CMPSC 465 Data Structures & Algorithms
Fall 2024 Mingfu Shao and Debarati Das

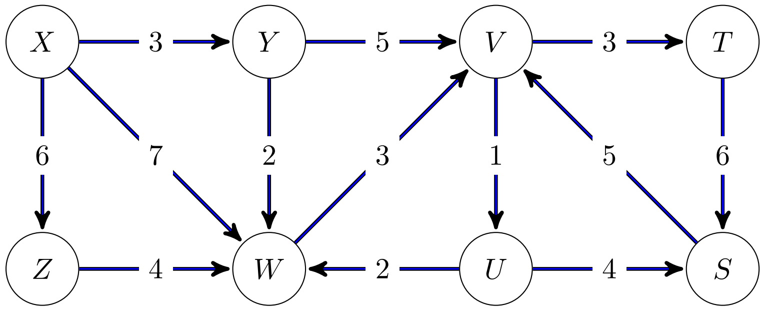
HW 5

Due October 4th (Friday), 11:59 pm

Formatting: Each problem should begin on a new page. When submitting in Gradescope, try to assign pages to problems from the rubric as much as you can. Make sure you write all your group members' names. For the full policy on assignments, consult the syllabus.

0. (0 pts.) Acknowledgements. List any resources besides the course material that you consulted in order to solve the assignment problems. If you did not consult anything, write "I did not consult any non-class materials." The assignment will receive a 0 if this question is not answered.

1. (8 pts.) Run Dijkstra's algorithm on the following graph, starting at node X. Whenever there is a choice of vertices with the same *dist* value, always pick the one that is alphabetically first. Please draw a table where each row shows the *dist* array at each iteration of the algorithm.



$$\begin{tikzcd}
X \arrow[r, "3"] \arrow[d, "6"] \arrow[dr, "7"] & Y \arrow[r, "5"] \arrow[d, "2"] & V \arrow[r, "3"] \arrow[d, "1"] & T \arrow[d, "6"] \\
Z \arrow[r, "4"] & W \arrow[ur, "3"] & U \arrow[l, "2"] \arrow[r, "4"] & S \arrow[ul, "5"]
\end{tikzcd}$$

2. (12 pts.) Let's see an application of (binary) heap. You are given an array A with n integers, and another integer k, $1 \leq k \leq n$. The numbers in A are either -1 or a positive integer, and you may assume that all positive integers are distinct (but there could be multiple -1s). You are asked to design data structures and algorithm to produce an output array X. Your algorithm should process the numbers in A one by one: when a positive number is met, you put it to the end of X; if a -1 is met, you need to remove the k-th smallest number in X. You may also assume that, you will never meet a -1 if the size of the current X is smaller than k. For example, if $A = [4, 2, -1, 7, 3, 8, -1, 6]$ and $k = 2$, the output X should be $X = [2, 7, 8, 6]$.

   Design an algorithm to complete this task and analyze its running time. Your algorithm should run in $O(n \log n)$ time. (Hint: consider using two binary-heaps, one max-heap and one min-heap; the max-heap maintains the smallest k numbers in X and the min-heap maintains rest of the numbers.)

3. (10 + 5 bonus pts.) A tree is defined as a connected, undirected graph without any cycle. It is obvious that there is a single, unique path between any two vertices in a tree. Let $T = (V, E)$ be a tree and let $\delta(u, v)$ be the length of the path between u and v. Here we assume unit edge length, i.e., $\delta(u, v)$ is the number of edges in the path between u and v. We aim to design an algorithm to find the length of the longest path in T, i.e., $\max_{u,v \in V} \delta(u, v)$. If (u', v') reaches this maximum, i.e., $\delta(u', v') = \max_{u,v \in V} \delta(u, v)$, we say (u', v') forms a furthest pair. Note that T may contain multiple furthest pairs.

a. (5 bonus pts) For any vertex $u \in V$, we say v' is the furthest vertex to u, if $\delta(u, v') = \max_{v \in V} \delta(u, v)$. Prove that, for any vertex $u \in V$ with v' being (any one of) its furthest vertex, there must exist vertex u' such that (u', v') is one furthest pair of T.

b. (10 pts) Given T, design an algorithm to find a furthest pair and analyze its running time. Your algorithm should run in $O(|V| + |E|)$ time. (Hint: consider using the results of part (a) and BFS.)