Lecture 7

Approximation Algorithms

Last Time

- Local Search Algo. for min degree spanning tree

- Greedy + Loc. Sear. Algo. for Edge Coloring

Chapters 1 & 2
(Williamson & Shmoys)

Today

- Bin Packing Approx. Algo. using Dynamic Programming

- Prize Collecting Steiner Tree Problem (LP rounding - deterministic)

Bin Packing

* Given n items of sizes $a_1, a_2, \cdots, a_n$ such that $1 > a_1 \geq a_2 \geq \cdots \geq a_n > 0$.

* A bin can hold a subset of items of size ≤ 1

* Goal: Pack items into as few bins as possible.

Theorem: Unless $P = NP$, $\exists$ no ρ-approx. algo for bin packing for any $\rho < \frac{3}{2}$.

(Reading exercise)

via reduction from partition problem

WARMUP

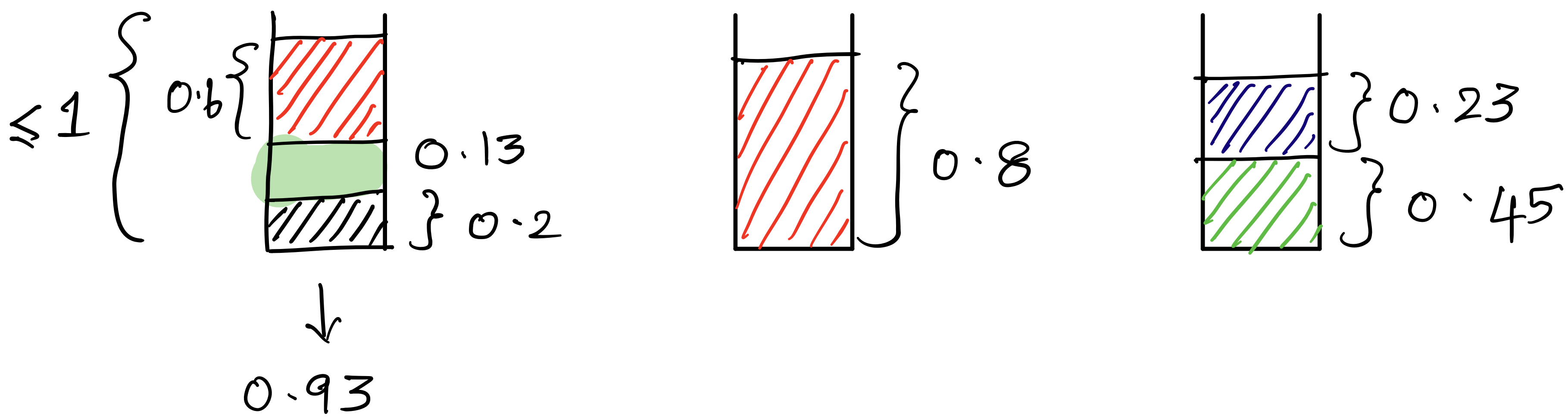
First-Fit Algorithm

(No need to sort items by sizes)

1. for $i \in [n]$:
   * if a_i does not fit into any opened bin so far, open a new bin and add a_i.

Suppose items have sizes

0.2 , 0.13 , 0.6 , 0.8 , 0.45 , 0.23



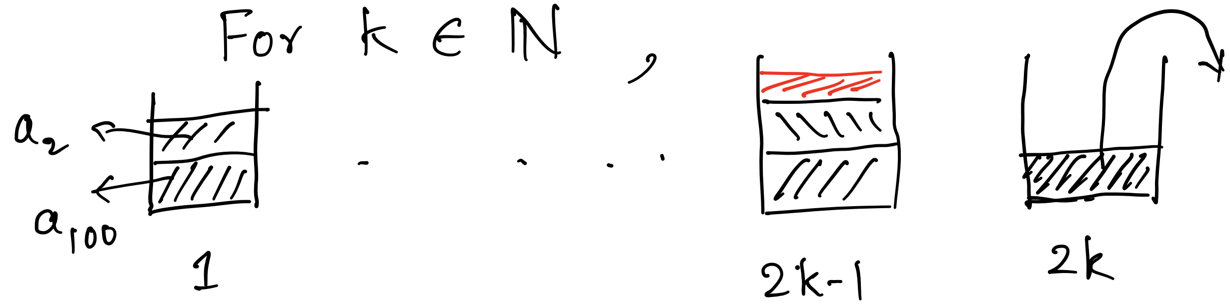
I - input instance ; FF (I) - solution value of first fit algo.

OPT (I) - optimal soln. value on instance

Lemma: $FF(I) \leq 2 \cdot OPT(I) + 1$

Proof:-

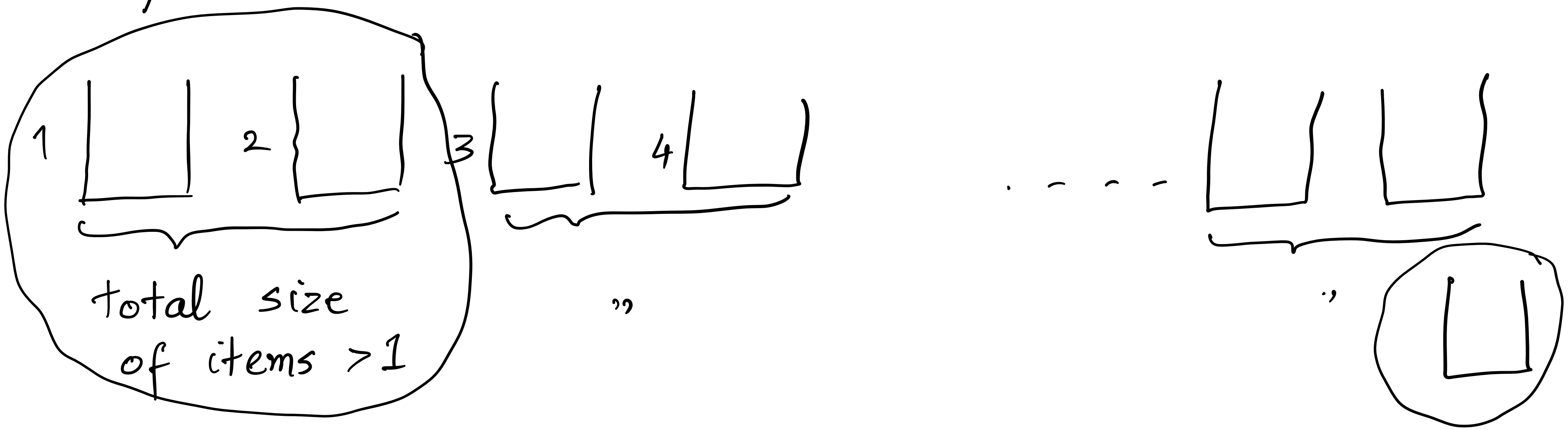
For $k \in \mathbb{N}$,



item could not be placed in $2k-1$

$\Rightarrow$ sum of sizes of items in $2k-1$ & $2k$ > 1

Similarly



$$\star \quad \sum a_i = \mathrm{SIZE}(I) \le \mathrm{OPT}(I)$$

$$\star \quad \mathrm{SIZE}(I) > \overbrace{\left\lfloor \frac{\#\text{boxes}}{2} \right\rfloor}^{\#\text{pairs of boxes}} = \left\lfloor \frac{\mathrm{FF}(I)}{2} \right\rfloor$$

$$\Rightarrow \quad \mathrm{FF}(I) \le 2\,\mathrm{OPT}(I) + 1$$

$$\left.\begin{array}{l}\text{Sum of sizes} \\ \text{of items in bins} \\ 2, 3, \ldots, \underbrace{\ell}_{FF(I)} - 1\end{array}\right\} \leq SIZE(I) - 1$$

$$\Rightarrow \; FF(I) - 1 \leq \left\{\begin{array}{l}\text{Sum of sizes} \\ \text{of items in} \\ (1,2), (2,3), \ldots, (\ell-1, \ell)\end{array}\right\} \leq 2 + 2\cdot\left(SIZE(I) - 1\right)$$

$$= 2\cdot SIZE(I) + 1$$

$$\Rightarrow \; FF(I) \leq 2\cdot SIZE(I) + 2$$

NEXT : Poly time algo. outputting a packing with $(1+\varepsilon)\,OPT(I) + 1$ bins.

for $\varepsilon > 0$

ASSUMPTION 1 : Each piece has size $> \frac{\varepsilon}{2}$

i/p : I

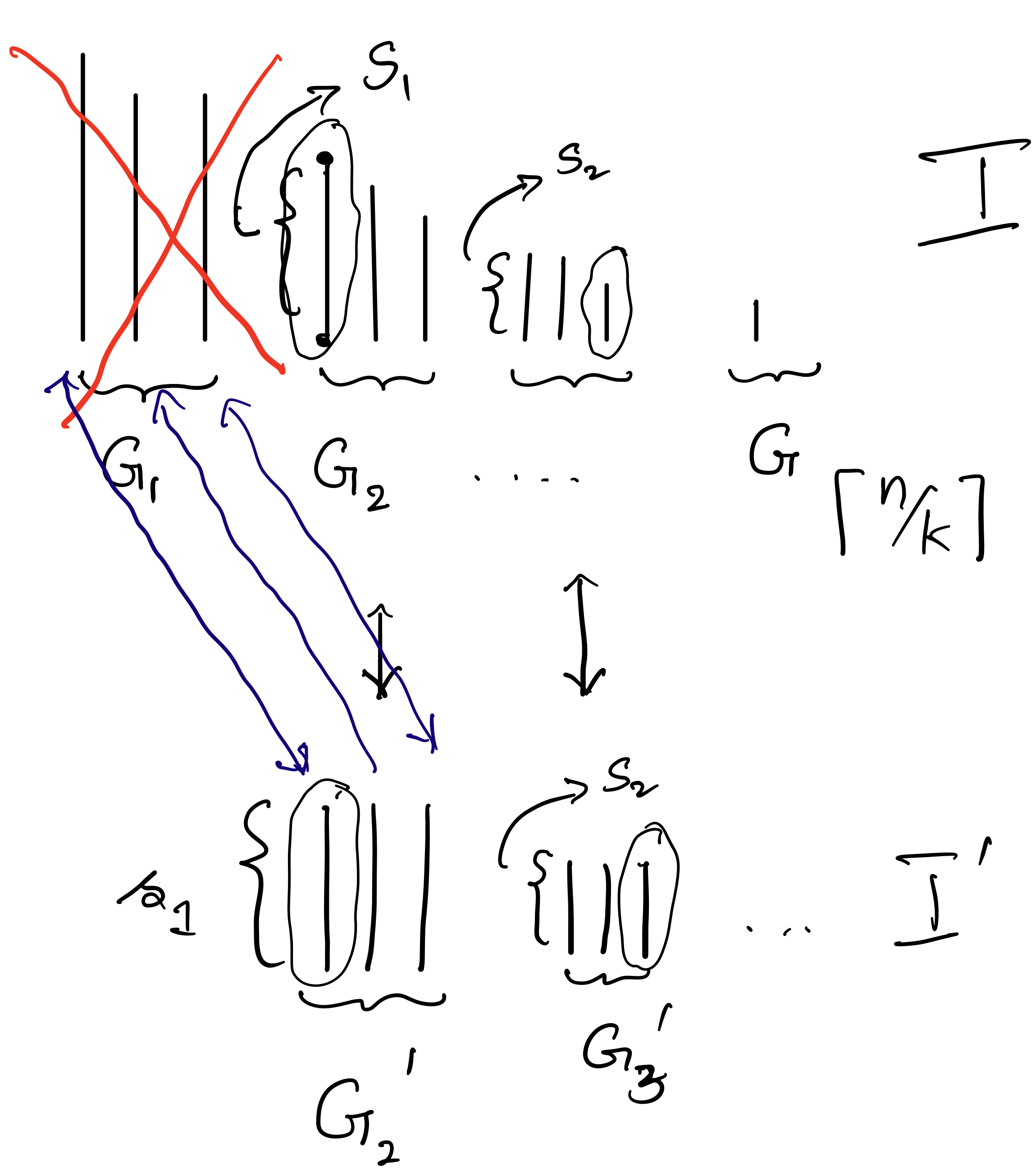
ALGORITHM$_\varepsilon$

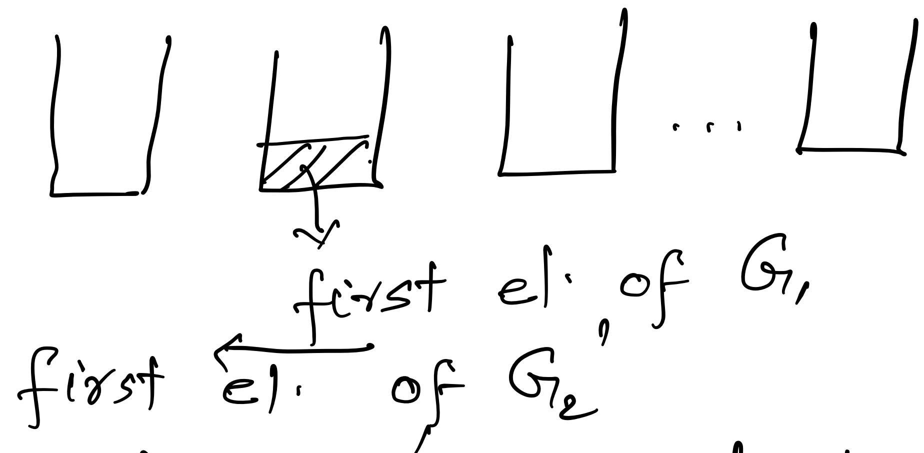
$\{A_\varepsilon\}_{\varepsilon > 0}$

(1) Let $k = \lfloor \varepsilon \cdot SIZE(I) \rfloor$

2 Linear grouping

* Consider pieces in decreasing order of size
* Group pieces into groups of size k
* Drop first group & equalize lengths in remaining groups to get new instance I'





Claim: I' packed into x bins $\Rightarrow$ I packed into $\leq x+k$ bins

Claim 2

I packed into y bins $\Rightarrow$ I' packed into y bins.

(3) Solve the bin packing problem on I' exactly using dynamic programming.

How ?

$\rightsquigarrow k = \lfloor \varepsilon \, \mathrm{SIZE}(I) \rfloor$

$\sum_{i \in [n]} a_i$ = Sum of sizes of all items in I

* if $k < 1$, $\mathrm{SIZE}(I) < \frac{1}{\varepsilon}$

grouping is meaningless

$\Rightarrow$ # ~~pieces~~ items $< \frac{1/\varepsilon}{\varepsilon/2} = \frac{2}{\varepsilon^2}$

Solve bin packing on I optimally via brute force

* else if $k = \lfloor \varepsilon \, SIZE(I) \rfloor \geq 1$,

t = # distinct item sizes in I'

$\leq \frac{n}{k} \leq \frac{2n}{\varepsilon \cdot SIZE(I)} \leq \frac{4}{\varepsilon^2}$

- $\lfloor \alpha \rfloor \geq \frac{\alpha}{2}, \forall \alpha \geq 1$
- $SIZE(I) \overset{?}{\geq} \frac{\varepsilon}{2} \cdot n$

Let these item sizes be $1 \geq s_1 > s_2 > \dots > s_t \geq \frac{\varepsilon}{2}$

$\therefore$ Represent I' as a vector

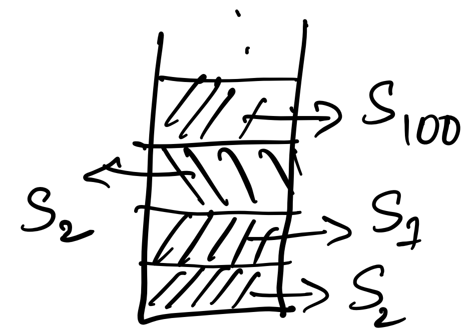
$(n_1, n_2, \dots, n_t)$ $\quad \sum n_i$ = # pieces in I'

n_i = # pieces of size s_i.

- A bin can contain $\leq \frac{2}{\varepsilon}$ pieces $(1, 2, \ldots, 1, \ldots)$

Defn:
- A bin configuration is a vector $(b_1, b_2, \ldots, b_t)$ s.t.

  b_i = # pieces in bin with size s_i



  $\sum b_i s_i \leq 1$

- #distinct bin configs $\leq \left(\frac{2}{\varepsilon}\right)^t$

$t \leq 4/\varepsilon^2$

$\leq \left(\frac{2}{\varepsilon}\right)^{(4/\varepsilon^2)}$

Recurrence

vector rep. of I'

$$\mathrm{OPT}(\overline{n_1, n_2, \ldots, n_t}) = \min_{\substack{(b_1, b_2, \ldots, b_t) \\ \text{bin config}}} \mathrm{OPT}(n_1 - b_1, \ldots, n_t - b_t) + 1$$

n_i = # items in I' of size s_i

Boundary cases:

one for each bin config.

$$\mathrm{OPT}(b_1, \ldots, b_t) = 1$$

Time to solve the recurrence by table filling $\Big\} = O(n^t)$

So, we solved bin packing exactly on I'.

$\underbrace{\text{packing exactly}}_{OPT(I')}$

Lemma: $OPT(I) \geq OPT(I')$

&

any packing of I' can be used to generate a packing of I with at most k more boxes

We get a packing of I using $OPT(I') + k$ bins $\leq OPT(I) + k$

__Thm:__ if each piece has size $\geq \frac{\varepsilon}{2}$, above algo. outputs a packing using $\leq OPT(I) + \lfloor \varepsilon \, SIZE(I) \rfloor$

$\leq (1+\varepsilon)\, OPT(I)$ bins.

Not all pieces need be large

<u>Claim</u>: Any packing of all pieces of size $> \gamma$ into l bins can be extended to a packing for the entire input with $\leq \max\left\{ l, \frac{SIZE(I)}{1-\gamma} + 1 \right\}$ bins

(Reading exercise: Lem 3.10 in book)

$$\gamma = \frac{\varepsilon}{2} \; ; \quad \ell = (1+\varepsilon)\,\mathrm{OPT}(I)$$

$$\max \left\{ \ell \, , \quad \frac{\mathrm{SIZE}(I)}{1 - \frac{\varepsilon}{2}} + 1 \right\}$$

$$\frac{\mathrm{SIZE}(I)}{1 - \frac{\varepsilon}{2}} \lesssim \mathrm{OPT}(I)\,(1+\varepsilon)$$