

**The Final Project Task**

**Maximum Flow and Dijkstra Algorithm**

**Third Year- Computer Science**

**Spring Semester**

**Submitted to:**

**Dr. Mostafa Reda El-Tantawi**

**Team members:**

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Email** |
| Ahmed Sayed Mansour | 20170022 | ahmed.mans20719@gmail.com |
| Abdelrahman Bahig | 20170143 | abdelrahmanbahig04@gmail.com |
| Ibrahim Ramadan Abdou | 20170002 | ibrahemramadan130@gmail.com |
| Atef Magdy Metwally | 20170136 | atefmagdy12@gmail.com |
| Hatem Sayed Ali | 20170084 | hatemgad98@gmail.com |

* **Introduction :**

One algorithm for finding the shortest path from a starting node to a target node in a weighted graph is **Dijkstra’s algorithm**. The algorithm creates a tree of shortest paths from the starting vertex, the source, to all other points in the graph.

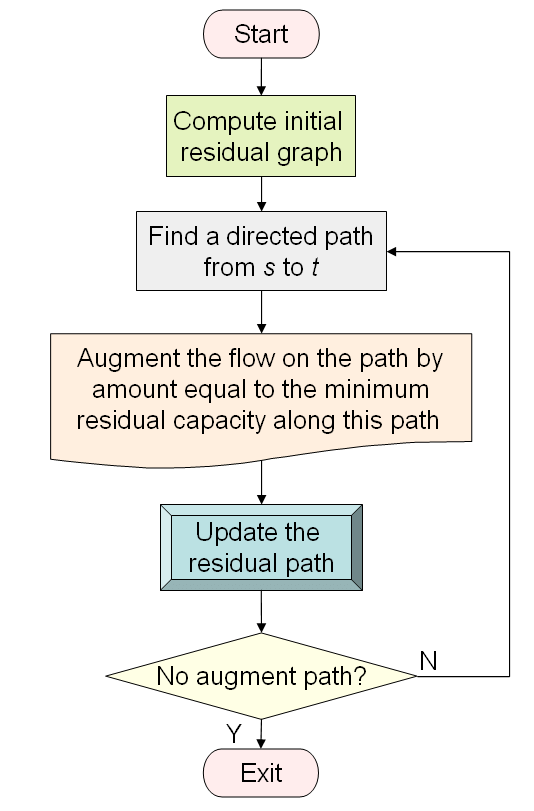
**Dijkstra’s algorithm**, published in 1959 and named after its creator Dutch computer scientist Edsger Dijkstra, can be applied on a weighted graph. The graph can be either directed or undirected. One stipulation to using the algorithm is that the graph needs to have a nonnegative weight on every edge.

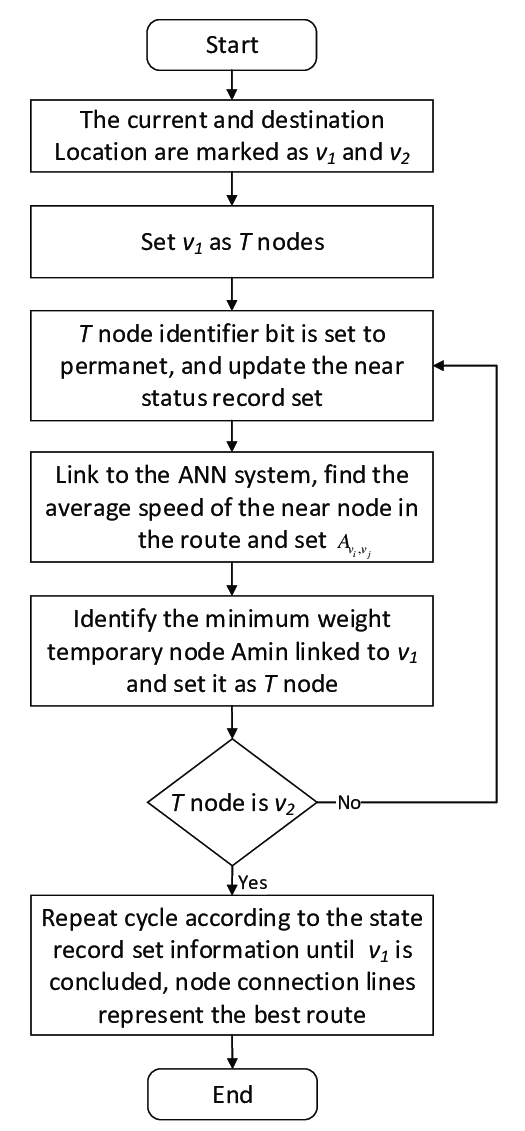
**Maximum flow problems** involve finding a feasible flow through a flow network that obtains the maximum possible flow rate.

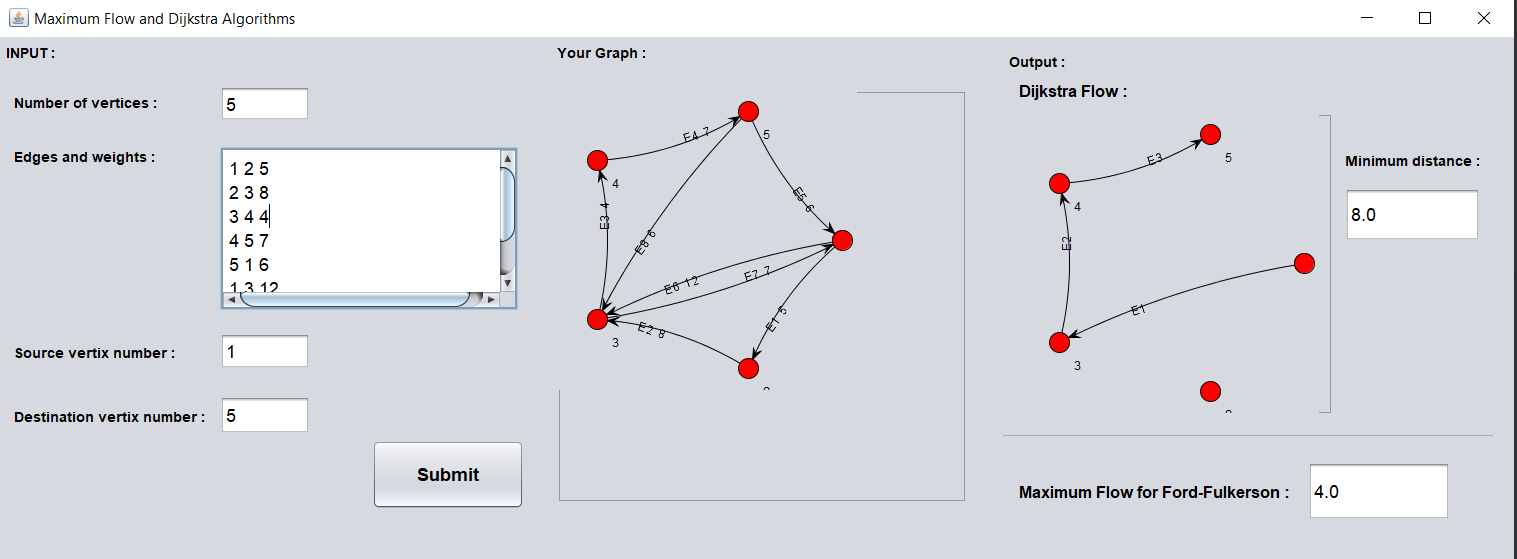
**The maximum flow problem** can be seen as a special case of more complex network flow problems, such as the circulation problem. The maximum value of an s-t flow (i.e., flow from source s to sink t) is equal to the minimum capacity of an s-t cut (i.e., cut severing s from t) in the network, as stated in the max-flow min-cut theorem.

**The Ford–Fulkerson** method or Ford–Fulkerson algorithm (**FFA**) is a greedy algorithm that computes the maximum flow in a flow network. It is sometimes called a "method" instead of an "algorithm" as the approach to finding augmenting paths in a residual graph is not fully specified or it is specified in several implementations with different running times. **L. R. Ford Jr.** and **D. R. Fulkerson** published the algorithm in 1956. The name "Ford–Fulkerson" is often also used for the Edmonds–Karp algorithm, which is a fully defined implementation of the Ford–Fulkerson method.

The idea behind the algorithm is as follows: as long as there is a path from the source (start node) to the sink (end node), with available capacity on all edges in the path, we send flow along one of the paths. Then we find another path, and so on. A path with available capacity is called an augmenting path.

* **flow charts :**
  + **Maximum flow for ford-Fulkerson’s algorithm :**
  + **Dijkstra’s algorithm :**



* **Snap Shots :**

3

9

8

7

1

10

Figure 1-GUI

6

5

4

2

|  |  |
| --- | --- |
| **Number** | **Description** |
| 1 | Input menu |
| 2 | Number of vertices |
| 3 | Edges : (Source – Destination – Weight) |
| 4 | Source Vertex |
| 5 | Destination Vertex |
| 6 | Submit button after finishing input data |
| 7 | Your input Graph |
| 8 | Dijkstra flow output |
| 9 | Minimum distance for Dijkstra algorithm |
| 10 | Maximum flow for Ford-Fulkerson Algorithm |

* **Output Discussion :**

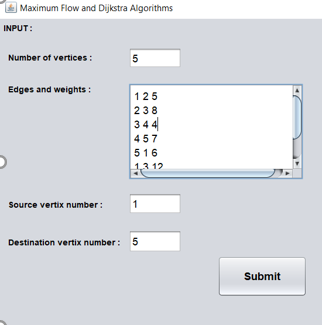
In this photo (Figure 1-GUI), you will see an entire Snap Shot for the program where you can see an input area on the left to input the information of the graph.

At Number seven in (Figure 1-GUI), you can see the graph that you have entered.

On the right the Output area, which contains the Dijkstra flow showing the flow from the source to destination also the minimum distance calculated.

In addition, the Maximum flow calculated using Ford-Fulkerson algorithm.

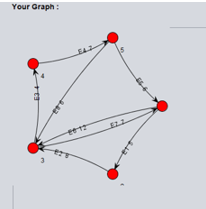
* **Program Step-By-Step :**

1. **Input :**

In the input, we find that the vertices are taken as number.

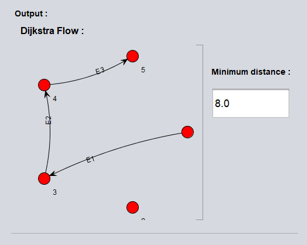
In addition, the edges taken formatted (source - destination - weight).

Submit button clicked after inserting all data.

****

1. **Your graph :**

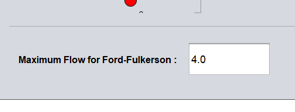
This layout shows your graph that you entered its data.

1. **Output :**
   1. **** **Dijkstra’s Algorithm :**

For Dijkstra’s algorithm we shows its output flow that show the minimum distance can be calculated throw which nodes.

In addition, the minimum distance is calculated.

* 1. **Maximum flow for ford-Fulkerson’s algorithm :**

For Maximum Flow for ford-Fulkerson’s algorithm we shows the value also in a box.

* **Pseudocodes :**
  + **Dijkstra's algorithm :**

1 **function** Dijkstra(*Graph*, *source*):

2

3 create vertex set Q

4

5 **for each** vertex *v* in *Graph*:

6 dist[*v*] ← INFINITY

7 prev[*v*] ← UNDEFINED

8 add *v* to *Q*

10 dist[*source*] ← 0

11

12 **while** *Q* is not empty:

13 *u* ← vertex in *Q* with min dist[u]

14

15 remove *u* from *Q*

16

17 **for each** neighbor *v* of *u*: *// only v that are still in Q*

18 *alt* ← dist[*u*] + length(*u*, *v*)

19 **if** *alt* < dist[*v*]:

20 dist[*v*] ← *alt*

21 prev[*v*] ← *u*

22

23 return dist[], prev[]

* + **Maximum flow for ford-Fulkerson’s algorithm :**
* function: FordFulkerson(Graph G,Node S,Node T):
* Initialise flow in all edges to 0
* while (there exists an augmenting path(P) between S and T in residual network graph):
* Augment flow between S to T along the path P
* Update residual network graph
* return
* **Suggestions for related future work :**

I suggest for a related future work with Dijkstra Algorithm and Maximum Flow algorithms is to apply those algorithms on a real problems in real life like:

* + - * + Geographical Maps.
        + Find locations of Map, which refers to vertices of graph.
        + IP routing to find Open shortest Path First.
        + The telephone network.
        + Airline Scheduling.

Appling our algorithms will also help us to understand those algorithms efficiently.

In addition working on such a big Project with better algorithms will increase the efficiency and performance of our applications.

* **References :**
  + - 1. Controversial, see Moshe Sniedovich (2006). "Dijkstra's algorithm revisited: the dynamic programming connexion".
      2. Frana, Phil (August 2010). "An Interview with Edsger W. Dijkstra". Communications of the ACM.
      3. Hall, A. How to use Dijkstra's algorithm. Retrieved April 27, 2016, from <https://www.youtube.com/watch?v=Cjzzx3MvOcU>
      4. Schrijver, A. (2002). "On the history of the transportation and maximum flow problems".
      5. Felner, Ariel (2011). Position Paper: Dijkstra's Algorithm versus Uniform Cost Search or a Case against Dijkstra's Algorithm. Proc. 4th Int'l Symp.
      6. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 26.2: The Ford–Fulkerson method". Introduction to Algorithms (Second Ed.). MIT Press and McGraw–Hill.
      7. George T. Heineman; Gary Pollice; Stanley Selkow (2008). "Chapter 8: Network Flow Algorithms". Algorithms in a Nutshell. Oreilly Media.
      8. Backman, Spencer; Huynh, Tony (2018). "Transfinite Ford–Fulkerson on a finite network".