CS 25 - Algorithms

10/16/95

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

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

Announcements
- X-hour

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.

  There, 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: Works 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 ∃ 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 ∃ an opt. soln. which begins w/ greedy choice (base case)
   - Show that if ∃ an opt. soln. which begins w/ i greedy choices, then ∃ opt. soln. which begins w/ $i+1$ greedy choices. (induction step)

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