Advanced Algorithms

Lecture 8: Greedy algorithms

Announcements

- HW 2 is out! Due next Friday — start early!
- HW 1 grades will be out by Monday

[TA Vivek].

Recap: dynamic programming

- Sequential decision making (eg. subset sum, paths in graphs, how much cake to eat, where-next in TSP tour, …)

  → Set of vertices "remaining"

- Some resource "depleting" (sub-problem defined by "amount remaining")

- **Key:** past decisions lead to some "state"; we can then solve sub-problem starting at the state (ignoring past)

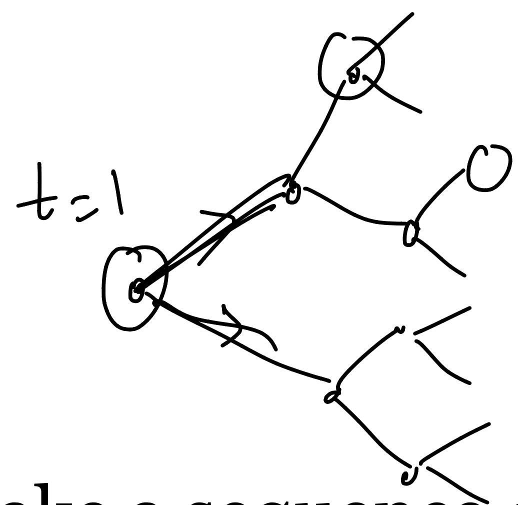
- One issue is memory

(Traveling Salesman → memory = $O(n \cdot 2^n)$.

$n \cdot 2^n$

Time/memory trade-offs.

DP template



- We need to make a sequence of choices

- Try all choices at time 1. For each one,
  cost = cost(choice 1) + cost("remaining" problem)

- Then pick the best value of choice 1

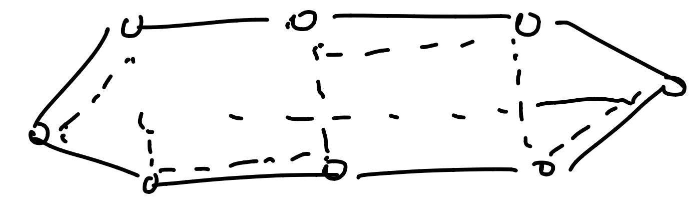
- <u>Key:</u> figure out how to define/parametrize the *remaining problem*

Greedy algorithms

Greedy paradigm

- Need to make a sequence of decisions
- "Myopic" choice — make *irrevocable* decision based on current state
- For each choice, associate *value* (only fn of present), make choice that has best value

Most "natural" algorithms



- E.g., traveling salesman problem (travel to closest unvisited node)
- Coin change: you are given coins of denominations 1c, 5c, 10c, 20c, 25c, 50c. Make change for say 75c using the fewest # of coins

40¢ → greedy uses 3 coins

↳ opt is to do 20+20¢.

- More complex problems… chess?

Moral: greedy algorithms "typically" aren't optimal, but give useful insights…

Scheduling jobs

Problem: suppose we have *n* jobs, with processing times $p_1, p_2, \ldots, p_n$. Find the best order of scheduling them so as to minimize the sum of "completion times"

p_1, p_2, p_3.

Completion times:

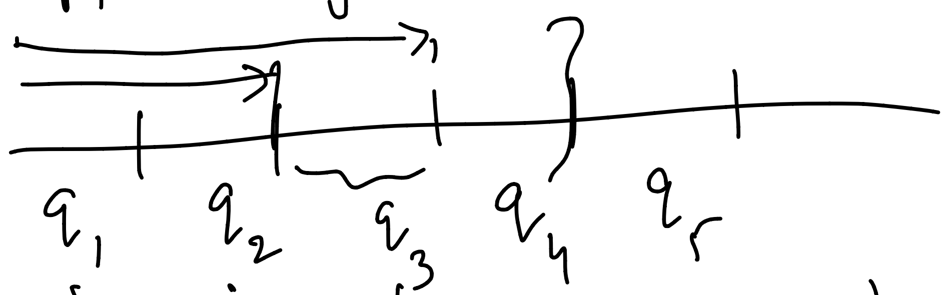
p_1 ; $p_1 + p_2$; $p_1 + p_2 + p_3$.

Key: correctness

Proof 1: closed form

permutation of $(1, 2, \ldots, n)$

Suppose jobs are done in some order: $(\sigma_1, \sigma_2, \ldots, \sigma_n)$



$$p_{\sigma_i} = q_i$$

q_1

$q_1 + q_2$

$\vdots$

$q_1 + q_2 + \ldots + q_n$

$n \cdot q_1 + (n-1) q_2 + \ldots + q_n \rightsquigarrow S$

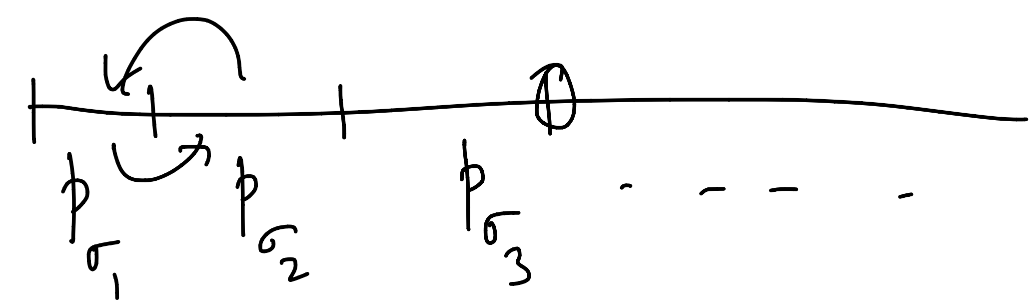
Claim: To minimize S, we should set $q_1 \leq q_2 \leq \ldots \leq q_n$.

$\therefore$ we must process jobs in increasing order of p_i.

Key: correctness

Proof 2: swapping — structure of opt solution $p_1, p_2, \ldots, p_n$.

Suppose $\sigma_1, \sigma_2, \ldots, \sigma_n$. is opt ordering.



Claim: $p_{\sigma_1} \leq p_{\sigma_2}$

Proof: Suppose if possible that $p_{\sigma_1} > p_{\sigma_2}$. Now swap σ_1 & σ_2

→ completion time of $\sigma_3, \sigma_4, \ldots$ will remain the same.

→ sum of completion times of σ_1 & σ_2 changes from $p_{\sigma_1} + (p_{\sigma_1} + p_{\sigma_2})$ to $p_{\sigma_2} + (p_{\sigma_2} + p_{\sigma_1})$

If $p_{\sigma_1} > p_{\sigma_2}$, then swapping gives a strictly better solution than optimum soln!

↓

contradiction.

Next claim: $p_{\sigma_2} \leq p_{\sigma_3}$

Exactly the same proof: swapping σ_2 & σ_3 does not change completion times of other jobs..

Can keep doing this → $p_{\sigma_1} \leq p_{\sigma_2} \leq \ldots$

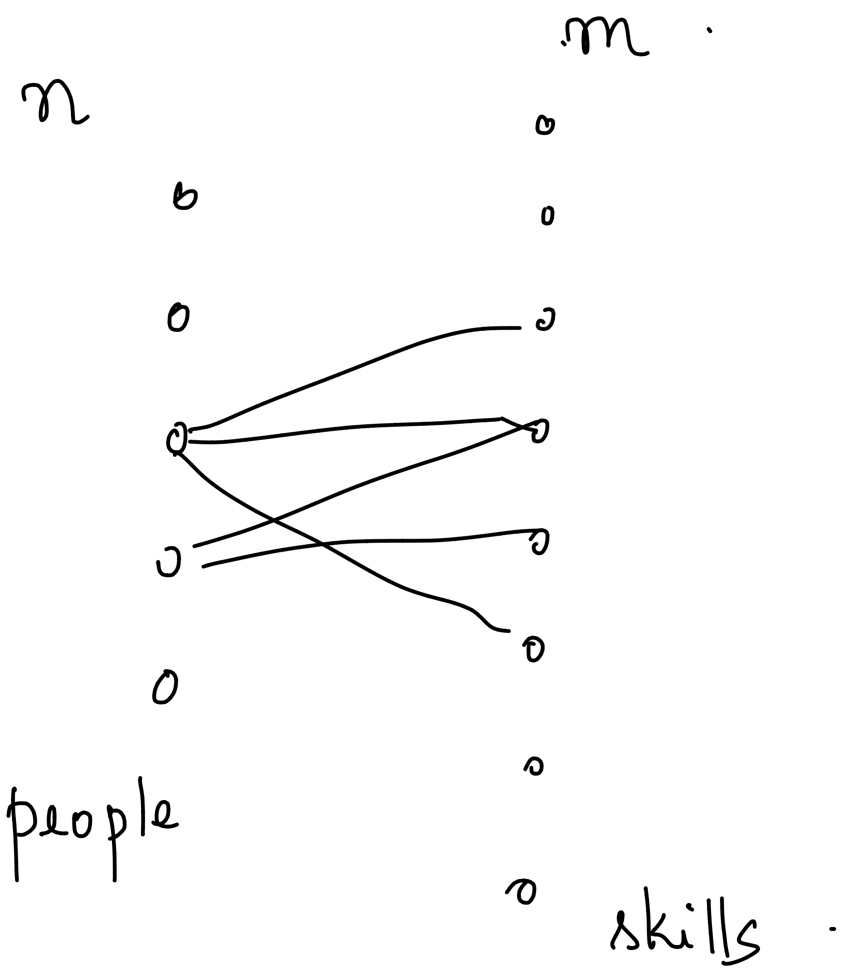
Key: correctness

Proof 3: induction — most common

Idea: prove by induction that first *k* choices are "correct"

Set cover

Problem: suppose we have *n* people, and *m* “desired skills”; each person has a subset of the skills. Pick the smallest subset of people such that every skill is *covered*



Another view: we are given

$S_1, S_2, \ldots, S_n \subseteq \{1, 2, \ldots, m\}$ → $[m]$.

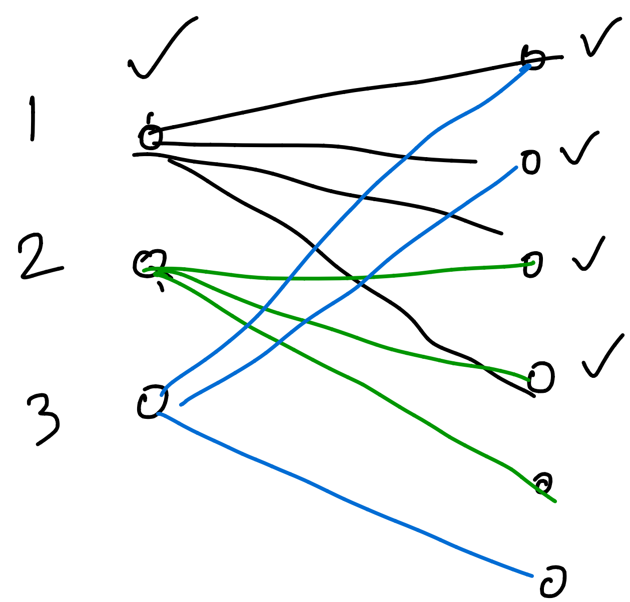
find some $i_1, i_2, \ldots, i_k$ s.t.

$S_{i_1} \cup S_{i_2} \cup \ldots \cup S_{i_k} = [m]$.

Greedy algorithm

- At each time, choose the person with the largest number of uncovered skills (breaking ties arbitrarily)

- Is this optimal?



greedy soln picks all 3 people.

opt solution only has {2, 3}.

Example

→ Greedy is not always optimal.

How bad can greedy be?

[Approx. algorithm].

Surprising theorem! suppose there is an optimum solution that uses k people. Then the greedy algorithm does not use more than $k \log n$.

$k = 5 \rightsquigarrow \leq 60$

$\ln n \approx 12$

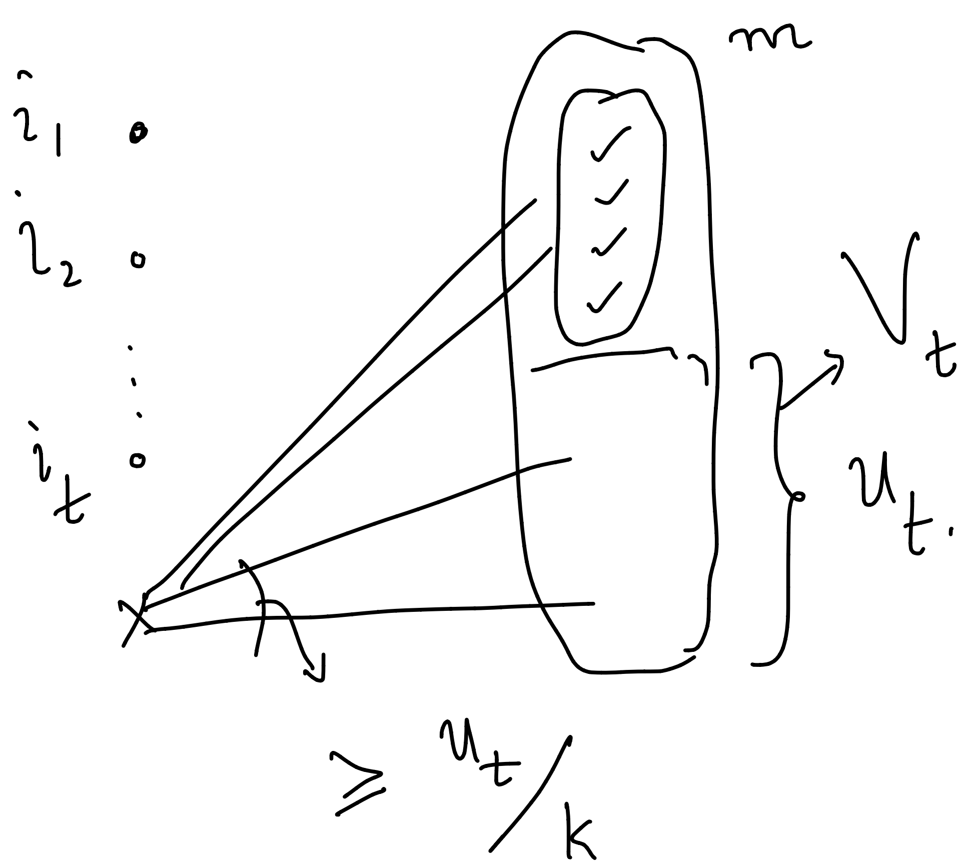
$n = 10000$

Challenge in the proof: showing that greedy alg. works well just by knowing the existence of a good soln.

Proof

Key idea: many skills are covered at each step!

Formally: Suppose we have u_t uncovered skills at iteration t. Then we claim that $u_{t+1} \leq u_t - \frac{u_t}{k}$.



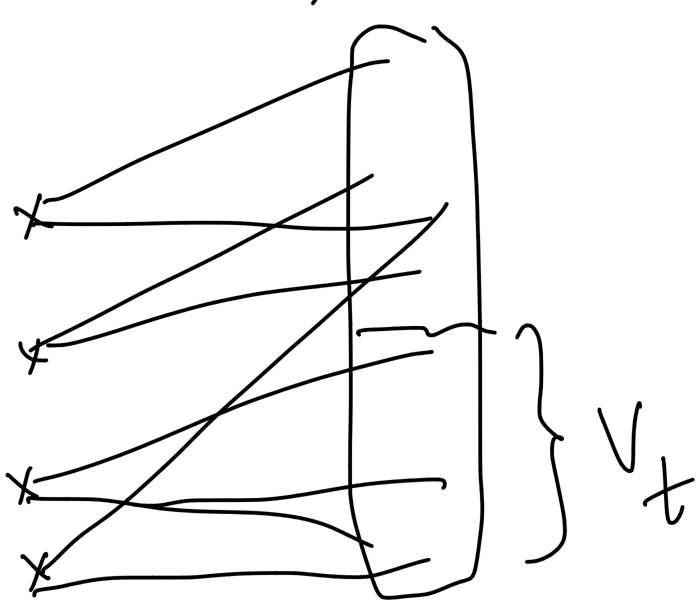
Suffices to show that there exists a person with at least $\frac{u_t}{k}$ uncovered skills.

Proof

Let $S_1, S_2, \ldots, S_k$ be the skill-sets of the optimal soln. $(j_1, j_2, \ldots, j_k)$

$$S_1 \cup S_2 \cup \ldots \cup S_k = [m] \supseteq V_t .$$

$$(S_1 \cap V_t) \cup (S_2 \cap V_t) \cup \ldots \cup (S_k \cap V_t) = V_t .$$



$\Downarrow$

one of $|S_r \cap V_t| \geq \frac{|V_t|}{k}$.

Greedy paradigm

- Need to make a sequence of decisions
- "Myopic" choice — make *irrevocable* decision based on current state
- For each choice, associate *value* (only fn of present), make choice that has best value