Data Structures and Algorithms - Homework 4

HK Rho

February 21, 2020

Problem 1

I just wanted to mention that I got help for this problem from the GeeksforGeeks algorithm. I worked together with Audrey, SeungU, and Manu.

Algorithm

To find the maximum sum of happiness scores by finding the optimal food interval from the array of stations, I would basically take the divide-and-conquer approach. Looking the algorithm at a broad, general level, I would approach finding the max_happiness by dividing the array of stations by half, test for the maximum interval in three regions: sub-array at the left, sub-array at the right, and the sub-array crossing the division between the previous two sub-arrays. This process would done recursively.

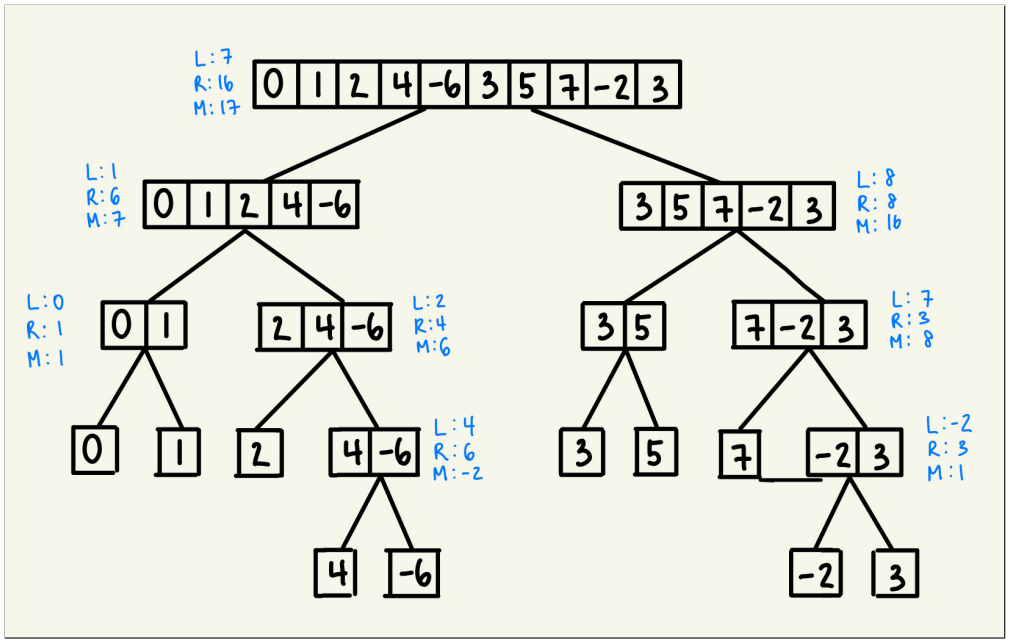

Figure 1: Visualization of the structure of the divide-and-conquer algorithm.

When using recursion to divide the array into two, the base case would be when we reach one happiness value, so we would return that value. As we traverse the tree upwards after reaching the base case, we pass in happiness scores from the child sub-arrays to the parent sub-arrays/array. For both the left part and the right part of the parent sub-array/array, it gets the maximum happiness values from its two child sub-arrays. As an example, in the case of analyzing the array shown in Figure 1:

$$[0, 1, 2, 4, -6]$$

The maximum happiness score of the left sub-array of the array above comes from $[0, 1]$, which is 1. The maximum happiness score of the right sub-array of the array above comes from $[2, 4, -6]$, which is 6. The maximum happiness score of the sub-array that crosses these two regions is found by adding the two happiness scores from each region, which is 7.Again, the maximum happiness score of this array must be a combination of both left and right sub-regions, since this sub-array crosses the left and the right sub-array. The maximum happiness score is propagated upward as this process is repeated.

Correctness

This algorithm has to be correct because it takes into account all of the possibilities where a potential array with the biggest happiness score would be: the left part of the divided sub-array, the right part of the divided sub-array, and the sub-array that crosses both parts. While going over all the possible places that the sub-array yielding the maximum happiness score might occur, we go over all of the happiness scores in the array so it is impossible to miss out on a sequence of happiness scores.

Run-time

The run-time for this algorithm is $O(nlogn)$. When trying to figure out where each n and $logn$ comes from, we learned that $logn$ comes from the height of this tree-like structure, especially with each array dividing up into two sub-arrays. For n, n comes from the idea that we are going through all of the happiness scores in the list.

Problem 2

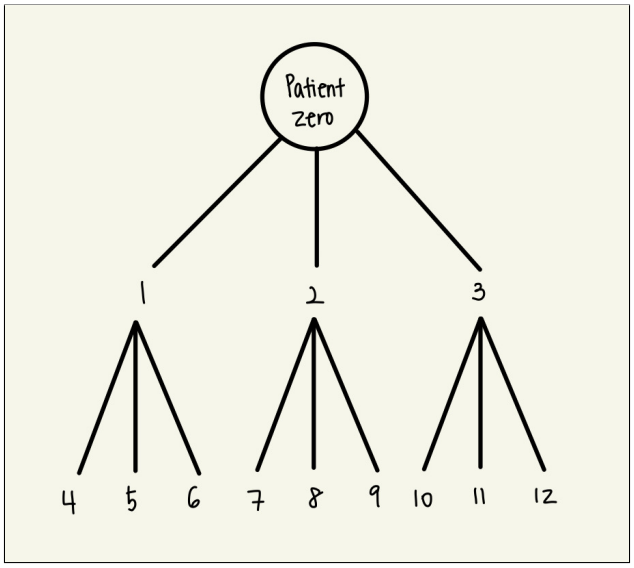

Figure 2: Sketch of overall structure of the connection between patient zero and all the groups of students in different classes that were in contact with patient zero. It also shows the contact in between different groups of students from different classes (e.g. 4, 5, 6) with the parent group of students that were in contact with all of them (e.g. 1)

Algorithm

The data structures that can be used for this algorithm are two queues, while one of the queues can be any linear data structure (e.g. array, stack, list). For this problem, A is a queue that holds all the students that store all the potentially affected students and B is another queue that holds all the students that are cured.

$$A = [\ ]\ and\ B = [\ ]$$

Since we are given who the patient zero is, we first add patient zero to A. After that, we find the number of classes that patient zero has attended, and add all the students in each class into A. For example, if patient zero attended DSA, Software Systems, and Material Science, we add all the students from DSA, Software Systems, and Material Science to A.

$$A = [students\ in\ DSA,\ students\ in\ Software\ Systems,\ students\ in\ Material\ Science]\ and\ B = [\ ]$$

Then after, we find the number of classes that each of the students from DSA, Software Systems, and Material Science were in, and add all the students in each of those classes into A, and continue this process. If a student is cured, patient zero for example, we remove patient zero from A and enqueue patient zero to B. We repeat the same general pattern of dequeueing a student that was cured from A and enqueueing them to B.

To prevent cases where we encounter duplicate students in A and B, we will loop through the students in A before adding a student to make sure that no same student is added. By doing this, we avoid putting same students in both A and B. There might be a concern regarding that the new duplicate student's contact not being added to A, but it is okay since the student was already added, the student's contacts were already added to A.

Correctness

This algorithm is correct because it adds all the potential candidates for being sick as we add all the students that were in contact with the potentially affected students by going through all their possible routines (which are classes in this case). This is kind of like tracking the route that a person who potentially has the Corona virus has gone through, and including people that visited that route as other potentially affected people. While doing so, we eliminate the students who have been added to A or B, so we do not get duplicates while continuously adding potentially affected students.

Problem 4

From running the part of the program that took the average of 100 runs of finding the maximum happiness score and the length of the sub-array that produced that value multiple times, I would like to state that the results were pretty constant–almost always falling in the range from $65 - 75$ for both values (maximum happiness, length of sub-array). An exemplary outcome was:

(Maximum happiness, Length of sub-array)= (69.89, 67.94)

Problem 5

From modifying Problem 4 by giving a probability distribution to the happiness values, the results I observed was that the maximum happiness scored to be extremely high of being higher than 220 for most of the times, and the length of the sub-array was also extremely high of falling on the higher end of the 90s. An exemplary outcome was:

(Maximum happiness, Length of sub-array)= (223.39737160418306, 99.0)

I believe that this outcome makes somewhat of a sense, because when I changed the probability distribution from 70%(happy) and 30%(less happy) to 50%(happy) and 50%(less happy) for the happiness scores, the following was the outcome I got:

(Maximum happiness, Length of sub-array)= (58.46248458776696, 54.59)

This makes sense as both the maximum happiness score and the length of that sub-array of the 50% 50% is smaller than that of 70% 30%, as it is less likely for higher happiness scores to occur.