CS 577: Introduction to Algorithms — Midterm 1

Out: 03/02/21 **Due: 03/09/21**

Name: ____________________ Wisc ID: ____________________

Problem 1 [30 points]

Assume that you are given a fully parenthesized numerical expression with positive and negative numbers and like the following

$$\Big(\, [+3] + ([+1] + [-7]) \,\Big) + \Big(\, ([-8] + [-3]) + ([+1] - [+4]) \,\Big)$$

where the only possible operations between two numbers are addition $+$ and subtraction $-$. Notice that to avoid confusion we use brackets, e.g., $[-7]$ to denote a number together with its sign. Now consider the same expression with the operators $+$ and $-$ missing, i.e, replaced by $?$

$$\Big(\, [+3] \,?\, ([+1] \,?\, [-7]) \,\Big) \,?\, \Big(\, ([-8] \,?\, [-3]) \,?\, ([+1] \,?\, [+4]) \,\Big)$$

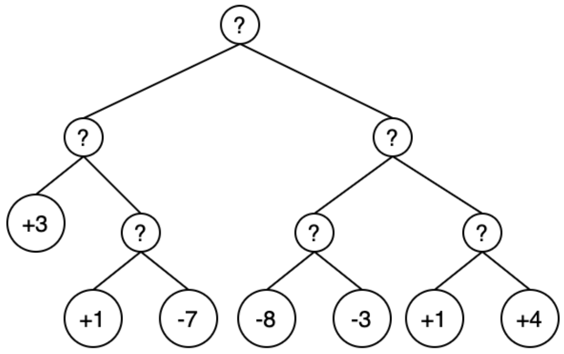
Your goal is figure out the **maximum and minimum** possible value of the expression when $?$ are replaced with either $+$ or $-$. For example, to maximize the above expression we have replace the $?$ as follows

$$\Big(\, [+3] + ([+1] - [-7]) \,\Big) - \Big(\, ([-8] + [-3]) - ([+1] + [+4]) \,\Big) = 27\,.$$

On the other hand to minimize it we have to replace them as

$$\Big(\, [+3] - ([+1] - [-7]) \,\Big) + \Big(\, ([-8] + [-3]) - ([+1] + [+4]) \,\Big) = -21\,.$$

In general you will be given an arithmetic expression with n numbers structured as a **binary tree** with root r where the leaves correspond to the numbers. For example, the input may be the following tree corresponding to the expression above:



In this case, the output of your algorithm should be $(27, -21)$.

(a) The following is an outline of a recursive algorithm for this problem. The main function call is to $\mathrm{FindMaxAndMin}(r)$ where r is the root of the tree. Fill in the missing parts.

```
Algorithm 1: FindMaxAndMin($v$)
/* Returns the maximum and minimum possible values of the expression. */
1 if /*Base Case*/
2 then
3     Return
4 else
5     /*Spawn recursive calls*/
6
7     Return (maxValue, minValue)
```

(b) State the running time of the algorithm in terms of the number n of numbers in the expression.

(c) Prove the correctness for your algorithm.

Problem 2 [30 points]

You are playing a videogame in which your character is attacked by a fleet of n enemy monsters. Each monster i inflicts damage d_i to your character every second it remains alive. To fight back, you start attacking monsters one by one until they are all defeated. It takes t_i seconds to defeat monster i and, during that time, both that monster as well as every other monster alive is constantly attacking you. Your goal is to find the order of attacks that minimizes the total damage you receive until all monsters are defeated.

Suppose there are $n = 2$ monsters with $(d_1, t_1) = (1, 2)$ and $(d_2, t_2) = (1, 1)$. It takes 3 seconds to defeat both monsters.

- Consider attacking monster 1 before monster 2. In the first 2 seconds until monster 1 is defeated, you receive 4 points of damage (2 from each monster). In the last second, only monster 2 attacks inflicting 1 additional damage point. In total you receive 5 points of damage.
- Consider attacking monster 2 before monster 1. In the first second until monster 2 is defeated, you receive 2 points of damage (1 from each monster). In the last 2 seconds, only monster 1 attacks inflicting 2 additional damage points. In total you receive 4 points of damage.

Thus, in this example, the optimal order is $[2, 1]$. For another example, consider again two monsters with $(d_1, t_1) = (3, 2)$ and $(d_2, t_2) = (1, 1)$. The optimal order in this case is $[1, 2]$ which leads to $2 \cdot 4 + 1 = 9$ points of damage.

(a) Design a greedy algorithm that finds the optimal order to attack the monsters.

(b) State the running time of your algorithm. No explanation is required.

(c) Prove the correctness of your algorithm.

Problem 3 [40 points]

Suppose you are given an **(undirected, unweighted)** graph $G = (V, E)$. The graph has colored edges, and so you are also given a partition of the edges $E = R \cup B$ into two sets, the red edges and the blue edges.

(a) Give a polynomial time algorithm to compute a spanning tree of G with as many red edges as possible. Prove your algorithm is correct and analyze its running time.

(b) Given two spanning trees T_1 and T_2 of G, one with r_1 red edges and one with r_2 red edges, where $r_1 \leq r_2$, it is always possible to construct a spanning tree with r red edges for any $r \in [r_1, r_2]$. Give a polynomial time algorithm that takes as input the trees T_1 and T_2 and the parameter $r \in [r_1, r_2]$ and constructs such a spanning tree T with r edges. Prove the correctness of your algorithm.
Hint: Try constructing a sequence of trees leading from T_1 to T_2 where consecutive trees differ by one edge.

(c) Suppose $n = |V|$ is odd, meaning that any spanning tree of G will have an even number of edges. Describe how to use the algorithms you developed in the previous parts to efficiently compute a spanning tree of G with an equal number of red and blue edges if one exists.
You do not need to prove correctness of your algorithm or analyze its runtime.