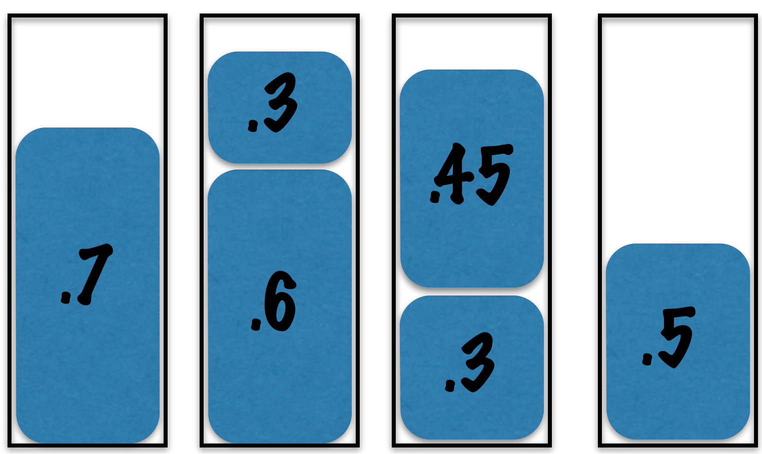
The Next Fit algorithm

"Next Fit" algorithm

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

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



can this instance
be packed better?

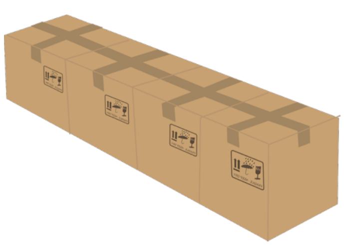
How good is Next Fit?

Capacity 100. Items:

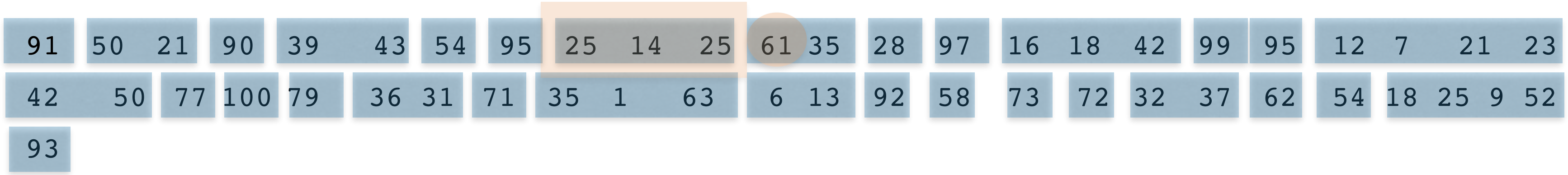
91 50 21 90 39 43 54 95 25 14 25 61 35
28 97 16 18 42 99 95 12 7 21 23 42 50 77
100 79 36 31 71 35 1 63 6 13 92 58 73 72
32 37 62 54 18 25 9 52 93

Next Fit:

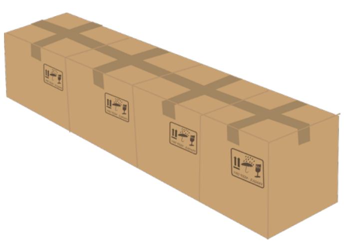
91	50 21	90	39 43	54	95	25 14 25	61 35	28	97	16 18 42	99	95	12 7 21 23		
42 50	77	100	79	36 31	71	35 1 63	6 13	92	58	73	72	32 37	62	54	18 25 9 52
93															

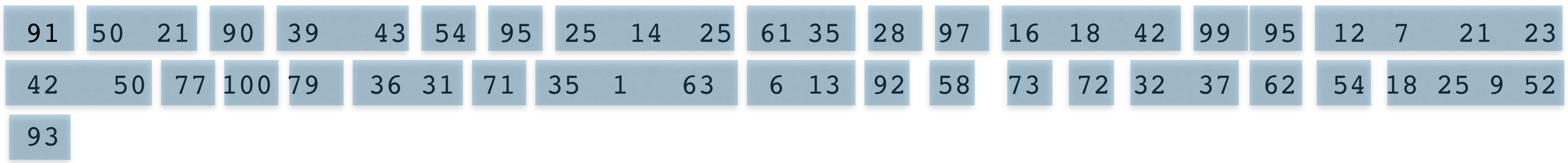


used 31 bins

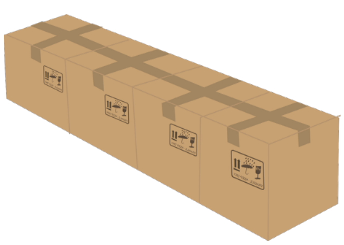


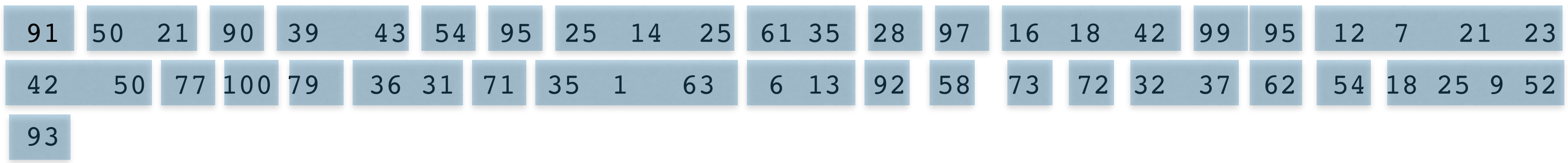
bin 7: (25+14+25)
next item: 61, but
(25+14+25) +61>100
so, close bin 7, open bin 8,
put item 61 in bin 8.



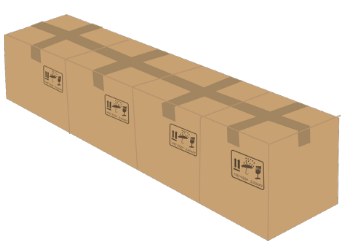


In general:
(items in bin 2i-1)+next item > 100
(items in bins 2i-1 or 2i) >100
31 bins: total item sizes > 15*100





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



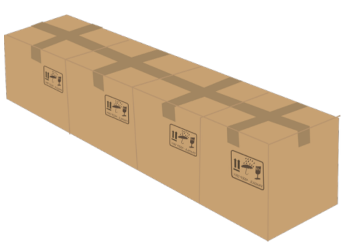
91 50 21 90 39 43 54 95 25 14 25 61 35 28 97 16 18 42 99 95 12 7 21 23

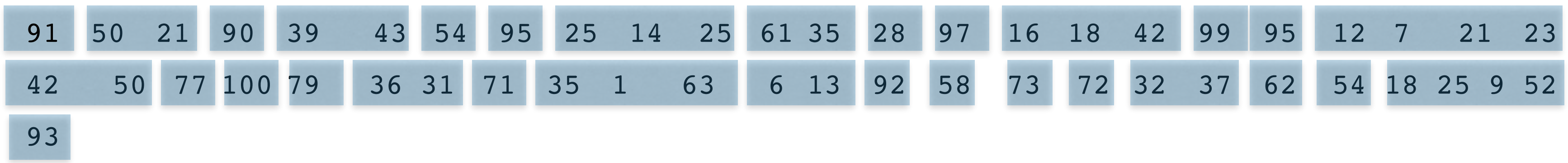
42 50 77 100 79 36 31 71 35 1 63 6 13 92 58 73 72 32 37 62 54 18 25 9 52

93

What about OPT?

Total item sizes < OPT*100



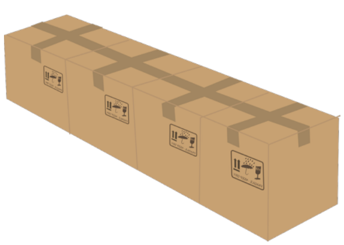


Combining:

k bins by Next Fit

$OPT*100 > (k-1)/2 * 100$

$\#(\text{bins of Next Fit}) < 2*OPT + 1$



Asymptotic 2 approximation

Special special case

Large items,
few distinct sizes

Example

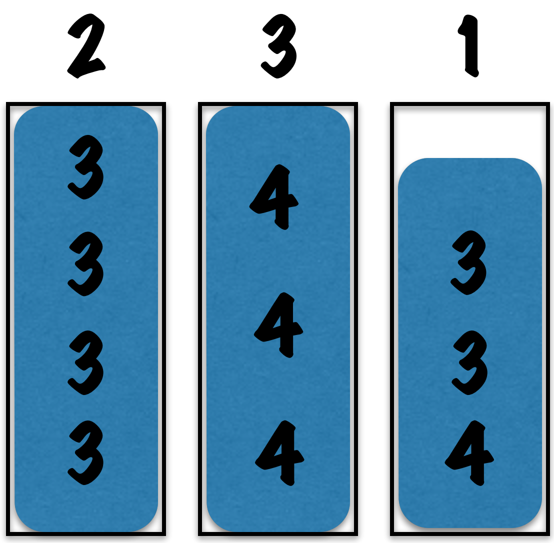
Bin capacity 12

sizes: { 3, 4 }

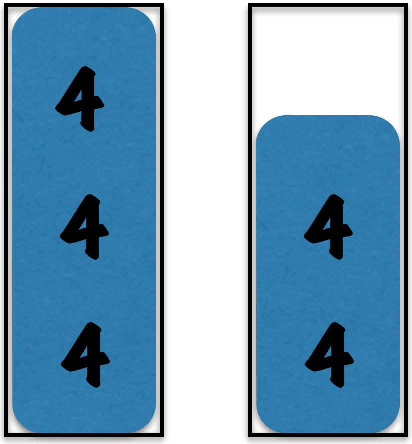
10 items of size 3,

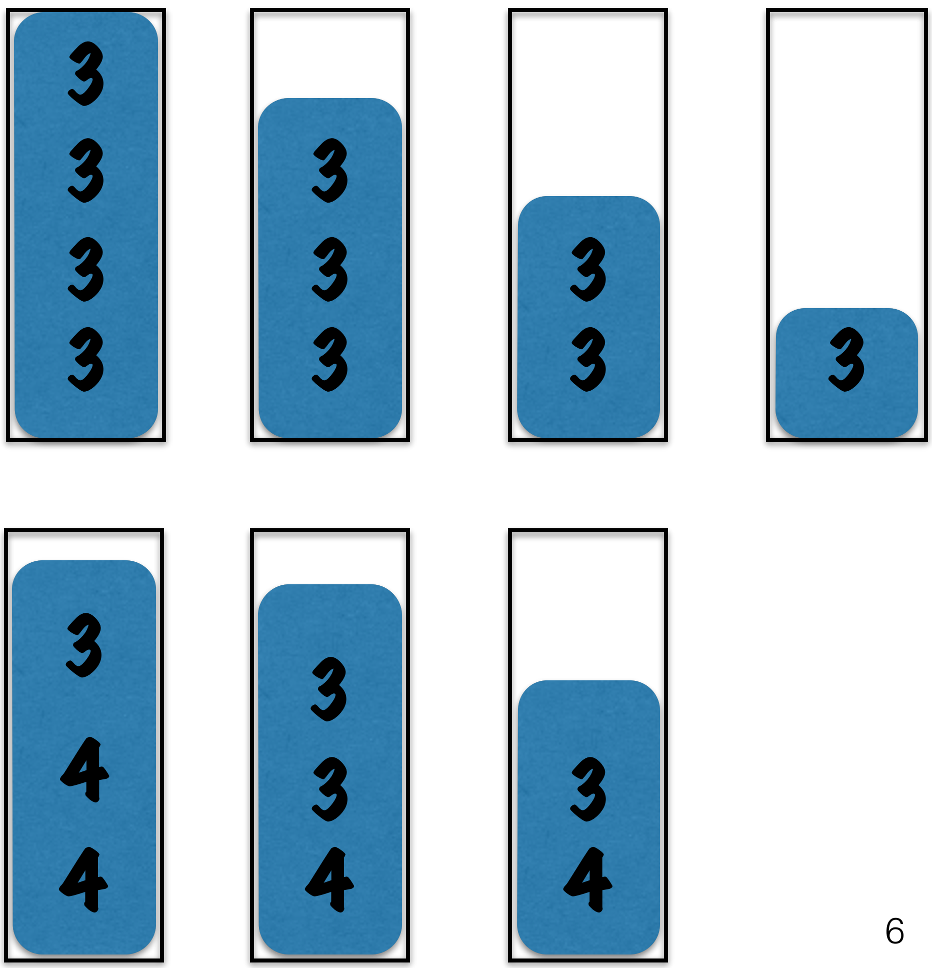
10 items of size 4.

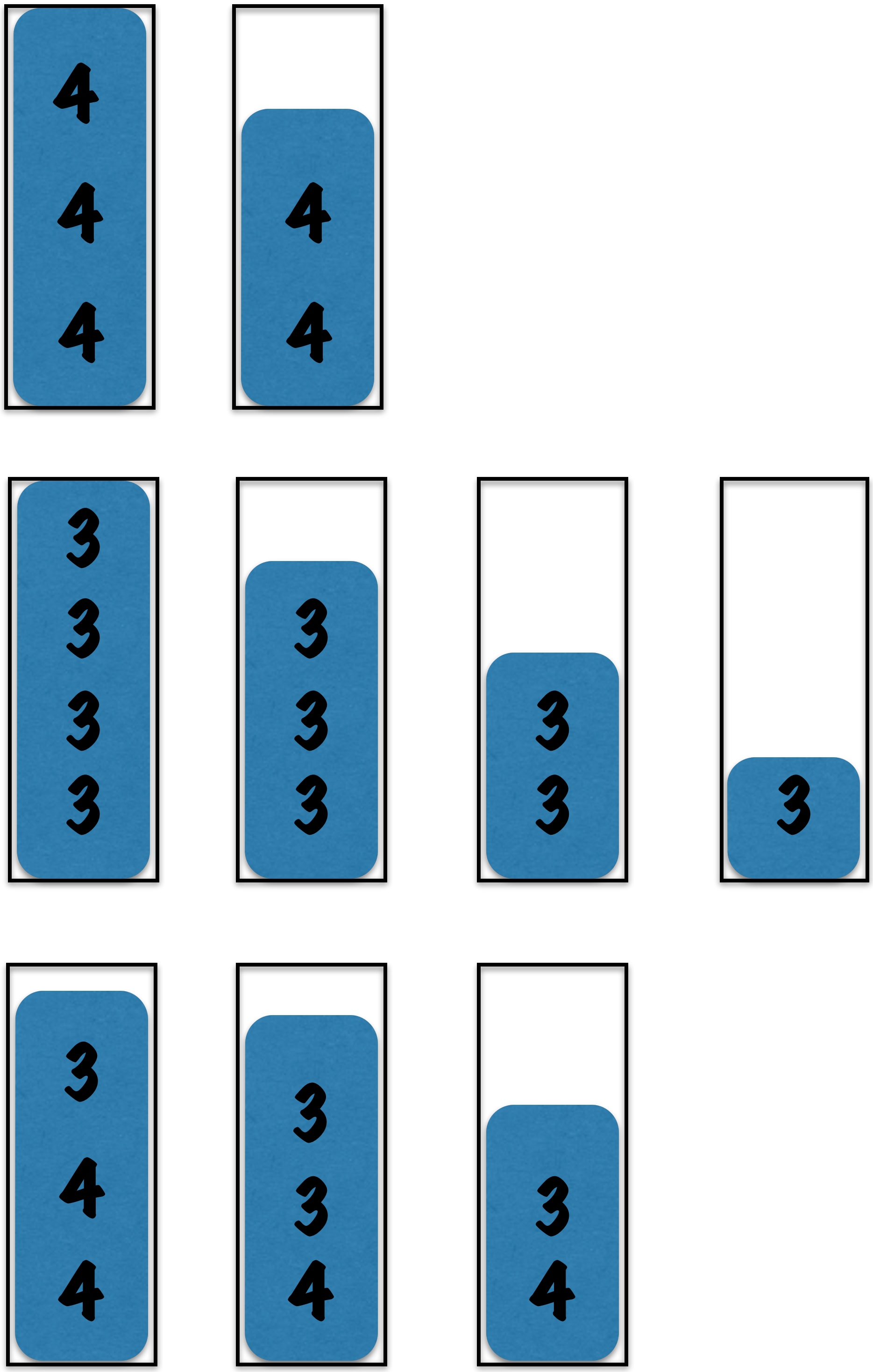
Bin capacity 12
sizes: { 3, 4 }
10 items of size 3,
10 items of size 4.



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: $\sum_c x_c$

integer

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

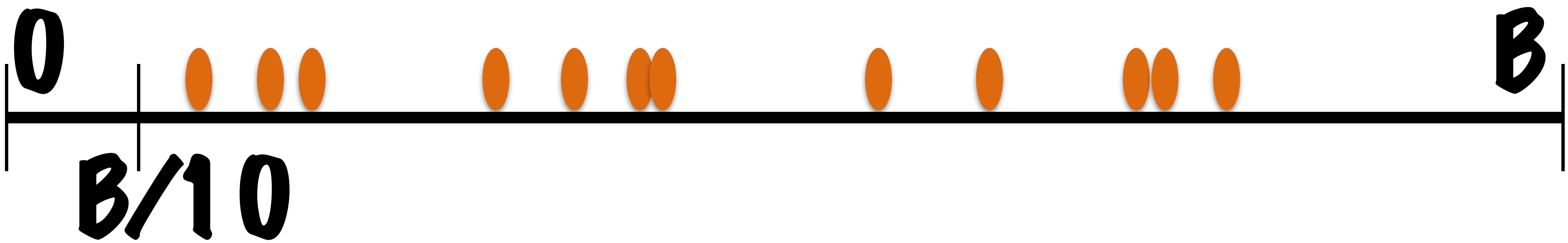
Check whether, for every size s, enough slots for items of size s

Output solution with min #bins

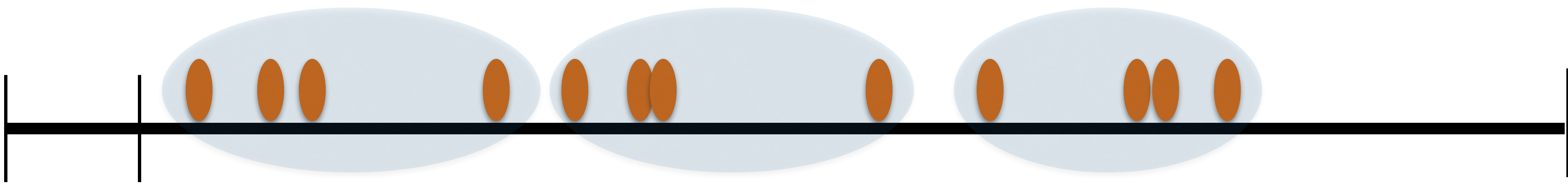
Runtime if size>capacity/10:

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

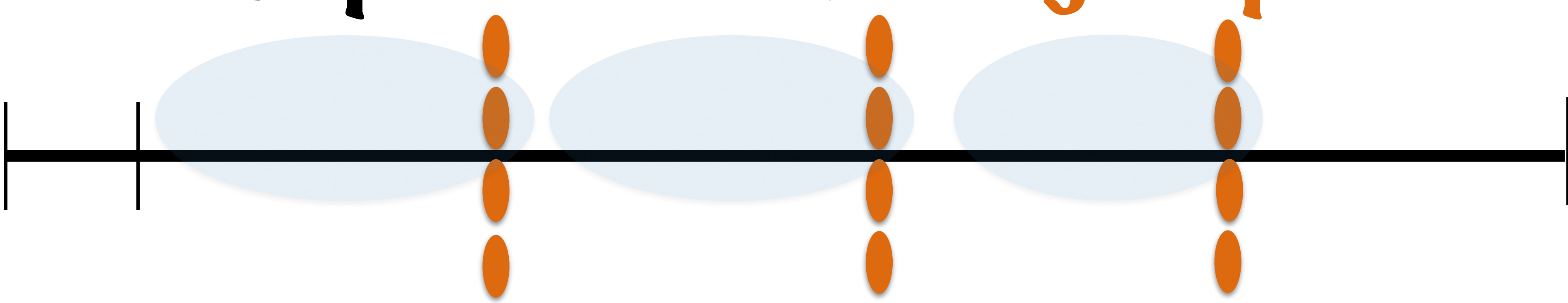
Adaptive rounding



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

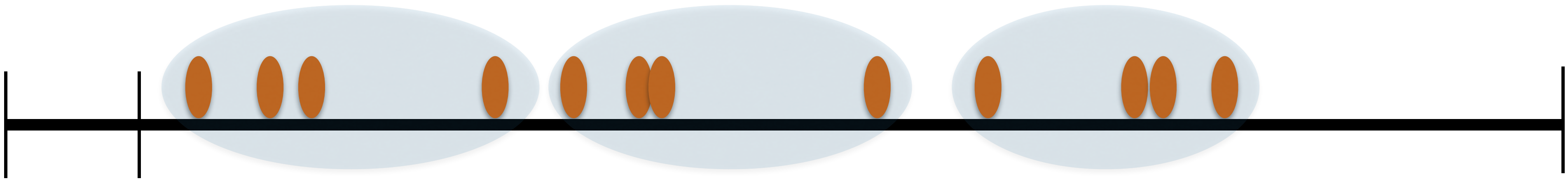
Observe: output is a packing

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

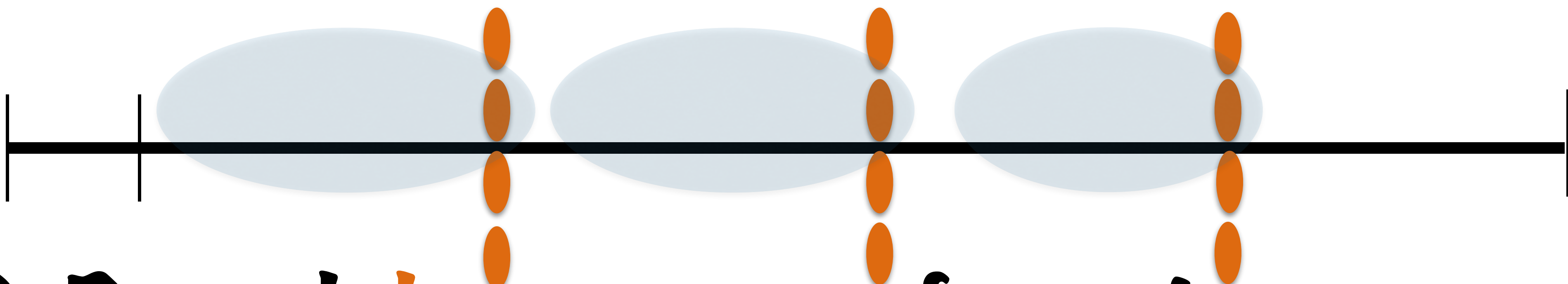
Observe: #distinct sizes < $1/\epsilon^2$

Runtime : polynomial

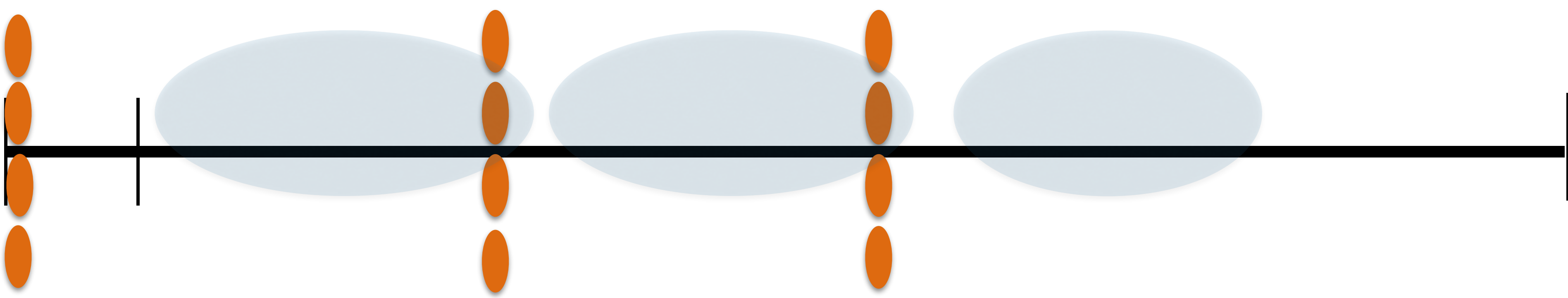
I: Input

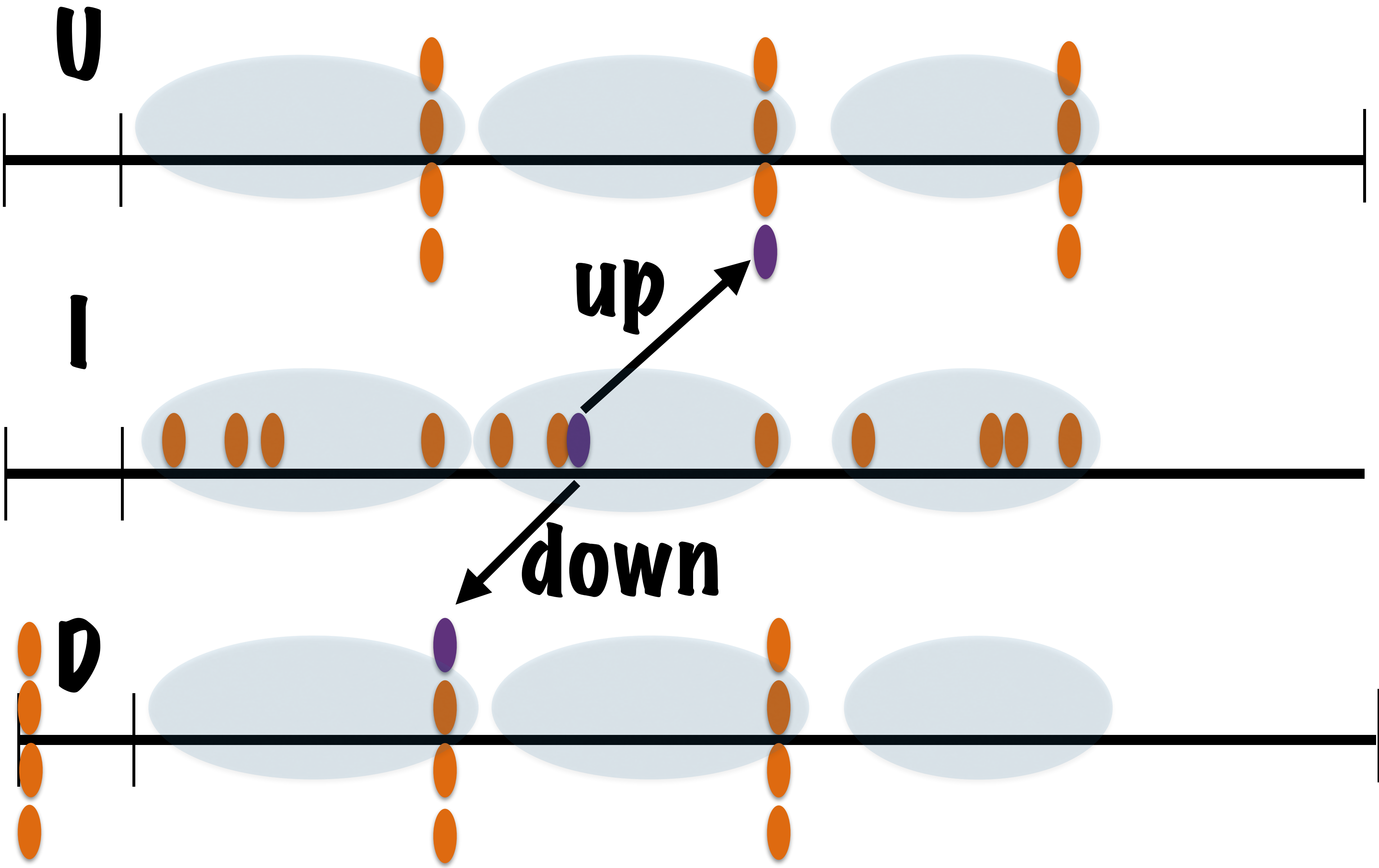


U: Round up: max of group



D: Round down: max of previous group



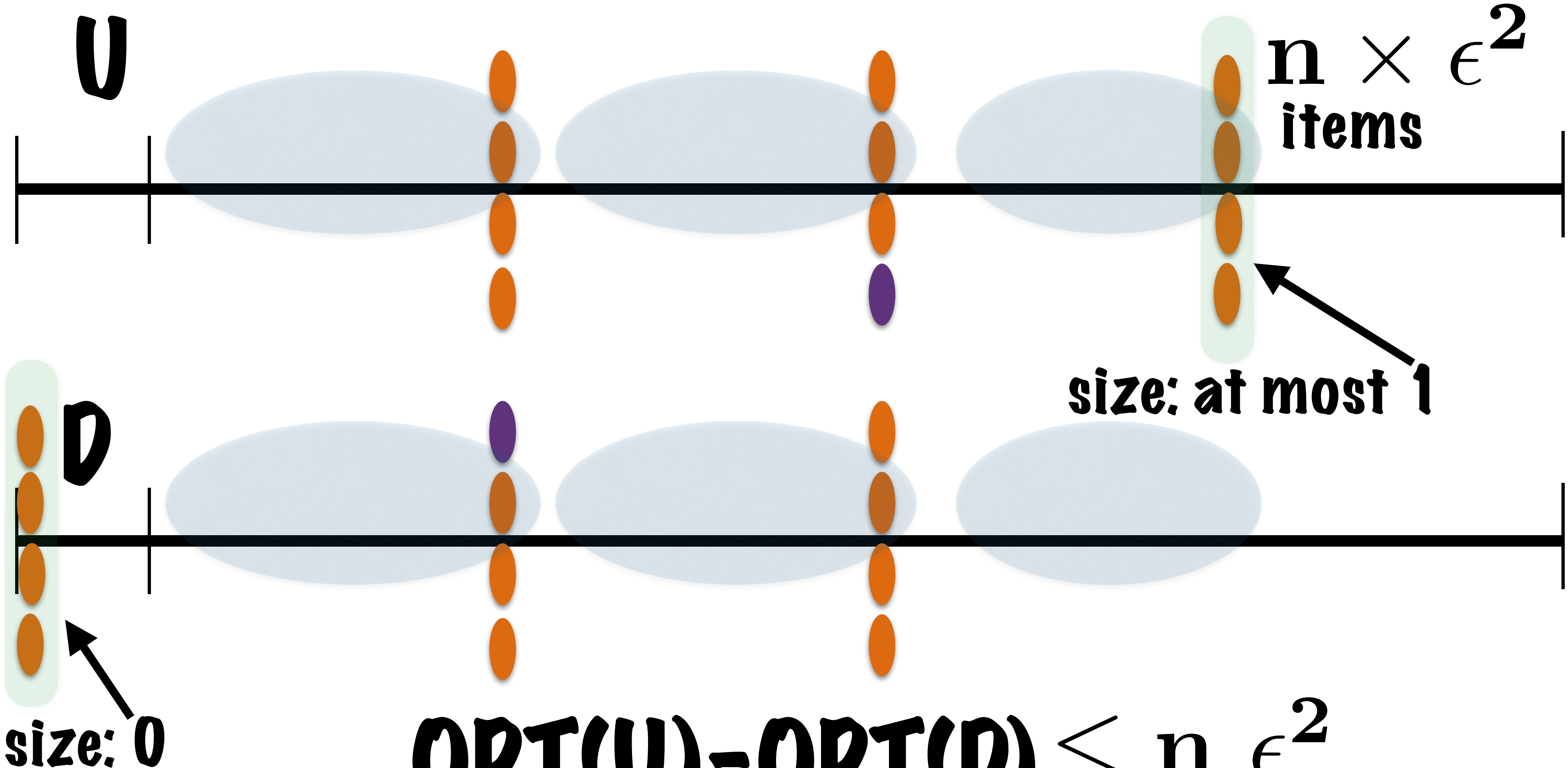

U
I
D
up
down

Relating input to rounded input

Observe:
Increasing sizes
can only increase OPT

$$\mathrm{OPT}(D) \leq \mathrm{OPT}(I) \leq \mathrm{OPT}(U)$$

U and D are similar!



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

Combine:

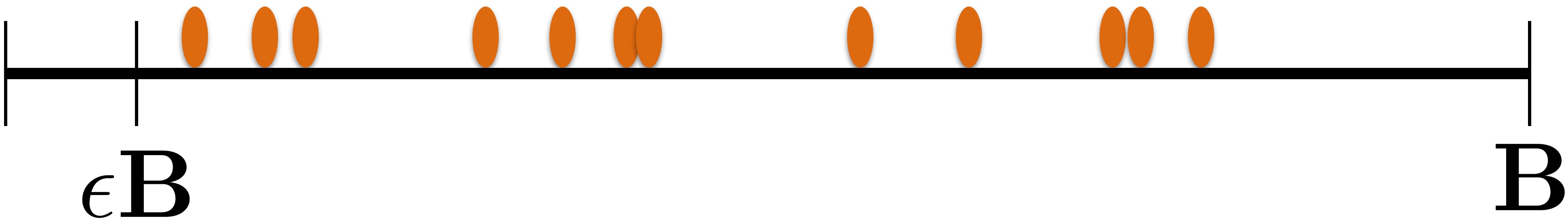
$$OPT(D) \leq OPT(I) \leq OPT(U)$$

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

$$OPT(U) \leq OPT(I) + \mathbf{n}\ \epsilon^{\mathbf{2}}$$

Additive error $\mathbf{n}\ \epsilon^{\mathbf{2}}$

Lower bound OPT



$$\frac{\text{n items}}{\text{max \#items per bin: } 1/\epsilon}$$

min #bins: ϵn

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

Theorem

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

General algorithm

Set aside: sizes < cap. * ϵ (small)

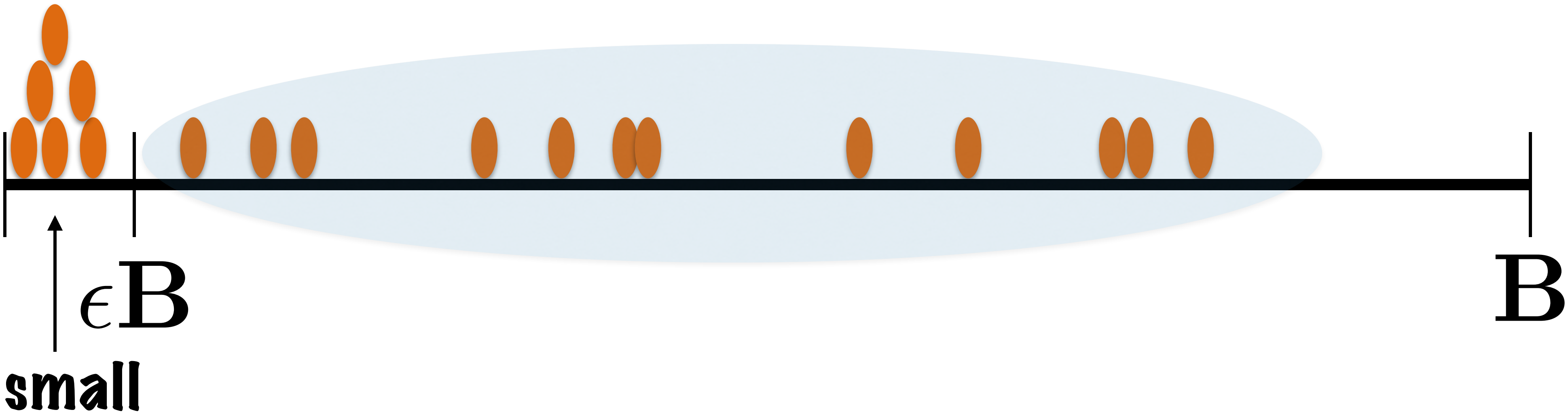
Sort remaining sizes

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

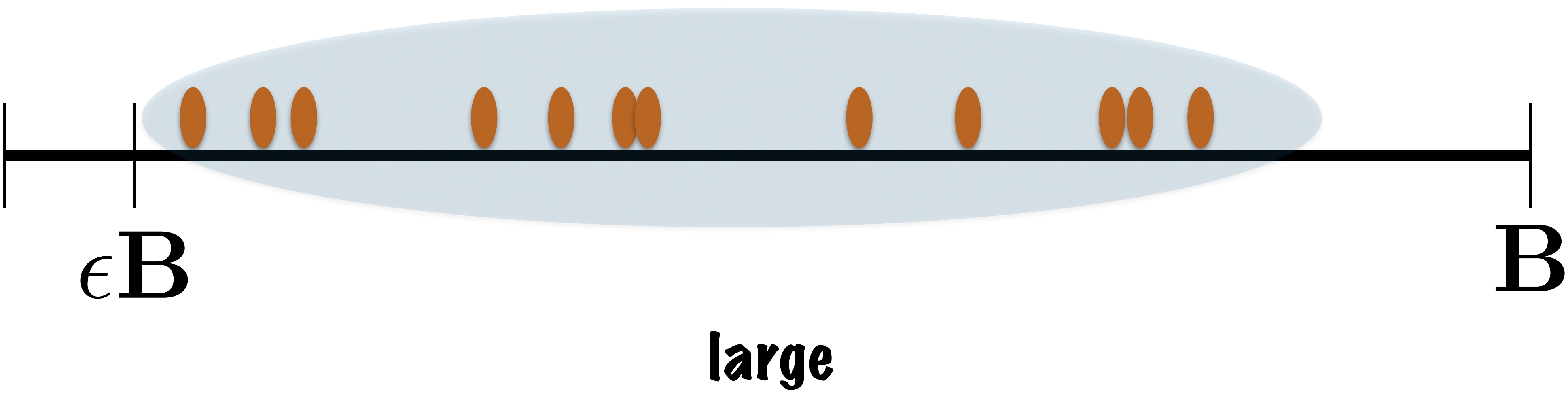
Round up to max size in group

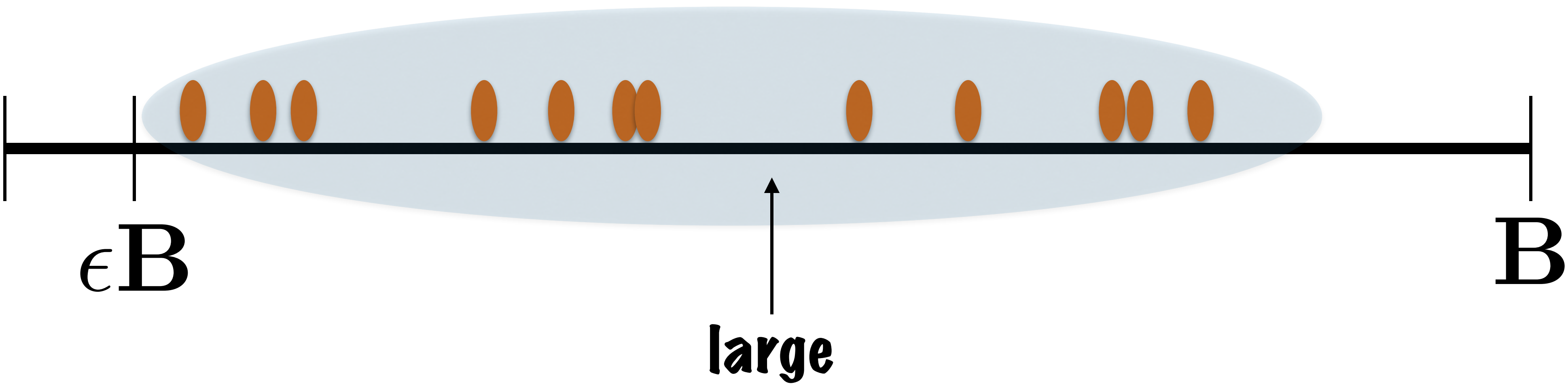
Solve rounded problem U

Greedily add small sizes

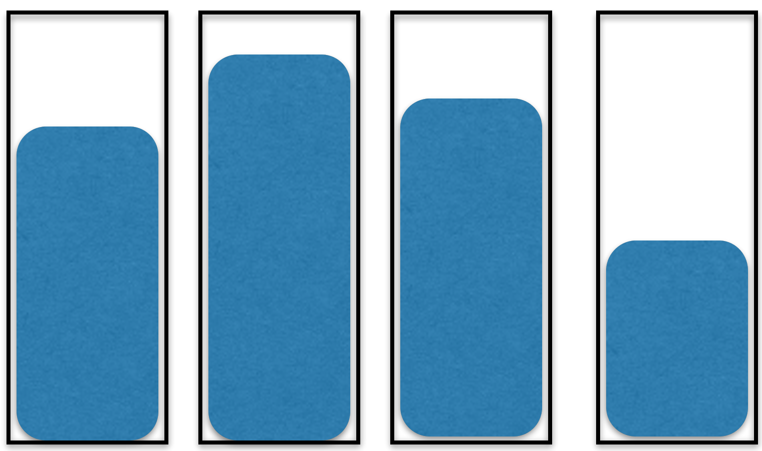


Set aside: sizes $< B \epsilon$ (small)

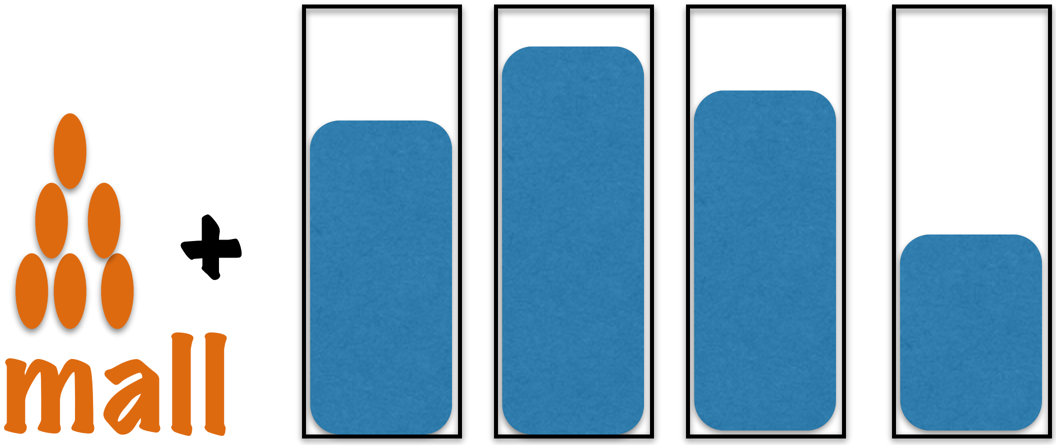




Solve for remaining sizes



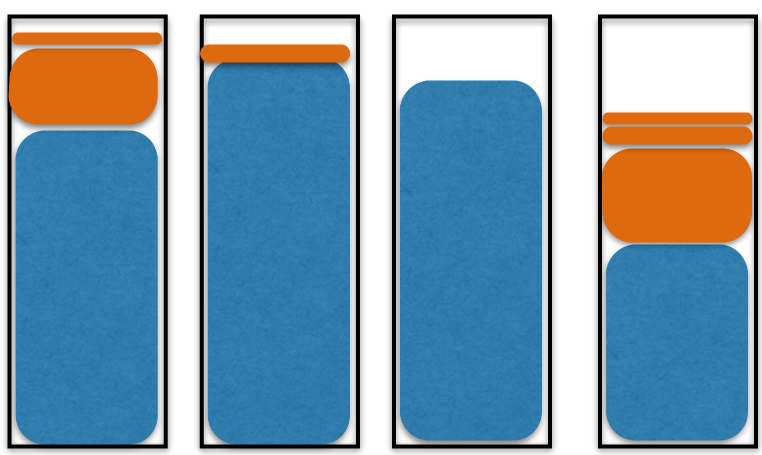
Packing of large items



small 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

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

Case 2

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

$$\geq 1 - \epsilon$$

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

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

Theorem

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

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