Midterm
Colorado CSCI 5454

Alex Book

October 13, 2021

Problem 1

Parts of DP solution

Let C be the table holding answers for our subproblems, and let C and D both be indexed starting at 1. Let M_i be the set of locations within range of location i (including location i itself).

Subproblem definition

Let $C[i, j]$ represent the maximum number of locations that can be covered with up to j stations using locations $1, \ldots, i$, necessarily including a station at location i.

Recurrence

Base case(s):
Let $C[1, j] = 1$ for all $j \in \{1, \ldots, b\}$ (with only 1 location available, we can only cover 1 location, regardless of the number of stations being placed).

Let $C[i, 1] = N_i$ for all $i \in \{1, \ldots, n\}$, where N_i is the number of locations in range to the left of location i, including location i itself (with only placing 1 station at location i (and using only locations $1, \ldots, i$), the total coverage will be the amount of locations in range to the left of location i, plus location i itself.

Inductive case:

$$C[i, j] = C[i, 1] + \max_{k \in \{1, \ldots, i-1\}} \begin{cases} C[k, j-1] + (i - k) - C[i, 1] & \text{if } |M_k| - C[k, 1] + k \geq i - C[i, 1] \\ C[k, j-1] + (|M_k| - C[k, 1]) & \text{else} \end{cases}$$

Correctness of recurrence:
To begin with, we take the leftward coverage of location i (including i itself) and add the maximum of the bracketed statement, which can be interpreted as follows:

The "if" statement will evaluate to true if all locations between i and k are covered by having hydration stations at both i and k (this is because $|M_k| - C[k,1] + k$ represents the farthest rightward location from k that a hydration station at k would cover, while $i - C[i,1]$ represents the closest leftward location to i that a hydration station at i wouldn't cover). In this case, we add $(C[k, j-1] + (i-k) - C[i,1])$ to the leftward coverage of i. $C[k, j-1]$ is the optimal solution for the subproblem at k using one less hydration station, and $(i-k) - C[i,1]$ is the number of locations between i and k that are covered only by the station at location k. Adding all terms together grants us the answer to the previous subproblem plus the locations between k and i that are covered, including i itself (which all must be covered due to the if statement).

Otherwise, not all locations between i and k are covered by having hydration stations at both i and k. In this case, we add $(C[k, j-1] + (|M_k| - C[k,1]))$ to the leftward coverage of i. $C[k, j-1]$ is the optimal solution for the subproblem at k using one less hydration station, and $(|M_k| - C[k,1])$ is the rightward coverage of a station at location k, not including k itself. Adding all terms together grants us the answer to the previous subproblem plus the locations between k and i that are covered, including i itself.

Computing the final answer/Reconstruction

Reconstruction will give us the final answer for this problem. First, we find the "earliest" occurrence (minimum value of i) of the maximum value in the right-most column of C (where $j = b$). This index/location will be our starting point, let it be called q. From there, we look at all locations $\{q - C[q,1], \ldots, q\}$ (all locations that are within range to the left of location q, including q itself). We will take whichever of these locations i has the largest total range M_i. If there is a tie, we take the earliest/leftmost location (let's call this q'). Once q' has been found, we add it to our answer as a location in which a hydration station should be built.

We then move 1 column to the left and 1 row above the range of q (if the range of $q = 7$ is locations 4-7, we go to row 3) and repeat this process, looking at the "earliest" occurrence of the next maximum value (essentially shrinking the size of the effective table as we move our way back to column 1 and back through the locations).

Correctness

This process consists of finding the station location q that maximizes total coverage when only considering covering stations to the left of itself (which is what we are looking for in each subproblem). We then find the leftmost location q' in the range of q with the largest total coverage (to the left and right). This guarantees that, while we include location q being

covered (since q is in the range of q') we also may cover other locations that aren't necessarily in the range of q (since q' may be to the left of q, it may cover locations outside the range of q). This method gives us the most optimal coverage possible, covering the locations that are considered optimal in the lookup table by building hydration stations in range of those optimal locations, while also possibly covering more locations by possibly extending beyond the range of those optimal locations (since where we actually build the hydration stations can be left of the optimal locations themselves, as explained above).

Running time

The running time of this algorithm is dominated by the process of solving the subproblems. If we pre-compute all coverages M_i (so that they don't need to be computed in the loops), the running time is as follows:

There is $n \cdot b$ cells in C, and in each cell, you would loop through, at worst, $n - 1$ possible values for k (to find the maximum). With the aforementioned pre-computation, the total running time for this algorithm is $O(n^2 \cdot b)$

Problem 2

Consider Problem 3 from Homework 4, where we were asked for an algorithm to find the minimum vertex cut of a given graph with given edge and vertex capacities. In the algorithm used in the problem, we take a given graph $G = (V, E)$ and transform it such that each vertex v is split into two new vertices v_{in}, v_{out}. Any edges coming into v are redirected into v_{in}, any edges coming out of v are redirected out of v_{out}, and an "interior" edge is drawn between v_{in} and v_{out}. All "interior" edge capacities are set equal to their respective vertex capacities, and all other edge capacities are set to ∞. We call this new graph $G' = (V', E')$. We can then find the min edge cut on G' (which must only be through "interior" edges, as highlighted in HW4) and convert it into the min vertex cut. The correctness of this algorithm was proven in Homework 4 solutions. The running time is dominated by finding the maximum flow through G', which is that of running a Ford-Fulkerson framework algorithm: $O(|V'| \cdot |E'|^2)$.

This algorithm can be applied to the given problem to find a satisfying solution. We treat all vertices as having equal capacities, as this will preserve the original unweighted nature of the given graph. Once the algorithm is applied and the max flow is found, we can simply find the minimum vertex cut as explained above, which will be representative of finding the set R with minimal size, which is what is desired.

Problem 3

Part a

The optimal offline algorithm is as follows:
If the trip is 2 days or less, we do single-day rentals each day (cost is either 1 or 2).
If the trip is 3 days, we either do a 5-day rental or 3 1-day rentals (cost is 3 either way).
If the trip is 4 or 5 days, we do a 5-day rental (cost is 3).
If the trip is 6 days, we do a 5-day rental and a 1-day rental (cost is 4).
If the trip is 7 days or longer, we purchase (cost is 5).

Part b

Consider the following online algorithm:
On day 1 we do a single-day rental (added cost of 1, total cost 1).
On day 2 if the trip is continuing we do a 5-day rental (added cost of 3, total cost 4).
On day 7 if the trip is continuing we purchase (added cost of 5, total cost 9).

	OPT	ALG
Day 1	1	1
Day 2	2	4
Day 3	3	4
Day 4	3	4
Day 5	3	4
Day 6	4	4
Day 7	5	9
Day 8	5	9
Day 9	5	9
Day 10	5	9
⋮	⋮	⋮

At worst, the competitive ratio of this algorithm is 2. Therefore, we can say $\text{ALG} \leq 2 \cdot \text{OPT}$.