CMPSC 465 Spring 2022 | Data Structures & Algorithms Chunhao Wang and Mingfu Shao | Final Part 2

- Please log into https://psu.zoom.us/j/93567164986?pwd=L3lXWTVQL28vTExldkc4V2VYZitmQT09 using your PSU credential.
- If you have a question during the exam, you may ask the Instructor privately via Zoom chat.
- Instructor will announce any major corrections vocally over Zoom.
- Write your solutions by hand. Typed solutions will not be accepted. You may handwrite on a tablet as well.
- **At 8:40pm, you must put your pens down. You have until 8:55 to upload your solutions to Gradescope.**
- You must use a scanning app and not just take pictures.

1. (15+15=30 pts.) Let $G = (V, E)$ be a graph whose edge weights are positive. A minimum spanning tree of G is $T = (V, E')$; assume G and T are given as adjacency lists. Now, the weight of a particular edge $e^* \in E$ is modified from $w(e^*)$ to a new value $\hat{w}(e^*)$. We wish to quickly update the minimum spanning tree T to reflect this change without recomputing the entire tree from scratch. Consider the following two cases. In each case, give a description of an algorithm for updating T, a proof of correctness, and a runtime analysis for the algorithm. Note that for some of the cases these may be quite brief.

(a) $e^* \notin E'$ and $\hat{w}(e^*) > w(e^*)$

(b) $e^* \notin E'$ and $\hat{w}(e^*) < w(e^*)$

Solution

(a) **Main Idea:** Do nothing.

Correctness: T's weight does not increase, and any other spanning tree's weight either stays the same or increases, so T must still be an MST.

Runtime: Doing nothing takes $O(1)$ time.

(b) **Main Idea:** Add e^* to T. Use DFS to find the cycle that now exists in T. Remove the heaviest edge in the cycle from T.

Correctness: The heaviest edge in a cycle is safe to exclude from the MST (because if it is in the MST, you can remove it from the MST and add some other edge to the MST, and the MST's cost will not increase), and any edge not in an MST is the heaviest edge in some cycle (in particular, the cycle formed by adding it to the MST). For any edge not in T except for e^*, decreasing e^*'s weight does not change that it is the heaviest edge in the cycle, so it is safe to exclude from the MST. By adding e^* to T and then removing the heaviest edge in the cycle in T, we remove an edge that is also safe to exclude from the MST. Thus after this update, all edges outside of T are safe to exclude from the MST.

Runtime: This takes $O(|V|)$ time since T has $|V|$ edges after adding e^*, so the DFS runs in $O(|V|)$ time.

2. (5+2.5+10+5+5+2.5 pts.) You own a food company and want to distribute food trucks along East College Avenue in front of Penn State main campus. There are n blocks in total along East College Avenue and your marketing team reported that if you place a food truck at block i, the expected revenue is $r_i > 0$. The city of State College has a regulation saying that each company can place at most one food truck at each block. In addition, if you placed a food truck at block i, then you are not allowed to place another food truck two blocks to its East or two blocks to its West. The goal is to construct a dynamic programming algorithm to figure out the locations of your food trucks that maximize total revenue. Answer the following questions.

(a) Define the subproblem as a one-dimensional table.

(b) Based on the subproblem defined above, what is the maximum total revenue?

(c) Give a recurrence for computing $R(i)$.

(d) Give the base cases.

(e) Define another table that helps you recover the optimal locations.

(f) What is the running time of this dynamic programming algorithm? Justify briefly.

Solution

(a) Let $R(i)$ be the maximum total revenue if food trucks can only be placed among blocks $1, \ldots, i$.

(b) $R(n)$

(c)
$$R(i) = \max\{R(i-1), r_i + R(i-3)\}$$

(d) $R(1) = r_1$, $R(2) = \max\{r_1, r_2\}$, $R(3) = \max\{r_1, r_2, r_3\}$
Alternatively, $R(0) = 0, R(1) = r_1$, $R(2) = \max\{r_1, r_2\}$

(e)
$$\text{prev}(i) = \begin{cases} i-3, & \text{if } R(i) = r_i + R(i-3) \\ i-1, & \text{Otherwise} \end{cases}$$
By using this table, we can keep track of where the maximum is achieved for each entry of R. Then starting with $R(n)$ and back-tracking prev, we can find the optimal locations for place the food trucks.

(f) $O(n)$ as there are $O(n)$ entries to compute and computing each entry takes $O(1)$ time.

Alternatively, other subproblems work too; for example, one could add to the above subproblem the condition that a food truck must be placed at block i. But the recurrence and bases cases are different.