Homework 5

Due Oct 9 by 8am **Points** 20 **Submitting** a file upload **File Types** pdf
Available Oct 2 at 8am - Oct 9 at 8am

This assignment was locked Oct 9 at 8am.

Suggested reading

Chapter 3 (Graph traversal), Chapter 4 (Dijkstra's), Chapter 5 (MST), Chapter 7 (Flows)

Instructions

For the **graded problems**, you are allowed to use the algorithms from class as black-boxes without further explanation. These include

- **DFS** (outputs connected components, topological sort on a DAG. You also have access to the prev, pre and post arrays), the **Explore** subroutine, and **BFS**.
- **Dijkstra's algorithm** to find the shortest distance from a source vertex to all other vertices and a path can be recovered backtracking over the prev labels.
- **Bellman-Ford** and **Floyd-Warshall** to compute the shortest path when weights are allowed to be negative.
- **SCC**s which outputs the strongly connected components, and the metagraph of connected components.
- **Kruskal's** and **Prim's** algorithms to find an MST.
- **Ford-Fulkerson** and **Edmonds-Karp** to find max flow on networks.
- **2-SAT** which takes a CNF with all clauses of size ≤ 2 and returns a satisfying assignment if it exists.

When using a black-box, make sure you clearly describe which input you are passing into it and how you use the output or take advantage of the data structures created by the algorithm. To receive full credit, your solution must:

- Include the description of your algorithm in words (no pseudocode!).
- Explain the correctness of your design.
- State and analyse the running time of your design (you can cite and use the running time of black-boxes without further explanations).

Unless otherwise indicated, black-box graph algorithms should be used without modification.

Example: I take the input graph G, I first find the vertex with largest degree, call it v*. I take the complement of the graph G, call it G'. Run Dijkstra's algorithm on G' with s = v* and then I get the array dist[v] of the shortest path lengths from s to every other vertex in the graph G'. I square each of these distances and return this new array.

We don't want you to go into the details of these algorithms and tinker with it, just use it as a black-box as shown with Dijkstra's algorithm above.

Practice Problems (do not turn in)

[DPV] Problem 7.10 (max-flow = min-cut example)

[DPV] Problem 7.17 (bottleneck edges).

[DPV] Problem 7.19 (verifying max-flow)

Graded Problem

You are given a ***undirected, connected,*** weighted graph G = (V, E) with positive edge weights. Give a linear time (O(|E| + |V|)) algorithm to decide if an input edge e = (u, v)***, an edge of the graph G***, is part of some MST of G or not. You should describe your algorithm in words (a list is okay); no pseudocode. Explain why it is correct and justify its running time.