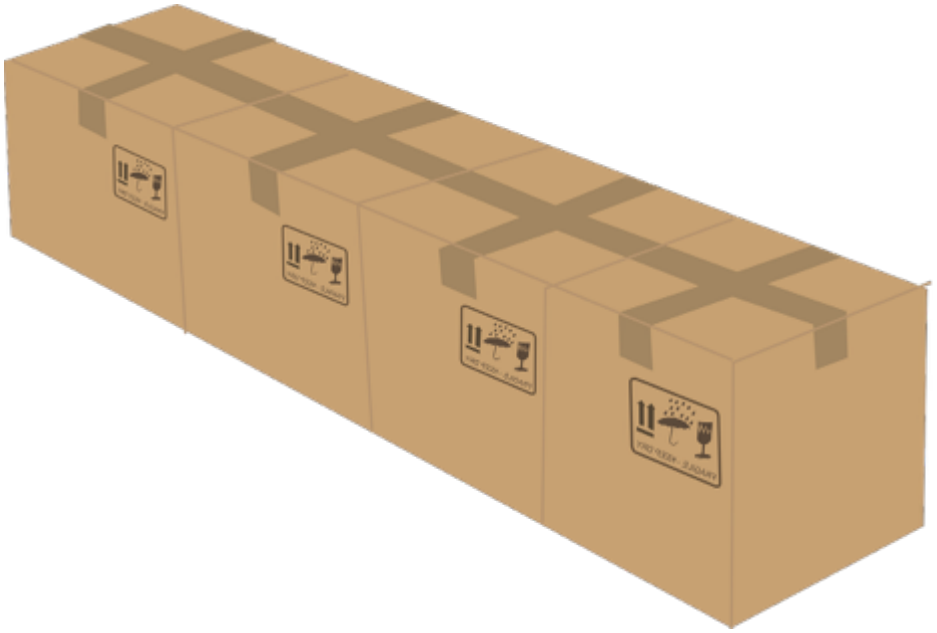
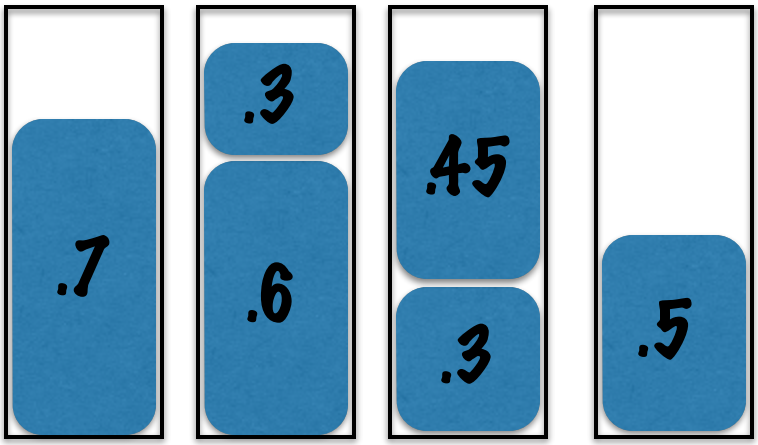
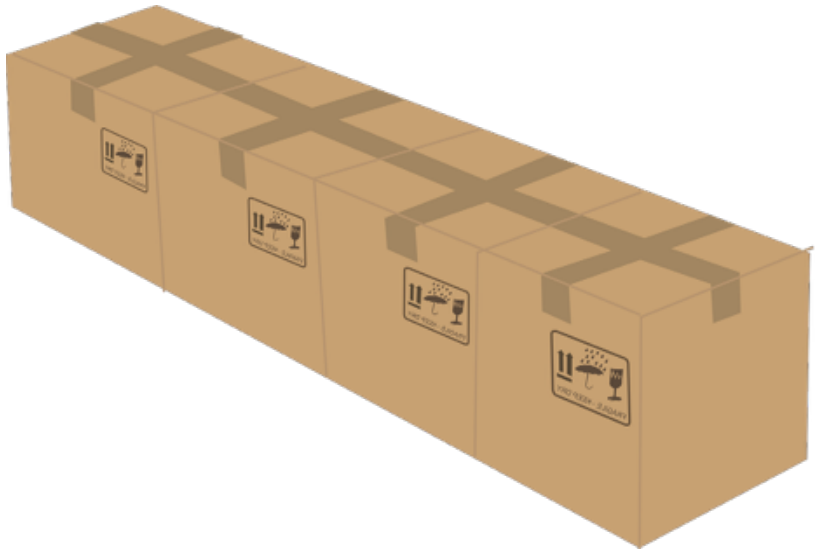
Bin packing, linear programming and rounding



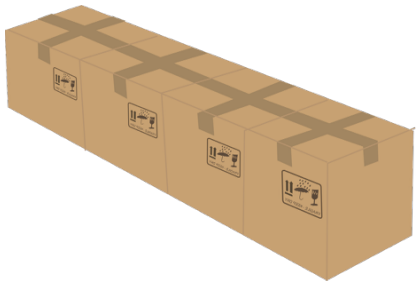


The bin packing problem

Pack your items
using
as few bins as possible



Given n items
item i has size si < 1
pack items into the fewest
unit capacity bins

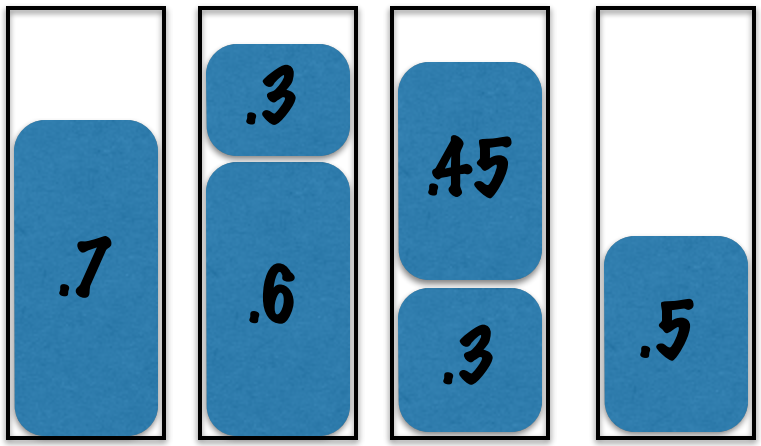


The Next Fit algorithm

One bin at a time:
If next item does not fit, close the bin and open a new bin

"Next Fit" algorithm

s1 =.7
s2=.6
s3=.4
s4=.3
s5=.45
s6=.5



can this instance be packed better?

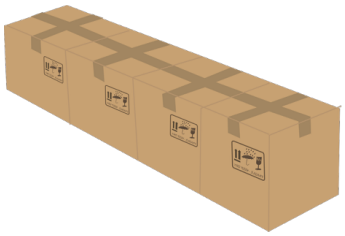
How good is Next Fit?

Capacity 100. Items:

91 50 21 90 39 43 54 95 25 14 25 61 35
28 97 16 18 42 99 95 12 7 21 23 42 50 77
100 79 36 31 71 35 1 63 6 13 92 58 73 72
32 37 62 54 18 25 9 52 93

Next Fit:

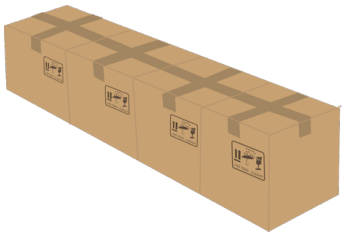
91 | 50 21 | 90 | 39 43 | 54 | 95 | 25 14 25 | 61 35 | 28 | 97 | 16 18 42 | 99 | 95 | 12 7 21 | 23 | 42 50 | 77 | 100 | 79 | 36 31 | 71 | 35 1 63 | 6 13 | 92 | 58 | 73 | 72 | 32 37 | 62 | 54 | 18 25 9 52 | 93



used 31 bins

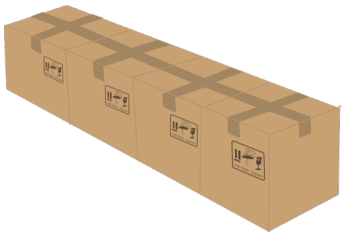
91 50 21 90 39 43 54 95 25 14 25 61 35 28 97 16 18 42 99 95 12 7 21 23
42 50 77 100 79 36 31 71 35 1 63 6 13 92 58 73 72 32 37 62 54 18 25 9 52
93

bin 7: (25+14+25)
next item: 61, but
(25+14+25) +61>100
so, close bin 7, open bin 8,
put item 61 in bin 8.



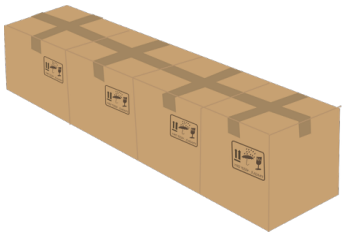
91 50 21 90 39 43 54 95 25 14 25 61 35 28 97 16 18 42 99 95 12 7 21 23
42 50 77 100 79 36 31 71 35 1 63 6 13 92 58 73 72 32 37 62 54 18 25 9 52
93

In general:
(items in bin 2i-1)+next item > 100
(items in bins 2i-1 or 2i) >100
31 bins: total item sizes > 15*100



91 50 21 90 39 43 54 95 25 14 25 61 35 28 97 16 18 42 99 95 12 7 21 23
42 50 77 100 79 36 31 71 35 1 63 6 13 92 58 73 72 32 37 62 54 18 25 9 52
93

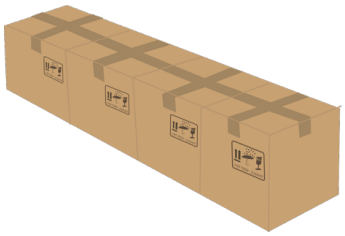
In general:
k bins by Next Fit
Total item sizes > (k-1)/2 * 100



91 50 21 90 39 43 54 95 25 14 25 61 35 28 97 16 18 42 99 95 12 7 21 23
42 50 77 100 79 36 31 71 35 1 63 6 13 92 58 73 72 32 37 62 54 18 25 9 52
93

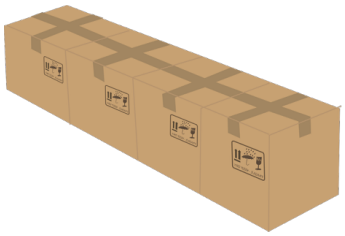
What about OPT?

Total item sizes < OPT*100



91 50 21 90 39 43 54 95 25 14 25 61 35 28 97 16 18 42 99 95 12 7 21 23
42 50 77 100 79 36 31 71 35 1 63 6 13 92 58 73 72 32 37 62 54 18 25 9 52
93

Combining:
k bins by Next Fit
OPT*100> (k-1)/2 * 100
#(bins of Next Fit)< 2*OPT +1



Asymptotic 2 approximation

Is this tight?

Example with
OPT=501 bins,
Next Fit=1000 bins

What about non-asymptotic?

Distinguishing between OPT=2 and OPT=3 is NP-hard

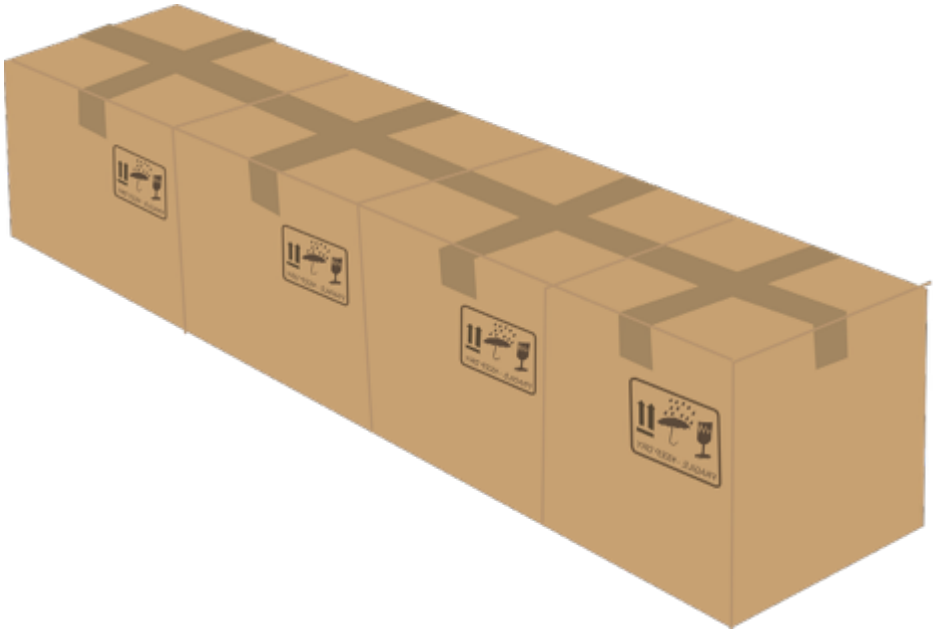
What have we learned?

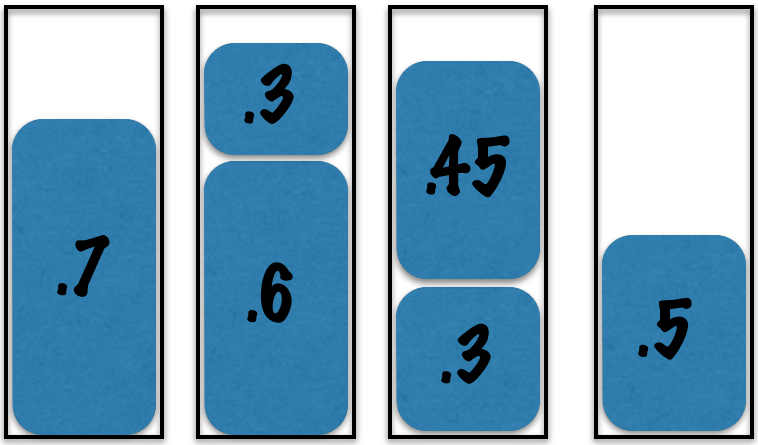
1. Crude algorithms can give good bounds

Message: first try the simplest algorithm

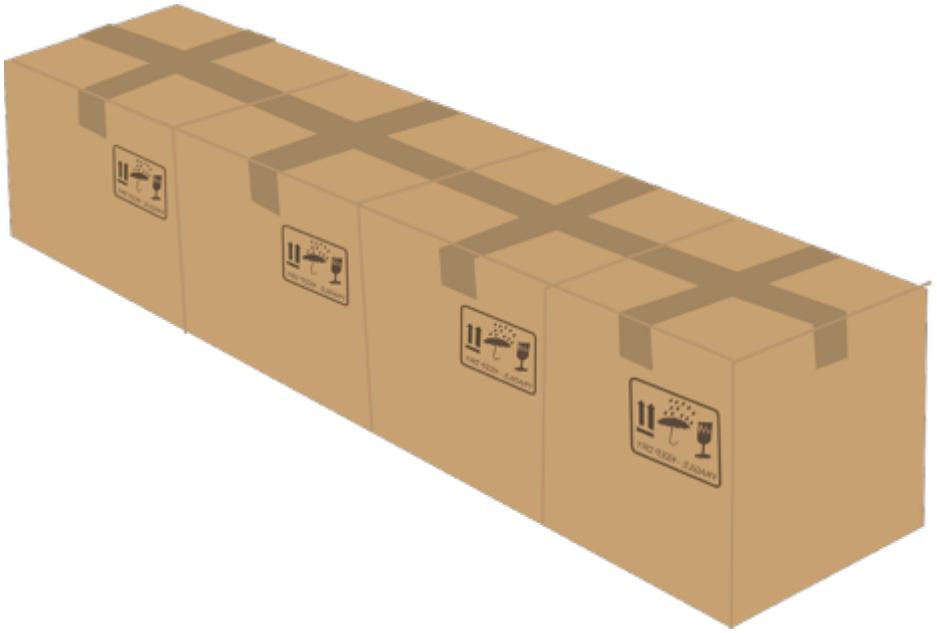
2. Analysis: for intuition, first execute it on some concrete examples

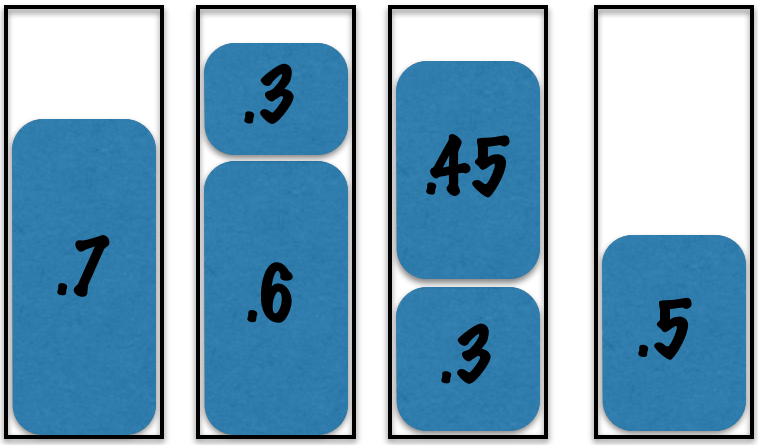
Bin packing, linear programming and rounding





Bin packing, linear programming and rounding





Can we do better than Next Fit?

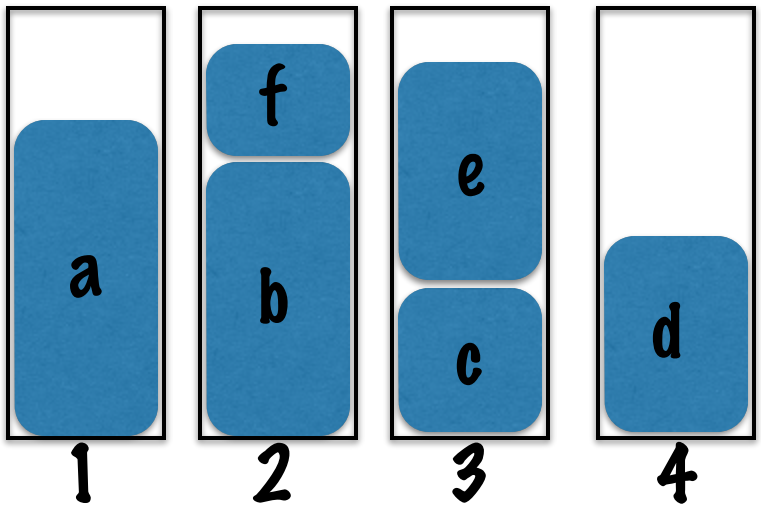
First tool: linear programming relaxation

An integer program

Given n items and K unit bins, is there a packing?

Variables: $x_{ij} \in \{0, 1\}$

$x_{ij} = 1$ iff item i is placed in bin j



$$x_{a1} = x_{b2} = x_{f2} = x_{c3} = x_{e3} = x_{d4} = 1,$$

$x_{ij} = 0$ otherwise

An integer program

Constraint: every item must go somewhere

Item b must go into bin 1,2,3 or 4

$$x_{b1} + x_{b2} + x_{b3} + x_{b4} = 1$$

An integer program

Constraint: must not exceed bin capacity

Item sizes in bin j sum to at most 1.

$$x_{aj}s_a + x_{bj}s_b + \cdots + x_{fj}s_f \leq 1$$

Integer program

n items, K bins

x_{ij} = whether item i goes into bin j

$\forall i: \ \sum_j x_{ij} = 1$

$\forall j: \ \sum_i x_{ij} s_i \leq 1$

$\forall i, j: \ x_{ij} \in \{0, 1\}$

feasible iff items can be packed into K bins

Linear programming relaxation

$$
\begin{aligned}
& n \text{ items, } K \text{ bins} \\
& \forall i: \ \textstyle\sum_j x_{ij} = 1 \\
& \forall j: \ \textstyle\sum_i x_{ij} s_i \leq 1 \\
& \forall i,j: \ 0 \leq x_{ij} \leq 1
\end{aligned}
$$

Algorithm: use the LP relaxation to pack items (somehow)

How good is the relaxation?

A bad example

n items, K bins

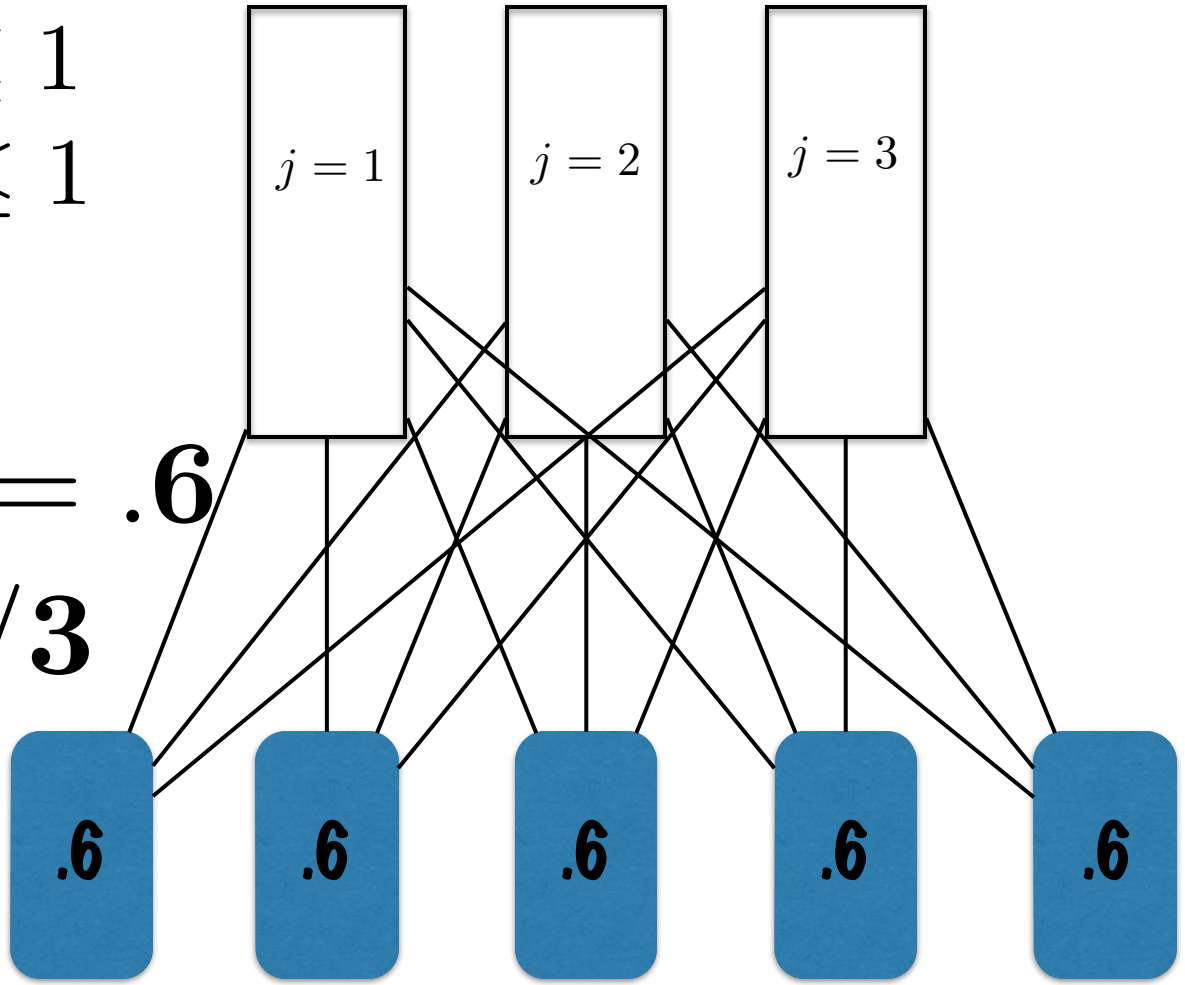
$$\forall i: \sum_j x_{ij} = 1$$
$$\forall j: \sum_i x_{ij} s_i \leq 1$$
$$\forall i,j: 0 \leq x_{ij} \leq 1$$

$$\mathbf{s_1 = \ldots = s_5 = .6}$$

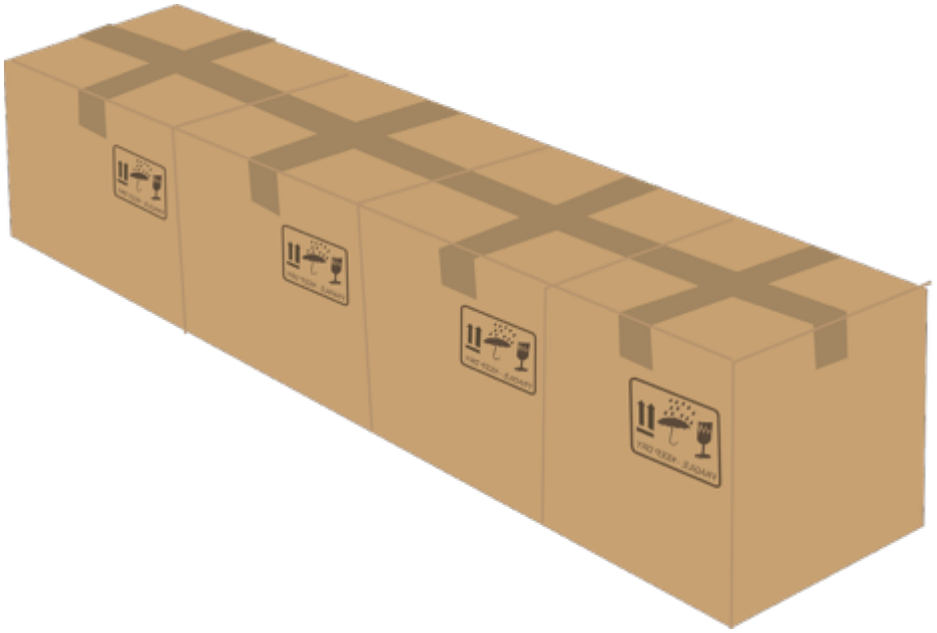
$$\mathbf{\forall i,j \;\; x_{ij} = 1/3}$$

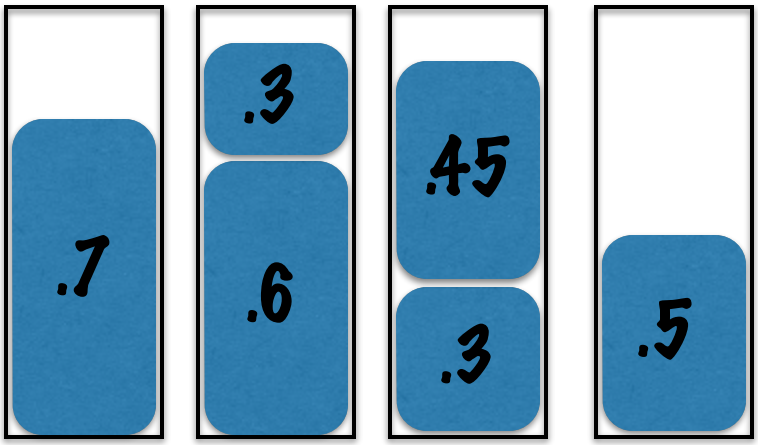
LP: 3 bins

OPT: 5 bins

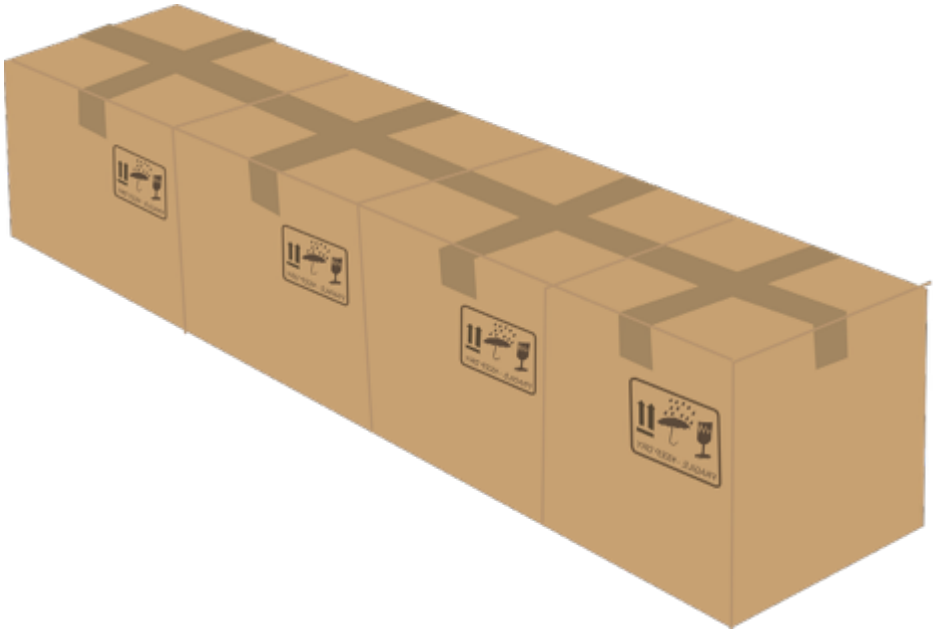


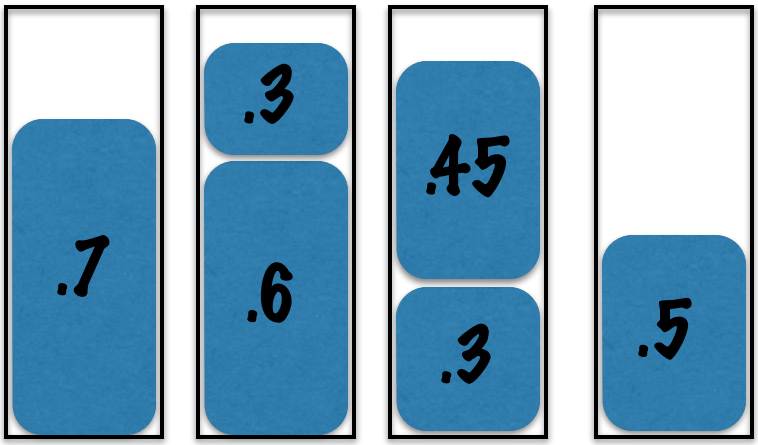
Bin packing, linear programming and rounding





Bin packing, linear programming and rounding





Remember:

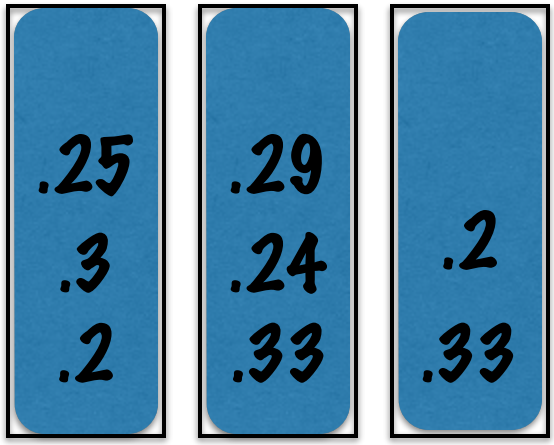
To analyze output vs. OPT, focus on LP value...

Try next meta-tool: special cases

What if items are smaller than 1/3*capacity?

.2,.3,.25,.33,.24,.29,.33,.2,...

What does **Next Fit** do?



.2+.3+.25=.75

.33+.24+.29=.86

.33+.2=.53

The next item will fit in bin 3:

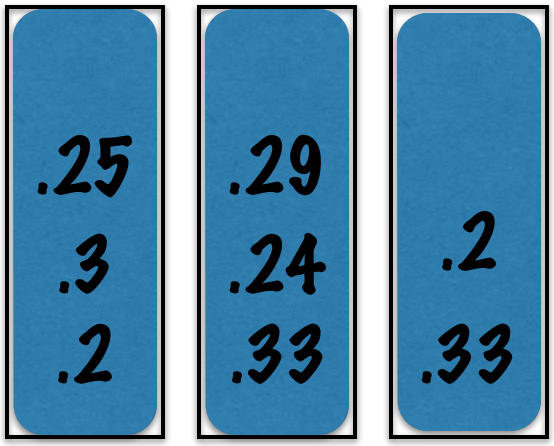
.53+(something less than 1/3) <1

Next Fit when items are smaller than 1/3*capacity

Bin filled to < 2/3 : next item fits

so:

only close bin when filled to > 2/3



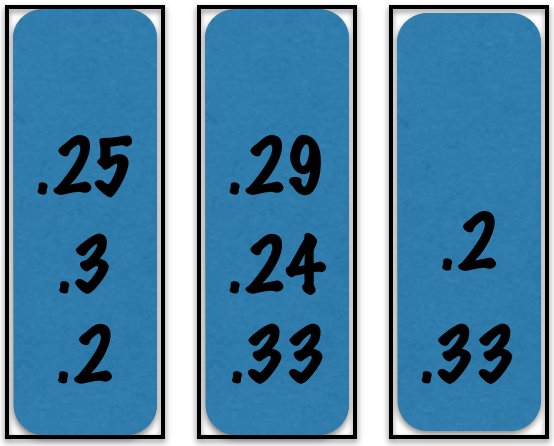
All bins except last are filled to > 2/3

Next Fit when items are smaller than 1/3*capacity

All bins except last are filled to > 2/3

Total size > 2/3 * (#bins -1)

But Total size < OPT

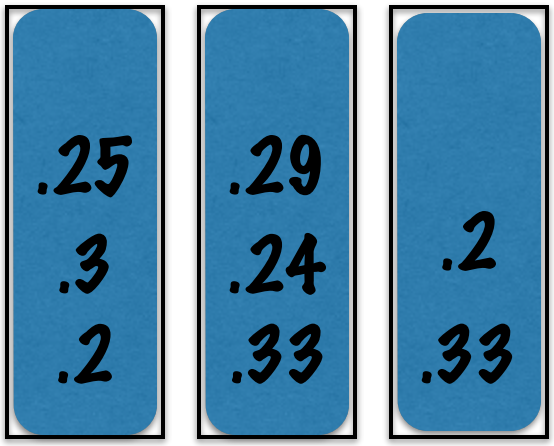


Combine:

#bins < 3/2 * OPT + 1

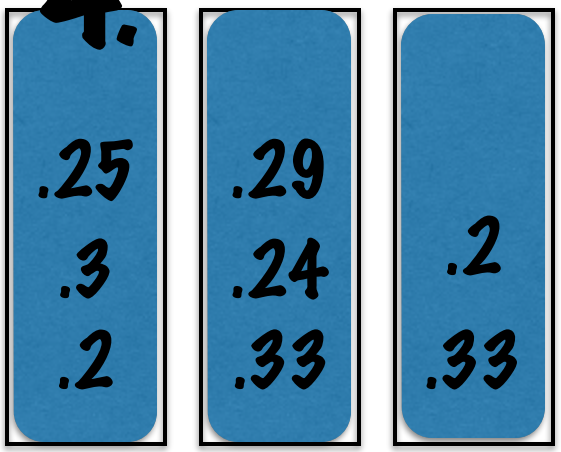
Theorem:

when items are smaller than 1/3*capacity, Next Fit uses at most 1+ (3/2) OPT bins

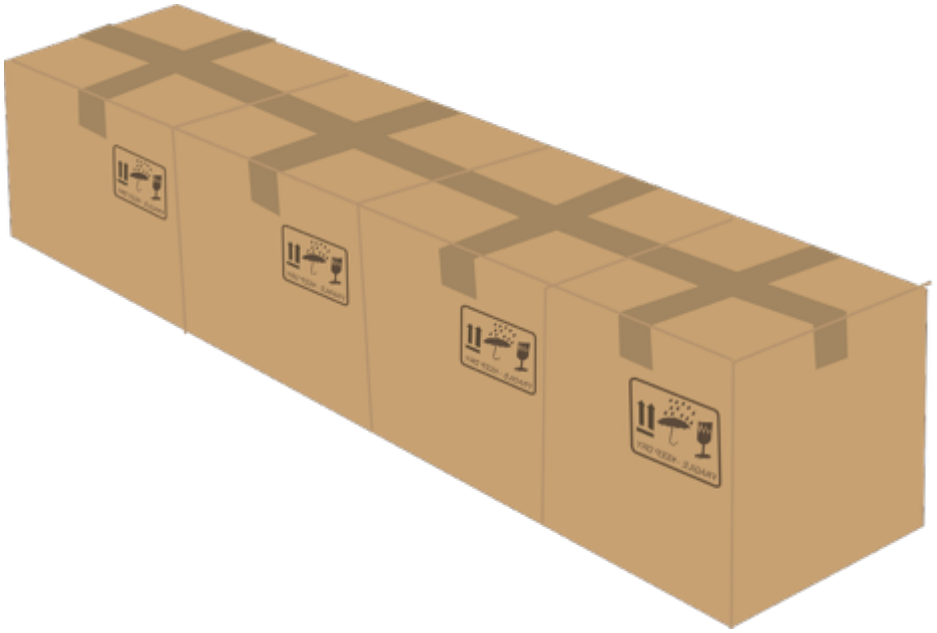


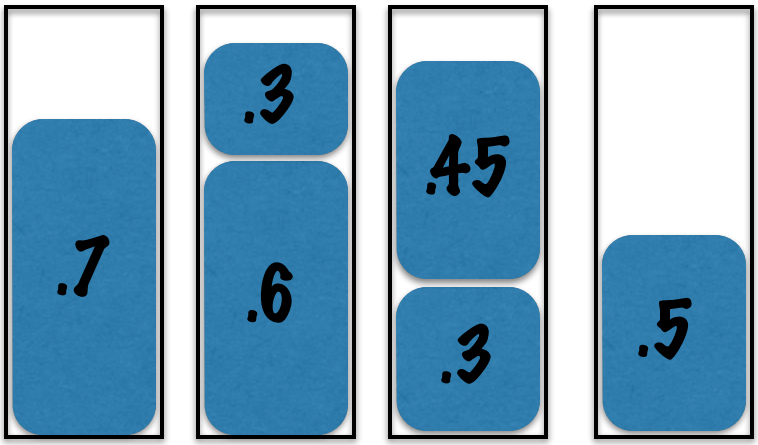
Message

1. From example to structural observation
2. With observation, upper bound algorithm
3. With different argument, lower bound OPT
4. Combine

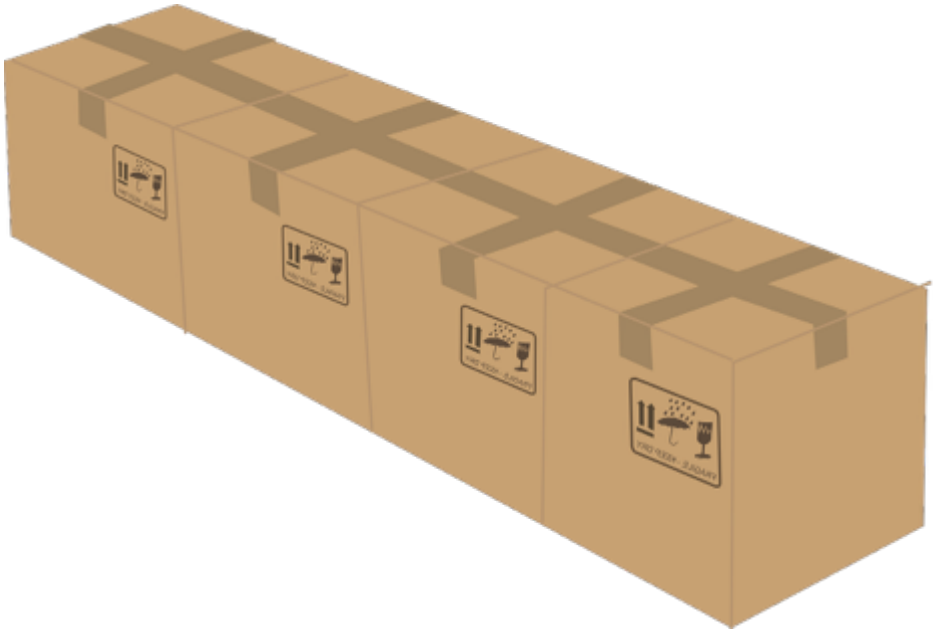


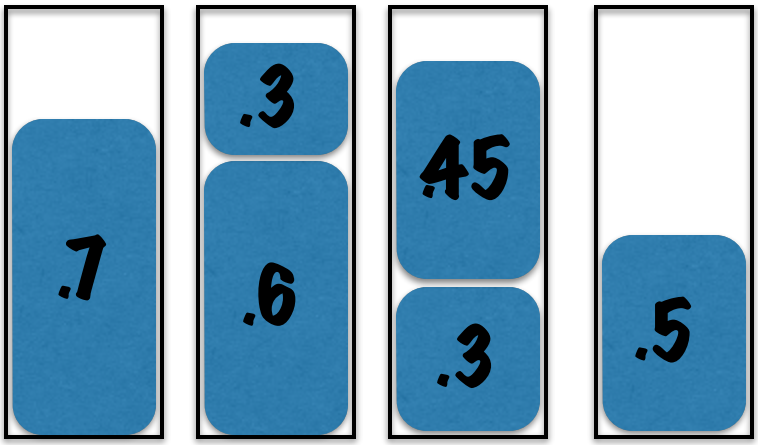
Bin packing, linear programming and rounding





Bin packing, linear programming and rounding





Meta-tool: special cases

Large items

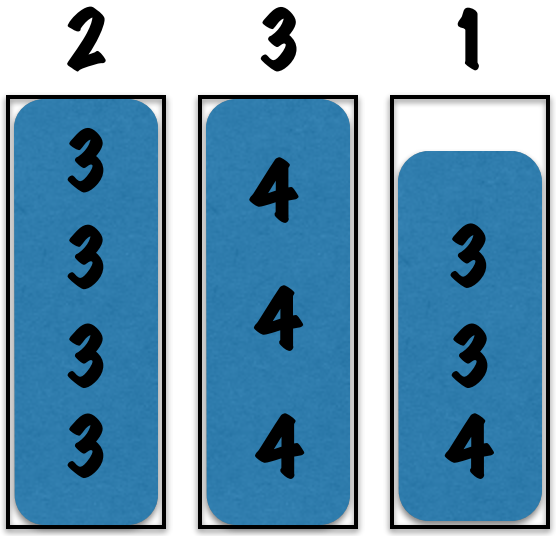
Special special case

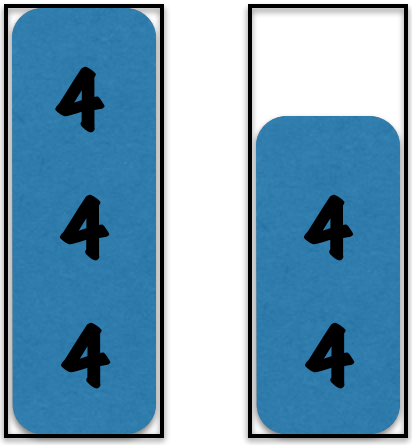
Large items,
few distinct sizes

Example

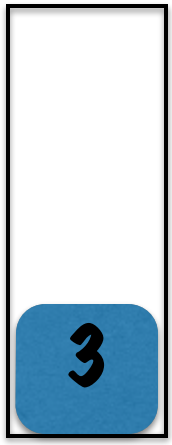
Bin capacity 12
sizes: { 3, 4 }
10 items of size 3,
10 items of size 4.

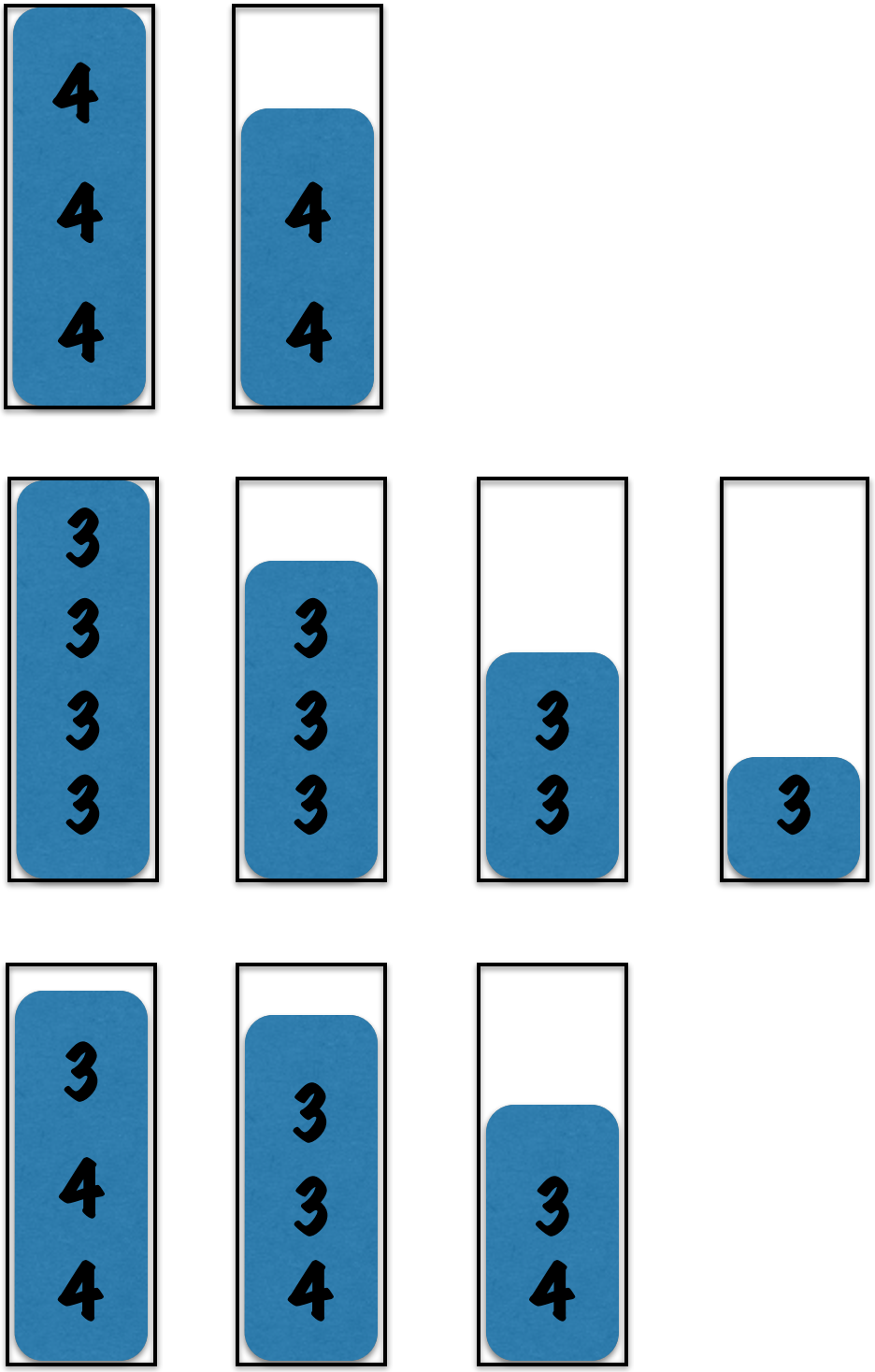
Bin capacity 12
sizes: { 3, 4 }
10 items of size 3,
10 items of size 4.





Observe:
few configurations





Large items, few distinct sizes C={configurations}

In configuration c size s occurs $a_{s,c}$ times

Integer program

Input: S={size}

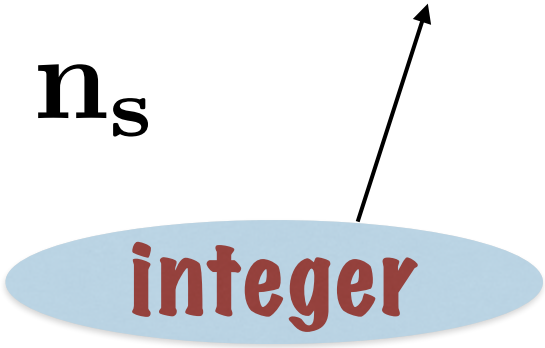
number of items of size s: n_s

Output: C={configurations}

number of bins in configuration c: x_c

Constraints: $\sum_c a_{s,c} x_c \geq n_s$

Number of bins: $\sum_c x_c$



If size > capacity/10 then:
< 10 items per configuration

If < 10 sizes then:
< 10^10 configurations

Solve LP relaxation
10 constraints, 10^10 variables
Round up to nearest integer

#bins < OPT + 10^10

(Exhaustive search also ok)

For every $(x_c)_{c \in \mathcal{C}} \in \{0, 1, \cdots, n\}^{|\mathcal{C}|}$

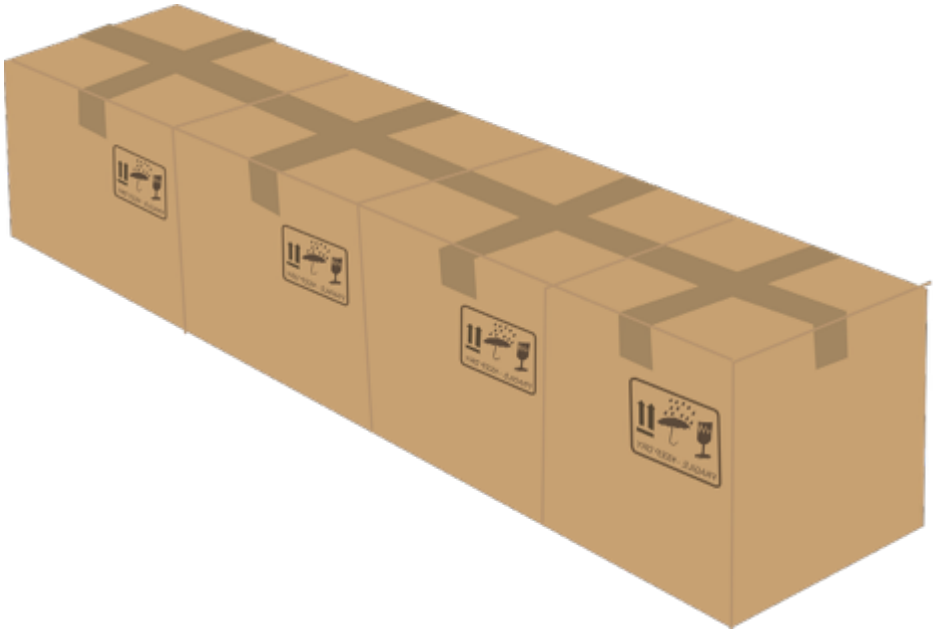
Check whether, for every size s,
enough slots for items of size s

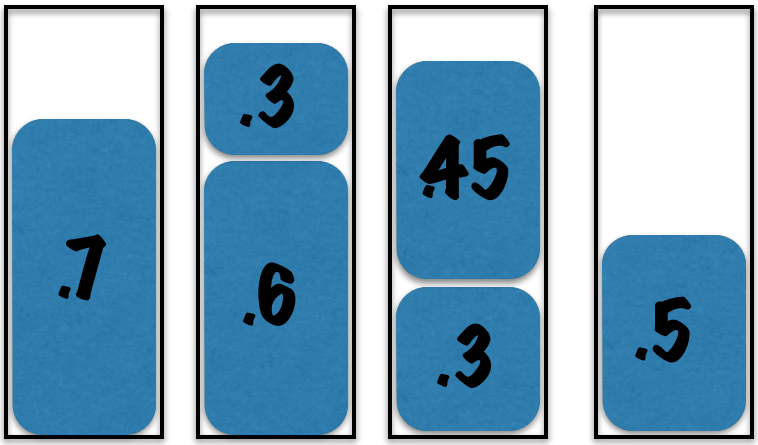
Output solution with min #bins

Runtime if size>capacity/10:

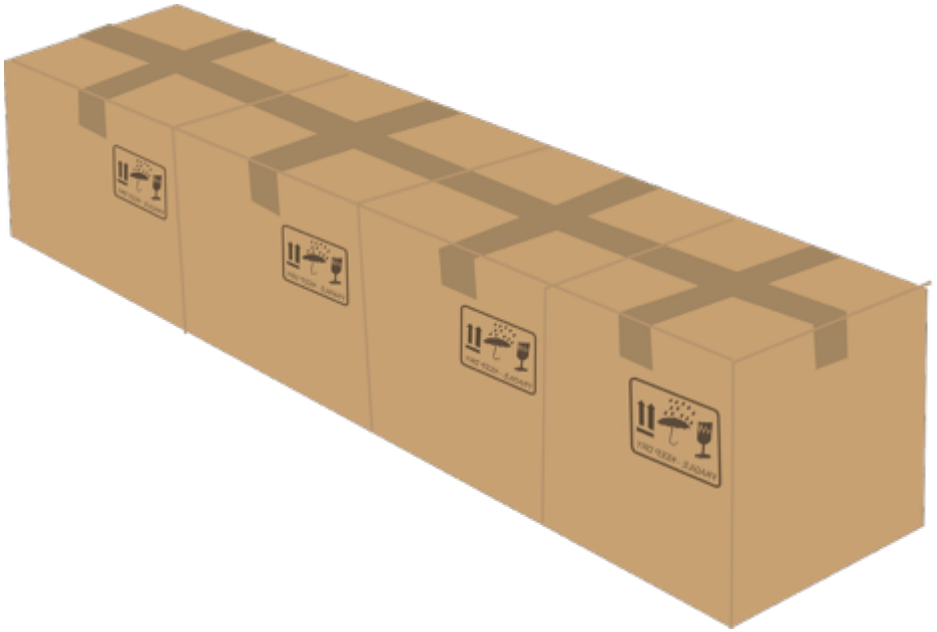
$$|S| \times n^{|S|^{10}}$$

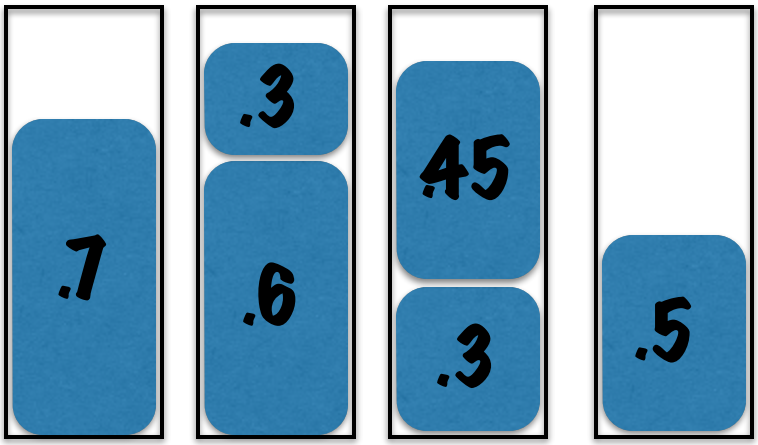
Bin packing, linear programming and rounding





Bin packing, linear programming and rounding





Less special special case

Large items, many sizes

Idea: Rounding

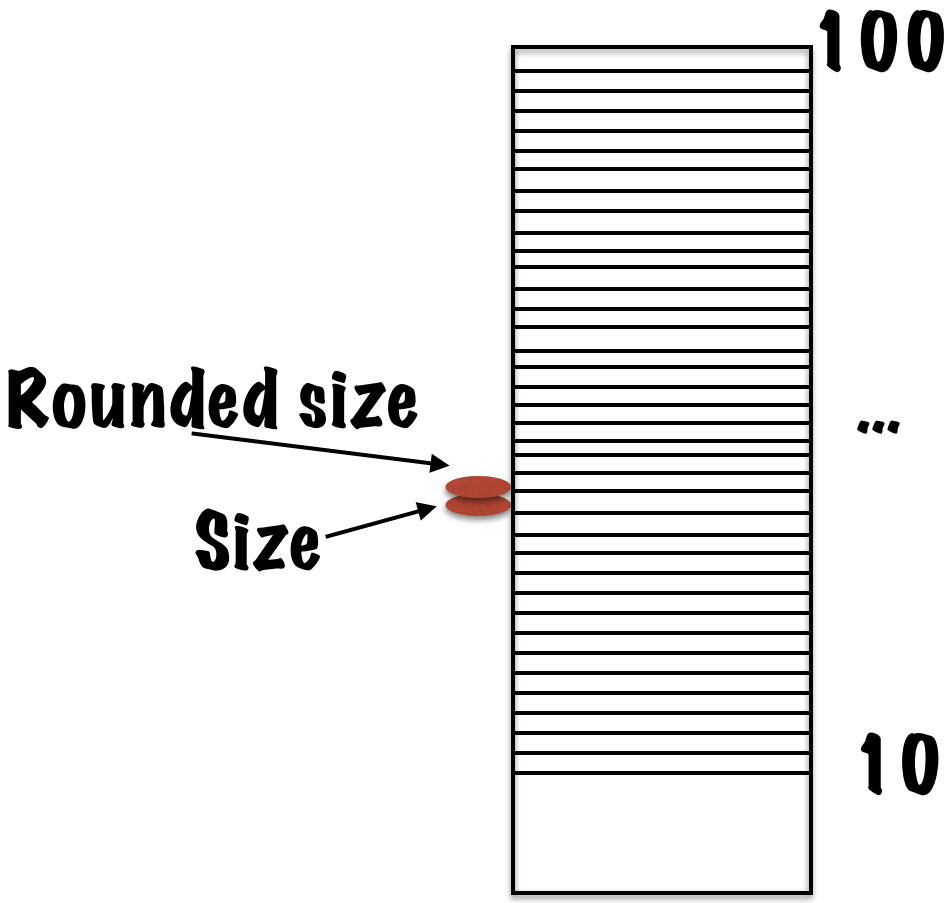
Round sizes
Reduce to special special case

How to round?

Capacity = 100 & Min size > 10:
Attempt:
Round sizes up to nearest integer
Solve rounded problem

Observe:
It's a solution to original problm

But how good is it?



Size increases a little bit

How much does OPT change?

Input:

Capacity 100

50 items with size 100*(1/3)

50 items with size 100*(2/3)

OPT=50

Rounded input:

Capacity 100

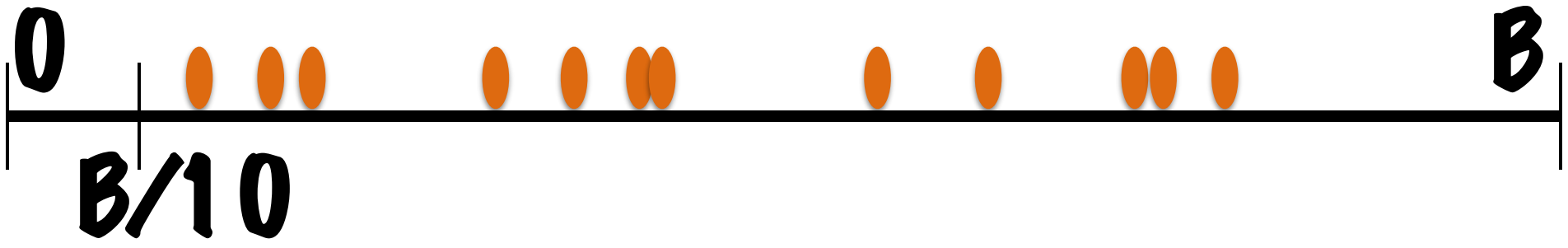
50 items with size 34

50 items with size 67

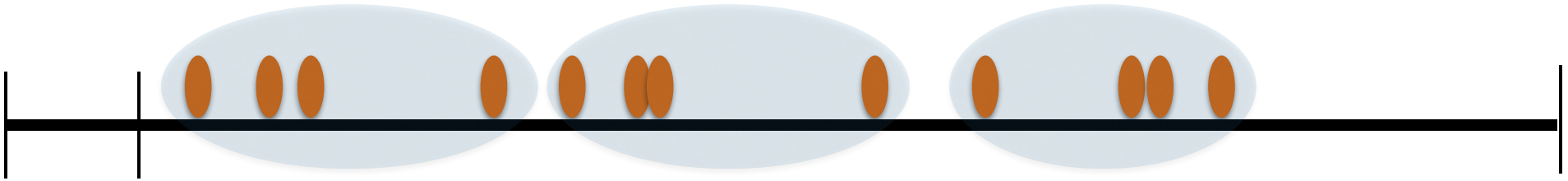
OPT'=75

This rounding fails

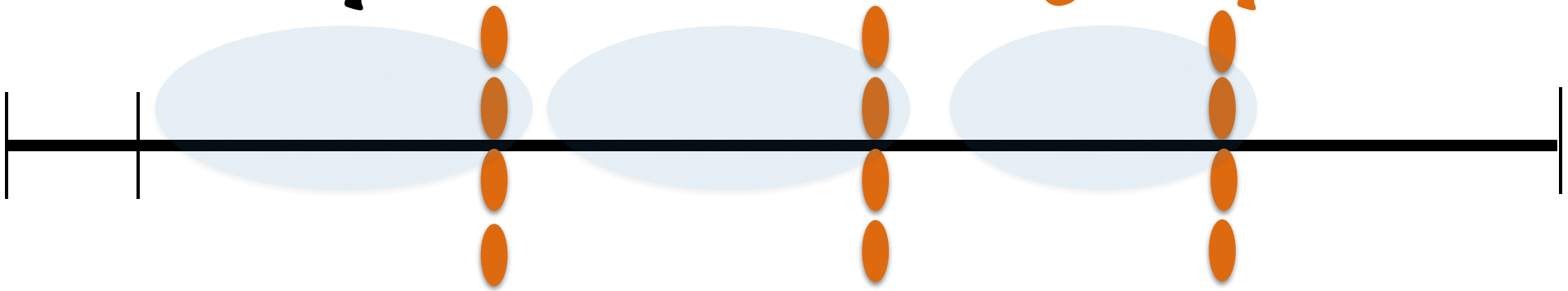
Adaptive rounding



Make groups of equal cardinality



Round up to max size in group



Algorithm - large items

Assume: sizes > capacity * ϵ

Sort sizes

Make groups of cardinality $n \times \epsilon^2$

Round up to max size in group

Solve rounded problem

Output corresponding packing

Observe: output is a packing

Observe: all sizes are $> \text{Capacity} \times \epsilon$

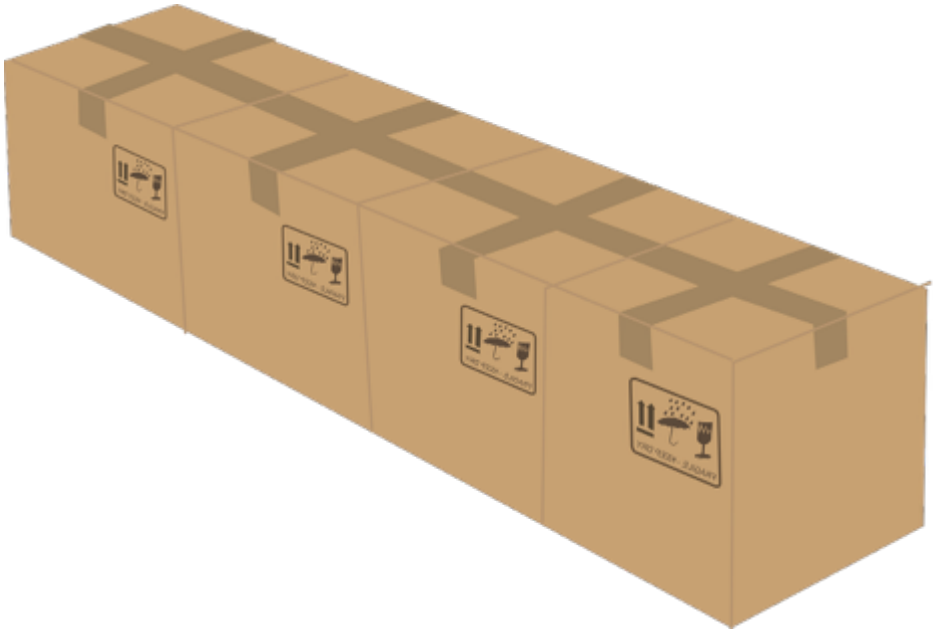
Observe: #distinct sizes $< 1/\epsilon^2$

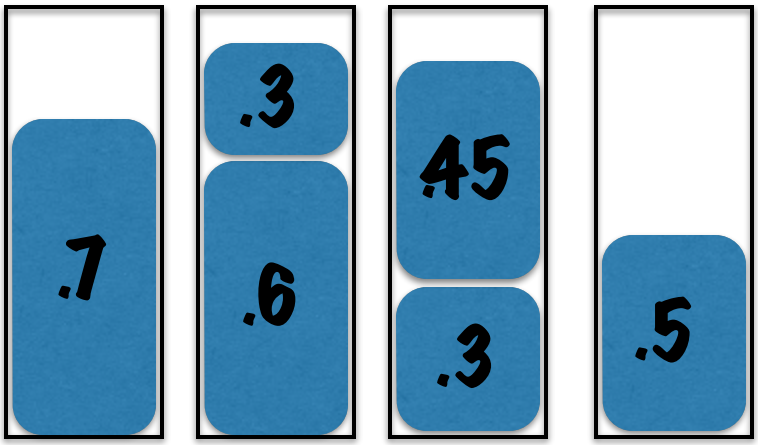
Runtime : polynomial

But how good is it?

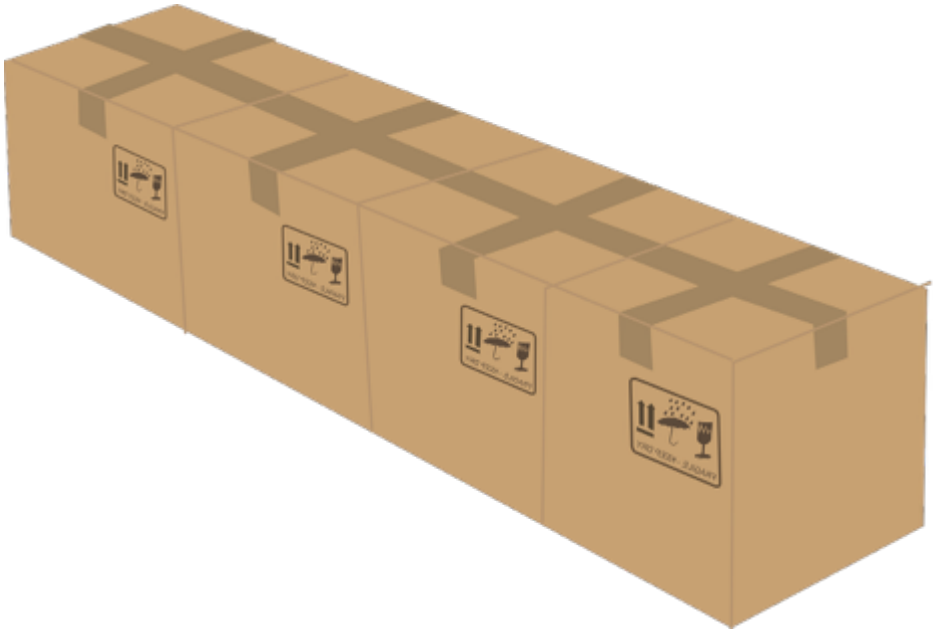
Value(Output) < OPT * $(1 + O(\epsilon))$

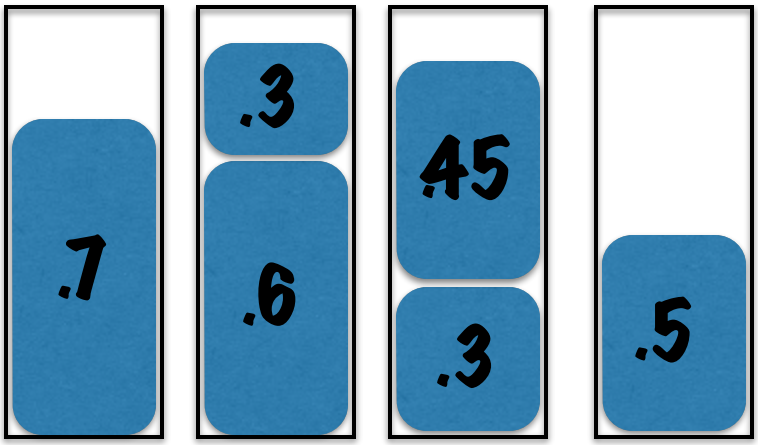
Bin packing, linear programming and rounding





Bin packing, linear programming and rounding





Algorithm - large items

Assume: sizes > capacity * ϵ

Sort sizes

Make groups of cardinality $n \times \epsilon^2$

Round up to max size in group

Solve rounded problem U

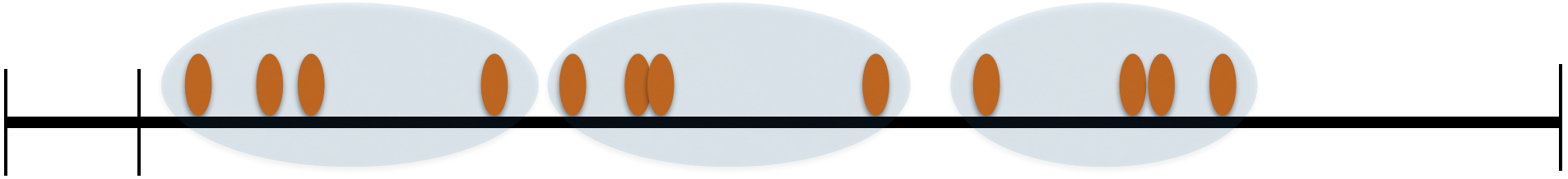
Output corresponding packing

But how good is it?

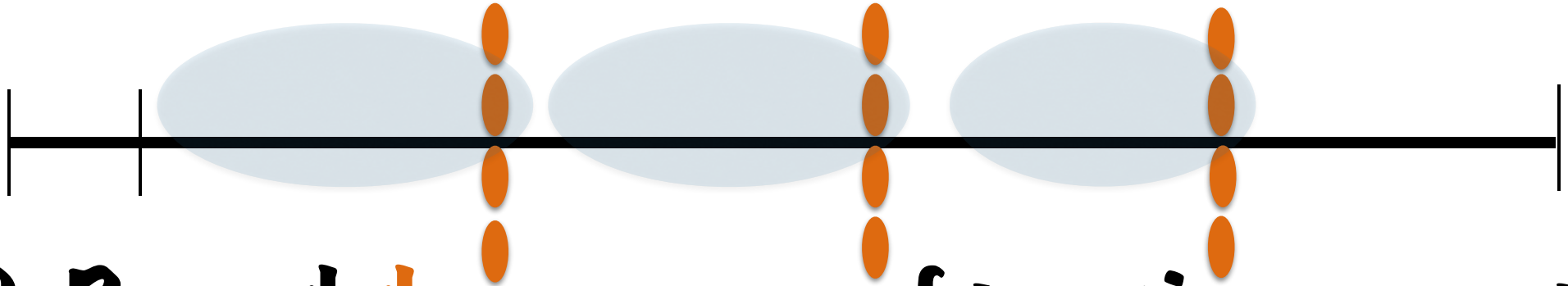
Value(Output) = OPT(U)

Must relate OPT(U) to OPT(I)

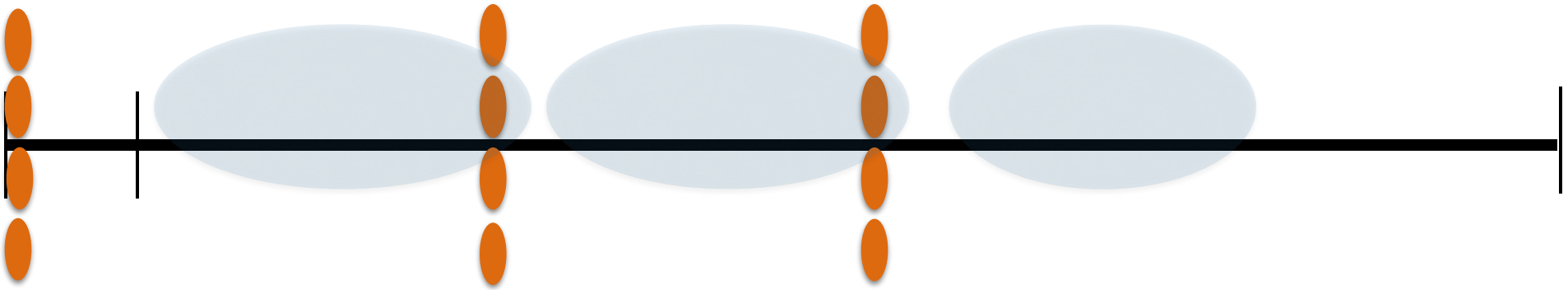
I: Input

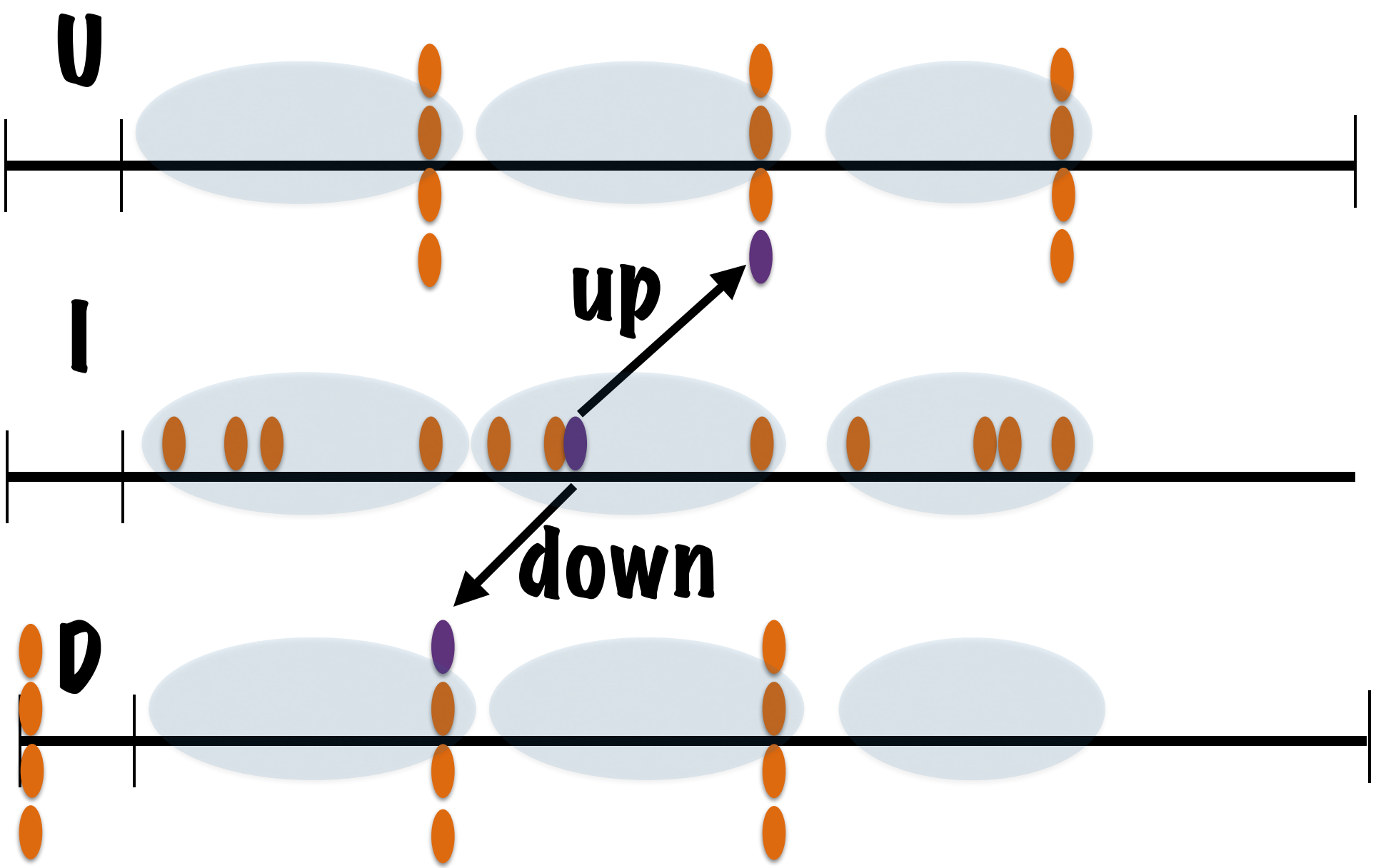


U: Round up: max of group



D: Round down: max of previous group



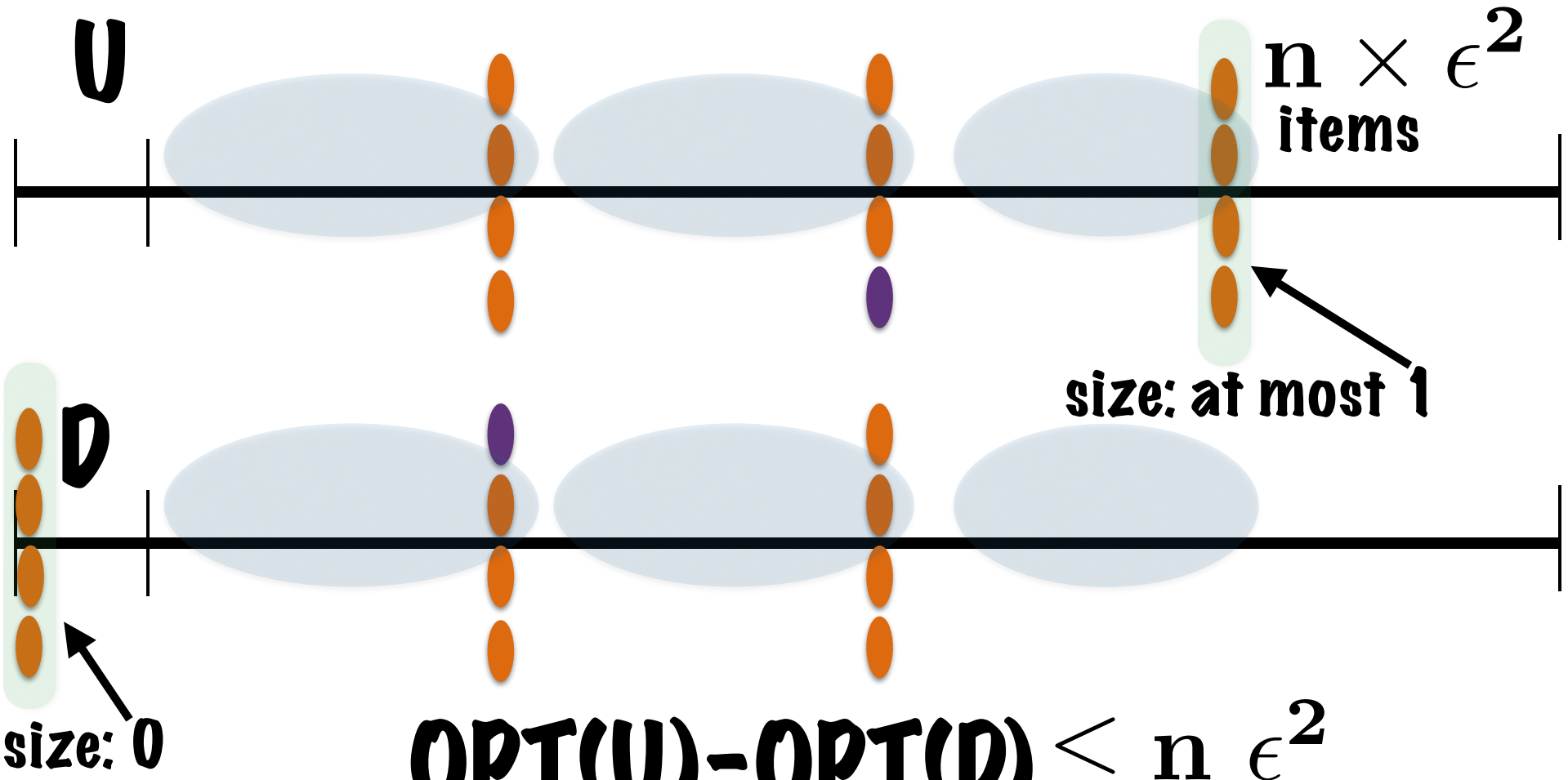

U
I
up
down
D

Relating input to rounded input

Observe:
Increasing sizes
can only increase OPT

$$OPT(D) \leq OPT(I) \leq OPT(U)$$

U and D are similar!



$$OPT(U) - OPT(D) \leq n\ \epsilon^2$$

Combine:

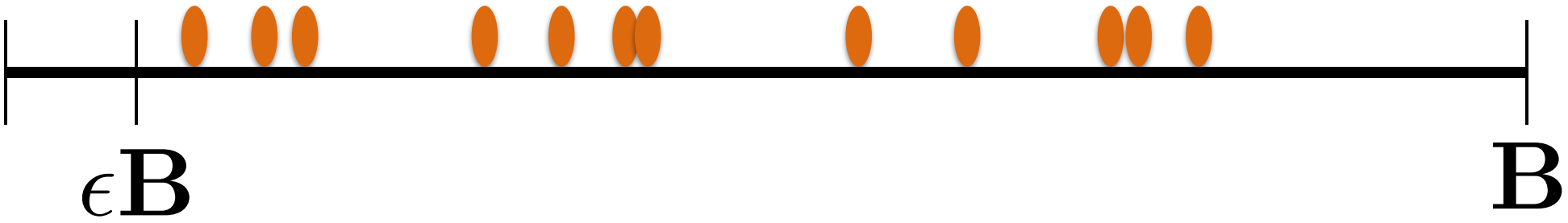
$$OPT(D) \leq OPT(I) \leq OPT(U)$$

$$OPT(U) - OPT(D) \leq n\,\epsilon^2$$

$$OPT(U) \leq OPT(I) + n\,\epsilon^2$$

Additive error $n\,\epsilon^2$

Lower bound OPT



n items

max #items per bin: $1/\epsilon$

---

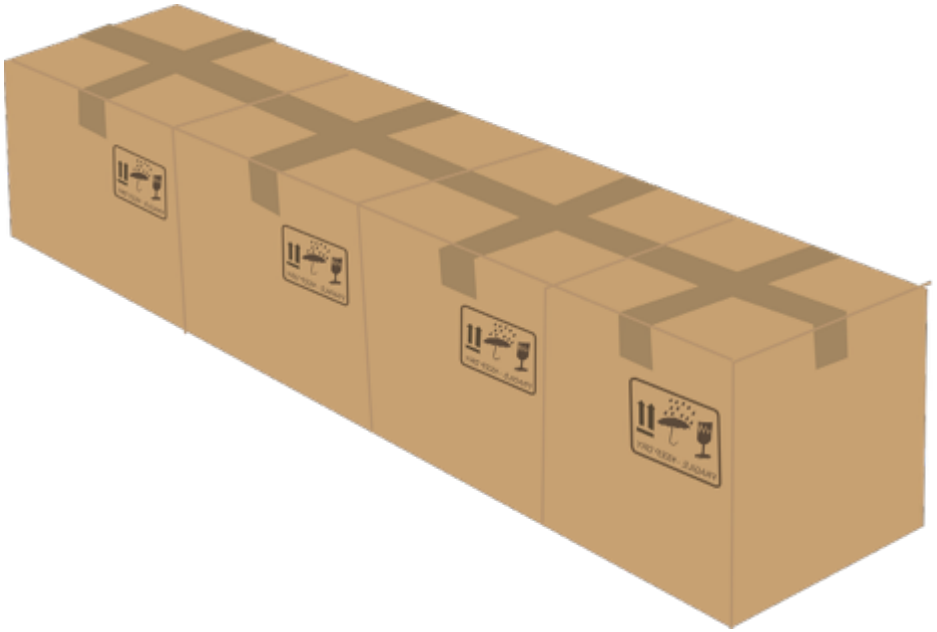
min #bins: ϵn

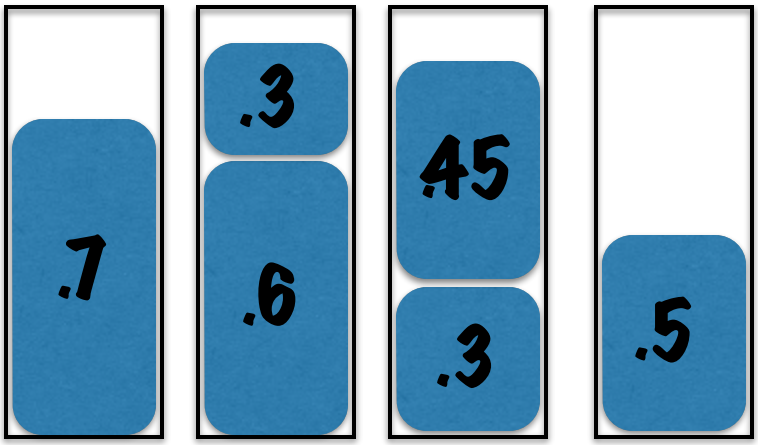
$$n\epsilon^2 \leq \epsilon \times (n\epsilon) \leq \epsilon \, \mathrm{OPT}$$

Theorem

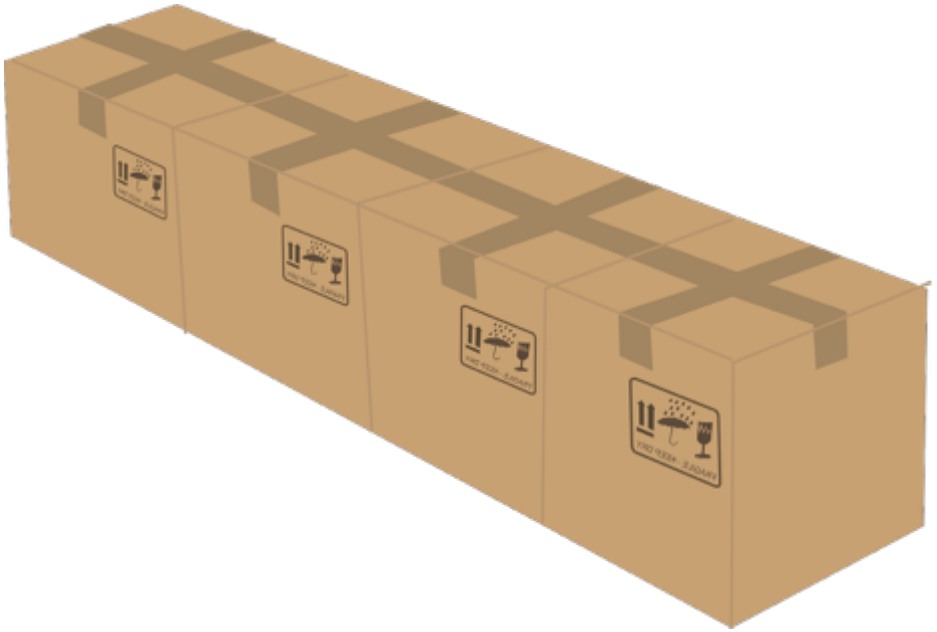
When all sizes are > ϵB
algorithm, in polynomial time
gives packing s.t.
Value(Output) < OPT $*(1+\epsilon)$

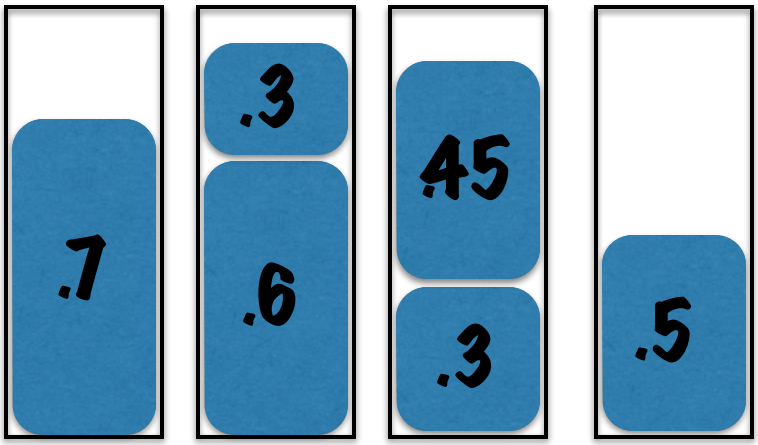
Bin packing, linear programming and rounding





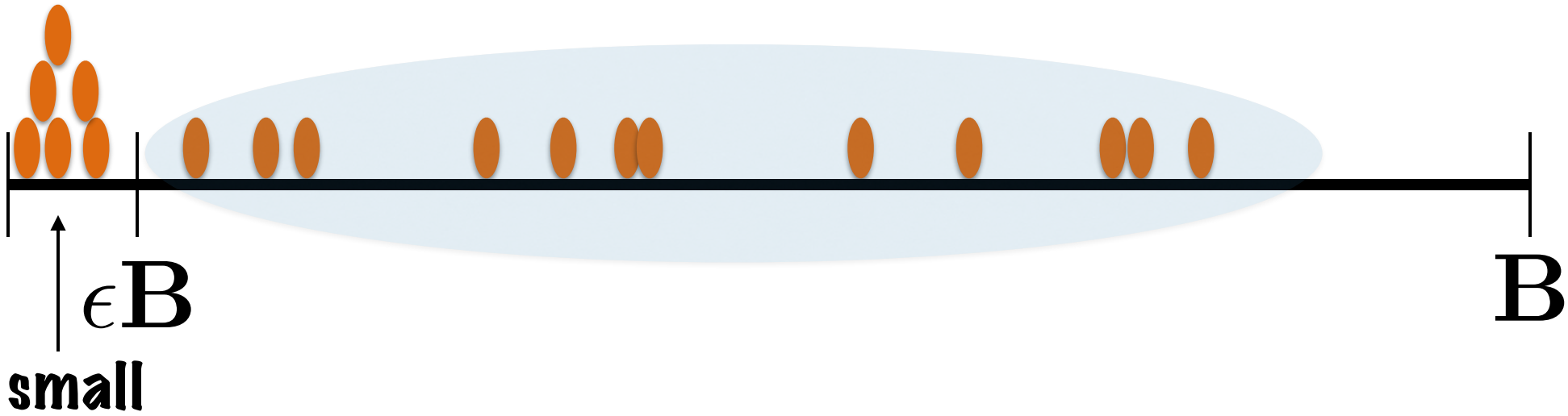
Bin packing, linear programming and rounding



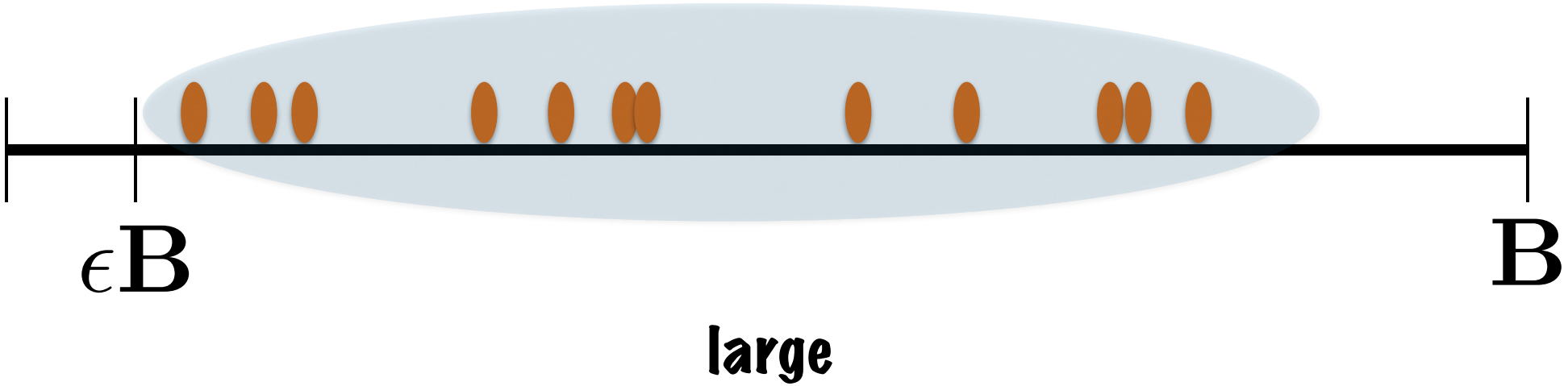


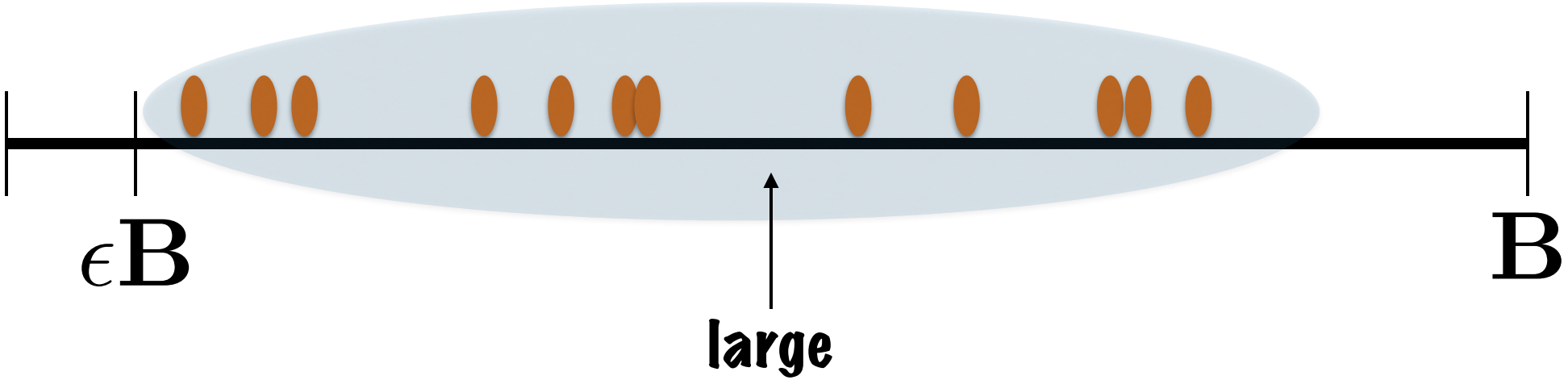
General algorithm

Set aside: sizes < cap. * ϵ (small)
Sort remaining sizes
Make groups of cardinality $n \times \epsilon^2$
Round up to max size in group
Solve rounded problem U
Greedily add small sizes

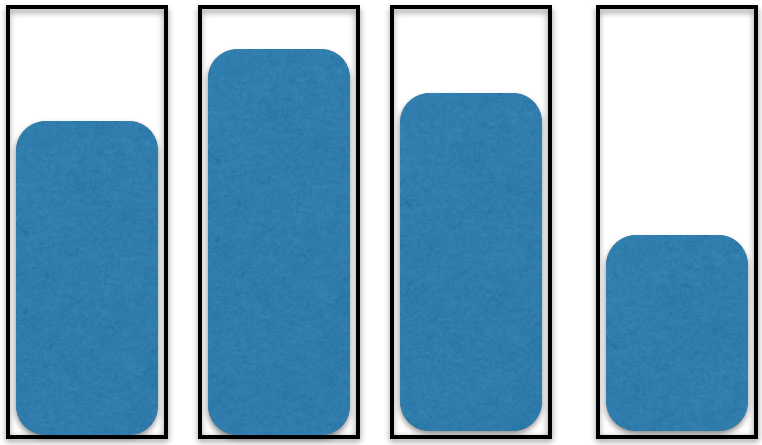


Set aside: sizes < $B\epsilon$ (small)

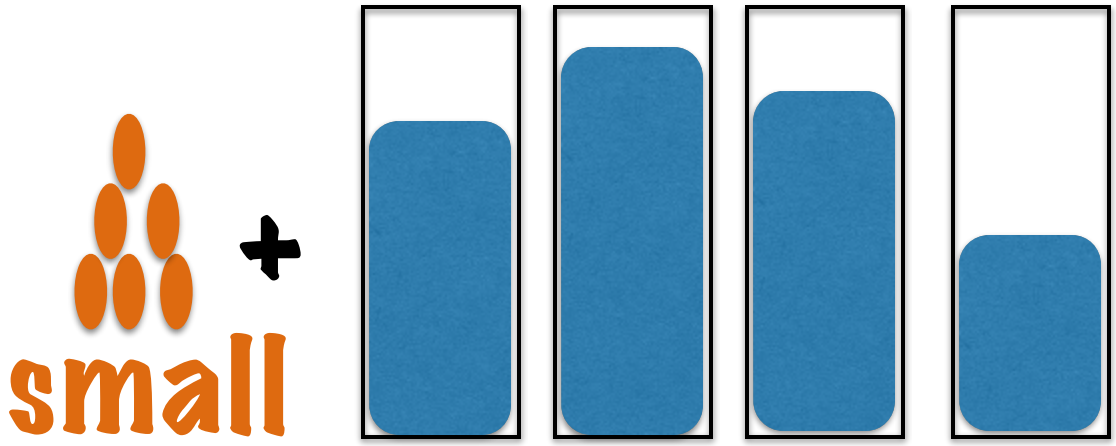




Solve for remaining sizes

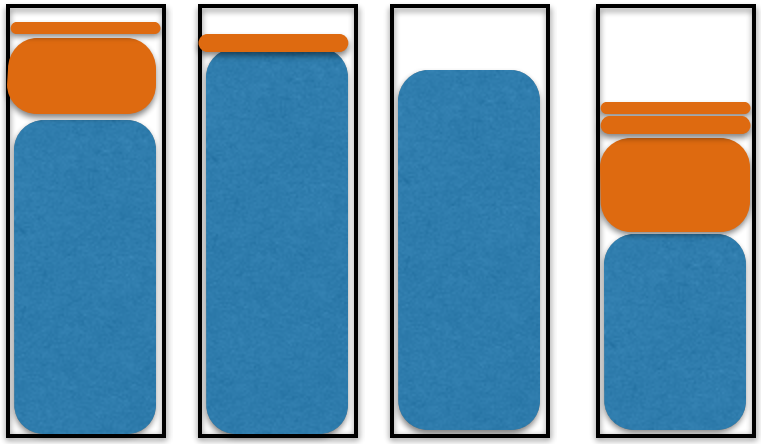


Packing of large items



Packing of large items

Greedily add small sizes



Analysis

$$\textbf{Input } \mathbf{I} = \mathbf{S} \cup \mathbf{L}$$

Case 1

No new bins opened by S: then

$$\begin{aligned}
&\mathbf{Value(Output)} = \\
&\mathbf{Value(packing\ of}\ L\mathbf{)} \\
&\leq (1+\epsilon) \cdot \mathbf{OPT(L)} \\
&\leq (1+\epsilon) \cdot \mathbf{OPT(I)}
\end{aligned}$$

Case 2

Some new bin opened by S: then all bins except last are filled to B times

$$\geq 1 - \epsilon$$

$$(1/\mathrm{B}) \sum \mathrm{s_i} \geq (\#\mathrm{bins} - 1)(1 - \epsilon)$$

$$(1/\mathrm{B}) \sum \mathrm{s_i} \leq \mathrm{OPT}$$

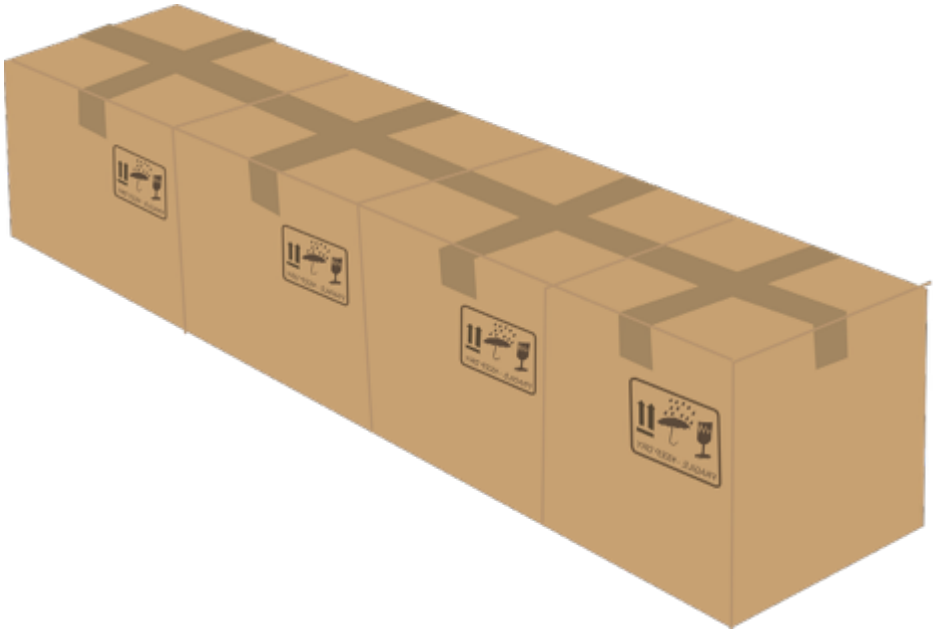
$$\mathrm{Value(Output)} \leq \frac{1}{1-\epsilon} \mathrm{OPT} + 1$$

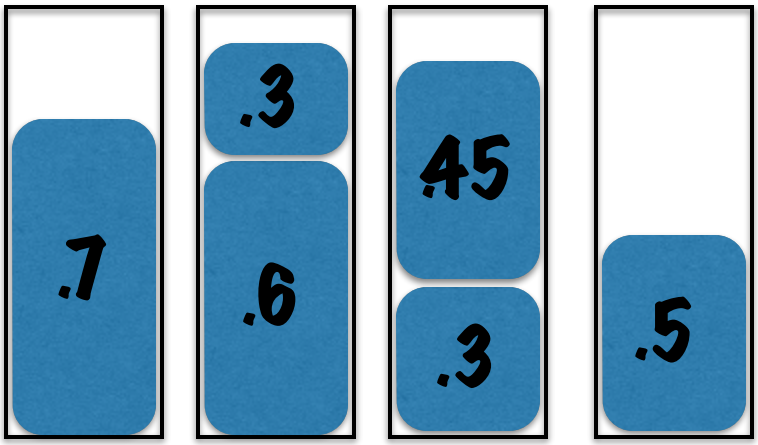
Theorem

Algorithm, in polynomial time gives packing s.t. Value(Output) <

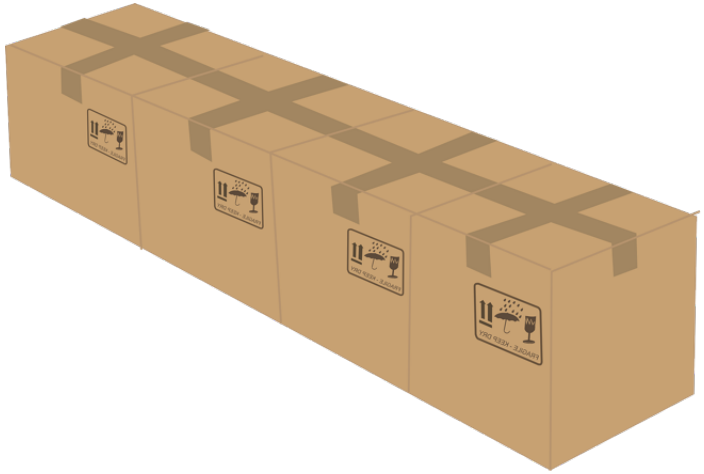
$$\mathbf{OPT}(1 + \mathbf{O}(\epsilon)) + 1$$

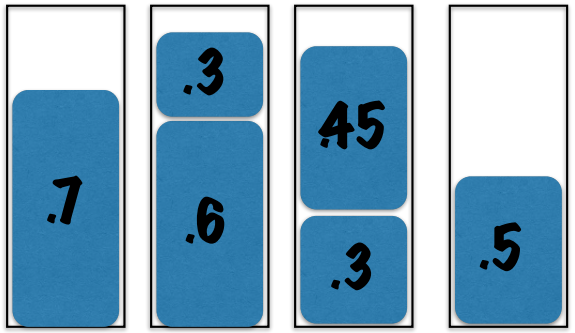
Bin packing, linear programming and rounding





Bin packing, linear programming and rounding





Adaptive rounding of input

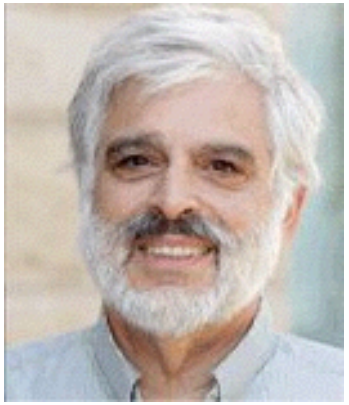
Bin packing variants

Two-dimensional,
three-dimensional

Online

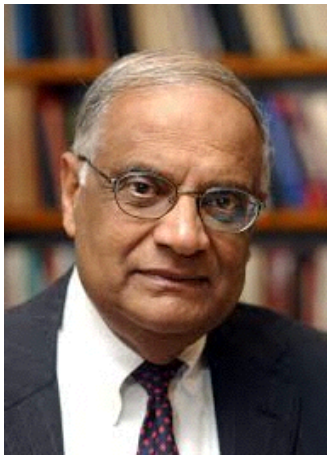
Karp (1972)
NP-complete

David S. Johnson
Constant factor
approximation algorithms
Jeffrey D. Ullman



The pioneers

Richard Karp



Narendra Karmarkar

Average case analysis

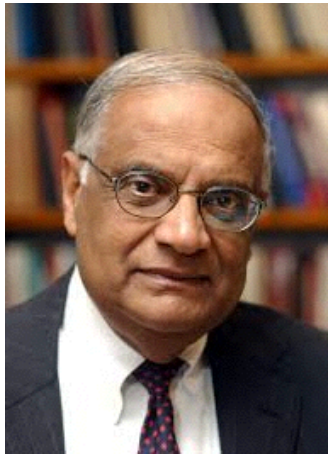
David S. Johnson

L. McGeoch ...

Wenceslas Fernandez de la Vega

George S. Lueker

$$\mathbf{OPT}(1+\mathbf{O}(\epsilon))+1$$



Narendra Karmarkar

Richard Karp

$$\mathbf{OPT}+\mathbf{O}(\log^2 \mathbf{OPT})$$

Asymptotic approximation schemes

Rebecca Hoberg
Thomas Rothvoss

$\mathrm{OPT} + \mathrm{O}(\log \mathrm{OPT})$

Asymptotic approximation schemes

Bin packing, linear programming and rounding