**Experiment 4:** Graph

**Student ID \_\_\_\_\_2023337621159\_\_\_\_\_\_ name:\_\_\_\_\_\_\_\_代翔\_\_\_\_\_\_\_\_\_\_\_**

**Purpose:**

1. Master the concepts of Graph;

2. Master some operations on Graph;

3. Understand the way of using different graph algorithms to solve problems.

**Problems and requirements:**

【Problem 1】: Is it a DAG?.

Given an unweighted directed graph, analyze the graph to determine whether it is a DAG.

If Yes, print out “yes” and the in-degree of each vertex in the graph.

If No, print out “No”, and show the vertices that form a cycle.

**Requirements:**

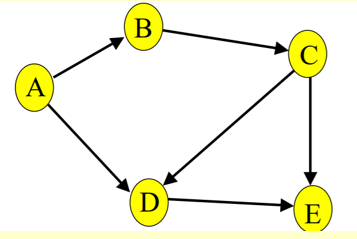
1. Suppose that each vertex takes a char data, such as ‘A’, ‘B’, ‘C’, etc.

2. Select the adjacency matrix or adjacency list to represent the graph;

3. Create and store a graph based on user input;

4. Please choose proper data structure to support your operations.

5. Please design two input graphs, one is acyclic, and the other one contains cycles. Then, demonstrate and explain your results based on these two inputs.



5 6

A B C D E

A->B

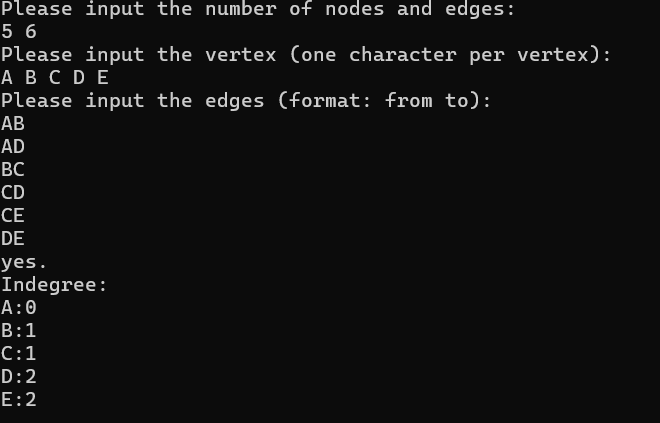
A->D

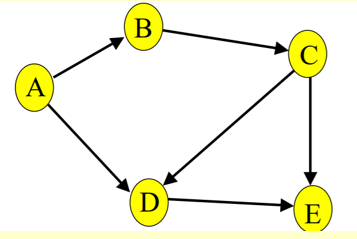
B->C

C->D

C->E

D->E





5 6

A B C D E

A->B

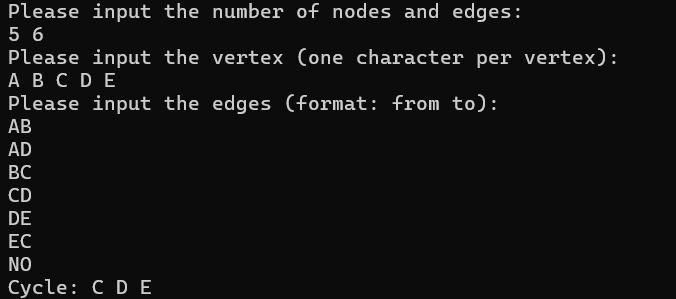
A->D

B->C

C->D

D->E

E->C



9 11

A B C D E F G H I

A->B

A->D

B->C

C->D

D->E

E->C

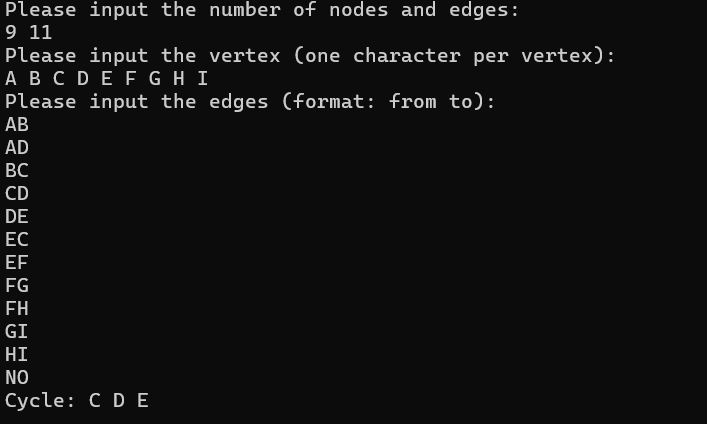
E->F

F->G

F->H

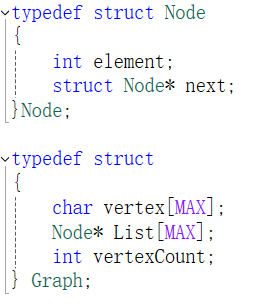
G->I

H->I

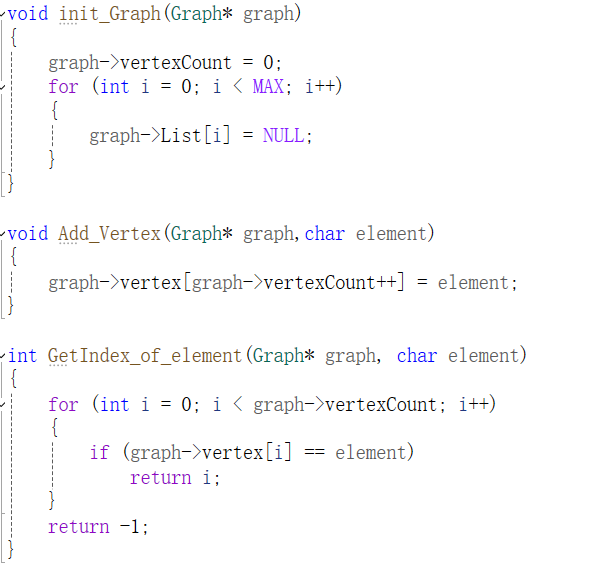


Method：

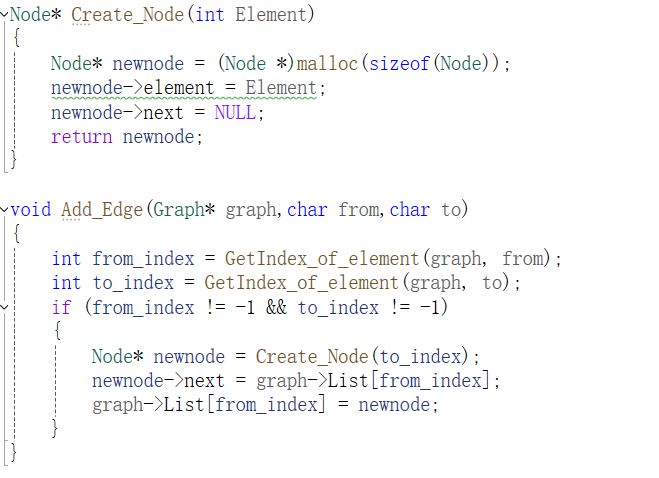
The core of the problem is to find out if the directed graph is a DAG, and to find the ring. To determine whether a graph is a directed graph, we can use topological sorting to determine whether it is a directed graph. In the process of finding rings, we can use the DST method to find them.



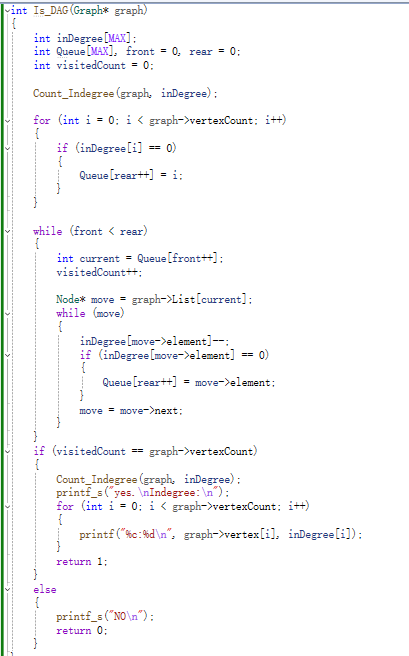
This code is used to define the data types Node and Graph, since the value of the node is a string this time, in order to facilitate the subsequent access to the nodes of the graph, I define a string array in the graph, which is used to establish the relationship between index and characters, so that the element in the Node is set to int, List is used to store edges, and vertexCount is used to record the number of nodes



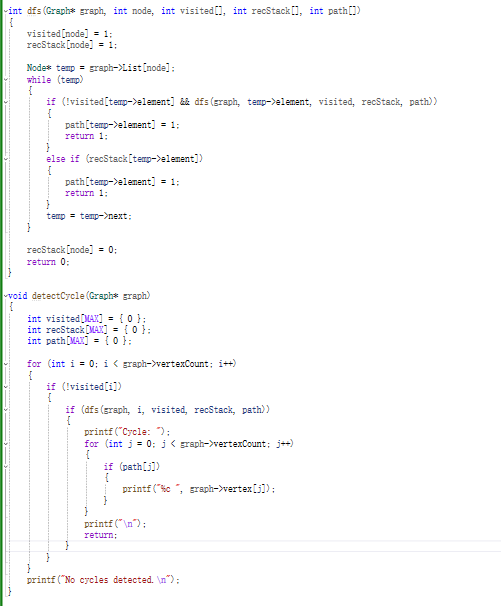
These codes are used to initialize the graph, add character elements, and establish the relationship between characters and index



These codes are used to create nodes and add relationships to edges



This code is used to determine whether this graph is a DAG graph, first establish an array indegree to store the in-degree of each node in the graph, and then establish a Queue for managing nodes, and use visitedCount to record how many nodes are visited now, if visitedCount is equal to the number of nodes in the graph, then it is a DAG, otherwise there is a loop, we let the in-degree be 0 in turn and then out of the stack, and let their neighbor's ins minus one, and then let the zero ins again, until there are no nodes in the stack.



detectCycle is used to output rings in the graph, dfs is used to deeply traverse to query and record rings into the path array, first, dfs is used to determine whether there are rings in it, and if there are rings, return 1, Otherwise, 0 is returned. When performing dfs traversal, use visited to store whether this node has been visited, and recStack is used to store the nodes in the path, if this node has not been visited and there is a ring in the next node, we will put the current node into the path, if this section has no loop or this node has been visited, we will go to recStack to check whether it is already in the path, if it is in the path, then the ring has been formed, and we will put the node into the path. If nothing happens, retrace to the previous node again and return to 0.