Project Report on Preflow Push and Ford Fulkerson Algorithms

By Ankit Khandelwal & Lakshmi Nookala

Introduction

In this project, we studied preflow push and Ford Fulkerson algorithms to find maximum flow in a flow network. Based on our study and understanding we have implemented both these algorithms and compared the performances of our implementation of preflow push and Ford Fulkerson algorithms. We will first give a brief overview of these two algorithms, and some details about our implementation. And conclude with our results.

Preflow Push Algorithm

Preflow push is one of the most efficient algorithms used to find the maximum flow in a flow network. The main idea behind preflow push is that each node in the graph has a height value and any flow can only flow from a higher height to a lower height. The algorithm for preflow push is as follows:

```
Initial Conditions:
        h(s) = n, h(t) = 0, h(v) = 0, for all other v
        f(e) = c(e) for e = (s, v), f(e) = 0, for all other edges e
Preflow-Push
while there is a node v≠t with $e_f(v)>0$
  if there is w, s.t. $(v,w)\in E_f$, and h(w)<h(v)
        push(f, h, v, w)
  else
     relabel(f, h, v)
```

```
push(f, h, v, w)
if e=(v,w) is a forward edge
    increase f(e) to $\min(c_e, f(e)+e_f(v))$
 else if (v,w) is a backward edge
   e=(w,v)
   decrease f(e) to $\max(0, f(e)-e_f(v))$
```

```
relabel(f, h, v)
If($e_f(v)>0$ and  (v,w), h(w)≥h(v))
   Increase h(v) by 1
```

Implementation Details

For our implementation, we have stored edges as hashmap(EdgeList<Integer, int[]>), the edge capacity, forward flow and backward flow as three hashmaps - (edgeCapacity< int[], Integer>), (edgeFFlow< int[], Integer>), (edgeBFlow< int[], Integer>) respectively. We are storing excess and height values in a 2D array excess_height[][].

Our implementation iterates over for all vertices and checks that as long as there is an edge through which flow can be pushed it will push. Otherwise the program terminates when it finds out that no more flow can be pushed. We have used Java and Eclipse IDE for the implementation. Complexity of our implementation is $O(mn^2)$ where m is the number of edges and n is the number of vertices.

Ford Fulkerson Algorithm

Ford Fulkerson is an algorithm proposed by and named after L. R. Ford, Jr. and D. R. Fulkerson. This algorithm is used to find the maximum flow in a flow network. The main idea behind Ford Fulkerson algorithm is that "*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 these paths. Then we find another path, and so on*." The algorithm for Ford Fulkerson is as follows:

```
Initial Conditions
        Set f(e)=0 for all e in G
Ford Fulkerson Algo
        while there is a s-t path in G in the residue graph Gf
                let P be a simple path from s to t in the residue graph Gf
                f' -> augment(f, P)
                f -> f'
                Gf -> Gf'
        Return f;
```

Implementation Details

For our implementation, we have stored edges as hashmap(EdgeList<Integer, int[]>), the edge capacity, forward flow, backward flow and paths as hashmaps - (edgeCapacity< int[], Integer>), (edgeFFlow< int[], Integer>), (edgeBFlow< int[], Integer>), allPaths<Integer, ArrayList<int[]>> respectively.

Our implementation iterates over for all edges and checks that as long as there is a path through which flow can be sent, it will send. If no flow can be pushed using forward edges, then it will check for any flow that can be pushed using backward edges. If no flow is possible, the program terminates. We have used Java and Eclipse IDE for the implementation. Complexity of our implementation is O(mnC) where m is the number of edges and n is the number of vertices and C is the maximum capacity.

Performance Analysis

Having described the algorithms and our implementation, we will evaluate the average performance of both our algorithms and compare them with each other by varying the number of nodes.

No of Nodes & No of Edges	Running time Preflow Push (ns)	Running time Ford Fulkerson(ns)	Comparison Ratio: PFP/FF
6/8	746637	2856549	0.26137
8/9	851406	1290579	0.65970
20/31	8896354	14086505	0.63155
40/63	20337945	39334832	0.51705

Table 1: Running time comparison for Preflow Push and Ford Fulkerson

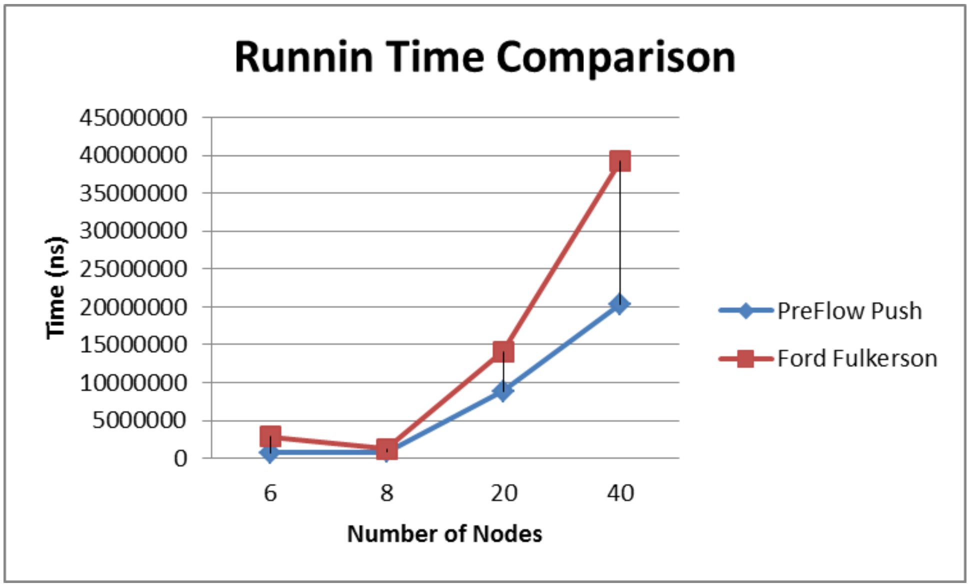

Figure 1: Running time comparison of Preflow Push and Ford Fulkerson

Conclusion

Based on our implementation, Preflow push performs better than Ford Fulkerson. We see that Preflow push algorithm runs in almost 3/5th time taken by Ford Fulkerson.