Algorithms II

Kevin Li

April 2022

1 Greedy Algorithms

There are many types of algorithms, including Brute Force, Divide and Conquer, and Dynamic Programming. If you are interested in computer science, you should explore these different types on your own. Today, we will be focusing on Greedy Algorithms.
A greedy algorithm is one that choosing **the best choice at every possible option** regardless of the long term goal. In some cases, this may mean the smallest or largest number, quickest path, etc. Let's look at an example of a problem which we can use a greedy algorithm.

Example 1 *Suppose you are given a list of n integers (positive and negative)* $\{x_1, \ldots, x_n\}$*. To design an algorithm which selects the largest sum of at most 3 numbers, there are two different approaches we can take.*

1. Go through the list, select the largest, the second largest, and the third largest numbers which are nonnegative.

2. Sort the list in increasing order, and select the last number if it is positive, then the second, then the third.

3. Find all 0, 1, 2, 3 element subsets of the given list and select the one with the maximum sum.

Approach 1 would be a greedy approach, since at each step we select the best element for our problem. Approach 2 would also be considered greedy, since our first step was sorting the list which is a tool to help select the best element next. Approach 3 is not considered greedy, because we checked all possible options and selected the largest one.

Problem 1 *What is the runtime of Approach 1 in the above example? As we hinted to last week, there is a sorting algorithm that has runtime $O(log(n))$. Given this information, is approach 2 better or worse than Approach 1? What is the runtime of algorithm 3? How does this compare to the rest?*

Problem 2 *Let's say you are given a set of n lecture times at university,* $[x_1, y_1], [x_2, y_2], \ldots, [x_n, y_n]$ *where* x_i *represents the start time, and* y_i *represents the finish time. Try to design an algorithm which selects the* ***maximum number*** *of lectures we can hold with no overlaps (except at the endpoints. For example [1,2] and [2,4] are fine to select). Can you prove your solution is correct? Discuss with the group.*

Given that the section is about greedy algorithms, let's try to use a greedy approach to this problem. (Hint: to not get too stuck on the following, consider the order in which problems 3-6 are placed.)

Problem 3 *To be able to use a greedy algorithm, we need to decide what "best" means in this problem. If we choose "best" to mean* ***earliest start time*** *of the remaining intervals, would this greedy algorithm work? (At each step, choose the interval with the earliest start time). If yes, try to prove it. If not, give a counterexample.*

Problem 4 *If we choose "best" to mean* ***shortest interval*** *of the remaining intervals, would this greedy algorithm work? (At each step, choose the interval with the shortest duration). If yes, try to prove it. If not, give a counterexample.*

Problem 5 *If we choose "best" to mean* ***interval with the least number of intersections among remaining intervals****, would this greedy algorithm work? (At each step, choose the interval with the least number of conflicting intervals. Ex. Given [1,3], [2,4], [3,5], [7,8] the interval [2,4] has 2 conflicts). If yes, try to prove it. If not, give a counterexample.*

Problem 6 *If we choose "best" to mean* ***earliest end time*** *of the remaining intervals, would this greedy algorithm work? (At each step, choose the interval with the earliest end time). If yes, try to prove it. If not, give a counterexample.*

Problem 7 *Given coins $c_1, c_2, \ldots, c_n$ with values $v_1, v_2, \ldots, v_n$ respectively, find the minimum number of coins which sum to a given number N. Try to design a greedy algorithm which solves this problem*

Problem 8 *Does the greedy algorithm in the previous problem always work? If yes, prove it. If not, find a counterexample.*

Problem 9 *Given what we have seen, give some pros and cons to greedy algorithms. (For example, runtime, correctness, use of information, etc.)*