Advanced Algorithms – COMS31900

Approximation algorithms part one

Constant factor approximations

Raphaël Clifford

Slides by Benjamin Sach

NP-completeness recap

NP is the class of *decision* problems we can check the answer to in polynomial time

A problem A is NP-complete if

A is in NP

Every B in NP has a polynomial time reduction to A

(this second part is the definition of NP-hard)

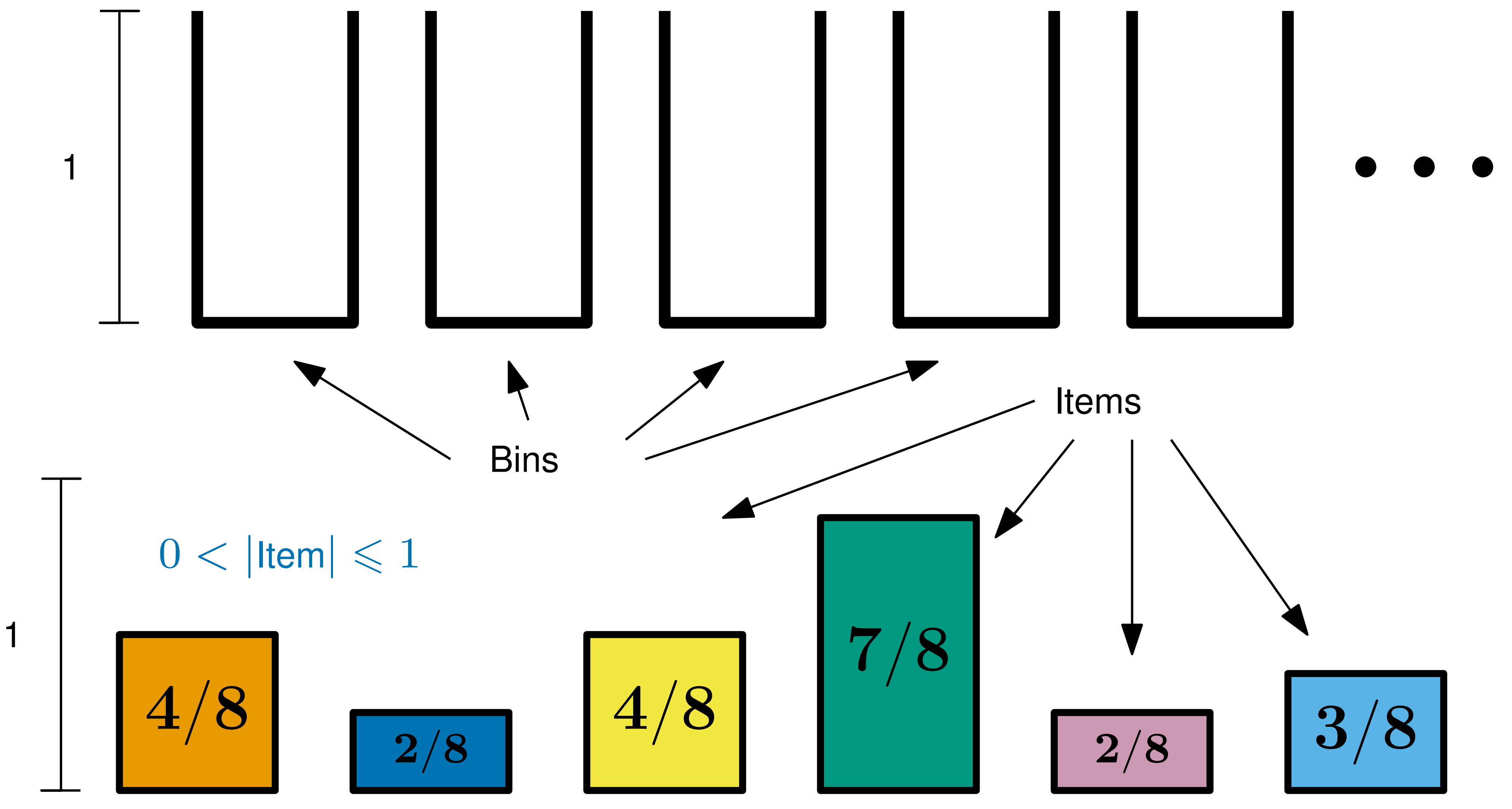
If we could solve A quickly we could solve every problem in NP quickly

They are the 'hardest' problems in NP

So if a problem is NP-complete, we give up right?

Bin packing

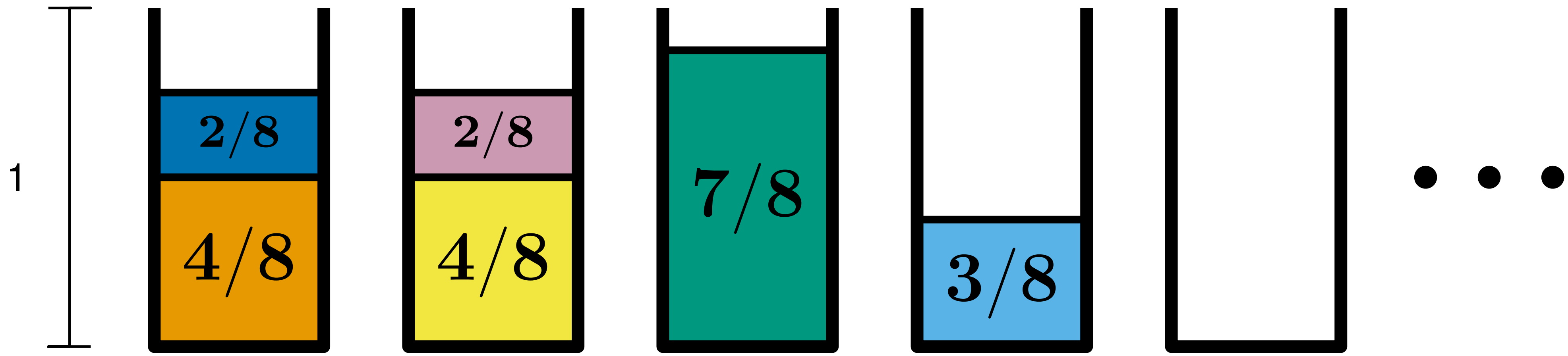
$|\text{Bin}| = 1$ and there is an unlimited number of bins...



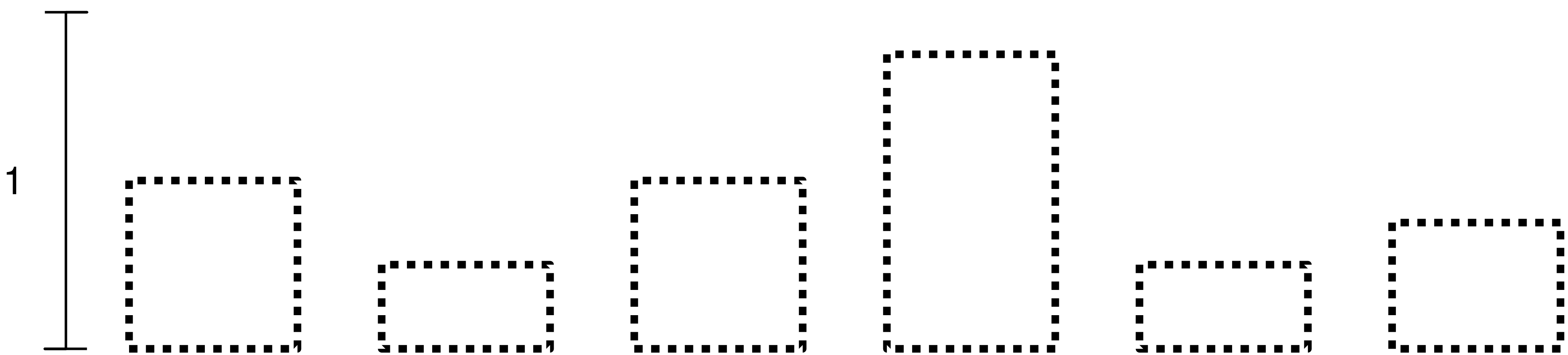
I is the sum of all item sizes

Bin packing

Problem pack all items into the fewest possible bins

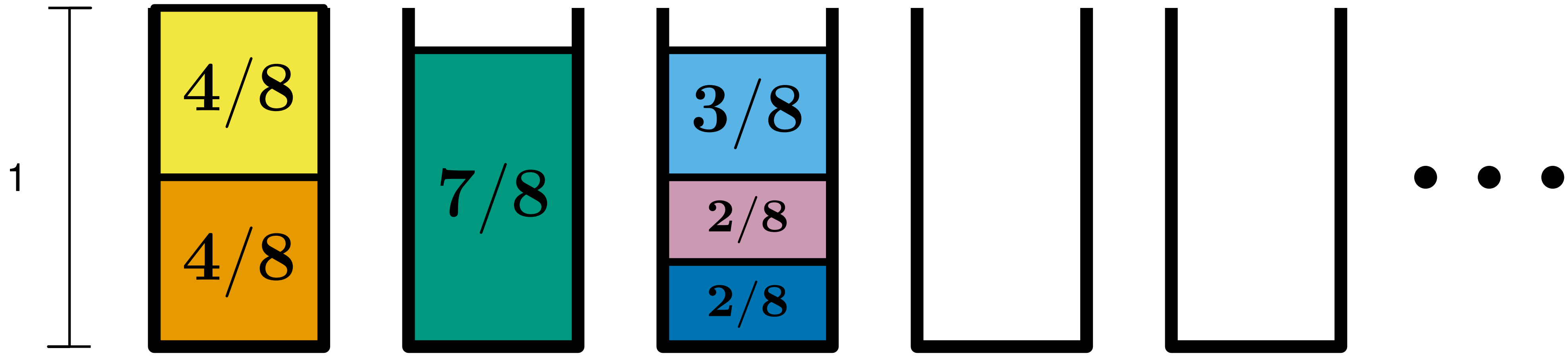


This is an example of an optimisation *problem*

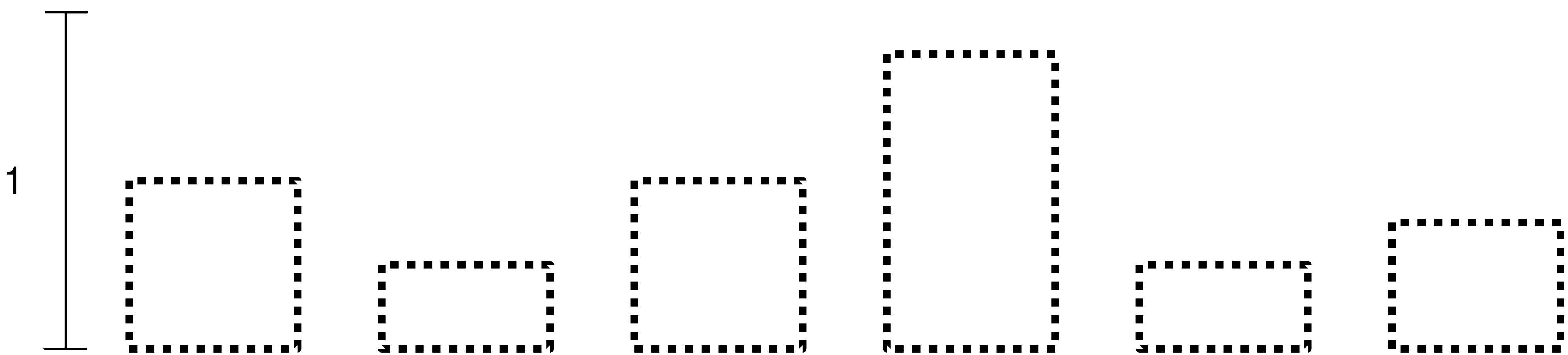


Bin packing

Problem pack all items into the fewest possible bins

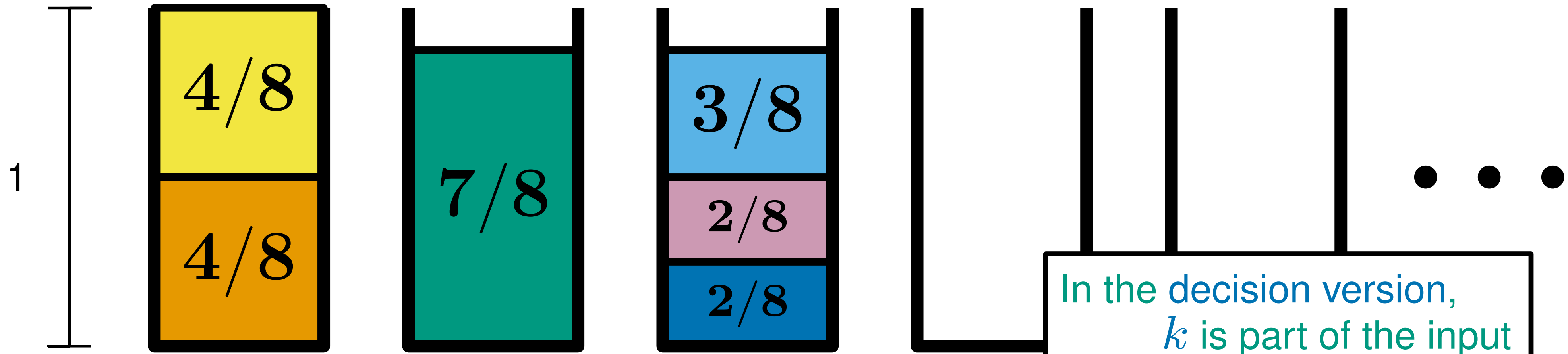


This is an example of an optimisation *problem*



Bin packing

Problem pack all items into the fewest possible bins

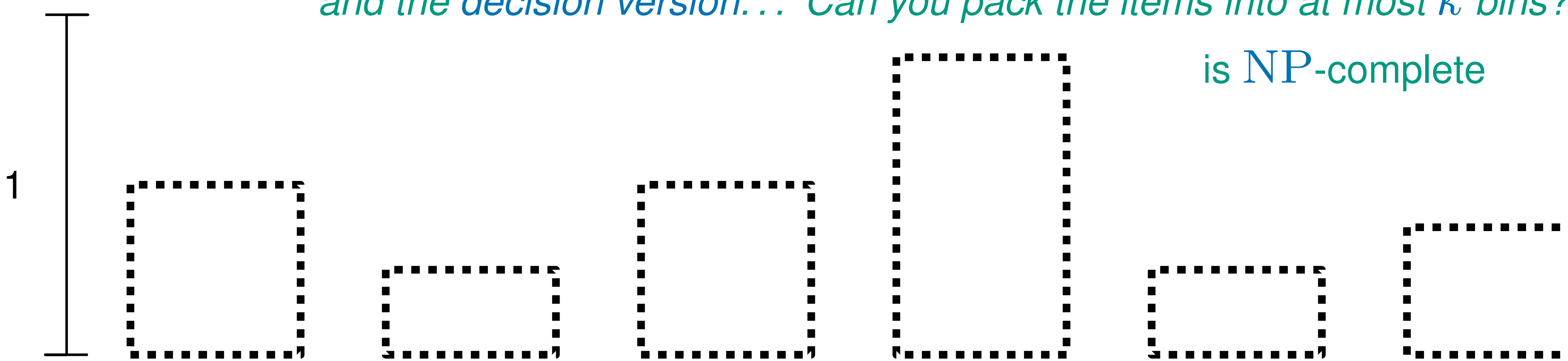


In the decision version,
k is part of the input

The BINPACKING problem is known to be NP-hard

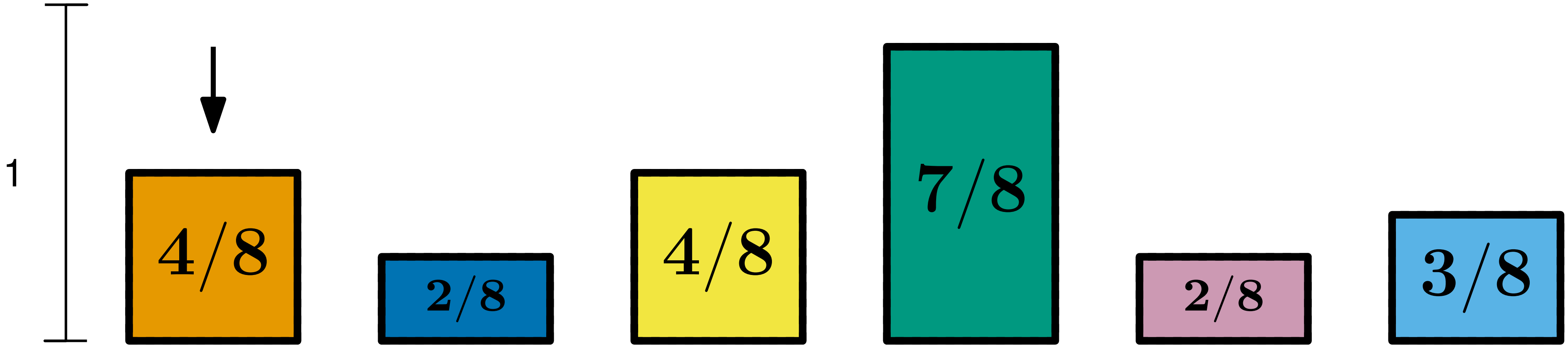
and the decision version... "Can you pack the items into at most k bins?"

is NP-complete

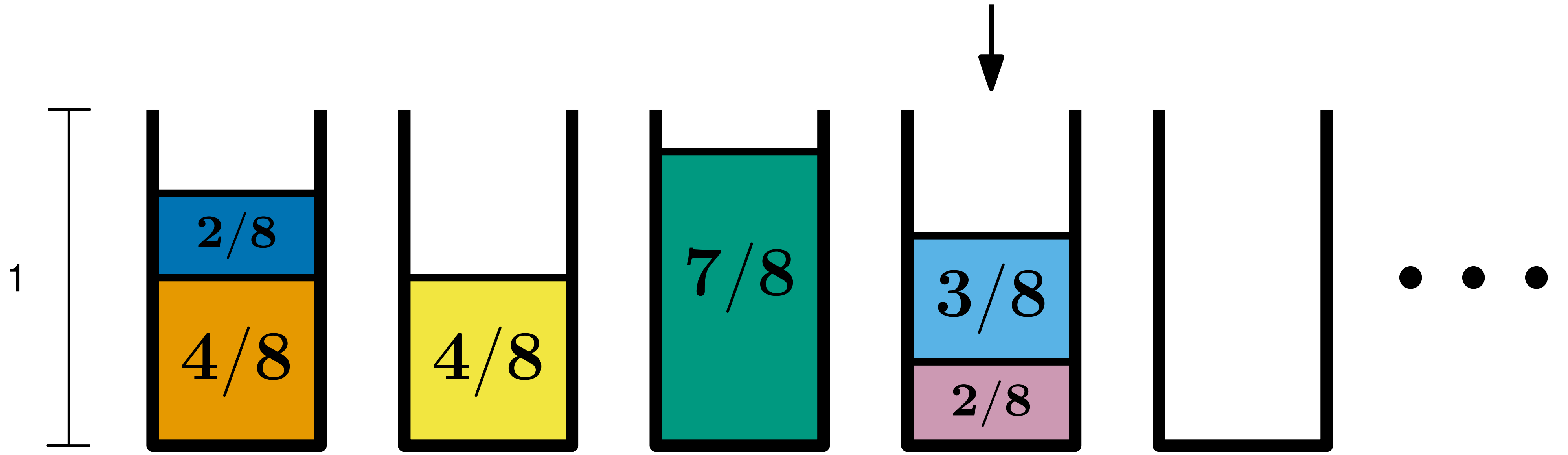


Next fit

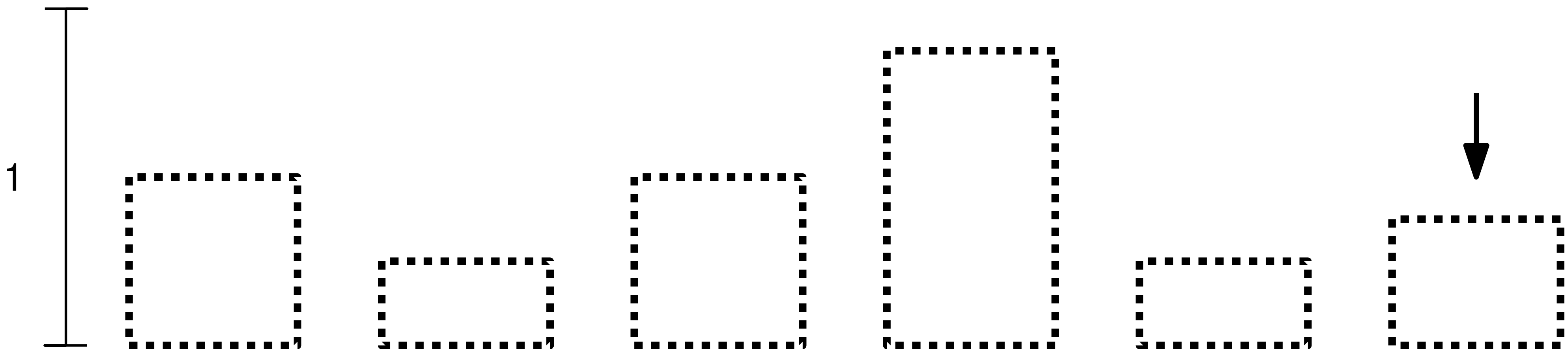
If item i fits into bin j: pack it, i++; **else** j++;



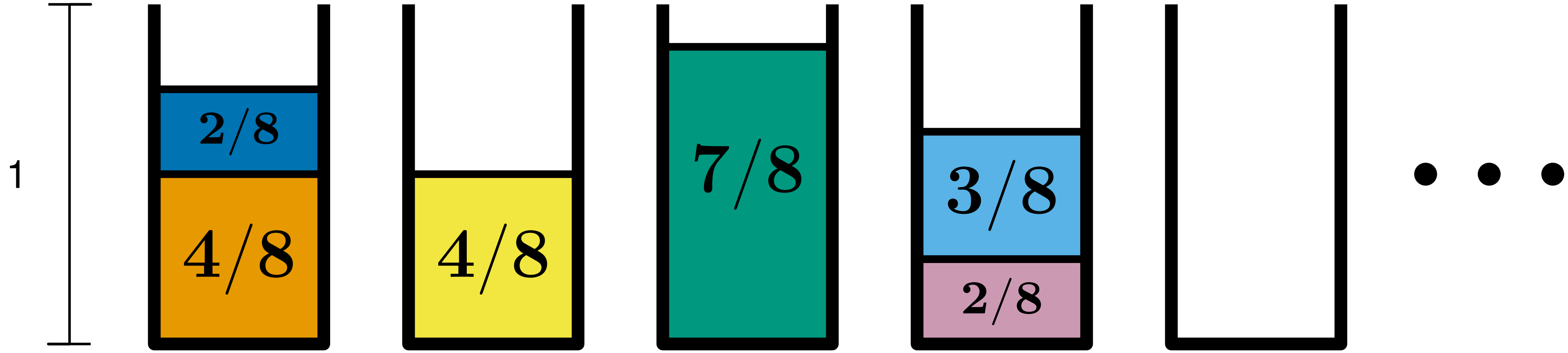
Next fit



If item `i` fits into bin `j`: pack it, `i++`; **else** `j++`;

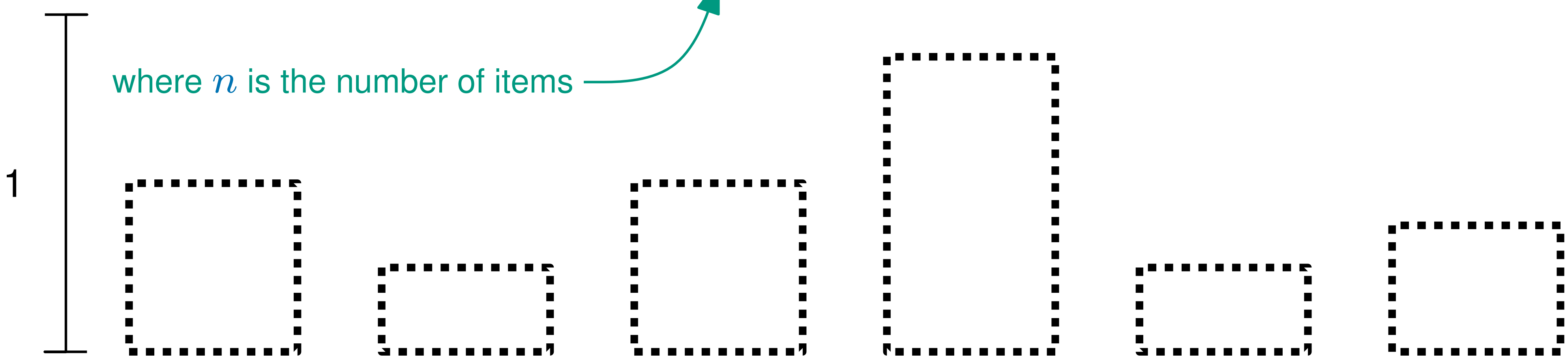


Next fit

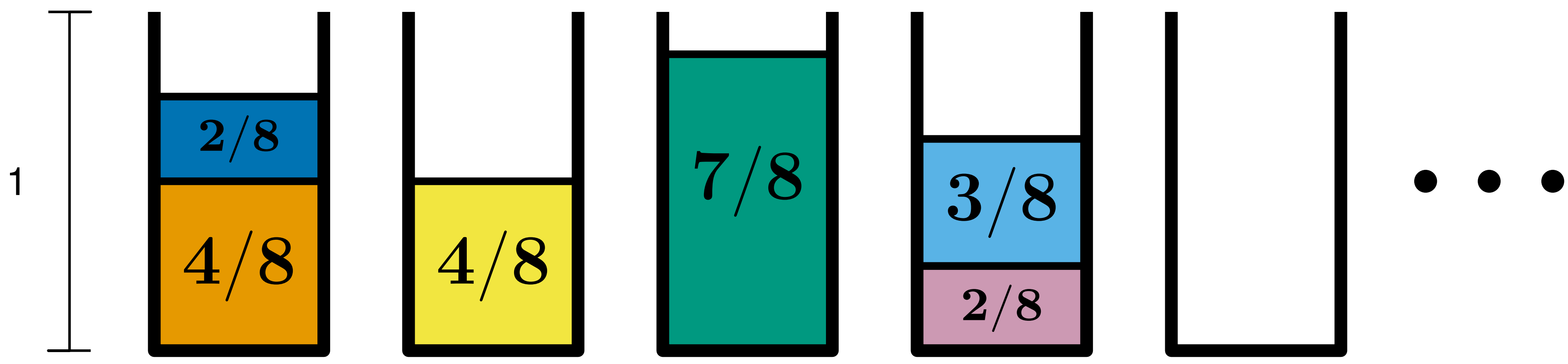


Next fit runs in $O(n)$ time but how good is it?

where n is the number of items



Next fit



Next fit runs in $O(n)$ time but how good is it?

Let fill(i) be the sum of item sizes in bin i

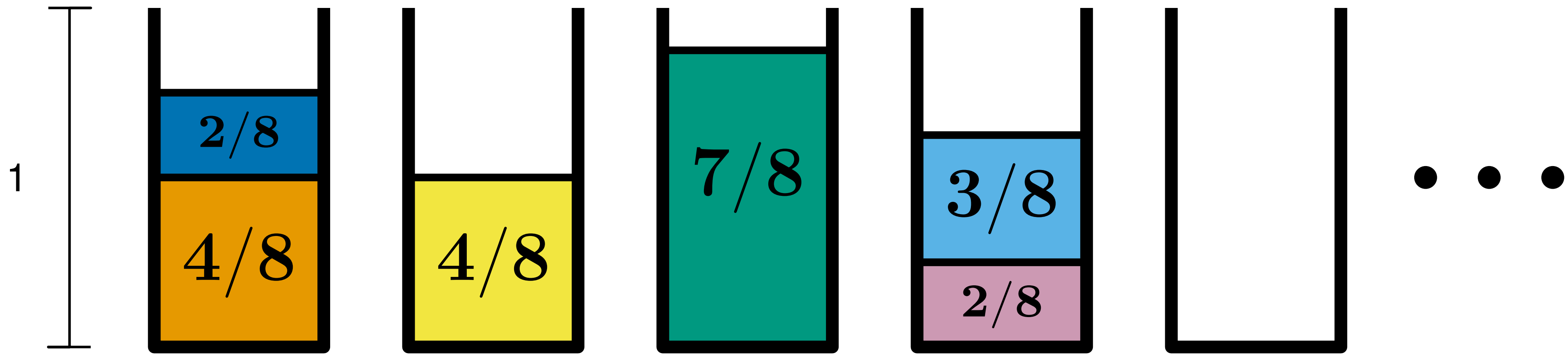
and s be the number of non-empty bins (using Next fit)

Observe that $\text{fill}(2i-1) + \text{fill}(2i) > 1$ (for $1 \leqslant 2i \leqslant s$)

so $\lfloor s/2 \rfloor < \sum_{1 \leqslant 2i \leqslant s} \text{fill}(2i-1) + \text{fill}(2i) \leqslant I$

the sum of the item weights

Next fit



Next fit runs in $O(n)$ time but how good is it?

Let fill(i) be the sum of item sizes in bin i

and s be the number of non-empty bins (using Next fit)

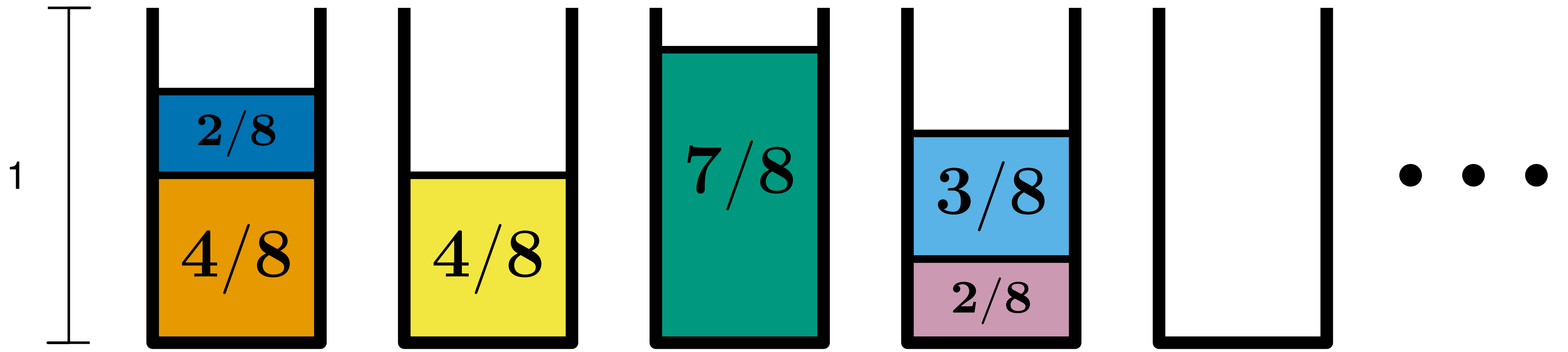
Observe that $\text{fill}(2i-1) + \text{fill}(2i) > 1$ (for $1 \leqslant 2i \leqslant s$)

so $$\lfloor s/2 \rfloor < \sum_{1 \leqslant 2i \leqslant s} \text{fill}(2i-1) + \text{fill}(2i) \leqslant I \leqslant \text{Opt}$$

the sum of the item weights

the optimal number of bins

Next fit



Next fit runs in $O(n)$ time but how good is it?

Let fill(i) be the sum of item sizes in bin i

and s be the number of non-empty bins (using Next fit)

Observe that $\text{fill}(2i-1) + \text{fill}(2i) > 1$ (for $1 \leqslant 2i \leqslant s$)

so $$\lfloor s/2 \rfloor < \sum_{1 \leqslant 2i \leqslant s} \text{fill}(2i-1) + \text{fill}(2i) \leqslant I \leqslant \text{Opt}$$

therefore $s \leqslant 2 \cdot \text{Opt}$ *in other words the Next Fit is never worse than twice the optimal*

Approximation Algorithms

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

- ○ A runs in polynomial time
- ○ A always outputs a solution with value s within an α factor of Opt

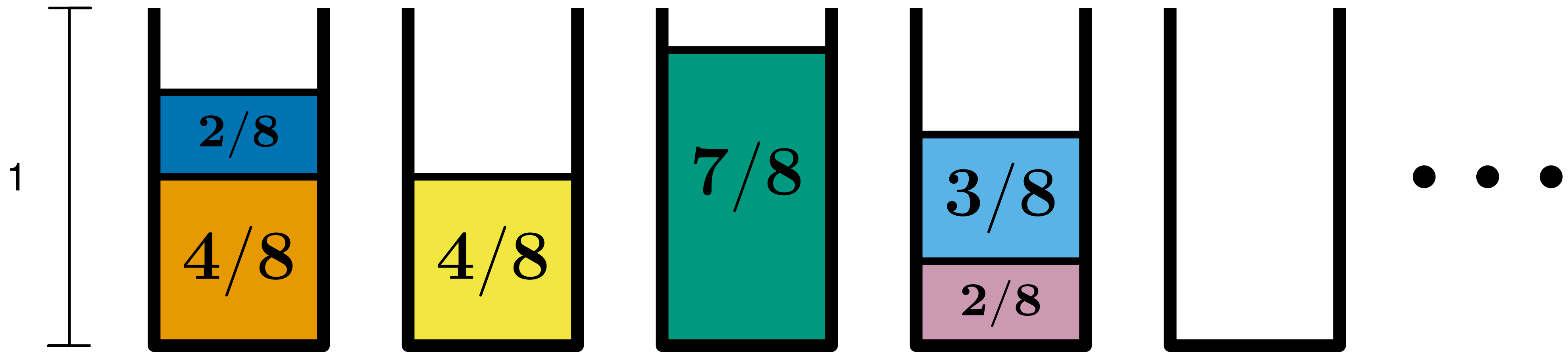
Here P is an optimisation problem with optimal solution of value Opt

- If P is a *maximisation* problem, $\frac{\mathrm{Opt}}{\alpha} \leqslant s \leqslant \mathrm{Opt}$
- If P is a *minimisation* problem (like BinPacking), $\mathrm{Opt} \leqslant s \leqslant \alpha \cdot \mathrm{Opt}$

We have seen a 2-approximation algorithm for BinPacking

the number of bins used, s is always between Opt and $2 \cdot \mathrm{Opt}$

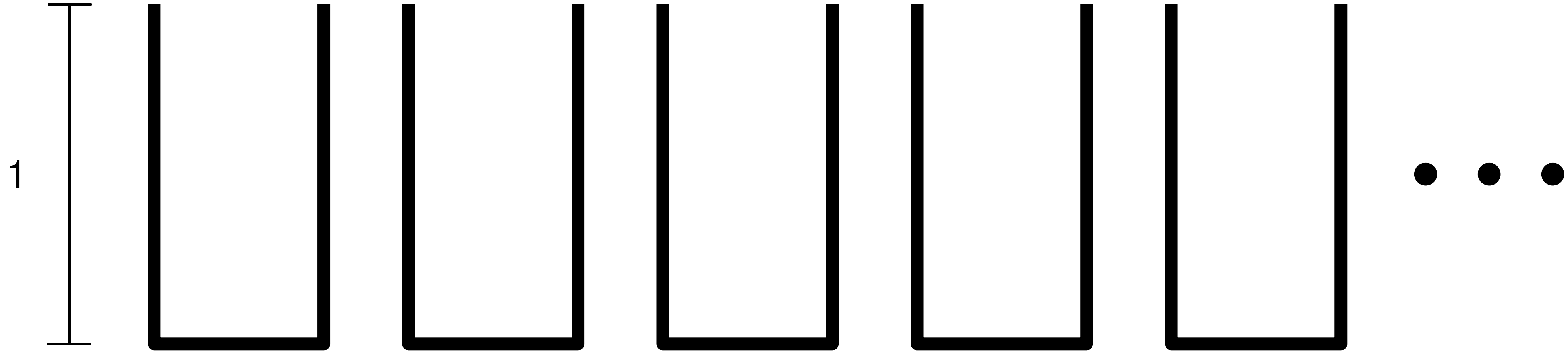
In the examples we consider, α will be a constant but it could depend on n (the input size)



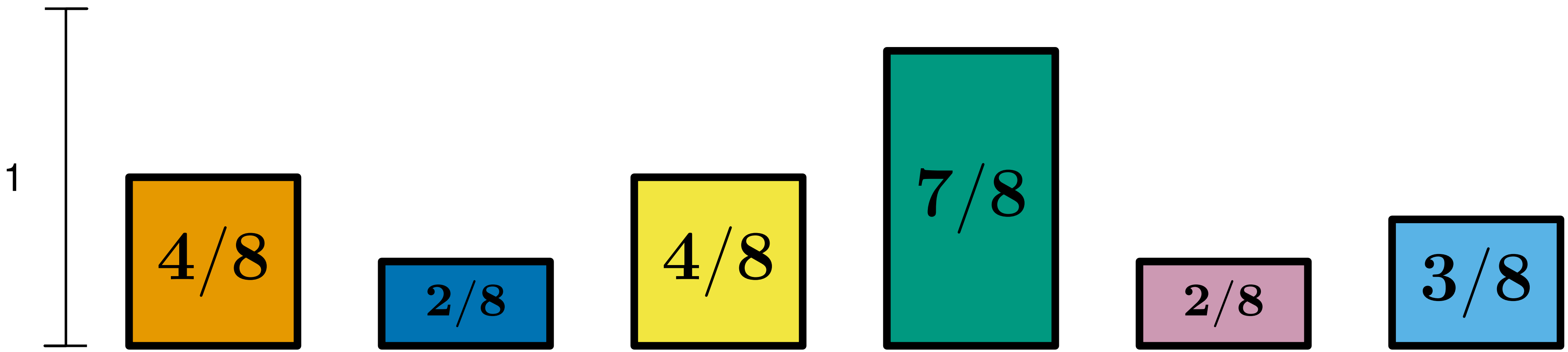
We have seen that Next fit is a 2-approximation algorithm for Bin packing which runs in $O(n)$ time

can we do better?

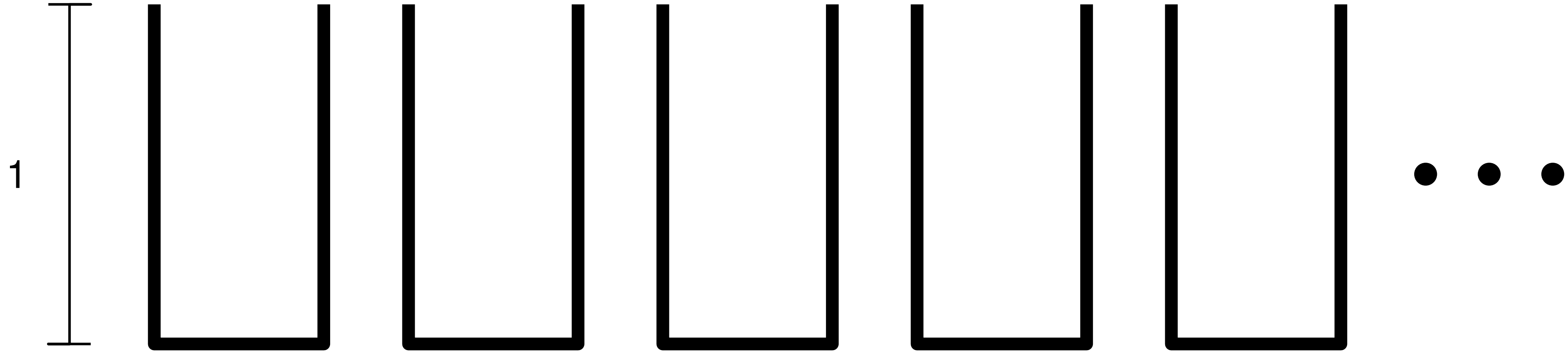
First fit decreasing (FFD)



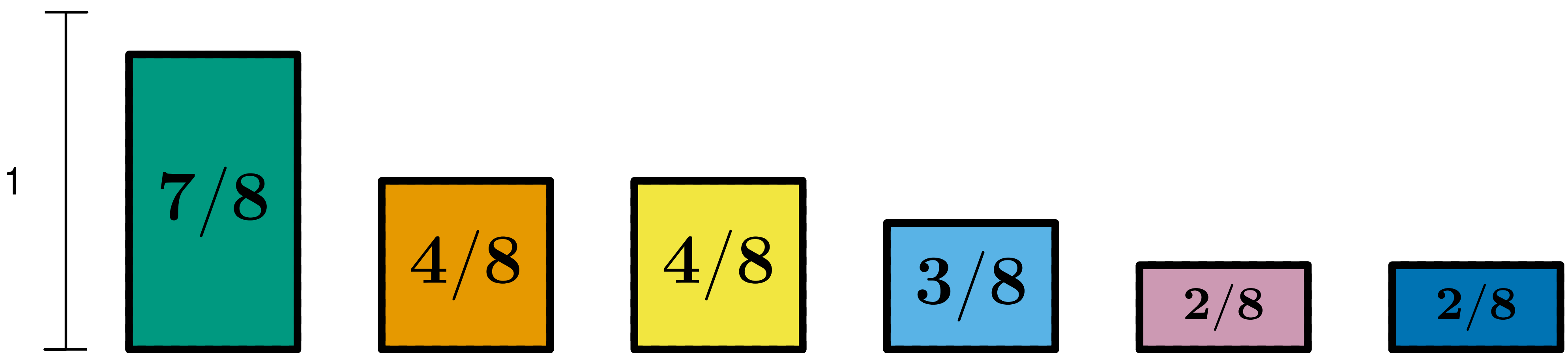
Step 1: Sort the items into non-increasing order



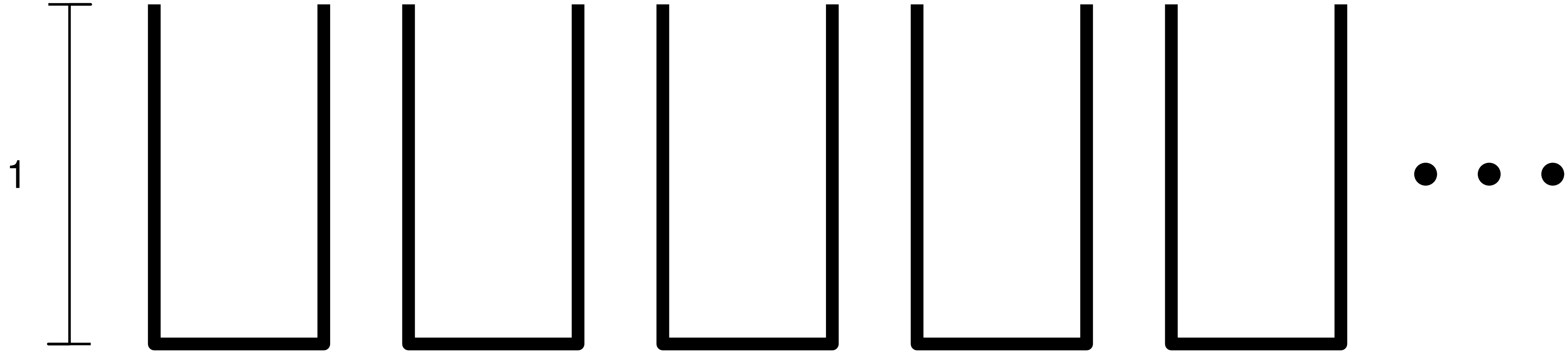
First fit decreasing (FFD)



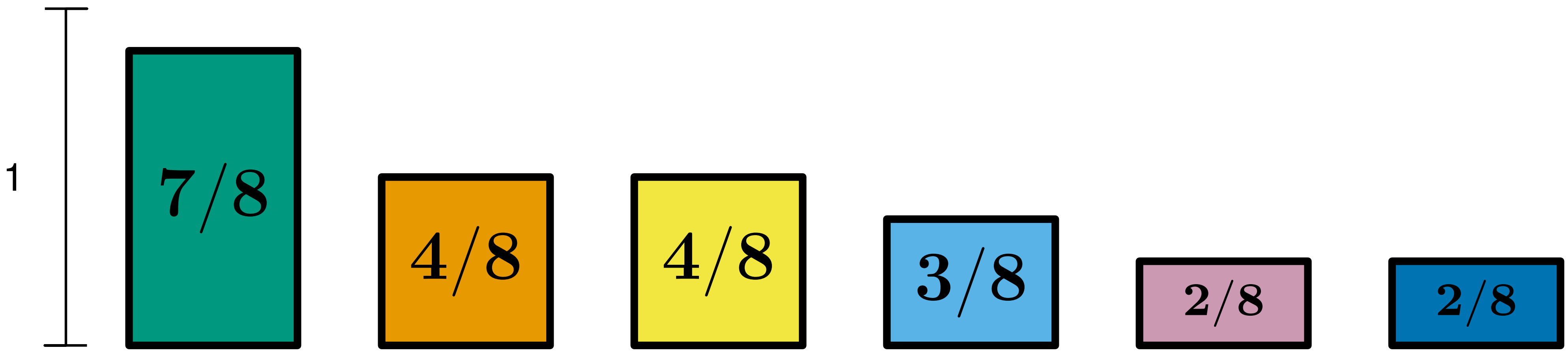
Step 1: Sort the items into non-increasing order



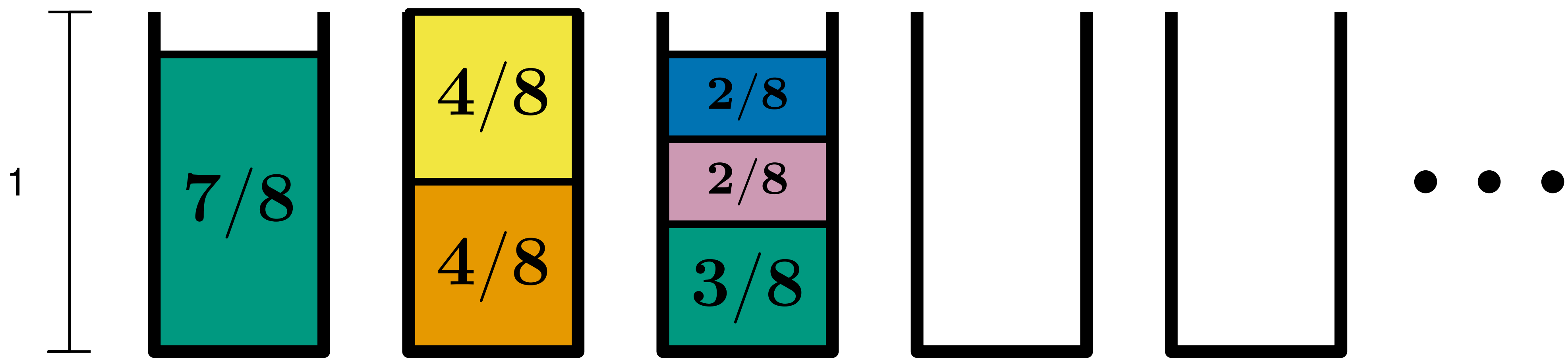
First fit decreasing (FFD)



Step 2: Put each item in the first **(left-most)** bin it fits in



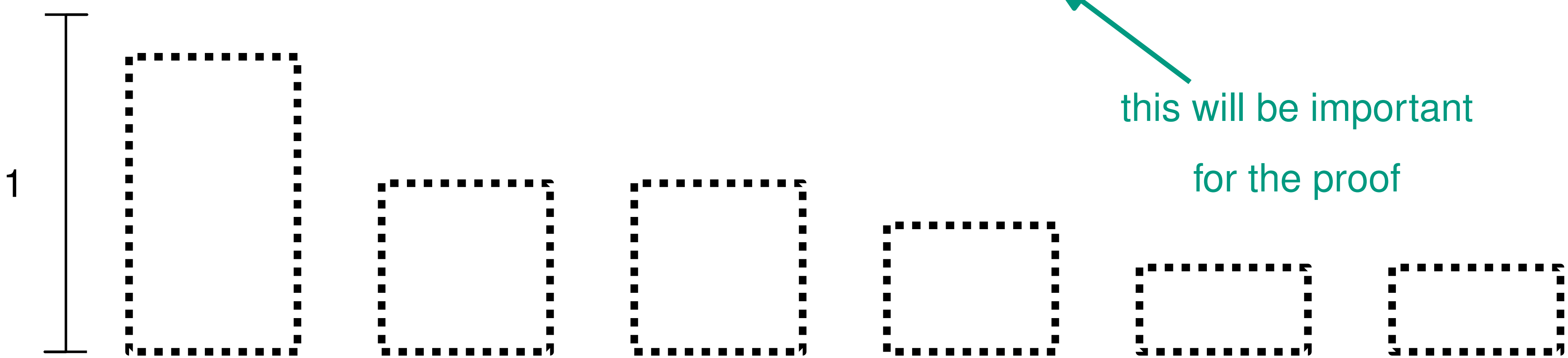
First fit decreasing (FFD)



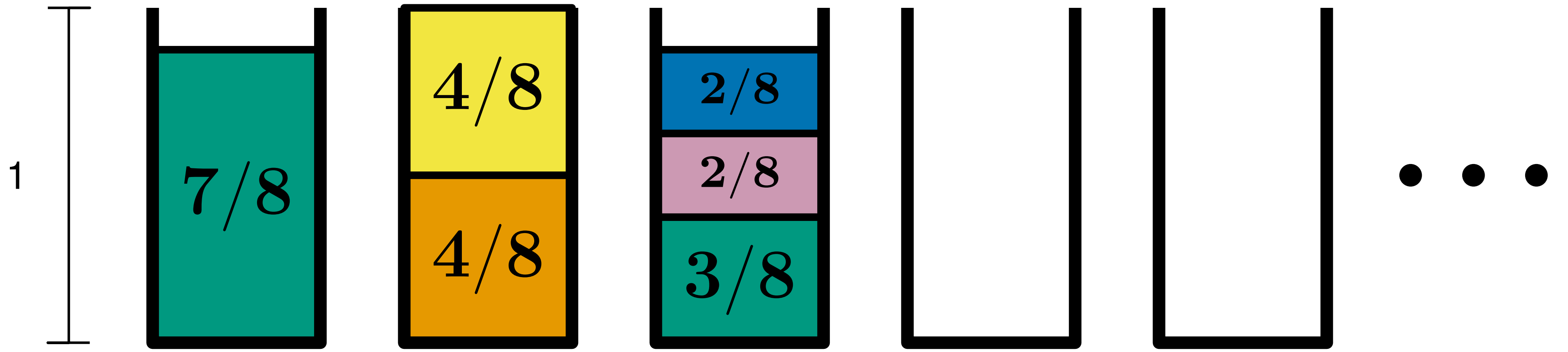
Step 2: Put each item in the first **(left-most)** bin it fits in

this will be important

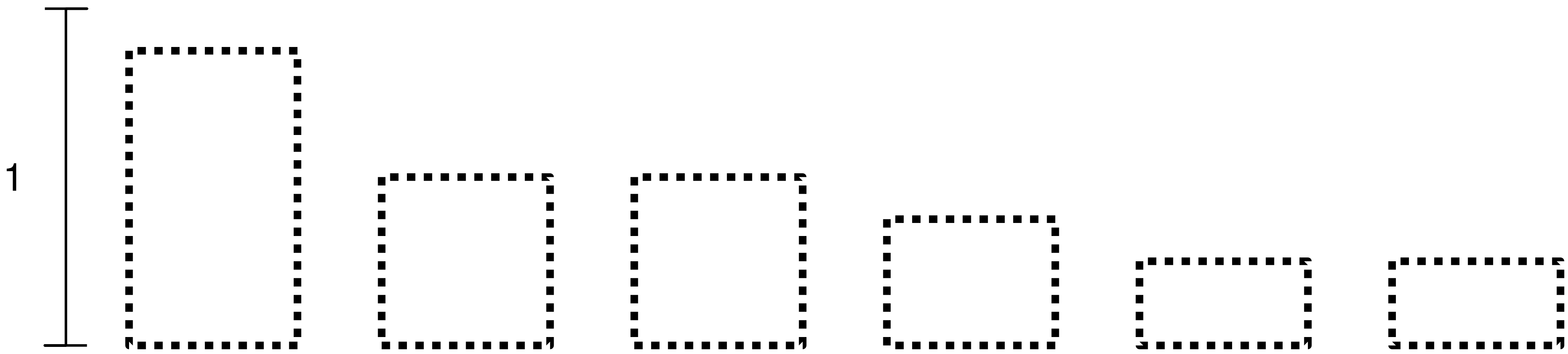
for the proof



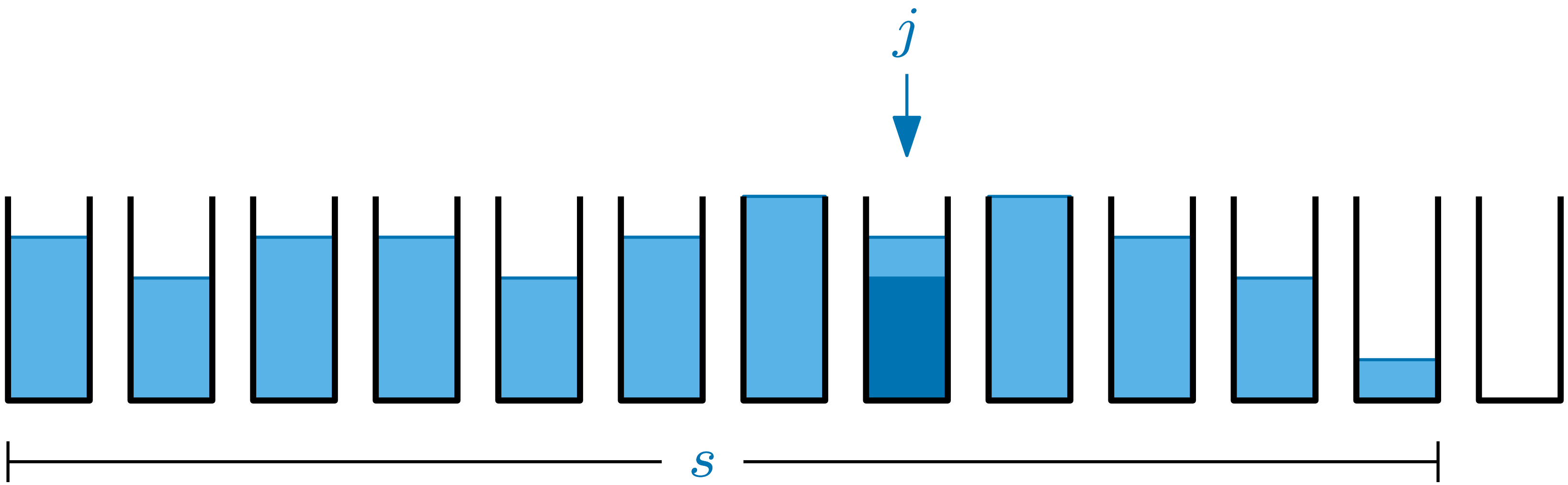
First fit decreasing (FFD)



FFD runs in $O(n^2)$ time but how good is it?



First fit decreasing (FFD)



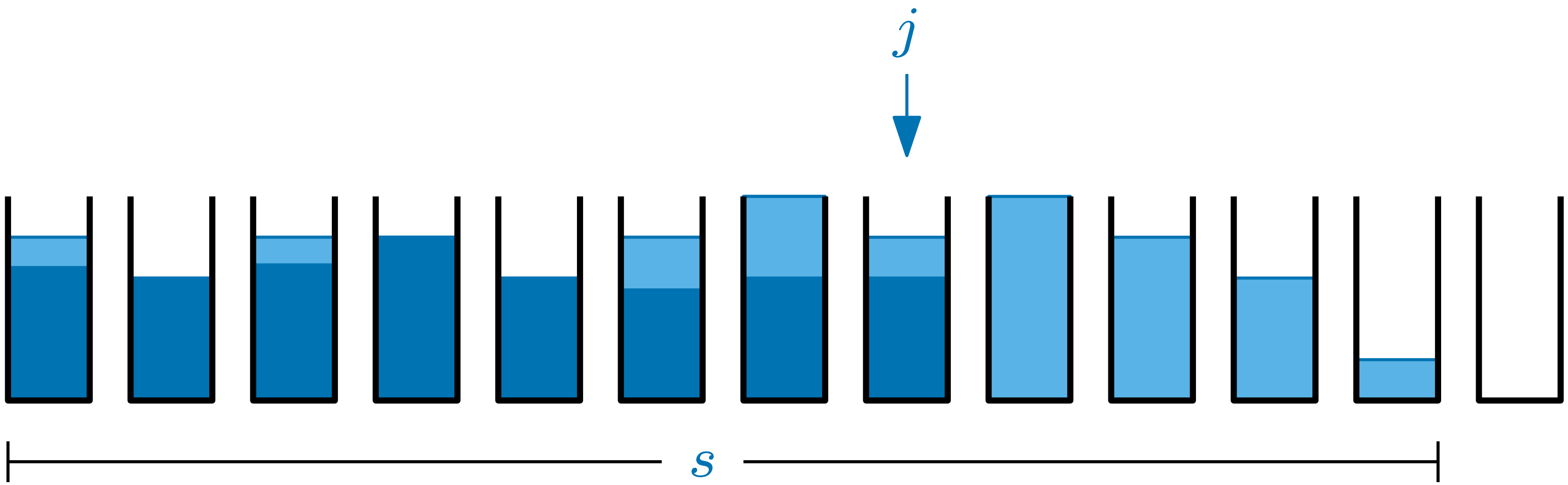
Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins FFD uses on this input)

Case 1: Bin j contains an item of size $> 1/2$

Every bin $j' \leqslant j$ contains an item of size $> 1/2$

because we packed big things first and each thing was packed in the lowest numbered bin

First fit decreasing (FFD)



Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins FFD uses on this input)

Case 1: Bin j contains an item of size $> 1/2$

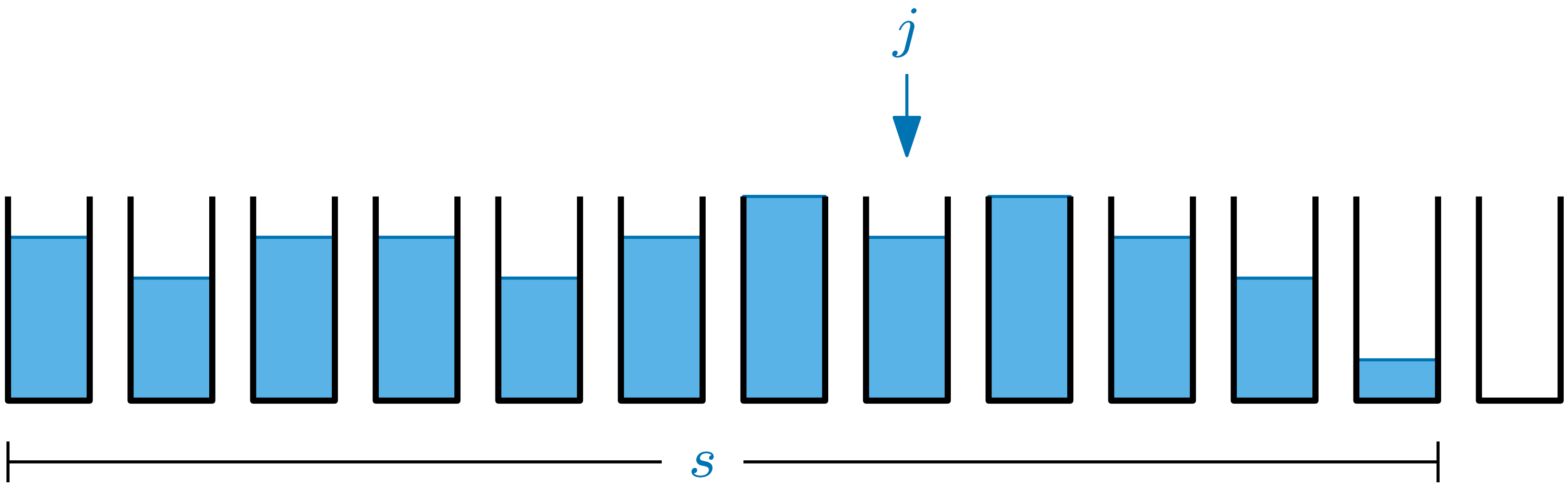
Every bin $j' \leqslant j$ contains an item of size $> 1/2$

each of these items has to be in a different bin (even in Opt)

So Opt uses at least $\frac{2s}{3}$ bins

or... $s \leqslant \dfrac{3\mathrm{Opt}}{2}$

First fit decreasing (FFD)

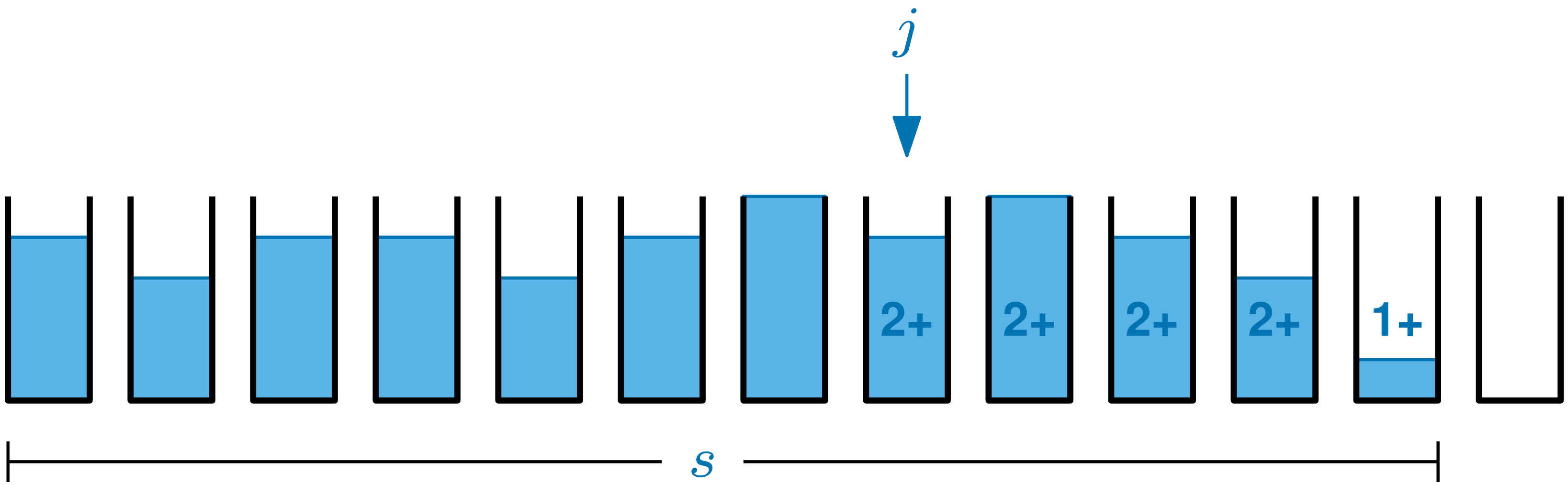


Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins FFD uses on this input)

Case 2: Bin j contains only items of size $\leqslant 1/2$

when FFD packed the first item into bin j,

First fit decreasing (FFD)



Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins FFD uses on this input)

Case 2: Bin j contains only items of size $\leqslant 1/2$

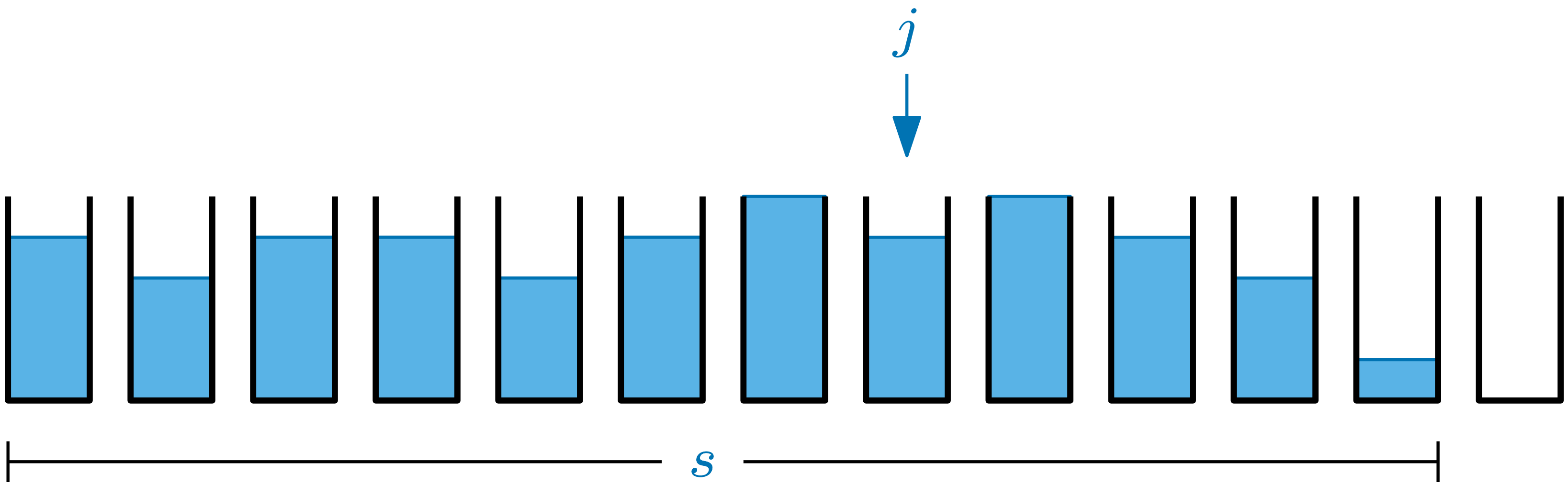
so Bins $j, (j+1), \ldots, (s-2), (s-1)$ each contain at least two items

and bin s contains at least one item

This gives a total of $2(s-j)+1$ items, none of which fits into bins $1, 2, 3, \ldots, (j-1)$

otherwise we would have packed them there

First fit decreasing (FFD)



Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins FFD uses on this input)

Case 2: Bin j contains only items of size $\leqslant 1/2$

consider pairing **these** with **these**

so Bins $j, (j+1), \ldots, (s-2), (s-1)$ each contain at least two items

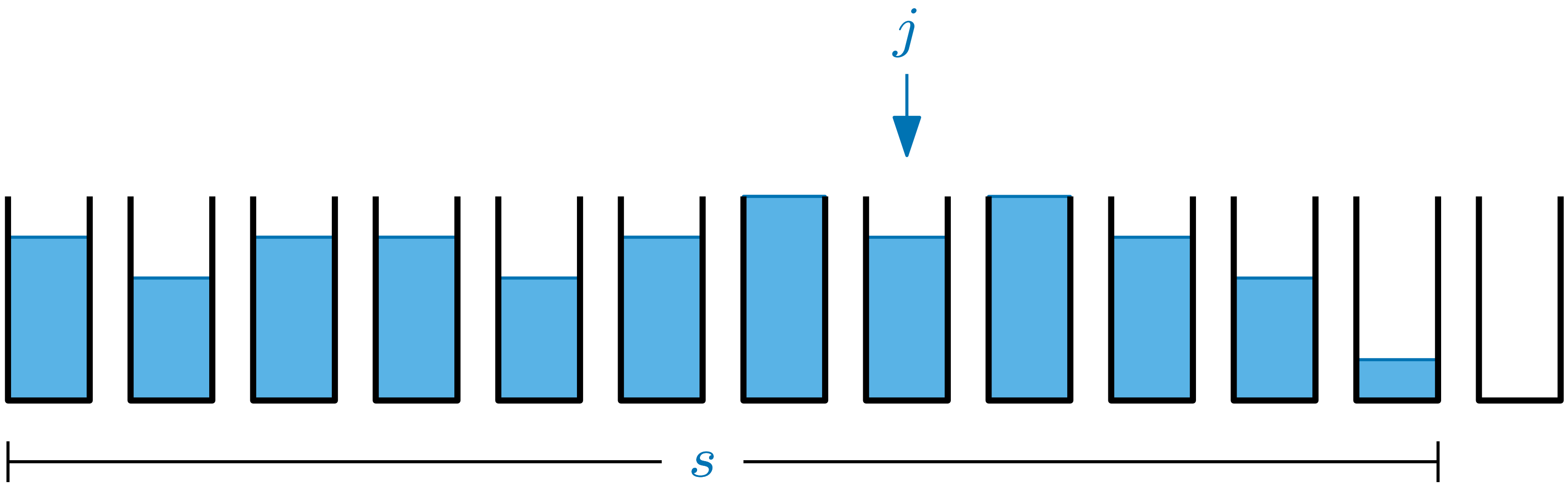
and bin s contains at least one item

This gives a total of $2(s-j)+1$ items, none of which fits into bins $1, 2, 3, \ldots, (j-1)$

so $I > \min\{j-1, 2(s-j)+1\}$

recall I is the total weight of all items

First fit decreasing (FFD)



Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins FFD uses on this input)

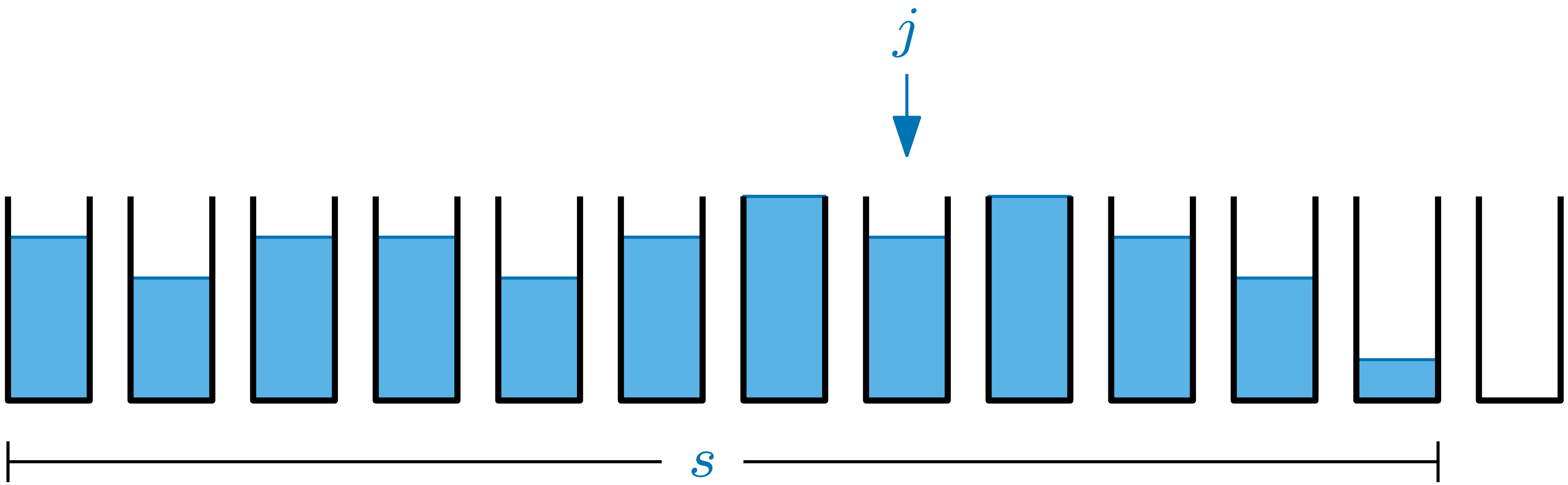
Case 2: Bin j contains only items of size $\leqslant 1/2$

so Bins $j, (j+1), \ldots, (s-2), (s-1)$ each contain at least two items
and bin s contains at least one item

This gives a total of $2(s-j)+1$ items, none of which fits into bins $1, 2, 3, \ldots, (j-1)$

so $I > \min\{j-1, 2(s-j)+1\}$

First fit decreasing (FFD)



Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins FFD uses on this input)

Case 2: Bin j contains only items of size $\leqslant 1/2$

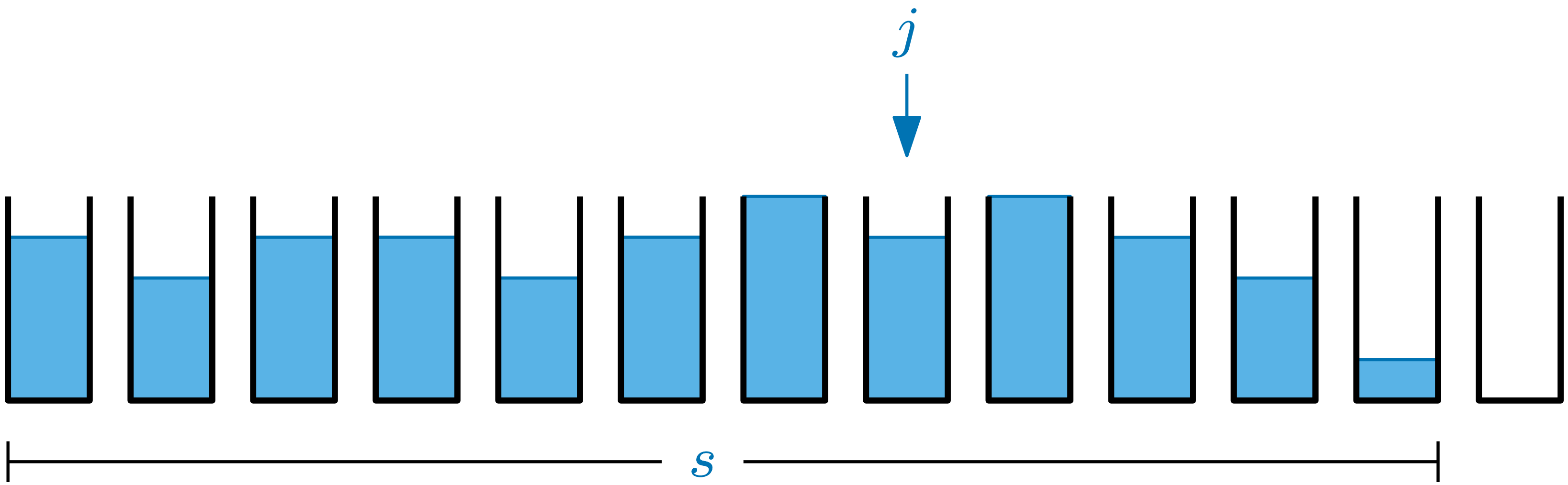
so Bins $j, (j+1), \ldots, (s-2), (s-1)$ each contain at least two items
and bin s contains at least one item

This gives a total of $2(s-j)+1$ items, none of which fits into bins $1, 2, 3, \ldots, (j-1)$

so $I > \min\{j-1, 2(s-j)+1\} \geqslant \lceil 2s/3 \rceil - 1$

by plugging in $j = \lceil 2s/3 \rceil$

First fit decreasing (FFD)



Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins FFD uses on this input)

Case 2: Bin j contains only items of size $\leqslant 1/2$

As $\lceil 2s/3 \rceil - 1 < I$ and $I \leqslant \mathrm{Opt}$

we have that $\lceil 2s/3 \rceil - 1 < \mathrm{Opt}$

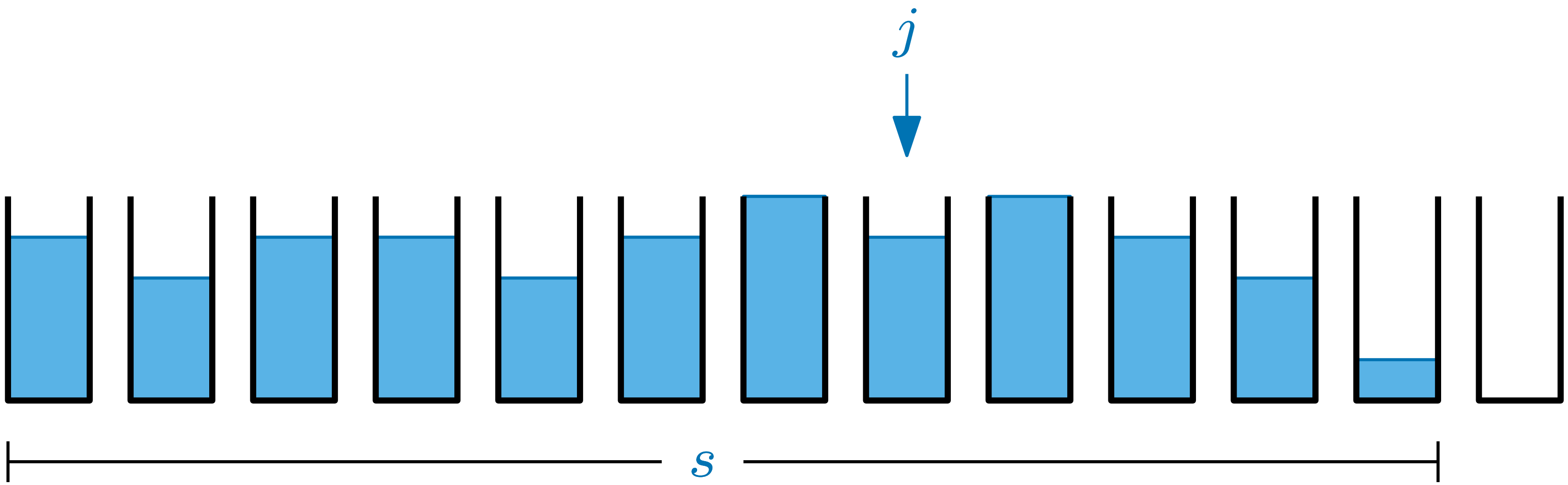
. . . but both sides are integers. . .

so $\lceil 2s/3 \rceil \leqslant \mathrm{Opt}$

finally . . . $2s/3 \leqslant \lceil 2s/3 \rceil \leqslant \mathrm{Opt}$

or $s \leqslant (3/2)\mathrm{Opt}$

First fit decreasing (FFD)



Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins FFD uses on this input)

Case 1: Bin j contains an item of size $> 1/2$

Case 2: Bin j contains only items of size $\leqslant 1/2$

in both cases... $s \leqslant \dfrac{3\text{Opt}}{2}$

So FFD is a $3/2$-approximation algorithm for BinPacking

Approximation Algorithms Summary

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

- A runs in polynomial time
- A always outputs a solution with value s within an α factor of Opt

Here P is an optimisation problem with optimal solution of value Opt

If P is a *maximisation* problem, $\frac{\mathrm{Opt}}{\alpha} \leqslant s \leqslant \mathrm{Opt}$

If P is a *minimisation* problem (like BinPacking), $\mathrm{Opt} \leqslant s \leqslant \alpha \cdot \mathrm{Opt}$

We have seen Next Fit which is a 2-approximation algorithm for BinPacking

which runs in $O(n)$ time

and First Fit Decreasing which is a $3/2$-approximation algorithm for BinPacking

which runs in $O(n^2)$ time

Bin Packing is NP-hard so solving it exactly in polynomial time would prove that $\mathrm{P} = \mathrm{NP}$