**SHORTEST PATH PROBLEM**

**by:**

Rachel Hera :

Beauty Tatenda Tasara : 2201849402

**ABSTRACT**

Graphical representation of networks is used to solve many real-life problems.The networks may include paths in a city or telephone network or circuit network. One of the problems related to the use of graphs is to find the shortest path between two nodes(eg cities) in the network considering the weight between the nodes (which may be the distance between the two cities or the condition of the road or the traffic in the way).In this documentation we find a solution to calculate the shortest path between a source vertex and every other vertex in a weighted directed graph. The solution uses Dijkstra’s Algorithm to calculate the distance and it also determines the path with the minimum weight from the source vertex to the destination.The results are saved in a csv file showing the source vertex , destination vertex ,the weight between the two and finally the path (consisting of vertices visited before the destination is reached). The csv file is used in a python script to visualize the shortest paths using *networkx* and *matplotlib*.

## 

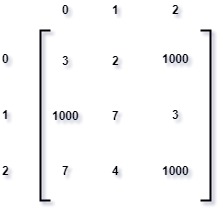
## **CHOICE OF DATA STRUCTURE**

## Graph theory is the common way used to represent network systems since the concept of Eulerian Graph constructed by Euler in 1735 . There are several types of software developed from graph theory today which are used in representation and visualisation of networks.

## **Used data structure**

In our implementation we used adjacency matrix representation to solve our problem. It is a 2d array with size where is the number of nodes of the graph.In the following example the entries with less than 1000 represents an edge between the vertices at the top and left side ,in our implementation 1000 represents infinity, meaning there is no edge between the two vertices.

Example:



Advantages

* Representation is easier to implement and follow.
* Removing an edge takes O(1) time.
* Queries like whether there is an edge from vertex ‘u’ to vertex ‘v’ are efficient and can be done in O(1).

Disadvantages

* Consumes more space O(V^2). Even if the graph is sparse(contains less number of edges), it consumes the same space.
* Adding a vertex takes O(V^2) time.

**Alternative data structure**

The other alternative was to use adjacency list .

example:(from <https://www.geeksforgeeks.org/graph-and-its-representations/>)

Adjacency List Representation of Graph

In this representation an array of pointers to linked list is used, a node in the linked list contains the vertex adjacent to the array indexes on the left and pointer to the next node .

Advantages

* Reduces the time complexity of dijkstra’s algorithm to O(E log N) with the help of binary heap(Min heap).
* Reduces the memory usage in the case of a sparse network. Memory is O(N + M) where N is the number of nodes and M is the number of edges. Worst case O(N^2) which will be similar to matrix representation.

Disadvantages

* Queries like whether there is an edge from vertex u to vertex v are not efficient and can be done O(V).
* It is more complicated to implement.

**Conclusion**

Our implementation would become better in the case of a dense network as listing all the adjacent edges would make the list long and the time taken to traverse the list would be O(N) unless a hash is used, therefore a matrix representation would condense the graph . In general, adjacency lists offer the fastest running time (especially for sparse graphs), while adjacency matrices are easier to implement.

## **THE GRAPH**

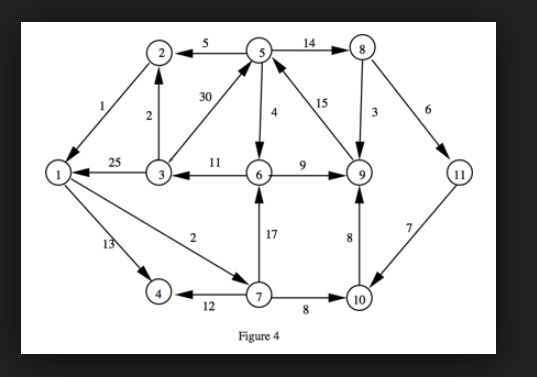
Graphs are structures that map relations between objects. The objects are referred to as nodes and the connections between them as edges in this documentation.

The starting graph is directed. That is, edges have an orientation: they are unidirectional.(eg A--->B != B--->A.

The graph is also an edge-weighted graph where the distance between each pair of adjacent nodes represents the weight of an edge. This is handled as an edge attribute named “weight”.

The graph is created from a txt file and has the properties : vertex, adjacent vertex and weight.

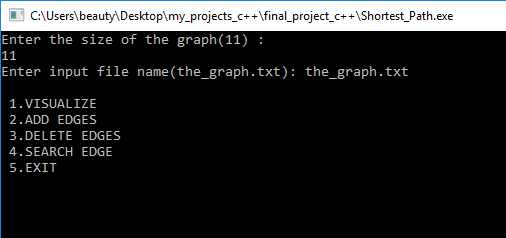
Picture from(google)

******

**PROGRAM MANUAL**

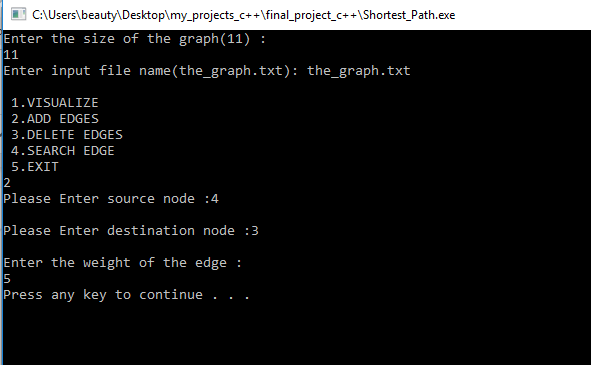
When you click the exe file, a window like the one below opens and you enter the graph size, and the filename from where you retrieve your graph data.

Below is a menu that you can choose from.

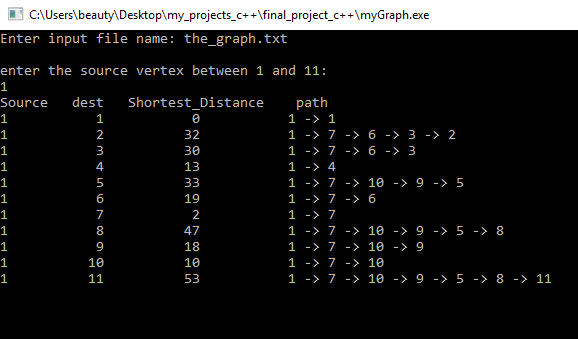


**Add edge:**

You can add an edge by specifying the source vertex , the destination vertex and the weight associated with the edge.

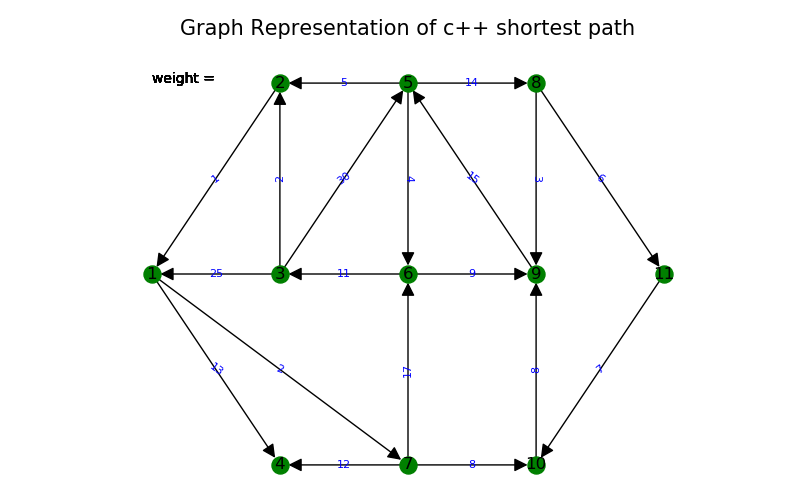


If the choice is 1 : When the correct entry is made the screen is updated and the shortest distance from the source vertex is shown as well as the path followed to get to the destination vertex.

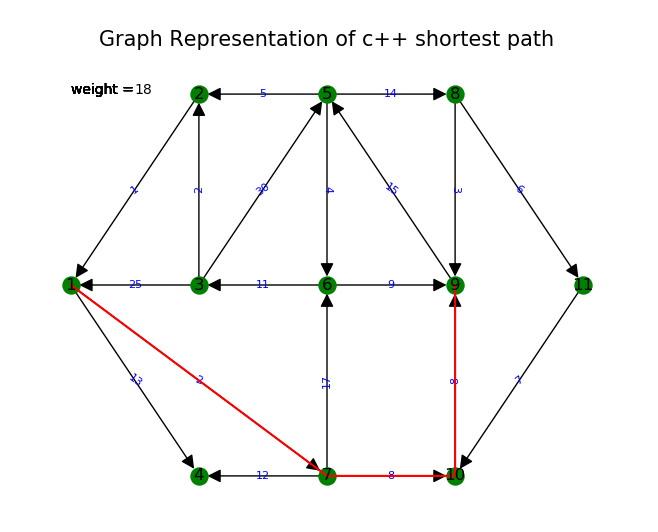


**Visualization**

After 3 seconds a matplotlib screen of the graph pops up and the paths followed to the destination vertices from the source are shown in turns on the graph.(the paths are in different colors to avoid confusion when the screen is updated).



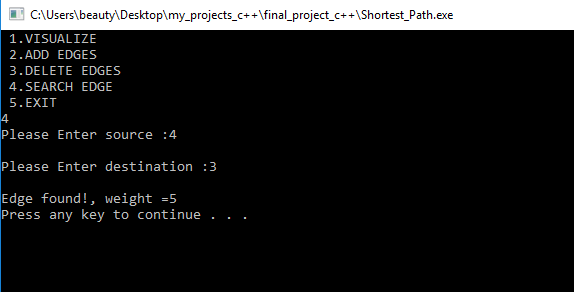
The screen updates every 3 seconds therefore it begins showing the paths from the source to nodes 1 through 11.In the image below the source vertex is 1 and the destination vertex is 9.The path followed to the vertex 9 is shown in red and the weight of the path is shown on the top left of the graph.



In some cases where there is no path from the source vertex to the destination vertex the weight on the graph is shown as 1000 which represents infinity. Also when the calculated shortest paths are shown infinity is represented by 1000.

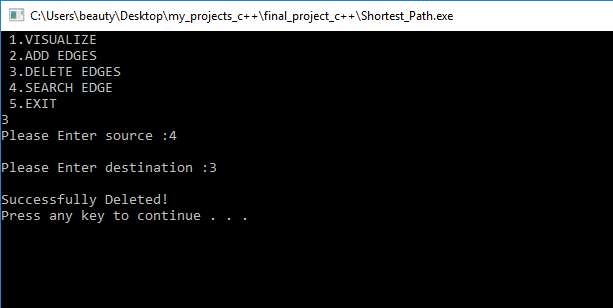
**To search for an edge**

You choose option 4 and enter the source and destination vertices, a message will be shown to notify if the edge is found or not.



**To delete an edge**

You choose option 3and enter the source and destination vertices, a message will be shown to notify if the edge is deleted or not.



You can repeat visualization on another source by (Y or y) or cancel by (N or n). Any other key is invalid and you are prompted again to make a choice.

Link to the application demo video:

Link to the git website: <https://github.com/beautytasara27/ShortestPathCPP>