CS648 : Randomized Algorithms
Semester I, 2011-12, CSE, IIT Kanpur

Assignment - 2 (due on 23 September : 9AM)

Note: Give complete details of the analysis of your solution. Be very rigorous in providing any mathematical detail in support of your arguments. Also mention any Lemma/Theorem you use.

1. **How to use random sampling ?**

   NOTE : *this exercise will help you realize the following well known lesson about randomization: random sampling can be sometimes used to achieve a goal which is inuitively promising but there does not seem to be any deteriministic way to achieve the same.*

   $G = (V, E)$ is an undirected weighted graph on $n = |V|$ vertices and $m = |E|$ edges. A subgraph (V, E_S) is said to be 3-approximate distance preserver if for each pair of vertices $u, v \in V$, the distance between u and v in the subgraph is at most 3 times their distance in the original graph G. We have to design a Las Vegas algorithm to compute a 3-approximate distance preserver which has $O(n^{3/2})$ edges. The intuition underlying the algorithm for achieving sparseness is the following.

   If a vertex has $\leq n^{1/2}$ edges, we can afford to add all its edges. The main problem is caused by those vertices which have much large number of edges. We use clustering *of vertices to achieve it. We select a small subset of vertices $A \subset V$ and cluster the vertices of the graph around them. For each unclustered vertex, just add all their edges to E_S For each clustered vertex, add least weight edge per neighboring cluster to E_S.*
   Help : hints will be provided in some doubt clearing session.

2. **Random choices may work!**

   Consider a uniform rooted tree of height h - every leaf is at distance h from the root. The root, as well as any internal node, has three children. Each leaf has a Boolean value associated with it. Each internal node returns the value returned by the majority of its children. The evaluation problem consists of determining the value returned by the value of the root.

   (a) (optional) Show that for each deterministic algorithm, there is an instance (a set of Boolean values for the leaves) that forces it to read all $n = 3^h$ leaves.

   (b) Design a randomized Las Vegas algorithm for this problem which reads expected $O(n^{0.9})$ leaves.

3. **Two-dimensional Pattern Matching**

   (a) In this problem we will use a different finger printing technique to solve the pattern matching problem. The idea is to map any bit string s into a 2×2 matrix $M(s)$, as follows.

      i. For the empty string ϵ, $M(\epsilon) = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$

      ii. $M(0) = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$

      iii. $M(1) = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$

      iv. For non-empty strings x and y, $M(xy) = M(x) \times M(y)$.

   Show that this fingerprint function has the following properties.

i. $M(x)$ is well defined for all $x \in \{0,1\}^*$.

ii. $M(x) = M(y) \Rightarrow x = y$.

iii. For $x \in \{0,1\}^n$, the entries in M(x) are bounded by Fibonacci number F_n.

By considering the matrices $M(x)$ modulo a suitable prime p, show how you would perform efficient randomized pattern matching. Explain how you would implement this as a "real-time" algorithm.

(b) Consider the two-dimensional version of the pattern matching problem. The text is an $n \times n$ matrix X, and the pattern is an $m \times m$ matrix Y. A pattern match occurs if Y appears as a (contiguous) sub-matrix of X. To apply the randomized algorithm described above, we convert the matrix Y into an m^2-bit vector using the row-major format. The possible occurences of Y in X are the m^2-bit vectors $X(j)$ obtained by taking all $(n-m+1)^2$ sub-matrices of X in a row-major form. It is clear that the earlier algorithm can now be applied to this scenario. Analyze the error probability in this case, and explain how the fingerprints of each $X(j)$ can be computed at a small incremental cost.

4. **Approximate Ham-sandwich cut**

Given n red points and n blue points in a plane, a line L is called ham-sandwich cut if it simultaneously bisects the red points as well as the blue points, that is, there are $n/2$ red (as well as blue) points on each of the two sides of the line.

There is a deterministic algorithm for this problem which uses point line duality concept and is quite nontrivial. For all practical purposes, even a slightly weaker version of the ham-sandwich cut, defined below, also works equally well.

a line L is said to be $(1+\epsilon)$-approximate ham-sandwich cut if the number of red (as well as blue) points on each side of the line L is at most $(1+\epsilon)n/2$.

You have to design an $O(n)$ time randomized Monte Carlo algorithm which computes an $(1+\epsilon)$-approximate ham-sandwich cut with probability $1 - n^{-c}$ for any given constant $c > 0$.
Hint: Think simple !!