

CSE 548 Homework Assignment 4

Due in Class on Thursday, November 12

November 15, 2009

There are four problems. Generic format of the solution is as follows:

1. Discuss your idea briefly.
2. State all the assumptions about the problems and meanings of random variables clearly.

Problems

1. Solve the following problems
 - a. This is a fairly common calculation done for routers on networks. As our router processes large number of packets, we want to understand some basic statistical properties of the traffic. We have independent Bernoulli random variables $X_1, X_2, X_3 \dots X_i \dots$ that represents the fact if packet arrived at time $t_1, t_2, t_3 \dots t_i \dots$. Given the same flow, lets simplify and say the parameter for all these variables is the same - p . ie $\text{Prob}[X_i = 1] = p$ and $\text{Prob}[X_i = 0] = 1 - p$.
Lets denote with S_n , the number of packet arrivals from time 1 to n . And let T_n be the number of time slots after $(n-1)$ th arrival until and including the n th arrival. Let Y_n be the total number of time slots from the beginning until and including the n th arrival. Find the distributions, expectations and variance of S_n, T_n and Y_n .
 - b. A permutation of S_n is a bijection $\pi : [n] \rightarrow [n]$, where $[n] = 1, 2, \dots, n$. The set $[n]$ can be viewed as vertices of a directed graph, where arcs of the form $(i, \pi(i))$. Every permutation corresponds to a disjoint set of cycles on this directed graph. Some cycles may be self loops, called fixed points of the permutation. Suppose we choose a permutation π from S_n at random uniformly:
What is the expected number of fixed points? Further what is the expected number of cycles?
2. Clustering is an important tool in wireless sensor networks. Its employed for a variety of reasons: congestion control, data aggregation etc. Suppose we have n sensors at n points: p_1, p_2, \dots, p_n . These sensors have some weights attached to them in the context of the application: w_1, w_2, \dots, w_n . We wish to find a location/point - p for the cluster center of these sensors. This point need not be one of p_1, p_2, \dots, p_n . But the point must minimize the sum $\sum_{i=1}^n w_i d(p, p_i)$ where $d(p, p_i)$ is some notion of distance between points p and p_i .
 - a. In case that all sensors line on a straight line in a simple Euclidian space, show how can we calculate the location of the cluster center of the points - p in $O(n \lg n)$ time. How about $O(n)$? In this setting, distance between two points a and b $d(a, b) = |a - b|$.
 - b. Repeat the above analysis when the sensors are placed in a grid like fashion and messages can travel only northwards, southwards, westwards and eastwards. That is the notion of distance in this setting is that of the Manhattan distance. If the co-ordinates of point a are (x_1, y_1) and that of point b are (x_2, y_2) , $d(a, b) = |x_1 - x_2| + |y_1 - y_2|$.
3. Consider the sorting problem where all the numbers are not known exactly. Instead, for each number, you know an interval on the real line to which it belongs. That is, you are given n closed intervals of the form $[a_i, b_i]$ where $a_i \leq b_i$. Assume that no interval completely contains any other

interval, that is if $a_i \leq a_j$, then $b_i \leq b_j$. You are asked to sort these intervals, that is produce a permutation $\langle i_1, i_2, \dots, i_n \rangle$ of intervals such that for all j , there exists a $c_j \in [a_{i_j}, b_{i_j}]$ stisfying $c_1 \leq c_2 \leq c_3 \dots \leq c_n$.

Suppose that each interval is guranteed to overlap atleast $d - 1$ intervals. Give an $O(n \log \frac{n}{d})$ algorithm to sort these intervals with d degrees of overlap.

4. Random anonymous chatting is another interesting development in the social networking domain. You work for one such firm called *hello-der!.com*. The site basically works like this: the user logs in and issues a “connect” command. Your chat server instantly connects him/her to some other completely random user who is currently logged in and has issued a “connect” command. So for you, your “connect” is equally likely to land up with any other user. Each subsequent “connect” is issued independently.

What are the expected number of *connects*, before your request lands up with a person, you had initially once connected to? That is what are the expected number of *hello-ders!*, till you *hello-der!* the same person again?

Solve this problem in the simple setting that you have issued a connect and there are n other users who can receive the connection uniformly randomly. After how many *connects*, will your “connect” take you to a user whom you had “connected-to” before.