CS 473 ✧ Spring 2020

Homework 7

Due Wednesday, April 1, 2020 at 9pm

---

1. Suppose you are given a directed graph $G = (V, E)$, two vertices s and t, a capacity function $c : E \to \mathbb{R}^+$, and a second function $f : E \to \mathbb{R}$.

   (a) Describe and analyze an efficient algorithm to determine whether f is a maximum (s, t)-flow in G.

   (b) Describe and analyze an efficient algorithm to determine whether f is the *unique* maximum (s, t)-flow in G.

   Do not assume ***anything*** about the function f.

2. A new assistant professor, teaching maximum flows for the first time, suggested the following greedy modification to the generic Ford-Fulkerson augmenting path algorithm. Instead of maintaining a residual graph, the greedy algorithm just reduces the capacity of edges along the augmenting path. In particular, whenever the algorithm saturates an edge, that edge is simply removed from the graph.

```
GreedyFlow(G, c, s, t):
  for every edge e in G
      f(e) ← 0

  while there is a path from s to t in G
      π ← arbitrary path from s to t in G
      F ← minimum capacity of any edge in π
      for every edge e in π
          f(e) ← f(e) + F
          if c(e) = F
              remove e from G
          else
              c(e) ← c(e) − F

  return f
```

   (a) Prove that GreedyFlow does not always compute a maximum flow.

   (b) Prove that GreedyFlow is not even guaranteed to compute a good approximation to the maximum flow. That is, for any constant $\alpha > 1$, describe a flow network G such that the value of the maximum flow is more than α times the value of the flow computed by GreedyFlow. *[Hint: Assume that GreedyFlow chooses the worst possible path π at each iteration.]*

3. Suppose you are given a flow network G with ***integer*** edge capacities and an ***integer*** maximum flow f^* in G. Describe algorithms for the following operations:

(a) INCREMENT(e): Increase the capacity of edge e by 1 and update the maximum flow.

(b) DECREMENT(e): Decrease the capacity of edge e by 1 and update the maximum flow.

Both algorithms should modify f^* so that it is still a maximum flow, but more quickly than recomputing a maximum flow from scratch.