CS 25 - Algorithms

10/16/95

Last time (chap 16, 17)
- DP
- Greedy

Today (chap 17, 18)
- Greedy Algs.
- Amortized Analysis

Announcements
- X-hour

Javed Aslam

Greedy algorithms

- Usually used to solve optimization problems
  - e.g. Multiple matrix multiplication (try to minimize # operations)
    LCS - longest common subsequence

- Key idea: Greedy algorithms always make the choice that looks best at the moment.
  - don't look ahead
  - don't consider future ramifications of current decisions

  * makes a locally optimal vs. globally optimal decision

- In some sense, its the opposite of dynamic programming.

  Dynamic prog: Best solution to a problem is the best combination of the best solutions of some subproblems. Best solution of subproblem is best combination of best solution of some subsubproblems, etc.

  Thus, dynamic programming works bottom up first finding best solutions to smallest subproblems and then combining optimally to find best solutions to larger subproblems and so on until the best solution to the entire problem is obtained.

  Greedy alg: Work top down, simply making the best decision at the moment and going on from there.

- Because of greedy decisions, greedy algorithms don't always obtain optimal solutions. But they often do, and they're simpler than dynamic prog.

Elements of a greedy strategy

(How to tell if greedy strategy will work)

1) Greedy-choice property: A globally optimal solution can be arrived at by making locally optimal (greedy) choices.

   e.g. In the A.S.P., while not every solution consists of greedy choices, we showed that $\exists$ a solution composed of greedy choices.

2) Optimal substructure: An optimal solution to the problem contains optimal solutions to subproblems.

   e.g. In the A.S.P., if the first activity of an optimal solution is removed, then the remaining activities are an optimal solution to the activities compatible with the removed activity.

Greedy Methodology

1. Characterize the "greedy choice"
2. Prove the greedy algorithm correct by induction
   - Show that $\exists$ an opt. soln. which begins w/ greedy choice (base case)
   - Show that if $\exists$ an opt. soln. which begins w/ i greedy choices, then $\exists$ opt. soln. which begins w/ $i+1$ greedy choices. (inductive step)

— each piece of the above proof will likely be by contradiction.

Example

Fractional Knapsack

- n items
- $\forall i$, item i has weight w_i and value v_i
- knapsack holds W pounds

Goal: optimize value of items taken

Restriction: • total weight cannot exceed W

• can, however, take fractions of items (e.g. gold dust vs. gold ingots)

Algorithm

- Sort items by $q_i = v_i / w_i$ (quality)
- Take as much of best item as possible
- Repeat on items of successively lesser quality until knapsack full.
- R.T. $\Theta(n \lg n)$ to sort, $O(n)$ to pick items : $\Theta(n \lg n)$ total

Proof of correctness: (follows)

- In any solution to the fractional knapsack problem, consider the items taken in order of their quality $q_i = v_i / w_i$.

<u>Base Case</u>: $\exists$ an opt. soln. that begins with a greedy choice.

<u>Pf</u>

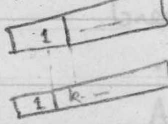
- W.l.o.g., assume items are numbered according to their order as sorted by quality. Thus, $q_1 \geq q_2 \geq q_3 \cdots$
- W.l.o.g., assume $q_i > q_{i+1}$ $\forall 1 \leq i < n$. (Otherwise, "combine" items with identical qualities.)
- Consider any opt. sol. S:
  - if S begins w/greedy choice (i.e., takes as much of Item 1 as possible), then done. ✓
  - if not, then S must take a positive amount of some other item $k > 1$

W.l.o.g., assume solution sorted by quality

$\Rightarrow \exists$ some weight $u > 0 \ni$ we can construct a new soln. S' which takes u less of Item k and u more of Item 1. ($u = \min\{$ amount of Item k taken, amount of Item 1 remaining $\}$)

Greedy:

Other:



$$\text{Value}(S') = \text{Value}(S) - u\frac{v_k}{w_k} + u\frac{v_1}{w_1}$$

$$= \text{Value}(S) + u(v_1/w_1 - v_k/w_k)$$

$$> \text{Value}(S) \quad \text{since } u > 0 \text{ and } q_1 > q_k$$

$\Rightarrow S$ is not optimal $\times$.

<u>Inductive Step</u>: If $\exists$ an opt. soln. which begins with $i-1$ greedy choices, then $\exists$ an opt. soln. which begins with i greedy choices.

<u>Pf</u> (Virtually identical to previous proof; leave as an exercise.)

Javed Aslam

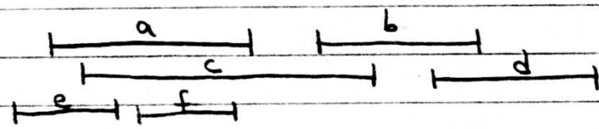
Another example

Activity Selection Problem

- Set S of n activities
  s_i = start time of activity i
  f_i = finish time of activity i

- find maximum subset A of compatible activities.

e.g. Given n activities that need ~~[illegible]~~ Spaulding auditorium, what is most activities you can schedule?



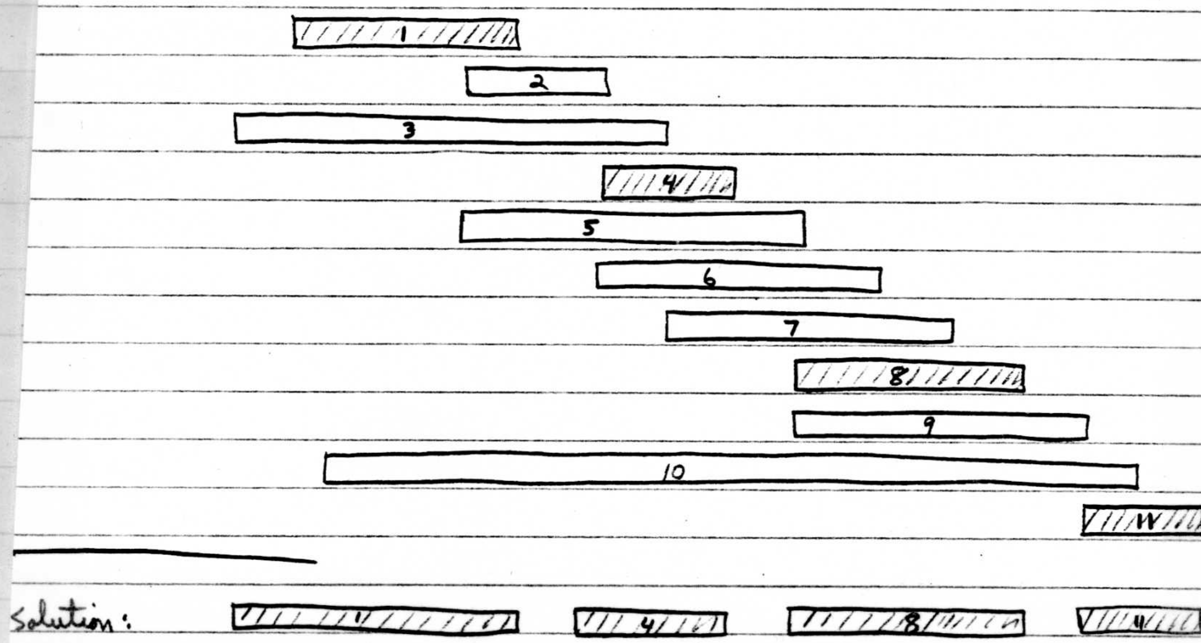
best: $\left\{ \begin{array}{l} e\ f\ b \\ e\ f\ d \end{array} \right.$

greedy decision : pick the compatible activity with the earliest finishing time so that you have maximum possible remaining time to schedule other activities.

```
Greedy-Activity-Selector (s, f)        % assume activities sorted by finishing time,
  n ← length [s]                       % e.g. f1 ≤ f2 ≤ ... ≤ fn
  A ← {1}
  j ← 1
  for i ← 2 to n
    do if si ≥ fj
         then A ← A ∪ {i}
              j ← i
  return A
```

Running time - $O(n)$ if already sorted, else $O(n \lg n)$ to sort.

Example (Shade activities in as they are selected)
(draw all 11 activities first ...)



Proof of correctness

- Let $S = \{1, 2, \ldots, n\}$ be a set of activities to schedule, ordered by finish times (i.e., $f_1 \le f_2 \le \ldots \le f_n$)

Greedy: $G = \{g_1, g_2, g_3, \ldots\}$
Opt: $A = \{a_1, a_2, a_3, \ldots\}$

Base Case: $\exists$ an optimal solution that begins with a greedy choice (i.e. activity g_1).

proof: Let $A \subseteq S$ be an optimal solution, and order the activities in A by their finish time. Suppose activity a_1 is the first activity in this order. $A = \{a_1, a_2, a_3, \ldots\}$

If $a_1 = g_1$, done ✓

If $a_1 \ne g_1$, then $(A - \{a_1\}) \cup \{g_1\}$ is an optimal solution (since $f_{g_1} \le f_{a_1}$)

Inductive Step: If $\exists$ an opt. soln. that begins with $i-1$ greedy choices, then $\exists$ an opt. soln. that begins w/ i g.c.

$A = \{g_1, g_2, g_3, \ldots, g_{i-1}, a_i, a_{i+1}, \ldots\}$

proof: Let $A \subseteq S$ be such an opt. soln. and let a_i be the i^{th} item picked (in sorted order)

If $a_i = g_i$, done ✓

If $a_i \ne g_i$, then $a_i > g_i$ and g_i not picked. (why?) (considering soln. in sorted order by f_i)

$\Rightarrow (A - \{a_i\}) \cup \{g_i\}$ is an opt. soln. (why?) ($f_{g_i} < f_{a_i}$ if $a_i > g_i$)

Greedy:
$G = \{g_1, g_2, \ldots, g_m\}$
(sorted order)