Merge k Sorted Lists

Difficulty: Hard

Link: https://leetcode.com/problems/merge-k-sorted-lists (https://leetcode.com/problems/merge-k-sorted-lists)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {


    }
};
```

Notes

Reverse Nodes in k-Group

Difficulty: Hard

Link: https://leetcode.com/problems/reverse-nodes-in-k-group (https://leetcode.com/problems/reverse-nodes-in-k-group)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* reverseKGroup(ListNode* head, int k) {

    }
};
```

Notes

Longest Valid Parentheses

Difficulty: Hard

Link: https://leetcode.com/problems/longest-valid-parentheses (https://leetcode.com/problems/longest-valid-parentheses)

C++ Code

```
class Solution {
public:
    int longestValidParentheses(string s) {

    }
};
```

Notes

First Missing Positive

Difficulty: Hard

Link: https://leetcode.com/problems/first-missing-positive (https://leetcode.com/problems/first-missing-positive)

C++ Code

```
class Solution {
public:
    int firstMissingPositive(vector<int>& nums) {

    }
};
```

Notes

Binary Tree Maximum Path Sum

Difficulty: Hard

Link: https://leetcode.com/problems/binary-tree-maximum-path-sum (https://leetcode.com/problems/binary-tree-maximum-path-sum)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int maxPathSum(TreeNode* root) {

    }
};
```

Notes

Word Ladder

Difficulty: Hard

Link: https://leetcode.com/problems/word-ladder (https://leetcode.com/problems/word-ladder)

C++ Code

```
class Solution {
public:
    int ladderLength(string beginWord, string endWord, vector<string>& wordList) {

    }
};
```

Notes

Candy

Difficulty: Hard

Link: https://leetcode.com/problems/candy (https://leetcode.com/problems/candy)

C++ Code

```
class Solution {
public:
    int candy(vector<int>& ratings) {

    }
};
```

Notes

Basic Calculator

Difficulty: Hard

Link: https://leetcode.com/problems/basic-calculator (https://leetcode.com/problems/basic-calculator)

C++ Code

```
class Solution {
public:
    int calculate(string s) {

    }
};
```

Notes

Sliding Window Maximum

Difficulty: Hard

Link: https://leetcode.com/problems/sliding-window-maximum (https://leetcode.com/problems/sliding-window-maximum)

C++ Code

```
class Solution {
public:
    vector<int> maxSlidingWindow(vector<int>& nums, int k) {

    }
};
```

Notes

Russian Doll Envelopes

Difficulty: Hard

Link: https://leetcode.com/problems/russian-doll-envelopes (https://leetcode.com/problems/russian-doll-envelopes)

C++ Code

```
class Solution {
public:
    int maxEnvelopes(vector<vector<int>>& envelopes) {

    }
};
```

Notes

Prefix and Suffix Search

Difficulty: Hard

Link: https://leetcode.com/problems/prefix-and-suffix-search (https://leetcode.com/problems/prefix-and-suffix-search)

C++ Code

```
class WordFilter {
public:
    WordFilter(vector<string>& words) {

    }

    int f(string pref, string suff) {

    }
};

/**
 * Your WordFilter object will be instantiated and called as such:
 * WordFilter* obj = new WordFilter(words);
 * int param_1 = obj->f(pref,suff);
 */
```

Notes

Sliding Puzzle

Difficulty: Hard

Link: https://leetcode.com/problems/sliding-puzzle (https://leetcode.com/problems/sliding-puzzle)

C++ Code

```
class Solution {
public:
    int slidingPuzzle(vector<vector<int>>& board) {

    }
};
```

Notes

Count Vowels Permutation

Difficulty: Hard

Link: https://leetcode.com/problems/count-vowels-permutation (https://leetcode.com/problems/count-vowels-permutation)

C++ Code

```
class Solution {
public:
    int countVowelPermutation(int n) {

    }
};
```

Notes

Minimum Difficulty of a Job Schedule

Difficulty: Hard

Link: https://leetcode.com/problems/minimum-difficulty-of-a-job-schedule (https://leetcode.com/problems/minimum-difficulty-of-a-job-schedule)

C++ Code

```
class Solution {
public:
    int minDifficulty(vector<int>& jobDifficulty, int d) {

    }
};
```

Notes

Maximum Score from Performing Multiplication Operations

Difficulty: Hard

Link: https://leetcode.com/problems/maximum-score-from-performing-multiplication-operations (https://leetcode.com/problems/maximum-score-from-performing-multiplication-operations)

C++ Code

```
class Solution {
public:
    int maximumScore(vector<int>& nums, vector<int>& multipliers) {

    }
};
```

Notes

Sum of Prefix Scores of Strings

Difficulty: Hard

Link: https://leetcode.com/problems/sum-of-prefix-scores-of-strings (https://leetcode.com/problems/sum-of-prefix-scores-of-strings)

C++ Code

```
class Solution {
public:
    vector<int> sumPrefixScores(vector<string>& words) {

    }
};
```

Notes

Add Two Numbers

Difficulty: Medium

Link: https://leetcode.com/problems/add-two-numbers (https://leetcode.com/problems/add-two-numbers)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {

    }
};
```

Notes

Longest Substring Without Repeating Characters

Difficulty: Medium

Link: https://leetcode.com/problems/longest-substring-without-repeating-characters (https://leetcode.com/problems/longest-substring-without-repeating-characters)

C++ Code

```
class Solution {
public:
    int lengthOfLongestSubstring(string s) {

    }
};
```

Notes

Longest Palindromic Substring

Difficulty: Medium

Link: https://leetcode.com/problems/longest-palindromic-substring (https://leetcode.com/problems/longest-palindromic-substring)

C++ Code

```
class Solution {
public:
    string longestPalindrome(string s) {

    }
};
```

Notes

Zigzag Conversion

Difficulty: Medium

Link: https://leetcode.com/problems/zigzag-conversion (https://leetcode.com/problems/zigzag-conversion)

C++ Code

```
class Solution {
public:
    string convert(string s, int numRows) {

    }
};
```

Notes

String to Integer (atoi)

Difficulty: Medium

Link: https://leetcode.com/problems/string-to-integer-atoi (https://leetcode.com/problems/string-to-integer-atoi)

C++ Code

```
class Solution {
public:
    int myAtoi(string s) {

    }
};
```

Notes

Container With Most Water

Difficulty: Medium

Link: https://leetcode.com/problems/container-with-most-water (https://leetcode.com/problems/container-with-most-water)

C++ Code

```
class Solution {
public:
    int maxArea(vector<int>& height) {

    }
};
```

Notes

3Sum

Difficulty: Medium

Link: https://leetcode.com/problems/3sum (https://leetcode.com/problems/3sum)

C++ Code

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {

    }
};
```

Notes

Letter Combinations of a Phone Number

Difficulty: Medium

Link: https://leetcode.com/problems/letter-combinations-of-a-phone-number (https://leetcode.com/problems/letter-combinations-of-a-phone-number)

C++ Code

```
class Solution {
public:
    vector<string> letterCombinations(string digits) {

    }
};
```

Notes

Remove Nth Node From End of List

Difficulty: Medium

Link: https://leetcode.com/problems/remove-nth-node-from-end-of-list (https://leetcode.com/problems/remove-nth-node-from-end-of-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {

    }
};
```

Notes

Generate Parentheses

Difficulty: Medium

Link: https://leetcode.com/problems/generate-parentheses (https://leetcode.com/problems/generate-parentheses)

C++ Code

```
class Solution {
public:
    vector<string> generateParenthesis(int n) {

    }
};
```

Notes

Swap Nodes in Pairs

Difficulty: Medium

Link: https://leetcode.com/problems/swap-nodes-in-pairs (https://leetcode.com/problems/swap-nodes-in-pairs)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* swapPairs(ListNode* head) {

    }
};
```

Notes

Divide Two Integers

Difficulty: Medium

Link: https://leetcode.com/problems/divide-two-integers (https://leetcode.com/problems/divide-two-integers)

C++ Code

```
class Solution {
public:
    int divide(int dividend, int divisor) {

    }
};
```

Notes

Next Permutation

Difficulty: Medium

Link: https://leetcode.com/problems/next-permutation (https://leetcode.com/problems/next-permutation)

C++ Code

```
class Solution {
public:
    void nextPermutation(vector<int>& nums) {

    }
};
```

Notes

Search in Rotated Sorted Array

Difficulty: Medium

Link: https://leetcode.com/problems/search-in-rotated-sorted-array (https://leetcode.com/problems/search-in-rotated-sorted-array)

C++ Code

```
class Solution {
public:
    int search(vector<int>& nums, int target) {

    }
};
```

Notes

Find First and Last Position of Element in Sorted Array

Difficulty: Medium

Link: https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array (https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array)

C++ Code

```
class Solution {
public:
    vector<int> searchRange(vector<int>& nums, int target) {

    }
};
```

Notes

Count and Say

Difficulty: Medium

Link: https://leetcode.com/problems/count-and-say (https://leetcode.com/problems/count-and-say)

C++ Code

```
class Solution {
public:
    string countAndSay(int n) {

    }
};
```

Notes

Combination Sum

Difficulty: Medium

Link: https://leetcode.com/problems/combination-sum (https://leetcode.com/problems/combination-sum)

C++ Code

```
class Solution {
public:
    vector<vector<int>> combinationSum(vector<int>& candidates, int target) {

    }
};
```

Notes

Combination Sum II

Difficulty: Medium

Link: https://leetcode.com/problems/combination-sum-ii (https://leetcode.com/problems/combination-sum-ii)

C++ Code

```
class Solution {
public:
    vector<vector<int>> combinationSum2(vector<int>& candidates, int target) {

    }
};
```

Notes

Multiply Strings

Difficulty: Medium

Link: https://leetcode.com/problems/multiply-strings (https://leetcode.com/problems/multiply-strings)

C++ Code

```
class Solution {
public:
    string multiply(string num1, string num2) {

    }
};
```

Notes

Jump Game II

Difficulty: Medium

Link: https://leetcode.com/problems/jump-game-ii (https://leetcode.com/problems/jump-game-ii)

C++ Code

```
class Solution {
public:
    int jump(vector<int>& nums) {

    }
};
```

Notes

Permutations

Difficulty: Medium

Link: https://leetcode.com/problems/permutations (https://leetcode.com/problems/permutations)

C++ Code

```
class Solution {
public:
    vector<vector<int>> permute(vector<int>& nums) {

    }
};
```

Notes

Permutations II

Difficulty: Medium

Link: https://leetcode.com/problems/permutations-ii (https://leetcode.com/problems/permutations-ii)

C++ Code

```
class Solution {
public:
    vector<vector<int>> permuteUnique(vector<int>& nums) {

    }
};
```

Notes

Rotate Image

Difficulty: Medium

Link: https://leetcode.com/problems/rotate-image (https://leetcode.com/problems/rotate-image)

C++ Code

```
class Solution {
public:
    void rotate(vector<vector<int>>& matrix) {

    }
};
```

Notes

Group Anagrams

Difficulty: Medium

Link: https://leetcode.com/problems/group-anagrams (https://leetcode.com/problems/group-anagrams)

C++ Code

```
class Solution {
public:
    vector<vector<string>> groupAnagrams(vector<string>& strs) {

    }
};
```

Notes

Pow(x, n)

Difficulty: Medium

Link: https://leetcode.com/problems/powx-n (https://leetcode.com/problems/powx-n)

C++ Code

```
class Solution {
public:
    double myPow(double x, int n) {

    }
};
```

Notes

Maximum Subarray

Difficulty: Medium

Link: https://leetcode.com/problems/maximum-subarray (https://leetcode.com/problems/maximum-subarray)

C++ Code

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {

    }
};
```

Notes

Spiral Matrix

Difficulty: Medium

Link: https://leetcode.com/problems/spiral-matrix (https://leetcode.com/problems/spiral-matrix)

C++ Code

```
class Solution {
public:
    vector<int> spiralOrder(vector<vector<int>>& matrix) {

    }
};
```

Notes

Jump Game

Difficulty: Medium

Link: https://leetcode.com/problems/jump-game (https://leetcode.com/problems/jump-game)

C++ Code

```
class Solution {
public:
    bool canJump(vector<int>& nums) {

    }
};
```

Notes

Merge Intervals

Difficulty: Medium

Link: https://leetcode.com/problems/merge-intervals (https://leetcode.com/problems/merge-intervals)

C++ Code

```
class Solution {
public:
    vector<vector<int>> merge(vector<vector<int>>& intervals) {

    }
};
```

Notes

Insert Interval

Difficulty: Medium

Link: https://leetcode.com/problems/insert-interval (https://leetcode.com/problems/insert-interval)

C++ Code

```
class Solution {
public:
    vector<vector<int>> insert(vector<vector<int>>& intervals, vector<int>& newInterval) {

    }
};
```

Notes

Rotate List

Difficulty: Medium

Link: https://leetcode.com/problems/rotate-list (https://leetcode.com/problems/rotate-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {

    }
};
```

Notes

Unique Paths

Difficulty: Medium

Link: https://leetcode.com/problems/unique-paths (https://leetcode.com/problems/unique-paths)

C++ Code

```
class Solution {
public:
    int uniquePaths(int m, int n) {

    }
};
```

Notes

Minimum Path Sum

Difficulty: Medium

Link: https://leetcode.com/problems/minimum-path-sum (https://leetcode.com/problems/minimum-path-sum)

C++ Code

```
class Solution {
public:
    int minPathSum(vector<vector<int>>& grid) {

    }
};
```

Notes

Simplify Path

Difficulty: Medium

Link: https://leetcode.com/problems/simplify-path (https://leetcode.com/problems/simplify-path)

C++ Code

```
class Solution {
public:
    string simplifyPath(string path) {

    }
};
```

Notes

Set Matrix Zeroes

Difficulty: Medium

Link: https://leetcode.com/problems/set-matrix-zeroes (https://leetcode.com/problems/set-matrix-zeroes)

C++ Code

```
class Solution {
public:
    void setZeroes(vector<vector<int>>& matrix) {

    }
};
```

Notes

Search a 2D Matrix

Difficulty: Medium

Link: https://leetcode.com/problems/search-a-2d-matrix (https://leetcode.com/problems/search-a-2d-matrix)

C++ Code

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {

    }
};
```

Notes

Sort Colors

Difficulty: Medium

Link: https://leetcode.com/problems/sort-colors (https://leetcode.com/problems/sort-colors)

C++ Code

```
class Solution {
public:
    void sortColors(vector<int>& nums) {

    }
};
```

Notes

Combinations

Difficulty: Medium

Link: https://leetcode.com/problems/combinations (https://leetcode.com/problems/combinations)

C++ Code

```
class Solution {
public:
    vector<vector<int>> combine(int n, int k) {

    }
};
```

Notes

Subsets

Difficulty: Medium

Link: https://leetcode.com/problems/subsets (https://leetcode.com/problems/subsets)

C++ Code

```
class Solution {
public:
    vector<vector<int>> subsets(vector<int>& nums) {

    }
};
```

Notes

Word Search

Difficulty: Medium

Link: https://leetcode.com/problems/word-search (https://leetcode.com/problems/word-search)

C++ Code

```
class Solution {
public:
    bool exist(vector<vector<char>>& board, string word) {

    }
};
```

Notes

Remove Duplicates from Sorted Array II

Difficulty: Medium

Link: https://leetcode.com/problems/remove-duplicates-from-sorted-array-ii (https://leetcode.com/problems/remove-duplicates-from-sorted-array-ii)

C++ Code

```
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {

    }
};
```

Notes

Search in Rotated Sorted Array II

Difficulty: Medium

Link: https://leetcode.com/problems/search-in-rotated-sorted-array-ii (https://leetcode.com/problems/search-in-rotated-sorted-array-ii)

C++ Code

```
class Solution {
public:
    bool search(vector<int>& nums, int target) {

    }
};
```

Notes

Remove Duplicates from Sorted List II

Difficulty: Medium

Link: https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii (https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {

    }
};
```

Notes

Partition List

Difficulty: Medium

Link: https://leetcode.com/problems/partition-list (https://leetcode.com/problems/partition-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* partition(ListNode* head, int x) {

    }
};
```

Notes

Subsets II

Difficulty: Medium

Link: https://leetcode.com/problems/subsets-ii (https://leetcode.com/problems/subsets-ii)

C++ Code

```
class Solution {
public:
    vector<vector<int>> subsetsWithDup(vector<int>& nums) {

    }
};
```

Notes

Decode Ways

Difficulty: Medium

Link: https://leetcode.com/problems/decode-ways (https://leetcode.com/problems/decode-ways)

C++ Code

```
class Solution {
public:
    int numDecodings(string s) {

    }
};
```

Notes

Reverse Linked List II

Difficulty: Medium

Link: https://leetcode.com/problems/reverse-linked-list-ii (https://leetcode.com/problems/reverse-linked-list-ii)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* reverseBetween(ListNode* head, int left, int right) {

    }
};
```

Notes

Unique Binary Search Trees II

Difficulty: Medium

Link: https://leetcode.com/problems/unique-binary-search-trees-ii (https://leetcode.com/problems/unique-binary-search-trees-ii)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<TreeNode*> generateTrees(int n) {

    }
};
```

Notes

Unique Binary Search Trees

Difficulty: Medium

Link: https://leetcode.com/problems/unique-binary-search-trees (https://leetcode.com/problems/unique-binary-search-trees)

C++ Code

```
class Solution {
public:
    int numTrees(int n) {

    }
};
```

Notes

Validate Binary Search Tree

Difficulty: Medium

Link: https://leetcode.com/problems/validate-binary-search-tree (https://leetcode.com/problems/validate-binary-search-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool isValidBST(TreeNode* root) {

    }
};
```

Notes

Recover Binary Search Tree

Difficulty: Medium

Link: https://leetcode.com/problems/recover-binary-search-tree (https://leetcode.com/problems/recover-binary-search-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    void recoverTree(TreeNode* root) {

    }
};
```

Notes

Binary Tree Level Order Traversal

Difficulty: Medium

Link: https://leetcode.com/problems/binary-tree-level-order-traversal (https://leetcode.com/problems/binary-tree-level-order-traversal)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {

    }
};
```

Notes

Binary Tree Zigzag Level Order Traversal

Difficulty: Medium

Link: https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal (https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {

    }
};
```

Notes

Construct Binary Tree from Preorder and Inorder Traversal

Difficulty: Medium

Link: https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal (https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* buildTree(vector<int>& preorder, vector<int>& inorder) {

    }
};
```

Notes

Construct Binary Tree from Inorder and Postorder Traversal

Difficulty: Medium

Link: https://leetcode.com/problems/construct-binary-tree-from-inorder-and-postorder-traversal (https://leetcode.com/problems/construct-binary-tree-from-inorder-and-postorder-traversal)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder) {

    }
};
```

Notes

Binary Tree Level Order Traversal II

Difficulty: Medium

Link: https://leetcode.com/problems/binary-tree-level-order-traversal-ii (https://leetcode.com/problems/binary-tree-level-order-traversal-ii)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<vector<int>> levelOrderBottom(TreeNode* root) {

    }
};
```

Notes

Convert Sorted List to Binary Search Tree

Difficulty: Medium

Link: https://leetcode.com/problems/convert-sorted-list-to-binary-search-tree (https://leetcode.com/problems/convert-sorted-list-to-binary-search-tree)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* sortedListToBST(ListNode* head) {

    }
};
```

Notes

Path Sum II

Difficulty: Medium

Link: https://leetcode.com/problems/path-sum-ii (https://leetcode.com/problems/path-sum-ii)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<vector<int>> pathSum(TreeNode* root, int targetSum) {

    }
};
```

Notes

Flatten Binary Tree to Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/flatten-binary-tree-to-linked-list (https://leetcode.com/problems/flatten-binary-tree-to-linked-list)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    void flatten(TreeNode* root) {

    }
};
```

Notes

Populating Next Right Pointers in Each Node

Difficulty: Medium

Link: https://leetcode.com/problems/populating-next-right-pointers-in-each-node (https://leetcode.com/problems/populating-next-right-pointers-in-each-node)

C++ Code

```
/*
// Definition for a Node.
class Node {
public:
    int val;
    Node* left;
    Node* right;
    Node* next;

    Node() : val(0), left(NULL), right(NULL), next(NULL) {}

    Node(int _val) : val(_val), left(NULL), right(NULL), next(NULL) {}

    Node(int _val, Node* _left, Node* _right, Node* _next)
        : val(_val), left(_left), right(_right), next(_next) {}
};
*/

class Solution {
public:
    Node* connect(Node* root) {

    }
};
```

Notes

Populating Next Right Pointers in Each Node II

Difficulty: Medium

Link: https://leetcode.com/problems/populating-next-right-pointers-in-each-node-ii (https://leetcode.com/problems/populating-next-right-pointers-in-each-node-ii)

C++ Code

```
/*
// Definition for a Node.
class Node {
public:
    int val;
    Node* left;
    Node* right;
    Node* next;

    Node() : val(0), left(NULL), right(NULL), next(NULL) {}

    Node(int _val) : val(_val), left(NULL), right(NULL), next(NULL) {}

    Node(int _val, Node* _left, Node* _right, Node* _next)
        : val(_val), left(_left), right(_right), next(_next) {}
};
*/

class Solution {
public:
    Node* connect(Node* root) {

    }
};
```

Notes

Triangle

Difficulty: Medium

Link: https://leetcode.com/problems/triangle (https://leetcode.com/problems/triangle)

C++ Code

```
class Solution {
public:
    int minimumTotal(vector<vector<int>>& triangle) {

    }
};
```

Notes

Best Time to Buy and Sell Stock II

Difficulty: Medium

Link: https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii (https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii)

C++ Code

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {

    }
};
```

Notes

Longest Consecutive Sequence

Difficulty: Medium

Link: https://leetcode.com/problems/longest-consecutive-sequence (https://leetcode.com/problems/longest-consecutive-sequence)

C++ Code

```
class Solution {
public:
    int longestConsecutive(vector<int>& nums) {

    }
};
```

Notes

Sum Root to Leaf Numbers

Difficulty: Medium

Link: https://leetcode.com/problems/sum-root-to-leaf-numbers (https://leetcode.com/problems/sum-root-to-leaf-numbers)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int sumNumbers(TreeNode* root) {

    }
};
```

Notes

Surrounded Regions

Difficulty: Medium

Link: https://leetcode.com/problems/surrounded-regions (https://leetcode.com/problems/surrounded-regions)

C++ Code

```
class Solution {
public:
    void solve(vector<vector<char>>& board) {

    }
};
```

Notes

Palindrome Partitioning

Difficulty: Medium

Link: https://leetcode.com/problems/palindrome-partitioning (https://leetcode.com/problems/palindrome-partitioning)

C++ Code

```
class Solution {
public:
    vector<vector<string>> partition(string s) {

    }
};
```

Notes

Clone Graph

Difficulty: Medium

Link: https://leetcode.com/problems/clone-graph (https://leetcode.com/problems/clone-graph)

C++ Code

```
/*
// Definition for a Node.
class Node {
public:
    int val;
    vector<Node*> neighbors;
    Node() {
        val = 0;
        neighbors = vector<Node*>();
    }
    Node(int _val) {
        val = _val;
        neighbors = vector<Node*>();
    }
    Node(int _val, vector<Node*> _neighbors) {
        val = _val;
        neighbors = _neighbors;
    }
};
*/

class Solution {
public:
    Node* cloneGraph(Node* node) {

    }
};
```

Notes

Gas Station

Difficulty: Medium

Link: https://leetcode.com/problems/gas-station (https://leetcode.com/problems/gas-station)

C++ Code

```
class Solution {
public:
    int canCompleteCircuit(vector<int>& gas, vector<int>& cost) {

    }
};
```

Notes

Single Number II

Difficulty: Medium

Link: https://leetcode.com/problems/single-number-ii (https://leetcode.com/problems/single-number-ii)

C++ Code

```
class Solution {
public:
    int singleNumber(vector<int>& nums) {

    }
};
```

Notes

Copy List with Random Pointer

Difficulty: Medium

Link: https://leetcode.com/problems/copy-list-with-random-pointer (https://leetcode.com/problems/copy-list-with-random-pointer)

C++ Code

```
/*
// Definition for a Node.
class Node {
public:
    int val;
    Node* next;
    Node* random;

    Node(int _val) {
        val = _val;
        next = NULL;
        random = NULL;
    }
};
*/

class Solution {
public:
    Node* copyRandomList(Node* head) {

    }
};
```

Notes

Word Break

Difficulty: Medium

Link: https://leetcode.com/problems/word-break (https://leetcode.com/problems/word-break)

C++ Code

```
class Solution {
public:
    bool wordBreak(string s, vector<string>& wordDict) {

    }
};
```

Notes

Linked List Cycle II

Difficulty: Medium

Link: https://leetcode.com/problems/linked-list-cycle-ii (https://leetcode.com/problems/linked-list-cycle-ii)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {

    }
};
```

Notes

Reorder List

Difficulty: Medium

Link: https://leetcode.com/problems/reorder-list (https://leetcode.com/problems/reorder-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    void reorderList(ListNode* head) {

    }
};
```

Notes

Sort List

Difficulty: Medium

Link: https://leetcode.com/problems/sort-list (https://leetcode.com/problems/sort-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* sortList(ListNode* head) {

    }
};
```

Notes

Reverse Words in a String

Difficulty: Medium

Link: https://leetcode.com/problems/reverse-words-in-a-string (https://leetcode.com/problems/reverse-words-in-a-string)

C++ Code

```
class Solution {
public:
    string reverseWords(string s) {

    }
};
```

Notes

Maximum Product Subarray

Difficulty: Medium

Link: https://leetcode.com/problems/maximum-product-subarray (https://leetcode.com/problems/maximum-product-subarray)

C++ Code

```
class Solution {
public:
    int maxProduct(vector<int>& nums) {

    }
};
```

Notes

Find Minimum in Rotated Sorted Array

Difficulty: Medium

Link: https://leetcode.com/problems/find-minimum-in-rotated-sorted-array (https://leetcode.com/problems/find-minimum-in-rotated-sorted-array)

C++ Code

```
class Solution {
public:
    int findMin(vector<int>& nums) {

    }
};
```

Notes

Min Stack

Difficulty: Medium

Link: https://leetcode.com/problems/min-stack (https://leetcode.com/problems/min-stack)

C++ Code

```
class MinStack {
public:
    MinStack() {

    }

    void push(int val) {

    }

    void pop() {

    }

    int top() {

    }

    int getMin() {

    }
};

/**
 * Your MinStack object will be instantiated and called as such:
 * MinStack* obj = new MinStack();
 * obj->push(val);
 * obj->pop();
 * int param_3 = obj->top();
 * int param_4 = obj->getMin();
 */
```

Notes

Find Peak Element

Difficulty: Medium

Link: https://leetcode.com/problems/find-peak-element (https://leetcode.com/problems/find-peak-element)

C++ Code

```
class Solution {
public:
    int findPeakElement(vector<int>& nums) {

    }
};
```

Notes

Fraction to Recurring Decimal

Difficulty: Medium

Link: https://leetcode.com/problems/fraction-to-recurring-decimal (https://leetcode.com/problems/fraction-to-recurring-decimal)

C++ Code

```
class Solution {
public:
    string fractionToDecimal(int numerator, int denominator) {

    }
};
```

Notes

Two Sum II - Input Array Is Sorted

Difficulty: Medium

Link: https://leetcode.com/problems/two-sum-ii-input-array-is-sorted (https://leetcode.com/problems/two-sum-ii-input-array-is-sorted)

C++ Code

```
class Solution {
public:
    vector<int> twoSum(vector<int>& numbers, int target) {

    }
};
```

Notes

Factorial Trailing Zeroes

Difficulty: Medium

Link: https://leetcode.com/problems/factorial-trailing-zeroes (https://leetcode.com/problems/factorial-trailing-zeroes)

C++ Code

```
class Solution {
public:
    int trailingZeroes(int n) {

    }
};
```

Notes

Binary Search Tree Iterator

Difficulty: Medium

Link: https://leetcode.com/problems/binary-search-tree-iterator (https://leetcode.com/problems/binary-search-tree-iterator)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class BSTIterator {
public:
    BSTIterator(TreeNode* root) {

    }

    int next() {

    }

    bool hasNext() {

    }
};

/**
 * Your BSTIterator object will be instantiated and called as such:
 * BSTIterator* obj = new BSTIterator(root);
 * int param_1 = obj->next();
 * bool param_2 = obj->hasNext();
 */
```

Notes

Largest Number

Difficulty: Medium

Link: https://leetcode.com/problems/largest-number (https://leetcode.com/problems/largest-number)

C++ Code

```
class Solution {
public:
    string largestNumber(vector<int>& nums) {

    }
};
```

Notes

Rotate Array

Difficulty: Medium

Link: https://leetcode.com/problems/rotate-array (https://leetcode.com/problems/rotate-array)

C++ Code

```
class Solution {
public:
    void rotate(vector<int>& nums, int k) {

    }
};
```

Notes

House Robber

Difficulty: Medium

Link: https://leetcode.com/problems/house-robber (https://leetcode.com/problems/house-robber)

C++ Code

```
class Solution {
public:
    int rob(vector<int>& nums) {

    }
};
```

Notes

Binary Tree Right Side View

Difficulty: Medium

Link: https://leetcode.com/problems/binary-tree-right-side-view (https://leetcode.com/problems/binary-tree-right-side-view)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<int> rightSideView(TreeNode* root) {

    }
};
```

Notes

Number of Islands

Difficulty: Medium

Link: https://leetcode.com/problems/number-of-islands (https://leetcode.com/problems/number-of-islands)

C++ Code

```
class Solution {
public:
    int numIslands(vector<vector<char>>& grid) {

    }
};
```

Notes

Bitwise AND of Numbers Range

Difficulty: Medium

Link: https://leetcode.com/problems/bitwise-and-of-numbers-range (https://leetcode.com/problems/bitwise-and-of-numbers-range)

C++ Code

```
class Solution {
public:
    int rangeBitwiseAnd(int left, int right) {

    }
};
```

Notes

Course Schedule

Difficulty: Medium

Link: https://leetcode.com/problems/course-schedule (https://leetcode.com/problems/course-schedule)

C++ Code

```
class Solution {
public:
    bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {

    }
};
```

Notes

Implement Trie (Prefix Tree)

Difficulty: Medium

Link: https://leetcode.com/problems/implement-trie-prefix-tree (https://leetcode.com/problems/implement-trie-prefix-tree)

C++ Code

```
class Trie {
public:
    Trie() {

    }

    void insert(string word) {

    }

    bool search(string word) {

    }

    bool startsWith(string prefix) {

    }
};

/**
 * Your Trie object will be instantiated and called as such:
 * Trie* obj = new Trie();
 * obj->insert(word);
 * bool param_2 = obj->search(word);
 * bool param_3 = obj->startsWith(prefix);
 */
```

Notes

Minimum Size Subarray Sum

Difficulty: Medium

Link: https://leetcode.com/problems/minimum-size-subarray-sum (https://leetcode.com/problems/minimum-size-subarray-sum)

C++ Code

```
class Solution {
public:
    int minSubArrayLen(int target, vector<int>& nums) {

    }
};
```

Notes

House Robber II

Difficulty: Medium

Link: https://leetcode.com/problems/house-robber-ii (https://leetcode.com/problems/house-robber-ii)

C++ Code

```
class Solution {
public:
    int rob(vector<int>& nums) {

    }
};
```

Notes

Kth Largest Element in an Array

Difficulty: Medium

Link: https://leetcode.com/problems/kth-largest-element-in-an-array (https://leetcode.com/problems/kth-largest-element-in-an-array)

C++ Code

```
class Solution {
public:
    int findKthLargest(vector<int>& nums, int k) {

    }
};
```

Notes

Combination Sum III

Difficulty: Medium

Link: https://leetcode.com/problems/combination-sum-iii (https://leetcode.com/problems/combination-sum-iii)

C++ Code

```
class Solution {
public:
    vector<vector<int>> combinationSum3(int k, int n) {

    }
};
```

Notes

Maximal Square

Difficulty: Medium

Link: https://leetcode.com/problems/maximal-square (https://leetcode.com/problems/maximal-square)

C++ Code

```
class Solution {
public:
    int maximalSquare(vector<vector<char>>& matrix) {

    }
};
```

Notes

Kth Smallest Element in a BST

Difficulty: Medium

Link: https://leetcode.com/problems/kth-smallest-element-in-a-bst (https://leetcode.com/problems/kth-smallest-element-in-a-bst)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int kthSmallest(TreeNode* root, int k) {

    }
};
```

Notes

Lowest Common Ancestor of a Binary Search Tree

Difficulty: Medium

Link: https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree (https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */

class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {

    }
};
```

Notes

Lowest Common Ancestor of a Binary Tree

Difficulty: Medium

Link: https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree (https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {

    }
};
```

Notes

Delete Node in a Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/delete-node-in-a-linked-list (https://leetcode.com/problems/delete-node-in-a-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    void deleteNode(ListNode* node) {

    }
};
```

Notes

Product of Array Except Self

Difficulty: Medium

Link: https://leetcode.com/problems/product-of-array-except-self (https://leetcode.com/problems/product-of-array-except-self)

C++ Code

```
class Solution {
public:
    vector<int> productExceptSelf(vector<int>& nums) {

    }
};
```

Notes

Search a 2D Matrix II

Difficulty: Medium

Link: https://leetcode.com/problems/search-a-2d-matrix-ii (https://leetcode.com/problems/search-a-2d-matrix-ii)

C++ Code

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {

    }
};
```

Notes

Different Ways to Add Parentheses

Difficulty: Medium

Link: https://leetcode.com/problems/different-ways-to-add-parentheses (https://leetcode.com/problems/different-ways-to-add-parentheses)

C++ Code

```
class Solution {
public:
    vector<int> diffWaysToCompute(string expression) {

    }
};
```

Notes

Meeting Rooms II

Difficulty: Medium

Link: https://leetcode.com/problems/meeting-rooms-ii (https://leetcode.com/problems/meeting-rooms-ii)

C++ Code

```

```

Notes

Single Number III

Difficulty: Medium

Link: https://leetcode.com/problems/single-number-iii (https://leetcode.com/problems/single-number-iii)

C++ Code

```
class Solution {
public:
    vector<int> singleNumber(vector<int>& nums) {

    }
};
```

Notes

Graph Valid Tree

Difficulty: Medium

Link: https://leetcode.com/problems/graph-valid-tree (https://leetcode.com/problems/graph-valid-tree)

C++ Code

```

```

Notes

H-Index

Difficulty: Medium

Link: https://leetcode.com/problems/h-index (https://leetcode.com/problems/h-index)

C++ Code

```
class Solution {
public:
    int hIndex(vector<int>& citations) {

    }
};
```

Notes

H-Index II

Difficulty: Medium

Link: https://leetcode.com/problems/h-index-ii (https://leetcode.com/problems/h-index-ii)

C++ Code

```
class Solution {
public:
    int hIndex(vector<int>& citations) {

    }
};
```

Notes

Find the Celebrity

Difficulty: Medium

Link: https://leetcode.com/problems/find-the-celebrity (https://leetcode.com/problems/find-the-celebrity)

C++ Code

```

```

Notes

Perfect Squares

Difficulty: Medium

Link: https://leetcode.com/problems/perfect-squares (https://leetcode.com/problems/perfect-squares)

C++ Code

```
class Solution {
public:
    int numSquares(int n) {

    }
};
```

Notes

Zigzag Iterator

Difficulty: Medium

Link: https://leetcode.com/problems/zigzag-iterator (https://leetcode.com/problems/zigzag-iterator)

C++ Code

Notes

Inorder Successor in BST

Difficulty: Medium

Link: https://leetcode.com/problems/inorder-successor-in-bst (https://leetcode.com/problems/inorder-successor-in-bst)

C++ Code

Notes

Walls and Gates

Difficulty: Medium

Link: https://leetcode.com/problems/walls-and-gates (https://leetcode.com/problems/walls-and-gates)

C++ Code

Notes

Find the Duplicate Number

Difficulty: Medium

Link: https://leetcode.com/problems/find-the-duplicate-number (https://leetcode.com/problems/find-the-duplicate-number)

C++ Code

```
class Solution {
public:
    int findDuplicate(vector<int>& nums) {

    }
};
```

Notes

Game of Life

Difficulty: Medium

Link: https://leetcode.com/problems/game-of-life (https://leetcode.com/problems/game-of-life)

C++ Code

```
class Solution {
public:
    void gameOfLife(vector<vector<int>>& board) {

    }
};
```

Notes

Longest Increasing Subsequence

Difficulty: Medium

Link: https://leetcode.com/problems/longest-increasing-subsequence (https://leetcode.com/problems/longest-increasing-subsequence)

C++ Code

```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums) {

    }
};
```

Notes

Range Sum Query - Mutable

Difficulty: Medium

Link: https://leetcode.com/problems/range-sum-query-mutable (https://leetcode.com/problems/range-sum-query-mutable)

C++ Code

```
class NumArray {
public:
    NumArray(vector<int>& nums) {

    }

    void update(int index, int val) {

    }

    int sumRange(int left, int right) {

    }
};

/**
 * Your NumArray object will be instantiated and called as such:
 * NumArray* obj = new NumArray(nums);
 * obj->update(index,val);
 * int param_2 = obj->sumRange(left,right);
 */
```

Notes

Best Time to Buy and Sell Stock with Cooldown

Difficulty: Medium

Link: https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-cooldown (https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-cooldown)

C++ Code

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {

    }
};
```

Notes

Bulb Switcher

Difficulty: Medium

Link: https://leetcode.com/problems/bulb-switcher (https://leetcode.com/problems/bulb-switcher)

C++ Code

```
class Solution {
public:
    int bulbSwitch(int n) {

    }
};
```

Notes

Generalized Abbreviation

Difficulty: Medium

Link: https://leetcode.com/problems/generalized-abbreviation (https://leetcode.com/problems/generalized-abbreviation)

C++ Code

```

```

Notes

Coin Change

Difficulty: Medium

Link: https://leetcode.com/problems/coin-change (https://leetcode.com/problems/coin-change)

C++ Code

```
class Solution {
public:
    int coinChange(vector<int>& coins, int amount) {

    }
};
```

Notes

Number of Connected Components in an Undirected Graph

Difficulty: Medium

Link: https://leetcode.com/problems/number-of-connected-components-in-an-undirected-graph (https://leetcode.com/problems/number-of-connected-components-in-an-undirected-graph)

C++ Code

```

```

Notes

Odd Even Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/odd-even-linked-list (https://leetcode.com/problems/odd-even-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* oddEvenList(ListNode* head) {

    }
};
```

Notes

Increasing Triplet Subsequence

Difficulty: Medium

Link: https://leetcode.com/problems/increasing-triplet-subsequence (https://leetcode.com/problems/increasing-triplet-subsequence)

C++ Code

```
class Solution {
public:
    bool increasingTriplet(vector<int>& nums) {

    }
};
```

Notes

House Robber III

Difficulty: Medium

Link: https://leetcode.com/problems/house-robber-iii (https://leetcode.com/problems/house-robber-iii)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int rob(TreeNode* root) {

    }
};
```

Notes

Sort Transformed Array

Difficulty: Medium

Link: https://leetcode.com/problems/sort-transformed-array (https://leetcode.com/problems/sort-transformed-array)

C++ Code

```

```

Notes

Bomb Enemy

Difficulty: Medium

Link: https://leetcode.com/problems/bomb-enemy (https://leetcode.com/problems/bomb-enemy)

C++ Code

```

```

Notes

Design Hit Counter

Difficulty: Medium

Link: https://leetcode.com/problems/design-hit-counter (https://leetcode.com/problems/design-hit-counter)

C++ Code

```

```

Notes

Water and Jug Problem

Difficulty: Medium

Link: https://leetcode.com/problems/water-and-jug-problem (https://leetcode.com/problems/water-and-jug-problem)

C++ Code

```
class Solution {
public:
    bool canMeasureWater(int x, int y, int target) {

    }
};
```

Notes

Find Leaves of Binary Tree

Difficulty: Medium

Link: https://leetcode.com/problems/find-leaves-of-binary-tree (https://leetcode.com/problems/find-leaves-of-binary-tree)

C++ Code

```

```

Notes

Plus One Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/plus-one-linked-list (https://leetcode.com/problems/plus-one-linked-list)

C++ Code

```
```

Notes

Sum of Two Integers

Difficulty: Medium

Link: https://leetcode.com/problems/sum-of-two-integers (https://leetcode.com/problems/sum-of-two-integers)

C++ Code

```
class Solution {
public:
    int getSum(int a, int b) {

    }
};
```

Notes

Find K Pairs with Smallest Sums

Difficulty: Medium

Link: https://leetcode.com/problems/find-k-pairs-with-smallest-sums (https://leetcode.com/problems/find-k-pairs-with-smallest-sums)

C++ Code

```
class Solution {
public:
    vector<vector<int>> kSmallestPairs(vector<int>& nums1, vector<int>& nums2, int k) {

    }
};
```

Notes

Combination Sum IV

Difficulty: Medium

Link: https://leetcode.com/problems/combination-sum-iv (https://leetcode.com/problems/combination-sum-iv)

C++ Code

```
class Solution {
public:
    int combinationSum4(vector<int>& nums, int target) {

    }
};
```

Notes

Kth Smallest Element in a Sorted Matrix

Difficulty: Medium

Link: https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix (https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix)

C++ Code

```
class Solution {
public:
    int kthSmallest(vector<vector<int>>& matrix, int k) {

    }
};
```

Notes

Lexicographical Numbers

Difficulty: Medium

Link: https://leetcode.com/problems/lexicographical-numbers (https://leetcode.com/problems/lexicographical-numbers)

C++ Code

```
class Solution {
public:
    vector<int> lexicalOrder(int n) {

    }
};
```

Notes

Elimination Game

Difficulty: Medium

Link: https://leetcode.com/problems/elimination-game (https://leetcode.com/problems/elimination-game)

C++ Code

```
class Solution {
public:
    int lastRemaining(int n) {

    }
};
```

Notes

Random Pick Index

Difficulty: Medium

Link: https://leetcode.com/problems/random-pick-index (https://leetcode.com/problems/random-pick-index)

C++ Code

```
class Solution {
public:
    Solution(vector<int>& nums) {

    }

    int pick(int target) {

    }
};

/**
 * Your Solution object will be instantiated and called as such:
 * Solution* obj = new Solution(nums);
 * int param_1 = obj->pick(target);
 */
```

Notes

Remove K Digits

Difficulty: Medium

Link: https://leetcode.com/problems/remove-k-digits (https://leetcode.com/problems/remove-k-digits)

C++ Code

```
class Solution {
public:
    string removeKdigits(string num, int k) {

    }
};
```

Notes

Partition Equal Subset Sum

Difficulty: Medium

Link: https://leetcode.com/problems/partition-equal-subset-sum (https://leetcode.com/problems/partition-equal-subset-sum)

C++ Code

```
class Solution {
public:
    bool canPartition(vector<int>& nums) {

    }
};
```

Notes

Pacific Atlantic Water Flow

Difficulty: Medium

Link: https://leetcode.com/problems/pacific-atlantic-water-flow (https://leetcode.com/problems/pacific-atlantic-water-flow)

C++ Code

```
class Solution {
public:
    vector<vector<int>> pacificAtlantic(vector<vector<int>>& heights) {

    }
};
```

Notes

Minimum Genetic Mutation

Difficulty: Medium

Link: https://leetcode.com/problems/minimum-genetic-mutation (https://leetcode.com/problems/minimum-genetic-mutation)

C++ Code

```
class Solution {
public:
    int minMutation(string startGene, string endGene, vector<string>& bank) {

    }
};
```

Notes

Non-overlapping Intervals

Difficulty: Medium

Link: https://leetcode.com/problems/non-overlapping-intervals (https://leetcode.com/problems/non-overlapping-intervals)

C++ Code

```
class Solution {
public:
    int eraseOverlapIntervals(vector<vector<int>>& intervals) {

    }
};
```

Notes

Find Right Interval

Difficulty: Medium

Link: https://leetcode.com/problems/find-right-interval (https://leetcode.com/problems/find-right-interval)

C++ Code

```
class Solution {
public:
    vector<int> findRightInterval(vector<vector<int>>& intervals) {

    }
};
```

Notes

Path Sum III

Difficulty: Medium

Link: https://leetcode.com/problems/path-sum-iii (https://leetcode.com/problems/path-sum-iii)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int pathSum(TreeNode* root, int targetSum) {

    }
};
```

Notes

Find All Duplicates in an Array

Difficulty: Medium

Link: https://leetcode.com/problems/find-all-duplicates-in-an-array (https://leetcode.com/problems/find-all-duplicates-in-an-array)

C++ Code

```
class Solution {
public:
    vector<int> findDuplicates(vector<int>& nums) {

    }
};
```

Notes

String Compression

Difficulty: Medium

Link: https://leetcode.com/problems/string-compression (https://leetcode.com/problems/string-compression)

C++ Code

```
class Solution {
public:
    int compress(vector<char>& chars) {

    }
};
```

Notes

Add Two Numbers II

Difficulty: Medium

Link: https://leetcode.com/problems/add-two-numbers-ii (https://leetcode.com/problems/add-two-numbers-ii)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {

    }
};
```

Notes

Ones and Zeroes

Difficulty: Medium

Link: https://leetcode.com/problems/ones-and-zeroes (https://leetcode.com/problems/ones-and-zeroes)

C++ Code

```
class Solution {
public:
    int findMaxForm(vector<string>& strs, int m, int n) {

    }
};
```

Notes

Target Sum

Difficulty: Medium

Link: https://leetcode.com/problems/target-sum (https://leetcode.com/problems/target-sum)

C++ Code

```
class Solution {
public:
    int findTargetSumWays(vector<int>& nums, int target) {

    }
};
```

Notes

Next Greater Element II

Difficulty: Medium

Link: https://leetcode.com/problems/next-greater-element-ii (https://leetcode.com/problems/next-greater-element-ii)

C++ Code

```
class Solution {
public:
    vector<int> nextGreaterElements(vector<int>& nums) {

    }
};
```

Notes

Find Largest Value in Each Tree Row

Difficulty: Medium

Link: https://leetcode.com/problems/find-largest-value-in-each-tree-row (https://leetcode.com/problems/find-largest-value-in-each-tree-row)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<int> largestValues(TreeNode* root) {

    }
};
```

Notes

Longest Palindromic Subsequence

Difficulty: Medium

Link: https://leetcode.com/problems/longest-palindromic-subsequence (https://leetcode.com/problems/longest-palindromic-subsequence)

C++ Code

```
class Solution {
public:
    int longestPalindromeSubseq(string s) {

    }
};
```

Notes

Minimum Time Difference

Difficulty: Medium

Link: https://leetcode.com/problems/minimum-time-difference (https://leetcode.com/problems/minimum-time-difference)

C++ Code

```
class Solution {
public:
    int findMinDifference(vector<string>& timePoints) {

    }
};
```

Notes

Single Element in a Sorted Array

Difficulty: Medium

Link: https://leetcode.com/problems/single-element-in-a-sorted-array (https://leetcode.com/problems/single-element-in-a-sorted-array)

C++ Code

```
class Solution {
public:
    int singleNonDuplicate(vector<int>& nums) {

    }
};
```

Notes

---

01 Matrix

Difficulty: Medium

Link: https://leetcode.com/problems/01-matrix (https://leetcode.com/problems/01-matrix)

C++ Code

```
class Solution {
public:
    vector<vector<int>> updateMatrix(vector<vector<int>>& mat) {

    }
};
```

Notes

Number of Provinces

Difficulty: Medium

Link: https://leetcode.com/problems/number-of-provinces (https://leetcode.com/problems/number-of-provinces)

C++ Code

```
class Solution {
public:
    int findCircleNum(vector<vector<int>>& isConnected) {

    }
};
```

Notes

Subarray Sum Equals K

Difficulty: Medium

Link: https://leetcode.com/problems/subarray-sum-equals-k (https://leetcode.com/problems/subarray-sum-equals-k)

C++ Code

```
class Solution {
public:
    int subarraySum(vector<int>& nums, int k) {

    }
};
```

Notes

Permutation in String

Difficulty: Medium

Link: https://leetcode.com/problems/permutation-in-string (https://leetcode.com/problems/permutation-in-string)

C++ Code

```
class Solution {
public:
    bool checkInclusion(string s1, string s2) {

    }
};
```

Notes

Maximum Length of Pair Chain

Difficulty: Medium

Link: https://leetcode.com/problems/maximum-length-of-pair-chain (https://leetcode.com/problems/maximum-length-of-pair-chain)

C++ Code

```
class Solution {
public:
    int findLongestChain(vector<vector<int>>& pairs) {

    }
};
```

Notes

Palindromic Substrings

Difficulty: Medium

Link: https://leetcode.com/problems/palindromic-substrings (https://leetcode.com/problems/palindromic-substrings)

C++ Code

```
class Solution {
public:
    int countSubstrings(string s) {

    }
};
```

Notes

---

2 Keys Keyboard

Difficulty: Medium

Link: https://leetcode.com/problems/2-keys-keyboard (https://leetcode.com/problems/2-keys-keyboard)

C++ Code

```
class Solution {
public:
    int minSteps(int n) {

    }
};
```

Notes

Maximum Binary Tree

Difficulty: Medium

Link: https://leetcode.com/problems/maximum-binary-tree (https://leetcode.com/problems/maximum-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* constructMaximumBinaryTree(vector<int>& nums) {

    }
};
```

Notes

Print Binary Tree

Difficulty: Medium

Link: https://leetcode.com/problems/print-binary-tree (https://leetcode.com/problems/print-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<vector<string>> printTree(TreeNode* root) {

    }
};
```

Notes

Find K Closest Elements

Difficulty: Medium

Link: https://leetcode.com/problems/find-k-closest-elements (https://leetcode.com/problems/find-k-closest-elements)

C++ Code

```
class Solution {
public:
    vector<int> findClosestElements(vector<int>& arr, int k, int x) {

    }
};
```

Notes

Maximum Width of Binary Tree

Difficulty: Medium

Link: https://leetcode.com/problems/maximum-width-of-binary-tree (https://leetcode.com/problems/maximum-width-of-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int widthOfBinaryTree(TreeNode* root) {

    }
};
```

Notes

Longest Univalue Path

Difficulty: Medium

Link: https://leetcode.com/problems/longest-univalue-path (https://leetcode.com/problems/longest-univalue-path)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int longestUnivaluePath(TreeNode* root) {

    }
};
```

Notes

Max Area of Island

Difficulty: Medium

Link: https://leetcode.com/problems/max-area-of-island (https://leetcode.com/problems/max-area-of-island)

C++ Code

```
class Solution {
public:
    int maxAreaOfIsland(vector<vector<int>>& grid) {

    }
};
```

Notes

Insert into a Binary Search Tree

Difficulty: Medium

Link: https://leetcode.com/problems/insert-into-a-binary-search-tree (https://leetcode.com/problems/insert-into-a-binary-search-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* insertIntoBST(TreeNode* root, int val) {

    }
};
```

Notes

Search in a Sorted Array of Unknown Size

Difficulty: Medium

Link: https://leetcode.com/problems/search-in-a-sorted-array-of-unknown-size (https://leetcode.com/problems/search-in-a-sorted-array-of-unknown-size)

C++ Code

```

```

Notes

Design Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/design-linked-list (https://leetcode.com/problems/design-linked-list)

C++ Code

```
class MyLinkedList {
public:
    MyLinkedList() {

    }

    int get(int index) {

    }

    void addAtHead(int val) {

    }

    void addAtTail(int val) {

    }

    void addAtIndex(int index, int val) {

    }

    void deleteAtIndex(int index) {

    }
};

/**
 * Your MyLinkedList object will be instantiated and called as such:
 * MyLinkedList* obj = new MyLinkedList();
 * int param_1 = obj->get(index);
 * obj->addAtHead(val);
 * obj->addAtTail(val);
 * obj->addAtIndex(index,val);
 * obj->deleteAtIndex(index);
 */
```

Notes

Subarray Product Less Than K

Difficulty: Medium

Link: https://leetcode.com/problems/subarray-product-less-than-k (https://leetcode.com/problems/subarray-product-less-than-k)

C++ Code

```
class Solution {
public:
    int numSubarrayProductLessThanK(vector<int>& nums, int k) {

    }
};
```

Notes

Split Linked List in Parts

Difficulty: Medium

Link: https://leetcode.com/problems/split-linked-list-in-parts (https://leetcode.com/problems/split-linked-list-in-parts)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    vector<ListNode*> splitListToParts(ListNode* head, int k) {

    }
};
```

Notes

Daily Temperatures

Difficulty: Medium

Link: https://leetcode.com/problems/daily-temperatures (https://leetcode.com/problems/daily-temperatures)

C++ Code

```
class Solution {
public:
    vector<int> dailyTemperatures(vector<int>& temperatures) {

    }
};
```

Notes

Delete and Earn

Difficulty: Medium

Link: https://leetcode.com/problems/delete-and-earn (https://leetcode.com/problems/delete-and-earn)

C++ Code

```
class Solution {
public:
    int deleteAndEarn(vector<int>& nums) {

    }
};
```

Notes

Max Chunks To Make Sorted

Difficulty: Medium

Link: https://leetcode.com/problems/max-chunks-to-make-sorted (https://leetcode.com/problems/max-chunks-to-make-sorted)

C++ Code

```
class Solution {
public:
    int maxChunksToSorted(vector<int>& arr) {

    }
};
```

Notes

Letter Case Permutation

Difficulty: Medium

Link: https://leetcode.com/problems/letter-case-permutation (https://leetcode.com/problems/letter-case-permutation)

C++ Code

```
class Solution {
public:
    vector<string> letterCasePermutation(string s) {

    }
};
```

Notes

Is Graph Bipartite?

Difficulty: Medium

Link: https://leetcode.com/problems/is-graph-bipartite (https://leetcode.com/problems/is-graph-bipartite)

C++ Code

```
class Solution {
public:
    bool isBipartite(vector<vector<int>>& graph) {

    }
};
```

Notes

Rotated Digits

Difficulty: Medium

Link: https://leetcode.com/problems/rotated-digits (https://leetcode.com/problems/rotated-digits)

C++ Code

```
class Solution {
public:
    int rotatedDigits(int n) {

    }
};
```

Notes

Domino and Tromino Tiling

Difficulty: Medium

Link: https://leetcode.com/problems/domino-and-tromino-tiling (https://leetcode.com/problems/domino-and-tromino-tiling)

C++ Code

```
class Solution {
public:
    int numTilings(int n) {

    }
};
```

Notes

All Paths From Source to Target

Difficulty: Medium

Link: https://leetcode.com/problems/all-paths-from-source-to-target (https://leetcode.com/problems/all-paths-from-source-to-target)

C++ Code

```
class Solution {
public:
    vector<vector<int>> allPathsSourceTarget(vector<vector<int>>& graph) {

    }
};
```

Notes

Find Eventual Safe States

Difficulty: Medium

Link: https://leetcode.com/problems/find-eventual-safe-states (https://leetcode.com/problems/find-eventual-safe-states)

C++ Code

```
class Solution {
public:
    vector<int> eventualSafeNodes(vector<vector<int>>& graph) {

    }
};
```

Notes

Binary Tree Pruning

Difficulty: Medium

Link: https://leetcode.com/problems/binary-tree-pruning (https://leetcode.com/problems/binary-tree-pruning)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* pruneTree(TreeNode* root) {

    }
};
```

Notes

Keys and Rooms

Difficulty: Medium

Link: https://leetcode.com/problems/keys-and-rooms (https://leetcode.com/problems/keys-and-rooms)

C++ Code

```
class Solution {
public:
    bool canVisitAllRooms(vector<vector<int>>& rooms) {

    }
};
```

Notes

Maximize Distance to Closest Person

Difficulty: Medium

Link: https://leetcode.com/problems/maximize-distance-to-closest-person (https://leetcode.com/problems/maximize-distance-to-closest-person)

C++ Code

```
class Solution {
public:
    int maxDistToClosest(vector<int>& seats) {

    }
};
```

Notes

Peak Index in a Mountain Array

Difficulty: Medium

Link: https://leetcode.com/problems/peak-index-in-a-mountain-array (https://leetcode.com/problems/peak-index-in-a-mountain-array)

C++ Code

```
class Solution {
public:
    int peakIndexInMountainArray(vector<int>& arr) {

    }
};
```

Notes

Score After Flipping Matrix

Difficulty: Medium

Link: https://leetcode.com/problems/score-after-flipping-matrix (https://leetcode.com/problems/score-after-flipping-matrix)

C++ Code

```
class Solution {
public:
    int matrixScore(vector<vector<int>>& grid) {

    }
};
```

Notes

All Nodes Distance K in Binary Tree

Difficulty: Medium

Link: https://leetcode.com/problems/all-nodes-distance-k-in-binary-tree (https://leetcode.com/problems/all-nodes-distance-k-in-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    vector<int> distanceK(TreeNode* root, TreeNode* target, int k) {

    }
};
```

Notes

Walking Robot Simulation

Difficulty: Medium

Link: https://leetcode.com/problems/walking-robot-simulation (https://leetcode.com/problems/walking-robot-simulation)

C++ Code

```
class Solution {
public:
    int robotSim(vector<int>& commands, vector<vector<int>>& obstacles) {

    }
};
```

Notes

Boats to Save People

Difficulty: Medium

Link: https://leetcode.com/problems/boats-to-save-people (https://leetcode.com/problems/boats-to-save-people)

C++ Code

```
class Solution {
public:
    int numRescueBoats(vector<int>& people, int limit) {

    }
};
```

Notes

Find and Replace Pattern

Difficulty: Medium

Link: https://leetcode.com/problems/find-and-replace-pattern (https://leetcode.com/problems/find-and-replace-pattern)

C++ Code

```
class Solution {
public:
    vector<string> findAndReplacePattern(vector<string>& words, string pattern) {

    }
};
```

Notes

Fruit Into Baskets

Difficulty: Medium

Link: https://leetcode.com/problems/fruit-into-baskets (https://leetcode.com/problems/fruit-into-baskets)

C++ Code

```
class Solution {
public:
    int totalFruit(vector<int>& fruits) {

    }
};
```

Notes

Minimum Falling Path Sum

Difficulty: Medium

Link: https://leetcode.com/problems/minimum-falling-path-sum (https://leetcode.com/problems/minimum-falling-path-sum)

C++ Code

```
class Solution {
public:
    int minFallingPathSum(vector<vector<int>>& matrix) {

    }
};
```

Notes

Shortest Bridge

Difficulty: Medium

Link: https://leetcode.com/problems/shortest-bridge (https://leetcode.com/problems/shortest-bridge)

C++ Code

```
class Solution {
public:
    int shortestBridge(vector<vector<int>>& grid) {

    }
};
```

Notes

K Closest Points to Origin

Difficulty: Medium

Link: https://leetcode.com/problems/k-closest-points-to-origin (https://leetcode.com/problems/k-closest-points-to-origin)

C++ Code

```
class Solution {
public:
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {

    }
};
```

Notes

Distribute Coins in Binary Tree

Difficulty: Medium

Link: https://leetcode.com/problems/distribute-coins-in-binary-tree (https://leetcode.com/problems/distribute-coins-in-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int distributeCoins(TreeNode* root) {

    }
};
```

Notes

Time Based Key-Value Store

Difficulty: Medium

Link: https://leetcode.com/problems/time-based-key-value-store (https://leetcode.com/problems/time-based-key-value-store)

C++ Code

```
class TimeMap {
public:
    TimeMap() {

    }

    void set(string key, string value, int timestamp) {

    }

    string get(string key, int timestamp) {

    }
};

/**
 * Your TimeMap object will be instantiated and called as such:
 * TimeMap* obj = new TimeMap();
 * obj->set(key,value,timestamp);
 * string param_2 = obj->get(key,timestamp);
 */
```

Notes

Minimum Cost For Tickets

Difficulty: Medium

Link: https://leetcode.com/problems/minimum-cost-for-tickets (https://leetcode.com/problems/minimum-cost-for-tickets)

C++ Code

```
class Solution {
public:
    int mincostTickets(vector<int>& days, vector<int>& costs) {

    }
};
```

Notes

String Without AAA or BBB

Difficulty: Medium

Link: https://leetcode.com/problems/string-without-aaa-or-bbb (https://leetcode.com/problems/string-without-aaa-or-bbb)

C++ Code

```
class Solution {
public:
    string strWithout3a3b(int a, int b) {

    }
};
```

Notes

Interval List Intersections

Difficulty: Medium

Link: https://leetcode.com/problems/interval-list-intersections (https://leetcode.com/problems/interval-list-intersections)

C++ Code

```
class Solution {
public:
    vector<vector<int>> intervalIntersection(vector<vector<int>>& firstList, vector<vector<int>>& secondList) {

    }
};
```

Notes

Satisfiability of Equality Equations

Difficulty: Medium

Link: https://leetcode.com/problems/satisfiability-of-equality-equations (https://leetcode.com/problems/satisfiability-of-equality-equations)

C++ Code

```
class Solution {
public:
    bool equationsPossible(vector<string>& equations) {

    }
};
```

Notes

Rotting Oranges

Difficulty: Medium

Link: https://leetcode.com/problems/rotting-oranges (https://leetcode.com/problems/rotting-oranges)

C++ Code

```
class Solution {
public:
    int orangesRotting(vector<vector<int>>& grid) {

    }
};
```

Notes

Max Consecutive Ones III

Difficulty: Medium

Link: https://leetcode.com/problems/max-consecutive-ones-iii (https://leetcode.com/problems/max-consecutive-ones-iii)

C++ Code

```
class Solution {
public:
    int longestOnes(vector<int>& nums, int k) {

    }
};
```

Notes

Minimum Domino Rotations For Equal Row

Difficulty: Medium

Link: https://leetcode.com/problems/minimum-domino-rotations-for-equal-row (https://leetcode.com/problems/minimum-domino-rotations-for-equal-row)

C++ Code

```
class Solution {
public:
    int minDominoRotations(vector<int>& tops, vector<int>& bottoms) {

    }
};
```

Notes

Construct Binary Search Tree from Preorder Traversal

Difficulty: Medium

Link: https://leetcode.com/problems/construct-binary-search-tree-from-preorder-traversal (https://leetcode.com/problems/construct-binary-search-tree-from-preorder-traversal)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* bstFromPreorder(vector<int>& preorder) {

    }
};
```

Notes

Next Greater Node In Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/next-greater-node-in-linked-list (https://leetcode.com/problems/next-greater-node-in-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    vector<int> nextLargerNodes(ListNode* head) {

    }
};
```

Notes

Number of Enclaves

Difficulty: Medium

Link: https://leetcode.com/problems/number-of-enclaves (https://leetcode.com/problems/number-of-enclaves)

C++ Code

```
class Solution {
public:
    int numEnclaves(vector<vector<int>>& grid) {

    }
};
```

Notes

Longest Arithmetic Subsequence

Difficulty: Medium

Link: https://leetcode.com/problems/longest-arithmetic-subsequence (https://leetcode.com/problems/longest-arithmetic-subsequence)

C++ Code

```
class Solution {
public:
    int longestArithSeqLength(vector<int>& nums) {

    }
};
```

Notes

Uncrossed Lines

Difficulty: Medium

Link: https://leetcode.com/problems/uncrossed-lines (https://leetcode.com/problems/uncrossed-lines)

C++ Code

```
class Solution {
public:
    int maxUncrossedLines(vector<int>& nums1, vector<int>& nums2) {

    }
};
```

Notes

Robot Bounded In Circle

Difficulty: Medium

Link: https://leetcode.com/problems/robot-bounded-in-circle (https://leetcode.com/problems/robot-bounded-in-circle)

C++ Code

```
class Solution {
public:
    bool isRobotBounded(string instructions) {

    }
};
```

Notes

The Earliest Moment When Everyone Become Friends

Difficulty: Medium

Link: https://leetcode.com/problems/the-earliest-moment-when-everyone-become-friends (https://leetcode.com/problems/the-earliest-moment-when-everyone-become-friends)

C++ Code

```

```

Notes

Longest Common Subsequence

Difficulty: Medium

Link: https://leetcode.com/problems/longest-common-subsequence (https://leetcode.com/problems/longest-common-subsequence)

C++ Code

```
class Solution {
public:
    int longestCommonSubsequence(string text1, string text2) {

    }
};
```

Notes

Snapshot Array

Difficulty: Medium

Link: https://leetcode.com/problems/snapshot-array (https://leetcode.com/problems/snapshot-array)

C++ Code

```
class SnapshotArray {
public:
    SnapshotArray(int length) {

    }

    void set(int index, int val) {

    }

    int snap() {

    }

    int get(int index, int snap_id) {

    }
};

/**
 * Your SnapshotArray object will be instantiated and called as such:
 * SnapshotArray* obj = new SnapshotArray(length);
 * obj->set(index,val);
 * int param_2 = obj->snap();
 * int param_3 = obj->get(index,snap_id);
 */
```

Notes

Longest Arithmetic Subsequence of Given Difference

Difficulty: Medium

Link: https://leetcode.com/problems/longest-arithmetic-subsequence-of-given-difference (https://leetcode.com/problems/longest-arithmetic-subsequence-of-given-difference)

C++ Code

```
class Solution {
public:
    int longestSubsequence(vector<int>& arr, int difference) {

    }
};
```

Notes

Path with Maximum Gold

Difficulty: Medium

Link: https://leetcode.com/problems/path-with-maximum-gold (https://leetcode.com/problems/path-with-maximum-gold)

C++ Code

```
class Solution {
public:
    int getMaximumGold(vector<vector<int>>& grid) {

    }
};
```

Notes

Number of Closed Islands

Difficulty: Medium

Link: https://leetcode.com/problems/number-of-closed-islands (https://leetcode.com/problems/number-of-closed-islands)

C++ Code

```
class Solution {
public:
    int closedIsland(vector<vector<int>>& grid) {

    }
};
```

Notes

XOR Queries of a Subarray

Difficulty: Medium

Link: https://leetcode.com/problems/xor-queries-of-a-subarray (https://leetcode.com/problems/xor-queries-of-a-subarray)

C++ Code

```
class Solution {
public:
    vector<int> xorQueries(vector<int>& arr, vector<vector<int>>& queries) {

    }
};
```

Notes

Delete Leaves With a Given Value

Difficulty: Medium

Link: https://leetcode.com/problems/delete-leaves-with-a-given-value (https://leetcode.com/problems/delete-leaves-with-a-given-value)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* removeLeafNodes(TreeNode* root, int target) {

    }
};
```

Notes

Time Needed to Inform All Employees

Difficulty: Medium

Link: https://leetcode.com/problems/time-needed-to-inform-all-employees (https://leetcode.com/problems/time-needed-to-inform-all-employees)

C++ Code

```
class Solution {
public:
    int numOfMinutes(int n, int headID, vector<int>& manager, vector<int>& informTime) {

    }
};
```

Notes

The k-th Lexicographical String of All Happy Strings of Length n

Difficulty: Medium

Link: https://leetcode.com/problems/the-k-th-lexicographical-string-of-all-happy-strings-of-length-n (https://leetcode.com/problems/the-k-th-lexicographical-string-of-all-happy-strings-of-length-n)

C++ Code

```
class Solution {
public:
    string getHappyString(int n, int k) {

    }
};
```

Notes

Count Good Nodes in Binary Tree

Difficulty: Medium

Link: https://leetcode.com/problems/count-good-nodes-in-binary-tree (https://leetcode.com/problems/count-good-nodes-in-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int goodNodes(TreeNode* root) {

    }
};
```

Notes

Maximum Number of Vowels in a Substring of Given Length

Difficulty: Medium

Link: https://leetcode.com/problems/maximum-number-of-vowels-in-a-substring-of-given-length (https://leetcode.com/problems/maximum-number-of-vowels-in-a-substring-of-given-length)

C++ Code

```
class Solution {
public:
    int maxVowels(string s, int k) {

    }
};
```

Notes

Longest Subarray of 1's After Deleting One Element

Difficulty: Medium

Link: https://leetcode.com/problems/longest-subarray-of-1s-after-deleting-one-element (https://leetcode.com/problems/longest-subarray-of-1s-after-deleting-one-element)

C++ Code

```
class Solution {
public:
    int longestSubarray(vector<int>& nums) {

    }
};
```

Notes

Range Sum of Sorted Subarray Sums

Difficulty: Medium

Link: https://leetcode.com/problems/range-sum-of-sorted-subarray-sums (https://leetcode.com/problems/range-sum-of-sorted-subarray-sums)

C++ Code

```
class Solution {
public:
    int rangeSum(vector<int>& nums, int n, int left, int right) {

    }
};
```

Notes

Maximum Number of Non-Overlapping Subarrays With Sum Equals Target

Difficulty: Medium

Link: https://leetcode.com/problems/maximum-number-of-non-overlapping-subarrays-with-sum-equals-target (https://leetcode.com/problems/maximum-number-of-non-overlapping-subarrays-with-sum-equals-target)

C++ Code

```
class Solution {
public:
    int maxNonOverlapping(vector<int>& nums, int target) {

    }
};
```

Notes

Minimum Time to Make Rope Colorful

Difficulty: Medium

Link: https://leetcode.com/problems/minimum-time-to-make-rope-colorful (https://leetcode.com/problems/minimum-time-to-make-rope-colorful)

C++ Code

```
class Solution {
public:
    int minCost(string colors, vector<int>& neededTime) {

    }
};
```

Notes

Minimum Deletions to Make String Balanced

Difficulty: Medium

Link: https://leetcode.com/problems/minimum-deletions-to-make-string-balanced (https://leetcode.com/problems/minimum-deletions-to-make-string-balanced)

C++ Code

```
class Solution {
public:
    int minimumDeletions(string s) {

    }
};
```

Notes

Determine if Two Strings Are Close

Difficulty: Medium

Link: https://leetcode.com/problems/determine-if-two-strings-are-close (https://leetcode.com/problems/determine-if-two-strings-are-close)

C++ Code

```
class Solution {
public:
    bool closeStrings(string word1, string word2) {

    }
};
```

Notes

Correct a Binary Tree

Difficulty: Medium

Link: https://leetcode.com/problems/correct-a-binary-tree (https://leetcode.com/problems/correct-a-binary-tree)

C++ Code

```

```

Notes

Max Number of K-Sum Pairs

Difficulty: Medium

Link: https://leetcode.com/problems/max-number-of-k-sum-pairs (https://leetcode.com/problems/max-number-of-k-sum-pairs)

C++ Code

```
class Solution {
public:
    int maxOperations(vector<int>& nums, int k) {

    }
};
```

Notes

Swapping Nodes in a Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/swapping-nodes-in-a-linked-list (https://leetcode.com/problems/swapping-nodes-in-a-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* swapNodes(ListNode* head, int k) {

    }
};
```

Notes

Distinct Numbers in Each Subarray

Difficulty: Medium

Link: https://leetcode.com/problems/distinct-numbers-in-each-subarray (https://leetcode.com/problems/distinct-numbers-in-each-subarray)

C++ Code

```
```

Notes

Find the Student that Will Replace the Chalk

Difficulty: Medium

Link: https://leetcode.com/problems/find-the-student-that-will-replace-the-chalk (https://leetcode.com/problems/find-the-student-that-will-replace-the-chalk)

C++ Code

```
class Solution {
public:
    int chalkReplacer(vector<int>& chalk, int k) {

    }
};
```

Notes

Count Sub Islands

Difficulty: Medium

Link: https://leetcode.com/problems/count-sub-islands (https://leetcode.com/problems/count-sub-islands)

C++ Code

```
class Solution {
public:
    int countSubIslands(vector<vector<int>>& grid1, vector<vector<int>>& grid2) {

    }
};
```

Notes

Nearest Exit from Entrance in Maze

Difficulty: Medium

Link: https://leetcode.com/problems/nearest-exit-from-entrance-in-maze (https://leetcode.com/problems/nearest-exit-from-entrance-in-maze)

C++ Code

```
class Solution {
public:
    int nearestExit(vector<vector<char>>& maze, vector<int>& entrance) {

    }
};
```

Notes

Maximum Number of Points with Cost

Difficulty: Medium

Link: https://leetcode.com/problems/maximum-number-of-points-with-cost (https://leetcode.com/problems/maximum-number-of-points-with-cost)

C++ Code

```
class Solution {
public:
    long long maxPoints(vector<vector<int>>& points) {

    }
};
```

Notes

Find Missing Observations

Difficulty: Medium

Link: https://leetcode.com/problems/find-missing-observations (https://leetcode.com/problems/find-missing-observations)

C++ Code

```
class Solution {
public:
    vector<int> missingRolls(vector<int>& rolls, int mean, int n) {

    }
};
```

Notes

Delete the Middle Node of a Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list (https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {

    }
};
```

Notes

Maximum Twin Sum of a Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/maximum-twin-sum-of-a-linked-list (https://leetcode.com/problems/maximum-twin-sum-of-a-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    int pairSum(ListNode* head) {

    }
};
```

Notes

Solving Questions With Brainpower

Difficulty: Medium

Link: https://leetcode.com/problems/solving-questions-with-brainpower (https://leetcode.com/problems/solving-questions-with-brainpower)

C++ Code

```
class Solution {
public:
    long long mostPoints(vector<vector<int>>& questions) {

    }
};
```

Notes

Coin Change II

Difficulty: Medium

Link: https://leetcode.com/problems/coin-change-ii (https://leetcode.com/problems/coin-change-ii)

C++ Code

```
class Solution {
public:
    int change(int amount, vector<int>& coins) {

    }
};
```

Notes

Equal Row and Column Pairs

Difficulty: Medium

Link: https://leetcode.com/problems/equal-row-and-column-pairs (https://leetcode.com/problems/equal-row-and-column-pairs)

C++ Code

```
class Solution {
public:
    int equalPairs(vector<vector<int>>& grid) {

    }
};
```

Notes

Remove Nodes From Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/remove-nodes-from-linked-list (https://leetcode.com/problems/remove-nodes-from-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* removeNodes(ListNode* head) {

    }
};
```

Notes

Minimum Operations to Make All Array Elements Equal

Difficulty: Medium

Link: https://leetcode.com/problems/minimum-operations-to-make-all-array-elements-equal (https://leetcode.com/problems/minimum-operations-to-make-all-array-elements-equal)

C++ Code

```
class Solution {
public:
    vector<long long> minOperations(vector<int>& nums, vector<int>& queries) {

    }
};
```

Notes

Insert Greatest Common Divisors in Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/insert-greatest-common-divisors-in-linked-list (https://leetcode.com/problems/insert-greatest-common-divisors-in-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* insertGreatestCommonDivisors(ListNode* head) {

    }
};
```

Notes

Double a Number Represented as a Linked List

Difficulty: Medium

Link: https://leetcode.com/problems/double-a-number-represented-as-a-linked-list (https://leetcode.com/problems/double-a-number-represented-as-a-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* doubleIt(ListNode* head) {

    }
};
```

Notes

Find the Length of the Longest Common Prefix

Difficulty: Medium

Link: https://leetcode.com/problems/find-the-length-of-the-longest-common-prefix (https://leetcode.com/problems/find-the-length-of-the-longest-common-prefix)

C++ Code

```
class Solution {
public:
    int longestCommonPrefix(vector<int>& arr1, vector<int>& arr2) {

    }
};
```

Notes

Maximize Happiness of Selected Children

Difficulty: Medium

Link: https://leetcode.com/problems/maximize-happiness-of-selected-children (https://leetcode.com/problems/maximize-happiness-of-selected-children)

C++ Code

```
class Solution {
public:
    long long maximumHappinessSum(vector<int>& happiness, int k) {

    }
};
```

Notes

Right Triangles

Difficulty: Medium

Link: https://leetcode.com/problems/right-triangles (https://leetcode.com/problems/right-triangles)

C++ Code

```
class Solution {
public:
    long long numberOfRightTriangles(vector<vector<int>>& grid) {

    }
};
```

Notes

Delete Nodes From Linked List Present in Array

Difficulty: Medium

Link: https://leetcode.com/problems/delete-nodes-from-linked-list-present-in-array (https://leetcode.com/problems/delete-nodes-from-linked-list-present-in-array)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* modifiedList(vector<int>& nums, ListNode* head) {

    }
};
```

Notes

Two Sum

Difficulty: Easy

Link: https://leetcode.com/problems/two-sum (https://leetcode.com/problems/two-sum)

C++ Code

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {

    }
};
```

Notes

Palindrome Number

Difficulty: Easy

Link: https://leetcode.com/problems/palindrome-number (https://leetcode.com/problems/palindrome-number)

C++ Code

```
class Solution {
public:
    bool isPalindrome(int x) {

    }
};
```

Notes

Roman to Integer

Difficulty: Easy

Link: https://leetcode.com/problems/roman-to-integer (https://leetcode.com/problems/roman-to-integer)

C++ Code

```
class Solution {
public:
    int romanToInt(string s) {

    }
};
```

Notes

Longest Common Prefix

Difficulty: Easy

Link: https://leetcode.com/problems/longest-common-prefix (https://leetcode.com/problems/longest-common-prefix)

C++ Code

```
class Solution {
public:
    string longestCommonPrefix(vector<string>& strs) {

    }
};
```

Notes

Valid Parentheses

Difficulty: Easy

Link: https://leetcode.com/problems/valid-parentheses (https://leetcode.com/problems/valid-parentheses)

C++ Code

```
class Solution {
public:
    bool isValid(string s) {

    }
};
```

Notes

Merge Two Sorted Lists

Difficulty: Easy

Link: https://leetcode.com/problems/merge-two-sorted-lists (https://leetcode.com/problems/merge-two-sorted-lists)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {

    }
};
```

Notes

Remove Duplicates from Sorted Array

Difficulty: Easy

Link: https://leetcode.com/problems/remove-duplicates-from-sorted-array (https://leetcode.com/problems/remove-duplicates-from-sorted-array)

C++ Code

```
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {

    }
};
```

Notes

Remove Element

Difficulty: Easy

Link: https://leetcode.com/problems/remove-element (https://leetcode.com/problems/remove-element)

C++ Code

```
class Solution {
public:
    int removeElement(vector<int>& nums, int val) {

    }
};
```

Notes

Search Insert Position

Difficulty: Easy

Link: https://leetcode.com/problems/search-insert-position (https://leetcode.com/problems/search-insert-position)

C++ Code

```
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {

    }
};
```

Notes

Length of Last Word

Difficulty: Easy

Link: https://leetcode.com/problems/length-of-last-word (https://leetcode.com/problems/length-of-last-word)

C++ Code

```
class Solution {
public:
    int lengthOfLastWord(string s) {

    }
};
```

Notes

Plus One

Difficulty: Easy

Link: https://leetcode.com/problems/plus-one (https://leetcode.com/problems/plus-one)

C++ Code

```
class Solution {
public:
    vector<int> plusOne(vector<int>& digits) {

    }
};
```

Notes

Add Binary

Difficulty: Easy

Link: https://leetcode.com/problems/add-binary (https://leetcode.com/problems/add-binary)

C++ Code

```
class Solution {
public:
    string addBinary(string a, string b) {

    }
};
```

Notes

Sqrt(x)

Difficulty: Easy

Link: https://leetcode.com/problems/sqrtx (https://leetcode.com/problems/sqrtx)

C++ Code

```
class Solution {
public:
    int mySqrt(int x) {

    }
};
```

Notes

Climbing Stairs

Difficulty: Easy

Link: https://leetcode.com/problems/climbing-stairs (https://leetcode.com/problems/climbing-stairs)

C++ Code

```
class Solution {
public:
    int climbStairs(int n) {

    }
};
```

Notes

Remove Duplicates from Sorted List

Difficulty: Easy

Link: https://leetcode.com/problems/remove-duplicates-from-sorted-list (https://leetcode.com/problems/remove-duplicates-from-sorted-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {

    }
};
```

Notes

Merge Sorted Array

Difficulty: Easy

Link: https://leetcode.com/problems/merge-sorted-array (https://leetcode.com/problems/merge-sorted-array)

C++ Code

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {

    }
};
```

Notes

Binary Tree Inorder Traversal

Difficulty: Easy

Link: https://leetcode.com/problems/binary-tree-inorder-traversal (https://leetcode.com/problems/binary-tree-inorder-traversal)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root) {

    }
};
```

Notes

Same Tree

Difficulty: Easy

Link: https://leetcode.com/problems/same-tree (https://leetcode.com/problems/same-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool isSameTree(TreeNode* p, TreeNode* q) {

    }
};
```

Notes

Symmetric Tree

Difficulty: Easy

Link: https://leetcode.com/problems/symmetric-tree (https://leetcode.com/problems/symmetric-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool isSymmetric(TreeNode* root) {


    }
};
```

Notes

Maximum Depth of Binary Tree

Difficulty: Easy

Link: https://leetcode.com/problems/maximum-depth-of-binary-tree (https://leetcode.com/problems/maximum-depth-of-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int maxDepth(TreeNode* root) {

    }
};
```

Notes

Convert Sorted Array to Binary Search Tree

Difficulty: Easy

Link: https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree (https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {

    }
};
```

Notes

Balanced Binary Tree

Difficulty: Easy

Link: https://leetcode.com/problems/balanced-binary-tree (https://leetcode.com/problems/balanced-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool isBalanced(TreeNode* root) {


    }
};
```

Notes

Minimum Depth of Binary Tree

Difficulty: Easy

Link: https://leetcode.com/problems/minimum-depth-of-binary-tree (https://leetcode.com/problems/minimum-depth-of-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int minDepth(TreeNode* root) {

    }
};
```

Notes

Path Sum

Difficulty: Easy

Link: https://leetcode.com/problems/path-sum (https://leetcode.com/problems/path-sum)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool hasPathSum(TreeNode* root, int targetSum) {

    }
};
```

Notes

Pascal's Triangle

Difficulty: Easy

Link: https://leetcode.com/problems/pascals-triangle (https://leetcode.com/problems/pascals-triangle)

C++ Code

```
class Solution {
public:
    vector<vector<int>> generate(int numRows) {

    }
};
```

Notes

Pascal's Triangle II

Difficulty: Easy

Link: https://leetcode.com/problems/pascals-triangle-ii (https://leetcode.com/problems/pascals-triangle-ii)

C++ Code

```
class Solution {
public:
    vector<int> getRow(int rowIndex) {

    }
};
```

Notes

Best Time to Buy and Sell Stock

Difficulty: Easy

Link: https://leetcode.com/problems/best-time-to-buy-and-sell-stock (https://leetcode.com/problems/best-time-to-buy-and-sell-stock)

C++ Code

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {

    }
};
```

Notes

Valid Palindrome

Difficulty: Easy

Link: https://leetcode.com/problems/valid-palindrome (https://leetcode.com/problems/valid-palindrome)

C++ Code

```
class Solution {
public:
    bool isPalindrome(string s) {

    }
};
```

Notes

Single Number

Difficulty: Easy

Link: https://leetcode.com/problems/single-number (https://leetcode.com/problems/single-number)

C++ Code

```
class Solution {
public:
    int singleNumber(vector<int>& nums) {

    }
};
```

Notes

Linked List Cycle

Difficulty: Easy

Link: https://leetcode.com/problems/linked-list-cycle (https://leetcode.com/problems/linked-list-cycle)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    bool hasCycle(ListNode *head) {

    }
};
```

Notes

Binary Tree Preorder Traversal

Difficulty: Easy

Link: https://leetcode.com/problems/binary-tree-preorder-traversal (https://leetcode.com/problems/binary-tree-preorder-traversal)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<int> preorderTraversal(TreeNode* root) {

    }
};
```

Notes

Binary Tree Postorder Traversal

Difficulty: Easy

Link: https://leetcode.com/problems/binary-tree-postorder-traversal (https://leetcode.com/problems/binary-tree-postorder-traversal)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<int> postorderTraversal(TreeNode* root) {


    }
};
```

Notes

Read N Characters Given Read4

Difficulty: Easy

Link: https://leetcode.com/problems/read-n-characters-given-read4 (https://leetcode.com/problems/read-n-characters-given-read4)

C++ Code

```

```

Notes

Intersection of Two Linked Lists

Difficulty: Easy

Link: https://leetcode.com/problems/intersection-of-two-linked-lists (https://leetcode.com/problems/intersection-of-two-linked-lists)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {

    }
};
```

Notes

Excel Sheet Column Title

Difficulty: Easy

Link: https://leetcode.com/problems/excel-sheet-column-title (https://leetcode.com/problems/excel-sheet-column-title)

C++ Code

```
class Solution {
public:
    string convertToTitle(int columnNumber) {

    }
};
```

Notes

Majority Element

Difficulty: Easy

Link: https://leetcode.com/problems/majority-element (https://leetcode.com/problems/majority-element)

C++ Code

```
class Solution {
public:
    int majorityElement(vector<int>& nums) {

    }
};
```

Notes

Reverse Bits

Difficulty: Easy

Link: https://leetcode.com/problems/reverse-bits (https://leetcode.com/problems/reverse-bits)

C++ Code

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {

    }
};
```

Notes

Number of 1 Bits

Difficulty: Easy

Link: https://leetcode.com/problems/number-of-1-bits (https://leetcode.com/problems/number-of-1-bits)

C++ Code

```
class Solution {
public:
    int hammingWeight(int n) {

    }
};
```

Notes

Happy Number

Difficulty: Easy

Link: https://leetcode.com/problems/happy-number (https://leetcode.com/problems/happy-number)

C++ Code

```
class Solution {
public:
    bool isHappy(int n) {

    }
};
```

Notes

Remove Linked List Elements

Difficulty: Easy

Link: https://leetcode.com/problems/remove-linked-list-elements (https://leetcode.com/problems/remove-linked-list-elements)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* removeElements(ListNode* head, int val) {

    }
};
```

Notes

Isomorphic Strings

Difficulty: Easy

Link: https://leetcode.com/problems/isomorphic-strings (https://leetcode.com/problems/isomorphic-strings)

C++ Code

```
class Solution {
public:
    bool isIsomorphic(string s, string t) {

    }
};
```

Notes

Reverse Linked List

Difficulty: Easy

Link: https://leetcode.com/problems/reverse-linked-list (https://leetcode.com/problems/reverse-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* reverseList(ListNode* head) {

    }
};
```

Notes

Contains Duplicate

Difficulty: Easy

Link: https://leetcode.com/problems/contains-duplicate (https://leetcode.com/problems/contains-duplicate)

C++ Code

```
class Solution {
public:
    bool containsDuplicate(vector<int>& nums) {

    }
};
```

Notes

Contains Duplicate II

Difficulty: Easy

Link: https://leetcode.com/problems/contains-duplicate-ii (https://leetcode.com/problems/contains-duplicate-ii)

C++ Code

```
class Solution {
public:
    bool containsNearbyDuplicate(vector<int>& nums, int k) {

    }
};
```

Notes

Count Complete Tree Nodes

Difficulty: Easy

Link: https://leetcode.com/problems/count-complete-tree-nodes (https://leetcode.com/problems/count-complete-tree-nodes)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int countNodes(TreeNode* root) {

    }
};
```

Notes

Invert Binary Tree

Difficulty: Easy

Link: https://leetcode.com/problems/invert-binary-tree (https://leetcode.com/problems/invert-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* invertTree(TreeNode* root) {

    }
};
```

Notes

Summary Ranges

Difficulty: Easy

Link: https://leetcode.com/problems/summary-ranges (https://leetcode.com/problems/summary-ranges)

C++ Code

```
class Solution {
public:
    vector<string> summaryRanges(vector<int>& nums) {

    }
};
```

Notes

Power of Two

Difficulty: Easy

Link: https://leetcode.com/problems/power-of-two (https://leetcode.com/problems/power-of-two)

C++ Code

```
class Solution {
public:
    bool isPowerOfTwo(int n) {

    }
};
```

Notes

Palindrome Linked List

Difficulty: Easy

Link: https://leetcode.com/problems/palindrome-linked-list (https://leetcode.com/problems/palindrome-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    bool isPalindrome(ListNode* head) {

    }
};
```

Notes

Valid Anagram

Difficulty: Easy

Link: https://leetcode.com/problems/valid-anagram (https://leetcode.com/problems/valid-anagram)

C++ Code

```
class Solution {
public:
    bool isAnagram(string s, string t) {

    }
};
```

Notes

Meeting Rooms

Difficulty: Easy

Link: https://leetcode.com/problems/meeting-rooms (https://leetcode.com/problems/meeting-rooms)

C++ Code

```

```

Notes

Binary Tree Paths

Difficulty: Easy

Link: https://leetcode.com/problems/binary-tree-paths (https://leetcode.com/problems/binary-tree-paths)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<string> binaryTreePaths(TreeNode* root) {

    }
};
```

Notes

Missing Number

Difficulty: Easy

Link: https://leetcode.com/problems/missing-number (https://leetcode.com/problems/missing-number)

C++ Code

```
class Solution {
public:
    int missingNumber(vector<int>& nums) {

    }
};
```

Notes

First Bad Version

Difficulty: Easy

Link: https://leetcode.com/problems/first-bad-version (https://leetcode.com/problems/first-bad-version)

C++ Code

```
// The API isBadVersion is defined for you.
// bool isBadVersion(int version);

class Solution {
public:
    int firstBadVersion(int n) {

    }
};
```

Notes

Move Zeroes

Difficulty: Easy

Link: https://leetcode.com/problems/move-zeroes (https://leetcode.com/problems/move-zeroes)

C++ Code

```
class Solution {
public:
    void moveZeroes(vector<int>& nums) {

    }
};
```

Notes

Nim Game

Difficulty: Easy

Link: https://leetcode.com/problems/nim-game (https://leetcode.com/problems/nim-game)

C++ Code

```
class Solution {
public:
    bool canWinNim(int n) {

    }
};
```

Notes

Range Sum Query - Immutable

Difficulty: Easy

Link: https://leetcode.com/problems/range-sum-query-immutable (https://leetcode.com/problems/range-sum-query-immutable)

C++ Code

```
class NumArray {
public:
    NumArray(vector<int>& nums) {

    }

    int sumRange(int left, int right) {

    }
};

/**
 * Your NumArray object will be instantiated and called as such:
 * NumArray* obj = new NumArray(nums);
 * int param_1 = obj->sumRange(left,right);
 */
```

Notes

Power of Three

Difficulty: Easy

Link: https://leetcode.com/problems/power-of-three (https://leetcode.com/problems/power-of-three)

C++ Code

```
class Solution {
public:
    bool isPowerOfThree(int n) {

    }
};
```

Notes

Counting Bits

Difficulty: Easy

Link: https://leetcode.com/problems/counting-bits (https://leetcode.com/problems/counting-bits)

C++ Code

```
class Solution {
public:
    vector<int> countBits(int n) {

    }
};
```

Notes

Reverse String

Difficulty: Easy

Link: https://leetcode.com/problems/reverse-string (https://leetcode.com/problems/reverse-string)

C++ Code

```
class Solution {
public:
    void reverseString(vector<char>& s) {

    }
};
```

Notes

Reverse Vowels of a String

Difficulty: Easy

Link: https://leetcode.com/problems/reverse-vowels-of-a-string (https://leetcode.com/problems/reverse-vowels-of-a-string)

C++ Code

```
class Solution {
public:
    string reverseVowels(string s) {

    }
};
```

Notes

Intersection of Two Arrays

Difficulty: Easy

Link: https://leetcode.com/problems/intersection-of-two-arrays (https://leetcode.com/problems/intersection-of-two-arrays)

C++ Code

```
class Solution {
public:
    vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {

    }
};
```

Notes

Intersection of Two Arrays II

Difficulty: Easy

Link: https://leetcode.com/problems/intersection-of-two-arrays-ii (https://leetcode.com/problems/intersection-of-two-arrays-ii)

C++ Code

```
class Solution {
public:
    vector<int> intersect(vector<int>& nums1, vector<int>& nums2) {

    }
};
```

Notes

Guess Number Higher or Lower

Difficulty: Easy

Link: https://leetcode.com/problems/guess-number-higher-or-lower (https://leetcode.com/problems/guess-number-higher-or-lower)

C++ Code

```
/**
 * Forward declaration of guess API.
 * @param  num   your guess
 * @return           -1 if num is higher than the picked number
 *                            1 if num is lower than the picked number
 *               otherwise return 0
 * int guess(int num);
 */

class Solution {
public:
    int guessNumber(int n) {

    }
};
```

Notes

Ransom Note

Difficulty: Easy

Link: https://leetcode.com/problems/ransom-note (https://leetcode.com/problems/ransom-note)

C++ Code

```
class Solution {
public:
    bool canConstruct(string ransomNote, string magazine) {

    }
};
```

Notes

Find the Difference

Difficulty: Easy

Link: https://leetcode.com/problems/find-the-difference (https://leetcode.com/problems/find-the-difference)

C++ Code

```
class Solution {
public:
    char findTheDifference(string s, string t) {

    }
};
```

Notes

Is Subsequence

Difficulty: Easy

Link: https://leetcode.com/problems/is-subsequence (https://leetcode.com/problems/is-subsequence)

C++ Code

```
class Solution {
public:
    bool isSubsequence(string s, string t) {

    }
};
```

Notes

Arranging Coins

Difficulty: Easy

Link: https://leetcode.com/problems/arranging-coins (https://leetcode.com/problems/arranging-coins)

C++ Code

```
class Solution {
public:
    int arrangeCoins(int n) {

    }
};
```

Notes

Find All Numbers Disappeared in an Array

Difficulty: Easy

Link: https://leetcode.com/problems/find-all-numbers-disappeared-in-an-array (https://leetcode.com/problems/find-all-numbers-disappeared-in-an-array)

C++ Code

```
class Solution {
public:
    vector<int> findDisappearedNumbers(vector<int>& nums) {

    }
};
```

Notes

Repeated Substring Pattern

Difficulty: Easy

Link: https://leetcode.com/problems/repeated-substring-pattern (https://leetcode.com/problems/repeated-substring-pattern)

C++ Code

```
class Solution {
public:
    bool repeatedSubstringPattern(string s) {

    }
};
```

Notes

Island Perimeter

Difficulty: Easy

Link: https://leetcode.com/problems/island-perimeter (https://leetcode.com/problems/island-perimeter)

C++ Code

```
class Solution {
public:
    int islandPerimeter(vector<vector<int>>& grid) {

    }
};
```

Notes

Number Complement

Difficulty: Easy

Link: https://leetcode.com/problems/number-complement (https://leetcode.com/problems/number-complement)

C++ Code

```
class Solution {
public:
    int findComplement(int num) {

    }
};
```

Notes

License Key Formatting

Difficulty: Easy

Link: https://leetcode.com/problems/license-key-formatting (https://leetcode.com/problems/license-key-formatting)

C++ Code

```
class Solution {
public:
    string licenseKeyFormatting(string s, int k) {

    }
};
```

Notes

Next Greater Element I

Difficulty: Easy

Link: https://leetcode.com/problems/next-greater-element-i (https://leetcode.com/problems/next-greater-element-i)

C++ Code

```
class Solution {
public:
    vector<int> nextGreaterElement(vector<int>& nums1, vector<int>& nums2) {

    }
};
```

Notes

Relative Ranks

Difficulty: Easy

Link: https://leetcode.com/problems/relative-ranks (https://leetcode.com/problems/relative-ranks)

C++ Code

```
class Solution {
public:
    vector<string> findRelativeRanks(vector<int>& score) {

    }
};
```

Notes

Fibonacci Number

Difficulty: Easy

Link: https://leetcode.com/problems/fibonacci-number (https://leetcode.com/problems/fibonacci-number)

C++ Code

```
class Solution {
public:
    int fib(int n) {

    }
};
```

Notes

Minimum Absolute Difference in BST

Difficulty: Easy

Link: https://leetcode.com/problems/minimum-absolute-difference-in-bst (https://leetcode.com/problems/minimum-absolute-difference-in-bst)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int getMinimumDifference(TreeNode* root) {

    }
};
```

Notes

Diameter of Binary Tree

Difficulty: Easy

Link: https://leetcode.com/problems/diameter-of-binary-tree (https://leetcode.com/problems/diameter-of-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int diameterOfBinaryTree(TreeNode* root) {

    }
};
```

Notes

Subtree of Another Tree

Difficulty: Easy

Link: https://leetcode.com/problems/subtree-of-another-tree (https://leetcode.com/problems/subtree-of-another-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool isSubtree(TreeNode* root, TreeNode* subRoot) {

    }
};
```

Notes

N-ary Tree Preorder Traversal

Difficulty: Easy

Link: https://leetcode.com/problems/n-ary-tree-preorder-traversal (https://leetcode.com/problems/n-ary-tree-preorder-traversal)

C++ Code

```
/*
// Definition for a Node.
class Node {
public:
    int val;
    vector<Node*> children;

    Node() {}

    Node(int _val) {
        val = _val;
    }

    Node(int _val, vector<Node*> _children) {
        val = _val;
        children = _children;
    }
};
*/

class Solution {
public:
    vector<int> preorder(Node* root) {

    }
};
```

Notes

Can Place Flowers

Difficulty: Easy

Link: https://leetcode.com/problems/can-place-flowers (https://leetcode.com/problems/can-place-flowers)

C++ Code

```
class Solution {
public:
    bool canPlaceFlowers(vector<int>& flowerbed, int n) {

    }
};
```

Notes

Merge Two Binary Trees

Difficulty: Easy

Link: https://leetcode.com/problems/merge-two-binary-trees (https://leetcode.com/problems/merge-two-binary-trees)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {

    }
};
```

Notes

Average of Levels in Binary Tree

Difficulty: Easy

Link: https://leetcode.com/problems/average-of-levels-in-binary-tree (https://leetcode.com/problems/average-of-levels-in-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    vector<double> averageOfLevels(TreeNode* root) {

    }
};
```

Notes

Maximum Average Subarray I

Difficulty: Easy

Link: https://leetcode.com/problems/maximum-average-subarray-i (https://leetcode.com/problems/maximum-average-subarray-i)

C++ Code

```
class Solution {
public:
    double findMaxAverage(vector<int>& nums, int k) {

    }
};
```

Notes

Search in a Binary Search Tree

Difficulty: Easy

Link: https://leetcode.com/problems/search-in-a-binary-search-tree (https://leetcode.com/problems/search-in-a-binary-search-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* searchBST(TreeNode* root, int val) {

    }
};
```

Notes

Binary Search

Difficulty: Easy

Link: https://leetcode.com/problems/binary-search (https://leetcode.com/problems/binary-search)

C++ Code

```
class Solution {
public:
    int search(vector<int>& nums, int target) {

    }
};
```

Notes

Find Pivot Index

Difficulty: Easy

Link: https://leetcode.com/problems/find-pivot-index (https://leetcode.com/problems/find-pivot-index)

C++ Code

```
class Solution {
public:
    int pivotIndex(vector<int>& nums) {

    }
};
```

Notes

Flood Fill

Difficulty: Easy

Link: https://leetcode.com/problems/flood-fill (https://leetcode.com/problems/flood-fill)

C++ Code

```
class Solution {
public:
    vector<vector<int>> floodFill(vector<vector<int>>& image, int sr, int sc, int color) {

    }
};
```

Notes

Find Smallest Letter Greater Than Target

Difficulty: Easy

Link: https://leetcode.com/problems/find-smallest-letter-greater-than-target (https://leetcode.com/problems/find-smallest-letter-greater-than-target)

C++ Code

```
class Solution {
public:
    char nextGreatestLetter(vector<char>& letters, char target) {

    }
};
```

Notes

Min Cost Climbing Stairs

Difficulty: Easy

Link: https://leetcode.com/problems/min-cost-climbing-stairs (https://leetcode.com/problems/min-cost-climbing-stairs)

C++ Code

```
class Solution {
public:
    int minCostClimbingStairs(vector<int>& cost) {

    }
};
```

Notes

Backspace String Compare

Difficulty: Easy

Link: https://leetcode.com/problems/backspace-string-compare (https://leetcode.com/problems/backspace-string-compare)

C++ Code

```
class Solution {
public:
    bool backspaceCompare(string s, string t) {

    }
};
```

Notes

Lemonade Change

Difficulty: Easy

Link: https://leetcode.com/problems/lemonade-change (https://leetcode.com/problems/lemonade-change)

C++ Code

```
class Solution {
public:
    bool lemonadeChange(vector<int>& bills) {

    }
};
```

Notes

Middle of the Linked List

Difficulty: Easy

Link: https://leetcode.com/problems/middle-of-the-linked-list (https://leetcode.com/problems/middle-of-the-linked-list)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* middleNode(ListNode* head) {

    }
};
```

Notes

Uncommon Words from Two Sentences

Difficulty: Easy

Link: https://leetcode.com/problems/uncommon-words-from-two-sentences (https://leetcode.com/problems/uncommon-words-from-two-sentences)

C++ Code

```
class Solution {
public:
    vector<string> uncommonFromSentences(string s1, string s2) {

    }
};
```

Notes

Monotonic Array

Difficulty: Easy

Link: https://leetcode.com/problems/monotonic-array (https://leetcode.com/problems/monotonic-array)

C++ Code

```
class Solution {
public:
    bool isMonotonic(vector<int>& nums) {

    }
};
```

Notes

Smallest Range I

Difficulty: Easy

Link: https://leetcode.com/problems/smallest-range-i (https://leetcode.com/problems/smallest-range-i)

C++ Code

```
class Solution {
public:
    int smallestRangeI(vector<int>& nums, int k) {

    }
};
```

Notes

DI String Match

Difficulty: Easy

Link: https://leetcode.com/problems/di-string-match (https://leetcode.com/problems/di-string-match)

C++ Code

```
class Solution {
public:
    vector<int> diStringMatch(string s) {

    }
};
```

Notes

Squares of a Sorted Array

Difficulty: Easy

Link: https://leetcode.com/problems/squares-of-a-sorted-array (https://leetcode.com/problems/squares-of-a-sorted-array)

C++ Code

```
class Solution {
public:
    vector<int> sortedSquares(vector<int>& nums) {

    }
};
```

Notes

Find the Town Judge

Difficulty: Easy

Link: https://leetcode.com/problems/find-the-town-judge (https://leetcode.com/problems/find-the-town-judge)

C++ Code

```
class Solution {
public:
    int findJudge(int n, vector<vector<int>>& trust) {

    }
};
```

Notes

Remove Outermost Parentheses

Difficulty: Easy

Link: https://leetcode.com/problems/remove-outermost-parentheses (https://leetcode.com/problems/remove-outermost-parentheses)

C++ Code

```
class Solution {
public:
    string removeOuterParentheses(string s) {

    }
};
```

Notes

Sum of Root To Leaf Binary Numbers

Difficulty: Easy

Link: https://leetcode.com/problems/sum-of-root-to-leaf-binary-numbers (https://leetcode.com/problems/sum-of-root-to-leaf-binary-numbers)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int sumRootToLeaf(TreeNode* root) {

    }
};
```

Notes

Divisor Game

Difficulty: Easy

Link: https://leetcode.com/problems/divisor-game (https://leetcode.com/problems/divisor-game)

C++ Code

```
class Solution {
public:
    bool divisorGame(int n) {

    }
};
```

Notes

Confusing Number

Difficulty: Easy

Link: https://leetcode.com/problems/confusing-number (https://leetcode.com/problems/confusing-number)

C++ Code

```

```

Notes

Index Pairs of a String

Difficulty: Easy

Link: https://leetcode.com/problems/index-pairs-of-a-string (https://leetcode.com/problems/index-pairs-of-a-string)

C++ Code

```

```

Notes

Duplicate Zeros

Difficulty: Easy

Link: https://leetcode.com/problems/duplicate-zeros (https://leetcode.com/problems/duplicate-zeros)

C++ Code

```
class Solution {
public:
    void duplicateZeros(vector<int>& arr) {

    }
};
```

Notes

Armstrong Number

Difficulty: Easy

Link: https://leetcode.com/problems/armstrong-number (https://leetcode.com/problems/armstrong-number)

C++ Code

```

```

Notes

N-th Tribonacci Number

Difficulty: Easy

Link: https://leetcode.com/problems/n-th-tribonacci-number (https://leetcode.com/problems/n-th-tribonacci-number)

C++ Code

```
class Solution {
public:
    int tribonacci(int n) {

    }
};
```

Notes

Check If a Number Is Majority Element in a Sorted Array

Difficulty: Easy

Link: https://leetcode.com/problems/check-if-a-number-is-majority-element-in-a-sorted-array (https://leetcode.com/problems/check-if-a-number-is-majority-element-in-a-sorted-array)

C++ Code

```
```

Notes

Diet Plan Performance

Difficulty: Easy

Link: https://leetcode.com/problems/diet-plan-performance (https://leetcode.com/problems/diet-plan-performance)

C++ Code

```
```

Notes

Unique Number of Occurrences

Difficulty: Easy

Link: https://leetcode.com/problems/unique-number-of-occurrences (https://leetcode.com/problems/unique-number-of-occurrences)

C++ Code

```
class Solution {
public:
    bool uniqueOccurrences(vector<int>& arr) {

    }
};
```

Notes

Convert Binary Number in a Linked List to Integer

Difficulty: Easy

Link: https://leetcode.com/problems/convert-binary-number-in-a-linked-list-to-integer (https://leetcode.com/problems/convert-binary-number-in-a-linked-list-to-integer)

C++ Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    int getDecimalValue(ListNode* head) {

    }
};
```

Notes

Count Negative Numbers in a Sorted Matrix

Difficulty: Easy

Link: https://leetcode.com/problems/count-negative-numbers-in-a-sorted-matrix (https://leetcode.com/problems/count-negative-numbers-in-a-sorted-matrix)

C++ Code

```
class Solution {
public:
    int countNegatives(vector<vector<int>>& grid) {

    }
};
```

Notes

Find a Corresponding Node of a Binary Tree in a Clone of That Tree

Difficulty: Easy

Link: https://leetcode.com/problems/find-a-corresponding-node-of-a-binary-tree-in-a-clone-of-that-tree (https://leetcode.com/problems/find-a-corresponding-node-of-a-binary-tree-in-a-clone-of-that-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */

class Solution {
public:
    TreeNode* getTargetCopy(TreeNode* original, TreeNode* cloned, TreeNode* target) {

    }
};
```

Notes

Counting Elements

Difficulty: Easy

Link: https://leetcode.com/problems/counting-elements (https://leetcode.com/problems/counting-elements)

C++ Code

Notes

Kids With the Greatest Number of Candies

Difficulty: Easy

Link: https://leetcode.com/problems/kids-with-the-greatest-number-of-candies (https://leetcode.com/problems/kids-with-the-greatest-number-of-candies)

C++ Code

```
class Solution {
public:
    vector<bool> kidsWithCandies(vector<int>& candies, int extraCandies) {

    }
};
```

Notes

Final Prices With a Special Discount in a Shop

Difficulty: Easy

Link: https://leetcode.com/problems/final-prices-with-a-special-discount-in-a-shop (https://leetcode.com/problems/final-prices-with-a-special-discount-in-a-shop)

C++ Code

```
class Solution {
public:
    vector<int> finalPrices(vector<int>& prices) {

    }
};
```

Notes

Running Sum of 1d Array

Difficulty: Easy

Link: https://leetcode.com/problems/running-sum-of-1d-array (https://leetcode.com/problems/running-sum-of-1d-array)

C++ Code

```
class Solution {
public:
    vector<int> runningSum(vector<int>& nums) {

    }
};
```

Notes

Can Make Arithmetic Progression From Sequence

Difficulty: Easy

Link: https://leetcode.com/problems/can-make-arithmetic-progression-from-sequence (https://leetcode.com/problems/can-make-arithmetic-progression-from-sequence)

C++ Code

```
class Solution {
public:
    bool canMakeArithmeticProgression(vector<int>& arr) {

    }
};
```

Notes

Kth Missing Positive Number

Difficulty: Easy

Link: https://leetcode.com/problems/kth-missing-positive-number (https://leetcode.com/problems/kth-missing-positive-number)

C++ Code

```
class Solution {
public:
    int findKthPositive(vector<int>& arr, int k) {

    }
};
```

Notes

Design Parking System

Difficulty: Easy

Link: https://leetcode.com/problems/design-parking-system (https://leetcode.com/problems/design-parking-system)

C++ Code

```
class ParkingSystem {
public:
    ParkingSystem(int big, int medium, int small) {

    }

    bool addCar(int carType) {

    }
};

/**
 * Your ParkingSystem object will be instantiated and called as such:
 * ParkingSystem* obj = new ParkingSystem(big, medium, small);
 * bool param_1 = obj->addCar(carType);
 */
```

Notes

Richest Customer Wealth

Difficulty: Easy

Link: https://leetcode.com/problems/richest-customer-wealth (https://leetcode.com/problems/richest-customer-wealth)

C++ Code

```
class Solution {
public:
    int maximumWealth(vector<vector<int>>& accounts) {

    }
};
```

Notes

Count the Number of Consistent Strings

Difficulty: Easy

Link: https://leetcode.com/problems/count-the-number-of-consistent-strings (https://leetcode.com/problems/count-the-number-of-consistent-strings)

C++ Code

```
class Solution {
public:
    int countConsistentStrings(string allowed, vector<string>& words) {

    }
};
```

Notes

Find the Highest Altitude

Difficulty: Easy

Link: https://leetcode.com/problems/find-the-highest-altitude (https://leetcode.com/problems/find-the-highest-altitude)

C++ Code

```
class Solution {
public:
    int largestAltitude(vector<int>& gain) {

    }
};
```

Notes

Sum of Unique Elements

Difficulty: Easy

Link: https://leetcode.com/problems/sum-of-unique-elements (https://leetcode.com/problems/sum-of-unique-elements)

C++ Code

```
class Solution {
public:
    int sumOfUnique(vector<int>& nums) {

    }
};
```

Notes

Merge Strings Alternately

Difficulty: Easy

Link: https://leetcode.com/problems/merge-strings-alternately (https://leetcode.com/problems/merge-strings-alternately)

C++ Code

```
class Solution {
public:
    string mergeAlternately(string word1, string word2) {

    }
};
```

Notes

Sign of the Product of an Array

Difficulty: Easy

Link: https://leetcode.com/problems/sign-of-the-product-of-an-array (https://leetcode.com/problems/sign-of-the-product-of-an-array)

C++ Code

```
class Solution {
public:
    int arraySign(vector<int>& nums) {

    }
};
```

Notes

Sum of Digits of String After Convert

Difficulty: Easy

Link: https://leetcode.com/problems/sum-of-digits-of-string-after-convert (https://leetcode.com/problems/sum-of-digits-of-string-after-convert)

C++ Code

```
class Solution {
public:
    int getLucky(string s, int k) {

    }
};
```

Notes

Convert 1D Array Into 2D Array

Difficulty: Easy

Link: https://leetcode.com/problems/convert-1d-array-into-2d-array (https://leetcode.com/problems/convert-1d-array-into-2d-array)

C++ Code

```
class Solution {
public:
    vector<vector<int>> construct2DArray(vector<int>& original, int m, int n) {

    }
};
```

Notes

Count Elements With Strictly Smaller and Greater Elements

Difficulty: Easy

Link: https://leetcode.com/problems/count-elements-with-strictly-smaller-and-greater-elements (https://leetcode.com/problems/count-elements-with-strictly-smaller-and-greater-elements)

C++ Code

```
class Solution {
public:
    int countElements(vector<int>& nums) {

    }
};
```

Notes

Find the Difference of Two Arrays

Difficulty: Easy

Link: https://leetcode.com/problems/find-the-difference-of-two-arrays (https://leetcode.com/problems/find-the-difference-of-two-arrays)

C++ Code

```
class Solution {
public:
    vector<vector<int>> findDifference(vector<int>& nums1, vector<int>& nums2) {

    }
};
```

Notes

Minimum Bit Flips to Convert Number

Difficulty: Easy

Link: https://leetcode.com/problems/minimum-bit-flips-to-convert-number (https://leetcode.com/problems/minimum-bit-flips-to-convert-number)

C++ Code

```
class Solution {
public:
    int minBitFlips(int start, int goal) {

    }
};
```

Notes

Find the Index of the First Occurrence in a String

Difficulty: Easy

Link: https://leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string (https://leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string)

C++ Code

```
class Solution {
public:
    int strStr(string haystack, string needle) {

    }
};
```

Notes

Evaluate Boolean Binary Tree

Difficulty: Easy

Link: https://leetcode.com/problems/evaluate-boolean-binary-tree (https://leetcode.com/problems/evaluate-boolean-binary-tree)

C++ Code

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool evaluateTree(TreeNode* root) {

    }
};
```

Notes

Largest Local Values in a Matrix

Difficulty: Easy

Link: https://leetcode.com/problems/largest-local-values-in-a-matrix (https://leetcode.com/problems/largest-local-values-in-a-matrix)

C++ Code

```
class Solution {
public:
    vector<vector<int>> largestLocal(vector<vector<int>>& grid) {

    }
};
```

Notes

Special Array I

Difficulty: Easy

Link: https://leetcode.com/problems/special-array-i (https://leetcode.com/problems/special-array-i)

C++ Code

```
class Solution {
public:
    bool isArraySpecial(vector<int>& nums) {

    }
};
```

Notes

Find the Winning Player in Coin Game

Difficulty: Easy

Link: https://leetcode.com/problems/find-the-winning-player-in-coin-game (https://leetcode.com/problems/find-the-winning-player-in-coin-game)

C++ Code

```
class Solution {
public:
    string losingPlayer(int x, int y) {

    }
};
```

Notes