**Implementation of Hiring Problem using multiple position scenario**

Theory:

The hiring problem is a simple model of decision-making under uncertainty

It is closely related to the well-known Secretary Problem

A sequence of n candidates is to be interviewed to fill a post. For each interviewed candidate we only learn about his/her relative rank among the candidates we've seen so far. After each interview, hire and finish, or discard and interview a new candidate. The nth candidate must be hired if we have reached that far. The goal: devise an strategy that maximizes the probability of hiring the best of the n candidates

Hire-Assistant(n)  
  1  best = 0                // fictional least qualified candidate  
  2  for i = 1 to n  
  3    interview candidate i // paying cost  ci  
  4    if candidate i is better than candidate best  
  5      best = i  
  6      hire candidate i    // paying cost ch

Code:

#include <iostream>  
#include <bits/stdc++.h>  
using namespace std;  
  
class Interviewee{  
    public:      
        string name;  
        int talent;  
};  
  
int compute\_cost(Interviewee I[], int n, int ci, int ch){  
    int best = -1;  
    int cost = 0;  
    for(int i=0; i<n; i++){  
        cost+=ci;  
        if(I[i].talent > best){  
            best = I[i].talent;  
            cost+=ch;  
        }  
    }  
    return cost;  
}  
  
int compute\_randomized\_cost(Interviewee I[], int n, int ci, int ch){  
    random\_shuffle(I, I + n);  
    int best = -1;  
    int cost = 0;  
    for(int i=0; i<n; i++){  
        cost+=ci;  
        if(I[i].talent > best){  
            best = I[i].talent;  
            cost+=ch;  
        }  
    }  
    return cost;  
}  
  
  
int main(){  
    cout<<"HIRING PROBLEM"<<endl;  
    cout<<endl;  
  
    int interview\_cost, hiring\_cost, n;  
  
    cout<<"enter interview cost: ";  
    cin>>interview\_cost;  
  
    cout<<"enter hiring cost: ";  
    cin>>hiring\_cost;  
  
    cout<<"enter number of applicants: ";  
    cin>>n;  
  
    cout<<endl;  
    Interviewee I[n];  
    for(int i=0; i<n; i++){  
        Interviewee j;  
        cout<<"Enter name of applicant "<<i<<": ";  
        cin>>j.name;  
        cout<<"Enter talent of applicant "<<i<<": ";  
        cin>>j.talent;  
        I[i] = j;  
        cout<<endl;  
    }  
  
    int cost = compute\_cost(I, n, interview\_cost, hiring\_cost);  
    int randcost = compute\_randomized\_cost(I, n, interview\_cost, hiring\_cost);  
  
    cout<<"Cost: "<<cost<<endl;  
    cout<<"Randomized Cost: "<<randcost<<endl;  
  
    return 0;  
}

Output:

HIRING PROBLEM         
   
enter interview cost: 2  
enter hiring cost: 5  
enter number of applicants: 5  
   
Enter name of applicant 0: Ram  
Enter talent of applicant 0: 10  
   
Enter name of applicant 1: Rahim  
Enter talent of applicant 1: 15  
   
Enter name of applicant 2: Sai  
Enter talent of applicant 2: 8  
   
Enter name of applicant 3: Arjun  
Enter talent of applicant 3: 5  
   
Enter name of applicant 4: Gopal  
Enter talent of applicant 4: 18  
   
Cost: 25  
Randomized Cost: 15

**Observations**:

If the list of people applying for interviews are randomly placed based on their talent value then shuffling their position and then taking the interview will help in minimizing the cost of hiring

**Conclusion**:

Applying randomization over the pool of candidates will help to minimize the cost

before hiring cost is O(m\*ch) where ch=cost of hiring

After randomization it is O(ln n).

**References**:

<https://www.cs.upc.edu/~conrado/research/talks/sem-UCT-hiring.pdf>

<http://www2.hawaii.edu/~suthers/courses/ics311f20/Notes/Topic-05.html>