CS 466/666, Spring 2011
Timothy Chan

Assignment 3 (due July 6 Wednesday 5pm)

Please read `http://www.student.cs.uwaterloo.ca/~cs466/policies.html` first for general instructions.

1. [*16 marks*]

   (a) [*5 marks*] Consider the following problem: given a directed graph $G = (V, E)$ with n vertices, decide whether G contains a directed cycle of length 5. Give a deterministic algorithm that solves this problem in $O(n^{2.376})$ time.

   (b) [*11 marks*] Consider the following problem: given an *undirected* graph $G = (V, E)$ with n vertices, decide whether G contains a simple cycle of length 5. (A cycle is simple if no vertices appear more than once in the cycle.) Give a Monte Carlo algorithm that solves this problem in $O(n^{2.376})$ time with error probability at most 0.001.

   [Hint: orient the edges according to some fixed vertex ordering, and solve a problem similar to (a) for the resulting directed graph. How should we choose the ordering?]

2. [*13 marks*] Consider the following Monte Carlo algorithm to sort a given array $A[1, \ldots, n]$, for a parameter d:

```
1.  repeat d times {
2.      i =random(1, n)
3.      j =random(1, n)
4.      if i < j and A[i] > A[j] then
5.          swap A[i] and A[j]
    }
```

   (a) [*4 marks*] Let Φ be the current number of pairs (k, ℓ) with $k < \ell$ and $A[k] > A[\ell]$. Show that if line 4 holds and line 5 is executed, then Φ decreases by at least 1.

   (b) [*4 marks*] Fix a moment in time during the execution of the algorithm. Suppose that $\Phi = k$ currently. Show that the expected number of iterations in the repeat loop till the next swap (line 5) is executed is $O(n^2/k)$.

   (c) [*5 marks*] Describe how to set d to guarantee that the algorithm solves the sorting problem with error probability at most 0.001. [Hint: use (b). Somehow, the resulting sum should involve Harmonic numbers... Aside: would you recommend this algorithm to practitioners?]

3. [*12 marks*] Suppose $m \geq n^2$. We are given an $m \times n$ matrix, where each of the m rows is sorted in increasing order. Present a Las Vegas algorithm that finds the median among all mn elements in $O(m \log n)$ expected time.

   [Hint: use a variant of an algorithm from class. You may use the following facts mentioned in class: if we take a random subset R of size r from a set S of size N, and compute the $(r/2 - c\sqrt{r})$-th smallest a and the $(r/2 + c\sqrt{r})$-th smallest b of R, then with probability $1 - O(1/c^2)$, the interval $[a, b]$ contains the median of S and has at most $O(cN/\sqrt{r})$ elements of S. How can we we enumerate all elements between a and b in the matrix?]

4. [*19 marks*] Consider the following problem (one of the numerous variations of "coin weighing puzzles"): We are given a set of n coins. Most of the coins are *genuine* and have the same weight, say, 1 unit, but there may be up to 2 *fake* coins, which have weight different from 1 unit. We have a scale that can weigh an any given subset of coins, i.e., determine the exact sum of the weights in the subset. The problem is to decide whether there are any fake coins (you are *not* required to find them, but just decide existence).

   [What makes the problem nontrivial is that the adversary/counterfeiter could create 1 fake coin of weight $1 + \delta$ and another of weight $1 - \delta$, so that we would not be able to detect the existence of fake coins if we simply weigh the whole set (the scale would report total weight n).]

   (a) [*5 marks*] Give a deterministic algorithm that solves the problem using $O(\log n)$ weighings. [Hint: write the indices in binary. If two indices are different, there must be a bit position where they are different...]

   (b) [*7 marks*] Prove, via an adversary argument, that any deterministic algorithm must require $\Omega(\log n)$ weighings in the worst case.

   (c) [*7 marks*] Give a Monte Carlo algorithm that solves the problem using $O(1)$ weighings with error probability at most 0.001. [Hint: take a random subset...]

   [Bonus: suppose we change the problem to allow for an arbitrary number of fake coins. Determine the asymptotic worst-case number of weighings required for deterministic algorithms (2 marks), and show that with randomization, the result in (c) still holds (2 marks).]