Approximation Algorithms
Chapter 9: Bin Packing

Presented By:

Piyush Ranjan Satapathy

Class CS260 By Dr Neal Young

(Original Slides From Nobuhisa Ueda's Webpage)

Overview (1/4)

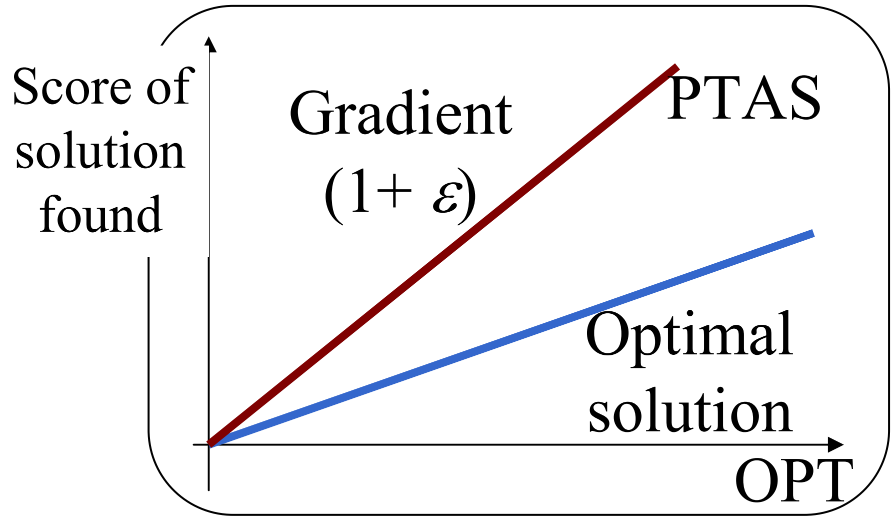
- Main issue: Asymptotic approximation algorithms for NP-hard problems
  - [Ideal case]: Given an instance, we can always obtain its solution with any approximation ratio.
    - PTAS (Polynomial Time Approximation Scheme)
      - See section 8 of the textbook.
  - [*Better* case]: For *almost* all instances, we can obtain its solution with any approximation ratio.
    - Bin Packing problem
    - Minimum approximation ratio = 3/2 if # bin is 2.

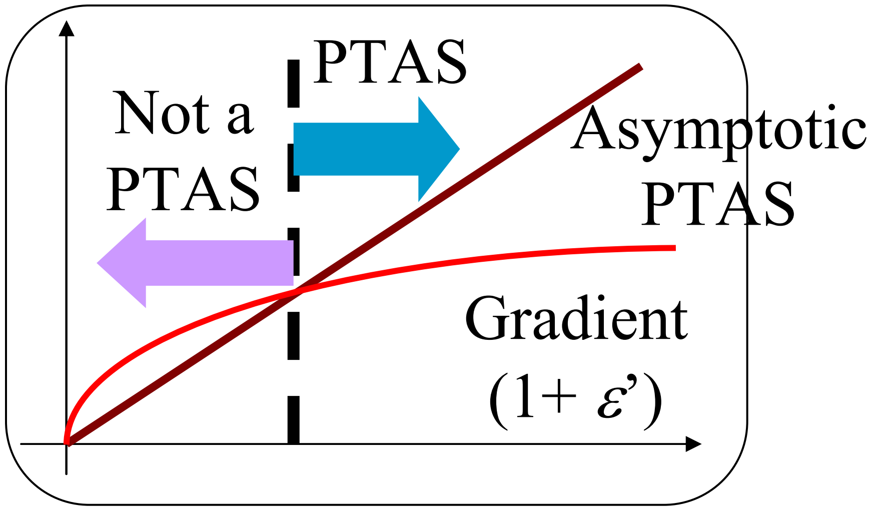
Overview(2/4)

- PTAS
  - Time bounded by a polynomial in (n), the problem size.
  - For any $\varepsilon > 0$ for a problem instance I the performance guarantee is $A(I) \leq (1+ \varepsilon)\, OPT(I)$
- FPTAS
  - Time bounded is polynomial in both problem size(n) and $(1/ \varepsilon)$.
  - We saw the Knapsack which is $O(n^2 \lfloor n / \varepsilon \rfloor)$
- FPTAAS
  - Time bounded is polynomial in both problem size, and $(1/ \varepsilon)$ and having a hidden constant in the order of (ε).
  - $A(I) \leq (1+ \varepsilon)\, OPT(I) + O_\varepsilon(1)$

Overview (3/4)

- PTAS
  - There is a polynomial-time algorithm that always finds a solution within a given approximation factor ε.
- Asymptotic PTAS
  - There is a polynomial-time algorithm for any large-sized instances that always finds a solution within a given approximation factor ε.



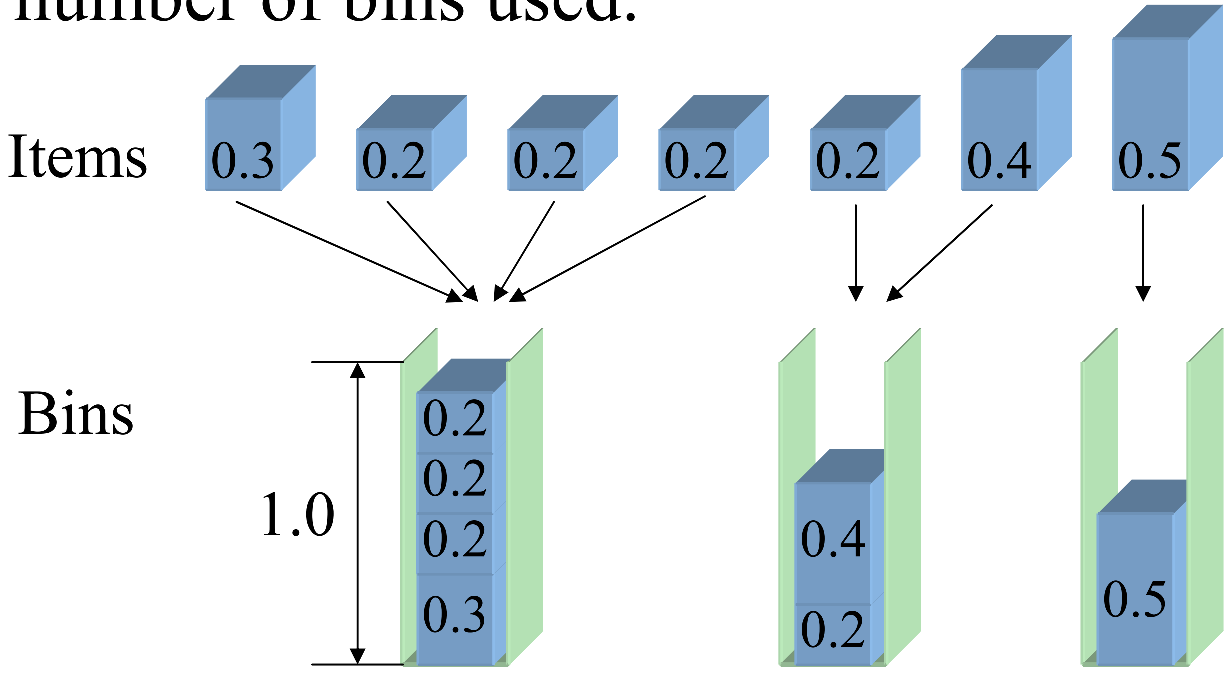


Overview (4/4)

- Bin packing problem
  - An example
  - The First-Fit algorithm.
    - Approximation factor is 2.
  - No approximation algorithm having a guarantee of 3/2.
    - Reduction from the set partition, an NP-complete problem.
  - Asymptotic PTAS A_ε.
    - The minimum size of bins=ε, # distinct sizes of bins= K.
    - Exact algorithm where ε and K are constants.
    - Approximation algorithm where ε is constant.

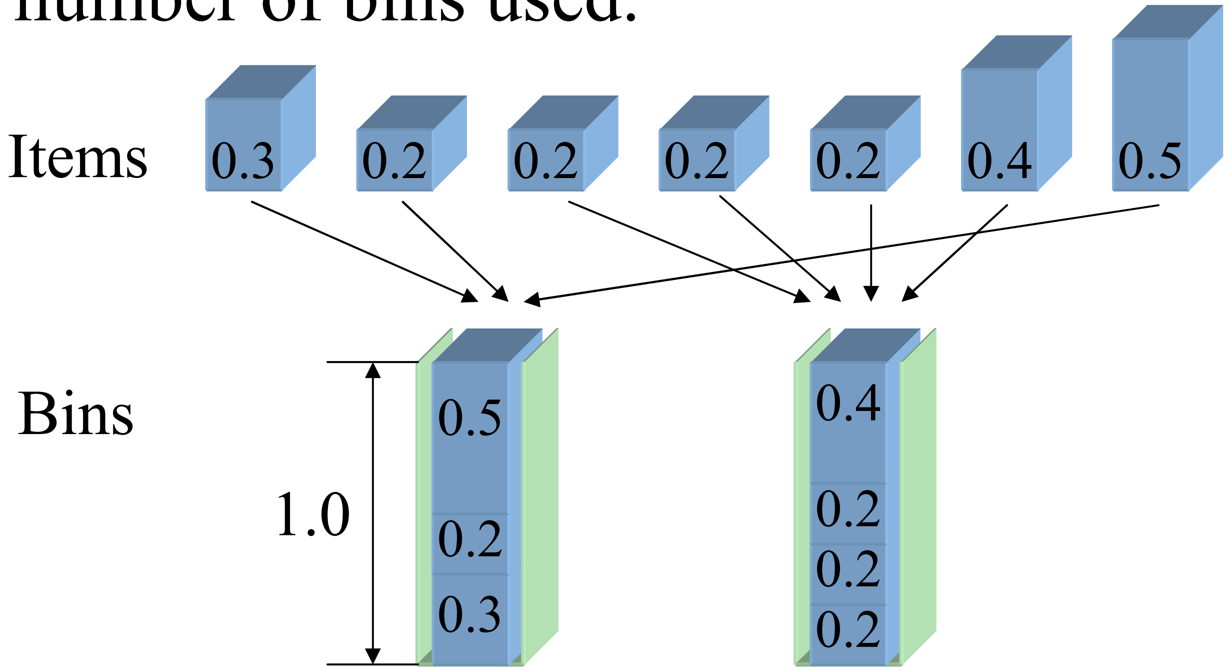
Bin packing problem

- Input:
  - n items with sizes $a_1, \ldots, a_n$ $(0 < a_i \leq 1)$.
- Task:
  - Find a packing in unit-sized bins that minimizes the number of bins used.



Bin packing problem

- Input:
  - n items with sizes $a_1, \ldots, a_n$ $(0 < a_i \leq 1)$.
- Task:
  - Find a packing in unit-sized bins that minimizes the number of bins used.

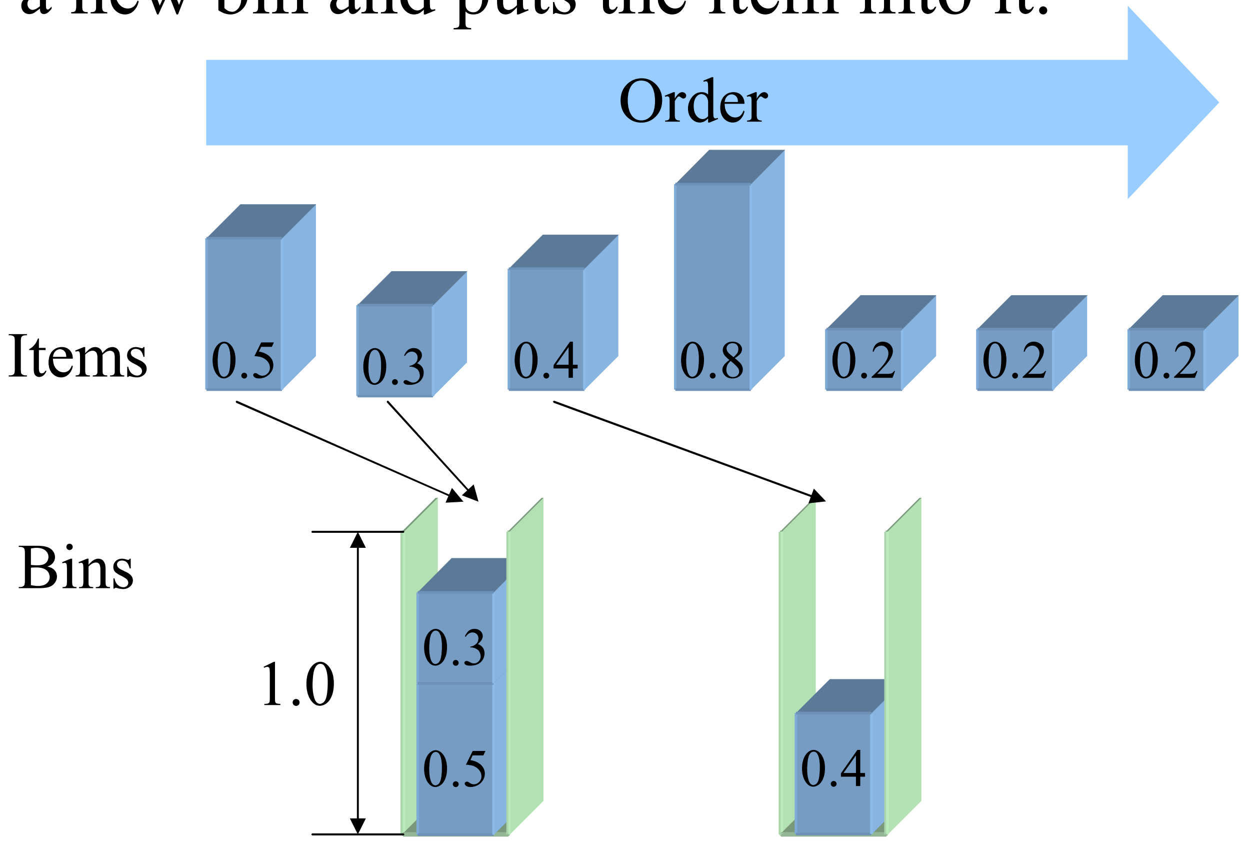


Overview (3/4)

- Bin packing problem
  - An example
  - The First-Fit algorithm.
    - Approximation factor is 2.
  - No approximation algorithm having a guarantee of 3/2.
    - Reduction from the set partition, an NP-complete problem.
  - Asymptotic PTAS A_{ε}.
    - Lower bound of bins: ε, # distinct sizes of bins: K.
    - Exact algorithm where ε and K are constants.
    - Approximation algorithm where ε is constant.

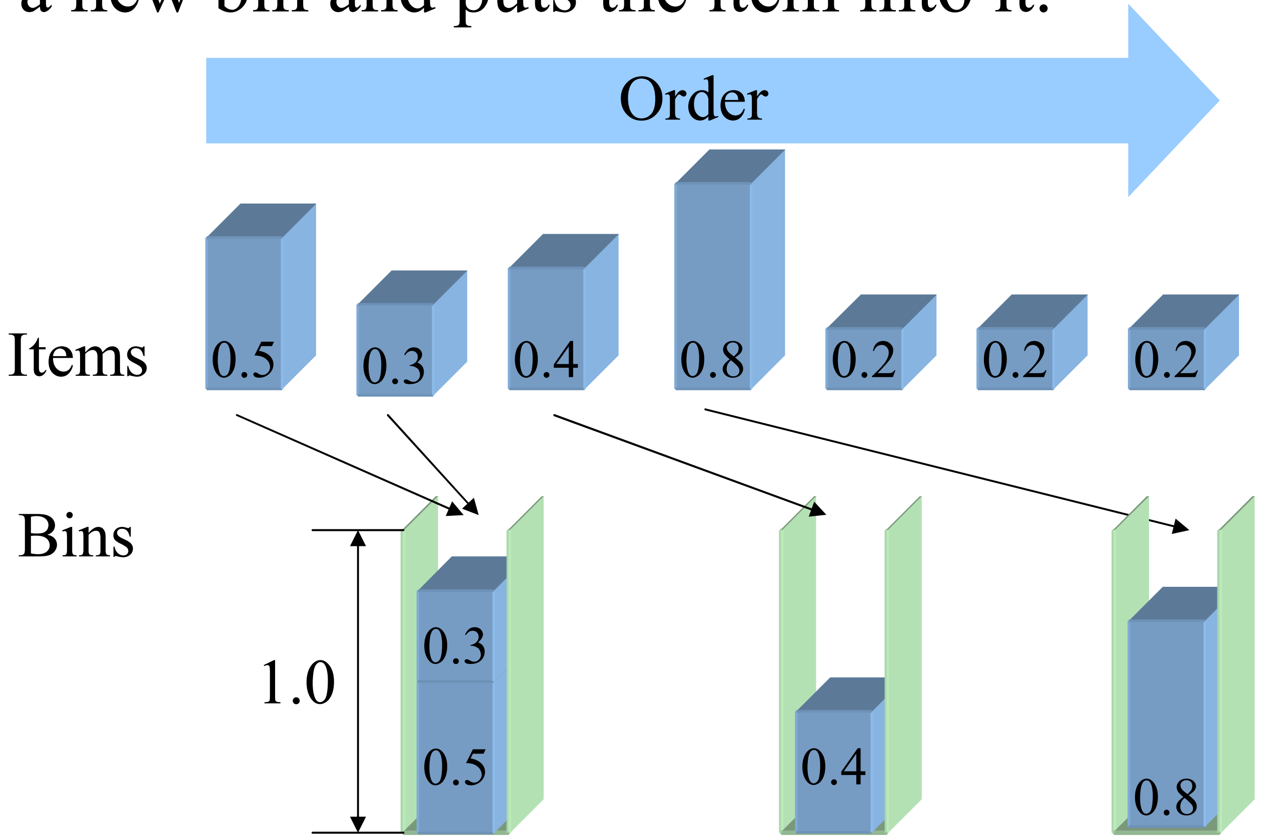
The First-Fit algorithm (1/4)

- This algorithm puts each item in one of partially packed bins.
  - If the item does not fit into any of these bins, it opens a new bin and puts the item into it.



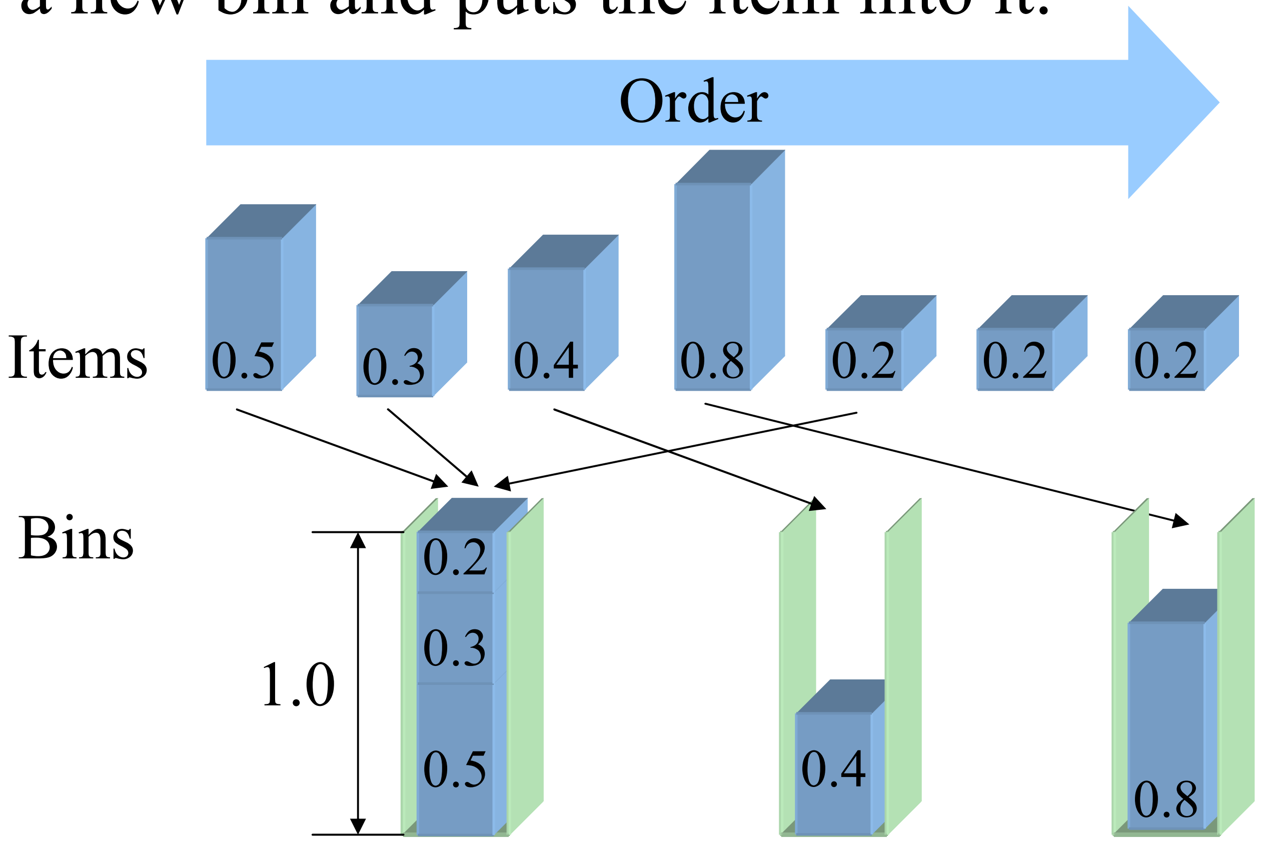
The First-Fit algorithm (2/4)

- This algorithm puts each item in one of partially packed bins.
  - If the item does not fit into any of these bins, it opens a new bin and puts the item into it.



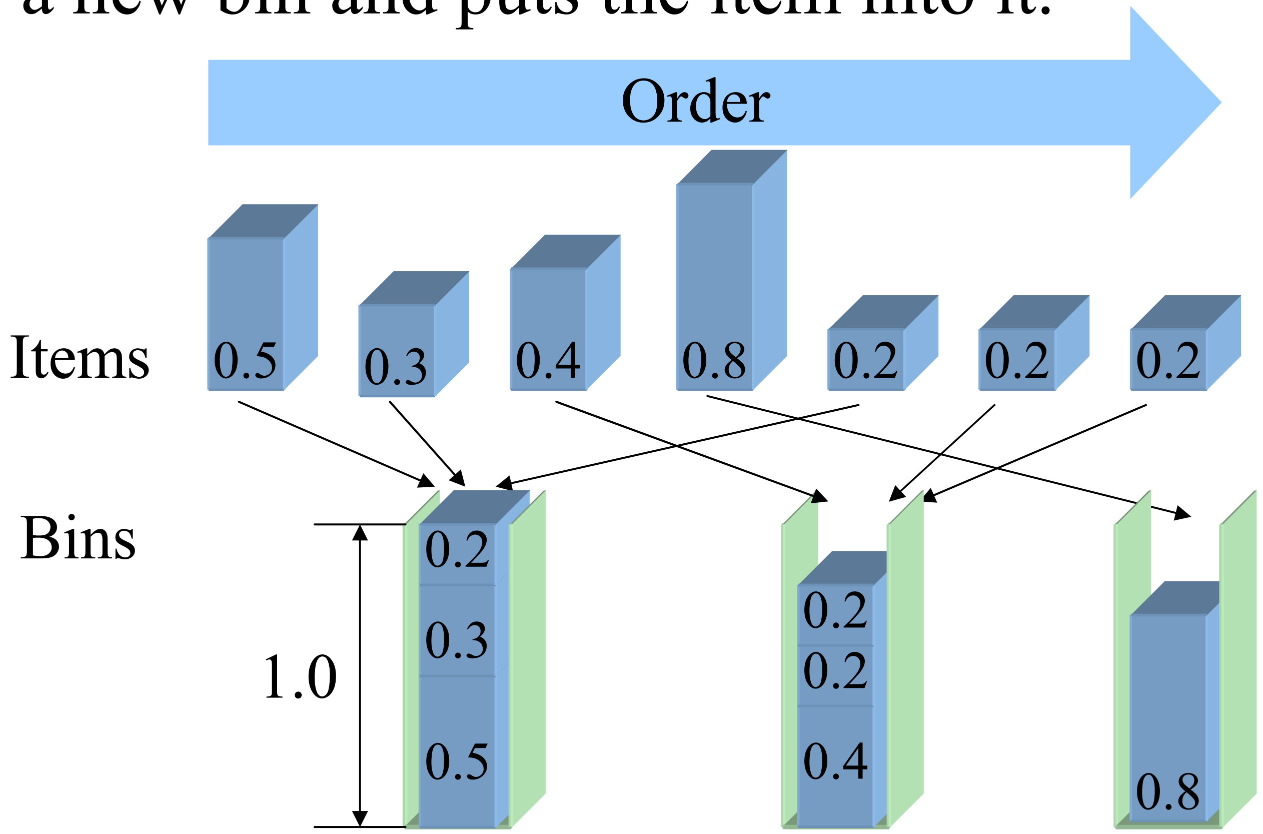
The First-Fit algorithm (3/4)

- This algorithm puts each item in one of partially packed bins.
  - If the item does not fit into any of these bins, it opens a new bin and puts the item into it.



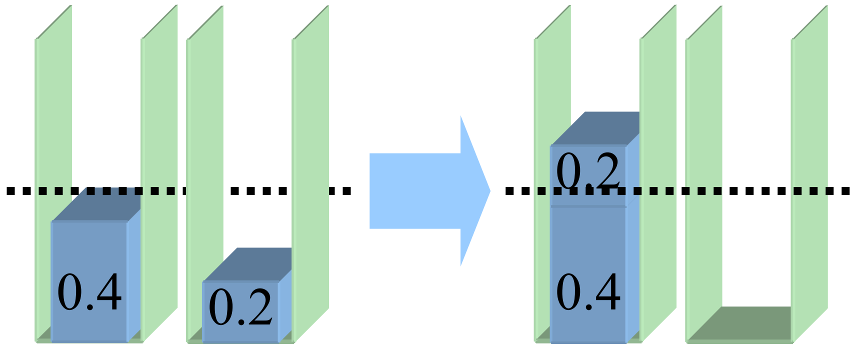
The First-Fit algorithm (4/4)

- This algorithm puts each item in one of partially packed bins.
  - If the item does not fit into any of these bins, it opens a new bin and puts the item into it.



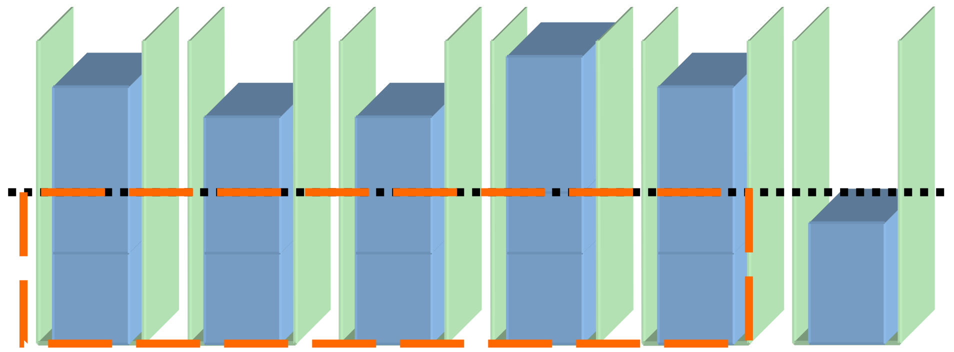
First-Fit finds a 2OPT solution (1/2)

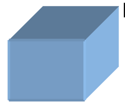
- OPT: # bins used in the optimal solution.
- [Proof]
  - Suppose that First-Fit uses *m* bins.
  - Then, at least (*m*-1) bins are more than half full.
    - We never have two bins less than half full.
      - If there are two bins less than half full, items in the second bin can be substituted into the first bin by First-Fit.



First-Fit finds a 2OPT solution (2/2)

- Suppose that First-Fit uses *m* bins.
- Then, at least (*m*-1) bins are more than half full.





Sum of sizes of the items

$$\mathrm{OPT} \geq \sum_{i=1}^{n} a_i > \frac{m-1}{2}$$

The size of

$$2\mathrm{OPT} > m-1$$

Since *m* and OPT are integers. $\longrightarrow$ $$2\mathrm{OPT} \geq m$$

Overview (3/4)

- Bin packing problem
  - An example
  - The First-Fit algorithm.
    - Approximation factor is 2.
  - No approximation algorithm having a guarantee of 3/2.
    - Reduction from the set partition, an NP-complete problem.
  - Asymptotic PTAS A_ε.
    - Lower bound of bins: ε, # distinct sizes of bins: K.
    - Exact algorithm where ε and K are constants.
    - Approximation algorithm where ε is constant.

No factor 3/2 approx. algorithms

- [Sketch of Proof]
  - Suppose that we have a factor 3/2 approximation algorithm *A*.
  - Then, *A* can find the optimal solution for the set partition problem in polynomial time.

  (Partitioning n +ve integers into two sets each adding up to half of the summation of all n numbers)

  This is Equivalent to n items to be packed in 2 bins.
    - Note that the set partition problem is NP-complete.
  - The result from the above assumption contradicts with $P \neq NP$.

Overview (3/4)

- Bin packing problem
  - An example
  - The First-Fit algorithm.
    - Approximation factor is 2.
  - No approximation algorithm having a guarantee of 3/2.
    - Reduction from the set partition, an NP-complete problem.
  - Asymptotic PTAS A_ε.
    - The minimum size of bins: ε, # distinct sizes of bins: K.
    - Exact algorithm where ε and K are constants.
    - Approximation algorithm where ε is constant.

Theorem 9.3

- We can find an approx. solution with factor [(1+2 ε) OPT+1]
  - where $0 < \varepsilon < 1/2$.
    - First-Fit is available if ε - 1/2.
      - The factor [(1+2 ε) OPT+1] > (2OPT+1) if ε □ 1/2.
  - 3 bins are required if OPT=2.
    - Consistent with the previous inapproximable result.
  - 1,001 bins are sufficient for an instance with OPT=1,000.
    - by setting ε=1/4,000.
  - Note: Its computation time is polynomial time but huge.
- We will follow the algorithm and proofs…

Algorithm

1. Remove items of size $< \varepsilon$ from the list

2. Partition all the items into groups of (k) where $k=[1/\varepsilon^2]$. Round items of each group to the largest size of the item belonging in it

3. Find an optimal packing

4. Use this packing for original item sizes

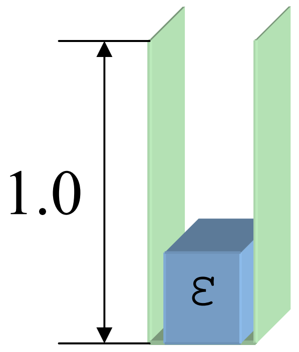
5. Pack items of size $< \varepsilon$ using First-Fit.

Lemma 9.4

- Consider bin packing with constraints (BP1)
  - The minimum size ε of items is a constant.
  - # distinct sizes of bins, K, is a constant.
- There exists a polynomial-time algorithm for BP1 that finds the best solution.
  - The algorithm searches for the solution exhaustively.
    - # combinations of items in a bin denotes R.
    - # combinations of n bins such that R distinct bins are available denotes P.
    - P is upper-bounded by a polynomial of n ($O(n^R)$).

Lemma 9.4

- Bin packing with constraints:
  - The minimum size ε of items is a constant.
  - # distinct bins, K, is a constant.

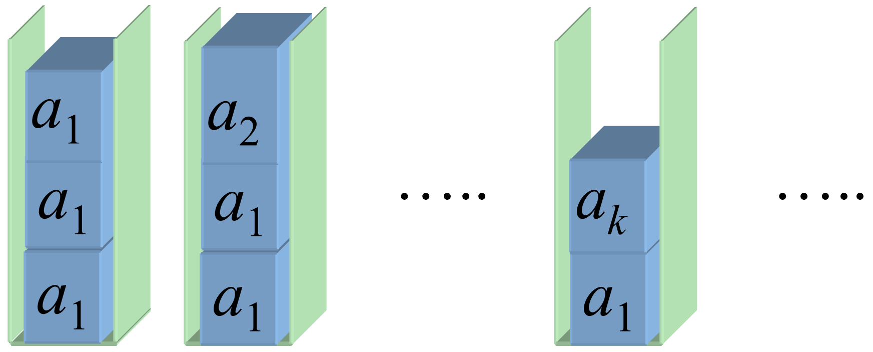


At most $\lfloor 1/\varepsilon \rfloor$ items are in a bin.

If ε=0.3, $\lfloor 1/\varepsilon \rfloor = \lfloor 3.33\ldots \rfloor = 3$.

M is the max. # of items in a bin.

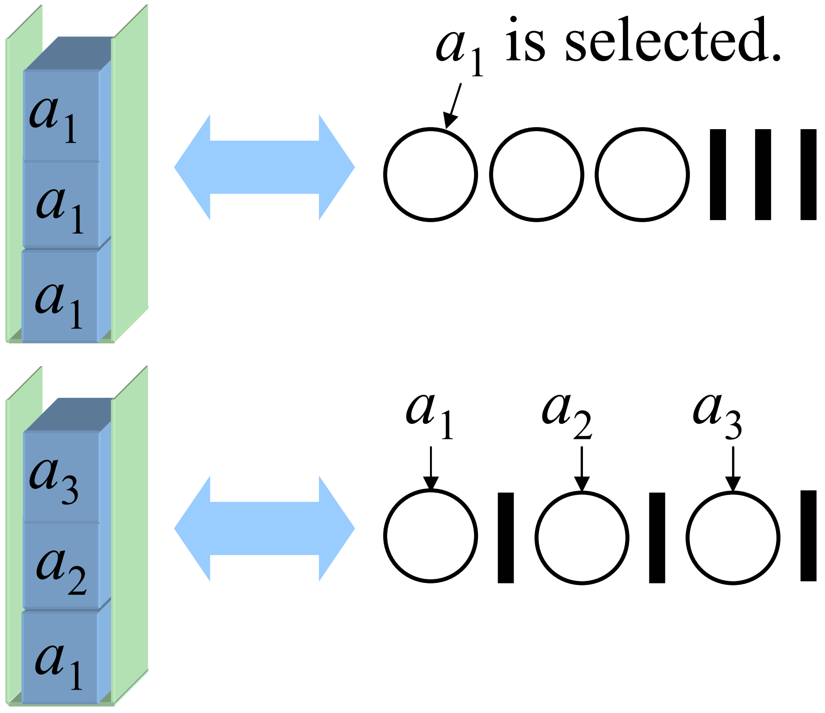
There are K distinct items.

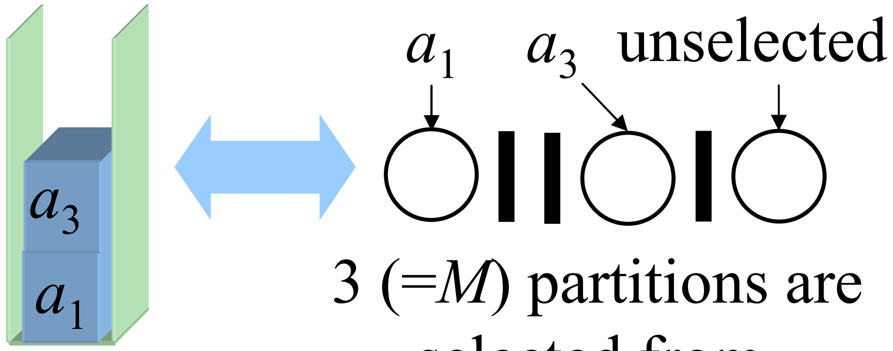


How many combinations of items are possible in a bin?

R, # combinations of items in a bin

- Pack M items in a bin from $(K+1)$ different items.
  - $K + 1$ = items with K sizes + empty (unselected).
  - E.g.) $K = 3$, $M = 3$.





3 (=M) partitions are selected from 6 (= $K + M$) elements.

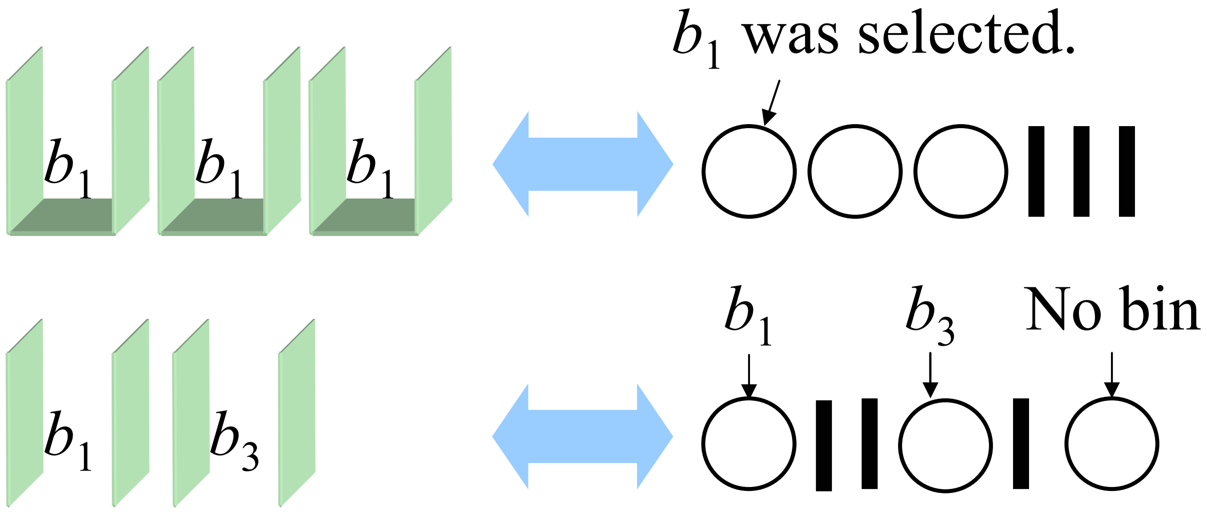
$$R = \binom{M+K}{M}$$

M and K are constants

$\rightarrow R$ is also constant.

P, # combinations of bins

- # combinations of bins with *R* different bins
  - We can find P in a similar way..
  - *P* can be bounded by a function of *n*.
  - E.g.) *R*=3, *n* = 3.



$$P = \binom{n+R}{R} \le (n+R)^R$$

$$= O(n^R).$$

# possible combination is bounded by a polynomial of *n*.

We can find the best packing by exhaustive search in polynomial time.

Examples of n^R

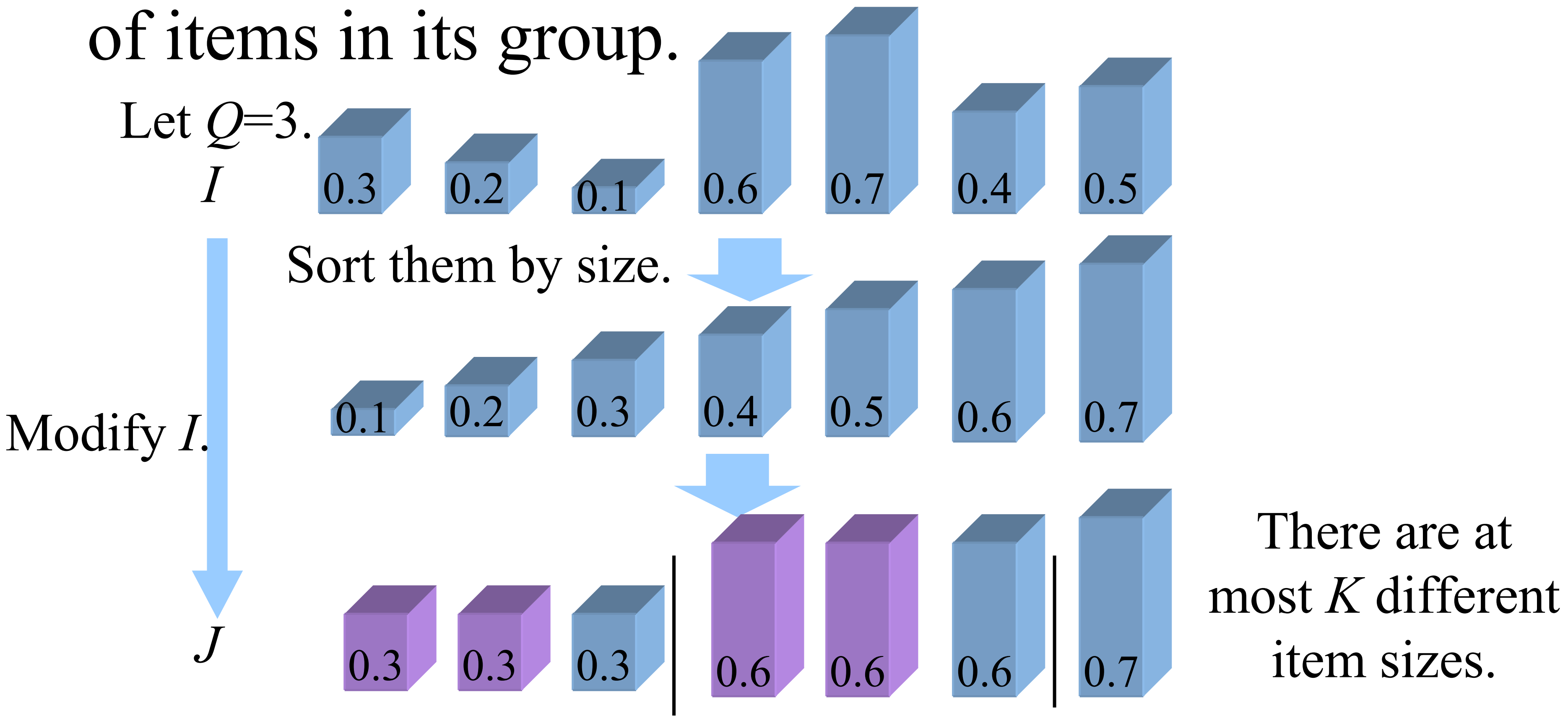
- ε: min. size of items, K: # different items.
  - $\varepsilon = 0.3$ ($M = 3$), $K=3$.
    - $R = {}_6C_3 = 20$, computation time $O(n^{20})$.
  - $\varepsilon = 0.1$ ($M = 10$), $K=3$.
    - $R = {}_{10}C_3 = 120$, computation time $O(n^{120})$.
  - $\varepsilon = 0.05$ ($M = 20$), $K=3$.
    - $R = {}_{20}C_3 = 1140$, computation time $O(n^{1140})$.

Lemma 9.5

- Bin packing with (less) constraints :
  - The minimum size ε of items is a constant.
- There exists a factor $(1+\varepsilon)$ approximation algorithm.
  - It first modifies the sizes of items into only K different ones.
  - It uses the exhaustive search used in Lemma 9.4.
    - $K = \lceil 1/\varepsilon^2 \rceil$, $Q = \lfloor n\varepsilon^2 \rfloor$.
      - Q: # items with the same size in a group.

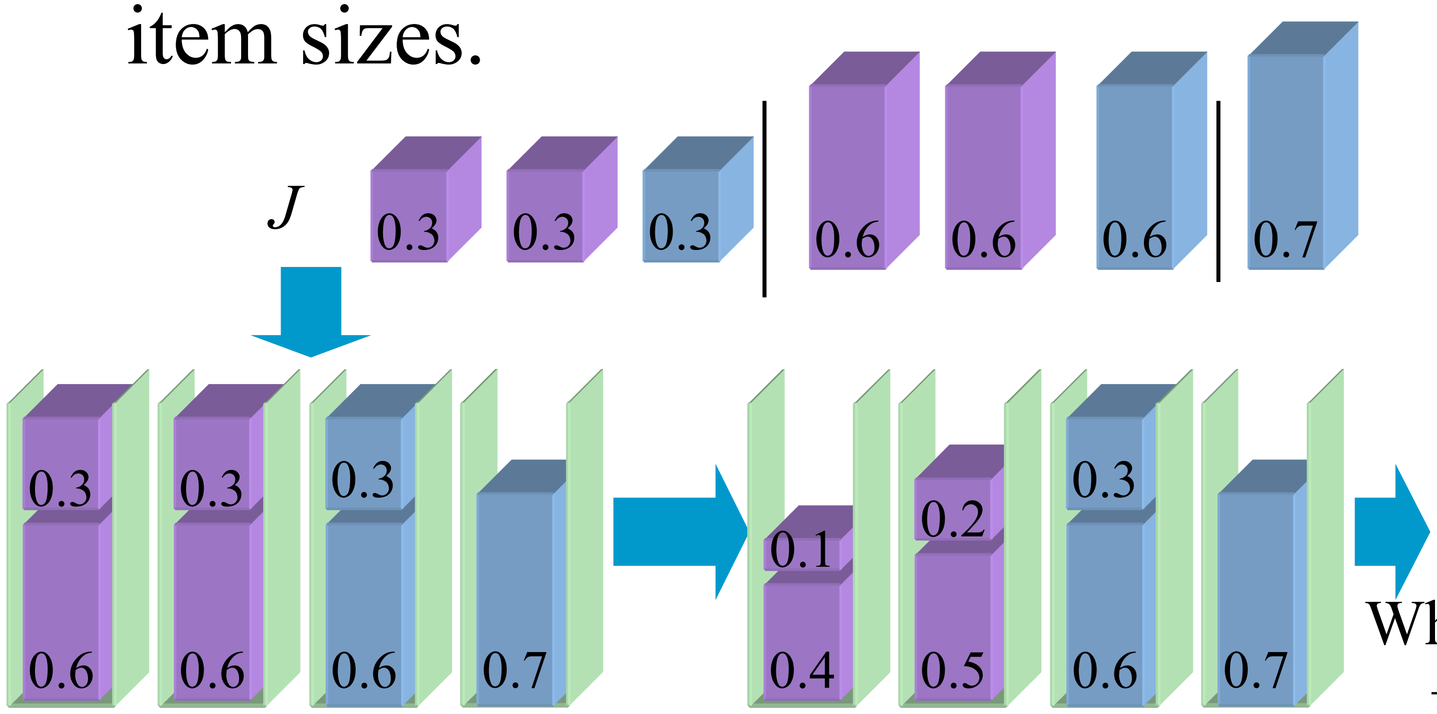
How to modify the sizes of items

- *I*: the original input, *J*: its modified one.
- *J* consists of *Q* groups.
- The size of each item is set to the maximum size of items in its group.



How to pack items

- From Lemma 9.4, the optimal packing for *J* can be found in polynomial time.
- The packing for *J* is also valid for the original item sizes.

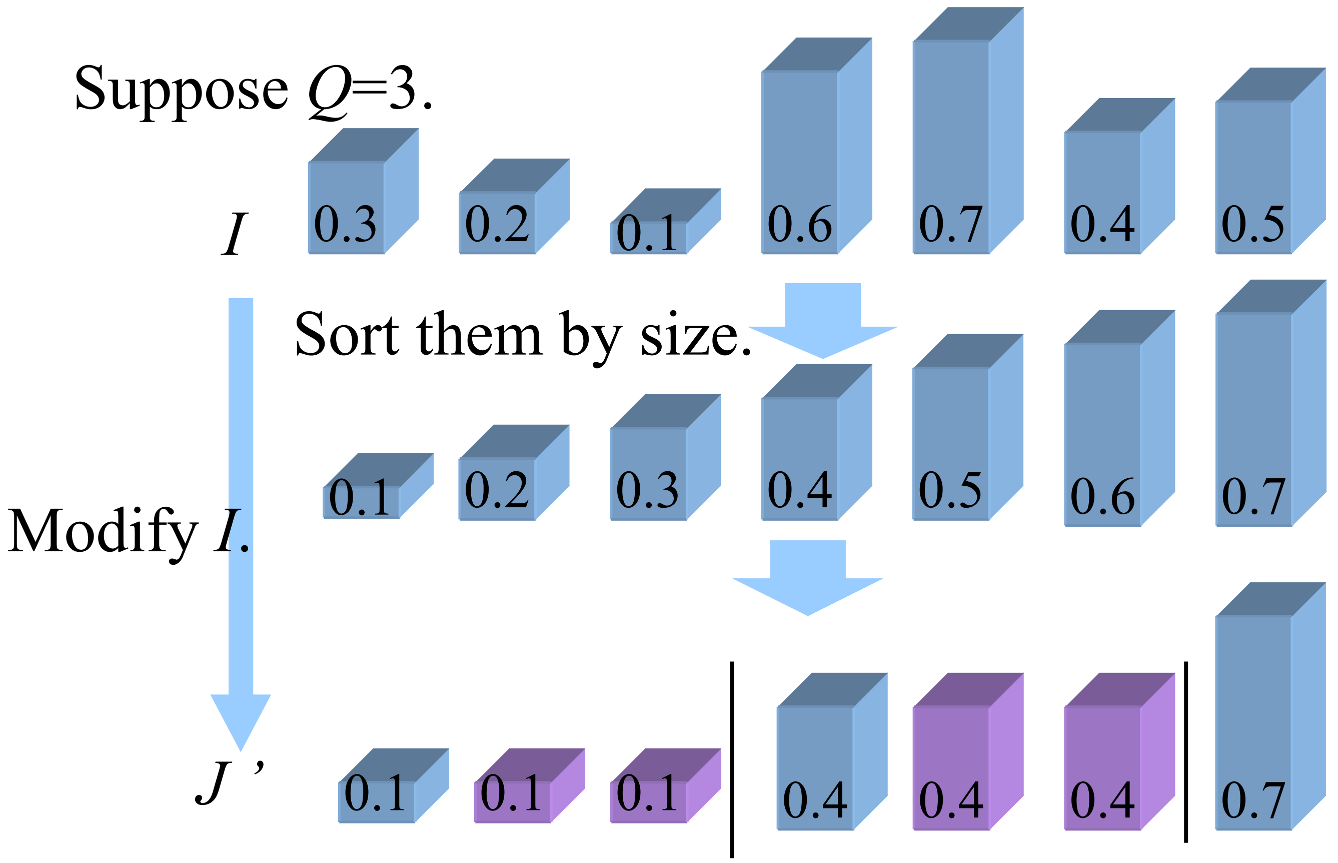


The optimal # bins denotes OPT(*J*).

What about approx. factor Z where OPT(*J*) $\leq z$ OPT(*I*)?

For evaluating OPT(I),…

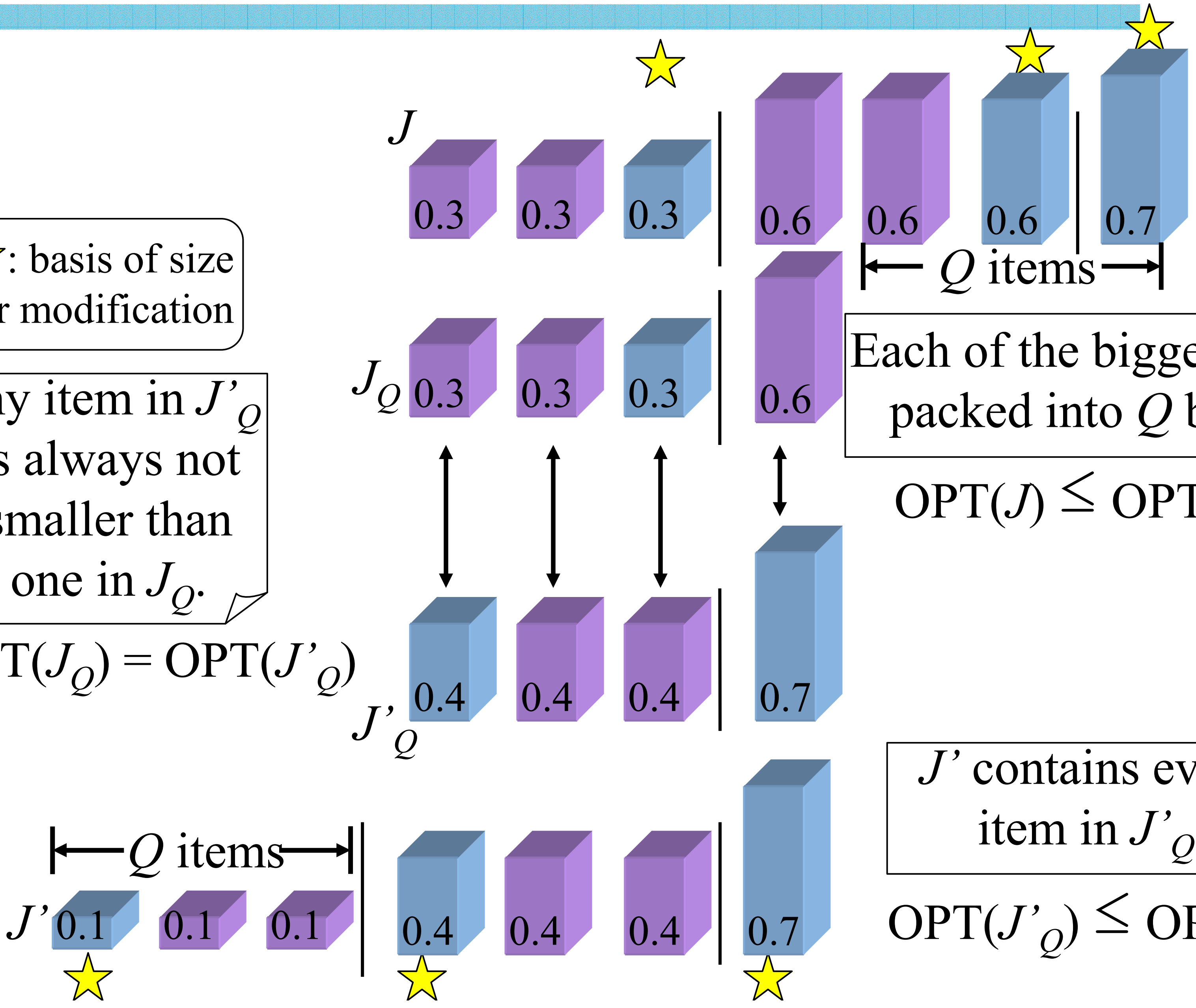
- We prepare another modified instance J'.
- J' consists of Q groups.
- Each item size is set to the minimum in its group.



Any item in J' is the same as or smaller than the original item in I.

OPT(J') ≤ OPT(I)

Diff. between OPT(J) and OPT(J')



☆: basis of size for modification

Any item in J'_Q is always not smaller than one in J_Q.

OPT(J_Q) = OPT(J'_Q)

Each of the biggest Q items packed into Q bins in J.

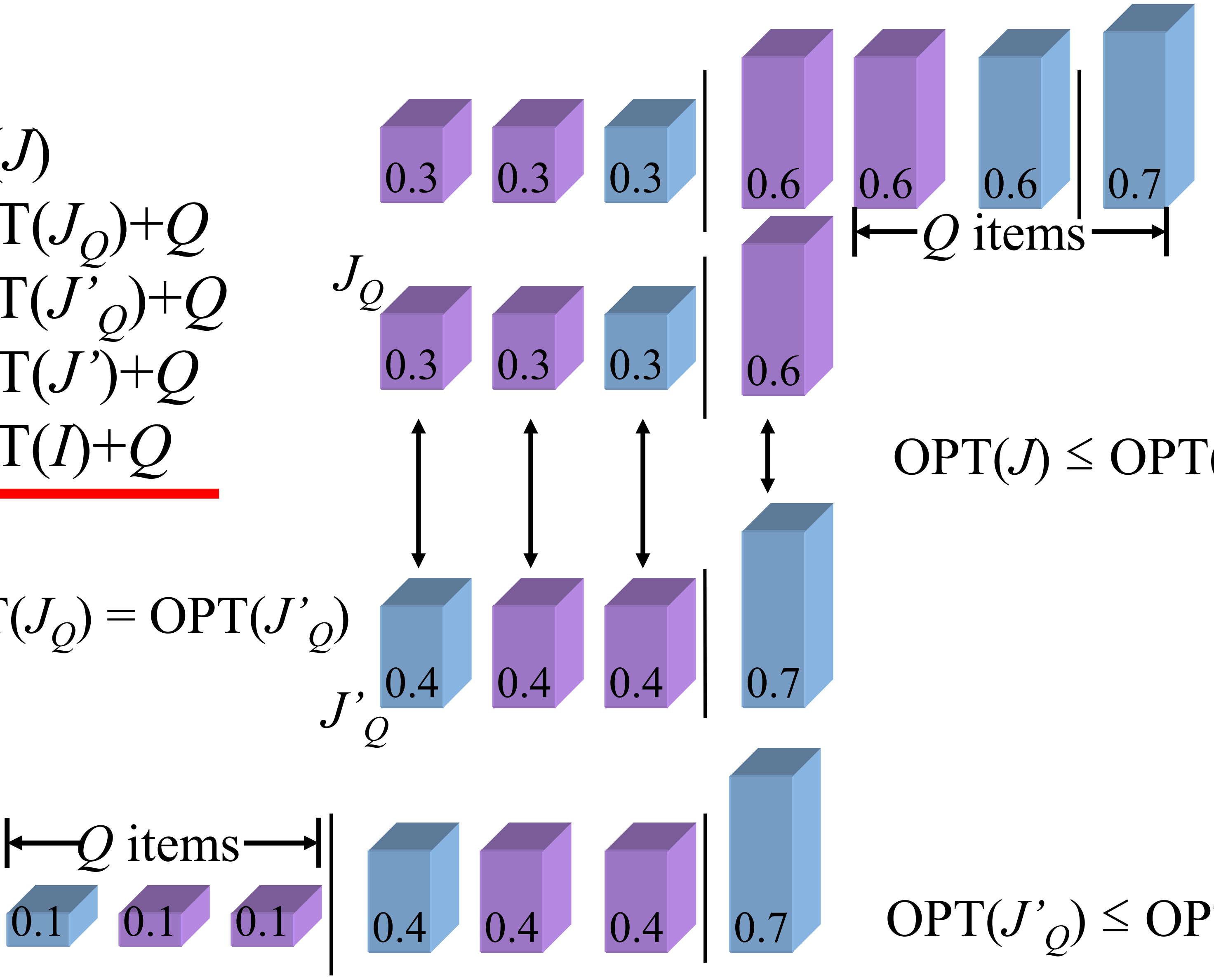
OPT(J) ≤ OPT(J_Q)+Q

J' contains every item in J'_Q.

OPT(J'_Q) ≤ OPT(J')

Diff. between OPT(J) and OPT(J')

OPT(J)
$\leq$ OPT(J_Q)+Q
= OPT(J'_Q)+Q
$\leq$ OPT(J')+Q
$\leq$ OPT(I)+Q



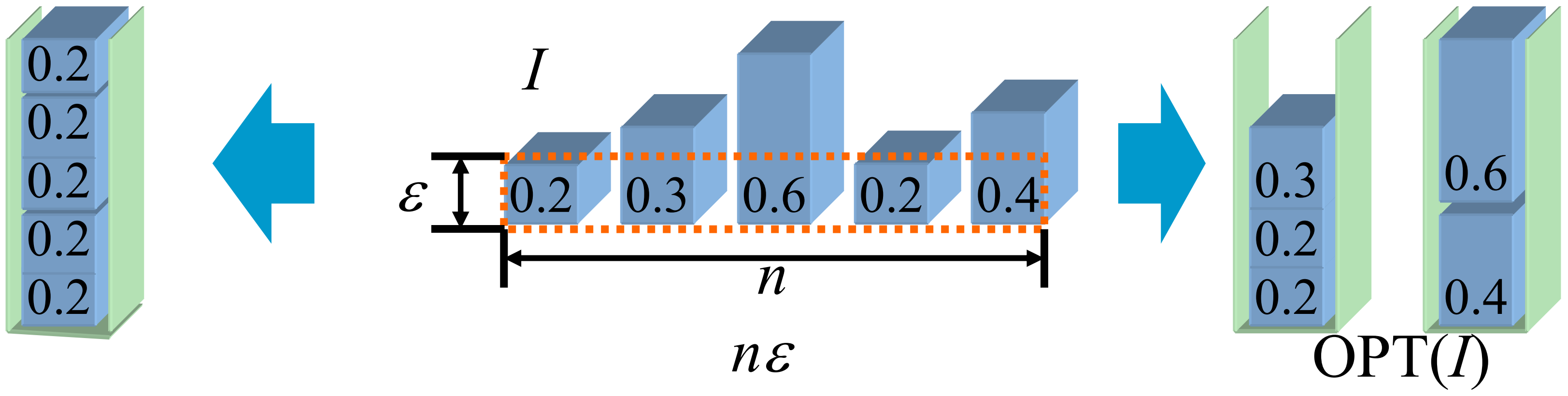
OPT(J) $\leq$ OPT(J_Q)+Q

OPT(J_Q) = OPT(J'_Q)

OPT(J'_Q) $\leq$ OPT(J')

Relation between Q and OPT(I)

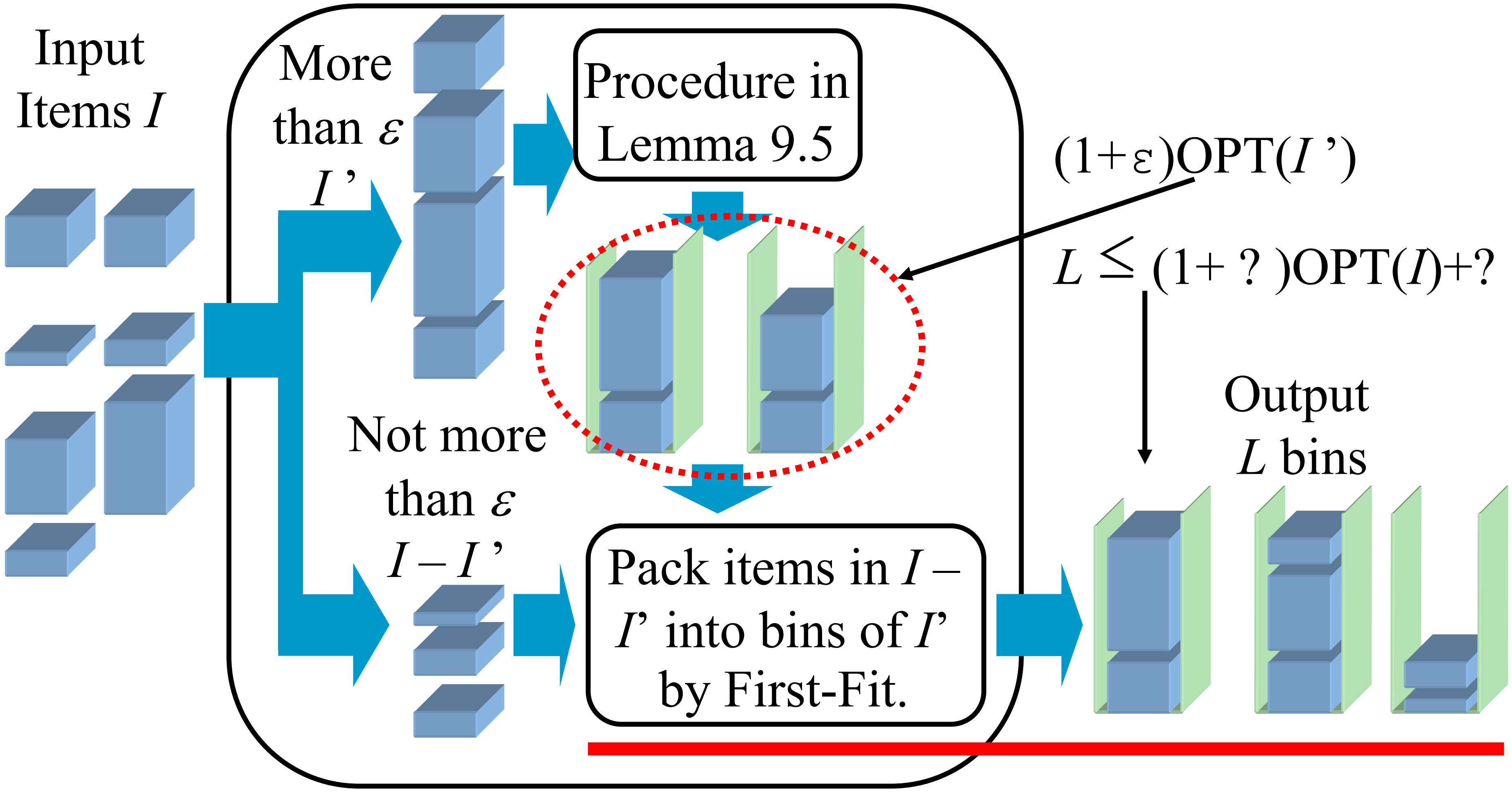
- $Q \le n\varepsilon^2$ since $Q = \lfloor n\varepsilon^2 \rfloor$.
- $\mathrm{OPT}(I) > n\varepsilon$ since.
  - ε = any item size,
  - $n\varepsilon$ = the total item size (n: # items),
  - The total item size = # bins (that is, OPT(I))
- Then, $Q \le n\varepsilon^2 = \varepsilon(n\varepsilon) \le \varepsilon\mathrm{OPT}(I)$.
- $\mathrm{OPT}(J) \le \mathrm{OPT}(I) + Q \le \underline{(1+\varepsilon)\ \mathrm{OPT}(I)}$.



Theorem 9.3

- We can find an approx. solution with factor [(1+2 ε) OPT+1]
  - where $0 < \varepsilon < 1/2$.
    - First-Fit is available if $\varepsilon = 1/2$.
      - The factor [(1+2 ε) OPT+1] > (2OPT+1) if $\varepsilon = 1/2$.
  - 3 bins are required if OPT=2.
    - Consistent with the previous inapproximable result.
  - 1,001 bins are sufficient for an instance with OPT=1,000.
    - by setting ε=1/4,000.

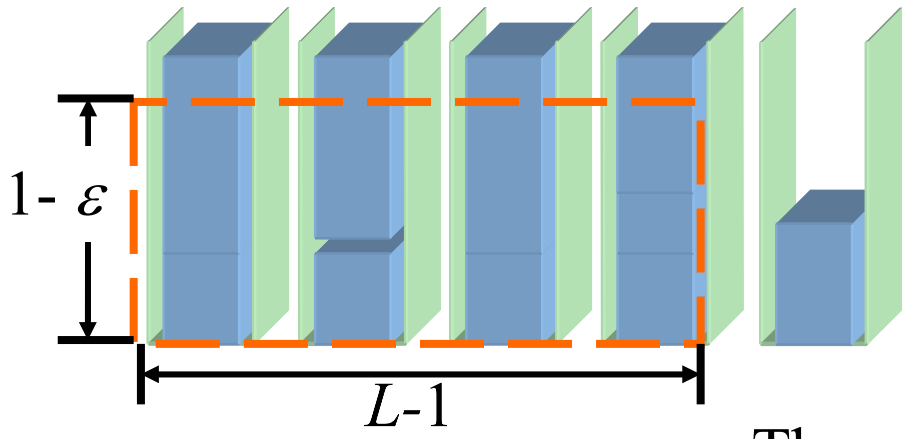
Algorithm A_ε


Input
Items I
More than ε
I'
Procedure in Lemma 9.5
$(1+\varepsilon)\mathrm{OPT}(I')$
$L \leq (1+?)\mathrm{OPT}(I)+?$
Not more than ε
$I - I'$
Pack items in $I - I'$ into bins of I' by First-Fit.
Output
L bins

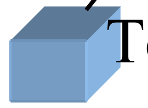
We consider two exclusive case to find the upper bound of L.

Evaluation of A_ε (1/2)

- Consider two exclusive cases:
  1. No extra bin was required for packing items in $I - I'$.
  - $L \leq (1+\varepsilon)\mathrm{OPT}(I') \leq (1+\varepsilon)\mathrm{OPT}(I)$.
    - Since there are more items in I than in I'.
  2. Extra bins were required for packing items in $I - I'$.
  - In each of L-1 bins, room is smaller than ε.



$$\mathrm{OPT} \geq \sum_{i=1}^{n} a_i > (L-1)(1-\varepsilon)$$



Total sum of item sizes

Item size in

Then, we have $\dfrac{\mathrm{OPT}}{1-\varepsilon} + 1 \geq L$.

Evaluation of algorithm A_ε (2/2)

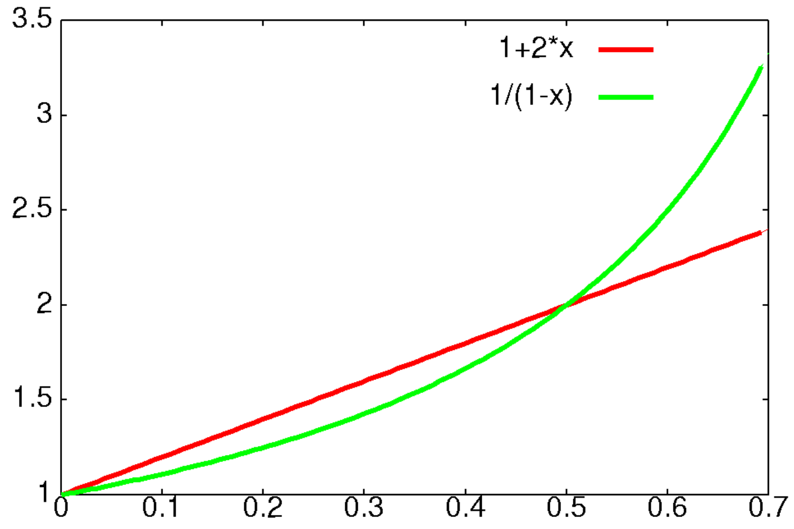
$$1+2\varepsilon-\left(\frac{1}{1-\varepsilon}\right)=\frac{1}{1-\varepsilon}\left((1-\varepsilon)(1+2\varepsilon)-1\right)$$
$$=\frac{1}{1-\varepsilon}\left(1+\varepsilon-2\varepsilon^2-1\right)$$
$$=\frac{\varepsilon}{1-\varepsilon}\left(1-2\varepsilon\right)\geq 0 \qquad (0<\varepsilon\leq\frac{1}{2}).$$

Then, we have

$$1+2\varepsilon\geq\frac{1}{1-\varepsilon}\ \ (0<\varepsilon\leq\frac{1}{2}).$$

Therefore,

$$L\leq\frac{\mathrm{OPT}}{1-\varepsilon}+1\leq(1+2\varepsilon)\mathrm{OPT}+1\ \left(0\leq\varepsilon\leq\frac{1}{2}\right).$$



Conclusion

- We consider the bin packing problem.
  - For *almost* all instances, we can obtain its solution with any approximation factor.
  - There is an approx. algorithm to find factor 2 solution.
  - It is impossible to find a solution with arbitrary approximation ratio under P is not equal to NP.
  - There is an approx. algorithm with arbitrary approximation ratio for large size instances.