

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

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];
    }
}
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

4. OUTPUT:

The screenshot shows a LeetCode problem solution page. On the left, there's a sidebar with 'Next question' (46. Permutations), 'More challenges' (1306. Jump Game III, 1871. Jump Game VII, 2297. Jump Game VIII), and filters for 'All statuses' and 'All languages'. The main area displays the solution for 'Jump Game' in Java. It shows a runtime of 61 ms, beating 22.14% of users, and a memory usage of 43 MB, beating 87.79% of users. Below the performance metrics, there's a 'Notes' section with a text input field. A 'Related Tags' section shows 'Select tags' with a count of 0/5. The solution code is displayed in a dark-themed editor.

```
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--;
                }
            }
        }
    }
}
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
        }else {break;}
    }
}
return w;
}
```

7. OUTPUT

Next question

• 503. Next Greater Element II

More challenges


• 2542. Maximum Subsequence Score

All statuses ▾ All languages ▾

Accepted
a few seconds ago

Accepted
Feb 23, 2023

Java



Runtime 115 ms Beats 51.9% Memory 61.9 MB Beats 36.65%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

```
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]));
```