Assignment 6

Aryan Choudhary - 170152
Abhinav Sharma - 180017

Algorithm

Algorithm 1: Ploy-FF(G, s, t)

```
f ← 0
k ← maximum capacity of any edge in G
while k ≥ 1 do
    while there exist a path of capacity ≥ k in G_f do
        Let P be any path in G_f with capacity at least k
        c ← bottleneck capacity of path P
        for each edge (x,y) ∈ P do
            if (x, y) is forward edge then f(x, y) ← f(x, y) + c
            if (x, y) is backward edge then f(y, x) ← f(y, x) − c
        end
    end
    k ← k/2
end
```

Modified Ford-Fulkerson algorithm

"In each iteration, pick the path with maximum capacity in the residual network G_f and use it to increase the flow in G during that iteration."

Complexity Analysis

Upper bound on Modified Ford-Fulkerson algorithm

To prove: For any graph G, the worst case number of augmenting paths used in the Modified Ford-Fulkerson algorithm is upper bounded by the worst case number of augmenting paths used in the algorithm Poly-FF(G, s, t).

In any iteration of outer while loop of Poly-FF algorithm for k = k_0, we can choose the path with capacity greater than or equal to k in any order, because this algorithm does not specifies anything about the order in which paths are chosen until the path capacity of chosen path is $\geq k_0$. If there exists a (s-t) path of capacity $\geq k_0$, then the capacity of maximum capacity path is also $\geq k_0$. Therefore, maximum capacity path is a possible path for this iteration. Also if there does not exist any path with capacity at least k_0, then the capacity of maximum capacity path will also be less than k_0. Therefore, the inner while loop will break. Hence, maximum capacity path satisfies all the properties of the path to be chosen.
Let G_{f1} be the residual graph for modified Ford-Fulkerson algorithm and G_{f2} be the residual graph for Poly-FF algorithm. Initially, residual graph is same as given graph. Therefore G_{f1} and G_{f2} are same.

Lemma - There exist a specific instance of Poly-FF algorithm which uses the same number of augmented paths as used by Modified Ford-Fulkerson algorithm and reaches the same instance of residual graph. i.e. at any iteration if p is number of augmented paths used by both the algorithms , then at this iteration G_{f1} is same as G_{f2}.

We will use induction to prove above lemma. Induction on p-
Base case - p = 0
Initially, G_{f1} is same as G_{f2}. Therefore, this case is true.
Induction Step -
Induction hypothesis - G_{f1} is same as G_{f2} after p augmented paths are used by each of these algorithms. where p is not greater than minimum of total number of augmented paths used by any of the above algorithms.
Let P be the path of maximum capacity in G_{f1} used as (p+1)th augmented path by Modified Ford-Fulkerson algorithm. Since, G_{f1} and G_{f2} are same at this step P is also a maximum capacity path in G_{f2}. Therefore, P is also a valid path to be chosen by Poly-FF algorithm in this iteration (as shown in paragraph just above) . Since, same paths are used as augmenting paths , G_{f1} and G_{f2} are modified in the same way in this iteration. Hence, after this iteration G_{f1} will be same as G_{f2}.

Using the above lemma, we can say that when there does not exist any augmenting path in G_{f1}, there does not exist any augmenting path in G_{f2} also. And, for every augmenting path used by modified Ford-Fulkerson algorithm, there was a augmenting path used by above specific instance of Poly-FF algorithm.
The number of augmenting paths used in this specific instance of Poly-FF algorithm is upper bounded by worst case number of augmenting paths used by Poly-FF algorithm. Therefore, the number of augmenting paths used by Modified Ford-Fulkerson algorithm is upper bounded by worst case number of augmenting paths used in the Poly-FF algorithm.

Analysis for Poly-FF algorithm

To prove: The number of augmenting path use by Poly-FF(G,s,t) algorithm is $O(mlog(c_{max}))$.

Lemma 1 If there does not exist a s-t path with capacity $\geq$ c in G_f, then there exist a s-t cut in G_f such that all the outgoing edges (i.e. edge from set containing s to set containing t) in this cut have capacity less than c.
Residual capacity in defined as -
For forward edge = Capacity of edge in original graph - current flow.
For backward edge = Current flow.

Proof- By valid path from u to v in proof means that a path from u to v in G with capacity $\geq c$. Lets prove it in general if there does not exist a s-t valid path with capacity $\geq$ c in $G = (V, E)$, then there exist a s-t cut in G such that all the outgoing edges in this cut have capacity less than c.
DFS/BFS or any other algorithm which we used to find an augmenting path doesnt makes any difference between residual capacity in G_f or if we replace it with an equivalent graph of same capacities.

We will prove it by contradiction. Lets assume that for all possible s-t cuts in G there exists an outgoing edge with capacity $\geq$ c, yet there doesn't exists a valid s-t path.

Let $u \in A$ be the set of all possible nodes in G such that there is a valid path from s to u. If $|A| = |V|$, then all nodes in V belong to A. Hence, t also belongs to A, which in turn means that there is a valid path from s to t.

If t belongs to A. then there exists a valid path from s-t in G (because $u \in A$ is the set of all possible nodes u in G such that there is a valid path from s to u).

If $|A| = |V|$, then all nodes in V belong to A which in turn mean that t belongs to A.

Let the size of $|A| < |V|$ and t does not belong to A. Since s is reachable from s $\implies$ $s \in A$ and t does not belong to A then lets look at s-t cut = $(S = A, T = V - A)$.

Since all s-t cuts have an outgoing edge with capacity $\geq c$. Let $(u, v) \in E$ such that $u \in S$, $v \in T$ and capacity$(u, v) \geq c$. Because $u \in A$ there is a valid path from s to u because there is an edge (u, v) with capacity $\geq c$, we can extend this valid path to u to include this edge and create a valid path from s to v. Therefore there exists v which doesnt belong to A yet there exists a valid path from s to v. This contradicts the fact that A contains all possible nodes u such that there is a valid from s to u.

Lemma 2 Consider the beginning of the iteration of outer while loop such that value of k = k_0. If f is the current value of the (s, t)-flow in G, then f $\geq f_{max} - 2mk_0$, where f_{max} is the maximum (s, t)-flow in G.
Proof- First of all, we will show that it is true in the beginning of algorithm.
Initially, f=0 and k_0 = capacity of maximum capacity edge in G. We know that flow from s to t is same as sum of flow through the outgoing edges from s. Therefore, the max-flow(f_{max}) from s to t is bounded by sum of capacities of outgoing edges from s.
$f_{max} \leq \sum_{x=s,(x,y)\in E}$ capacity of edge(x, y)
$f_{max} \leq m * k_0$
or, $f_{max} - 2mk_0 \leq -mk_0 \leq 0 =$ f
Therefore, in the beginning of algorithm , f $\geq f_{max} - 2mk_0$.
Now, iteration for k = k_0 begins after the end of the iteration for k = $2k_0$. This means when iteration of outer while loop begins with k = k_0. there does not exist any (s-t) path of capacity greater than or equal to $2k_0$. Considering c=$2k_0$ in Lemma 1, we can say that there exist a s-t cut in G_f such that all the outgoing edges in this cut have the residual capacity less than $2k_0$. Let this cut be defined by set S and T = V-S, where S contains all the nodes u $\in$ V such that there exists a valid path (i.e. a path with capacity $2k_0$) from s to u. Consider this cut in the original graph G. We know that the capacity of any cut is the sum of capacity of the outgoing edges. We will prove that the capacity of this cut is bounded by (current flow + 2mk_0).
Consider any outgoing edge (x,y) in G where x $\in$ S and y $\in$ T. The capacity of this edge is the sum of the current flow through this edge and residual capacity of this edge. This is because of the way the residual capacity is defined $c_f(x, y) = c(x, y) - f(x, y)$, where $c_f(x, y)$ is the residual capacity of edge (x,y), $c(x, y)$ is the original capacity of this edge and $f(x, y)$ is the flow through this edge.
The edge (x,y) is also an outgoing edge from S to T in G_f. From lemma 1, we know that capacity of this edge in G_f is less than to $2k_0$.
$c_f(x, y) < 2k_0$
$\implies c(x, y) - f(x, y) < 2k_0$
or, $\implies c(x, y) < f(x, y) + 2k_0$

Let the capacity of this cut be $C(S, T)$
$C(S, T) = \sum_{(x,y)\in E,\ x\in S,\ y\in T} c(x, y)$
$\implies C(S, T) \leq \sum_{(x,y)\in E,\ x\in S,\ y\in T}(f(x, y) + 2k_0) \ldots (i)$

Now, let f_{curr} be the current flow from s to t , then
f_{curr} = out-flow − in-flow
$\implies f_{curr} = \sum_{(x,y)\in E,\ x\in S,\ y\in T} f(x, y) - \sum_{(x,y)\in E,\ x\in T,\ y\in S} f(x, y)$

Consider the incoming edge (x,y) in G , where x $\in$ T and y $\in$ S. If the flow through this edge is f(x,y) , then there exist a backward edge from S to T in G_f whose capacity is equal to f(x,y). Since, this backward edge is an edge from S to T in G_f, it is also an outgoing edge for S to T in G_f, therefore from lemma 1, f(x,y) $\leq 2k_0$.
$f_{curr} \geq \sum_{(x,y)\in E,\ x\in S,\ y\in T} f(x, y) - \sum_{(x,y)\in E,\ x\in T,\ y\in S} 2k_0$
$\implies \sum_{(x,y)\in E,\ x\in S,\ y\in T} f(x, y) \leq f_{curr} + \sum_{(x,y)\in E,\ x\in T,\ y\in S} 2k_0 \ldots (ii)$

Using (i) and (ii)
$C(S, T) \leq \sum_{(x,y)\in E,\ x\in S,\ y\in T}(f(x, y) + 2k_0) \leq f_{curr} + \sum_{(x,y)\in E,\ x\in T,\ y\in S} 2k_0 + \sum_{(x,y)\in E,\ x\in S,\ y\in T} 2k_0$

Since, the number of edges (x,y) such that x $\in$ S and y $\in$ T or x $\in$ T and y $\in$ S is less than or equal to m (no of edges in G). Therefore,
$C(S, T) \leq f_{curr} + 2mk_0$
$\implies f_{curr} \geq C(S, T) - 2mk_0$

Since min-cut is less than or equal to Capacity of any cut , Therefore
$f_{curr} \geq C(S, T) - 2mk_0 \geq$ min-cut(G) $- 2mk_0$

Using min-cut = max-flow
$f_{curr} \geq$ min-cut(G) $- 2mk_0 = f_{max} - 2mk_0$
$\implies f_{curr} \geq f_{max} - 2mk_0$

Lemma 3 In an iteration of the inner while loop for a given value k_0 of variable k , the lower bound on the amount by which flow increase is k_0.
Proof The inner while loop is executed only when there exist a path P from s to t with capacity at least k_0. After this iteration the flow through each edge of path P increases by the bottleneck capacity(c) of path P. Since, the capacity of path P is at least k_0, therefore the bottleneck capacity(c) $\geq k_0$. The first edge in this path is an outgoing edge from s. Therefore, the outgoing flow from s increases by c ($\geq k_0$).
We know that total flow from s to t is equal to the amount of flow coming out from s. Therefore, total flow increases by more than k_0 in an iteration of inner while loop or the lower bound on the amount by which flow increase is k_0.

To Prove: The number of augmenting path used by Poly-FF algorithm is $\mathbf{O(mlog(}c_{max}\mathbf{))}$, where c_{max} is the edge with maximum capacity in G.
Proof Using Lemma 2, we know that in the start of the While loop for k = k_0, the current flow (f_{curr}) is away from the maximum flow (f_{max}) by $2mk_0$. Using Lemma 3, we know that in each iteration of the inner loop for k = k_0, the flow increases by c ($\geq k_0$). Therefore, the maximum number of iteration of inner while loop for k = k_0 is $\frac{2mk_0}{c} \leq \frac{2mk_0}{k_0} = 2m$. Hence maximum number of iteration of inner loop for a particular value of k is bounded by $O(m)$. Also each iteration of inner while loop can be performed in $O(m)$ using DFS/BFS by ignoring the edges with capacity less than k.
In each iteration of the outer loop k reduces to k/2. Therefore, outer while loop runs for (1 + log(c_{max})) times, where c_{max} is the edge with maximum capacity in G.
Hence, the number of augmenting path from s to t used by algorithm Poly-FF is $O(mlog(c_{max}))$.
The overall complexity of Poly-FF algorithm is given by $O(m * m * log(m)) = O(m^2 log(m))$.