22**C** : 231 (CS : 5350 : 0001) **Design and Analysis of Algorithms**
Homework 5: Network Flow

The homework has two types of problems – reinforcement problems and regular problems. For the reinforcement problems, we are not concerned with originality in coming up with the solution, but rather with how well you write up the solution. You can get help in coming up with the solution – from friends, online, etc. – but understand the solution and explain it in your own words. For the regular problems, the only type of help you can get is collaboration with classmates, and discussion with the instructor or TA. No record or notes, electronic or written, should be taken from such collaborations. For these problems we do care about originality in coming up with the solution.

The homework is worth 10 points (each problem is worth 2), and is due in class on Tuesday, April 30. **For the purpose of mastering the material, it is highly recommended that you make a serious effort to solve even the reinforcement problems by yourself.**

Reinforcement Problems

1. Exercise 3 of Chapter 7. (Use residual network computation as much as possible to get some practice on that.)
2. Exercise 8 of Chapter 7.
3. Exerices 11 of Chapter 7.
4. Exercise 15 of Chapter 7.

Regular Problems

1. We are given a directed network G with integer capacities on its edges, and two designated nodes s and t in G. In this problem, we consider another variant of the Ford-Fulkerson algorithm (Sections 7.1 and 7.2) for computing a maximum s-t flow in G. Given some flow f in G, consider the residual network G_f. For a path P from s to t in G_f, we define its bottleneck capacity `bottleneck`(P) to be the minimum residual capacity of any edge in P.

   (a) Describe an efficient algorithm that, given G_f, computes a path from s to t in G_f with *maximum* bottleneck capacity (or reports that no path from s to t in G_f exists).

   (b) Suppose we modify the Ford-Fulkerson algorithm so that in each iteration, it updates the current flow f using an s-t path of maximum bottleneck capacity in G_f (rather than just any s-t path in G_f). Show that the modified algorithm terminates in $O(m(1 + \log D))$ iterations, where m is the number of edges in G, and D is the maximum capacity of any edge in G. (We assume that m is at least the number of nodes in G.)