EXPERIMENT 8

Student Name: Praduman Kumar

UID: 20BCS9446

Branch: CSE

Section/Group: 20BCS_DM_714-A

Semester: 06

Subject Name: Competitive Coding

Subject Code: 20CSP-351

1. **AIM:** To demonstrate the concept of Greedy Approach

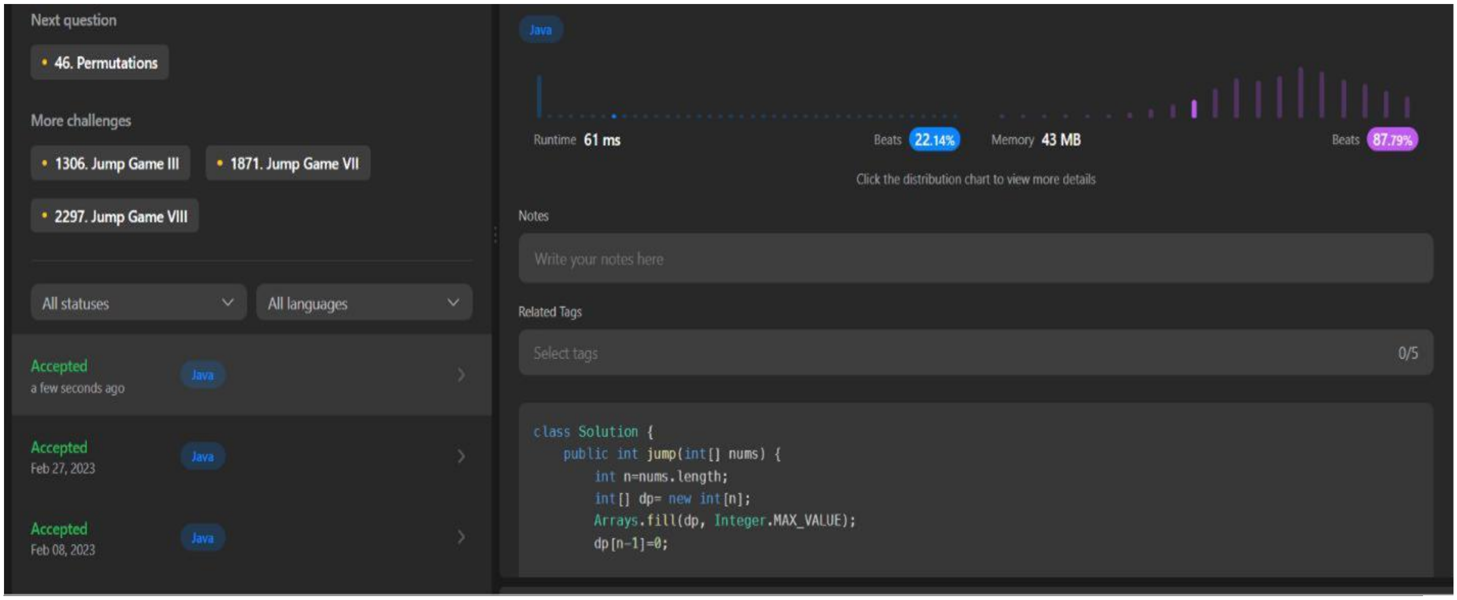
2. **OBJECTIVE 1:** Jump Game 2

3. **CODE:**

```
class Solution {
    public int jump(int[] nums) {
        int n=nums.length;
        int[] dp= new int[n];
        Arrays.fill(dp, Integer.MAX_VALUE);
        dp[n-1]=0;

        for(int i=n-2;i>=0;i--){
            int min= Integer.MAX_VALUE;
            for(int j=i+1;j<=Math.min(n-1,i+nums[i]);j++){
                min=Math.min(min,dp[j]);
            }
            if(min!=Integer.MAX_VALUE)  dp[i] = min+1;
        }
        return dp[0];
    }
}
```

4. OUTPUT:



Accepted a few seconds ago Java

Accepted Feb 27, 2023 Java

Accepted Feb 08, 2023 Java

Runtime 61 ms Beats 22.14%

Memory 43 MB Beats 87.79%

```
class Solution {
    public int jump(int[] nums) {
        int n=nums.length;
        int[] dp= new int[n];
        Arrays.fill(dp, Integer.MAX_VALUE);
        dp[n-1]=0;
```

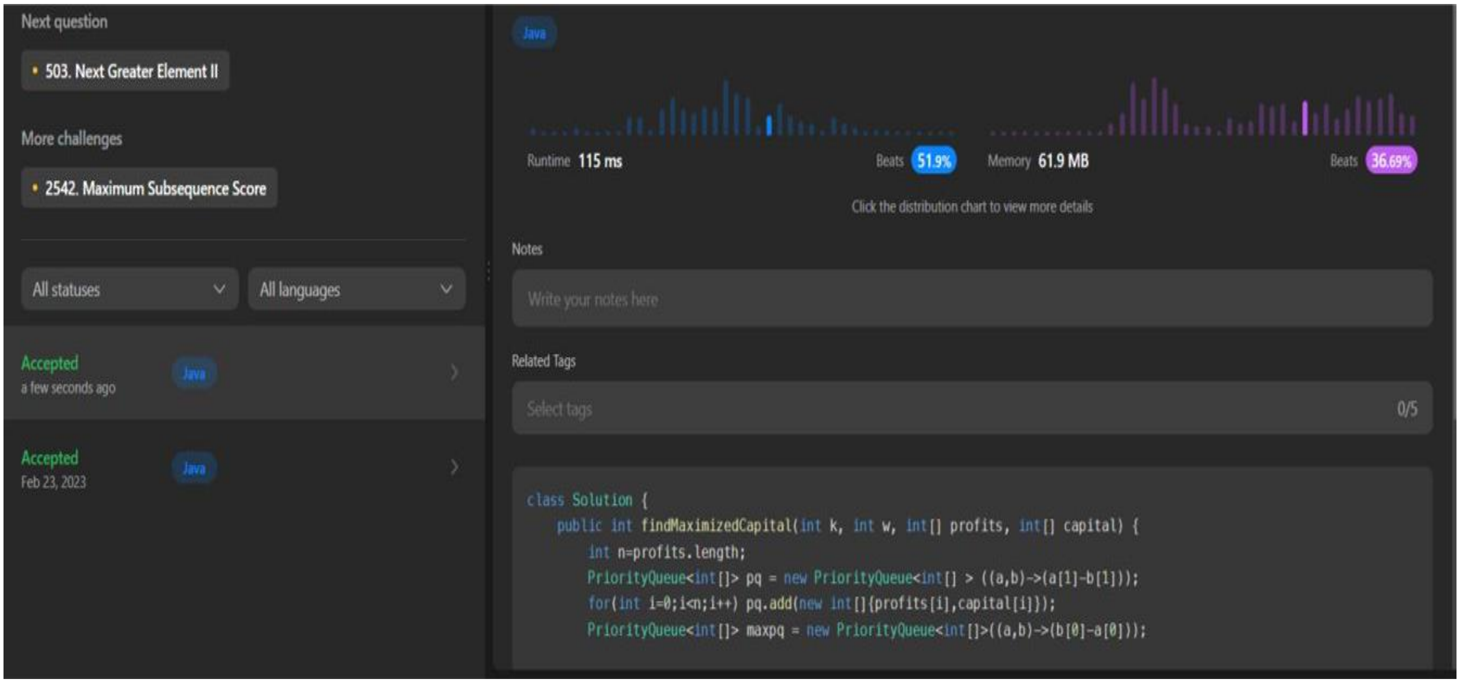
5. OBJECTIVE 2: IPO

6. CODE:

```
class Solution {
    public int findMaximizedCapital(int k, int w, int[] profits, int[] capital) {
        int n=profits.length;
        PriorityQueue<int[]> pq = new PriorityQueue<int[] > ((a,b)->(a[1]-b[1]));
        for(int i=0;i<n;i++) pq.add(new int[]{profits[i],capital[i]});
        PriorityQueue<int[]> maxpq = new PriorityQueue<int[]>((a,b)->(b[0]-a[0]));

        while((!pq.isEmpty()|| !maxpq.isEmpty())&& k>0){
            if(!pq.isEmpty()&&pq.peek()[1]<=w) maxpq.add(pq.poll());
            else{
                if(!maxpq.isEmpty()){
                    w+=maxpq.poll()[0];
                    k--;
                }else {break;}
            }
        }
        return w;
    }
}
```

7. OUTPUT



Accepted Java
a few seconds ago

Accepted Java
Feb 23, 2023

Runtime 115 ms Beats 51.9%

Memory 61.9 MB Beats 36.69%

Click the distribution chart to view more details

```
class Solution {
    public int findMaximizedCapital(int k, int w, int[] profits, int[] capital) {
        int n=profits.length;
        PriorityQueue<int[]> pq = new PriorityQueue<int[] > ((a,b)->(a[1]-b[1]));
        for(int i=0;i<n;i++) pq.add(new int[]{profits[i],capital[i]});
        PriorityQueue<int[]> maxpq = new PriorityQueue<int[]>((a,b)->(b[0]-a[0]));
```