CSCI 470 Homework 4

Please write your solutions in the LaTeX. You may use online compiler such as Overleaf or any other compiler you are comfortable with to write your solutions in the LaTeX.

Due date: Monday Nov. 06, 2023, 11:59 PM EDT.

Please submit a PDF (preferably written in LaTeX) or a scanned copy of your handwritten solutions to Homework 03 on Canvas. We are no longer accepting physical copies. Points will be deducted if handwritten solutions are not legible.

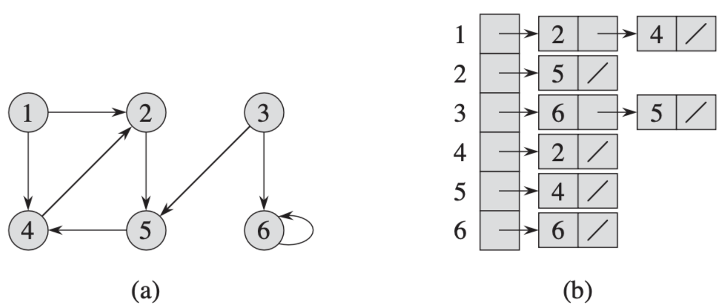
You can use the LaTeX submission template I have shared along with the homework. There are two .tex files (“macros.tex”, and “main.tex”). You can upload the zipped folder directly to Overleaf or create a blank project on Overleaf and upload macros.tex and main.tex files, and edit main.tex to write your solutions. “macros.tex” is mostly for macros (predefined commands).

Representation of graphs [10 points]

Problem 4-1. (10 points) Give an adjacency-list representation for a **complete binary tree** on 7 vertices. Give an equivalent adjacency-matrix representation. Assume that vertices are numbered from 1 to 7 as in a binary heap.

Breadth-first search [35 points]

Problem 4-2. (10 points) Show that d and π values that result from running breadth-first search on the directed graph of Figure 22.2(a), using vertex 3 as the source.



	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

(c)

Figure 22.2 Two representations of a directed graph. **(a)** A directed graph G with 6 vertices and 8 edges. **(b)** An adjacency-list representation of G. **(c)** The adjacency-matrix representation of G.

Problem 4-3. (10 points) Show that d and π values that result from running breadth-first search on the undirected graph of Figure 22.3, using vertex u as the source.

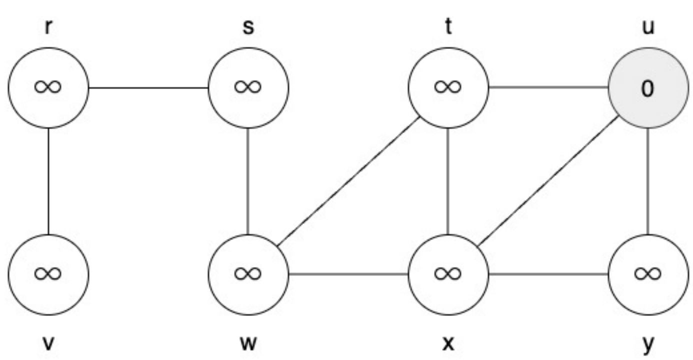


Figure 22.3

Problem 4-4. (5 points) Give a shortest path from vertex u to v in Figure 22.3 using the ***predecessor subgraph*** of G in problem 4-2. We define the predecessor subgraph of G as $G_\pi = (V_\pi, E_\pi)$, where $V_\pi = \{v \in V : v.\pi \neq \text{NIL}\} \cup \{s\}$ and $E_\pi = \{(v.\pi, v) : v \in V_\pi - \{s\}\}$.

Problem 4-5. (10 points) What is the running time of BFS if we represent its input graph by an adjacency matrix and modify the algorithm to handle this form of input?

Depth-first search [15 points]

Problem 4-6. (10 points) Show how depth-first search works on the graph of Figure 22.6. Assume that the **for** loop of lines 5-7 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show discovery and finishing times for each vertex.

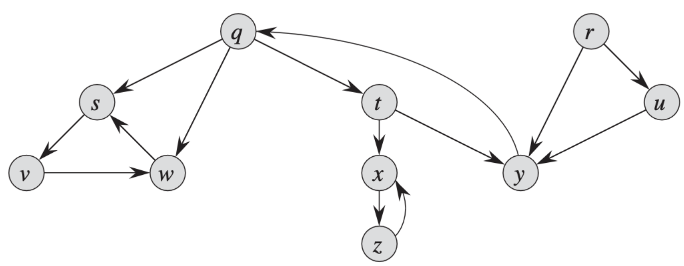


Figure 22.6

Problem 4-7. (5 points) Show the parenthesis structure of the depth-first search of Figure 22.4, under the assumption of Problem 4-6.

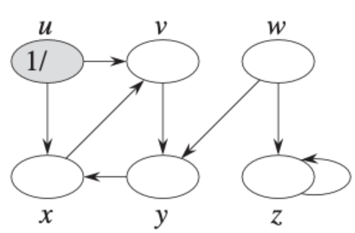


Figure 22.4

Topological sort [10 points]

Problem 4-8. (10 points) Show the ordering of vertices produced by TOPOLOGICAL-SORT when it is run on the dag of Figure 22.8. Assume that the **for** loop of lines 5-7 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically.

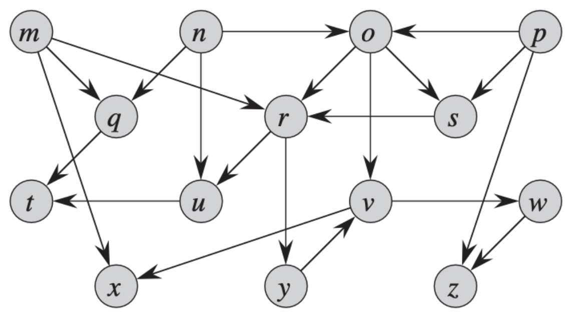


Figure 22.8 A dag for topological sorting.

Strongly connected components [10 points]

Problem 4-9. (10 points) Show how the procedure STRONGLY-CONNECTED-COMPONENTS works on the graph of Figure 22.6. Specifically, show the finishing times computed in line 1 and the forest produced in line 3. Assume that the loop of lines 5-7 of DFS considers vertices in alphabetical order and that the adjacency lists are in alphabetical order.

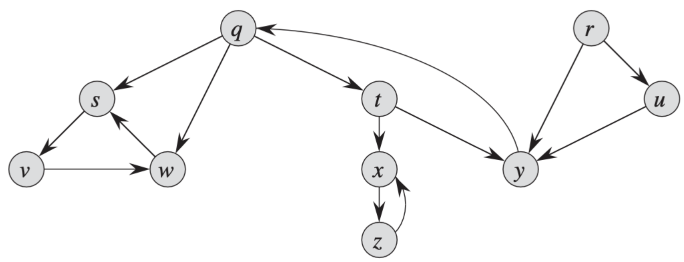


Figure 22.6

Extra Credit [30 points]

Problem 4-10. (10 points) Argue that in a breadth-first search, the value $u.d$ assigned to a vertex u is independent of the order in which the vertices appear in each adjacency list. Using Figure 22.3 as an example, show that the breadth-first tree computed by BFS can depend on the ordering within adjacency lists.

Problem 4-11. (10 points) Rewrite the procedure DFS, using a stack to eliminate recursion.

Problem 4-12. (10 points) Professor Bacon claims that the algorithm for strongly connected components would be simpler if it used the original (instead of the transpose) graph in the second depth-first search and scanned the vertices in order of *increasing* finishing times. Does this simpler algorithm always produce correct results? Your illustration or argument must include a graph with more than 3 vertices.