Due date: Monday, November 19, 9:00 PM

Homework should be submitted via Moodle in PDF, or plain text. To avoid reduced marks, please submit **word/latex-formated 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.* Please solve Problem 22-3 a (6 points) on page 623.

2. *Purpose: Reinforce your understanding of minimum spanning trees* (9 points). Please sole problem 23-4 on page 641. You do NOT have to describe the most efficient implementation of each algorithm.

3. *Purpose: Reinforce your understanding of minimum spanning trees, practice algorithm design*. Suppose you are given a connected graph G=(V,E), with edge costs that you may assume are all distinct. A particular edge e\*=(u,v)∈E of G is specified.

a) (6 points) Show that e\* does not belong to any MST of G if and only if v and u can be joined by a path consisting entirely of edges that are cheaper than e\*.

b) (6 points) Give an algorithm with running time O(|V|+|E|) to decide whether e\* is contained in any minimum spanning tree of G. Tip: use a).

4. *Purpose: Reinforce your understanding of Dijkstra’s shortest path algorithm, and practice algorithm design (6 points).* Suppose you have a weighted, undirected graph *G* with positive edge weights and a start vertex *s*. 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.

*5. Purpose: Reinforce your understanding of Dijkstra’s shortest path algorithm, learn about multiple solutions, and practice algorithm design and Implementation (12 points).* In the usual formulation of Dijkstra’s algorithm, the number of edges in the shortest (= lightest) path is not a consideration. Here, we assume that there might be multiple shortest paths. Implement an algorithm that takes as input an undirected graph **G = (V, E)**, a nonnegative cost function **w** on **E**, a source vertex **s** and a destination vertex **t**, and produces a path with the fewest edges amongst all shortest paths from **s** to **t**. If there are multiple such shortest paths with the fewest edges, your algorithm should output the *unique* path with the lexicographically smallest sequence of vertices amongst all such paths. Please describe (6 points) and implement your algorithm (6 points). **Your implementation should follow the description of Dijkstra's algorithm given in class.**

**I/O Specification:**

The input should be read from the console, and the output should be printed to the console.

The first line of input contains two integers separated by a space: the number of vertices **|V| (1<=|V|<=50)**, and the number of edges **|E| (0<=|E|<=100)**. The next **|E|** lines each describes an undirected edge. An edge is described by three integers separated by space: the end-points **u** and **v (0<=u, v<=|V|-1)**, and the cost **w(u,v) (0<=w(u,v)<=1000)**. The last line of input contains the source-destination **s** and **t (0<=s, t<=|V|-1)** separated by space.

Your program should print two lines. In the first line, output **the cost of the optimal path** computed by your algorithm. In the second line, output **the sequence of vertices in the path** (for details, see the problem description) separated by spaces. ***Do not* print an extra space after the last vertex in the sequence; *do* print a newline after the sequence**. You can assume that there will always be a solution.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 13 13  0 1 2  1 3 4  3 6 12  6 7 2  0 4 3  4 2 2  2 5 7  5 7 8  7 10 3  10 12 2  7 9 1  9 11 1  11 12 3  0 12 | 25  0 1 3 6 7 10 12 |
| 5 5  0 1 1  1 2 2  2 4 2  0 3 4  3 4 1  0 4 | 5  0 3 4 |
| 5 5  3 4 1  2 4 2  0 1 1  1 2 2  0 3 4  2 1 | 2  2 1 |

Like in the previous homework, we will use the **AutoGradr (**[**https://app.autogradr.com/**](https://app.autogradr.com/)**)** for grading your programs. Please submit your code in the **“Homework 4: LSFESP”** project. **The project will be posted in the site 2 weeks before the deadline. The submission rules will remain the same as before.**