Brandon Perry

Coding Assignment #4

CS 221

Dr. Troung Tran

Documentation

**GraphType**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Requirements**

***CRC Card***

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: GraphType | Superclass: | | Subclasses: |
| Primary Responsibility: | | | |
| Responsibilities | | Collaborations | |
| Initialize | |  | |
| Add vertex to graph (vertex) | | VertexType | |
| Add edge to graph (vertex, vertex, weight) | | VertexType | |
| Get a queue of to vertices (vertex, queue) | | VertexType, Queue | |
| Clear marked vertices | |  | |
| Mark a vertex | | VertexType | |
| Return true if vertex is marked | | VertexType | |
| Index position of vertex | | VertexType | |

***ADT Specs***

Structure: The graph consists of a set of vertices and a set of weighted edges that connect some of or all the vertices to one another

Assumption: before any call is made to a graph operation, the graph has been declared and a constructor has been applied

Definition: const int NULL\_EDGE = 0

Operations:

GraphType GraphType()

Function: Default constructor for GraphType

Pre: None

Post: Parameters and variables are initialized

GraphType ~GraphType()

Function: Default deconstructor for GraphType

Pre: none

Post: arrays are deleted

Void AddVertex (VertexType vertex)

Function: Adds vertex to the graph

Pre: Graph is not full

Post: Vertex is in V(graph)

Void AddEdge (VertexType fromVertex, VertexType toVertex, int weight)

Function: Adds an edge with the specified weight from fromVertex to toVertex

Pre: fromVertex and toVertex are in V(graph)

Post: (fromVertex, toVertex) is in E(graph) with the specified weight

Void GetToVertices (VertexType vertex, Queue& vertexQ)

Function: Returns a queue of the vertices that are adjacent from vertex

Pre: vertex is in V(graph)

Post: vertex contains the names of all vertices that are adjacent form vertex

Void ClearMarks ()

Function: Sets marks for all vertices to false

Pre: none

Post: All marks have been set to false

Void MarkVertex (VertexType vertex)

Function: Sets mark for vertex to true

Pre: vertex is in V(graph)

Post: IsMarked(vertex) is true

Bool IsMarked (VertexType vertex)

Function: Determines if vertex has been marked

Pre: vertex is in V(graph)

Post: Function value = (vertex is marked true)

Int IndexIs (VertexType\* vertices, VertexType vertex)

Function: Returns the index of vertex in vertices

Pre: vertex is in vertices

Post: returns the index

***UML DIAGRAM***

|  |
| --- |
| GraphType |
| -numVertices: int  -maxVertices: int  -vertices: VertexType\*  -edges[7][7]: int  -marks: bool\* |
| +GraphType ()  +~GraphType ()  +AddVertex (vertex: VertexType): void  +AddEdge (startVertex: VertexType, endVertex: VertexType, weight: int): void  +GetToVertices (toVertex: VertexType, endVertex: queue<VertexType>&): void  +ClearMarks (): void  +MarkVertex (vertex: VertexType): void  +IsMarked (vertex: VertexType): bool  +IndexIs (vertices: VertexType\*, vertex: VertexType): int; |

Test Plan

Intentions:

This test plan is to test the GraphType class as well as the Depth First Search algorithm. Seven vertices will be added calling AddVertex seven times. Then, AddEdge will be called ten times in order to create a graph. Before the main function ends, the Depth First Search algorithm is called and executed to a file. startVertex and endVertex are redefined right before it is called in order to create the proper path.

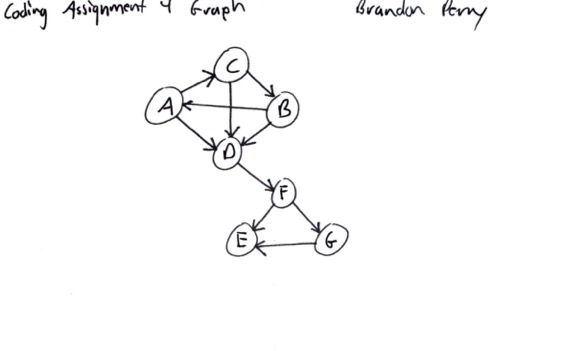
Input File:

|  |
| --- |
| Input File: Input |
| AddVertex A  AddVertex B  AddVertex C  AddVertex D  AddVertex E  AddVertex F  AddVertex G  AddEdge A C 1  AddEdge A D 1  AddEdge B A 1  AddEdge B D 1  AddEdge C B 1  AddEdge C D 1  AddEdge D F 1  AddEdge F E 1  AddEdge F G 1  AddEdge G E 1  Quit |

Expected Output File:

|  |
| --- |
| OutPut File: Output |
| ADFE |

Hand Drawn Graph:



Results:

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Seventeen commands were executed which is correct. Program executed with input and output to the proper file.

Error Analysis:

In the output file, ADFGE is the result rather than ADFE. I believe that ADFE should be the expected output. I am unsure as to what exactly is causing this issue and making it go to G rather than straight to E.

One error I faced during the coding process what the deconstructor making the program fail. This was because I believe either multiple redefinitions of the arrays within or other functions access to them. Again, I am not entirely sure. I commented out the code (as noted in the program) in order to make it run without crashing.