Experiment 8

Student Name: Aman Gokul
Branch: BE CSE
Semester: 6th
Subject Name: CC-2 Lab

UID: 20BCS5449
Section/Group: 607/A
Date of Performance: 05/05/2023
Subject Code: 20CSP-351

1. Aim/Overview of the practical:

Best Time to Buy and Sell Stock

You are given an array prices where prices[i] is the price of a given stock on the i^{th} day.
You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return 0.

https://leetcode.com/problems/best-time-to-buy-and-sell-stock/

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

3. Objective:

- To understand the concept of Greedy.

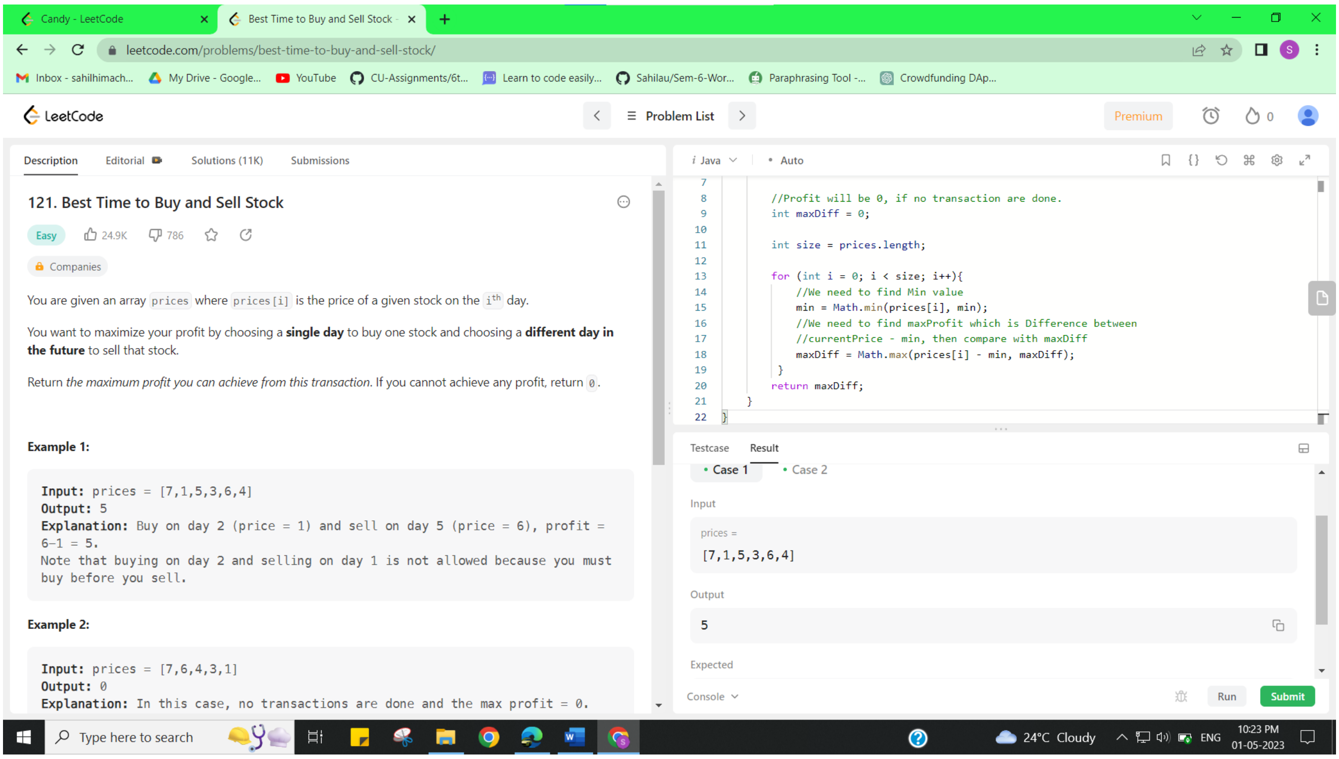
A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result. The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.

4. Code:

```
class Solution {
  public int maxProfit(int[] prices) {

      //In constraints it is given that
      //0 <= prices[i] <= 104
      int min = 10000;

      //Profit will be 0, if no transaction are done.
      int maxDiff = 0;

      int size = prices.length;

      for (int i = 0; i < size; i++){
        //We need to find Min value
        min = Math.min(prices[i], min);
        //We need to find maxProfit which is Difference between
        //currentPrice - min, then compare with maxDiff
```

```
            maxDiff = Math.max(prices[i] - min, maxDiff);
         }
        return maxDiff;
    }}
```

4. Result/Output/Writing Summary:



121. Best Time to Buy and Sell Stock

You are given an array `prices` where `prices[i]` is the price of a given stock on the `i`th day.

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return `0`.

Example 1:

```
Input: prices = [7,1,5,3,6,4]
Output: 5
Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.
```

Example 2:

```
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transactions are done and the max profit = 0.
```

```

        //Profit will be 0, if no transaction are done.
        int maxDiff = 0;

        int size = prices.length;

        for (int i = 0; i < size; i++){
            //We need to find Min value
            min = Math.min(prices[i], min);
            //We need to find maxProfit which is Difference between
            //currentPrice - min, then compare with maxDiff
            maxDiff = Math.max(prices[i] - min, maxDiff);
         }
        return maxDiff;
    }
}
```

Input

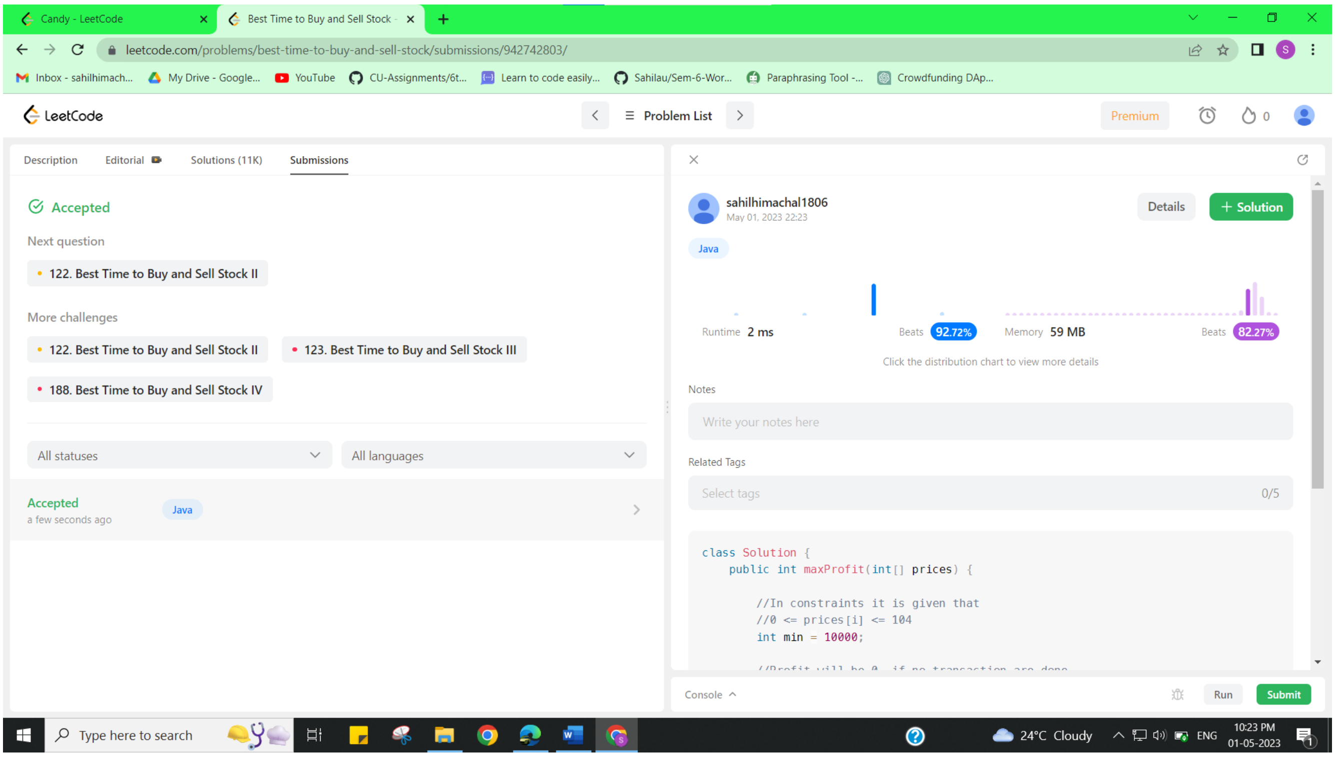
prices =

```
[7,1,5,3,6,4]
```

Output

```
5
```

Expected



Accepted

Next question

122. Best Time to Buy and Sell Stock II

More challenges

122. Best Time to Buy and Sell Stock II

123. Best Time to Buy and Sell Stock III

188. Best Time to Buy and Sell Stock IV

All statuses

All languages

Accepted Java
a few seconds ago

sahilhimachal1806
May 01, 2023 22:23

Runtime 2 ms Beats 92.72%

Memory 59 MB Beats 82.27%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

```
class Solution {
    public int maxProfit(int[] prices) {

        //In constraints it is given that
        //0 <= prices[i] <= 104
        int min = 10000;

        //Profit will be 0, if no transaction are done
```

Experiment 8.2

1. Aim/Overview of the practical:

Candy

There are n children standing in a line. Each child is assigned a rating value given in the integer array ratings.

You are giving candies to these children subjected to the following requirements:

- Each child must have at least one candy.
- Children with a higher rating get more candies than their neighbors.

Return *the minimum number of candies you need to have to distribute the candies to the children*.

https://leetcode.com/problems/candy/

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

3. Objective:

- To understand the concept of Greedy.

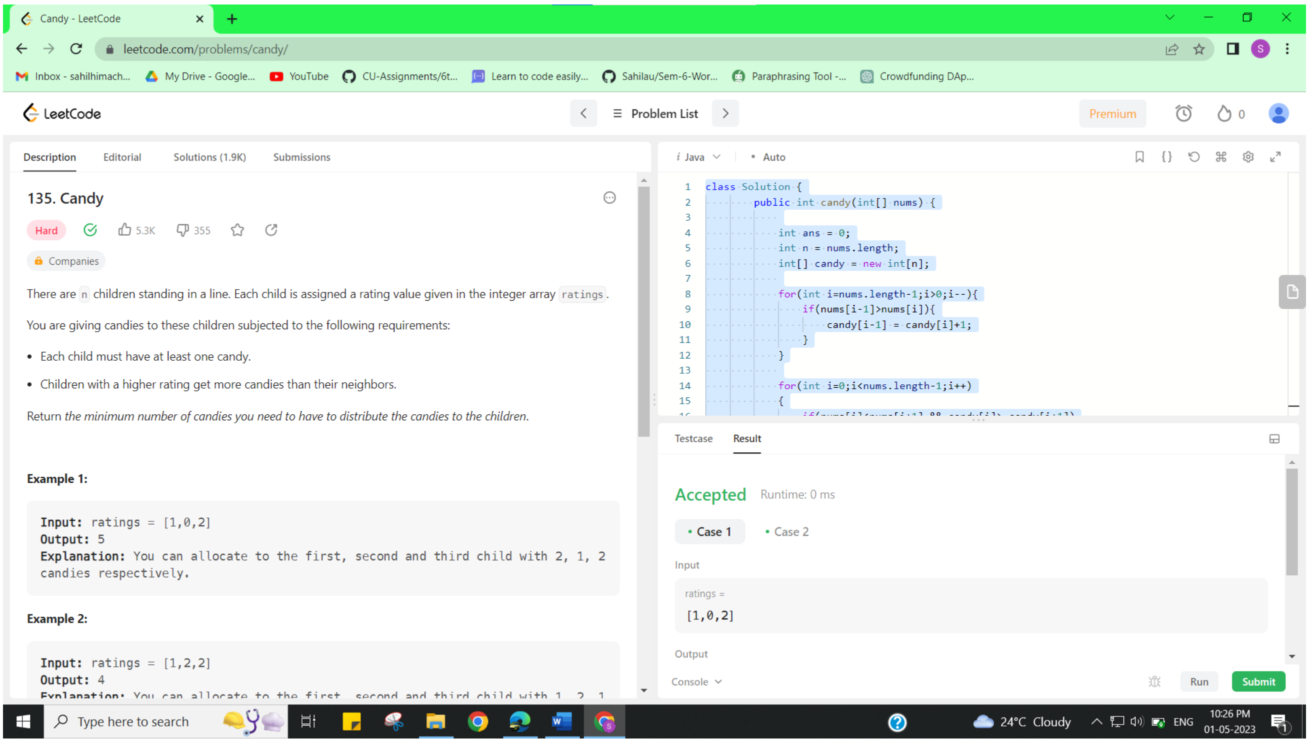
A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result. The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.

4. Code:

```
class Solution {
    public int candy(int[] nums) {

      int ans = 0;
      int n = nums.length;
      int[] candy = new int[n];

      for(int i=nums.length-1;i>0;i--){
        if(nums[i-1]>nums[i]){
          candy[i-1] = candy[i]+1;
        }
      }

      for(int i=0;i<nums.length-1;i++)
      {
        if(nums[i]<nums[i+1] && candy[i]>=candy[i+1])
        {
          candy[i+1] = candy[i]+1;
        }
        ans+=candy[i];
      }

      return n+ans+candy[n-1];
    }
```

}

5. Result/Output/Writing Summary:



135. Candy

Hard

There are n children standing in a line. Each child is assigned a rating value given in the integer array ratings.

You are giving candies to these children subjected to the following requirements:

- Each child must have at least one candy.
- Children with a higher rating get more candies than their neighbors.

Return *the minimum number of candies you need to have to distribute the candies to the children*.

Example 1:

```
Input: ratings = [1,0,2]
Output: 5
Explanation: You can allocate to the first, second and third child with 2, 1, 2 candies respectively.
```

Example 2:

```
Input: ratings = [1,2,2]
Output: 4
```

```
class Solution {
    public int candy(int[] nums) {

        int ans = 0;
        int n = nums.length;
        int[] candy = new int[n];

        for(int i=nums.length-1;i>0;i--){
            if(nums[i-1]>nums[i]){
                candy[i-1] = candy[i]+1;
            }
        }

        for(int i=0;i<nums.length-1;i++)
        {
```

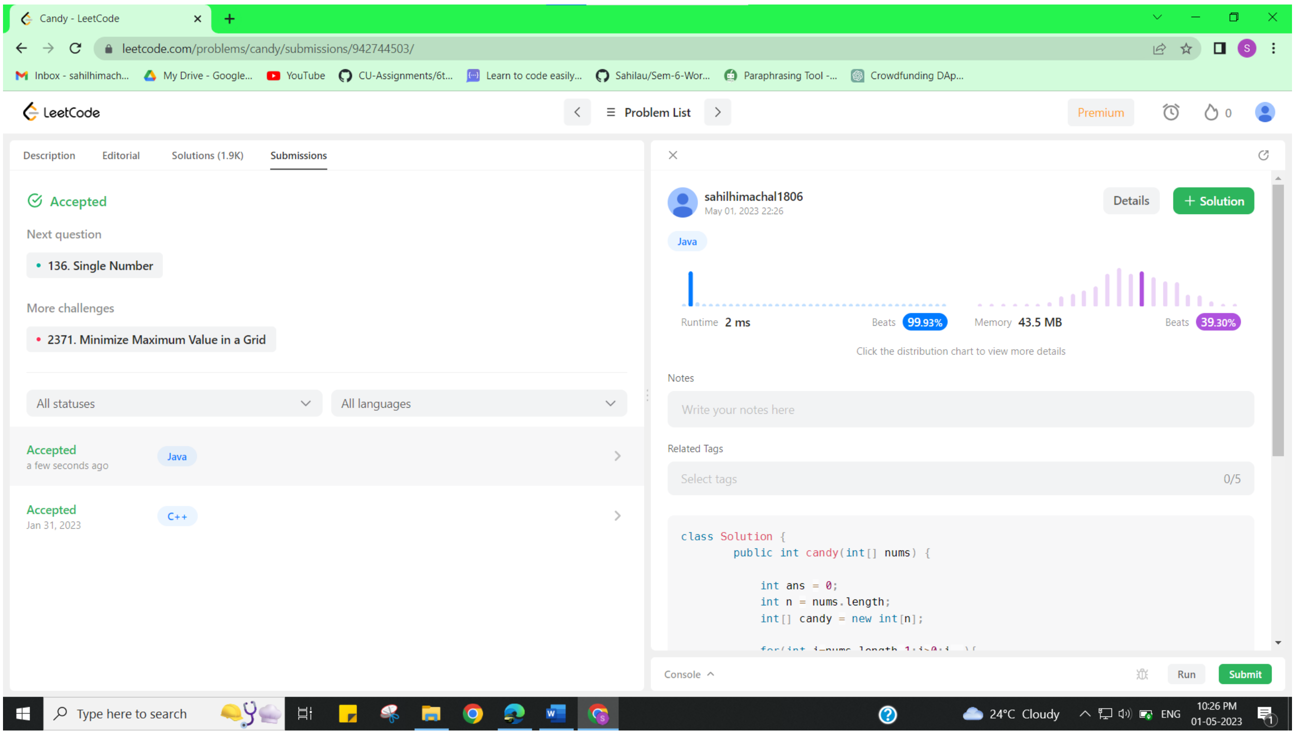
Accepted Runtime: 0 ms

Case 1 Case 2

Input

ratings =
[1,0,2]

Output



Accepted

Next question

136. Single Number

More challenges

2371. Minimize Maximum Value in a Grid

Accepted a few seconds ago	Java
Accepted Jan 31, 2023	C++

sahilhimachal1806
May 01, 2023 22:26

Java

Runtime 2 ms Beats 99.93%

Memory 43.5 MB Beats 39.30%

Click the distribution chart to view more details

```
class Solution {
        public int candy(int[] nums) {

            int ans = 0;
            int n = nums.length;
            int[] candy = new int[n];
```

Learning outcomes (What I have learnt):

- Learned the concept of Greedy Approach.