CS345 : Algorithms II
Semester I, 2020-21, CSE, IIT Kanpur

Assignment 6

Deadline : 11:55 PM, 10th November 2020.

Most Important guidelines

- It is only through the assignments that one learns the most about the algorithms and data structures. You are advised to refrain from searching for a solution on the net or from a notebook or from other fellow students. Remember - **Before cheating the instructor, you are cheating yourself**. The onus of learning from a course lies first on you. So act wisely while working on this assignment.
- **Grading policy:**
  There will be no penalty for submission based on the time of submission as long as the submission is before the deadline.
- Refrain from collaborating with the students of other groups. If any evidence is found that confirms copying, the penalty will be very harsh. Refer to the website at the link: https://cse.iitk.ac.in/pages/AntiCheatingPolicy.html regarding the departmental policy on cheating.

General guidelines

1. There is only problem in this assignment. However, hints are given at multiple levels in the solution of this problem. You may use these hints if required. There is no penalty for using these hints.

2. You are strongly discouraged to submit the scanned copy of a handwritten solution. Instead, you should prepare your answer using any text processing software (LaTex, Microsoft word, ...). The final submission should be a single pdf file.

3. You need to justify any claim that you make during the analysis of the algorithm. But you must be formal, concise, and precise. You may use the results proved in the class. But, if you wish to use any homework problem in your solution, you must provide its solution as well.

4. If you are asked to design an algorithm, you may state the algorithm either in plain English or a pseudocode. But it must be formal, complete, unambiguous, and easy to read. You must not submit any code (in C++ or C, python, ...).

5. **Naming the file**:
   The submission file has to be given a name that reflects the information about the assignment number, and the roll numbers of the 2 students of the group. For example, you should name the file as **Assign_i_Rollnumber1_Rollnumber2.pdf** if you are submitting the solution for the ith assignment.

6. **Each student of a group** has to upload the submission file separately unlike in the past when only one submission file per group had to be uploaded. Be careful during the submission of an assignment. Once submitted, it can not be re-submitted.

7. Deadline is strict. Make sure you upload the assignment well in time to avoid last minute rush.

8. Contact TA at the email address: **Kbhanja@cse.iitk.ac.in** for all queries related to the submission of this assignment. Avoid sending any such queries to the instructor.

Ford-Fulkerson algorithm in polynomial time for integer capacity

In Lecture 20, 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 so that it runs in polynomial time for all integer capacity graphs. In fact, the intuition underlying the modified algorithm is based on the graph that we discussed in Lecture 20.

Spend sufficient time to ponder over the slight modification in the Ford-Fulkerson algorithm. Thereafter, turn over the page.

The modification required in the Ford-Fulkerson algorithm is just the following.

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.

Spend sufficient time to show that the Ford-Fulkerson algorithm after this modification will use only $O(m \log_2 c_{\max})$ augmenting paths to compute the maximum flow from s to t. Hence, its time complexity is polynomial in the input size. If you don't succeed, 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 Poly-FF(G, s, t) 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 algorithm described on the previous page is upper bounded by the worst case number of augmenting paths used in the algorithm Poly-FF(G, s, t). Hence, in order to show that the algorithm on the previous page uses $O(m \log_2 c_{\max})$ augmenting paths, it suffices to show that the algorithm Poly-FF(G, s, t) uses $O(m \log_2 c_{\max})$ augmenting paths.

3. A useful hint, that you may use, is the following: The outermost While loop 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 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.

4. Note that the running time of one iteration of the inner While loop is $O(m)$. So, if we are able to establish the validity of Step 3, the running time of the algorithm Poly-FF(G, s, t) is $O(m^2 \log_2 c_{\max})$.

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. 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.*

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 now able to complete the above steps, turn over the page.

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

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

   But if you are still now able to complete the above steps, turn over the page.

2. The flow increases by at least k_0. 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.