Experiment No. - 10

Student Name: Vivek Kumar **UID: 21BCS8129**
Branch: BE-CSE(LEET) **Section/Group: 20BCS-ST-801/B**
Semester: 6th **Date of Performance: 09/05/2023**
Subject Name: Competitive coding - II **Subject Code: 20CSP-351**

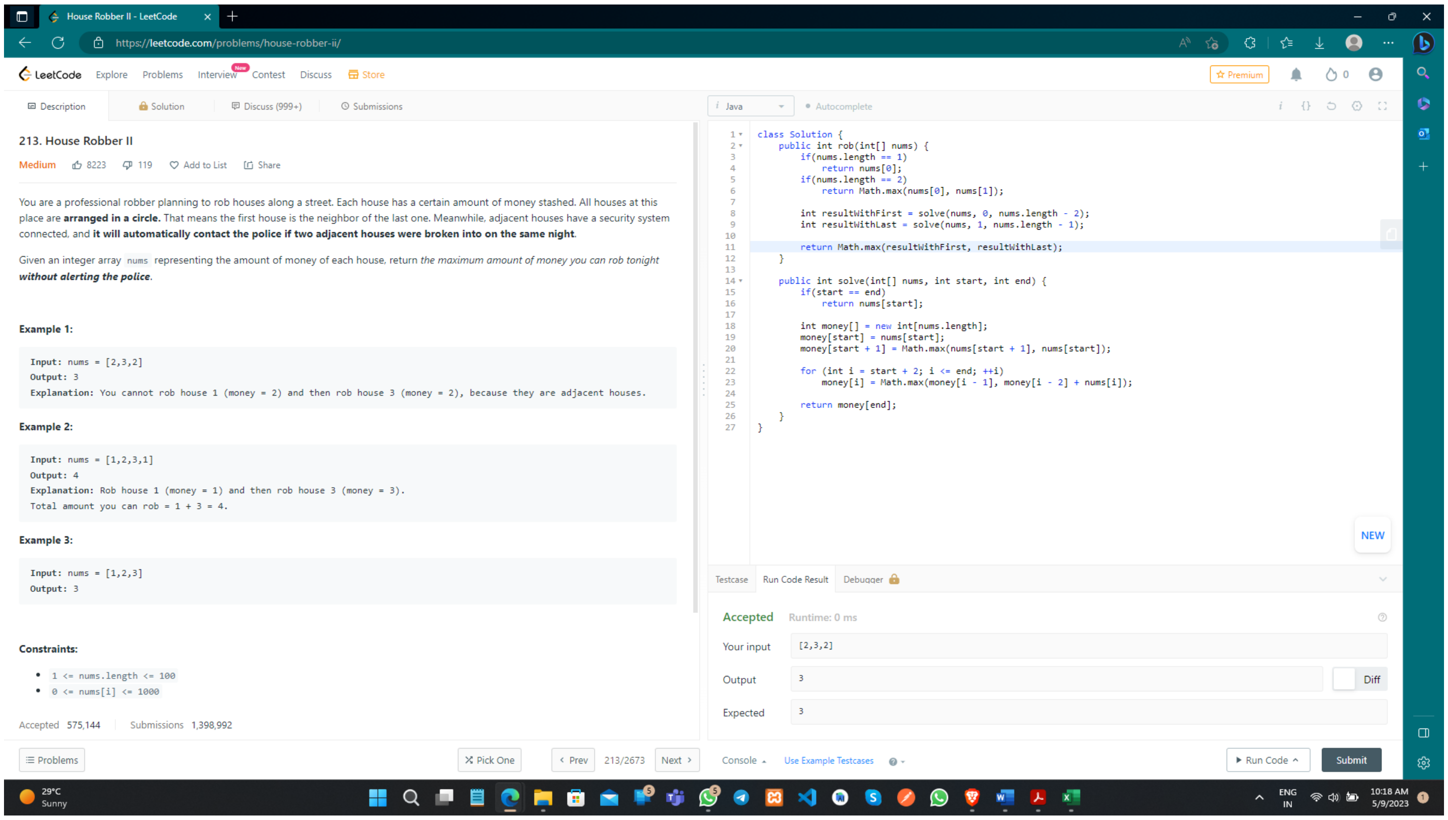
1. **Aim/Overview of the practical:**

   Q.1 House Robber - ii.

   hthttps://leetcode.com/problems/house-robber-ii/
2. **Apparatus / Simulator Used:**
   - Windows 7 or above
   - Google Chrome
3. **Objective:**
   - To understand the concept of Dynamic Programming.
   - To implement the concept of House Robber.
4. **Code:**

```
class Solution {
    public int rob(int[] nums) {
        if(nums.length == 1)
            return nums[0];
        if(nums.length == 2)
            return Math.max(nums[0], nums[1]);

        int resultWithFirst = solve(nums, 0, nums.length - 2);
        int resultWithLast = solve(nums, 1, nums.length - 1);

        return Math.max(resultWithFirst, resultWithLast);
    }

    public int solve(int[] nums, int start, int end) {
        if(start == end)
            return nums[start];

        int money[] = new int[nums.length];
        money[start] = nums[start];
        money[start + 1] = Math.max(nums[start + 1], nums[start]);

        for (int i = start + 2; i <= end; ++i)
            money[i] = Math.max(money[i - 1], money[i - 2] + nums[i]);

        return money[end];
    }
}
```

5. Result/Output/Writing Summary:



https://leetcode.com/problems/house-robber-ii/

213. House Robber II

Medium 8223 119 Add to List Share

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed. All houses at this place are **arranged in a circle.** That means the first house is the neighbor of the last one. Meanwhile, adjacent houses have a security system connected, and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given an integer array `nums` representing the amount of money of each house, return *the maximum amount of money you can rob tonight **without alerting the police***.

Example 1:

```
Input: nums = [2,3,2]
Output: 3
Explanation: You cannot rob house 1 (money = 2) and then rob house 3 (money = 2), because they are adjacent houses.
```

Example 2:

```
Input: nums = [1,2,3,1]
Output: 4
Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).
Total amount you can rob = 1 + 3 = 4.
```

Example 3:

```
Input: nums = [1,2,3]
Output: 3
```

Constraints:

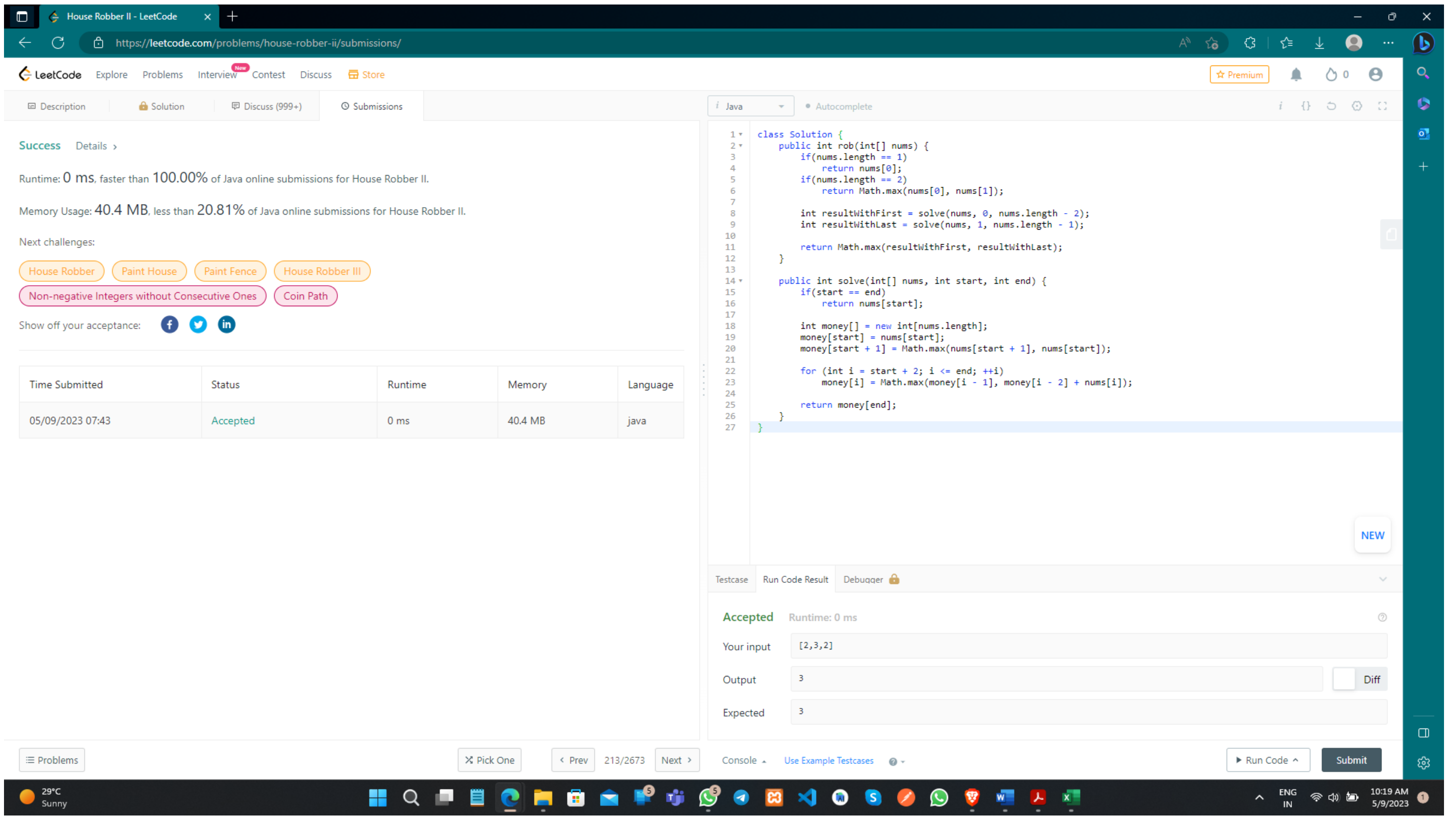
- `1 <= nums.length <= 100`
- `0 <= nums[i] <= 1000`

Accepted 575,144 Submissions 1,398,992

```java
class Solution {
    public int rob(int[] nums) {
        if(nums.length == 1)
            return nums[0];
        if(nums.length == 2)
            return Math.max(nums[0], nums[1]);

        int resultWithFirst = solve(nums, 0, nums.length - 2);
        int resultWithLast = solve(nums, 1, nums.length - 1);

        return Math.max(resultWithFirst, resultWithLast);
    }

    public int solve(int[] nums, int start, int end) {
        if(start == end)
            return nums[start];

        int money[] = new int[nums.length];
        money[start] = nums[start];
        money[start + 1] = Math.max(nums[start + 1], nums[start]);

        for (int i = start + 2; i <= end; ++i)
            money[i] = Math.max(money[i - 1], money[i - 2] + nums[i]);

        return money[end];
    }
}
```

Accepted Runtime: 0 ms

Your input	[2,3,2]
Output	3
Expected	3



https://leetcode.com/problems/house-robber-ii/submissions/

Success Details

Runtime: 0 ms, faster than 100.00% of Java online submissions for House Robber II.

Memory Usage: 40.4 MB, less than 20.81% of Java online submissions for House Robber II.

Next challenges:

House Robber · Paint House · Paint Fence · House Robber III · Non-negative Integers without Consecutive Ones · Coin Path

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
05/09/2023 07:43	Accepted	0 ms	40.4 MB	java

```java
class Solution {
    public int rob(int[] nums) {
        if(nums.length == 1)
            return nums[0];
        if(nums.length == 2)
            return Math.max(nums[0], nums[1]);

        int resultWithFirst = solve(nums, 0, nums.length - 2);
        int resultWithLast = solve(nums, 1, nums.length - 1);

        return Math.max(resultWithFirst, resultWithLast);
    }

    public int solve(int[] nums, int start, int end) {
        if(start == end)
            return nums[start];

        int money[] = new int[nums.length];
        money[start] = nums[start];
        money[start + 1] = Math.max(nums[start + 1], nums[start]);

        for (int i = start + 2; i <= end; ++i)
            money[i] = Math.max(money[i - 1], money[i - 2] + nums[i]);

        return money[end];
    }
}
```

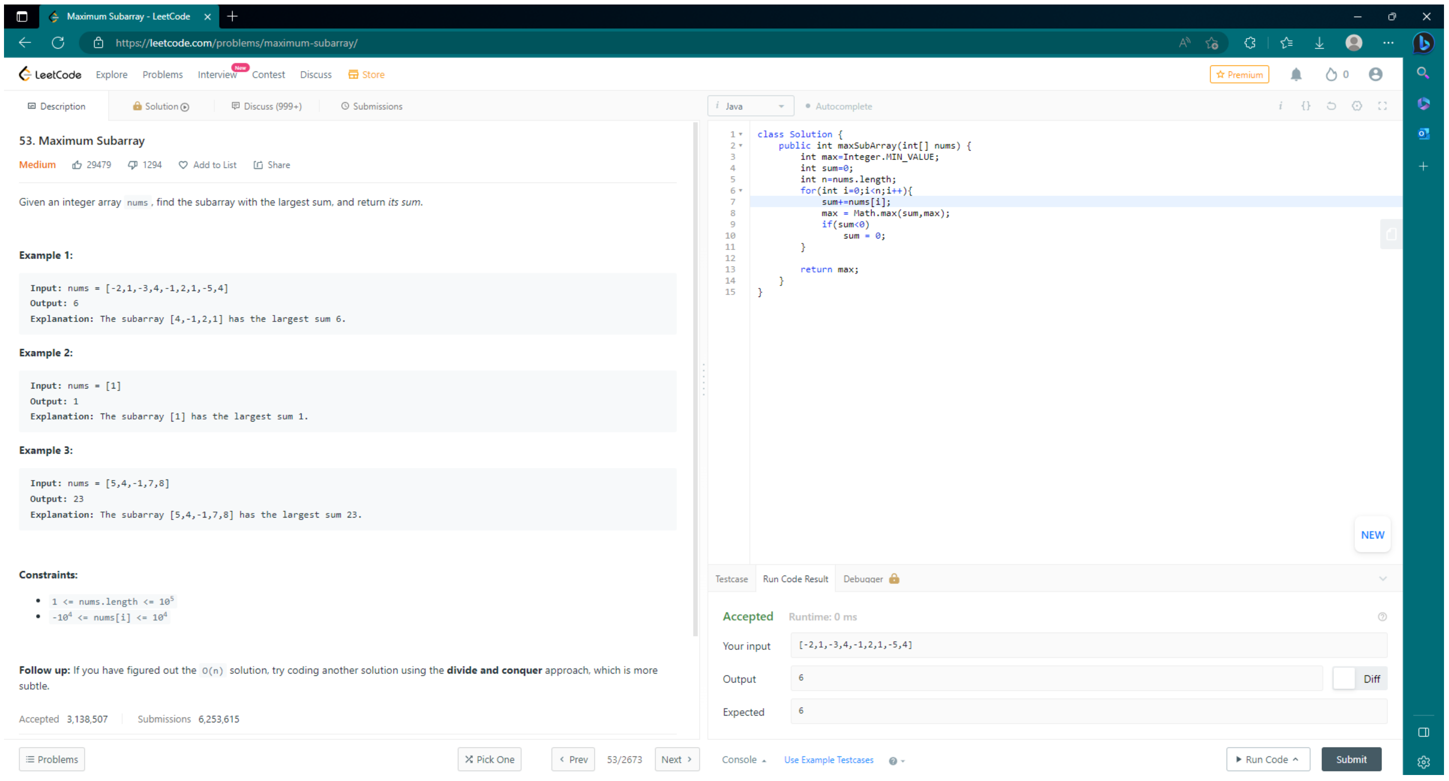
Accepted Runtime: 0 ms

Your input	[2,3,2]
Output	3
Expected	3

1. **Aim/Overview of the practical:**
   Q.2 Maximum - Subarray.
   https://leetcode.com/problems/maximum-subarray/
2. **Apparatus / Simulator Used:**
   - Windows 7 or above
   - Google Chrome
3. **Objective:**
   - To understand the concept of Dynamic Programming.
   - To implement the concept of Maximum Subarray.
4. **Code:**

```
class Solution {
    public int maxSubArray(int[] nums) {
        int max=Integer.MIN_VALUE;
        int sum=0;
        int n=nums.length;
        for(int i=0;i<n;i++){
            sum+=nums[i];
            max = Math.max(sum,max);
            if(sum<0)
                sum = 0;
        }

        return max;
    }
}
```

5. **Result/Output/Writing Summary:**



Maximum Subarray - LeetCode
https://leetcode.com/problems/maximum-subarray/

53. Maximum Subarray

Medium 29479 1294 Add to List Share

Given an integer array `nums`, find the subarray with the largest sum, and return *its sum*.

Example 1:

```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the largest sum 6.
```

Example 2:

```
Input: nums = [1]
Output: 1
Explanation: The subarray [1] has the largest sum 1.
```

Example 3:

```
Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.
```

Constraints:

- `1 <= nums.length <=` 10^5
- -10^4 `<= nums[i] <=` 10^4

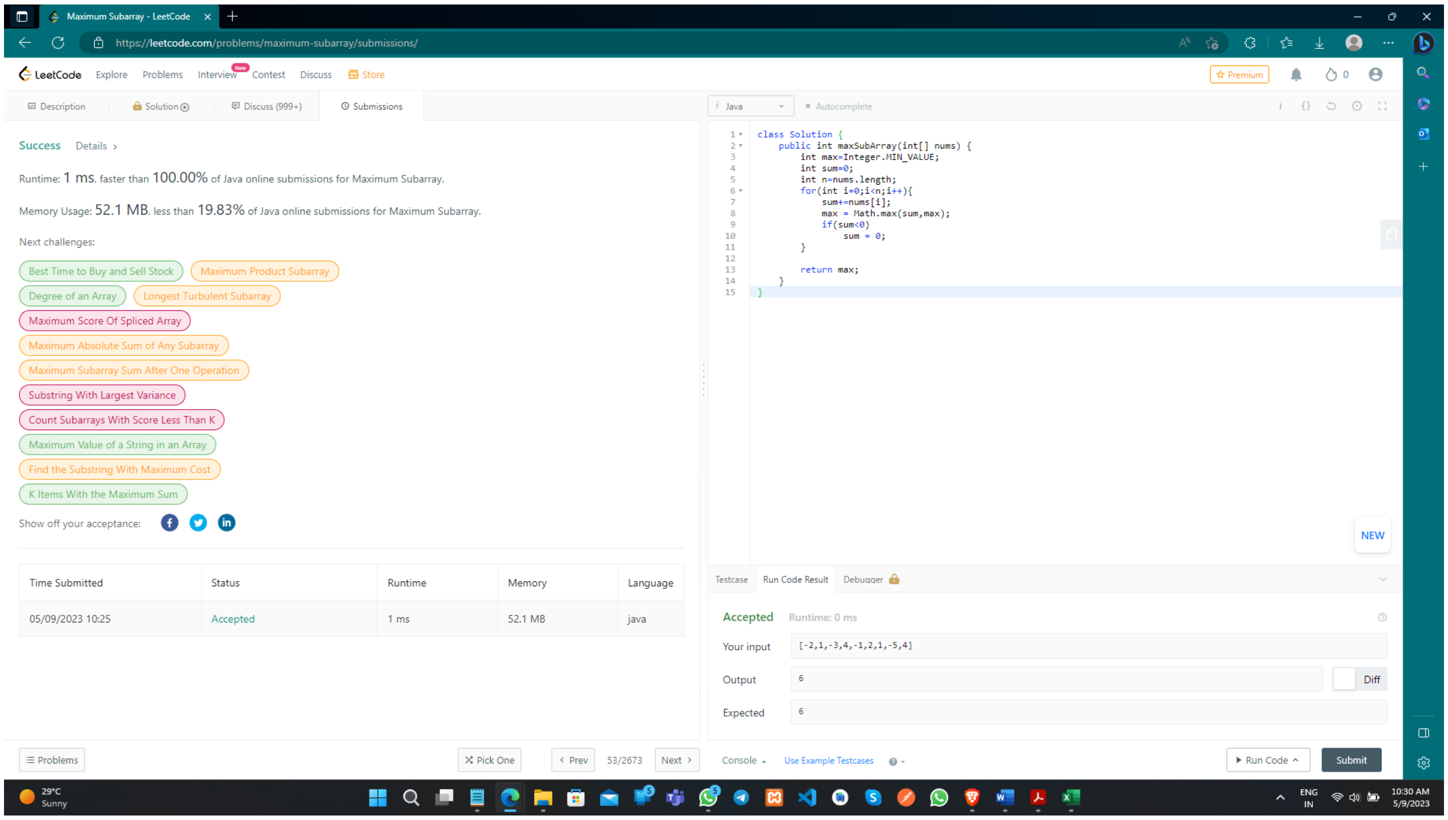
Follow up: If you have figured out the `O(n)` solution, try coding another solution using the **divide and conquer** approach, which is more subtle.

Accepted 3,138,507 Submissions 6,253,615

```
class Solution {
    public int maxSubArray(int[] nums) {
        int max=Integer.MIN_VALUE;
        int sum=0;
        int n=nums.length;
        for(int i=0;i<n;i++){
            sum+=nums[i];
            max = Math.max(sum,max);
            if(sum<0)
                sum = 0;
        }

        return max;
    }
}
```

Accepted Runtime: 0 ms

Your input
```
[-2,1,-3,4,-1,2,1,-5,4]
```
Output
```
6
```
Expected
```
6
```



Success Details

Runtime: 1 ms, faster than 100.00% of Java online submissions for Maximum Subarray.

Memory Usage: 52.1 MB, less than 19.83% of Java online submissions for Maximum Subarray.

Time Submitted	Status	Runtime	Memory	Language
05/09/2023 10:25	Accepted	1 ms	52.1 MB	java

```java
class Solution {
    public int maxSubArray(int[] nums) {
        int max=Integer.MIN_VALUE;
        int sum=0;
        int n=nums.length;
        for(int i=0;i<n;i++){
            sum+=nums[i];
            max = Math.max(sum,max);
            if(sum<0)
                sum = 0;
        }

        return max;
    }
}
```

```
Accepted   Runtime: 0 ms
Your input [-2,1,-3,4,-1,2,1,-5,4]
Output     6
Expected   6
```

Maximum Subarray - LeetCode
https://leetcode.com/problems/maximum-subarray/submissions/
Description
Solution
Discuss (999+)
Submissions
Next challenges:
Best Time to Buy and Sell Stock
Maximum Product Subarray
Degree of an Array
Longest Turbulent Subarray
Maximum Score Of Spliced Array
Maximum Absolute Sum of Any Subarray
Maximum Subarray Sum After One Operation
Substring With Largest Variance
Count Subarrays With Score Less Than K
Maximum Value of a String in an Array
Find the Substring With Maximum Cost
K Items With the Maximum Sum
Show off your acceptance:
Java
Autocomplete
Run Code
Submit
53/2673

Learning outcomes (What I have learnt):

- Learned the concept of Dynamic Programming in Fibonacci Sequence and so on.
- Learnt about House Robber-ii to Target & Maximum Subarray.