CAS CS 330. Problem Set 10 (Randomized Algorithms)

Problem 1

part 1

The probability that one person can pick out his cap correctly is $\frac{1}{n}$

There are n people going to pick up their caps with probability of $\frac{1}{n}$ each, therefore, the expected number of correct cap = n* $\frac{1}{n}$ = 1

Part 2

Algorithm:

Modified Quicksort (S,C): #S for campers and C for caps

If |S| $\leq$3:

Sort S;

Output the sorted list

Else:

while no central splitter has been found

choose a splitter $c_i \in C$ uniformly at random

For each element a_j in S:

put a_j in S^- if the cap c_i is too big for a_j

put a_j in S^+ if the cap c_i is too small for a_j

put a_i with the splitter if a_i matches the cap c_i

If |S^-| ≥ |S|/4 and |S^+| ≥ |S|/4 then

c_i is a central splitter

For each cap other than c_j

put c_j in C^- if the cap is too small for a_j

put c_j in C^+ if the cap is too big for a_j

Recursively call Quicksort(S^-, C^-) and Quicksort(S^+, C^+)

Output the sorted set S^-, then a_i, then the sorted set S^+ for people list

Output the sorted set C^- ,then c_i, then the sorted set C^+ for caps corresponding to the people list.

End of the Algorithm

Analyze:

Because we can not compare two heads and two caps directly, in order to separate them, we need to find a way to range them by order. First, find a cap randomly, and let every people try it on, and we will find the owner of the cap a_i eventually. For those people who think the cap is too small, we know that those people have a bigger head than a_i , so we can put them in list S^+ . For those people who think the cap is too big, we know that they have a smaller head and we put them in list S^- . In this way, we would be able to create two subgroups by people's head size.

After that, we can use similar way to split caps. Let a_i try on all the rest caps, if it is too small for a_i , then we know that this cap belongs to people in S^- . On the other hand, if the cap is too big for a_i , then we know that this cap belongs to people in S^+ . In this way, we would be able to separate caps into two subgroups as well. Then we just need to do the same thing recursively.

In order to get a good splitter a_i , i use a while loop to choose a_i until we find a good splitter.

Running Time:

Dividing caps and campers into two subgroup will be in Theta(2n), and finding a good splitter will repeat this several time if we have a bad luck, but the running time is still in Theta(n). In the Recursive part, we keep cutting the size down to approximately half because we choose a good splitter. We need to run two subgroups of size half on each split, so the running time for this part will be Theta(logn). Therefore, the total running time is Theta(nlogn)

Problem 2

Algorithm:

If we have N candidates, randomly select N/2 candidates, or select first N/2 candidates. Reject all N/2 candidates, compute the Rank and store the highest Rank r. For the rest N/2 candidates, go through them one by one, if the Rank is higher than r, then marry this person, otherwise, go to see next person.

Analyze:

We skip the first half candidates in order to compute the sample maximum, from this subgroup, we will know what are these candidates like, and what kind of candidate is a good choice. Among all the candidates, we only care about two candidates: the candidate has the highest rank and the candidate has the second highest rank. We want to have the second-highest-rank candidate in our sample, and the highest-rank candidate in the rest subgroup. Therefore, we would be able to know the second highest Rank by compute the sample maximum. Then we can determine the highest-rank candidate compare the rest with our maximum.

Because we are taking half candidates as sample, the probability of having second-highest-rank candidate in our sample would be ½. The probability that given the second-highest-rank candidate in group one and highest-rank candidate in group two would be ½.

x1 = highest-rank candidate

x2 = second-highest-rank candidate

Pr[x2] = ½

Pr [x1 | x2] = ½

Therefore, the probability of finding the highest-rank candidate = ½ * ½ = ¼