

AP Assignment 4

1. Longest Nice Substring: <https://leetcode.com/problems/longest-nice-substring/description/>

The screenshot shows the LeetCode interface for the 'Longest Nice Substring' problem. The submission is accepted, with a runtime of 2 ms and memory usage of 42.97 MB. The code is written in Java and uses a HashSet to track characters. The test result shows the input string 'YazaAay'.

```
1 class Solution {
2     public String longestNiceSubstring(String s) {
3         if(s.length() < 2) return "";
4
5         char[] arr = s.toCharArray();
6
7         HashSet<Character> set = new HashSet<>();
8
9         for(char c : arr) {
```

Runtime: 2 ms | Beats 81.07%
Memory: 42.97 MB | Beats 40.75%

Accepted | Runtime: 0 ms
Case 1: Input: s = "YazaAay"

2. Reverse Bits: <https://leetcode.com/problems/reverse-bits/description/>

The screenshot shows the LeetCode interface for the 'Reverse Bits' problem. The submission is accepted, with a runtime of 1 ms and memory usage of 41.78 MB. The code is written in Java and uses a loop to reverse the bits of the input integer. The test result shows the input integer 00000010100101000001111010011100.

```
1 public class Solution {
2     // you need treat n as an unsigned value
3     public int reverseBits(int n) {
4         int ans = 0;
5
6         for (int i = 0; i < 32; i++) { // why 32, for reverse 32 bits
7             ans = (ans << 1) | (n & 1); // left shift by 1 and find lsb and set it
8             n = n >> 1; // delete last lsb
9         }
```

Runtime: 1 ms | Beats 15.75%
Memory: 41.78 MB | Beats 74.02%

Accepted | Runtime: 0 ms
Case 1: Input: n = 00000010100101000001111010011100

3. Number of 1 Bits: <https://leetcode.com/problems/number-of-1-bits/description/>

AP Assignment 4

This screenshot shows a LeetCode submission for the problem "Number of 1-bits" (LeetCode 231). The submission is by user AayushBatra, submitted on Mar 17, 2025 at 13:22. The status is "Accepted" with 598 / 598 testcases passed. The runtime is 0 ms, beating 100.00% of submissions. The memory usage is 41.20 MB, beating 13.36% of submissions. The code is written in Java and implements a function to calculate the number of 1-bits in an integer using a while loop. The test result shows the submission is accepted with a runtime of 0 ms.

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

4. Maximum Subarray: <https://leetcode.com/problems/maximum-subarray/description/>

This screenshot shows a LeetCode submission for the problem "Maximum Subarray" (LeetCode 53). The submission is by user AayushBatra, submitted on Mar 17, 2025 at 13:28. The status is "Accepted" with 210 / 210 testcases passed. The runtime is 1 ms, beating 99.51% of submissions. The memory usage is 56.89 MB, beating 81.08% of submissions. The code is written in Java and implements a function to find the maximum subarray sum using Kadane's algorithm. The test result shows the submission is accepted with a runtime of 0 ms. The input for the test case is an array of numbers: [-2, 1, -3, 4, -1, 2, 1, -5, 4].

```
class Solution {  
    public int maxSubArray(int[] nums) {  
        int maxSum = Integer.MIN_VALUE;  
        int currentSum = 0;  
        for (int i = 0; i < nums.length; i++) {  
            currentSum += nums[i];  
            // (currentSum > maxSum) ? maxSum = currentSum : maxSum;  
        }  
        return maxSum;  
    }  
}
```

5. Search a 2D Matrix II: <https://leetcode.com/problems/search-a-2d-matrix-ii/description/>

AP Assignment 4

This screenshot shows a LeetCode submission for the problem "Search a 2D Matrix II". The submission is by user "AayushBatra" and is marked as "Accepted". The runtime is 5 ms, which beats 99.66% of other submissions. The memory usage is 46.12 MB, which beats 38.16%. The code is written in Java and implements a search algorithm for a target in a 2D matrix. The test result shows that the solution passed all test cases.

```
public class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        if(matrix == null || matrix.length < 1 || matrix[0].length < 1) {
            return false;
        }
        int col = matrix[0].length-1;
        int row = 0;
        while(col >= 0 && row <= matrix.length-1) {
            if(target == matrix[row][col]) {
                return true;
            } else if(target < matrix[row][col]) {
                col--;
            } else {
                row++;
            }
        }
        return false;
    }
}
```

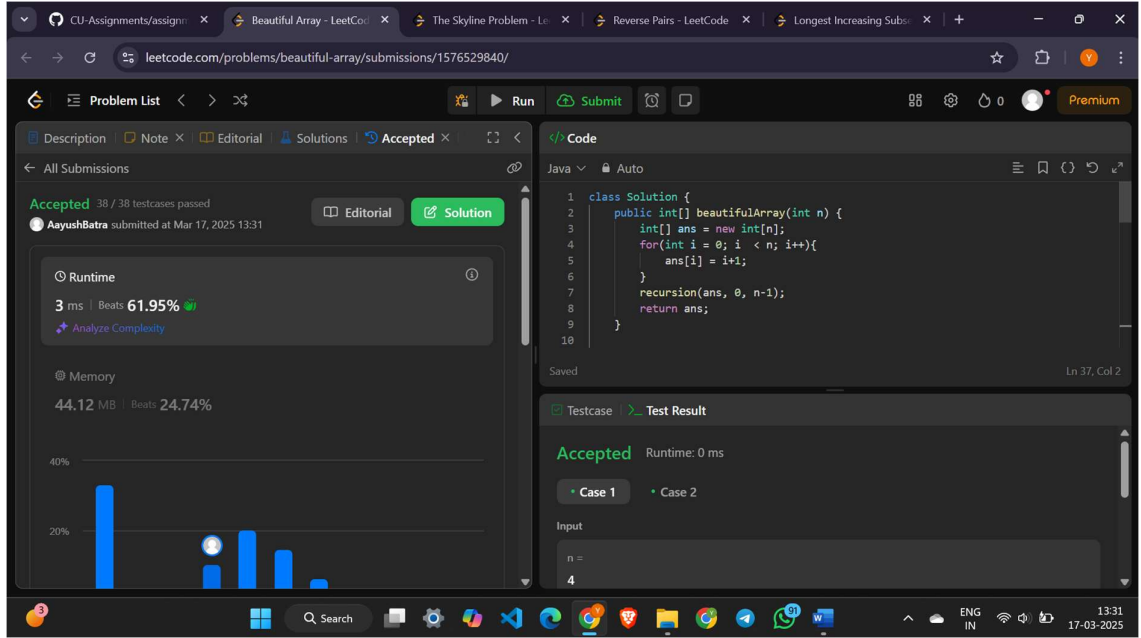
6. Super Pow: <https://leetcode.com/problems/super-pow/description/>

This screenshot shows a LeetCode submission for the problem "Super Pow". The submission is by user "AayushBatra" and is marked as "Accepted". The runtime is 3 ms, which beats 84.06% of other submissions. The memory usage is 44.32 MB, which beats 58.89%. The code is written in Java and implements a recursive solution to calculate the super power of a number. The test result shows that the solution passed all test cases.

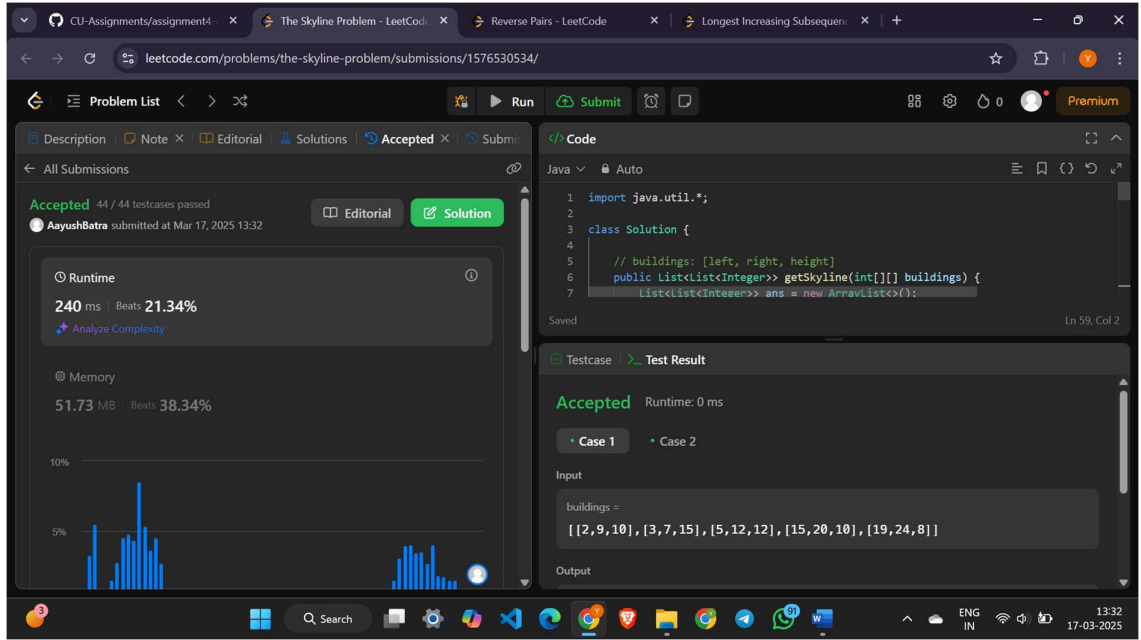
```
class Solution {
    private final int MOD = 1337;
    public int superPow(int a, int[] b) {
        return helper(a, b);
    }
    private int helper(int a, int[] b) {
        if(b.length == 0) return 1;
        int last = b[b.length-1];
        int[] newB = new int[b.length-1];
        for(int i = 0; i < b.length-1; i++) {
            newB[i] = b[i];
        }
        return (int) Math.pow(a, last) * helper(a, newB) % MOD;
    }
}
```

7. Beautiful Array: <https://leetcode.com/problems/beautiful-array/description/>

AP Assignment 4



8. The Skyline Problem: <https://leetcode.com/problems/the-skyline-problem/description/>



9. Reverse Pairs: <https://leetcode.com/problems/reverse-pairs/description/>

AP Assignment 4

This screenshot shows a LeetCode submission for the 'Reverse Pairs' problem. The submission is accepted, with 140 out of 140 test cases passed. The runtime is 476 ms, which is 5.02% faster than the fastest solution. The memory usage is 56.02 MB, which is 5.09% less than the least memory solution. The code is written in Java and uses a recursive approach to count the number of reverse pairs in an array. The test case input is [1, 3, 2, 3, 1], and the output is 4.

```
class Solution {
    public int reversePairs(int[] nums) {
        int ans = 0;
        List<Long> res = new ArrayList<>();
        res.add((long) nums[nums.length - 1] * 2);
        for (int i = nums.length - 2; i >= 0; i--) {
            ans += LessThan(res, nums[i]);
            update(res, (long) nums[i] * 2);
        }
        return ans;
    }
}
```

10. Longest Increasing Subsequence II: <https://leetcode.com/problems/longest-increasing-subsequence-ii/description/>

This screenshot shows a LeetCode submission for the 'Longest Increasing Subsequence II' problem. The submission is accepted, with 84 out of 84 test cases passed. The runtime is 29 ms, which is 85.59% faster than the fastest solution. The memory usage is 55.45 MB, which is 94.92% less than the least memory solution. The code is written in Java and uses a dynamic programming approach to find the longest increasing subsequence. The test case input is [4, 2, 1, 4, 3, 4, 5, 8, 15], and the output is 6.

```
class Solution {
    public int longestIncreasingSubsequence(int[] nums) {
        int N = 100001;
        int[] seg = new int[2*N];
        void update(int pos, int val) { // update max
            pos += N;
            seg[pos] = val;
        }
        // ... (rest of the code)
    }
}
```