[1]IMPLEMENTATION OF FORD-FULKERSON ALGORITHM

EMPIRICAL STUDIES

-ABOUT THE ALGORITHM

Ford-Fulkerson algorithm is used to find the maximum possible flow in a directed graph from a given source to a given destination.

The algorithm finds a path from source to sink with available capacity, called the augmenting path. As long as there is a path from source to sink with the residual capacities from the previously assigned flows, it augments the path to the previous flow. This algorithm ensures that all the three properties of the flow networks

1. Capacity Constraint:

   For all edges in the graph,

   $f(u,v) <= c(u,v)$, where (u,v) is any edge in the graph,

   $f(u,v)$ is the flow along the edge (u,v)

   $c(u,v)$ is the capacity along the edge (u,v)

2. Flow Conservation Property:

   Sum of incoming flows= Sum of the outgoing flows , for any vertex in the graph except the source and the sink.

3. Skew Symmetry:

   $f(u,v) = -f(v,u)$ where $f(u,v)$ is the flow along the edge (u,v)

are maintained.

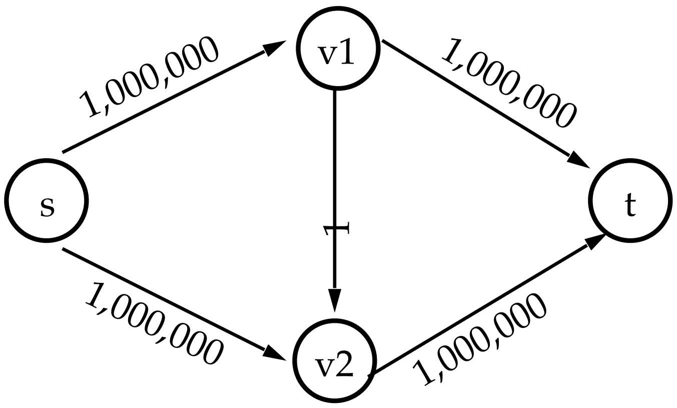
An improvised version - Edmond-Karp's solution uses Breadth-First Search to find the augmenting path.

ANALYSIS ON THE IMPLEMENTED SOLUTION:

Driver.java takes in input graph and a source and destination and outputs the maximum flow possible from the source to the sink for the given graph.

[1]*Nevhetha Radhamani Shanmugam* *nxr153230*

-SCENARIO-1



Sample I/O Driver.java for Scenario 1

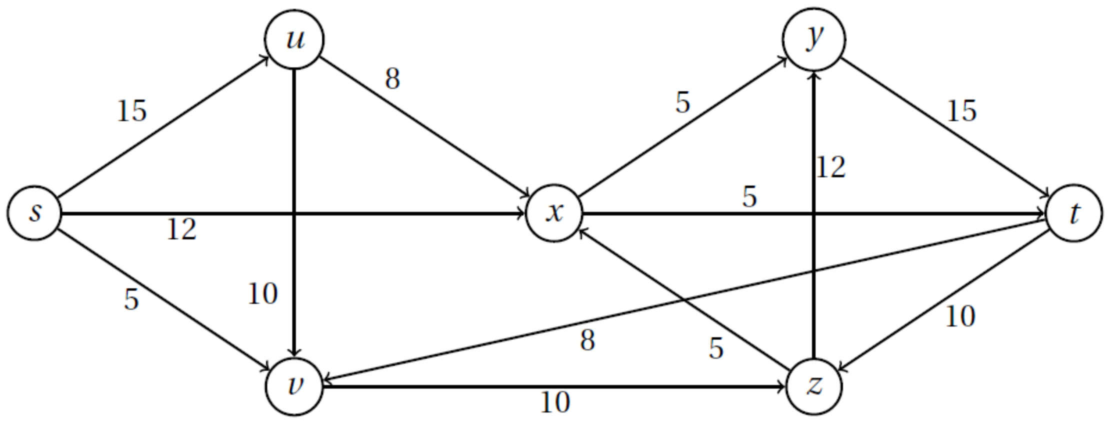
```
4 5
1 2 1000000
1 3 1000000
2 3 1
2 4 1000000
3 4 1000000
Input Graph:
1: (1,2) (1,3)
2: (2,3) (2,4)
3: (3,4)
4:
Source:
1[2]
Sink:
4


Flow along the edges of the given network:
============================================================

 (1,2)      Capacity: 1000000    Flow: 1000000
 (1,3)      Capacity: 1000000    Flow: 1000000
 (2,3)      Capacity: 1          Flow: 0
 (2,4)      Capacity: 1000000    Flow: 1000000
 (3,4)      Capacity: 1000000    Flow: 1000000
============================================================
Maximum flow for the given network is: 2000000
============================================================
```

[2] *Nevhetha Radhamani Shanmugam* *nxr153230*

-SCENARIO-2



Sample I/O Driver.java for Scenario 2:

```
7 13
1 2 15
2 4 8
1 4 12
1 5 5
2 5 10
7 6 10
5 6 10
6 4 5
4 7 5
7 5 8

6 3 12
4 3 5
3 7 15
Input Grap[3]h:
1: (1,2) (1,4) (1,5)
2: (2,4) (2,5)
3: (3,7)
4: (4,7) (4,3)
5: (5,6)
6: (6,4) (6,3)
7: (7,6) (7,5)
Source:
1
```

[3] *Nevhetha Radhamani Shanmugam* *nxr153230*

```
Sink:
7


Flow along the edges of the given network:
============================================================

 (1,2)      Capacity: 15   Flow: 5
 (1,4)      Capacity: 12   Flow: 10
 (1,5)      Capacity: 5    Flow: 5
 (2,4)      Capacity: 8    Flow: 0
 (2,5)      Capacity: 10   Flow: 5
 (3,7)      Capacity: 15   Flow: 15
 (4,7)      Capacity: 5    Flow: 5
 (4,3)      Capacity: 5    Flow: 5
 (5,6)      Capacity: 10   Flow: 10
 (6,4)      Capacity: 5    Flow: 0
 (6,3)      Capacity: 12   Flow: 10
 (7,6)      Capacity: 10   Flow: 0
 (7,5[4])     Capacity: 8    Flow: 0
============================================================
Maximum flow for the given network is: 20
============================================================
```

[4] *Nevhetha Radhamani Shanmugam* *nxr153230*