Data Structures & Algorithms

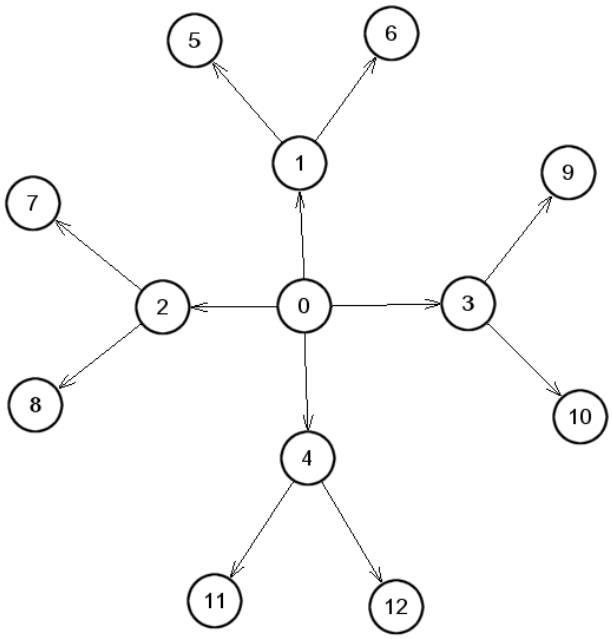
Lab Exercise 3 – Graph Project

Due: to be demonstrated by Thursday 13th November

Learning outcomes
At the end of this lab you should be able to:

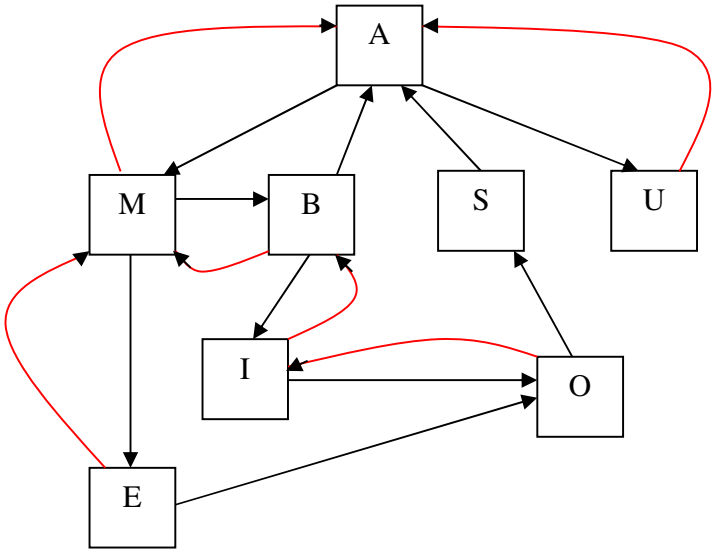
- Explain the architecture of the Graph implementation (i.e. classes Graph, GraphNode, GraphArc).
- Construct any directed graph using the Graph implementation.
- Implement the adapted breath first search algorithm as a member function of class Graph.

Q1. Use the `Graph` class to create a graph with the following structure:



Invoke the `breadthFirst()` function defined in the `Graph` class. Is the output correct?

Q2. Before attempting this question, review Chapter 2, Lecture 14 on blackboard, in particular the section on the Breadth-First Search algorithm adapted for pathfinding (referred to as adapted BFS below).
The adapted breadth-first search seeks a target node. As the neighbouring nodes of an origin node are discovered, a previous pointer is set from the neighbouring node back to the origin node.
Consider a short example where our search commences from node A, seeking target node O (previous pointers are illustrated in red)



Visit sequence:
M, U (immediate neighbours of A) *Set previous pointer from M toA and from U to A*
E, B (immediate neighbours of M) *Set previous pointer from E toM and from B to M*
I (immediate neighbour of B) *Set previous pointer from I toB*
O (immediate neighbour of E) *Set previous pointer from O toI*

The search terminates at the target node O.

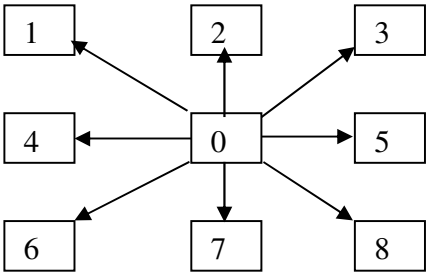
The search terminates when the target node is discovered. The previous pointers can be traversed from the target node O which will yield the path back to the origin node.

What are the nodes on this search path? (Ask for assistance if you are unsure).

(i) Now we will create a 5x5 grid, where each cell in the grid is represented as a GraphNode. For example, consider the innermost 9 cells of this grid:

	8	1	2	
	7	0	3	
	6	5	4	

Note that cell 0 has eight immediate neighbours numbered 0 to 8. This implies there should be an outgoing arc from cell 0 to each of these neighbouring cells:



Create the grid as described making sure to build the correct adjacency set for each of the 25 nodes. Store an integer as the data component of each GraphNode which will provide us with a way of identifying each cell. Use the following numbering system:

23	24	9	10	11
22	8	1	2	12
21	7	0	3	13
20	6	5	4	14
19	18	17	16	15

Note: The text file on blackboard "Q2Arcs.txt" has the correct adjacency set for nodes 0 to 19. Create the adjacency sets for the remaining nodes 20 to 24. Again, if you are unsure what to do ask for assistance.

(ii) The adapted BFS algorithm requires each node (cell) to store a pointer to the previous node used to reach it during traversal. Modify the GraphNode class to add this "previous" pointer field. Provide a way of initializing this pointer field in the constructor function. You should also provide member functions to read/set this pointer field.

(iii) Implement the adapted BFS algorithm as a new member function of the Graph class.

(iv) In the main function, test out your new BFS algorithm. Try searching for node 15, commencing the search from node 0. Print the final "path" used to visit node 15 (simply print the ID of the appropriate cells).