CS 344 - Sections 1,2,3 - Fall 2018
Homework 6
Due Friday, Dec 14, 11:55 pm
40 points total plus extra credit

1 Problem 1 (10 points)

Consider the directed graph in Figure 1 on the next page. Think of the vertex with number i as vertex v_i.

- Part 1 (5 points): what does the adjacency matrix look like for this graph?
- Part 2 (5 points): what is the adjacency list representation of this graph?

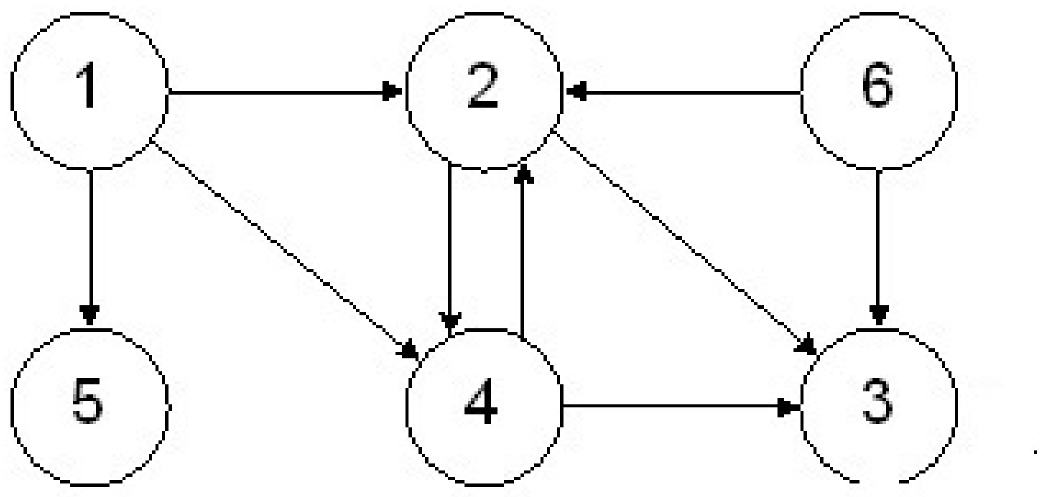


Figure 1: Graph for Problem 1.

2 Problem 2 (10 points)

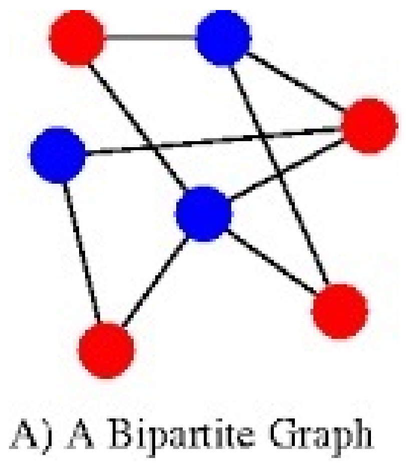
Give an example of a graph G with positive weights, and a pair of vertices s and t, such that if we run BFS(G,s), the s-t path returned by BFS is NOT the shortest path from s to t in the graph.

In addition to the graph, make sure to indicate the the $s-t$ path returned by BFS, as well as the actual shortest path from s to t in the graph.

Your graph should contain at most 7 nodes. You can get away with a lot fewer.

3 Problem 3 (20 points)

We say that an undirected graph $G = (V, E)$ is bipartite if it is possible to color every vertex either red or blue in such a way that for every edge $(u, v) \in E$, u and v have different colors. For example, in Figure 3, the graph on the left is bipartite because such a red-blue coloring is possible, but the graph on the right is not bipartite: you can check for yourself no matter how you color the vertices of the right graph red/blue, there will always be an edge between two vertices of the same color.



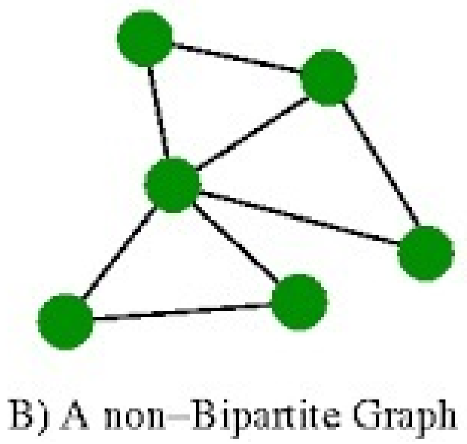


Figure 2: Graphs for Problem 3. The left grah is bipartite. It is not hard to check that the right graph is non-bipartite, because no matter how you color the vertices, there will be a red-red edge or a blue-blue edge.

The Question: Assume you are given a graph G that is connected: that is, there is a path from every vertex to every other vertex. Describe an algorithm CheckBipartite(G) that colors the vertices red and blue in the appropriate way, or returns "no solution" if the graph is not bipartite. Make sure to explain why in the case that your algorithm returns "no solution", there is in fact no valid red-blue coloring. Also, make sure to analyze the running time of your algorithm. (Full credit will only be given if the running time is $O(|E|)$).

HINT 1: Start by picking an arbitrary vertex v and coloring it red. Now, proceeding from v, try to color the rest of the vertices in a way that doesn't create any illegal edge (by illegal I mean blue-blue or red-red.)

HINT 2: BFS will prove very useful here. You don't have to re-describe how BFS works.

4 Extra Credit

Given a directed graph $G = (V, E)$, a vertex $v \in V$ is said to be a *universal sink* if v has $n - 1$ incoming edges, but zero outgoing edges; that is, v has an incoming edge from every other vertex, but an outgoing edge to none.

If G is given in adjacency list format, it is not hard to check that you need $|E|$ time to solve the problem.

The Question: Now, say that G is given in adjacency matrix format. This turns out to be very helpful for this problem. Write pseudocode for an algorithm that finds the universal sink in time $O(|V|)$.