CSE 2321 WorkSheet 8

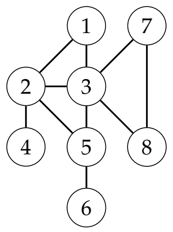
(Prof. Baha Alzalg)

April 20, 2022

Question 1.

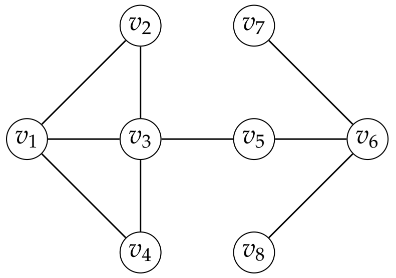
[1+1 points] Represent the following graph using:

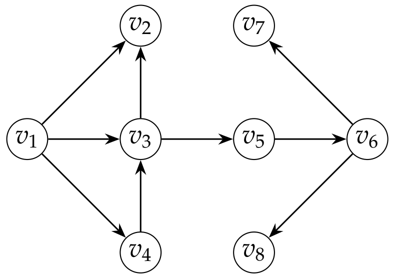
1. Adjacency list representation.
2. Adjacency matrix representation.



Question 2.

[2+2+3+1+2 points] Consider the following two graphs; one of them is undirected and the other is directed





1. Run a breadth-first search on the undirected graph to obtain the breadth-first tree.
2. Use item 1 to find the shortest path from vertex v_1 to vertex v_8. Is this shortest path unique?
3. Run a depth-first search on the directed graph to compute a depth-first tree. Classify the edges as tree edges, back edges, forward edges, and cross edges.
4. Does the directed graph has a directed cycle? Why or why not? Link your answer to the depth-first search completed in item (c).
5. Compute a topological ordering for the directed graph.

Question 3.

[3 points] Breadth-first search can be used to test bipartiteness. The algorithm starts the search at any vertex and gives alternating labels to the vertices visited during the search. That is, give label 0 to the starting vertex, 1 to all its neighbors, 0 to those neighbors' neighbors, and so on. If at any step a vertex has (visited) neighbors with the same label as itself, then the graph is not bipartite. If the search ends without such a situation occurring, then the graph is bipartite.

The following workflow to test bipartiteness uses the concept of graph coloring (recall that a graph is bipartite iff it is 2-colorable) and breadth-first search.

Workflow 1. *We test the bipartiteness of an input graph G by following two steps:*

Step 1. *Assign a color (say red) to a random source vertex.*

Step 2. *Assign all the neighbors of the above vertex another color (say blue).*

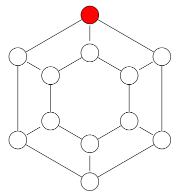
Step 3. *Take one neighbour at a time to assign all the neighbour's neighbors the color red.*

Step 4. *Continue in this manner till all the vertices have been assigned a color.*

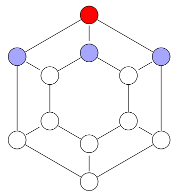
Step 5. *If at any stage, we find a neighbour which has been assigned the same color as that of the current vertex, stop the process. The graph cannot be colored using two colors. Thus the graph is not bipartite.*

We visually show the steps of Workflow 1 in Figure 1.

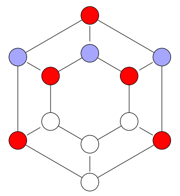
In this question, I want you to use the breadth-first search for testing the undirected graph in Question 2 for bipartiteness.



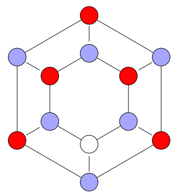
(*i*) We test the bipartiteness of this graph. Color source vertex, say dark red.



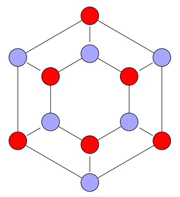
(*ii*) Assign neighbors of the source vertex another color, say light blue.



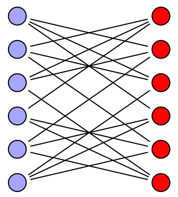
(*iii*) Assign all neighbors of the vertices colored in (*ii*) the color dark red.



(*iv*) Assign all neighbors of the vertices colored in (*iii*) the color light blue.



(*v*) Repeat until all vertices are colored, or a conflicting assignment occurs.



(*vi*) Since no conflicting evidence was found, the graph is bipartite.

Figure 1: Illustrating the progress of breadth-first search on testing bipartiteness of a graph.