Experiment1.1

Student Name: Himanshu
UID:20BCS7944
Branch: CSE
Section/Group:905/A
Semester:6th
Date of Performance:08/02/2023
Subject Name: Competitive Coding-II
Subject Code:20CSP-351

1. Aim:

⇨ To implement the concept of arrays, queues , stacks and Linked-list.

2. Objective:

⇨ The objective is to build problem solving capability and to learn the basic concepts of data structures.
⇨ The implementation of Jump Game-II which shows and brushes up the concept of arrays.
⇨ The implementation of removing the duplicacy in the linked-list.

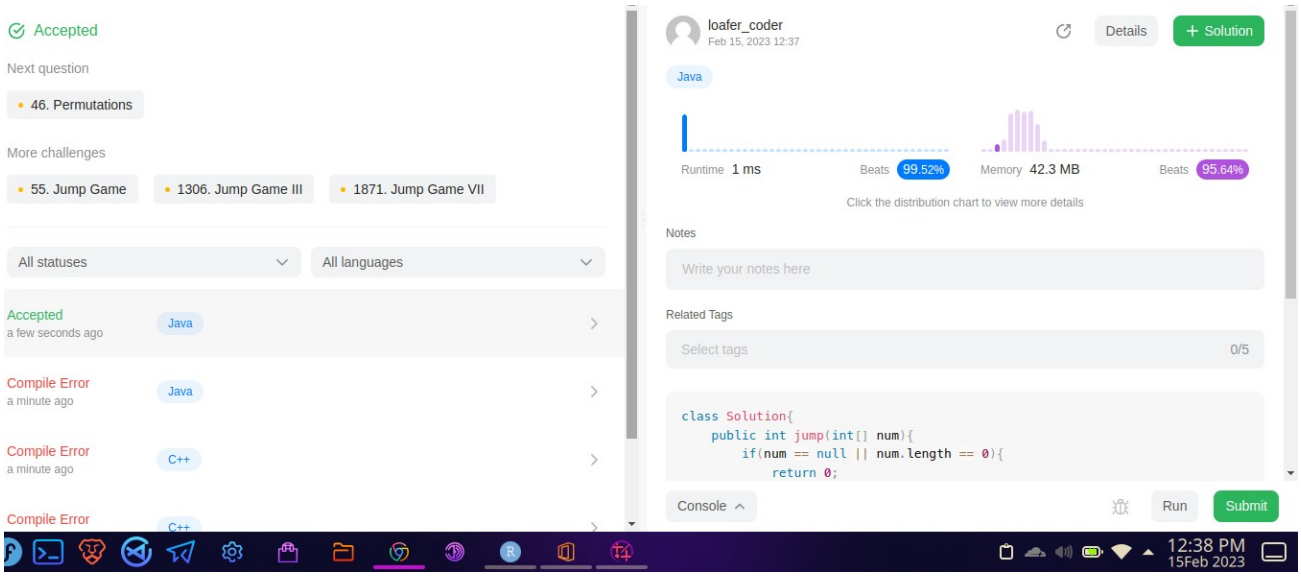
3. Leetcode code and output:

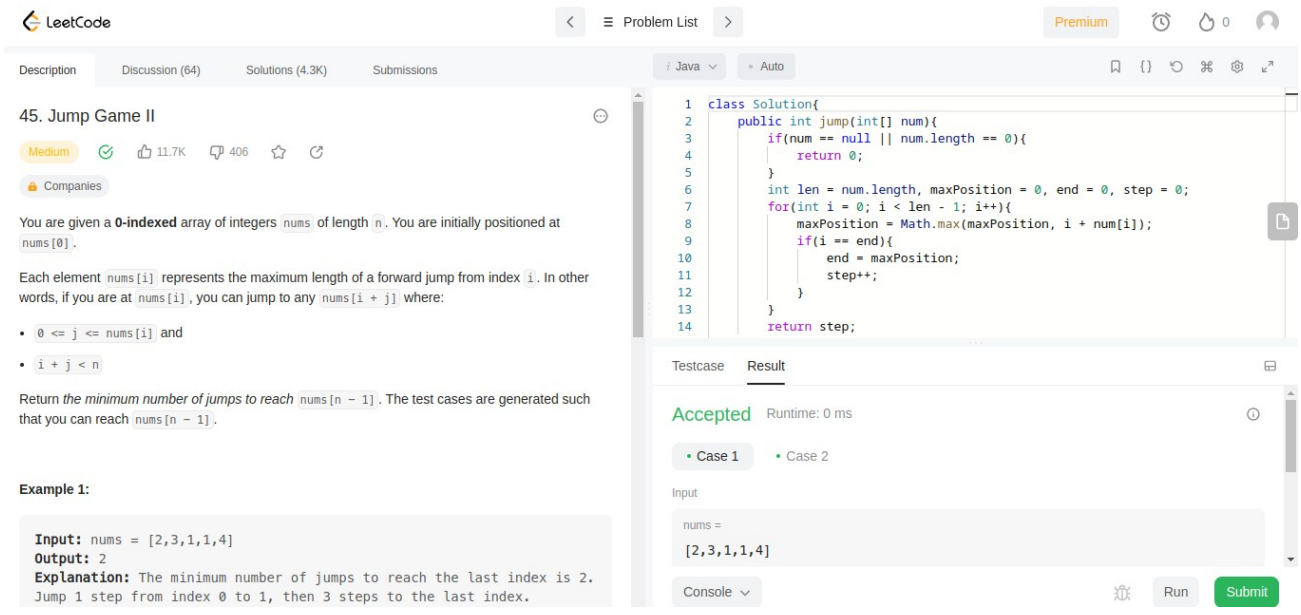
- **JUMP GAME-II**

Code-

```
class Solution{
        public int jump(int[] num){
                if(num == null || num.length == 0){
                        return 0;
                }
int len = num.length, maxPosition = 0, end = 0, step = 0;
for(int i = 0; i < len - 1; i++){
        maxPosition = Math.max(maxPosition, i + num[i]);
                if(i == end){
                        end = maxPosition;
                        step++;
                        }
                }
                return step;
        }
}
```

OUTPUT-


Accepted
Next question
46. Permutations
More challenges
55. Jump Game
1306. Jump Game III
1871. Jump Game VII
All statuses
All languages
Accepted
a few seconds ago
Java
Compile Error
a minute ago
Java
Compile Error
a minute ago
C++
Compile Error
C++
loafer_coder
Feb 15, 2023 12:37
Details
+ Solution
Java
Runtime 1 ms
Beats 99.52%
Memory 42.3 MB
Beats 95.64%
Click the distribution chart to view more details
Notes
Write your notes here
Related Tags
Select tags
0/5
class Solution{
public int jump(int[] num){
if(num == null || num.length == 0){
return 0;
Console
Run
Submit
12:38 PM
15Feb 2023


LeetCode
Problem List
Premium
Description
Discussion (64)
Solutions (4.3K)
Submissions
45. Jump Game II
Medium
11.7K
406
Companies
You are given a 0-indexed array of integers nums of length n. You are initially positioned at nums[0].
Each element nums[i] represents the maximum length of a forward jump from index i. In other words, if you are at nums[i], you can jump to any nums[i + j] where:
0 <= j <= nums[i] and
i + j < n
Return the minimum number of jumps to reach nums[n - 1]. The test cases are generated such that you can reach nums[n - 1].
Example 1:
Input: nums = [2,3,1,1,4]
Output: 2
Explanation: The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index.
Java
Auto
class Solution{
public int jump(int[] num){
if(num == null || num.length == 0){
return 0;
}
int len = num.length, maxPosition = 0, end = 0, step = 0;
for(int i = 0; i < len - 1; i++){
maxPosition = Math.max(maxPosition, i + num[i]);
if(i == end){
end = maxPosition;
step++;
}
}
return step;
Testcase
Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
nums =
[2,3,1,1,4]
Console
Run
Submit

- **REMOVING THE DUPLICATE ELEMENTS IN THE LINKED-LIST**

CODE-

```
class Solution {
        public ListNode deleteDuplicates(ListNode head) {
                HashSet<Integer> set=new HashSet<Integer>();
                while(head!=null){
                set.add(head.val);
                head=head.next; }
List<Integer> list=new ArrayList<Integer>(set);
        Collections.sort(list);
        ListNode dummy=new ListNode();
        ListNode temp=dummy;
for(int i=0;i<list.size();i++){
        ListNode Head=new ListNode(list.get(i));
                dummy.next=Head;
                dummy=dummy.next; }
        return temp.next;
        }
 }
```