Experiment 3.3

Student Name: Yash Gupta **UID: 20BCS5009**
Branch: BE-CSE **Section/Group:20BCS_DM-716 B**
Semester: 6 **Date of Performance: 08/05/23**
Subject Name: CC LAB **Subject Code: 20CSP_351**

1. Aim:

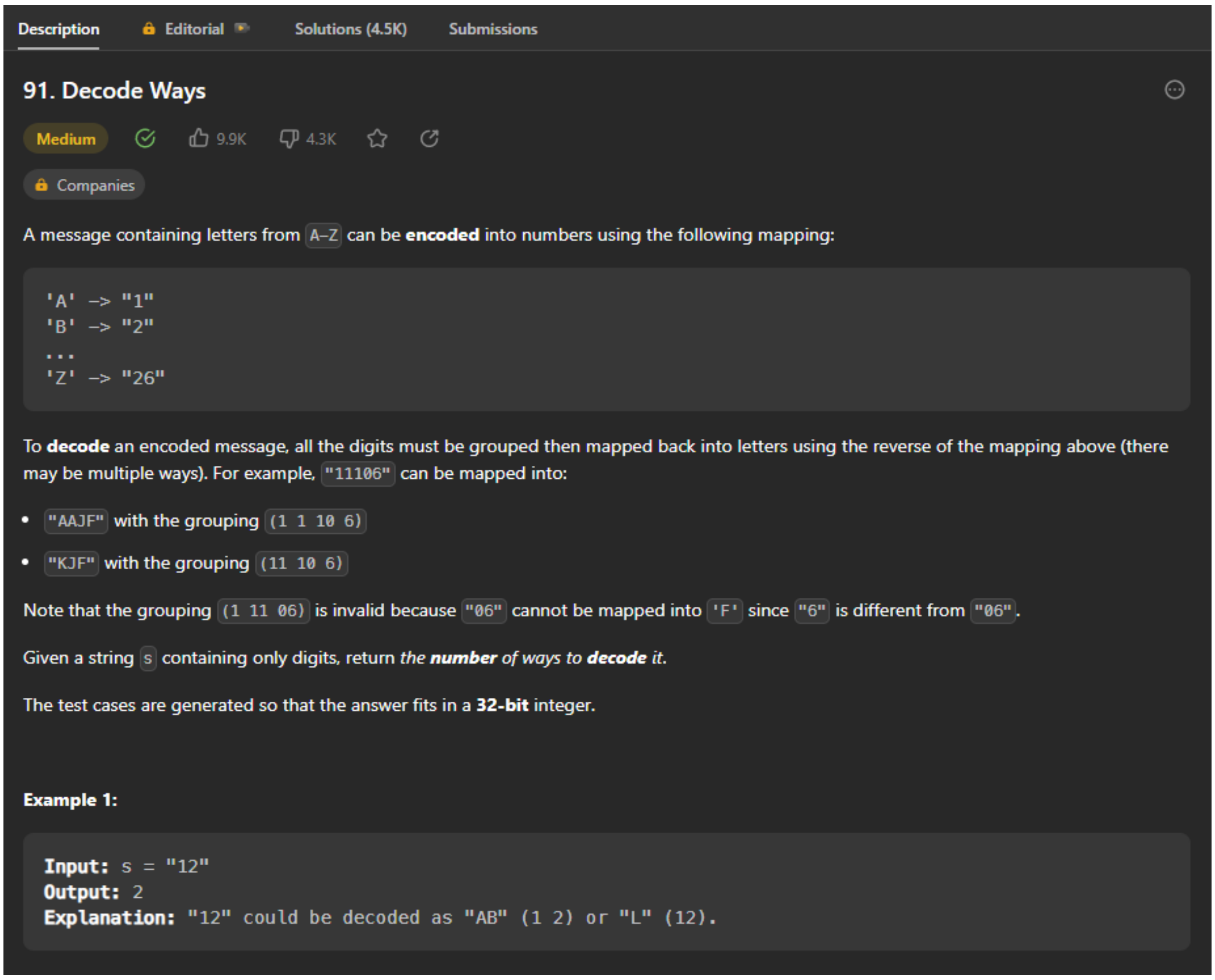
To implement the concept of dynamic programming.

2. Objective:

- The objective is to build problem solving capability and to learn the basic concepts of data structures.
- Understand the problem and find out better approach to solve particular problem

3. LeetCode code and output:

- **Decode Ways**



91. Decode Ways

A message containing letters from `A-Z` can be **encoded** into numbers using the following mapping:

```
'A' -> "1"
'B' -> "2"
...
'Z' -> "26"
```

To **decode** an encoded message, all the digits must be grouped then mapped back into letters using the reverse of the mapping above (there may be multiple ways). For example, `"11106"` can be mapped into:

- `"AAJF"` with the grouping `(1 1 10 6)`
- `"KJF"` with the grouping `(11 10 6)`

Note that the grouping `(1 11 06)` is invalid because `"06"` cannot be mapped into `'F'` since `"6"` is different from `"06"`.

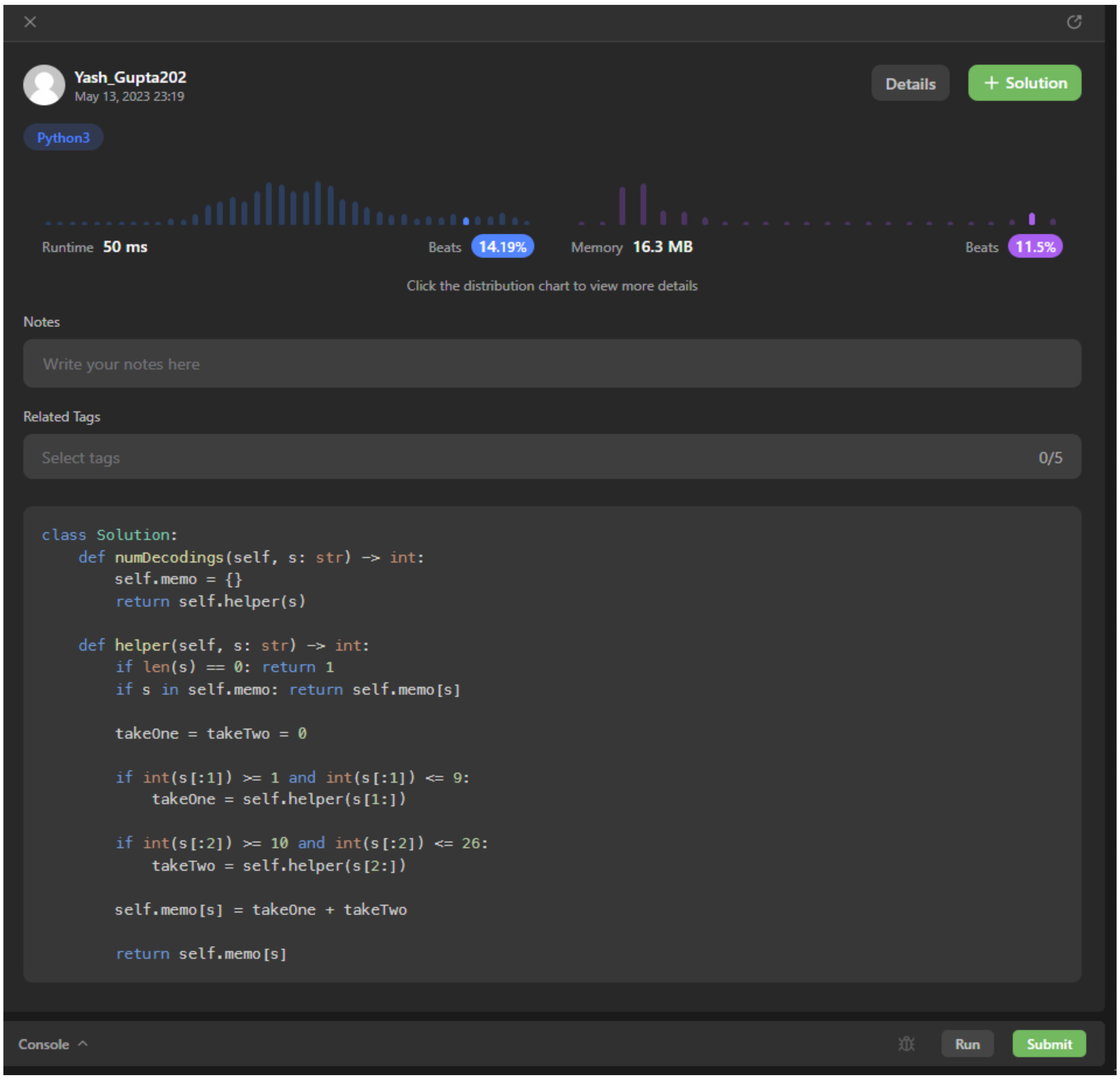
Given a string `s` containing only digits, return *the* ***number*** *of ways to* ***decode*** *it*.

The test cases are generated so that the answer fits in a **32-bit** integer.

Example 1:

```
Input: s = "12"
Output: 2
Explanation: "12" could be decoded as "AB" (1 2) or "L" (12).
```

```
class Solution:
    def numDecodings(self, s: str) -> int:
        self.memo = {}
        return self.helper(s)

    def helper(self, s: str) -> int:
        if len(s) == 0: return 1
        if s in self.memo: return self.memo[s]

        takeOne = takeTwo = 0

        if int(s[:1]) >= 1 and int(s[:1]) <= 9:
            takeOne = self.helper(s[1:])

        if int(s[:2]) >= 10 and int(s[:2]) <= 26:
            takeTwo = self.helper(s[2:])

        self.memo[s] = takeOne + takeTwo

        return self.memo[s]
```



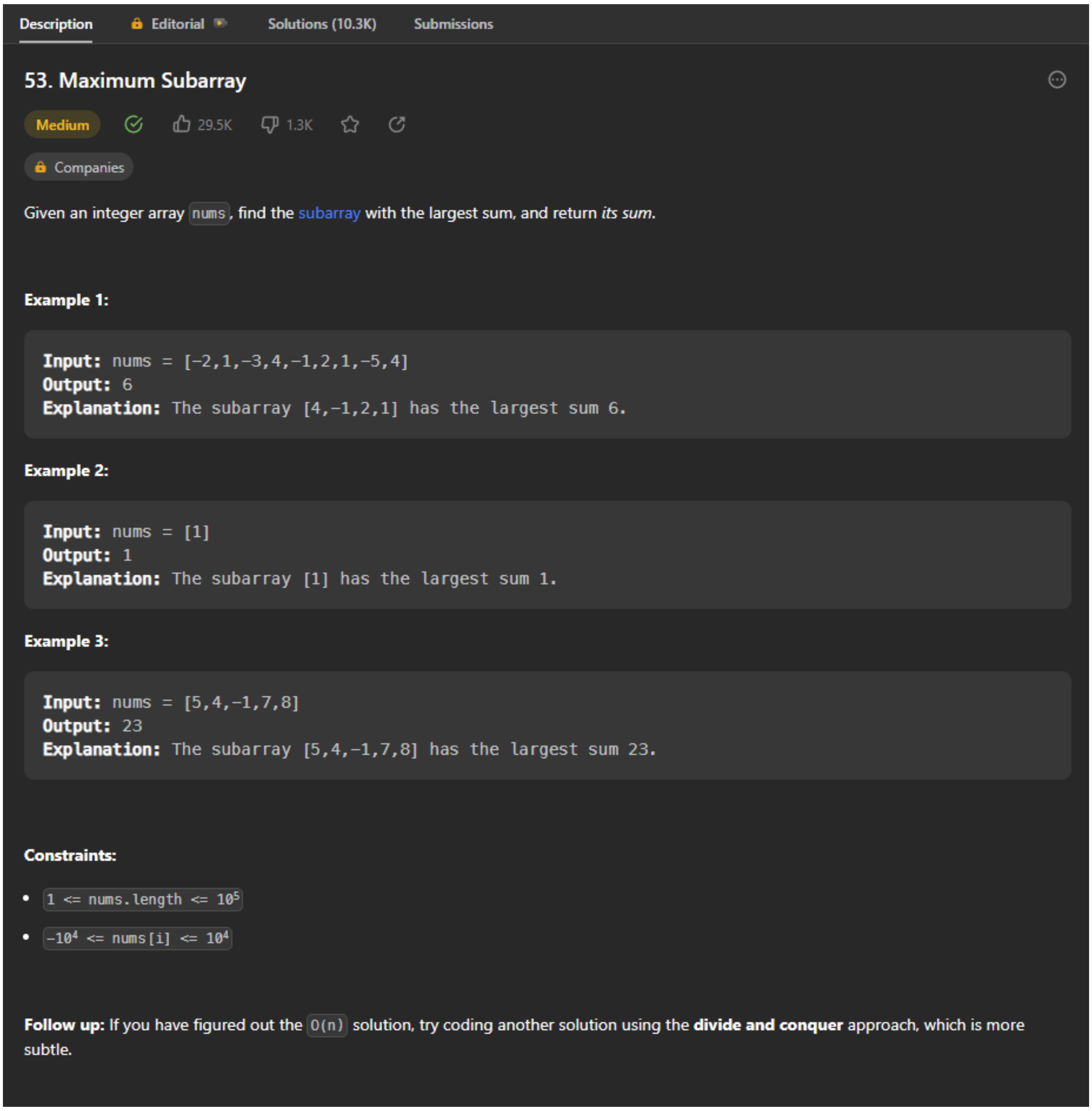
Yash_Gupta202
May 13, 2023 23:19

Runtime 50 ms Beats 14.19%
Memory 16.3 MB Beats 11.5%

Click the distribution chart to view more details

```
class Solution:
    def numDecodings(self, s: str) -> int:
        self.memo = {}
        return self.helper(s)

    def helper(self, s: str) -> int:
        if len(s) == 0: return 1
        if s in self.memo: return self.memo[s]

        takeOne = takeTwo = 0

        if int(s[:1]) >= 1 and int(s[:1]) <= 9:
            takeOne = self.helper(s[1:])

        if int(s[:2]) >= 10 and int(s[:2]) <= 26:
            takeTwo = self.helper(s[2:])

        self.memo[s] = takeOne + takeTwo

        return self.memo[s]
```

• **Maximum Subarray**



Description | Editorial | Solutions (10.3K) | Submissions

53. Maximum Subarray

Medium 29.5K 1.3K

Companies

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:

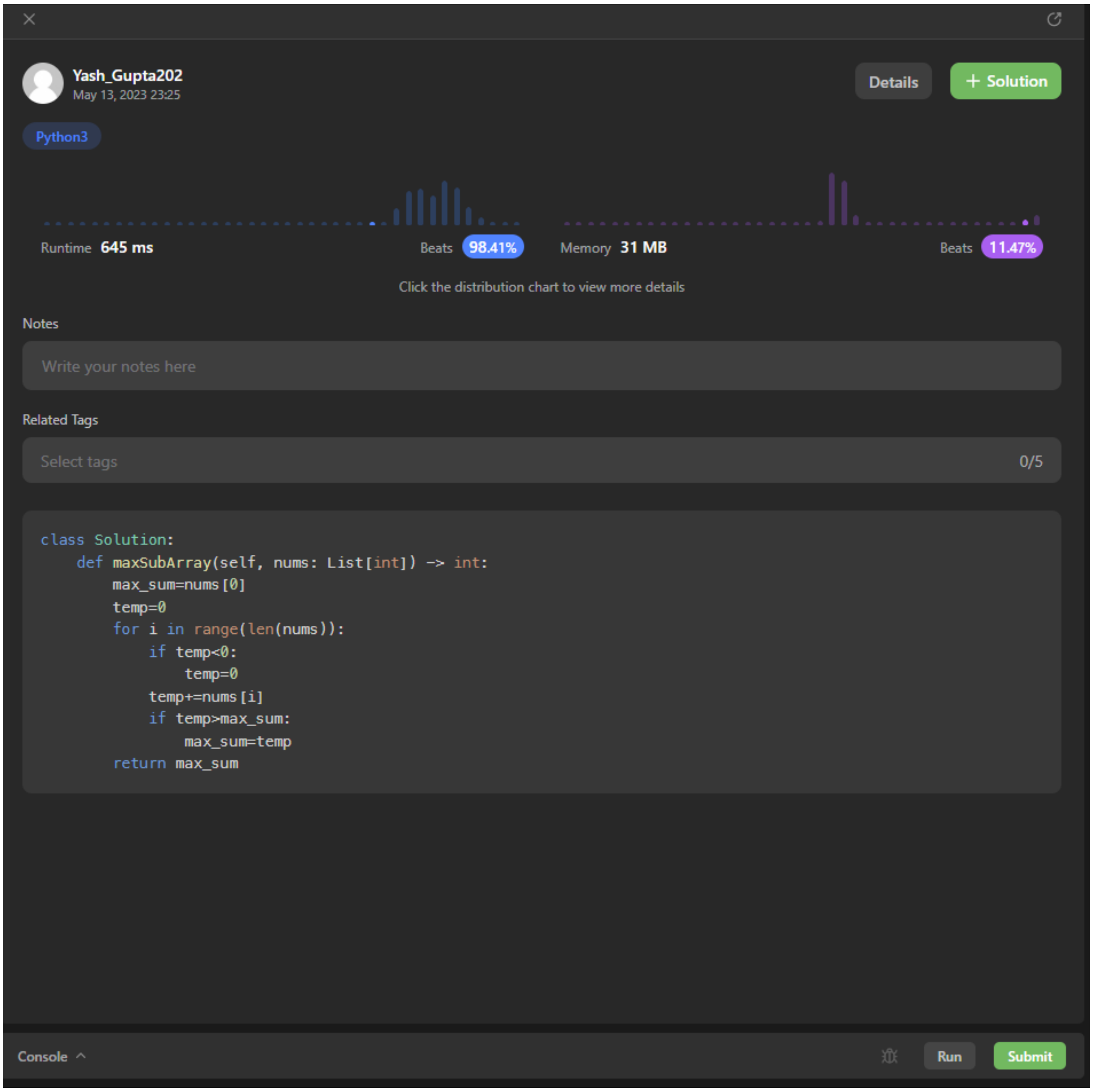
```
Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.
```

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.

```
class Solution:
    def maxSubArray(self, nums: List[int]) -> int:
        max_sum=nums[0]
        temp=0
        for i in range(len(nums)):
            if temp<0:
                temp=0
            temp+=nums[i]
            if temp>max_sum:
                max_sum=temp
           return max_sum
```



```python
class Solution:
    def maxSubArray(self, nums: List[int]) -> int:
        max_sum=nums[0]
        temp=0
        for i in range(len(nums)):
            if temp<0:
                temp=0
            temp+=nums[i]
            if temp>max_sum:
                max_sum=temp
        return max_sum
```