

Name: - Navneet Sharma

UID: - 22BCS15680

Sec/Group: - IOT_609/B

AP worksheet

1. Longest Nice Substring:

1763. Longest Nice Substring Solved

1522 Easy Topics Companies Hint

Microsoft

A string *s* is **nice** if, for every letter of the alphabet that *s* contains, it appears **both** in uppercase and lowercase. For example, "a**bb**B" 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* = "YazaAay"
Output: "aAa"
Explanation: "aAa" is a nice string because 'A/a' is the only letter of the alphabet in *s*, and both 'A' and 'a' appear. "aAa" is the longest nice substring.

Example 2:
Input: *s* = "Bb"
Output: "Bb"
Explanation: "Bb" is a nice string because both 'B' and 'b' appear. The whole string is a substring.

```
1 class Solution {
2 private:
3     bool isNice(string& str){
4         for(char c:str){
5             if(islower(c)&&str.find(toupper(c))==string::npos){
6                 return false;
7             }
8             if(isupper(c)&&str.find(tolower(c))==string::npos){
9                 return false;
10            }
11        }
12        return true;
13    }
14 public:
15     string longestNiceSubstring(string s) {
16         string ans="";
17         int n=s.length();
18         for(int i=0;i<n;i++){
19             for(int j=i+1;j<n;j++){
20                 string sub=s.substr(i,j-i+1);
21                 if(isNice(sub)){
22                     if(sub.length()>ans.length()){
23                         ans=sub;
24                     }
25                 }
26             }
27         }
28         return ans;
29     }
30 }
```

2. Reverse Bits

190. Reverse Bits Solved

Easy Topics Companies

Apple Facebook Qualcomm Microsoft Google

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 -43261596 and the output represents the signed integer -964176192.

Example 1:
Input: *n* = 0000001010010100000111011001100
Output: 964176192 (0011100101110000010100101000000)
Explanation: The input binary string 0000001010010100000111011001100 represents the unsigned integer 43261596, so return 964176192 which its binary representation is 0011100101110000010100101000000.

```
1 class Solution {
2 public:
3     uint32_t reverseBits(uint32_t n) {
4         uint32_t ans = 0;
5
6         for (int i = 0; i < 32; ++i) {
7             if (n >> i & 1) {
8                 ans |= 1 << 31 - i;
9             }
10        }
11        return ans;
12    }
13 }
```

Accepted Runtime: 2 ms

Case 1 Case 2

Input

n = 0000001010010100000111011001100

Output

964176192 (0011100101110000010100101000000)

3. Number of 1 Bits:

The screenshot displays a coding problem interface for '191. Number of 1 Bits'. The problem description states: '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)'. Two examples are provided: Example 1 with input $n = 11$ and output 3, and Example 2 with input $n = 128$ and output 1. The solution is implemented in C++ as follows:

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

The test result shows 'Accepted' with a runtime of 0 ms. The input field contains $n = 11$ and the output field contains 3.

4. Maximum Subarray:

The screenshot displays a coding problem interface for '53. Maximum Subarray'. The problem description states: 'Given an integer array $nums$, find the subarray with the largest sum, and return its sum'. Three examples are provided: Example 1 with input $nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]$ and output 6; Example 2 with input $nums = [1]$ and output 1; and Example 3 with input $nums = [5, 4, -1, 7, 8]$ and output 23. The solution is implemented in C++ as follows:

```
1 class Solution {
2 public:
3     int maxSubArray(vector<int>& nums) {
4         int sum = 0;
5         int maxi = nums[0];
6
7         for (int i = 0; i < nums.size(); i++) {
8             sum = sum + nums[i];
9             maxi = max(maxi, sum);
10            if (sum < 0) {
11                sum = 0;
12            }
13        }
14    }
15};
```

The test result shows 'Accepted' with a runtime of 0 ms. The input field contains $nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]$ and the output field contains 6.

5. Search a 2D Matrix II:

240. Search a 2D Matrix II Solved

Medium Topics Companies

Amazon Microsoft Bloomberg Apple Facebook

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:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24

12.3K 87 50 Online

```
1 class Solution {
2 public:
3     bool searchMatrix(vector<vector<int>>& matrix, int target) {
4         int row = matrix.size();
5         int col = matrix[0].size();
6
7         int rowIndex = 0;
8         int colIndex = col-1;
9
10        while(rowIndex < row && colIndex >= 0){
11            int element = matrix[rowIndex][colIndex];
12
13            if(element == target){
14                return true;
15            }
16        }
17        return false;
18    }
19 }
```

Accepted Runtime: 2 ms

Case 1 Case 2

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

6. Super Pow:

372. Super Pow

Medium Topics Companies

Adobe

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

Example 2:

Input: $a = 2, b = [1,0]$

Output: 1024

Example 3:

Input: $a = 1, b = [4,3,3,8,5,2]$

Output: 1

Constraints:

999 24 6 Online

```
12 private:
13     static constexpr int kMod = 1337;
14
15     long modPow(long x, long n) {
16         if (n == 0)
17             return 1;
18         if (n % 2 == 1)
19             return x * modPow(x % kMod, (n - 1) / 2) % kMod;
20         return modPow(x * x % kMod, n / 2) % kMod;
21     }
22
23 }
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

a =

2

b =

[3]

7. Beautiful Array:

932. Beautiful Array

2294 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`.

Example 1:

Input: $n = 4$

Output: [2,1,4,3]

Example 2:

Input: $n = 5$

Output: [3,1,2,5,4]

Constraints:

1.3K 32 6 Online

```
15 divide(arr, l, r, mask << 1);
16 divide(arr, r + 1, r, mask << 1);
17 }
18
19 int partition(vector<int>& arr, int l, int r, int mask) {
20     int nextSwapped = 1;
21     for (int i = l; i < r; ++i)
22         if (arr[i] & mask)
23             swap(arr[i], arr[nextSwapped++]);
24     return nextSwapped - 1;
25 }
26 }
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

n =

4

Output

[3,1,2,4]

8. The Skyline Problem:

218. The Skyline Problem

Hard Topics Companies

Microsoft Cruise Automation Google Uber

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.

You may assume all buildings are perfect rectangles grounded on an absolutely flat surface at height 0.

The **skyline** should be represented as a list of "key points" sorted by their x-coordinate in the form `[[x1,y1],[x2,y2],...]`. Each key point is the left endpoint of some horizontal segment in the skyline except the last point in the list, which always has a y-coordinate 0 and is used to mark the skyline's termination where the rightmost building ends. Any ground between the leftmost and rightmost buildings should be part of the skyline's contour.

Note: There must be no consecutive horizontal lines of equal height in the output.

6K 31 56 Online

```
C++
void addPoint(vector<vector<int>>& ans, int x, int y) {
    if (ans.empty() && ans.back()[0] == x) {
        ans.back()[1] = y;
        return;
    }
    if (ans.empty() && ans.back()[1] < y) {
        return;
    }
    ans.push_back({x, y});
}
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

buildings =
[[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]

Output

[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]

9. Reverse Pairs:

493. Reverse Pairs

Hard Topics Companies Hint

Amazon

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
- $\text{nums[i]} > 2 * \text{nums[j]}$.

Example 1:

Input: `nums = [1,3,2,3,1]`

Output: 2

Explanation: The reverse pairs are:
(1, 4) $\rightarrow \text{nums}[1] = 3, \text{nums}[4] = 1, 3 > 2 * 1$
(3, 4) $\rightarrow \text{nums}[3] = 3, \text{nums}[4] = 1, 3 > 2 * 1$

Example 2:

Input: `nums = [2,4,3,5,1]`

Output: 3

Explanation: The reverse pairs are:
(1, 4) $\rightarrow \text{nums}[1] = 4, \text{nums}[4] = 1, 4 > 2 * 1$

6.4K 71 52 Online

```
C++
class FenwickTree {
public:
    FenwickTree(int n) : sums(n + 1) {}

    void add(int i, int delta) {
        while (i < sums.size()) {
            sums[i] += delta;
            i += lowbit(i);
        }
    }

    int get(int i) const {
        int sum = 0;
    }
};
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =
[1,3,2,3,1]

Output

2

10. Longest Increasing Subsequence II:

2407. Longest Increasing Subsequence II

2280 Hard Topics Companies Hint

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.

Example 1:

Input: `nums = [4,2,1,4,3,4,5,8,15], k = 3`

Output: 5

Explanation:
The longest subsequence that meets the requirements is `[1,3,4,5,8]`.
The subsequence has a length of 5, so we return 5.
Note that the subsequence `[1,3,4,5,8,15]` does not meet the requirements because `15 - 8 = 7` is larger than 3.

917 26 7 Online

```
C++
class SegmentTree {
public:
    explicit SegmentTree() : root(make_unique<SegmentTreeNode>(0, 1e5 + 1, 0)) {}

    void updateRange(int i, int j, int maxLength) {
        update(root, i, j, maxLength);
    }

    int queryRange(int i, int j) {
        return query(root, i, j);
    }

private:
    std::unique_ptr<SegmentTreeNode> root;
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =
[4,2,1,4,3,4,5,8,15]

k =
3