CS 345A Assignment 6

Yasharth Bajpai (170822), Nidhi Hegde (180472)

November 11, 2020

Difficult

Pseudo Code:

Algorithm 1 Poly-FF(G, s, t)

```
$f \leftarrow 0$
$k \leftarrow$ maximum capacity of any edge in $G$
while $k \geq 1$ do
    while there exists a path of capacity $\geq k$ in $G_f$ do
        Let $P$ be any path in $G_f$ with capacity at least $k$
        for each edge $(x, y) \in P$ do
                                                    ▷ Note: the change of $G_f$ to $P$
            if $(x, y)$ is a forward edge then
                $f(x, y) \leftarrow f(x, y) + k$
            end
            if $(x, y)$ is a backward edge then
                $f(y, x) \leftarrow f(y, x) - k$
            end
        end
    end
    $k \leftarrow k/2$
end
return $f$
```

Time Complexity:

Following the steps as guided in the assignment.

Lemma 0.001: Worst case number of augmenting paths used in the maximum capacity path algorithm is upper bounded by the worst case number of augmenting paths used in the Poly-FF(G, s, t).

Consider the maximum capacity algorithm, let's assume that the capacities bottle-necking the paths at every iteration are c_i where $i \in [1, n]$ such that $c_i \geq c_{i+1} \forall i \in [1, n)$. Note that in this case, at every iteration i, the flow gets increased by c_i.

On the other hand, in the Poly-FF algorithm, for a particular scaling $k = k_o$, and considering the same set of paths. Note that in this case, at every iteration, the flow gets increased by a factor k_o which we know from the algorithm is at most c_i (Bottle neck capacity of the path being considered in the iteration).

From both these realizations of flow increase for the same set of paths in consideration,we can say that, the increase in flow in each iteration is higher for the Maximum capacity algorithm by $c_i - k_o$, which is greater than equal to 0. Using this result, we can conclude that the Poly-FF will need a higher number of augmenting paths as compared to the maximum capacity paths to reach the optimal solution(In the worst case).

Lemma 0.01: The outermost While loop should run for $\mathbf{O(\log_2 C_{max})}$ times only. One can see this trivially that the scaling variable, k is halved at every step of the outer loop iteration. The maximum possible value for k being c_{max} and minimum for the loop condition to be true being 1. We can say that The number of iterations of the outer While loop is at most $1 + \lceil log_2 c_{max} \rceil$. Thus, making complexity of the outer loop to be $O(\log_2 c_{max})$.

Lemma 0.1: If f is the current value of the (s, t)-flow in G, then show that $f \geq f_{max} - 2mk_o$, where f_{max} is the maximum (s, t)-flow in G.

Let the value of k in the current iteration of the outer while loop be k_o. Consider the s-t cut defined as follows:
A: The set of all vertices v in G such that there exists a path (s,v) in G_f with capacity $\geq 2k_o$.
Vertex t is not a part of A, since all such paths with capacity $\geq 2k_o$ from s to t are already taken care of by the previous iteration of the outer while loop, when $k = 2k_o$. Hence, the cut defined is a valid cut.
Observation 1:
There is no edge {u, v} in G for which $\mathbf{u \in A}$ and $\mathbf{v \in \bar{A}}$, such that $\mathbf{C_{uv} \geq f_{uv} + 2k_o}$
Proof by Contradiction: If such an edge exists, there will be a corresponding forward edge in G_f with $e_f(u, v) = C_{uv} - f_{uv}$. Thus, it will have $e_f(u, v) \geq 2k_o$. Hence, by definition of the cut, if vertex u lies in A, vertex v will also lie in A, since the entire flow reaching u can be transmitted across the edge (u,v). This contradicts our assumption. Hence, we prove this observation
Observation 2:
There is no edge {u', v'} in G for which $\mathbf{u' \in \bar{A}}$ and $\mathbf{v' \in A}$, such that $\mathbf{f_{u'v'} \geq 2k_o}$

Proof by Contradiction: If such an edge exists, there will be a corresponding backward edge in G_f with $e_f(v', u') = f_{u'v'}$. Thus, there is an edge from A to $\bar{A}$ in G_f with $e_f(v', u') \geq 2k_o$. As there exists a path from s to v' with a capacity $\geq 2k_o$, so this entire flow reaching v' can be transmitted across the edge (v',u') to get a path with capacity $\geq 2k_o$ from s to u'. Hence, by definition of the cut, vertex u' will lie in A, which contradicts our assumption. Hence, we prove this observation.
Thus, from these two observations, all edges coming out of A must satisfy $C_e < f_e + 2k_o$, while all edges coming into A must satisfy $f_e < 2k_o$.
We also know that the capacity of this cut is defined as:

$$C_{A\bar{A}} = \Sigma_{u \in A, v \in \bar{A}}\, C_{uv}$$

And,

$$\begin{aligned} f_A^{out} - f_A^{in} &= \Sigma_{u \in A, v \in \bar{A}}\, f_{uv} - \Sigma_{v' \in \bar{A}, u' \in A}\, f_{v'u'} \\ &\geq \Sigma_{u \in A, v \in \bar{A}}\, [C_{uv} - 2k_o] - \Sigma_{v' \in \bar{A}, u' \in A}\, 2k_o \\ &= \Sigma_{u \in A, v \in \bar{A}}\, C_{uv} - \left[\Sigma_{u \in A, v \in \bar{A}}\, 2k_o + \Sigma_{v' \in \bar{A}, u' \in A}\, 2k_o\right] \\ &= C_{A\bar{A}} - m.2k_o \end{aligned}$$

The first term in the last line is obtained by the definition of capacity of the cut and the second term is implied by the fact that there can be a maximum of m edges passing through any cut. Now, by **MaxFlowMinCut Theorem**, the value of max flow in the network is bounded by the capacity of any s-t cut and thus is equal to the capacity of min-cut. Hence, the capacity of the cut defined above must be greater than equal to f_{max}, and hence can be replaced by it in the relation above.
Also, the flow through this cut is equal to f (proved in class), which is the current value of flow from s to t in the network G. From these observations, we get the relation below, proving Lemma 0.1

$$f = f_A^{out} - f_A^{in} \geq\ f_{max} - m.2k_o$$

Now, for each iteration of augmentation(Inner While Loop) with $k = k_o$, one can note that the flow at least increases by a quantum of k_o. This is because the algorithm picks a path of a capacity at least k_o in each iteration of the inner while loop, and the flow from s to t is incremented by this quantity thereafter.

Using this and *Lemma* 0.1(that the current flow is away from the maximum flow by at most $2mk_0$ and that each iteration of the inner loop increases the flow by at least k_o), we can say that the entire augmentation procedure(for the current value of k) shall run for at most $2m$ iterations.

Therefore, we can say for a complete augmentation $O(m)$ time operations(The for loop) are executed for $O(m)$ time. Thus, making the complexity of an augmentation to be $O(m^2)$.

Using *Lemma* 0.01 and the above realizations. We can now, conclude that the Outer Loop executes $O(m^2)$ operations for $O(\log_2 c_{max})$.
Making the complexity of the entire procedure Poly-FF to be $O(m^2 \log_2 c_{max})$.