Assignment II

Jan. 2013 : Combinatorial Algorithms

1. We have seen the expected number of steps required for the randomized 2 SAT algorithm in class. Write down the recurrence equation for the case when the formula contains exactly 3 literals per clause. What is the expected number of steps required for solving 3 SAT with the same random walk strategy?

2. Suppose you toss an n headed fair dice. If the outcome is r for some $1 \leq r \leq n$, next time throw an r headed dice. Repeat this experiment till you get a 1 as outcome. What is the expected number of throws required? (Let X_n denote the random variable that counts the number of throws necessary starting with a n headed dice. You can write a recurrence for $E(X_n)$ which will contain terms involving $E(X_k)$ for each $1 \leq k \leq n$. Write a similar recurrence for $E(X_{n-1})$ and Subtract equation for $E(X_{n-1})$ from that of $E(X_n)$ to cancel out terms and get an easy recurrence equation).

3. This is a re-visit on the average case analysis of the quick-sort algorithm using the method of conditional expectations. Given n elements to sort, Let X_n denote a random variable denoting the total number of comparisons required to sort n elements. the *Quicksort* algorithm runs a partitioning algorithm requiring n comparisons after choosing a random pivot. Suppose the pivot is the i^{th} element in order (an event that occurs with probability $\frac{1}{n}$), then after pivoting you are left with sorting a set of $i-1$ elements and another set of $n-i$ elements. The problem is to set up a recurrence for $E(X_n)$ using conditional expectations. Solve the recurrence.

4. There are n bins. Balls are thrown to the bins till two bins contain at least one ball. What is the expected number of throws required? Generalize this to yield the expected number of throws required for every bin to have at least one ball. (Suppose $i-1$ bins contain at least one ball. Let X_i denote the number of throws required for a new bin to get a ball. X_i follows geometric distribution. Let $X = \sum_{i=1}^{n} X_i$.)

5. There are n men and n women wishing to find a partner. Each man writes down the names of all n women in the order of his preference. Similarly each women writes down the names of all n men according to her preference. The problem here is to find a '*stable marriage*. A marriage is a bijective function mapping men to women satisfying the following: If we marry man i to women x and man j is married to women y, it must not be the case that both i has higher preference for y over x and y has higher preference for i than her assigned partner j according to the preference list. The problem is to find a stable marriage (or stable matching).

   The *Gale Shapely* algorithm does the following. Initially all men are in a queue and unmarried. All women also are unmarried and no proposals are made. At each step, the man in the front of the queue proposes to the first women in his list to whom he has not already proposed. If the women is unmarried, they are paired temporarily. If the women is already paired and if the present proposal has higher preference for the women than her existing partner, then the existing partner is de-paired from her and he goes to the rear of the queue. The new proposer is paired with her temporarily. Finally, if the existing partner is of higher preference to the women, the new proposer is continues proposing to the next women in the list.

   - Argue that a women paired once remains paired for ever.
   - Prove that a new women gets paired after at most n proposals.
   - Hence argue that the algorithm requires at most $O(n^2)$ proposals before termination.
   - Show that the marriage produced at the time of termination is stable.
   - Analyze the algorithm in the average case, assume that preference lists are uniformly at random. (It takes a sequence of proposals for a new women to be paired, but once paired the women remains paired for the rest of the algorithm. Observe that this is similar to throwing balls to bins till a ball lands into a new bin.)

6. Describe how you can de-randomize the LP based algorithm discussed in the class for MAXSAT using the method of conditional expectations.

7. Let F_2 denote the binary field. Suppose you are given m equations over n binary variables with $m > n$. The problem is to find an assignment to the variables that satisfies maximum number of equations. The problem is NP hard when $m > n$ (and polynomial time solvable otherwise - why?). Devise a randomized $\frac{1}{2}$ approximation algorithm for the problem. De-randomize your algorithm using the method of conditional expectations.

8. This question develops a proof for *Jensen's inequality*. Let f be a real valued *convex* function defined on some closed interval $[a, b]$ of the real line. Let $x_1, x_2, ..., x_n$ be points in $[a, b]$ and let $p_1, p_2, ..p_n$ be positive numbers such that $p_1 + p_2 + ... + p_n = 1$. Use induction to show that $f(p_1x_1 + p_2x_2 + ... + p_nx_n) \leq p_1f(x_1) + p_2f(x_2) + ... + p_nf(x_n)$.

9. Consider the problem of finding a large independent set in a graph $G = (V, E)$. Suppose V has n vertices and let for each vertex $v \in V$, d_v denote the degree of V. Let D denote the average degree of the graph. Consider the following algorithm: generate a random permuation of $\{1, 2, .., n\}$ and assign each vertex with a (unique random) number from the set. Now, if a vertex v is assigned a number smaller than all its neighbhours, pick vertex v into the indepenent set. Show that the independent set picked has size at least $\sum_{v \in V} \frac{1}{d_v+1} \geq \frac{n}{D+1}$ (Use Jensen's inequality). Read and understand the following link to figure out how to de-randomize this algorithm (This question is important for the final examination).. http://cmurandomized.wordpress.com/2011/01/27/notes-on-lecture-6/