# Table of contents

[Table of contents 1](#_Toc51790358)

[Inbox 3](#_Toc51790359)

[Simple puzzles 5](#_Toc51790360)

[Inbox 6](#_Toc51790361)

[Data Structures 8](#_Toc51790362)

[Linked lists 9](#_Toc51790363)

[Tries 10](#_Toc51790364)

[Directed Graphs 12](#_Toc51790365)

[Undirected graphs 14](#_Toc51790366)

[BST 17](#_Toc51790367)

[Stacks 18](#_Toc51790368)

[Queues 19](#_Toc51790369)

[Heap 20](#_Toc51790370)

[Trees 21](#_Toc51790371)

[Matrix 23](#_Toc51790372)

[Segment trees 25](#_Toc51790373)

[Suffix arrays 26](#_Toc51790374)

[Greedy Approach 27](#_Toc51790375)

[Dynamic Programming 32](#_Toc51790376)

[Recursion and backtracking 36](#_Toc51790377)

[Binary Search 37](#_Toc51790378)

[Hashing 40](#_Toc51790379)

[Stacking 42](#_Toc51790380)

[Number theory 44](#_Toc51790381)

[Sorting and searching 46](#_Toc51790382)

[Bit Masking 47](#_Toc51790383)

[2 pointers 49](#_Toc51790384)

[Important Algorithms 50](#_Toc51790385)

[Geometry 53](#_Toc51790386)

[Divide and conquer 54](#_Toc51790387)

[Debugging 55](#_Toc51790388)

# Inbox

#### Range minimum queries – Segment Trees

#### [Coin jam](https://codingcompetitions.withgoogle.com/codejam/round/0000000000201bee/0000000000201b6d)

A *jam coin* is a string of **N** ≥ 2 digits with the following properties:

* Every digit is either 0 or 1.
* The first digit is 1 and the last digit is 1.
* If you interpret the string in any base between 2 and 10, inclusive, the resulting number is not prime

Given N and J, print J jam coins along with the proof?

#### [Integer Product](https://atcoder.jp/contests/agc047/tasks/agc047_a)

atcoder.jp

#### Regular expression matching

Given a string s and a pattern p. Check if s matches the pattern p. (p contains only . and \* and alphabets [a-z], s contains)

<https://leetcode.com/problems/regular-expression-matching/solution/>

#### Task scheduler

Given an array of alphabets which represents tasks. The CPU can execute the task or remain idle. After executing a task there is a cool down period for executing the same task (given n). You can change the order of the tasks. Find the minimum time taken by CPU for executing all the tasks.

<https://leetcode.com/problems/task-scheduler/>

# Simple puzzles

## Inbox

#### Element extermination

[**Question**](https://codeforces.com/contest/1375/problem/C): Given a permutation . You can choose any index such that and remove either from the permutation. Find whether you can attain permutation with only one element.

Ans is YES if else NO

#### [Replace by MEX](https://codeforces.com/contest/1375/problem/D)

**Question**: Given an array containing n integers between 0 and n (inclusive). In one operation you can change any element to the MEX (minimum excluded) of the array. In at most 2n operations you must make the array non – decreasing.

Print any one of the solutions as the sequence of operations taken.

**Solution**:

We want to make the array as [0, n – 1].

while (array not sorted)

{

Find mex = mex(array)

array[mex] = mex

}

When mex = n, put it in any index , such that .

#simple

#### Inversion swapsort

Given an array, list out inversions after correcting them you get array in non – descending order. Such that

1. The list **must include all the inversion pairs**
2. All the pairs in the list should be inversions

Solution:

For int i = n - 1 to 0

max\_idx = max\_index(0, i)

If (arr[max\_idx] > arr[i])

Swap(Max, i)

Inversion Swapsort: My logic was incorrect. Think about the correct logic

#### [Omkar and the baseball](http://codeforces.com/contest/1372/problem/C)

Given an array of positive integers. Choose a subarray and permute such that no element is in the same position as before. Find the minimum number of such permutations before the array is sorted.

**Solution**:

#### D: Prefix flip

**Question**: Given two binary strings a, b. convert into by at most 3n prefix flip operations. Prefix flip = chose any prefix and invert (1 → 0, 0 → 1) and **reverse** the prefix.

**Solution**:

Converting a → all zeroes can be done in at most n operations.

Convert a → all zeroes, convert b → all zeroes

<http://codeforces.com/contest/1382/problem/C2>

#### C: [Koa and the beach](http://codeforces.com/contest/1384/problem/B1)

**Question**: A beach consists of a shore (at 0) sea from 1 to n and island at n + 1. The height of the sea at is . Because of tide height increments 1 unit for k seconds and decrements for the next k seconds. Height at any given time is . Where the array p is . At t = 0, Koa is at the shore. Find whether he reaches the island?

**Solution**:

Create a graph with vertices **distance (0 to n+1)**, **tide level (1 to k)**, **flag** (whether increasing or decreasing). Now use dfs to find if possible or not?

#pending

# Data Structures

## Linked lists

#### Cycle in a linked list

Given a linked list header. Find whether the linked list contains cycles or not and if the cycle is present return the start of the cycle?

ptr1 = ptr2 = head

while (ptr1 != null)

{

if (count and ptr1 == ptr2)

{

ptr2 = head;

while (ptr1)

if (ptr1 == ptr2) return ptr1

ptr1 = ptr1->next

ptr2 = ptr2->next

}

count++

if (ptr2 and ptr2->next) ptr2 = ptr2->next->next

else break;

ptr1 = ptr1->next;

}

#### Intersection of linked lists

Given two linked lists. Find the intersection node of the two linked lists if they intersect.

## Tries

### Implementation

class Trie {

public:

    struct node {

        char ch;

        bool is\_word = false;

        vector<node \*> children;

        node (char ch)

        {

            this->ch = ch;

            is\_word = false;

            children.resize(alpha);

        }

    };

    node \*root;

    Trie() {

        root = new node('a' - 1);

    }

    void insert(string word) {

        node \*n = root;

        int i = 0;

        for (i = 0; i < word.length(); i++)

        {

            if (n->children[word[i] - 'a'])

                n = n->children[word[i] - 'a'];

            else break;

        }

        for (int j = i; j < word.length(); j++)

        {

            n->children[word[j] - 'a'] = new node(word[j]);

            n = n->children[word[j] - 'a'];

        }

        n->is\_word = true;

    }

    bool search(string word) {

        node \*n = root;

        for (int i = 0; i < word.length(); i++)

        {

            if (n->children[word[i] - 'a'])

                n = n->children[word[i] - 'a'];

            else return false;

        }

        return n->is\_word == true;

    }

    bool startsWith(string prefix) {

        node \*n = root;

        for (int i = 0; i < prefix.length(); i++)

        {

            if (n->children[prefix[i] - 'a'])

                n = n->children[prefix[i] - 'a'];

            else return false;

        }

        return true;

    }

};

## Directed Graphs

#### Detect cycle in directed graph

Given a directed graph, find whether the graph contains the cycle or not?

bool cycle(i):

vis1[i] = true;

vis2[i] = true; // recursion visit

is\_cycle = false

for j in neighbors[i]

if vis2[j] return true;

if !vis1[j] is\_cycle |= cycle(j)

vis2[i] = false;

return is\_cycle

for i in nodes:

if !vis1[i]: if cycle(i): return true

return false

#### Topological sort (leetcode)

Given a directed graph, return the topological sort of the graph or nothing if it does not exist.

vector<int> deg(n)

queue<int> que // fill with vertices whose indegree is zero

while (que.size())

idx = que.front(), que.pop()

ans.push\_back()

for i in graph[idx]:

deg[i]--

if deg[i] == 0: que.push(i)

if (ans.size() == n) return ans

else return {}

#### Longest path in directed acyclic graph

Given a directed acyclic graph, find the length of the longest path from any node?

int dfs(i, dist)

if (dist[i]) return dist[i]

ans = 1

for j in neighbors[i]:

if (!dist[j]) dfs(j, dist)

ans = max(ans, 1 + dist[j])

dist[i] = ans

return ans

for i in nodes:

if (!dist[j])

ans = max(ans, dfs(i, dist))

return ans

#### Longest string chain (google interview: Errichto)

Given a list of strings. Find the longest string chain where you can add a letter to a string at any position and reach a new string. (all strings should be present in the word list)

Create a graph of strings.

Find the longest path in the directed acyclic graph. O(n)

## Undirected graphs

#### Return the path from s to t in a graph

Given an undirected graph, return the list of nodes in the simple path between s and t.

First need to calculate the distance array to t.

We can get the path using the distance array.

#### Detect cycle in undirected graph

bool cycle(i):

vis[i] = true;

is\_cycle = false

for j in neighbors[i]

if vis[j] : return true

else is\_cycle |= cycle(j)

return is\_cycle

for i in nodes:

if !vis[i]: if cycle(i): return true

return false

#### Deep clone an undirected connected graph (hashing)

unordered\_map<Node\*, Node\*> mp;

Node\* cloneGraph(Node\* node)

{

if (!node) return nullptr;

Node \*n = new Node(node->val);

mp[node] = n;

for (auto p : node->neighbors)

{

if (mp[p])

n->neighbors.push\_back(mp[p]);

else

{

Node \*nb = cloneGraph(p);

n->neighbors.push\_back(nb);

}

}

return n;

}

#### Remove maximum number of edges to keep the graph fully traversal (leetcode)

Alice and bob have an undirected graph of n nodes. It has edges of three types.

1. Type I, only Alice can traverse through this
2. Type II, only bob can traverse through this
3. Type III, both Alice and bob can traverse through this

Union find algorithm

ans = total edges – useful edges

union u1(n), u2(n)

for edge in type III

useful\_edges += u1.unite(edge.a, edge.b), u2.unite(edge.a, edge.b)

for edge in type I

useful\_edges += u1.unite(edge.a, edge.b)

for edge in type II

useful\_edges += u2.unite(edge.a, edge.b)

if (u1.components > 1 or u2.components > 1) return -1

else return total\_edges – useful\_edges

#### Longest increasing path in a matrix

Given a matrix, you can move left, right, up, and down. Find the length of the longest increasing sequence.

Create a directed graph from the matrix.

For e in matrix:

Add directed edge to all neighbors greater than e.

vector<int> dist(n \* m, 0);

for (int i = 0; i < n \* m; i++)

if (!dist[i]) dfs(i, graph, dist)

ans = max(ans, dist[i])

return ans

And I think the complexity will be O(n \* m)

#### [Beauty of the tree](https://codingcompetitions.withgoogle.com/kickstart/round/000000000019ff08/0000000000386edd)

Amadea and Bilva are decorating a rooted tree containing **N** nodes, labelled from 1 to **N**. Node 1 is the root of the tree, and all other nodes have a node with a numerically smaller label as their parent.

Amadea and Bilva's decorate the tree as follows:

1. Amadea picks a node of the tree uniformly at random and paints it. Then, she travels up the tree painting every **A**-th node until she reaches the root.
2. Bilva picks a node of the tree uniformly at random and paints it. Then, she travels up the tree painting every **B**-th node until she reaches the root.

The *beauty* of the tree is equal to the number of nodes painted *at least once* by either Amadea or Bilva. Note that even if they both paint a node, it only counts once.

What is the [expected](https://en.wikipedia.org/wiki/Expected_value) beauty of the tree?

**Solution**:

Where denotes the probability that node gets colored.

Since are independent events.

For each node:

Travel to the root and increment nodes along the way

### Bipartite graph

Given an undirected graph, check if it is bipartite graph or not?

## BST

#### Custom data structure

Given pairs of **index** and **value**. Design a data structure that supports

1. Changing value of index
2. Adding index and value
3. Get the index of least value

I think it can be done by simply maintaining a set of pairs with a custom comparator. And an additional map of index and value (will be used for deleting an index from the set).

set<pair<int, int>, compare> s; map<int, int> mp;

adding is O(logn)

changing value is like deleting (O(logn)) and adding (O(logn)).

getting index of least value is O(1) i.e. s.begin().first

#### Infix to postfix

Given an expression (standard infix) convert it into postfix form.

Generate n + 1 strings, such that lengths of the longest common prefixes between the consecutive strings given as .

**Solution**:

string s(100, ‘a’);

for (int i = 0; i < n; i++)

{

int x;

cin ≫ x;

if (s[x] == ‘a’) s[x] = ‘b’;

else s[x] = ‘a’;

cout ≪ s ≪ endl;

}

## Stacks

### Recursion vs iteration

Convert any recursion to iteration using stack.

### DFS

## Queues

### BFS

## Heap

## Trees

#### Link cut centroids

Given a tree, a **centroid** is defined as the vertex upon removing it, the size of the largest connected component is least. You can remove one edge and add one edge. You must make the centroid unique. Print the edge removed and the edge added.

**Logic:** There can be at most two centroids if they are they must be connect by an edge.

#### Minimum height trees

Given a tree, when a node is selected as the root, the height of the tree is defined as the longest path from the root. Among n trees, return **the list of roots whose height is minimum**.

**Logic**: removing leaves one by one util it is empty

vector<int> ans

vector<int> deg (contains the degrees of the nodes in the tree)

queue<int> leaves

// compute leaves, degs first

while (leaves.size())

ans.clear()

int size = leaves.size()

for i = 0 to size – 1

idx = leaves.front(), leaves.pop(), ans.push\_back(idx)

for j : neighbors[i]

deg[j]--

if deg[j]] == 1: leaves.push(j)

return ans

#### Count complete tree nodes

Given a complete binary tree. Find the number of nodes in the tree.

I think the complexity of the algo should be less than O(n)

Since the tree is complete, we just need to figure out the number of nodes in the last level. If d is the depth. Last level can have elements 1 to 2^d.

We can do binary search to figure out the number of nodes the last level. Complexity will be O(d^2).

#### Diameter of the tree (codeforces)

Given a tree find the diameter of the tree.

**Logic**: find the longest from 1 → u, now find the longest path from u → v (diameter)

## Matrix

#### Shortest distance between two points

Given a matrix consisting of 0’s and 1’s. 0 for obstacle. Given two coordinates, find the shortest path length between them?

#### Set matrix zeroes

Given a matrix containing integers. For all the cells containing zero, replace cells in the same row or in the same column with zero. Do this **in-place**.

Use the first row and the storage.

bool first\_zero = first row has zero

For each column in matrix:

if column has zero, matrix[0][col] = 0

for each row in matrix:

if row has zero: set all elements in row to zero

for each element

if matrix[0][col] == 0: matrix[row][col] = 0

if first\_zero:

set all elements in first row to zero.

#### E Bomber

Given a matrix of size H, W. Also given the list of locations where bombs were placed. You have igniter. You place it anywhere in the matrix once. It can ignite all the bombs in its row and all bombs in its column. Find the maximum number of bombs you can ignite.

Creating two maps one for row and one for column.

Mpr[i] – # points in ith row

Mpc[i] - # points in the ith column

Find for which mpr, mpc becomes maximum.

<https://atcoder.jp/contests/abc176/tasks/abc176_e>

Given a matrix of size . Containing any of ‘.’ - empty, ‘#’ – wall, ‘G’ – good, ‘B’ – bad. And the cell at is always ‘.’. (Only cell having a common edge are connected, and you cannot pass through ‘#’)

**You can add one or more walls in empty cells**. Make it such that only good people will be able to escape and not a single bad person escape.

**Solution**:

Block all the neighboring cells of bad people. Check if all good people can escape.

Given a matrix (of size n × m) containing 0’s and 1’s. you can change the cells from 0 to 1 or 1 to 0. What is the least number of cell changes such all the paths from (0, 0) → (n - 1, m - 1) are palindromic?

**Solution**:

Given a matrix of size n × m. The matrix is called good if for every cell, if it has number k > 0, then it should have exactly k positive integers. If it has 0, nothing needed.

You can increase any cell by 1 any number of times. Print the good matrix after conversion. Print “NO” if not possible.

**Solution**:

Each cell should have . Otherwise there is no solution. If yes,

Then we can make

## Segment trees

## Suffix arrays

# Greedy Approach

### Inbox

#### Jump game II

Given an array of non-negative integers, you are initially positioned at the first index of the array. Each element in the array represents your maximum jump length at that position. Your goal is to reach the last index in the minimum number of jumps.

The problem is acutally greedy

max\_steps = nums[0]

max\_reach = nums[0]

jump = 1

for i = 1 to n – 1

if i > max\_steps:

jump++

max\_steps = max\_reach

max\_reach = max(max\_reach, i + nums[i])

return jump

#### Dist Max (Atcoder)

Given n points in the 2d plane. The Manhattan distance between two points is defined as . Find the largest Manhattan distance between any two points.

**Logic:** only two possibilities same sign or different sign or

vector<int> v1, v2;

For each point in points:

v1.insert(x + y)

v2.insert(x - y)

ans = max(ans, max(v1) - min(v1))

ans = max(ans, max(v2) - min(v2))

return ans;

#datastructures, #greedy

### Line sweep

### Games: Two **Player**

#### Stoned game (codeforces)

There are n piles of stones, initially pile consists of stones.

T and HL take turns and pick a stone from any of the pile which is not chosen in the previous move. The game ends when either of them is unable to choose a stone. If both play optimally, determine who wins.

If ∃ a\_i > sum / 2

return T

else

if sum is odd return T

else return HL

#### Maximum product subarray (leetcode)

Given an integer array, find the maximum product of all the subarrays.

mine = nums[0], maxe = nums[0]

int ans = INT\_MIN

for i = 1 to nums.size() – 1

temp\_min = mine

mine = min(nums[i], min(mine \* nums[i], maxe \* nums[i]))

maxe = max(nums[i], max(mine \* nums[i], maxe \* nums[i]))

ans = max(ans, maxe)

## Sliding window

#### Longest Substring with At Most Two Distinct Characters (leetcode)

Given a string, find the longest substring which has at most 2 distinct characters.

Sliding window approach.

map<int, int> mp

ptr = 0

for (int i = 0; i < str.length(); i++)

{

if (mp.find(str[i]) != mp.end()) mp[str[i]]++;

else if (mp.size() < 2) mp[str[i]]++;

else

{

ans = max(ans, sum(mp.values())

while (mp.size() != 1)

{

mp[str[ptr]]--;

if (mp[str[ptr]] == 0) mp.erase(str[ptr])

ptr++;

}

}

}

ans = max(ans, sum(mp.values())

#### Rational lee (codeforces)

Given n integers, . You need to group them into k groups of sizes . Such that

Is maximized.

First sort the numbers

All the larger should be in the groups of least sizes.

All the smaller numbers should be in the groups of larger sizes.

Pass 1:

Insert the largest numbers in the increasing order the groups sizes

Pass 2:

Insert the smallest numbers in the decreasing order of the group sizes.

Topics: Greedy approach.

Todo: Code this

#### Container with most water

Given an array of integers denoting the height of poles at locations 1, 2, … n. Find maximum area of the container you can make using two poles.

Left = 0, right = n – 1

While (right > left)

ans = max(ans, min(left, right) \* (right – left))

If (left < right) left++

else right++

Two pointer technique: <https://leetcode.com/problems/container-with-most-water/>

#### Longest valid parenthesis

Given a string containing open and closed parenthesis. Find the length of the longest substring which is valid parenthesis?

open = closed = 0

For i = 0 to str.length()

if (str[i] == ‘(’) open++

if (str[i] == ‘)’) closed++

if (closed > open) closed = open = 0

else if (closed == open) ans = max(ans, 2 \* closed)

Now **reverse and flip** the string and do the above one more time.

#### E Reading Books

Given list of n books. Each book takes time for both Alice and bob to read. And each book has two Booleans denoting whether Alice or bob likes, respectively. **Select some books** such that **at least k books are liked by Alice and at least k books are liked by bob** and the sum of times of reading the books is minimized.

**Solution**:

Sort the books according the to the time and traverse.

#### String Transformation

**Question**: Given two strings A and B of length n made of first 20 characters in the alphabet. In one move.

1. Select subset of indices such that
2. Select alphabet
3. Make

Transform A → B in least number of moves? Print the #moves.

**Solution**:

if not

return not possible

ans = 0;

for c = ‘a’ to ‘t’

∀ i, s.t. A[i] ≠ B[i] and A[i] = c add to temp vector.

Min = min(B[i] ∀ i ∈ temp)

A[i] = Min ∀ i ∈ temp

#### Minimum window substring

Given two strings s and t. Find the smallest substring in s, which contains all the characters in t.

<https://leetcode.com/problems/minimum-window-substring/>

## Two pointers

# Dynamic Programming

The **non-aftereffect property**: Does not affect the state of the subsystem in future when determining the state of another subsystem.

#### Maximal rectangle

Given a matrix containing 0’s and 1’s. Find the area of the largest rectangle containing all 1’s.

cum\_row

For each row in matrix

add(row, cum\_row)

ans = max(ans, max\_rectangle\_histogram(cum\_row))

return ans

### Two Pointers

#### Two platforms (codeforces)

Given n coordinates (x, y). You have two horizontal platforms of length k. You need to place the two platforms at integer coordinates such that when the points fall vertically, they both catch maximum number of points. (n ≤ 1e5, x, y, k ≤ 1e9)

We always place the platforms on the x – axis. We can discard y – coordinates and just consider x – coordinates.

For each point

Calculate the #points to left with in ≤ k

Calculate the #points to right with in ≤ k

Build **prefix maximum** array on left

Build **suffix maximum** array on right

For i = 1 to n

ans = max(ans, left[i] + right[i + 1])

return ans

#### Best time to buy and sell a stock III

Given an array of numbers denoting the time series values of a stock. Find the maximum profit possible with at most two transaction? (you cannot engage in multiple transactions at the same time, you must sell a stock before you buy again)

vector<int> left(n), right(n); // maximum profit iterating to left and iterating to right respectively.

int ans = right[0]

for i = 0 to n – 2

ans = max(ans, left[i] + right[i + 1])

ans = max(ans, left[n – 1])

### Edit Distance

Given two strings a, b. You have 3 kinds of operations.

1. Insert a string
2. Delete a string
3. Replace a string

Find the minimum number of operations need to make the strings equal.

int dp[m][n]

if s[i] == s[j]: dp[i][j] = min(dp[i – 1][j – 1], dp[i][j – 1], dp[i – 1][j])

else

dp[i][j] = dp[i][j] = min(dp[i – 1][j – 1] + 1, dp[i][j – 1] + 1, dp[i – 1][j] + 1)

return dp[m – 1][n – 1]

#### Coin change

Given different **denominations** and total money needed ***amount***. Find the fewest number of coins that you need to make up the *amount* (**repetitions are allowed**)*.* If it is not possible return -1

dp[amount] = min(dp[amount – den[i]] + 1) for all i

worst case complexity will be O(n \* amount)

#### Jump game

Given an array of non-negative integers, you are initially positioned at the first index of the array. Each element in the array represents your maximum jump length at that position. Determine if you can reach the last index.

dp[i] denotes whether he will be able to reach the end.

cum[i] = cum[i + 1]

reach = min(nums.size(), i + nums[i] + 1)

if (cum[i + 1] – cum[reach]) dp[i] = true, cum[i]++

return dp[0]

#### [Colored Rectangles](https://codeforces.com/contest/1398/problem/D)

Given **R** pairs of red sticks, **G** pairs of green sticks , **B** pairs of blue sticks. You can select **two pairs** of **different color** and form a rectangle and add the area to **total area**. Find the maximum total area possible?

**Solution**:

dp[r][g][b] = max(dp[r – 1][g – 1][b] + len[r] \* len[g],

dp[r][g – 1][b - 1] + len[g] \* len[b]

dp[r – 1][g][b - 1] + len[r] \* len[b])

### knapsack

#### weight possible

Given n fruits, each of weight w[i], find whether a total weight W can be achieved with the fruits (repetitions not allowed)

bool can[max\_w];

for (int i = 0; i < n; i++)

for (int j = W; j ≥ w[i]; j--)

can[j] |= can[j – w[i]];

return can[W];

#### Split array largest sum

Given an array of n elements. You need to group them into m subarrays such that the largest sum in any group is least possible.

### others

#### Number of palindromic substrings

Given a string s. Find the number of palindromic substrings in

Use dynamic programming

dp[i][j] = dp[i + 1][j – 1] and s[i] == s[j]

for i = n to 1

for j = i to n

dp[i][j] = dp[i + 1][j – 1] and s[i] == s[j]

#### Three sequences (codeforces)

Given an array of integers. You need to come up with two other arrays and . Such that all elements in are non-decreasing and all elements in are non-increasing and max(, ) is least possible. There are n queries. Each query is of the form i.e. add to each to . For each query print the least possible .

**Logic**: , so maintain a diff array.

# Recursion and backtracking

#### Return the list of all combinations

Given an array of elements, return the list of all the combinations possible from the given array?

void combination(vector<vector<int>> &all, vector<int> &ans, vector<int> &arr, int i)

{

    if (i == arr.size())

    {

        all.push\_back(ans);

        return;

    }

    combination(all, ans, arr, i + 1);

    ans.push\_back(arr[i]);

    combination(all, ans, arr, i + 1);

    ans.pop\_back();

}

vector<vector<int>> all\_combinations(vector<int> &arr)

{

    vector<vector<int>> all;

    vector<int> ans;

    combination(all, ans, arr, 0);

    return all;

}

#### Traverse all the permutations

#### Combination sum II (leetcode)

Given an array of positive integers, find the list of unique combinations whose sum is equal to target.

vvans

void combination(i, rem, vans, nums):

if rem == 0: vvans.push\_back(vans)

if (i == n or rem < 0) return

for j = i to n – 1

if (j < i and nums[j] == nums[j – 1]) continue

vans.push\_back(nums[i])

combination(i + 1, rem – nums[i], vans, nums)

vans.pop\_back()

vans

sort(nums.begin(), nums.end())

combination(0, target, vans, nums)

return vvans

#recursion #backtracking

Time complexity: ?

#### Generate Parenthesis

Given n. return all the valid parenthesis of size 2n.

Generate(open, closed, str):

if (open == n and closed == n)

ans.pb(str)

if (open < n) Generate(open + 1, closed, str + ‘(’)

if (closed < open) Generate(open, closed + 1, str + ‘)’)

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

# Binary Search

### Simple binary search

L = 0, R = n – 1

while (L ≤ R)

mid = L + (R – L)/2

if a[mid] == target

return mid

if a[mid] < target

L = mid + 1

else R = mid – 1

return -1

#### Is a Square?

Given n, check if it is a perfect square or not?

L = 1, R = n

while (L ≤ R)

mid = L + (R – L)/2

if (mid \* mid == N)

return true

if (mid \* mid < N)

L = mid + 1

else R = mid – 1

return false

#### closest to the right

Find the least (nearest) element in the sorted array ≥ X

L = 0, R = n – 1

ans = -1

while (L ≤ R)

mid = L + (R – L)/2

if (a[mid] ≥ X)

R = mid – 1

ans = mid

else L = mid + 1

return ans

#### closest to the left

Find the largest (and farthest) element in the sorted array ≤ X

L = 0, R = n – 1

while (L ≤ R)

mid = L + (R – L)/2

if (arr[mid] ≤ X)

R = mid – 1

ans = mid

else L = mid + 1

#### Rotated array

Somebody rotated (shifted) a sorted array. Find the smallest element.

L = 0, R = n – 1

ans = -1

while (L ≤ R)

mid = L + (R – L)/2

if (a[mid] > a[n – 1])

ans = mid

L = mid + 1

else R = mid – 1

return ans + 1

#### Array increases and decreases

The array first increases and then decreases. Find the maximum?

L = 0, R = n – 1

ans = -1

while (L ≤ R)

mid = L + (R – L)/2

if (a[mid] > a[mid + 1])

ans = mid

R = mid – 1

else L = mid + 1

return ans

#### Find peak element (leetcode, Errichto)

Given an array of integers. A peak element is the one larger than its neighbors. Find a peak element (the array may contain multiple peaks) in **logarithmic time** complexity. (arr[i] != arr[i + 1] ∀ i)

left = 0, right = n – 1

we need to be able to deduce the next sub interval (left or right) based on some values in the array.

if arr[mid] < arr[right] consider the right interval

if arr[mid] < arr[left] consider the left interval

if arr[mid] > arr[right], arr[left]

if arr[mid] < arr[mid – 1] consider the left interval

else consider the right interval

### Binary search optimization

#### Split array largest sum

Given an array of elements. You need to group them into subarrays such that the largest sum in any group is least possible.

**is\_possible**(nums, ans, m) // greedy

left = 1, right = accumulate(nums)

while (right – left > 1)

if is\_possible((left + right)/2) right = (left + right)/2

else left = (left + right)/2

if is\_possible(nums, right, m) return right

else return left

# Hashing

#### Number of pairs whose product is k (leetcode)

Given an array of integers. Find the number of pairs of integers whose product is k.

int ans = 0;

for i = 0 to n – 1

mp[nums[i]]++

for i = 0 to n – 1

if (k % nums[i] == 0)

if (k/nums[i] != nums[i])

ans += mp[k/nums[i]]

else ans += mp[nums[i]] – 1

return ans/2;

#### Number of Subarray with given sum

Given an array of size . Find the number of subarrays with sum

**Solution**:

for i = 1 to n

ans += mp[cum[i] – s];

mp[cum[i]]++;

return ans;

#### Good Subarrays

Given an array of size . A subarray is good if the **sum of elements** in the subarray is equal to the **length of the subarray**. Find the number of good subarrays in the given array?

**Solution**:

Decrease every element in the subarray by 1. Now the problem reduces to the # subarrays with sum **zero.**

#### 3Sum (leetcode)

Find the set of unique triplets whose sum is zero in the given array.

For i = 0 to n – 1

if arr[i] == arr[i – 1] continue

for j = i + 1 to n – 1

if arr[j] == arr[j – 1] continue

for k =

#### D Zigzags

You are given an array . Find the number of pairs such that

**Logic**: Fix and then search for

For j = 1 to n

For k = n to j + 1

Ans += count(aj after k) \* count(ak before j)

return ans

#dynamicprogramming, #hashing

#### Rotation matching

Given two permutations () which can be cyclically matched. Find the maximum number of matches in any rotation?

Suppose . Then can be matched by right shifting We can use map <shift, #matches>.

Calculating the differences between the two permutations is n.

# Stacking

#### Discrete centrifugal jumps (codeforces)

Given n heights. You can jump from to , if and are either strictly greater or strictly smaller than all heights in between (can also jump if ). Reach nth height in minimum jumps.

v1(n) – first element to left which is larger (in between are strictly less)

v2(n) – first element to left which is smaller (in between are strictly more)

v3(n) – first element to right which is larger (strictly less)

v4(n) – first element to right which is smaller (strictly more)

#### Array cancellation

Given an array whose sum of elements is zero. In one operation chose and , decrement be one and increment by one. If , this operation is free otherwise it costs 1. Find the minimum cost for making all the elements zero.

Negative of the smallest prefix sum would be the answer. Or the largest suffix sum.

int ans = INT\_MAX, sum = 0

for i = 0 to n – 1

sum += arr[i]

ans = min(ans, sum)

return -ans

#### Best time to buy and sell a stock

Given an array of numbers denoting the time series values of a stock. Find the maximum profit possible with at most one transaction?

min = INT\_MAX

int ans = 0

for i = 0 to n – 1

ans = max(ans, arr[i] – min)

min = min(min, arr[i])

return ans

#### Largest rectangle in histogram

Given an array of heights in the histogram. Find the maximum area of the rectangle possible.

A screenshot of a cell phone

Description automatically generated

For each element

Find first element to right less than the current value

Find first element to left less than the current value

For each element

Find maximum possible rect with current height.

<https://leetcode.com/problems/largest-rectangle-in-histogram/>

# Number theory

### Prime factorization

Given n, find all the prime factors of n.

For i = 2 to i \* i <= n

if (n % i == 0) **prime\_factors**.push\_back(i)

while (n % i == 0) n /= i

if (n != 1) prime\_factors.push\_back(n)

return prime\_factors

### nCr implementation

nCr % p = (fac[n]×inverse(fact[r])×inverse(fact[n-r]))% p

#### Inverse modulo p

Given two integers and . find the modular multiplicative inverse of .

a \* x ≡ 1 mod m

when m is prime

So, the inverse of a is can be computed in O(log m)

when a and m are coprime,

#### Coprime

Given n integers . Find if they are pair wise coprime (gcd of all pairs is 1)?

Set s – set of primes till now

For e in elements

vector prime\_factors = find\_prime\_factors(e)

for p in prime\_factors

if s contains p

return not coprime

return coprime

complexity =

<https://atcoder.jp/contests/abc177/tasks/abc177_e>

### Ugly number

A number is called ugly if it has only 2, 3 and 5 as the prime factors. Given n (< 1700) find the nth ugly number.

**Method** 1: using set or heap

1st number is 1. So, push 1 into the set.

For i = 1 to n - 1

x = set.begin();

s.insert(2 \* x);

s.insert(3 \* x);

s.insert(5 \* x);

return s.begin();

**Method 2:**

Maintain three pointers **p1 = 0, p2 = 0, p3 = 0**

int ans[n];

ans[0] = 1;

for i = 1 to n – 1

ans[i] = min(ans[p1] \* 2, ans[p2] \* 3, ans[p3] \* 5)

if (ans[i] = ans[p1] \* 2) p1++;

if (ans[i] = ans[p2] \* 3) p2++;

if (ans[i] = ans[p3] \* 5) p3++;

return ans[n – 1];

<https://leetcode.com/problems/ugly-number-ii/>

# Sorting and searching

#### Search a 2d matrix (leetcode)

Given a matrix, all row and all columns are sorted. Find whether a number is present in the matrix

#### Median of two sorted arrays

Given two sorted arrays of length and respectively. Find the median of the two arrays combined in O(m + n).

Given two arrays, arr1, arr2

while (arr1.size() ≥ 2 and arr2.size() ≥ 2)

m1 = median(arr1)

m2 = median(arr2)

if m1 > m2: arr1 = arr1[0: n/2], arr2 = arr2[n/2: 0]

else arr1 = arr1[n/2: 0], arr2 = arr2[0: n/2]

return median (easy)

# Bit Masking

bitset<1000> a; // creates a binary number a of length 1000

a.count() // returns the number of ones

a & b // bitwise and

a | b // bitwise or

a[i] // random access

#### knapsack

Given N (≤ 1000) each with some weight w[i] is there a subset with total weight exactly W(≤ 10^6) ?

bitset<MAX\_W> can;

for (int i = 0; i < n; i++)

can |= (can ≪ w[i]);

return can[W];

#### Triangles in a graph

Given a graph with N ≤ 2000 vertices, count number of triangles (a, b, c)

Create an adjacency list as

vector<bitset<MAX\_N≫ graph;

for i = 0 to N

for j = i + 1 to N

if (edge(i, j))

ans += (graph[i] & graph[j]).count()

ans /= 3;

return ans;

#### odd topic (codechef)

Given two arrays a, b of lengths n and m, respectively. There are q queries. Each query contains Consider l1 to r1 elements in a and l2 to r2 elements in b. Find the number of elements that occur odd number of times in the union.

vector<bitset<4001≫ cum\_a(n), cum\_b(m);

for i = 1 to n

cum\_a[i][a[i]] = 1

cum\_a[i] ^= cum\_a[i – 1]

for i = 1 to m

cum\_b[i][b[i]] = 1

cum\_b[i] ^= cum\_b[i – 1]

for i = 1 to q

cin ≫ l1 ≫ r1 ≫ l2 ≫ r2; l1--, l2--;

cout ≪ (a[l1] ^ a[r1] ^ b[l2] ^ b[r2]).count() ≪ endl;

# Important Algorithms

### Union find algorithm

class unf

{

    vector<int> par, sz;

    int components;

    unf(int n) : par(n), sz(n, 1)

    {

        components = n;

        iota(par.begin(), par.end(), 0);

    }

    int find(int x)

    {

        return par[x] == x ? x : (par[x] = find(par[x]));

    }

    bool unite(int x, int y)

    {

        int parx = find(x), pary = find(y);

        if (parx == pary) return false;

        if (sz[parx] > sz[pary])

        {

            par[pary] = parx;

            sz[parx] += sz[pary];

        }

        else

        {

            par[parx] = pary;

            sz[pary] += sz[parx];

        }

        return true;

    }

};

### Kadane’s algorithm

Given an array containing non-negative integers. Find the maximum sum in any subarray.

#### maximum product subarray

given an array containing non-negative integers. Find the maximum product in any subarray.

### KMPs Algorithm

Given two strings pat, str, find all the locations where pat is matched with str.

First computing LPS[pat.len] array.

LPS[i] = length of longest **proper prefix** of pat[0…i]

Which is also a suffix of pat[0…i]

i = 0, j = 0

while (i < n)

if (str[i] == pat[j])

i++, j++

if j == m: ans.push\_back(i – j), i++, j = 0

if (str[i] != pat[j])

if (j == 0) i++

else j = lps[j – 1]

#### D Maximum sum on even positions

Given an array of size n. You can reverse at most one subarray. Find the maximum sum in even positions you can achieve.

<https://codeforces.com/problemset/problem/1373/D>

### Dijkstra

#### element in a subarray

Given an array of size whose values are also between . Suppose you need to find the smallest element in an arbitrary subarray multiple times.

**Technique**:

Maintain an array . Which is defined as shown below

For a subarray

is the smallest element if

# Geometry

#### Check if two line-segments intersect

### Convex hull

Given a set of points in the plane. The convex hull of the set is the smallest polygon that contains all the points.

### A - Takahashikun, The Strider

Takahashi is standing on a two-dimensional plane, facing north. Find the minimum positive integer  such that Takahashi will be at the starting position again after he does the following action K times: *Go one meter in the direction he is facing. Then, turn  degrees counterclockwise.*

Ans

# Divide and conquer

### Merge sort

### In place merge sort

# Debugging

1. Using --set.rbegin() <https://codeforces.com/contest/1393/submission/89992264>
2. Using #define 1e9 + 7 giving wrong results <https://codeforces.com/problemset/problem/431/C>