Lec 4
Wednesday, August 27, 2025 2:15 PM

Announcements

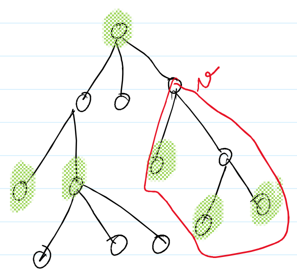
- HW 1 due Today
- Office Hours in Klaus 2108 (see Canvas or Piazza for Google Calender)
- Exam 1 on Sep 18 (instead of Sep 16)
- Recordings on Canvas, Handwritten on Piazza
- HW-2 to be out soon (due Thurs Sep 11)

Plan for today: More DP — Max Indep Set,
Knapsack pb $\leftarrow$ Poly vs Pseudo poly time
$(1-\epsilon)$- approx for knapsack

Max Independent Set on a Tree

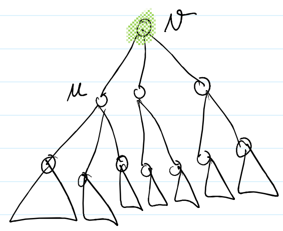
Given a rooted tree $T = (V, E)$

Find largest $S \subseteq V$
s.t. $\forall\ u, v \in S$
$(u, v) \notin E$



Define: $M[v]$ denotes size of largest indep set in subtree rooted at v

Ans: $M[\text{root}]$



Thm: We can compute $M[\ ]$ in $O(n)$ time

Pf: Base case: $M[\text{leaf}] = 1$

Induc Step: $M[v] = \max \begin{cases} 1 + \sum\limits_{u \in \text{Grand Child}(v)} M(u), \\ \sum\limits_{u \in \text{Child}(v)} M(u) \end{cases}$

[illegible]: Total time is still $O(n)$

$\sum_{u \in \text{Child}(v)}$

Claim: Total time is still $O(n)$

$\because$ each vertex u appears once in it's parent's calculation
& once " grand-parent's calculation

Knapsack Problem

Given a budget B ← integer

n jobs : Size s_i and value v_i (assume $s_i \le B$)

Goal: Find $A \subseteq \overset{\{1,2,\ldots,n\}}{[n]}$ s.t.

$\sum_{i \in A} s_i \le B$ and we maximize $\sum_{i \in A} v_i$

E.g. 4 jobs

sizes $\{6, 3, 4, 2\}$ Budget $= 10$

values $\{30, 14, 16, 9\}$

Remark: Knapsack is an 'NP Hard' problem, so we don't think it's polytime solvable.

Polytime vs Pseudo-Polytime algo

↑ Running time poly in # of $\underbrace{\text{input bits}}$

$\left(\sum_i \log s_i\right) + \left(\sum_i \log v_i\right) + \log B$

Runtime might assume some input paramters are small
↑ or given in unary representation

Thm 1: We can solve Knapsack in time $O(n^2 \cdot \underbrace{\max_i v_i})$.

think of all values being small

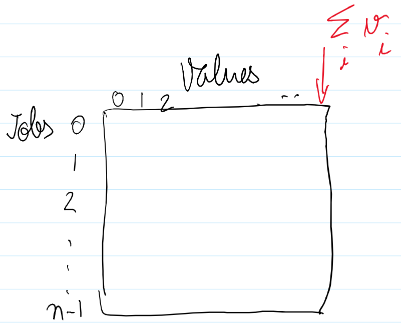
Define: $M(i, v) =$ Min total knapsack size that suffice to get exactly value v using jobs $\{0, \ldots i\}$

Define - $M(i, v) =$ min total [illegible]
suffice to get exactly value v using jobs $\{0, \dots i\}$

We set $M(i,v) = \infty$ if v is not achievable

Base Case: $\leftarrow$ Exercise

$$M(i, v) = \min \begin{cases} M(i-1, v), \\ s_i + M(i-1, v - v_i) & \text{for } v \geq v_i \end{cases}$$



$$\text{Time} = O\left(n \cdot \sum_i v_i\right) = O\left(n^2 \cdot v_{max}\right)$$
$$= \text{Space}$$

<u>Ans:</u> max v such that $M(n-1, v) \leq B$

Approx Algorithm

What if we don't want to assume values & want a polytime algo?

Give up on optimality.

Given a fixed error parameter ϵ (think $\epsilon = 0.01$)
$\uparrow$ constant

Design an Alg with value $\geq (1-\epsilon)$ Optimum and run time
poly$\left(n, \log B, \sum_i \log v_i\right)$.
$\sum_i \log s_i$

E.g. $n^{1/\epsilon} \left(\log B \cdot \log v_{max}\right)^{10}$

Such an Alg is called Polytime Approx Scheme (PTAS), i.e., runtime is poly(# input bits) assuming ϵ is a constant.

Thm : For knapsack, there exists a PTAS (infact, depency on ϵ is just $\text{poly}(1/\epsilon)$)

Plan : Replace v_i by $\tilde{v}_i$ such we can apply pseudo-polytime algo.

Then argue the obtained soln is $(1-\epsilon)$ approx.

Proof : Define $K = \frac{\epsilon \max_i v_i}{n}$

let $\tilde{v}_i := K \left\lfloor \frac{v_i}{K} \right\rfloor \Rightarrow v_i \geq \tilde{v}_i \geq v_i - K$

Moreover, for any set S of jobs

$$\sum_{i \in S} \tilde{v}_i \geq \sum_{i \in S} (v_i - K) \geq \sum_{i \in S} v_i - \epsilon v_{\max}$$

$$\geq \sum_{i \in S} v_i - \epsilon \, opt \quad \text{since } opt \geq v_{\max}$$

Alg : (1) Run Pseudo-polytime algo with jobs having value $\frac{\tilde{v}_i}{K}$ & size s_i and budget B. (Simplified instance)

(2) Return the obtained set S of jobs

Runtime : $O(n^2 v_{\max}) = O\left(n^2 \cdot \frac{\max \tilde{v}_i}{K}\right) = O\left(n^2 \cdot \frac{\max v_i}{K}\right)$

Ignores bit-complexity $= O(n^3/\epsilon)$

Alg's value : Want to show $\sum_{i \in S} v_i \geq (1-\epsilon)\, opt$