

DAA HOLIDAY ASSIGNMENT

- 1) find-the-index-of-the-first-occurrence-in-a-string <https://leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string/>

The screenshot shows the LeetCode interface for the problem "28. Find the Index of the First Occurrence in a String". The problem is marked as "Solved" and "Easy". The description states: "Given two strings `needle` and `haystack`, return the index of the first occurrence of `needle` in `haystack`, or `-1` if `needle` is not part of `haystack`." Example 1: Input: `haystack = "sadbutsad"`, `needle = "sad"`; Output: `0`; Explanation: "sad" occurs at index 0 and 6. The first occurrence is at index 0, so we return 0. Example 2: Input: `haystack = "leetcode"`, `needle = "leeto"`; Output: `-1`; Explanation: "leeto" did not occur in "leetcode", so we return -1. Constraints: $1 \leq \text{haystack.length}, \text{needle.length} \leq 10^4$. The code editor shows a C++ solution using `string::find`. The test result shows a successful run for the input `needle = "sad"` with output `0`.

```
class Solution {
public:
    int strStr(string haystack, string needle) {
        // Find the position of the first occurrence of needle in haystack
        size_t pos = haystack.find(needle);

        // If found, return the position; otherwise, return -1
        return (pos != string::npos) ? pos : -1;
    }
};
```

- 2) Bitwise AND of numberrange <https://leetcode.com/problems/bitwise-and-of-numbers-range/description/>

The screenshot shows the LeetCode interface for the problem "201. Bitwise AND of Numbers Range". The problem is marked as "Solved" and "Medium". The description states: "Given two integers `left` and `right` that represent the range `[left, right]`, return the bitwise AND of all numbers in this range, inclusive." Example 1: Input: `left = 5`, `right = 7`; Output: `4`. Example 2: Input: `left = 0`, `right = 0`; Output: `0`. Example 3: Input: `left = 1`, `right = 2147483647`; Output: `0`. Constraints: $0 \leq \text{left} \leq \text{right} \leq 2^{31} - 1$. The code editor shows a Python solution using a loop to shift bits until `left` and `right` are equal. The test result shows "Accepted" with a runtime of 0 ms for Case 1 with input `left = 5` and `right = 7`.

```
class Solution(object):
    def rangeBitwiseAnd(self, left, right):
        # Shift left and right until they are equal
        shift = 0
        while left < right:
            left >>= 1
            right >>= 1
            shift += 1
        # Restore the common prefix
        return left << shift
```

3)SQRT(X)

<https://leetcode.com/problems/sqrtx>

The screenshot shows the LeetCode interface for problem 69, "Sqrt(x)". The left sidebar contains the problem description, which states: "Given a non-negative integer x , return the square root of x , rounded down to the nearest integer. The returned integer should be non-negative as well. You must not use any built-in exponent function or operator." It includes two examples: Example 1 with input $x = 4$ and output 2, and Example 2 with input $x = 8$ and output 2. The constraints are $0 \leq x \leq 2^{31} - 1$. The right sidebar shows a C++ code editor with a solution using binary search. The code defines a class Solution with a method mySqrt that returns the integer square root of x. The bottom status bar shows the user is logged in as 'Humid' and the system time is 07:12 on 16-01-2023.

69. Sqrt(x)

Easy Topics Companies Hint

Given a non-negative integer x , return the square root of x , rounded down to the nearest integer. The returned integer should be non-negative as well.

You must not use any built-in exponent function or operator.

- For example, do not use `pow(x, 0.5)` in c++ or `x ** 0.5` in python.

Example 1:

Input: $x = 4$
Output: 2
Explanation: The square root of 4 is 2, so we return 2.

Example 2:

Input: $x = 8$
Output: 2
Explanation: The square root of 8 is 2.8284..., and since we round it down to the nearest integer, 2 is returned.

Constraints:

- $0 \leq x \leq 2^{31} - 1$

8.6K 273 89 Online

```
1 #include <iostream>
2 using namespace std;
3
4 class Solution {
5 public:
6     int mySqrt(int x) {
7         if (x == 0 || x == 1) return x; // Handle edge cases
8
9         int left = 0, right = x, result = 0;
10
11         while (left <= right) {
12             long long mid = left + (right - left) / 2; // Use long long to avoid overflow
13             long long square = mid * mid;
14
15             if (square == x) {
16                 return mid; // Perfect square
17             } else if (square < x) {
18                 result = mid; // Update result
19                 left = mid + 1; // Search in the right half
20             } else {
21                 right = mid - 1; // Search in the left half
22             }
23         }
24
25         return result; // Return the largest integer whose square <= x
26     }
27 };
28
29
```

Ln 29, Col 1 Saved Run Submit

Testcase Test Result

Humid Now Search 07:12 16-01-2023

4)Largest Number

<https://leetcode.com/problems/largest-number/description/>

The screenshot shows the LeetCode interface for problem 179, "Largest Number". The left sidebar contains the problem description, which states: "Given a list of non-negative integers `nums`, arrange them such that they form the largest number and return it. Since the result may be very large, so you need to return a string instead of an integer." It includes two examples: Example 1 with input `nums = [10, 2]` and output "210", and Example 2 with input `nums = [3, 30, 34, 5, 9]` and output "9534330". The constraints are $1 \leq \text{nums.length} \leq 100$ and $0 \leq \text{nums}[i] \leq 10^9$. The right sidebar shows a C++ code editor with a solution using a custom comparator for sorting the numbers. The code defines a class Solution with a method largestNumber that returns the largest number as a string. The bottom status bar shows the user is logged in as 'Humid' and the system time is 07:14 on 16-01-2023.

179. Largest Number

Medium Topics Companies

Given a list of non-negative integers `nums`, arrange them such that they form the largest number and return it.

Since the result may be very large, so you need to return a string instead of an integer.

Example 1:

Input: `nums = [10, 2]`
Output: "210"

Example 2:

Input: `nums = [3, 30, 34, 5, 9]`
Output: "9534330"

Constraints:

- $1 \leq \text{nums.length} \leq 100$
- $0 \leq \text{nums}[i] \leq 10^9$

Seen this question in a real interview before? 1/5

Accepted 679.1K Submissions 1.7M Acceptance Rate 40.6%

```
1 #include <vector>
2 #include <string>
3 #include <algorithm>
4 using namespace std;
5
6 class Solution {
7 public:
8     string largestNumber(vector<int>& nums) {
9         vector<string> strNums;
10         for (int num : nums) {
11             strNums.push_back(to_string(num));
12         }
13
14         auto compare = [](const string& a, const string& b) {
15             return a + b > b + a;
16         };
17
18         sort(strNums.begin(), strNums.end(), compare);
19
20         string result;
21         for (const string& str : strNums) {
22             result += str;
23         }
24
25         if (result[0] == '0') {
26             return "0";
27         }
28
29         return result;
30     }
31 };
32
```

Ln 32, Col 1 Saved Run Submit

Testcase Test Result

Internet access

65°F Partly sunny Search 07:14 16-01-2023

5)Valid Parenthesis

<https://leetcode.com/problems/valid-parentheses/>

20. Valid Parentheses

Given a string `s` containing just the characters `'('`, `)'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

Example 1:

Input: `s = "()"`
Output: `true`

Example 2:

Input: `s = "([)]"`
Output: `false`

Example 3:

Input: `s = "{[]}"`
Output: `true`

Example 4:

Input: `s = "([)]"`
Output: `false`

```
1 #include <iostream>
2 #include <stack>
3 #include <string>
4 using namespace std;
5
6 class Solution {
7 public:
8     bool isValid(string s) {
9         stack<char> stack;
10
11         for (char c : s) {
12             // If the character is an opening bracket, push it onto the stack
13             if (c == '(' || c == '[' || c == '{') {
14                 stack.push(c);
15             }
16             // If the character is a closing bracket, check if it matches the top of the stack
17             else if (c == ')' || c == ']' || c == '}') {
18                 if (stack.empty()) {
19                     return false; // No opening bracket to match with
20                 }
21                 char top = stack.top();
22                 stack.pop();
23
24                 // Check if the top of the stack matches the corresponding opening bracket
25                 if ((c == ')' && top != '(') || (c == ']' && top != '[') || (c == '}' && top != '{')) {
26                     return false;
27                 }
28             }
29         }
30
31         // If the stack is empty, all brackets were matched, otherwise return false
32         return stack.empty();
33     }
34 };
35
```

6)Merge Two Sorted Lists

<https://leetcode.com/problems/merge-two-sorted-lists/>

21. Merge Two Sorted Lists

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.

Example 1:

Input: `list1 = [1,2,4]`, `list2 = [1,3,4]`
Output: `[1,1,2,3,4,4]`

Example 2:

Input: `list1 = []`, `list2 = []`
Output: `[]`

Example 3:

Input: `list1 = [0]`, `list2 = [0]`
Output: `[0]`

```
1 // If there are remaining nodes in list1 or list2, attach them
2 if (list1 != nullptr) {
3     current->next = list1;
4 } else if (list2 != nullptr) {
5     current->next = list2;
6 }
7
8 return dummy->next; // Return the merged list starting from the first node
9
10
11 // Helper function to create a linked list from a vector
12 ListNode* createList(const vector<int>& nums) {
13     ListNode* head = nullptr;
14     ListNode* current = nullptr;
15     for (int num : nums) {
16         if (!head) {
17             head = new ListNode(num);
18             current = head;
19         } else {
20             current->next = new ListNode(num);
21             current = current->next;
22         }
23     }
24     return head;
25 }
26
27 // Helper function to print the linked list
28 void printList(ListNode* head) {
29     while (head != nullptr) {
30         cout << head->val << " ";
31         head = head->next;
32     }
33     cout << endl;
34 }
35
```

7) Remove Duplicates from sorted list

<https://leetcode.com/problems/remove-duplicates-from-sorted-list/>

83. Remove Duplicates from Sorted List

Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list *sorted* as well.

Example 1:

Input: head = [1,1,2]
Output: [1,2]

Example 2:

Input: head = [1,1,2,3,3]
Output: [1,2,3]

```
1 #include <iostream>
2 using namespace std;
3
4 // Assume ListNode structure is already defined and precompiled.
5
6 class Solution {
7 public:
8     ListNode* deleteDuplicates(ListNode* head) {
9         ListNode* current = head;
10        while (current != nullptr && current->next != nullptr) {
11            if (current->val == current->next->val) {
12                // Skip the duplicate node
13                current->next = current->next->next;
14            } else {
15                // Move to the next node
16                current = current->next;
17            }
18        }
19        return head;
20    }
21 };
22
23 // Helper function to print the linked list
24 void printList(ListNode* head) {
25     while (head != nullptr) {
26         cout << head->val << " ";
27         head = head->next;
28     }
29     cout << endl;
30 }
31
32
33
34
```

8) Find peak Element

<https://leetcode.com/problems/find-peak-element/>

162. Find Peak Element

A peak element is an element that is strictly greater than its neighbors.

Given a 0-indexed integer array `nums`, find a peak element, and return its index. If the array contains multiple peaks, return the index to *any* of the peaks.

You may imagine that `nums[-1] = nums[n] = -∞`. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in $O(\log n)$ time.

Example 1:

Input: `nums = [1,2,3,1]`
Output: `2`
Explanation: 3 is a peak element and your function should return the index number 2.

Example 2:

Input: `nums = [1,2,1,3,5,6,4]`
Output: `5`
Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.

Constraints:

- `1 <= nums.length <= 104`
- `-109 <= nums[i] <= 109`

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 class Solution {
6 public:
7     int findPeakElement(vector<int>& nums) {
8         int left = 0, right = nums.size() - 1;
9
10        while (left < right) {
11            int mid = left + (right - left) / 2;
12
13            if (nums[mid] > nums[mid + 1]) {
14                right = mid;
15            } else {
16                left = mid + 1;
17            }
18        }
19        return left;
20    }
21 };
22
23
```

9) Binary Tree Inorder Traversal

<https://leetcode.com/problems/binary-tree-inorder-traversal/>

94. Binary Tree Inorder Traversal

Given the *root* of a binary tree, return the *inorder traversal* of its nodes' values.

Example 1:
Input: *root* = [1,null,2,3]
Output: [1,3,2]

Example 2:
Input: *root* = [1,2,3,4,5,null,8,null,null,6,7,9]

Explanation:

```
graph TD
    1((1)) --> 3((3))
    1 --> 2((2))
    2 --> 3
```

Example 1:
Input: *root* = [1,null,2,3]
Output: [1,3,2]

Example 2:
Input: *root* = [1,2,3,4,5,null,8,null,null,6,7,9]

Code:

```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 class Solution {
7 public:
8     void inorderTraversalHelper(TreeNode* root, vector<int>& result) {
9         if (root == nullptr) {
10             return;
11         }
12         // Traverse left subtree
13         inorderTraversalHelper(root->left, result);
14         // Visit the root node
15         result.push_back(root->val);
16         // Traverse right subtree
17         inorderTraversalHelper(root->right, result);
18     }
19
20     vector<int> inorderTraversal(TreeNode* root) {
21         vector<int> result;
22         inorderTraversalHelper(root, result);
23         return result;
24     }
25 };
26
27 In 30, Col 1 - Saved
```

10) N-Queens

<https://leetcode.com/problems/n-queens/>

51. N-Queens

The *n*-queens puzzle is the problem of placing *n* queens on an *n* × *n* chessboard such that no two queens attack each other.

Given an integer *n*, return all distinct solutions to the *n*-queens puzzle. You may return the answer in *any order*.

Each solution contains a distinct board configuration of the *n*-queens' placement, where *Q* and *.* both indicate a queen and an empty space, respectively.

Example 1:

Input: *n* = 4
Output: [[".Q..", "...Q.", "Q...", "..Q."], ["Q...", "Q...", "...Q.", ".Q.."]]
Explanation: There exist two distinct solutions to the 4-queens puzzle as shown above

Example 2:

Code:

```
1 #include <vector>
2 #include <string>
3 using namespace std;
4
5 class Solution {
6 public:
7     vector<vector<string>> solveNQueens(int n) {
8         vector<vector<string>> solutions;
9         vector<string> board(n, string(n, '.')); // Initialize an empty n x n board
10         vector<int> leftRow(n, 0), upperDiag(2 * n - 1, 0), lowerDiag(2 * n - 1, 0);
11         backtrack(0, board, solutions, leftRow, upperDiag, lowerDiag);
12         return solutions;
13     }
14
15 private:
16     void backtrack(int col, int n, vector<string>& board, vector<vector<string>>& solutions,
17         vector<int>& leftRow, vector<int>& upperDiag, vector<int>& lowerDiag) {
18         if (col == n) {
19             solutions.push_back(board);
20             return;
21         }
22         for (int row = 0; row < n; ++row) {
23             if (leftRow[row] == 0 && upperDiag[row + col] == 0 && lowerDiag[row - col + n - 1] == 0) {
24                 board[row][col] = 'Q';
25                 leftRow[row] = 1;
26                 upperDiag[row + col] = 1;
27                 lowerDiag[row - col + n - 1] = 1;
28                 backtrack(col + 1, n, board, solutions, leftRow, upperDiag, lowerDiag);
29                 // Undo the current placement
30                 board[row][col] = '.';
31                 leftRow[row] = 0;
32                 upperDiag[row + col] = 0;
33                 lowerDiag[row - col + n - 1] = 0;
34             }
35         }
36     }
37 }
```

G.CHARITHA
2211CS020162
AIML-BETA

