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

Final Part 2

Instructions:

- Please log into https://psu.zoom.us/j/95679319901?pwd=aW41MGpIMTdxUUVrSnlkMXMxblNGUT09 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 2:10pm, you must put your pens down. You have until 2:20 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) $e^* \in E'$ and $\hat{w}(e^*) < w(e^*)$

(b) $e^* \in E'$ and $\hat{w}(e^*) > w(e^*)$

Solution

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

Correctness: T's weight decreases by $w(e) - \hat{w}(e)$, and any other spanning tree's weight either stays the same or also decreases by this much, so T must still be an MST.

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

(b) **Main Idea:** Delete e from T. Now T has two components, A and B. Find the lightest edge with one endpoint in each of A and B, and add this edge to T.

Correctness: Every edge besides e in the MST is a lightest edge in some cut prior to changing e's weight, and increasing e's weight cannot affect this property. So all edges besides e are safe to keep in the MST. Then, whatever edge we add is also the lightest edge in the cut (A, B) and is thus also safe to include in the MST.

Runtime: This takes $O(|V| + |E|)$ time since it might be the case that almost all edges in the graph might one endpoint in both A and B and thus almost all edges will be looked at.

2. (5+5+10+5+5 pts.) **S**uppose you are collecting stamps for a reward. There are n possible stamps in total and the i-th stamps has value v_i and area a_i. You have a stamp book of total area A. You are qualified for the reward only if you have collected at least K different stamps, and amount of the reward is equal to the total value of your stamps. The goal is to construct a dynamic programming algorithm to compute the maximum value for the reward. Answer the following questions. (Note that you can have multiple copies of a stamp.)

(a) Complete this sentence: "Define the subproblem as follows. Let $S(a, k, i)$ be the maximum ..."

(b) Based on the subproblem defined above, what is the maximum value of reward.

(c) Give a recurrence for computing $S(a, k, i)$.

(d) Give the base cases.

(e) What is the running time of this dynamic programming algorithm?

Solution

(a) Let $S(a, k, i)$ be the maximum value for having at tleast k different stamps among $1, \ldots, i$, with total area no more than a.

(b) the maximum value of reward is obtained by

$$\max_{k:k\geq K} S(A, k, n).$$

(c) There are three cases regarding item i

  (a) The optimal solution doesn't use i, then the optimal value is the same as $S(a, k, i-1)$

  (b) The optimal solution uses i once, then the optimal value is the same as $S(a-a_i, k-1, i-1)+v_i$

  (c) The optimal solution uses i more than once. In this case we just take one copy away, and the value is $S(a-a_i, k, i)+v_i$, notice we don't decrease i, since we will (potentially) use i again, and we don't decrease k here, since we want to decrease it only once for all copies of stamp i, and we wait until the last time we use i to decrease k.

Thus we have the recurrence $S(a, k, i) = \max\{S(a, k, i-1), S(a-a_i, k-1, i-1)+v_i, S(a-a_i, k, i)+v_i\}$

(d) For the above recurrence, we need to deal with three base cases

  (a) $a < 0$: this case violates the weight constraint, we need $S(a, k, i) = -\infty$

  (b) $a \geq 0, i = 0, k = 0$: We have no more items to choose, so $S(a, k, i) = 0$

  (c) $a \geq 0, i = 0, k > 0$: this case violates the number of different items constraint, we need $S(a, k, i) = -\infty$.

Notice in the base cases, if you have 0 instead of $-\infty$ for those cases, you may end up with a solution not satisfying the constraints.

(e) The total running time is $O(AKn)$ as there are $O(AKn)$ entries to copute and computing each entry takes $O(1)$ time.

Alternatively, you can have the recurrence $S(a, k, i) = \max\{\max_{1\leq c\leq\lfloor a/a_i\rfloor}\{S(a-ca_i, k-1, i-1)+cv_i\}, S(a, k, i-1)\}$ if you want to make the decision of how many copies of stamp i to use all at once. You won't need the $a < 0$ base case then, since your recurrence will never incur $S(a, k, i)$ for $a < 0$.