

Name : Rakshit Uniyal

Uid: 22BCS12396

607 – B

AP assignment 4

1. (1763) Longest Nice Substring

1763. Longest Nice Substring Solved

A string s is **nice** if, for every letter of the alphabet that s contains, it appears **both** in uppercase and lowercase. For example, "abAB" is nice because 'A' and 'a' appear, and 'B' and 'b' appear. However, "abA" is not because 'b' appears, but 'B' does not.

Given a string s , return the **longest substring** of s that is **nice**. If there are multiple, return the substring of the **earliest** occurrence. If there are none, return an empty string.

Example 1:

Input: $s = \text{"YazaAay"}$

Output: "aA"

Submissions

Status	Language	Runtime	Memory	Notes
Accepted	C++	6 ms	14.3 MB	
Accepted	C++	0 ms	9.7 MB	

```
1 class Solution {
2 public:
3     string longestNiceSubstring(string s) {
4         if(s.length() < 2){
5             return "";
6         }
7         unordered_set<char> seen(s.begin(), s.end());
8         for(int i = 0; i < s.length(); i++){
9             char ch = s[i];
10            if(seen.count(tolower(ch)) && seen.count(toupper(ch))){
11                continue;
12            }
13            string left = longestNiceSubstring(s.substr(0, i));
14            string right = longestNiceSubstring(s.substr(i+1));
15            return left.length() >= right.length() ? left : right;
16        }
17    }
18 }
```

Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input: $s = \text{"YazaAay"}$

Output: "aA"

2. (190) Reverse Bits

190. Reverse Bits Solved

Reverse bits of a given 32 bits unsigned integer.

Note:

- Note that in some languages, such as Java, there is no unsigned integer type. In this case, both input and output will be given as a signed integer type. They should not affect your implementation, as the integer's internal binary representation is the same, whether it is signed or unsigned.
- In Java, the compiler represents the signed integers using 2's complement notation. Therefore, in Example 2 above, the input represents the signed integer -3 and the output represents the signed integer -1073741825.

Submissions

Status	Language	Runtime	Memory	Notes
Accepted	C++	0 ms	7.7 MB	

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

Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input: $n = 000000101001010000011110011001100$

Output: "10011001100110000010100101000000"

3. (191) Number of 1 Bits

191. Number of 1 Bits Solved

Easy Topics Companies

Given a positive integer n , write a function that returns the number of **set bits** in its binary representation (also known as the **Hamming weight**).

Example 1:
Input: $n = 11$
Output: 3

6.8K 177 59 Online

Submissions

Status	Language	Runtime	Memory	Notes
Accepted a minute ago	C++	0 ms	8.3 MB	

Code

```
1 class Solution {
2 public:
3     int hammingWeight(uint32_t n) {
4         int count = 0;
5         while (n) {
6             count += n & 1;
7             n >>= 1;
8         }
9         return count;
10    }
11 };
```

Ln 6, Col 29 Saved

Testcase **Test Result**

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input: $n = 11$

Output:

Right sidebar: Rakshit Uniyal, My Lists, Notebook, Submissions, Progress, Points, Try New Features, Orders, My Playgrounds, Settings, Classic Mode, Appearance, Sign Out, Submit

4. (53) Maximum Subarray

53. Maximum Subarray Solved

Medium Topics Companies

Given an integer array `nums`, find the **subarray** with the largest sum, and return its sum.

Example 1:
Input: `nums = [-2,1,-3,4,-1,2,1,-5,4]`
Output: 6
Explanation: The subarray `[4,-1,2,1]` has the largest sum 6.

Example 2:

35.4K 342 472 Online

Submissions

Status	Language	Runtime	Memory	Notes
Accepted Dec 31, 2024	C++	0 ms	71.8 MB	
Accepted Dec 31, 2024	C++	0 ms	71.5 MB	
Accepted Jul 07, 2024	C++	90 ms	70.3 MB	
Wrong Answer Jul 07, 2024	C++	N/A	N/A	

Code

```
1 class Solution {
2 public:
3     int maxSubArray(vector<int>& nums) {
4         int maxSum = INT_MIN, currSum = 0;
5         for(int val : nums){
6             currSum += val;
7             maxSum = max(currSum, maxSum);
8             if(currSum < 0){
9                 currSum = 0;
10            }
11        }
12        return maxSum;
13    }
14 };
```

Ln 1, Col 1 Saved Upgrade to Cloud Saving

Testcase **Test Result**

Case 1 Case 2 Case 3 +

nums = [-2,1,-3,4,-1,2,1,-5,4]

</> Source

5. (240) Search a 2D matrix II

240. Search a 2D Matrix II Solved

Medium Topics Companies

Write an efficient algorithm that searches for a value `target` in an `m x n` integer matrix `matrix`. This matrix has the following properties:

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

Example 1:

Input: `matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]`, `target = 5`

Output: `true`

Accepted Runtime: 0 ms

```
1 class Solution {
2 public:
3     bool searchMatrix(vector<vector<int>>& matrix,
4                       int target) {
5         int row = 0;
6         int col = matrix[0].size() - 1;
7
8         while (row < matrix.size() && col >= 0) {
9             if (matrix[row][col] == target) {
10                 return true;
11             } else if (matrix[row][col] > target) {
12                 col--;
13             } else {
14                 row++;
15             }
16         }
17     }
18 }
```

6. (372) Super Pow

372. Super Pow Solved

Medium Topics Companies

Your task is to calculate $a^b \bmod 1337$ where a is a positive integer and b is an extremely large positive integer given in the form of an array.

Example 1:

Input: `a = 2, b = [3]`

Output: `8`

Accepted Runtime: 0 ms

```
1 class Solution {
2     const int base = 1337;
3     int powmod(int a, int k) {
4         if (k == 0) return 1;
5         a %= base;
6         int result = 1;
7         for (int i = 0; i < k; ++i) {
8             result = (result * a) % base;
9         }
10        return result;
11    }
12    int superPow(int a, vector<int>& b) {
13        if (b.empty()) return 1;
14        int last_digit = b.back();
15        b.pop_back();
16        return powmod(superPow(a, b), 10) * powmod(a, last_digit);
17    }
18 }
```

7. (932) Beautiful Array

932. Beautiful Array Solved

Medium Topics Companies

An array `nums` of length `n` is **beautiful** if:

- `nums` is a permutation of the integers in the range `[1, n]`.
- For every $0 \leq i < j < n$, there is no index `k` with $i < k < j$ where $2 * \text{nums}[k] == \text{nums}[i] + \text{nums}[j]$.

Given the integer `n`, return **any beautiful array** `nums` of length `n`. There will be at least one valid answer for the given `n`.

Submissions: 1 Accepted a few seconds ago C++ 2 ms 9.4 MB

Testcase: 1/1 Test Result: Accepted Runtime: 0 ms

```
void sort(vector<int> &v, int start, int end, int mid) {
    if(start >= end) return;
    int mid = partition(v, start, end, mid);
    sort(v, start, mid - 1, mid << 1);
    sort(v, mid, end, mid << 1);
}

vector<int> beautifulArray(int N) {
    vector<int> ans;
    for(int i = 0; i < N; i++) ans.push_back(i);
    sort(ans, 0, N - 1, 1);
    return ans;
}
```

8. (218) The Skyline Problem

218. The Skyline Problem Solved

Hard Topics Companies

A city's **skyline** is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Given the locations and heights of all the buildings, return the **skyline** formed by these buildings collectively.

The geometric information of each building is given in the array `buildings`, where `buildings[i] = [lefti, righti, heighti]`:

- `lefti` is the x coordinate of the left edge of the `i`th building.
- `righti` is the x coordinate of the right edge of the `i`th building.
- `heighti` is the height of the `i`th building.

Submissions: 1 Accepted a few seconds ago C++ 16 ms 27.9 MB

Testcase: 1/1 Test Result: Accepted Runtime: 0 ms

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>> buildings) {
        vector<pair<int, int>> events;
        multiset<int> heights = {0};
        vector<vector<int>> result;

        for(auto &b : buildings) {
            events.emplace_back(b[0], -b[2]); // left edge
            events.emplace_back(b[1], b[2]); // right edge
        }

        sort(events.begin(), events.end());
        int prevHeight = 0;

        for(auto [x, h] : events) {
            if(h < 0) {
                heights.insert(h);
            } else {
                heights.erase(heights.find(h));
            }
            int curHeight = *heights.rbegin();
            if(curHeight > prevHeight) {
                result.push_back({x, curHeight});
                prevHeight = curHeight;
            }
        }

        return result;
    }
};
```

9. (493) Reverse Pairs

493. Reverse Pairs Solved

Hard Topics Companies

Given an integer array `nums`, return the number of **reverse pairs** in the array.

A **reverse pair** is a pair `(i, j)` where:

- $0 \leq i < j < \text{nums.length}$ and
- $\text{num}[i] > 2 * \text{num}[j]$.

Submissions: 1 Accepted a few seconds ago C++ 167 ms 99.3 MB

Testcase: 1/1 Test Result: Accepted Runtime: 167 ms

```
int mergeSort(vector<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 && (long)nums[i] >= 2 * (long)nums[j]) {
            count++;
            j++;
        }
        inplace_merge(nums.begin() + i, nums.begin() + j, nums.begin() + mid + 1);
    }

    return count;
}
```

10.(2407) Longest Increasing Subsequence II

The screenshot shows the LeetCode interface for problem 2407. The problem description states: "You are given an integer array `nums`, and an integer `k`. Find the longest subsequence of `nums` that meets the following requirements: The subsequence is **strictly increasing** and The difference between adjacent elements in the subsequence is **at most** `k`. Return the length of the **longest subsequence** that meets the requirements. A **subsequence** is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements." The problem is marked as "Hard" and "Solved".

The "Submissions" table shows two attempts:

Status	Language	Runtime	Memory	Notes
Accepted a few seconds ago	C++	102 ms	143.6 MB	
Time Limit Exceeded a minute ago	C++	N/A	N/A	

The "Code" section shows the following C++ solution:

```
36 };
37
38 class Solution {
39 public:
40     int lengthOfLIS(vector<int>& nums, int k) {
41         MaxSegmentTree tree(1e5 + 1);
42         for (int i : nums) {
43             int lower = max(0, i - k);
44             int cur = 1 + tree.query(lower, i - 1);
45             tree.update(i, cur);
46         }
47         return tree.max_value();
48     }
49 };
50 }
```

The "Testcase" section shows the input: `nums = [4, 2, 1, 4, 3, 4, 5, 8, 15]` and `k =`.

The right sidebar shows the user profile for Rakshit Uniyal, with a Premium subscription. The sidebar also contains links to My Lists, Notebook, Submissions, Progress, Points, Try New Features, Orders, My Playgrounds, Settings, Classic Mode, Appearance, and Sign Out.