Computer Science C73
Scarborough Campus

September 13, 2023
University of Toronto

Homework Assignment #2
Due: September 20, 2023, by 11:59 pm

• ***You must submit your assignment through the Crowdmark system.*** *You will receive by email an invitation through which you can submit your work in the form of separate PDF docucments with your answers to each question of the assignment. To work with a partner, you and your partner must form a group on Crowdmark. Crowdmark does not enforce a limit on the size of groups.* ***The course policy that limits the size of each group to at most two remains in effect:*** *submissions by groups of more than two persons will not be graded.*
• *It is your responsibility to ensure that the PDF files you submit are legible. To this end, I encourage you to learn and use the LaTex typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You are not required to produce the PDF files you submit using LaTex; you may produce it any way you wish, as long as the resulting document is legible.*
• *By virtue of submitting this assignment you (and your partner, if you have one) acknowledge that you are aware of the policy on homework collaboration for this course.*[a]
• *For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbooks, by referring to it.*
• *Unless we explicitly state otherwise, you may describe algorithms in high-level pseudocode or point-form English, whichever leads to a clearer and simpler description. Do not provide executable code.*
• *Unless we explicitly state otherwise, you should justify your answers. Your paper will be graded based on* ***the correctness and efficiency of your answers, and the clarity, precision, and conciseness of your presentation****.*

[a] "In each homework assignment you may collaborate with at most one other student who is currently taking CSCC73. If you collaborate with another student on an assignment, you and your partner must submit only one copy of your solution, with both of your names. The solution will be graded in the usual way and both partners will receive the same mark. Collaboration involving more than two students is not allowed. **For help with your homework you may consult only the instructor, TAs, your homework partner (if you have one), your textbook, and your class notes. You may not consult any other source.**"

Question 1. (20 marks) The optimal length of a paddle for a canoeist is his/her shoulder height. We have n canoeists with shoulder heights $h_1, h_2, \ldots, h_n$, and n paddles of length $\ell_1, \ell_2, \ldots, \ell_n$. We must assign exactly one paddle to each canoeist (and, of course, different paddles to different canoeists). Such an assignment can be conveniently represented by a permutation π of $1, 2 \ldots, n$: canoeist i is assigned paddle $\pi(i)$. The ***penalty*** of such an assignment for canoeist i is $|h_i - \ell_{\pi(i)}|$ — i.e., how far off the optimal is the length of the paddle assigned to the canoeist. The ***average penalty*** of such an assignment is $\frac{1}{n}\sum_1^n |h_i - \ell_{\pi(i)}|$. We wish to find an assignment that minimises the average penalty.

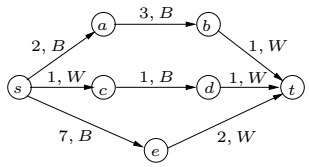
For each of the following two greedy strategies, either show that it determines a desired assignment or provide a counterexample.

a. Best-match first strategy: Find a pair (i, j) that minimizes $|h_i - \ell_j|$ (i.e., a canoeist-paddle pair of minimum penalty). Assign paddle j to canoeist i (i.e., $\pi(i) = j$) and apply the same strategy to the remaining canoeists and paddles, until all canoeists are assigned paddles.

b. Increasing height strategy: Sort the canoeists in increasing shoulder height and the paddles in increasing length, and assign to each canoeist the corresponding paddle. In other words, the canoeist of minimum shoulder height gets the shortest paddle, the canoeist of second minimum shoulder height gets the second shortest paddle, and so on.

Question 2. (15 marks) We are given a directed graph $G = (V, E)$ and two distinguished nodes $s, t \in V$. Each edge e has a positive weight $\mathbf{wt}(e)$ and a colour $\mathbf{colour}(e) \in \{\text{black}, \text{white}\}$. The ***coloured-weight*** of a path in G is the sum of the weights of the edges plus the fixed amount 5 ***each time*** the path changes colour — i.e., it leaves a node with an edge of the opposite colour than the edge with which it enters the node.

For example, the colour-length of the three $s \to t$ paths in the graph shown to the right (where the label 'x, y' of an edge indicates its weight x and color y) are: $2 + 3 + 5 + 1 = 11$ (for s, a, b, t), $1 + 5 + 1 + 5 + 1 = 13$ (for s, c, d, t), and $7 + 5 + 2 = 14$ (for s, e, t).



Given as input the graph G, the nodes s, t, and the edge weight and colour functions **wt** and **colour**, we want to find the minimum coloured-weight of an $s \to t$ path in G. Give an efficient algorithm to solve this problem ***by reducing it to a shortest-path problem***.[1] Justify the correctness of your algorithm and analyze its running time.

[1] Problem A is reduced to problem B if there is an algorithm that solves A using ***as a black box*** a solution to B. By "using as a black box" we mean that the reduction algorithm that solves A can make calls to a subroutine about which we know ***only*** that it solves B — not ***how*** it does so. (In particular, we cannot "look inside" the black box that solves B and change the way it works.) Thus, if A reduces to B and we know of an actual algorithm b that solves B, we can obtain a solution to A by plugging b into the black box used by the reduction. Reductions are a very powerful algorithm design tool, as they allow us to solve new problems by leveraging algorithms that have already been devised to solve other problems. Certain problems play a particularly important role in computer science precisely because many other problems reduce to them. Shortest paths is such an example. Later in the course we will encounter two other such problems: maximum flow and linear programming. When you take CSCC63 you will learn about a different use of reductions, namely as a tool for proving that certain problems are impossible to solve algorithmically, or are unlikely to solve efficiently.

When you analyze the running time of a reduction algorithm you must take into account the cost of the calls to the black box. So, for example, if you know that B can be solved by an algorithm whose running time is $O\big(f(n)\big)$, you must charge $O\big(f(k)\big)$ for each use of the black box on an input of size k by the reduction algorithm.