CS 577: Introduction to Algorithms	**Homework 8**
Out: 04/07/17	**Due: 04/14/17**

Ground Rules

- Review problems will be discussed at the following week's review.
- Self-graded problems will be discussed at the following week's review, and are to be self-graded by you during the review. Only students present at the review session and turning in their graded solutions in person will earn points.
- Graded problems are to be turned in on the due date at the beginning of the lecture.
- Self-graded problems should be done individually.
- Graded problems should be done in pairs.
- Please follow page limits for all self-graded and graded problems.
- Write your name(s) clearly on your submissions, and turn in each problem on **separate sheets of paper**.

Review Problems

1. An edge in a flow network is called *lower-binding* if reducing its capacity by one unit decreases the maximum flow in the network. Describe and analyze an algorithm for finding all the lower-binding edges in a network G when given the edge capacities, the source and the sink, as well as a maximum flow f^* in G. Your algorithm should run in time $O(mn)$.

2. Describe and analyze an efficient algorithm to determine whether a given flow network contains a *unique* maximum flow.

Self-graded Problems

3. **(Page limit: 1 sheet; 2 sides)** (Taken from "Algorithms, Etc.") A data stream is an extremely long sequence of items that you can read only once, in order. A good example of a data stream is the sequence of packets that pass through a router. Data stream algorithms must process each item in the stream quickly, using very little memory; there is simply too much data to store, and it arrives too quickly for any complex computations. Every data stream algorithm looks roughly like this:

Algorithm 1: DoSomethingInteresting(stream S)

```
repeat
    x ← next item in S.
    ⟪Do something fast with x⟫.
until S ends;
return ⟪something⟫.
```

Describe and analyze an algorithm that chooses one element uniformly at random from a data stream, *without knowing the length of the stream in advance*. Your algorithm should spend $O(1)$ time per stream element and use $O(1)$ space (not counting the stream itself).

Graded Problems

4. **(Page limit: 2 sheets; 3 sides)** (Taken from "Algorithms, Etc.") Let $M[1 \ldots n, 1 \ldots n]$ be an $n \times n$ matrix in which every row and every column is sorted. Such a matrix is called *totally monotone*. No two elements of M are equal.

   (a) Describe and analyze an algorithm to solve the following problem in $O(n)$ time: Given indices i, j, i', j' as input, compute the number of elements of M smaller than $M[i, j]$ and larger than $M[i', j']$.

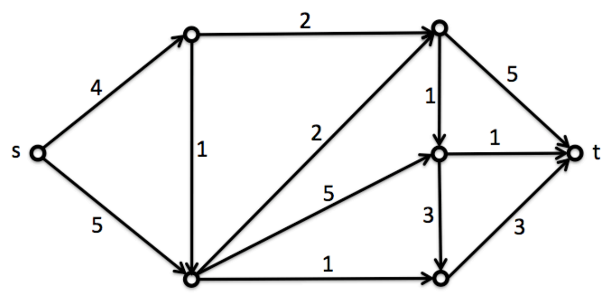
   (b) Describe and analyze an algorithm to solve the following problem in $O(n)$ time: Given indices i, j, i', j' as input, return an element of M chosen uniformly at random from the elements smaller than $M[i, j]$ and larger than $M[i', j']$. Assume the requested range is always non-empty, and that you have a random number generator that returns a uniformly random integer in a specified range.

   (c) Describe and analyze a randomized algorithm to compute the median element of M in $O(n \log n)$ expected time.

   Hint: Use a randomized version of binary search, as in HW 7 problem 3.

5. **(Page limit: 1 sheet; 2 sides)**

   (a) For the network G below determine the max s-t flow, f^*, the residual network G_{f^*}, and a minimum s-t cut.



   (b) An edge in a flow network is called *upper-binding* if increasing its capacity by one unit increases the maximum flow in the network. See problem 1 for the definition of lower-binding edges. Identify all of the upper-binding and all of the lower-binding edges in the above flow network.

   (c) Describe and analyze an algorithm for finding all the upper-binding edges in a network G when given a maximum flow f^* in G. Your algorithm should run in linear time.