

Practice Problems on Greedy Algorithms

September 4, 2003

Below are a set of three practice problems on designing and proving the correctness of greedy algorithms. For those of you who feel like you need us to guide you through some additional problems (that you first try to solve on your own), these problems will serve that purpose. *If anyone would like a help session where I guide you through the process of solving these problems, please let me know.*

The first four problems have fairly straight forward solutions. So if you want to just be sure you understand how to develop a greedy algorithm and prove it is correct (or incorrect) then you should work these problems. The last three problems are harder in both the algorithm needed and in the proof of correctness.

To help you check your work the solutions to problems 1,2, and 5 are included here. Solutions for the rest are available upon request. You can also bring your solution/write-up to any of our office hours and we'll let you know if there were any problems. If you are having trouble and need a lot of guidance the TAs and I will use these practice problems to help you so that you can then do the HW problems on your own.

Practice Problems

1. Given a set $\{x_1 \leq x_2 \leq \dots \leq x_n\}$ of points on the real line, determine the smallest set of unit-length closed intervals (e.g. the interval $[1.25, 2.25]$ includes all x_i such that $1.25 \leq x_i \leq 2.25$) that contains all of the points.

Give the most efficient algorithm you can to solve this problem, prove it is correct and analyze the time complexity.

2. Suppose you were to drive from St. Louis to Denver along I-70. Your gas tank, when full, holds enough gas to travel m miles, and you have a map that gives distances between gas stations along the route. Let $d_1 < d_2 < \dots < d_n$ be the locations of all the gas stations along the route where d_i is the distance from St. Louis to the gas station. You can assume that the distance between neighboring gas stations is at most m miles.

Your goal is to make as few gas stops as possible along the way. Give the most efficient algorithm you can find to determine at which gas stations you should stop and prove that your strategy yields an optimal solution. Be sure to give the time complexity of your algorithm as a function of n .

3. Suppose we want to make change for n cents, using the least number of coins of denominations 1, 10, and 25 cents. Consider the following greedy strategy: suppose the amount left to change is m ; take the largest coin that is no more than m ; subtract this coin's value from m , and repeat.

Either give a counterexample, to prove that this algorithm can output a non-optimal solution, or prove that this algorithm always outputs an optimal solution.

4. You are given a sequence of n songs where the i th song is ℓ_i minutes long. You want to place all of the songs on an ordered series of CDs (e.g. CD 1, CD 2, CD 3, \dots , CD k) where each CD can hold m minutes. Furthermore,
- (1) The songs must be recorded in the given order, song 1, song 2, \dots , song n .
 - (2) All songs must be included.
 - (3) No song may be split across CDs.

Your goal is to determine how to place them on the CDs as to minimize the number of CDs needed. Give the most efficient algorithm you can to find an optimal solution for this problem, prove the algorithm is correct and analyze the time complexity.

5. You are given n events where each takes one unit of time. Event i will provide a profit of g_i dollars ($g_i > 0$) if started at or before time t_i where t_i is an arbitrary real number. (Note: If an event is not started by t_i then there is no benefit in scheduling it at all. All events can start as early as time 0.)

Given the most efficient algorithm you can to find a schedule that maximizes the profit.

6. Consider the problem of making change from n cents using the fewest coins when the available coins are quarters, dimes, nickels and pennies. Does the greedy strategy of outputting the largest coin that does not exceed the amount of change that must still be returned yield an optimal solution? Prove your answer is correct.

7. Consider the following scheduling problem. You are given n jobs. Job i is specified by an earliest start time s_i , and a processing time p_i . We consider a preemptive version of the problem where a job's execution can be suspended at any time and then completed later. For example if $n = 2$ and the input is $s_1 = 2$, $p_1 = 5$ and $s_2 = 0$, $p_2 = 3$, then a legal preemptive schedule is one in which job 2 runs from time 0 to 2 and is then suspended. Then job 1 runs from time 2 to 7 and finally, job 2 is completed from time 7 to 8. The goal is to output a schedule that minimizes $\sum_{j=1}^n C_j$ where C_j is the time when job j is completed. In the example schedule given above, $C_1 = 7$ and $C_2 = 8$.

Give the most efficient algorithm you can that computes an optimal preemptive schedule. Be sure to prove that your algorithm is correct and analyze the time complexity of your algorithm.

Solutions (solve the problems before reading this)

1. The greedy algorithm we use is to place the first interval at $[x_1, x_1 + 1]$, remove all points in $[x_1, x_1 + 1]$ and then repeat this process on the remaining points.

Clearly the above is an $O(n)$ algorithm. We now prove it is correct.

Greedy Choice Property: Let S be an optimal solution. Suppose S places its leftmost interval at $[x, x + 1]$. By definition of our greedy choice $x \leq x_1$ since it puts the first point as far right as possible while still covering x_1 . Let S' be the scheduled obtained by starting with S and replacing $[x, x + 1]$ by $[x_1, x_1 + 1]$. We now argue that all points contained in $[x, x + 1]$ are covered by $[x_1, x_1 + 1]$. The region covered by $[x, x + 1]$ which is not covered by $[x_1, x_1 + 1]$ is $[x, x_1)$ which is the points from x up until x_1 (but not including x_1). However, since x_1 is the leftmost point there are no points in this region. (There could be additional points covered by $[x + 1, x_1 + 1]$ that are not covered in $[x, x + 1]$ but that does not affect the validity of S'). Hence S' is a valid solution with the same number of points as S and hence S' is an optimal solution.

Optimal Substructure Property: Let P be the original problem with an optimal solution S . After including the interval $[x_1, x_1 + 1]$, the subproblem P' is to find an solution for covering the points to the right of $x_1 + 1$. Let S' be an optimal solution to P' . Since, $\text{cost}(S) = \text{cost}(S') + 1$, clearly an optimal solution to P includes within it an optimal solution to P' .

2. The greedy algorithm we use is to go as far as possible before stopping for gas. Let c_i be the city with distance d_i from St. Louis. Here is the pseudo-code.

```
S = ∅
last = 0
for i = 1 to n
    if (di - last) > m
        S = S ∪ {ci-1}
        last = ti-1
```

Clearly the above is an $O(n)$ algorithm. We now prove it is correct.

Greedy Choice Property: Let S be an optimal solution. Suppose that its sequence of stops is s_1, s_2, \dots, s_k where s_i is the stop corresponding to distance t_i . Suppose that g is the first stop made by the above greedy algorithm. We now show that there is an optimal solution with a first stop at g . If $s_1 = g$ then S is such a solution. Now suppose that $s_1 \neq g$. Since the greedy algorithm stops at the latest possible city then it follows that s_1 is before g . We now argue that $S' = \langle g, s_2, s_3, \dots, s_k \rangle$ is an optimal solution. First note that $|S'| = |S|$. Second, we argue that S' is legal (i.e. you never run out of gas). By definition of the greedy choice you can reach g . Finally, since S is optimal and the distance between g and s_2 is no more than the distance between s_1 and s_2 , there is enough gas to get from g to s_2 . The rest of S' is like S and thus legal.

Optimal Substructure Property: Let P be the original problem with an optimal solution S . Then after stopping at the station g at distance d_i the subproblem P' that remains is given by d_{i+1}, \dots, d_n (i.e. you start at the current city instead of St. Louis).

Let S' be an optimal solution to P' . Since, $\text{cost}(S) = \text{cost}(S') + 1$, clearly an optimal solution to P includes within it an optimal solution to P' .

5. We first argue that there always exists an optimal solution in which all of the events start at integral times. Take an optimal solution S — you can always have the first job in S start at time 0, the second start at time 1, and so on. Hence, in any optimal solution, event i will start at or before time $\lfloor t_i \rfloor$.

This observation leads to the following greedy algorithm. First, we sort the jobs according to $\lfloor t_i \rfloor$ (sorted from largest to smallest). Let time t be the current time being considered (where initially $t = \lfloor t_1 \rfloor$). All jobs i where $\lfloor t_i \rfloor = t$ are inserted into a priority queue with the profit g_i used as the key. An `extractMax` is performed to select the job to run at time t . Then t is decremented and the process is continued. Clearly the time complexity is $O(n \log n)$. The sort takes $O(n \log n)$ and there are at most n insert and `extractMax` operations performed on the priority queue, each which takes $O(\log n)$ time.

We now prove that this algorithm is correct by showing that the greedy choice and optimal substructure properties hold.

Greedy Choice Property: Consider an optimal solution S in which $x + 1$ events are scheduled at times $0, 1, \dots, x$. Let event k be the last job run in S . The greedy schedule will run event 1 last (at time $\lfloor t_1 \rfloor$). From the greedy choice property we know that $\lfloor t_1 \rfloor \geq \lfloor t_k \rfloor$. We consider the following cases:

Case 1: $\lfloor t_1 \rfloor = \lfloor t_k \rfloor$. By our greedy choice, we know that $g_1 \geq g_k$. If event 1 is not in S then we can just replace event k by event 1. The resulting solution S' is at least as good as S since $g_1 \geq g_k$. The other possibility is that event 1 is in S at an earlier time. Since $\lfloor t_1 \rfloor = \lfloor t_k \rfloor$, we can switch the times in which they run to create a schedule S' which has the same profit as S and is hence optimal.

Case 2: $\lfloor t_k \rfloor < \lfloor t_1 \rfloor$. In this case, S does not run any event at time $\lfloor t_1 \rfloor$ since job k was its last job. If event 1 is not in S , then we could add it to S contradicting the optimality of S . If event 1 is in S we can run it instead at time $\lfloor t_1 \rfloor$ creating a schedule S' that makes the greedy choice and has the same profit as S and is hence also optimal.

Optimal Substructure Property: Let P be the original problem of scheduling events $1, \dots, n$ with an optimal solution S . Given that event 1 is scheduled first we are left with the subproblem P' of scheduling events $2, \dots, n$. Let S' be an optimal solution to P' . Clearly $\text{profit}(S) = \text{profit}(S') + g_1$ and hence an optimal solution for P includes within it an optimal solution to P' .