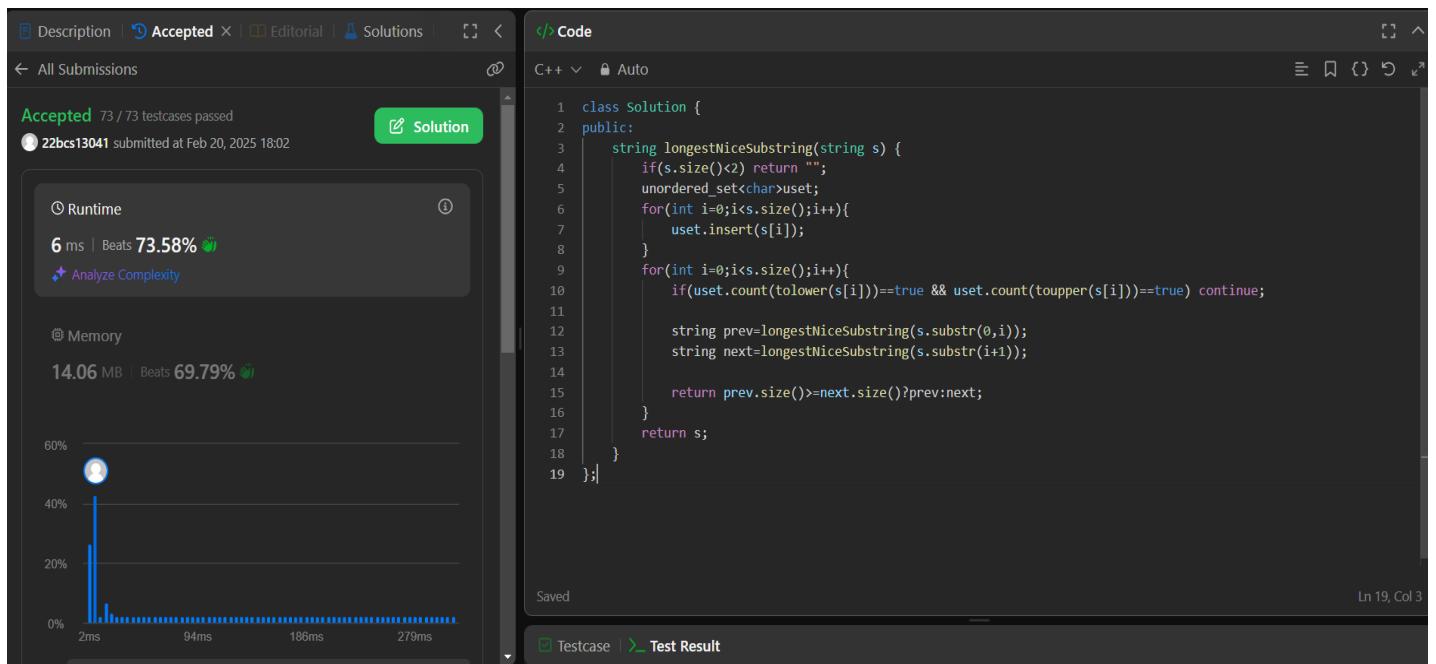


1 Longest Nice Substring

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        if(s.size()<2) return "";
        unordered_set<char>uset;
        for(int i=0;i<s.size();i++){
            uset.insert(s[i]);
        }
        for(int i=0;i<s.size();i++){
            if(uset.count(tolower(s[i]))==true && uset.count(toupper(s[i]))==true) continue;

            string prev=longestNiceSubstring(s.substr(0,i));
            string next=longestNiceSubstring(s.substr(i+1));

            return prev.size()>=next.size()?prev:next;
        }
        return s;
    }
};
```



The screenshot displays the LeetCode submission interface for the 'Longest Nice Substring' problem. The left sidebar shows the submission status as 'Accepted' with 73/73 test cases passed. The runtime is 6 ms, beating 73.58% of submissions, and the memory usage is 14.06 MB, beating 69.79%. A performance graph at the bottom left shows a single data point at 2ms. The main area on the right displays the C++ code for the solution, which is a recursive function that checks for nice substrings by comparing the current string with its prefixes and suffixes. The code is saved and the current cursor position is at line 19, column 3.

Accepted 73 / 73 testcases passed
22bcs13041 submitted at Feb 20, 2025 18:02

Runtime
6 ms | Beats 73.58%
[Analyze Complexity](#)

Memory
14.06 MB | Beats 69.79%

60%
40%
20%
0%

2ms 94ms 186ms 279ms

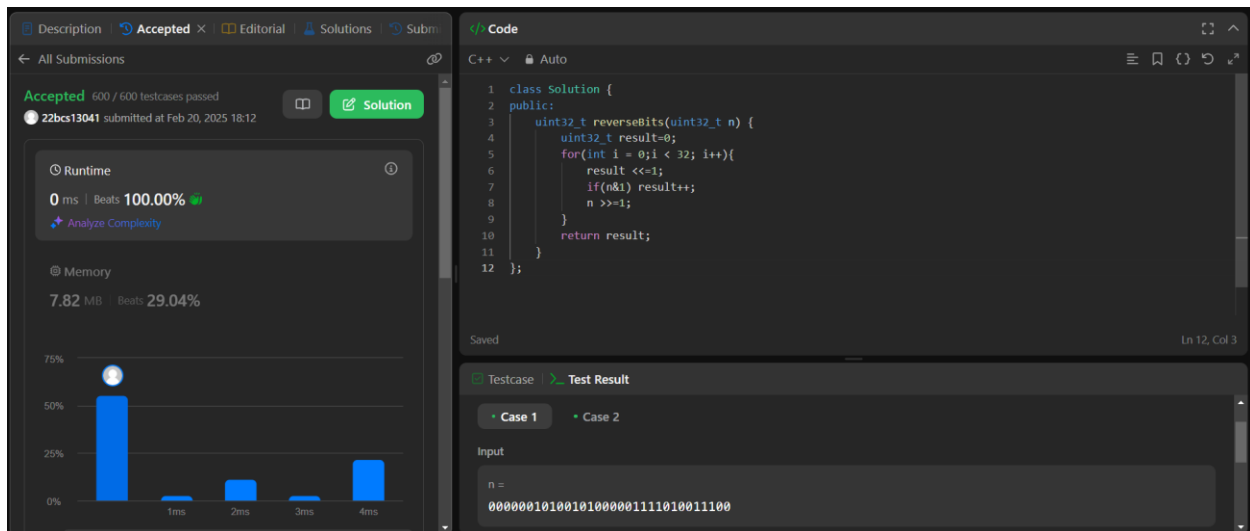
```
1 class Solution {
2 public:
3     string longestNiceSubstring(string s) {
4         if(s.size()<2) return "";
5         unordered_set<char>uset;
6         for(int i=0;i<s.size();i++){
7             uset.insert(s[i]);
8         }
9         for(int i=0;i<s.size();i++){
10            if(uset.count(tolower(s[i]))==true && uset.count(toupper(s[i]))==true) continue;
11
12            string prev=longestNiceSubstring(s.substr(0,i));
13            string next=longestNiceSubstring(s.substr(i+1));
14
15            return prev.size()>=next.size()?prev:next;
16        }
17        return s;
18    }
19 };
```

Saved
Ln 19, Col 3

Testcase Test Result

2. Reverse bits

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result=0;
        for(int i = 0; i < 32; i++){
            result <<=1;
            if(n&1) result++;
            n >>=1;
        }
        return result;
    }
};
```



3 Number of 1 bit

```
class Solution {
public:
    int hammingWeight(int n) {
        int count=0;
        while(n){
            if(n & 1) count++;
            n >>= 1;
        }
        return count;
    }
};
```

Accepted 598 / 598 testcases passed
22bcs13041 submitted at Feb 20, 2025 18:16

Runtime: 0 ms | Beats 100.00%
Memory: 8.20 MB | Beats 80.44%

```

1 class Solution {
2 public:
3     int hammingWeight(int n) {
4         int count=0;
5         while(n){
6             if(n & 1) count++;
7             n >>= 1;
8         }
9         return count;
10    }
11 };

```

Testcase 1: n = 11

4. maximum Subarray

```

class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int current_sum = nums[0];
        int max_sum = nums[0];

        for (size_t i = 1; i < nums.size(); ++i) {
            current_sum = std::max(nums[i], current_sum + nums[i]);
            max_sum = std::max(max_sum, current_sum);
        }

        return max_sum;
    }
};

```

Accepted 210 / 210 testcases passed
22bcs13041 submitted at Feb 20, 2025 18:23

Runtime: 0 ms | Beats 100.00%
Memory: 71.66 MB | Beats 80.65%

```

1 class Solution {
2 public:
3     int maxSubArray(vector<int>& nums) {
4         int current_sum = nums[0];
5         int max_sum = nums[0];
6
7         for (size_t i = 1; i < nums.size(); ++i) {
8             current_sum = std::max(nums[i], current_sum + nums[i]);
9             max_sum = std::max(max_sum, current_sum);
10        }
11
12        return max_sum;
13    }
14 };

```

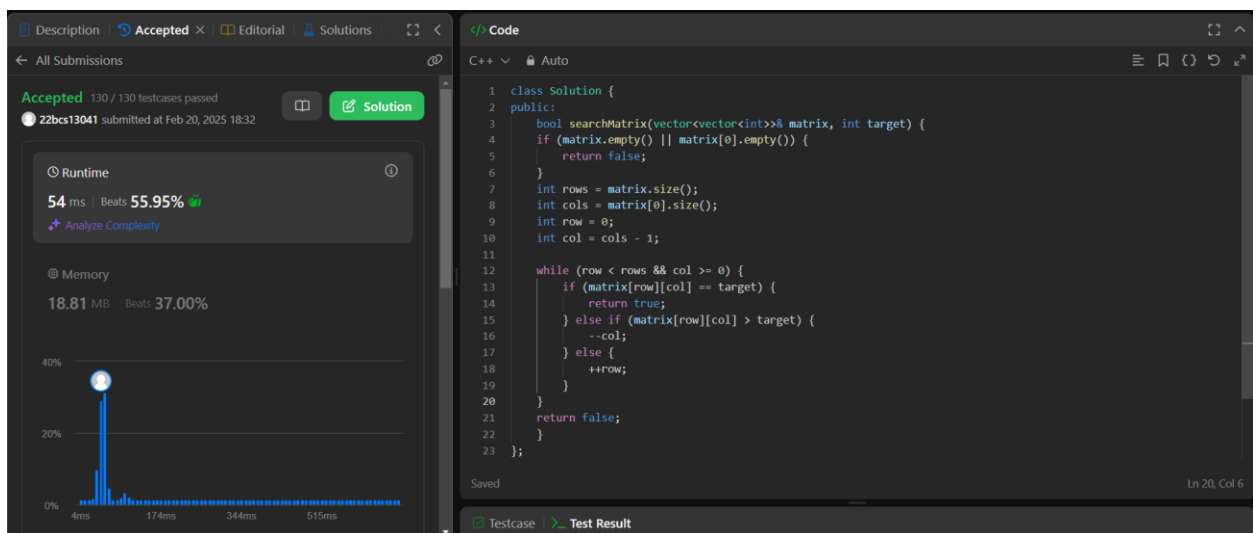
Testcase 1: Runtime: 0 ms

Input: nums =

5. Search a 2D Matrix

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        if (matrix.empty() || matrix[0].empty()) {
            return false;
        }
        int rows = matrix.size();
        int cols = matrix[0].size();
        int row = 0;
        int col = cols - 1;

        while (row < rows && col >= 0) {
            if (matrix[row][col] == target) {
                return true;
            } else if (matrix[row][col] > target) {
                --col;
            } else {
                ++row;
            }
        }
        return false;
    }
};
```



6. Search Pow

```
class Solution {
public:
    int modPow(int a, int b, int mod) {
        int result = 1;
        a %= mod;
        while (b > 0) {
            if (b % 2 == 1) {
                result = (result * a) % mod;
            }
            a = (a * a) % mod;
            b /= 2;
        }
        return result;
    }
    int superPow(int a, vector<int>& b) {
        const int mod = 1337;
        int result = 1;
        for (int digit : b) {
            result = modPow(result, 10, mod) * modPow(a, digit, mod) % mod;
        }
        return result;
    }
};
```

