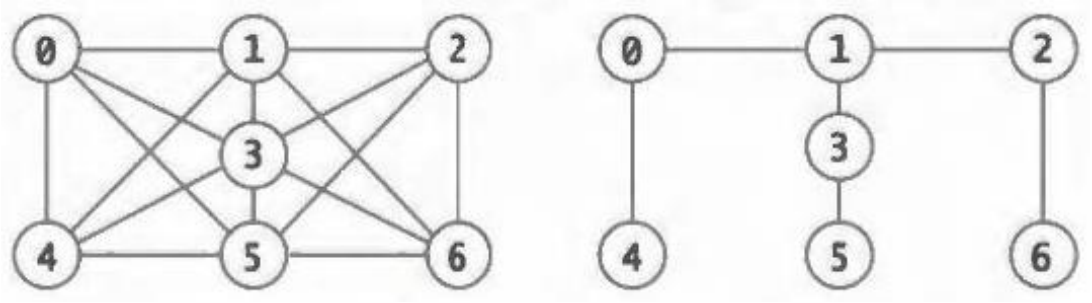
# GTU Department of Computer Engineering
# CSE 222/505 - Spring 2020
# Homework 8
# Due date: 14 June 2020 – 23:55

***This assignment consists of 3 questions.***

**Q1:**



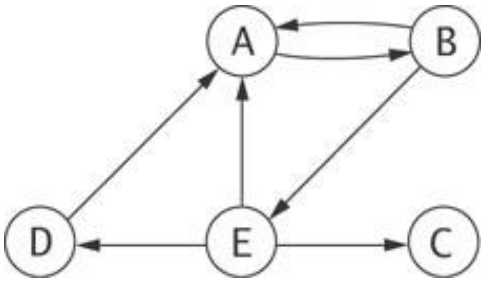

- Represent the graphs above using adjacency lists. Draw the corresponding data structure.
- Represent the graphs above using an adjacency matrix. Draw the corresponding data structure.
- For each graph above, what are the IVI=n, the IEI=m, and the density? Which representation is better for each graph? Explain your answers.
- Draw DFS tree starting from vertex 2 and traversing the vertices adjacent to a vertex in descending order (largest to smallest).
- Draw BFS tree starting from vertex 2 and traversing the vertices adjacent to a vertex in descending order (largest to smallest).

Show your work step by step and make your explanation. Write your solution by latex, word or handwriting (scan or take picture) and upload it as a single *StudentNumber.pdf* file.

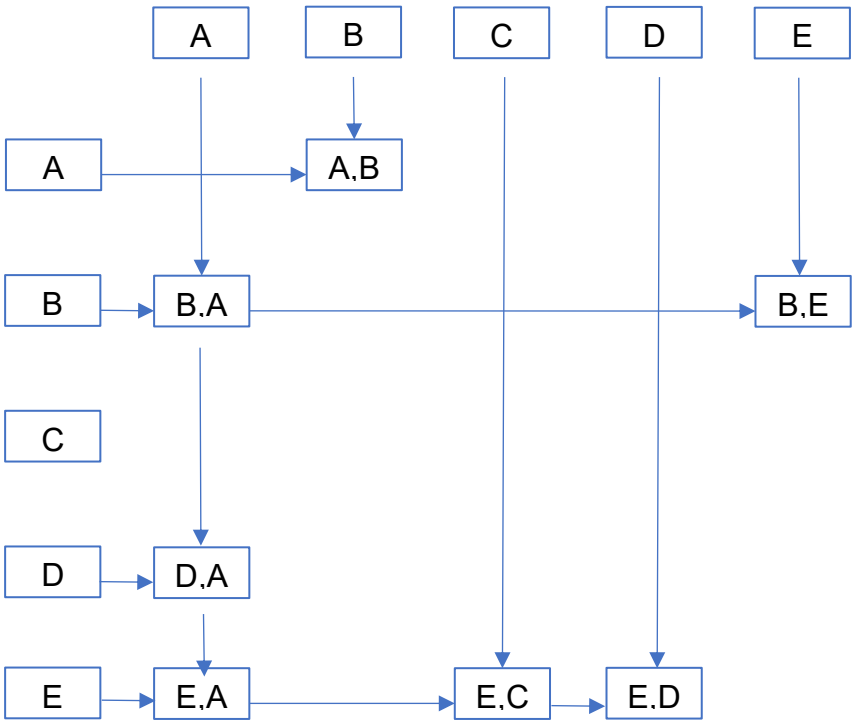
**Q2:**
Extend the graph ADT defined in the book so that it includes the following operations.

- Deletion of an individual edge.
- Insertion and deletion of an individual vertex. Give proper definition of the operations. Note that if you can delete a node, node IDs cannot be from 0…(n-1) anymore.
- Perform breadth-first search of the graph
- Perform depth-first search of the graph.

Implement the extended graph ADT using 2-D linked-list structure. In 2-D linked-list structure, each node has two row links (**rprev** to row-predecessor and **rnext** to row-successor) and two column links (**cprev** to column-predecessor and **cnext** to column-successor). In our graph representation, each row linked-list represents adjacent vertices to a vertex and each column linked-list represents vertices adjacent to a vertex. Consider the example graph below;




2-D linked-list representation is given below. Note that predecessor links are not shown for simplicity. The first column represents the source vertices (in accesding order) and the first row represents the destination vertices (in accesding order). Each edge is represented by a node which belong to two linked lists; a row linked-list and a column linked-list.

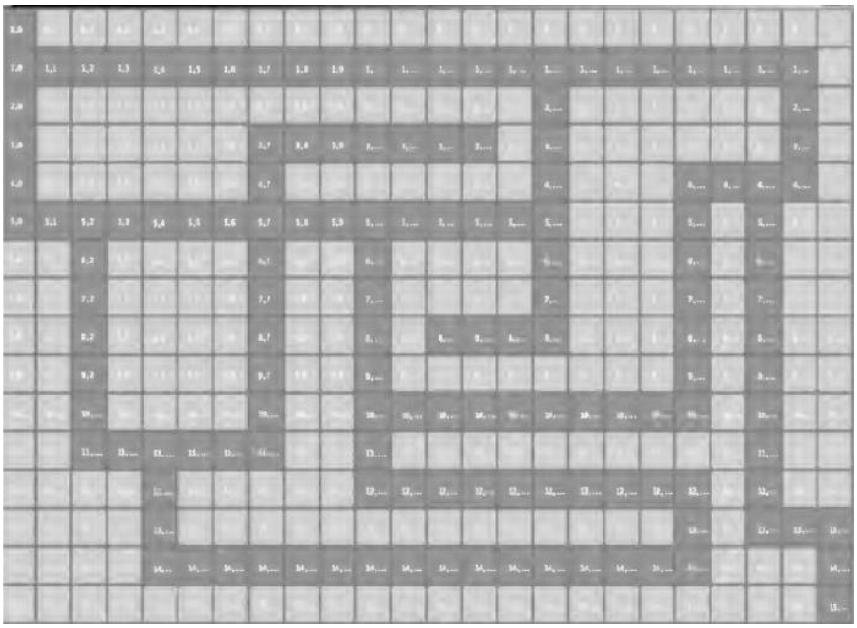



Write a program which tests your implementation using a randomly created directed and undirected graphs. You may create random adjacency matrices and convert it to your representation. You can also use random vertices and edges to test your operations.

**Q3:**

Create a MazeSolver which solves a maze. It should read a rectangular maze as a sequence of lines consisting of Os and 1s, where a 0 represents an open square and a 1 represents a closed one. You can find a maze below as an example. This maze created based on the below input file. After reading the input in this format, your program should convert it to a **weighted graph** (where the vertices are junction squares and the weight of an edge is the distance defined by the number of squares from the junction square represented by the source vertex to the next junction square represented by the destination vertex) and should find the shortest path from upper-left corner to lower-right corner. You should prefer the best graph representation for your graph and required operations.

```
011111111111111111111111
000000000000000000000001
011111111111111011111101
011111100000001011111101
011111101111111011000001
000000000000000011011011
110111101101111011011011
110111101101111011011011
110111101101000011011011
110111101101111111011011
110111101100000000011011
110000001101111111111011
111101111100000000001011
111101111111111111101000
111100000000000000001110
111111111111111111111110
```



**RESTRICTIONS:**

- Use only specified data types
- Can be only one main class in each question
- Don't use any other third part library

**GENERAL RULES:**

- For any question firstly use course news forum in Moodle, and then the contact TA.
- You can submit assignment one day late and will be evaluated over sixty percent (%40).

**TECHNICAL RULES:**

- Use given CSE222-VM to develop and test your Homeworks (your code must be working on CSE222-VM), CSE222-VM download link will be given on Moodle.
- Implement clean code standards in your code;
    - Classes, methods and variables names must be meaningful and related with the functionality.
    - Your functions and classes must be simple, general, reusable and focus on one topic.
    - Use standard java code name conventions.

**REPORT RULES:**

- Add all javadoc documentations for classes, methods, variables …etc. All explanation must be meaningful and understandable.
- You should submit your homework code, Javadoc and report to Moodle in a "studentid_hw3.tar.gz" file.
- Use the given homework format including selected parts from the table below:

| | |
|---|---|
| Detailed system requirements | |
| Use case diagrams (extra points) | |
| Class diagrams | X |
| Other diagrams | |
| Problem solutions approach | X |
| Test cases | X |
| Running command and results | X |

**GRADING :**

- **No OOP design: -100**
- **No interface: -95**
- **No method overriding: -95**
- **No error handling: -50**
- **No inheritance: -95**
- **No polymorphism: -95**
- No javadoc documentation: -50
- No report: -90
- Disobey restrictions: -100
- **Cheating : -200**
- Your solution is evaluated over 100 as your performance.

**CONTACT :**


- Teaching Assistant : Ümit Murat Akkaya
- Email: umit.akkaya@gtu.edu.tr