CS3000: Algorithms & Data — Summer 2023 — Laney Strange

Homework 5 - Long
Due Tuesday June 13 @ 9pm Gradescope

Name:
Collaborators:

- Put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This assignment is due Tuesday June 13 @ 9pm Gradescope. You may submit up to 48 hours late for no penalty, but expect a delay in grading.
- You will have an opportunity to resubmit one short homework and one long homework for new grades, at the end of the semester.
- Solutions must be typeset, preferably in LaTeX. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.
- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class, is strictly forbidden.

Problem 1. *BFS and Bipartiteness (2 + 4 = 6 points)*

An unweighted, undirected graph can be labelled *bipartite* if its vertices can be divided into two independent sets, U and V, such that every edge (u, v) connects a vertex from U to V.

(a) Draw an example of a bipartite graph with at least 3 vertices in U and at least three vertices in V.

Solution:

(b) Describe an algorithm that determines whether a graph is bipartite. (You don't need formal pseudocode here; a clear description of the algorithm suffices.) (*Hint:* Try painting the vertices different colors like we've done in some graph algorithms.)

Solution:

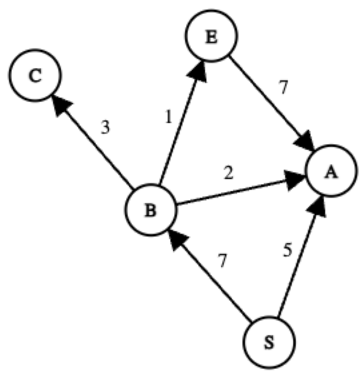
Problem 2. *Minimum Spanning Tree (4 points)*

Give a counterexample, along with a clear explanation, to show that the following theorem is False: Let $G = (V, E)$ be a connected, weighted, undirected graph. Let A be a subset of E that is included in some minimum spanning tree for G, let $(S, V - S)$ be any cut of G such that no edge in A crosses the cut. Finally, let (u, v) be a safe edge for A crossing $(S, V - S)$. Then (u, v) is a light edge for the cut.

Solution:

Problem 3. *SSLP* ($2 + 4 + 2 = 8$ *points)*

For this problem, we'll modify Dijktra's to compute the **longest** paths from a source vertex s to all other vertices in a weighted, directed, acyclic graph. (Note that this is a hard problem for a general graph. But doable for a DAG!)



(a) For the graph above, give a valid topological sort of the vertices.

Solution:

(b) Give pseudocode for an algorithm to compute the longest paths on a DAG like the one above. Your algorithm should take a graph G represented in an adjacency list, a source vertex s, and a weight function w, and you can assume that the vertices are topologically sorted. (*Hint:* You can call/modify any helper functions we've seen in class.)

Solution:

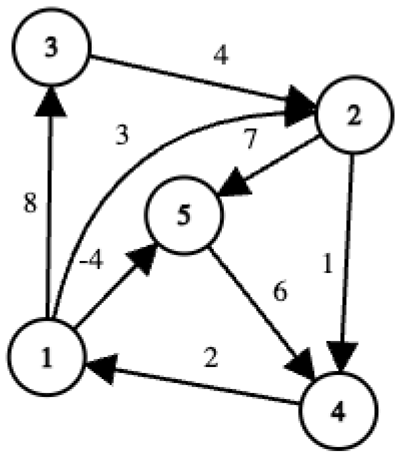
(c) Give the final $v.d$ and $v.\pi$ values your algorithm would generate for the graph above.

vertex	S	A	B	C	E
$v.d$					
$v.\pi$					

Solution:

Problem 4. *APSP* ($4 + 2 + 2 = 8$ *points)*

This problem is concerned with the graph below.



Recall that the recursive formula for the distances computed by the Floyd-Warshall APSP algorithm defines an entry (i, j) in the kth D-matrix as follows:

$d_{ij}^{(k)} = w_{ij}$ if $k = 0$ and
$\min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$ otherwise.

When $k = 0$, we also need to consider when there is no edge from i to j. So we also have $d_{ij}^{(0)} = 0$ if $i == j$ and ∞ if $i \neq j$ but there is no edge from i to j.

(a) Fill in the first two D matrices, $D^{(0)}$ and $D^{(1)}$ following the recursive formula above.

$D^{(0)}$

vertex	1	2	3	4	5
1					
2					
3					
4					
5					

$D^{(1)}$

vertex	1	2	3	4	5
1					
2					
3					
4					
5					

Solution:

(b) We're now interested in Π, the predecessor matrix that would help us construct a shortest path between any two vertices. We define a value $\pi_{ij}^{(k)}$ in the predecessor matrix $\Pi^{(k)}$ as $\pi_{ij}^{(k)}$ = predecessor of vertex j on a shortest path from vertex i.

Give a formula for $\pi_{ij}^{(0)}$, the entry for i, j in the initialization matrix.

Solution:

(c) Give a recursive formula for entry (i, j) in the kth Π matrix. You can assume all relevant D matrices exist and you can refer back to them.

Solution: