Worth: 7.5% (*best four of the five assignments*) **Due:** *before* 6:00pm on Wed 23 Jan

Required filename for MarkUs submission: `a1.pdf`

Remember to write the *full name* and *MarkUs username* of *every* group member (up to three) prominently on your submission.

---

*Please read and understand the policy on Collaboration given on the Course Information Sheet. Then, to protect yourself, list on the front of your submission **every** source of information you used to complete this homework (other than your own lecture and tutorial notes). For example, indicate clearly the **name** of every student from another group with whom you had discussions, the **title and sections** of every textbook you consulted (including the course textbook), the **source** of every web document you used (including documents from the course webpage), etc.*

*For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks **will** be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions.*

---

1. Suppose the Canadian Armed Forces is conducting a training exercise for n soldiers in an elite unit. Each soldier must first climb up an down a 10 meter rope and then run a kilometer on a track with n lanes. There is only one rope (due to budget cuts) so only one soldier can be on the rope at any time, whereas it does not matter how many soldiers are on the track. Suppose we know the time $c_i > 0$ for soldier i to climb the rope (c_i is the total time to climb both up and down) and the time $r_i > 0$ for soldier i to complete a kilometer run. The goal is to minimize the overall completion time (the moment when all soldiers have completed the exercise, assuming that the exercise begins at time 0).
   (a) Suppose we perform a greedy algorithm by sorting the order in which the soldiers will climb the rope so that $r_i + c_i \geqslant r_{i+1} + c_{i+1}$. Show that this algorithm will not always produce an optimal schedule.
   (b) Design a greedy scheduling algorithm so as to produce an optimal schedule for the order in which the soldiers will climb the rope.
   (c) Prove that your scheduling algorithm is optimal by using an exchange argument.

2. Give an efficient algorithm that takes the following inputs:
   - a positive integer $n \in \mathbb{Z}^+$;
   - a sequence with n integers, $a_1, a_2, \ldots, a_n$ (note that each a_i can be either *positive* or *negative*);

   and that outputs two integers j and k such that $a_j, \ldots, a_k$ is the *maximum subsequence* of $a_1, \ldots, a_n$. Specifically, j and k satisfy:
   - $1 \leqslant j \leqslant k \leqslant n$;
   - for any j' and k' such that $1 \leqslant j' \leqslant k' \leqslant n$, $\sum_{i=j'}^{k'} a_i \leqslant \sum_{i=j}^{k} a_i$.

   For full marks, your algorithm must have the time complexity $\mathcal{O}(n)$. Write a detailed proof that your algorithm is correct and justify your algorithm's time complexity. (Note that this argument of correctness will be worth *significantly more* than the algorithm itself.)

3. Consider the following inputs:

   - $G = (V, E)$, a connected undirected graph,
   - $w : E \to \mathbb{Z}^+$, a weight function for the edges of G.

   Also assume the minimum spanning tree (MST) of G, T^*, is unique, i.e., there is no other spanning tree has the same total weight as T^*. Design an algorithm that outputs a *second-minimum* spanning tree for G, i.e., a spanning tree $\hat{T}$ of G such that:

   - $w(\hat{T}) > w(T^*)$—remember that T^* is the unique MST of G,
   - $w(\hat{T}) \leqslant w(T)$ for all spanning trees T of G that are **not** T^*

   (where $w(T)$ is the sum of the weights of the edges of T, for any spanning tree T). If there is no spanning tree $\hat{T}$ that satisfies these conditions, your algorithm should output the special value nil.

   For full marks, your algorithm must have asymptotic time complexity no worse than computing a MST for G from scratch. Prove that your algorithm is correct and justify your algorithm's time complexity. (Note that this argument of correctness will be worth *more* than the algorithm itself).

   Hint: One possible efficient algorithm is to compute the minimum spanning tree T^* first and then modify T^* to obtain the second minimum spanning tree. In this case, simply make a call to the appropriate algorithm from class and please **do not** repeat its proof of correctness in your solution: you can just *use* the fact that the algorithm is known to be correct.

   Note: Your algorithm is likely to be different from a pure Greedy algorithm. Keep this in mind when writing your proof!