CSC 505, Homework 4
Due date: Friday November 5, 9 PM

Homework should be submitted via Gradescope in PDF, or plain text. To avoid reduced marks, please submit **a word/PDF file**, **NOT scanned writing in pdf format**. All assignments are due on 9 PM of the due date. Late homework will be accepted only in circumstances that are grounds for excused absence under university policy. The university provides mechanisms for documenting such reasons (severe illness, death in the family, etc.). Arrangements for turning in late homework must be made by the day preceding the due date if possible.

All assignments for this course are intended to be individual work. Turning in an assignment which is not your own work is cheating. The Internet is not an allowed resource! Copying of text, code or other content from the Internet (or other sources) is plagiarism. Any tool/resource must be approved in advance by the instructor and identified and acknowledged clearly in any work turned in, anything else is plagiarism.

If an academic integrity violation occurs, the offending student(s) will be assessed a penalty that is at least as severe as getting a 0 for the whole homework for which the violation occurred, and the case will be reported to the Office of Student Conduct.

Instructions about how to "give/describe" an algorithm (taken from Erik Demaine): Try to be concise, correct, and complete. To avoid deductions, you should provide (1) a textual description of the algorithm, and, if helpful, flow charts and pseudocode; (2) at least one worked example or diagram to illustrate how your algorithm works; (3) a proof (or other indication) of the correctness of the algorithm; and (4) an analysis of the time complexity (and, if relevant, the space complexity) of the algorithm. Remember that, above all else, your goal is to communicate. If a grader cannot understand your solution, they cannot give you appropriate credit for it.

1. Purpose: *Learn about Euler tours, practice describing algorithms.* Please solve Problem 22-3 (12 points) on page 623.

2. Purpose: *Reinforce your understanding of MST algorithms, and practice algorithm design (16 points).* Let T be the Minimum Spanning Tree of a graph G=(V,E,w). Suppose g is connected, $|E| \geq |V|$, and that all edge-weights are distinct. Denote T* the MST of G and ST(G) be the set of all spanning trees of G. A second-best MST is a spanning tree T such that $w(T) = \min\{w(T): T \in ST(G)-\{T^*\}\}$.

a) (4 points) Show that T* is unique, but that the second-best MST T2 need not be unique.

b) (4 points) Prove that G contains an edge $(u,v) \in T^*$ and another edge $(s,t) \notin T^*$ such that $(T^*-\{(u,v)\}) \cup \{(s,t)\}$ is a second-best minimum spanning tree of G.

c) (6 points) Please use Kruskal's algorithm to design an efficient algorithm to compute the second-best minimum spanning tree of G.

3. *Purpose: Reinforce your understanding of Dijkstra's shortest path algorithm, and practice algorithm design (16 points).* Suppose you have a weighted, undirected graph *G* with positive edge weights and a start vertex *s*.

a) (6 points) Describe a modification of Dijkstra's algorithm that runs (asymptotically) as fast as the original algorithm, and assigns a binary value usp[*u*] to every vertex *u* in *G*, so that usp[*u*]=1 if and only if there is **a unique shortest path from *s* to *u***. We set usp[*s*]=1. In addition to your modification, be sure to provide arguments for both the correctness and time bound of your algorithm, and an example.

b) (10 points) Implement the algorithm described in a). As you can see in the code framework provided (DijkstraFramework), Dijkstra.py/java file has a method: Dijkstra_alg. The method has an input parameter list (n, e, mat, s), where n = number of vertices of G, e = number of *undirected* edges of G, mat = an e x 3 matrix that defines the edges of G, and s = the source vertex of Dijkstra's algorithm. **Assume the vertices of G are numbered 1…n.** In mat each row consists of three integers <u, v, weight>. Here, u and v define the edge (u,v), and weight is the corresponding edge weight. The method returns an n x 2 matrix, where the i*th* row contains the path length and the usp value for the shortest path between source and vertex *i* for $i \in \{1, \ldots n\}$. More detailed examples (that refers to the first two test cases) are given below. You can create additional methods if required but do not change the name of existing methods and any existing code - points will be cut if you do. To avoid loss of marks, you should use basic arrays to implement the priority queue in Dijkstra's algorithm, and not use any packages. You can code this in either Java or Python. Another file DijkstraTest.py/java has been provided which tests your code for various inputs. You need to test your code on VCL using this file (instructions for VCL setup and test can be referred from the file *HW4 Programming Assignment Setup Instructions* on our moodle page). Please only submit the file Dijkstra.py/java and not the test file. Your code will be checked on VCL by us automatically using the provided cases, plus some (unknown) inputs. To avoid loss of marks please make sure that all provided test cases pass on VCL by using the test file.

Some examples:

Testcase1

Input values

n = 5,
e = 6,

mat = [[1, 2, 9], [1, 3, 6], [1, 4, 5], [1, 5, 3], [3, 2, 2], [3, 4, 4]],
s = 1

Output

Ans = [[0, 1], → Shortest path weight source to vertex 1 is 0, usp is 1
[8, 1], → Shortest path weight source to vertex 2 is 8, usp is 1
[6, 1], → Shortest path weight source to vertex 3 is 6, usp is 1
[5, 1], → Shortest path weight source to vertex 4 is 5, usp is 1
[3, 1]] → Shortest path weight source to vertex 5 is 3, usp is 1

All shortest paths originating from source vertex 1 are unique, and the corresponding usp value is 1.

Testcase2:

Input values

n = 5,
e = 5,
mat = [[1, 3, 2], [1, 5, 3], [2, 5, 3], [4, 1, 1], [4, 2, 1]]
s = 4

Output

Ans = [[1, 1], → Shortest path weight source to vertex 1 is 1, usp is 1
[1, 1], → Shortest path weight source to vertex 2 is 1, usp is 1
[3, 1], → Shortest path weight source to vertex 3 is 3, usp is 1
[0, 1], → Shortest path weight source to vertex 4 is 0, usp is 1
[4, 0]] → Shortest path weight source to vertex 5 is 4, usp is 0

There are two shortest paths of length 4 from source vertex 4 to vertex 5. Therefore, the usp value of fifth row is 0. All other shortest paths originating from vertex 4 are unique, and therefore the corresponding usp value is 1.