Computer Science C73 September 14, 2022
Scarborough Campus University of Toronto

Homework Assignment #2
Due: September 21, 2022, 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.*
• *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. (15 marks) A summer camp has n campers, labeled $1, 2, \ldots, n$, and wants to organize a canoe trip. Camper i weighs w_i. Each canoe must have exactly two campers, whose combined weight must not exceed C; naturally, no camper can be in two canoes! Our job is to pair-up as many campers as possible subject to the weight constraint.

More precisely, a set of (unordered) pairs of campers is ***feasible*** if (a) no camper belongs to two different pairs in the set; and (b) for any pair $\{i, j\}$ in the set, $w_i + w_j \leq C$. A feasible set of pairs of campers is ***optimal*** if there is no feasible set with greater cardinality. Given $w_1, \ldots, w_n$ and C, we want to find an optimal set of pairs of campers.

a. Prove that the following greedy algorithm does not necessarily find an optimal set: If $n < 2$, return the empty set. Otherwise, let p and q be two lightest campers (ties broken arbitrarily). If $w_p + w_q \leq C$, return the set of pairs obtained by adding the pair $\{p, q\}$ to the set returned by recursively applying the algorithm to the remaining campers; otherwise, return the empty set.

b. Prove that the following greedy algorithm always finds an optimal set: If $n < 2$, return the empty set. Otherwise, let p be a lightest camper and q be a heaviest camper (ties broken arbitrarily). If $w_p + w_q \leq C$, return the set of pairs obtained by adding the pair $\{p, q\}$ to the set returned by recursively applying the algorithm to the remaining campers; otherwise, discard q and return the set returned by recursively applying the algorithm to the remaining campers. The recursion ends when the number of remaining campers is less than two. (**Hint.** Prove that if the combined weight of a lightest and heaviest camper does not exceed C, there is an optimal set that contains a pair consisting of these two campers.)

c. Suppose the camp has only four-person canoes; i.e., each canoe must carry exactly four campers. Consider the following greedy algorithm for this case: Take the two lightest and two heaviest campers (ties broken arbitrarily). If the combined weight of these four campers is at most C add this group of four campers to the set returned by recursively applying the algorithm to the remaining campers; otherwise discard the heaviest of the four campers and return the set returned by recursively applying the algorithm to the remaining campers. The recursion ends when the number of remaining campers is less than four. Does this algorithm work? Justify your answer.

Question 2. (10 marks) A computer network can be described as a directed graph $G = (V, E)$ with positive edge weights $\mathbf{wt} : E \to \mathbb{Z}^+$. The nodes represent routers, the edges represent unidirectional links connecting routers, and the weight of edge (u, v) is the rate at which router u can transmit data to router v, measured in bits per second. To stream a file from node s to node t we must select a path from s to t, along which to send the file's bits in their sequence. The rate at which we can transmit the file along a path is limited by the ***slowest*** link on that path. Thus, we define the transmission rate of path $p = v_1, v_2, \ldots, v_k$ as $\mathbf{tr}(p) = \min_{1 \le i < k} \mathbf{wt}(v_i, v_{i+1})$. To transmit a file from node s to node t as fast as possible we would like to use an $s \to t$ path of ***maximum*** transmission rate.

Modify Dijkstra's algorithm to compute an $s \to t$ path of maximum transmission rate given the graph G, the edge weight function $\mathbf{wt}$, and the nodes s, t. Explain the information that your algorithm maintains. You do not need to prove that your algorithm is correct, but you should re-state the five claims in the proof of correctness of Dijkstra's algorithm, adjusted to the information maintained by your algorithm. (You don't need to prove these lemmas: If you do things right, the proofs are essentially the same as for Dijkstra's algorithm, and you can use this fact to check if you did things right!)