5SENG001W Algorithms
Week 9
Tutorial Exercises: Graphs

These exercises cover: Representations of graphs, DFS and BFS traversal algorithms

You will also make use of David Galles' web based *Data Structure Visualizations* tools. During its use it may help to slow the animation right down so that you can follow the steps. You should also take screen shots and/or record your DFS and BFS example runs on at least 1 graph each.

Given the following definition of a graph G:

V = { 1, 2, 3, 4, 5, 6, 7 }

E = { (1,2), (1,3), (1,4), (2,5), (3,1), (3,5), (3,6), (4,6), (5,7), (6,3), (6,5), (6,7) }

Use the definition of **G** in the following exercises.

Exercise 1.

(a) Draw graph G, using either pen and paper or a software drawing tool, e.g. Draw.io https://app.diagrams.net/

(b) Create the **adjacency matrix** for graph G, using either pen and paper or a software tool.

(c) Create the **adjacency list** for graph G, using either pen and paper or a software tool.

Exercise 2.

(a) Create a Java or C++ program that implements your adjacency matrix for graph G. Test the program by printing out the adjacency matrix.

(b) Create a Java or C++ program or extend your program from (a), that implements your adjacency list for graph G. Test the program by printing out the adjacency list.

Exercise 3.

(a) Use David Galles' visualisation tools of the DFS and BFS algorithms explore how they work in practice, they are available at:

https://www.cs.usfca.edu/~galles/visualization/DFS.html

https://www.cs.usfca.edu/~galles/visualization/BFS.html

(b) Using either your adjacency matrix or adjacency list implementations of graph G, create a Java or C++ program that implements the **DFS traversal algorithm** using it. The "**ProcessVertex(sv)**" procedure should simply print out the value stored in the "sv" vertex.

(c) Using either your adjacency matrix or adjacency list implementations of graph G, create a Java or C++ program that implements the B**FS traversal algorithm** using it. The "**ProcessVertex(sv)**" procedure should simply print out the value stored in the "sv" vertex.

Exercise 4.

(a) Modify graph G to define a new graph **GW** by adding **weights** to each of the edges of graph G.

(b) Select one of your programs from **Exercise 3** for either (b) or (c). Extend and modify it so that it can deal with the weighted graph **GW**. For example, it should:

- Print out your chosen representation of GW including the **weights of the edges.**
- Depending on your chosen program, in either **dfs(sv)** or **bfs(sv),** after checking that an adjacent edge exists, it prints out that edge's details, i.e.

  edge = (source vertex, destination vertex, edge weight).

(c) Test your modified implementation by running your program on GW.