DSA Homework 4

Katie Foster

Part 1

I used a recursive algorithm to find the interval with the maximum happiness score. The algorithm's base case is the length of the list being one. If the length of the list is one, the algorithm either returns the list twice if the list element is positive, otherwise returns two empty lists. The reason that the algorithm needs to return two lists is that the first list (l1) represents the highest happiness interval so far. The second list (l2) represents the maximum interval (in terms of happiness) that ends on the last element of the list that we are currently considering.

Besides the base cases, the other main work that the function has to do is to decide whether the current list or the best list that ends on the last element is the better option. First the function checks if the best list that ends on the last element of the current list in question has a value of greater than 0. If not, it sets that list to an empty list, and if it does, the last item is appended to list2. If the value of list2 is above zero, it is compared with the current best list overall, and whichever one of those lists has a higher value is set to the best overall list.

The runtime of this function is O(n) amortized. For the base case, the function takes O(1) time because it just uses some if statements and returns values. Each recursive function call only performs actions that take O(1) time (also if statements), and there are n-1 recursive calls, so the total runtime for the function is O(n) amortized.

Part 2

I would use a list of lists to store the names of the students in each class. Then I would use a dictionary to keep track of all the students and their states of being potentially infected. Each key in the dictionary would be a student in a class, and the value would be a boolean representing whether or not they are potentially infected. The order of the classes in the list would be based on the order in which they occur in the day, and it is assumed that no student is in more than one class happening at the same time.

- For class in class_list:
    - For student in class:
        - Look up student in dictionary
        - If student is infected:
            - Set all students in class to infected in dictionary

This method would be best for keeping track of how many people are infected throughout the day, and it controls for the fact that someone who gets infected in their afternoon class could not

infect someone in their morning class.

Another possible way would be to use depth first search to build up a stack using the information from a heap where each student is represented by a node, and each student/node ends up being connected to all the other students/nodes that it has class with. Each student/node keeps track of its neighbors (classmates) and it's value (the name of the student).

The first step is to find patient zero within the heap. Then patient zero, along with all connecting nodes (all students in the same class as patient zero) get pushed to the stack as well. Once a student is pushed to the stack, they are marked as infected, so they will not be pushed again. Keep pushing nodes until the stack is empty by calling the function recursively.

Part 4

The average length of the interval is about 42 elements, and the average value is 68. The average length seems longer than what I had expected, but in hindsight it makes sense given an even distribution of preferences. The average value makes sense given the length, because it means that on average all the values had a happiness of around 1.5. There would have been a few very high happiness items but they would be cancelled out by the negative items within the interval.

Part 5

I got an average interval length of 93 and an average total value of 222. This is a very big difference from the last experiment. In this case, you are more likely happy, which leads to much higher total values and interval length.