UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

5SENG001W: Algorithms Design & Implementation

Coursework (2020/21)

Explanation Report

Student ID: 2019437

UoW ID: W1761107

Name: Vajith Chamuditha

Course: BEng (Hons) Software Engineering

Module Leader: Mr. Sudarshana Welihida

Submission Date:

Data Structure: -

Adjacency matrix is used as the data structure. Adjacency matrix is a V*V array. The entry of the matrix is the capacity. If there is no path it is represented by 0 and if there is a path it is represented by its capacity (weight). In adjacency matrix we can check if two nodes are connected or not by o(1) approach. Inserting an edge also can done by o(1) times. So the time efficiency is high. For represent this 2D array is used which can hold fixed number of data of same data type. The reason to choose that is we can simply access any element by index, ability of storing large data and we already know the array size and data type.

Ford Fulkerson Algorithm with BFS : -

Ford Fulkerson algorithm is a greedy approach to calculate the maximum flow of a network of a graph. The Ford-fulkerson algorithm is a dynamic network routing algorithm that has advantages over static routing algorithms in that each router maintains a table of the best known distances to neighbours on the network.

The breadth first search (BFS) is used to find the path. This is a path finding algorithm that is capable of always finding the solution if one exists. BFS uses more memory but it will find the shortest path first. In here queue used which is a built in data structure and designed to have elements inserted at the end of the queue. It is working on first in first out method.

Short run of the algorithm: -

Pseudocode;

Ford Fulkerson

```
Begin
        create residual graph
        initialize max_flow to 0 run
        breadth first search (bfs)
        while true:
                run breadth first search
                check minimum value path_flow, residual graph check
                max_flow
        return flow
End
```

Breadth first search

Begin

create graph to check visited values

create queue

while queue is empty

drop front element from queue

mark visited elements

end while

return if visited

end

Output;

```
Connected from 0 to 1, capacity: 1
Connected from 0 to 4, capacity: 4
Connected from 1 to 2, capacity: 1
Connected from 1 to 3, capacity: 2
Connected from 2 to 3, capacity: 1
Connected from 2 to 4, capacity: 2
Connected from 3 to 4, capacity: 1
Connected from 1 to 5, capacity: 4
Connected from 4 to 5, capacity: 1

Graph Representation: Adjacency Matrix
0   1   0   0   4   0
0   0   1   2   0   4
0   0   0   1   2   0
0   0   0   0   1   0
0   0   0   0   0   1
0   0   0   0   0   0

Current flow: 0 + 1 = 1
Current flow: 1 + 1 = 2
Number of Edges: 9
Number of Vertices: 6
Maximum possible flow from source: 0 to destination: 5 is: 2
Time taken: 0.001 seconds.
```

Part C

Vertices	Time	Ratio	Log 2 ratio
6	0.001	-	-
12	0.001	1	0
24	0.001	1	0
48	0.004	4	2
96	0.013	3.25	1.6
192	0.02	1.5	0.5
384	0.085	4.25	2.08
786	0.575	6.7	2.7
1536	4.731	8.2	3

By using the doubling hypothesis, it can demonstrate that log ratio value is around 2. As the log ratio value is 2 the big O notation for this algorithm is $o(n^2)$. In my algorithm I have used while loop inside a for loop which should have time complexity $o(n^2)$.