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[[1]](#footnote-1)** 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.*

1. The in-degree of a vertex v is the sum of all of the edges directed towards v. The out-degree of v is the sum of all of the edges directed away from v. In the case of an undirected graph, the out-degree will be equal to the in-degree, and they will both equal the number of edges connected to v.

Exercise 1 (15 points)

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

1. Propose the **pseudocode** of Algorithm A to compute the **out-degree** of each vertex in V.

OutDegreeCalculator(G)

1. int[] result = new int[];
2. int n = G[1].Length;
3. int sum;
4. for int i = 1 to n
5. sum = 0;
6. for int j = 1 to n
7. if G[i,j] == 1
8. sum += 1;
9. result.add(sum);
10. return result;
11. What is the time complexity of A?
    1. To measure the time complexity, comparisons will be used. The number of comparisons for the for loop in line 4 is n + 1. There are also n + 1 comparisons in the for loop in line 6, and since it is executed n times, the if statement on line 7 performs in total n comparisons. This gives a tie complexity of This equation grows as Ɵ(n2).
12. Propose the **pseudocode** of Algorithm B to compute the **in-degree** of each vertex in V.

InDegreeCalculator(G)

1. int[] result = new int[];
2. int n = G[1].Length;
3. int sum;
4. for int i = 1 to n
5. sum = 0;
6. for int j = 1 to n
7. if G[j,i] == 1
8. sum += 1;
9. result.add(sum);
10. return result;
11. What is the time complexity of B?
    1. Since the algorithm is the same as in part a except for the switch of the positions of i and j, the time complexity will be the same, totaling 2n2 + 3n + 1 and growing as Ɵ(n2).

Exercise 2 (35 points) Breadth-First Search

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



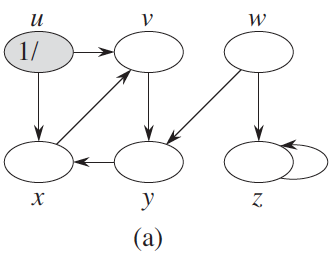
1. Complete V= {y, x, ….} (Fill in the blanks. Sort V **alphabetically in reverse order** z🡪a)
   1. {y, x, w, v, u, t, s, r}
2. Complete E = {(y,x), …} (**Pay attention**: G is undirected)
   1. {(y,u), (u,y), (x,u), (u,x), (w,t), (t,w), (w,s), (s,w), (u,t), (t,u), (s,r), (r,s)}
3. Complete the adjacency list as a table {sort Adj[u] **alphabetically in reverse order** z🡪a}

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

1. Execute Breadth-First Search (**BFS(G,x)**) on Graph G with taking Vertex ***x*** 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. As long as it is **neat**, you can draw by hand, take pictures, and insert the graphs

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**)
   1. {z, y, x, w, v, u}
2. Complete E = {(z,z), …}
   1. {(z,z), (y,x), (x,v), (w,z), (w,y), (v,y), (u,v), (u,x)}
3. 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.

Exercise 4 (10 points) Topological Sort

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

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

1. List the linear ordering (explain how you build it).

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.

**Appendix**: Grading: What is an OBVIOUS and CLEAR LINK?

Here is an example to explain what an **obvious and clear link** is and how we grade your work.

Consider the following problem:

"(100 points) John travels from Auburn to Atlanta in his car at a speed of 50 mph. Leaving at 8am, at what time will John reach Atlanta".

Here are the answers of three students and their scores:

**Student 1** answers: "10am". Student 1 will get 25 points.

**Student 2**answers : "John will reach Atlanta at 10am". Student 2 will get 25+15 = 40 points

**Student 3** answers: "The time t to travel a distance d at speed v is equal to d/v = d/50mph. The problem does not provide the distance d from Auburn to Atlanta. Based on Google, the distance from Auburn to Atlanta is approximately 100 miles (**document is here**). Therefore, the time t = 100 miles/50mph = 2 hours. Since John left at 8am, he will then reach Atlanta at 8am + 2 hours = 10 am".

**Student 3** will get 25 + 15 + 60 = 100 points

Do you see the **direct** **link** going from the data provided in the question to the final answer, using general knowledge/formula and documents?.... Can you now solve the following problem and get 100 points?

"(100 points) Alice travels from Auburn to Atlanta in her car at a speed of 50 mph. Leaving at 8am, at what time will Alice reach Atlanta assuming that she had a flat tire that delayed her 30 minutes".

1. Read on the appendix about what *an obvious and clear* link is. [↑](#footnote-ref-1)