5.2 Randomized Algorithms and Random Inputs

Las-Vegas algorithms, unlike Monte-Carlo, never give wrong answers. Unlucky coin-flips just make them run longer than expected. Quick-Sort is a simple example. It is about as fast as deterministic sorters, but is popular due to its simplicity. It sorts an array $a[1 \dots n]$ of $n > 2$ numbers by choosing in it a random ***pivot***, splitting the remaining array in two by comparing with the pivot, and calling itself recursively on each half.

For easy reference, rename the array entries with their positions $1, \dots, n$ in the *sorted output* (no effect on the algorithm). Denote $t(i)$ the (random) time i is chosen as a pivot. Then i will ever be compared with j iff either $t(i)$ or $t(j)$ is the smallest among $t(i), \dots, t(j)$. This has 2 out of $|j-i|+1$ chances. So, the expected number of comparisons is $\sum_{i,j>i} 2/(1+j-i) = -4n + (n+1)\sum_{k=1}^{n} 2/k = 2n(\ln n - O(1))$. Note, that the expectation of the sum of variables is the sum of their expectations (not true, say, for product).

The above Monte-Carlo and Las-Vegas algorithms require choosing strings *at random* with uniform distribution. We mentally picture that as flipping a coin. (Computers use ***pseudo-random generators*** rather than coins in hope, rarely supported by proofs, that their outputs have all the statistical properties of truly random coin flips needed for the analysis of the algorithm.)

Random Inputs to Deterministic Algorithms are analyzed similarly to algorithms that flip coins themselves and the two should not be confused. Consider an example: Someone is interested in knowing whether or not certain graphs contain Hamiltonian Cycles. He offers graphs and pays \$100 if we show either that the graph *has* or that it *has not* Hamiltonian Cycles. Hamiltonian Cycle problem is NP-Complete, so it should be very hard for *some*, but not necessarily for *most* graphs. In fact, if our patron chooses the graphs uniformly, a fast algorithm can earn us the \$100 *most of the time*! Let all graphs have n nodes and, say, $d < (\ln n)/2$ mean degree and be equally likely. Then we can use the following (deterministic) algorithm:
Output "**No** Hamiltonian Cycles" and collect the \$100, if the graph has an isolated node. Otherwise, pass on that graph and the money. Now, how often do we get our \$100. The probability that a given node A of the graph is isolated is $(1-1/n)^{dn} > (1-O(1/n))/\sqrt{n}$. Thus, the probability that *none* of n nodes is isolated (and we lose our \$100) is $O((1-1/\sqrt{n})^n) = O(1)/e^{\sqrt{n}}$ and vanishes fast. Similar calculations can be made whenever $r = \lim(d/\ln n) < 1$. If $r > 1$, other fast algorithms can actually find a Hamiltonian Cycle.
See: [Johnson 84, Karp 76, Gurevich 85]. See also [Levin Venkatesan 18] for a proof that another graph problem is NP-complete even on average. How do this HC algorithm and the above primality test differ?

- The primality algorithm works for *all* instances. It tosses the coin itself and can repeat it for a more reliable answer. The HC algorithms only work for *most* instances (with isolated nodes or generic HC).
- In the HC algorithms, we must trust the customer to follow the presumed random procedure. If he cheats and produces rare graphs often, the analysis breaks down.

Symmetry Breaking. Randomness comes into Computer Science in many other ways besides those we considered. Here is a simple example: avoiding conflicts for shared resources.

Dining Philosophers. They sit at a circular table. Between each pair is either a knife or a fork, alternating. The problem is, neighboring diners must share the utensils, cannot eat at the same time. How can the philosophers complete the dinner given that all of them must act in the same way without any central organizer? Trying to grab the knives and forks at once may turn them into fighting philosophers. Instead they could each flip a coin, and sit still if it comes up heads, otherwise try to grab the utensils. If two diners try to grab the same utensil, neither succeeds. If they repeat this procedure enough times, most likely each philosopher will eventually get both a knife and a fork without interference.

We have no time to actually analyze this and many other scenaria, where randomness is crucial. Instead we will take a look into the concept of Randomness itself.