ADVANCE ALGORITHM
Experiment 2

Ayush Jain 60004200132 B3

Aim : To implement Hiring Problem.

Theory:
Amortized Analysis.

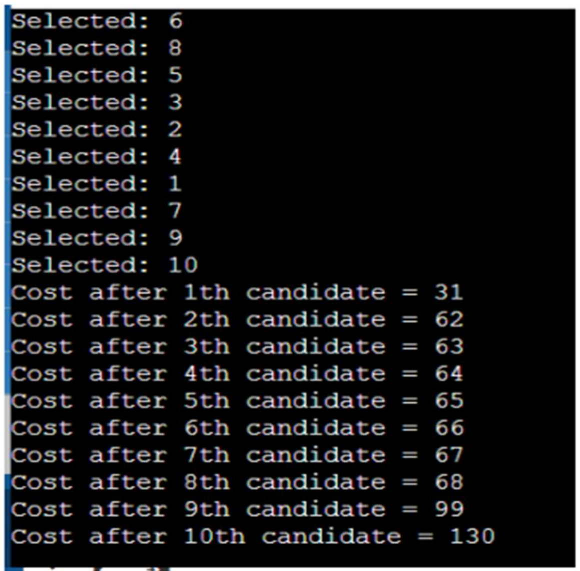
Hiring Problem. The hiring problem is a classic problem in probability theory and decision theory that involves selecting the best candidate from a pool of applicants who arrive sequentially. The goal is to maximize the probability of selecting the best candidate while minimizing the expected number of applicants that are interviewed. One common approach to the hiring problem is to use the "optimal stopping" or "secretary problem" strategy. This strategy involves interviewing a fixed number of applicants (n) and then selecting the best candidate out of the n. The value of n is determined by a formula that depends on the total number of applicants and the probability distribution of their quality. Another approach to the hiring problem is to use a Bayesian decision-theoretic framework. This approach involves updating a prior probability distribution on the quality of the applicants as each new applicant is interviewed. The decision to hire a candidate is then based on the updated probability distribution and a decision threshold that balances the cost of making a mistake (hiring a suboptimal candidate) with the cost of delaying the decision. Other decision-making strategies have also been proposed for the hiring problem, including the "rank and select" strategy, which involves ranking the candidates as they arrive and then selecting the best candidate from a fixed number of the top-ranked candidates, and the "threshold acceptance" strategy, which involves setting a threshold quality level for the applicants and accepting the first applicant who meets or exceeds that threshold. In practice, the optimal strategy for the hiring problem may depend on the specific details of the hiring situation, such as the size of the applicant pool, the distribution of applicant qualities, and the cost of interviewing candidates.

Code:

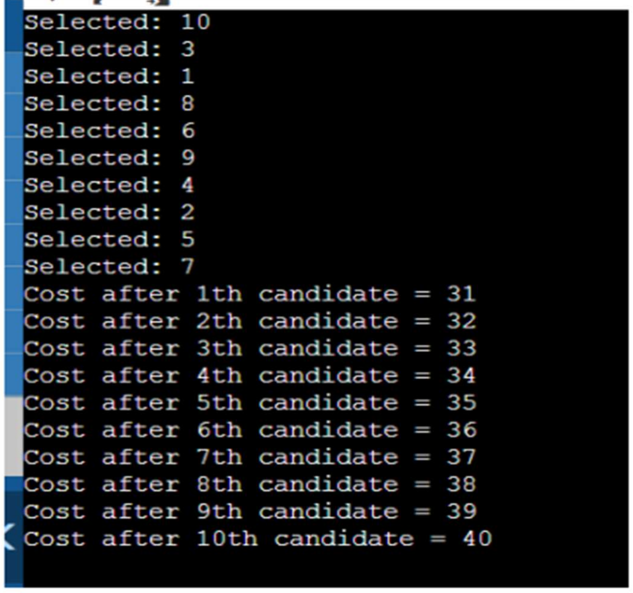
```
import java.util.*;

public class Main
{
public static void main(String[] args) {
int order[]=new int[10];
int k=0,hc=20,fc=10,threshold=3,cost=0;
ArrayList<Integer> list = new ArrayList<Integer>(10);
 for(int i = 1; i <= 10; i++) {
 list.add(i);
 }
 Random rand = new Random();
 while(list.size() > 0) {
 int index = rand.nextInt(list.size());
 System.out.println("Selected: "+list.get(index));
 order[k++]=list.remove(index);
 }
 for(int j=0;j<10;j++){
 cost++;
 if(order[j]>threshold){
 cost+=hc+fc;
 threshold=order[j];
 }
 System.out.println("Cost after "+(j+1)+"th candidate = "+cost);
 }
}
}
```

Output:



```
Selected: 6
Selected: 8
Selected: 5
Selected: 3
Selected: 2
Selected: 4
Selected: 1
Selected: 7
Selected: 9
Selected: 10
Cost after 1th candidate = 31
Cost after 2th candidate = 62
Cost after 3th candidate = 63
Cost after 4th candidate = 64
Cost after 5th candidate = 65
Cost after 6th candidate = 66
Cost after 7th candidate = 67
Cost after 8th candidate = 68
Cost after 9th candidate = 99
Cost after 10th candidate = 130
```



```
Selected: 10
Selected: 3
Selected: 1
Selected: 8
Selected: 6
Selected: 9
Selected: 4
Selected: 2
Selected: 5
Selected: 7
Cost after 1th candidate = 31
Cost after 2th candidate = 32
Cost after 3th candidate = 33
Cost after 4th candidate = 34
Cost after 5th candidate = 35
Cost after 6th candidate = 36
Cost after 7th candidate = 37
Cost after 8th candidate = 38
Cost after 9th candidate = 39
Cost after 10th candidate = 40
```

Conclusion: In conclusion, we learned the Hiring Problem where by randomization we can decrease the cost for hiring a best candidate