Approximation Algorithms

Week 10

COMP 1201 (Algorithmics)

ECS, University of Southampton

15 May 2020

Previously...

Linear Programming

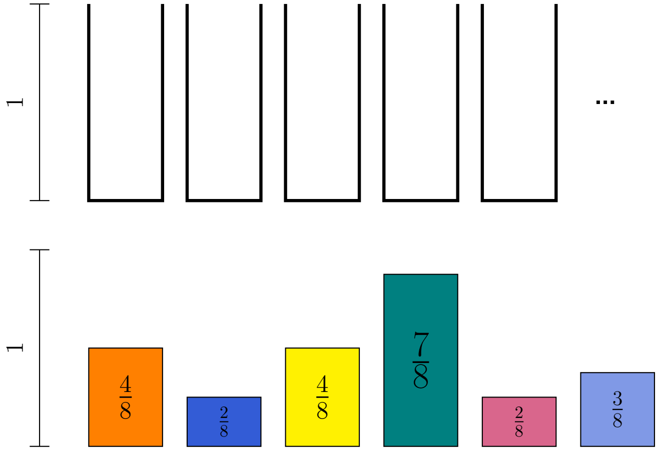
- Optimisation of linear functions subject to linear inequality constraints.
- Incredibly powerful tool in applied mathematics. Many practical problems can be cast as Linear Programs.
- Efficient algorithms exist for solving *very large* Linear Programs.
- The Simplex Algorithm runs in *exponential time* in the worst case, but is polynomial time in practice.
- Alternative *interior point* algorithms have been developed that have polynomial time complexity.

A Little Complexity Theory

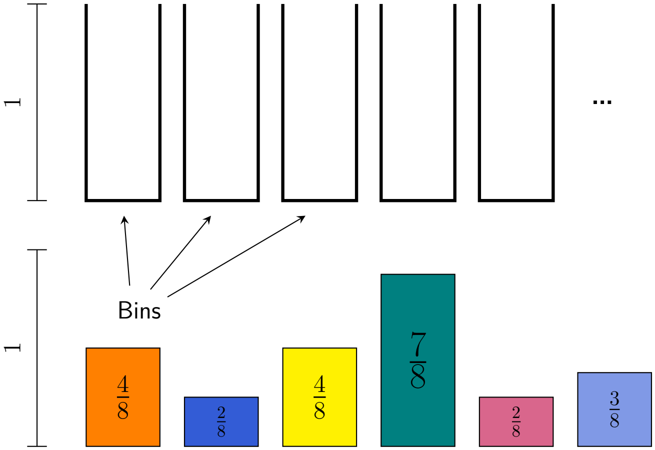
NP-Completeness in short.

- **NP** is a class of decision problems (i.e. **yes/no** problems) that we can check in polynomial time.
- A problem A is **NP-Complete** if
  1. A is in **NP**, and
  2. Every B in **NP** has a polynomial time reduction to A.
     [This second part is the definition of **NP-Hard**.]
- Thus, if we could solve A quickly, we could solve **every** problem in **NP** quickly.
- **NP-Complete** problems are the 'hardest' in **NP**.
- Most computer scientists believe that you cannot solve them in polynomial time (**P** $\neq$ **NP**).

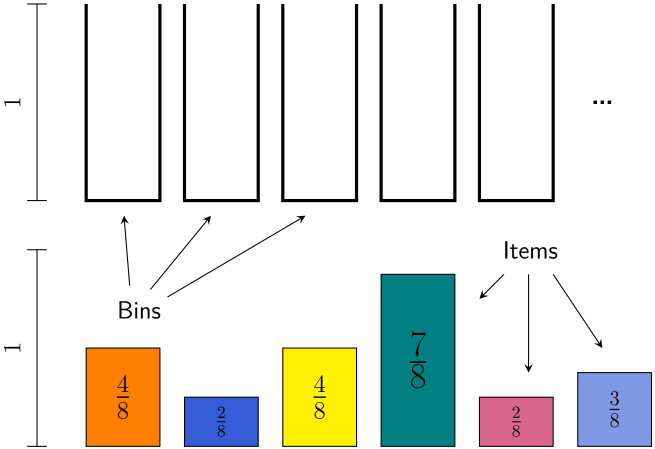
Bin Packing



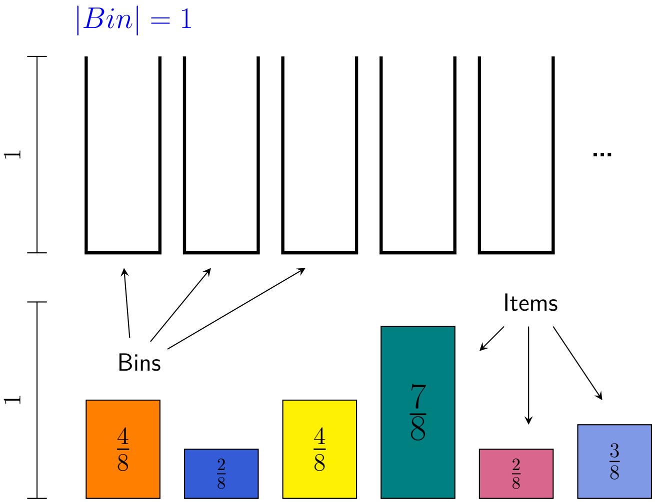
Bin Packing



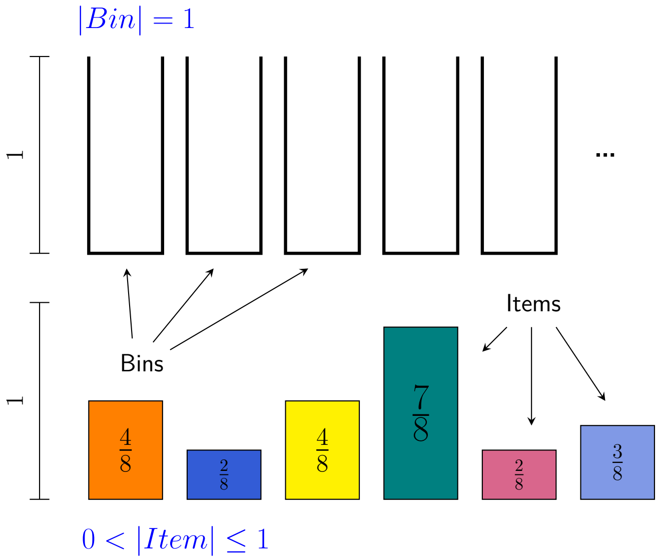
Bin Packing



Bin Packing


$|Bin| = 1$
1
...
Items
Bins
1
$\frac{4}{8}$
$\frac{2}{8}$
$\frac{4}{8}$
$\frac{7}{8}$
$\frac{2}{8}$
$\frac{3}{8}$

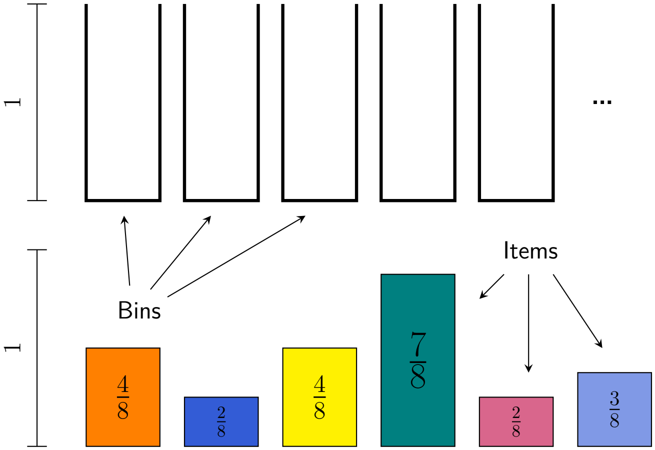
Bin Packing



$|Bin| = 1$
1
...
Bins
Items
1
$\frac{4}{8}$
$\frac{2}{8}$
$\frac{4}{8}$
$\frac{7}{8}$
$\frac{2}{8}$
$\frac{3}{8}$
$0 < |Item| \leq 1$

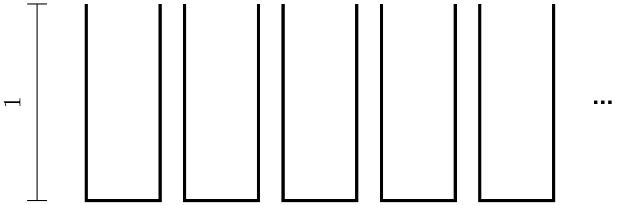
Bin Packing

Problem: pack all the items into fewest bins possible.

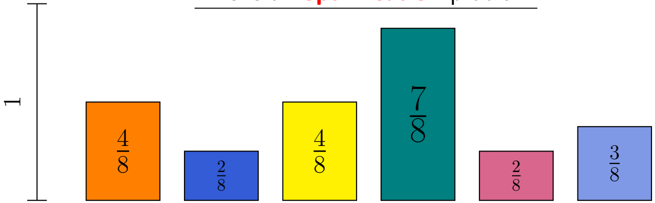


Bin Packing

Problem: pack all the items into fewest bins possible.

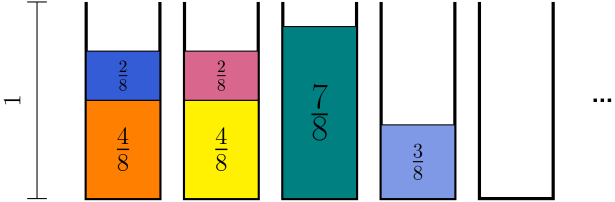


This is an **optimisation** problem.

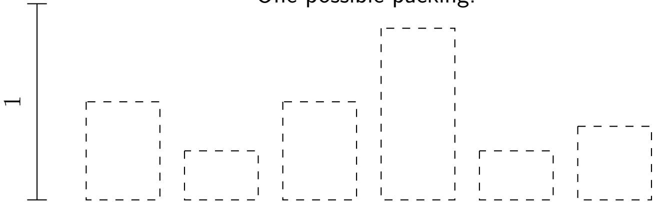


Bin Packing

Problem: pack all the items into fewest bins possible.

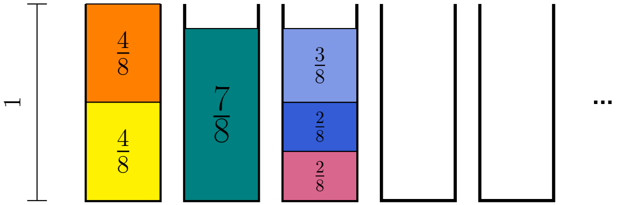


One possible packing.

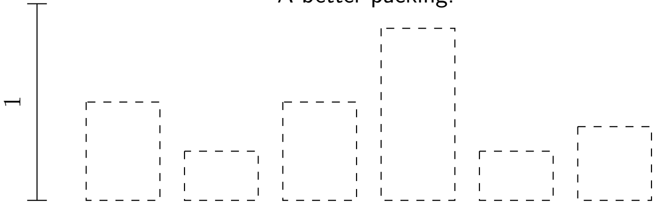


Bin Packing

Problem: pack all the items into fewest bins possible.

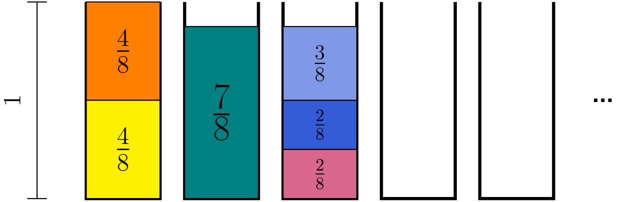


A better packing.

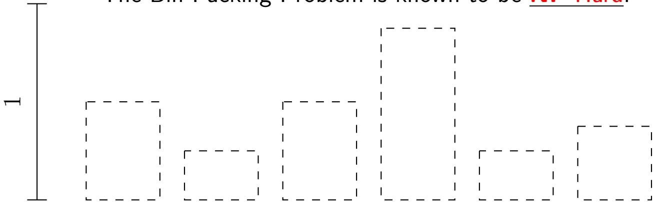


Bin Packing

Problem: pack all the items into fewest bins possible.

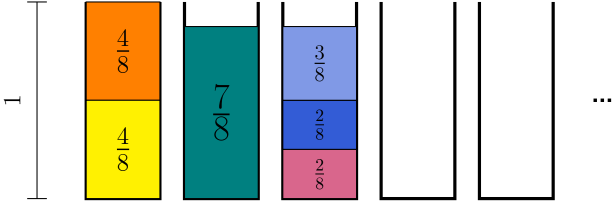


The Bin Packing Problem is known to be **NP**-Hard.



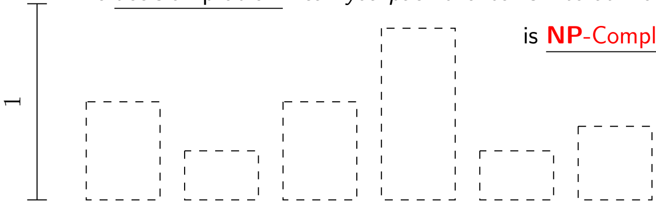
Bin Packing

Problem: pack all the items into fewest bins possible.



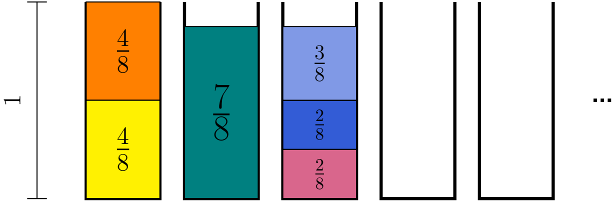
The decision problem "*can you pack the items into at most k bins?*"

is **NP**-Complete.



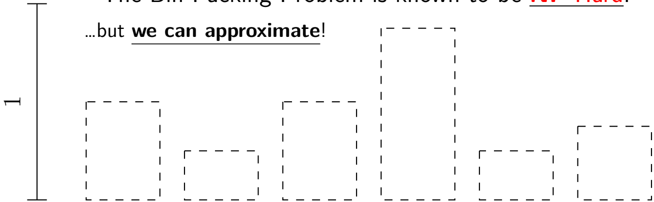
Bin Packing

Problem: pack all the items into fewest bins possible.

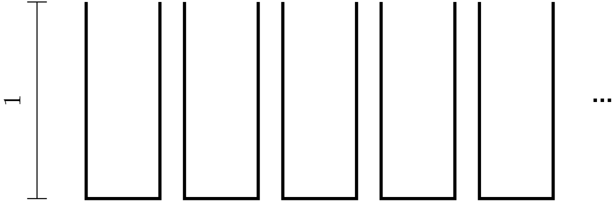


The Bin Packing Problem is known to be **NP**-Hard.

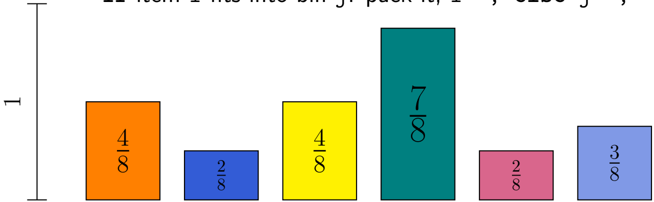
...but **we can approximate**!



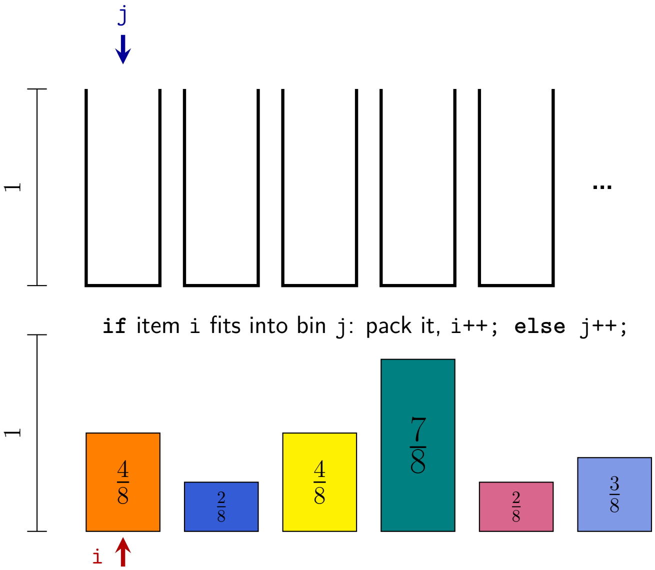
Next Fit



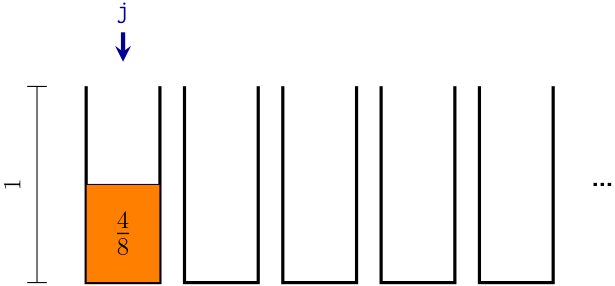
if item `i` fits into bin `j`: pack it, `i++;` **else** `j++;`



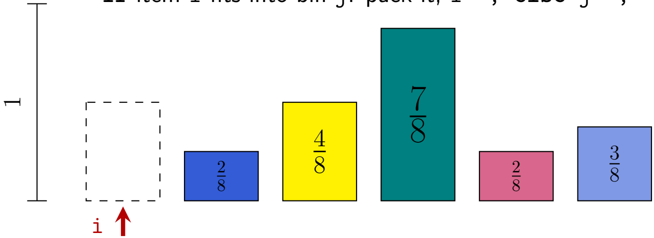
Next Fit



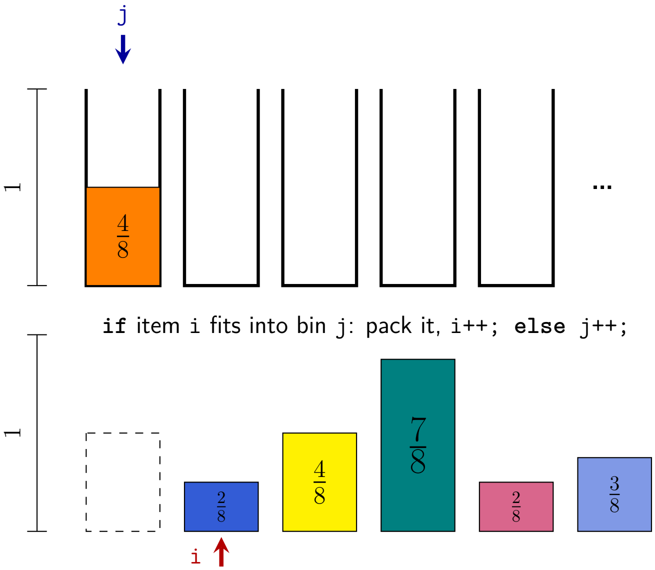
Next Fit



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

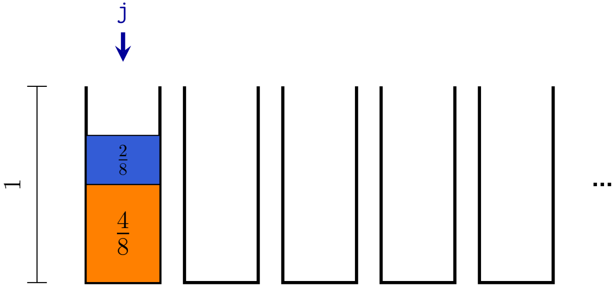


Next Fit

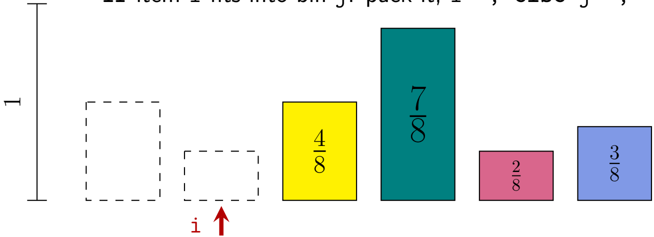


if item `i` fits into bin `j`: pack it, `i++;` **else** `j++;`

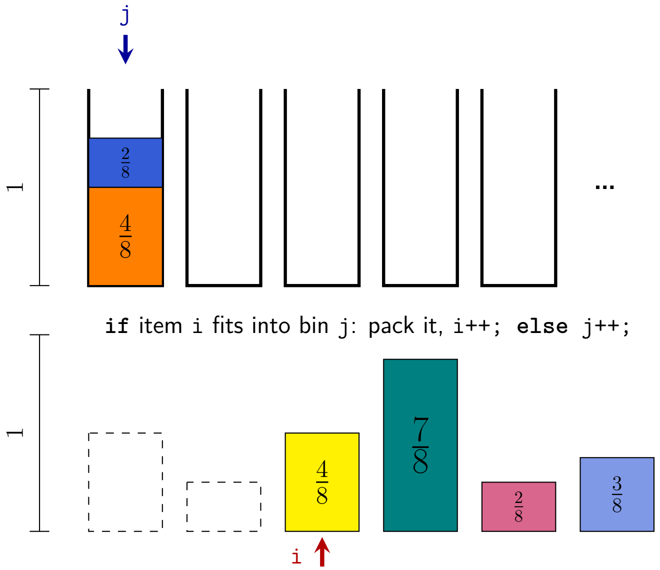
Next Fit



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

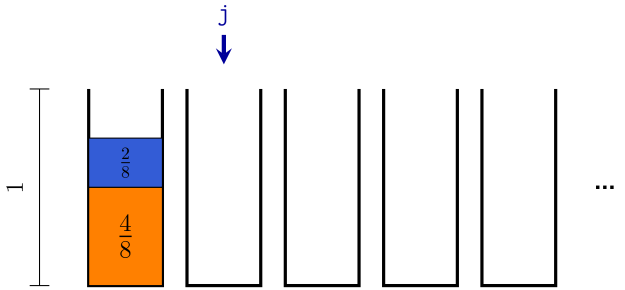


Next Fit

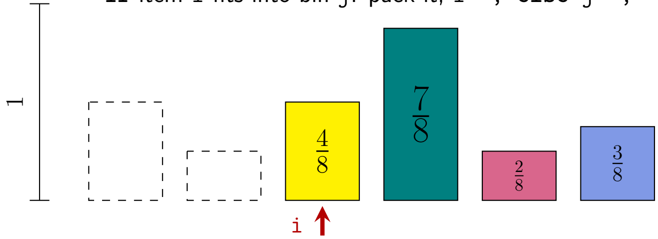


j
1
$\frac{2}{8}$
$\frac{4}{8}$
...
if item i fits into bin j: pack it, i++; else j++;
1
$\frac{4}{8}$
$\frac{7}{8}$
$\frac{2}{8}$
$\frac{3}{8}$
i

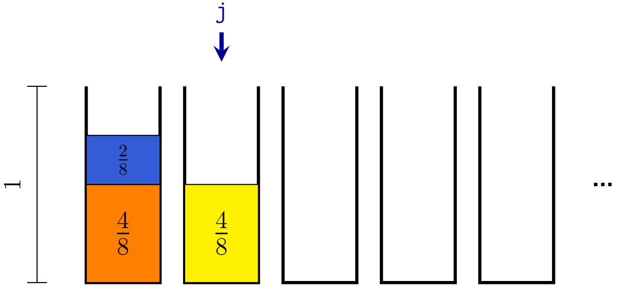
Next Fit



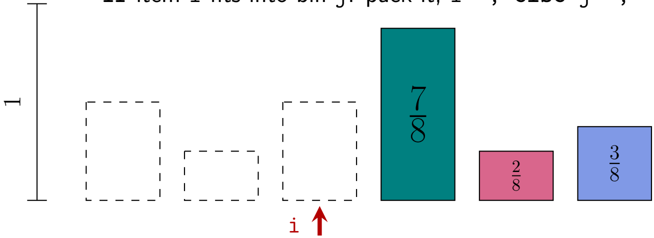
if item i fits into bin j: pack it, i++; **else** j++;



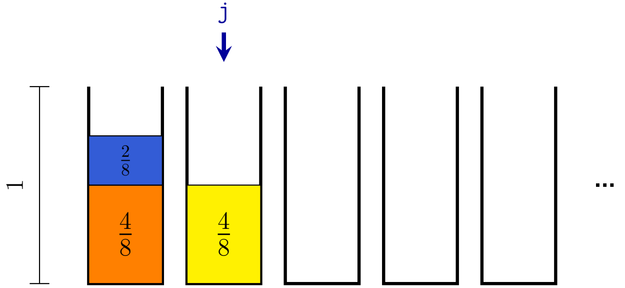
Next Fit



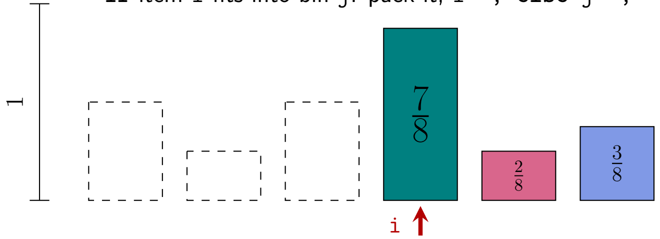
if item `i` fits into bin `j`: pack it, `i++;` **`else`** `j++;`



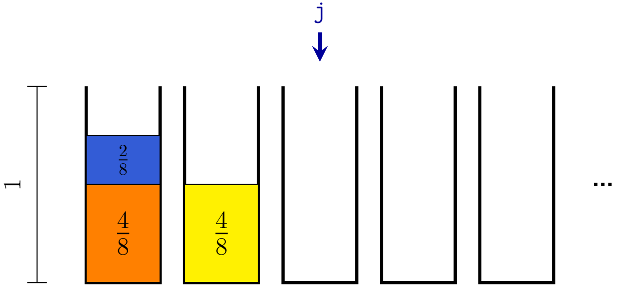
Next Fit



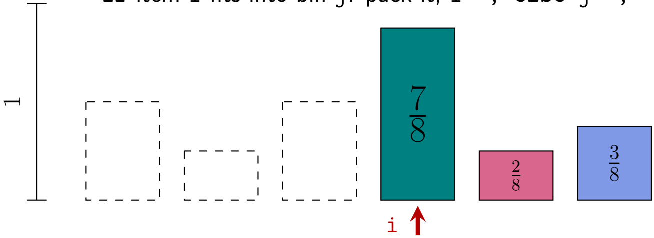
if item i fits into bin j: pack it, i++; **else** j++;



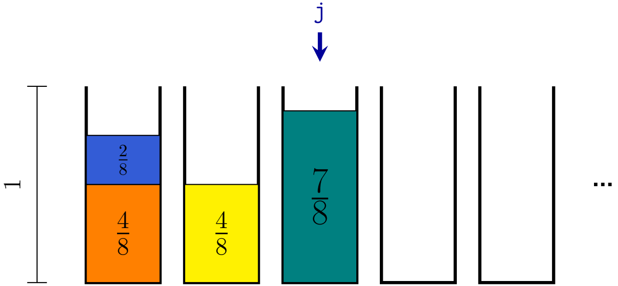
Next Fit



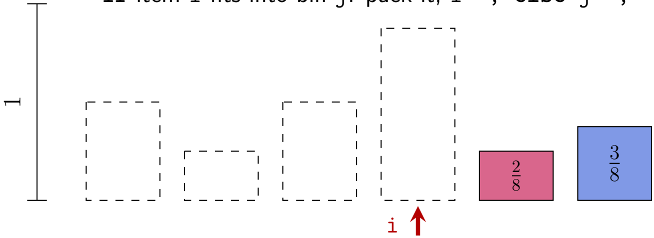
if item `i` fits into bin `j`: pack it, `i++;` **else** `j++;`



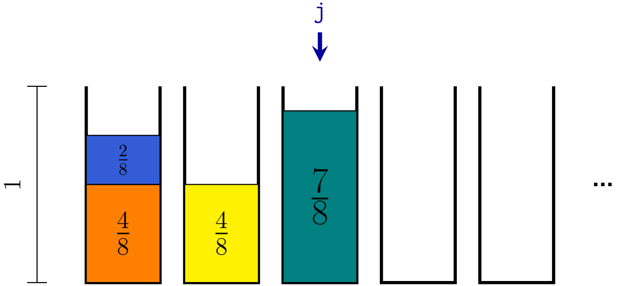
Next Fit



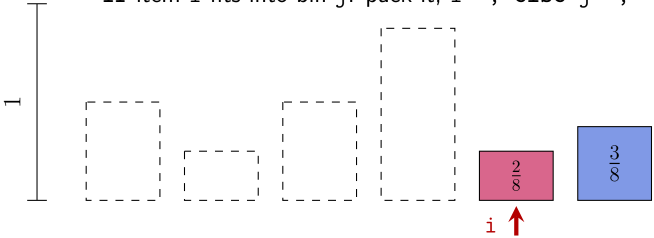
if item `i` fits into bin `j`: pack it, `i++;` **else** `j++;`



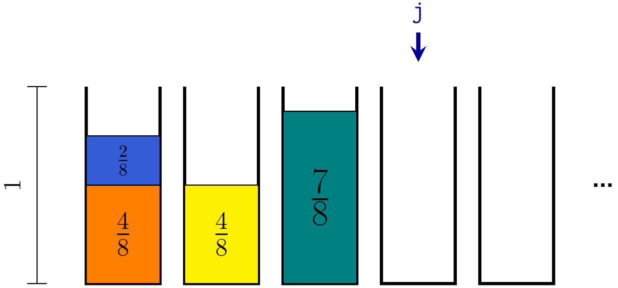
Next Fit



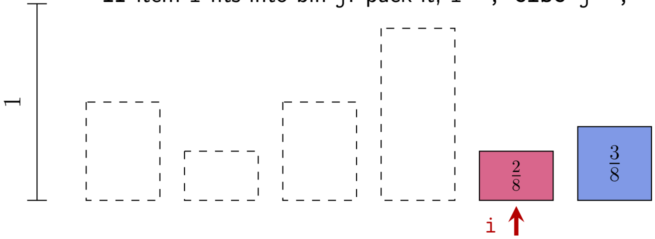
if item `i` fits into bin `j`: pack it, `i++`; **else** `j++`;



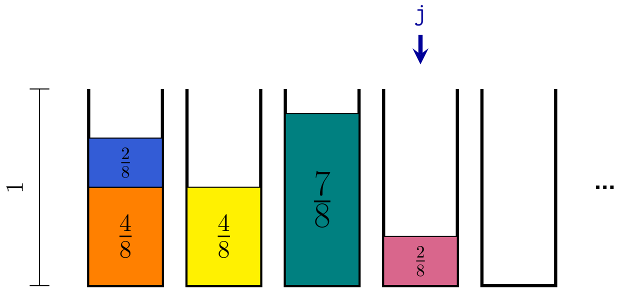
Next Fit



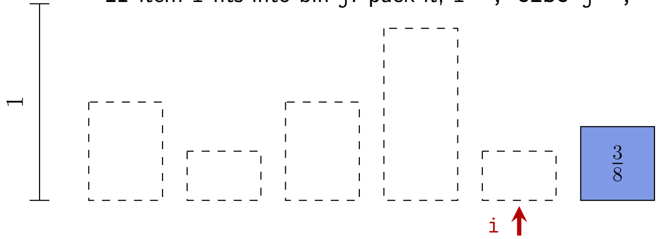
if item `i` fits into bin `j`: pack it, `i++;` **else** `j++;`



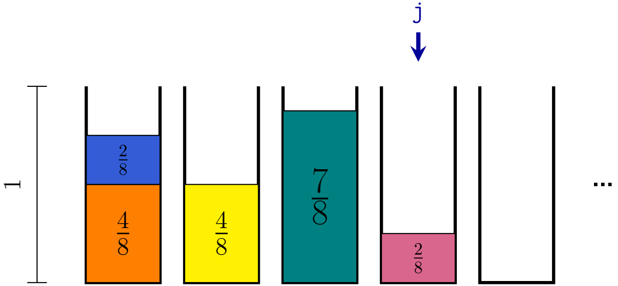
Next Fit



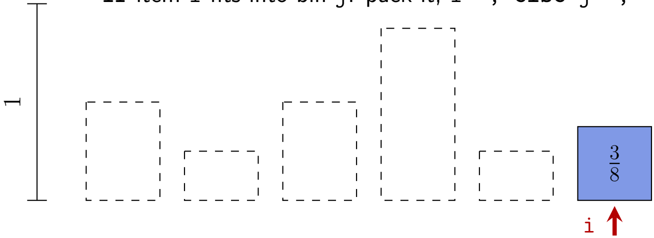
if item i fits into bin j: pack it, i++; **else** j++;



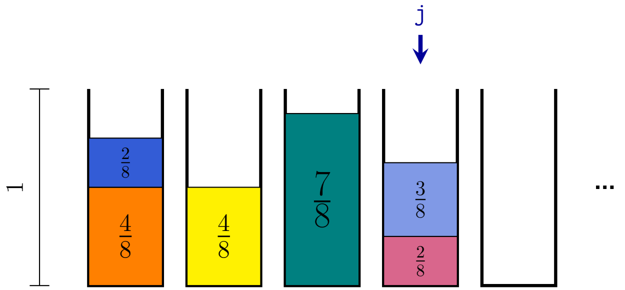
Next Fit



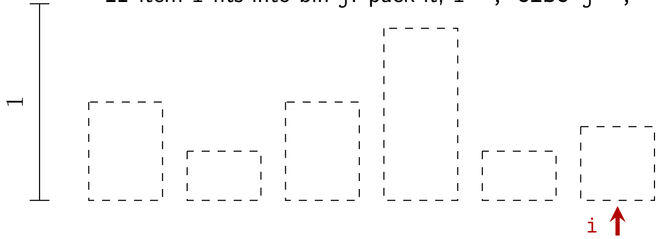
if item `i` fits into bin `j`: pack it, `i++`; **else** `j++`;



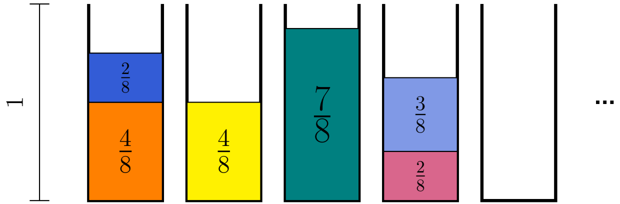
Next Fit



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

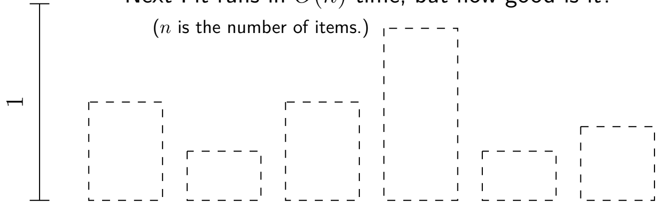


Next Fit

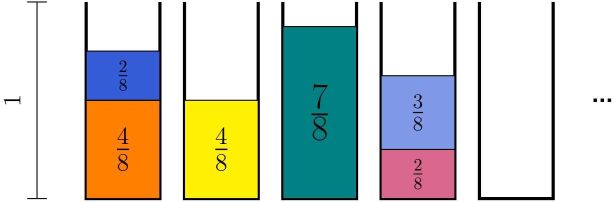


Next Fit runs in $O(n)$ time, but how good is it?

(n is the number of items.)



Next Fit



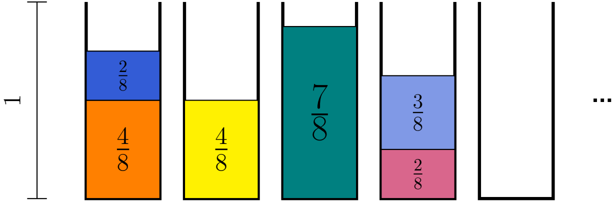
Next Fit runs in $O(n)$ time, but how good is it?

(n is the number of items.)

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

and `s` be the number of non-empty bins (using Next Fit).

Next Fit



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

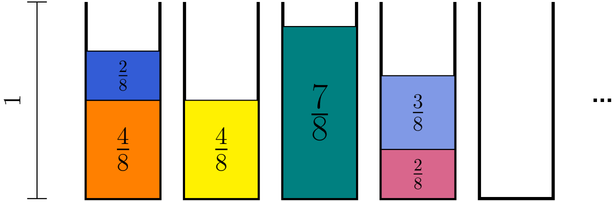
(n is the number of items.)

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 `fill`$(2i-1)$ + `fill`$(2i)$ > 1 (for $1 \leq 2i \leq s$).

Next Fit



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

(n is the number of items.)

Let $\texttt{fill}(i)$ be the sum of item sizes in Bin i

and $\texttt{s}$ be the number of non-empty bins (using Next Fit).

$$\left\lfloor \frac{s}{2} \right\rfloor < \sum_{1 \le 2i \le s} \texttt{fill}(2i-1) + \texttt{fill}(2i)$$

Next Fit

Let $\texttt{fill}(i)$ be the sum of item sizes in Bin i and $\texttt{s}$ be the number of non-empty bins (using Next Fit).

Observe that $\texttt{fill}(2i-1)\ \texttt{+}\ \texttt{fill}(2i) > 1$ (for $1 \leq 2i \leq s$).

So $\left\lfloor \frac{s}{2} \right\rfloor < \sum_{1\leq 2i\leq s} \texttt{fill}(2i-1)\ \texttt{+}\ \texttt{fill}(2i) \leq I \leq Opt$

Here I is the sum of the item weights and Opt is the optimal solution.

Next Fit

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 `fill(`$2i-1$`) + fill(`$2i$`)` > 1 (for $1 \leq 2i \leq s$).

So $\left\lfloor \frac{s}{2} \right\rfloor < \sum_{1 \leq 2i \leq s} \texttt{fill}(2i-1) \texttt{ + } \texttt{fill}(2i) \leq I \leq Opt$

Here I is the sum of the item weights and Opt is the optimal solution.

Therefore $s \leq 2 \cdot Opt$, i.e. the output of Next Fit is never worse that twice of what is optimal.

Approximation Algorithms

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

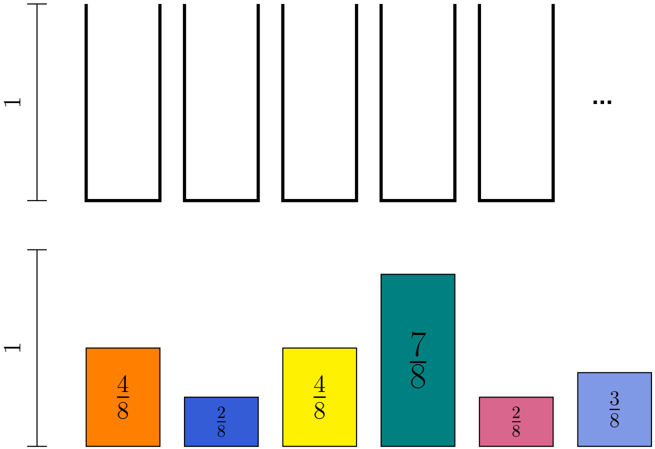
- A runs in polynomial time.
- Always outputs a solution with value s within an α factor of Opt. [Here P is an optimisation problem with optimal solution value Opt.]

If P is a maximisation problem $\frac{Opt}{\alpha} \leq s \leq Opt$.
If P is a minimisation problem $Opt \leq s \leq \alpha Opt$.

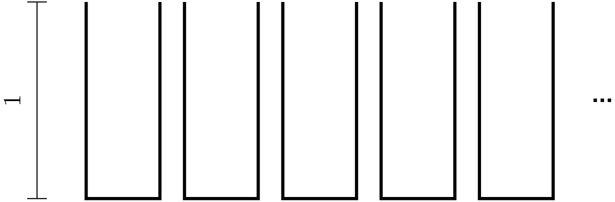
We have now seen Next Fit, which is a 2-approximation algorithm for Bin Packing, which runs in $O(n)$ time.

In our examples α will be a constant, but in general it can depend on n (i.e. problem input size).

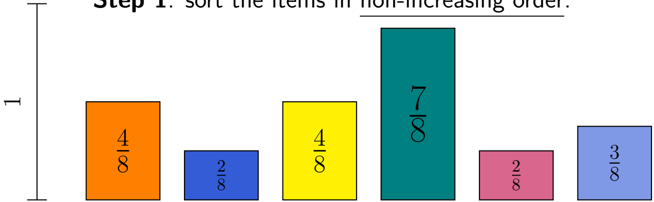
First Fit Decreasing (FFD)



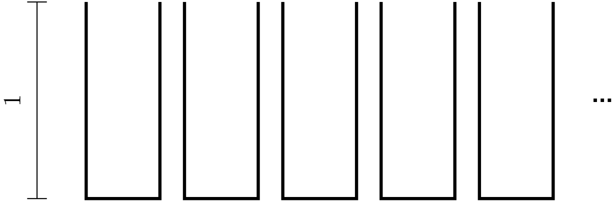
First Fit Decreasing (FFD)



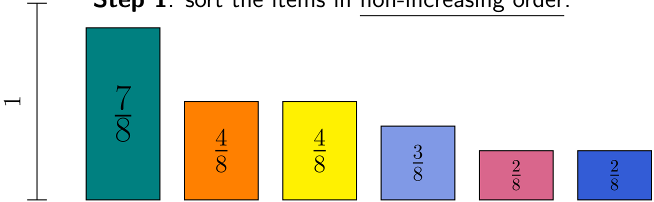
Step 1: sort the items in non-increasing order.



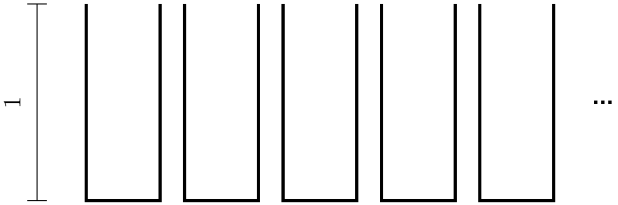
First Fit Decreasing (FFD)



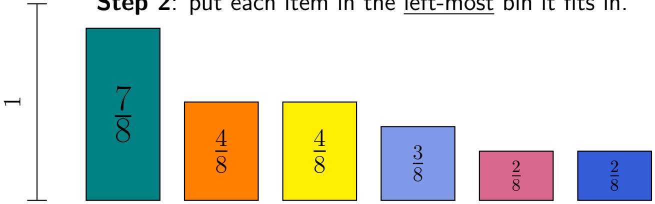
Step 1: sort the items in non-increasing order.



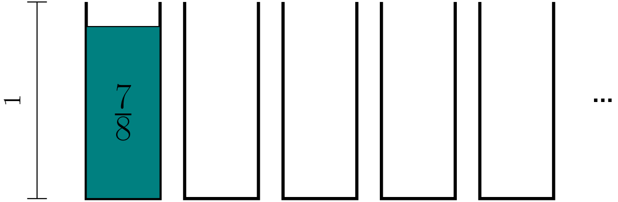
First Fit Decreasing (FFD)



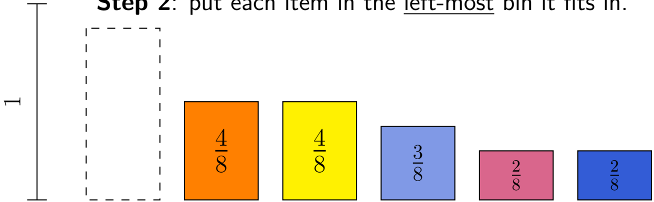
Step 2: put each item in the <u>left-most</u> bin it fits in.



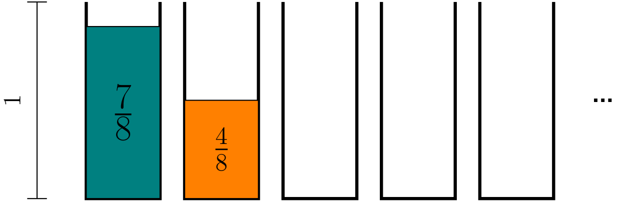
First Fit Decreasing (FFD)



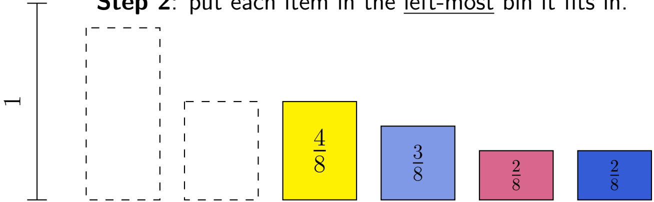
Step 2: put each item in the left-most bin it fits in.



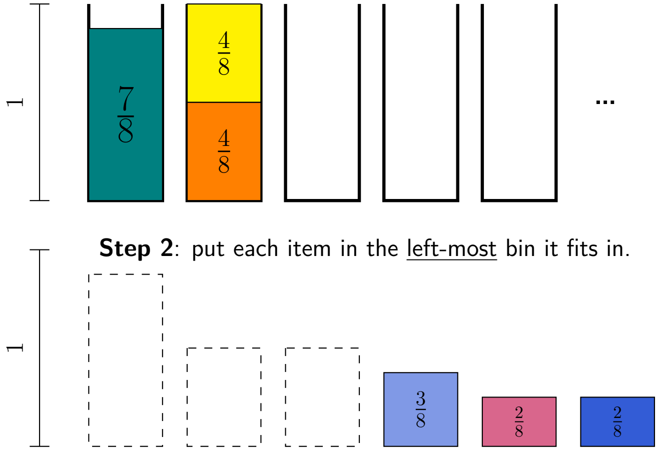
First Fit Decreasing (FFD)



Step 2: put each item in the left-most bin it fits in.

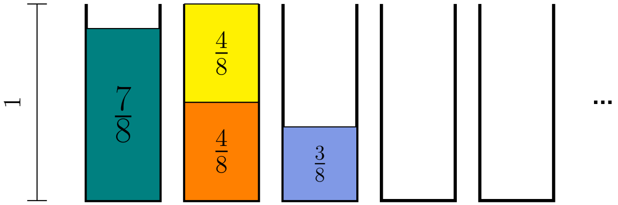


First Fit Decreasing (FFD)

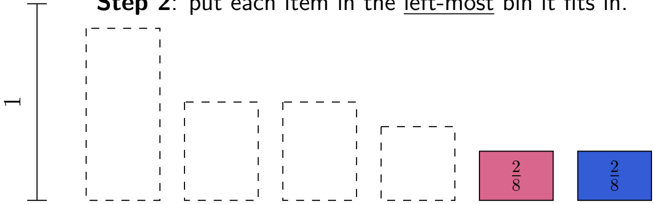


Step 2: put each item in the left-most bin it fits in.

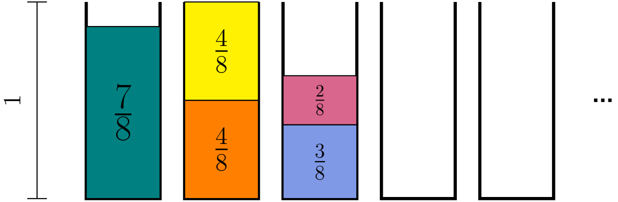
First Fit Decreasing (FFD)



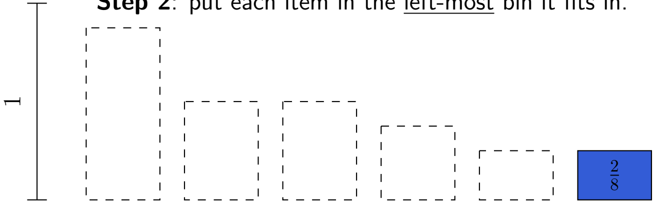
Step 2: put each item in the left-most bin it fits in.



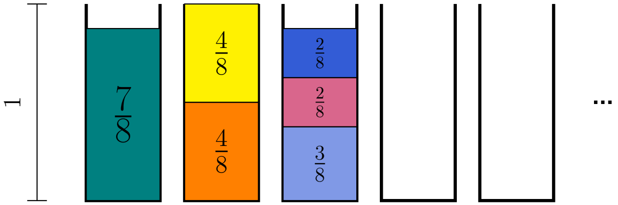
First Fit Decreasing (FFD)



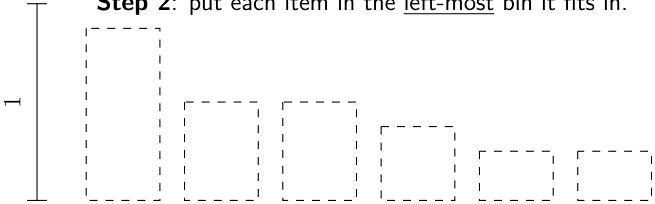
Step 2: put each item in the left-most bin it fits in.



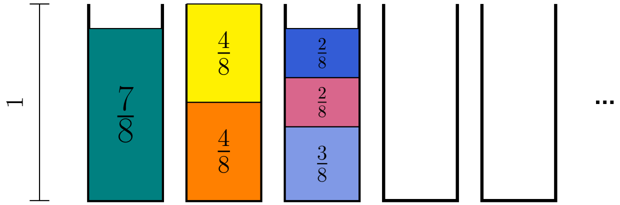
First Fit Decreasing (FFD)



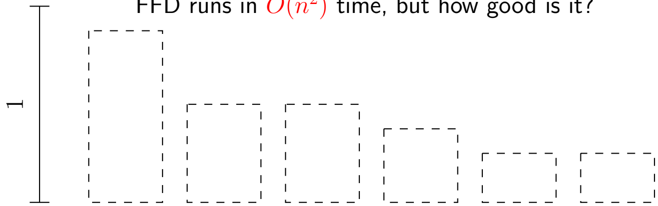
Step 2: put each item in the left-most bin it fits in.



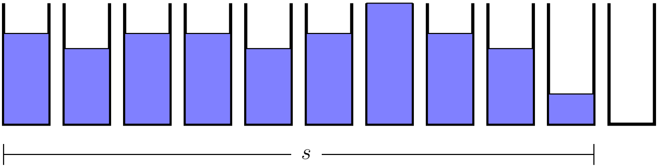
First Fit Decreasing (FFD)



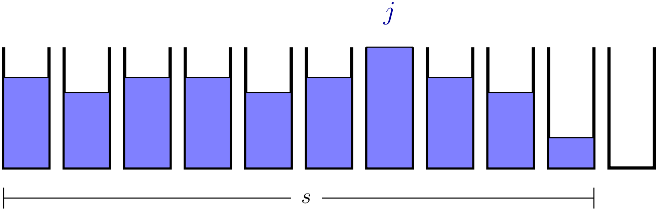
FFD runs in $O(n^2)$ time, but how good is it?



First Fit Decreasing (FFD)

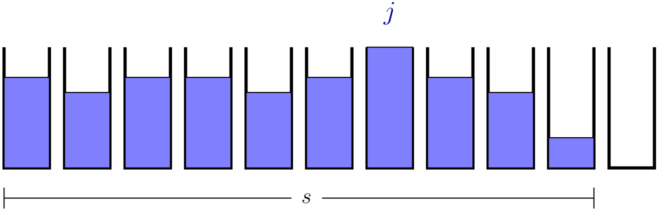


First Fit Decreasing (FFD)



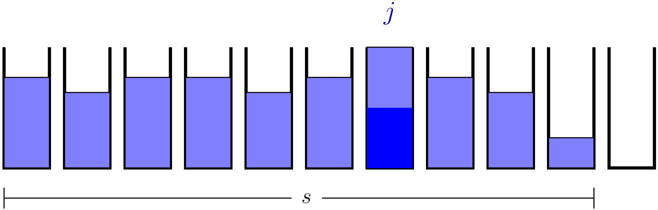
- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)

First Fit Decreasing (FFD)



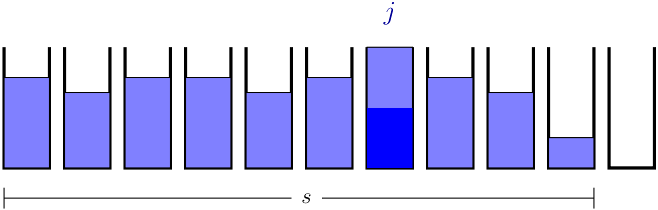
- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 1: bin j contains an item of size $> \frac{1}{2}$.

First Fit Decreasing (FFD)



- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 1: bin j contains an item of size $> \frac{1}{2}$.

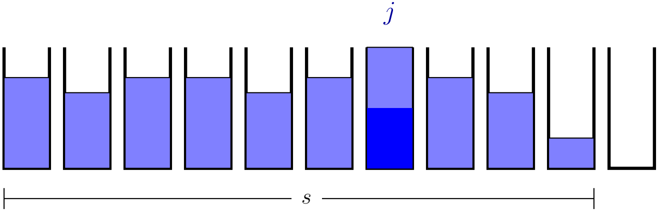
First Fit Decreasing (FFD)



- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 1: bin j contains an item of size $> \frac{1}{2}$.

Every bin $j' \leq j$ contains an item of size $> \frac{1}{2}$.

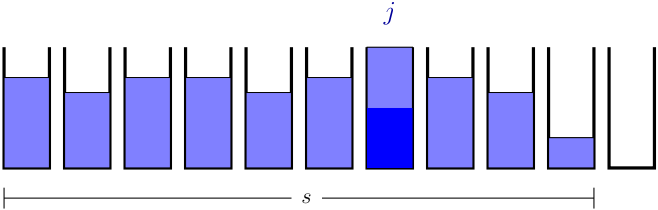
First Fit Decreasing (FFD)



- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 1: bin j contains an item of size $> \frac{1}{2}$.

Every bin $j' \leq j$ contains an item of size $> \frac{1}{2}$. [Because FFD packed big items first and each item was packed into the lowest numbered bin.]

First Fit Decreasing (FFD)

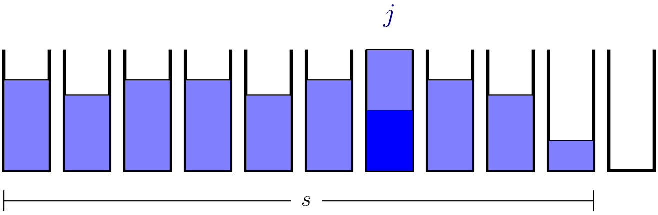


- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 1: bin j contains an item of size $> \frac{1}{2}$.

Every bin $j' \leq j$ contains an item of size $> \frac{1}{2}$. [Because FFD packed big items first and each item was packed into the lowest numbered bin.]

Each of these items has to be in a different bin even in Opt.

First Fit Decreasing (FFD)



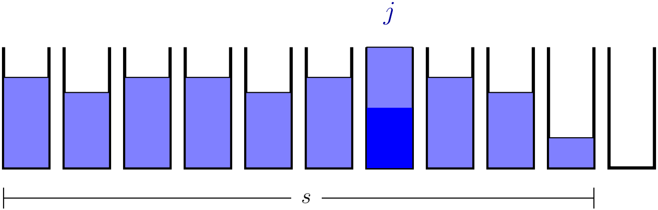
- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 1: bin j contains an item of size $> \frac{1}{2}$.

Every bin $j' \leq j$ contains an item of size $> \frac{1}{2}$. [Because FFD packed big items first and each item was packed into the lowest numbered bin.]

Each of these items has to be in a different bin even in Opt.

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

First Fit Decreasing (FFD)



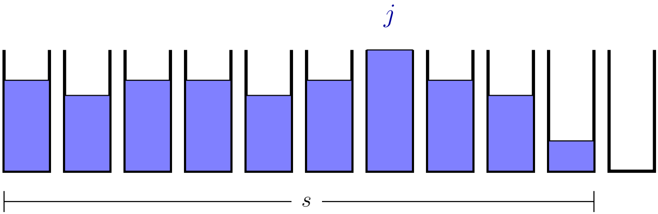
- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 1: bin j contains an item of size $> \frac{1}{2}$.

Every bin $j' \leq j$ contains an item of size $> \frac{1}{2}$. [Because FFD packed big items first and each item was packed into the lowest numbered bin.]

Each of these items has to be in a different bin even in Opt.

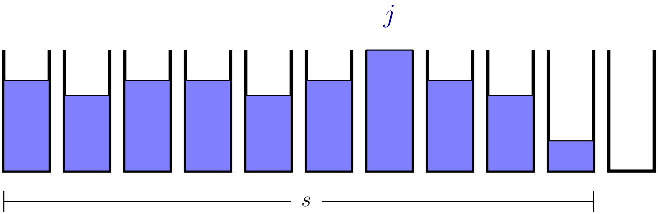
So Opt uses at least $\frac{2s}{3}$ bins. That is, $s \leq \frac{3Opt}{2}$.

First Fit Decreasing (FFD)



- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

First Fit Decreasing (FFD)

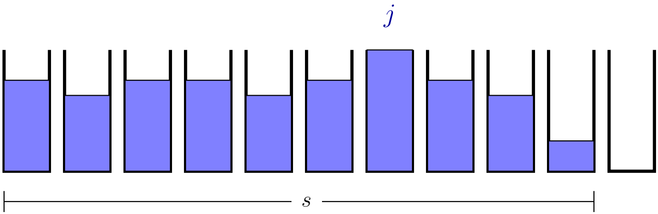


- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

When FFD packed the first item into bin j:

1. All bins $j, (j+1), \ldots, (s-2), (s-1)$ were empty.

First Fit Decreasing (FFD)

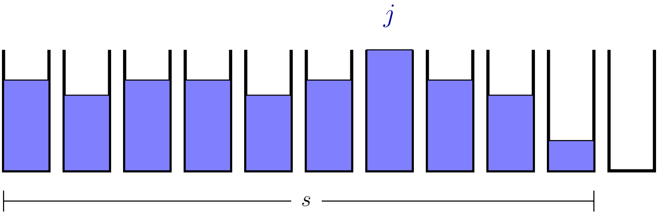


- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

When FFD packed the first item into bin j:

1. All bins $j, (j+1), \ldots, (s-2), (s-1)$ were empty.
2. All unpacked items had size $\leq \frac{1}{2}$. [Because FFD packed the items in non-increasing order.]

First Fit Decreasing (FFD)



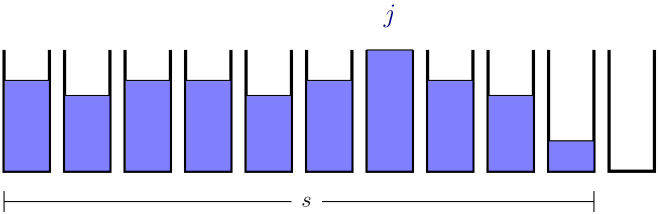
- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

When FFD packed the first item into bin j:

1. All bins $j, (j+1), \ldots, (s-2), (s-1)$ were empty.
2. All unpacked items had size $\leq \frac{1}{2}$. [Because FFD packed the items in non-increasing order.]

So bins $j, (j+1), \ldots, (s-2), (s-1)$ each contain **at least two items**. [We only use a new bin when the item won't fit in any previous bin.]

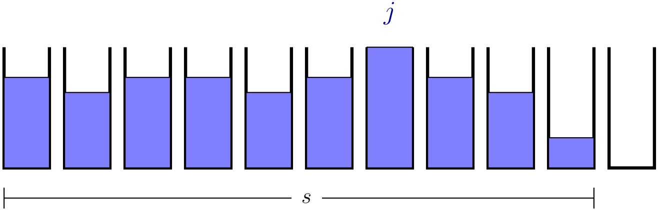
First Fit Decreasing (FFD)



- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

So bins $j, (j+1), \ldots, (s-2), (s-1)$ each contain **at least two items**. [We only use a new bin when the item won't fit in any previous bin.]

First Fit Decreasing (FFD)

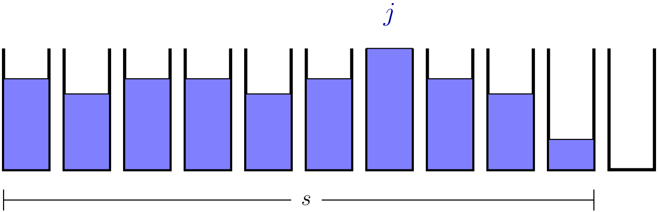


- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

So bins $j, (j+1), \ldots, (s-2), (s-1)$ each contain **at least two items**. [We only use a new bin when the item won't fit in any previous bin.]

Bin s contains at least one item.

First Fit Decreasing (FFD)

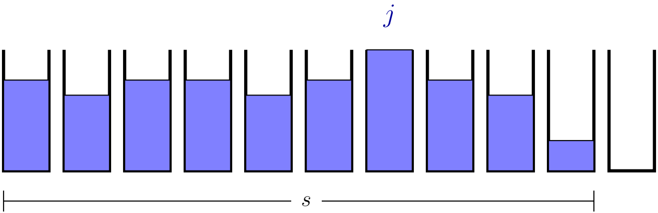


- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

So bins $j, (j+1), \ldots, (s-2), (s-1)$ each contain **at least two items**. [We only use a new bin when the item won't fit in any previous bin.]

Bin s contains at least one item. This gives a total of $2(s-j)+1$ items, none of which fit into bins $1, 2, 3, \ldots, (j-1)$.
[Otherwise FFD would have packed them there.]

First Fit Decreasing (FFD)



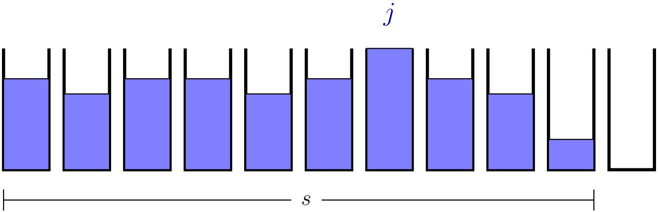
- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

So bins $j, (j+1), \ldots, (s-2), (s-1)$ each contain **at least two items**. [We only use a new bin when the item won't fit in any previous bin.]

Bin s contains at least one item. This gives a total of $2(s-j)+1$ items, none of which fit into bins $1, 2, 3, \ldots, (j-1)$.
[Otherwise FFD would have packed them there.]

So $I > \min(j-1, 2(s-j)+1)$ [where I is the total weight of all items].

First Fit Decreasing (FFD)



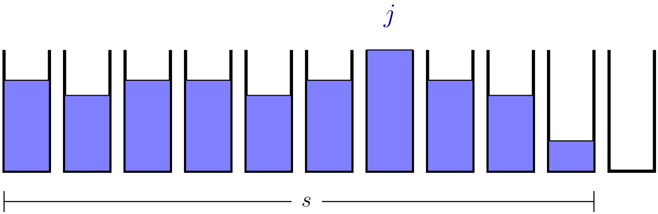
- Consider bin $j = \lceil \frac{2s}{3} \rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

So bins $j, (j+1), \ldots, (s-2), (s-1)$ each contain **at least two items**. [We only use a new bin when the item won't fit in any previous bin.]

Bin s contains at least one item. This gives a total of $2(s-j)+1$ items, none of which fit into bins $1, 2, 3, \ldots, (j-1)$.
[Otherwise FFD would have packed them there.]

So $I > \min(j-1, 2(s-j)+1) \geq \lceil \frac{2s}{3} \rceil - 1$ [Plugging in $j = \lceil \frac{2s}{3} \rceil$.]

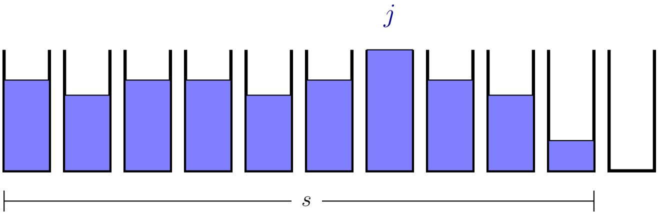
First Fit Decreasing (FFD)



- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

So, since $\left\lceil \frac{2s}{3} \right\rceil - 1 < I$, and $I \leq Opt$, we have $\left\lceil \frac{2s}{3} \right\rceil - 1 < Opt$.

First Fit Decreasing (FFD)

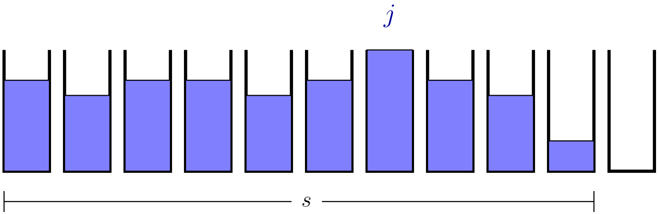


- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

So, since $\left\lceil \frac{2s}{3} \right\rceil - 1 < I$, and $I \leq Opt$, we have $\left\lceil \frac{2s}{3} \right\rceil - 1 < Opt$.

But both sides are **integers**, so $\left\lceil \frac{2s}{3} \right\rceil \leq Opt$.

First Fit Decreasing (FFD)



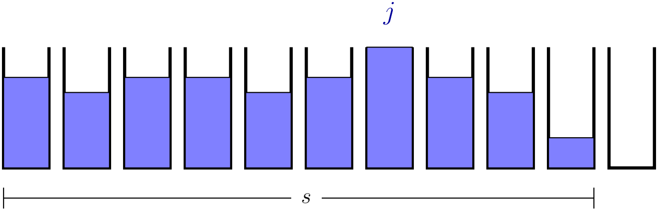
- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

So, since $\left\lceil \frac{2s}{3} \right\rceil - 1 < I$, and $I \leq Opt$, we have $\left\lceil \frac{2s}{3} \right\rceil - 1 < Opt$.

But both sides are **integers**, so $\left\lceil \frac{2s}{3} \right\rceil \leq Opt$.

Finally, $\frac{2s}{3} \leq \left\lceil \frac{2s}{3} \right\rceil \leq Opt$, so $s \leq \frac{3}{2} Opt$.

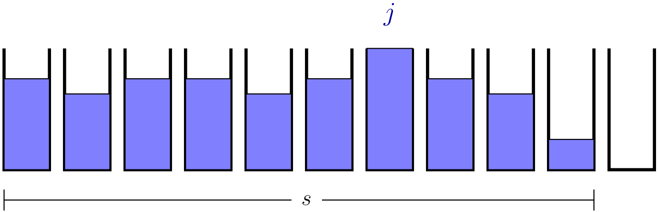
First Fit Decreasing (FFD)



- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 1: bin j contains an item of size $> \frac{1}{2}$.
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

In **both cases** we have, $s \leq \frac{3}{2} Opt$.

First Fit Decreasing (FFD)



- Consider bin $j = \left\lceil \frac{2s}{3} \right\rceil$ (s is the number of bins used by FFD)
  - Case 1: bin j contains an item of size $> \frac{1}{2}$.
  - Case 2: bin j contains only items of size $\leq \frac{1}{2}$.

In **both cases** we have, $s \leq \frac{3}{2} Opt$.

So FFD is a $\frac{3}{2}$-approximation algorithm of Bin Packing.

Approximation Algorithms: Summary

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

- A runs in polynomial time.
- Always outputs a solution with value s within an α factor of Opt. [Here P is an optimisation problem with optimal solution value Opt.]

If P is a maximisation problem $\frac{Opt}{\alpha} \leq s \leq Opt$.

If P is a minimisation problem $Opt \leq s \leq \alpha Opt$.

We have seen Next Fit, which is a 2-approximation algorithm for Bin Packing, which runs in $O(n)$ time.

We have seen First Fit Decreasing, which is a $\frac{3}{2}$-approximation algorithm for Bin Packing, which runs in $O(n^2)$ time.

Bin Packing is known to be **NP-Hard** so solving this exactly in polynomial time would prove that **P=NP**.

Further Reading:

Optional:

1. **David P. Williamson, David D. Shmoys**
   "The Design of Approximation Algorithms"
   https://www.designofapproxalgs.com/

Acknowledgements: Partly based on earlier COMP 1201 slides by Dr Benjamin Sach and Dr Baharak Rastegari.