CS 473: Fundamental Algorithms, Spring 2011

HW 8

Question 1 is due by **Sunday, 23:59:59, April 3**

Questions 2-4 are due by **Monday, 23:59:59, April 4**

This homework contains four problems. **Read the instructions for submitting homework on the course webpage**.

Collaboration Policy: For this homework, Problems 2–4 can be worked in groups of up to three students.

Problem 1 should be answered in Compass as part of the assessment HW8-Online and should be done individually.

---

1. HW8-Online. (10 pts.)
2. Coins and stuff. (30 pts.)

   (A) (5 pts.) You are given a fair coin that returns heads with probability $1/2$. Prove that there is no algorithm that can perform a constant number of coin flip operations, and output 1 with probability $1/3$ and 0 with probability $2/3$.

   (B) (5 pts.) Describe an algorithm that using a fair coin outputs 1 with probability $1/3$ and 0 with probability $2/3$. The expected number of coin flips of your algorithm uses should be $O(1)$.

   (C) (10 pts.) Describe a simple algorithm that given n picks uniformly in random a number between 1 and n (where all the numbers have the same probability to be picked). The algorithm can use only fair coin flips. How many coin flips does your algorithm uses in expectation? How many coin flips does your algorithm uses in the worst case?

   (D) (10 pts.) Describe an algorithm that given n outputs a random permutation of $\{1, \ldots, n\}$. How many coin flips does your algorithm uses in expectation? Argue why asymptotically (up to a constant) there is no algorithm that uses fewer coin flips (in expectation) that can perform this task.

3. The good, the bad, and the middle. (30 pts.)

   Suppose you're looking at a flow network G with source s and sink t, and you want to be able to express something like the following intuitive notion: Some nodes are clearly on the "source side" of the main bottlenecks; some nodes are clearly on the "sink side" of the main bottlenecks; and some nodes are in the middle. However, G can have many minimum cuts, so we have to be careful in how we try making this idea precise.

   Here's one way to divide the nodes of G into three categories of this sort.

- We say a node v is ***upstream*** if, for all minimum s-t cuts (A, B), we have $v \in A$ – that is, v lies on the source side of every minimum cut.
- We say a node v is ***downstream*** if, for all minimum s-t cuts (A, B), we have $v \in B$ – that is, v lies on the sink side of every minimum cut.
- We say a node v is ***central*** if it is neither upstream nor downstream; there is at least one minimum s-t cut (A, B) for which $v \in A$, and at least one minimum s-t cut (A', B') for which $v \in B'$.

Give an algorithm that takes a flow network G and classifies each of its nodes as being upstream, downstream, or central. (Hint: The running time of your algorithm should be within a constant factor of the time required to compute a *single* maximum flow.)

4. Maximum Flow By Doubling. (30 pts.)

Let $G = (V, E)$ be a flow network with source s, sink t, and an integer capacity $c(u, v)$ on each edge $(u, v) \in E$. Let $C = \max_{(u,v)\in E} c(u, v)$.

(A) (4 pts.) Argue that a minimum cut of G has capacity at most $C|E|$.

(B) (6 pts.) For a given number K, show that an augmenting path of capacity at least K can be found in $O(|E|)$ time, if such a path exists.

The following modification of the Ford-Fulkerson can be used to compute a maximum flow in G.

```
algMaxFlowDoubling($G, s, t$)
    $C \leftarrow max_{(u,v)\in E} c(u, v)$
    initialize flow $f$ to 0
    $K \leftarrow 2^{\lfloor \lg C \rfloor}$
    while $K \geq 1$ do      (*)
        while ( $\exists$ augmenting path $p$ of capacity at least $K$) do
            augment flow $f$ along $p$

        $K \leftarrow K/2$
    return $f$
```

(C) (4 pts.) Argue that **algMaxFlowDoubling** returns a maximum flow.

(D) (6 pts.) Show that the capacity of a minimum cut of the residual graph G_f is at most $2K|E|$ each time line (*) is executed.

(E) (5 pts.) Argue that the inner **while** loop is executed $O(|E|)$ times for each value of K.

(F) (5 pts.) Conclude that **algMaxFlowDoubling** can be implemented so that it runs in $O\big(|E|^2 \lg C\big)$ time.