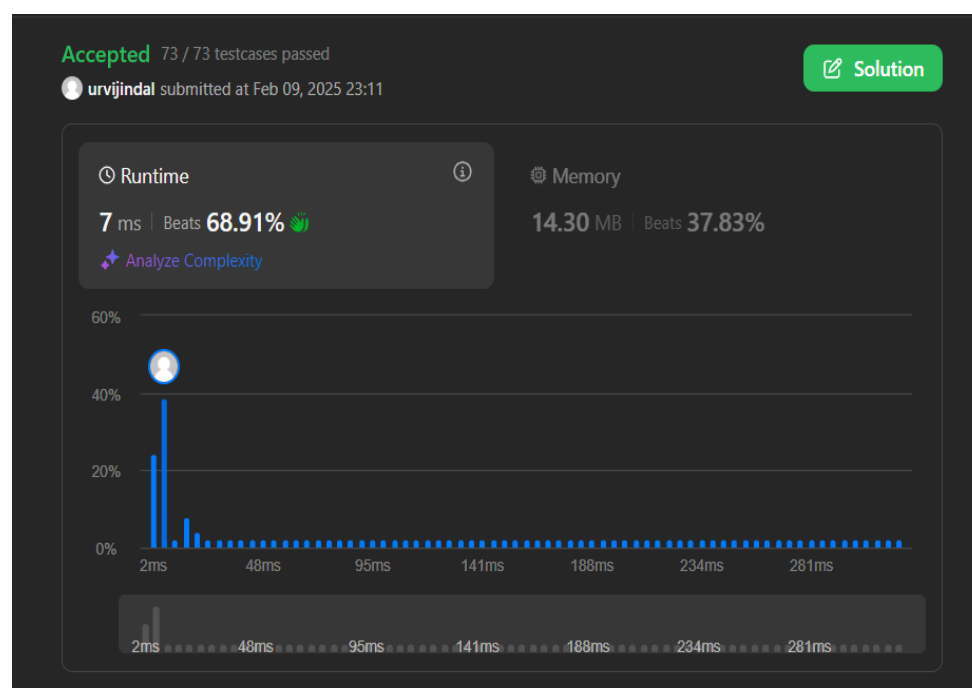


1763. Longest Nice Substring



```
C++ v Auto

1  class Solution {
2  public:
3      string longestNiceSubstring(string s) {
4          if (s.length() < 2) return "";
5          unordered_set<char> charSet(s.begin(), s.end());
6          for (int i = 0; i < s.length(); i++) {
7              if (charSet.count(tolower(s[i])) && charSet.count(toupper(s[i])))
8                  continue;
9              string left = longestNiceSubstring(s.substr(0, i));
10             string right = longestNiceSubstring(s.substr(i + 1));
11             return left.length() >= right.length() ? left : right;
12         }
13         return s;
14     }
15 };
16
```



190. Reverse Bits

</> Code


C++   Auto


```
1  class Solution {
2  public:
3      uint32_t reverseBits(uint32_t n) {
4          int result=0;
5          for(int i=0;i<32;i++)
6          {
7              result=result<<1;
8              result=result|(n&1);
9              n=n>>1;
10         }
11         return result;
12     }
13 };
```


Saved

← All Submissions

Accepted 600 / 600 testcases passed

 urvijindal submitted at Feb 09, 2025 23:08

 Editorial

 Solution

⌚ Runtime

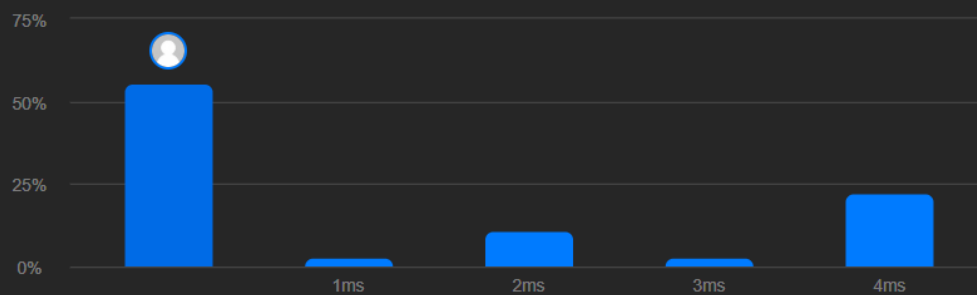
ⓘ

0 ms | Beats 100.00% 🏆

🔮 Analyze Complexity

💾 Memory

7.81 MB | Beats 29.19%

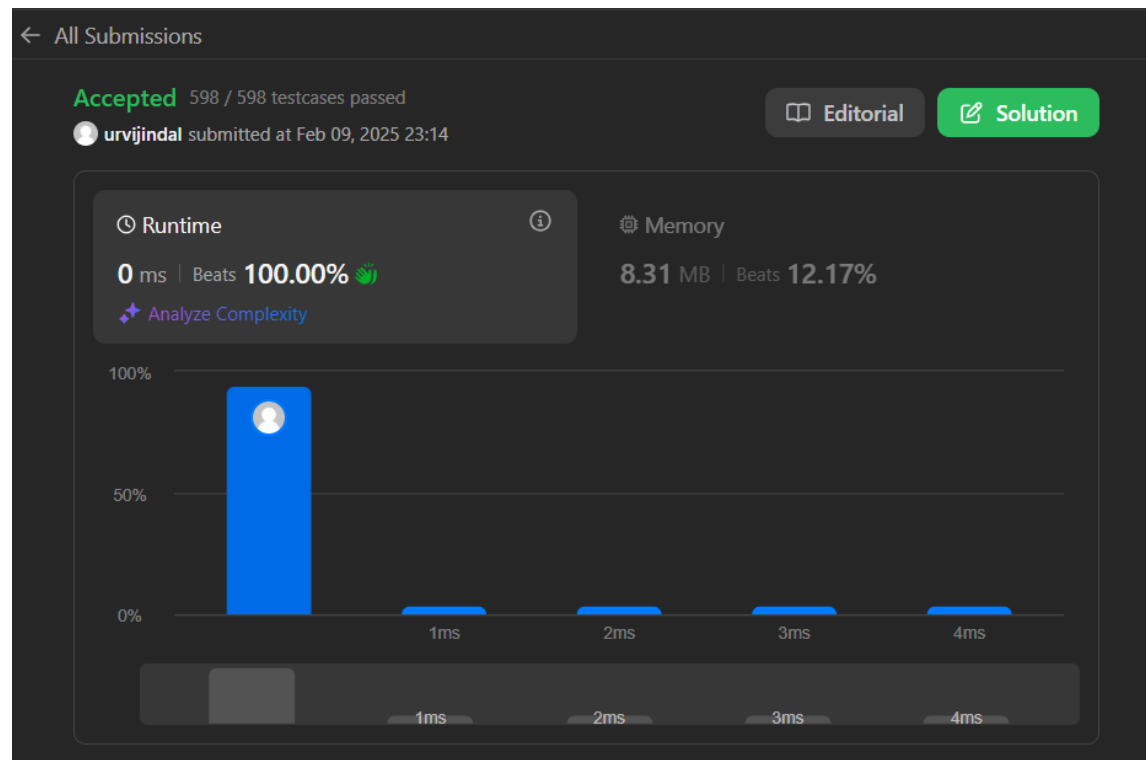


Code | C++

191. Number of 1 Bits

```
C++  Auto

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



53. Maximum Subarray

C++ Auto

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

Accepted 210 / 210 testcases passed

urvijindal submitted at Feb 09, 2025 23:17

Editorial

Solution

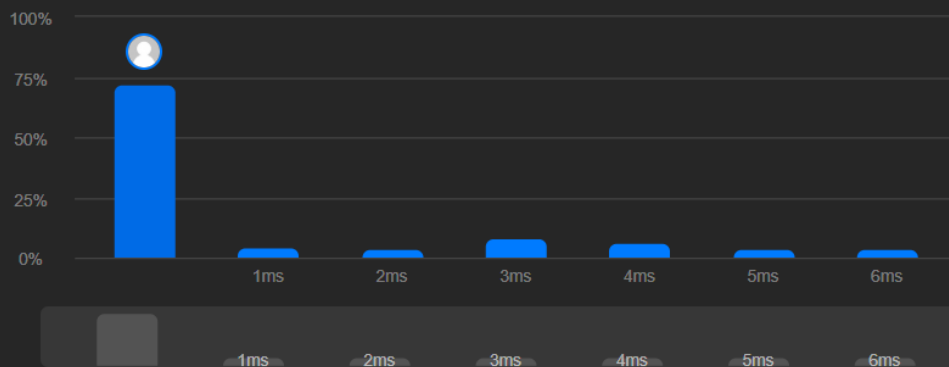
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

71.63 MB | Beats 80.64%



240. [Search a 2D Matrix II](#)


</> Code

Java  Auto

```
1 public class Solution {
2     public boolean searchMatrix(int[][] matrix, int target) {
3         for (int i = 0; i < matrix.length; i++) {
4             for (int j = 0; j < matrix[i].length; j++) {
5                 if (matrix[i][j] == target) {
6                     return true;
7                 }
8             }
9         }
10        return false;
11    }
12 }
13
```

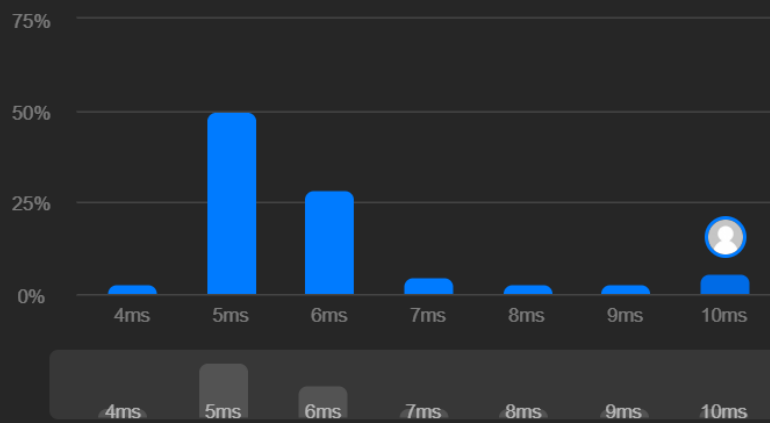
⌚ Runtime

10 ms | Beats 15.50%

 Analyze Complexity

⚙️ Memory

46.31 MB | Beats 11.44%



372. Super Pow

</> Code

Java   Auto

```
1  class Solution {
2      public int superPow(int a, int[] b) {
3          int ans = 1;
4
5          a %= kMod;
6          for (final int i : b)
7              ans = modPow(ans, 10) * modPow(a, i) % kMod;
8
9          return ans;
10     }
11
12     private static final int kMod = 1337;
13
14     private int modPow(int x, int n) {
15         if (n == 0)
16             return 1;
17         if (n % 2 == 1)
18             return x * modPow(x % kMod, (n - 1)) % kMod;
19         return modPow(x * x % kMod, (n / 2)) % kMod;
20     }
21 }
```

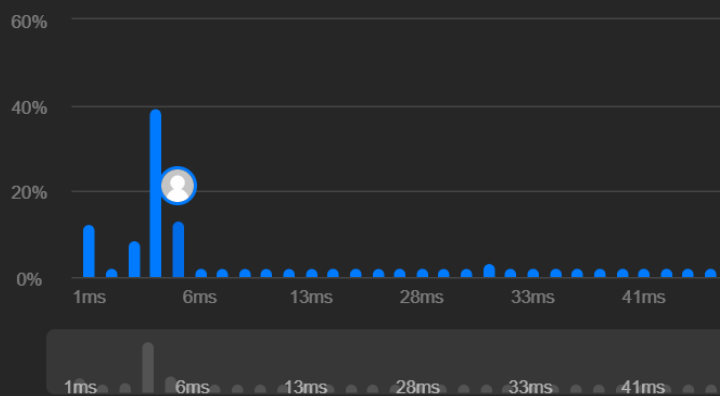
🕒 Runtime

5 ms | Beats 36.52%

🔮 Analyze Complexity

⚙️ Memory

44.49 MB | Beats 49.64%




932. Beautiful Array


 Code


Java   Auto


```
1 class Solution {
2     public int[] beautifulArray(int n) {
3         int[] arr = new int[n];
4         for (int i = 0; i < n; ++i)
5             arr[i] = i + 1;
6         divide(arr, 0, n - 1, 1);
7         return arr;
8     }
9
10    private void divide(int[] arr, int l, int r, int mask) {
11        if (l >= r)
12            return;
13        final int m = partition(arr, l, r, mask);
14        divide(arr, l, m, mask << 1);
15        divide(arr, m + 1, r, mask << 1);
16    }
17
18    private int partition(int[] arr, int l, int r, int mask) {
19        int nextSwapped = l;
20        for (int i = l; i <= r; ++i)
21            if ((arr[i] & mask) > 0)
22                swap(arr, i, nextSwapped++);
23        return nextSwapped - 1;
24    }
25
26    private void swap(int[] arr, int i, int j) {
27        final int temp = arr[i];
28        arr[i] = arr[j];
29    }
30 }
```

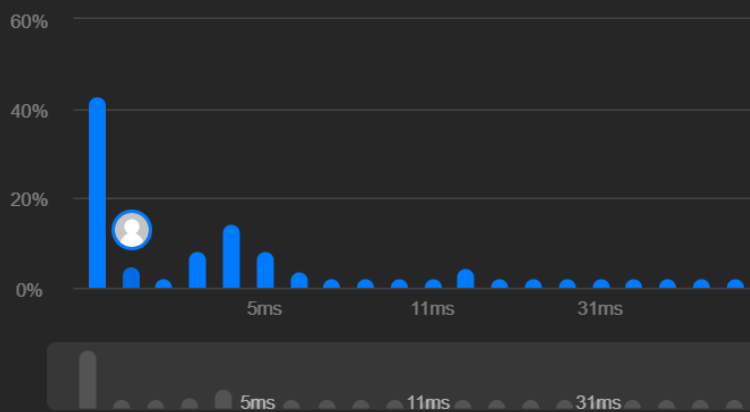
 Runtime

1 ms | Beats 57.10% 

 Analyze Complexity

 Memory

41.68 MB | Beats 98.39% 



88. Merge Sorted Array

```
C++ v Auto

1 class Solution {
2 public:
3     void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
4         int i = m - 1, j = n - 1, k = m + n - 1;
5         while (i >= 0 && j >= 0) {
6             if (nums1[i] > nums2[j]) {
7                 nums1[k--] = nums1[i--];
8             } else {
9                 nums1[k--] = nums2[j--];
10            }
11        }
12        while (j >= 0) {
13            nums1[k--] = nums2[j--];
14        }
15    }
16 };
17
```

Accepted 59 / 59 testcases passed

urvijindal submitted at Feb 09, 2025 23:23

Editorial

Solution

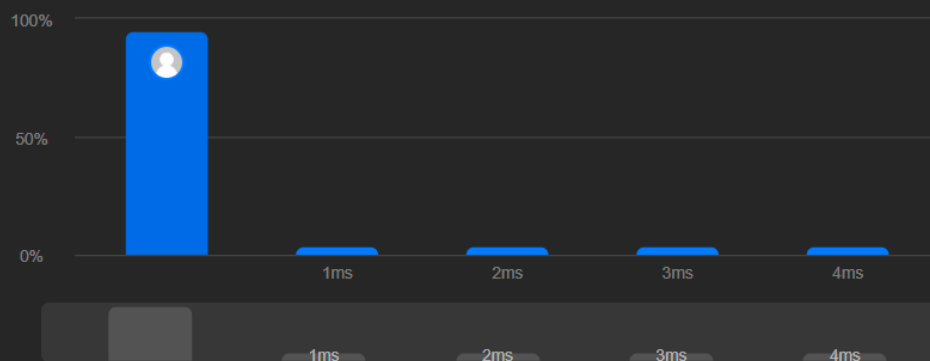
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

12.24 MB | Beats 70.68%

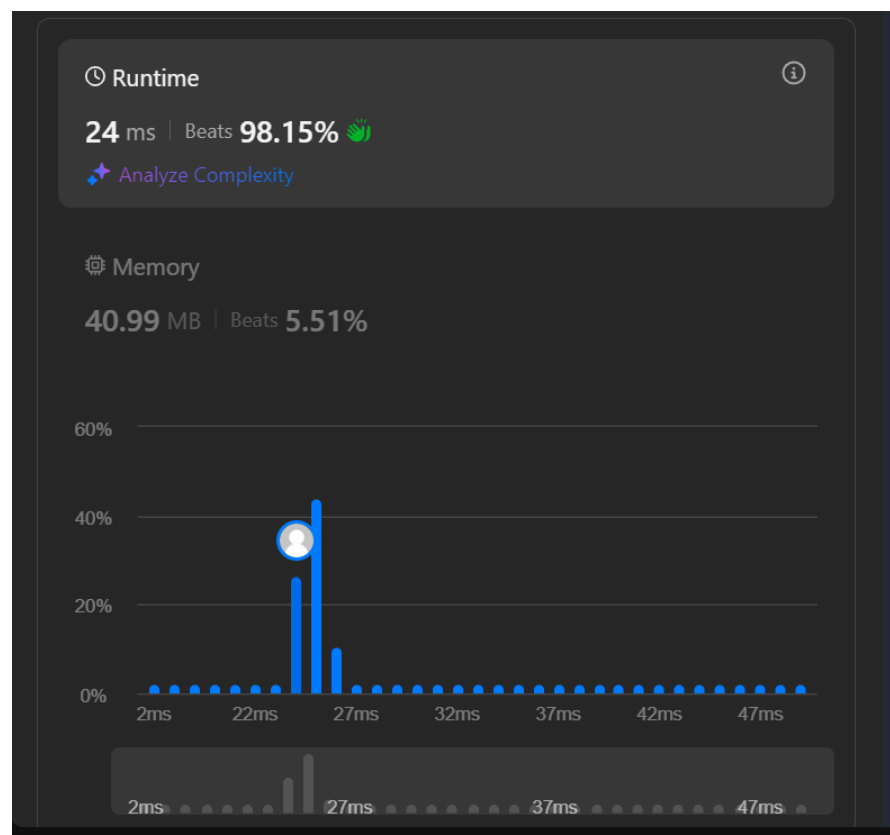


278. [First Bad Version](#)

```
</> Code

Java ▾ 🔒 Auto

1  public class Solution extends VersionControl {
2      public int firstBadVersion(int n) {
3          int l = 1;
4          int r = n;
5
6          while (l < r) {
7              final int m = 1 + (r - l) / 2;
8              if (isBadVersion(m))
9                  r = m;
10             else
11                 l = m + 1;
12         }
13
14         return l;
15     }
16 }
```



75. Sort Colors

</> Code

Java ▾ 🔒 Auto

```
1  class Solution {
2      public void sortColors(int[] nums) {
3          HashMap<Integer,Integer> map=new HashMap<>();
4          for(int i: nums){
5              map.put(i,map.getOrDefault(i,0)+1);
6          }
7          int x=0;
8          for(int i:map.keySet()){
9              int y=map.get(i);
10             for(int j=0;j<y;j++){
11                 nums[x++]=i;
12             }
13             // System.out.println(i);
14         }
15     }
16 }
```

🕒 Runtime

1 ms | Beats 22.28%

💎 Analyze Complexity

🧠 Memory

42.13 MB | Beats 29.92%

