COMP 2080 Summer 2024 – Assignment 5

This assignment is due by 11:59pm on **Monday June 10**.

Your assignment will be submitted in Crowdmark and *not* in UM Learn. A Crowdmark invitation will be sent to your U of M email.

Proofs will be graded on both correctness and presentation. Clearly explain all of your steps.

The total number of points is 28.

Questions

Questions 1 – 4 are about the following problem, which we will call the **music festival problem**. Suppose that I am attending a music festival, which has several concerts scheduled. Each concert has a scheduled start time s and a scheduled end time f. Let $P = \{(s_1, f_1), (s_2, f_2), \ldots, (s_n, f_n)\}$ be the set of all concerts taking place at the music festival, where the i-th concert has start time s_i and end time f_i.

Now, my goal is to see as many concerts as possible (I am unconcerned with the quality of the concerts that I see) and, as I only have one body, I can only see one concert a time. How can I choose which concerts I attend so that I do not attend two concerts at the same time and so that the total number of concerts I attend is **maximized**?

1. For the music festival problem:

   (a) [**3 marks**] What are the items in a solution S? What is a **feasible** solution? What is an **optimal** solution?

   (b) [**2 marks**] Given a solution S, if we remove one item and apply it to the problem, what is the resulting subproblem?

2. (a) [**2 marks**] State the Optimal Substructure Property for the music festival problem. Your statement should not use the words “optimal”, “item”, or “subproblem” (though “maximal” and/or “minimal” are allowed).

   (b) [**4 marks**] Prove the Optimal Substructure Property that you stated in part (a).

3. [**2 marks**] Here is a potential greedy strategy for the music festival problem. I will repeatedly choose the **shortest** concert that does not overlap any of the concerts I have already decided to attend until there are no concerts left that do not overlap any of the concerts I have already decided to attend.

   Prove that this greedy algorithm does **not** satisfy the Greedy Choice property when applied to the music festival problem. That is, construct an instance of the music festival problem so that no optimal solution exists in which I attend the shortest concert. Explain why no such solution exists.

4. Here is a greedy algorithm that will solve the music festival problem. I will repeatedly choose the concert with the **earliest end time** that does not overlap with any of the concerts that I have already decided to attend.

   (a) [**2 marks**] State the Greedy Choice Property for this algorithm as applied to the music festival problem. Your answer should not include the phrases "optimal solution" or "greedy choice". (Again, "maximal" and/or "minimal" are allowed).

   (b) [**4 marks**] Prove the Greedy Choice Property that you stated in part (a).

Questions 5 – 7 are about the following problem, which we will call the **maximum sum sub-array problem**. Let A be an integer array of length n. The entries in A can be positive, negative, or zero. We want to find the indices a, b, where $0 \le a \le b < n$, so that the sum

$$A[a] + A[a+1] + \cdots + A[b]$$

is maximized. For simplicity, we will only find the maximal sum S, not the indices a and b.

We want to come up with a divide-and-conquer algorithm to solve the maximum sum sub-array problem. First, we decide that we will recursively solve the problem on the first half and the second half of the array. That means that we need to come up with the base case and the 'combine' step for our algorithm. Below is the start of a pseudocode implementation of our algorithm.

```
// Pre: A is an integer of length >= 1
int maxSubArray(A):
    n←A.length
    return maxSubArrayRecursive(A, 0, n-1)

int maxSubArrayRecursive(A, j, k):
    ...
```

5. First, we consider the base case of our algorithm.

   (a) [**1 mark**] When should the base case of our algorithm occur?

   (b) [**1 mark**] What should our algorithm return in the base case?

6. [**4 marks**] Write a pseudocode implementation of the **recursive case** of our algorithm. (You do not need to write the base case).

7. Now, let's analyze the running time of our algorithm.

   (a) [**2 marks**] Write a recurrence relation for the cost of our algorithm.

   (b) [**1 mark**] Use the Master Theorem to find a closed-form solution to your recurrence relation from part (a).