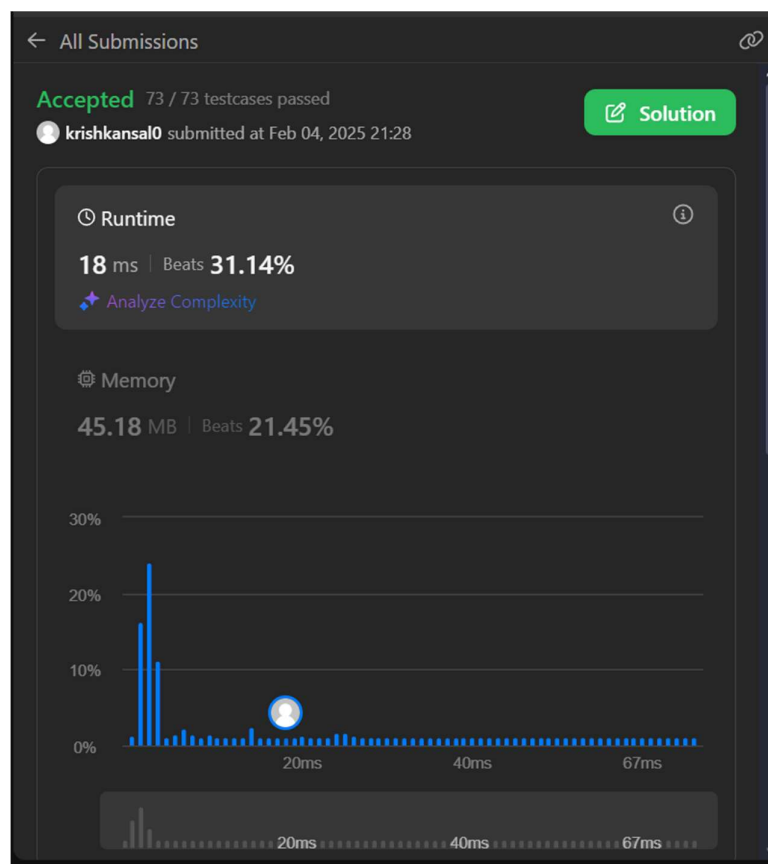


1763. [Longest Nice Substring](#)


```
</> Code
Java Auto

1 class Solution {
2     public String longestNiceSubstring(String s) {
3         if (s.length() == 0) return "";
4         for (int len = s.length(); len > 1; len--) {
5             for (int i = 0; i <= s.length() - len; i++) {
6                 String sub = s.substring(i, i + len);
7                 if (isNice(sub)) {
8                     return sub;
9                 }
10            }
11        }
12        return "";
13    }
14
15    private boolean isNice(String s) {
16        for (int i = 0; i < s.length(); i++) {
17            char c = s.charAt(i);
18            if (Character.isLowerCase(c) && !s.contains(String.valueOf(Character.toUpperCase(c)))) {
19                return false;
20            }
21            if (Character.isUpperCase(c) && !s.contains(String.valueOf(Character.toLowerCase(c)))) {
22                return false;
23            }
24        }
25        return true;
26    }
27 }
28
```

Saved



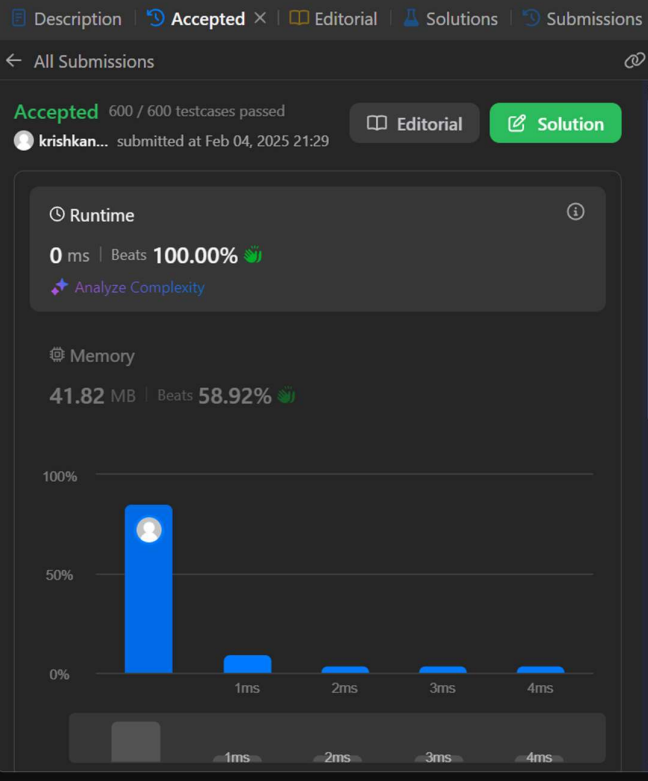
190. [Reverse Bits](#)

 Code

Java   Auto

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

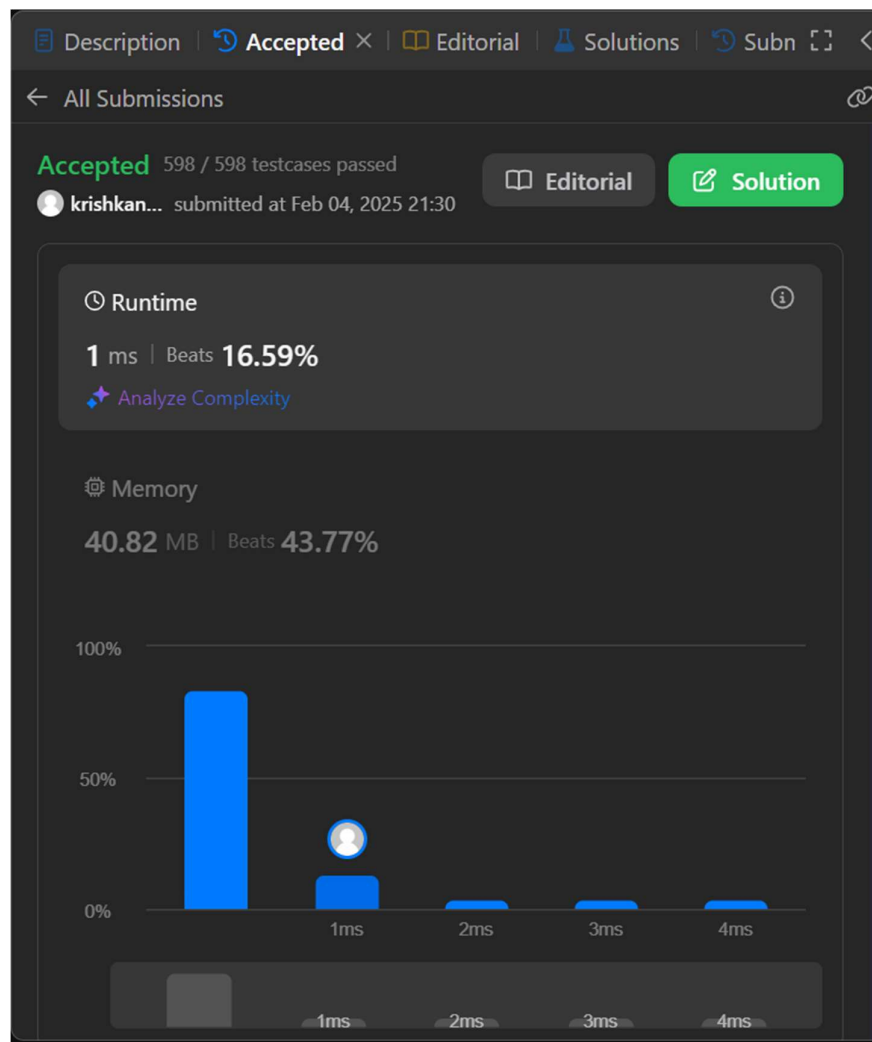
Saved



191. [Number of 1 Bits](#)

```
</>Code
Java ▾ 🔒 Auto

1  class Solution {
2      public int hammingWeight(int n) {
3          int temp=0;
4          String a=(String)Integer.toBinaryString(n);
5          for(int i=0;i<a.length();i++){
6              if(a.charAt(i)=='1'){
7                  temp++;
8              }
9          }
10         return temp;
11     }
12 }
```



53. [Maximum Subarray](#)

```
</> Code
Java Auto

1 class Solution {
2     public int maxSubArray(int[] ar) {
3         int sum=0;
4         int max=ar[0];
5         for(int c:ar){
6             sum+=c;
7             if(sum>max){
8                 max=sum;
9             }
10            if(sum<0){
11                sum=0;
12            }
13        }
14        return max;
15    }
16 }
```



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
```

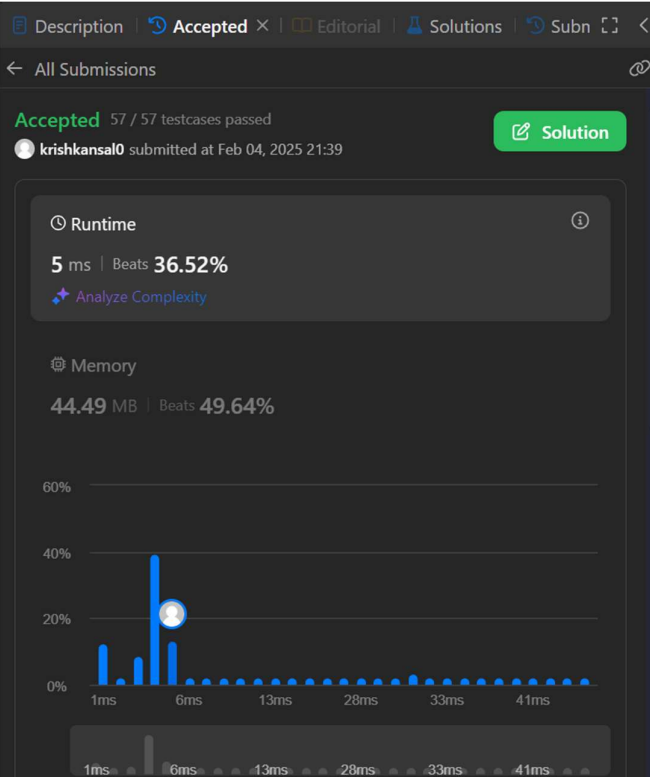


372. Super Pow

</> 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 }
```

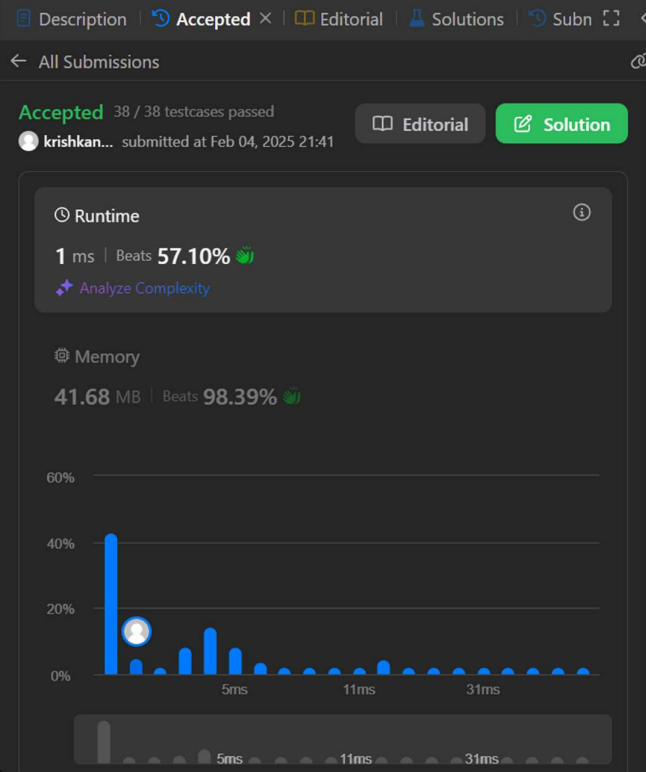


932. Beautiful Array

</> 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];
```



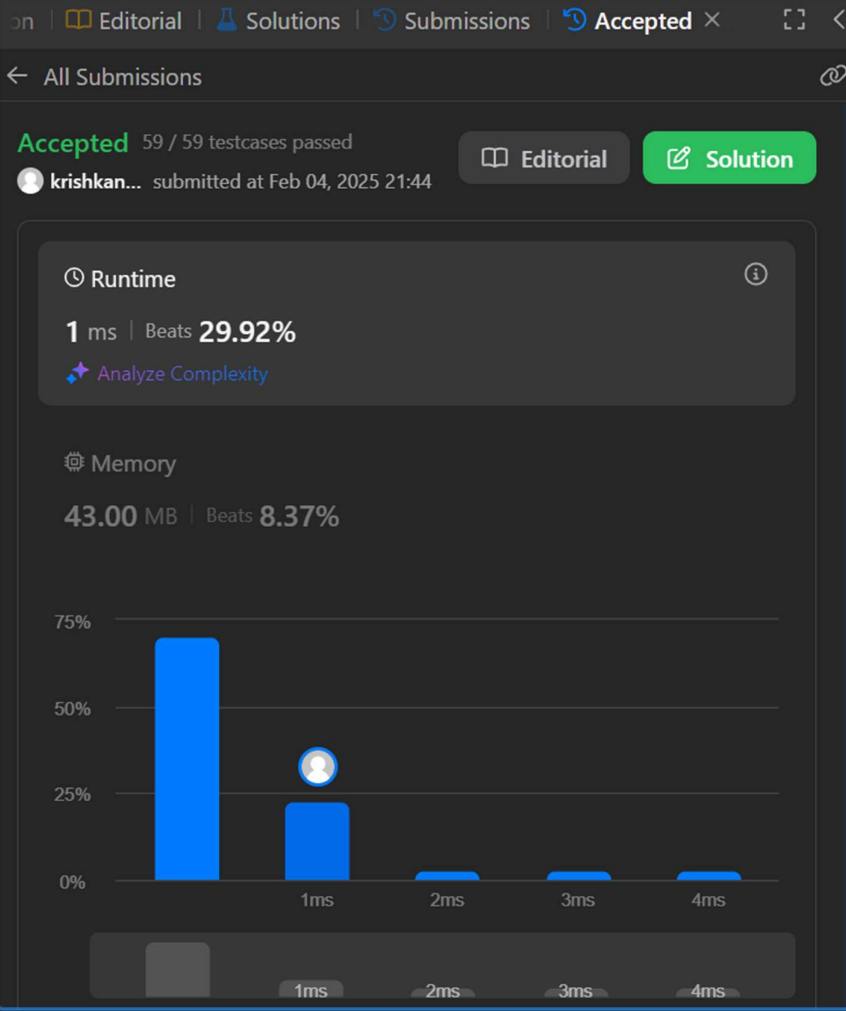
88. Merge Sorted Array

</> Code

Java ▾ 🔒 Auto

```
1 class Solution {
2     public void merge(int[] nums1, int m, int[] nums2, int n) {
3         int x=0;
4         for(int i=m;i<nums1.length;i++){
5             nums1[i]=nums2[x++];
6         }
7         Arrays.sort(nums1);
8     }
9 }
```

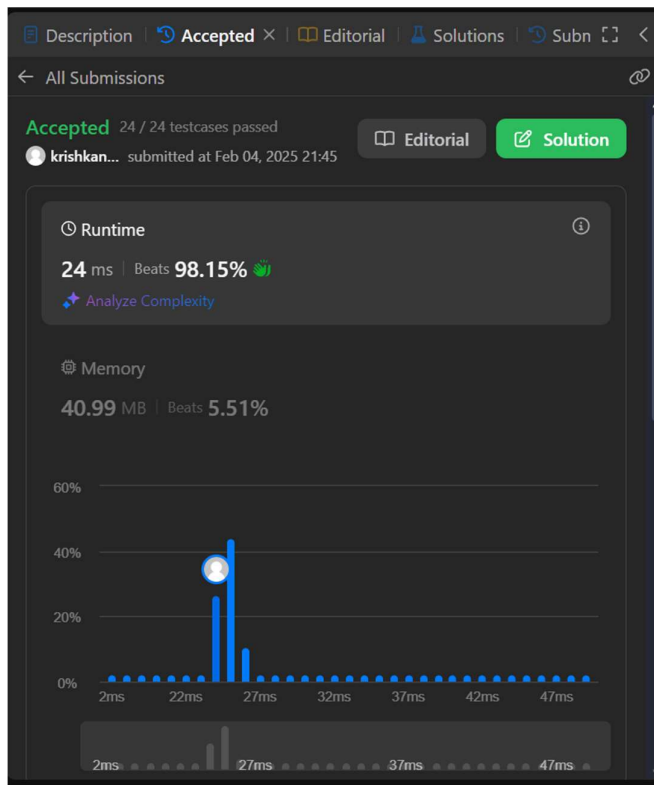
Saved



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 = l + (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.getDefault(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 }
```

