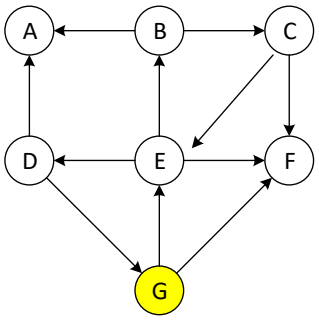
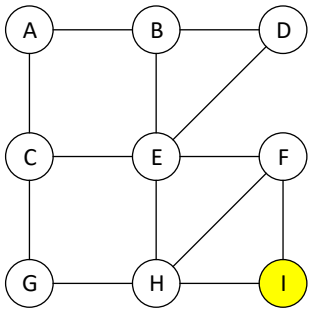
Assignment 6

Problem 1 (10 points). Run the DFS on the following graph beginning at node G and show the sequence of nodes generated by the search. When you have two or more choices as the next node to visit, choose them in the alphabetical order.

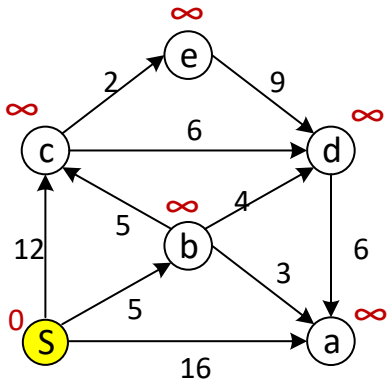


After completing the DFS, classify each edge as a *tree edge,* a *forward edge,* a *back edge,* or a *cross edge*.

Problem 2 (10 points). Run the BFS on the following graph beginning at node I and show the sequence of nodes generated by the search. When you have two or more choices as the next node to visit, choose them in the alphabetical order.

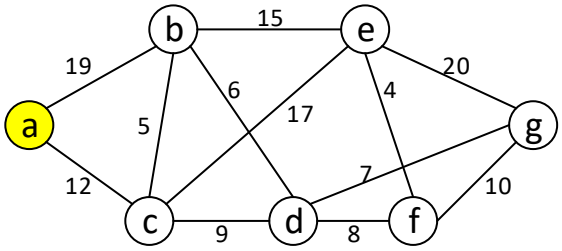


Problem 3 (10 points). Run the Dijkstra's algorithm on the following graph beginning at node S.



Problem 3-(1). After each iteration, show the D values of all nodes (initial D values are shown above each node in red).
Problem 3-(2). Show the shortest path from S to every other node generated by the algorithm

Problem 4 (10 points). Run the Prim-Jarnik algorithm on the following graph beginning at node *a*.



Problem 4-(1). Show the sequence of nodes in the order they are brought into the "cloud."
Problem 4-(2). Show the minimum spanning tree T, generated by the algorithm, as a set of edges.

Problem 5 (60 points) A friend relationship is a binary relation defined for two persons denoted *f*(*p*1, *p*2), where *p*1 and *p*2 are two persons. The meaning is that *p*1 is a friend of *p*2. Properties of the friend relationship are:

- If *f*(*p*1, *p*2), then *f*(*p*2, *p*1): If *p*1 is a friend of *p*2, then *p*2 is a friend of *p*1.
- If *f*(*p*1, *p*2) and *f*(*p*2, *p*3), then *f*(*p*1, *p*3): If *p*1 is a friend of *p*2 and *p*2 is a friend of *p*3, then *p*1 is a friend of *p*3.

You must write a program named *Hw6_P5.java* that implements the following requirements.

- Your program must read friend relationships among people from an input file named *friends_input.txt*. The format of an input file is shown below:

  John, Isobel
  Nathan, John
  Rebecka, Yorst
  Maria, Yorst

Doloris, Maria
Yorst, Doloris

- Your program must store the friend relationships in a (simplified) adjacency matrix, which is a two-dimensional matrix. For example, given the above friend relationships, the corresponding adjacency matrix should be:

	Doloris	Isobel	John	Maria	Nathan	Rebecka	Yorst
Doloris				1			1
Isobel			1				
John		1			1		
Maria	1						1
Nathan			1				
Rebecka							1
Yorst	1			1		1	

Note that in the matrix above, names of persons are shown instead of array indexes for easy illustration.

- Your program must print the adjacency matrix on the screen (to show that the adjacency matrix was correctly constructed), like the one shown above.
- Then, your program must display the following main menu:

Main Menu

Search options:
1. Friends of a person
2. Friend or not?
3. Exit

Enter option number:

- Your program must read an option a user enters and perform the following:
- If the option entered is smaller than 1 or greater than 3, then your program must issue an appropriate error message and redisplay the main menu.
- If the option entered is 1, then your program must prompt the user to enter the name of a person and perform the following
    - If the name does not exist in the adjacency matrix, then issue an appropriate error message and redisplay the main menu.
    - If the name exists, then display all friends of the person.
    - Redisplay the main menu.
- If the option entered is 2, then your program must prompt the user to enter names of two persons separated by a space and perform the following
    - If any of the two names does not exist in the adjacency matrix, then issue an appropriate error message and redisplay the main menu.
    - If both names exist, then display *Yes* if they are friends and *No* otherwise.

    - Redisplay the main menu
- If the option entered is 3, then your program must display an appropriate parting message on the screen and terminate the program.

Note that this program can be implemented using many data structures. However, for this assignment, you must use an adjacency matrix to store friend relationships and you may use other additional data structures if you want.

Deliverables

You must submit the following files:

You need to submit the following files:

- *Hw6_P1_P4.pdf*: This file must include:
  - Answers to problems 1 through 5.
  - Discussion/observation of Problem 6: This part must include what you observed and learned from this experiment and it must be "substantive."
- *Hw6_P5.java*
- Other files, if any.

Combine all files into a single archive file and name it *LastName_FirstName_hw6.EXT*, where *EXT* is an appropriate archive file extension, such as *zip* or *rar*.

Grading

- Problem 1 through Problem 4:
  - For each problem, up to 6 points will be deducted if your answer is wrong.

- Problem 5:
  - There is no one correct output. As far as your output is consistent with generally expected output, no point will be deducted. Otherwise, up to 36 points will be deducted.
  - If there are no sufficient inline comments, up to 10 points will be deducted.
  - If your conclusion/observation/discussion is not **substantive**, points will be deducted up to 10 points