CS 362, Lecture: Greedy Algorithms

Jared Saia
University of New Mexico

Outline

- Greedy Algorithm Intro
- Activity Selection
- Knapsack

Greedy Algorithms

"Greed is Good" - Michael Douglas in Wall Street

- A greedy algorithm always makes the choice that looks best at the moment
- Greedy algorithms do not always lead to optimal solutions, but for many problems they do
- In the next week, we will see several problems for which greedy algorithms produce optimal solutions including: activity selection, fractional knapsack.
- When we study graph theory, we will also see that greedy algorithms can work well for computing shortest paths and finding minimum spanning trees.

Activity Selection

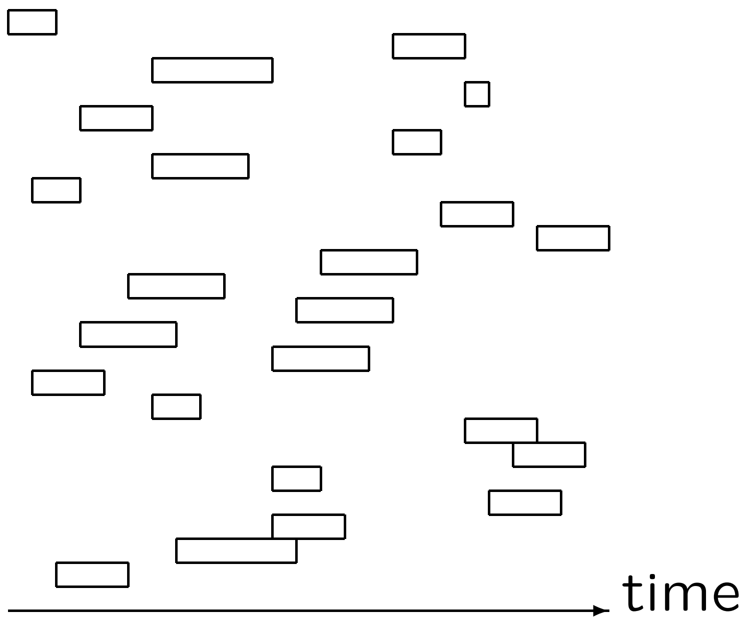
- You are given a list of programs to run on a single processor
- Each program has a start time and a finish time
- However the processor can only run one program at any given time, and there is no preemption (i.e. once a program is running, it must be completed)

Another Motivating Problem

- Suppose you are at a film fest, all movies look equally good, and you want to see as many complete movies as possible
- This problem is also exactly the same as the activity selection problem.

Example

Imagine you are given the following set of start and stop times for activities



Ideas

- There are many ways to optimally schedule these activities
- Brute Force: examine every possible subset of the activites and find the largest subset of non-overlapping activities
- Q: If there are n activities, how many subsets are there?
- The book also gives a DP solution to the problem

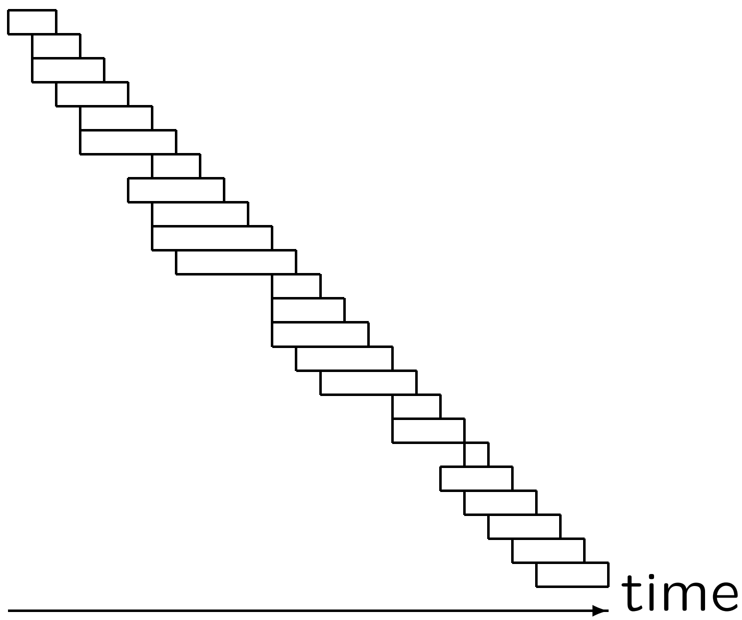
Greedy Activity Selector

1. Sort the activities by their finish times
2. Schedule the first activity in this list
3. Now go through the rest of the sorted list in order, scheduling activities whose start time is after (or the same as) the last scheduled activity

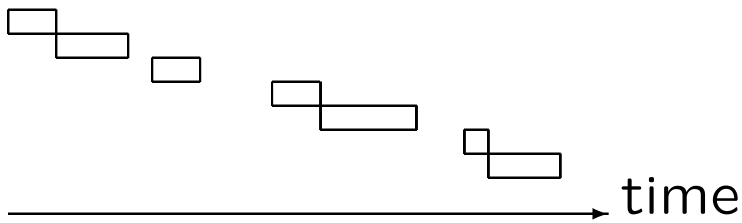
(note: code for this algorithm is in section 16.1)

Greedy Algorithm

Sorting the activities by their finish times



Greedy Scheduling of Activities



Analysis

- Let n be the total number of activities
- The algorithm first sorts the activities by finish time taking $O(n \log n)$
- Then the algorithm visits each activity exactly once, doing a constant amount of work each time. This takes $O(n)$
- Thus total time is $O(n \log n)$

Optimality

- The big question here is: Does the greedy algorithm give us an optimal solution???
- Surprisingly, the answer turns out to be yes
- We can prove this is true by something called an exchange argument.

Proof by Exchange Argument

- Let A be the set of activities selected by the greedy algorithm
- Consider *any* non-overlapping set of activities B
- We will show that $|A| \geq |B|$ by showing that we can replace each activity in B with a unique activity in A
- This will show that A has as many activities as any other valid schedule. Thus A is optimal.
- This type of proof is called an *Exchange Argument*

Proof Exchange Argument

- Let a_x be the *first* activity in A that is different than an activity in B
- Then $A = a_1, a_2, \ldots, a_x, a_{x+1}, \ldots$
  and $B = a_1, a_2, \ldots, b_x, b_{x+1}, \ldots$
- But since A was chosen by the greedy algorithm, a_x must have a finish time which is earlier than the finish time of b_x
- Thus $B' = a_1, a_2, \ldots, a_x, b_{x+1}, \ldots$ is also a valid schedule ($B' = B - \{b_x\} \cup \{a_x\}$)
- Continuing this process, we see that we can replace each activity in B with an activity in A. QED

What?

- We wanted to show that the schedule, A, chosen by greedy was optimal
- To do this, we showed that the number of activities in A was at least as large as the number of activities in any other non-overlapping set of activities
- To show this, we considered any arbitrary, non-overlapping set of activities, B. We showed that we could replace each activity in B with an activity in A

Greedy pattern

- **The problem has a solution that can be given some numerical value**. The "best" (optimal) solution has the highest/lowest value.
- **The solutions can be broken down into steps.** The steps have some order and at each step there is a choice that makes up the solution.
- **The choice is based on what's best at a given moment.** Need a criterion that will distinguish one choice from another.
- Finally, need to **prove** that the solution that you get by making these local choices is indeed optimal

Activity Selection Pattern

- The value of the solution is the number of non-overlapping activities. The best solution has the highest number.
- The sorting gives the order to the activities. Each step is examining the next activity in order and decide whether to include it.
- In each step, the greedy algorithm chooses the activity which extends the length of the schedule as little as possible

Knapsack Problem

- Those problems for which greedy algorithms can be used are a subset of those problems for which dynamic programming can be used
- So, it's easy to mistakenly generate a dynamic program for a problem for which a greedy algorithm suffices
- *Or* to try to use a greedy algorithm when, in fact, dynamic programming is required
- The knapsack problem illustrates this difference
- The 0-1 knapsack problem requires dynamic programming, whereas for the fractional knapsack problem, a greedy algorithm suffices

0-1 Knapsack

The problem:

- A thief robbing a store finds n items, the i-th item is worth v_i dollars and weighs w_i pounds, where w_i and v_i are integers
- The thief has a knapsack which can only hold W pounds for some integer W
- The thief's goal is to take as valuable a load as possible
- Which values should the thief take?

(This is called the 0-1 knapsack problem because each item is either taken or not taken, the thief can not take a fractional amount)

Fractional Knapsack

- In this variant of the problem, the thief can take fractions of items rather than the whole item
- An item in the 0-1 knapsack is like a gold ingot whereas an item in the fractional knapsack is like gold dust

Greedy

We can solve the fractional knapsack problem with a greedy algorithm:

1. Compute the value per pound (v_i/w_i) for each item
2. Sort the items by value per pound
3. The thief then follows the greedy strategy of always taking as much as possible of the item remaining which has highest value per pound.

Analysis

- If there are n items, this greedy algorithm takes $O(n \log n)$ time
- We'll show in the in-class exercise that it returns the correct solution
- Note however that the greedy algorithm does *not* work on the $0-1$ knapsack

Failure on 0-1 Knapsack

- Say the knapsack holds weight 5, and there are three items
- Let item 1 have weight 1 and value 3, let item 2 have weight 2 and value 5, let item 3 have weight 3 and value 6
- Then the value per pound of the items are: $3, 5/2, 2$ respectively
- The greedy algorithm will then choose item 1 and item 2, for a total value of 8
- However the optimal solution is to choose items 2 and 3, for a total value of 11

Optimality of Greedy on Fractional

- Greedy is not optimal on 0-1 knapsack, but it is optimal on fractional knapsack
- To show this, we can use a proof by contradiction

Proof

- Assume the objects are sorted in order of cost per pound. Let v_i be the value for item i and let w_i be its weight.
- Let x_i be the *fraction* of object i selected by greedy and let V be the total value obtained by greedy
- Consider some arbitrary solution, B, and let x'_i be the fraction of object i taken in B and let V' be the total value obtained by B
- We want to show that $V' \leq V$ or that $V - V' \geq 0$

Proof

- Let k be the smallest index with $x_k < 1$
- Note that for $i < k$, $x_i = 1$ and for $i > k$, $x_i = 0$
- You will show that for all i,

$$(x_i - x_i')\frac{v_i}{w_i} \geq (x_i - x_i')\frac{v_k}{w_k}$$

Proof

$$
\begin{aligned}
V - V' &= \sum_{i=1}^{n} x_i v_i - \sum_{i=1}^{n} x_i' v_i \\
&= \sum_{i=1}^{n} (x_i - x_i') v_i \\
&= \sum_{i=1}^{n} (x_i - x_i') w_i \left(\frac{v_i}{w_i}\right) \\
&\geq \sum_{i=1}^{n} (x_i - x_i') w_i \left(\frac{v_k}{w_k}\right) \\
&\geq \left(\frac{v_k}{w_k}\right) \sum_{i=1}^{n} (x_i - x_i') * w_i \\
&\geq 0
\end{aligned}
$$

Proof

- The last step follows because $\frac{v_k}{w_k}$ is positive and because:

$$\sum_{i=1}^{n}(x_i - x'_i) * w_i = \sum_{i=1}^{n} x_i w_i - \sum_{i=1}^{n} x'_i w_i \quad (1)$$
$$= W - W' \quad (2)$$
$$\geq 0. \quad (3)$$

- Where W is the total weight taken by greedy and W' is the total weight for the strategy B
- We know that $W \geq W'$

Exercise

Consider the inequality:

$$(x_i - x_i')\frac{v_i}{w_i} \geq (x_i - x_i')\frac{v_k}{w_k}$$

- Q1: Show this inequality is true for $i < k$
- Q2: Show it's true for $i = k$
- Q3: Show it's true for $i > k$