# CS345 Assignment 6

Chinmay Goyal(180206) and Supreeth Baliga(180801)

## 1 Fold Fulkerson in Polynomial Time for Integral Capacity:

**Solution:**

**ALGORITHM:** We first start by analyzing the algorithm given in the hints of the questions. Then we discuss the time complexity of the given algorithm, and after this we will show that the number of iterations of the Modified version of Ford-Fulkerson Algorithm in which we take the path with maximum capacity is of the same order as the number of iterations of Algo-1, hence providing us with the required polynomial time complexity of the algorithm.

```
Algorithm 1: Poly-FF(G,s,t):
f <- 0;
k <- 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 $\in$ $G_f$ do {
            if (x,y) is a forward edge then
                f(x,y) <- f(x,y) + BottleNeckCapacity(P);
            if (x,y) is a backward edge then
                f(y,x) <- f(y,x) - BottleNeckCapacity(P);
            // Note: BottleNeckCapacity(P) $\geq$ k
        }
    }
    k <- k/2;
}
return f;
```

**PROOF OF CORRECTNESS:**
**Lemma 1:** Since the edge capacities are integers, there will be a flow in which flow along all the edges will be integral.

**Proof:** The proof of this lemma is straightforward, we can use induction to show that at the beginning of the inner while loop, all the edge capacities in the residual graph are integral and the flow across all the edges is integral, hence after one iteration of the while loop, we only add/subtract integer values from the flow which ensures that at the end of the inner while loop, still all values remain integers. This would complete the induction proof (similar to the one done in class).

**Proof that Algorithm - 1 upon termination gives the maximum flow:**
Firstly, we show that the algorithm does terminate. Using Lemma 1, we can say that when k = 1 in the above algorithm, we will exit from the inner while loop only when there exists no path from s-t in $G_f$, (we use the above lemma to ensure that no paths with fractional capacity can exist from s-t), which will ensure that we have covered all the augmenting paths from s-t. Thus, the algorithm terminates. Now, imagine that we are applying Ford-Fulkerson algorithm and we use the same sequence of paths as we did in this algorithm. Clearly, we will get the same residual graph in this case too. Now, we proved in the case of Ford-Fulkerson algorithm that the flow we get is a max-flow. Using this, we infer that our algorithm has also computed a max flow since both the Ford-Fulkerson algorithm and our algorithm have computed the same residual graph.

**ANALYSING TIME COMPLEXITY:** Here, for the outermost while loop we can see that after each iteration we are dividing the value of k by 2, hence this while loop will iterate for $O(\log_2 C_{MAX})$ times, where $C_{MAX}$ is the maximum of capacity of all the edges. Finding a path in $G_f$, with each edge having capacity of at least k requires a simple DFS on $G_f$ with a bit of modification (In this DFS, we ignore edges with value less than k), hence this will take $O(n+m) \implies O(m)$ time. Now we only have to show that for any value of k, the number of iterations of the inner while loop is $O(m)$ only.

**Lemma 2**: If at the beginning of any iteration of the outermost while loop, say k= $k_0$, $f$ is the current flow in the graph. then $f \geq f_{max} - 2mk_0$, where $f_{max}$ is the maximum value of the flow that is possible in the graph.

**Proof:** Let the value of k at the beginning of an iteration of inner while loop be $k_0$, then from the structure of while loop we can say that there is no path currently from s to t in $G_f$ such that the capacity of the path is at least $2k_0$, hence we know that all the paths from s-t have capacities less than $2k_0$ at the beginning of the while loop. After this we will use the maxflow-mincut theorem to prove the above lemma. Let set A be defined such that it is the set of all reachable vertices in $G_f$ from s such that all the edges have capacities more than $2k_0$, and let $\bar{A}$, be the set of the remaining vertices, then as there is no path in $G_f$, from s to t with path capacity atleast $2k_0$, we can say that $s \in A$, and $t \in \bar{A}$.

**Claim 1 :** Let e = (u,v) be an edge in $G$, such that u $\in$ A and v $\in$ $\bar{A}$, we say that the flow $f$ in this edge has the following property: $c_e < f(e) + 2k_0$. We prove this claim by contradiction. Let us assume that this is not the case, then there is a forward edge from u to v in $G_f$, with capacity more than $2k_0$, which means that the vertex v is reachable from s and it must belong to the set A, but this is a contradiction, hence we prove that $c_e < f(e) + 2k_0$.

**Claim 2 :**Let e = (u,v) be an edge in $G$, such that v $\in$ A and u $\in$ $\bar{A}$, then we claim that the value of flow through this edge has the following property: $f(e) < 2k_0$. We will prove this by contradiction. Let us assume that this is not the case, hence there will be a backward edge in $G_f$, from v to u with capacity more than $2k_0$, which would mean that the vertex u is reachable from s, but this is a contradiction, hence we prove that $f(e) < 2k_0$.

Now, we will use Claim 1 and Claim 2 to prove the above Lemma.

$$f = \sum_{x \in A} f_{out}(x) - \sum_{x \in A} f_{in}(x)$$

$$f = \sum_{(x,y) \in E, x \in A, y \in \bar{A}} f_{out}(e) - \sum_{(x,y) \in E, y \in A, x \in \bar{A}} f_{in}(e)$$

Using the claims proved above:

$$f \geq \sum_{\text{edge e going out of A}} (c_e - 2k_0) - \sum_{\text{edge e coming into A}} 2k_0$$

$$f \geq \sum_{\text{edge e going out of A}} (c_e) - 2mk_0$$

But $\sum_{\text{edge e going out of A}}(c_e)$ is the capacity of the cut (A,$\bar{A}$). So, we get

$$f \geq Cap(A, \bar{A}) - 2mk_0$$

From the maxflow -mincut theorem we know that $f_{max} \leq Cap(\text{any s} - \text{t cut})$, hence:

$$f + 2mk_0 \geq \sum_{\text{edge e going out of A}} (c_e) \geq f_{max}$$

$$f \geq f_{max} - 2mk_0$$

Hence Proved.

Say, we are currently at k=$k_0$. By the above lemma,

$$f \geq f_{max} - 2mk_0$$

where $f$ denotes the flow at the beginning of the iteration. So, we have

$$f_{max} - f \leq 2mk_0$$

Also, in each iteration of the inner while loop, we increase the flow in the graph by at least $k_0$. Using this and the fact that f cannot be more than $f_{max}$, we get number of iterations of the inner while loop for any iteration of the outer while loop will be at most 2m. So, we get that number of iterations of the inner while loop for any specific value of k is O(m) only. Hence the total time complexity of the given algorithm is $O(m \times m \times log_2 C_{MAX}) = O(m^2 log_2 C_{MAX})$.

Now the only thing left to be shown is that the worst case number of augmenting paths used in the modified Ford-Fulkerson algorithm (the one where we find the max-capacity path in each iteration) is upper bounded by the worst case number of augmenting paths used in Algorithm-1. Now, it is possible that in a run of Algorithm-1, it chooses the same sequence of augmenting paths as the modified Ford-Fulkerson algorithm(i.e. the Poly-FF Algorithm always chooses the path with maximum capacity). But we proved that for any run of Algorithm-1, the worst case number of augmenting paths is O($mlog_2 C_{MAX}$)(We saw that for any value of k the inner while loop iterates $O(m)$ times and the number of different values that k can take is $O(log_2 C_{MAX})$, hence the total number of augmenting paths in Poly-FF is $O(mlog_2 C_{MAX})$). This implies that the worst case number of augmenting paths used in the modified Ford-Fulkerson algorithm is O($mlog_2 C_{MAX}$). Also, finding the path of max-capacity in each iteration of modified Ford-Fulkerson method takes polynomial time (By making modifications in Dijkstra'a algorithm). Hence, the time complexity of modified Ford-Fulkerson method is polynomial in the input size.