Week 9: Lab

Module 4: Techniques

Collaboration level 0 (no restrictions). Open notes.

As you solve each problem, write down your answers clearly in the space provided. To get solution to any of this lab's problems you will need to DM Laura in Slack and send along your attempted solution.

1. **Greedy pharmacist?** A pharmacist has W pills and n empty bottles. Bottle i can hold p_i pills and has an associated cost c_i. Given $W, p_1, p_2, ..., p_n$ and $c_1, c_2, ..., c_n$, you want to store all pills using a set of bottles in such a way that the total cost of the bottles is minimized. Note: If you use a bottle you have to pay for its cost no matter if you fill it to capacity or not.

   Someone proposes the following greedy strategy to solve the problem: Pick the bottle with the smallest cost-per-pill, and recurse on the remaining pills with the remaining bottles.

   Show that this greedy strategy is not correct by giving a counterexample.

2. **A different pharmacist problem**: Assume you have a number of pills W, and n bottles which can hold $\{p_1, p_2, ..., p_n\}$ pills, respectively. Assume W and all p_i are natural numbers. Describe a greedy algorithm, which, given W and $\{p_1, p_2, ..., p_n\}$, determines the fewest number of bottles needed to store the pills. (Note: all bottles are "equal" in terms of cost). Remember to argue that your algorithm is correct.

   The algorithm:

   Analysis:

   Correctness:

   Remember that in order to prove that a greedy algorithm is correct, it is sufficient to prove that **there exists an optimal solution that contains the first greedy choice**. In this case, you want to show that there axists an optimal solution which contains the first bottle chosen by your greedy algorithm. Use an exchange argument to show that an optimal solution can be changed into another optimal solution that contains the first bottle chosen by the greedy algorithm.

3. **Longest `true` interval**: Suppose we are given an array $A[1..n]$ of booleans. We want to find the longest interval $A[i..j]$ such that every element in the interval is true – in other words, $A[i], A[i+1], .., A[j]$ are all true.

   (a) Helen proposes using dynamic programming, with the following choice of subproblems: a subproblem is specified by parameters r, s such that $1 \leq r \leq s \leq n$. She suggests we define $F(r, s) =$ true if all of the entries $A[r], A[r+1], .., A[s]$ are true, and $F(r, s) =$ false otherwise.

   How many subproblems are there, in Helen's approach? Use big-O notation (but try to get a precise count as well).

   (b) Show a recursive formula for $F(r, s)$ that you could use to build a dynamic programming algorithm. Your recursive formula for $F(r, s)$ should be defined in terms of the solution(s) to smaller/easier subproblem(s).

   (c) Describe how you can find the longest true interval using $F(r, s)$. How long does that take? Assume of course that you compute $F(r, s)$ using dynamic programming.

(d) Michelle suggests a different approach for the same problem. In her approach, a subproblem is specified by the parameter x, where $1 \leq x \leq n$. Michelle suggests we define $G(x)$ to be the length of the longest suffix[1] of $A[1..x]$ that is all true. In other words, $G(x)$ is the largest integer l such that $A[x-l+1], A[x-l+2], .., A[x]$ are all true, or 0 if $A[x]$ is false.

If someone gave you a black-box that can compute $G(x)$ for any x, how would you use it to find the longest true interval in A?

(e) Show a recursive formula for $G(x)$. Your recursive formula for $G(x)$ should be defined in terms of the solution(s) to smaller/easier subproblem(s), and do not forget the base cases.

(f) Describe a dynamic programming algorithm to compute $G(n)$. How long does it take to find the longest interval?

[1]An array $B[1..m]$ is a suffix of an array $A[1..n]$ if $A[n-k] = B[m-k]$ for $0 \leq k < m$

4. **Art gallery guarding:** In the *art gallery guarding* problem we are given a line L that represents a long hallway in an art gallery. We are also given a set $X = \{x_0, x_1, x_2, ..., x_{n-1}\}$ of real numbers that specify the positions pf paintings in this hallway; assume that each painting is a point. Suppose that a single guard can protect all the paintings within a distance at most 1 of his or her position, on both sides.

   Design an algorithm for finding a placement of guards that uses the minimum number of guards to guard all the paintings in X. Briefly argue why your algorithm is correct and analyze its running time.

   The algorithm:

   Analysis:

   Why is this correct?:

5. Consider the activity selection problem: Given a set of n activities each with its start and finish times, $\{s_i, f_i\}$, find the maximum number of non-overlapping activities.

   Consider the following greedy algorithm: for each activity compute how many activities it overlaps; pick the activity with smallest number of overlaps; repeat.

   Prove that this algorithm is wrong by showing a counter-example where it fails to produce an optimal solution.

Additional problems–OPTIONAL

1. **The house robber problem[2]:** This problem will train you for a different type of robbing scenario. This time, instead of robbing Alibaba's cave with a knapsack, imagine you are a robber who has set his eye on a block of houses to rob. Each house i has a non-negative $v(i)$ worth of value inside that you can steal (imagine a stash of money in the house). However, due to the way security cameras work, you'll get caught if you rob two adjacent houses in the same night.

   Given a number of houses on the block and a list of non-negative integers representing the amount of money in each house, describe an algorithm to find the maximum amount of money you can rob in one night from the block (and the houses associated with this).

   ```
   Example 1: v = [1, 2, 3, 1]

   Max amount you can rob is 4 (rob house 1 and 3)

   Example 2:  v = [2, 7, 9, 3, 1]

   Max amount you can rob is 12 (rob house 2, 9 and 1)

   Example 3:  v = [6, 7, 1, 3, 8, 2, 4]

   Max amount you can rob is: 19 (rob houses  6, 1, 8 and 4)

   Example 4:  v = [5, 3, 4, 11, 2]

   Max amount you can rob is 16 (rob house 11 and 5)
   ```

[2]Frequently asked in interviews

2. Let's play a game where the input is a sequence $A[1], .., A[n]$ of numbers. At each turn, you can either take the first number in the sequence, add it to your score, and then delete that element from the sequence; or you can delete the first number in the sequence, take the second number in the sequence and add it to your score, and then finally delete the second number from the sequence. You keep playing until all of elements in the sequence have been deleted. You want to find a choice of moves that maximize your score.

   For instance, if the sequence is $5, -2, 4, -4, -1, 5$, your best choice of moves is "take first, take second, take second, take first", for a total score of $5 + 4 + -1 + 5 = 13$.

   Suppose we want to solve this problem using dynamic programming. Let $f(i)$ denote the best score attainable starting from the sequence $A[i], .., A[n]$. Write a recursive formula for $f(i)$.