Design and Analysis of Algorithms : CS345A
CSE, IIT Kanpur

Worked out Assignment on *Maximum Flow*

An Important guideline

- It is only through the assignments that one learns the most about the algorithms and data structures. For the current assignment, a sequence of hints are given. Try to look at a hint only after trying on your own for sufficiently long period of time. The onus of learning from a course lies first on you. So act wisely while working on this assignment.

Ford-Fulkerson algorithm in polynomial time for integer capacity

While covering the topic on Maximum Flow, we showed a graph with integer capacities on which the Ford-Fulkerson algorithm can be made to run in $\Theta(mc_{\max})$ time, where $c_{\max}$ is the maximum capacity of any edge in G. This running time is not a polynomial in the input size. The objective of this assignment problem is to slightly modify the algorithm as follows.

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.

The motivation behind this slight modification is quite intuitive. Isn't it? Interestingly, the modified algorithm will use only $O(m \log_2 c_{\max})$ augmenting paths to compute the maximum flow from s to t. Hence, the time complexity of the modified algorithm will be a polynomial in the input size.

Spend sufficient time on the modified algorithm and the above mentioned bound. Only after spending sufficient time, turn over the page.

Consider the following algorithm.

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

```
f ← 0;
k ← maximum capacity of any edge in G;
while k ≥ ... do
    while there exists a path of capacity ≥ ... in G_f do
        Let P be any path in G_f with capacity at least ...;
        for each edge (x, y) ∈ G_f do
            if (x, y) is a forward edge then f(x, y) ← f(x, y) + ...;
            if (x, y) is a backward edge then f(y, x) ← f(y, x) − ...;
    k ← ...;
return f;
```

1. Fill in the blanks of Algorithm 1 suitably. This will add to your insight into the algorithm.

2. Show that, for any graph G, the worst case number of augmenting paths used in the modified Ford-Fulkerson algorithm described on the previous page is upper bounded by the worst case number of augmenting paths used in Algorithm 1. Hence, in order to show that the modified algorithm on the previous page uses $O(m \log_2 c_{\max})$ augmenting paths, it suffices to show that Algorithm 1 uses $O(m \log_2 c_{\max})$ augmenting paths.

3. A useful hint, that you may use, is the following: The outermost While loop of Algorithm 1 should run for $O(\log_2 c_{\max})$ times only. So all that is left for you to show is that the number of iterations of the inner While loop of Algorithm 1 for any specific value of variable k is $O(m)$ only. Spend sufficient time to establish this. If you don't succeed, please turn over the page.

Note:
The running time of one iteration of the inner While loop of Algorithm 1 is $O(m)$. So, if we are able to establish the validity of Step 3, the running time of Algorithm 1 is $O(m^2 \log_2 c_{\max})$; so our objective will be achieved ☺.

Proceed along the following steps.

1. Consider the beginning of the iteration of the outermost While loop for any value, say k_0, of variable k. We wish to show that the value of the (s,t)-flow in G at this stage is away from maximum (s,t)-flow by at most $2mk_0$.
2. What is the lower bound on the amount by which the flow increases in an iteration of the inner While loop for a given value k_0 of variable k ?
3. Use (1) and (2) to provide suitable arguments to establish that the inner While loop will run for $O(m)$ times only for any specific value of k.

If you are still not able to complete the above steps, turn over the page.

The following are the hints for the steps mentioned on the previous page.

1. You need to prove the following lemma:

   Lemma 0.1 *If f is the current value of the (s,t)-flow in G, then show that $f \geq f_{\max} - 2mk_0$, where $f_{\max}$ is the maximum (s,t)-flow in G.*

   In order to prove Lemma 0.1, carefully analyse the residual network G_f. If you have fully internalized the maxflow-mincut theorem discussed in the lecture, you should have no difficulty to do this task.

   But if you are still not able to complete the above steps, turn over the page.

2. The flow increases by at least k_0 in each iteration. Give suitable arguments to support this claim.

3. It follows from (1) that the current flow is away from the maximum flow by at most $2mk_0$. It follows from (2) that each iteration of the inner loop increases the flow by at least k. Hence the number of iterations of the inner While loop for any fixed value of k is $O(m)$ only.

For Step 1 on the previous page:
Notice that there is no (s,t)-path in G_f with capacity $\geq 2k_0$. Now suitably define a (s,t)-cut in G_f and then try to give a bound on the capacities of its forward and backward edges. Use it to derive a lower bound on the current flow in terms of the capacity of the cut. You might have to use the fact that the maximum (s,t)-flow is bounded by the capacity of any (s,t)-cut.

But if you are still not able to define the desired (s,t)-cut in G_f, turn over the page.

Let A be the set of vertices in G_f reachable by a path of capacity at least $2k_0$.
Show that $(A, \overline{A})$ is the desired cut.

The arguments required to show this are quite similar to the arguments used in the the proof of maxflow-mincut theorem.