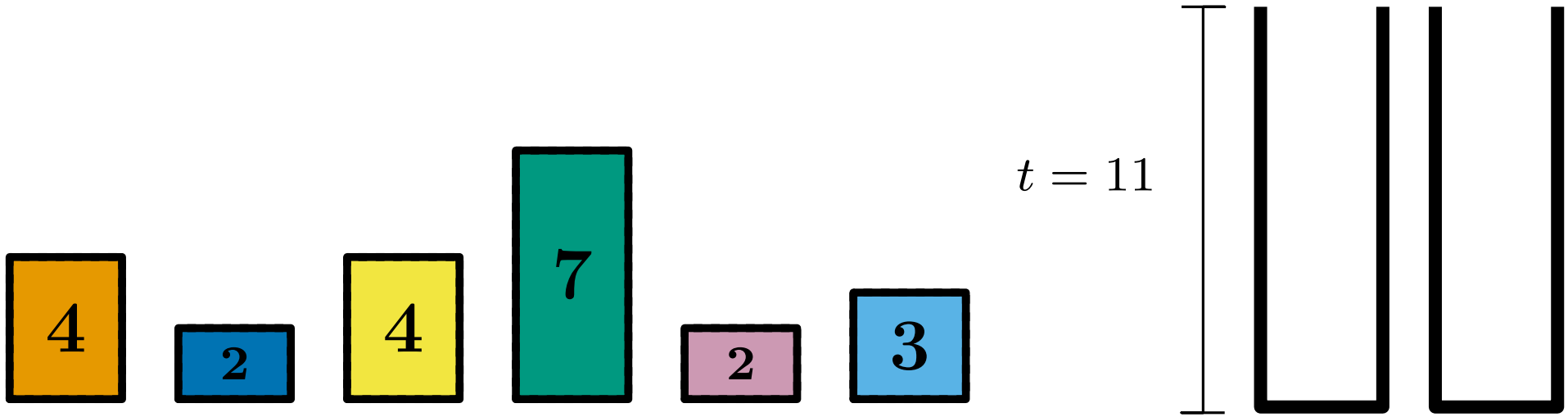
but t is always half the sum of item sizes

$$t = \mathit{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

Decision Problem Is there a subset, $S' \subseteq S$ with $\mathrm{SIZE}(S') = \mathrm{SIZE}(S)/2$?

Alternatively... Can we pack S into two bins of size $\mathrm{SIZE}(S)/2$?

The PARTITION problem



The PARTITION problem is a special case of SUBSETSUM

The input S is a multi-set of integers

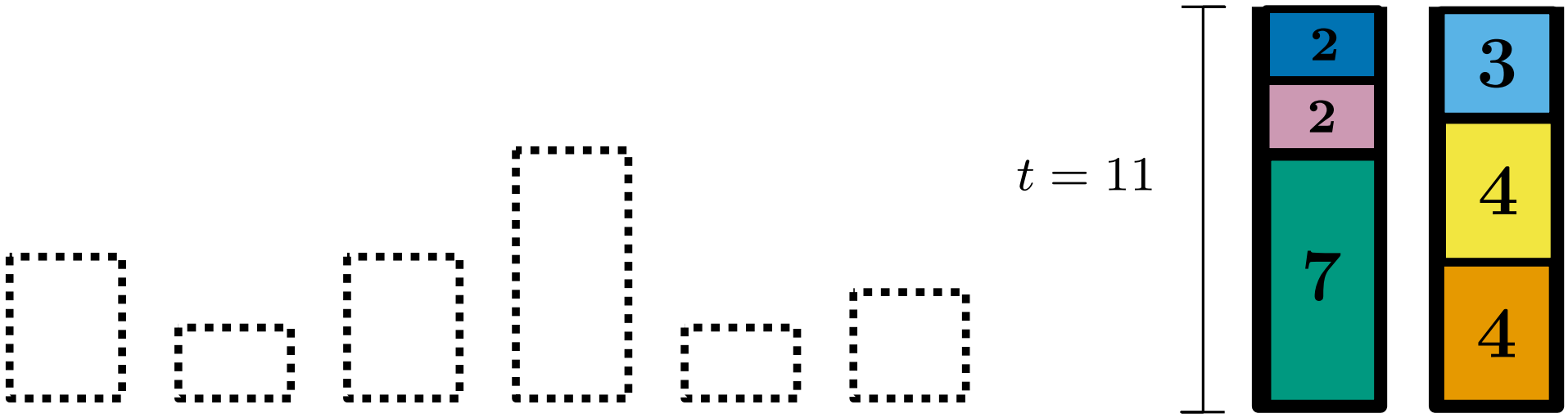
but t is always half the sum of item sizes

$$t = \mathit{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

Decision Problem Is there a subset, $S' \subseteq S$ with $\mathrm{SIZE}(S') = \mathrm{SIZE}(S)/2$?

Alternatively... Can we pack S into two bins of size $\mathrm{SIZE}(S)/2$?

The PARTITION problem



The PARTITION problem is a special case of SUBSETSUM

The input S is a multi-set of integers

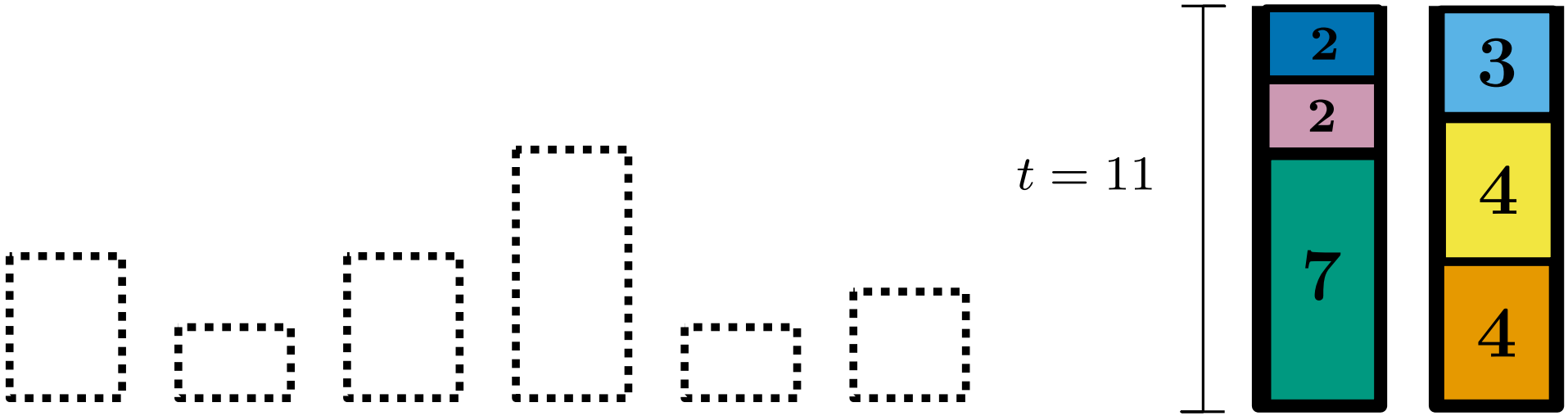
but t is always half the sum of item sizes

$$t = \mathit{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

Decision Problem Is there a subset, $S' \subseteq S$ with $\mathrm{SIZE}(S') = \mathrm{SIZE}(S)/2$?

Alternatively... Can we pack S into two bins of size $\mathrm{SIZE}(S)/2$?

The PARTITION problem



The PARTITION problem is a special case of SUBSETSUM

The input S is a multi-set of integers

but t is always half the sum of item sizes

$$t = \mathit{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

Decision Problem Is there a subset, $S' \subseteq S$ with $\mathrm{SIZE}(S') = \mathrm{SIZE}(S)/2$?

Alternatively... Can we pack S into two bins of size $\mathrm{SIZE}(S)/2$?

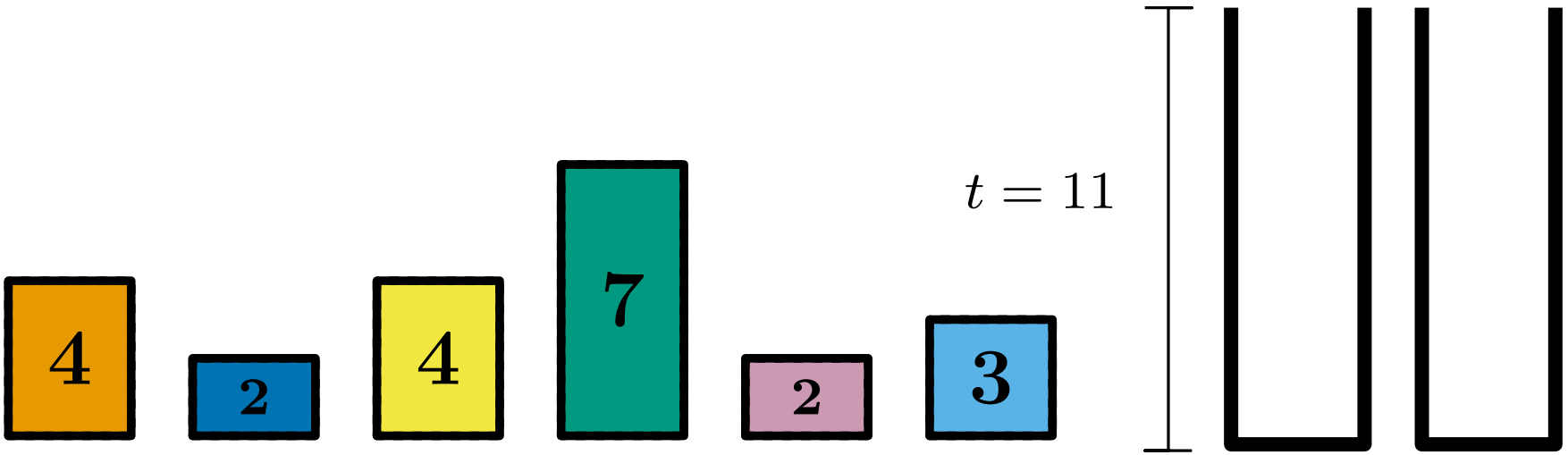
The PARTITION problem is also NP-complete

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING

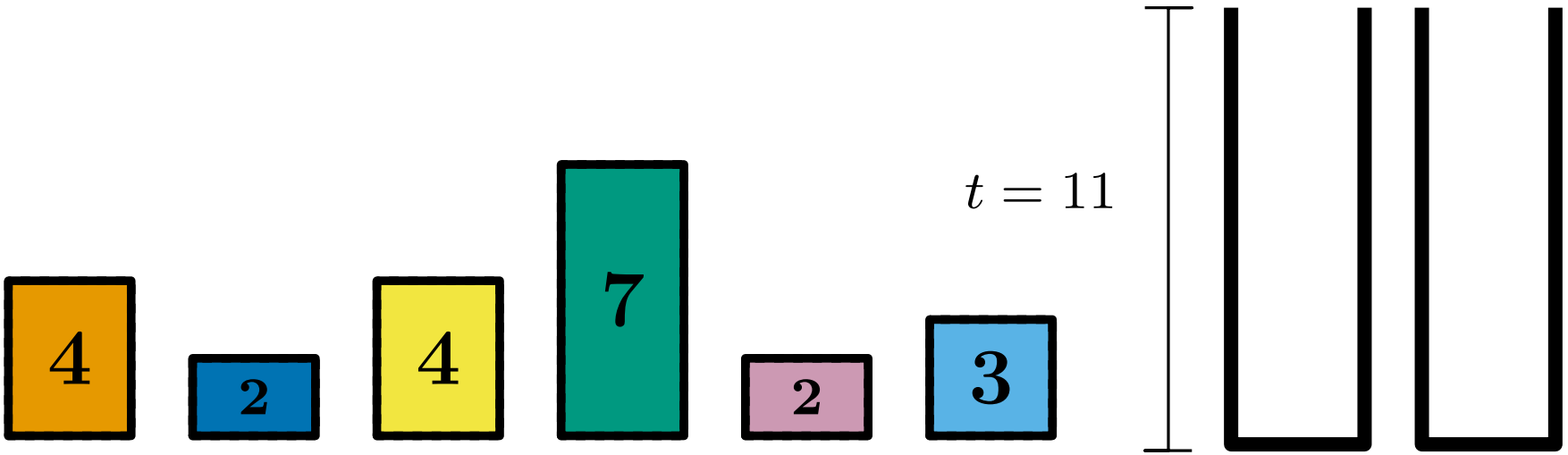
PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING

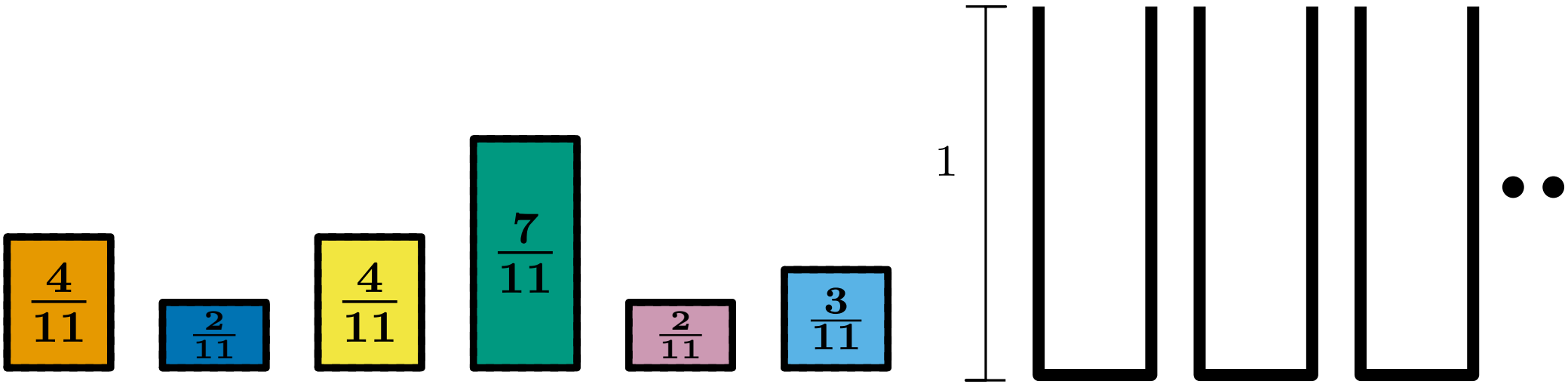


- Convert the PARTITION instance into a BINPACKING problem

  by dividing all item and bin sizes by $t = \text{SIZE}(S)/2$

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING

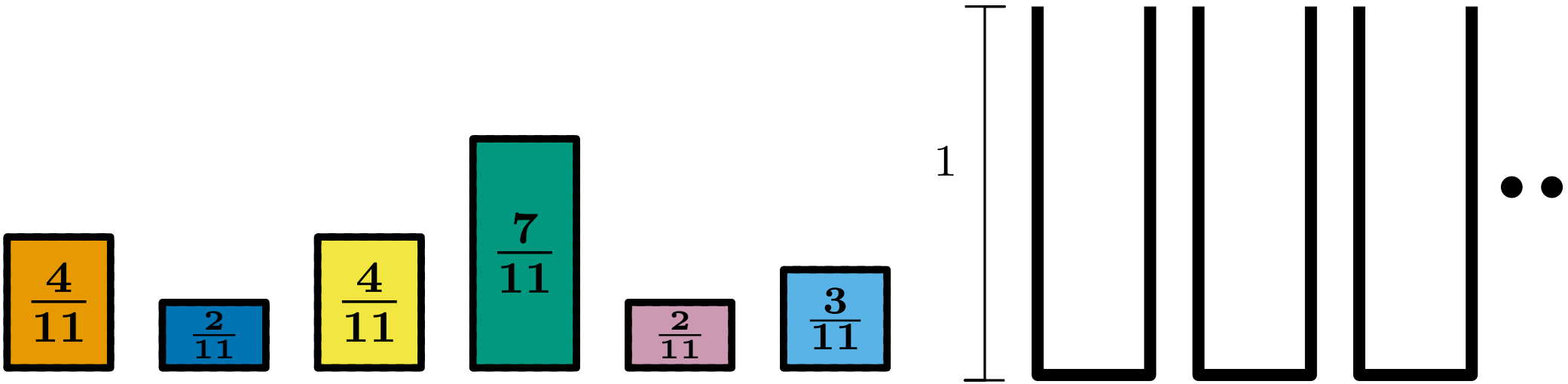


- Convert the PARTITION instance into a BINPACKING problem

  by dividing all item and bin sizes by $t = \text{SIZE}(S)/2$

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



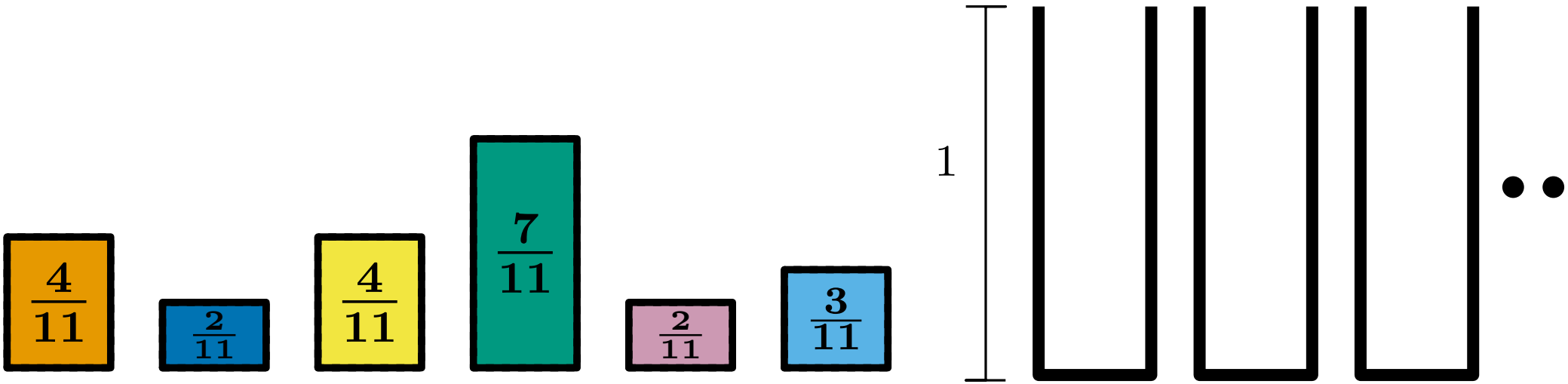
- Convert the PARTITION instance into a BINPACKING problem

  by dividing all item and bin sizes by $t = \text{SIZE}(S)/2$

- Now all items are have size at most 1 and the bins have size 1

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



- Convert the PARTITION instance into a BINPACKING problem

  by dividing all item and bin sizes by $t = \text{SIZE}(S)/2$

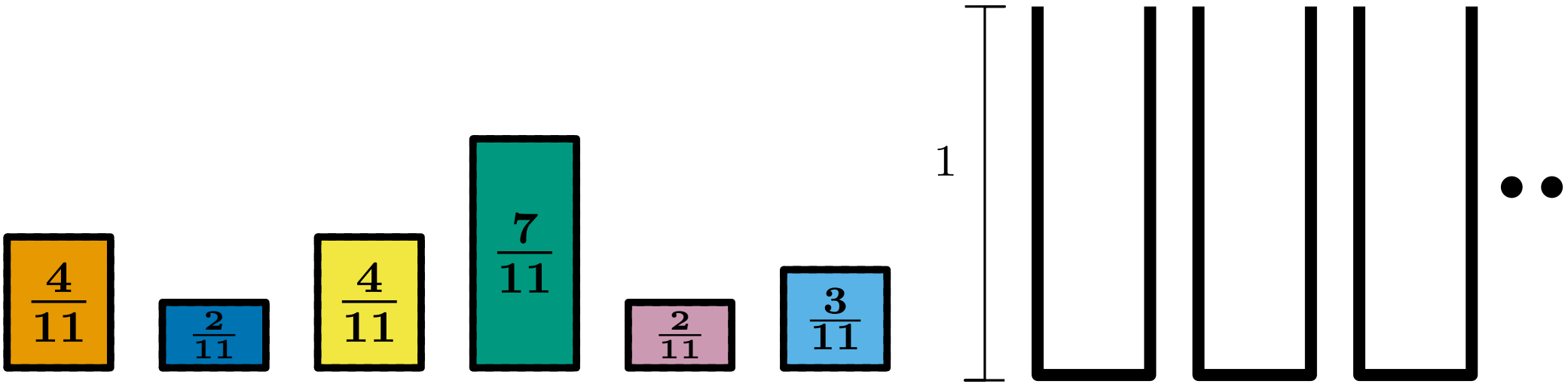
- Now all items are have size at most 1 and the bins have size 1

- The optimal number of bins Opt_b is 2 iff

  the answer to the PARTITION instance is 'yes'

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



- Convert the PARTITION instance into a BINPACKING problem

  by dividing all item and bin sizes by $t = \text{SIZE}(S)/2$

- Now all items are have size at most 1 and the bins have size 1

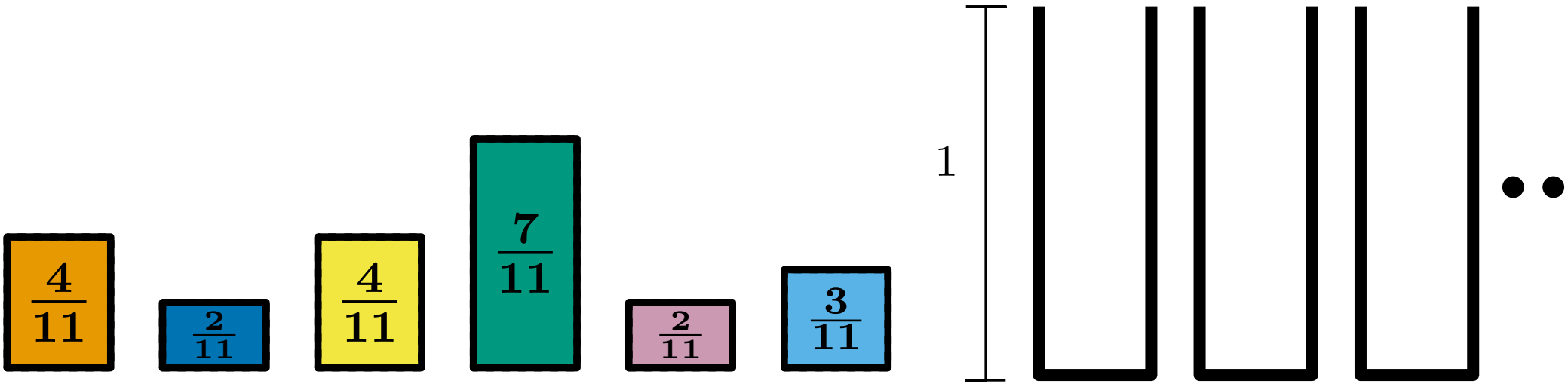
- The optimal number of bins Opt_b is 2 iff

  the answer to the PARTITION instance is 'yes'

- What does this tell us about approximating BINPACKING?

PARTITION and BINPACKING

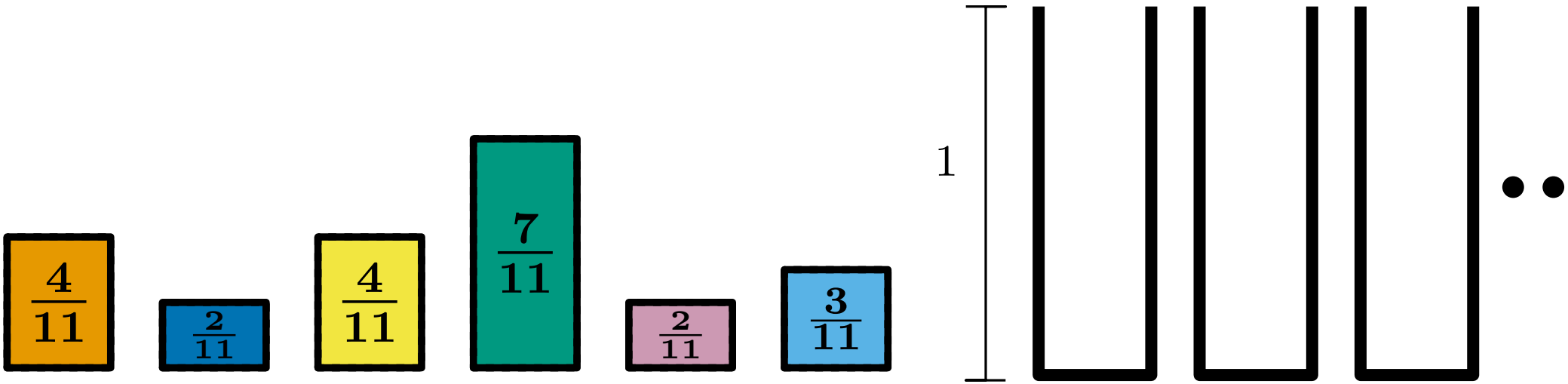
Key Idea Solve the PARTITION problem by approximating BINPACKING



- Assume A is an α-approximation for BINPACKING with $\alpha < 3/2$

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING

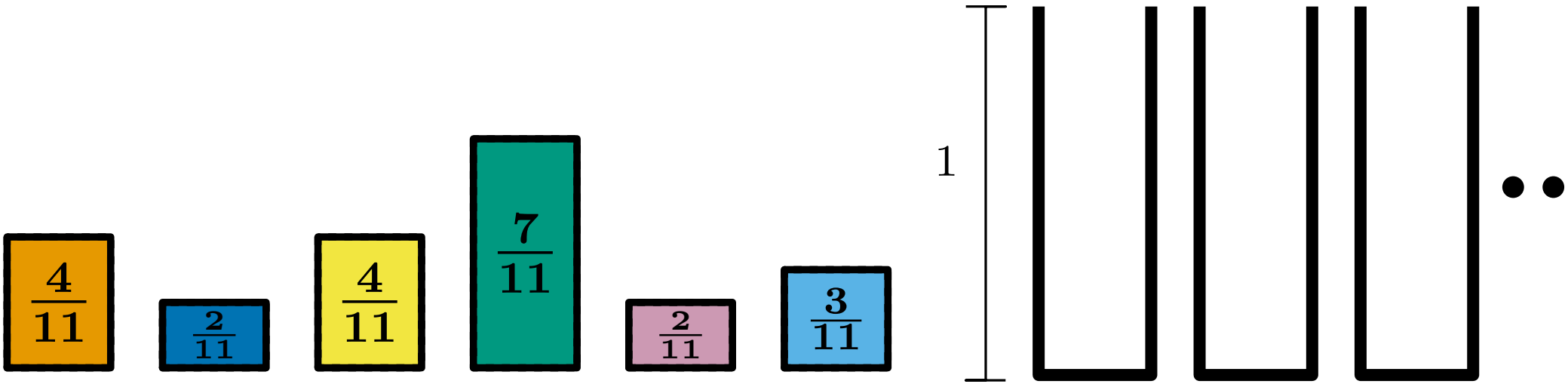


- Assume A is an α-approximation for BINPACKING with $\alpha < 3/2$

  A outputs a solution using s bins with $\mathrm{Opt}_b \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b$

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



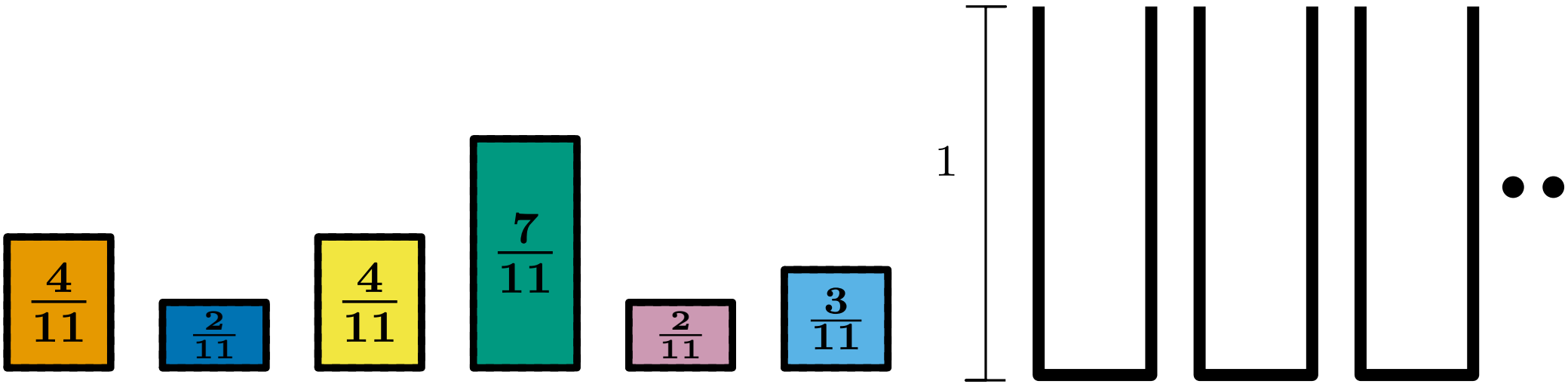
- Assume A is an α-approximation for BINPACKING with $\alpha < 3/2$

  A outputs a solution using s bins with $\mathrm{Opt}_b \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b$

- If $\mathrm{Opt}_b > 2$ then $s > 2$

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



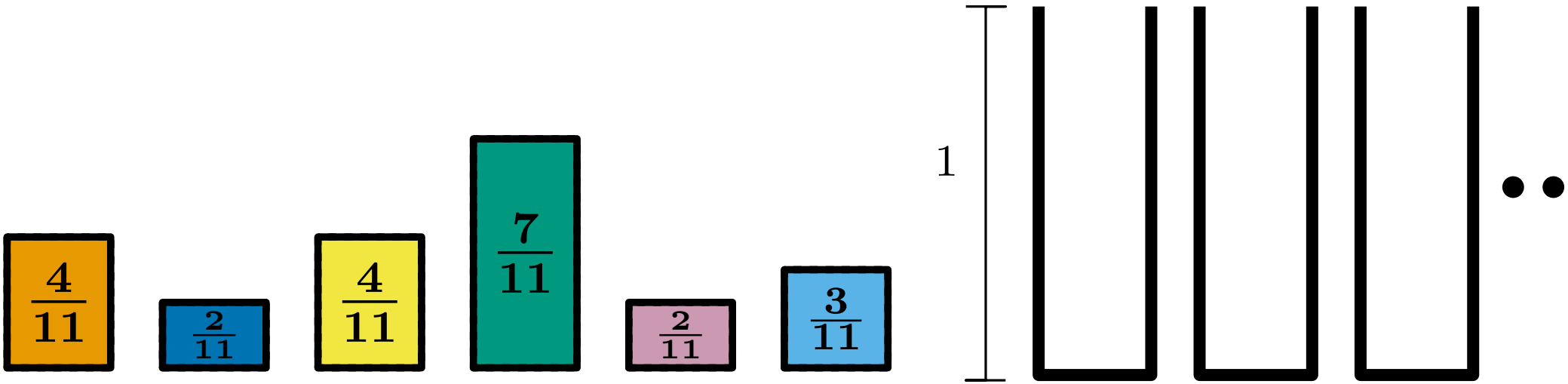
- Assume A is an α-approximation for BINPACKING with $\alpha < 3/2$

  A outputs a solution using s bins with $\mathrm{Opt}_b \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b$

- If $\mathrm{Opt}_b > 2$ then $s > 2$
- If $\mathrm{Opt}_b = 2$ then $2 \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b < (3/2) \cdot \mathrm{Opt}_b = 3$

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



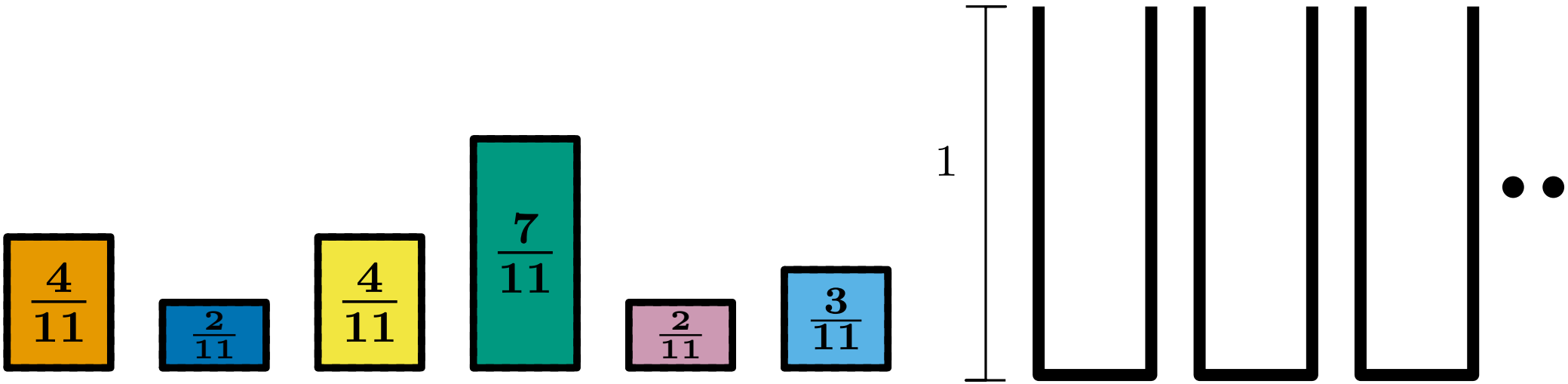
- Assume A is an α-approximation for BINPACKING with $\alpha < 3/2$

  A outputs a solution using s bins with $\mathrm{Opt}_b \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b$

- If $\mathrm{Opt}_b > 2$ then $s > 2$
- If $\mathrm{Opt}_b = 2$ then $2 \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b < (3/2) \cdot \mathrm{Opt}_b = 3$

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



- Assume A is an α-approximation for BINPACKING with $\alpha < 3/2$

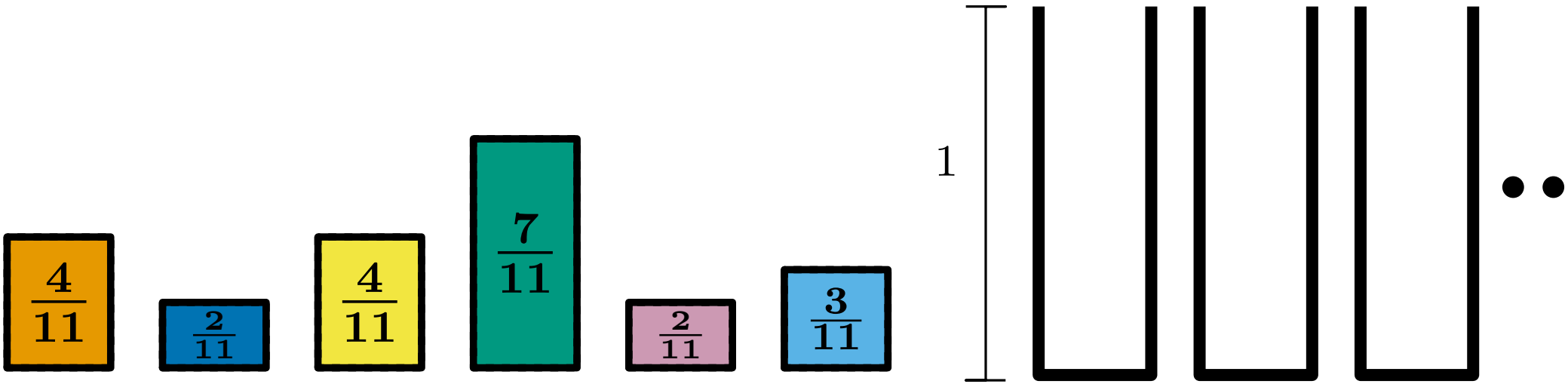
  A outputs a solution using s bins with $\mathrm{Opt}_b \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b$

- If $\mathrm{Opt}_b > 2$ then $s > 2$

- If $\mathrm{Opt}_b = 2$ then $2 \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b < (3/2) \cdot \mathrm{Opt}_b = 3$ so $s = 2$

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



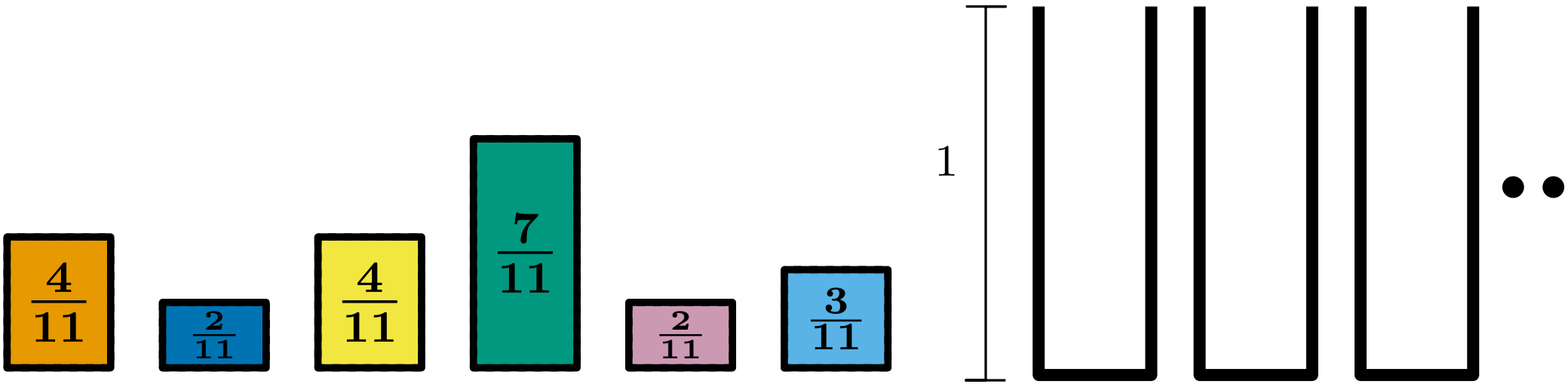
- Assume A is an α-approximation for BINPACKING with $\alpha < 3/2$

  A outputs a solution using s bins with $\mathrm{Opt}_b \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b$

- If $\mathrm{Opt}_b > 2$ then $s > 2$
- If $\mathrm{Opt}_b = 2$ then $2 \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b < (3/2) \cdot \mathrm{Opt}_b = 3$ so $s = 2$
- So A can solve the NP-complete PARTITION problem in polynomial time!

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



- Assume A is an α-approximation for BINPACKING with $\alpha < 3/2$

  A outputs a solution using s bins with $\mathrm{Opt}_b \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b$

- If $\mathrm{Opt}_b > 2$ then $s > 2$

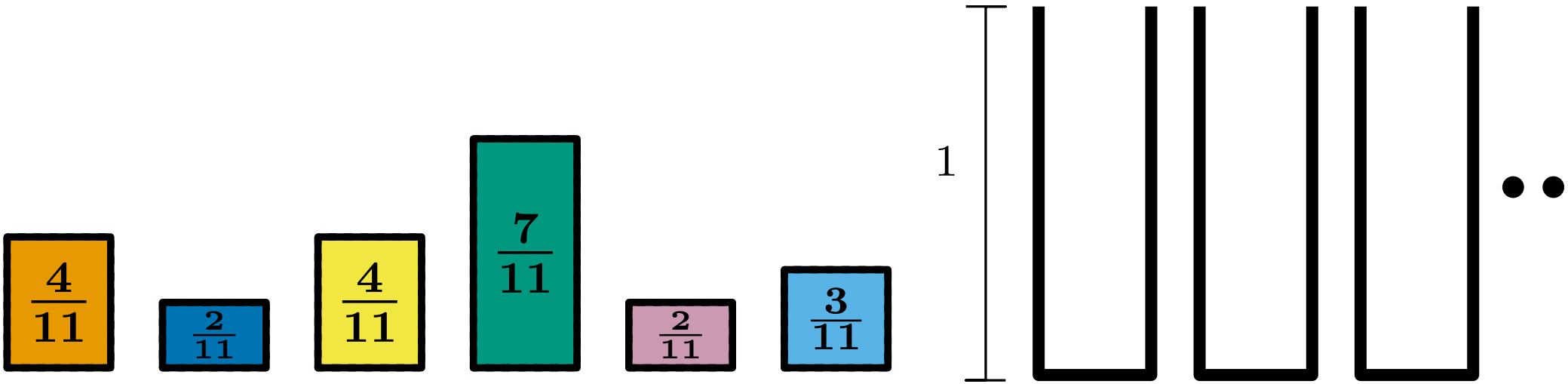
- If $\mathrm{Opt}_b = 2$ then $2 \leqslant s \leqslant \alpha \cdot \mathrm{Opt}_b < (3/2) \cdot \mathrm{Opt}_b = 3$ so $s = 2$

- So A can solve the NP-complete PARTITION problem in polynomial time!

  which implies that $\mathrm{P} = \mathrm{NP}$

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING

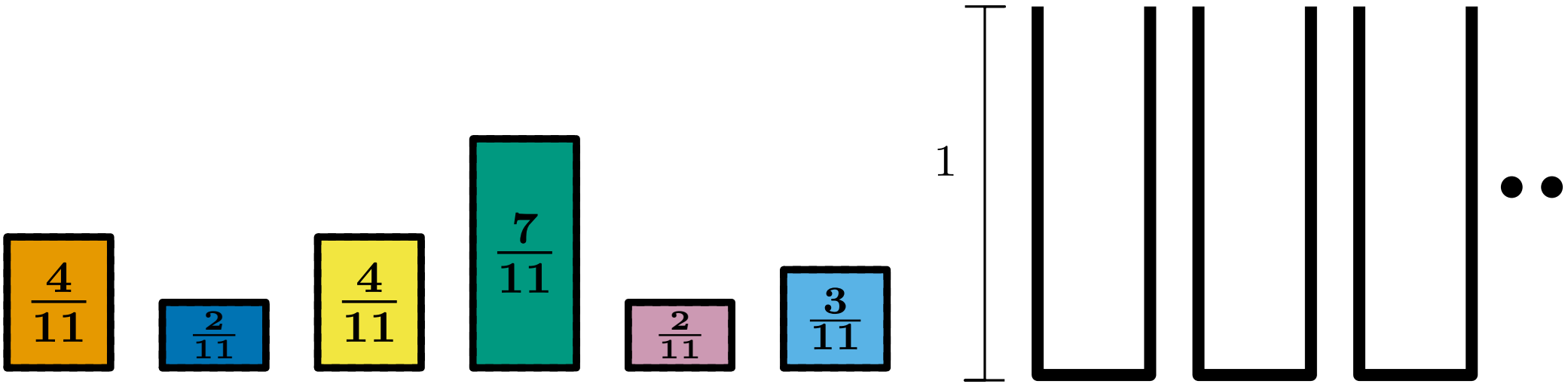


Lemma There is no α-approximation for BINPACKING with $\alpha < 3/2$

unless $\mathrm{P} = \mathrm{NP}$

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



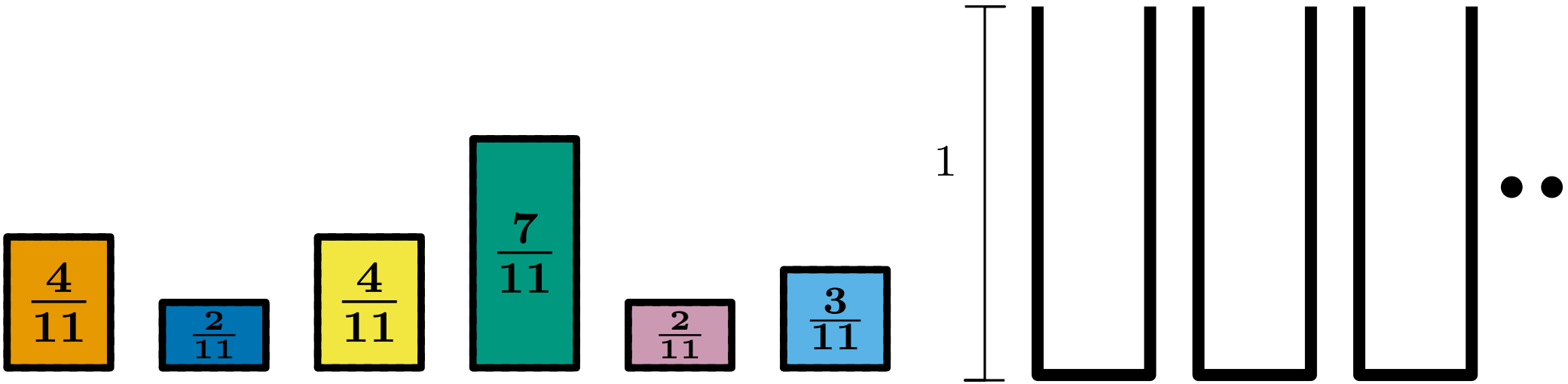
Lemma There is no α-approximation for BINPACKING with $\alpha < 3/2$

unless $\mathrm{P} = \mathrm{NP}$

We saw that First Fit Decreasing (FFD) is a $3/2$-approximation

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



Lemma There is no α-approximation for BINPACKING with $\alpha < 3/2$

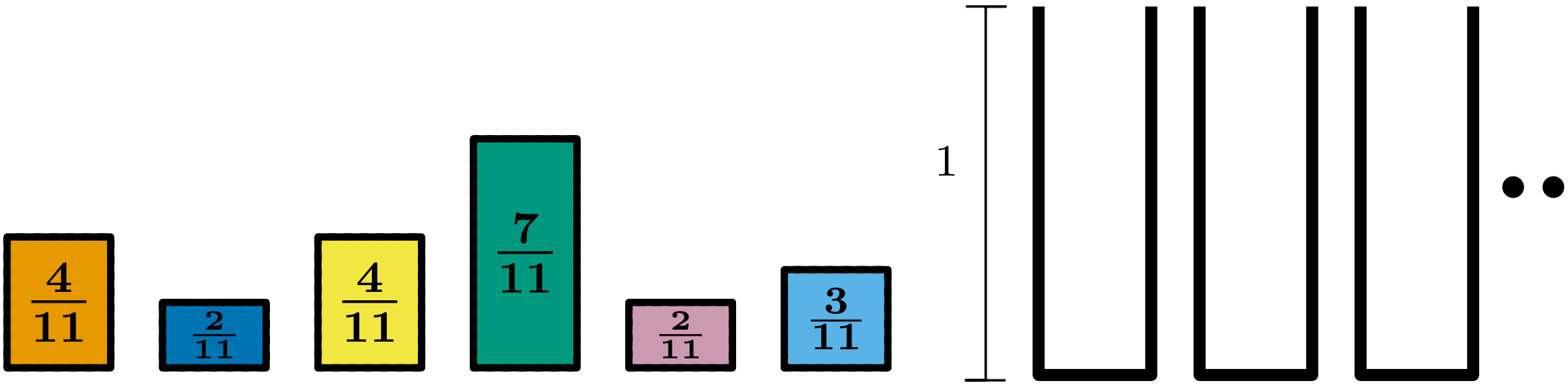
unless $\mathrm{P} = \mathrm{NP}$

We saw that First Fit Decreasing (FFD) is a $3/2$-approximation

so this is the best we can do?

PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING



Lemma There is no α-approximation for BINPACKING with $\alpha < 3/2$

unless $\mathrm{P} = \mathrm{NP}$

We saw that First Fit Decreasing (FFD) is a $3/2$-approximation

so this is the best we can do?

In fact, FFD gives a solution using s bins with

$$\mathrm{Opt}_b \leqslant s \leqslant \frac{11}{9} \cdot \mathrm{Opt}_b + 1$$

Asymptotic Polynomial time approximation schemes

An asymptotic polynomial time approximation scheme (APTAS)

for problem P is a family of algorithms:

There is a constant $c > 0$ such that

For any constant $\epsilon > 0$, A_ϵ runs in polynomial time (poly-time)

and outputs a solution s with $\mathrm{Opt} \leqslant s \leqslant (1+\epsilon) \cdot \mathrm{Opt} + c$

Asymptotic Polynomial time approximation schemes

An asymptotic polynomial time approximation scheme (APTAS)

for problem P is a family of algorithms:

There is a constant $c > 0$ such that

For any constant $\epsilon > 0$, A_ϵ runs in polynomial time (poly-time)

and outputs a solution s with $\mathrm{Opt} \leqslant s \leqslant (1+\epsilon) \cdot \mathrm{Opt} + c$

(this is the minimisation version of the definition)

Asymptotic Polynomial time approximation schemes

An asymptotic polynomial time approximation scheme (APTAS)

for problem P is a family of algorithms:

There is a constant $c > 0$ such that

For any constant $\epsilon > 0$, A_ϵ runs in polynomial time (poly-time)

and outputs a solution s with $\mathrm{Opt} \leqslant s \leqslant (1+\epsilon) \cdot \mathrm{Opt} + c$

(this is the minimisation version of the definition)

An APTAS does not have to have running time polynomial in $1/\epsilon$

Asymptotic Polynomial time approximation schemes

An asymptotic polynomial time approximation scheme (APTAS)

for problem P is a family of algorithms:

There is a constant $c > 0$ such that

For any constant $\epsilon > 0$, A_ϵ runs in polynomial time (poly-time)

and outputs a solution s with $\mathrm{Opt} \leqslant s \leqslant (1+\epsilon) \cdot \mathrm{Opt} + c$

(this is the minimisation version of the definition)

An APTAS does not have to have running time polynomial in $1/\epsilon$

An asymptotic fully PTAS (AFPTAS) is also polynomial in $1/\epsilon$

Asymptotic Polynomial time approximation schemes

An asymptotic polynomial time approximation scheme (APTAS)

for problem P is a family of algorithms:

There is a constant $c > 0$ such that

For any constant $\epsilon > 0$, A_ϵ runs in polynomial time (poly-time)

and outputs a solution s with $\mathrm{Opt} \leqslant s \leqslant (1+\epsilon) \cdot \mathrm{Opt} + c$

(this is the minimisation version of the definition)

An APTAS does not have to have running time polynomial in $1/\epsilon$

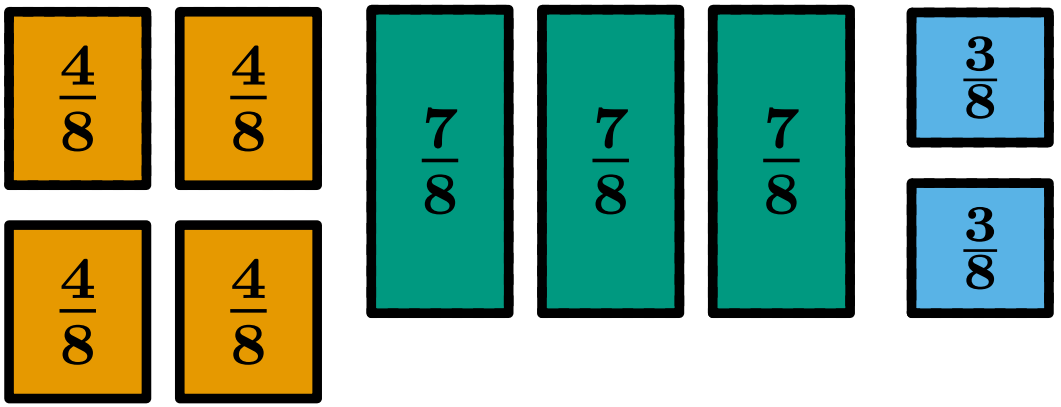
An asymptotic fully PTAS (AFPTAS) is also polynomial in $1/\epsilon$

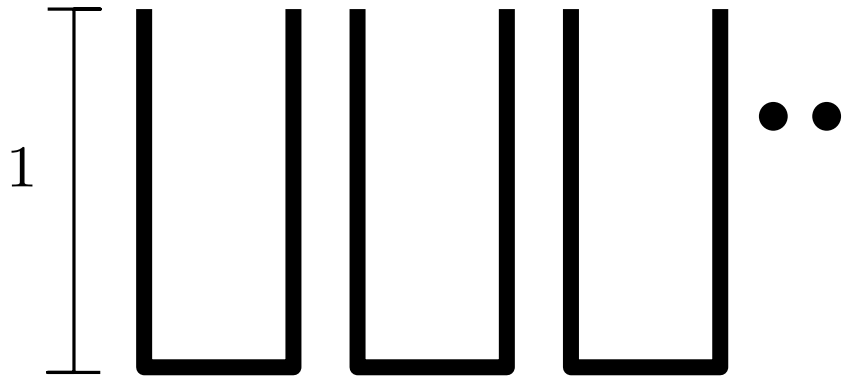
We will see an APTAS for BinPacking

which uses at most $(1+\epsilon) \cdot \mathrm{Opt} + 1$ bins

A special case of BINPACKING

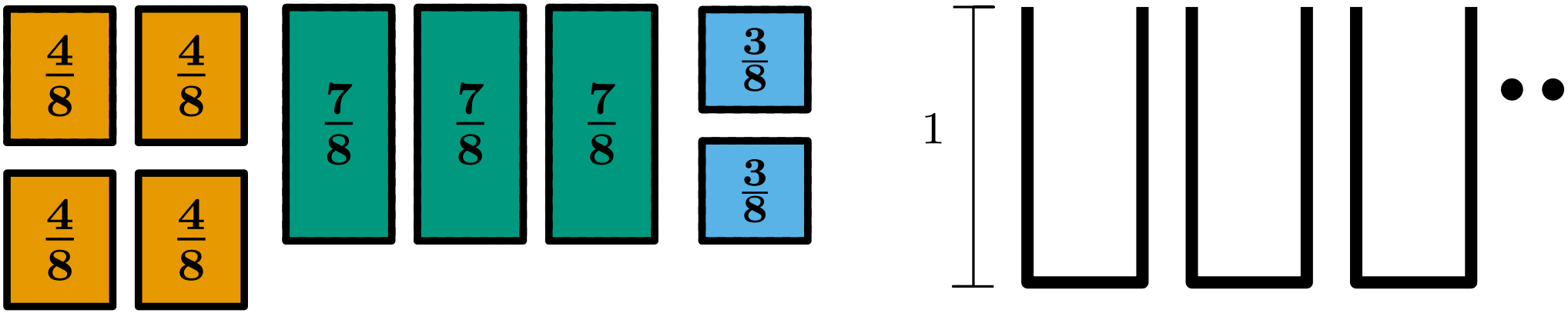
We will now see a special case of BINPACKING which we can solve optimally in polynomial time





A special case of BINPACKING

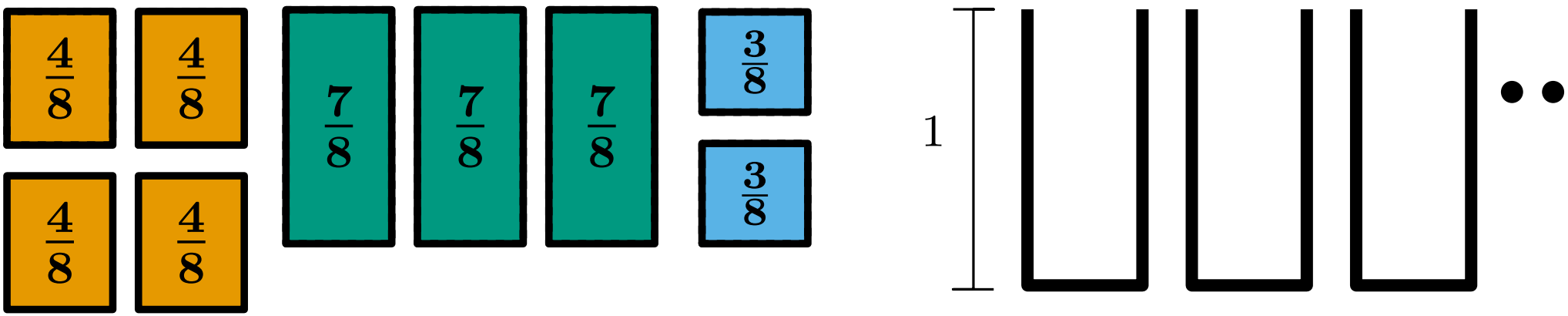
We will now see a special case of BINPACKING which we can solve optimally in polynomial time



We have n items but

A special case of BINPACKING

We will now see a special case of BINPACKING which we can solve optimally in polynomial time

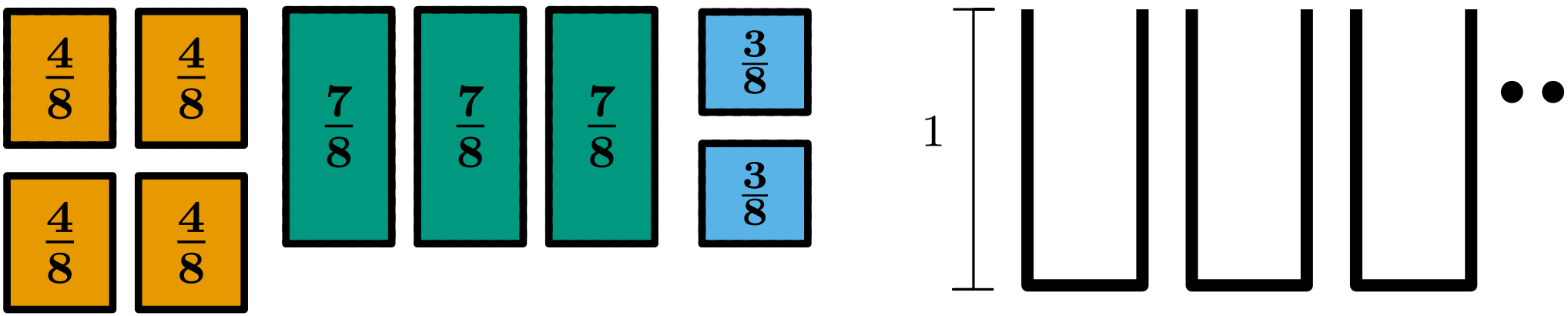


We have n items but

- There are c_s (a constant) number of different item sizes

A special case of BINPACKING

We will now see a special case of BINPACKING which we can solve optimally in polynomial time

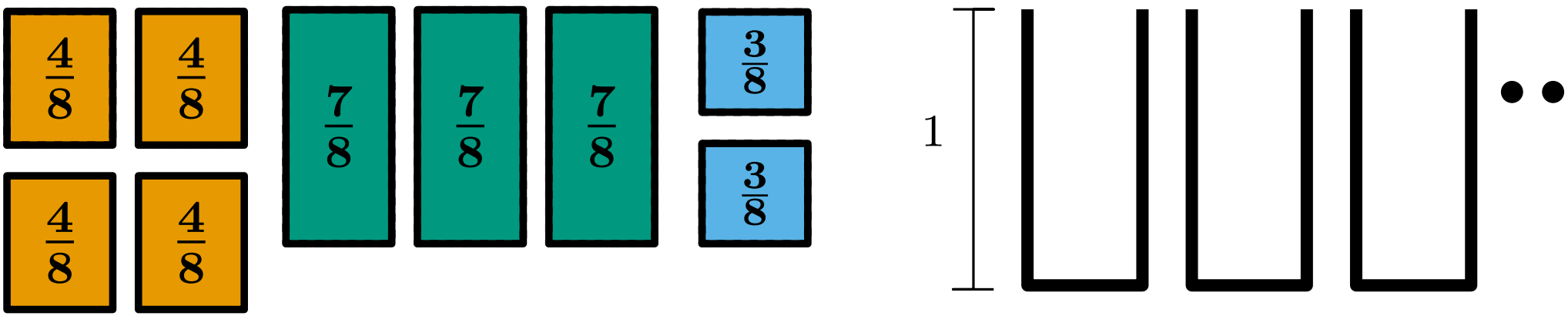


We have n items but

- There are c_s (a constant) number of different item sizes
- At most c_b (another constant) items fit in each bin

A special case of BINPACKING

We will now see a special case of BINPACKING which we can solve optimally in polynomial time



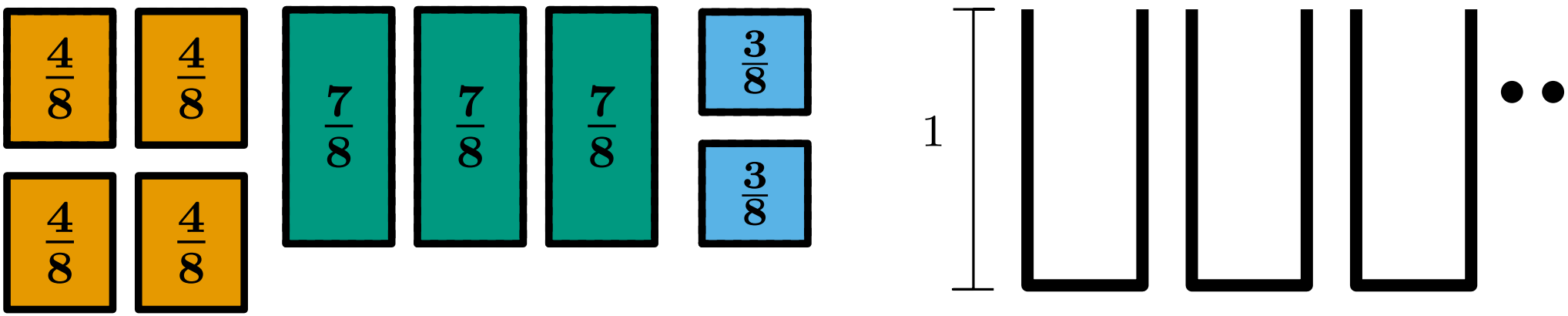
Here $c_s = 3$ and $c_b = 2$

We have n items but

- There are c_s (a constant) number of different item sizes
- At most c_b (another constant) items fit in each bin

A special case of BINPACKING

We will now see a special case of BINPACKING which we can solve optimally in polynomial time



Here $c_s = 3$ and $c_b = 2$

We have n items but

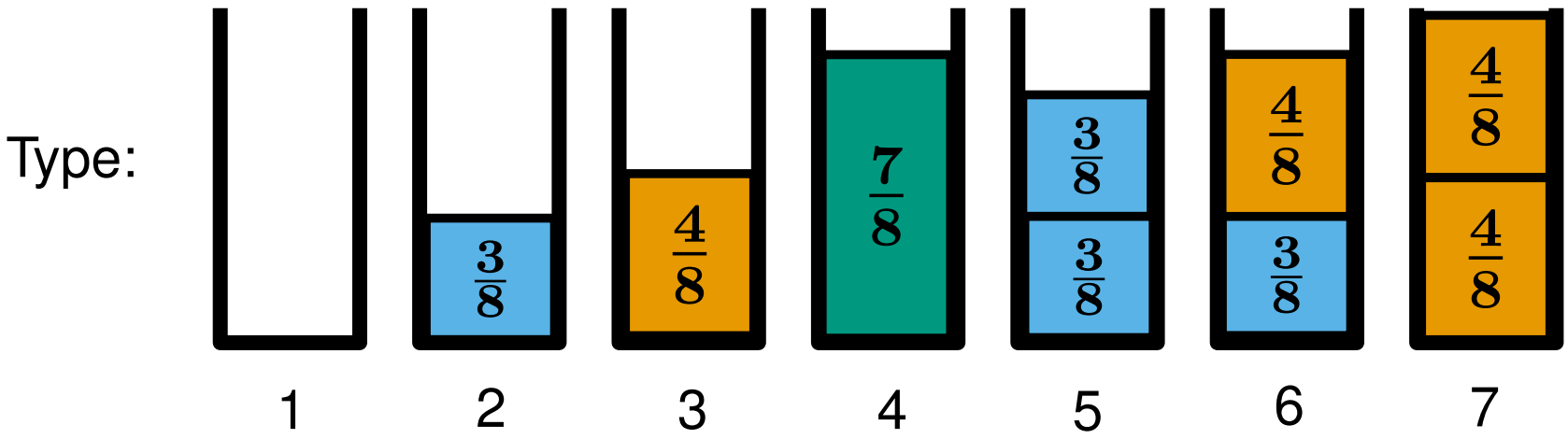
- There are c_s (a constant) number of different item sizes
- At most c_b (another constant) items fit in each bin

How many different ways are there to fill a single bin?

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes

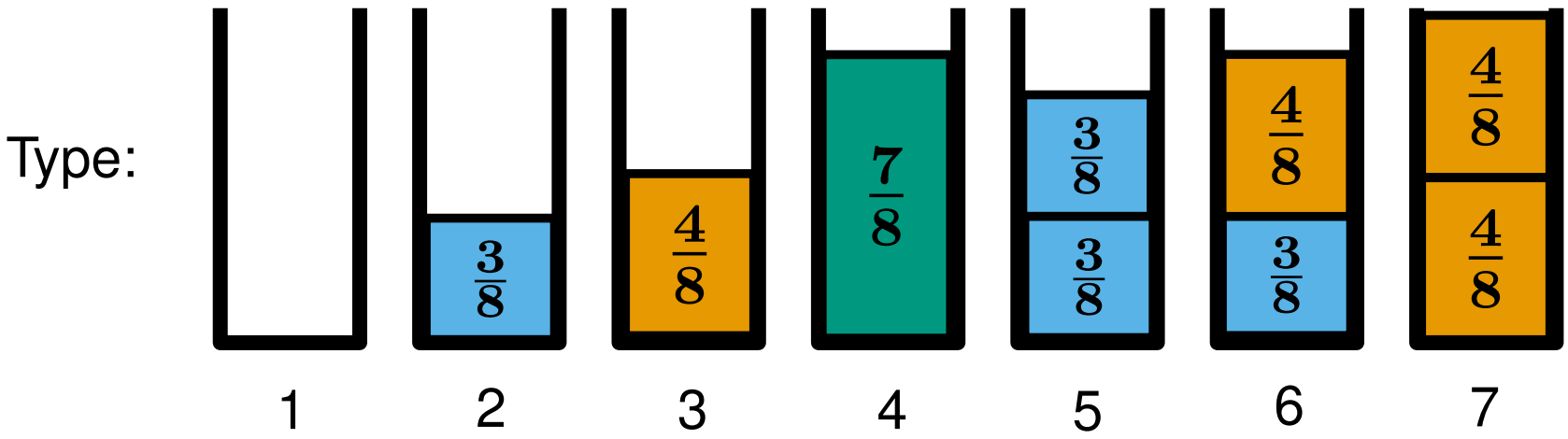


- We can describe any packing of items into a single bin by its *type*

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes

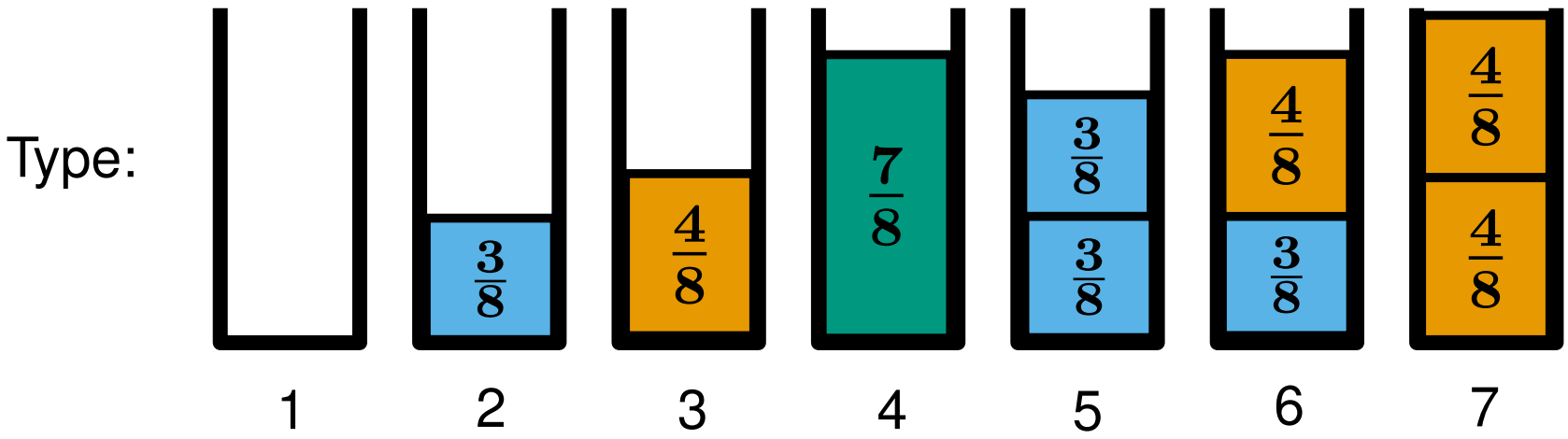


- We can describe any packing of items into a single bin by its *type*
- The type of a packed bin determines how many of each item is packed

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



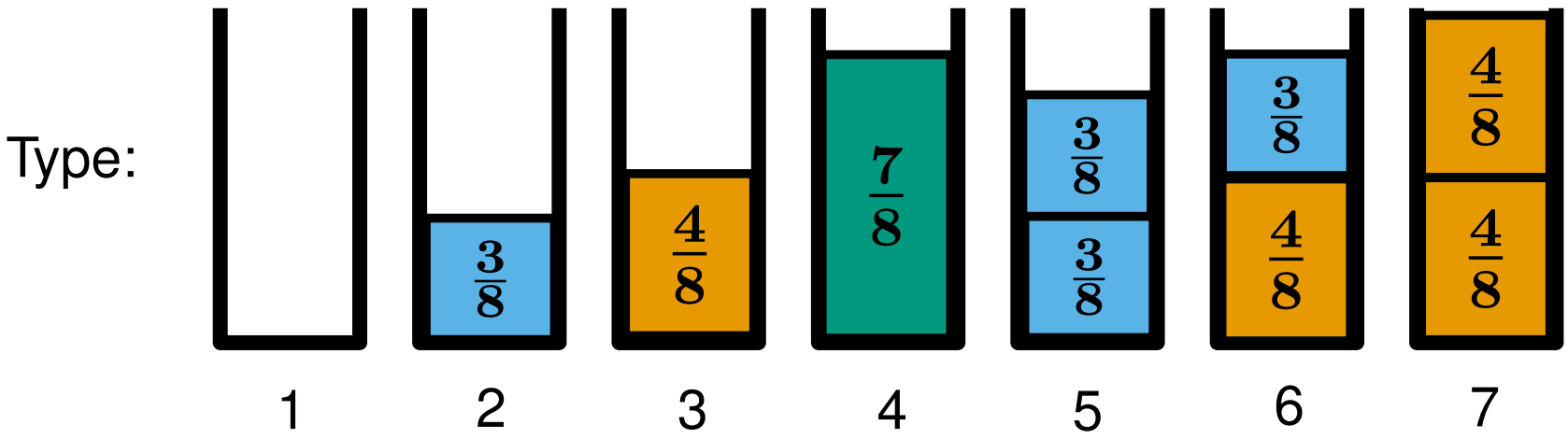
- We can describe any packing of items into a single bin by its *type*
- The type of a packed bin determines how many of each item is packed

we ignore rearrangement of items

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



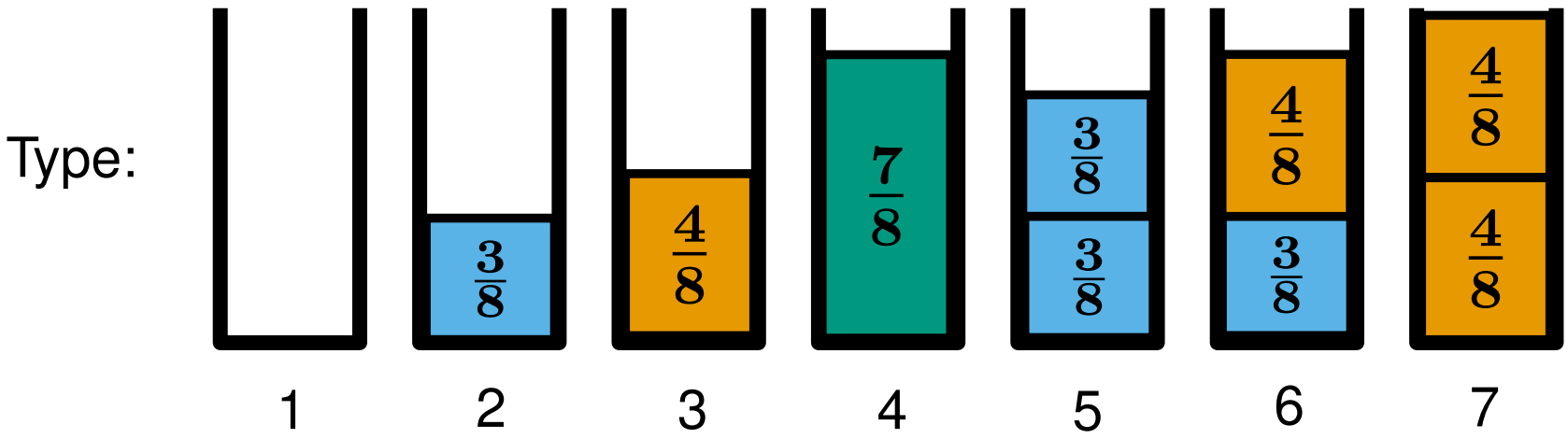
- We can describe any packing of items into a single bin by its *type*
- The type of a packed bin determines how many of each item is packed

we ignore rearrangement of items

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



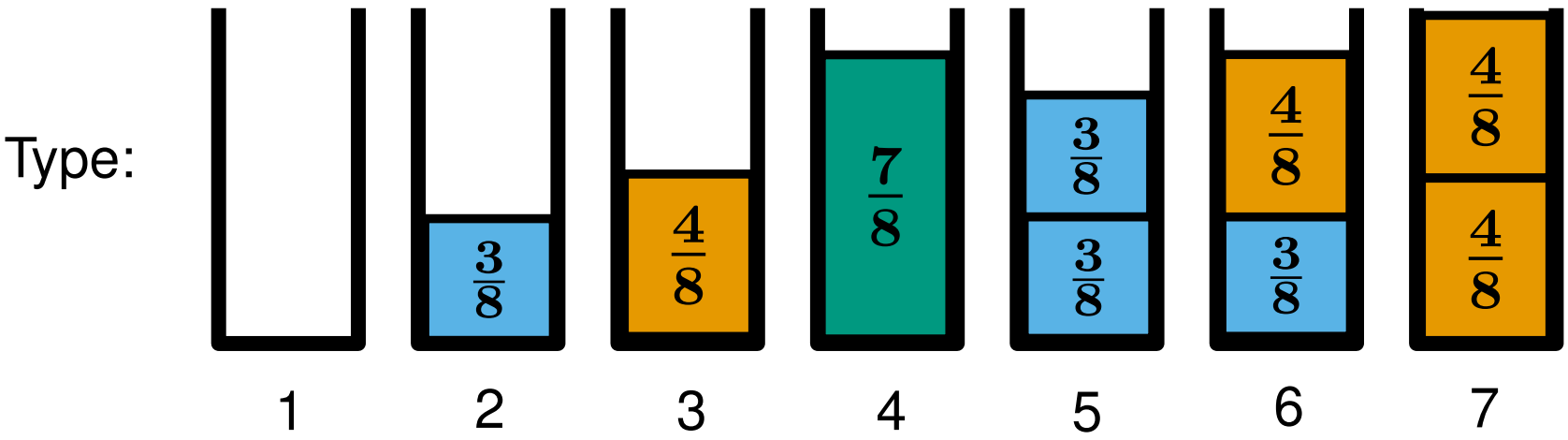
- We can describe any packing of items into a single bin by its *type*
- The type of a packed bin determines how many of each item is packed

we ignore rearrangement of items

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



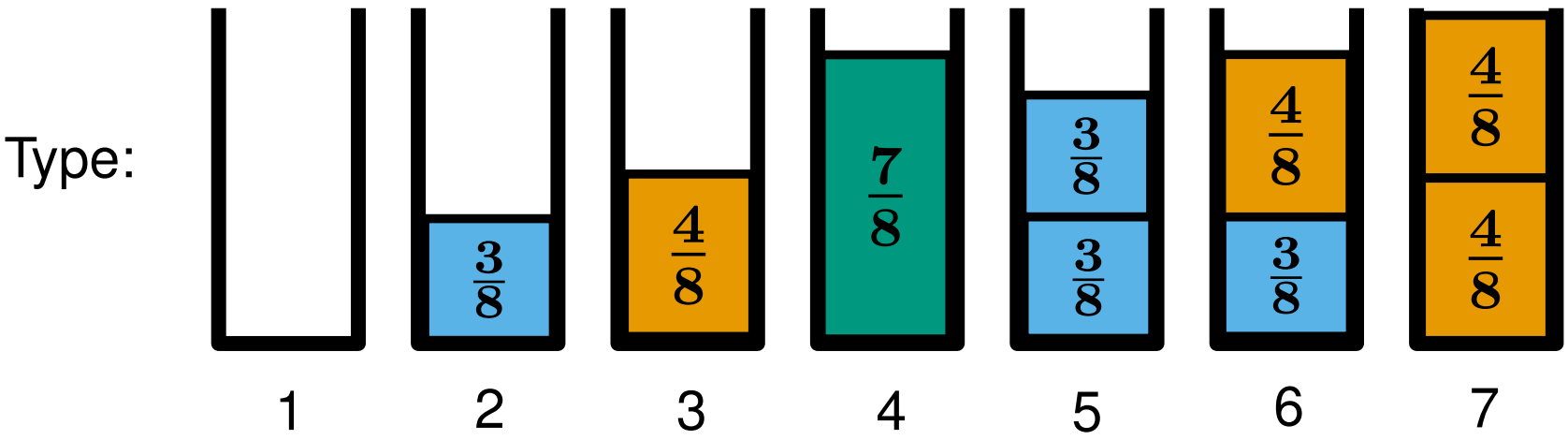
- We can describe any packing of items into a single bin by its *type*
- The type of a packed bin determines how many of each item is packed

  we ignore rearrangement of items
- How many types can there be?

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



- We can describe any packing of items into a single bin by its *type*
- The type of a packed bin determines how many of each item is packed

  we ignore rearrangement of items

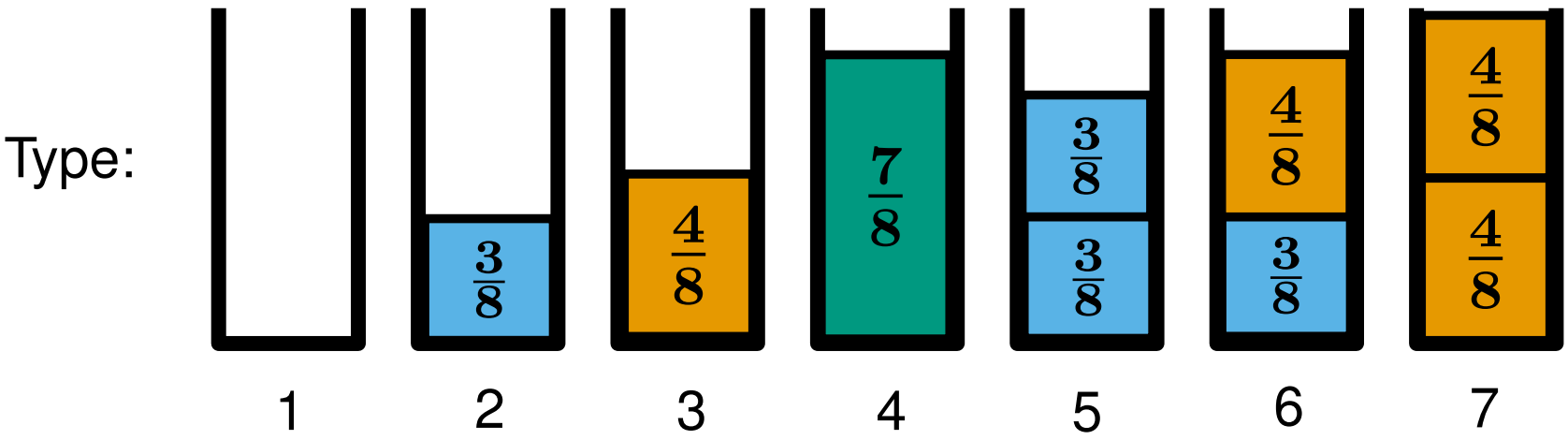
- How many types can there be?

  There are between 0 and c_b items of any one size

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



- We can describe any packing of items into a single bin by its *type*
- The type of a packed bin determines how many of each item is packed

  we ignore rearrangement of items

- How many types can there be?

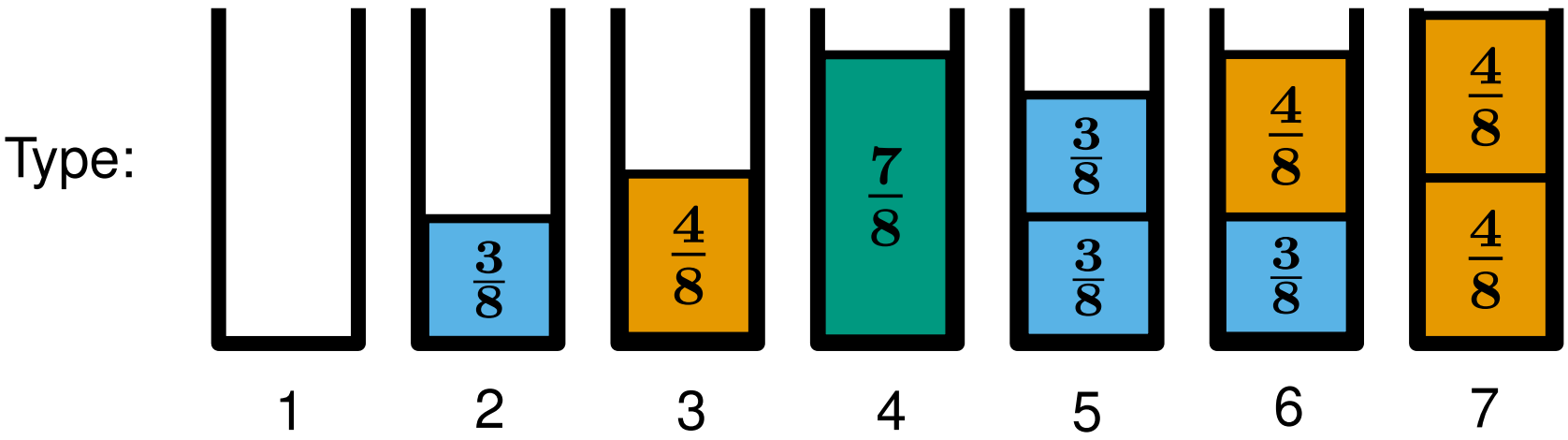
  There are between 0 and c_b items of any one size

  There are c_s different sizes

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



- We can describe any packing of items into a single bin by its *type*
- The type of a packed bin determines how many of each item is packed

  we ignore rearrangement of items
- How many types can there be?

  There are between 0 and c_b items of any one size

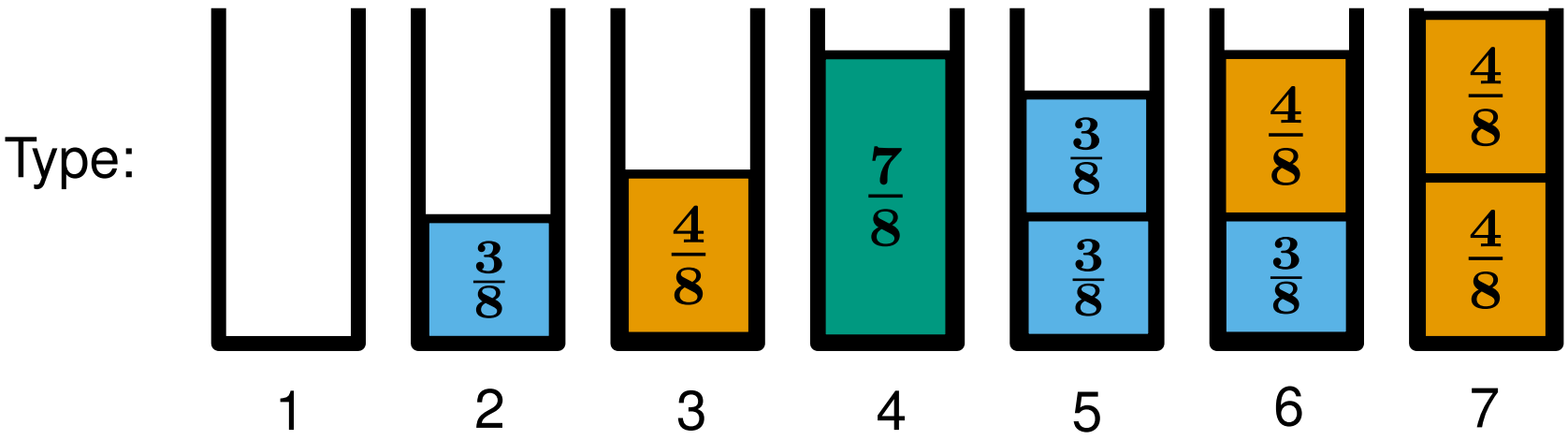
  There are c_s different sizes

  The number of types $\leqslant$

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



- We can describe any packing of items into a single bin by its *type*
- The type of a packed bin determines how many of each item is packed

  we ignore rearrangement of items
- How many types can there be?

  There are between 0 and c_b items of any one size

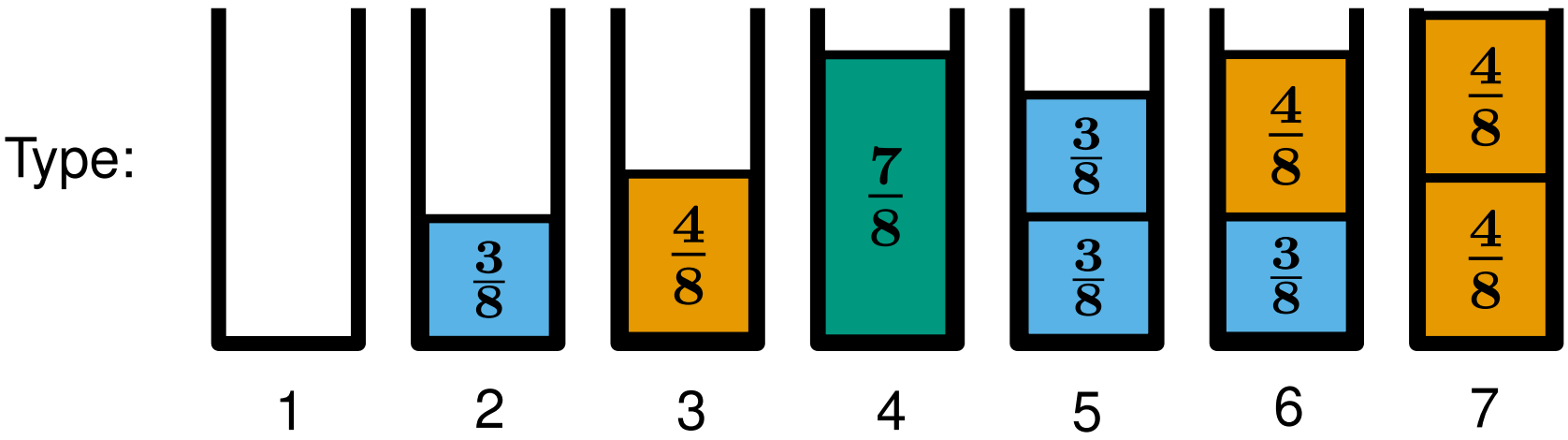
  There are c_s different sizes

  The number of types $\leqslant \underbrace{(c_b+1) \times (c_b+1) \times \ldots \times (c_b+1)}_{c_s}$

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



- We can describe any packing of items into a single bin by its *type*
- The type of a packed bin determines how many of each item is packed

  we ignore rearrangement of items

- How many types can there be?

  There are between 0 and c_b items of any one size

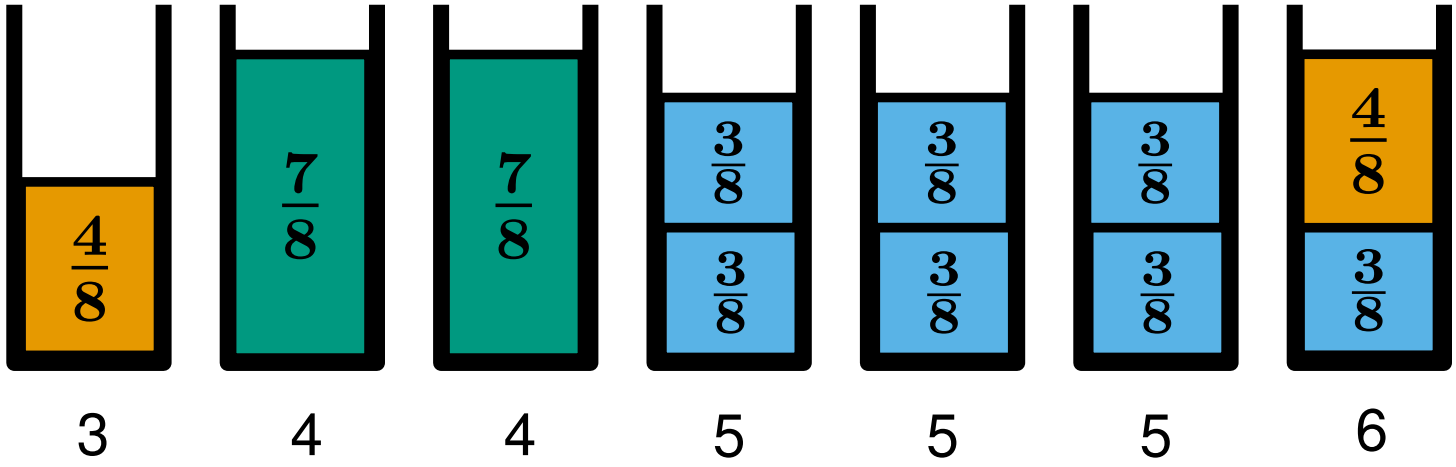
  There are c_s different sizes

  The number of types $\leqslant \underbrace{(c_b+1)\times(c_b+1)\times\ldots\times(c_b+1)}_{c_s} = (c_b+1)^{c_s}$

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



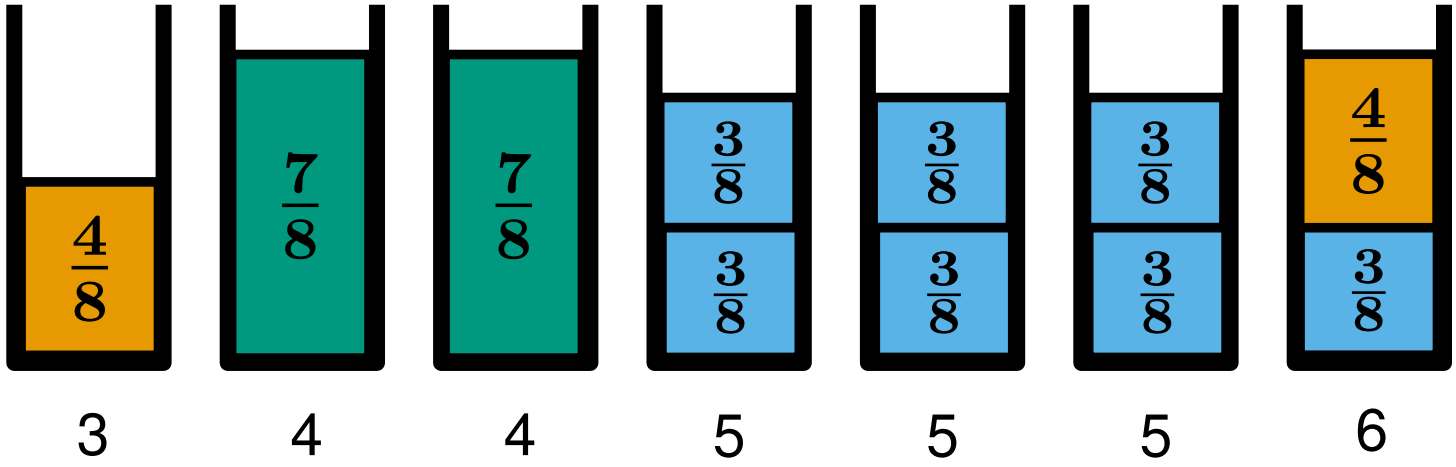
- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

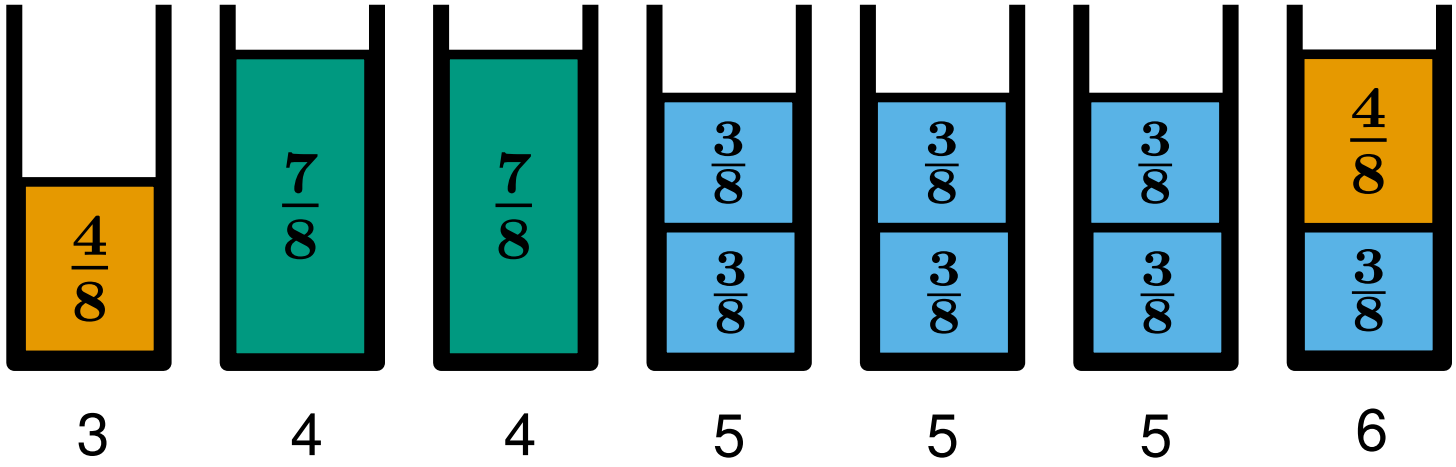
- We can completely describe any packing of S into $b \leqslant n$ bins

by the number of bins of each type used

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



- $0\times$ Type 1
- $0\times$ Type 2
- $1\times$ Type 3
- $2\times$ Type 4
- $3\times$ Type 5
- $1\times$ Type 6
- $0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

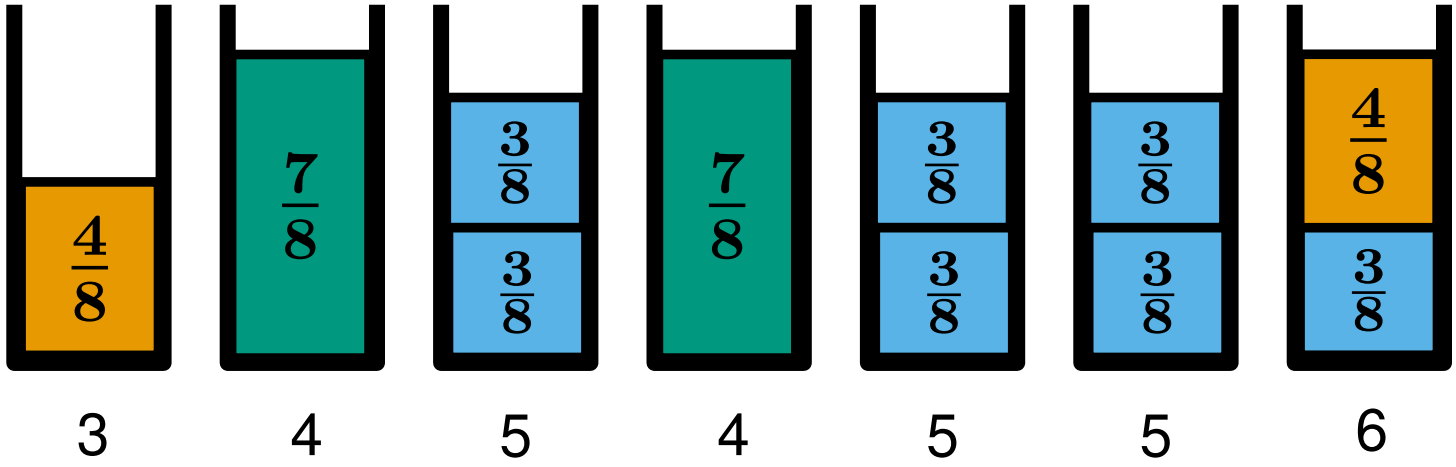
  by the number of bins of each type used

  we ignore rearrangement of bins

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

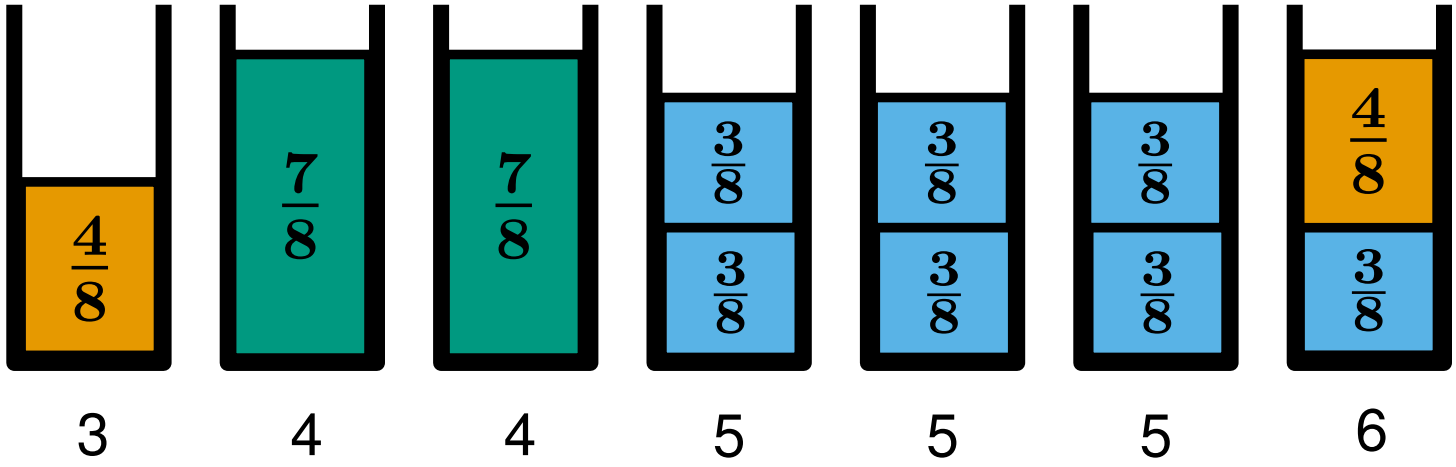
by the number of bins of each type used

we ignore rearrangement of bins

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



- $0\times$ Type 1
- $0\times$ Type 2
- $1\times$ Type 3
- $2\times$ Type 4
- $3\times$ Type 5
- $1\times$ Type 6
- $0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

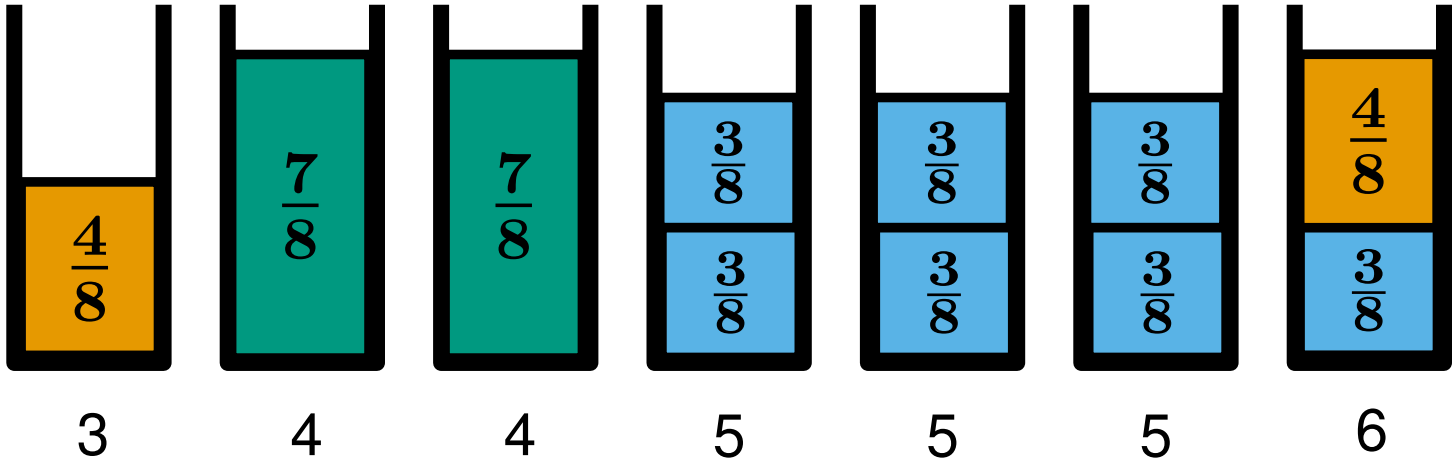
by the number of bins of each type used

we ignore rearrangement of bins

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

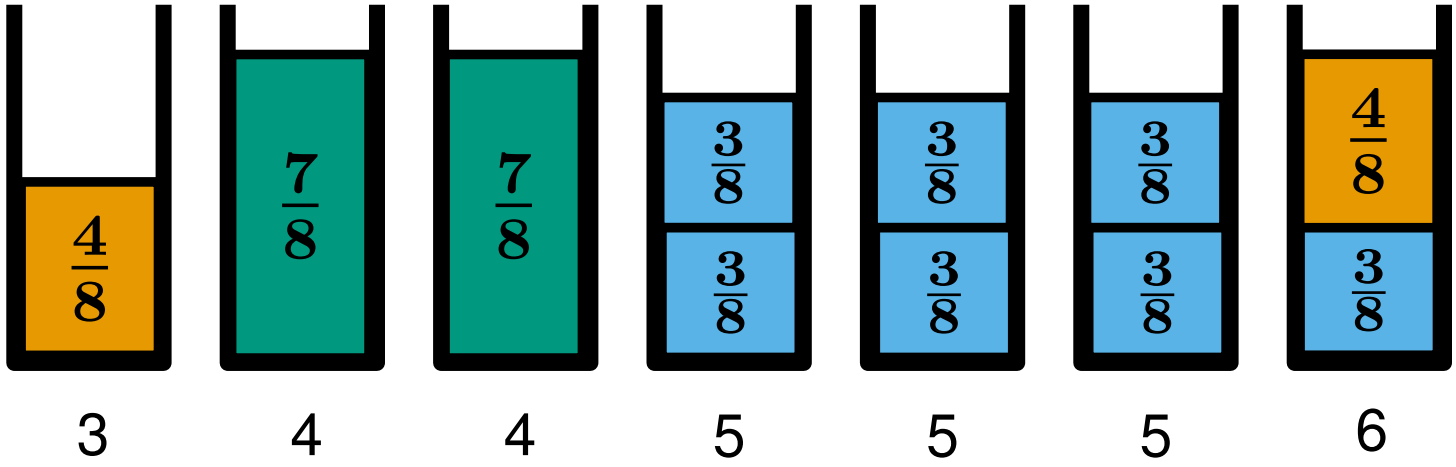
  we ignore rearrangement of bins

- How *different* packings can there be?

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

  we ignore rearrangement of bins

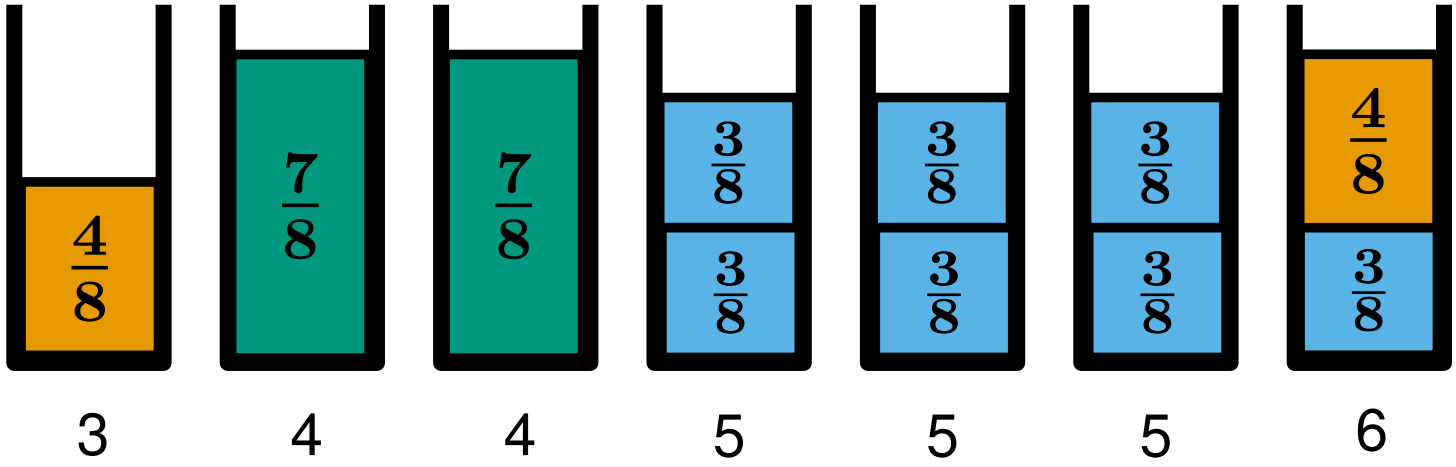
- How *different* packings can there be?

  There are between 0 and n bins of any type

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

  we ignore rearrangement of bins

- How *different* packings can there be?

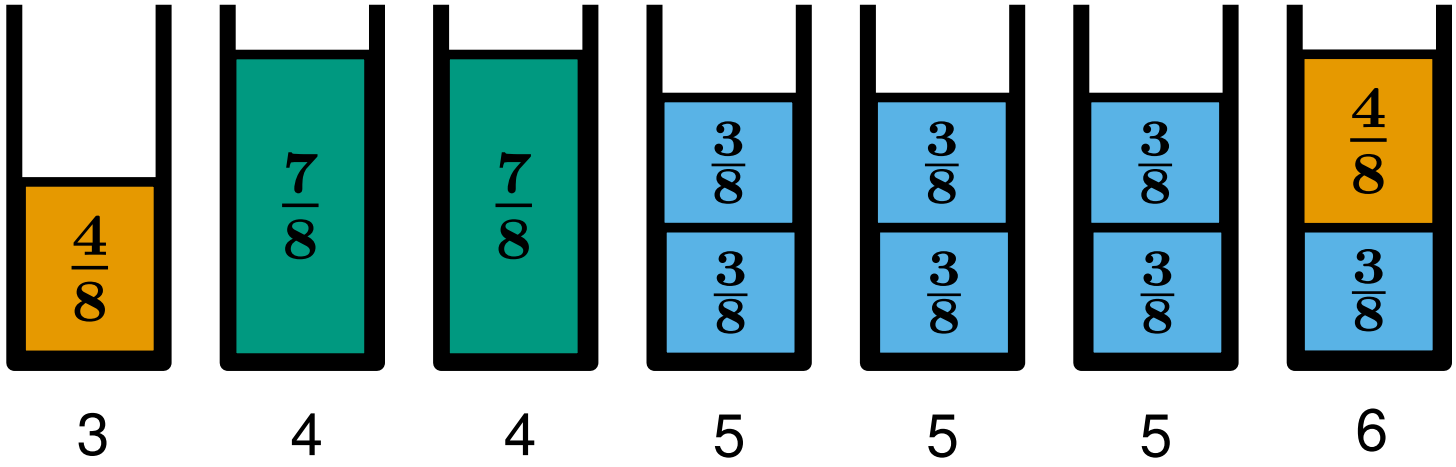
  There are between 0 and n bins of any type

  There are at most $(c_b + 1)^{c_s}$ different types

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

  we ignore rearrangement of bins

- How *different* packings can there be?

  There are between 0 and n bins of any type

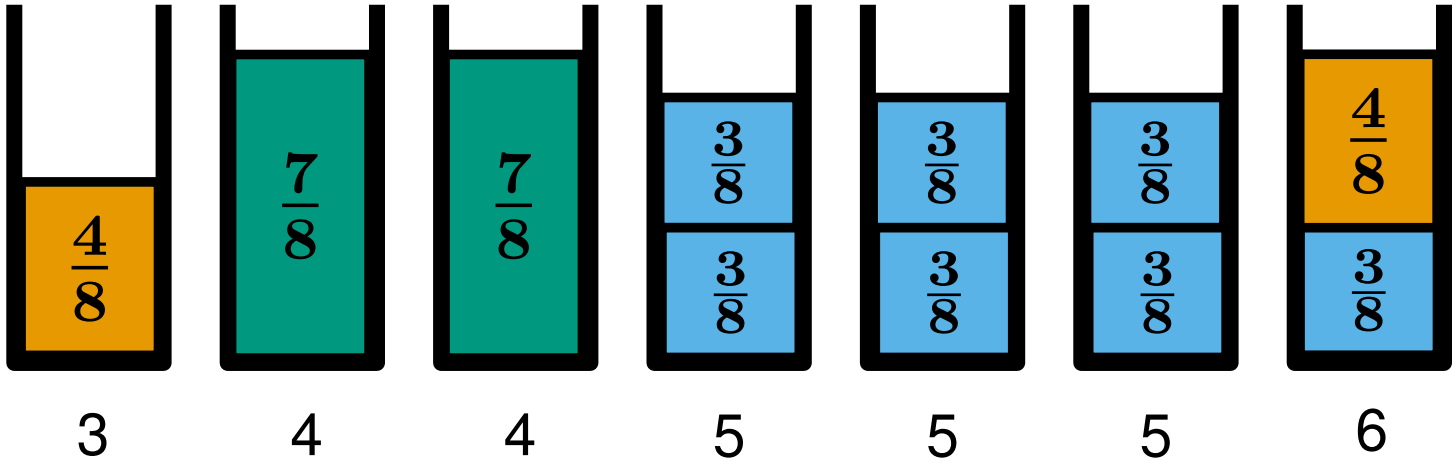
  There are at most $(c_b + 1)^{c_s}$ different types

  number of packings $\leqslant$

c_b items fit in each bin

A special case of BinPacking

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

  we ignore rearrangement of bins

- How *different* packings can there be?

  There are between 0 and n bins of any type

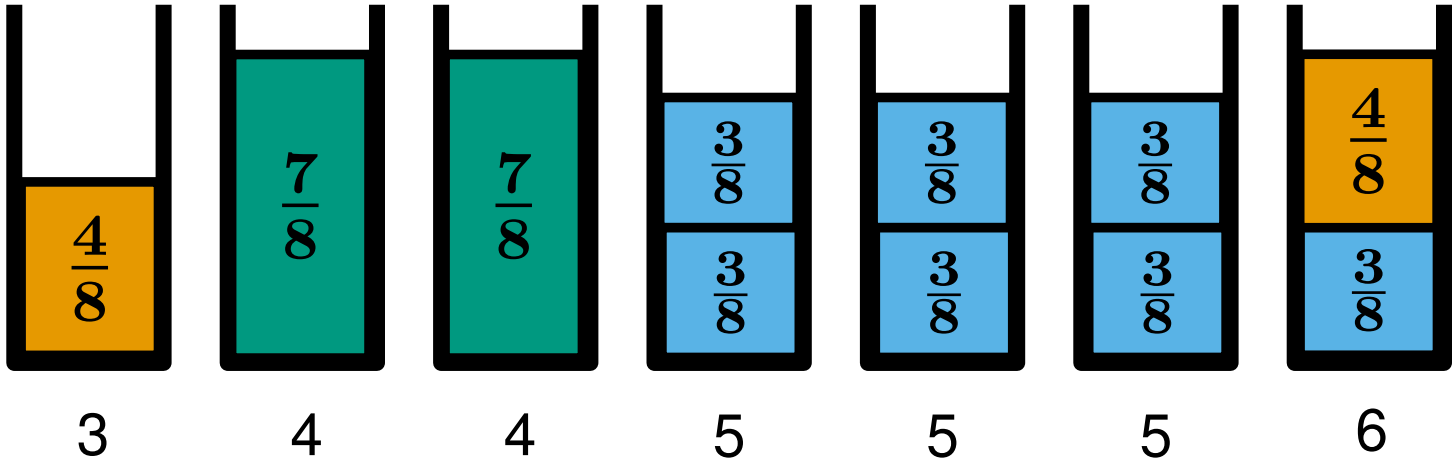
  There are at most $(c_b + 1)^{c_s}$ different types

$$\text{number of packings} \leqslant \underbrace{(n+1) \times (n+1) \times \ldots \times (n+1)}_{(c_b+1)^{c_s}}$$

c_b items fit in each bin

A special case of BinPacking

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

  we ignore rearrangement of bins

- How *different* packings can there be?

  There are between 0 and n bins of any type

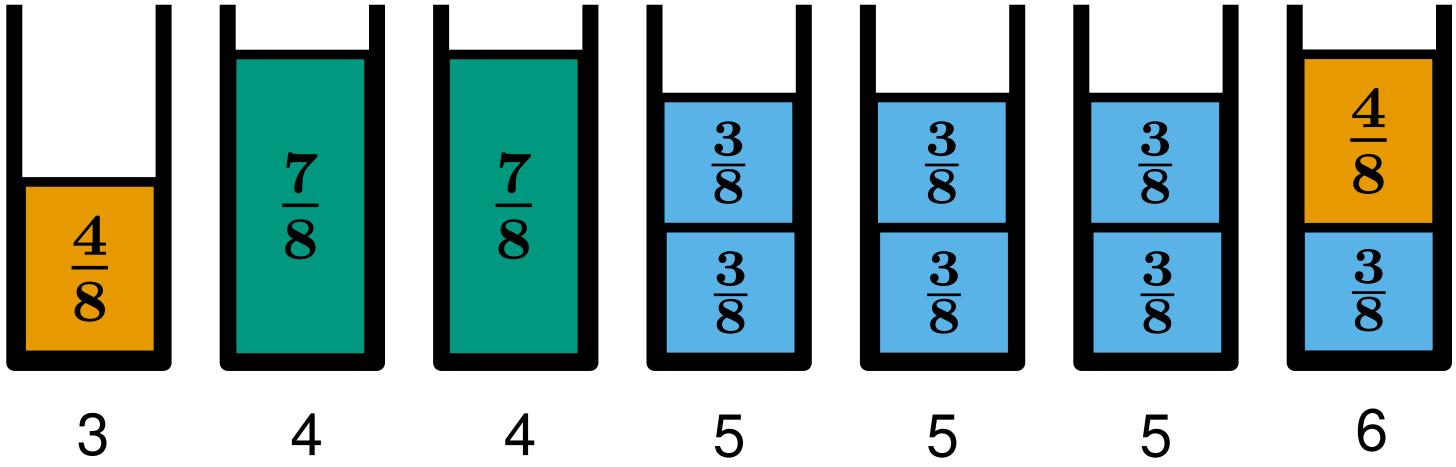
  There are at most $(c_b + 1)^{c_s}$ different types

$$\text{number of packings} \leqslant \underbrace{(n+1) \times (n+1) \times \ldots \times (n+1)}_{(c_b+1)^{c_s}} = (n+1)^{(c_b+1)^{c_s}}$$

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



- $0\times$ Type 1
- $0\times$ Type 2
- $1\times$ Type 3
- $2\times$ Type 4
- $3\times$ Type 5
- $1\times$ Type 6
- $0\times$ Type 7

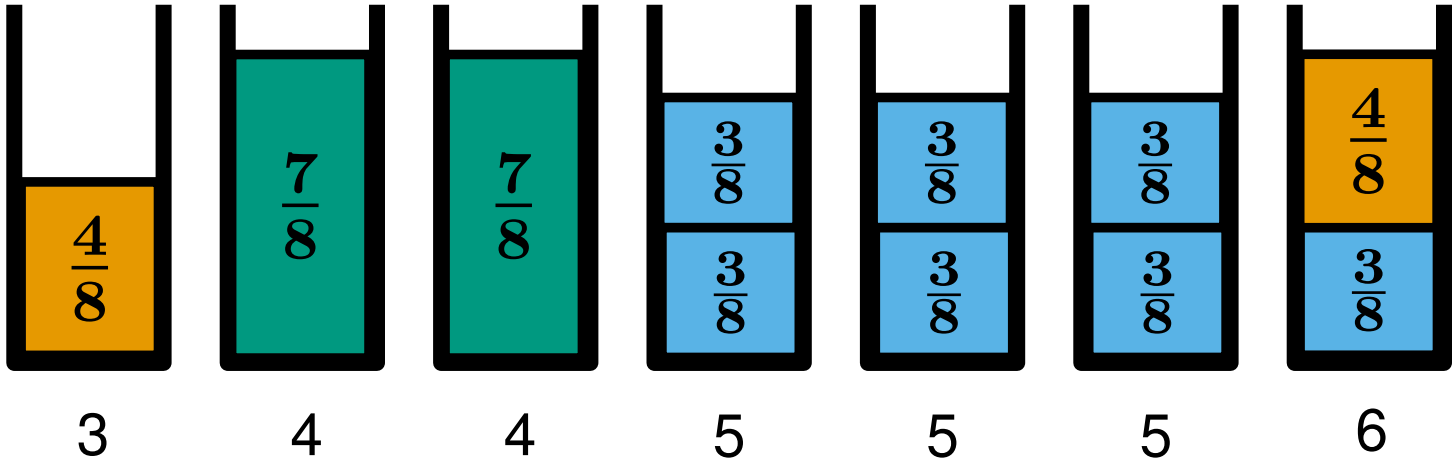
- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

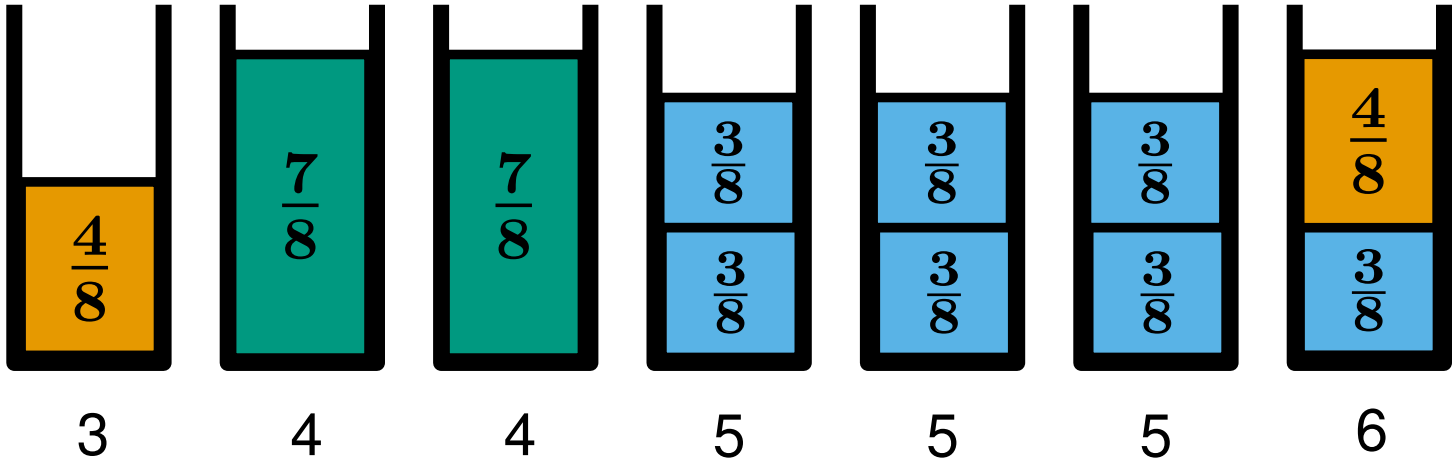
  by the number of bins of each type used

- However, not all these different packings are *valid*

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

- However, not all these different packings are *valid*

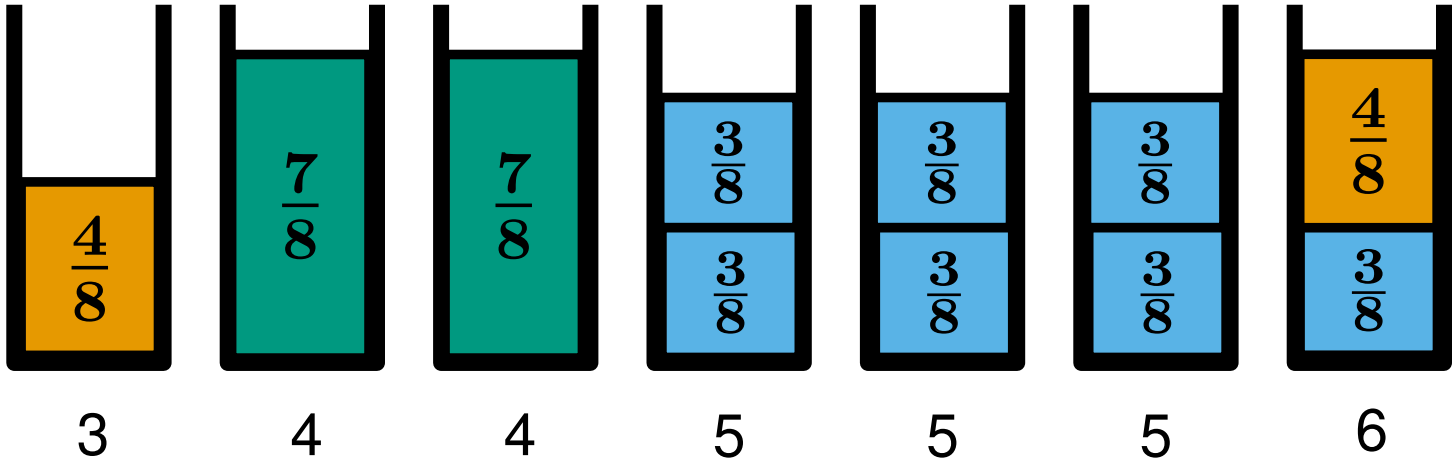
  Many of them use too few or many items of a particular size

  (in particular the example packing uses too many items of size $3/8$)

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

- However, not all these different packings are *valid*

  Many of them use too few or many items of a particular size

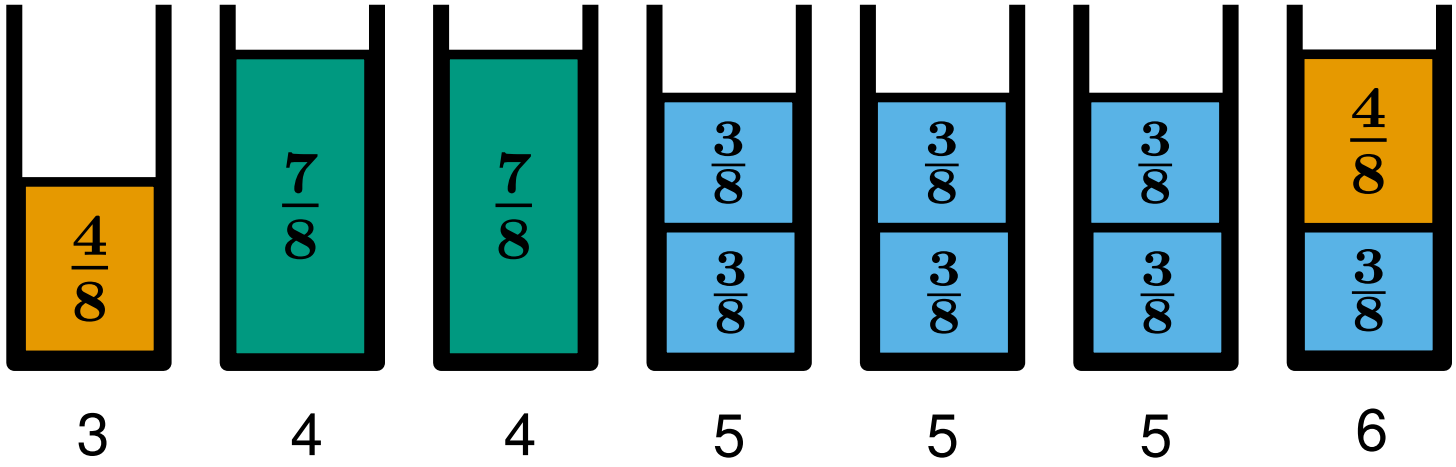
  (in particular the example packing uses too many items of size $3/8$)

- We check each of the different packings to see if it is valid

c_b items fit in each bin

A special case of BINPACKING

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

- However, not all these different packings are *valid*

  Many of them use too few or many items of a particular size

  (in particular the example packing uses too many items of size $3/8$)

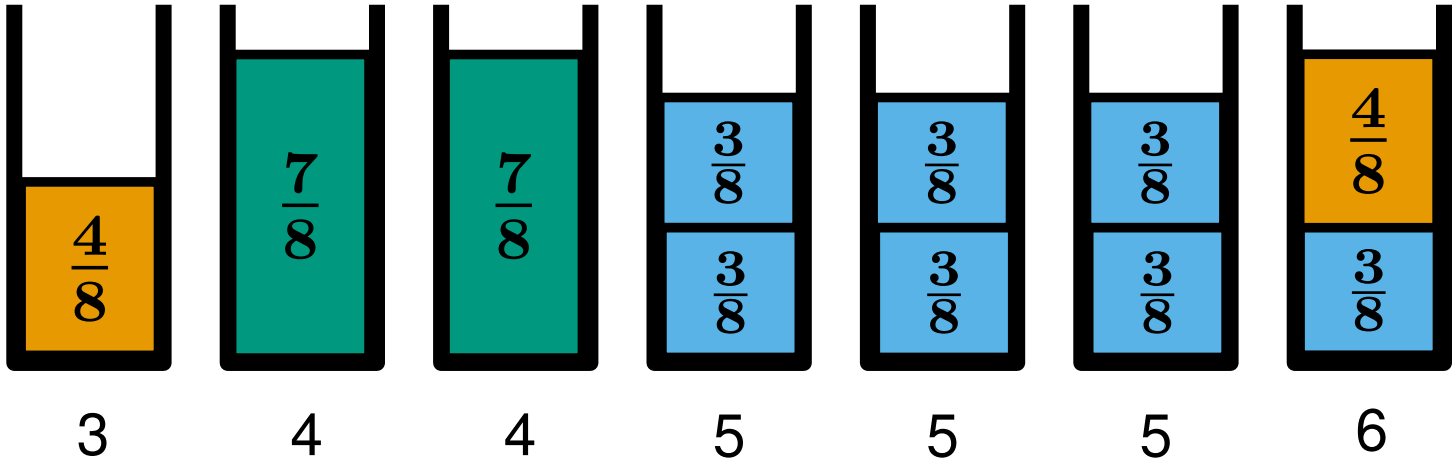
- We check each of the different packings to see if it is valid

  and output the valid one which uses the fewest bins

c_b items fit in each bin

A special case of BinPacking

c_s diff. item sizes



$0\times$ Type 1
$0\times$ Type 2
$1\times$ Type 3
$2\times$ Type 4
$3\times$ Type 5
$1\times$ Type 6
$0\times$ Type 7

- We can completely describe any packing of S into $b \leqslant n$ bins

  by the number of bins of each type used

- However, not all these different packings are *valid*

  Many of them use too few or many items of a particular size

  (in particular the example packing uses too many items of size $3/8$)

- We check each of the different packings to see if it is valid

  and output the valid one which uses the fewest bins

- This takes $O(n \cdot (n+1)^{(c_b+1)^{c_s}})$ time and exactly solves BinPacking

  (i.e. it outputs an optimal packing)

Towards an APTAS

The APTAS for BINPACKING will use a four step process:

Step 1 Remove all the *small* items

Only c_b of the remaining large items will fit into a single bin

Step 2 Divide the items into a constant number of groups

Sizes of items in each group are then rounded up

to match the size of the largest member

(and the largest group is removed)

This will leave only c_s different item sizes

Step 3 Use the poly-time algorithm to optimally pack the remaining special case

the *constants* c_b and c_s will depend on ϵ

Step 4 Reverse the grouping from **Step 2** and then

greedily pack all the small items removed in **Step 1**

Remember that the goal is to use b bins where $\mathrm{Opt} \leqslant b \leqslant (1+\epsilon) \cdot \mathrm{Opt} + c$ (for some c)

Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

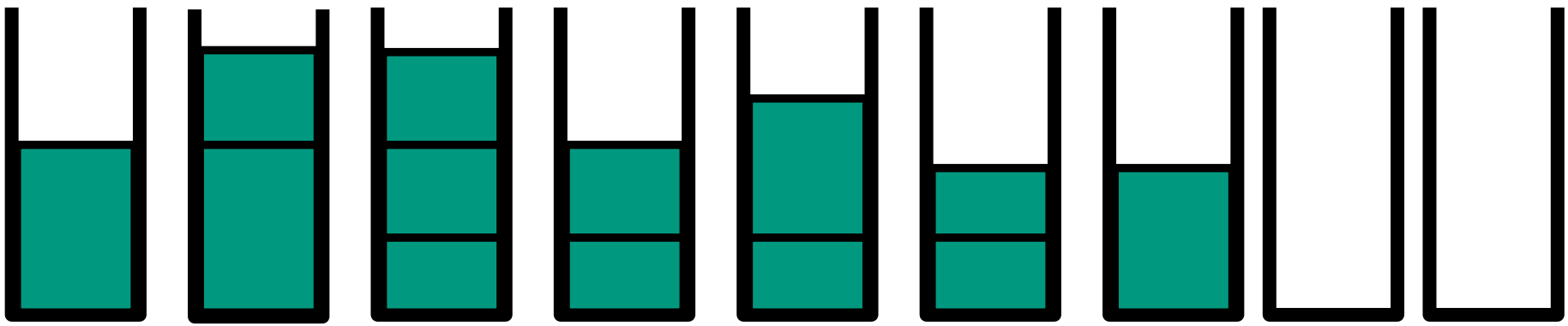
$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

After packing of large items ($> \epsilon/2$)



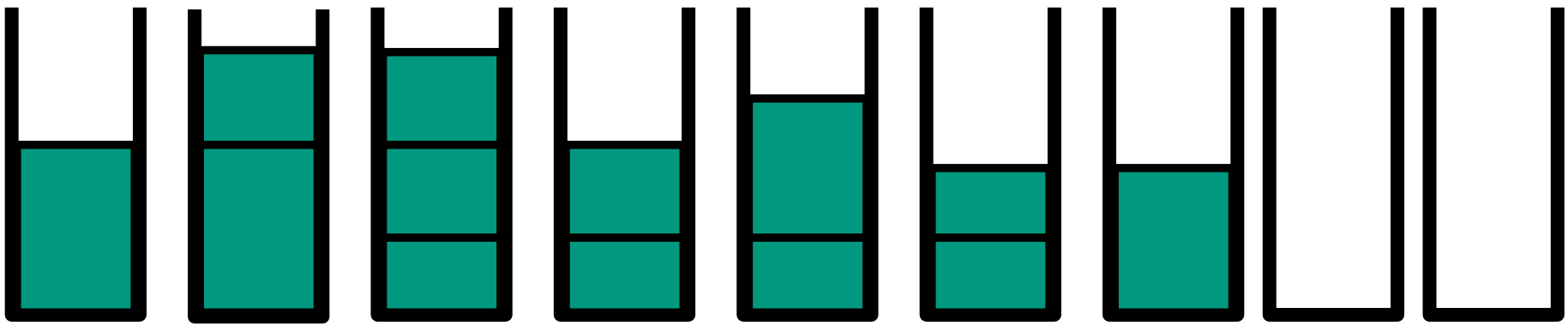
Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

After packing of large items ($> \epsilon/2$)

we pack the small items ($\leqslant \epsilon/2$) on top of the large items greedily



Removing the small items

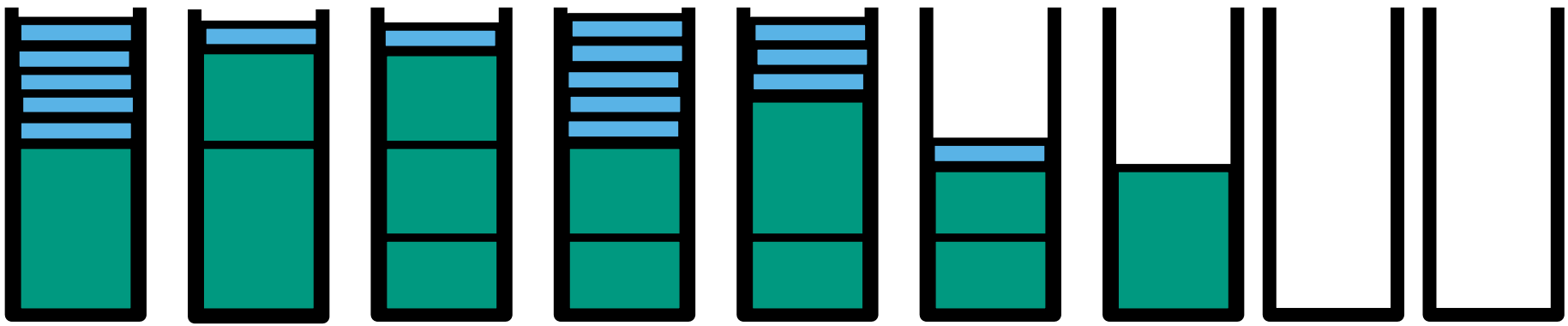
Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins,

in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

After packing of large items ($> \epsilon/2$)

we pack the small items ($\leqslant \epsilon/2$) on top of the large items greedily



Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins,

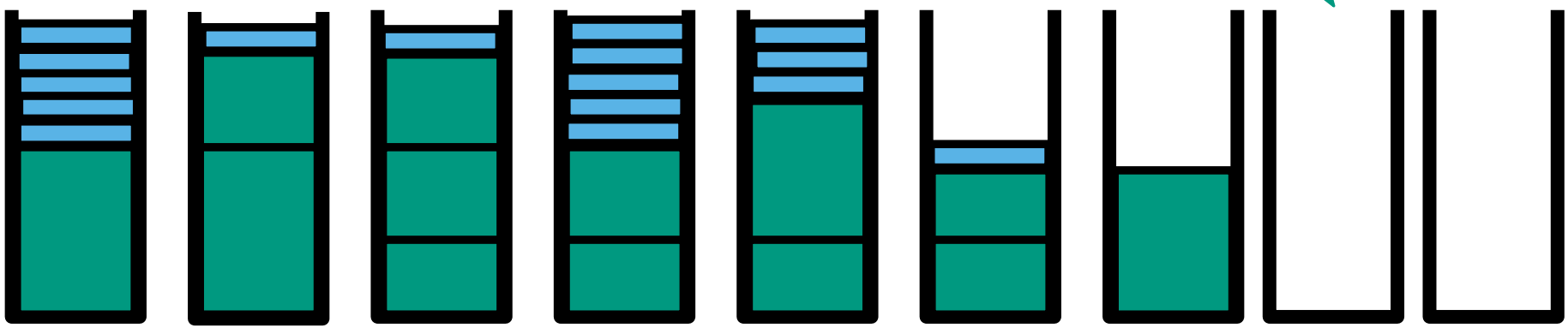
in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

pack the small items anywhere you like - just but don't use an extra bin unless you have to

After packing of large items ($> \epsilon/2$)

we pack the small items ($\leqslant \epsilon/2$) on top of the large items greedily



Removing the small items

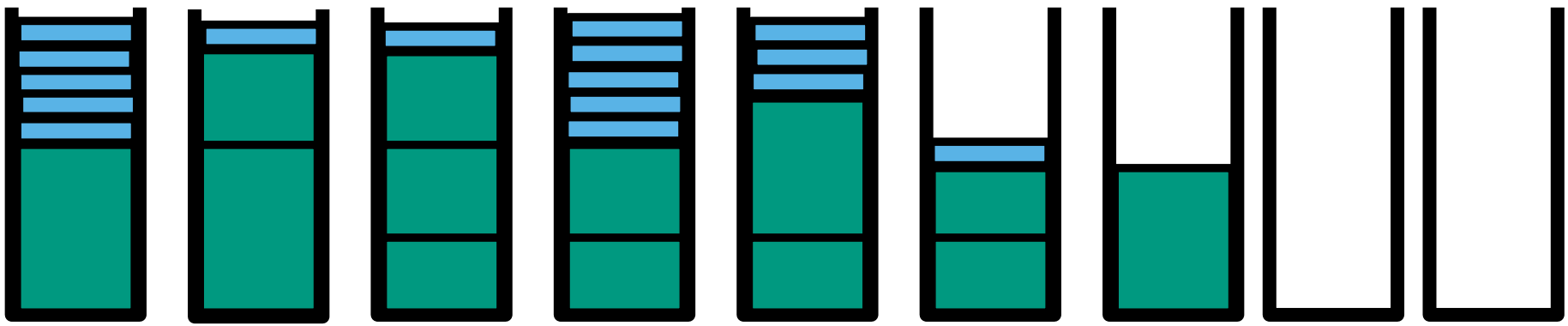
Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins,

in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

After packing of large items ($> \epsilon/2$)

we pack the small items ($\leqslant \epsilon/2$) on top of the large items greedily



Removing the small items

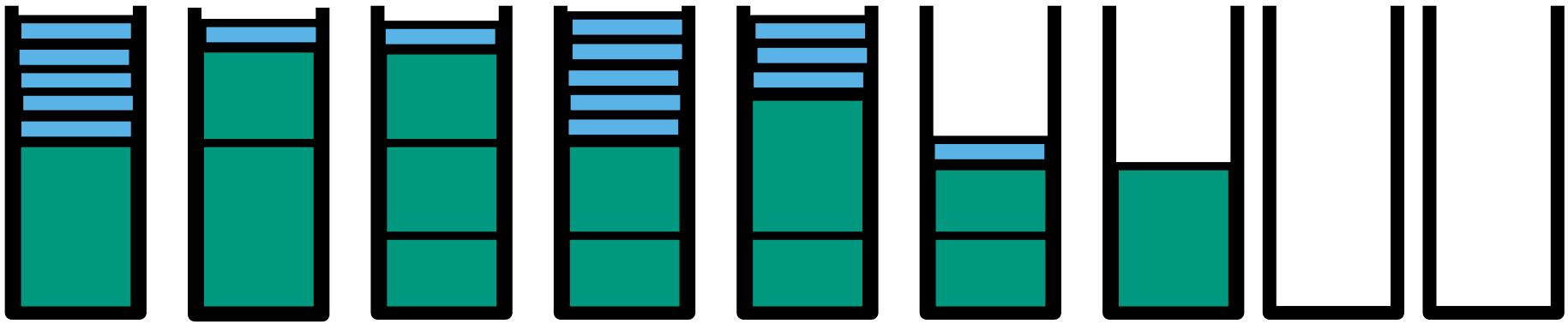
Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins,

in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

After packing of large items ($> \epsilon/2$)

we pack the small items ($\leqslant \epsilon/2$) on top of the large items greedily



Either: We don't use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full

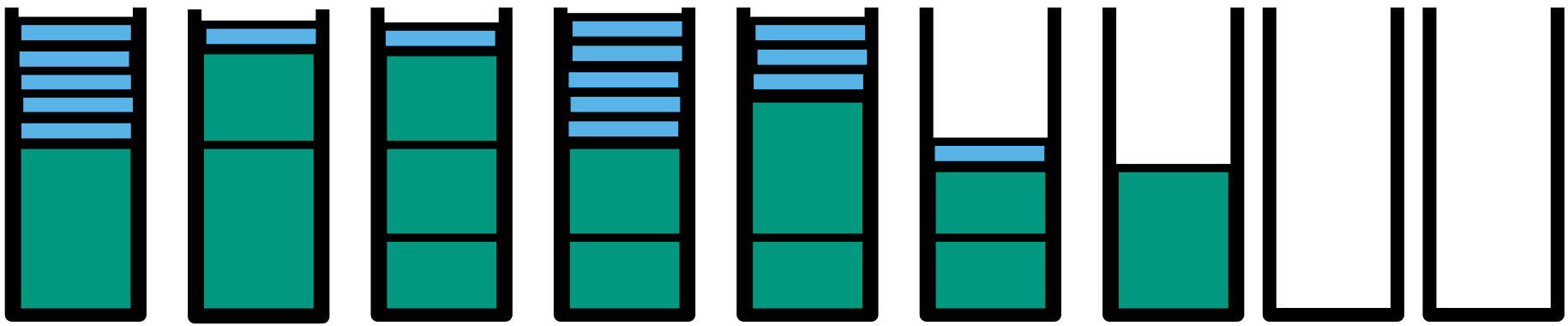
Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

After packing of large items ($> \epsilon/2$)

we pack the small items ($\leqslant \epsilon/2$) on top of the large items greedily



Either: We don't use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full

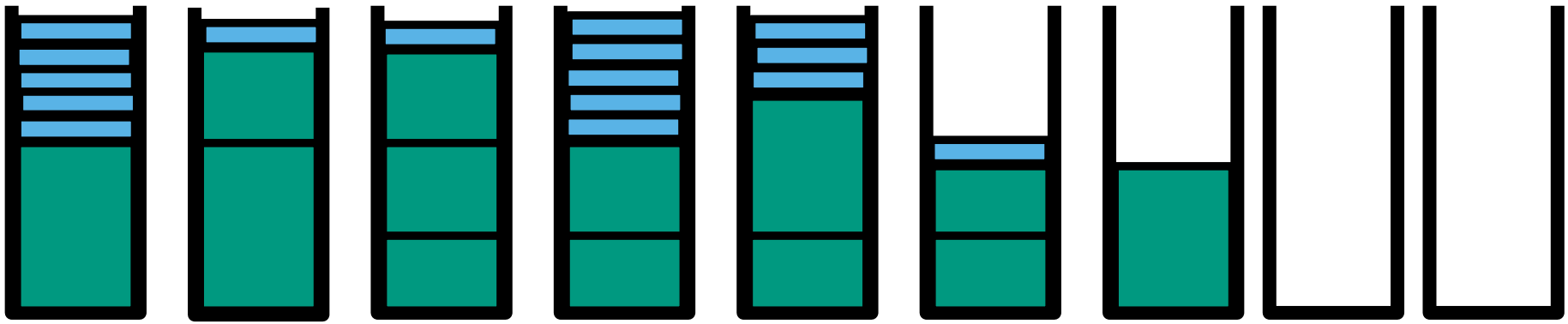
Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

After packing of large items ($> \epsilon/2$)

we pack the small items ($\leqslant \epsilon/2$) on top of the large items greedily



Either: We don't use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full

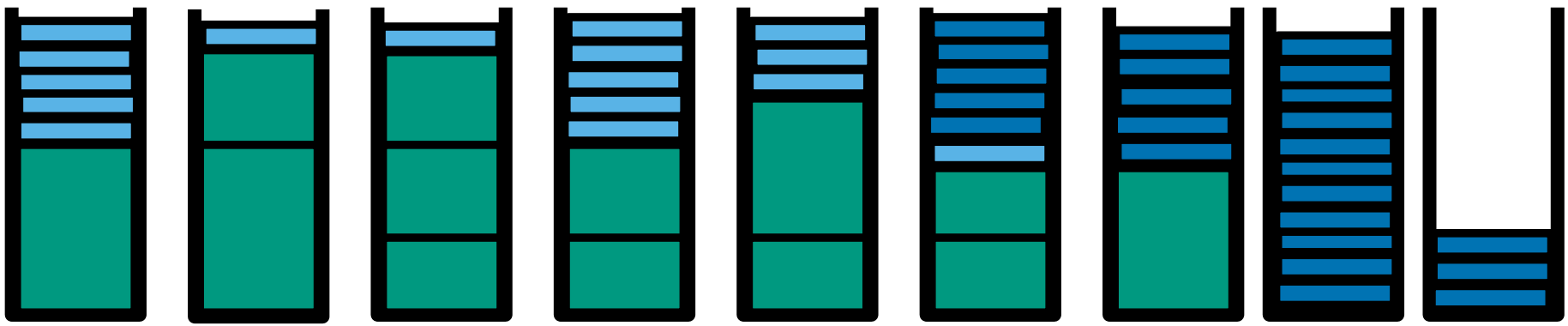
Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

After packing of large items ($> \epsilon/2$)

we pack the small items ($\leqslant \epsilon/2$) on top of the large items greedily



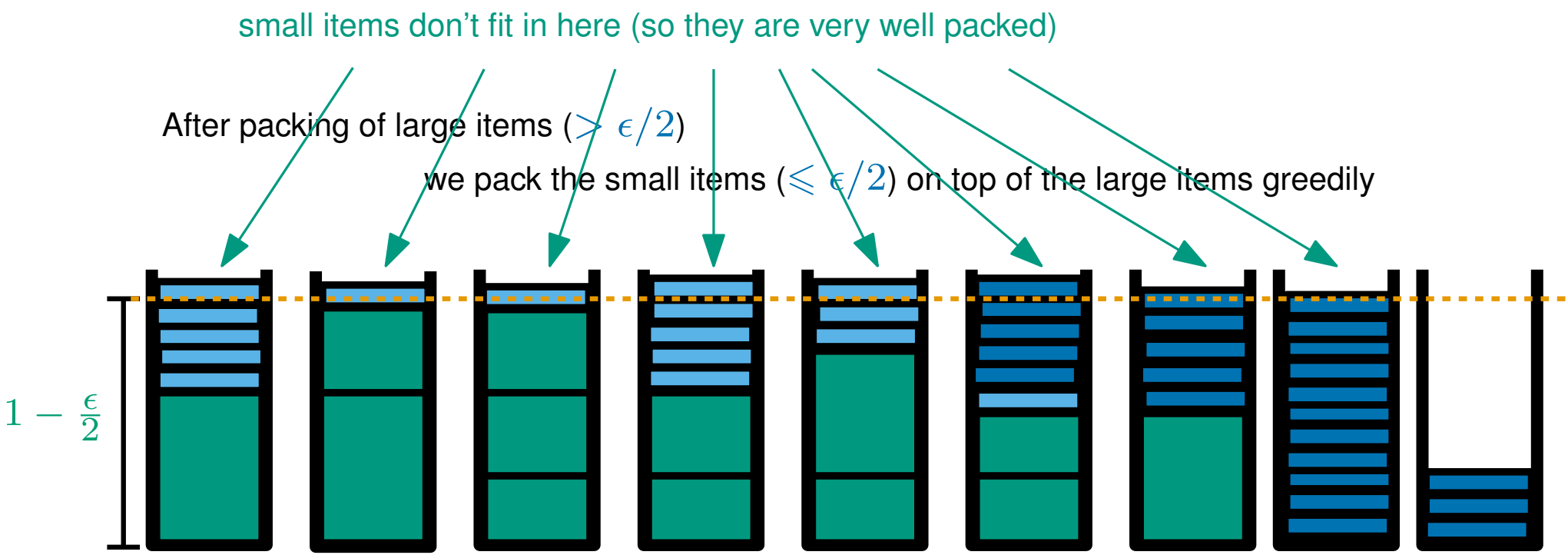
Either: We don't use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full

Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$



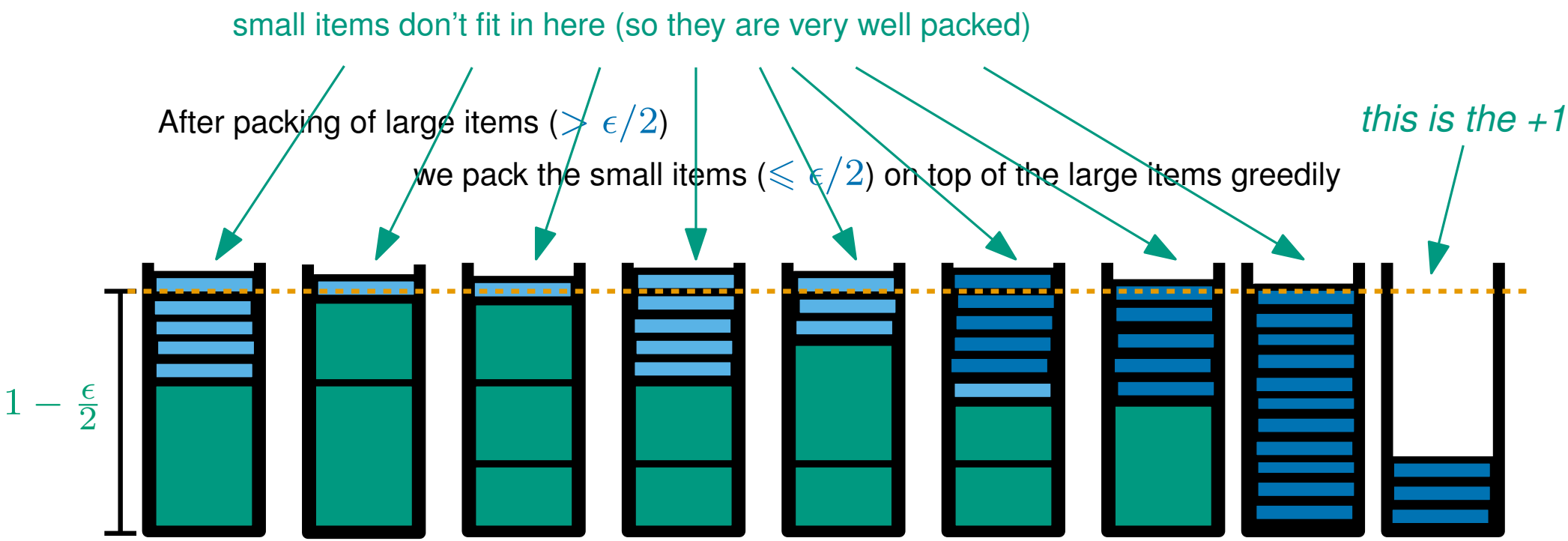
Either: We don't use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full

Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$



Either: We don't use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full

Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

We can now ignore all the small items

and focus on finding a good packing of the large items

Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

We can now ignore all the small items

and focus on finding a good packing of the large items

How many large items fit in a single bin?

Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

We can now ignore all the small items

and focus on finding a good packing of the large items

How many large items fit in a single bin?

Each large item is larger than $\epsilon/2 \ldots$

Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

$$b \text{ bins} \quad \textit{or} \quad (1+\epsilon)\mathrm{Opt} + 1 \text{ bins}$$

We can now ignore all the small items

and focus on finding a good packing of the large items

How many large items fit in a single bin?

Each large item is larger than $\epsilon/2\ldots$

so at most $2/\epsilon$

Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into b bins, in *polynomial time* we can find a packing of *all items* in S which either uses:

b bins *or* $(1+\epsilon)\mathrm{Opt} + 1$ bins

We can now ignore all the small items

and focus on finding a good packing of the large items

How many large items fit in a single bin?

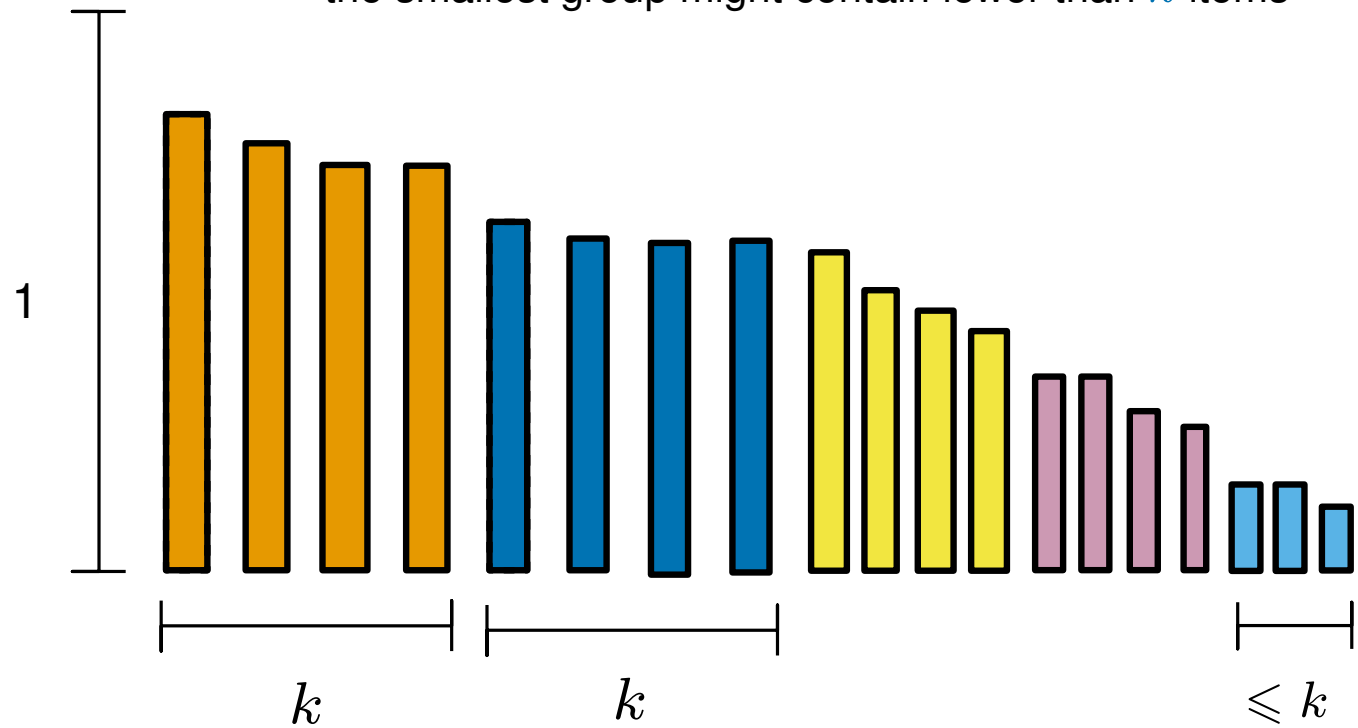
Each large item is larger than $\epsilon/2\ldots$

so at most $2/\epsilon$ which is a constant :)

Reducing the number of item sizes

- We divide the items by size, into groups of size k

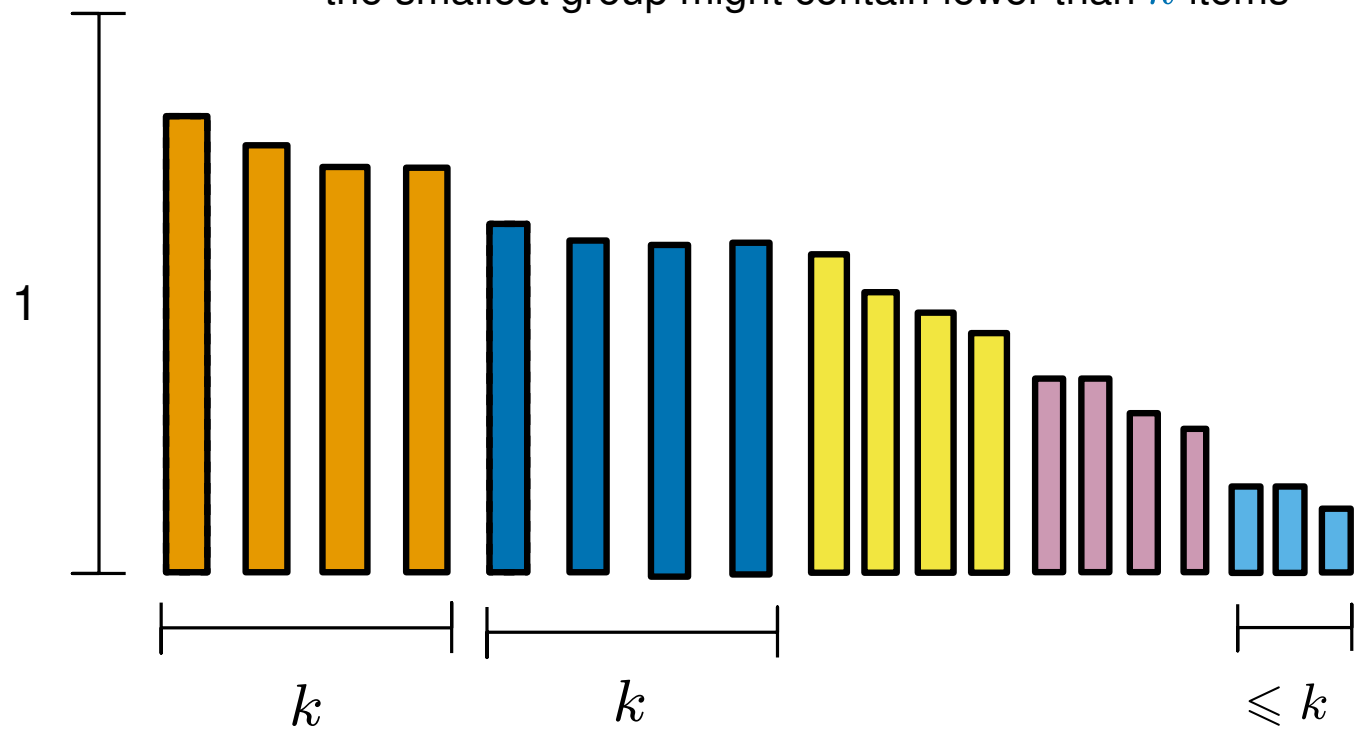
the smallest group might contain fewer than k items



Reducing the number of item sizes

- We divide the items by size, into groups of size k

  the smallest group might contain fewer than k items



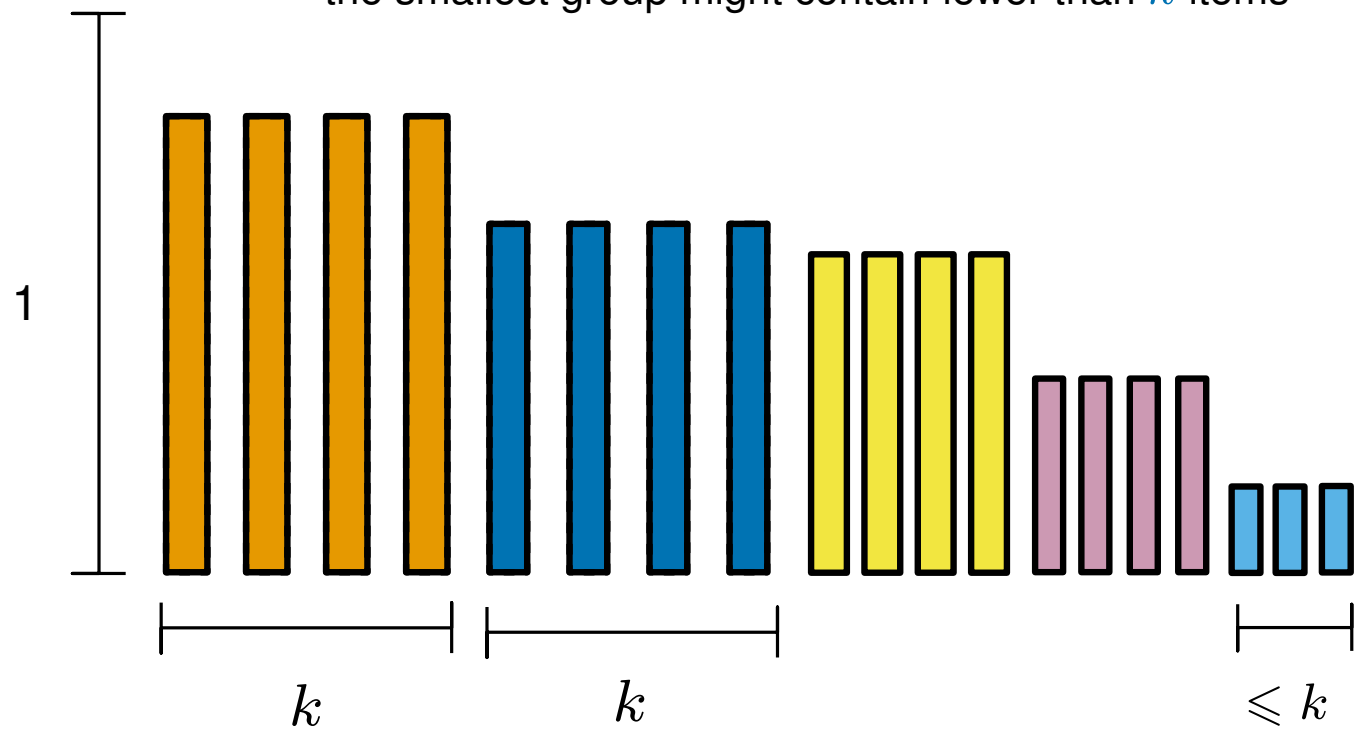
- We define a new set of items S' where each item is rounded up

  so each item in a group has the same size

Reducing the number of item sizes

- We divide the items by size, into groups of size k

  the smallest group might contain fewer than k items



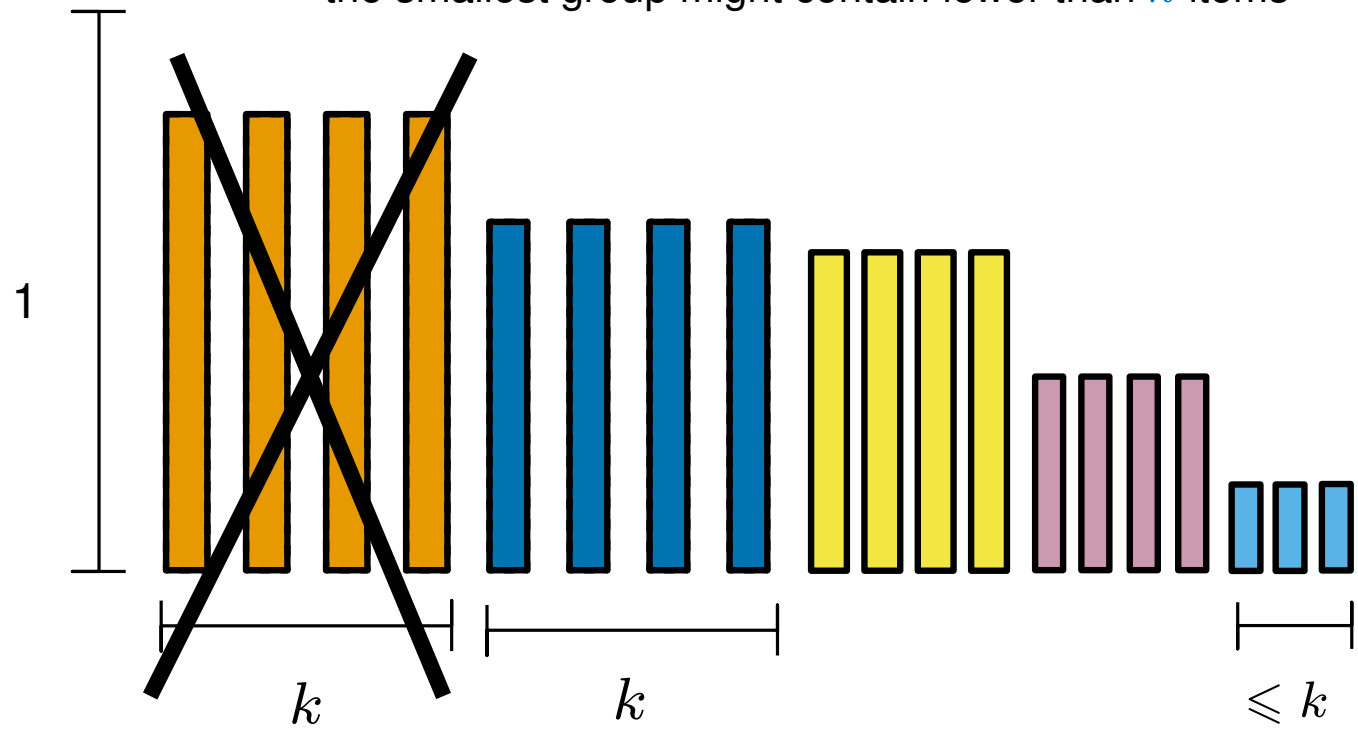
- We define a new set of items S' where each item is rounded up

  so each item in a group has the same size

Reducing the number of item sizes

- We divide the items by size, into groups of size k

  the smallest group might contain fewer than k items



- We define a new set of items S' where each item is rounded up

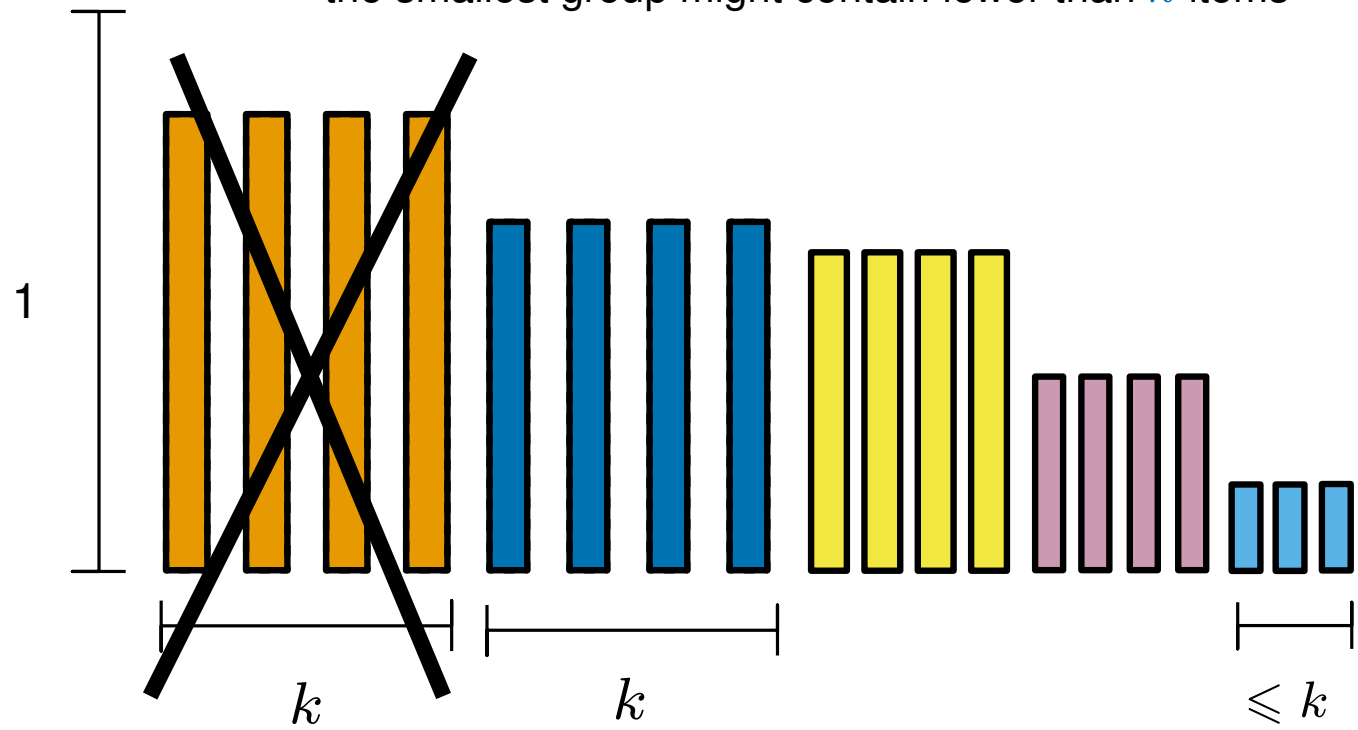
  so each item in a group has the same size

  and the largest group is removed

Reducing the number of item sizes

- We divide the items by size, into groups of size k

  the smallest group might contain fewer than k items



- We define a new set of items S' where each item is rounded up

  so each item in a group has the same size

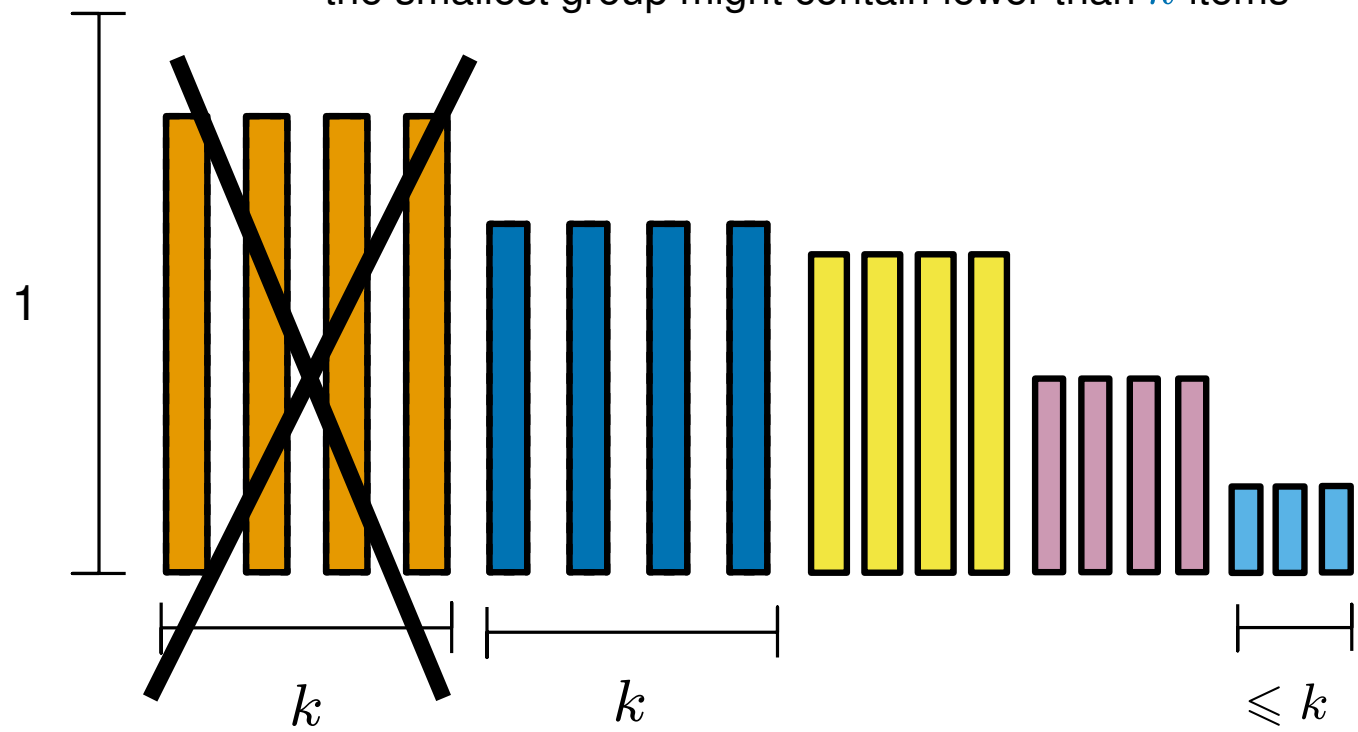
  and the largest group is removed

- Notice that S' contains only n/k distinct item sizes

Reducing the number of item sizes

- We divide the items by size, into groups of size k

  the smallest group might contain fewer than k items



- We define a new set of items S' where each item is rounded up

  so each item in a group has the same size

  and the largest group is removed

- Notice that S' contains only n/k distinct item sizes

  (k will be big enough so that $n/k \leqslant 4/\epsilon^2$ - which is a constant)

Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

. Further, any packing of S' can be converted into a packing of S in polynomial time using at most k extra bins

Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

. Further, any packing of S' can be converted into a packing of S in polynomial time using at most k extra bins

Here $\mathrm{Opt}(S)$ is the optimal number of bins required to pack S

Similarly $\mathrm{Opt}(S')$ is the optimal number of bins required to pack S'

Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

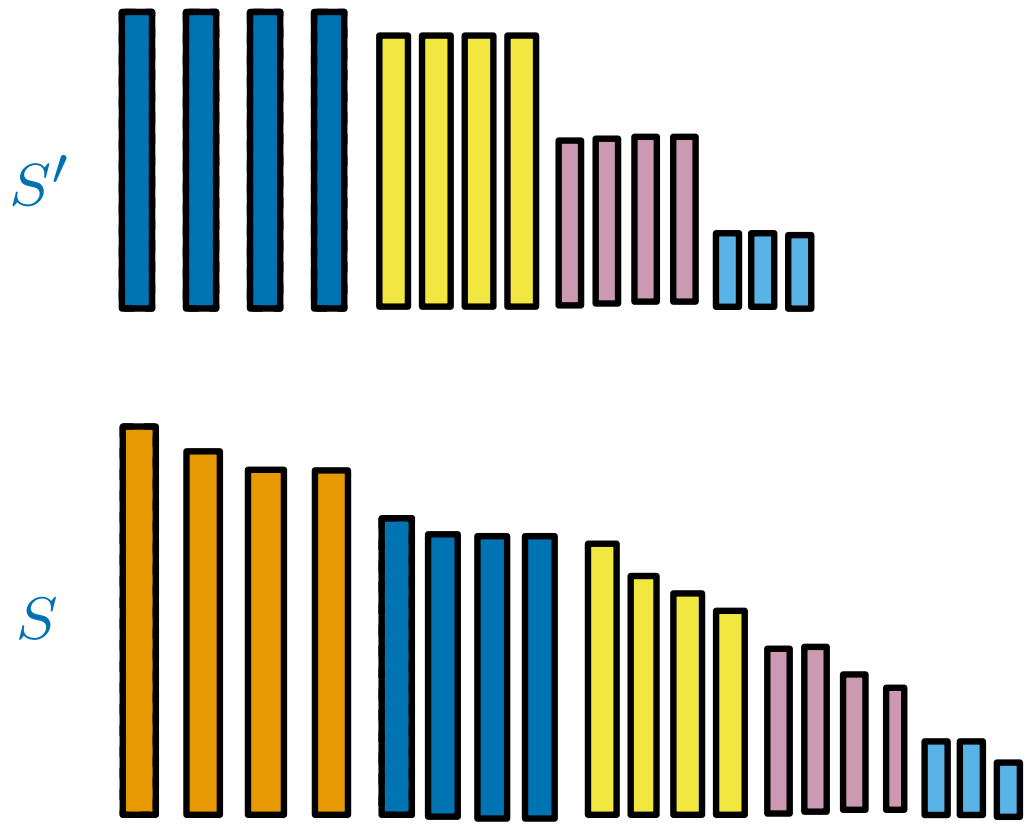
$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

. Further, any packing of S' can be converted into a packing of S in polynomial time using at most k extra bins

Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

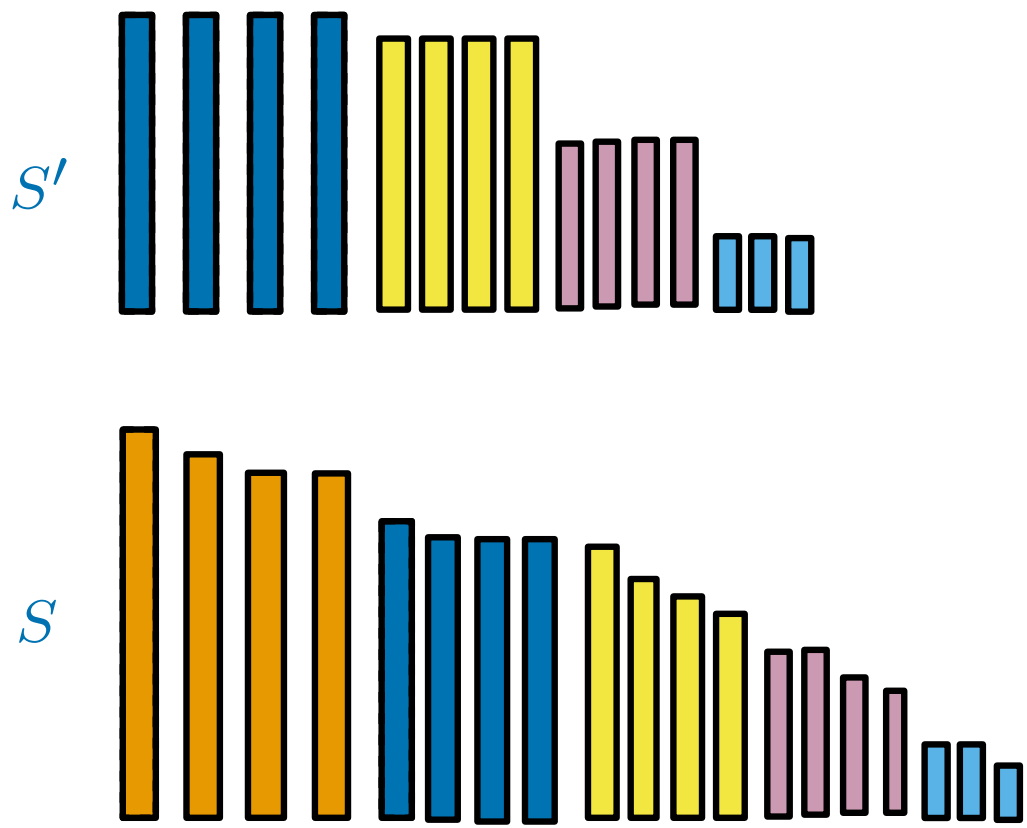


Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof



Reducing the number of item sizes

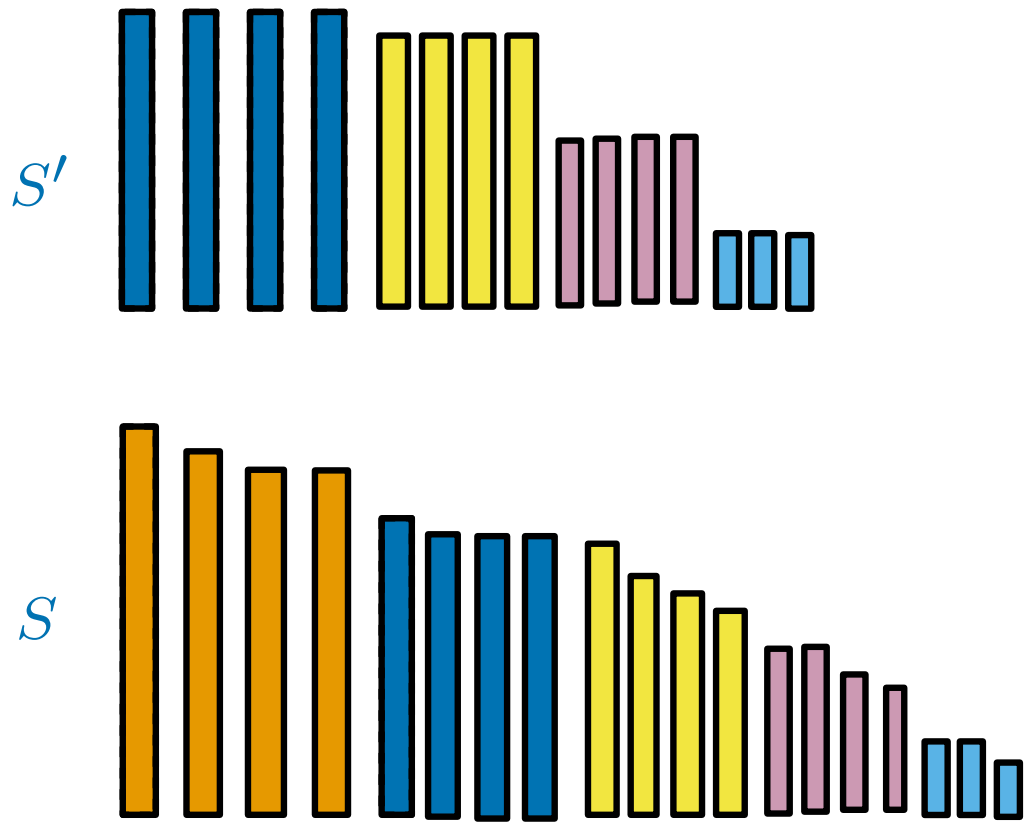
Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

If you can pack S into b bins,

you can pack S' into b bins



Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

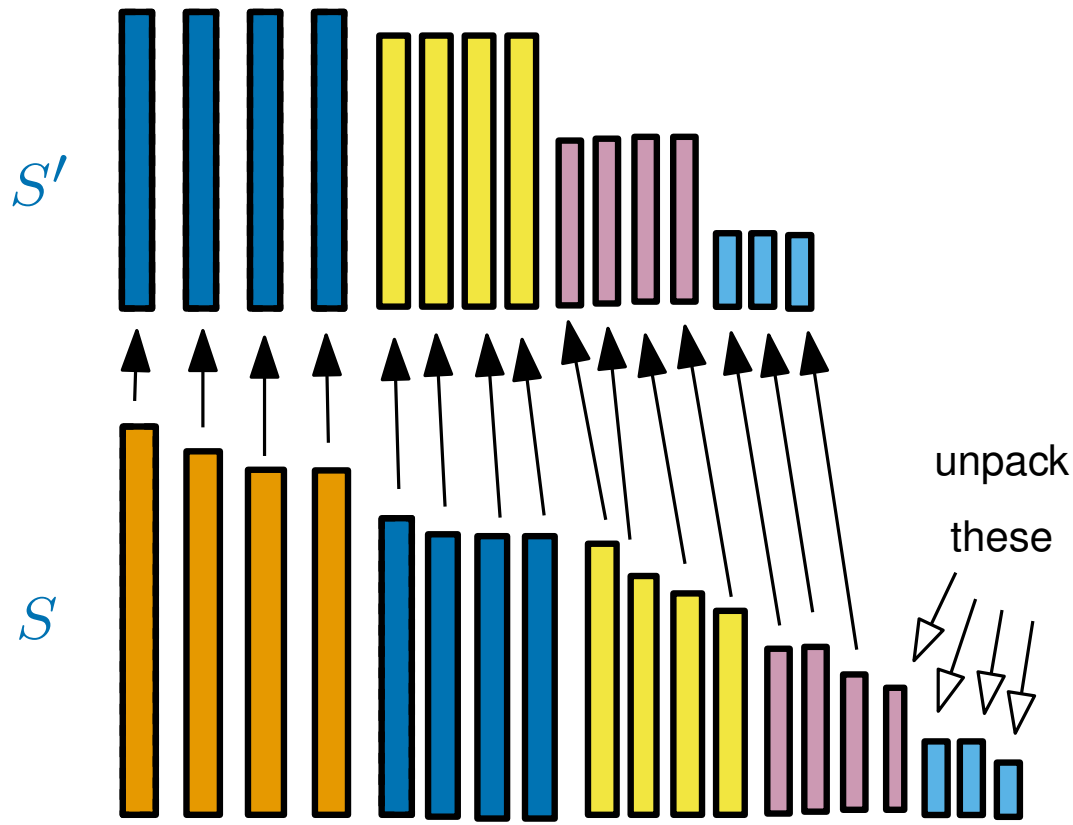
Proof

If you can pack S into b bins,

you can pack S' into b bins

Take the packing of S

and replace each item as shown



Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

If you can pack S into b bins,

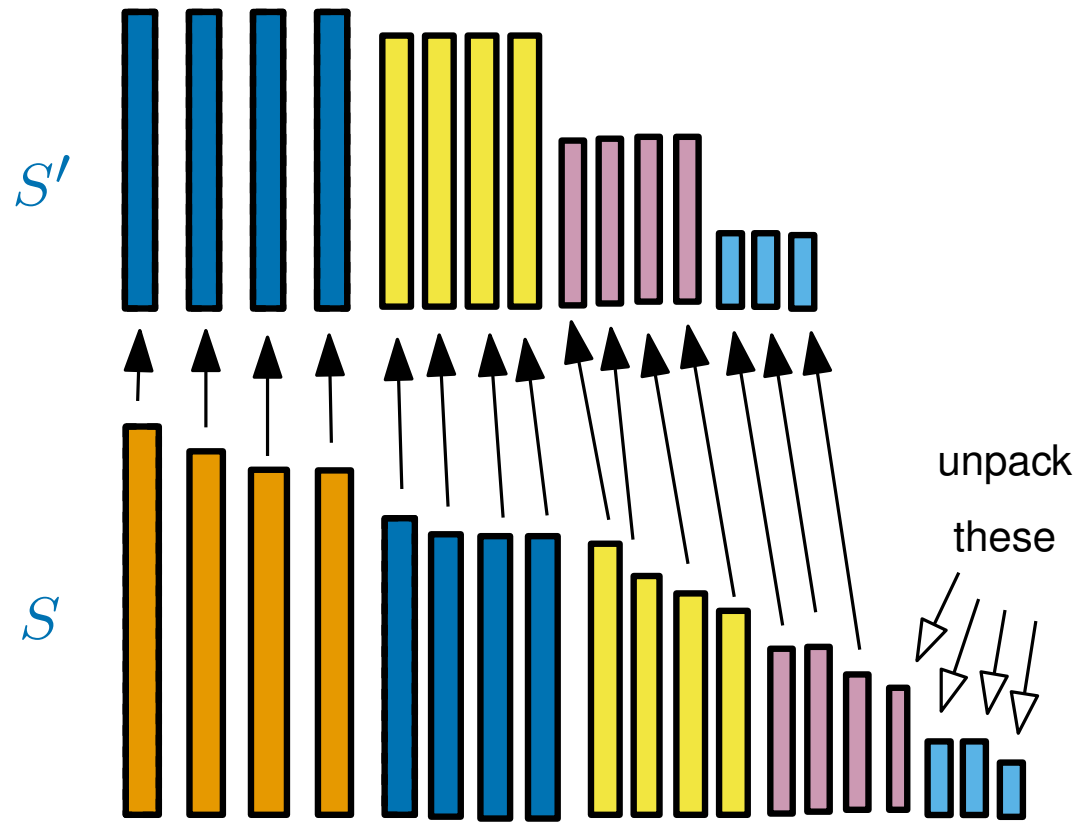
you can pack S' into b bins

Take the packing of S

and replace each item as shown

Each item from S is replaced

with one no larger from S'



Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

If you can pack S into b bins,

you can pack S' into b bins

Take the packing of S

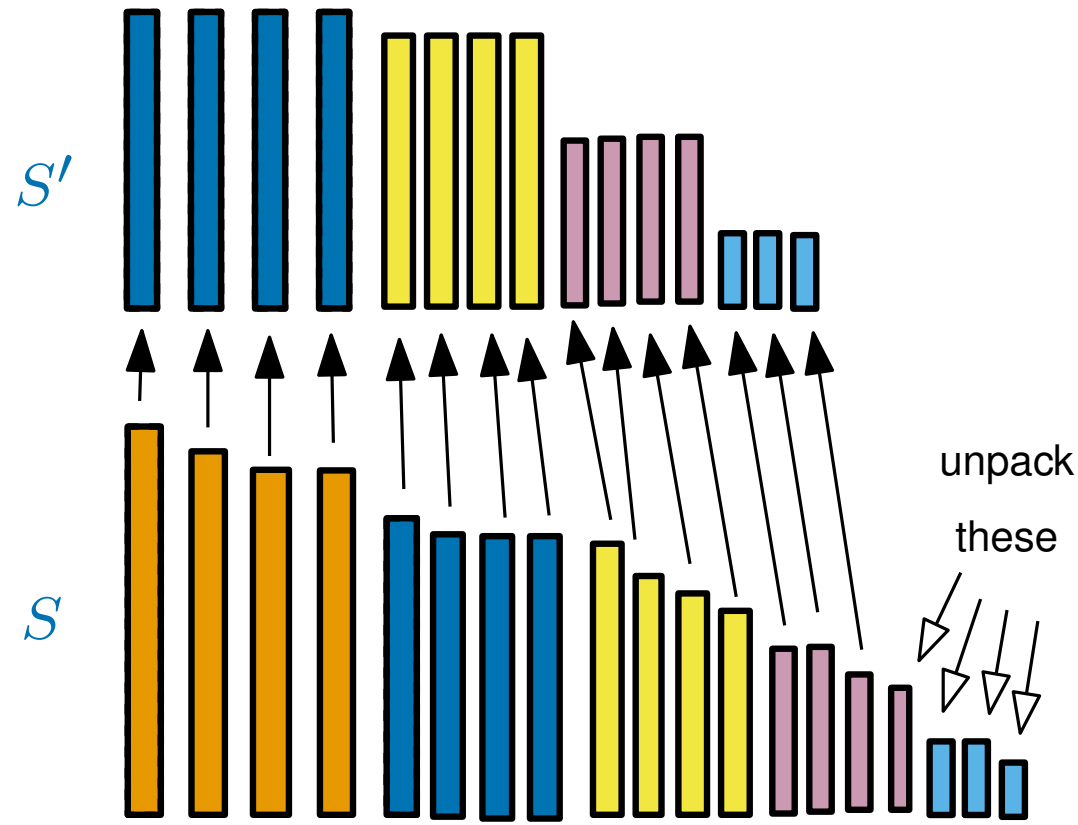
and replace each item as shown

Each item from S is replaced

with one no larger from S'

So the packing is valid and hence

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S)$$



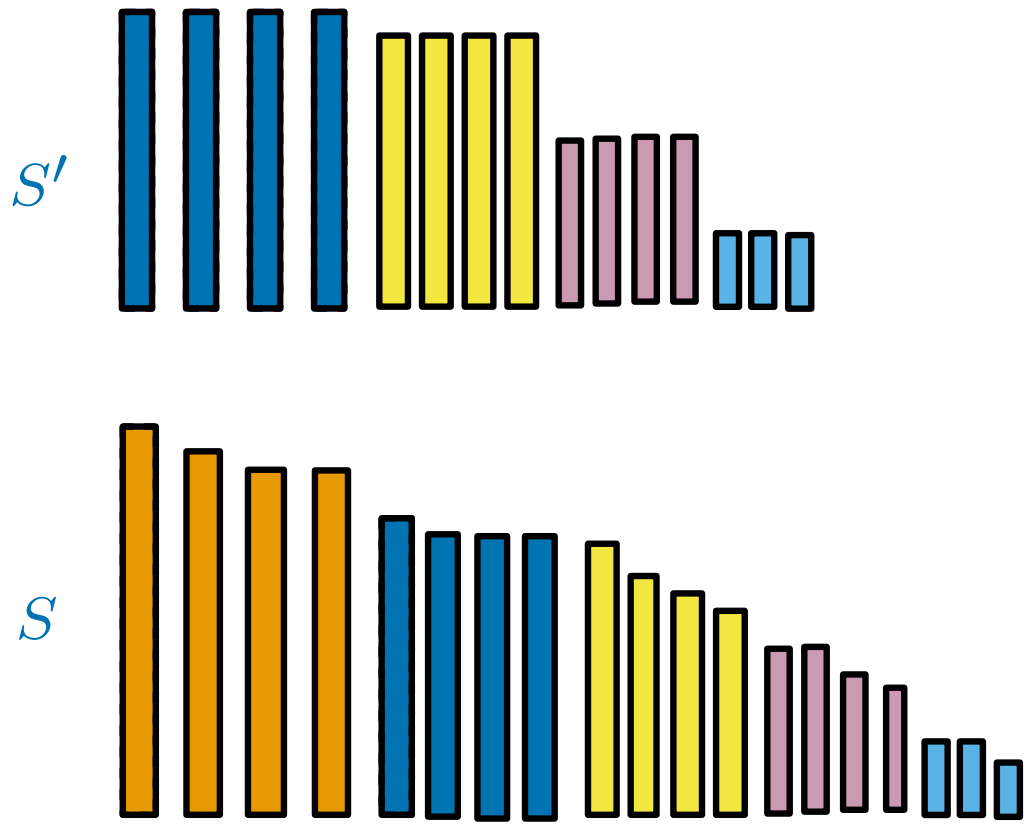
Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

If you can pack S' into b bins,
you can pack S into $b + k$ bins



Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

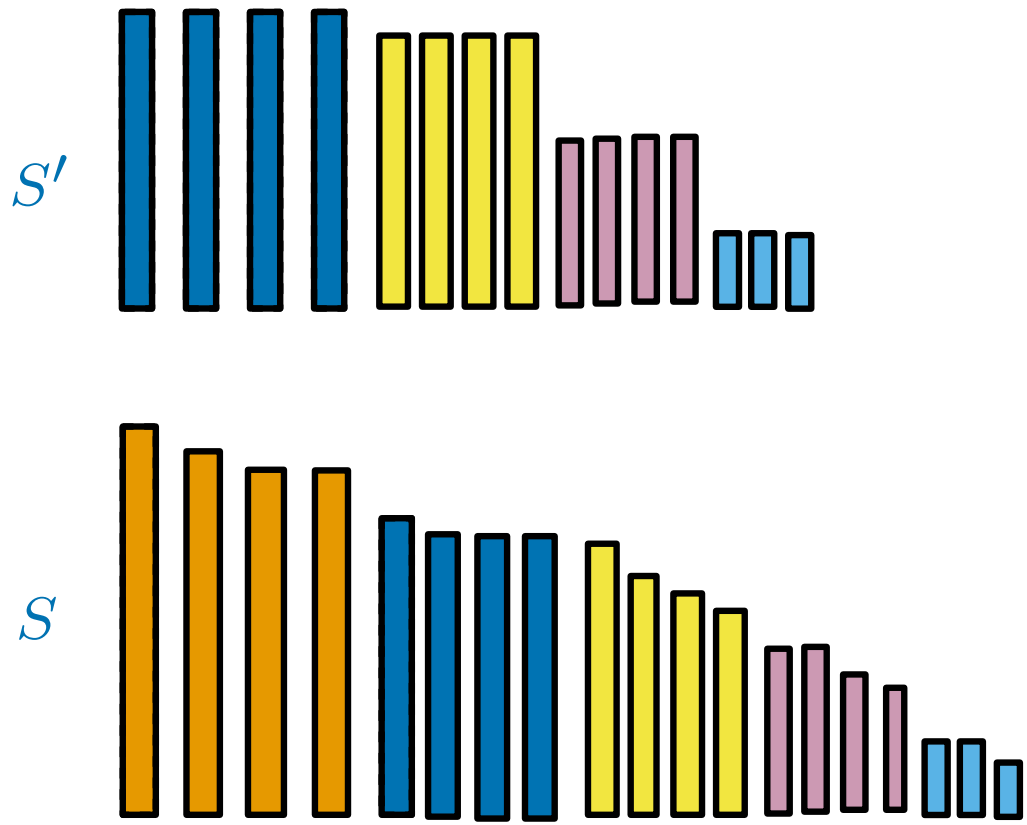
$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

If you can pack S' into b bins,
you can pack S into $b + k$ bins

Take the packing of S'

and replace each item as shown



Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

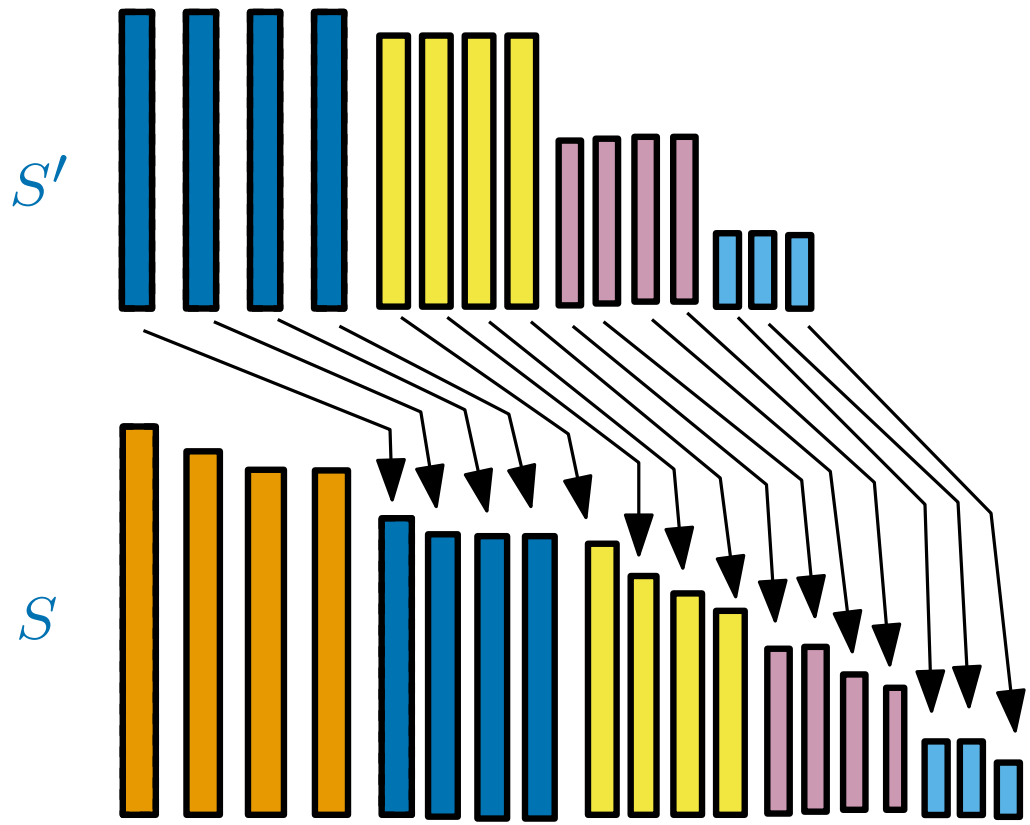
$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

If you can pack S' into b bins,
you can pack S into $b + k$ bins

Take the packing of S'

and replace each item as shown



Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

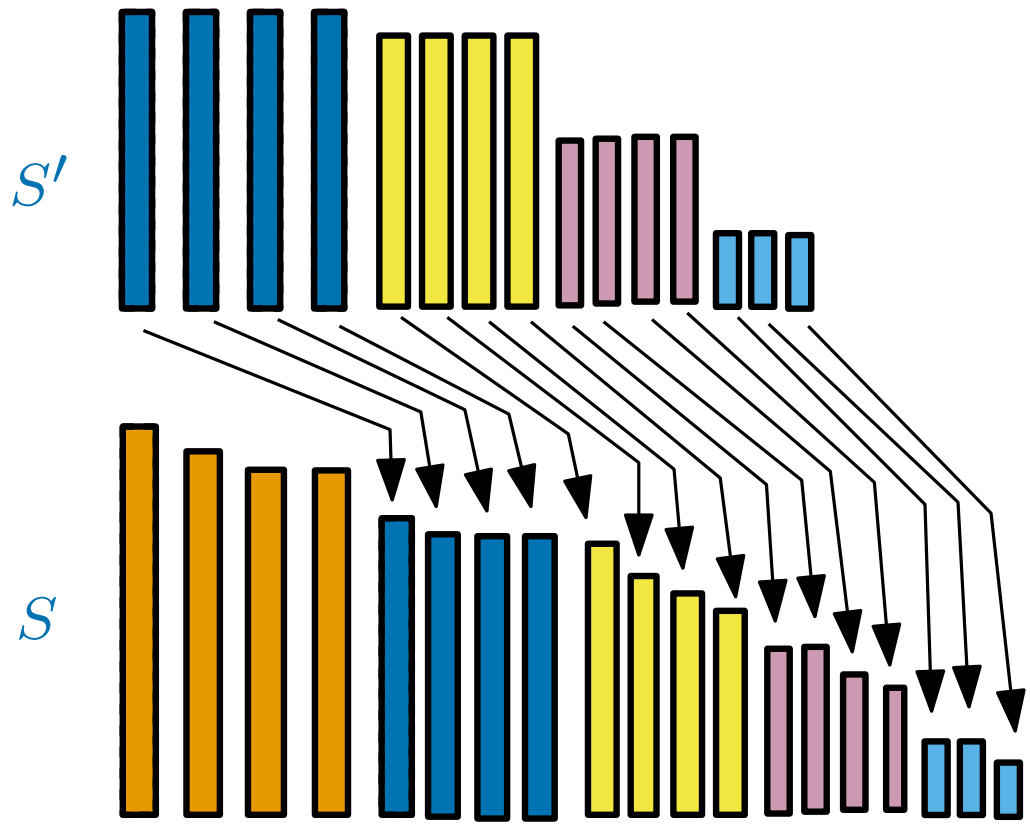
If you can pack S' into b bins,
you can pack S into $b + k$ bins

Take the packing of S'

and replace each item as shown

Each item from S' is replaced

with one no larger from S



Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

If you can pack S' into b bins,
you can pack S into $b + k$ bins

Take the packing of S'

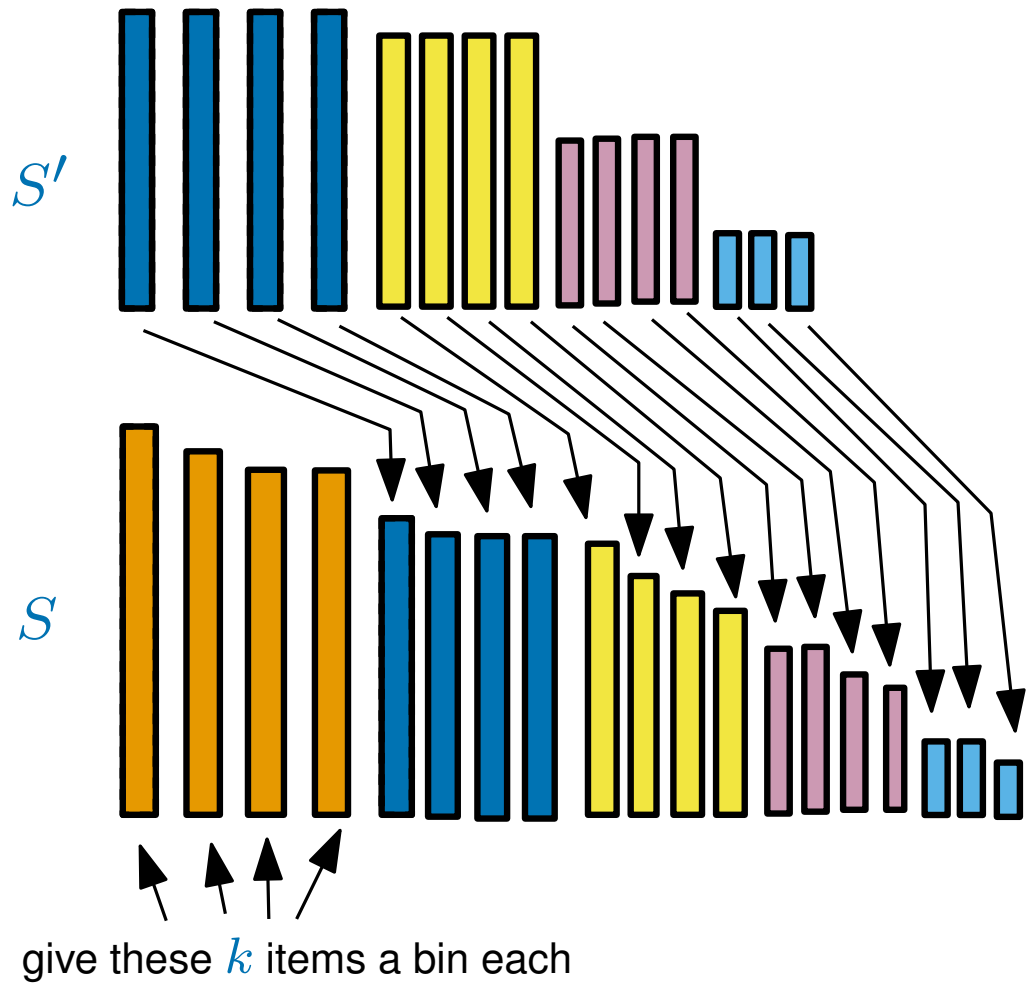
and replace each item as shown

Each item from S' is replaced

with one no larger from S

The k largest items are

given their own extra bins



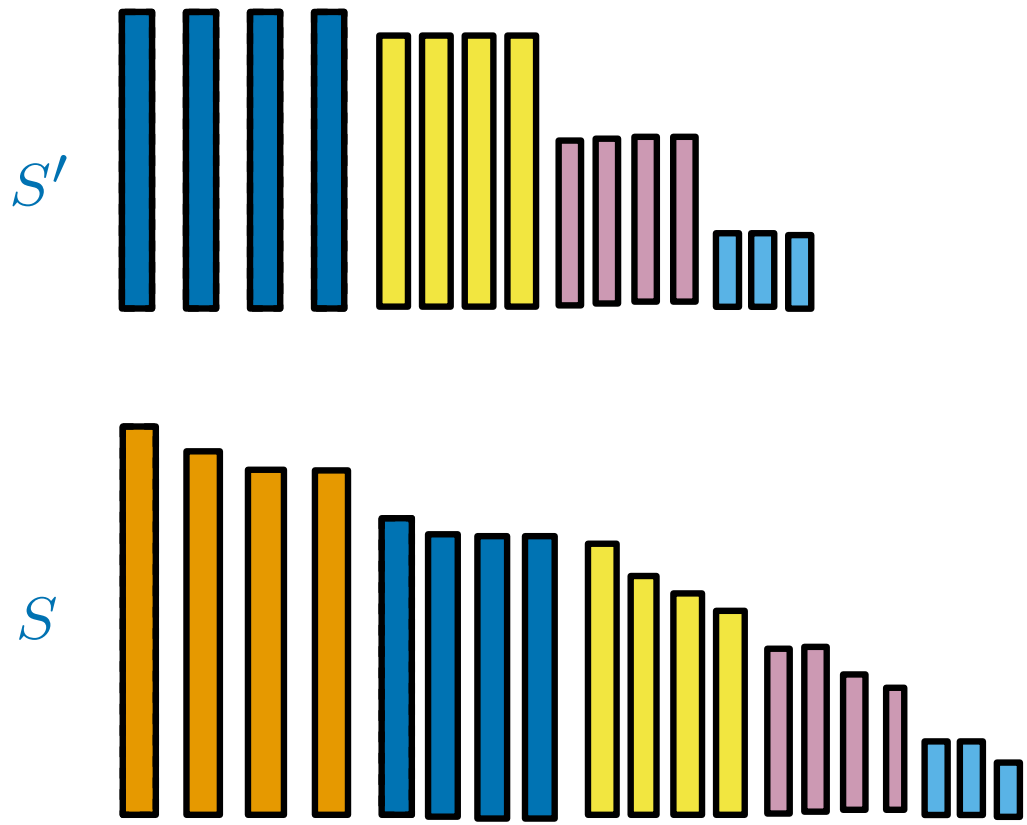
Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

If you can pack S' into b bins,
you can pack S into $b + k$ bins



Reducing the number of item sizes

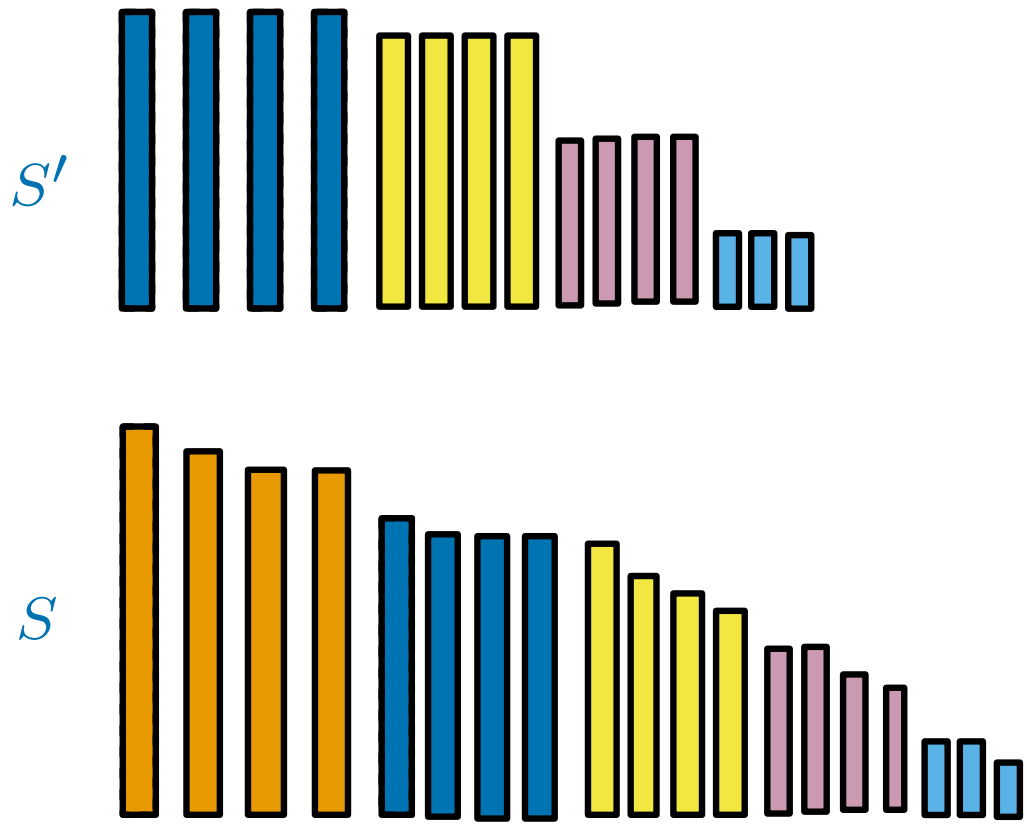
Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

If you can pack S' into b bins,
you can pack S into $b + k$ bins

This gives a valid packing of S



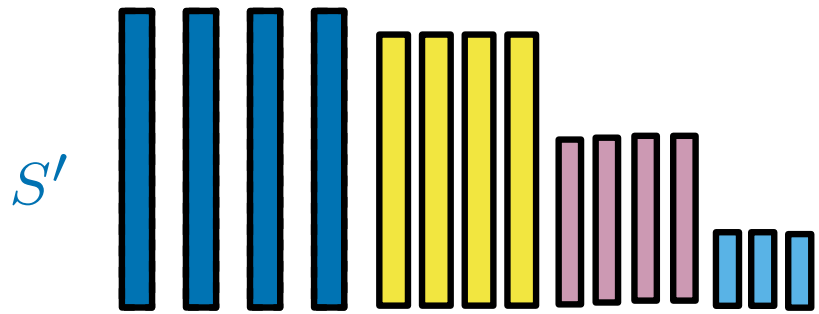
Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

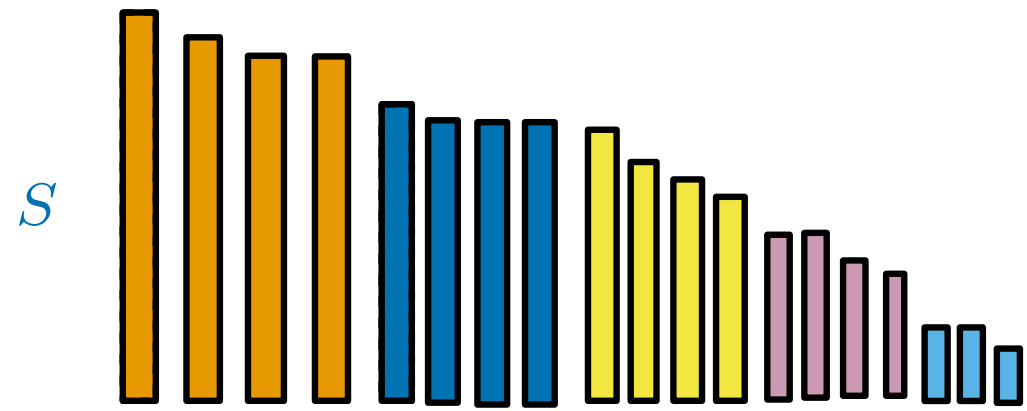
Proof

If you can pack S' into b bins,
you can pack S into $b + k$ bins



This gives a valid packing of S

Hence, $\mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k$



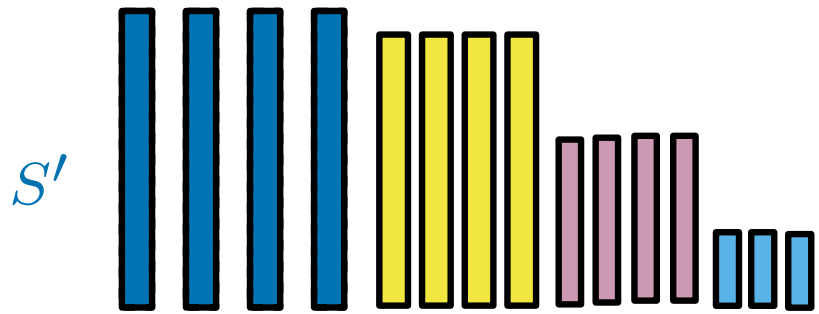
Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

Proof

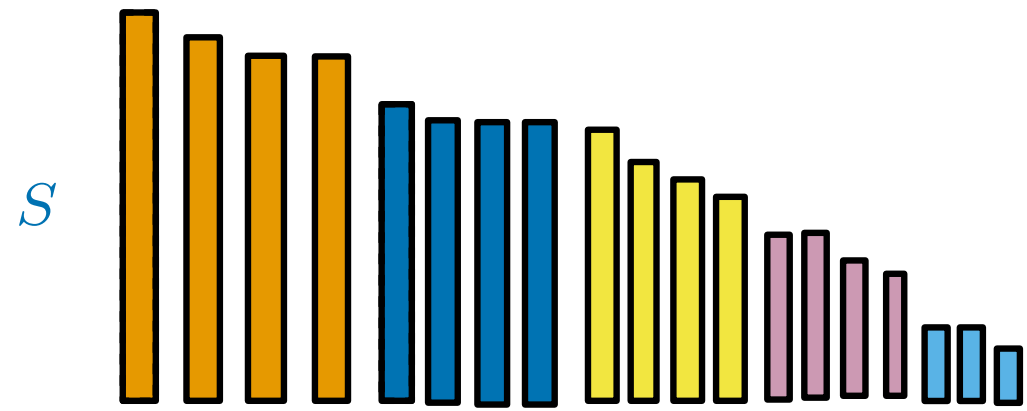
If you can pack S' into b bins,
you can pack S into $b + k$ bins



This gives a valid packing of S

Hence, $\mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k$

Note that both transformations
take polynomial time



Reducing the number of item sizes

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

. Further, any packing of S' can be converted into a packing of S in polynomial time using at most k extra bins

Reducing the number of item sizes

$n = |S|$

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

. Further, any packing of S' can be converted into a packing of S in polynomial time using at most k extra bins

We set $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ and let S be the set of large items which implies that...

Reducing the number of item sizes

$n = |S|$

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

. Further, any packing of S' can be converted into a packing of S in polynomial time using at most k extra bins

We set $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ and let S be the set of large items which implies that...

$$k \leqslant \epsilon \cdot \mathrm{Opt}(S)$$

Reducing the number of item sizes

$n = |S|$

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

. Further, any packing of S' can be converted into a packing of S in polynomial time using at most k extra bins

We set $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ and let S be the set of large items which implies that...

$k \leqslant \epsilon \cdot \mathrm{Opt}(S)$ (because each of the n items in S has size at least $\epsilon/2$)

Reducing the number of item sizes

$n = |S|$

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

. Further, any packing of S' can be converted into a packing of S in polynomial time using at most k extra bins

We set $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ and let S be the set of large items which implies that...

$k \leqslant \epsilon \cdot \mathrm{Opt}(S)$ (because each of the n items in S has size at least $\epsilon/2$)

If we can find the optimal packing of S', which uses $\mathrm{Opt}(S')$ bins

we can convert it into a packing of S which uses

$\mathrm{Opt}(S') + k \leqslant (1+\epsilon)\mathrm{Opt}(S)$ bins

Reducing the number of item sizes

$n = |S|$

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

. Further, any packing of S' can be converted into a packing of S in polynomial time using at most k extra bins

We set $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ and let S be the set of large items which implies that...

$k \leqslant \epsilon \cdot \mathrm{Opt}(S)$ (because each of the n items in S has size at least $\epsilon/2$)

If we can find the optimal packing of S', which uses $\mathrm{Opt}(S')$ bins

we can convert it into a packing of S which uses

$\mathrm{Opt}(S') + k \leqslant (1+\epsilon)\mathrm{Opt}(S)$ bins

S' contains $4/\epsilon^2$ distinct item sizes

and only $2/\epsilon$ items fit in each bin...

Reducing the number of item sizes

$n = |S|$

Lemma Let S' be S after linear grouping (with groups of size k).

$$\mathrm{Opt}(S') \leqslant \mathrm{Opt}(S) \leqslant \mathrm{Opt}(S') + k\,.$$

. Further, any packing of S' can be converted into a packing of S in polynomial time using at most k extra bins

We set $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ and let S be the set of large items which implies that...

$k \leqslant \epsilon \cdot \mathrm{Opt}(S)$ (because each of the n items in S has size at least $\epsilon/2$)

If we can find the optimal packing of S', which uses $\mathrm{Opt}(S')$ bins

we can convert it into a packing of S which uses

$\mathrm{Opt}(S') + k \leqslant (1+\epsilon)\mathrm{Opt}(S)$ bins

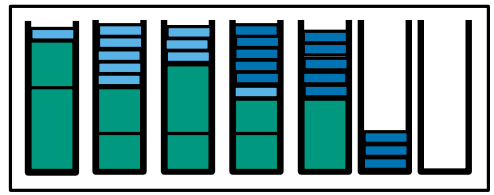
S' contains $4/\epsilon^2$ distinct item sizes

and only $2/\epsilon$ items fit in each bin...

I.e. we *can* optimally pack S' in polynomial time...

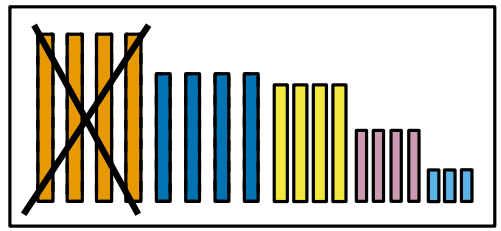
The overall APTAS

Step 1 Remove all the *small* items



Only $c_b = 2/\epsilon$ of the remaining large items will fit into a single bin

Step 2 Divide the items into $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ different groups



Sizes of items in each group are then rounded up

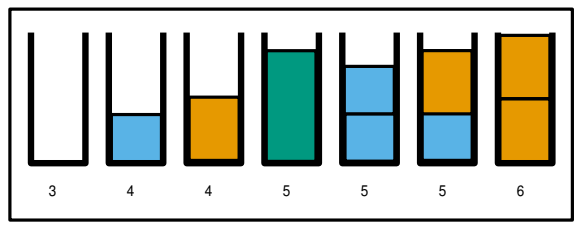
to match the size of the largest member

(and the largest group is removed)

This will leave only $c_s = 4/\epsilon^2$ different item sizes

Step 3 Use the poly-time algorithm to optimally pack the remaining special case

This takes $O\left(n \cdot (n+1)^{(4/\epsilon^2+1)^{2/\epsilon}}\right)$ time



Step 4 Reverse the grouping from **Step 2** and then

greedily pack all the small items removed in **Step 1**

Theorem For any $0 < \epsilon < 1$, the algorithm presented runs in polynomial time and returns a packing of any set of items using at most $(1+\epsilon)\mathrm{Opt} + 1$ bins

Conclusions

- There is no α-approximation for BinPacking with $\alpha < 3/2$ unless $\mathrm{P} = \mathrm{NP}$
- This in turn implies that there is no PTAS for BinPacking unless $\mathrm{P} = \mathrm{NP}$
- The First Fit Decreasing algorithm uses at most,

$$\frac{11}{9} \cdot \mathrm{Opt} + 1 \text{ bins}$$

- We saw an APTAS for BinPacking. (which uses at most $(1+\epsilon) \cdot \mathrm{Opt} + 1$ bins)

there is also an AFPTAS for bin packing

- There is a poly-time algorithm which outputs a solution using at most,

$$\mathrm{Opt} + O(\log^2 \mathrm{Opt}) = \left(1 + \frac{O(\log^2 \mathrm{Opt})}{\mathrm{Opt}}\right) \cdot \mathrm{Opt} \text{ bins}$$

Conclusions

- There is no α-approximation for BinPacking with $\alpha < 3/2$ unless $\mathrm{P} = \mathrm{NP}$
- This in turn implies that there is no PTAS for BinPacking unless $\mathrm{P} = \mathrm{NP}$
- The First Fit Decreasing algorithm uses at most,

$$\frac{11}{9} \cdot \mathrm{Opt} + 1 \text{ bins}$$

- We saw an APTAS for BinPacking. (which uses at most $(1+\epsilon) \cdot \mathrm{Opt} + 1$ bins)

there is also an AFPTAS for bin packing

- There is a poly-time algorithm which outputs a solution using at most,

$$\mathrm{Opt} + O(\log^2 \mathrm{Opt}) = \left(1 + \frac{O(\log^2 \mathrm{Opt})}{\mathrm{Opt}}\right) \cdot \mathrm{Opt} \text{ bins}$$

Whether there is a poly-time algorithm which uses at most $\mathrm{Opt} + 1$ bins is an open problem