1 A Variation of the Hiring Problem

We now consider a variation of the hiring problem we studied in class. As before, you have $n$ job candidates with distinct skills (i.e. no ties in the skill level), and we already randomly shuffle these candidates (so that each of the $n!$ order of the candidates in terms of their skill level is equally likely). The difference is that in this problem only one hiring is allowed. Note that when we evaluate a candidate, we only know how this candidate ranks against the previously seen candidates. Also we have to decide whether to hire or reject right after the interview, and once rejected, the person can not be hired later. We will take the following simple strategy.

1. We first interview and reject the first $k$ candidates (for some carefully chosen $k$).
2. Then, starting from the $k+1$-th candidate, we will hire the first candidate who ranks higher than all the first $k$ candidates. After this hiring, stop.

For example, let $k = 4$. We look at candidate 5, and found this one is not better than the best of the first 4 candidates, and so we reject. Then we found candidate 6 is better than all of the first 4 candidates, we then decide to hire candidate 6. Of course, there is a chance that we will not be able to hire anyone using this strategy. But this is what sometimes happens in the real world.

Now, what is the probability of the best candidate among all the $n$ candidates is hired using this strategy? If you can, try to find for each $n$, what value of $k$ should we choose so that the probability of hiring the best candidate is maximized? This is not required but can be interesting to think about yourself.

2 Heap

Do parts (a), (b) and (c) of Problem 6.2 (p.167) in textbook. For part (a), tell me where the $d$ children (if exists) of a node $i$ will be located in the array, and where the parent of a node $i$ will be in the array.

3 Lower bound

Consider the task of searching a sorted array $A[1 \ldots n]$ for a given element $x$: a task we usually perform by binary search in time $O(\log n)$. Show that any algorithm that accesses the array only via comparisons (that is, by asking questions of the form “is $A[i] \leq z$?”), must take $\Omega(\log n)$ steps.

4 Coin Change

We now prove the simple greedy algorithm for the coin change problem with quarters, dimes, nickels and pennies are optimal (i.e. the number of coins in the given change is minimized), when the supply of each coin type is unlimited.

Let $q_o, d_o, k_o, p_o$ be the number of quarters, dimes, nickels and pennies used for changing $n$ cents in an optimal solution.

1. First, show that $d_o \leq 2, k_o \leq 1, p_o \leq 4$. Also, show that if $k_o = 1$, then $d_o \leq 1$. 

2. Now show it is always optimal to choose as many quarters as possible.

3. Finally, show that the greedy algorithm is optimal in its choice for dimes, nickels and pennies.

5 Activity Selection

Let us consider an alternative greedy strategy for the activity selection problem. Let \( \mathcal{R} = \{R_1, \ldots, R_n\} \) be the set of activities. Recall that each activity \( R_i = [s_i, f_i] \).

1: while \( \mathcal{R} \) is not empty do
2: Select the activity \( R_i \in \mathcal{R} \) that overlaps with the fewest activities in \( \mathcal{R} \) and add this activity to solution set \( A \).
3: Remove \( R_i \) and any activity \( R_j \) that overlaps with \( R_i \) from \( \mathcal{R} \).
4: end while

Now prove or disprove that the greedy strategy solves the activity selection problem optimally.