CS648 : Randomized Algorithms
CSE, IIT Kanpur

Practice sheet

Contents:

1. How and when can we convert a Monte Carlo algorithm of a problem to a Las Vegas algorithm ?
2. A simple by efficient coloring algorithm.
3. An example problem where randomization beats deterministic algorithm.

1. **Monte Carlo to Las Vaga algorithm**

   Suppose Q is an algorithmic problem. Let I be any instance of problem Q of size n.

   There exists a Monte Carlo algorithm $\mathcal{M}$ for Q with the following property.

   The Monte Carlo algorithm runs in $\mathcal{O}(f(n))$ time on I and outputs a correct solution with probability p.

   You are also given a deterministic algorithm $\mathcal{V}$ that takes as input an instance I of problem Q and a proposed solution of I. It can verifies whether the proposed solution of instance I is a correct solution of I. The running time of this algorithm on I is $\mathcal{O}(g(n))$.

   Design an efficient Las Vegas algorithm of problem Q using algorithms $\mathcal{M}$ and $\mathcal{V}$. What is the expected running time of this algorithm.

2. **3-Coloring a graph**

   Suppose you are given a graph $G = (V, E)$, and we want to color each node with one of three colors. An edge (u, v) is said to be satisfied in a 3-coloring if the colors assigned to u and v are different.

   Let c^* be the maximum number of satisfied edges by a 3-coloring. Design a randomized algorithm that produces a 3-coloring such that expected number of satisfied edges is at least $\frac{2}{3}c^*$.
   Hint: Think of an algorithm that uses randomization in the simplest possible manner.

3. **A randomized algorithm beats deterministic algorithm.**

   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. For example, if two children return 0 and the third child returns 1, then the value returned by the node will be 0. The evaluation problem consists of determining the value of the root; at each step, an algorithm can choose one leaf whose value it wishes to read.

   Design a randomized algorithm for this problem such that the expected number of leaves read by the algorithm is $O(n^{1-c})$ for some constant c such that $c > 0$.

   Those whose aim is more than A^* should try to solve the following problem as well. Show that for any deterministic algorithm, there exists an instance (a set of Boolean values for the leaves) that forces it to read all $n = 3^h$ leaves.

   Hint: Focus on a tree with 3 leaves only. Design a randomized algorithm to evaluate it by querying fewer than 3 leaves on expectation. Now try to generalize ...