目录

[601. （SQL） 3](#_Toc37148421)

[605. Can Place Flowers 4](#_Toc37148422)

[606. Construct String from Binary Tree 5](#_Toc37148423)

[609. Find Duplicate File in System 7](#_Toc37148424)

[611. Valid Triangle Number 9](#_Toc37148425)

[617. Merge Two Binary Trees 10](#_Toc37148426)

[620. SQL 11](#_Toc37148427)

[621. Task Scheduler 12](#_Toc37148428)

[622. Design Circular Queue 13](#_Toc37148429)

[623. Add One Row to Tree 15](#_Toc37148430)

[626. SQL 18](#_Toc37148431)

[627. SQL 19](#_Toc37148432)

[628. Maximum Product of Three Numbers 20](#_Toc37148433)

[629. K Inverse Pairs Array 21](#_Toc37148434)

[630. Course Schedule III 22](#_Toc37148435)

[632. Smallest Range Covering Elements from K Lists 23](#_Toc37148436)

[633. Sum of Square Numbers 24](#_Toc37148437)

[636. Exclusive Time of Functions 25](#_Toc37148438)

[637. Average of Levels in Binary Tree 27](#_Toc37148439)

[638. Shopping Offers 28](#_Toc37148440)

[639. Decode Ways II 30](#_Toc37148441)

[640. Solve the Equation 31](#_Toc37148442)

[641. Design Circular Deque 32](#_Toc37148443)

[643. Maximum Average Subarray I 34](#_Toc37148444)

[645. Set Mismatch 35](#_Toc37148445)

[646. Maximum Length of Pair Chain 36](#_Toc37148446)

[647. Palindromic Substrings 37](#_Toc37148447)

[648. Replace Words 38](#_Toc37148448)

[649. Dota2 Senate 39](#_Toc37148449)

[650. 2 Keys Keyboard 41](#_Toc37148450)

[652. Find Duplicate Subtrees 42](#_Toc37148451)

[653. Two Sum IV - Input is a BST 43](#_Toc37148452)

[654. Maximum Binary Tree 45](#_Toc37148453)

[655. Print Binary Tree 46](#_Toc37148454)

[657. Robot Return to Origin 48](#_Toc37148455)

[658. Find K Closest Elements 49](#_Toc37148456)

[659. Split Array into Consecutive Subsequences 50](#_Toc37148457)

[661. Image Smoother 52](#_Toc37148458)

[662. Maximum Width of Binary Tree 53](#_Toc37148459)

[664. Strange Printer 56](#_Toc37148460)

[665. Non-decreasing Array 57](#_Toc37148461)

[667. Beautiful Arrangement II 58](#_Toc37148462)

[668. Kth Smallest Number in Multiplication Table 59](#_Toc37148463)

[669. Trim a Binary Search Tree 61](#_Toc37148464)

[670. Maximum Swap 63](#_Toc37148465)

[672. Bulb Switcher II 65](#_Toc37148466)

[673. Number of Longest Increasing Subsequence 67](#_Toc37148467)

[674. Longest Continuous Increasing Subsequence 68](#_Toc37148468)

[675. Cut Off Trees for Golf Event 69](#_Toc37148469)

[676. Implement Magic Dictionary 71](#_Toc37148470)

[677. Map Sum Pairs 72](#_Toc37148471)

[678. Valid Parenthesis String 73](#_Toc37148472)

[679. 24 Game 74](#_Toc37148473)

[680. Valid Palindrome II 75](#_Toc37148474)

[682. Baseball Game 76](#_Toc37148475)

[684. Redundant Connection 78](#_Toc37148476)

[685. Redundant Connection II 80](#_Toc37148477)

[686. Repeated String Match 82](#_Toc37148478)

[687. Longest Univalue Path 83](#_Toc37148479)

[688. Knight Probability in Chessboard 85](#_Toc37148480)

[689. Maximum Sum of 3 Non-Overlapping Subarrays 87](#_Toc37148481)

[690. Employee Importance 88](#_Toc37148482)

[691. Stickers to Spell Word 89](#_Toc37148483)

[692. Top K Frequent Words 91](#_Toc37148484)

[693. Binary Number with Alternating Bits 92](#_Toc37148485)

[695. Max Area of Island 93](#_Toc37148486)

[696. Count Binary Substrings 94](#_Toc37148487)

[697. Degree of an Array 95](#_Toc37148488)

[698. Partition to K Equal Sum Subsets 96](#_Toc37148489)

[699. Falling Squares 97](#_Toc37148490)

[700. Search in a Binary Search Tree 99](#_Toc37148491)

### 601. （SQL）

### 605. Can Place Flowers

Easy

Suppose you have a long flowerbed in which some of the plots are planted and some are not. However, flowers cannot be planted in adjacent plots - they would compete for water and both would die.

Given a flowerbed (represented as an array containing 0 and 1, where 0 means empty and 1 means not empty), and a number **n**, return if **n** new flowers can be planted in it without violating the no-adjacent-flowers rule.

**Example 1:**

**Input:** flowerbed = [1,0,0,0,1], n = 1

**Output:** True

**Example 2:**

**Input:** flowerbed = [1,0,0,0,1], n = 2

**Output:** False

**Note:**

1. The input array won't violate no-adjacent-flowers rule.
2. The input array size is in the range of [1, 20000].
3. **n** is a non-negative integer which won't exceed the input array size.

class Solution {

public:

    bool canPlaceFlowers(vector<int>& v, int n) {

        if (n <= 0) return true;

        int N = v.size();

        for(int i = 0; i < N; i++) {

            if ((!i || !v[i-1]) && !v[i] && (i == N-1 || !v[i+1])) {

                if (--n == 0) return true;

                v[i] = 1;

            }

        }

        return false;

    }

};

### 606. Construct String from Binary Tree

Easy

You need to construct a string consists of parenthesis and integers from a binary tree with the preorder traversing way.

The null node needs to be represented by empty parenthesis pair "()". And you need to omit all the empty parenthesis pairs that don't affect the one-to-one mapping relationship between the string and the original binary tree.

**Example 1:**

**Input:** Binary tree: [1,2,3,4]

1

/ \

2 3

/

4

**Output:** "1(2(4))(3)"

**Explanation:** Originallay it needs to be "1(2(4)())(3()())",   
but you need to omit all the unnecessary empty parenthesis pairs.   
And it will be "1(2(4))(3)".

**Example 2:**

**Input:** Binary tree: [1,2,3,null,4]

1

/ \

2 3

\

4

**Output:** "1(2()(4))(3)"

**Explanation:** Almost the same as the first example,   
except we can't omit the first parenthesis pair to break the one-to-one mapping relationship between the input and the output.

/\*\*

 \* 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:

    string tree2str(TreeNode\* t) {

        if (!t) return "";

        string l = tree2str(t->left);

        string r = tree2str(t->right);

        if (l == "" && r == "") return to\_string(t->val);

        else if (r == "") return to\_string(t->val) + "(" + l + ")";

        else return to\_string(t->val) + "(" + l + ")"+ "(" + r + ")";

    }

};

### 609. Find Duplicate File in System

Medium

Given a list of directory info including directory path, and all the files with contents in this directory, you need to find out all the groups of duplicate files in the file system in terms of their paths.

A group of duplicate files consists of at least **two** files that have exactly the same content.

A single directory info string in the **input** list has the following format:

"root/d1/d2/.../dm f1.txt(f1\_content) f2.txt(f2\_content) ... fn.txt(fn\_content)"

It means there are **n** files (f1.txt, f2.txt ... fn.txt with content f1\_content, f2\_content ... fn\_content, respectively) in directory root/d1/d2/.../dm. Note that n >= 1 and m >= 0. If m = 0, it means the directory is just the root directory.

The **output** is a list of group of duplicate file paths. For each group, it contains all the file paths of the files that have the same content. A file path is a string that has the following format:

"directory\_path/file\_name.txt"

**Example 1:**

**Input:**

["root/a 1.txt(abcd) 2.txt(efgh)", "root/c 3.txt(abcd)", "root/c/d 4.txt(efgh)", "root 4.txt(efgh)"]

**Output:**

[["root/a/2.txt","root/c/d/4.txt","root/4.txt"],["root/a/1.txt","root/c/3.txt"]]

**Note:**

1. No order is required for the final output.
2. You may assume the directory name, file name and file content only has letters and digits, and the length of file content is in the range of [1,50].
3. The number of files given is in the range of [1,20000].
4. You may assume no files or directories share the same name in the same directory.
5. You may assume each given directory info represents a unique directory. Directory path and file info are separated by a single blank space.

**Follow-up beyond contest:**

1. Imagine you are given a real file system, how will you search files? DFS or BFS?
2. If the file content is very large (GB level), how will you modify your solution?
3. If you can only read the file by 1kb each time, how will you modify your solution?
4. What is the time complexity of your modified solution? What is the most time-consuming part and memory consuming part of it? How to optimize?
5. How to make sure the duplicated files you find are not false positive?

class Solution {

public:

    vector<vector<string>> findDuplicate(vector<string>& paths) {

        unordered\_map<string, vector<string>> mp;

        for (const auto &s : paths) {

            stringstream ss(s);

            string root, file;

            ss >> root;

            while (ss >> file) {

               auto pos = file.find('(');

               string content = file.substr(pos);

               mp[content].push\_back(root + "/" + file.substr(0, pos));

            }

        }

        vector<vector<string>> res;

        for (auto i : mp) {

            if (i.second.size() > 1) {

               res.push\_back(i.second);

            }

        }

        return res;

    }

};

### 611. Valid Triangle Number

Medium

Given an array consists of non-negative integers, your task is to count the number of triplets chosen from the array that can make triangles if we take them as side lengths of a triangle.

**Example 1:**

**Input:** [2,2,3,4]

**Output:** 3

**Explanation:**

Valid combinations are:

2,3,4 (using the first 2)

2,3,4 (using the second 2)

2,2,3

**Note:**

1. The length of the given array won't exceed 1000.
2. The integers in the given array are in the range of [0, 1000].

### 617. Merge Two Binary Trees

Easy

Given two binary trees and imagine that when you put one of them to cover the other, some nodes of the two trees are overlapped while the others are not.

You need to merge them into a new binary tree. The merge rule is that if two nodes overlap, then sum node values up as the new value of the merged node. Otherwise, the NOT null node will be used as the node of new tree.

**Example 1:**

**Input:**

Tree 1 Tree 2

1 2

/ \ / \

3 2 1 3

/ \ \

5 4 7

**Output:**

Merged tree:

3

/ \

4 5

/ \ \

5 4 7

**Note:** The merging process must start from the root nodes of both trees.

/\*\*

 \* 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\* mergeTrees(TreeNode\* t1, TreeNode\* t2) {

        if (!t1 || !t2) return t1 ? t1 : t2;

        t1->val += t2->val;

        t1->left = mergeTrees(t1->left, t2->left);

        t1->right = mergeTrees(t1->right, t2->right);

        return t1;

    }

};

### 620. SQL

### 621. Task Scheduler

Medium

Given a char array representing tasks CPU need to do. It contains capital letters A to Z where different letters represent different tasks. Tasks could be done without original order. Each task could be done in one interval. For each interval, CPU could finish one task or just be idle.

However, there is a non-negative cooling interval **n** that means between two **same tasks**, there must be at least n intervals that CPU are doing different tasks or just be idle.

You need to return the **least** number of intervals the CPU will take to finish all the given tasks.

**Example:**

**Input:** tasks = ["A","A","A","B","B","B"], n = 2

**Output:** 8

**Explanation:** A -> B -> idle -> A -> B -> idle -> A -> B.

**Constraints:**

* The number of tasks is in the range [1, 10000].
* The integer n is in the range [0, 100].

class Solution {

public:

    int leastInterval(vector<char>& tasks, int n) {

        unordered\_map<char, int> m;

        for (char t : tasks)  m[t]++;

        priority\_queue<int> pq;

        for (auto it : m) pq.push(it.second);

        int res = 0;

        int cycle = n + 1;

        while (!pq.empty()) {

            int time = 0;

            vector<int> tmp;

            for (int i = 0; i < cycle; i++) {

                if (pq.empty()) break;

                tmp.push\_back(pq.top());

                pq.pop();

                time++;

            }

            for (int cnt : tmp) {

                if (--cnt) pq.push(cnt);

            }

            res += pq.empty() ? time : cycle;

        }

        return res;

    }

};

class Solution {

public:

    int leastInterval(vector<char>& tasks, int n) {

        unordered\_map<char, int> m;

        int cnt = 0;

        for (auto c : tasks) {

            m[c]++;

            cnt = max(cnt, m[c]);

        }

        int res = (cnt-1) \* (n+1);

        for(auto c : m) if (c.second == cnt) res++;

        return max((int)tasks.size(), res);

    }

};

### 622. Design Circular Queue

Medium

Design your implementation of the circular queue. The circular queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called "Ring Buffer".

One of the benefits of the circular queue is that we can make use of the spaces in front of the queue. In a normal queue, once the queue becomes full, we cannot insert the next element even if there is a space in front of the queue. But using the circular queue, we can use the space to store new values.

Your implementation should support following operations:

* MyCircularQueue(k): Constructor, set the size of the queue to be k.
* Front: Get the front item from the queue. If the queue is empty, return -1.
* Rear: Get the last item from the queue. If the queue is empty, return -1.
* enQueue(value): Insert an element into the circular queue. Return true if the operation is successful.
* deQueue(): Delete an element from the circular queue. Return true if the operation is successful.
* isEmpty(): Checks whether the circular queue is empty or not.
* isFull(): Checks whether the circular queue is full or not.

**Example:**

MyCircularQueue circularQueue = new MyCircularQueue(3); // set the size to be 3

circularQueue.enQueue(1);  // return true

circularQueue.enQueue(2);  // return true

circularQueue.enQueue(3);  // return true

circularQueue.enQueue(4);  // return false, the queue is full

circularQueue.Rear();  // return 3

circularQueue.isFull();  // return true

circularQueue.deQueue();  // return true

circularQueue.enQueue(4);  // return true

circularQueue.Rear();  // return 4

**Note:**

* All values will be in the range of [0, 1000].
* The number of operations will be in the range of [1, 1000].
* Please do not use the built-in Queue library.

/\*\*

 \* Your MyCircularQueue object will be instantiated and called as such:

 \* MyCircularQueue\* obj = new MyCircularQueue(k);

 \* bool param\_1 = obj->enQueue(value);

 \* bool param\_2 = obj->deQueue();

 \* int param\_3 = obj->Front();

 \* int param\_4 = obj->Rear();

 \* bool param\_5 = obj->isEmpty();

 \* bool param\_6 = obj->isFull();

 \*/

class MyCircularQueue {

public:

    MyCircularQueue(int k) {

        que.resize(N = k+1);

    }

    bool enQueue(int value) {

        if (isFull()) return false;

        que[q] = value;

        q = ++q % N;

        return true;

    }

    bool deQueue() {

        if (isEmpty()) return false;

        ++p;

        return true;

    }

    int Front() {

        if (isEmpty()) return -1;

        return que[p];

    }

    int Rear() {

        if (isEmpty()) return -1;

        return que[(q-1+N) % N];

    }

    bool isEmpty() {

        return p == q;

    }

    bool isFull() {

        return (q+1) % N == p;

    }

private:

    vector<int> que;

    int N;

    int p = 0, q = 0;

};

### 623. Add One Row to Tree

Medium

Given the root of a binary tree, then value v and depth d, you need to add a row of nodes with value v at the given depth d. The root node is at depth 1.

The adding rule is: given a positive integer depth d, for each NOT null tree nodes N in depth d-1, create two tree nodes with value v as N's left subtree root and right subtree root. And N's **original left subtree** should be the left subtree of the new left subtree root, its **original right subtree** should be the right subtree of the new right subtree root. If depth d is 1 that means there is no depth d-1 at all, then create a tree node with value **v** as the new root of the whole original tree, and the original tree is the new root's left subtree.

**Example 1:**

**Input:**

A binary tree as following:

4

/ \

2 6

/ \ /

3 1 5

**v = 1**

**d = 2**

**Output:**

4

/ \

1 1

/ \

2 6

/ \ /

3 1 5

**Example 2:**

**Input:**

A binary tree as following:

4

/

2

/ \

3 1

**v = 1**

**d = 3**

**Output:**

4

/

2

/ \

1 1

/ \

3 1

**Note:**

1. The given d is in range [1, maximum depth of the given tree + 1].
2. The given binary tree has at least one tree node.

/\*\*

 \* 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\* addOneRow(TreeNode\* root, int v, int d) {

        auto dummp = new TreeNode(-1);

        dummp->left = root;

        dfs(dummp, v, d);

        return dummp->left;

    }

private:

    void dfs(TreeNode\* root, int v, int d) {

        if (!root) return;

        if (--d) {

           dfs(root->left, v, d);

           dfs(root->right, v, d);

        }

        else {

            auto l = root->left, r = root->right;

            root->left = new TreeNode(v);

            root->left->left = l;

            root->right = new TreeNode(v);

            root->right->right = r;

        }

    }

};

### 626. SQL

### 627. SQL

### 628. Maximum Product of Three Numbers

Easy

Given an integer array, find three numbers whose product is maximum and output the maximum product.

**Example 1:**

**Input:** [1,2,3]

**Output:** 6

**Example 2:**

**Input:** [1,2,3,4]

**Output:** 24

**Note:**

1. The length of the given array will be in range [3,104] and all elements are in the range [-1000, 1000].
2. Multiplication of any three numbers in the input won't exceed the range of 32-bit signed integer.

class Solution {

public:

    int maximumProduct(vector<int>& nums) {

        int min1 =INT\_MAX, min2 = INT\_MAX;

        int max1 = INT\_MIN, max2 = INT\_MIN, max3 = INT\_MIN;

        for (auto &n : nums) {

            if (n <= min1) {

                min2 = min1;

                min1 = n;

            } else if (n <= min2) {

                min2 = n;

            }

            if (n >= max1) {

                max3 = max2;

                max2 = max1;

                max1 = n;

            } else if (n >= max2) {

                max3 = max2;

                max2 = n;

            } else if (n >= max3) {

                max3 = n;

            }

        }

        return max1\*max(min1 \* min2, max2 \* max3);

    }

};

### 629. K Inverse Pairs Array

Hard

Given two integers n and k, find how many different arrays consist of numbers from 1 to n such that there are exactly k inverse pairs.

We define an inverse pair as following: For ith and jth element in the array, if i < j and a[i] > a[j] then it's an inverse pair; Otherwise, it's not.

Since the answer may be very large, the answer should be modulo 109 + 7.

**Example 1:**

**Input:** n = 3, k = 0

**Output:** 1

**Explanation:**

Only the array [1,2,3] which consists of numbers from 1 to 3 has exactly 0 inverse pair.

**Example 2:**

**Input:** n = 3, k = 1

**Output:** 2

**Explanation:**

The array [1,3,2] and [2,1,3] have exactly 1 inverse pair.

**Note:**

1. The integer n is in the range [1, 1000] and k is in the range [0, 1000].

class Solution {

public:

    int kInversePairs(int n, int k) {

        vector<vector<long long>> dp(2, vector<long long>(k+1));

        dp[0][0] = 1;

        int p = 0;

        for(int i = 1; i <= n; ++i){

            auto &pre = dp[p], &cur = dp[p^1];

            cur[0] = 1;

            for(int j = 1; j <= k; ++j){

                cur[j] = (cur[j-1] - (j >= i ? pre[j-i]:0)

+ pre[j] + mod) % mod;

            }

            p ^= 1;

        }

        return static\_cast<int> (dp[p][k]);

    }

private:

    const long long mod = 1e9 + 7;

};

### 630. Course Schedule III

Hard

There are n different online courses numbered from 1 to n. Each course has some duration(course length) t and closed on dth day. A course should be taken **continuously** for t days and must be finished before or on the dth day. You will start at the 1st day.

Given n online courses represented by pairs (t,d), your task is to find the maximal number of courses that can be taken.

**Example:**

**Input:** [[100, 200], [200, 1300], [1000, 1250], [2000, 3200]]

**Output:** 3

**Explanation:**

There're totally 4 courses, but you can take 3 courses at most:

First, take the 1st course, it costs 100 days so you will finish it on the 100th day, and ready to take the next course on the 101st day.

Second, take the 3rd course, it costs 1000 days so you will finish it on the 1100th day, and ready to take the next course on the 1101st day.

Third, take the 2nd course, it costs 200 days so you will finish it on the 1300th day.

The 4th course cannot be taken now, since you will finish it on the 3300th day, which exceeds the closed date.

**Note:**

1. The integer 1 <= d, t, n <= 10,000.
2. You can't take two courses simultaneously.

### 632. Smallest Range Covering Elements from K Lists

Hard

You have k lists of sorted integers in ascending order. Find the **smallest** range that includes at least one number from each of the k lists.

We define the range [a,b] is smaller than range [c,d] if b-a < d-c or a < c if b-a == d-c.

**Example 1:**

**Input:** [[4,10,15,24,26], [0,9,12,20], [5,18,22,30]]

**Output:** [20,24]

**Explanation:**

List 1: [4, 10, 15, 24,26], 24 is in range [20,24].

List 2: [0, 9, 12, 20], 20 is in range [20,24].

List 3: [5, 18, 22, 30], 22 is in range [20,24].

**Note:**

1. The given list may contain duplicates, so ascending order means >= here.
2. 1 <= k <= 3500
3. -105 <= value of elements <= 105.

### 633. Sum of Square Numbers

Easy

Given a non-negative integer c, your task is to decide whether there're two integers a and b such that a2 + b2 = c.

**Example 1:**

**Input:** 5

**Output:** True

**Explanation:** 1 \* 1 + 2 \* 2 = 5

**Example 2:**

**Input:** 3

**Output:** False

### 636. Exclusive Time of Functions

Medium

On a **single threaded** CPU, we execute some functions.  Each function has a unique id between 0 and N-1.

We store logs in timestamp order that describe when a function is entered or exited.

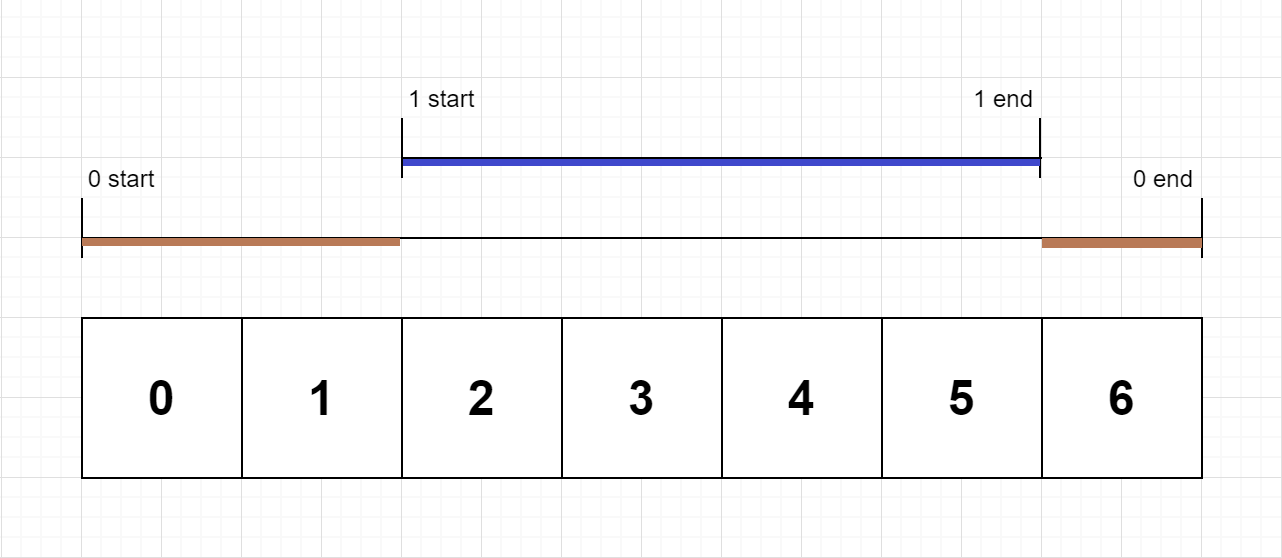
Each log is a string with this format: "{function\_id}:{"start" | "end"}:{timestamp}".  For example, "0:start:3" means the function with id 0 **started at the beginning** of timestamp 3.  "1:end:2" means the function with id 1 **ended at the end** of timestamp 2.

A function's *exclusive time* is the number of units of time spent in this function.  Note that this does **not** include any recursive calls to child functions.

The CPU is **single threaded** which means that only one function is being executed at a given time unit.

Return the exclusive time of each function, sorted by their function id.

**Example 1:**

****

**Input:**

n = 2

logs = ["0:start:0","1:start:2","1:end:5","0:end:6"]

**Output:** [3, 4]

**Explanation:**

Function 0 starts at the beginning of time 0, then it executes 2 units of time and reaches the end of time 1.

Now function 1 starts at the beginning of time 2, executes 4 units of time and ends at time 5.

Function 0 is running again at the beginning of time 6, and also ends at the end of time 6, thus executing for 1 unit of time.

So function 0 spends 2 + 1 = 3 units of total time executing, and function 1 spends 4 units of total time executing.

**Note:**

1. 1 <= n <= 100
2. Two functions won't start or end at the same time.
3. Functions will always log when they exit.

### 637. Average of Levels in Binary Tree

Easy

Given a non-empty binary tree, return the average value of the nodes on each level in the form of an array.

**Example 1:**

**Input:**

3

/ \

9 20

/ \

15 7

**Output:** [3, 14.5, 11]

**Explanation:**

The average value of nodes on level 0 is 3, on level 1 is 14.5, and on level 2 is 11. Hence return [3, 14.5, 11].

**Note:**

1. The range of node's value is in the range of 32-bit signed integer.

### 638. Shopping Offers

Medium

In LeetCode Store, there are some kinds of items to sell. Each item has a price.

However, there are some special offers, and a special offer consists of one or more different kinds of items with a sale price.

You are given the each item's price, a set of special offers, and the number we need to buy for each item. The job is to output the lowest price you have to pay for **exactly** certain items as given, where you could make optimal use of the special offers.

Each special offer is represented in the form of an array, the last number represents the price you need to pay for this special offer, other numbers represents how many specific items you could get if you buy this offer.

You could use any of special offers as many times as you want.

**Example 1:**

**Input:** [2,5], [[3,0,5],[1,2,10]], [3,2]

**Output:** 14

**Explanation:**

There are two kinds of items, A and B. Their prices are $2 and $5 respectively.

In special offer 1, you can pay $5 for 3A and 0B

In special offer 2, you can pay $10 for 1A and 2B.

You need to buy 3A and 2B, so you may pay $10 for 1A and 2B (special offer #2), and $4 for 2A.

**Example 2:**

**Input:** [2,3,4], [[1,1,0,4],[2,2,1,9]], [1,2,1]

**Output:** 11

**Explanation:**

The price of A is $2, and $3 for B, $4 for C.

You may pay $4 for 1A and 1B, and $9 for 2A ,2B and 1C.

You need to buy 1A ,2B and 1C, so you may pay $4 for 1A and 1B (special offer #1), and $3 for 1B, $4 for 1C.

You cannot add more items, though only $9 for 2A ,2B and 1C.

**Note:**

1. There are at most 6 kinds of items, 100 special offers.
2. For each item, you need to buy at most 6 of them.
3. You are **not** allowed to buy more items than you want, even if that would lower the overall price.

### 639. Decode Ways II

Hard

A message containing letters from A-Z is being encoded to numbers using the following mapping way:

'A' -> 1

'B' -> 2

...

'Z' -> 26

Beyond that, now the encoded string can also contain the character '\*', which can be treated as one of the numbers from 1 to 9.

Given the encoded message containing digits and the character '\*', return the total number of ways to decode it.

Also, since the answer may be very large, you should return the output mod 109 + 7.

**Example 1:**

**Input:** "\*"

**Output:** 9

**Explanation:** The encoded message can be decoded to the string: "A", "B", "C", "D", "E", "F", "G", "H", "I".

**Example 2:**

**Input:** "1\*"

**Output:** 9 + 9 = 18

**Note:**

1. The length of the input string will fit in range [1, 105].
2. The input string will only contain the character '\*' and digits '0' - '9'.

### 640. Solve the Equation

Medium

206479Add to ListShare

Solve a given equation and return the value of x in the form of string "x=#value". The equation contains only '+', '-' operation, the variable x and its coefficient.

If there is no solution for the equation, return "No solution".

If there are infinite solutions for the equation, return "Infinite solutions".

If there is exactly one solution for the equation, we ensure that the value of x is an integer.

**Example 1:**

**Input:** "x+5-3+x=6+x-2"

**Output:** "x=2"

**Example 2:**

**Input:** "x=x"

**Output:** "Infinite solutions"

**Example 3:**

**Input:** "2x=x"

**Output:** "x=0"

**Example 4:**

**Input:** "2x+3x-6x=x+2"

**Output:** "x=-1"

**Example 5:**

**Input:** "x=x+2"

**Output:** "No solution"

### 641. Design Circular Deque

Medium

Design your implementation of the circular double-ended queue (deque).

Your implementation should support following operations:

* MyCircularDeque(k): Constructor, set the size of the deque to be k.
* insertFront(): Adds an item at the front of Deque. Return true if the operation is successful.
* insertLast(): Adds an item at the rear of Deque. Return true if the operation is successful.
* deleteFront(): Deletes an item from the front of Deque. Return true if the operation is successful.
* deleteLast(): Deletes an item from the rear of Deque. Return true if the operation is successful.
* getFront(): Gets the front item from the Deque. If the deque is empty, return -1.
* getRear(): Gets the last item from Deque. If the deque is empty, return -1.
* isEmpty(): Checks whether Deque is empty or not.
* isFull(): Checks whether Deque is full or not.

**Example:**

MyCircularDeque circularDeque = new MycircularDeque(3); // set the size to be 3

circularDeque.insertLast(1); // return true

circularDeque.insertLast(2); // return true

circularDeque.insertFront(3); // return true

circularDeque.insertFront(4); // return false, the queue is full

circularDeque.getRear(); // return 2

circularDeque.isFull(); // return true

circularDeque.deleteLast(); // return true

circularDeque.insertFront(4); // return true

circularDeque.getFront(); // return 4

**Note:**

* All values will be in the range of [0, 1000].
* The number of operations will be in the range of [1, 1000].
* Please do not use the built-in Deque library.

### 643. Maximum Average Subarray I

Easy

Given an array consisting of n integers, find the contiguous subarray of given length k that has the maximum average value. And you need to output the maximum average value.

**Example 1:**

**Input:** [1,12,-5,-6,50,3], k = 4

**Output:** 12.75

**Explanation:** Maximum average is (12-5-6+50)/4 = 51/4 = 12.75

**Note:**

1. 1 <= k <= n <= 30,000.
2. Elements of the given array will be in the range [-10,000, 10,000].

### 645. Set Mismatch

Easy

The set S originally contains numbers from 1 to n. But unfortunately, due to the data error, one of the numbers in the set got duplicated to **another** number in the set, which results in repetition of one number and loss of another number.

Given an array nums representing the data status of this set after the error. Your task is to firstly find the number occurs twice and then find the number that is missing. Return them in the form of an array.

**Example 1:**

**Input:** nums = [1,2,2,4]

**Output:** [2,3]

**Note:**

1. The given array size will in the range [2, 10000].
2. The given array's numbers won't have any order.

### 646. Maximum Length of Pair Chain

Medium

You are given n pairs of numbers. In every pair, the first number is always smaller than the second number.

Now, we define a pair (c, d) can follow another pair (a, b) if and only if b < c. Chain of pairs can be formed in this fashion.

Given a set of pairs, find the length longest chain which can be formed. You needn't use up all the given pairs. You can select pairs in any order.

**Example 1:**

**Input:** [[1,2], [2,3], [3,4]]

**Output:** 2

**Explanation:** The longest chain is [1,2] -> [3,4]

**Note:**

1. The number of given pairs will be in the range [1, 1000].

### 647. Palindromic Substrings

Medium

Given a string, your task is to count how many palindromic substrings in this string.

The substrings with different start indexes or end indexes are counted as different substrings even they consist of same characters.

**Example 1:**

**Input:** "abc"

**Output:** 3

**Explanation:** Three palindromic strings: "a", "b", "c".

**Example 2:**

**Input:** "aaa"

**Output:** 6

**Explanation:** Six palindromic strings: "a", "a", "a", "aa", "aa", "aaa".

**Note:**

1. The input string length won't exceed 1000.

### 648. Replace Words

MediumIn English, we have a concept called root, which can be followed by some other words to form another longer word - let's call this word successor. For example, the root an, followed by other, which can form another word another.

Now, given a dictionary consisting of many roots and a sentence. You need to replace all the successor in the sentence with the root forming it. If a successor has many roots can form it, replace it with the root with the shortest length.

You need to output the sentence after the replacement.

**Example 1:**

**Input:** dict = ["cat", "bat", "rat"]

sentence = "the cattle was rattled by the battery"

**Output:** "the cat was rat by the bat"

**Note:**

1. The input will only have lower-case letters.
2. 1 <= dict words number <= 1000
3. 1 <= sentence words number <= 1000
4. 1 <= root length <= 100
5. 1 <= sentence words length <= 1000

### 649. Dota2 Senate

Medium

In the world of Dota2, there are two parties: the Radiant and the Dire.

The Dota2 senate consists of senators coming from two parties. Now the senate wants to make a decision about a change in the Dota2 game. The voting for this change is a round-based procedure. In each round, each senator can exercise one of the two rights:

1. Ban one senator's right:  
   A senator can make another senator lose **all his rights** in this and all the following rounds.
2. Announce the victory:  
   If this senator found the senators who still have rights to vote are all from **the same party**, he can announce the victory and make the decision about the change in the game.

Given a string representing each senator's party belonging. The character 'R' and 'D' represent the Radiant party and the Dire party respectively. Then if there are n senators, the size of the given string will be n.

The round-based procedure starts from the first senator to the last senator in the given order. This procedure will last until the end of voting. All the senators who have lost their rights will be skipped during the procedure.

Suppose every senator is smart enough and will play the best strategy for his own party, you need to predict which party will finally announce the victory and make the change in the Dota2 game. The output should be Radiant or Dire.

**Example 1:**

**Input:** "RD"

**Output:** "Radiant"

**Explanation:** The first senator comes from Radiant and he can just ban the next senator's right in the round 1.

And the second senator can't exercise any rights any more since his right has been banned.

And in the round 2, the first senator can just announce the victory since he is the only guy in the senate who can vote.

**Example 2:**

**Input:** "RDD"

**Output:** "Dire"

**Explanation:**

The first senator comes from Radiant and he can just ban the next senator's right in the round 1.

And the second senator can't exercise any rights anymore since his right has been banned.

And the third senator comes from Dire and he can ban the first senator's right in the round 1.

And in the round 2, the third senator can just announce the victory since he is the only guy in the senate who can vote.

**Note:**

1. The length of the given string will in the range [1, 10,000].

### 650. 2 Keys Keyboard

Medium

Initially on a notepad only one character 'A' is present. You can perform two operations on this notepad for each step:

1. Copy All: You can copy all the characters present on the notepad (partial copy is not allowed).
2. Paste: You can paste the characters which are copied **last time**.

Given a number n. You have to get **exactly** n 'A' on the notepad by performing the minimum number of steps permitted. Output the minimum number of steps to get n 'A'.

**Example 1:**

**Input:** 3

**Output:** 3

**Explanation:**

Intitally, we have one character 'A'.

In step 1, we use **Copy All** operation.

In step 2, we use **Paste** operation to get 'AA'.

In step 3, we use **Paste** operation to get 'AAA'.

**Note:**

1. The n will be in the range [1, 1000].

### 652. Find Duplicate Subtrees

Medium

Given a binary tree, return all duplicate subtrees. For each kind of duplicate subtrees, you only need to return the root node of any **one** of them.

Two trees are duplicate if they have the same structure with same node values.

**Example 1:**

1

/ \

2 3

/ / \

4 2 4

/

4

The following are two duplicate subtrees:

2

/

4

and

4

Therefore, you need to return above trees' root in the form of a list.

### 653. Two Sum IV - Input is a BST

Easy

1243130Add to ListShare

Given a Binary Search Tree and a target number, return true if there exist two elements in the BST such that their sum is equal to the given target.

**Example 1:**

**Input:**

5

/ \

3 6

/ \ \

2 4 7

Target = 9

**Output:** True

**Example 2:**

**Input:**

5

/ \

3 6

/ \ \

2 4 7

Target = 28

**Output:** False

### 654. Maximum Binary Tree

Medium

Given an integer array with no duplicates. A maximum tree building on this array is defined as follow:

1. The root is the maximum number in the array.
2. The left subtree is the maximum tree constructed from left part subarray divided by the maximum number.
3. The right subtree is the maximum tree constructed from right part subarray divided by the maximum number.

Construct the maximum tree by the given array and output the root node of this tree.

**Example 1:**

**Input:** [3,2,1,6,0,5]

**Output:** return the tree root node representing the following tree:

6

/ \

3 5

\ /

2 0

\

1

**Note:**

1. The size of the given array will be in the range [1,1000].

### 655. Print Binary Tree

Medium

Print a binary tree in an m\*n 2D string array following these rules:

1. The row number m should be equal to the height of the given binary tree.
2. The column number n should always be an odd number.
3. The root node's value (in string format) should be put in the exactly middle of the first row it can be put. The column and the row where the root node belongs will separate the rest space into two parts (**left-bottom part and right-bottom part**). You should print the left subtree in the left-bottom part and print the right subtree in the right-bottom part. The left-bottom part and the right-bottom part should have the same size. Even if one subtree is none while the other is not, you don't need to print anything for the none subtree but still need to leave the space as large as that for the other subtree. However, if two subtrees are none, then you don't need to leave space for both of them.
4. Each unused space should contain an empty string "".
5. Print the subtrees following the same rules.

**Example 1:**

**Input:**

1

/

2

**Output:**

[["", "1", ""],

["2", "", ""]]

**Example 2:**

**Input:**

1

/ \

2 3

\

4

**Output:**

[["", "", "", "1", "", "", ""],

["", "2", "", "", "", "3", ""],

["", "", "4", "", "", "", ""]]

**Example 3:**

**Input:**

1

/ \

2 5

/

3

/

4

**Output:**

[["", "", "", "", "", "", "", "1", "", "", "", "", "", "", ""]

["", "", "", "2", "", "", "", "", "", "", "", "5", "", "", ""]

["", "3", "", "", "", "", "", "", "", "", "", "", "", "", ""]

["4", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]]

**Note:** The height of binary tree is in the range of [1, 10].

### 657. Robot Return to Origin

Easy

There is a robot starting at position (0, 0), the origin, on a 2D plane. Given a sequence of its moves, judge if this robot **ends up at (0, 0)** after it completes its moves.

The move sequence is represented by a string, and the character moves[i] represents its ith move. Valid moves are R (right), L (left), U (up), and D (down). If the robot returns to the origin after it finishes all of its moves, return true. Otherwise, return false.

**Note**: The way that the robot is "facing" is irrelevant. "R" will always make the robot move to the right once, "L" will always make it move left, etc. Also, assume that the magnitude of the robot's movement is the same for each move.

**Example 1:**

**Input:** "UD"

**Output:** true

**Explanation**: The robot moves up once, and then down once. All moves have the same magnitude, so it ended up at the origin where it started. Therefore, we return true.

**Example 2:**

**Input:** "LL"

**Output:** false

**Explanation**: The robot moves left twice. It ends up two "moves" to the left of the origin. We return false because it is not at the origin at the end of its moves.

### 658. Find K Closest Elements

Medium

Given a sorted array, two integers k and x, find the k closest elements to x in the array. The result should also be sorted in ascending order. If there is a tie, the smaller elements are always preferred.

**Example 1:**

**Input:** [1,2,3,4,5], k=4, x=3

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

**Example 2:**

**Input:** [1,2,3,4,5], k=4, x=-1

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

**Note:**

1. The value k is positive and will always be smaller than the length of the sorted array.
2. Length of the given array is positive and will not exceed 104
3. Absolute value of elements in the array and x will not exceed 104

### 659. Split Array into Consecutive Subsequences

Medium

Given an array nums sorted in ascending order, return true if and only if you can split it into 1 or more subsequences such that each subsequence consists of consecutive integers and has length at least 3.

**Example 1:**

**Input:** [1,2,3,3,4,5]

**Output:** True

**Explanation:**

You can split them into two consecutive subsequences :

1, 2, 3

3, 4, 5

**Example 2:**

**Input:** [1,2,3,3,4,4,5,5]

**Output:** True

**Explanation:**

You can split them into two consecutive subsequences :

1, 2, 3, 4, 5

3, 4, 5

**Example 3:**

**Input:** [1,2,3,4,4,5]

**Output:** False

**Constraints:**

* 1 <= nums.length <= 10000

### 661. Image Smoother

Easy

Given a 2D integer matrix M representing the gray scale of an image, you need to design a smoother to make the gray scale of each cell becomes the average gray scale (rounding down) of all the 8 surrounding cells and itself. If a cell has less than 8 surrounding cells, then use as many as you can.

**Example 1:**

**Input:**

[[1,1,1],

[1,0,1],

[1,1,1]]

**Output:**

[[0, 0, 0],

[0, 0, 0],

[0, 0, 0]]

**Explanation:**

For the point (0,0), (0,2), (2,0), (2,2): floor(3/4) = floor(0.75) = 0

For the point (0,1), (1,0), (1,2), (2,1): floor(5/6) = floor(0.83333333) = 0

For the point (1,1): floor(8/9) = floor(0.88888889) = 0

**Note:**

1. The value in the given matrix is in the range of [0, 255].
2. The length and width of the given matrix are in the range of [1, 150].

### 662. Maximum Width of Binary Tree

Medium

Given a binary tree, write a function to get the maximum width of the given tree. The width of a tree is the maximum width among all levels. The binary tree has the same structure as a **full binary tree**, but some nodes are null.

The width of one level is defined as the length between the end-nodes (the leftmost and right most non-null nodes in the level, where the null nodes between the end-nodes are also counted into the length calculation.

**Example 1:**

**Input:**

1

/ \

3 2

/ \ \

5 3 9

**Output:** 4

**Explanation:** The maximum width existing in the third level with the length 4 (5,3,null,9).

**Example 2:**

**Input:**

1

/

3

/ \

5 3

**Output:** 2

**Explanation:** The maximum width existing in the third level with the length 2 (5,3).

**Example 3:**

**Input:**

1

/ \

3 2

/

5

**Output:** 2

**Explanation:** The maximum width existing in the second level with the length 2 (3,2).

**Example 4:**

**Input:**

1

/ \

3 2

/ \

5 9

/ \

6 7

**Output:** 8

**Explanation:**The maximum width existing in the fourth level with the length 8 (6,null,null,null,null,null,null,7).

**Note:** Answer will in the range of 32-bit signed integer.

### 664. Strange Printer

Hard

There is a strange printer with the following two special requirements:

1. The printer can only print a sequence of the same character each time.
2. At each turn, the printer can print new characters starting from and ending at any places, and will cover the original existing characters.

Given a string consists of lower English letters only, your job is to count the minimum number of turns the printer needed in order to print it.

**Example 1:**

**Input:** "aaabbb"

**Output:** 2

**Explanation:** Print "aaa" first and then print "bbb".

**Example 2:**

**Input:** "aba"

**Output:** 2

**Explanation:** Print "aaa" first and then print "b" from the second place of the string, which will cover the existing character 'a'.

**Hint**: Length of the given string will not exceed 100.

### 665. Non-decreasing Array

Easy

Given an array nums with n integers, your task is to check if it could become non-decreasing by modifying **at most** 1 element.

We define an array is non-decreasing if nums[i] <= nums[i + 1] holds for every i (0-based) such that (0 <= i <= n - 2).

**Example 1:**

**Input:** nums = [4,2,3]

**Output:** true

**Explanation:** You could modify the first 4 to 1 to get a non-decreasing array.

**Example 2:**

**Input:** nums = [4,2,1]

**Output:** false

**Explanation:** You can't get a non-decreasing array by modify at most one element.

**Constraints:**

* 1 <= n <= 10 ^ 4
* - 10 ^ 5 <= nums[i] <= 10 ^ 5

### 667. Beautiful Arrangement II

Medium

Given two integers n and k, you need to construct a list which contains n different positive integers ranging from 1 to n and obeys the following requirement:  
Suppose this list is [a1, a2, a3, ... , an], then the list [|a1 - a2|, |a2 - a3|, |a3 - a4|, ... , |an-1 - an|] has exactly k distinct integers.

If there are multiple answers, print any of them.

**Example 1:**

**Input:** n = 3, k = 1

**Output:** [1, 2, 3]

**Explanation:** The [1, 2, 3] has three different positive integers ranging from 1 to 3, and the [1, 1] has exactly 1 distinct integer: 1.

**Example 2:**

**Input:** n = 3, k = 2

**Output:** [1, 3, 2]

**Explanation:** The [1, 3, 2] has three different positive integers ranging from 1 to 3, and the [2, 1] has exactly 2 distinct integers: 1 and 2.

**Note:**

1. The n and k are in the range 1 <= k < n <= 104.

### 668. Kth Smallest Number in Multiplication Table

Hard

Nearly every one have used the [Multiplication Table](https://en.wikipedia.org/wiki/Multiplication_table). But could you find out the k-th smallest number quickly from the multiplication table?

Given the height m and the length n of a m \* n Multiplication Table, and a positive integer k, you need to return the k-th smallest number in this table.

**Example 1:**

**Input:** m = 3, n = 3, k = 5

**Output:**

**Explanation:**

The Multiplication Table:

1 2 3

2 4 6

3 6 9

The 5-th smallest number is 3 (1, 2, 2, 3, 3).

**Example 2:**

**Input:** m = 2, n = 3, k = 6

**Output:**

**Explanation:**

The Multiplication Table:

1 2 3

2 4 6

The 6-th smallest number is 6 (1, 2, 2, 3, 4, 6).

**Note:**

1. The m and n will be in the range [1, 30000].
2. The k will be in the range [1, m \* n]

### 669. Trim a Binary Search Tree

Easy

Given a binary search tree and the lowest and highest boundaries as L and R, trim the tree so that all its elements lies in [L, R] (R >= L). You might need to change the root of the tree, so the result should return the new root of the trimmed binary search tree.

**Example 1:**

**Input:**

1

/ \

0 2

L = 1

R = 2

**Output:**

1

\

2

**Example 2:**

**Input:**

3

/ \

0 4

\

2

/

1

L = 1

R = 3

**Output:**

3

/

2

/

1

### 670. Maximum Swap

Medium

Given a non-negative integer, you could swap two digits **at most** once to get the maximum valued number. Return the maximum valued number you could get.

**Example 1:**

**Input:** 2736

**Output:** 7236

**Explanation:** Swap the number 2 and the number 7.

**Example 2:**

**Input:** 9973

**Output:** 9973

**Explanation:** No swap.

**Note:**

1. The given number is in the range [0, 108]
2. **671. Second Minimum Node In a Binary Tree**
3. Easy
4. 523759Add to ListShare
5. Given a non-empty special binary tree consisting of nodes with the non-negative value, where each node in this tree has exactly two or zero sub-node. If the node has two sub-nodes, then this node's value is the smaller value among its two sub-nodes. More formally, the property root.val = min(root.left.val, root.right.val) always holds.
6. Given such a binary tree, you need to output the **second minimum** value in the set made of all the nodes' value in the whole tree.
7. If no such second minimum value exists, output -1 instead.
8. **Example 1:**
9. **Input:**
10. 2
11. / \
12. 2 5
13. / \
14. 5 7
15. **Output:** 5
16. **Explanation:** The smallest value is 2, the second smallest value is 5.
18. **Example 2:**
19. **Input:**
20. 2
21. / \
22. 2 2
23. **Output:** -1
24. **Explanation:** The smallest value is 2, but there isn't any second smallest value.

### 672. Bulb Switcher II

Medium

There is a room with n lights which are turned on initially and 4 buttons on the wall. After performing exactly m unknown operations towards buttons, you need to return how many different kinds of status of the n lights could be.

Suppose n lights are labeled as number [1, 2, 3 ..., n], function of these 4 buttons are given below:

1. Flip all the lights.
2. Flip lights with even numbers.
3. Flip lights with odd numbers.
4. Flip lights with (3k + 1) numbers, k = 0, 1, 2, ...

**Example 1:**

**Input:** n = 1, m = 1.

**Output:** 2

**Explanation:** Status can be: [on], [off]

**Example 2:**

**Input:** n = 2, m = 1.

**Output:** 3

**Explanation:** Status can be: [on, off], [off, on], [off, off]

**Example 3:**

**Input:** n = 3, m = 1.

**Output:** 4

**Explanation:** Status can be: [off, on, off], [on, off, on], [off, off, off], [off, on, on].

**Note:** n and m both fit in range [0, 1000].

### 673. Number of Longest Increasing Subsequence

Medium

Given an unsorted array of integers, find the number of longest increasing subsequence.

**Example 1:**

**Input:** [1,3,5,4,7]

**Output:** 2

**Explanation:** The two longest increasing subsequence are [1, 3, 4, 7] and [1, 3, 5, 7].

**Example 2:**

**Input:** [2,2,2,2,2]

**Output:** 5

**Explanation:** The length of longest continuous increasing subsequence is 1, and there are 5 subsequences' length is 1, so output 5.

**Note:** Length of the given array will be not exceed 2000 and the answer is guaranteed to be fit in 32-bit signed int.

### 674. Longest Continuous Increasing Subsequence

Easy

Given an unsorted array of integers, find the length of longest continuous increasing subsequence (subarray).

**Example 1:**

**Input:** [1,3,5,4,7]

**Output:** 3

**Explanation:** The longest continuous increasing subsequence is [1,3,5], its length is 3.

Even though [1,3,5,7] is also an increasing subsequence, it's not a continuous one where 5 and 7 are separated by 4.

**Example 2:**

**Input:** [2,2,2,2,2]

**Output:** 1

**Explanation:** The longest continuous increasing subsequence is [2], its length is 1.

**Note:** Length of the array will not exceed 10,000.

### 675. Cut Off Trees for Golf Event

Hard

You are asked to cut off trees in a forest for a golf event. The forest is represented as a non-negative 2D map, in this map:

1. 0 represents the obstacle can't be reached.
2. 1 represents the ground can be walked through.
3. The place with number bigger than 1 represents a tree can be walked through, and this positive number represents the tree's height.

In one step you can walk in any of the four directions top, bottom, left and right also when standing in a point which is a tree you can decide whether or not to cut off the tree.

You are asked to cut off **all** the trees in this forest in the order of tree's height - always cut off the tree with lowest height first. And after cutting, the original place has the tree will become a grass (value 1).

You will start from the point (0, 0) and you should output the minimum steps **you need to walk** to cut off all the trees. If you can't cut off all the trees, output -1 in that situation.

You are guaranteed that no two trees have the same height and there is at least one tree needs to be cut off.

**Example 1:**

**Input:**

[

[1,2,3],

[0,0,4],

[7,6,5]

]

**Output:** 6

**Example 2:**

**Input:**

[

[1,2,3],

[0,0,0],

[7,6,5]

]

**Output:** -1

**Example 3:**

**Input:**

[

[2,3,4],

[0,0,5],

[8,7,6]

]

**Output:** 6

**Explanation:** You started from the point (0,0) and you can cut off the tree in (0,0) directly without walking.

**Constraints:**

* 1 <= forest.length <= 50
* 1 <= forest[i].length <= 50
* 0 <= forest[i][j] <= 10^9

### 676. Implement Magic Dictionary

Medium

Implement a magic directory with buildDict, and search methods.

For the method buildDict, you'll be given a list of non-repetitive words to build a dictionary.

For the method search, you'll be given a word, and judge whether if you modify **exactly** one character into **another** character in this word, the modified word is in the dictionary you just built.

**Example 1:**

Input: buildDict(["hello", "leetcode"]), Output: Null

Input: search("hello"), Output: False

Input: search("hhllo"), Output: True

Input: search("hell"), Output: False

Input: search("leetcoded"), Output: False

**Note:**

1. You may assume that all the inputs are consist of lowercase letters a-z.
2. For contest purpose, the test data is rather small by now. You could think about highly efficient algorithm after the contest.
3. Please remember to **RESET** your class variables declared in class MagicDictionary, as static/class variables are **persisted across multiple test cases**. Please see [here](https://leetcode.com/faq/#different-output) for more details.

### 677. Map Sum Pairs

Medium

Implement a MapSum class with insert, and sum methods.

For the method insert, you'll be given a pair of (string, integer). The string represents the key and the integer represents the value. If the key already existed, then the original key-value pair will be overridden to the new one.

For the method sum, you'll be given a string representing the prefix, and you need to return the sum of all the pairs' value whose key starts with the prefix.

**Example 1:**

Input: insert("apple", 3), Output: Null

Input: sum("ap"), Output: 3

Input: insert("app", 2), Output: Null

Input: sum("ap"), Output: 5

### 678. Valid Parenthesis String

Medium

Given a string containing only three types of characters: '(', ')' and '\*', write a function to check whether this string is valid. We define the validity of a string by these rules:

1. Any left parenthesis '(' must have a corresponding right parenthesis ')'.
2. Any right parenthesis ')' must have a corresponding left parenthesis '('.
3. Left parenthesis '(' must go before the corresponding right parenthesis ')'.
4. '\*' could be treated as a single right parenthesis ')' or a single left parenthesis '(' or an empty string.
5. An empty string is also valid.

**Example 1:**

**Input:** "()"

**Output:** True

**Example 2:**

**Input:** "(\*)"

**Output:** True

**Example 3:**

**Input:** "(\*))"

**Output:** True

**Note:**

1. The string size will be in the range [1, 100].

### 679. 24 Game

Hard

You have 4 cards each containing a number from 1 to 9. You need to judge whether they could operated through \*, /, +, -, (, ) to get the value of 24.

**Example 1:**

**Input:** [4, 1, 8, 7]

**Output:** True

**Explanation:** (8-4) \* (7-1) = 24

**Example 2:**

**Input:** [1, 2, 1, 2]

**Output:** False

**Note:**

1. The division operator / represents real division, not integer division. For example, 4 / (1 - 2/3) = 12.
2. Every operation done is between two numbers. In particular, we cannot use - as a unary operator. For example, with [1, 1, 1, 1] as input, the expression -1 - 1 - 1 - 1 is not allowed.
3. You cannot concatenate numbers together. For example, if the input is [1, 2, 1, 2], we cannot write this as 12 + 12.

### 680. Valid Palindrome II

Easy

Given a non-empty string s, you may delete **at most** one character. Judge whether you can make it a palindrome.

**Example 1:**

**Input:** "aba"

**Output:** True

**Example 2:**

**Input:** "abca"

**Output:** True

**Explanation:** You could delete the character 'c'.

**Note:**

1. The string will only contain lowercase characters a-z. The maximum length of the string is 50000.

### 682. Baseball Game

Easy

You're now a baseball game point recorder.

Given a list of strings, each string can be one of the 4 following types:

1. Integer (one round's score): Directly represents the number of points you get in this round.
2. "+" (one round's score): Represents that the points you get in this round are the sum of the last two valid round's points.
3. "D" (one round's score): Represents that the points you get in this round are the doubled data of the last valid round's points.
4. "C" (an operation, which isn't a round's score): Represents the last valid round's points you get were invalid and should be removed.

Each round's operation is permanent and could have an impact on the round before and the round after.

You need to return the sum of the points you could get in all the rounds.

**Example 1:**

**Input:** ["5","2","C","D","+"]

**Output:** 30

**Explanation:**

Round 1: You could get 5 points. The sum is: 5.

Round 2: You could get 2 points. The sum is: 7.

Operation 1: The round 2's data was invalid. The sum is: 5.

Round 3: You could get 10 points (the round 2's data has been removed). The sum is: 15.

Round 4: You could get 5 + 10 = 15 points. The sum is: 30.

**Example 2:**

**Input:** ["5","-2","4","C","D","9","+","+"]

**Output:** 27

**Explanation:**

Round 1: You could get 5 points. The sum is: 5.

Round 2: You could get -2 points. The sum is: 3.

Round 3: You could get 4 points. The sum is: 7.

Operation 1: The round 3's data is invalid. The sum is: 3.

Round 4: You could get -4 points (the round 3's data has been removed). The sum is: -1.

Round 5: You could get 9 points. The sum is: 8.

Round 6: You could get -4 + 9 = 5 points. The sum is 13.

Round 7: You could get 9 + 5 = 14 points. The sum is 27.

**Note:**

 The size of the input list will be between 1 and 1000.

 Every integer represented in the list will be between -30000 and 30000.

### 684. Redundant Connection

Medium

In this problem, a tree is an **undirected** graph that is connected and has no cycles.

The given input is a graph that started as a tree with N nodes (with distinct values 1, 2, ..., N), with one additional edge added. The added edge has two different vertices chosen from 1 to N, and was not an edge that already existed.

The resulting graph is given as a 2D-array of edges. Each element of edges is a pair [u, v] with u < v, that represents an **undirected** edge connecting nodes u and v.

Return an edge that can be removed so that the resulting graph is a tree of N nodes. If there are multiple answers, return the answer that occurs last in the given 2D-array. The answer edge [u, v] should be in the same format, with u < v.

**Example 1:**

**Input:** [[1,2], [1,3], [2,3]]

**Output:** [2,3]

**Explanation:** The given undirected graph will be like this:

1

/ \

2 - 3

**Example 2:**

**Input:** [[1,2], [2,3], [3,4], [1,4], [1,5]]

**Output:** [1,4]

**Explanation:** The given undirected graph will be like this:

5 - 1 - 2

| |

4 - 3

**Note:**

 The size of the input 2D-array will be between 3 and 1000.

 Every integer represented in the 2D-array will be between 1 and N, where N is the size of the input array.

### 685. Redundant Connection II

Hard

In this problem, a rooted tree is a **directed** graph such that, there is exactly one node (the root) for which all other nodes are descendants of this node, plus every node has exactly one parent, except for the root node which has no parents.

The given input is a directed graph that started as a rooted tree with N nodes (with distinct values 1, 2, ..., N), with one additional directed edge added. The added edge has two different vertices chosen from 1 to N, and was not an edge that already existed.

The resulting graph is given as a 2D-array of edges. Each element of edges is a pair [u, v] that represents a **directed** edge connecting nodes u and v, where u is a parent of child v.

Return an edge that can be removed so that the resulting graph is a rooted tree of N nodes. If there are multiple answers, return the answer that occurs last in the given 2D-array.

**Example 1:**

**Input:** [[1,2], [1,3], [2,3]]

**Output:** [2,3]

**Explanation:** The given directed graph will be like this:

1

/ \

v v

2-->3

**Example 2:**

**Input:** [[1,2], [2,3], [3,4], [4,1], [1,5]]

**Output:** [4,1]

**Explanation:** The given directed graph will be like this:

5 <- 1 -> 2

^ |

| v

4 <- 3

**Note:**

 The size of the input 2D-array will be between 3 and 1000.

 Every integer represented in the 2D-array will be between 1 and N, where N is the size of the input array.

### 686. Repeated String Match

Easy

Given two strings A and B, find the minimum number of times A has to be repeated such that B is a substring of it. If no such solution, return -1.

For example, with A = "abcd" and B = "cdabcdab".

Return 3, because by repeating A three times (“abcdabcdabcd”), B is a substring of it; and B is not a substring of A repeated two times ("abcdabcd").

**Note:**  
The length of A and B will be between 1 and 10000.

### 687. Longest Univalue Path

Easy

Given a binary tree, find the length of the longest path where each node in the path has the same value. This path may or may not pass through the root.

The length of path between two nodes is represented by the number of edges between them.

**Example 1:**

**Input:**

5

/ \

4 5

/ \ \

1 1 5

**Output:** 2

**Example 2:**

**Input:**

1

/ \

4 5

/ \ \

4 4 5

**Output:** 2

**Note:** The given binary tree has not more than 10000 nodes. The height of the tree is not more than 1000.

### 688. Knight Probability in Chessboard

Medium

On an NxN chessboard, a knight starts at the r-th row and c-th column and attempts to make exactly K moves. The rows and columns are 0 indexed, so the top-left square is (0, 0), and the bottom-right square is (N-1, N-1).

A chess knight has 8 possible moves it can make, as illustrated below. Each move is two squares in a cardinal direction, then one square in an orthogonal direction.



Each time the knight is to move, it chooses one of eight possible moves uniformly at random (even if the piece would go off the chessboard) and moves there.

The knight continues moving until it has made exactly K moves or has moved off the chessboard. Return the probability that the knight remains on the board after it has stopped moving.

**Example:**

**Input:** 3, 2, 0, 0

**Output:** 0.0625

**Explanation:** There are two moves (to (1,2), (2,1)) that will keep the knight on the board.

From each of those positions, there are also two moves that will keep the knight on the board.

The total probability the knight stays on the board is 0.0625.

**Note:**

* N will be between 1 and 25.
* K will be between 0 and 100.
* The knight always initially starts on the board.

### 689. Maximum Sum of 3 Non-Overlapping Subarrays

Hard

In a given array nums of positive integers, find three non-overlapping subarrays with maximum sum.

Each subarray will be of size k, and we want to maximize the sum of all 3\*k entries.

Return the result as a list of indices representing the starting position of each interval (0-indexed). If there are multiple answers, return the lexicographically smallest one.

**Example:**

**Input:** [1,2,1,2,6,7,5,1], 2

**Output:** [0, 3, 5]

**Explanation:** Subarrays [1, 2], [2, 6], [7, 5] correspond to the starting indices [0, 3, 5].

We could have also taken [2, 1], but an answer of [1, 3, 5] would be lexicographically larger.

**Note:**

* nums.length will be between 1 and 20000.
* nums[i] will be between 1 and 65535.
* k will be between 1 and floor(nums.length / 3).

### 690. Employee Importance

Easy

You are given a data structure of employee information, which includes the employee's **unique id**, his **importance value** and his **direct** subordinates' id.

For example, employee 1 is the leader of employee 2, and employee 2 is the leader of employee 3. They have importance value 15, 10 and 5, respectively. Then employee 1 has a data structure like [1, 15, [2]], and employee 2 has [2, 10, [3]], and employee 3 has [3, 5, []]. Note that although employee 3 is also a subordinate of employee 1, the relationship is **not direct**.

Now given the employee information of a company, and an employee id, you need to return the total importance value of this employee and all his subordinates.

**Example 1:**

**Input:** [[1, 5, [2, 3]], [2, 3, []], [3, 3, []]], 1

**Output:** 11

**Explanation:**

Employee 1 has importance value 5, and he has two direct subordinates: employee 2 and employee 3. They both have importance value 3. So the total importance value of employee 1 is 5 + 3 + 3 = 11.

**Note:**

1. One employee has at most one **direct** leader and may have several subordinates.
2. The maximum number of employees won't exceed 2000.

### 691. Stickers to Spell Word

Hard

We are given N different types of stickers. Each sticker has a lowercase English word on it.

You would like to spell out the given target string by cutting individual letters from your collection of stickers and rearranging them.

You can use each sticker more than once if you want, and you have infinite quantities of each sticker.

What is the minimum number of stickers that you need to spell out the target? If the task is impossible, return -1.

**Example 1:**

Input:

["with", "example", "science"], "thehat"

Output:

3

Explanation:

We can use 2 "with" stickers, and 1 "example" sticker.

After cutting and rearrange the letters of those stickers, we can form the target "thehat".

Also, this is the minimum number of stickers necessary to form the target string.

**Example 2:**

Input:

["notice", "possible"], "basicbasic"

Output:

-1

Explanation:

We can't form the target "basicbasic" from cutting letters from the given stickers.

**Note:**

 stickers has length in the range [1, 50].

 stickers consists of lowercase English words (without apostrophes).

 target has length in the range [1, 15], and consists of lowercase English letters.

 In all test cases, all words were chosen randomly from the 1000 most common US English words, and the target was chosen as a concatenation of two random words.

 The time limit may be more challenging than usual. It is expected that a 50 sticker test case can be solved within 35ms on average.

### 692. Top K Frequent Words

Medium

Given a non-empty list of words, return the *k* most frequent elements.

Your answer should be sorted by frequency from highest to lowest. If two words have the same frequency, then the word with the lower alphabetical order comes first.

**Example 1:**

**Input:** ["i", "love", "leetcode", "i", "love", "coding"], k = 2

**Output:** ["i", "love"]

**Explanation:** "i" and "love" are the two most frequent words.

Note that "i" comes before "love" due to a lower alphabetical order.

**Example 2:**

**Input:** ["the", "day", "is", "sunny", "the", "the", "the", "sunny", "is", "is"], k = 4

**Output:** ["the", "is", "sunny", "day"]

**Explanation:** "the", "is", "sunny" and "day" are the four most frequent words,

with the number of occurrence being 4, 3, 2 and 1 respectively.

**Note:**

1. You may assume *k* is always valid, 1 ≤ *k* ≤ number of unique elements.
2. Input words contain only lowercase letters.

**Follow up:**

1. Try to solve it in *O*(*n* log *k*) time and *O*(*n*) extra space.

### 693. Binary Number with Alternating Bits

Easy

Given a positive integer, check whether it has alternating bits: namely, if two adjacent bits will always have different values.

**Example 1:**

**Input:** 5

**Output:** True

**Explanation:**

The binary representation of 5 is: 101

**Example 2:**

**Input:** 7

**Output:** False

**Explanation:**

The binary representation of 7 is: 111.

**Example 3:**

**Input:** 11

**Output:** False

**Explanation:**

The binary representation of 11 is: 1011.

**Example 4:**

**Input:** 10

**Output:** True

**Explanation:**

The binary representation of 10 is: 1010.

### 695. Max Area of Island

Medium

Given a non-empty 2D array grid of 0's and 1's, an **island** is a group of 1's (representing land) connected 4-directionally (horizontal or vertical.) You may assume all four edges of the grid are surrounded by water.

Find the maximum area of an island in the given 2D array. (If there is no island, the maximum area is 0.)

**Example 1:**

[[0,0,1,0,0,0,0,1,0,0,0,0,0],

[0,0,0,0,0,0,0,1,1,1,0,0,0],

[0,1,1,0,1,0,0,0,0,0,0,0,0],

[0,1,0,0,1,1,0,0,**1**,0,**1**,0,0],

[0,1,0,0,1,1,0,0,**1**,**1**,**1**,0,0],

[0,0,0,0,0,0,0,0,0,0,**1**,0,0],

[0,0,0,0,0,0,0,1,1,1,0,0,0],

[0,0,0,0,0,0,0,1,1,0,0,0,0]]

Given the above grid, return 6. Note the answer is not 11, because the island must be connected 4-directionally.

**Example 2:**

[[0,0,0,0,0,0,0,0]]

Given the above grid, return 0.

**Note:** The length of each dimension in the given grid does not exceed 50.

### 696. Count Binary Substrings

Easy

Give a string s, count the number of non-empty (contiguous) substrings that have the same number of 0's and 1's, and all the 0's and all the 1's in these substrings are grouped consecutively.

Substrings that occur multiple times are counted the number of times they occur.

**Example 1:**

**Input:** "00110011"

**Output:** 6

**Explanation:** There are 6 substrings that have equal number of consecutive 1's and 0's: "0011", "01", "1100", "10", "0011", and "01".

Notice that some of these substrings repeat and are counted the number of times they occur.

Also, "00110011" is not a valid substring because **all** the 0's (and 1's) are not grouped together.

**Example 2:**

**Input:** "10101"

**Output:** 4

**Explanation:** There are 4 substrings: "10", "01", "10", "01" that have equal number of consecutive 1's and 0's.

**Note:**

 s.length will be between 1 and 50,000.

 s will only consist of "0" or "1" characters.

### 697. Degree of an Array

Easy

Given a non-empty array of non-negative integers nums, the **degree** of this array is defined as the maximum frequency of any one of its elements.

Your task is to find the smallest possible length of a (contiguous) subarray of nums, that has the same degree as nums.

**Example 1:**

**Input:** [1, 2, 2, 3, 1]

**Output:** 2

**Explanation:**

The input array has a degree of 2 because both elements 1 and 2 appear twice.

Of the subarrays that have the same degree:

[1, 2, 2, 3, 1], [1, 2, 2, 3], [2, 2, 3, 1], [1, 2, 2], [2, 2, 3], [2, 2]

The shortest length is 2. So return 2.

**Example 2:**

**Input:** [1,2,2,3,1,4,2]

**Output:** 6

**Note:**

 nums.length will be between 1 and 50,000.

 nums[i] will be an integer between 0 and 49,999.

### 698. Partition to K Equal Sum Subsets

Medium

Given an array of integers nums and a positive integer k, find whether it's possible to divide this array into k non-empty subsets whose sums are all equal.

**Example 1:**

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

**Output:** True

**Explanation:** It's possible to divide it into 4 subsets (5), (1, 4), (2,3), (2,3) with equal sums.

**Note:**

* 1 <= k <= len(nums) <= 16.
* 0 < nums[i] < 10000.

### 699. Falling Squares

Hard

On an infinite number line (x-axis), we drop given squares in the order they are given.

The i-th square dropped (positions[i] = (left, side\_length)) is a square with the left-most point being positions[i][0] and sidelength positions[i][1].

The square is dropped with the bottom edge parallel to the number line, and from a higher height than all currently landed squares. We wait for each square to stick before dropping the next.

The squares are infinitely sticky on their bottom edge, and will remain fixed to any positive length surface they touch (either the number line or another square). Squares dropped adjacent to each other will not stick together prematurely.

Return a list ans of heights. Each height ans[i] represents the current highest height of any square we have dropped, after dropping squares represented by positions[0], positions[1], ..., positions[i].

**Example 1:**

**Input:** [[1, 2], [2, 3], [6, 1]]

**Output:** [2, 5, 5]

**Explanation:**

After the first drop of positions[0] = [1, 2]: \_aa \_aa ------- The maximum height of any square is 2.

After the second drop of positions[1] = [2, 3]: \_\_aaa \_\_aaa \_\_aaa \_aa\_\_ \_aa\_\_ -------------- The maximum height of any square is 5. The larger square stays on top of the smaller square despite where its center of gravity is, because squares are infinitely sticky on their bottom edge.

After the third drop of positions[1] = [6, 1]: \_\_aaa \_\_aaa \_\_aaa \_aa \_aa\_\_\_a -------------- The maximum height of any square is still 5. Thus, we return an answer of [2, 5, 5].

**Example 2:**

**Input:** [[100, 100], [200, 100]]

**Output:** [100, 100]

**Explanation:** Adjacent squares don't get stuck prematurely - only their bottom edge can stick to surfaces.

**Note:**

* 1 <= positions.length <= 1000.
* 1 <= positions[i][0] <= 10^8.
* 1 <= positions[i][1] <= 10^6.

### 700. Search in a Binary Search Tree

Easy

Given the root node of a binary search tree (BST) and a value. You need to find the node in the BST that the node's value equals the given value. Return the subtree rooted with that node. If such node doesn't exist, you should return NULL.

For example,

Given the tree:

4

/ \

2 7

/ \

1 3

And the value to search: 2

You should return this subtree:

2

/ \

1 3

In the example above, if we want to search the value 5, since there is no node with value 5, we should return NULL.

Note that an empty tree is represented by NULL, therefore you would see the expected output (serialized tree format) as [], not null.