Questions and Exercises to work out and turn in:

Grading Guidelines:

* A right answer will get full credit when:

1. It is right (worth 25%)
2. It is right **AND** neatly presented making it easy and pleasant to read. (worth an **extra** 15%)
3. There is an **obvious and clear link** between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth an **extra** 60%).
4. Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.

You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, **personal** writing is expected.

* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **DO NOT DELETE ANYTHING FROM THIS FILE:** JUST **INSERT** YOUR ANSWERS.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), **USE THIS FILE** BY CREATING SUFFICIENT SPACE AND WRITE IN YOUR ANSWERS.
* FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST **A 30% PENALTY.**

Objectives of this assignment:

* to use and manipulate the concepts presented in this module
* to propose and write algorithms in pseudocode
* to analyze the time complexity of algorithms
* to analyze the space complexity of algorithms
* to learn autonomously new concepts

What you need to do:

Answer the questions and/or solve the exercises described below.

Questions (5 points):

Research online what *out-degree* and *in-degree* of a vertex v means. Use then your own words to explain the definition of *out-degree* and *in-degree.*

First given a graph of G = (V, E), which defines all the vertices and edges of the given graph G. Given an individual v, of the larger group V, there are any number of the given edges that will either go toward or away from a given vertex. Simply from the nature of this, only a directed graph can be used to represent the out-degree or in-degree of a given vertex. So, the out-degree of a given vertex, in only a directed graph, would be the total number of edges that start with the given vertex and reach out to another vertex. Each outgoing edge adds to the overall out-degree of a given vertex v. Then opposite that the in-degree of a given vertex would be the number of directed edges starting from some other vertex and being directed into the given vertex v. Each incoming edge adds to the overall in-degree of a given vertex v.

Exercise 1 (15 points)

Consider an adjacency-list representation of a directed graph G=(V,E).

1. Propose in pseudocode an algorithm A to compute the in-degree of each vertex in V.

// Algorithm A

Given a graph G = (V, E).

// Initialize in\_degree as empty dictionary.

For each vertex v in V:

Set in\_degree[v] = 0

For each vertex u in V:

For each neighbor v in adj\_list[u]:

Increment in\_degree[v] by 1

Return in\_degree // For each v in V

The idea should be to be to make a dictionary that will keep track of the in-degree for each vertex in G. First initializing the in-degree variable to 0 and then iterating over each vertex “u” and its neighbor’s “v” from the adjacency list. This will give a simple representation of the total in-degree of each vertex in G returning the value stored within “in\_degree”.

1. What is the time complexity of A?

The time complexity of the above algorithm A would be the cardinality of the vertices and edges. This can be represented as O(∣V∣) which will be the outer loop for each vertex in G. And then the inner loop would be represented as O(∣E∣). Thus, **the time complexity of Algorithm A is O(∣V∣+∣E∣).**

1. Propose in pseudocode an algorithm B to compute the out-degree of each vertex in V.

// Algorithm B

Given a graph G = (V, E)

// Initialize out\_degree as empty dictionary.

For each vertex v in V:

Set out\_degree[v] = length of adj\_list[v]

Return out\_degree

1. What is the time complexity of B?

The loop runs O(∣V∣) times, and for each vertex, which determines the length of its adjacency list takes O(1) time. Thus, the time complexity of Algorithm B is O(∣V∣).

Exercise 2 (35 points) Breadth-First Search

Consider the following graph G=(V,E):



1. Complete V= {y, x, ….} (Fill in the blanks. Sort V **reverse** alphabetically z🡪a)

V = {y, x, w, v, u, t, s, r}

1. Complete E = {(y,x), …} (**Pay attention**: G is undirected)

E = {(y,x),(y,u),(x,w),(x,u),(x,t),(w,r),(w,u),(w,t),(v,r),(u,y),(u,x),(u,t),(t,x),(t,w),(t,u),(s,w),(s,r),(r,v)(r,s)}

1. Complete the adjacency list as a table {sort Adj[u] **reverse** alphabetically z🡪a}

|  |  |
| --- | --- |
| Vertices u | Adj[u] |
| y | {x, u} |
| x | {y, w, u, t} |
| w | {x, t, s} |
| v | {r} |
| u | {y, x, t} |
| t | {x, w, u} |
| s | {w, r} |
| r | {v, s} |

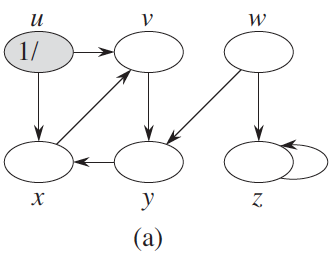
1. Execute Breadth-First Search (**BFS(G,v)**) on Graph G with taking Vertex ***v*** as the source. **Respect** the order of the adjacency list as completed in the previous question. Show all figures (a) through (i) just like Figure 22.3 in the textbook. The number of figures may differ.

A screenshot of a video game

Description automatically generated

Exercise 3 (35 points) Depth-First Search

Consider the following graph G=(V,E):



1. Complete V= {z, ….} (Fill in the blanks. Sort V alphabetically **in reverse z🡪a**)

V = {z, y, x, w, v, u}

1. Complete E = {(z,z), (y,x), (x,v), (w,z), (w,y), (v,y), (u,x), (u,v)}
2. Complete the adjacency list as a table {sort Adj[u] alphabetically **in reverse z🡪a**}

|  |  |
| --- | --- |
| Vertices u | Adj[u] |
| z | {z} |
| y | {x} |
| x | {v} |
| w | {z, y} |
| v | {y} |
| u | {x, v} |

1. Execute Depth-First Search (**DFS(G)**) on Graph G. **Respect** the order of the adjacency list as completed in the previous question. Show all figures (a) through (p) just like Figure 22.4 in the textbook. The number of figures may differ.

On next page

A group of white ovals on a black background

Description automatically generated

Exercise 4 (10 points) Topological Sort

Consider the following graph G=(V,E):

|  |  |
| --- | --- |
| Vertices u | Adj[u] |
| u | {v, x} |
| v | {y} |
| w | {y, z} |
| x |  |
| y | {x} |
| z |  |

1. Execute Topological Sort on Graph G. **Respect** the order of the adjacency list above. Make sure to collect the finish times and report them. No need to draw all intermediary steps.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Vertices r | u | v | w | x | y | z |
| Finish time r.f | 12 | 11 | 8 | 10 | 9 | 7 |

1. List the linear ordering.

In regards to topological sort of the given graph, the linear ordering would be: {u, v, x, y, w, z}

What you need to turn in:

* Electronic copy of this file (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.
* Recall that answers must be well written, documented, justified, and presented to get full credit.
* How this assignment will be graded:
* A right answer will get full credit when:
* It is right (worth 25%)
* It is right AND neatly presented making it easy and pleasant to read. (worth 15%)
* There is an obvious and clear link between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth 60%).
* Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.
* You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, personal writing is expected.