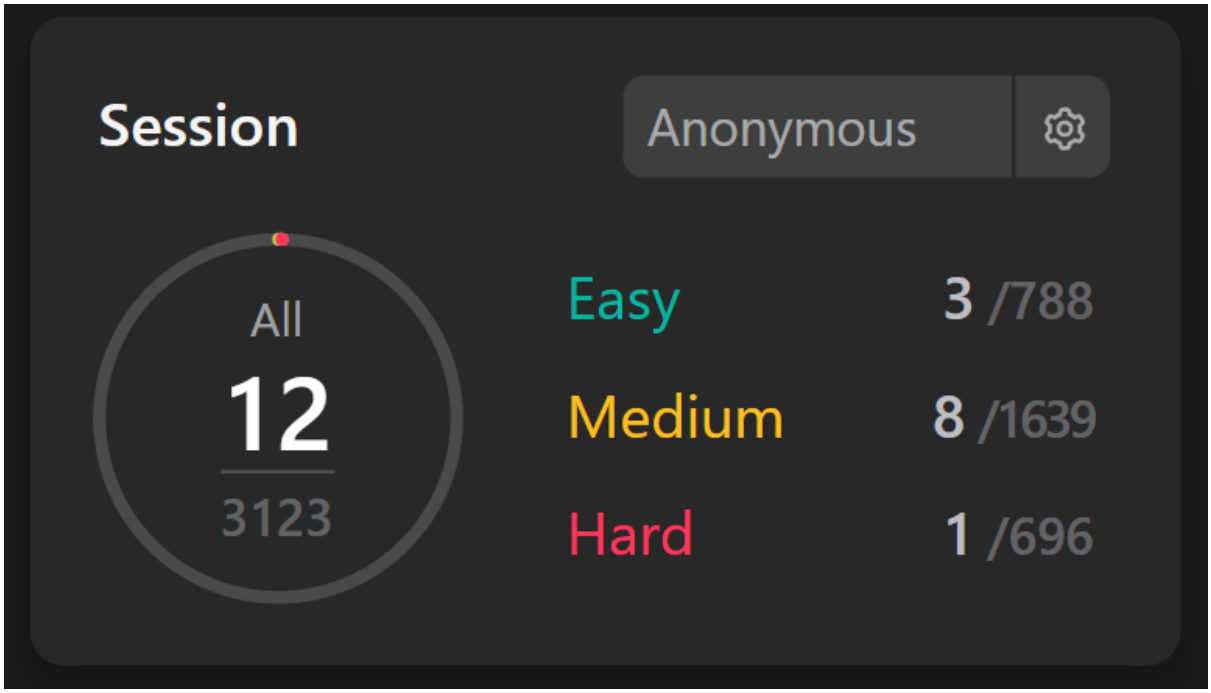
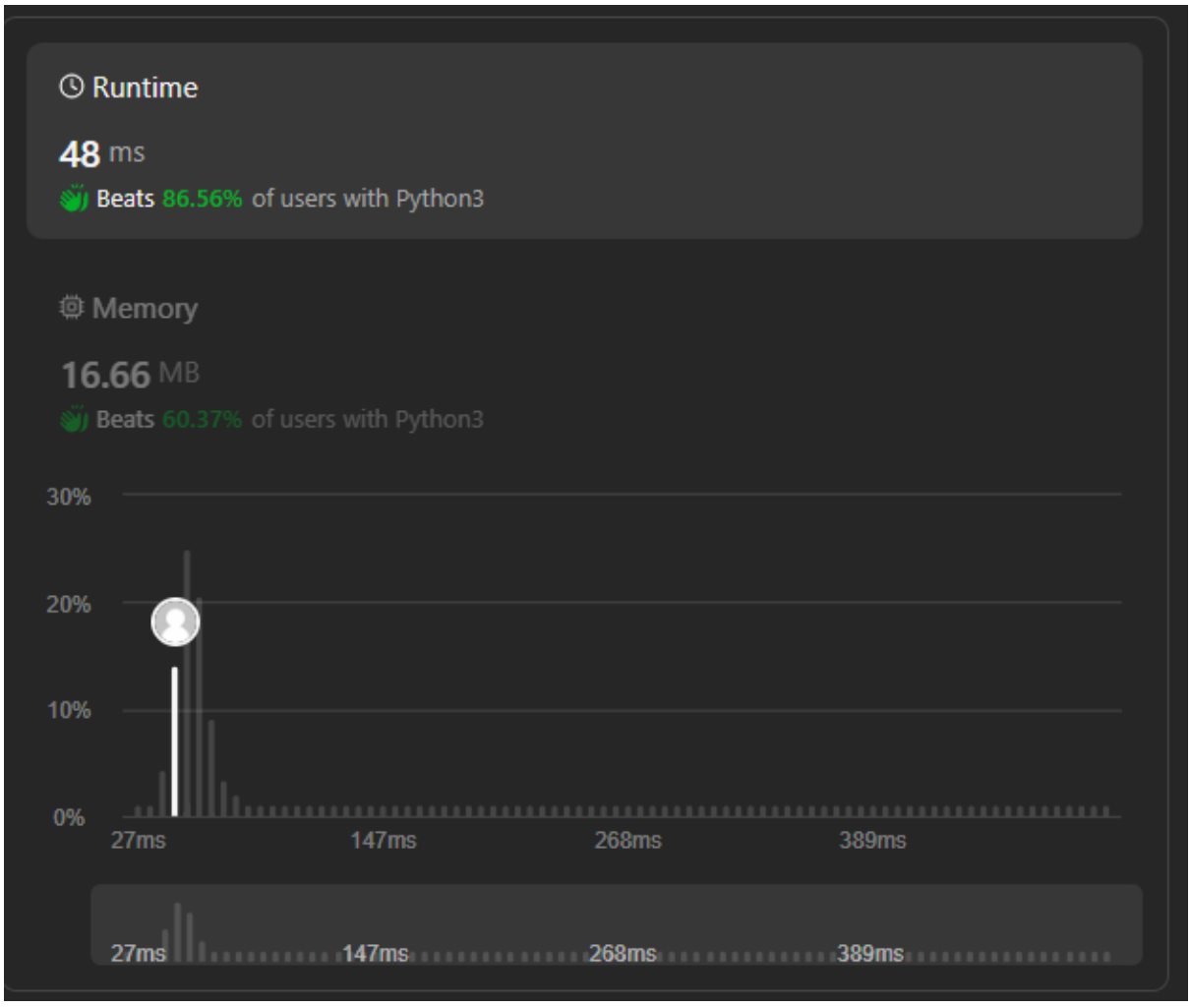
Raghav Sejpal – LeetCode Problems



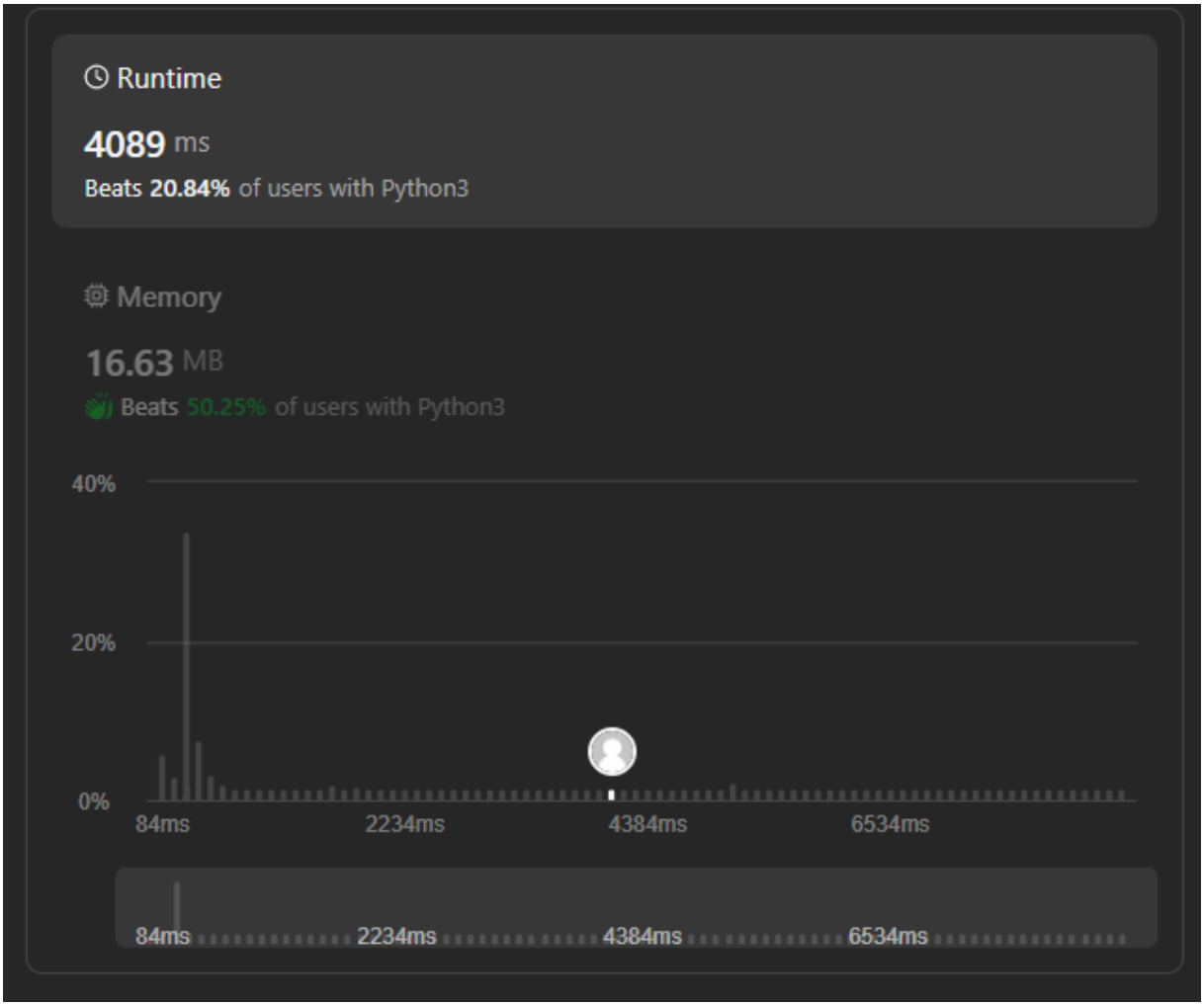
Problem 3 : Longest substring without repeating characters

```python
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        char={}
        start_idx=0
        maxLen=0

        for i in range(len(s)):
            if s[i] not in char or char[s[i]] < start_idx:
                char[s[i]]=i
                maxLen=max(maxLen,char[s[i]]-start_idx+1)
            else:
                start_idx=char[s[i]]+1
                char[s[i]]=i

        return maxLen
```



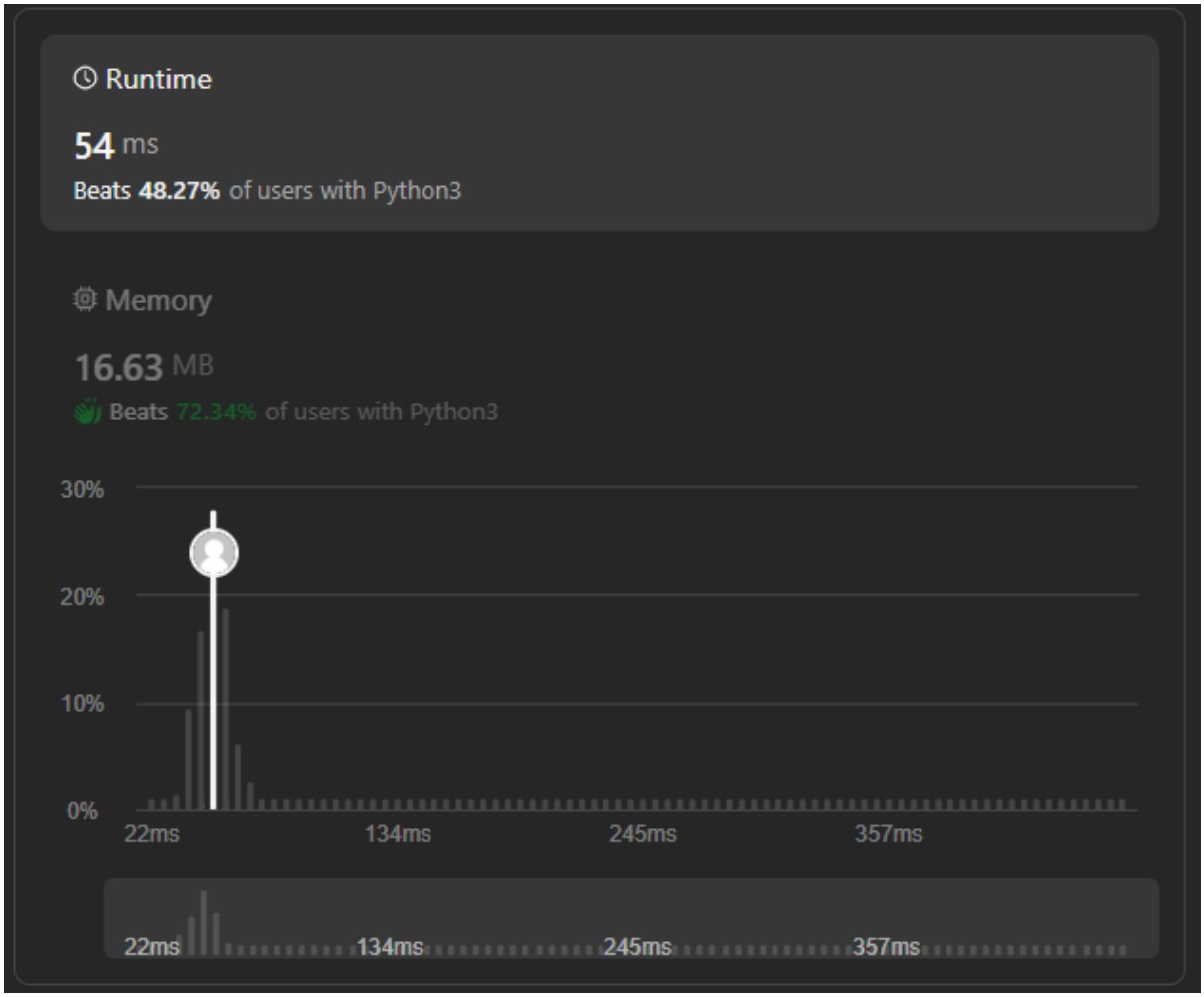
Problem 5 : Longest Palindromic Substring

```python
class Solution:
    def longestPalindrome(self, s: str) -> str:
        max_len=0
        ans=''

        if not s:
            return s

        if s[::-1]==s:
            return s

        for i in range(len(s)):
            for j in range(i+1,len(s)+1):
                cur=s[i:j]

                if cur==cur[::-1]:
                    l=len(cur)

                    if l>max_len:
                        max_len=l
                        ans=cur
        return ans
```


Runtime
4089 ms
Beats 20.84% of users with Python3
Memory
16.63 MB
Beats 50.25% of users with Python3
40%
20%
0%
84ms
2234ms
4384ms
6534ms
84ms
2234ms
4384ms
6534ms

Problem 6 : Zig Zag Conversion

```python
class Solution:
    def convert(self, s: str, numRows: int) -> str:
        if numRows<=1:return s

        l = ['']*numRows
        i=0
        idx=0
        switch=False

        while i <= numRows and idx<len(s) :

            if i in [0,numRows] :
                i=0
                switch = not switch

            if switch:
                l[i] += s[idx]

            else:

                if numRows-2-i>0:
                    l[numRows-2-i]+=s[idx]

                elif numRows-2-i==0 or i==numRows-1:
                    idx-=1

            i+=1
            idx+=1

        return ''.join(l)
```

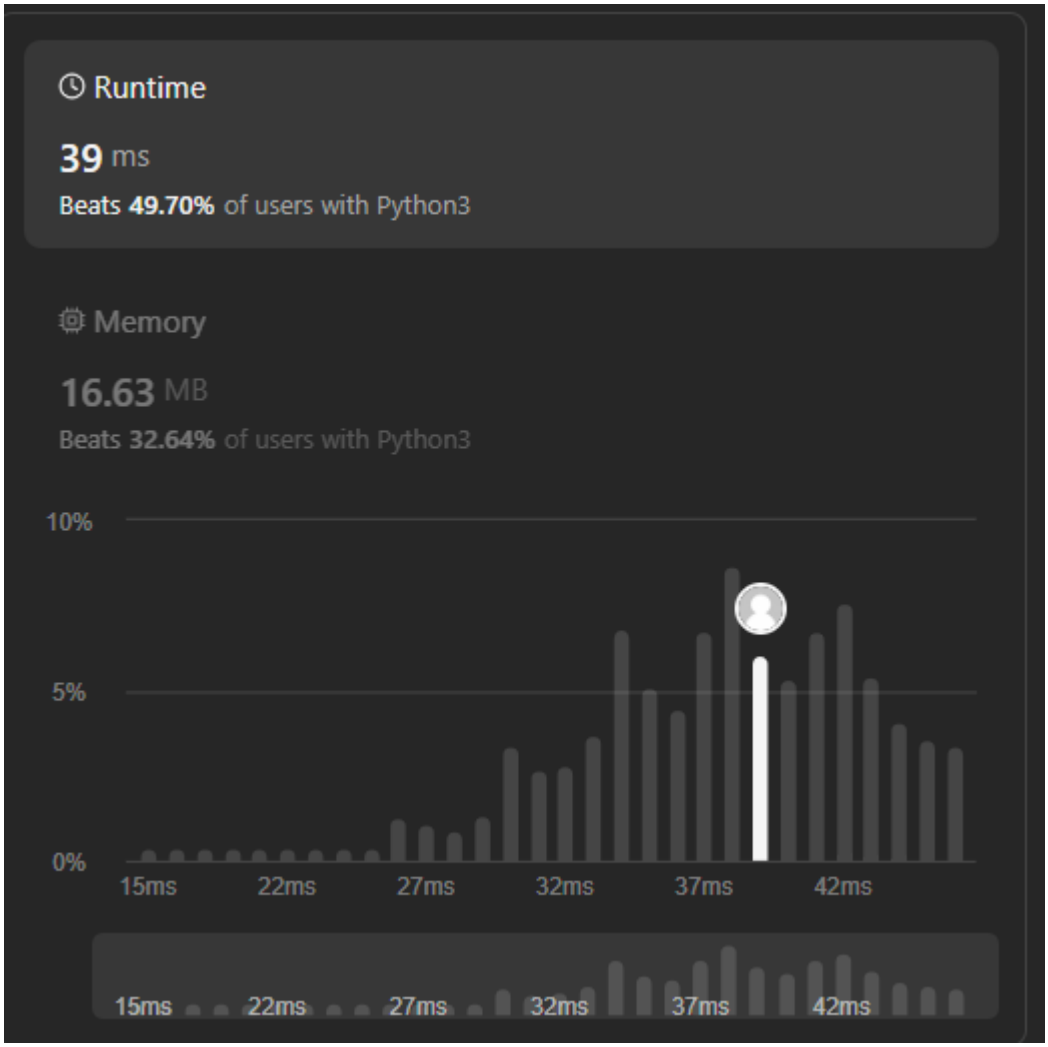

Runtime
54 ms
Beats 48.27% of users with Python3
Memory
16.63 MB
Beats 72.34% of users with Python3
30%
20%
10%
0%
22ms
134ms
245ms
357ms
22ms
134ms
245ms
357ms

Problem 7 : Reverse Integer

```python
class Solution:
    def reverse(self, x: int) -> int:
        int_min, int_max = -2**31, 2**31 - 1
        sign = -1 if x<0 else 1
        x=abs(x)
        ans=0
        while x>0:
            ans=(10*ans+x%10)
            x=x//10


        if sign*ans < int_min or sign*ans > int_max:
            return 0
        return sign*ans
```

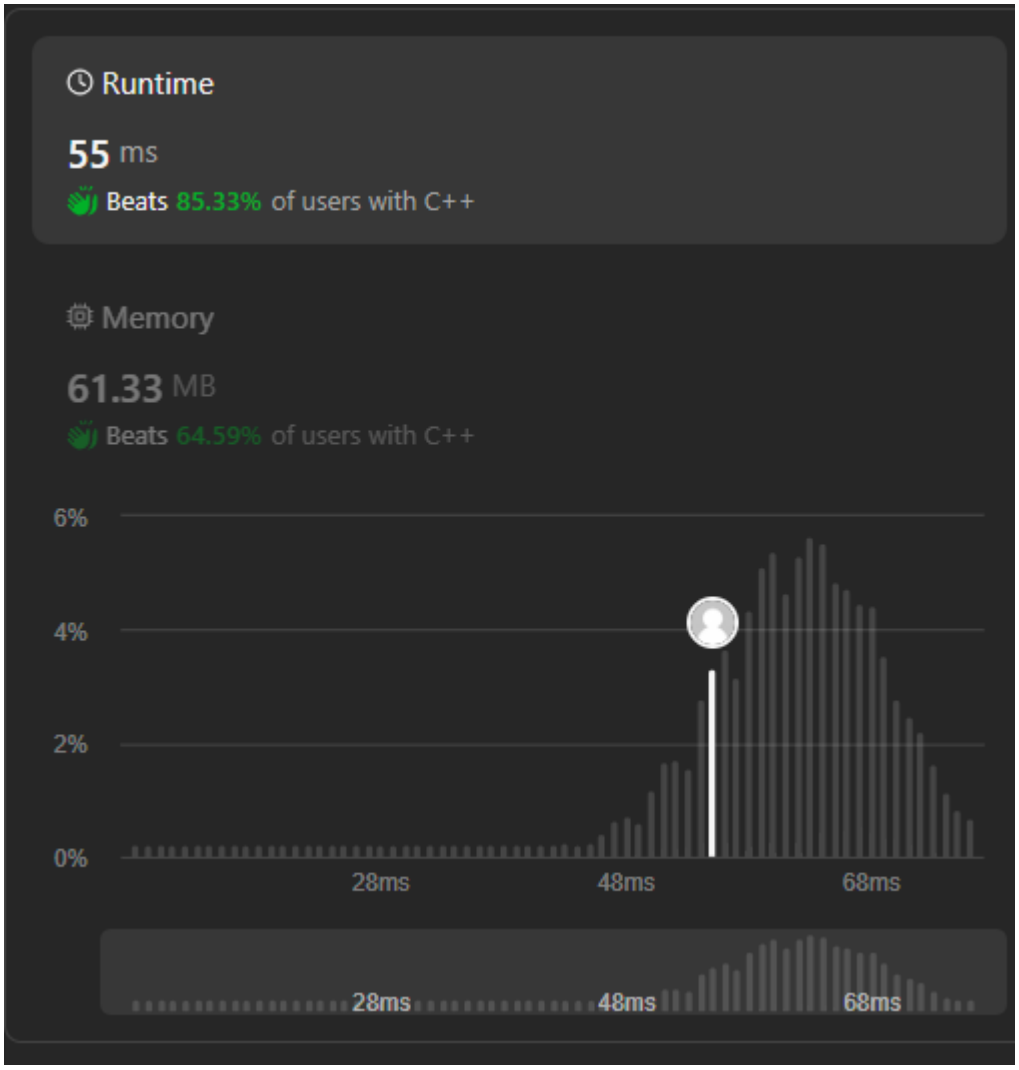
Problem 8 : String to Integer (atoi)

```python
class Solution:
    def myAtoi(self, s: str) -> int:
        int_min, int_max = -2**31, 2**31 - 1

        i = 0
        while i < len(s) and s[i] == ' ':
            i += 1

        if i < len(s) and (s[i] == '-' or s[i] == '+'):
            sign = -1 if s[i] == '-' else 1
            i += 1
        else:
            sign = 1

        result = 0
        while i < len(s) and s[i].isdigit():
            result = result * 10 + ord(s[i]) - ord('0')

            if result * sign > int_max:
                return int_max
            elif result * sign < int_min:
                return int_min

            i += 1

        result *= sign

        return result
```


Runtime
39 ms
Beats 49.70% of users with Python3
Memory
16.63 MB
Beats 32.64% of users with Python3
10%
5%
0%
15ms
22ms
27ms
32ms
37ms
42ms
15ms
22ms
27ms
32ms
37ms
42ms

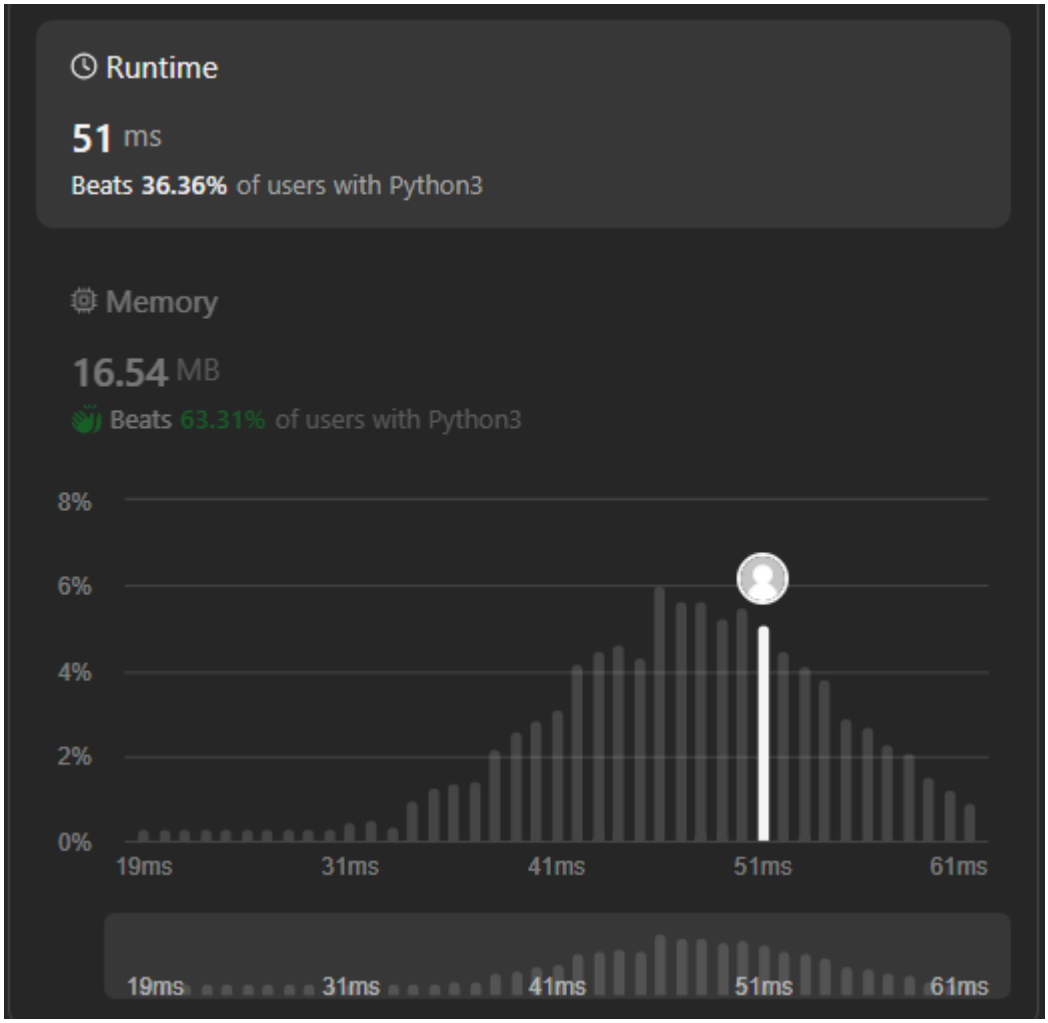
Problem 11 : Container with most water

```cpp
class Solution {
public:
    int maxArea(vector<int>& height) {
        int n = 0;
        // Calculates the length of the array height
        for(auto k : height){
            n++;
        }
        int l = 0;
        int r=n-1;
        int temp = 0;
        int m = 0;
        while(l<r) {
            temp = (r-l) * min(height[l], height[r]);
            m = max(m, temp);
            if(height[l] < height[r]){
                l++;
            }
            else{
                r--;
            }
        }
        return m;
    }
};
```


Runtime
55 ms
Beats 85.33% of users with C++
Memory
61.33 MB
Beats 64.59% of users with C++
6%
4%
2%
0%
28ms
48ms
68ms
28ms
48ms
68ms

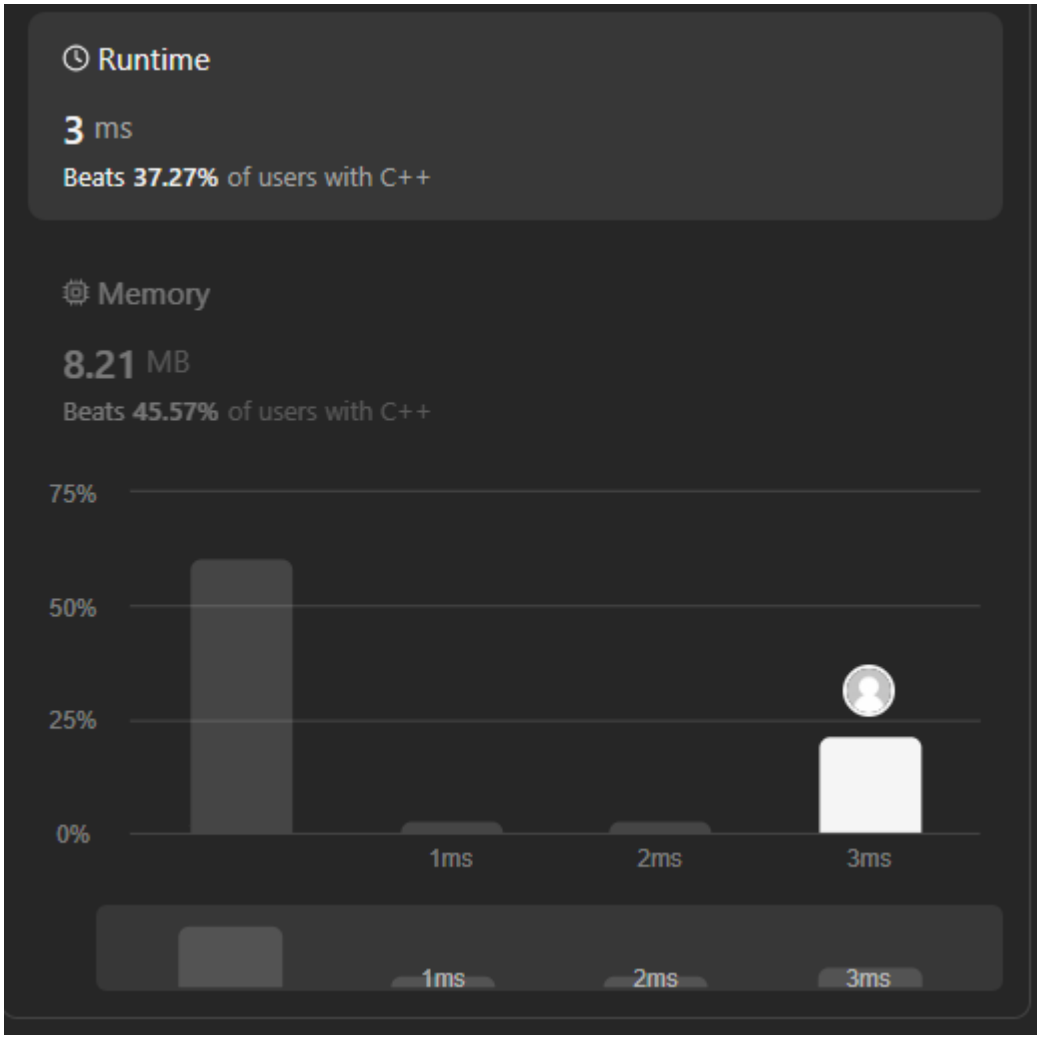
Problem 12 : Integer to Roman

```python
class Solution:
    def intToRoman(self, num: int) -> str:
        ans=''
        d={
            'M' : 1000,
            'D' : 500,
            'C' : 100,
            'L' : 50,
            'X' : 10,
            'V' : 5,
            'I' : 1
            }
        keys=list(d.keys())
        for i in range(len(keys)) :
            while num >= d[keys[i]] :
                if i == len(keys)-1 :
                    if num >= 4*d[keys[i]] :
                        ans += f'{keys[i]}{keys[i-1]}'
                        num -= 4*d[keys[i]]
                    elif num >= d[keys[i]]:
                        ans += keys[i]
                        num -= d[keys[i]]

                else:
                    if num >= 4*d[keys[i+1]]+d[keys[i]] and i not in [0,2,4] :
                        ans += f'{keys[i+1]}{keys[i-1]}'
                        num -= 4*d[keys[i+1]]+d[keys[i]]
                    elif num >= 4*d[keys[i]] :
                        ans += f'{keys[i]}{keys[i-1]}'
                        num -= 4*d[keys[i]]
                    elif num >= d[keys[i]]:
                        ans += keys[i]
                        num -= d[keys[i]]
        return ans
```


Runtime
51 ms
Beats 36.36% of users with Python3
Memory
16.54 MB
Beats 63.31% of users with Python3
8%
6%
4%
2%
0%
19ms
31ms
41ms
51ms
61ms
19ms
31ms
41ms
51ms
61ms

Problem 17 : Letter Combinations of a Phone Number

```
using namespace std;
class Solution {
public:
    vector<string> letterCombinations(string digits) {
        if (digits.empty()) return {};

        string letters[] = {"abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv",
"wxyz"};
        vector<string> result;
        recursive("", digits, letters, result);
        return result;
    }

private:
    void recursive(string combination, string next_digits, string letters[],
vector<string>& result) {
        if (next_digits.empty()) {
            result.push_back(combination);
        } else {
            string l = letters[next_digits[0] - '2'];
            for (char letter : l) {
                recursive(combination + letter, next_digits.substr(1), letters,
result);
            }
        }
    }
};
```


Runtime
3 ms
Beats 37.27% of users with C++
Memory
8.21 MB
Beats 45.57% of users with C++
75%
50%
25%
0%
1ms
2ms
3ms
1ms
2ms
3ms

Problem 2448 : Minimum Cost to Make an Array Equal (**Hard** Problem)

```python
class Solution:
    def minCost(self, nums: List[int], cost: List[int]) -> int:
        pairs=sorted(list(zip(nums,cost)))
        med=sum(cost)/2
        c=0
        for i in range(len(pairs)):
            c=c+pairs[i][1]
            if c >= med:
                break
        target=pairs[i][0]
        finCost=0
        for i in range(len(pairs)):
            finCost=finCost+(abs(pairs[i][0]-target) * pairs[i][1])
        return finCost
```