Experiment 2

Student Name: Khushmn Sangha	**UID:** 22BCS14585
Branch: BE-CSE	**Section/Group:** 602/A
Semester: 6th	**Date of Performance:** 4-02-25
Subject Name: Advanced Programming - II	**Subject Code:** 22CSP-351

Aim: To solve the following problems on Leetcode, with the goal of optimizing solutions in terms of time complexity and space efficiency:

(I) **No. of 1 Bits** –
Aim: Given a positive integer `n`, the objective is to return the number of set bits in its binary representation (also known as the Hamming weight). This helps in understanding bit manipulation and the efficient counting of 1-bits.

(II) **Median of Two Sorted Arrays** –
Aim: Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, the objective is to find and return the median of these two sorted arrays with an optimal time complexity of O(log(m+n)), which involves leveraging binary search techniques.

(III) **Sort Colors** –
Aim: Given an array `nums` with `n` objects colored red, white, or blue, the goal is to sort them in-place such that objects of the same color are adjacent, in the order red, white, and blue. The problem focuses on sorting without using library sort functions and utilizing counting or the Dutch National Flag algorithm.

(IV) **Reverse Bits** –
Aim: Given a 32-bit unsigned integer, the task is to reverse its bits. This problem emphasizes bitwise operations and understanding the concept of bit reversal.

(V) **Maximum Subarray** –
Aim: Given an integer array `nums`, the objective is to find the contiguous subarray that has the largest sum, using the Kadane's algorithm to achieve an optimal O(n) time complexity.

(VI) **Search a 2D Matrix II** –
Aim: Given an m x n matrix where each row and column is sorted, the goal is to search for a

target value in the matrix. This problem requires efficient traversal techniques to minimize time complexity, using the "top-right" search approach to find the target in O(m + n) time.

(VII) **First Bad Version** –
Aim: Given a versioning system, the goal is to find the first bad version using a binary search approach, ensuring optimal performance with O(log n) time complexity.

(VII) **First Bad Version** –
Aim: Given a versioning system, the goal is to find the first bad version using a binary search approach, ensuring optimal performance with O(log n) time complexity.

(VIII) **Top K Frequent Elements** –
Aim: Given an integer array nums, the objective is to find the k most frequent elements. This problem aims to utilize a priority queue (max-heap) or bucket sort to efficiently retrieve the top k elements with an optimal time complexity of O(n log k).

(IX) **Merge Sorted Arrays** –
Aim: Given two sorted integer arrays nums1 and nums2, the goal is to merge them into a single sorted array. This problem focuses on merging with an optimal in-place solution by iterating from the end of the arrays, ensuring a time complexity of O(m + n).

(X) ***Reverse Pairs***–
Aim: Count the number of reverse pairs (i, j) where nums[i] > 2 * nums[j] and i < j using an efficient **merge sort** approach.

(XI) ***The SkyLine problem***–
Aim: Find the outline of the buildings' tops (skyline) formed by their start, end, and heights using a **sweep line** algorithm with a **priority queue**.

(XII) **Beautiful Array** –
Aim: Find the outline of the buildings' tops (skyline) formed by their start, end, and heights using a **sweep line** algorithm with a **priority queue**.

(XIII) **Super Pow**–
Aim: Calculate a^b % 1337 efficiently using **modular exponentiation** to handle large exponents in **O(log b)** time.

(XIV) **Longest nice substring** –
Aim: Find the longest substring where every character has both its lowercase and uppercase versions using a **sliding window** approach.

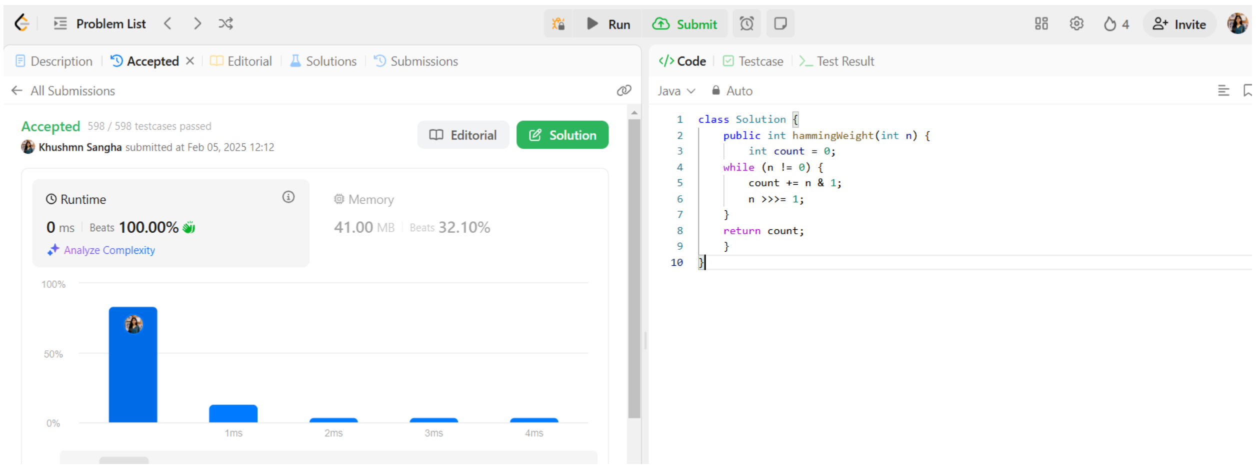
1. **Objective:** To efficiently solve a set of algorithmic problems on Leetcode, optimizing for time and space complexity, by implementing advanced techniques such as bit manipulation,

binary search, sorting algorithms, and dynamic programming, with the goal of enhancing problem-solving skills and achieving optimal solutions for real-world applications.

2. Implementation/Code:

Problem No. 191 : Number of 1 Bits
Code:

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



Accepted 598 / 598 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:12

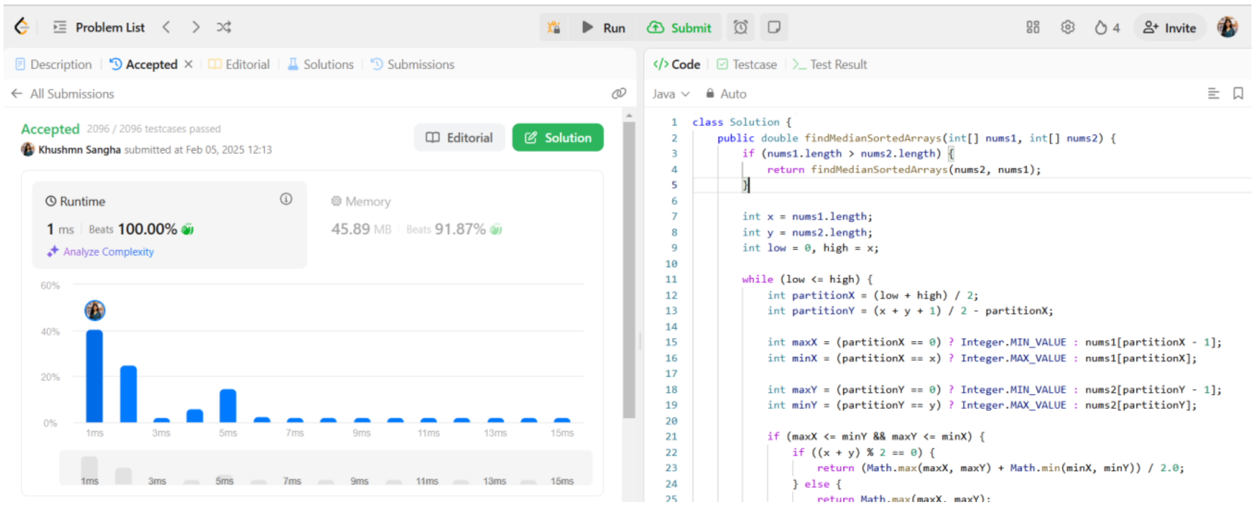
Runtime
0 ms | Beats 100.00%

Memory
41.00 MB | Beats 32.10%

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

Problem No. 4 : Median of Two Sorted Arrays
Code:

```
class Solution {
    public double findMedianSortedArrays(int[] nums1, int[] nums2) {
        if (nums1.length > nums2.length) {
            return findMedianSortedArrays(nums2, nums1);
        }
        int x = nums1.length;
        int y = nums2.length;
        int low = 0, high = x;

        while (low <= high) {
            int partitionX = (low + high) / 2;
            int partitionY = (x + y + 1) / 2 - partitionX;

            int maxX = (partitionX == 0) ? Integer.MIN_VALUE : nums1[partitionX - 1];
            int minX = (partitionX == x) ? Integer.MAX_VALUE : nums1[partitionX];

            int maxY = (partitionY == 0) ? Integer.MIN_VALUE : nums2[partitionY - 1];
            int minY = (partitionY == y) ? Integer.MAX_VALUE : nums2[partitionY];

            if (maxX <= minY && maxY <= minX) {
                if ((x + y) % 2 == 0) {
                    return (Math.max(maxX, maxY) + Math.min(minX, minY)) / 2.0;
                } else {
                    return Math.max(maxX, maxY);
                }
            } else if (maxX > minY) {
                high = partitionX - 1;
            } else {
                low = partitionX + 1;
            }
        }

        throw new IllegalArgumentException("Input arrays are not sorted.");
    }
}
```



Accepted 2096 / 2096 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:13

Runtime: 1 ms | Beats 100.00%
Memory: 45.89 MB | Beats 91.87%

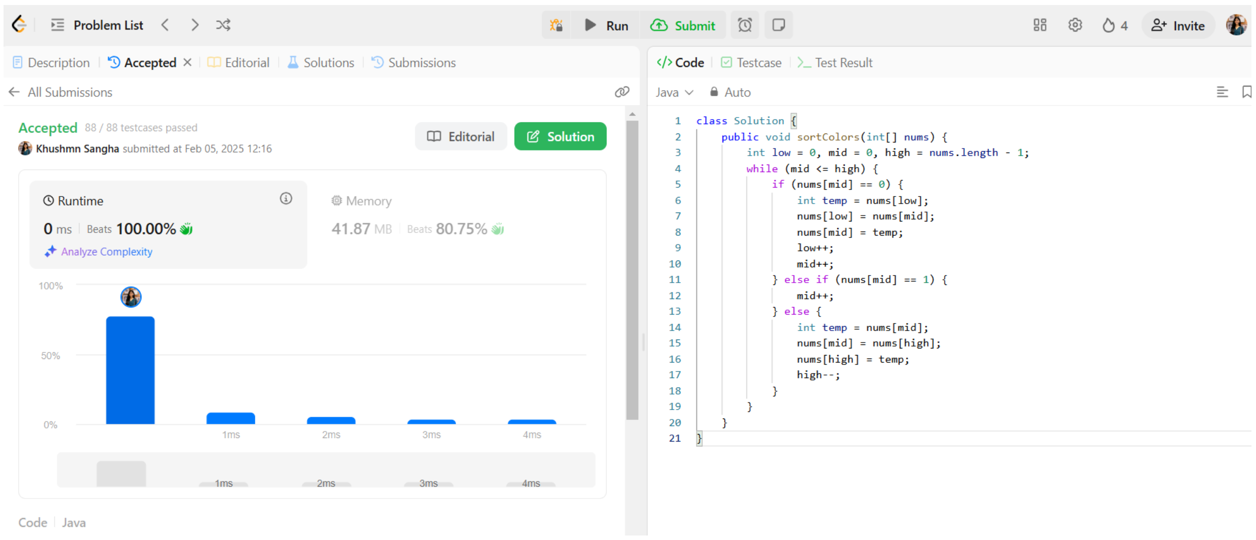
```java
class Solution {
    public double findMedianSortedArrays(int[] nums1, int[] nums2) {
        if (nums1.length > nums2.length) {
            return findMedianSortedArrays(nums2, nums1);
        }

        int x = nums1.length;
        int y = nums2.length;
        int low = 0, high = x;

        while (low <= high) {
            int partitionX = (low + high) / 2;
            int partitionY = (x + y + 1) / 2 - partitionX;

            int maxX = (partitionX == 0) ? Integer.MIN_VALUE : nums1[partitionX - 1];
            int minX = (partitionX == x) ? Integer.MAX_VALUE : nums1[partitionX];

            int maxY = (partitionY == 0) ? Integer.MIN_VALUE : nums2[partitionY - 1];
            int minY = (partitionY == y) ? Integer.MAX_VALUE : nums2[partitionY];

            if (maxX <= minY && maxY <= minX) {
                if ((x + y) % 2 == 0) {
                    return (Math.max(maxX, maxY) + Math.min(minX, minY)) / 2.0;
                } else {
                    return Math.max(maxX, maxY);
```

Problem No. 75 : Sort Colors

Code:

```java
class Solution {
    public void sortColors(int[] nums) {
        int low = 0, mid = 0, high = nums.length - 1;
        while (mid <= high) {
            if (nums[mid] == 0) {
                int temp = nums[low];
                nums[low] = nums[mid];
                nums[mid] = temp;
                low++;
                mid++;
            } else if (nums[mid] == 1) {
                mid++;
            } else {
                int temp = nums[mid];
                nums[mid] = nums[high];
                nums[high] = temp;
                high--;
            }
        }
    }
```

}



Accepted 88 / 88 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:16

Runtime
0 ms | Beats 100.00%

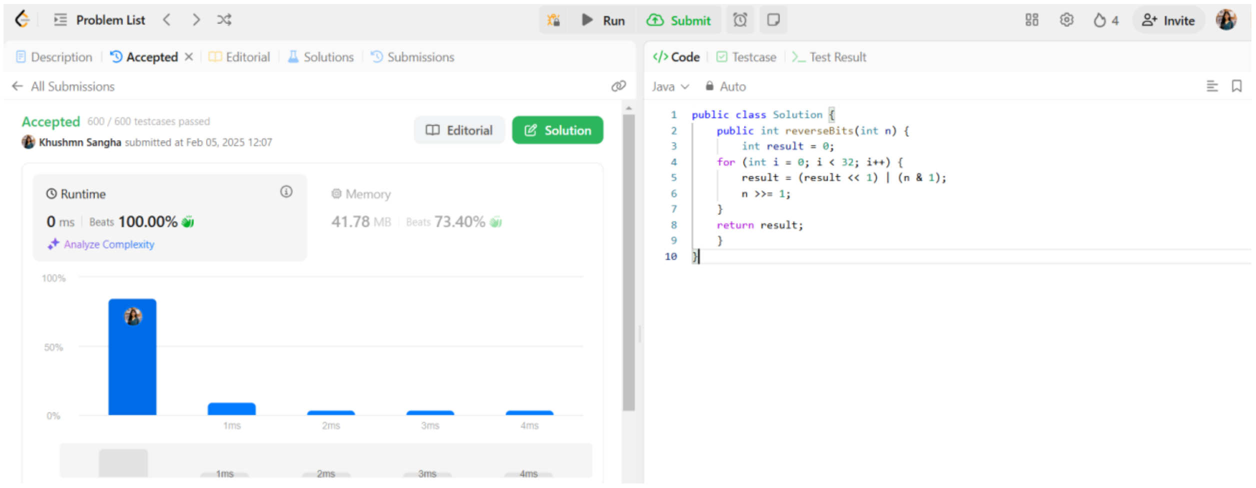
Memory
41.87 MB | Beats 80.75%

```java
class Solution {
    public void sortColors(int[] nums) {
        int low = 0, mid = 0, high = nums.length - 1;
        while (mid <= high) {
            if (nums[mid] == 0) {
                int temp = nums[low];
                nums[low] = nums[mid];
                nums[mid] = temp;
                low++;
                mid++;
            } else if (nums[mid] == 1) {
                mid++;
            } else {
                int temp = nums[mid];
                nums[mid] = nums[high];
                nums[high] = temp;
                high--;
            }
        }
    }
}
```

Problem No. 190 : Reverse Bits

Code:

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



Accepted 600 / 600 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:07

Runtime
0 ms | Beats 100.00%

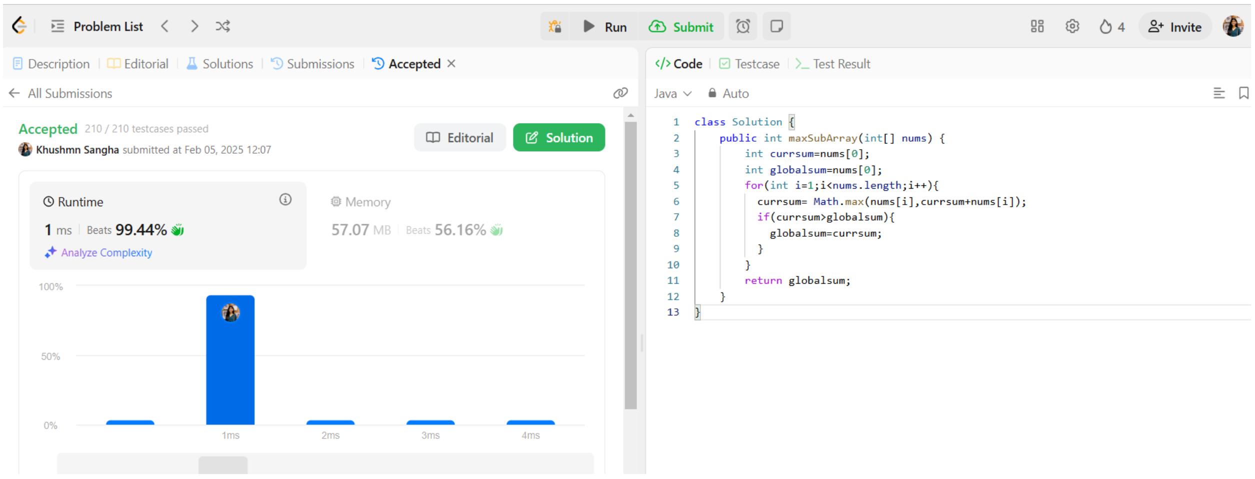
Memory
41.78 MB | Beats 73.40%

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

Problem No. 53 : Maximum Subarray -

Code:

```
class Solution {
    public int maxSubArray(int[] nums) {
        int currsum=nums[0];
        int globalsum=nums[0];
        for(int i=1;i<nums.length;i++){
          currsum= Math.max(nums[i],currsum+nums[i]);
          if(currsum>globalsum){
            globalsum=currsum;
          }
        }
        return globalsum;
    }
}
```



Accepted 210 / 210 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:07

Runtime
1 ms | Beats 99.44%
Analyze Complexity

Memory
57.07 MB | Beats 56.16%

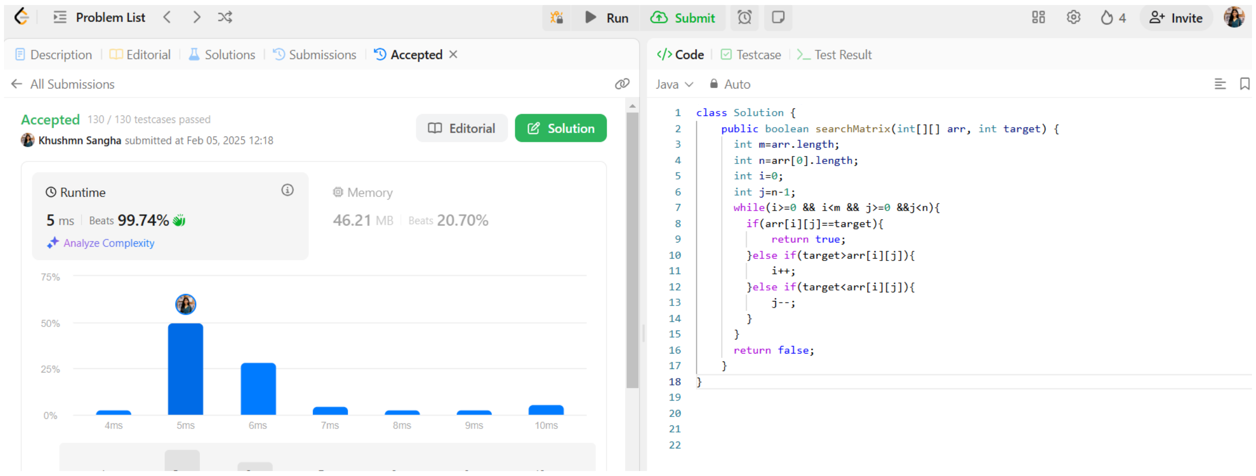
```java
class Solution {
    public int maxSubArray(int[] nums) {
        int currsum=nums[0];
        int globalsum=nums[0];
        for(int i=1;i<nums.length;i++){
          currsum= Math.max(nums[i],currsum+nums[i]);
          if(currsum>globalsum){
            globalsum=currsum;
          }
        }
        return globalsum;
    }
}
```

Problem No. 240 : Search a 2D Matrix –

Code:

```
class Solution {
    public boolean searchMatrix(int[][] arr, int target) {
      int m=arr.length;
      int n=arr[0].length;
      int i=0;
      int j=n-1;
      while(i>=0 && i<m && j>=0 &&j<n){
        if(arr[i][j]==target){
            return true;
        }else if(target>arr[i][j]){
            i++;
        }else if(target<arr[i][j]){
            j--;
        }
      }
```

```
        return false;
    }
}
```



Accepted 130 / 130 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:18

Runtime
5 ms Beats 99.74%
Analyze Complexity

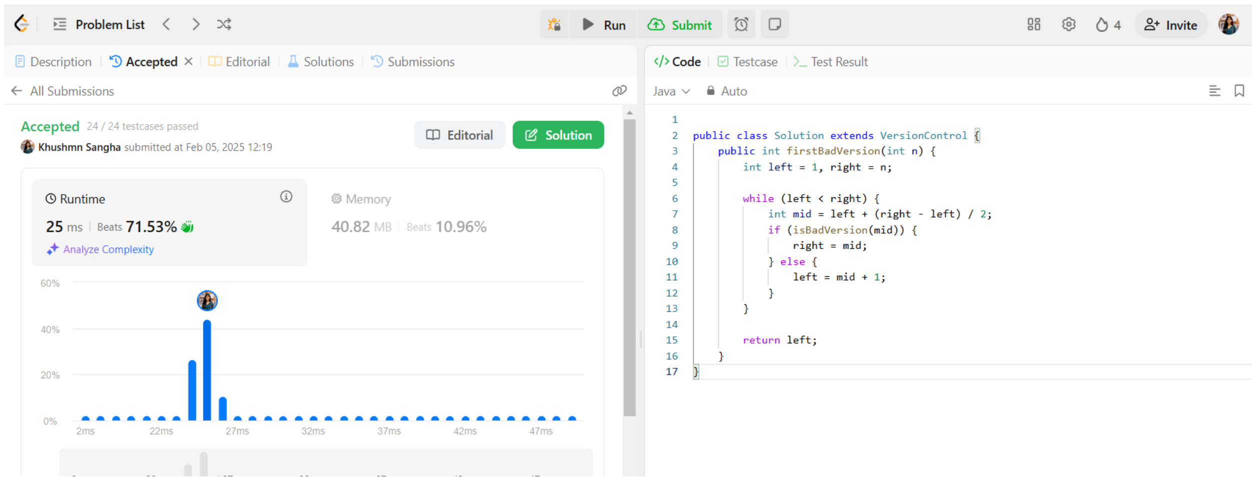
Memory
46.21 MB Beats 20.70%

```java
class Solution {
    public boolean searchMatrix(int[][] arr, int target) {
      int m=arr.length;
      int n=arr[0].length;
      int i=0;
      int j=n-1;
      while(i>=0 && i<m && j>=0 &&j<n){
        if(arr[i][j]==target){
            return true;
        }else if(target>arr[i][j]){
            i++;
        }else if(target<arr[i][j]){
            j--;
        }
      }
      return false;
    }
}
```

Problem No. 278 First Bad Version

Code:

```java
public class Solution extends VersionControl {
    public int firstBadVersion(int n) {
        int left = 1, right = n;

        while (left < right) {
            int mid = left + (right - left) / 2;
            if (isBadVersion(mid)) {
                right = mid;
            } else {
                left = mid + 1;
            }
        }

        return left;
    }
```



Accepted 24 / 24 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:19

Runtime
25 ms | Beats 71.53%

Memory
40.82 MB | Beats 10.96%

Java Auto

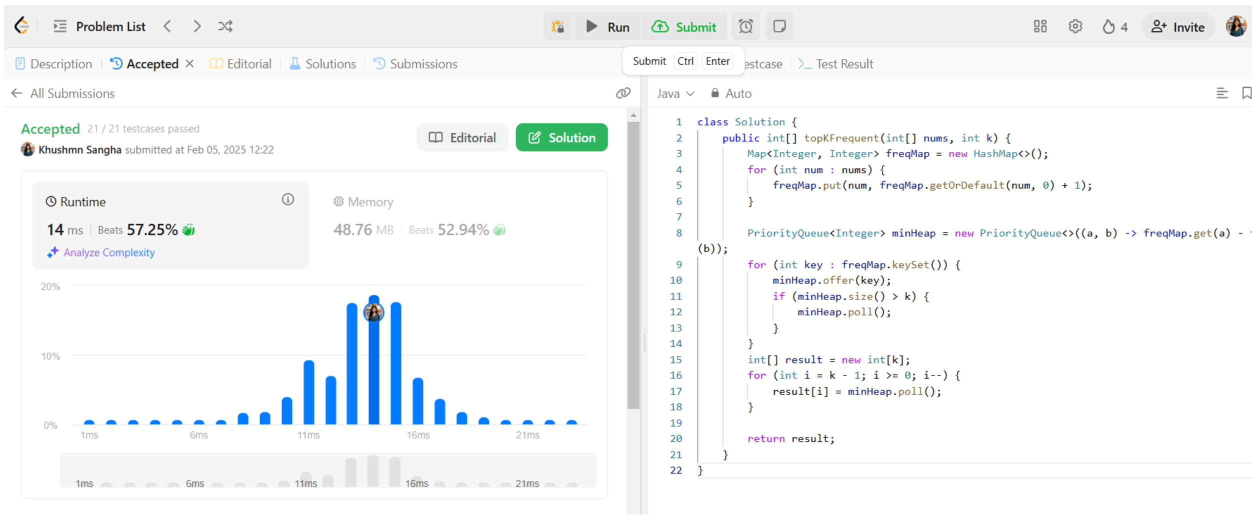
```java

public class Solution extends VersionControl {
    public int firstBadVersion(int n) {
        int left = 1, right = n;

        while (left < right) {
            int mid = left + (right - left) / 2;
            if (isBadVersion(mid)) {
                right = mid;
            } else {
                left = mid + 1;
            }
        }

        return left;
    }
}
```

Prolem No. 347: Top K Frequent Elements –

Code:

```
class Solution {
    public int[] topKFrequent(int[] nums, int k) {
        Map<Integer, Integer> freqMap = new HashMap<>();
        for (int num : nums) {
            freqMap.put(num, freqMap.getOrDefault(num, 0) + 1);
        }

        PriorityQueue<Integer> minHeap = new PriorityQueue<>((a, b) -> freqMap.get(a) -
freqMap.get(b));
```

```
        for (int key : freqMap.keySet()) {
            minHeap.offer(key);
            if (minHeap.size() > k) {
                minHeap.poll();
            }
        }
        int[] result = new int[k];
        for (int i = k - 1; i >= 0; i--) {
            result[i] = minHeap.poll();
        }

        return result;
    }
}
```



Accepted 21 / 21 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:22

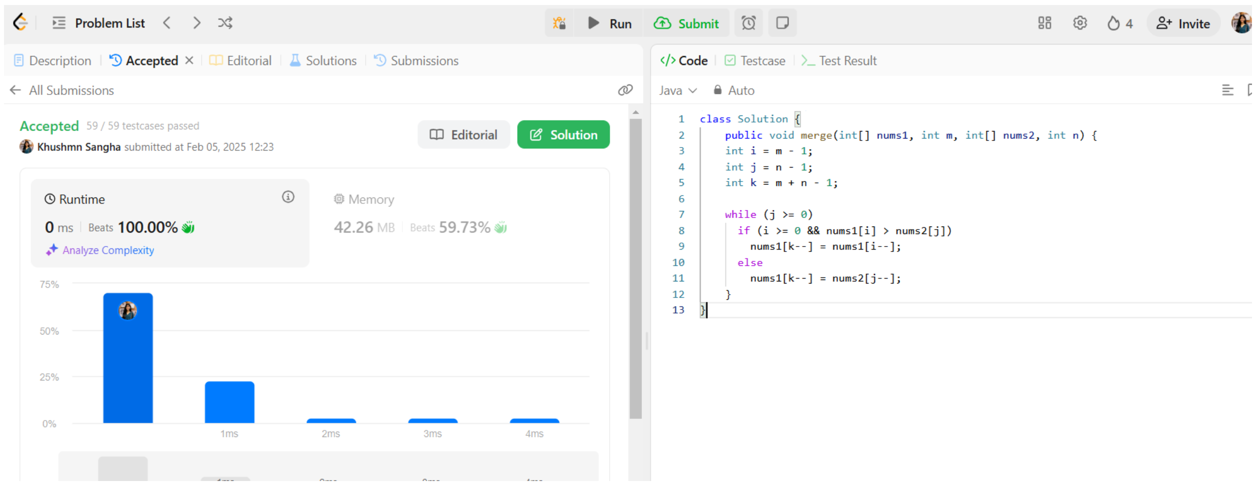
Runtime 14 ms | Beats 57.25%
Memory 48.76 MB | Beats 52.94%

```
class Solution {
    public int[] topKFrequent(int[] nums, int k) {
        Map<Integer, Integer> freqMap = new HashMap<>();
        for (int num : nums) {
            freqMap.put(num, freqMap.getOrDefault(num, 0) + 1);
        }

        PriorityQueue<Integer> minHeap = new PriorityQueue<>((a, b) -> freqMap.get(a) - 
(b));

        for (int key : freqMap.keySet()) {
            minHeap.offer(key);
            if (minHeap.size() > k) {
                minHeap.poll();
            }
        }
        int[] result = new int[k];
        for (int i = k - 1; i >= 0; i--) {
            result[i] = minHeap.poll();
        }

        return result;
    }
}
```

Problem No. 88 : Merge Sorted Arrays

Code:

```
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
    int i = m - 1;
    int j = n - 1;
    int k = m + n - 1;

    while (j >= 0)
      if (i >= 0 && nums1[i] > nums2[j])
        nums1[k--] = nums1[i--];
      else
        nums1[k--] = nums2[j--];
    }
}
```



Accepted 59 / 59 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:23

Runtime
0 ms | Beats 100.00%

Memory
42.26 MB | Beats 59.73%

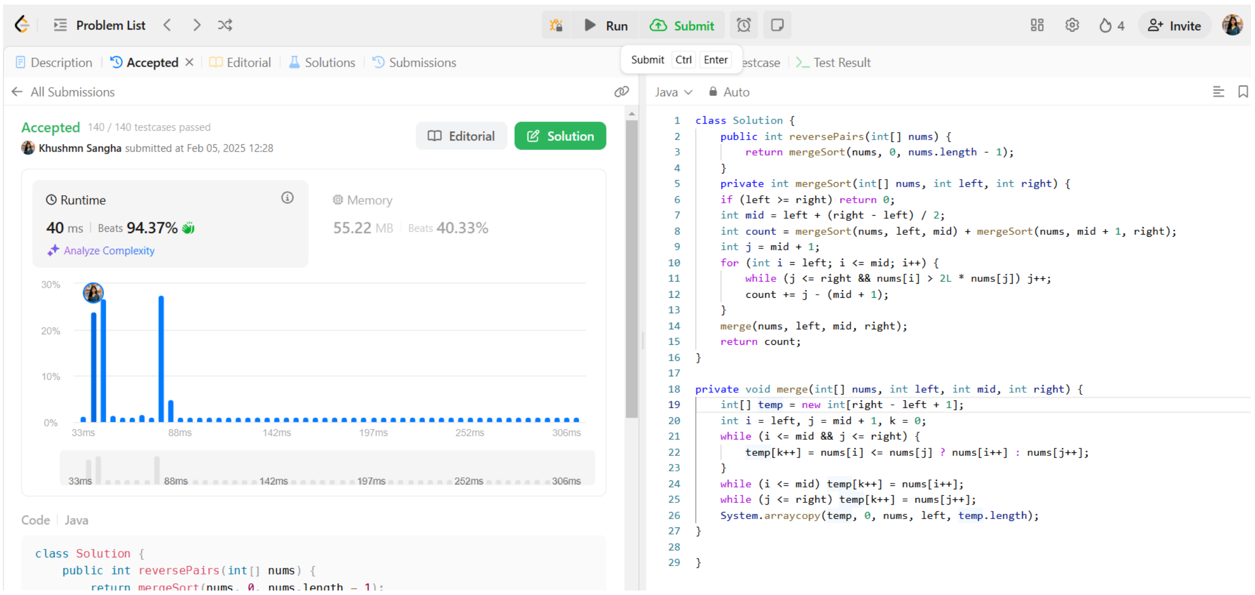
```
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
    int i = m - 1;
    int j = n - 1;
    int k = m + n - 1;

    while (j >= 0)
      if (i >= 0 && nums1[i] > nums2[j])
        nums1[k--] = nums1[i--];
      else
        nums1[k--] = nums2[j--];
    }
}
```

Problem No. 493: Reverse Pairs

Code:

```
class Solution {
    public int reversePairs(int[] nums) {
        return mergeSort(nums, 0, nums.length - 1);
    }
    private int mergeSort(int[] nums, int left, int right) {
    if (left >= right) return 0;
    int mid = left + (right - left) / 2;
    int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1, right);
    int j = mid + 1;
    for (int i = left; i <= mid; i++) {
        while (j <= right && nums[i] > 2L * nums[j]) j++;
        count += j - (mid + 1);
    }
    merge(nums, left, mid, right);
    return count;
}

private void merge(int[] nums, int left, int mid, int right) {
    int[] temp = new int[right - left + 1];
    int i = left, j = mid + 1, k = 0;
    while (i <= mid && j <= right) {
        temp[k++] = nums[i] <= nums[j] ? nums[i++] : nums[j++];
    }
    while (i <= mid) temp[k++] = nums[i++];
    while (j <= right) temp[k++] = nums[j++];
    System.arraycopy(temp, 0, nums, left, temp.length);
}
```

}



Accepted 140 / 140 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:28

Runtime 40 ms | Beats 94.37%
Memory 55.22 MB | Beats 40.33%

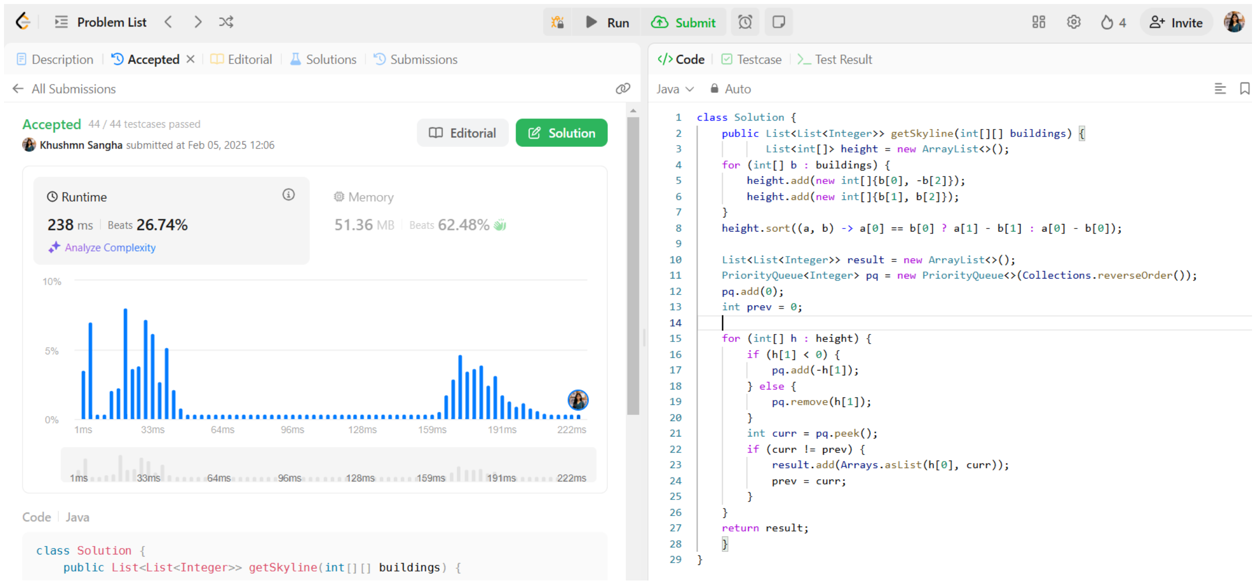
```java
class Solution {
    public int reversePairs(int[] nums) {
        return mergeSort(nums, 0, nums.length - 1);
    }
    private int mergeSort(int[] nums, int left, int right) {
    if (left >= right) return 0;
    int mid = left + (right - left) / 2;
    int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1, right);
    int j = mid + 1;
    for (int i = left; i <= mid; i++) {
        while (j <= right && nums[i] > 2L * nums[j]) j++;
        count += j - (mid + 1);
    }
    merge(nums, left, mid, right);
    return count;
}

private void merge(int[] nums, int left, int mid, int right) {
    int[] temp = new int[right - left + 1];
    int i = left, j = mid + 1, k = 0;
    while (i <= mid && j <= right) {
        temp[k++] = nums[i] <= nums[j] ? nums[i++] : nums[j++];
    }
    while (i <= mid) temp[k++] = nums[i++];
    while (j <= right) temp[k++] = nums[j++];
    System.arraycopy(temp, 0, nums, left, temp.length);
}

}
```

Problem No. 218: The Skyline Problem

Code:

```java
class Solution {
   public List<List<Integer>> getSkyline(int[][] buildings) {
        List<int[]> height = new ArrayList<>();
   for (int[] b : buildings) {
      height.add(new int[]{b[0], -b[2]});
      height.add(new int[]{b[1], b[2]});
   }
   height.sort((a, b) -> a[0] == b[0] ? a[1] - b[1] : a[0] - b[0]);

   List<List<Integer>> result = new ArrayList<>();
   PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
```

```
        pq.add(0);
        int prev = 0;

        for (int[] h : height) {
            if (h[1] < 0) {
                pq.add(-h[1]);
            } else {
                pq.remove(h[1]);
            }
            int curr = pq.peek();
            if (curr != prev) {
                result.add(Arrays.asList(h[0], curr));
                prev = curr;
            }
        }
        return result;
        }
    }
```



Accepted 44 / 44 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:06

Runtime
238 ms | Beats 26.74%
Analyze Complexity

Memory
51.36 MB | Beats 62.48%

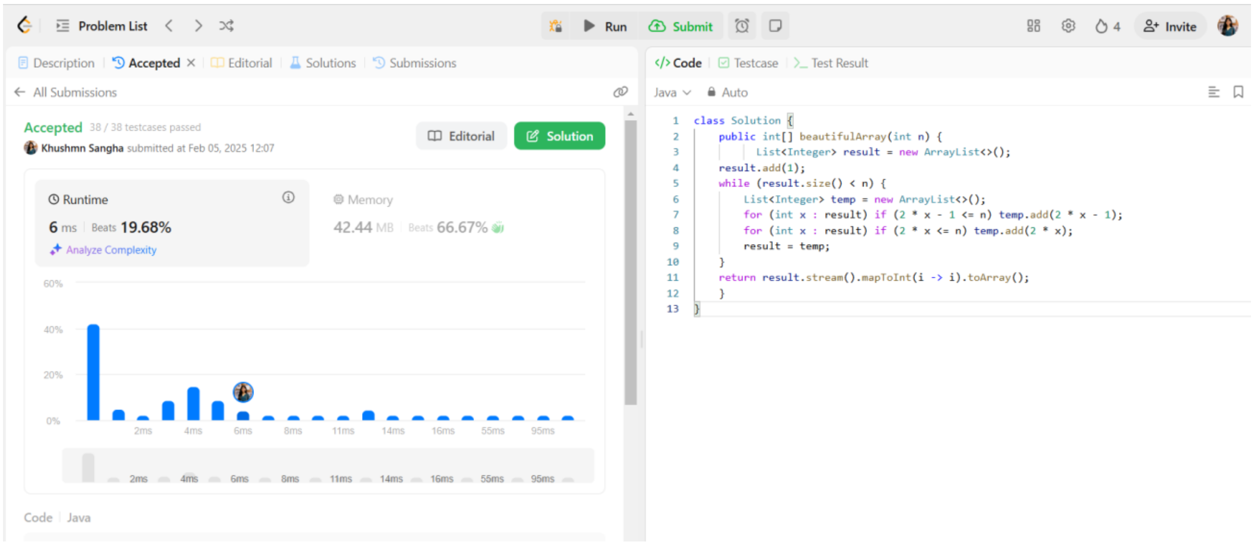
Code | Java

```java
class Solution {
    public List<List<Integer>> getSkyline(int[][] buildings) {
            List<int[]> height = new ArrayList<>();
    for (int[] b : buildings) {
        height.add(new int[]{b[0], -b[2]});
        height.add(new int[]{b[1], b[2]});
    }
    height.sort((a, b) -> a[0] == b[0] ? a[1] - b[1] : a[0] - b[0]);

    List<List<Integer>> result = new ArrayList<>();
    PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
    pq.add(0);
    int prev = 0;

    for (int[] h : height) {
        if (h[1] < 0) {
            pq.add(-h[1]);
        } else {
            pq.remove(h[1]);
        }
        int curr = pq.peek();
        if (curr != prev) {
            result.add(Arrays.asList(h[0], curr));
            prev = curr;
        }
    }
    return result;
    }
}
```

Problem No. 932: [Beautiful Array]

Code:

```java
class Solution {
    public int[] beautifulArray(int n) {
        List<Integer> result = new ArrayList<>();
    result.add(1);
    while (result.size() < n) {
        List<Integer> temp = new ArrayList<>();
        for (int x : result) if (2 * x - 1 <= n) temp.add(2 * x - 1);
        for (int x : result) if (2 * x <= n) temp.add(2 * x);
        result = temp;
    }
    return result.stream().mapToInt(i -> i).toArray();
    }
}
```


Accepted 38 / 38 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:07

Runtime
6 ms | Beats 19.68%

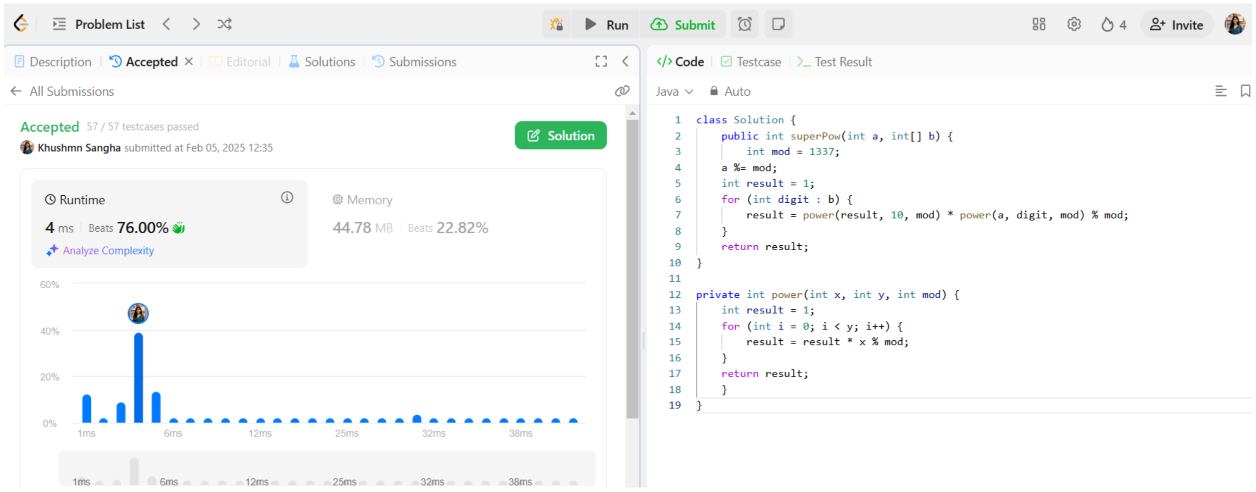
Memory
42.44 MB | Beats 66.67%

```java
class Solution {
    public int[] beautifulArray(int n) {
            List<Integer> result = new ArrayList<>();
    result.add(1);
    while (result.size() < n) {
        List<Integer> temp = new ArrayList<>();
        for (int x : result) if (2 * x - 1 <= n) temp.add(2 * x - 1);
        for (int x : result) if (2 * x <= n) temp.add(2 * x);
        result = temp;
    }
    return result.stream().mapToInt(i -> i).toArray();
    }
}
```

Problem No. 372:* *[Super Pow]()

Code:

```
class Solution {
    public int superPow(int a, int[] b) {
        int mod = 1337;
    a %= mod;
    int result = 1;
    for (int digit : b) {
        result = power(result, 10, mod) * power(a, digit, mod) % mod;
    }
    return result;
}

private int power(int x, int y, int mod) {
    int result = 1;
    for (int i = 0; i < y; i++) {
        result = result * x % mod;
    }
    return result;
    }
}
```



Accepted 57 / 57 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:35

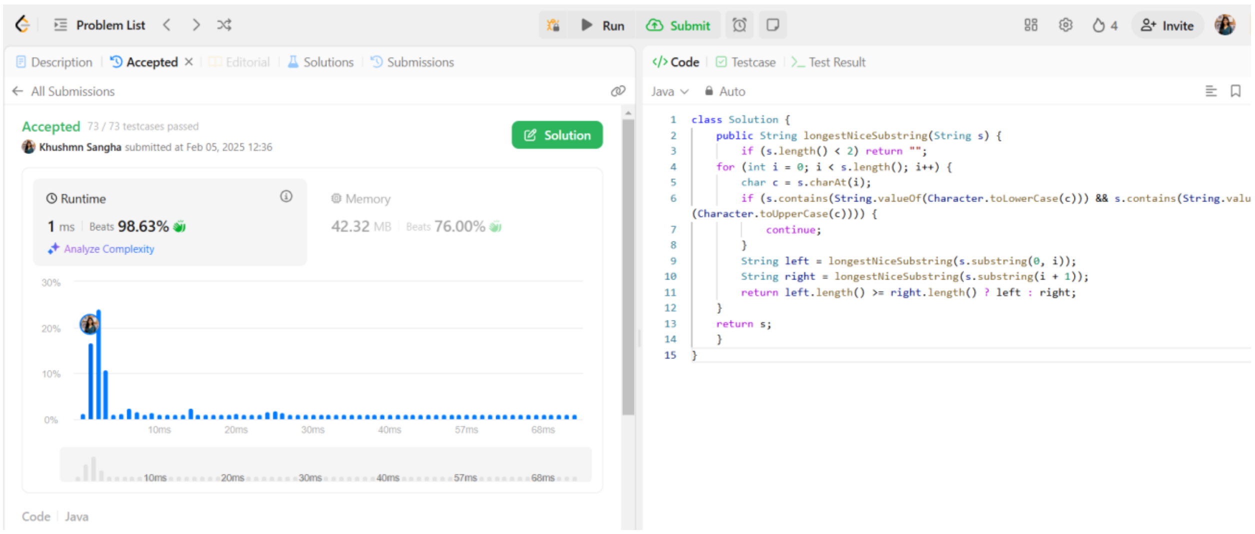
Runtime 4 ms | Beats 76.00%
Memory 44.78 MB | Beats 22.82%

```java
class Solution {
    public int superPow(int a, int[] b) {
        int mod = 1337;
    a %= mod;
    int result = 1;
    for (int digit : b) {
        result = power(result, 10, mod) * power(a, digit, mod) % mod;
    }
    return result;
}

private int power(int x, int y, int mod) {
    int result = 1;
    for (int i = 0; i < y; i++) {
        result = result * x % mod;
    }
    return result;
    }
}
```

Problem No. 1763: [Longest Nice Substring](#)

Code:

```
class Solution {
    public String longestNiceSubstring(String s) {
        if (s.length() < 2) return "";
    for (int i = 0; i < s.length(); i++) {
        char c = s.charAt(i);
        if (s.contains(String.valueOf(Character.toLowerCase(c))) &&
s.contains(String.valueOf(Character.toUpperCase(c)))) {
            continue;
        }
        String left = longestNiceSubstring(s.substring(0, i));
        String right = longestNiceSubstring(s.substring(i + 1));
        return left.length() >= right.length() ? left : right;
    }
    return s;
    }
}
```



Accepted 73 / 73 testcases passed
Khushmn Sangha submitted at Feb 05, 2025 12:36

Runtime
1 ms | Beats 98.63%

Memory
42.32 MB | Beats 76.00%

```
class Solution {
    public String longestNiceSubstring(String s) {
        if (s.length() < 2) return "";
    for (int i = 0; i < s.length(); i++) {
        char c = s.charAt(i);
        if (s.contains(String.valueOf(Character.toLowerCase(c))) && s.contains(String.valu
(Character.toUpperCase(c)))) {
            continue;
        }
        String left = longestNiceSubstring(s.substring(0, i));
        String right = longestNiceSubstring(s.substring(i + 1));
        return left.length() >= right.length() ? left : right;
    }
    return s;
    }
}
```