**CISC2006: Algorithm Design and Analysis**
**Homework 4**                                  **Due Date:** 23:59, 26 April 2024
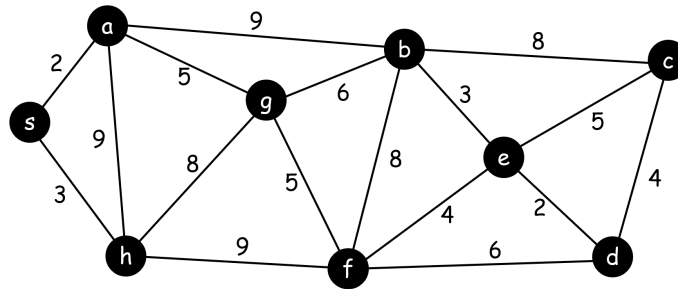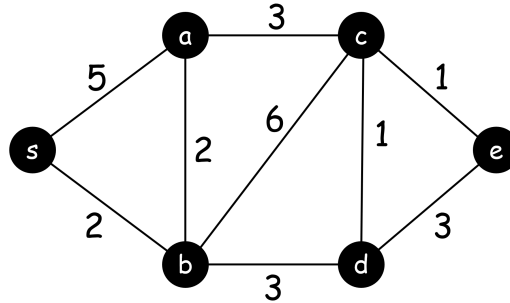
Rules: Discussion of the problems is permitted, but writing the assignment together is not (i.e. you are not allowed to see the actual pages of another student). You can get at most 100 points if attempting all problems. Please make your answers precise and concise.

1. (20 pts) Consider the problem of computing the Minimum Spanning Tree (MST) of the following edge-weighted graph.



- List the edges that appear in the MST in the order they are included using Kruskal's algorithm. For example, if edge $(a, b)$ is included as the first edge in the MST and $(c, d)$ is the second,..., then the list is $(a, b), (c, d), \ldots$

- List the edges that appear in the MST in the order they are included using Prim's algorithm. Suppose we begin with node $s$.

2. (15 pts) Consider running Dijkstra's Algorithm on the following graph to compute the distance $d(s, u)$ between $s$ and every other node $u \in V$.



Let $S \subseteq V$ contain the set of nodes $u$ whose distance $d(s, u)$ is known.

Initially we have $S = \{s\}$ and $d(s, s) = 0$. In each step each node maintains an estimate $d'(s, u)$ (which is also an upper bound) of $d(s, u)$.

Fill in the values for each step of the execution.

Initialization: set $d'(s, a) = $ _____, $d'(s, b) = 2$, $d'(s, c) = d'(s, d) = d'(s, e) = \infty$.

Pick the node $b$ with the smallest estimate, and include it into $S$: $S \leftarrow S \cup \{b\}$; set $d(s, b) = 2$ .

Update the estimates of nodes in $V \setminus S$ as follows:

$d'(s, a) = \min\{5, d(s, b) + $ _____$\} = $ _____, $d'(s, c) = $ _____

$d'(s, d) = $ _____, $d'(s, e)$ remains the same value as before.

Pick the node $a$ with the smallest estimate, and include it into $S$: $S \leftarrow S \cup \{a\}$; set $d(s, a) = $ _____.

Update the estimates of nodes in $V \setminus S$ as follows:

$d'(s, c) = \min\{8, d(s, a) + $ _____$\} = $ _____.

$d'(s, d)$ and $d'(s, e)$ remain the same values as before.

Pick the node _____ with the smallest estimate, and include it into $S$: _____;
set $d(s,$ _____$) = $ _____.

Update the estimates of nodes in $V \setminus S$ as follows: _____,

_____.

Pick the node _____ with the smallest estimate, and include it into $S$: _____;
set $d(s,$ _____$) = $ _____.

Update the estimates of nodes in $V \setminus S$ as follows: $d'(s, e) = $ _____.

Pick the node $e$ with the smallest estimate, and include it into $S$: $S \leftarrow S \cup \{e\}$; set $d(s, e) = 7$. Since $S = V$, the algorithm terminates.

3. (15 pts) Let $V$ denote a set of $n$ nodes $v_1, v_2, \ldots, v_n$. A degree sequence is a list of $n$ positive integers $d_1 \geq d_2 \geq \ldots \geq d_n$. Given a degree sequence, determine whether there exists an undirected graph $G = (V, E)$ such that for all $i$, the degree of $v_i$ is $d_i$.

Draw the graph if such graph exists; otherwise prove that it does not exist. Note that $G$ is not allowed to contain any self-loop (edge with two endpoints being the same node) or multiple edges between a pair of nodes.

(1) Degree sequence: $(4, 3, 3, 2, 2)$

(2) Degree sequence: $(3, 2, 2, 1, 1)$

(3) Degree sequence: $(4, 4, 3, 2, 1, 1)$

4. (10 pts) Let $G(V, E)$ be an undirected connected graph. Give an algorithm that finds an edge $e$ in $G$ such that removing $e$ from $G$ would disconnect $G$ (and reports none if no such edge exists). Your algorithm should run in $O(n + m)$ time, where $n = |V|$ is the number of nodes and $m = |E|$ the number of edges.

5. (20 pts) **Implementation of the Graph Traversal Algorithms**.

In this problem you need to implement the graph traversal algorithms using DFS and BFS. In the main function, your algorithm needs to (1) read an integer $n$ that denotes the number of nodes; (2) read a sequence of edges, where each edge is specified by $(i, j)$, indicating that there is an edge between node $i$ and node $j$.

You are guaranteed that both $i, j \in \{1, 2, \ldots, n\}$.

Based on the input, you need to construct an undirected unweighted graph using adjacency lists. That is, every node $u$ stores its set of incident edges as a list.

Then a function is called to traverse the graph from node 1, and output the sequence of nodes in the order they are visited by the algorithm.

You need to implement each of the following 2 algorithms as a function.

- Depth First Search (DFS);
- Breadth First Search (BFS).

Several test cases of input will be provided. For each test case, the main function calls the two traversal algorithms, each of them outputs a traversal sequence of nodes.

6. (20 pts) **Implementing the Dijkstra's Algorithm**.

In this problem you need to implement the Dijkstra's algorithms for computing the single source shortest path (SSSP) problem. In the main function, your algorithm needs to (1) read an integer $n$ that denotes the number of nodes; (2) read a sequence of edges, where each edge is specified by $(i, j, w)$, indicating that there is an edge between node $i$ and node $j$ with integer edge weight $w$.

You are guaranteed that both $i, j \in \{1, 2, \ldots, n\}$ and $w \geq 1$.

Based on the input, you need to construct an undirected weighted graph using adjacency lists. That is, every node $u$ stores its set of incident edges as a list. Then a function is called to compute the distance $d(1, i)$ between node 1 and node $i$, for every $i \in \{2, 3, \ldots, n\}$, and output the distances $d(1, 2), d(1, 3), \ldots, d(1, n)$.

Recall that to implement Dijkstra's algorithm we need to update and maintain an estimate $d'(1, i)$ for the distance between 1 and $i$, for every $i \in \{2, 3, \ldots, n\}$. You need to store the estimates in a Binary Search Tree (BST) that supports the query of minimum value. You can import libraries for the BST data structure.

Several test cases of input will be provided. For each test case, the main function calls the Dijkstra's algorithm to compute and output the distances.

7. (20 pts) **Implementation of Algorithms for the MST Problem**.

In this problem you need to implement the Kruskal's algorithm and Prim's algorithm for the Minimum Spanning Tree (MST) problem. In the main function, your algorithm needs to (1) read an integer $n$ that denotes the number of nodes; (2) read a sequence of edges, where each edge is specified by $(i, j, w)$, indicating that there is an edge between node $i$ and node $j$ with integer edge weight $w$.

You are guaranteed that $i, j \in \{1, 2, \ldots, n\}$ and $w \geq 1$.

Based on the input, you need to construct an undirected weighted graph using adjacency lists. That is, every node $u$ stores its set of incident edges as a list. Then a function is called to compute the MST of the input graph, output the edges in the MST and also output the total edge weight in the MST.

You need to implement each of the following 2 algorithms as a function.

- Kruskal's algorithm that repeatedly includes an edge with minimum weight as long as it does not create a cycle. Recall that to test whether a new edge creates a cycle or not, you need to maintain a Union-Find data structure for the nodes. You can import libraries for the Union-Find data structure.

- Prim's algorithm that repeatedly includes an edge with minimum weight to grow the current partial tree. You should run Prim's algorithm with $s = 1$, i.e., start growing the tree from node 1 outwards. Recall that to find the minimum weight edge, you need to maintain a Binary Search Tree (BST) that supports the query of minimum value. You can import libraries for the BST data structure.

Several test cases of input will be provided. For each test case, the main function calls the two algorithms, each of them outputs a MST and its total edge weight.