Homework 4

Start Assignment

- Due Monday by 8am
- Points 20
- Submitting a file upload
- File Types pdf
- Available Feb 12 at 8am - Feb 19 at 8am

Suggested reading

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

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 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 pre 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 3.3 (Topological ordering example)

[DPV] Problem 3.4 (SCC algorithm example)

[DPV] Problem 3.5 (Reverse of graph)

[DPV] Problem 3.15 (Computopia)

[DPV] Problem 4.14 (Shortest path through a given vertex)

[DPV] Problems 5.1, 5.2 (Practice fundamentals of MST designs)

[DPV] Problem 5.9 (multiple statements about MST. We will provide the answer to a few, you are welcome to try them all)

Graded Problem

We ~~said~~ say a directed graph G=(V,E) is *well-connected* if for every pair of distinct vertices u and v in V, there exists a path from u to v OR a path from v to u (both may exist, but at least one must be present). Design an algorithm that takes as input a directed graph G and determines if it is well-connected.
Faster (in asymptotic Big O notation) and correct solutions are worth more credit.