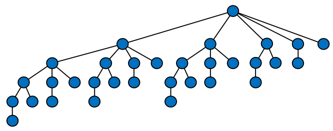
Lecture 17. Greedy Algorithms

CpSc 8400: Algorithms and Data Structures
Brian C. Dean



School of Computing
Clemson University
Spring, 2016

Discrete Optimization

- **Optimization problems** are everywhere, and a significant fraction of computer science (and related disciplines) is devoted to the pursuit of simple and efficient algorithms for these problems.
- We'll tend to focus on **discrete**, or **combinatorial** optimization problems, where we want to choose the "best" answer from a finite set of alternatives.
  - E.g., shortest paths, minimum spanning trees, minimum cuts, optimal schedules (permutations) of jobs.
  - These problems are "easy" in the sense that we could solve them by enumerating all possible solutions, but the # of such solutions is usually exponentially large, so the goal is to develop more efficient approaches.

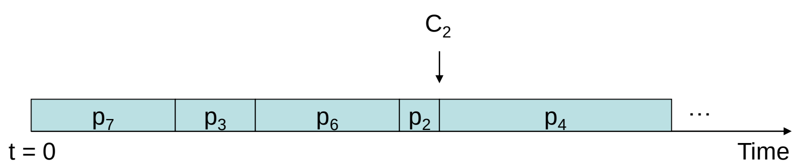
Greedy Algorithms

- "Incremental construction" approach: build a solution step by step, making near-sighted "greedy" decisions.
- We rarely go back and revise old decisions.
- Typically very simple to implement, and usually have very fast running times.
- **Caution!** Many students make the mistake of applying greedy methods when they ought not to be applied!
  - Careful analysis needed to convince oneself that a greedy solution is always optimal!

3

Example: Scheduling to Minimize Avg. Completion Time

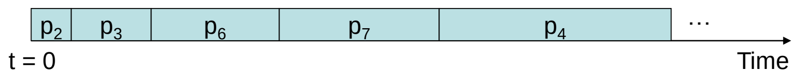
- **Input**: n jobs with processing times $p_1 \dots p_n$.
- **Goal**: Order the jobs so as to minimize their average completion time, $(^1/_n)\Sigma C_j$.
- Note that this is equivalent to minimizing ΣC_j.



4

Example: Scheduling to Minimize Avg. Completion Time

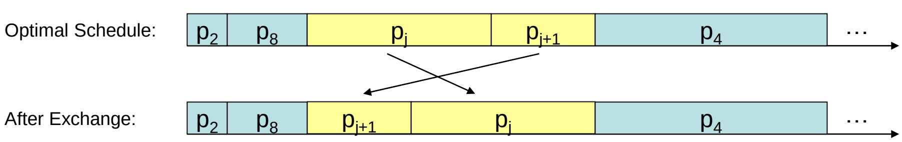
- **Greedy Algorithm**: order jobs in non-decreasing order of processing time.
- Since this just amounts to sorting, it takes only O(n log n) time.
- But why does it minimize ΣC_j? How would we prove this fact?



5

The Exchange Argument

- **Claim:** Our greedy algorithm always produces an optimal solution.
- **Proof**: By contradiction. Suppose it doesn't. Consider some instance where the greedy algorithm outputs a sub-optimal schedule.
  - Every optimal schedule for this instance contains an adjacent pair of jobs (j, j+1) satisfying $p_j > p_{j+1}$.
  - Suppose we were to swap these two jobs…



  - This would improve the objective value ΣC_j, contradicting the fact that our solution was optimal!

6

Example: Scheduling to Minimize Weighted Avg. Completion Time

- **Input**:
  - n jobs, with processing times $p_1 \ldots p_n$.
  - weights $w_1 \ldots w_n$.
- **Goal**: Order the jobs so as to minimize the average weighted completion time, $(^1/_n)\Sigma w_j C_j$. (note: equivalent to minimizing $\Sigma w_j C_j$)
- What is a simple greedy algorithm for this problem?

7

Example: Scheduling to Minimize Weighted Avg. Completion Time

- **Greedy algorithm:** Order jobs in non-increasing order of w_j / p_j (weight over processing time).
- O(n log n) running time.
- Again, we can prove this is optimal using an exchange argument:
  - Suppose that greedy is not always optimal.
  - Consider an instance where the greedy solution is ≠ an optimal solution, and look at an "optimal" solution.
  - Find some place where the "optimal" solution does something different from greedy, and make an exchange to bring the "optimal" solution closer to agreement with the greedy solution.
  - Prove that this doesn't make the objective value of the "optimal" solution any worse.
  - Therefore, by repeated exchanges we can transform the "optimal" solution into the greedy solution without harming its objective value!

8

Example: Scheduling to Minimize Weighted Avg. Completion Time

- Two similar approaches:
  - Show how you can repeatedly make exchanges in optimal solution (that don't hurt its objective value) so as to transform it into the greedy solution. This leads to a contradiction to the fact that our greedy solution was not optimal.
  - Start by considering, among all possible optimal solutions, one that agrees the most with our greedy solution. Now our very first exchange leads to a contradiction, since it allows us to find an optimal solution that agrees even more with our greedy solution.

9

Example: Activity Selection

- **Input**: n intervals $[a_1, b_1]$ .. $[a_n, b_n]$.
- **Goal**: select a set S of disjoint intervals, where |S| is maximized.
- Think of the intervals as the times at which different activities are scheduled. We'd like to attend the maximum # of activities.
- What is a simple greedy algorithm for this problem, and why is it optimal?

10

Example: The Quiz Problem

- **Input**:
  - n quiz questions
  - values $v_1 \ldots v_n$
  - probabilities of answering correctly $p_1 \ldots p_n$
- **Goal**: Find an ordering of the quiz problems that maximizes the expected total point value you obtain.
- You keep answering questions the order you choose until the first incorrect answer, at which point the quiz stops (and no value is received for the incorrect answer).

Example: The Quiz Problem

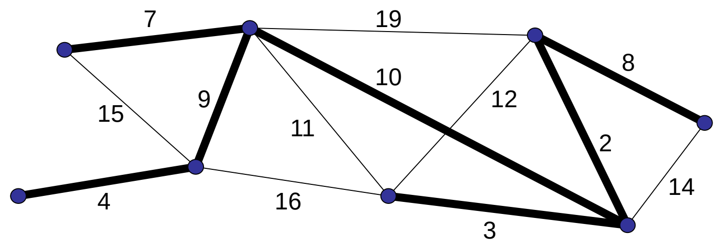
- Is there a greedy algorithm for the quiz problem?
- Let's try to use an exchange argument to "reverse-engineer" the right algorithm:
  - Suppose problems ordered 1, 2, …, n.
  - **E**[value] = $\Sigma_j\, v_j$ **Pr**[problems 1…j answered correctly]

    $= v_1p_1 + v_2p_1p_2 + v_3p_1p_2p_3 + \ldots$
  - Consider swapping two adjacent problems j and j+1.
  - Change in **E**[value] =

    $[p_1p_2\ldots p_{j-1}]\ [v_j\, p_j\, (1 - p_{j+1}) - v_{j+1}\, p_{j+1}\, (1 - p_j)]$.
  - This change ought to be ≤ 0, so:

    $v_j\, p_j / (1 - p_j) \geq v_{j+1}\, p_{j+1} / (1 - p_{j+1})$
- Greedy algorithm: non-increasing by $v_j\, p_j / (1 - p_j)$.

Example: The Minimum Spanning Tree Problem



- **Goal:** Find a minimum-cost subset of the edges in a graph that forms a tree, and that connects together all nodes.
- Very well-studied problem, and can be solved very efficiently.