

User Manual

User Manual

Graph

This program reads in a file that contains vertex names with the respective edges and weights. Each vertex is separated by #. The program opens a menu to allow the user to read in this formatted file as well as offering extensive options for interacting with the graph. Vertices and edges can be added and deleted. The graph may be printed simply or by using a breadth first traversal or a depth first traversal. The shortest path between any two vertices can also be calculated as well as the path taken.

1. Executing the program

Turn on and boot your computer.

Insert the flash drive containing the program file Project3.exe in the appropriate drive. (We will assume it is the A drive, but it could be any drive. Substitute the appropriate letter where you see a:)

Enter the following at the prompt.

a:Project1

The menu will appear, along with a prompt allowing you to input your choice for the menu options.

2. Input

2.1 Input Requirements

The input file consists of VertexName EdgeName Weight EdgeName Weight # with the # denoting the end of the edge list.

The main menu options require you to enter an integer associated with an option. The number will be next to the option it performs.

Most options will ask for either one or two vertex names. These names must be typed exactly as they appear in the file or were entered previously. They are case sensitive.

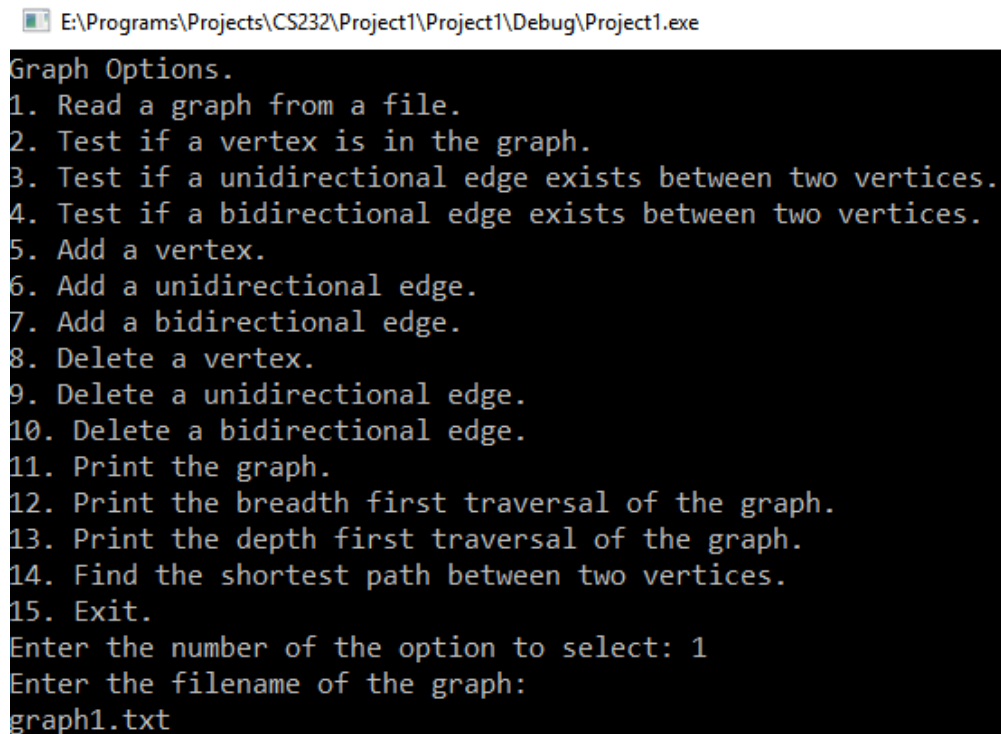
2.2 Input Restrictions

The input file must be formatted in the proper way. Any extraneous data in the file will cause the program to behave incorrectly. When prompted for a vertex name, the name must be a vertex that exists in the graph. If it does not exist, you will be taken back to the main menu.

3. Output

For the functions that add or remove vertices or edges, the program will output if it successfully completed that action or not. When the graph is printed, it will be printed by each vertex with its associated edgelist. The breadth first and depth first traversals will print out the starting vertex and then the vertices visited on the traversal. The shortest path option will print out the path taken from the original vertex to the final vertex as well as the distance of this path.

Example run



The screenshot shows a Windows command prompt window titled "E:\Programs\Projects\CS232\Project1\Project1\Debug\Project1.exe". The window contains a list of 15 menu options for a graph application. The user has entered '1' for the first option and 'graph1.txt' for the filename.

```
E:\Programs\Projects\CS232\Project1\Project1\Debug\Project1.exe
Graph Options.
1. Read a graph from a file.
2. Test if a vertex is in the graph.
3. Test if a unidirectional edge exists between two vertices.
4. Test if a bidirectional edge exists between two vertices.
5. Add a vertex.
6. Add a unidirectional edge.
7. Add a bidirectional edge.
8. Delete a vertex.
9. Delete a unidirectional edge.
10. Delete a bidirectional edge.
11. Print the graph.
12. Print the breadth first traversal of the graph.
13. Print the depth first traversal of the graph.
14. Find the shortest path between two vertices.
15. Exit.
Enter the number of the option to select: 1
Enter the filename of the graph:
graph1.txt
```

```
Graph Options.
1. Read a graph from a file.
2. Test if a vertex is in the graph.
3. Test if a unidirectional edge exists between two vertices.
4. Test if a bidirectional edge exists between two vertices.
5. Add a vertex.
6. Add a unidirectional edge.
7. Add a bidirectional edge.
8. Delete a vertex.
9. Delete a unidirectional edge.
10. Delete a bidirectional edge.
11. Print the graph.
12. Print the breadth first traversal of the graph.
13. Print the depth first traversal of the graph.
14. Find the shortest path between two vertices.
15. Exit.
Enter the number of the option to select: 11
```

Vertex: Atlanta
->(Austin with weight 475)
->(Buffalo with weight 550)
->(Chicago with weight 925)
->(Dallas with weight 625)
->(Denver with weight 800)
->(Houston with weight 650)
->(Newark with weight 725)
->(Washington with weight 600)

Vertex: Austin
->(Buffalo with weight 825)

Vertex: Buffalo
->(Chicago with weight 625)

Vertex: Chicago
->(Dallas with weight 1500)
->(Denver with weight 550)
->(New_York with weight 950)

Vertex: Dallas
->(Denver with weight 1250)
->(Chicago with weight 1500)

Vertex: Denver
->(Houston with weight 875)
->(Chicago with weight 550)

Vertex: Houston
->(Newark with weight 1050)

Vertex: Newark
->(New_York with weight 50)

Vertex: New_York
->(Washington with weight 550)
->(Buffalo with weight 450)
->(Chicago with weight 2000)

Vertex: Washington
->(Austin with weight 725)

Graph Options.

1. Read a graph from a file.
2. Test if a vertex is in the graph.
3. Test if a unidirectional edge exists between two vertices.
4. Test if a bidirectional edge exists between two vertices.
5. Add a vertex.
6. Add a unidirectional edge.
7. Add a bidirectional edge.
8. Delete a vertex.
9. Delete a unidirectional edge.
10. Delete a bidirectional edge.
11. Print the graph.
12. Print the breadth first traversal of the graph.
13. Print the depth first traversal of the graph.
14. Find the shortest path between two vertices.
15. Exit.

Enter the number of the option to select: 5

Enter the name of the vertex to add (-1 to cancel): Los_Angeles

Los_Angeles added to the graph.

Graph Options.

1. Read a graph from a file.
2. Test if a vertex is in the graph.
3. Test if a unidirectional edge exists between two vertices.
4. Test if a bidirectional edge exists between two vertices.
5. Add a vertex.
6. Add a unidirectional edge.
7. Add a bidirectional edge.
8. Delete a vertex.
9. Delete a unidirectional edge.
10. Delete a bidirectional edge.
11. Print the graph.
12. Print the breadth first traversal of the graph.
13. Print the depth first traversal of the graph.
14. Find the shortest path between two vertices.
15. Exit.

Enter the number of the option to select: 6

Enter the names of the two vertices to add a unidirectional edge between. (-1 to cancel)

Starting vertex: New_York

Ending vertex: Los_Angeles

Enter the distance between the two vertices: 2000

Unidirectional edge added between New_York and Los_Angeles

Graph Options.

1. Read a graph from a file.
2. Test if a vertex is in the graph.
3. Test if a unidirectional edge exists between two vertices.
4. Test if a bidirectional edge exists between two vertices.
5. Add a vertex.
6. Add a unidirectional edge.
7. Add a bidirectional edge.
8. Delete a vertex.
9. Delete a unidirectional edge.
10. Delete a bidirectional edge.
11. Print the graph.
12. Print the breadth first traversal of the graph.
13. Print the depth first traversal of the graph.
14. Find the shortest path between two vertices.
15. Exit.

Enter the number of the option to select: 14

Enter the names of the two vertices to find the shortest path between them. (-1 to cancel)

Vertex 1: Austin

Vertex 2: Los_Angeles

The shortest path from Austin to Los_Angeles is:

(Austin)

->(Buffalo)

->(Chicago)

->(New_York)

->(Los_Angeles)

The shortest distance between Austin and Los_Angeles is 4400

Graph Options.

1. Read a graph from a file.
2. Test if a vertex is in the graph.
3. Test if a unidirectional edge exists between two vertices.
4. Test if a bidirectional edge exists between two vertices.
5. Add a vertex.
6. Add a unidirectional edge.
7. Add a bidirectional edge.
8. Delete a vertex.
9. Delete a unidirectional edge.
10. Delete a bidirectional edge.
11. Print the graph.
12. Print the breadth first traversal of the graph.
13. Print the depth first traversal of the graph.
14. Find the shortest path between two vertices.
15. Exit.

Enter the number of the option to select:

Graph Options.

1. Read a graph from a file.
2. Test if a vertex is in the graph.
3. Test if a unidirectional edge exists between two vertices.
4. Test if a bidirectional edge exists between two vertices.
5. Add a vertex.
6. Add a unidirectional edge.
7. Add a bidirectional edge.
8. Delete a vertex.
9. Delete a unidirectional edge.
10. Delete a bidirectional edge.
11. Print the graph.
12. Print the breadth first traversal of the graph.
13. Print the depth first traversal of the graph.
14. Find the shortest path between two vertices.
15. Exit.

Enter the number of the option to select: 8

Enter the name of the vertex to delete (-1 to cancel): Buffalo

Buffalo deleted.

Graph Options.

1. Read a graph from a file.
2. Test if a vertex is in the graph.
3. Test if a unidirectional edge exists between two vertices.
4. Test if a bidirectional edge exists between two vertices.
5. Add a vertex.
6. Add a unidirectional edge.
7. Add a bidirectional edge.
8. Delete a vertex.
9. Delete a unidirectional edge.
10. Delete a bidirectional edge.
11. Print the graph.
12. Print the breadth first traversal of the graph.
13. Print the depth first traversal of the graph.
14. Find the shortest path between two vertices.
15. Exit.

Enter the number of the option to select: 12

Breadth first traversal:

Enter the name of the vertex to start at (-1 to cancel): New_York

The Breadth First Traversal of the graph is:

(New_York)

->(Washington)
->(Chicago)
->(Los_Angeles)
->(Austin)
->(Dallas)
->(Denver)
->(Houston)
->(Newark)
->(Atlanta)

Graph Options.

1. Read a graph from a file.
2. Test if a vertex is in the graph.
3. Test if a unidirectional edge exists between two vertices.
4. Test if a bidirectional edge exists between two vertices.
5. Add a vertex.
6. Add a unidirectional edge.
7. Add a bidirectional edge.
8. Delete a vertex.
9. Delete a unidirectional edge.
10. Delete a bidirectional edge.
11. Print the graph.
12. Print the breadth first traversal of the graph.
13. Print the depth first traversal of the graph.
14. Find the shortest path between two vertices.
15. Exit.

Enter the number of the option to select: 10

Enter the names of the two vertices to delete a bidirectional edge between. (-1 to cancel):

Vertex 1: Chicago

Vertex 2: Dallas

Edge deleted between Chicago and Dallas

Graph Options.

1. Read a graph from a file.
2. Test if a vertex is in the graph.
3. Test if a unidirectional edge exists between two vertices.
4. Test if a bidirectional edge exists between two vertices.
5. Add a vertex.
6. Add a unidirectional edge.
7. Add a bidirectional edge.
8. Delete a vertex.
9. Delete a unidirectional edge.
10. Delete a bidirectional edge.
11. Print the graph.
12. Print the breadth first traversal of the graph.
13. Print the depth first traversal of the graph.
14. Find the shortest path between two vertices.
15. Exit.

Enter the number of the option to select: 8

Enter the name of the vertex to delete (-1 to cancel): Tokyo

Tokyo is not in the graph.