Experiment-3.3

Student Name: Praduman Kumar **UID: 20BCS9446**
Branch: BE-CSE **Section/Group: 714/A**
Semester: 6th **Subject Code: 20CSP-351**
Subject Name: Competitive coding - II

1. **Aim/Overview of the practical:**
   To implement the concept of dynamic programming.

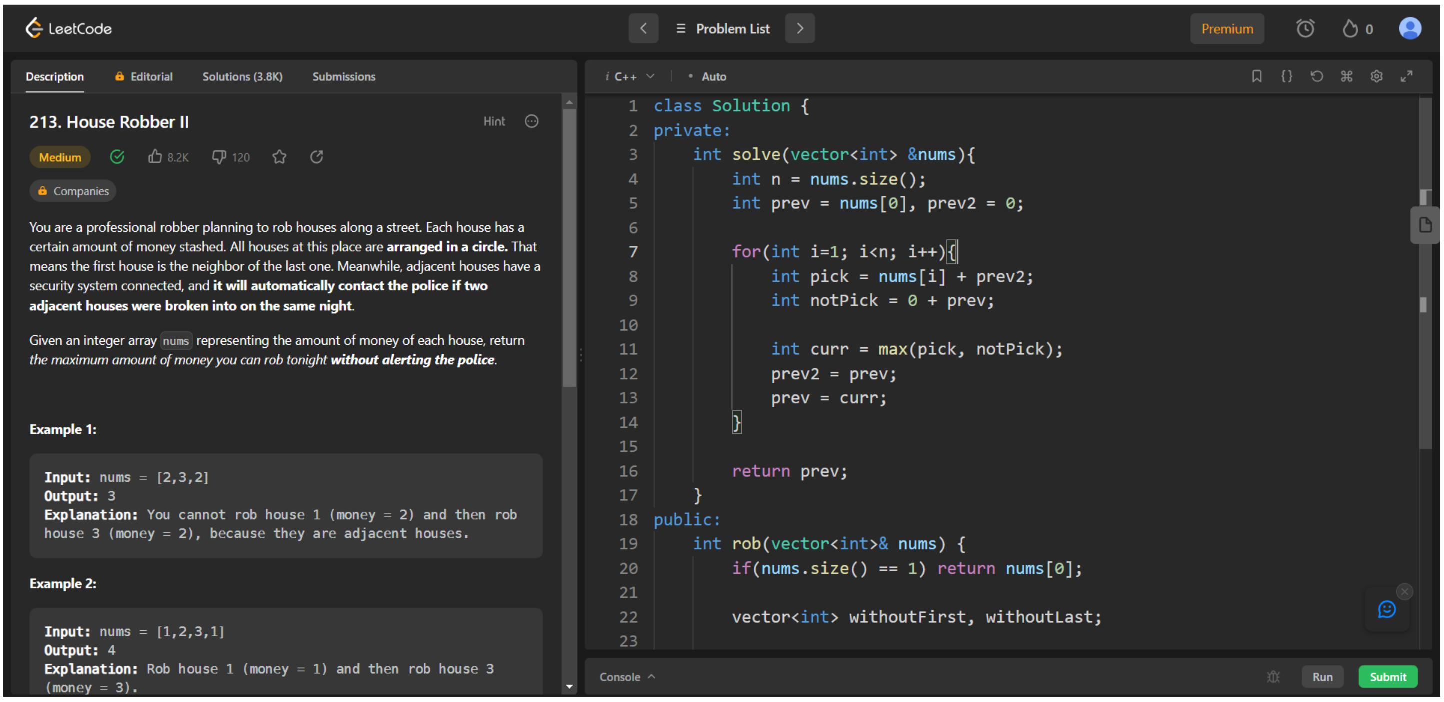
   House Robber - ii.
   https://leetcode.com/problems/house-robber-ii/

   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];
    }
}
```

Result/Output/Writing Summary:



213. House Robber II

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***.

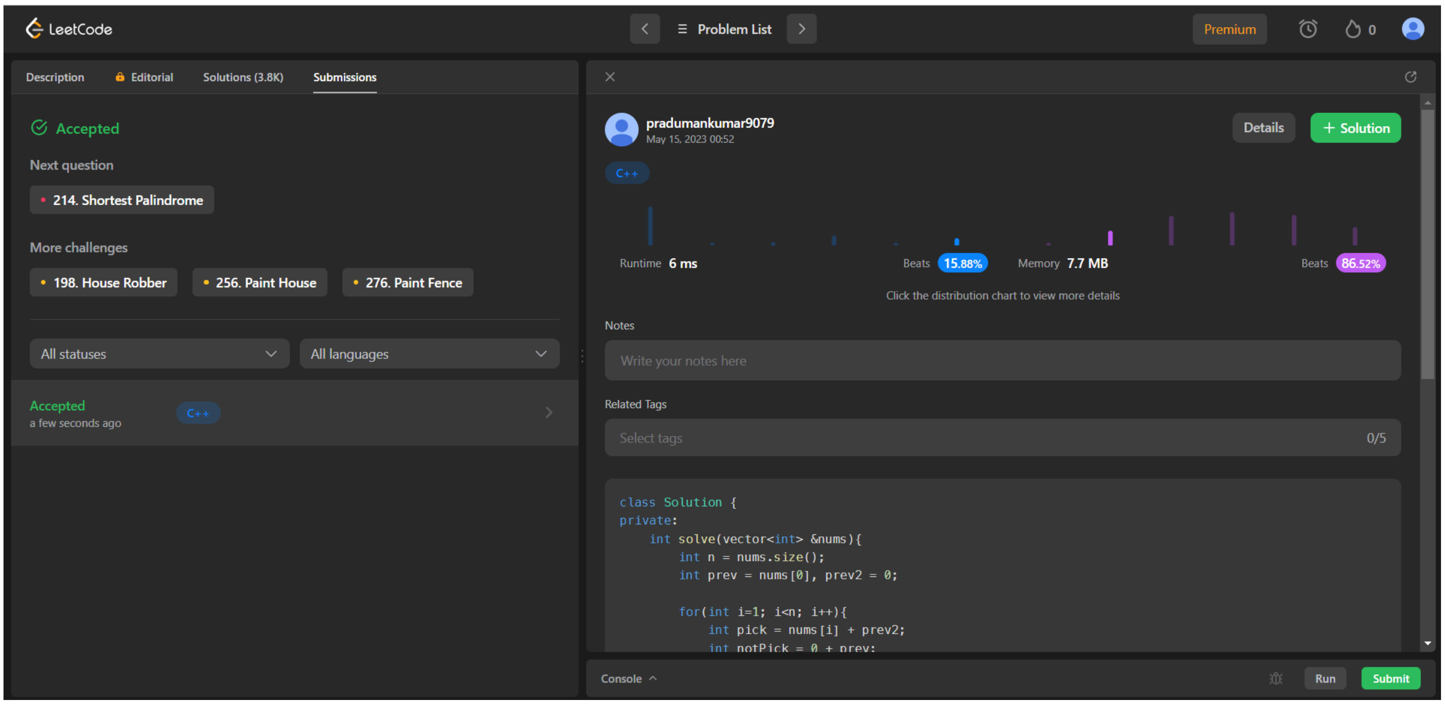
Example 1:

```
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).
```

```cpp
class Solution {
private:
    int solve(vector<int> &nums){
        int n = nums.size();
        int prev = nums[0], prev2 = 0;

        for(int i=1; i<n; i++){
            int pick = nums[i] + prev2;
            int notPick = 0 + prev;

            int curr = max(pick, notPick);
            prev2 = prev;
            prev = curr;
        }

        return prev;
    }
public:
    int rob(vector<int>& nums) {
        if(nums.size() == 1) return nums[0];

        vector<int> withoutFirst, withoutLast;

```



Accepted

Next question

• 214. Shortest Palindrome

More challenges

• 198. House Robber • 256. Paint House • 276. Paint Fence

All statuses | All languages

Accepted
a few seconds ago
C++

pradumankumar9079
May 15, 2023 00:52

Details | + Solution

C++

Runtime 6 ms Beats 15.88%

Memory 7.7 MB Beats 86.52%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

```
class Solution {
private:
    int solve(vector<int> &nums){
        int n = nums.size();
        int prev = nums[0], prev2 = 0;

        for(int i=1; i<n; i++){
            int pick = nums[i] + prev2;
            int notPick = 0 + prev;
```

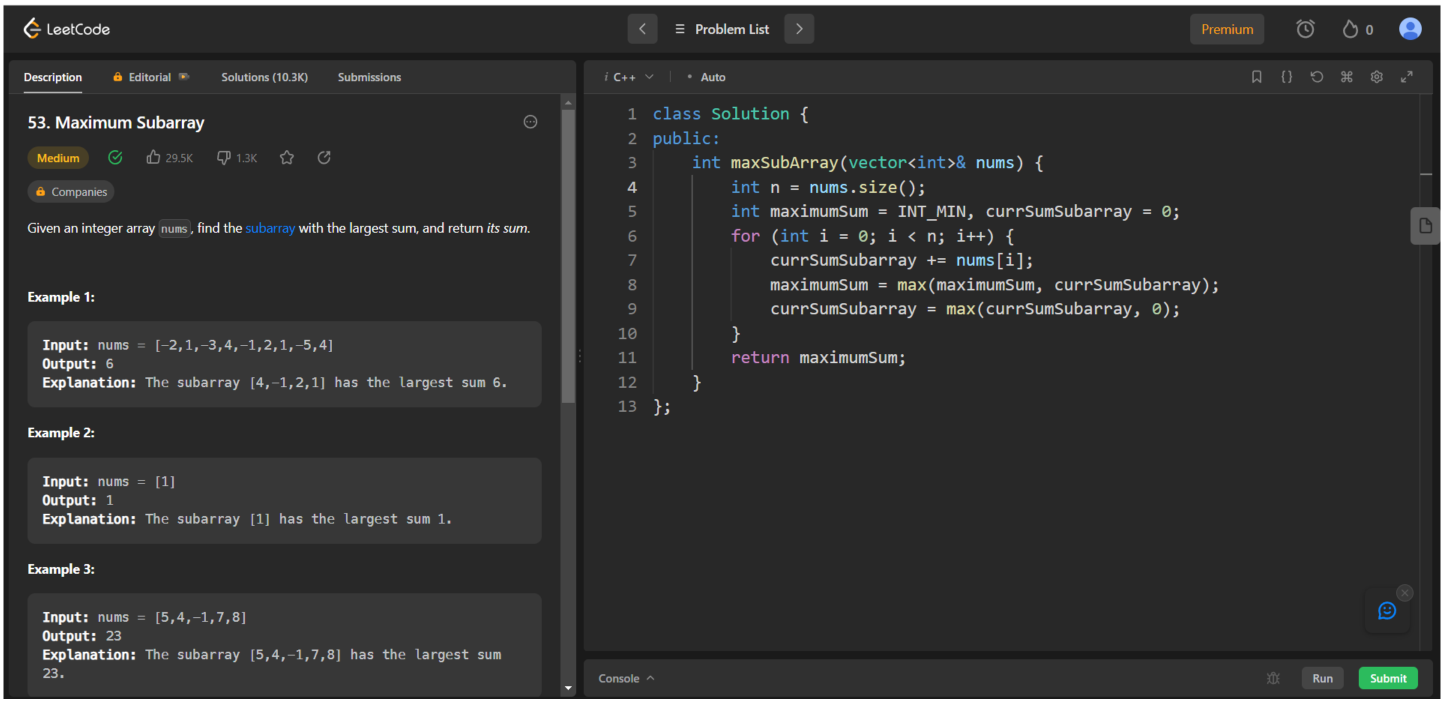
2. Maximum - Subarray.

https://leetcode.com/problems/maximum-subarray/

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;
    }
}
```

Result/Output/Writing Summary:



53. Maximum Subarray

Medium

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.
```

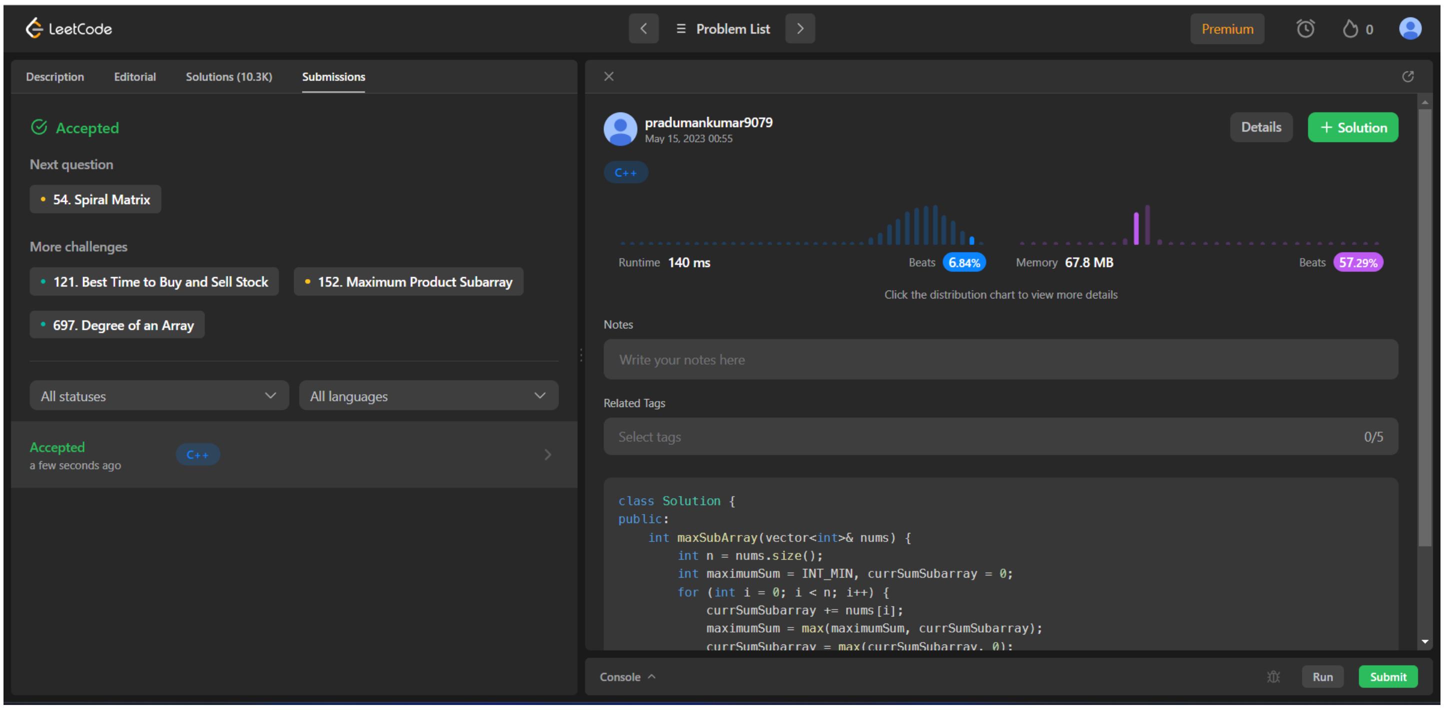
Example 2:

```
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.
```

```cpp
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int n = nums.size();
        int maximumSum = INT_MIN, currSumSubarray = 0;
        for (int i = 0; i < n; i++) {
            currSumSubarray += nums[i];
            maximumSum = max(maximumSum, currSumSubarray);
            currSumSubarray = max(currSumSubarray, 0);
        }
        return maximumSum;
    }
};
```



Accepted

Next question: 54. Spiral Matrix

More challenges: 121. Best Time to Buy and Sell Stock, 152. Maximum Product Subarray, 697. Degree of an Array

Accepted — C++ — a few seconds ago

pradumankumar9079
May 15, 2023 00:55

Runtime 140 ms Beats 6.84%
Memory 67.8 MB Beats 57.29%

```cpp
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int n = nums.size();
        int maximumSum = INT_MIN, currSumSubarray = 0;
        for (int i = 0; i < n; i++) {
            currSumSubarray += nums[i];
            maximumSum = max(maximumSum, currSumSubarray);
            currSumSubarray = max(currSumSubarray, 0);
```

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.