Department of Computing
BEng (Hons) Software Engineering

Module: 5SENG003C.2
Algorithms: Theory, Design and Implementation
Module Leader: Ragu Sivaraman

COURSEWORK REPORT

Name: Hasni Haleemdeen

Student ID: 20211337

UOW ID: W1898945

Level: 05

Tutorial Group: F/SE

A) A SHORT EXPLANATION OF YOUR CHOICE OF DATA STRUCTURE AND ALGORITHM.

Depth-first search is an algorithm for exploring or searching through tree or graph data structures. This algorithm starts at the root node and explores every branch as far as it can before turning around.

A directed graph is defined as G = (V, E), where V denotes the vertex set and E denotes the edge set. Assuming the graph is represented as an adjacency structure, all the vertices that can be reached by following an edge from vertex v are contained in the set adj(v). To execute a depth-first search on each vertex v, we save two pieces of data for each vertex v.

This categorization of the edges is not a unique aspect of the graph. Additionally, it relies on how the vertices are organized in adj(v) and in the loop that carries out the DFS procedure. To do a thorough search of the graph, it is not always necessary to use the num and mark fields. To achieve it, only a single bit that indicates whether a vertex has been examined previously is needed.

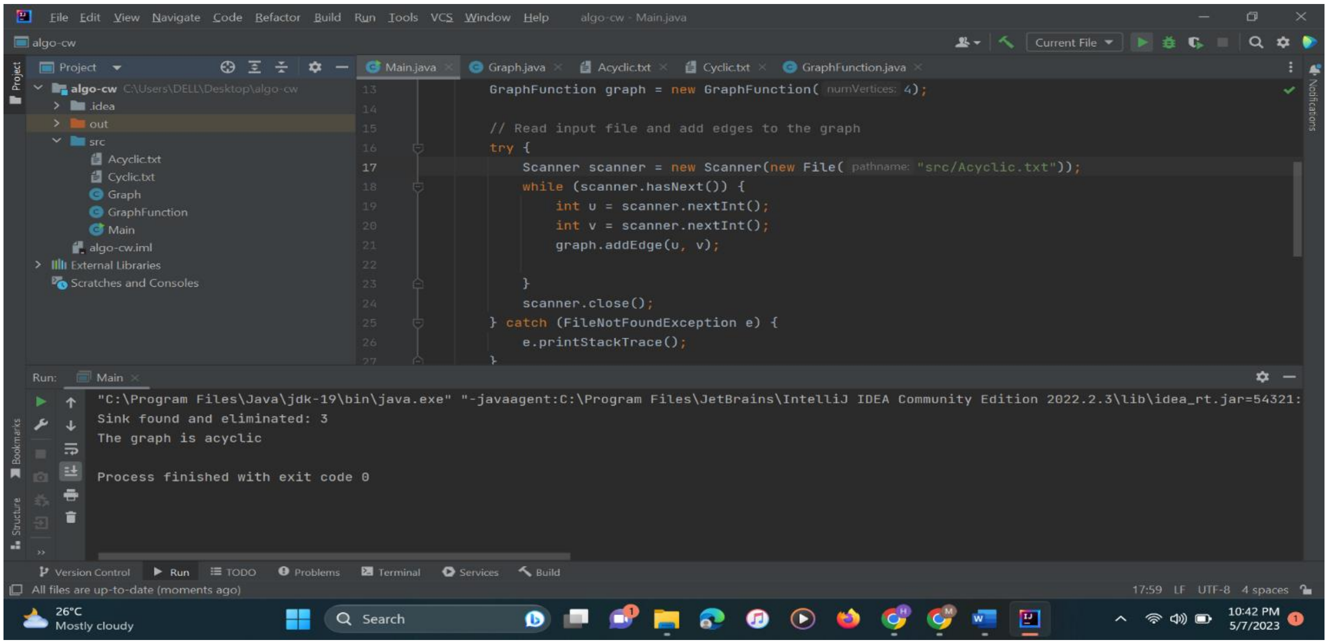
A cycle is obviously present if there is a back edge. To see if the graph contains a cycle, we can create a special version of DFS. The only reason we require the numbers is to check whether we have already visited a node.

Assume that the graph is acyclic. The arrangement of the vertices in this case is referred to as topological. This is a list of every vertex in the graph, ordered so that for any edge (u, v), vertex u comes before vertex v. One is said to have a reverse topological ordering if their reversal is a topological ordering. Therefore, for any edge (u, v), v comes before u in the reverse topological ordering.

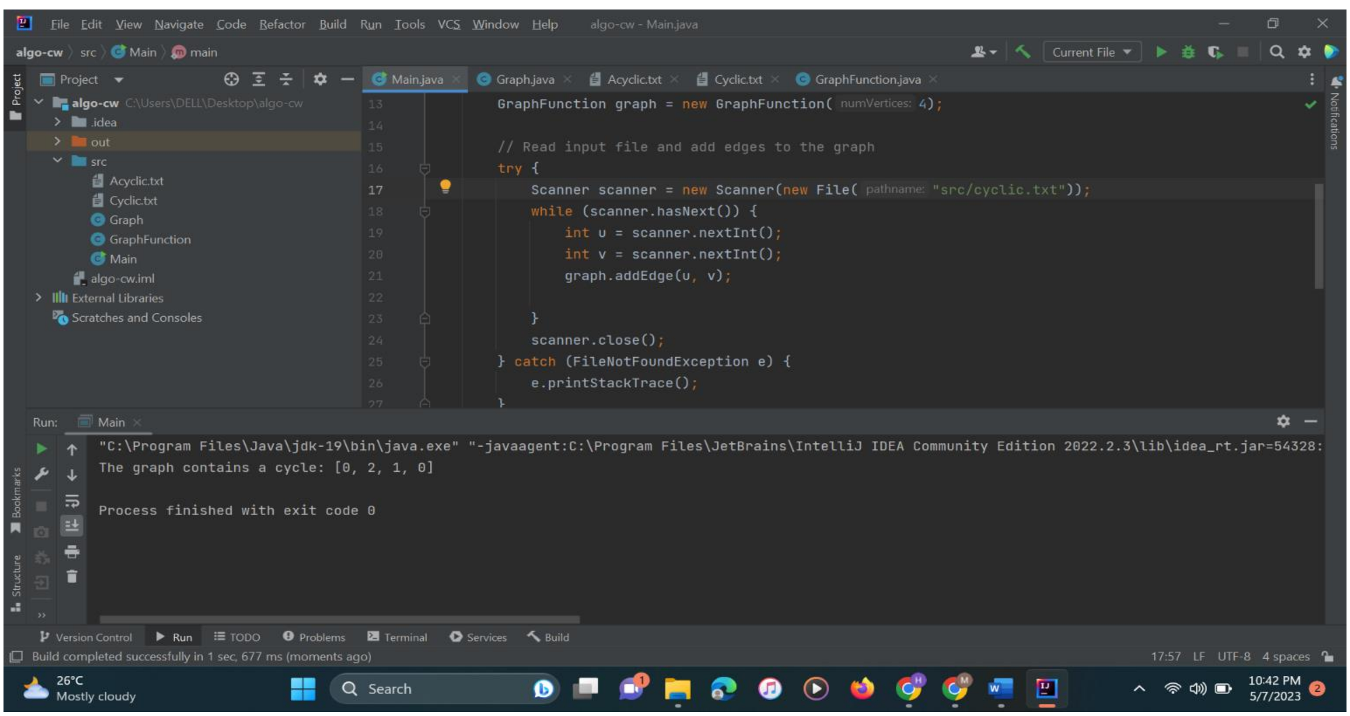
The DFS algorithm's time complexity is expressed as O (V + E), where V stands for the number of nodes and E for the number of edges. The algorithm's space complexity is O (V).

B) A RUN OF YOUR ALGORITHM ON TWO SMALL BENCHMARK EXAMPLES, ONE OF WHICH IS ACYCLIC AND THE OTHER ONE ISN'T. THIS SHOULD INCLUDE THE SUPPORTING INFORMATION AS DESCRIBED IN TASK 4.

ACYCLIC GRAPH



CYCLIC GRAPH



C) A PERFORMANCE ANALYSIS OF YOUR ALGORITHMIC DESIGN AND IMPLEMENTATION. THIS CAN BE BASED EITHER ON AN EMPIRICAL STUDY, E.G., DOUBLING HYPOTHESIS, OR ON PURELY THEORETICAL CONSIDERATIONS, AS DISCUSSED IN THE LECTURES AND TUTORIALS. IT SHOULD INCLUDE A SUGGESTED ORDER-OF-GROWTH CLASSIFICATION (BIG-O NOTATION).

Once a text file is inputted in the path, isACyclicGraph() method will check whether the given benchmark is a cyclic graph or not. If it's an Acyclic graph, it'll move onto sinkElimination() method, where it checks and removes the empty edges(no outgoing edges).

The input file is represented by the x axis in a performance analysis table, and the time is represented by the y axis. The algorithm complies to the theory of O(n) (n being the number of nodes). Time Complexity: O [N * M)

NM: This specifies how many rows and columns are involved. Since I and j go up to v, a depth first search suggests that the large O notation would be O(n2). The output displayed provides the result based on dynamically doubled inputs.

The results indicate that the method is naturally linear and that it has its own constraints, such as the length of time needed to search for locations that are farther away and have more barriers.

The primary benefit of choosing Depth First Search is that it will identify the graph's shortest route between the source and sink vertices. This BFS approach can therefore also be applied in this situation.