Reading: 11.8

Last Time:

- approximation
- metric TSP
- knapsack

Today:

- pseudo polynomial time
- knapsack $(1+\epsilon)$ approx.

Def: $\mathcal{A}$ is an β-approximation the value of its solutions is at least OPT/β (maximization problems)

Recall: knapsack problem

input:

- n objects
- sizes s_i (non-negative real number)
- values v_i
- capacity C.

output: subset S that

- fits: $\sum_{i \in S} s_i \leq C$
- maximizes values: $\sum_{i \in S} v_i$.

Pseudo-polynomial Time

"polynomial if numbers in input are written in unary (not binary)"

Integer Knapsack

input:
- n objects $S = \{1, \ldots, n\}$
- s_i = size of object i (integer).
- v_i = value of object i.
- capacity C of knapsack (integer)

output:

- subset $K \subseteq S$ of objects that
  (a) fit in knapsack together (i.e., $\sum_{i \in K} s_i \leq C$)
  (b) maximize total value (i.e., $\sum_{i \in K} v_i$)

Find a subproblem:

- consider object $i \in S$.
- if i in knapsack:

  value of knapsack is v_i + optimal knapsack value on $S \setminus \{i\}$ with capacity $C - s_i$.
- if i not in knapsack:

  value of knapsack is optimal knapsack on $S \setminus \{i\}$ with capacity C.

Succinct description:

- remaining objects $\{j, \ldots, n\}$ represented by "j"
- remaining capacity represented by $D \in \{0, \ldots, C\}$.

Step I: identify subproblem in English

$\mathrm{OPT}(j, D)$

$=$ "value of optimal size D knapsack on $\{j, \ldots, n\}$"

Step II: write recurrence

$\mathrm{OPT}(j, D)$

$$= \max(\underbrace{v_j + \mathrm{OPT}(j+1, D - s_j)}_{\text{if } s_j \leq D}, \mathrm{OPT}(j + 1, D))$$

Step III: base case

$\mathrm{OPT}(n + 1, D) = 0$ (for all D)

Step IV: iterative DP

Algorithm: knapsack

1. $\forall D$, $\text{memo}[n + 1, D] = 0$.
2. for $i = n$ down to 1,

   for $D = C$ down to 0,

   (a) if i fits (i.e., $s_i \leq D$)

   $\text{memo}[j, D] = \max[\mathrm{OPT}(j + 1, D),$

   $v_j + \mathrm{OPT}(j + 1, D - s_j)]$

   (b) else,

   $\text{memo}[j, D] = \mathrm{OPT}(j + 1, D)$
3. return $\text{memo}[1, C]$

Correctness

induction

Runtime

$$\begin{aligned} T(n, C) &= O(\# \text{ of subprobs} \times \text{cost per subprob}) \\ &= O(nC). \end{aligned}$$

Note: Knapsack DP is <u>pseudo-polynomial time</u>.

Polynomial Time Approximation Scheme (PTAS)

"for any constant ϵ, get $(1+\epsilon)$-approximation algorithm in polynomial time."

Note: often pseudo-polynomial time alg can be converted into PTAS by rounding..

Knapsack PTAS

Goal: output $(1+\epsilon)$-approximation to optimal knapsack value.

Idea: round so that numbers are integers in range from 0 to poly(n).

Recall: for old knapsack dynamic program, need sizes to be integer, but approximation would allow for rounding values not sizes.

Approach:

1. write new dynamic program that is pseudo-polynomial in values not capacitiy. $O(n^2 v_{\max})$
2. divide values by $\epsilon v_{\max}/n$ and round up. (range from 0 to n/ϵ.)
3. solve dynamic program on rounded values.

Value-based Knapsack DP

Idea: instead of maximizing value, let's minimize size.

Part I: Subproblem

MinSize(i, V) = smallest total size of subset of $\{i, \ldots, n\}$ with total value at least V.

Part II: Recurrence

$$\begin{aligned}\text{MinSize}(i, V) &= \max\{s_i + \text{MinSize}(i+1, \max\{V - v_i, 0\}),\\ &\quad \text{MinSize}(i+1, V)\}\end{aligned}$$

Part III: Invocation

1. $V \leftarrow \sum_i v_i$
2. while MinSize$(1, V) > C$

   $V \leftarrow V - 1$
3. output V.

Part IV: Base case

$$\text{MinSize}(n+1, V) = \begin{cases} 0 & \text{if } V = 0 \\ \infty & \text{o.w.} \end{cases}$$

Theorem: ALG has pseudo-polynomial runtime $O(n^2 v_{\max})$ if v_is are integer.

Proof: table size $= n \times \sum_i v_i \leq n \times n v_{\max}$

Polynomial Time Approximation Scheme

Algorithm: Knapsack $(1+\epsilon)$-approx

1. round v_i up to multiple of $\epsilon v_{max}/n \rightarrow \tilde{v}_i$
2. divide $\tilde{v}_i$ by $\epsilon v_{max}/n \rightarrow \hat{v}_i$ (integer)
3. solve integral knapsack on $\hat{v}_1, \ldots, \hat{v}_n \rightarrow S$
4. output $\max(v_{\max}, \sum_{i \in S} v_i)$

Correctness

Lemma: ALG is optimal for $\hat{v}_i$s and $\tilde{v}_i$s.

Proof: via correctness of DP.

Lemma: ALG is polynomial in n (for const. ϵ)

Proof:

- $\hat{v}_{max} = v_{max} \times \frac{n}{\epsilon v_{max}} = n/\epsilon$
- runtime is $O(n^2 \hat{v}_{max}) = O(n^3/\epsilon) = O(n^3)$.

Lemma: ALG is $(1+\epsilon)$-approx for v_is.

Proof:

1. lower bound on OPT

$$\begin{aligned} OPT &= \sum_{i \in S^*} v_i && \text{(OPT's actual values)} \\ &\leq \sum_{i \in S^*} \tilde{v}_i && \text{(OPT's rounded values)} \\ &\leq \sum_{i \in S} \tilde{v}_i && \text{(ALG's rounded values)} \end{aligned}$$

   Last step by optimality of ALG on $\tilde{v}$s and $\hat{v}$s.

2. upper bound on algorithm

   - bound 1:
   $$ALG = \sum_{i \in S} v_i$$
   (ALGs's actual values)
   $$\begin{aligned} &= \sum_{i \in S} \tilde{v}_i - \sum_{i \in S} \underbrace{(\tilde{v}_i - v_i)}_{\leq \epsilon v_{max}/n} \\ &\geq \sum_{i \in S} \tilde{v}_i - n \times \epsilon v_{max}/n \\ &= \sum_{i \in S} \tilde{v}_i - \epsilon v_{max} \end{aligned}$$
   - bound 2: ALG $\geq v_{max}$.

3. combine:

$$\begin{aligned} ALG &\geq \underbrace{\sum_{i \in S} \tilde{v}_i}_{\geq OPT} - \epsilon \underbrace{v_{max}}_{\leq ALG} \\ &\geq OPT - \epsilon ALG \end{aligned}$$

   So $(1+\epsilon) ALG \geq OPT$.

QED

Complexity of Approximation

Def: APX = class of problems with constant approximations

Def: PTAS = class of problems with PTASs.

DRAW PICTURE of $P \leq PTAS \leq APX \leq NP$