

## 801. Minimum Swaps To Make Sequences Increasing

Hard

2276128Add to ListShare

You are given two integer arrays of the same length `nums1` and `nums2`. In one operation, you are allowed to swap `nums1[i]` with `nums2[i]`.

For example, if `nums1 = [1, 2, 3, 8]`, and `nums2 = [5, 6, 7, 4]`, you can swap the element at `i = 3` to obtain `nums1 = [1, 2, 3, 4]` and `nums2 = [5, 6, 7, 8]`.

Return the minimum number of needed operations to make `nums1` and `nums2` **strictly increasing**.

The test cases are generated so that the given input always makes it possible.

An array `arr` is **strictly increasing** if and only if `arr[0] < arr[1] < arr[2] < ... < arr[arr.length - 1]`.

**Example 1:**

**Input:** `nums1 = [1, 3, 5, 4]`, `nums2 = [1, 2, 3, 7]`

**Output:** 1

**Explanation:**

Swap `nums1[3]` and `nums2[3]`. Then the sequences are:

`nums1 = [1, 3, 5, 7]` and `nums2 = [1, 2, 3, 4]`

which are both strictly increasing.

**Example 2:**

**Input:** `nums1 = [0, 3, 5, 8, 9]`, `nums2 = [2, 1, 4, 6, 9]`

**Output:** 1

**Constraints:**

`2 <= nums1.length <= 105`

`nums2.length == nums1.length`

`0 <= nums1[i], nums2[i] <= 2 * 105`

## 802. Find Eventual Safe States

### Medium

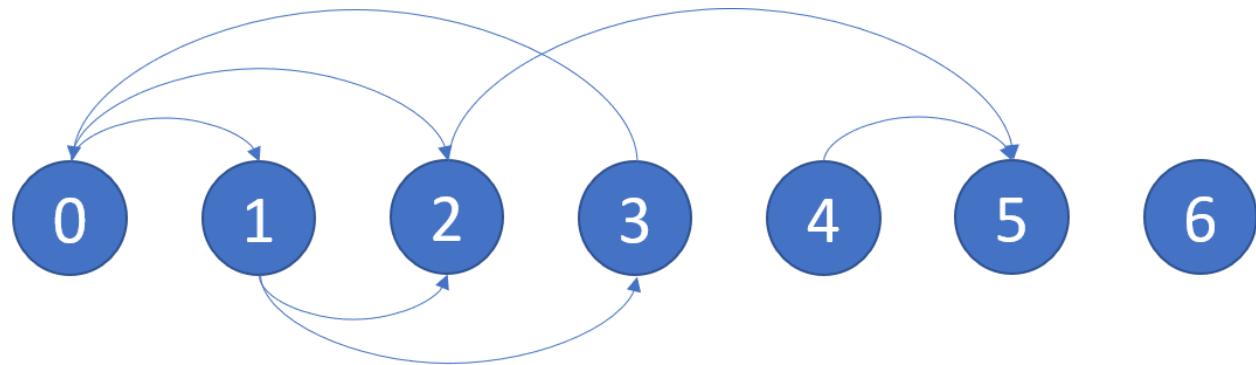
2802339Add to ListShare

There is a directed graph of  $n$  nodes with each node labeled from  $0$  to  $n - 1$ . The graph is represented by a **0-indexed** 2D integer array `graph` where `graph[i]` is an integer array of nodes adjacent to node  $i$ , meaning there is an edge from node  $i$  to each node in `graph[i]`.

A node is a **terminal node** if there are no outgoing edges. A node is a **safe node** if every possible path starting from that node leads to a **terminal node** (or another safe node).

Return *an array containing all the safe nodes of the graph*. The answer should be sorted in **ascending** order.

#### Example 1:



**Input:** `graph = [[1,2],[2,3],[5],[0],[5],[],[]]`

**Output:** `[2,4,5,6]`

**Explanation:** The given graph is shown above.

Nodes 5 and 6 are terminal nodes as there are no outgoing edges from either of them.

Every path starting at nodes 2, 4, 5, and 6 all lead to either node 5 or 6.

#### Example 2:

**Input:** `graph = [[1,2,3,4],[1,2],[3,4],[0,4],[]]`

**Output:** `[4]`

**Explanation:**

Only node 4 is a terminal node, and every path starting at node 4 leads to node 4.

#### Constraints:

```

n == graph.length

1 <= n <= 104

0 <= graph[i].length <= n

0 <= graph[i][j] <= n - 1

```

graph[i] is sorted in a strictly increasing order.

The graph may contain self-loops.

The number of edges in the graph will be in the range [1, 4 \* 10<sup>4</sup>].

## 803. Bricks Falling When Hit

Hard

908175Add to ListShare

You are given an  $m \times n$  binary grid, where each 1 represents a brick and 0 represents an empty space. A brick is **stable** if:

It is directly connected to the top of the grid, or

At least one other brick in its four adjacent cells is **stable**.

You are also given an array `hits`, which is a sequence of erasures we want to apply. Each time we want to erase the brick at the location `hits[i] = (rowi, coli)`. The brick on that location (if it exists) will disappear. Some other bricks may no longer be stable because of that erasure and will **fall**. Once a brick falls, it is **immediately** erased from the `grid` (i.e., it does not land on other stable bricks).

Return an array `result`, where each `result[i]` is the number of bricks that will **fall** after the  $i^{\text{th}}$  erasure is applied.

**Note** that an erasure may refer to a location with no brick, and if it does, no bricks drop.

### Example 1:

**Input:** `grid = [[1,0,0,0],[1,1,1,0]], hits = [[1,0]]`

**Output:** [2]

**Explanation:** Starting with the grid:

```

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

```

We erase the underlined brick at  $(1,0)$ , resulting in the grid:

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

The two underlined bricks are no longer stable as they are no longer connected to the top nor adjacent to another stable brick, so they will fall. The resulting grid is:

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

Hence the result is  $[2]$ .

**Example 2:**

**Input:** grid =  $[[1,0,0,0],[1,1,0,0]]$ , hits =  $[[1,1],[1,0]]$

**Output:**  $[0,0]$

**Explanation:** Starting with the grid:

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

We erase the underlined brick at  $(1,1)$ , resulting in the grid:

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

All remaining bricks are still stable, so no bricks fall. The grid remains the same:

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

Next, we erase the underlined brick at  $(1,0)$ , resulting in the grid:

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

Once again, all remaining bricks are still stable, so no bricks fall.

Hence the result is  $[0,0]$ .

**Constraints:**

```

m == grid.length

n == grid[i].length

1 <= m, n <= 200

grid[i][j] is 0 or 1.

1 <= hits.length <= 4 * 104

hits[i].length == 2

0 <= xi <= m - 1

0 <= yi <= n - 1

All (xi, yi) are unique.

```

## 804. Unique Morse Code Words

Easy

20731381 Add to List Share

International Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes, as follows:

```

'a' maps to ".-",
'b' maps to "-...",
'c' maps to "-.-.", and so on.

```

For convenience, the full table for the 26 letters of the English alphabet is given below:

```

[".-","-...","-.-.","-..",".","...-","-.-","....","..-",".-..","-..-","-.-..","-.-..-","-.-.-","-.-.-..","-.-.-.-","-.-.-.-..","-.-.-.-.-..","-.-.-.-.-..-","-.-.-.-.-.-..-..","-.-.-.-.-.-..-..-.."]

```

Given an array of strings `words` where each word can be written as a concatenation of the Morse code of each letter.

For example, "cab" can be written as "-.-.-..", which is the concatenation of "-.-", ".-", and "-..". We will call such a concatenation the **transformation** of a word.

Return *the number of different **transformations** among all words we have.*

**Example 1:**

**Input:** words = ["gin", "zen", "gig", "msg"]

**Output:** 2

**Explanation:** The transformation of each word is:

"gin" -> "--...-."

"zen" -> "--...-."

"gig" -> "--...--."

"msg" -> "--...--."

There are 2 different transformations: "--...-." and "--...--.".

**Example 2:**

**Input:** words = ["a"]

**Output:** 1

**Constraints:**

`1 <= words.length <= 100`

`1 <= words[i].length <= 12`

`words[i]` consists of lowercase English letters.

## 805. Split Array With Same Average

Hard

964122Add to ListShare

You are given an integer array `nums`.

You should move each element of `nums` into one of the two arrays `A` and `B` such that `A` and `B` are non-empty, and `average(A) == average(B)`.

Return `true` if it is possible to achieve that and `false` otherwise.

**Note** that for an array `arr`, `average(arr)` is the sum of all the elements of `arr` over the length of `arr`.

**Example 1:**

**Input:** `nums = [1,2,3,4,5,6,7,8]`

Output: true

**Explanation:** We can split the array into  $[1,4,5,8]$  and  $[2,3,6,7]$ , and both of them have an average of 4.5.

## Example 2:

**Input:** nums = [3,1]

**Output:** false

## Constraints:

1 <= nums.length <= 30

$0 \leq \text{nums}[i] \leq 10^4$

## 806. Number of Lines To Write String

Easy

4281170Add to ListShare

You are given a string `s` of lowercase English letters and an array `widths` denoting **how many pixels wide** each lowercase English letter is. Specifically, `widths[0]` is the width of '`a`', `widths[1]` is the width of '`b`', and so on.

You are trying to write `s` across several lines, where **each line is no longer than 100 pixels**. Starting at the beginning of `s`, write as many letters on the first line such that the total width does not exceed 100 pixels. Then, from where you stopped in `s`, continue writing as many letters as you can on the second line. Continue this process until you have written all of `s`.

Return an array result of length 2 where:

`result[0]` is the total number of lines.

`result[1]` is the width of the last line in pixels.

### Example 1:

**Output:** [3,60]

**Explanation:** You can write  $s$  as follows:

```
abcdefghijkl // 100 pixels wide  
klmnopqrst // 100 pixels wide  
uvwxyz      // 60 pixels wide
```

There are a total of 3 lines, and the last line is 60 pixels wide.

## Example 2:

**Output:** [2,4]

**Explanation:** You can write  $s$  as follows:

bbbcccddaa // 98 pixels wide

a // 4 pixels wide

There are a total of 2 lines, and the last line is 4 pixels wide.

## Constraints:

```
widths.length == 26
```

```
2 <= widths[i] <= 10
```

1 <= s.length <= 1000

`s` contains only lowercase English letters.

## 807. Max Increase to Keep City Skyline

## Medium

1959458Add to ListShare

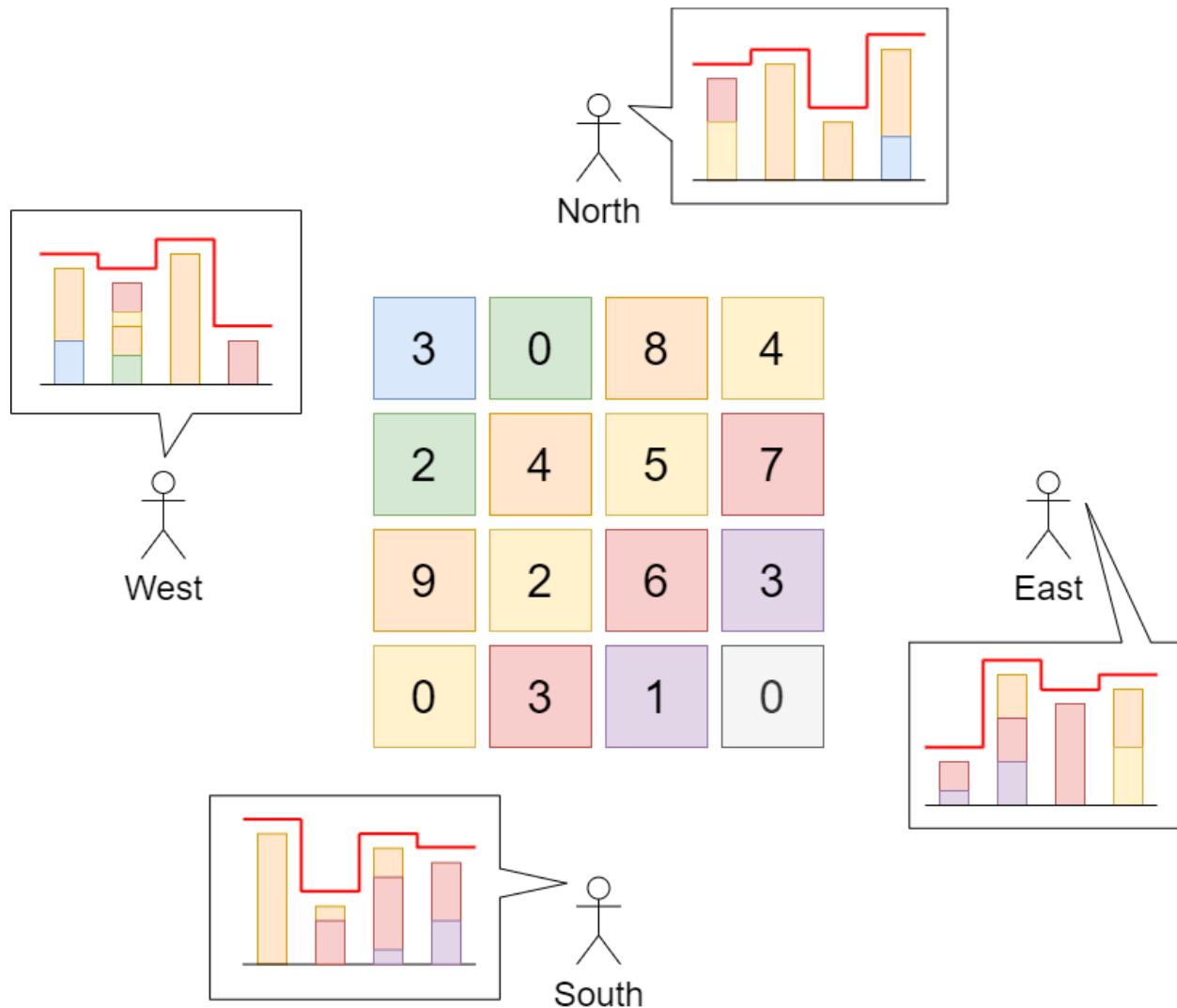
There is a city composed of  $n \times n$  blocks, where each block contains a single building shaped like a vertical square prism. You are given a **0-indexed**  $n \times n$  integer matrix `grid` where `grid[r][c]` represents the **height** of the building located in the block at row `r` and column `c`.

A city's **skyline** is the the outer contour formed by all the building when viewing the side of the city from a distance. The **skyline** from each cardinal direction north, east, south, and west may be different.

We are allowed to increase the height of **any number of buildings by any amount** (the amount can be different per building). The height of a 0-height building can also be increased. However, increasing the height of a building should **not** affect the city's **skyline** from any cardinal direction.

Return the **maximum total sum** that the height of the buildings can be increased by **without** changing the city's **skyline** from any cardinal direction.

**Example 1:**



**Input:** grid = [[3,0,8,4],[2,4,5,7],[9,2,6,3],[0,3,1,0]]

**Output:** 35

**Explanation:** The building heights are shown in the center of the above image.

The skylines when viewed from each cardinal direction are drawn in red.

The grid after increasing the height of buildings without affecting skylines is:

```
gridNew = [ [8, 4, 8, 7],
            [7, 4, 7, 7],
            [9, 4, 8, 7],
            [3, 3, 3, 3] ]
```

**Example 2:**

**Input:** grid = [[0,0,0],[0,0,0],[0,0,0]]

**Output:** 0

**Explanation:** Increasing the height of any building will result in the skyline changing.

**Constraints:**

```
n == grid.length
n == grid[r].length
2 <= n <= 50
0 <= grid[r][c] <= 100
```

## 808. Soup Servings

Medium

287807Add to ListShare

There are two types of soup: **type A** and **type B**. Initially, we have  $n$  ml of each type of soup. There are four kinds of operations:

Serve 100 ml of **soup A** and 0 ml of **soup B**,

Serve 75 ml of **soup A** and 25 ml of **soup B**,

Serve 50 ml of **soup A** and 50 ml of **soup B**, and

Serve 25 ml of **soup A** and 75 ml of **soup B**.

When we serve some soup, we give it to someone, and we no longer have it. Each turn, we will choose from the four operations with an equal probability 0.25. If the remaining volume of soup is not enough to complete the operation, we will serve as much as possible. We stop once we no longer have some quantity of both types of soup.

**Note** that we do not have an operation where all 100 ml's of **soup B** are used first.

Return the probability that **soup A** will be empty first, plus half the probability that **A** and **B** become empty at the same time. Answers within  $10^{-5}$  of the actual answer will be accepted.

### Example 1:

**Input:** n = 50

**Output:** 0.62500

**Explanation:** If we choose the first two operations, A will become empty first.

For the third operation, A and B will become empty at the same time.

For the fourth operation, B will become empty first.

So the total probability of A becoming empty first plus half the probability that A and B become empty at the same time, is  $0.25 * (1 + 1 + 0.5 + 0) = 0.625$ .

### Example 2:

**Input:** n = 100

**Output:** 0.71875

### Constraints:

$0 \leq n \leq 10^9$

## 809. Expressive Words

Medium

7461730Add to ListShare

Sometimes people repeat letters to represent extra feeling. For example:

"hello" -> "heeeellooo"

"hi" -> "hiiii"

In these strings like "heeeellooo", we have groups of adjacent letters that are all the same: "h", "eee", "ll", "ooo".

You are given a string `s` and an array of query strings `words`. A query word is **stretchy** if it can be made to be equal to `s` by any number of applications of the following extension operation: choose a

group consisting of characters `c`, and add some number of characters `c` to the group so that the size of the group is **three or more**.

For example, starting with "hello", we could do an extension on the group "o" to get "hellooo", but we cannot get "helloo" since the group "oo" has a size less than three. Also, we could do another extension like "ll"  $\rightarrow$  "lllll" to get "helllllooo". If `s` = "helllllooo", then the query word "hello" would be **stretchy** because of these two extension operations: `query` = "hello"  $\rightarrow$  "hellooo"  $\rightarrow$  "helllllooo" = `s`.

Return *the number of query strings that are stretchy*.

### Example 1:

**Input:** `s` = "heeellooo", `words` = ["hello", "hi", "helo"]

**Output:** 1

#### Explanation:

We can extend "e" and "o" in the word "hello" to get "heeellooo".

We can't extend "helo" to get "heeellooo" because the group "ll" is not size 3 or more.

### Example 2:

**Input:** `s` = "zzzzzyyyyy", `words` = ["zzyy", "zy", "zyy"]

**Output:** 3

#### Constraints:

`1 <= s.length, words.length <= 100`

`1 <= words[i].length <= 100`

`s` and `words[i]` consist of lowercase letters.

## 810. Chalkboard XOR Game

### Hard

155250Add to ListShare

You are given an array of integers `nums` represents the numbers written on a chalkboard.

Alice and Bob take turns erasing exactly one number from the chalkboard, with Alice starting first. If erasing a number causes the bitwise XOR of all the elements of the chalkboard to become `0`, then that player loses. The bitwise XOR of one element is that element itself, and the bitwise XOR of no elements is `0`.

Also, if any player starts their turn with the bitwise XOR of all the elements of the chalkboard equal to `0`, then that player wins.

Return `true` if and only if Alice wins the game, assuming both players play optimally.

### Example 1:

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

**Output:** `false`

**Explanation:**

Alice has two choices: erase 1 or erase 2.

If she erases 1, the `nums` array becomes `[1, 2]`. The bitwise XOR of all the elements of the chalkboard is  $1 \text{ XOR } 2 = 3$ . Now Bob can remove any element he wants, because Alice will be the one to erase the last element and she will lose.

If Alice erases 2 first, now `nums` become `[1, 1]`. The bitwise XOR of all the elements of the chalkboard is  $1 \text{ XOR } 1 = 0$ . Alice will lose.

### Example 2:

**Input:** `nums = [0,1]`

**Output:** `true`

### Example 3:

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

**Output:** `true`

### Constraints:

`1 <= nums.length <= 1000`

`0 <= nums[i] < 216`

## 811. Subdomain Visit Count

## Medium

12521193Add to ListShare

A website domain "discuss.leetcode.com" consists of various subdomains. At the top level, we have "com", at the next level, we have "leetcode.com" and at the lowest level, "discuss.leetcode.com". When we visit a domain like "discuss.leetcode.com", we will also visit the parent domains "leetcode.com" and "com" implicitly.

A **count-paired domain** is a domain that has one of the two formats "rep d1.d2.d3" or "rep d1.d2" where `rep` is the number of visits to the domain and `d1.d2.d3` is the domain itself.

For example, "9001 discuss.leetcode.com" is a **count-paired domain** that indicates that `discuss.leetcode.com` was visited 9001 times.

Given an array of **count-paired domains** `cpdomains`, return *an array of the count-paired domains of each subdomain in the input*. You may return the answer in **any order**.

### Example 1:

**Input:** cpdomains = ["9001 discuss.leetcode.com"]

**Output:** ["9001 leetcode.com", "9001 discuss.leetcode.com", "9001 com"]

**Explanation:** We only have one website domain: "discuss.leetcode.com".

As discussed above, the subdomain "leetcode.com" and "com" will also be visited. So they will all be visited 9001 times.

### Example 2:

**Input:** cpdomains = ["900 google.mail.com", "50 yahoo.com", "1 intel.mail.com", "5 wiki.org"]

**Output:** ["901 mail.com", "50 yahoo.com", "900 google.mail.com", "5 wiki.org", "5 org", "1 intel.mail.com", "951 com"]

**Explanation:** We will visit "google.mail.com" 900 times, "yahoo.com" 50 times, "intel.mail.com" once and "wiki.org" 5 times.

For the subdomains, we will visit "mail.com"  $900 + 1 = 901$  times, "com"  $900 + 50 + 1 = 951$  times, and "org" 5 times.

### Constraints:

`1 <= cpdomain.length <= 100`

`1 <= cpdomain[i].length <= 100`

`cpdomain[i]` follows either the "`repi d1i.d2i.d3i`" format or the "`repi d1i.d2i`" format.

`repi` is an integer in the range  $[1, 10^4]$ .

`d1i, d2i, and d3i` consist of lowercase English letters.

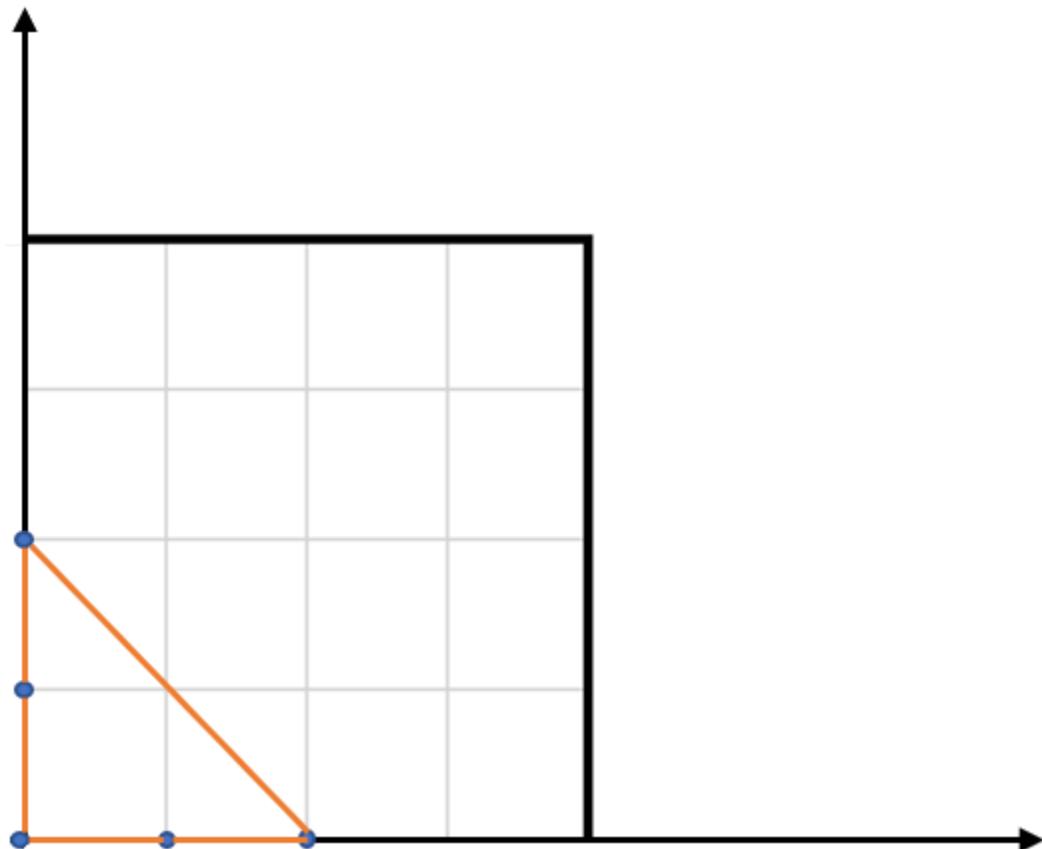
## 812. Largest Triangle Area

Easy

3811398Add to ListShare

Given an array of points on the **X-Y** plane `points` where `points[i] = [xi, yi]`, return *the area of the largest triangle that can be formed by any three different points*. Answers within  $10^{-5}$  of the actual answer will be accepted.

**Example 1:**



**Input:** `points = [[0,0],[0,1],[1,0],[0,2],[2,0]]`

**Output:** `2.00000`

**Explanation:** The five points are shown in the above figure. The red triangle is the largest.

**Example 2:**

**Input:** points = [[1,0],[0,0],[0,1]]

**Output:** 0.50000

**Constraints:**

3  $\leq$  points.length  $\leq$  50

-50  $\leq$   $x_i, y_i \leq$  50

All the given points are **unique**.

## 813. Largest Sum of Averages

Medium

169688Add to ListShare

You are given an integer array `nums` and an integer `k`. You can partition the array into **at most** `k` non-empty adjacent subarrays. The **score** of a partition is the sum of the averages of each subarray.

Note that the partition must use every integer in `nums`, and that the score is not necessarily an integer.

Return *the maximum score you can achieve of all the possible partitions*. Answers within  $10^{-6}$  of the actual answer will be accepted.

**Example 1:**

**Input:** nums = [9,1,2,3,9], k = 3

**Output:** 20.00000

**Explanation:**

The best choice is to partition `nums` into [9], [1, 2, 3], [9]. The answer is  $9 + (1 + 2 + 3) / 3 + 9 = 20$ .

We could have also partitioned `nums` into [9, 1], [2], [3, 9], for example.

That partition would lead to a score of  $5 + 2 + 6 = 13$ , which is worse.

**Example 2:**

**Input:** nums = [1,2,3,4,5,6,7], k = 4

**Output:** 20.50000

**Constraints:**

1 <= nums.length <= 100

1 <= nums[i] <= 10<sup>4</sup>

1 <= k <= nums.length

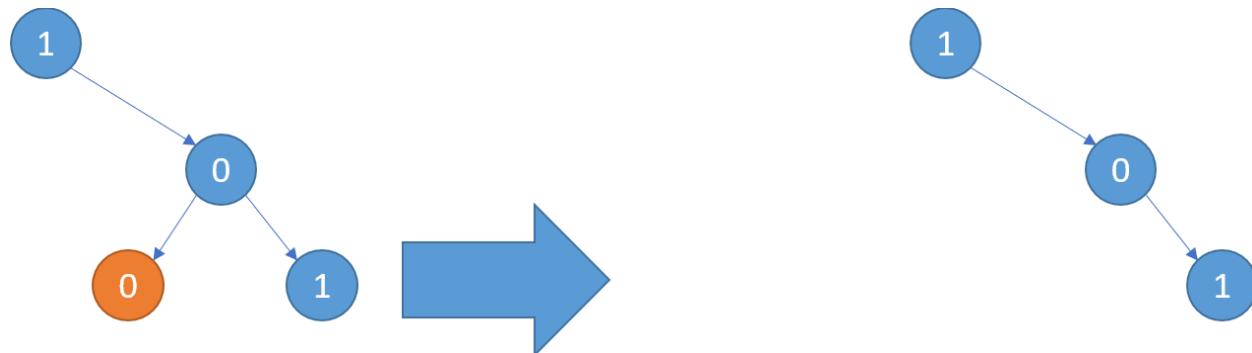
**814. Binary Tree Pruning**

Medium

3945103Add to ListShare

Given the `root` of a binary tree, return *the same tree where every subtree (of the given tree) not containing a 1 has been removed*.

A subtree of a node `node` is `node` plus every node that is a descendant of `node`.

**Example 1:**

**Input:** root = [1,null,0,0,1]

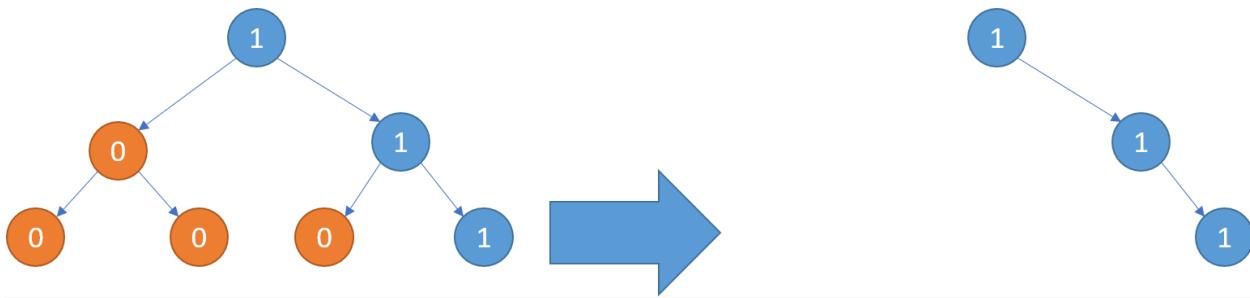
**Output:** [1,null,0,null,1]

**Explanation:**

Only the red nodes satisfy the property "every subtree not containing a 1".

The diagram on the right represents the answer.

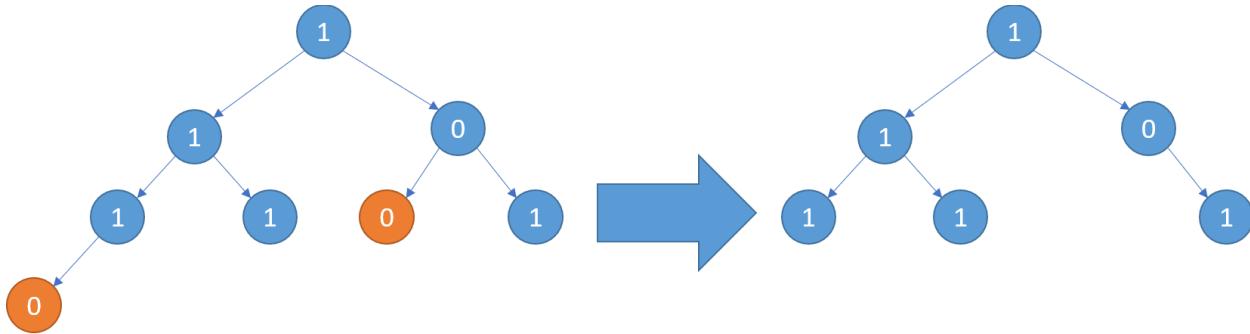
**Example 2:**



**Input:** root = [1,0,1,0,0,0,1]

**Output:** [1,null,1,null,1]

**Example 3:**



**Input:** root = [1,1,0,1,1,0,1,0]

**Output:** [1,1,0,1,1,null,1]

**Constraints:**

The number of nodes in the tree is in the range [1, 200].

Node.val is either 0 or 1.

## 815. Bus Routes

**Hard**

258365Add to ListShare

You are given an array routes representing bus routes where routes[i] is a bus route that the *i<sup>th</sup>* bus repeats forever.

For example, if routes[0] = [1, 5, 7], this means that the 0<sup>th</sup> bus travels in the sequence 1  
-> 5 -> 7 -> 1 -> 5 -> 7 -> 1 -> ... forever.

You will start at the bus stop source (You are not on any bus initially), and you want to go to the bus stop target. You can travel between bus stops by buses only.

Return the least number of buses you must take to travel from `source` to `target`. Return `-1` if it is not possible.

**Example 1:**

**Input:** `routes = [[1,2,7],[3,6,7]]`, `source = 1`, `target = 6`

**Output:** `2`

**Explanation:** The best strategy is take the first bus to the bus stop 7, then take the second bus to the bus stop 6.

**Example 2:**

**Input:** `routes = [[7,12],[4,5,15],[6],[15,19],[9,12,13]]`, `source = 15`, `target = 12`

**Output:** `-1`

**Constraints:**

`1 <= routes.length <= 500.`

`1 <= routes[i].length <= 105`

All the values of `routes[i]` are **unique**.

`sum(routes[i].length) <= 105`

`0 <= routes[i][j] < 106`

`0 <= source, target < 106`

## 816. Ambiguous Coordinates

Medium

272606Add to ListShare

We had some 2-dimensional coordinates, like `"(1, 3)"` or `"(2, 0.5)"`. Then, we removed all commas, decimal points, and spaces and ended up with the string `s`.

For example, `"(1, 3)"` becomes `s = "(13)"` and `"(2, 0.5)"` becomes `s = "(205)"`.

Return a list of strings representing all possibilities for what our original coordinates could have been.

Our original representation never had extraneous zeroes, so we never started with numbers like `"00"`, `"0.0"`, `"0.00"`, `"1.0"`, `"001"`, `"00.01"`, or any other number that can be represented

with fewer digits. Also, a decimal point within a number never occurs without at least one digit occurring before it, so we never started with numbers like ".1".

The final answer list can be returned in any order. All coordinates in the final answer have exactly one space between them (occurring after the comma.)

**Example 1:**

**Input:** s = "(123)"

**Output:** ["(1, 2.3)", "(1, 23)", "(1.2, 3)", "(12, 3)"]

**Example 2:**

**Input:** s = "(0123)"

**Output:** ["(0, 1.23)", "(0, 12.3)", "(0, 123)", "(0.1, 2.3)", "(0.1, 23)", "(0.12, 3)"]

**Explanation:** 0.0, 00, 0001 or 00.01 are not allowed.

**Example 3:**

**Input:** s = "(00011)"

**Output:** ["(0, 0.011)", "(0.001, 1)"]

**Constraints:**

```
4 <= s.length <= 12
s[0] == '(' and s[s.length - 1] == ')'.
```

The rest of s are digits.

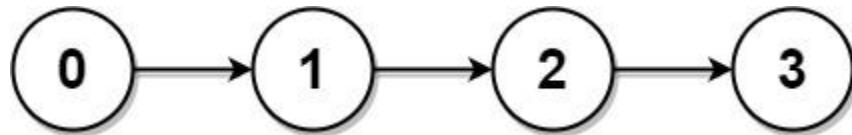
## 817. Linked List Components

Medium

7841901Add to ListShare

You are given the `head` of a linked list containing unique integer values and an integer array `nums` that is a subset of the linked list values.

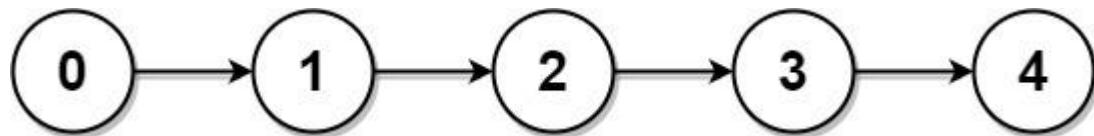
Return the number of connected components in `nums` where two values are connected if they appear **consecutively** in the linked list.

**Example 1:**

**Input:** head = [0,1,2,3], nums = [0,1,3]

**Output:** 2

**Explanation:** 0 and 1 are connected, so [0, 1] and [3] are the two connected components.

**Example 2:**

**Input:** head = [0,1,2,3,4], nums = [0,3,1,4]

**Output:** 2

**Explanation:** 0 and 1 are connected, 3 and 4 are connected, so [0, 1] and [3, 4] are the two connected components.

**Constraints:**

The number of nodes in the linked list is `n`.

`1 <= n <= 104`

`0 <= Node.val < n`

All the values `Node.val` are **unique**.

`1 <= nums.length <= n`

`0 <= nums[i] < n`

All the values of `nums` are **unique**.

**818. Race Car**

Hard

1413141Add to ListShare

Your car starts at position `0` and speed `+1` on an infinite number line. Your car can go into negative positions. Your car drives automatically according to a sequence of instructions `'A'` (accelerate) and `'R'` (reverse):

When you get an instruction `'A'`, your car does the following:

```
position += speed
speed *= 2
```

When you get an instruction `'R'`, your car does the following:

If your speed is positive then `speed = -1`

otherwise `speed = 1`

Your position stays the same.

For example, after commands "AAR", your car goes to positions `0 --> 1 --> 3 --> 3`, and your speed goes to `1 --> 2 --> 4 --> -1`.

Given a target position `target`, return *the length of the shortest sequence of instructions to get there*.

### Example 1:

**Input:** `target = 3`

**Output:** `2`

**Explanation:**

The shortest instruction sequence is "AA".

Your position goes from `0 --> 1 --> 3`.

### Example 2:

**Input:** `target = 6`

**Output:** `5`

**Explanation:**

The shortest instruction sequence is "AAARA".

Your position goes from `0 --> 1 --> 3 --> 7 --> 7 --> 6`.

**Constraints:**

```
1 <= target <= 104
```

**819. Most Common Word****Easy**

13862735Add to ListShare

Given a string `paragraph` and a string array of the banned words `banned`, return *the most frequent word that is not banned*. It is **guaranteed** there is **at least one word** that is not banned, and that the answer is **unique**.

The words in `paragraph` are **case-insensitive** and the answer should be returned in **lowercase**.

**Example 1:**

**Input:** `paragraph = "Bob hit a ball, the hit BALL flew far after it was hit.", banned = ["hit"]`

**Output:** "ball"

**Explanation:**

"hit" occurs 3 times, but it is a banned word.

"ball" occurs twice (and no other word does), so it is the most frequent non-banned word in the paragraph.

Note that words in the paragraph are not case sensitive, that punctuation is ignored (even if adjacent to words, such as "ball,"), and that "hit" isn't the answer even though it occurs more because it is banned.

**Example 2:**

**Input:** `paragraph = "a.", banned = []`

**Output:** "a"

**Constraints:**

```
1 <= paragraph.length <= 1000
```

paragraph consists of English letters, space ' ', or one of the symbols: " !? ', ; .".

```
0 <= banned.length <= 100
```

```
1 <= banned[i].length <= 10
banned[i] consists of only lowercase English letters.
```

## 820. Short Encoding of Words

Medium

1612617Add to ListShare

A **valid encoding** of an array of `words` is any reference string `s` and array of indices `indices` such that:

```
words.length == indices.length
```

The reference string `s` ends with the '#' character.

For each index `indices[i]`, the **substring** of `s` starting from `indices[i]` and up to (but not including) the next '#' character is equal to `words[i]`.

Given an array of `words`, return the **length of the shortest reference string** `s` possible of any **valid encoding** of `words`.

### Example 1:

**Input:** `words = ["time", "me", "bell"]`

**Output:** 10

**Explanation:** A valid encoding would be `s = "time#bell#"` and `indices = [0, 2, 5]`.

`words[0] = "time"`, the substring of `s` starting from `indices[0] = 0` to the next '#' is underlined in `"timeme#bell#"`

`words[1] = "me"`, the substring of `s` starting from `indices[1] = 2` to the next '#' is underlined in `"timeme#bell#"`

`words[2] = "bell"`, the substring of `s` starting from `indices[2] = 5` to the next '#' is underlined in `"time#bell#"`

### Example 2:

**Input:** `words = ["t"]`

**Output:** 2

**Explanation:** A valid encoding would be `s = "t#"` and `indices = [0]`.

**Constraints:**

```
1 <= words.length <= 2000
1 <= words[i].length <= 7
words[i] consists of only lowercase letters.
```

**821. Shortest Distance to a Character****Easy**

2455129Add to ListShare

Given a string `s` and a character `c` that occurs in `s`, return an array of integers `answer` where `answer.length == s.length` and `answer[i]` is the **distance** from index `i` to the **closest** occurrence of character `c` in `s`.

The **distance** between two indices `i` and `j` is `abs(i - j)`, where `abs` is the absolute value function.

**Example 1:**

**Input:** `s = "loveleetcode", c = "e"`

**Output:** `[3,2,1,0,1,0,0,1,2,2,1,0]`

**Explanation:** The character 'e' appears at indices 3, 5, 6, and 11 (0-indexed).

The closest occurrence of 'e' for index 0 is at index 3, so the distance is `abs(0 - 3) = 3`.

The closest occurrence of 'e' for index 1 is at index 3, so the distance is `abs(1 - 3) = 2`.

For index 4, there is a tie between the 'e' at index 3 and the 'e' at index 5, but the distance is still the same: `abs(4 - 3) == abs(4 - 5) = 1`.

The closest occurrence of 'e' for index 8 is at index 6, so the distance is `abs(8 - 6) = 2`.

**Example 2:**

**Input:** `s = "aaab", c = "b"`

**Output:** `[3,2,1,0]`

**Constraints:**

```
1 <= s.length <= 104
```

`s[i]` and `c` are lowercase English letters.

It is guaranteed that `c` occurs at least once in `s`.

## 822. Card Flipping Game

Medium

126657Add to ListShare

You are given two **0-indexed** integer arrays `fronts` and `backs` of length `n`, where the `ith` card has the positive integer `fronts[i]` printed on the front and `backs[i]` printed on the back. Initially, each card is placed on a table such that the front number is facing up and the other is facing down. You may flip over any number of cards (possibly zero).

After flipping the cards, an integer is considered **good** if it is facing down on some card and **not** facing up on any card.

Return *the minimum possible good integer after flipping the cards*. If there are no good integers, return 0.

### Example 1:

**Input:** `fronts = [1,2,4,4,7], backs = [1,3,4,1,3]`

**Output:** 2

**Explanation:**

If we flip the second card, the face up numbers are `[1,3,4,4,7]` and the face down are `[1,2,4,1,3]`.

2 is the minimum good integer as it appears facing down but not facing up.

It can be shown that 2 is the minimum possible good integer obtainable after flipping some cards.

### Example 2:

**Input:** `fronts = [1], backs = [1]`

**Output:** 0

**Explanation:**

There are no good integers no matter how we flip the cards, so we return 0.

### Constraints:

```

n == fronts.length == backs.length

1 <= n <= 1000

1 <= fronts[i], backs[i] <= 2000

```

## 823. Binary Trees With Factors

Medium

2381172Add to ListShare

Given an array of unique integers, `arr`, where each integer `arr[i]` is strictly greater than 1.

We make a binary tree using these integers, and each number may be used for any number of times. Each non-leaf node's value should be equal to the product of the values of its children.

Return *the number of binary trees we can make*. The answer may be too large so return the answer **modulo**  $10^9 + 7$ .

### Example 1:

**Input:** arr = [2,4]

**Output:** 3

**Explanation:** We can make these trees: [2], [4], [4, 2, 2]

### Example 2:

**Input:** arr = [2,4,5,10]

**Output:** 7

**Explanation:** We can make these trees: [2], [4], [5], [10], [4, 2, 2], [10, 2, 5], [10, 5, 2].

### Constraints:

```

1 <= arr.length <= 1000

2 <= arr[i] <= 10^9

```

All the values of `arr` are **unique**.

## 824. Goat Latin

Easy

7251145Add to ListShare

You are given a string `sentence` that consists of words separated by spaces. Each word consists of lowercase and uppercase letters only.

We would like to convert the sentence to "Goat Latin" (a made-up language similar to Pig Latin.) The rules of Goat Latin are as follows:

If a word begins with a vowel ('a', 'e', 'i', 'o', or 'u'), append "ma" to the end of the word.

For example, the word "apple" becomes "applema".

If a word begins with a consonant (i.e., not a vowel), remove the first letter and append it to the end, then add "ma".

For example, the word "goat" becomes "oatgma".

Add one letter 'a' to the end of each word per its word index in the sentence, starting with 1.

For example, the first word gets "a" added to the end, the second word gets "aa" added to the end, and so on.

Return *the final sentence representing the conversion from sentence to Goat Latin*.

### Example 1:

**Input:** sentence = "I speak Goat Latin"

**Output:** "Imaa peaksmaaa oatGmaaaa atinLmaaaaa"

### Example 2:

**Input:** sentence = "The quick brown fox jumped over the lazy dog"

**Output:** "heTmaa uickqmaaa rownbmaaaa oxfmaaaaa umpedjmaaaaaa overmaaaaaaa hetmaaaaaaaaa azylmaaaaaaaaa ogdmaaaaaaaaa"

### Constraints:

`1 <= sentence.length <= 150`

`sentence` consists of English letters and spaces.

`sentence` has no leading or trailing spaces.

All the words in `sentence` are separated by a single space.

## 825. Friends Of Appropriate Ages

Medium

5871092Add to ListShare

There are `n` persons on a social media website. You are given an integer array `ages` where `ages[i]` is the age of the `ith` person.

A Person `x` will not send a friend request to a person `y` (`x != y`) if any of the following conditions is true:

```
age[y] <= 0.5 * age[x] + 7
age[y] > age[x]
age[y] > 100 && age[x] < 100
```

Otherwise, `x` will send a friend request to `y`.

Note that if `x` sends a request to `y`, `y` will not necessarily send a request to `x`. Also, a person will not send a friend request to themselves.

Return *the total number of friend requests made*.

### Example 1:

**Input:** `ages = [16,16]`

**Output:** 2

**Explanation:** 2 people friend request each other.

### Example 2:

**Input:** `ages = [16,17,18]`

**Output:** 2

**Explanation:** Friend requests are made  $17 \rightarrow 16$ ,  $18 \rightarrow 17$ .

### Example 3:

**Input:** `ages = [20,30,100,110,120]`

**Output:** 3

**Explanation:** Friend requests are made  $110 \rightarrow 100$ ,  $120 \rightarrow 110$ ,  $120 \rightarrow 100$ .

**Constraints:**

```

n == ages.length

1 <= n <= 2 * 104

1 <= ages[i] <= 120

```

**826. Most Profit Assigning Work****Medium**

1099107Add to ListShare

You have `n` jobs and `m` workers. You are given three arrays: `difficulty`, `profit`, and `worker` where:

`difficulty[i]` and `profit[i]` are the difficulty and the profit of the `ith` job, and `worker[j]` is the ability of `jth` worker (i.e., the `jth` worker can only complete a job with difficulty at most `worker[j]`).

Every worker can be assigned **at most one job**, but one job can be **completed multiple times**.

For example, if three workers attempt the same job that pays `$1`, then the total profit will be `$3`. If a worker cannot complete any job, their profit is `$0`.

Return the maximum profit we can achieve after assigning the workers to the jobs.

**Example 1:**

**Input:** `difficulty = [2,4,6,8,10]`, `profit = [10,20,30,40,50]`, `worker = [4,5,6,7]`

**Output:** `100`

**Explanation:** Workers are assigned jobs of difficulty `[4,4,6,6]` and they get a profit of `[20,20,30,30]` separately.

**Example 2:**

**Input:** `difficulty = [85,47,57]`, `profit = [24,66,99]`, `worker = [40,25,25]`

**Output:** `0`

**Constraints:**

```

n == difficulty.length

n == profit.length

m == worker.length

1 <= n, m <= 104

1 <= difficulty[i], profit[i], worker[i] <= 105

```

## 827. Making A Large Island

Hard

267957Add to ListShare

You are given an  $n \times n$  binary matrix `grid`. You are allowed to change **at most one** 0 to be 1.

Return *the size of the largest island in `grid` after applying this operation*.

An **island** is a 4-directionally connected group of 1s.

### Example 1:

**Input:** `grid = [[1,0],[0,1]]`

**Output:** 3

**Explanation:** Change one 0 to 1 and connect two 1s, then we get an island with area = 3.

### Example 2:

**Input:** `grid = [[1,1],[1,0]]`

**Output:** 4

**Explanation:** Change the 0 to 1 and make the island bigger, only one island with area = 4.

### Example 3:

**Input:** `grid = [[1,1],[1,1]]`

**Output:** 4

**Explanation:** Can't change any 0 to 1, only one island with area = 4.

### Constraints:

```

n == grid.length

n == grid[i].length

1 <= n <= 500

grid[i][j] is either 0 or 1.

```

## 828. Count Unique Characters of All Substrings of a Given String

Hard

1727230Add to ListShare

Let's define a function `countUniqueChars(s)` that returns the number of unique characters on `s`.

For example, calling `countUniqueChars(s)` if `s = "LEETCODE"` then "L", "T", "C", "O", "D" are the unique characters since they appear only once in `s`, therefore `countUniqueChars(s) = 5`.

Given a string `s`, return the sum of `countUniqueChars(t)` where `t` is a substring of `s`. The test cases are generated such that the answer fits in a 32-bit integer.

Notice that some substrings can be repeated so in this case you have to count the repeated ones too.

### Example 1:

**Input:** `s = "ABC"`

**Output:** 10

**Explanation:** All possible substrings are: "A", "B", "C", "AB", "BC" and "ABC".

Every substring is composed with only unique letters.

Sum of lengths of all substring is  $1 + 1 + 1 + 2 + 2 + 3 = 10$

### Example 2:

**Input:** `s = "ABA"`

**Output:** 8

**Explanation:** The same as example 1, except `countUniqueChars("ABA") = 1`.

### Example 3:

**Input:** `s = "LEETCODE"`

**Output:** 92

**Constraints:**

$1 \leq s.length \leq 10^5$

$s$  consists of uppercase English letters only.

## 829. Consecutive Numbers Sum

Hard

11501306Add to ListShare

Given an integer  $n$ , return *the number of ways you can write  $n$  as the sum of consecutive positive integers.*

**Example 1:**

**Input:**  $n = 5$

**Output:** 2

**Explanation:**  $5 = 2 + 3$

**Example 2:**

**Input:**  $n = 9$

**Output:** 3

**Explanation:**  $9 = 4 + 5 = 2 + 3 + 4$

**Example 3:**

**Input:**  $n = 15$

**Output:** 4

**Explanation:**  $15 = 8 + 7 = 4 + 5 + 6 = 1 + 2 + 3 + 4 + 5$

**Constraints:**

$1 \leq n \leq 10^9$

## 830. Positions of Large Groups

**Easy**

682114Add to ListShare

In a string `s` of lowercase letters, these letters form consecutive groups of the same character.

For example, a string like `s = "abbxxxxzzy"` has the groups `"a"`, `"bb"`, `"xxxx"`, `"z"`, and `"yy"`.

A group is identified by an interval `[start, end]`, where `start` and `end` denote the start and end indices (inclusive) of the group. In the above example, `"xxxx"` has the interval `[3, 6]`.

A group is considered **large** if it has 3 or more characters.

Return the intervals of every **large** group sorted in **increasing order by start index**.

**Example 1:**

**Input:** `s = "abbxxxxzzy"`

**Output:** `[[3,6]]`

**Explanation:** `"xxxx"` is the only large group with start index 3 and end index 6.

**Example 2:**

**Input:** `s = "abc"`

**Output:** `[]`

**Explanation:** We have groups `"a"`, `"b"`, and `"c"`, none of which are large groups.

**Example 3:**

**Input:** `s = "abcdddeeeeaaabbbcd"`

**Output:** `[[3,5],[6,9],[12,14]]`

**Explanation:** The large groups are `"ddd"`, `"eee"`, and `"bbb"`.

**Constraints:**

`1 <= s.length <= 1000`

`s` contains lowercase English letters only.

**831. Masking Personal Information****Medium**

127412Add to ListShare

You are given a personal information string `s`, representing either an **email address** or a **phone number**. Return the **masked** personal information using the below rules.

### **Email address:**

An email address is:

A **name** consisting of uppercase and lowercase English letters, followed by

The '@' symbol, followed by

The **domain** consisting of uppercase and lowercase English letters with a dot '.' somewhere in the middle (not the first or last character).

To mask an email:

The uppercase letters in the **name** and **domain** must be converted to lowercase letters.

The middle letters of the **name** (i.e., all but the first and last letters) must be replaced by 5 asterisks "\*\*\*\*\*".

### **Phone number:**

A phone number is formatted as follows:

The phone number contains 10-13 digits.

The last 10 digits make up the **local number**.

The remaining 0-3 digits, in the beginning, make up the **country code**.

**Separation characters** from the set `{ '+', '- ', '(', ') ', ' ' }` separate the above digits in some way.

To mask a phone number:

Remove all **separation characters**.

The masked phone number should have the form:

"\*\*\*\*\*-\*\*\*\*\*-XXXX" if the country code has 0 digits.

"+\*-\*\*-\*\*-\*\*-XXXX" if the country code has 1 digit.

"+\*\*-\*\*-\*\*-\*\*-XXXX" if the country code has 2 digits.

"`*****-****-***-XXXX`" if the country code has 3 digits.

"`XXXX`" is the last 4 digits of the **local number**.

### Example 1:

**Input:** `s = "LeetCode@LeetCode.com"`

**Output:** `"l*****e@leetcode.com"`

**Explanation:** `s` is an email address.

The name and domain are converted to lowercase, and the middle of the name is replaced by 5 asterisks.

### Example 2:

**Input:** `s = "AB@qq.com"`

**Output:** `"a*****b@qq.com"`

**Explanation:** `s` is an email address.

The name and domain are converted to lowercase, and the middle of the name is replaced by 5 asterisks.

Note that even though "ab" is 2 characters, it still must have 5 asterisks in the middle.

### Example 3:

**Input:** `s = "1(234)567-890"`

**Output:** `"***-***-7890"`

**Explanation:** `s` is a phone number.

There are 10 digits, so the local number is 10 digits and the country code is 0 digits.

Thus, the resulting masked number is `"***-***-7890"`.

### Constraints:

`s` is either a **valid** email or a phone number.

If `s` is an email:

```
8 <= s.length <= 40
```

`s` consists of uppercase and lowercase English letters and exactly one '@' symbol and '.' symbol.

If `s` is a phone number:

```
10 <= s.length <= 20
```

`s` consists of digits, spaces, and the symbols '(', ')', '−', and '+'.

## 832. Flipping an Image

**Easy**

2516210Add to ListShare

Given an  $n \times n$  binary matrix `image`, flip the image **horizontally**, then invert it, and return *the resulting image*.

To flip an image horizontally means that each row of the image is reversed.

For example, flipping `[1, 1, 0]` horizontally results in `[0, 1, 1]`.

To invert an image means that each `0` is replaced by `1`, and each `1` is replaced by `0`.

For example, inverting `[0, 1, 1]` results in `[1, 0, 0]`.

### Example 1:

**Input:** `image = [[1,1,0],[1,0,1],[0,0,0]]`

**Output:** `[[1,0,0],[0,1,0],[1,1,1]]`

**Explanation:** First reverse each row: `[[0,1,1],[1,0,1],[0,0,0]]`.

Then, invert the image: `[[1,0,0],[0,1,0],[1,1,1]]`

### Example 2:

**Input:** `image = [[1,1,0,0],[1,0,0,1],[0,1,1,1],[1,0,1,0]]`

**Output:** `[[1,1,0,0],[0,1,1,0],[0,0,0,1],[1,0,1,0]]`

**Explanation:** First reverse each row: `[[0,0,1,1],[1,0,0,1],[1,1,1,0],[0,1,0,1]]`.

Then invert the image: `[[1,1,0,0],[0,1,1,0],[0,0,0,1],[1,0,1,0]]`

**Constraints:**

```

n == image.length

n == image[i].length

1 <= n <= 20

images[i][j] is either 0 or 1.

```

**833. Find And Replace in String****Medium**

963878Add to ListShare

You are given a **0-indexed** string `s` that you must perform `k` replacement operations on. The replacement operations are given as three **0-indexed** parallel arrays, `indices`, `sources`, and `targets`, all of length `k`.

To complete the `ith` replacement operation:

Check if the **substring** `sources[i]` occurs at index `indices[i]` in the **original string** `s`.

If it does not occur, **do nothing**.

Otherwise if it does occur, **replace** that substring with `targets[i]`.

For example, if `s = "abcd"`, `indices[0] = 0`, `sources[0] = "ab"`, and `targets[0] = "eee"`, then the result of this replacement will be `"eeecd"`.

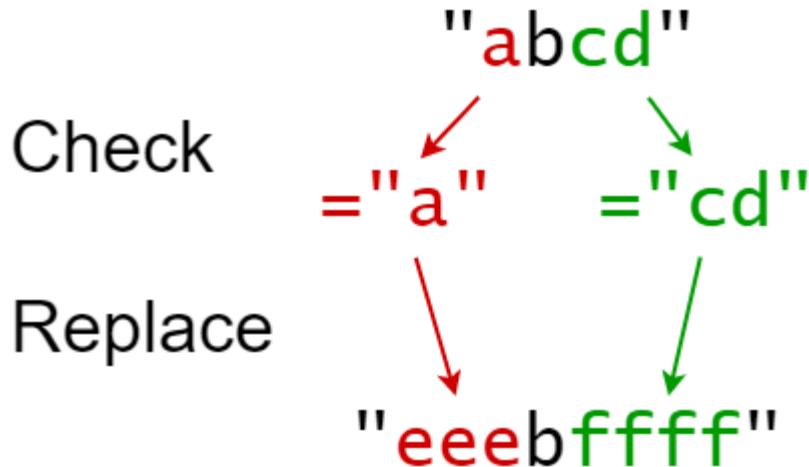
All replacement operations must occur **simultaneously**, meaning the replacement operations should not affect the indexing of each other. The testcases will be generated such that the replacements will **not overlap**.

For example, a testcase with `s = "abc"`, `indices = [0, 1]`, and `sources = ["ab", "bc"]` will not be generated because the `"ab"` and `"bc"` replacements overlap.

Return *the resulting string* after performing all replacement operations on `s`.

A **substring** is a contiguous sequence of characters in a string.

**Example 1:**



**Input:** s = "abcd", indices = [0, 2], sources = ["a", "cd"], targets = ["eee", "ffff"]

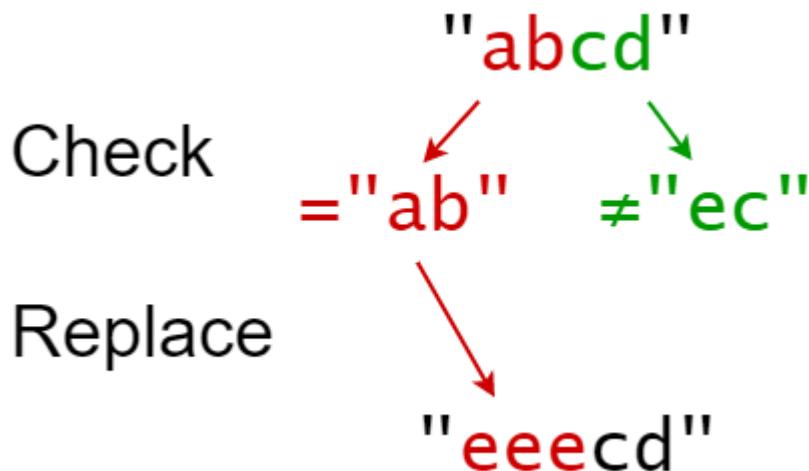
**Output:** "eeebffff"

**Explanation:**

"a" occurs at index 0 in s, so we replace it with "eee".

"cd" occurs at index 2 in s, so we replace it with "ffff".

**Example 2:**



**Input:** s = "abcd", indices = [0, 2], sources = ["ab", "ec"], targets = ["eee", "ffff"]

**Output:** "eeecd"

**Explanation:**

"ab" occurs at index 0 in s, so we replace it with "eee".

"ec" does not occur at index 2 in s, so we do nothing.

**Constraints:**

```

1 <= s.length <= 1000

k == indices.length == sources.length == targets.length

1 <= k <= 100

0 <= indexes[i] < s.length

1 <= sources[i].length, targets[i].length <= 50

s consists of only lowercase English letters.

sources[i] and targets[i] consist of only lowercase English letters.

```

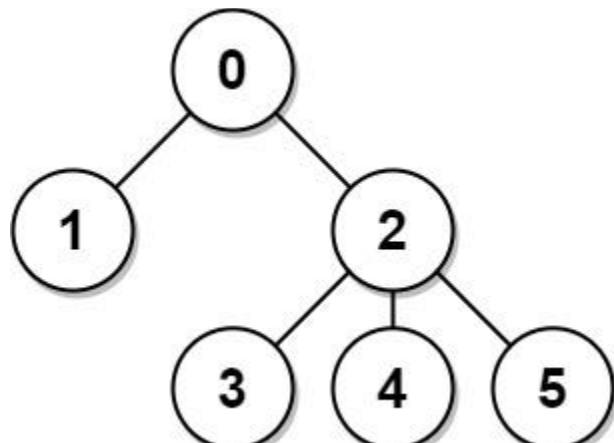
**834. Sum of Distances in Tree****Hard**

295663Add to ListShare

There is an undirected connected tree with  $n$  nodes labeled from  $0$  to  $n - 1$  and  $n - 1$  edges.

You are given the integer  $n$  and the array  $\text{edges}$  where  $\text{edges}[i] = [a_i, b_i]$  indicates that there is an edge between nodes  $a_i$  and  $b_i$  in the tree.

Return an array  $\text{answer}$  of length  $n$  where  $\text{answer}[i]$  is the sum of the distances between the  $i^{\text{th}}$  node in the tree and all other nodes.

**Example 1:**

**Input:**  $n = 6$ ,  $\text{edges} = [[0,1],[0,2],[2,3],[2,4],[2,5]]$

**Output:** [8,12,6,10,10,10]

**Explanation:** The tree is shown above.

We can see that  $\text{dist}(0,1) + \text{dist}(0,2) + \text{dist}(0,3) + \text{dist}(0,4) + \text{dist}(0,5)$

equals  $1 + 1 + 2 + 2 + 2 = 8$ .

Hence,  $\text{answer}[0] = 8$ , and so on.

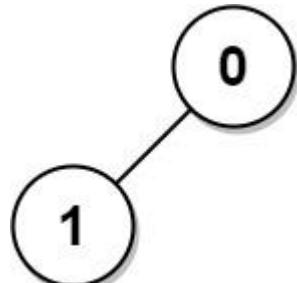
**Example 2:**



**Input:**  $n = 1$ ,  $\text{edges} = []$

**Output:** [0]

**Example 3:**



**Input:**  $n = 2$ ,  $\text{edges} = [[1,0]]$

**Output:** [1,1]

**Constraints:**

```

1 <= n <= 3 * 10^4
edges.length == n - 1
edges[i].length == 2
0 <= a_i, b_i < n
a_i != b_i
  
```

The given input represents a valid tree.

## 835. Image Overlap

### Medium

15341 Add to List Share

You are given two images, `img1` and `img2`, represented as binary, square matrices of size  $n \times n$ . A binary matrix has only `0s` and `1s` as values.

We **translate** one image however we choose by sliding all the `1` bits left, right, up, and/or down any number of units. We then place it on top of the other image. We can then calculate the **overlap** by counting the number of positions that have a `1` in **both** images.

Note also that a translation does **not** include any kind of rotation. Any `1` bits that are translated outside of the matrix borders are erased.

Return *the largest possible overlap*.

#### Example 1:

1	1	0
0	1	0
0	1	0

img1

0	0	0
0	1	1
0	0	1

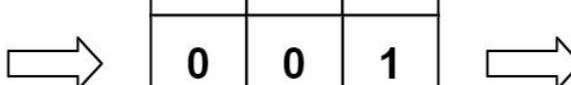
img2

**Input:** `img1 = [[1,1,0],[0,1,0],[0,1,0]]`, `img2 = [[0,0,0],[0,1,1],[0,0,1]]`

**Output:** 3

**Explanation:** We translate `img1` to right by 1 unit and down by 1 unit.

1	1	0
0	1	0
0	1	0



0	1	1
0	0	1
0	0	1

0	0	0
0	1	1
0	0	1

The number of positions that have a 1 in both images is 3 (shown in red).

0	0	0
0	1	1
0	0	1

img1

0	0	0
0	1	1
0	0	1

img2

**Example 2:**

**Input:** img1 = [[1]], img2 = [[1]]

**Output:** 1

**Example 3:**

**Input:** img1 = [[0]], img2 = [[0]]

**Output:** 0

**Constraints:**

n == img1.length == img1[i].length

n == img2.length == img2[i].length

1 <= n <= 30

img1[i][j] is either 0 or 1.

img2[i][j] is either 0 or 1.

## 836. Rectangle Overlap

Easy

1543404Add to ListShare

An axis-aligned rectangle is represented as a list `[x1, y1, x2, y2]`, where  $(x_1, y_1)$  is the coordinate of its bottom-left corner, and  $(x_2, y_2)$  is the coordinate of its top-right corner. Its top and bottom edges are parallel to the X-axis, and its left and right edges are parallel to the Y-axis.

Two rectangles overlap if the area of their intersection is **positive**. To be clear, two rectangles that only touch at the corner or edges do not overlap.

Given two axis-aligned rectangles `rec1` and `rec2`, return `true` if they overlap, otherwise return `false`.

### Example 1:

**Input:** `rec1 = [0,0,2,2]`, `rec2 = [1,1,3,3]`

**Output:** `true`

### Example 2:

**Input:** `rec1 = [0,0,1,1]`, `rec2 = [1,0,2,1]`

**Output:** `false`

### Example 3:

**Input:** `rec1 = [0,0,1,1]`, `rec2 = [2,2,3,3]`

**Output:** `false`

### Constraints:

```
rec1.length == 4
rec2.length == 4
-109 <= rec1[i], rec2[i] <= 109
```

`rec1` and `rec2` represent a valid rectangle with a non-zero area.

## 837. New 21 Game

Medium

1011685Add to ListShare

Alice plays the following game, loosely based on the card game "**21**".

Alice starts with `0` points and draws numbers while she has less than `k` points. During each draw, she gains an integer number of points randomly from the range `[1, maxPts]`, where `maxPts` is an integer. Each draw is independent and the outcomes have equal probabilities.

Alice stops drawing numbers when she gets `k` or more points.

Return the probability that Alice has `n` or fewer points.

Answers within  $10^{-5}$  of the actual answer are considered accepted.

### Example 1:

**Input:** `n = 10, k = 1, maxPts = 10`

**Output:** `1.00000`

**Explanation:** Alice gets a single card, then stops.

### Example 2:

**Input:** `n = 6, k = 1, maxPts = 10`

**Output:** `0.60000`

**Explanation:** Alice gets a single card, then stops.

In 6 out of 10 possibilities, she is at or below 6 points.

### Example 3:

**Input:** `n = 21, k = 17, maxPts = 10`

**Output:** `0.73278`

### Constraints:

`0 <= k <= n <= 104`

`1 <= maxPts <= 104`

## 838. Push Dominoes

Medium

1784120Add to ListShare

There are `n` dominoes in a line, and we place each domino vertically upright. In the beginning, we simultaneously push some of the dominoes either to the left or to the right.

After each second, each domino that is falling to the left pushes the adjacent domino on the left. Similarly, the dominoes falling to the right push their adjacent dominoes standing on the right.

When a vertical domino has dominoes falling on it from both sides, it stays still due to the balance of the forces.

For the purposes of this question, we will consider that a falling domino expends no additional force to a falling or already fallen domino.

You are given a string `dominoes` representing the initial state where:

`dominoes[i] = 'L'`, if the  $i^{\text{th}}$  domino has been pushed to the left,

`dominoes[i] = 'R'`, if the  $i^{\text{th}}$  domino has been pushed to the right, and

`dominoes[i] = '.'`, if the  $i^{\text{th}}$  domino has not been pushed.

Return a string representing the final state.

### Example 1:

**Input:** `dominoes = "RR.L"`

**Output:** `"RR.L"`

**Explanation:** The first domino expends no additional force on the second domino.

### Example 2:



**Input:** `dominoes = ".L.R...LR..L.."`

**Output:** `"LL.RR.LLRRLL.."`

### Constraints:

```

n == dominoes.length

1 <= n <= 105

dominoes[i] is either 'L', 'R', or '.'.

```

## 839. Similar String Groups

Hard

971179Add to ListShare

Two strings  $X$  and  $Y$  are similar if we can swap two letters (in different positions) of  $X$ , so that it equals  $Y$ . Also two strings  $X$  and  $Y$  are similar if they are equal.

For example, "tars" and "rats" are similar (swapping at positions 0 and 2), and "rats" and "arts" are similar, but "star" is not similar to "tars", "rats", or "arts".

Together, these form two connected groups by similarity: {"tars", "rats", "arts"} and {"star"}. Notice that "tars" and "arts" are in the same group even though they are not similar. Formally, each group is such that a word is in the group if and only if it is similar to at least one other word in the group.

We are given a list  $strs$  of strings where every string in  $strs$  is an anagram of every other string in  $strs$ . How many groups are there?

### Example 1:

**Input:**  $strs = ["tars", "rats", "arts", "star"]$

**Output:** 2

### Example 2:

**Input:**  $strs = ["omv", "ovm"]$

**Output:** 1

### Constraints:

$1 \leq strs.length \leq 300$

$1 \leq strs[i].length \leq 300$

$strs[i]$  consists of lowercase letters only.

All words in  $strs$  have the same length and are anagrams of each other.

## 840. Magic Squares In Grid

Medium

2651501Add to ListShare

A  $3 \times 3$  magic square is a  $3 \times 3$  grid filled with distinct numbers **from 1 to 9** such that each row, column, and both diagonals all have the same sum.

Given a `row x col grid` of integers, how many  $3 \times 3$  "magic square" subgrids are there? (Each subgrid is contiguous).

**Example 1:**

4	3	8	4
9	5	1	9
2	7	6	2

**Input:** `grid = [[4,3,8,4],[9,5,1,9],[2,7,6,2]]`

**Output:** 1

**Explanation:**

The following subgrid is a  $3 \times 3$  magic square:

4	3	8
9	5	1
2	7	6

while this one is not:

3	8	4
5	1	9
7	6	2

In total, there is only one magic square inside the given grid.

**Example 2:**

**Input:** grid = [[8]]

**Output:** 0

**Constraints:**

```
row == grid.length
col == grid[i].length
1 <= row, col <= 10
0 <= grid[i][j] <= 15
```

## 841. Keys and Rooms

Medium

3677190Add to ListShare

There are  $n$  rooms labeled from 0 to  $n - 1$  and all the rooms are locked except for room 0. Your goal is to visit all the rooms. However, you cannot enter a locked room without having its key.

When you visit a room, you may find a set of **distinct keys** in it. Each key has a number on it, denoting which room it unlocks, and you can take all of them with you to unlock the other rooms.

Given an array `rooms` where `rooms[i]` is the set of keys that you can obtain if you visited room `i`, return `true` if you can visit **all the rooms**, or `false` otherwise.

**Example 1:**

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

**Output:** true

**Explanation:**

We visit room 0 and pick up key 1.

We then visit room 1 and pick up key 2.

We then visit room 2 and pick up key 3.

We then visit room 3.

Since we were able to visit every room, we return true.

**Example 2:**

**Input:** rooms = [[1,3],[3,0,1],[2],[0]]

**Output:** false

**Explanation:** We can not enter room number 2 since the only key that unlocks it is in that room.

**Constraints:**

```

n == rooms.length
2 <= n <= 1000
0 <= rooms[i].length <= 1000
1 <= sum(rooms[i].length) <= 3000
0 <= rooms[i][j] < n

```

All the values of `rooms[i]` are **unique**.

## 842. Split Array into Fibonacci Sequence

Medium

925271 Add to List Share

You are given a string of digits `num`, such as "123456579". We can split it into a Fibonacci-like sequence [123, 456, 579].

Formally, a **Fibonacci-like** sequence is a list  $f$  of non-negative integers such that:

$0 \leq f[i] < 2^{31}$ , (that is, each integer fits in a **32-bit** signed integer type),

$f.length \geq 3$ , and

$f[i] + f[i + 1] == f[i + 2]$  for all  $0 \leq i < f.length - 2$ .

Note that when splitting the string into pieces, each piece must not have extra leading zeroes, except if the piece is the number `0` itself.

Return any Fibonacci-like sequence split from `num`, or return `[]` if it cannot be done.

### Example 1:

**Input:** `num = "1101111"`

**Output:** `[11,0,11,11]`

**Explanation:** The output `[110, 1, 111]` would also be accepted.

### Example 2:

**Input:** `num = "112358130"`

**Output:** `[]`

**Explanation:** The task is impossible.

### Example 3:

**Input:** `num = "0123"`

**Output:** `[]`

**Explanation:** Leading zeroes are not allowed, so `"01", "2", "3"` is not valid.

### Constraints:

$1 \leq num.length \leq 200$

`num` contains only digits.

## 843. Guess the Word

Hard

12981551Add to ListShare

You are given an array of unique strings `words` where `words[i]` is six letters long. One word of `words` was chosen as a secret word.

You are also given the helper object `Master`. You may call `Master.guess(word)` where `word` is a six-letter-long string, and it must be from `words`. `Master.guess(word)` returns:

`-1` if `word` is not from `words`, or

an integer representing the number of exact matches (value and position) of your guess to the secret word.

There is a parameter `allowedGuesses` for each test case where `allowedGuesses` is the maximum number of times you can call `Master.guess(word)`.

For each test case, you should call `Master.guess` with the secret word without exceeding the maximum number of allowed guesses. You will get:

`"Either you took too many guesses, or you did not find the secret word."` if you called `Master.guess` more than `allowedGuesses` times or if you did not call `Master.guess` with the secret word, or

`"You guessed the secret word correctly."` if you called `Master.guess` with the secret word with the number of calls to `Master.guess` less than or equal to `allowedGuesses`.

The test cases are generated such that you can guess the secret word with a reasonable strategy (other than using the bruteforce method).

### Example 1:

**Input:** `secret = "acckzz", words = ["acckzz", "ccbazz", "eiowzz", "abcczz"], allowedGuesses = 10`

**Output:** You guessed the secret word correctly.

#### Explanation:

`master.guess("aaaaaa")` returns `-1`, because `"aaaaaa"` is not in wordlist.

`master.guess("acckzz")` returns `6`, because `"acckzz"` is secret and has all 6 matches.

`master.guess("ccbazz")` returns `3`, because `"ccbazz"` has 3 matches.

`master.guess("eiowzz")` returns `2`, because `"eiowzz"` has 2 matches.

`master.guess("abcczz")` returns `4`, because `"abcczz"` has 4 matches.

We made 5 calls to `master.guess`, and one of them was the secret, so we pass the test case.

### Example 2:

**Input:** `secret = "hamada", words = ["hamada", "khaled"]`, `allowedGuesses = 10`

**Output:** You guessed the secret word correctly.

**Explanation:** Since there are two words, you can guess both.

### Constraints:

`1 <= words.length <= 100`

`words[i].length == 6`

`words[i]` consist of lowercase English letters.

All the strings of `wordlist` are **unique**.

`secret` exists in `words`.

`10 <= allowedGuesses <= 30`

## 844. Backspace String Compare

Easy

5565246Add to ListShare

Given two strings `s` and `t`, return `true` if they are equal when both are typed into empty text editors. '`#`' means a backspace character.

Note that after backspacing an empty text, the text will continue empty.

### Example 1:

**Input:** `s = "ab#c", t = "ad#c"`

**Output:** `true`

**Explanation:** Both `s` and `t` become "ac".

### Example 2:

**Input:** `s = "ab##", t = "c#d#"`

**Output:** true

**Explanation:** Both s and t become "".

**Example 3:**

**Input:** s = "a#c", t = "b"

**Output:** false

**Explanation:** s becomes "c" while t becomes "b".

**Constraints:**

`1 <= s.length, t.length <= 200`

`s` and `t` only contain lowercase letters and '`#`' characters.

## 845. Longest Mountain in Array

Medium

218762Add to ListShare

You may recall that an array `arr` is a **mountain array** if and only if:

`arr.length >= 3`

There exists some index `i` (**0-indexed**) with `0 < i < arr.length - 1` such that:

`arr[0] < arr[1] < ... < arr[i - 1] < arr[i]`

`arr[i] > arr[i + 1] > ... > arr[arr.length - 1]`

Given an integer array `arr`, return *the length of the longest subarray, which is a mountain*. Return 0 if there is no mountain subarray.

**Example 1:**

**Input:** arr = [2,1,4,7,3,2,5]

**Output:** 5

**Explanation:** The largest mountain is [1,4,7,3,2] which has length 5.

**Example 2:**

**Input:** arr = [2,2,2]

**Output:** 0

**Explanation:** There is no mountain.

**Constraints:**

1 <= arr.length <= 10<sup>4</sup>

0 <= arr[i] <= 10<sup>4</sup>

## 846. Hand of Straights

Medium

1616128Add to ListShare

Alice has some number of cards and she wants to rearrange the cards into groups so that each group is of size `groupSize`, and consists of `groupSize` consecutive cards.

Given an integer array `hand` where `hand[i]` is the value written on the `ith` card and an integer `groupSize`, return `true` if she can rearrange the cards, or `false` otherwise.

**Example 1:**

**Input:** hand = [1,2,3,6,2,3,4,7,8], groupSize = 3

**Output:** true

**Explanation:** Alice's hand can be rearranged as [1,2,3],[2,3,4],[6,7,8]

**Example 2:**

**Input:** hand = [1,2,3,4,5], groupSize = 4

**Output:** false

**Explanation:** Alice's hand can not be rearranged into groups of 4.

**Constraints:**

1 <= hand.length <= 10<sup>4</sup>

```
0 <= hand[i] <= 109
1 <= groupSize <= hand.length
```

## 847. Shortest Path Visiting All Nodes

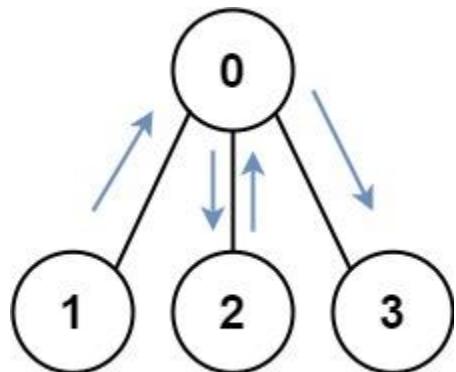
Hard

2860133Add to ListShare

You have an undirected, connected graph of  $n$  nodes labeled from 0 to  $n - 1$ . You are given an array `graph` where `graph[i]` is a list of all the nodes connected with node  $i$  by an edge.

Return *the length of the shortest path that visits every node*. You may start and stop at any node, you may revisit nodes multiple times, and you may reuse edges.

**Example 1:**

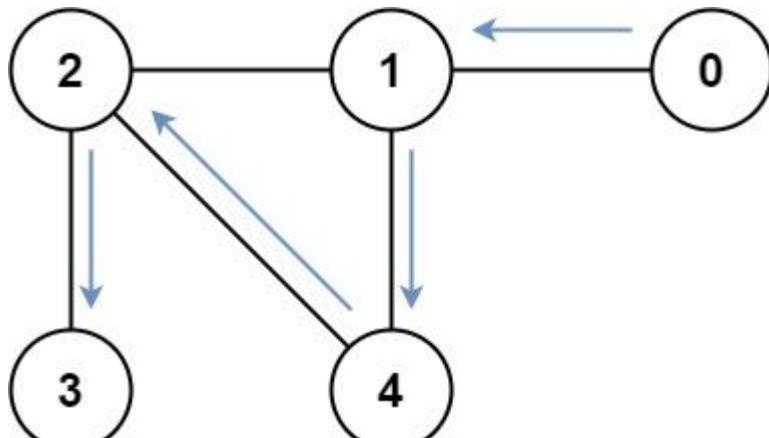


**Input:** `graph = [[1,2,3],[0],[0],[0]]`

**Output:** 4

**Explanation:** One possible path is [1,0,2,0,3]

**Example 2:**



**Input:** graph = [[1],[0,2,4],[1,3,4],[2],[1,2]]

**Output:** 4

**Explanation:** One possible path is [0,1,4,2,3]

### Constraints:

```

n == graph.length
1 <= n <= 12
0 <= graph[i].length < n
graph[i] does not contain i.

```

If graph[a] contains b, then graph[b] contains a.

The input graph is always connected.

## 848. Shifting Letters

Medium

1050104Add to ListShare

You are given a string `s` of lowercase English letters and an integer array `shifts` of the same length.

Call the `shift()` of a letter, the next letter in the alphabet, (wrapping around so that 'z' becomes 'a').

For example, `shift('a') = 'b'`, `shift('t') = 'u'`, and `shift('z') = 'a'`.

Now for each `shifts[i] = x`, we want to shift the first `i + 1` letters of `s`, `x` times.

Return the final string after all such shifts to `s` are applied.

**Example 1:**

**Input:** `s = "abc", shifts = [3,5,9]`

**Output:** `"rpl"`

**Explanation:** We start with `"abc"`.

After shifting the first 1 letters of `s` by 3, we have `"dbc"`.

After shifting the first 2 letters of `s` by 5, we have `"igc"`.

After shifting the first 3 letters of `s` by 9, we have `"rpl"`, the answer.

**Example 2:**

**Input:** `s = "aaa", shifts = [1,2,3]`

**Output:** `"gfd"`

**Constraints:**

`1 <= s.length <= 105`

`s` consists of lowercase English letters.

`shifts.length == s.length`

`0 <= shifts[i] <= 109`

## 849. Maximize Distance to Closest Person

**Medium**

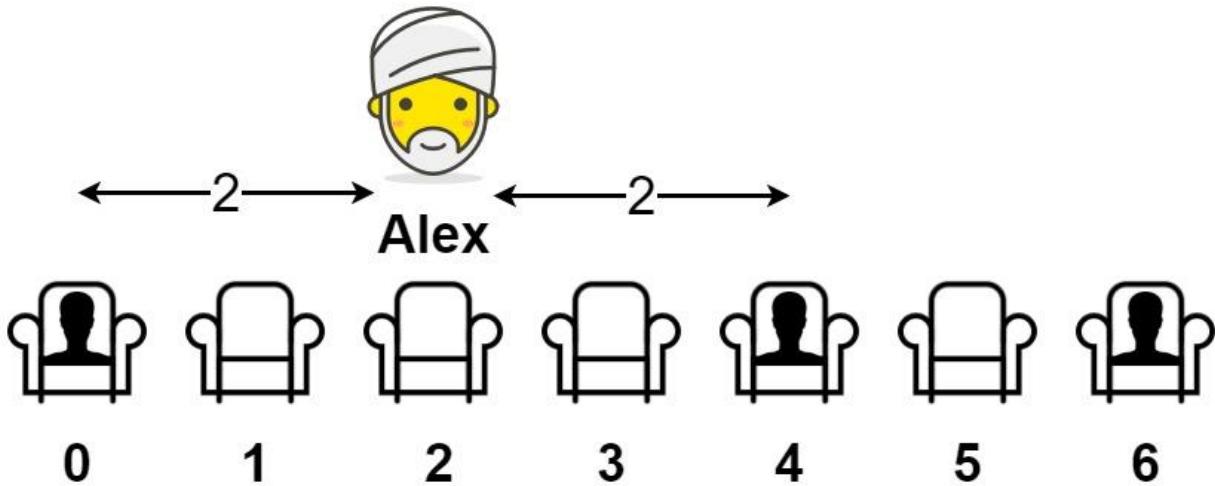
2772176Add to ListShare

You are given an array representing a row of `seats` where `seats[i] = 1` represents a person sitting in the `ith` seat, and `seats[i] = 0` represents that the `ith` seat is empty (**0-indexed**).

There is at least one empty seat, and at least one person sitting.

Alex wants to sit in the seat such that the distance between him and the closest person to him is maximized.

Return *that maximum distance to the closest person*.

**Example 1:**

**Input:** seats = [1,0,0,0,1,0,1]

**Output:** 2

**Explanation:**

If Alex sits in the second open seat (i.e. seats[2]), then the closest person has distance 2.

If Alex sits in any other open seat, the closest person has distance 1.

Thus, the maximum distance to the closest person is 2.

**Example 2:**

**Input:** seats = [1,0,0,0]

**Output:** 3

**Explanation:**

If Alex sits in the last seat (i.e. seats[3]), the closest person is 3 seats away.

This is the maximum distance possible, so the answer is 3.

**Example 3:**

**Input:** seats = [0,1]

**Output:** 1

**Constraints:**

2 <= seats.length <= 2 \* 10<sup>4</sup>

`seats[i]` is 0 or 1.

At least one seat is **empty**.

At least one seat is **occupied**.

## 850. Rectangle Area II

Hard

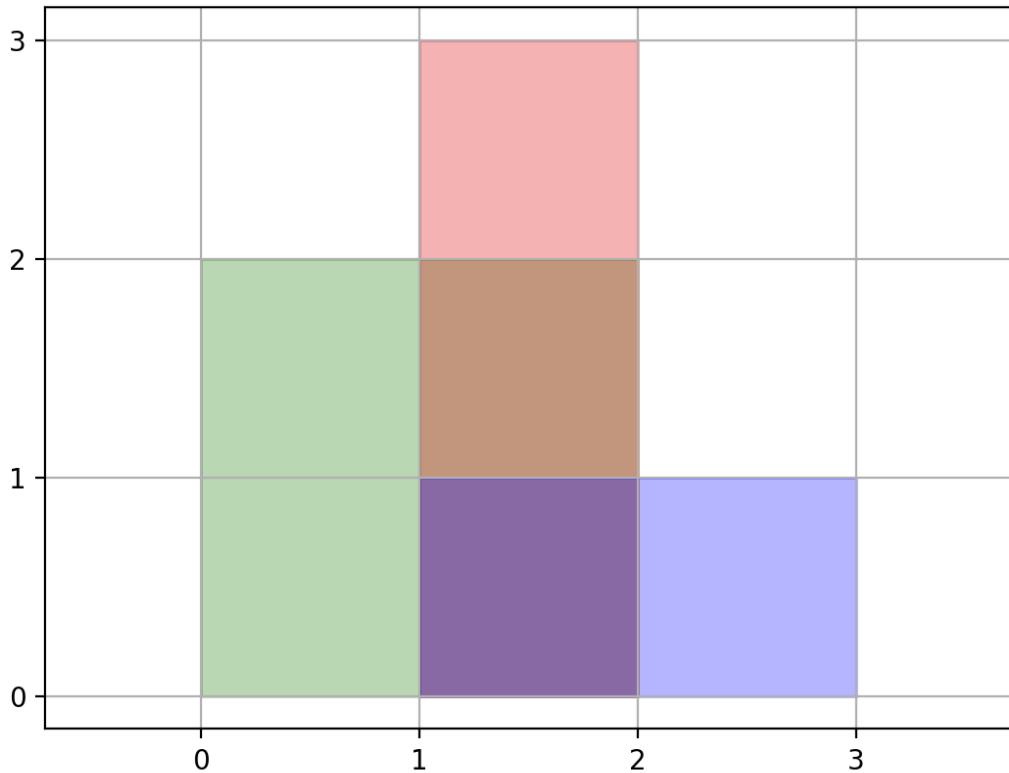
84055Add to ListShare

You are given a 2D array of axis-aligned rectangles. Each `rectangle[i] = [xi1, yi1, xi2, yi2]` denotes the *i<sup>th</sup>* rectangle where  $(x_{i1}, y_{i1})$  are the coordinates of the **bottom-left corner**, and  $(x_{i2}, y_{i2})$  are the coordinates of the **top-right corner**.

Calculate the **total area** covered by all `rectangles` in the plane. Any area covered by two or more rectangles should only be counted **once**.

Return *the total area*. Since the answer may be too large, return it **modulo**  $10^9 + 7$ .

**Example 1:**



**Input:** rectangles = [[0,0,2,2],[1,0,2,3],[1,0,3,1]]

**Output:** 6

**Explanation:** A total area of 6 is covered by all three rectangles, as illustrated in the picture.

From (1,1) to (2,2), the green and red rectangles overlap.

From (1,0) to (2,3), all three rectangles overlap.

**Example 2:**

**Input:** rectangles = [[0,0,1000000000,1000000000]]

**Output:** 49

**Explanation:** The answer is  $10^{18}$  modulo  $(10^9 + 7)$ , which is 49.

**Constraints:**

```
1 <= rectangles.length <= 200
```

```
rectanges[i].length == 4
0 <= xi1, yi1, xi2, yi2 <= 109
```

## 851. Loud and Rich

Medium

820636Add to ListShare

There is a group of  $n$  people labeled from  $0$  to  $n - 1$  where each person has a different amount of money and a different level of quietness.

You are given an array `richer` where  $\text{richer}[i] = [a_i, b_i]$  indicates that  $a_i$  has more money than  $b_i$  and an integer array `quiet` where  $\text{quiet}[i]$  is the quietness of the  $i^{\text{th}}$  person. All the given data in `richer` are **logically correct** (i.e., the data will not lead you to a situation where  $x$  is richer than  $y$  and  $y$  is richer than  $x$  at the same time).

Return an integer array `answer` where  $\text{answer}[x] = y$  if  $y$  is the least quiet person (that is, the person  $y$  with the smallest value of  $\text{quiet}[y]$ ) among all people who definitely have equal to or more money than the person  $x$ .

### Example 1:

**Input:** `richer` =  $[[1,0],[2,1],[3,1],[3,7],[4,3],[5,3],[6,3]]$ , `quiet` =  $[3,2,5,4,6,1,7,0]$

**Output:**  $[5,5,2,5,4,5,6,7]$

### Explanation:

`answer[0] = 5`.

Person 5 has more money than 3, which has more money than 1, which has more money than 0.

The only person who is quieter (has lower  $\text{quiet}[x]$ ) is person 7, but it is not clear if they have more money than person 0.

`answer[7] = 7`.

Among all people that definitely have equal to or more money than person 7 (which could be persons 3, 4, 5, 6, or 7), the person who is the quietest (has lower  $\text{quiet}[x]$ ) is person 7.

The other answers can be filled out with similar reasoning.

### Example 2:

**Input:** `richer` =  $[]$ , `quiet` =  $[0]$

**Output:** [0]

**Constraints:**

`n == quiet.length`

`1 <= n <= 500`

`0 <= quiet[i] < n`

All the values of `quiet` are **unique**.

`0 <= richer.length <= n * (n - 1) / 2`

`0 <= ai, bi < n`

`ai != bi`

All the pairs of `richer` are **unique**.

The observations in `richer` are all logically consistent.

## 852. Peak Index in a Mountain Array

Medium

37561730Add to ListShare

An array `arr` a **mountain** if the following properties hold:

`arr.length >= 3`

There exists some `i` with `0 < i < arr.length - 1` such that:

`arr[0] < arr[1] < ... < arr[i - 1] < arr[i]`

`arr[i] > arr[i + 1] > ... > arr[arr.length - 1]`

Given a mountain array `arr`, return the index `i` such that `arr[0] < arr[1] < ... < arr[i - 1] < arr[i] > arr[i + 1] > ... > arr[arr.length - 1]`.

You must solve it in `O(log(arr.length))` time complexity.

**Example 1:**

**Input:** `arr = [0,1,0]`

**Output:** 1

**Example 2:**

**Input:** arr = [0,2,1,0]

**Output:** 1

**Example 3:**

**Input:** arr = [0,10,5,2]

**Output:** 1

**Constraints:**

3 <= arr.length <= 10<sup>5</sup>

0 <= arr[i] <= 10<sup>6</sup>

arr is **guaranteed** to be a mountain array.

## 853. Car Fleet

Medium

1892481Add to ListShare

There are n cars going to the same destination along a one-lane road. The destination is target miles away.

You are given two integer array position and speed, both of length n, where position[i] is the position of the  $i^{\text{th}}$  car and speed[i] is the speed of the  $i^{\text{th}}$  car (in miles per hour).

A car can never pass another car ahead of it, but it can catch up to it and drive bumper to bumper **at the same speed**. The faster car will **slow down** to match the slower car's speed. The distance between these two cars is ignored (i.e., they are assumed to have the same position).

A **car fleet** is some non-empty set of cars driving at the same position and same speed. Note that a single car is also a car fleet.

If a car catches up to a car fleet right at the destination point, it will still be considered as one car fleet.

Return *the number of car fleets* that will arrive at the destination.

**Example 1:**

**Input:** target = 12, position = [10,8,0,5,3], speed = [2,4,1,1,3]

**Output:** 3

**Explanation:**

The cars starting at 10 (speed 2) and 8 (speed 4) become a fleet, meeting each other at 12.

The car starting at 0 does not catch up to any other car, so it is a fleet by itself.

The cars starting at 5 (speed 1) and 3 (speed 3) become a fleet, meeting each other at 6. The fleet moves at speed 1 until it reaches target.

Note that no other cars meet these fleets before the destination, so the answer is 3.

**Example 2:**

**Input:** target = 10, position = [3], speed = [3]

**Output:** 1

**Explanation:** There is only one car, hence there is only one fleet.

**Example 3:**

**Input:** target = 100, position = [0,2,4], speed = [4,2,1]

**Output:** 1

**Explanation:**

The cars starting at 0 (speed 4) and 2 (speed 2) become a fleet, meeting each other at 4. The fleet moves at speed 2.

Then, the fleet (speed 2) and the car starting at 4 (speed 1) become one fleet, meeting each other at 6. The fleet moves at speed 1 until it reaches target.

**Constraints:**

`n == position.length == speed.length`

`1 <= n <= 105`

`0 < target <= 106`

`0 <= position[i] < target`

All the values of `position` are **unique**.

```
0 < speed[i] <= 106
```

## 854. K-Similar Strings

Hard

92252Add to ListShare

Strings  $s_1$  and  $s_2$  are  **$k$ -similar** (for some non-negative integer  $k$ ) if we can swap the positions of two letters in  $s_1$  exactly  $k$  times so that the resulting string equals  $s_2$ .

Given two anagrams  $s_1$  and  $s_2$ , return the smallest  $k$  for which  $s_1$  and  $s_2$  are  **$k$ -similar**.

### Example 1:

**Input:**  $s_1 = "ab"$ ,  $s_2 = "ba"$

**Output:** 1

**Explanation:** The two strings are 1-similar because we can use one swap to change  $s_1$  to  $s_2$ : "ab"  $\rightarrow$  "ba".

### Example 2:

**Input:**  $s_1 = "abc"$ ,  $s_2 = "bca"$

**Output:** 2

**Explanation:** The two strings are 2-similar because we can use two swaps to change  $s_1$  to  $s_2$ : "abc"  $\rightarrow$  "bac"  $\rightarrow$  "bca".

### Constraints:

```
1 <= s1.length <= 20
```

```
s2.length == s1.length
```

```
s1 and s2 contain only lowercase letters from the set {'a', 'b', 'c', 'd', 'e', 'f'}.  
s2 is an anagram of s1.
```

## 855. Exam Room

Medium

1062416Add to ListShare

There is an exam room with  $n$  seats in a single row labeled from 0 to  $n - 1$ .

When a student enters the room, they must sit in the seat that maximizes the distance to the closest person. If there are multiple such seats, they sit in the seat with the lowest number. If no one is in the room, then the student sits at seat number 0.

Design a class that simulates the mentioned exam room.

Implement the `ExamRoom` class:

`ExamRoom(int n)` Initializes the object of the exam room with the number of the seats `n`.

`int seat()` Returns the label of the seat at which the next student will sit.

`void leave(int p)` Indicates that the student sitting at seat `p` will leave the room. It is guaranteed that there will be a student sitting at seat `p`.

### Example 1:

#### Input

```
["ExamRoom", "seat", "seat", "seat", "seat", "leave", "seat"]
[[10], [], [], [], [4], []]
```

#### Output

```
[null, 0, 9, 4, 2, null, 5]
```

#### Explanation

```
ExamRoom examRoom = new ExamRoom(10);

examRoom.seat(); // return 0, no one is in the room, then the student sits at seat number 0.

examRoom.seat(); // return 9, the student sits at the last seat number 9.

examRoom.seat(); // return 4, the student sits at the last seat number 4.

examRoom.seat(); // return 2, the student sits at the last seat number 2.

examRoom.leave(4);

examRoom.seat(); // return 5, the student sits at the last seat number 5.
```

**Constraints:**

```
1 <= n <= 109
```

It is guaranteed that there is a student sitting at seat  $p$ .

At most  $10^4$  calls will be made to `seat` and `leave`.

**856. Score of Parentheses****Medium**

4700158Add to ListShare

Given a balanced parentheses string  $s$ , return the **score** of the string.

The **score** of a balanced parentheses string is based on the following rule:

"()" has score 1.

$AB$  has score  $A + B$ , where  $A$  and  $B$  are balanced parentheses strings.

$(A)$  has score  $2 * A$ , where  $A$  is a balanced parentheses string.

**Example 1:**

**Input:**  $s = "()"$

**Output:** 1

**Example 2:**

**Input:**  $s = "((()))"$

**Output:** 2

**Example 3:**

**Input:**  $s = "(()())"$

**Output:** 2

**Constraints:**

```
2 <= s.length <= 50
```

$s$  consists of only '(' and ')'.  
)

`s` is a balanced parentheses string.

## 857. Minimum Cost to Hire K Workers

Hard

1823225Add to ListShare

There are `n` workers. You are given two integer arrays `quality` and `wage` where `quality[i]` is the quality of the `ith` worker and `wage[i]` is the minimum wage expectation for the `ith` worker.

We want to hire exactly `k` workers to form a paid group. To hire a group of `k` workers, we must pay them according to the following rules:

Every worker in the paid group should be paid in the ratio of their quality compared to other workers in the paid group.

Every worker in the paid group must be paid at least their minimum wage expectation.

Given the integer `k`, return *the least amount of money needed to form a paid group satisfying the above conditions*. Answers within  $10^{-5}$  of the actual answer will be accepted.

### Example 1:

**Input:** `quality = [10,20,5], wage = [70,50,30], k = 2`

**Output:** `105.00000`

**Explanation:** We pay 70 to 0<sup>th</sup> worker and 35 to 2<sup>nd</sup> worker.

### Example 2:

**Input:** `quality = [3,1,10,10,1], wage = [4,8,2,2,7], k = 3`

**Output:** `30.66667`

**Explanation:** We pay 4 to 0<sup>th</sup> worker, 13.33333 to 2<sup>nd</sup> and 3<sup>rd</sup> workers separately.

### Constraints:

`n == quality.length == wage.length`

`1 <= k <= n <= 104`

`1 <= quality[i], wage[i] <= 104`

## 858. Mirror Reflection

**Medium**

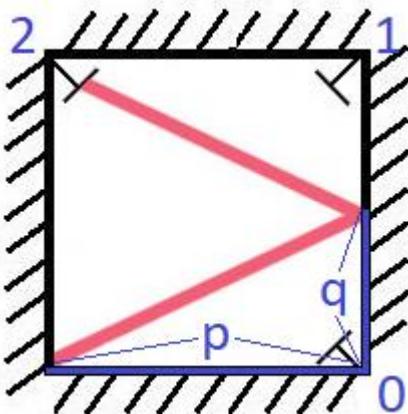
10202395Add to ListShare

There is a special square room with mirrors on each of the four walls. Except for the southwest corner, there are receptors on each of the remaining corners, numbered 0, 1, and 2.

The square room has walls of length  $p$  and a laser ray from the southwest corner first meets the east wall at a distance  $q$  from the  $0^{\text{th}}$  receptor.

Given the two integers  $p$  and  $q$ , return *the number of the receptor that the ray meets first*.

The test cases are guaranteed so that the ray will meet a receptor eventually.

**Example 1:**

**Input:**  $p = 2$ ,  $q = 1$

**Output:** 2

**Explanation:** The ray meets receptor 2 the first time it gets reflected back to the left wall.

**Example 2:**

**Input:**  $p = 3$ ,  $q = 1$

**Output:** 1

**Constraints:**

$1 \leq q \leq p \leq 1000$

**859. Buddy Strings**

**Easy**

15761000Add to ListShare

Given two strings `s` and `goal`, return `true` if you can swap two letters in `s` so the result is equal to `goal`, otherwise, return `false`.

Swapping letters is defined as taking two indices `i` and `j` (0-indexed) such that `i != j` and swapping the characters at `s[i]` and `s[j]`.

For example, swapping at indices `0` and `2` in `"abcd"` results in `"cbad"`.

**Example 1:**

**Input:** `s = "ab"`, `goal = "ba"`

**Output:** `true`

**Explanation:** You can swap `s[0] = 'a'` and `s[1] = 'b'` to get `"ba"`, which is equal to `goal`.

**Example 2:**

**Input:** `s = "ab"`, `goal = "ab"`

**Output:** `false`

**Explanation:** The only letters you can swap are `s[0] = 'a'` and `s[1] = 'b'`, which results in `"ba" != goal`.

**Example 3:**

**Input:** `s = "aa"`, `goal = "aa"`

**Output:** `true`

**Explanation:** You can swap `s[0] = 'a'` and `s[1] = 'a'` to get `"aa"`, which is equal to `goal`.

**Constraints:**

`1 <= s.length, goal.length <= 2 * 104`

`s` and `goal` consist of lowercase letters.

**860. Lemonade Change****Easy**

1446132Add to ListShare

At a lemonade stand, each lemonade costs \$5. Customers are standing in a queue to buy from you and order one at a time (in the order specified by bills). Each customer will only buy one lemonade and pay with either a \$5, \$10, or \$20 bill. You must provide the correct change to each customer so that the net transaction is that the customer pays \$5.

Note that you do not have any change in hand at first.

Given an integer array `bills` where `bills[i]` is the bill the  $i^{\text{th}}$  customer pays, return `true` if you can provide every customer with the correct change, or `false` otherwise.

### Example 1:

**Input:** `bills = [5,5,5,10,20]`

**Output:** `true`

**Explanation:**

From the first 3 customers, we collect three \$5 bills in order.

From the fourth customer, we collect a \$10 bill and give back a \$5.

From the fifth customer, we give a \$10 bill and a \$5 bill.

Since all customers got correct change, we output true.

### Example 2:

**Input:** `bills = [5,5,10,10,20]`

**Output:** `false`

**Explanation:**

From the first two customers in order, we collect two \$5 bills.

For the next two customers in order, we collect a \$10 bill and give back a \$5 bill.

For the last customer, we can not give the change of \$15 back because we only have two \$10 bills.

Since not every customer received the correct change, the answer is false.

### Constraints:

`1 <= bills.length <= 105`

`bills[i]` is either 5, 10, or 20.

## 861. Score After Flipping Matrix

Medium

1219166Add to ListShare

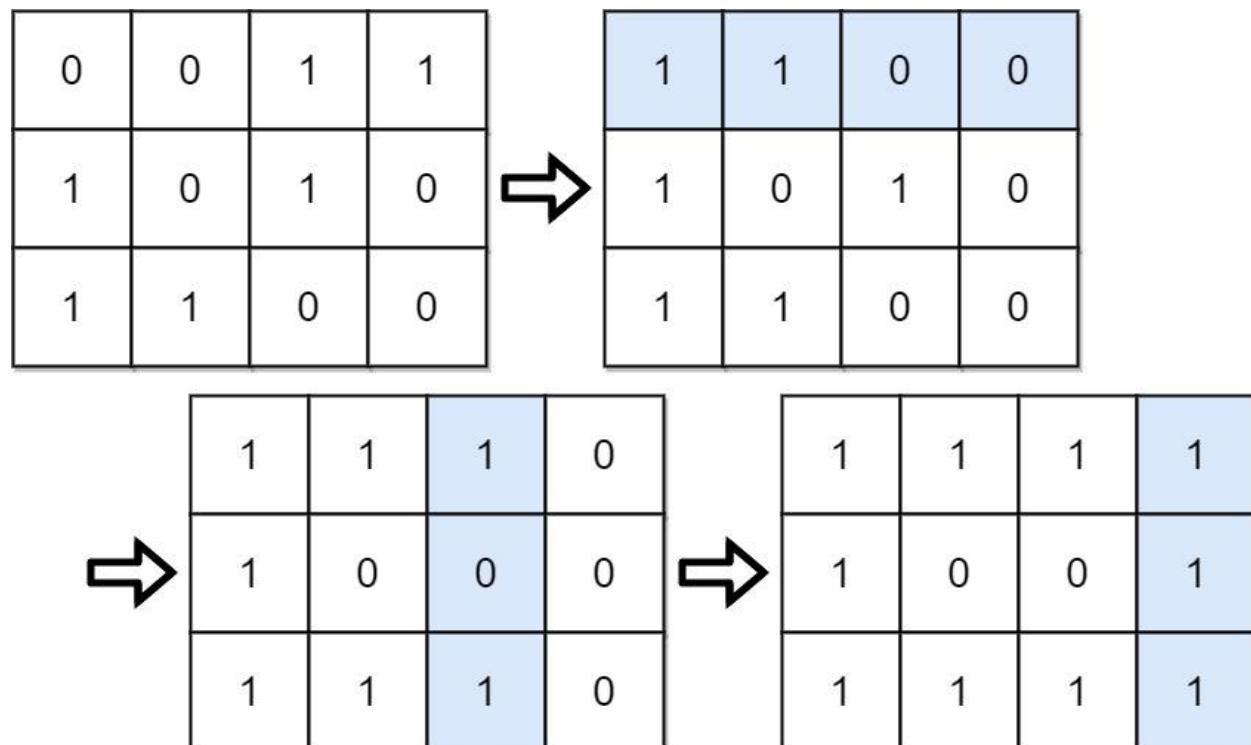
You are given an  $m \times n$  binary matrix `grid`.

A **move** consists of choosing any row or column and toggling each value in that row or column (i.e., changing all 0's to 1's, and all 1's to 0's).

Every row of the matrix is interpreted as a binary number, and the **score** of the matrix is the sum of these numbers.

Return *the highest possible score* after making any number of **moves** (including zero moves).

**Example 1:**



**Input:** `grid = [[0,0,1,1],[1,0,1,0],[1,1,0,0]]`

**Output:** 39

**Explanation:**  $0b1111 + 0b1001 + 0b1111 = 15 + 9 + 15 = 39$

**Example 2:**

**Input:** grid = [[0]]

**Output:** 1

**Constraints:**

`m == grid.length`

`n == grid[i].length`

`1 <= m, n <= 20`

`grid[i][j]` is either 0 or 1.

## 862. Shortest Subarray with Sum at Least K

Hard

336988Add to ListShare

Given an integer array `nums` and an integer `k`, return *the length of the shortest non-empty **subarray** of `nums` with a sum of at least `k`*. If there is no such **subarray**, return `-1`.

A **subarray** is a **contiguous** part of an array.

**Example 1:**

**Input:** `nums = [1], k = 1`

**Output:** 1

**Example 2:**

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

**Output:** -1

**Example 3:**

**Input:** `nums = [2,-1,2], k = 3`

**Output:** 3

**Constraints:**

`1 <= nums.length <= 105`

```
-105 <= nums[i] <= 105
```

```
1 <= k <= 109
```

## 863. All Nodes Distance K in Binary Tree

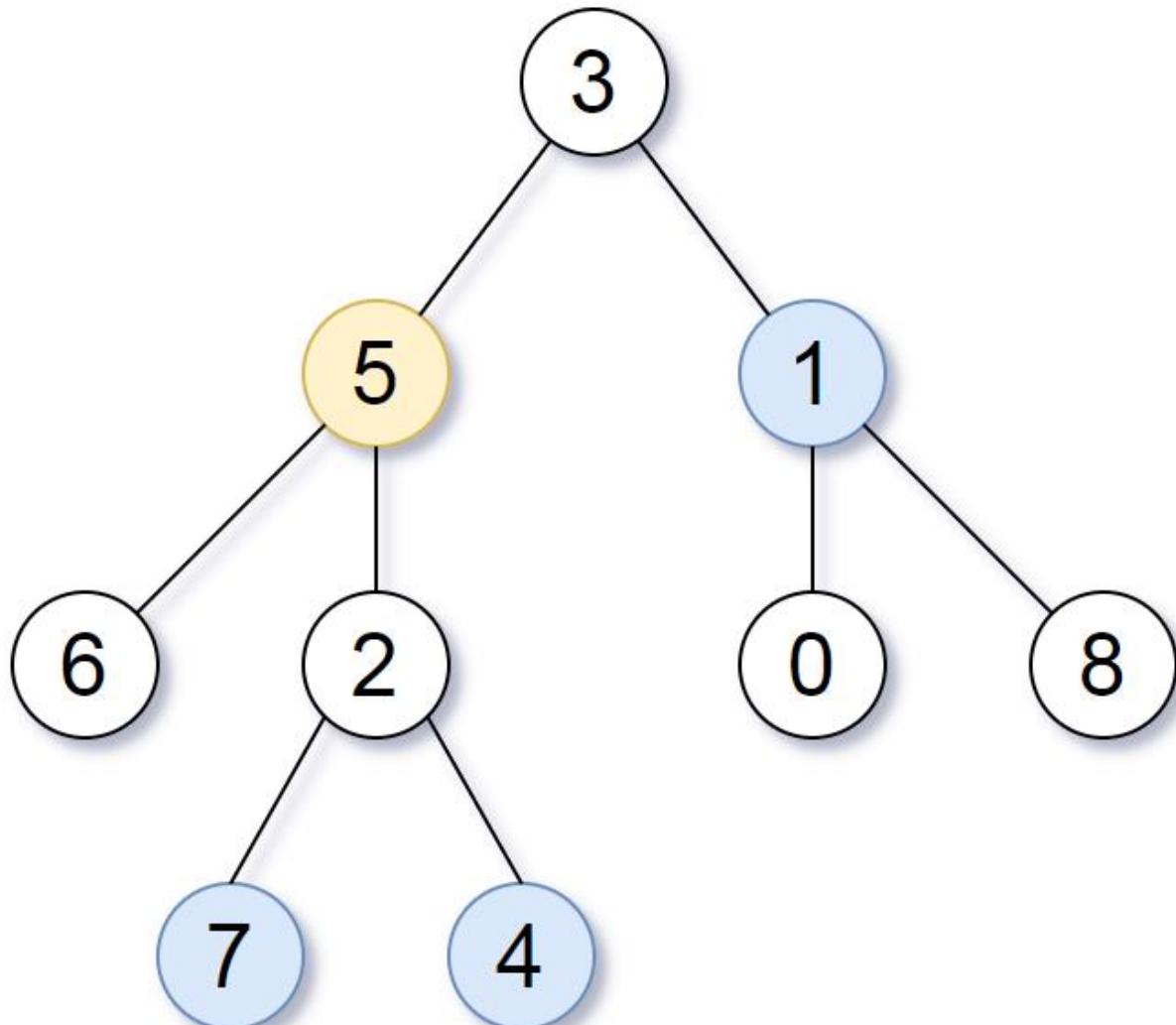
Medium

7432146Add to ListShare

Given the `root` of a binary tree, the value of a target node `target`, and an integer `k`, return *an array of the values of all nodes that have a distance `k` from the target node*.

You can return the answer in **any order**.

**Example 1:**



```
Input: root = [3,5,1,6,2,0,8,null,null,7,4], target = 5, k = 2
```

**Output:** [7,4,1]

Explanation: The nodes that are a distance 2 from the target node (with value 5) have values 7, 4, and 1.

**Example 2:**

**Input:** root = [1], target = 1, k = 3

**Output:** []

**Constraints:**

The number of nodes in the tree is in the range [1, 500].

0 <= Node.val <= 500

All the values `Node.val` are **unique**.

`target` is the value of one of the nodes in the tree.

0 <= k <= 1000

## 864. Shortest Path to Get All Keys

Hard

87234Add to ListShare

You are given an `m x n` grid `grid` where:

'.' is an empty cell.

'#' is a wall.

'@' is the starting point.

Lowercase letters represent keys.

Uppercase letters represent locks.

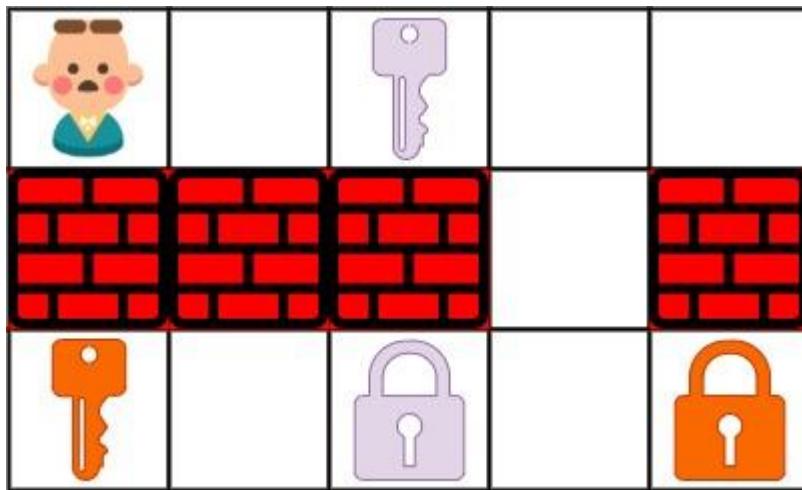
You start at the starting point and one move consists of walking one space in one of the four cardinal directions. You cannot walk outside the grid, or walk into a wall.

If you walk over a key, you can pick it up and you cannot walk over a lock unless you have its corresponding key.

For some  $1 \leq k \leq 6$ , there is exactly one lowercase and one uppercase letter of the first  $k$  letters of the English alphabet in the grid. This means that there is exactly one key for each lock, and one lock for each key; and also that the letters used to represent the keys and locks were chosen in the same order as the English alphabet.

Return *the lowest number of moves to acquire all keys*. If it is impossible, return  $-1$ .

**Example 1:**

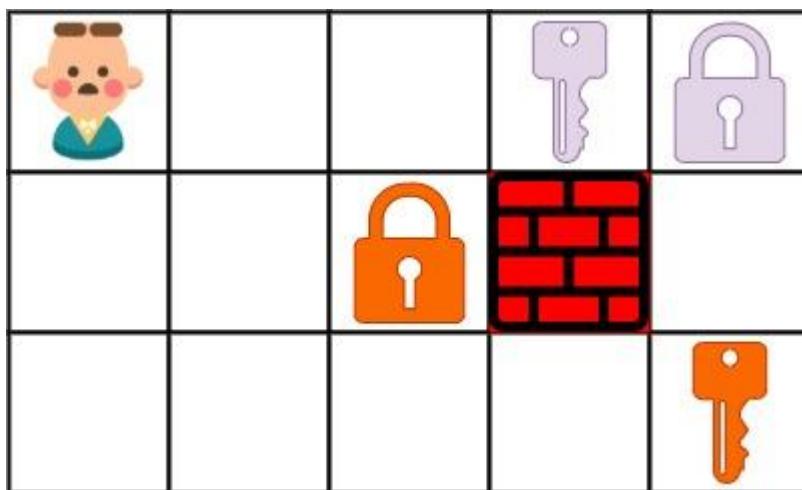


**Input:** grid = ["@.a..", "###.#", "b.A.B"]

**Output:** 8

**Explanation:** Note that the goal is to obtain all the keys not to open all the locks.

**Example 2:**



**Input:** grid = ["@..aA", "...B#.", "....b"]

**Output:** 6

**Example 3:**

**Input:** grid = ["@Aa"]

**Output:** -1

**Constraints:**

`m == grid.length`

`n == grid[i].length`

`1 <= m, n <= 30`

`grid[i][j]` is either an English letter, '.', '#', or '@'.

The number of keys in the grid is in the range `[1, 6]`.

Each key in the grid is **unique**.

Each key in the grid has a matching lock.

## 865. Smallest Subtree with all the Deepest Nodes

Medium

2175338Add to ListShare

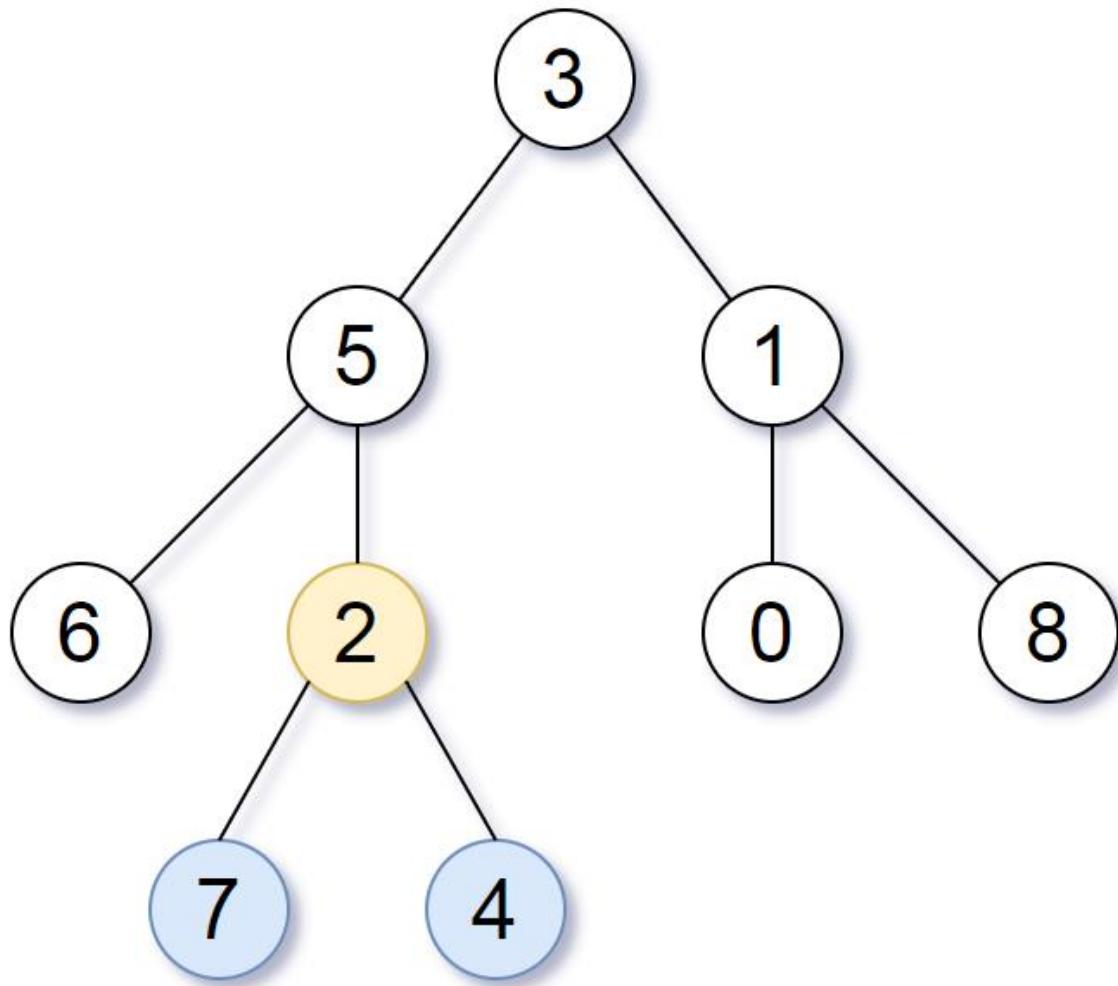
Given the `root` of a binary tree, the depth of each node is **the shortest distance to the root**.

Return *the smallest subtree* such that it contains **all the deepest nodes** in the original tree.

A node is called **the deepest** if it has the largest depth possible among any node in the entire tree.

The **subtree** of a node is a tree consisting of that node, plus the set of all descendants of that node.

**Example 1:**



**Input:** root = [3,5,1,6,2,0,8,null,null,7,4]

**Output:** [2,7,4]

**Explanation:** We return the node with value 2, colored in yellow in the diagram.

The nodes coloured in blue are the deepest nodes of the tree.

Notice that nodes 5, 3 and 2 contain the deepest nodes in the tree but node 2 is the smallest subtree among them, so we return it.

### Example 2:

**Input:** root = [1]

**Output:** [1]

**Explanation:** The root is the deepest node in the tree.

### Example 3:

**Input:** root = [0,1,3,null,2]

**Output:** [2]

**Explanation:** The deepest node in the tree is 2, the valid subtrees are the subtrees of nodes 2, 1 and 0 but the subtree of node 2 is the smallest.

### Constraints:

The number of nodes in the tree will be in the range [1, 500].

0 <= Node.val <= 500

The values of the nodes in the tree are **unique**.

## 866. Prime Palindrome

Medium

343739Add to ListShare

Given an integer n, return *the smallest prime palindrome greater than or equal to n*.

An integer is **prime** if it has exactly two divisors: 1 and itself. Note that 1 is not a prime number.

For example, 2, 3, 5, 7, 11, and 13 are all primes.

An integer is a **palindrome** if it reads the same from left to right as it does from right to left.

For example, 101 and 12321 are palindromes.

The test cases are generated so that the answer always exists and is in the range [2, 2 \* 10<sup>8</sup>].

### Example 1:

**Input:** n = 6

**Output:** 7

### Example 2:

**Input:** n = 8

**Output:** 11

### Example 3:

**Input:** n = 13

**Output:** 101

**Constraints:**

1 <= n <= 10<sup>8</sup>

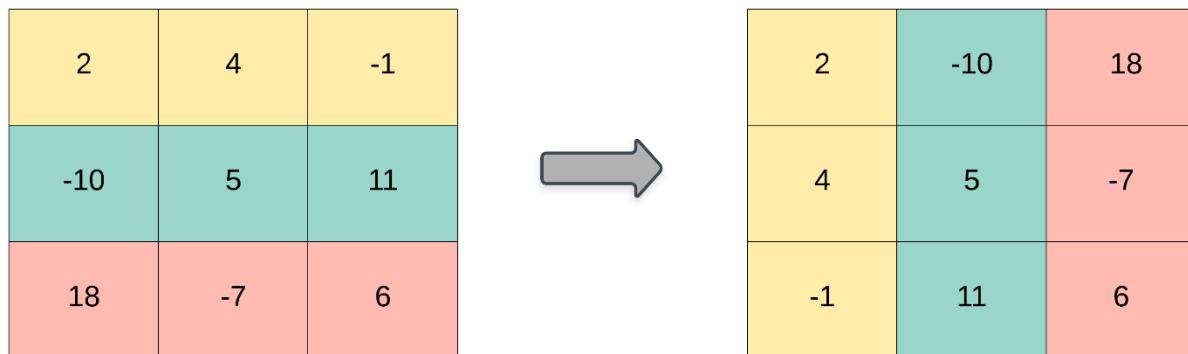
## 867. Transpose Matrix

Easy

2355410Add to ListShare

Given a 2D integer array `matrix`, return *the transpose of matrix*.

The **transpose** of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.



**Example 1:**

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

**Output:** [[1,4,7],[2,5,8],[3,6,9]]

**Example 2:**

**Input:** matrix = [[1,2,3],[4,5,6]]

**Output:** [[1,4],[2,5],[3,6]]

**Constraints:**

```

m == matrix.length

n == matrix[i].length

1 <= m, n <= 1000

1 <= m * n <= 105

-109 <= matrix[i][j] <= 109

```

## 868. Binary Gap

Easy

475610Add to ListShare

Given a positive integer  $n$ , find and return the **longest distance** between any two **adjacent** 1's in the binary representation of  $n$ . If there are no two adjacent 1's, return 0.

Two 1's are **adjacent** if there are only 0's separating them (possibly no 0's). The **distance** between two 1's is the absolute difference between their bit positions. For example, the two 1's in "1001" have a distance of 3.

### Example 1:

**Input:**  $n = 22$

**Output:** 2

**Explanation:** 22 in binary is "10110".

The first adjacent pair of 1's is "10110" with a distance of 2.

The second adjacent pair of 1's is "10110" with a distance of 1.

The answer is the largest of these two distances, which is 2.

Note that "10110" is not a valid pair since there is a 1 separating the two 1's underlined.

### Example 2:

**Input:**  $n = 8$

**Output:** 0

**Explanation:** 8 in binary is "1000".

There are not any adjacent pairs of 1's in the binary representation of 8, so we return 0.

**Example 3:****Input:** n = 5**Output:** 2**Explanation:** 5 in binary is "101".**Constraints:** $1 \leq n \leq 10^9$ **869. Reordered Power of 2****Medium**

1902376Add to ListShare

You are given an integer `n`. We reorder the digits in any order (including the original order) such that the leading digit is not zero.

Return `true` if and only if we can do this so that the resulting number is a power of two.

**Example 1:****Input:** n = 1**Output:** true**Example 2:****Input:** n = 10**Output:** false**Constraints:** $1 \leq n \leq 10^9$ **870. Advantage Shuffle****Medium**

134784Add to ListShare

You are given two integer arrays `nums1` and `nums2` both of the same length.

The **advantage** of `nums1` with respect to `nums2` is the number of indices `i` for which `nums1[i] > nums2[i]`.

Return *any permutation of* `nums1` *that maximizes its **advantage** with respect to* `nums2`.

**Example 1:**

**Input:** `nums1 = [2,7,11,15]`, `nums2 = [1,10,4,11]`

**Output:** `[2,11,7,15]`

**Example 2:**

**Input:** `nums1 = [12,24,8,32]`, `nums2 = [13,25,32,11]`

**Output:** `[24,32,8,12]`

**Constraints:**

```
1 <= nums1.length <= 105
nums2.length == nums1.length
0 <= nums1[i], nums2[i] <= 109
```

## 871. Minimum Number of Refueling Stops

Hard

406176Add to ListShare

A car travels from a starting position to a destination which is `target` miles east of the starting position.

There are gas stations along the way. The gas stations are represented as an array `stations` where `stations[i] = [positioni, fueli]` indicates that the *i<sup>th</sup>* gas station is `positioni` miles east of the starting position and has `fueli` liters of gas.

The car starts with an infinite tank of gas, which initially has `startFuel` liters of fuel in it. It uses one liter of gas per one mile that it drives. When the car reaches a gas station, it may stop and refuel, transferring all the gas from the station into the car.

Return *the minimum number of refueling stops the car must make in order to reach its destination*. If it cannot reach the destination, return `-1`.

Note that if the car reaches a gas station with `0` fuel left, the car can still refuel there. If the car reaches the destination with `0` fuel left, it is still considered to have arrived.

**Example 1:**

**Input:** target = 1, startFuel = 1, stations = []

**Output:** 0

**Explanation:** We can reach the target without refueling.

**Example 2:**

**Input:** target = 100, startFuel = 1, stations = [[10,100]]

**Output:** -1

**Explanation:** We can not reach the target (or even the first gas station).

**Example 3:**

**Input:** target = 100, startFuel = 10, stations = [[10,60],[20,30],[30,30],[60,40]]

**Output:** 2

**Explanation:** We start with 10 liters of fuel.

We drive to position 10, expending 10 liters of fuel. We refuel from 0 liters to 60 liters of gas.

Then, we drive from position 10 to position 60 (expending 50 liters of fuel),

and refuel from 10 liters to 50 liters of gas. We then drive to and reach the target.

We made 2 refueling stops along the way, so we return 2.

**Constraints:**

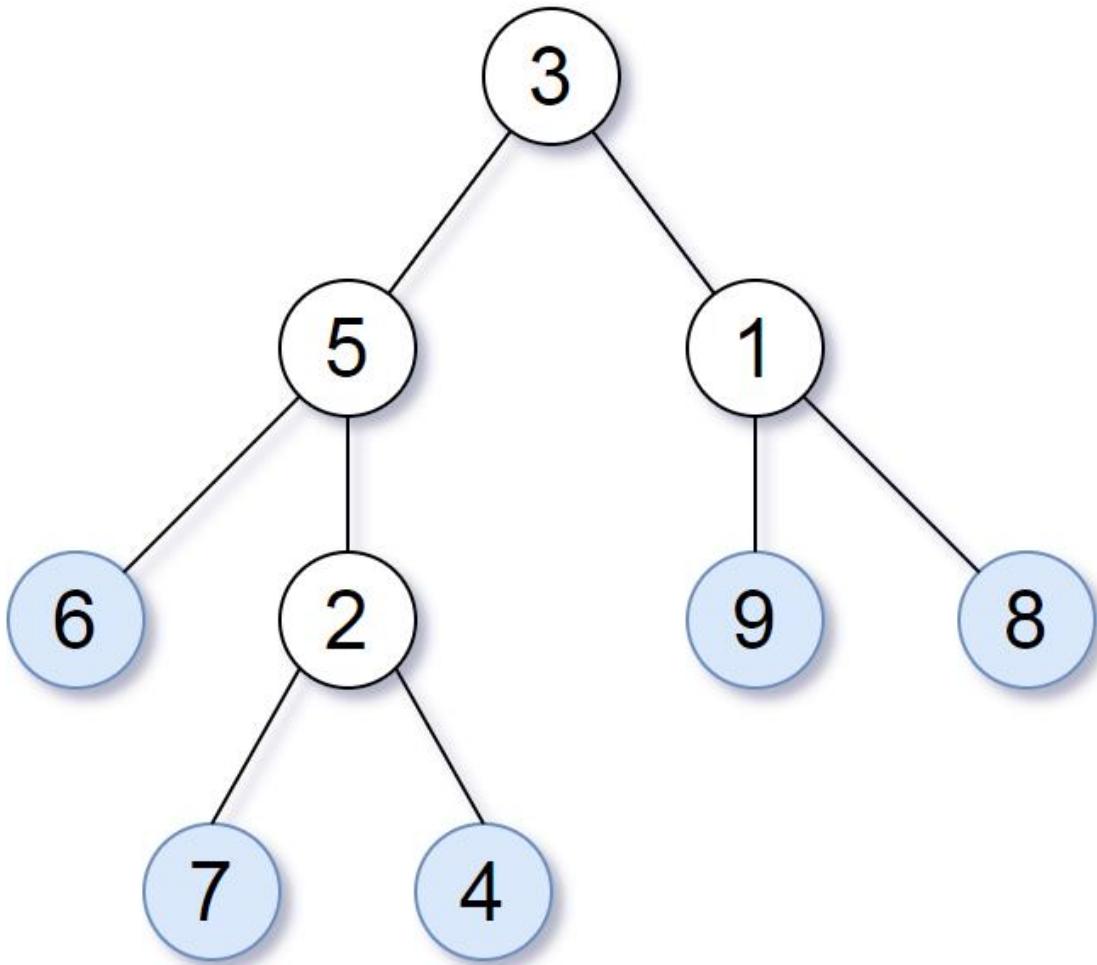
```
1 <= target, startFuel <= 109
0 <= stations.length <= 500
1 <= positioni < positioni+1 < target
1 <= fueli < 109
```

**872. Leaf-Similar Trees**

Easy

189856Add to ListShare

Consider all the leaves of a binary tree, from left to right order, the values of those leaves form a **leaf value sequence**.

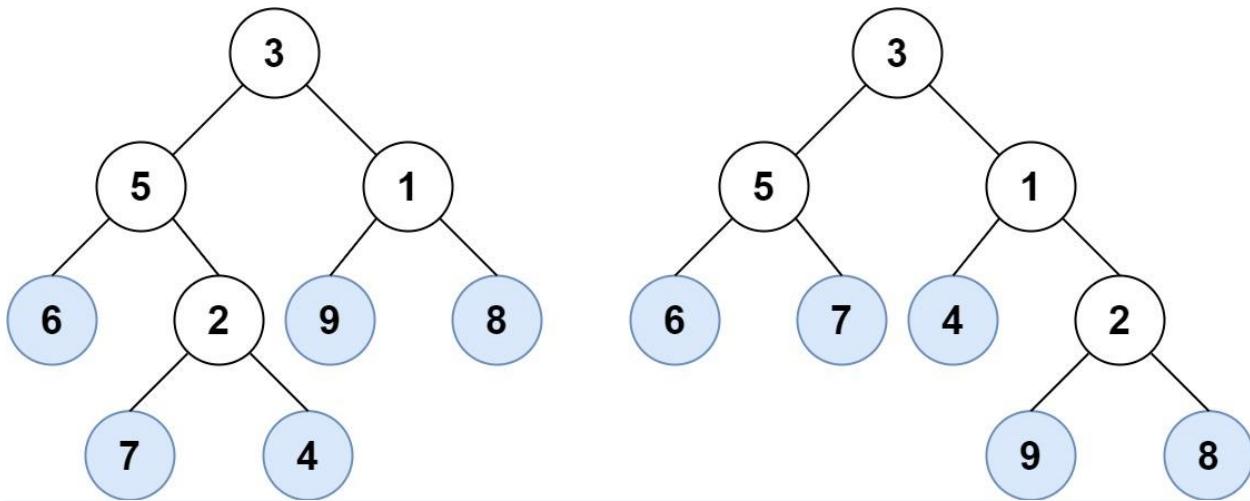


For example, in the given tree above, the leaf value sequence is `(6, 7, 4, 9, 8)`.

Two binary trees are considered *leaf-similar* if their leaf value sequence is the same.

Return `true` if and only if the two given trees with head nodes `root1` and `root2` are leaf-similar.

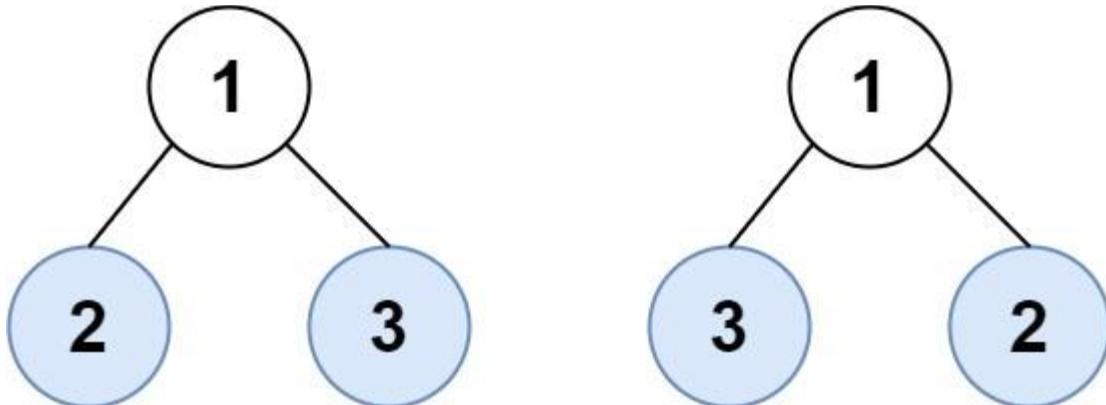
**Example 1:**



**Input:** root1 = [3,5,1,6,2,9,8,null,null,7,4], root2 = [3,5,1,6,7,4,2,null,null,null,null,null,9,8]

**Output:** true

**Example 2:**



**Input:** root1 = [1,2,3], root2 = [1,3,2]

**Output:** false

**Constraints:**

The number of nodes in each tree will be in the range [1, 200].

Both of the given trees will have values in the range [0, 200].

## 873. Length of Longest Fibonacci Subsequence

Medium

163658Add to ListShare

A sequence  $x_1, x_2, \dots, x_n$  is *Fibonacci-like* if:

```
n >= 3
```

```
xi + xi+1 == xi+2 for all i + 2 <= n
```

Given a **strictly increasing** array `arr` of positive integers forming a sequence, return *the length of the longest Fibonacci-like subsequence of arr*. If one does not exist, return `0`.

A **subsequence** is derived from another sequence `arr` by deleting any number of elements (including none) from `arr`, without changing the order of the remaining elements. For example, `[3, 5, 8]` is a subsequence of `[3, 4, 5, 6, 7, 8]`.

### Example 1:

**Input:** `arr = [1,2,3,4,5,6,7,8]`

**Output:** `5`

**Explanation:** The longest subsequence that is fibonacci-like: `[1,2,3,5,8]`.

### Example 2:

**Input:** `arr = [1,3,7,11,12,14,18]`

**Output:** `3`

**Explanation:** The longest subsequence that is fibonacci-like: `[1,11,12]`, `[3,11,14]` or `[7,11,18]`.

### Constraints:

```
3 <= arr.length <= 1000
```

```
1 <= arr[i] < arr[i + 1] <= 109
```

## 874. Walking Robot Simulation

Medium

11618Add to ListShare

A robot on an infinite XY-plane starts at point `(0, 0)` facing north. The robot can receive a sequence of these three possible types of `commands`:

`-2`: Turn left 90 degrees.

`-1`: Turn right 90 degrees.

`1 <= k <= 9`: Move forward `k` units, one unit at a time.

Some of the grid squares are `obstacles`. The  $i^{\text{th}}$  obstacle is at grid point `obstacles[i] = (xi, yi)`. If the robot runs into an obstacle, then it will instead stay in its current location and move on to the next command.

Return *the maximum Euclidean distance that the robot ever gets from the origin squared* (i.e. if the distance is 5, return 25).

**Note:**

North means +Y direction.

East means +X direction.

South means -Y direction.

West means -X direction.

**Example 1:**

**Input:** `commands = [4, -1, 3]`, `obstacles = []`

**Output:** 25

**Explanation:** The robot starts at  $(0, 0)$ :

1. Move north 4 units to  $(0, 4)$ .
2. Turn right.
3. Move east 3 units to  $(3, 4)$ .

The furthest point the robot ever gets from the origin is  $(3, 4)$ , which squared is  $3^2 + 4^2 = 25$  units away.

**Example 2:**

**Input:** `commands = [4, -1, 4, -2, 4]`, `obstacles = [[2, 4]]`

**Output:** 65

**Explanation:** The robot starts at  $(0, 0)$ :

1. Move north 4 units to  $(0, 4)$ .
2. Turn right.

3. Move east 1 unit and get blocked by the obstacle at (2, 4), robot is at (1, 4).
4. Turn left.
5. Move north 4 units to (1, 8).

The furthest point the robot ever gets from the origin is (1, 8), which squared is  $1^2 + 8^2 = 65$  units away.

**Example 3:**

**Input:** commands = [6, -1, -1, 6], obstacles = []

**Output:** 36

**Explanation:** The robot starts at (0, 0):

1. Move north 6 units to (0, 6).
2. Turn right.
3. Turn right.
4. Move south 6 units to (0, 0).

The furthest point the robot ever gets from the origin is (0, 6), which squared is  $6^2 = 36$  units away.

**Constraints:**

`1 <= commands.length <= 104`

`commands[i]` is either `-2`, `-1`, or an integer in the range `[1, 9]`.

`0 <= obstacles.length <= 104`

`-3 * 104 <= xi, yi <= 3 * 104`

The answer is guaranteed to be less than `231`.

## 875. Koko Eating Bananas

Medium

5151237 Add to List Share

Koko loves to eat bananas. There are `n` piles of bananas, the `ith` pile has `piles[i]` bananas. The guards have gone and will come back in `h` hours.

Koko can decide her bananas-per-hour eating speed of  $k$ . Each hour, she chooses some pile of bananas and eats  $k$  bananas from that pile. If the pile has less than  $k$  bananas, she eats all of them instead and will not eat any more bananas during this hour.

Koko likes to eat slowly but still wants to finish eating all the bananas before the guards return.

Return *the minimum integer  $k$  such that she can eat all the bananas within  $h$  hours.*

**Example 1:**

**Input:** piles = [3,6,7,11], h = 8

**Output:** 4

**Example 2:**

**Input:** piles = [30,11,23,4,20], h = 5

**Output:** 30

**Example 3:**

**Input:** piles = [30,11,23,4,20], h = 6

**Output:** 23

**Constraints:**

$1 \leq \text{piles.length} \leq 10^4$

$\text{piles.length} \leq h \leq 10^9$

$1 \leq \text{piles}[i] \leq 10^9$

## 876. Middle of the Linked List

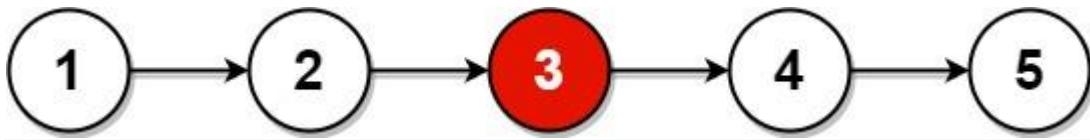
Easy

7076189Add to ListShare

Given the `head` of a singly linked list, return *the middle node of the linked list.*

If there are two middle nodes, return **the second middle** node.

**Example 1:**

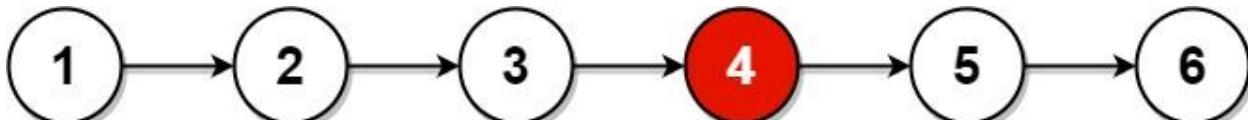


**Input:** head = [1,2,3,4,5]

**Output:** [3,4,5]

**Explanation:** The middle node of the list is node 3.

**Example 2:**



**Input:** head = [1,2,3,4,5,6]

**Output:** [4,5,6]

**Explanation:** Since the list has two middle nodes with values 3 and 4, we return the second one.

**Constraints:**

The number of nodes in the list is in the range [1, 100].

1 <= Node.val <= 100

## 877. Stone Game

Medium

22902255Add to ListShare

Alice and Bob play a game with piles of stones. There are an **even** number of piles arranged in a row, and each pile has a **positive** integer number of stones `piles[i]`.

The objective of the game is to end with the most stones. The **total** number of stones across all the piles is **odd**, so there are no ties.

Alice and Bob take turns, with **Alice starting first**. Each turn, a player takes the entire pile of stones either from the **beginning** or from the **end** of the row. This continues until there are no more piles left, at which point the person with the **most stones wins**.

Assuming Alice and Bob play optimally, return `true` if Alice wins the game, or `false` if Bob wins.

**Example 1:**

**Input:** piles = [5,3,4,5]

**Output:** true

**Explanation:**

Alice starts first, and can only take the first 5 or the last 5.

Say she takes the first 5, so that the row becomes [3, 4, 5].

If Bob takes 3, then the board is [4, 5], and Alice takes 5 to win with 10 points.

If Bob takes the last 5, then the board is [3, 4], and Alice takes 4 to win with 9 points.

This demonstrated that taking the first 5 was a winning move for Alice, so we return true.

**Example 2:**

**Input:** piles = [3,7,2,3]

**Output:** true

**Constraints:**

`2 <= piles.length <= 500`

`piles.length is even.`

`1 <= piles[i] <= 500`

`sum(piles[i]) is odd.`

## 878. Nth Magical Number

Hard

1054142Add to ListShare

A positive integer is *magical* if it is divisible by either `a` or `b`.

Given the three integers `n`, `a`, and `b`, return the `nth` magical number. Since the answer may be very large, **return it modulo  $10^9 + 7$** .

**Example 1:**

**Input:** `n = 1, a = 2, b = 3`

**Output:** 2

**Example 2:**

**Input:** n = 4, a = 2, b = 3

**Output:** 6

**Constraints:**

1 <= n <=  $10^9$

2 <= a, b <= 4 \*  $10^4$

## 879. Profitable Schemes

**Hard**

53444Add to ListShare

There is a group of n members, and a list of various crimes they could commit. The  $i^{\text{th}}$  crime generates a profit[i] and requires group[i] members to participate in it. If a member participates in one crime, that member can't participate in another crime.

Let's call a **profitable scheme** any subset of these crimes that generates at least minProfit profit, and the total number of members participating in that subset of crimes is at most n.

Return the number of schemes that can be chosen. Since the answer may be very large, **return it modulo  $10^9 + 7$** .

**Example 1:**

**Input:** n = 5, minProfit = 3, group = [2,2], profit = [2,3]

**Output:** 2

**Explanation:** To make a profit of at least 3, the group could either commit crimes 0 and 1, or just crime 1.

In total, there are 2 schemes.

**Example 2:**

**Input:** n = 10, minProfit = 5, group = [2,3,5], profit = [6,7,8]

**Output:** 7

**Explanation:** To make a profit of at least 5, the group could commit any crimes, as long as they commit one.

There are 7 possible schemes: (0), (1), (2), (0,1), (0,2), (1,2), and (0,1,2).

### Constraints:

```
1 <= n <= 100
0 <= minProfit <= 100
1 <= group.length <= 100
1 <= group[i] <= 100
profit.length == group.length
0 <= profit[i] <= 100
```

## 880. Decoded String at Index

Medium

1246196Add to ListShare

You are given an encoded string `s`. To decode the string to a tape, the encoded string is read one character at a time and the following steps are taken:

If the character read is a letter, that letter is written onto the tape.

If the character read is a digit `d`, the entire current tape is repeatedly written `d - 1` more times in total.

Given an integer `k`, return *the  $k^{\text{th}}$  letter (1-indexed) in the decoded string*.

### Example 1:

**Input:** `s = "leet2code3", k = 10`

**Output:** "o"

**Explanation:** The decoded string is "leetleetcodeleetleetcodeleetleetcode".

The  $10^{\text{th}}$  letter in the string is "o".

### Example 2:

**Input:** `s = "ha22", k = 5`

**Output:** "h"

**Explanation:** The decoded string is "hahahaha".

The 5<sup>th</sup> letter is "h".

**Example 3:**

**Input:** s = "a2345678999999999999999", k = 1

**Output:** "a"

**Explanation:** The decoded string is "a" repeated 8301530446056247680 times.

The 1<sup>st</sup> letter is "a".

**Constraints:**

`2 <= s.length <= 100`

`s` consists of lowercase English letters and digits `2` through `9`.

`s` starts with a letter.

`1 <= k <= 109`

It is guaranteed that `k` is less than or equal to the length of the decoded string.

The decoded string is guaranteed to have less than `263` letters.

## 881. Boats to Save People

Medium

334989Add to ListShare

You are given an array `people` where `people[i]` is the weight of the `ith` person, and an **infinite number of boats** where each boat can carry a maximum weight of `limit`. Each boat carries at most two people at the same time, provided the sum of the weight of those people is at most `limit`.

Return *the minimum number of boats to carry every given person*.

**Example 1:**

**Input:** `people = [1,2]`, `limit = 3`

**Output:** 1

**Explanation:** 1 boat (1, 2)

**Example 2:**

**Input:** people = [3,2,2,1], limit = 3

**Output:** 3

**Explanation:** 3 boats (1, 2), (2) and (3)

**Example 3:**

**Input:** people = [3,5,3,4], limit = 5

**Output:** 4

**Explanation:** 4 boats (3), (3), (4), (5)

**Constraints:**

1 <= people.length <= 5 \* 10<sup>4</sup>

1 <= people[i] <= limit <= 3 \* 10<sup>4</sup>

**882. Reachable Nodes In Subdivided Graph****Hard**

614208Add to ListShare

You are given an undirected graph (the "**original graph**") with  $n$  nodes labeled from 0 to  $n - 1$ . You decide to **subdivide** each edge in the graph into a chain of nodes, with the number of new nodes varying between each edge.

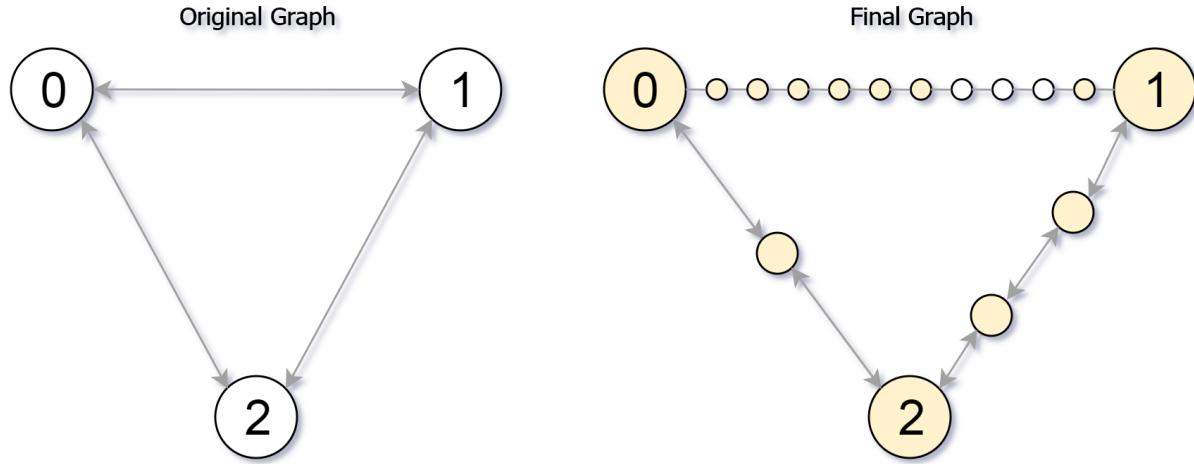
The graph is given as a 2D array of `edges` where `edges[i] = [ui, vi, cnti]` indicates that there is an edge between nodes  $u_i$  and  $v_i$  in the original graph, and  $cnt_i$  is the total number of new nodes that you will **subdivide** the edge into. Note that  $cnt_i == 0$  means you will not subdivide the edge.

To **subdivide** the edge  $[u_i, v_i]$ , replace it with  $(cnt_i + 1)$  new edges and  $cnt_i$  new nodes. The new nodes are  $x_1, x_2, \dots, x_{cnt_i}$ , and the new edges are  $[u_i, x_1], [x_1, x_2], [x_2, x_3], \dots, [x_{cnt_i-1}, x_{cnt_i}], [x_{cnt_i}, v_i]$ .

In this **new graph**, you want to know how many nodes are **reachable** from the node 0, where a node is **reachable** if the distance is `maxMoves` or less.

Given the original graph and `maxMoves`, return *the number of nodes that are **reachable** from node 0 in the new graph*.

**Example 1:**



**Input:** edges = [[0,1,10],[0,2,1],[1,2,2]], maxMoves = 6, n = 3

**Output:** 13

**Explanation:** The edge subdivisions are shown in the image above.

The nodes that are reachable are highlighted in yellow.

**Example 2:**

**Input:** edges = [[0,1,4],[1,2,6],[0,2,8],[1,3,1]], maxMoves = 10, n = 4

**Output:** 23

**Example 3:**

**Input:** edges = [[1,2,4],[1,4,5],[1,3,1],[2,3,4],[3,4,5]], maxMoves = 17, n = 5

**Output:** 1

**Explanation:** Node 0 is disconnected from the rest of the graph, so only node 0 is reachable.

**Constraints:**

```
0 <= edges.length <= min(n * (n - 1) / 2, 104)
```

```
edges[i].length == 3
```

```
0 <= ui < vi < n
```

There are **no multiple edges** in the graph.

```
0 <= cnti <= 104
```

```
0 <= maxMoves <= 109
```

```
1 <= n <= 3000
```

## 883. Projection Area of 3D Shapes

Easy

4441218Add to ListShare

You are given an  $n \times n$  grid where we place some  $1 \times 1 \times 1$  cubes that are axis-aligned with the  $x$ ,  $y$ , and  $z$  axes.

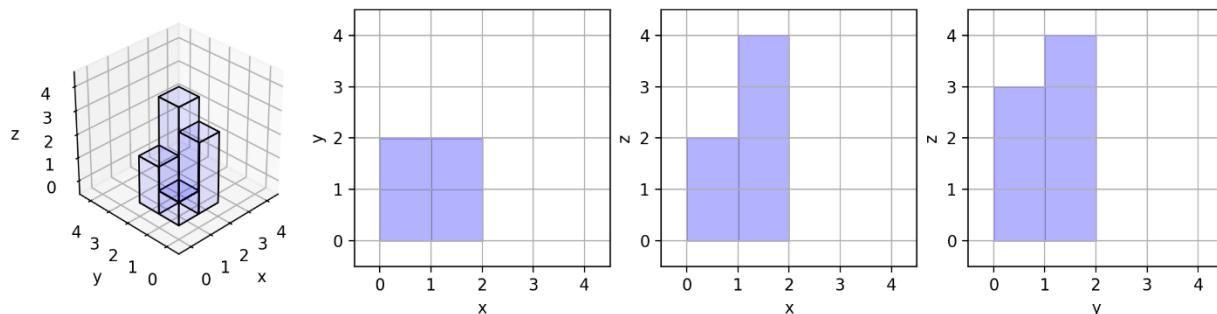
Each value  $v = \text{grid}[i][j]$  represents a tower of  $v$  cubes placed on top of the cell  $(i, j)$ .

We view the projection of these cubes onto the  $xy$ ,  $yz$ , and  $zx$  planes.

A **projection** is like a shadow, that maps our **3-dimensional** figure to a **2-dimensional** plane. We are viewing the "shadow" when looking at the cubes from the top, the front, and the side.

Return *the total area of all three projections*.

### Example 1:



**Input:**  $\text{grid} = [[1,2],[3,4]]$

**Output:** 17

**Explanation:** Here are the three projections ("shadows") of the shape made with each axis-aligned plane.

### Example 2:

**Input:**  $\text{grid} = [[2]]$

**Output:** 5

### Example 3:

**Input:**  $\text{grid} = [[1,0],[0,2]]$

**Output:** 8

**Constraints:**

```
n == grid.length == grid[i].length
1 <= n <= 50
0 <= grid[i][j] <= 50
```

## 884. Uncommon Words from Two Sentences

Easy

1026148Add to ListShare

A **sentence** is a string of single-space separated words where each word consists only of lowercase letters.

A word is **uncommon** if it appears exactly once in one of the sentences, and **does not appear** in the other sentence.

Given two **sentences** `s1` and `s2`, return *a list of all the uncommon words*. You may return the answer in **any order**.

**Example 1:**

**Input:** `s1 = "this apple is sweet", s2 = "this apple is sour"`

**Output:** `["sweet", "sour"]`

**Example 2:**

**Input:** `s1 = "apple apple", s2 = "banana"`

**Output:** `["banana"]`

**Constraints:**

```
1 <= s1.length, s2.length <= 200
```

`s1` and `s2` consist of lowercase English letters and spaces.

`s1` and `s2` do not have leading or trailing spaces.

All the words in `s1` and `s2` are separated by a single space.

## 885. Spiral Matrix III

Medium

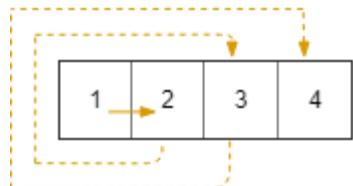
616657Add to ListShare

You start at the cell `(rStart, cStart)` of an `rows x cols` grid facing east. The northwest corner is at the first row and column in the grid, and the southeast corner is at the last row and column.

You will walk in a clockwise spiral shape to visit every position in this grid. Whenever you move outside the grid's boundary, we continue our walk outside the grid (but may return to the grid boundary later.). Eventually, we reach all `rows * cols` spaces of the grid.

Return *an array of coordinates representing the positions of the grid in the order you visited them.*

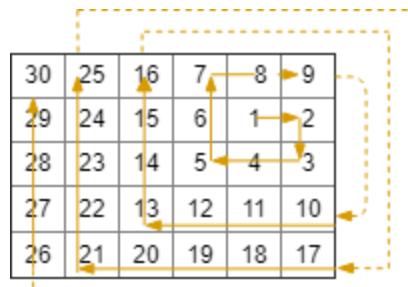
### Example 1:



**Input:** `rows = 1, cols = 4, rStart = 0, cStart = 0`

**Output:** `[[0,0],[0,1],[0,2],[0,3]]`

### Example 2:



**Input:** `rows = 5, cols = 6, rStart = 1, cStart = 4`

**Output:**

```
[[1,4],[1,5],[2,5],[2,4],[2,3],[1,3],[0,3],[0,4],[0,5],[3,5],[3,4],[3,3],[3,2],[2,2],[1,2],[0,2],[4,5],[4,4],[4,3],[4,2],[4,1],[3,1],[2,1],[1,1],[0,1],[4,0],[3,0],[2,0],[1,0],[0,0]]
```

### Constraints:

```
1 <= rows, cols <= 100
0 <= rStart < rows
0 <= cStart < cols
```

## 886. Possible Bipartition

Medium

272160Add to ListShare

We want to split a group of  $n$  people (labeled from 1 to  $n$ ) into two groups of **any size**. Each person may dislike some other people, and they should not go into the same group.

Given the integer  $n$  and the array `dislikes` where `dislikes[i] = [ai, bi]` indicates that the person labeled  $a_i$  does not like the person labeled  $b_i$ , return `true` if it is possible to split everyone into two groups in this way.

### Example 1:

**Input:**  $n = 4$ , `dislikes = [[1,2],[1,3],[2,4]]`

**Output:** `true`

**Explanation:** group1 [1,4] and group2 [2,3].

### Example 2:

**Input:**  $n = 3$ , `dislikes = [[1,2],[1,3],[2,3]]`

**Output:** `false`

### Example 3:

**Input:**  $n = 5$ , `dislikes = [[1,2],[2,3],[3,4],[4,5],[1,5]]`

**Output:** `false`

### Constraints:

```
1 <= n <= 2000
0 <= dislikes.length <= 104
dislikes[i].length == 2
1 <= dislikes[i][j] <= n
```

$a_i < b_i$

All the pairs of `dislikes` are **unique**.

## 887. Super Egg Drop

Hard

2906146Add to ListShare

You are given  $k$  identical eggs and you have access to a building with  $n$  floors labeled from  $1$  to  $n$ .

You know that there exists a floor  $f$  where  $0 \leq f \leq n$  such that any egg dropped at a floor **higher** than  $f$  will **break**, and any egg dropped **at or below** floor  $f$  will **not break**.

Each move, you may take an unbroken egg and drop it from any floor  $x$  (where  $1 \leq x \leq n$ ). If the egg breaks, you can no longer use it. However, if the egg does not break, you may **reuse** it in future moves.

Return *the minimum number of moves* that you need to determine **with certainty** what the value of  $f$  is.

### Example 1:

**Input:**  $k = 1, n = 2$

**Output:** 2

#### Explanation:

Drop the egg from floor 1. If it breaks, we know that  $f = 0$ .

Otherwise, drop the egg from floor 2. If it breaks, we know that  $f = 1$ .

If it does not break, then we know  $f = 2$ .

Hence, we need at minimum 2 moves to determine with certainty what the value of  $f$  is.

### Example 2:

**Input:**  $k = 2, n = 6$

**Output:** 3

### Example 3:

**Input:**  $k = 3, n = 14$

**Output:** 4

**Constraints:**

```
1 <= k <= 100
```

```
1 <= n <= 104
```

**888. Fair Candy Swap****Easy**

1513275Add to ListShare

Alice and Bob have a different total number of candies. You are given two integer arrays `aliceSizes` and `bobSizes` where `aliceSizes[i]` is the number of candies of the  $i^{\text{th}}$  box of candy that Alice has and `bobSizes[j]` is the number of candies of the  $j^{\text{th}}$  box of candy that Bob has.

Since they are friends, they would like to exchange one candy box each so that after the exchange, they both have the same total amount of candy. The total amount of candy a person has is the sum of the number of candies in each box they have.

Return an integer array `answer` where `answer[0]` is the number of candies in the box that Alice must exchange, and `answer[1]` is the number of candies in the box that Bob must exchange. If there are multiple answers, you may **return any** one of them. It is guaranteed that at least one answer exists.

**Example 1:**

**Input:** `aliceSizes` = [1,1], `bobSizes` = [2,2]

**Output:** [1,2]

**Example 2:**

**Input:** `aliceSizes` = [1,2], `bobSizes` = [2,3]

**Output:** [1,2]

**Example 3:**

**Input:** `aliceSizes` = [2], `bobSizes` = [1,3]

**Output:** [2,3]

**Constraints:**

```
1 <= aliceSizes.length, bobSizes.length <= 104
```

```
1 <= aliceSizes[i], bobSizes[j] <= 105
```

Alice and Bob have a different total number of candies.

There will be at least one valid answer for the given input.

## 889. Construct Binary Tree from Preorder and Postorder Traversals

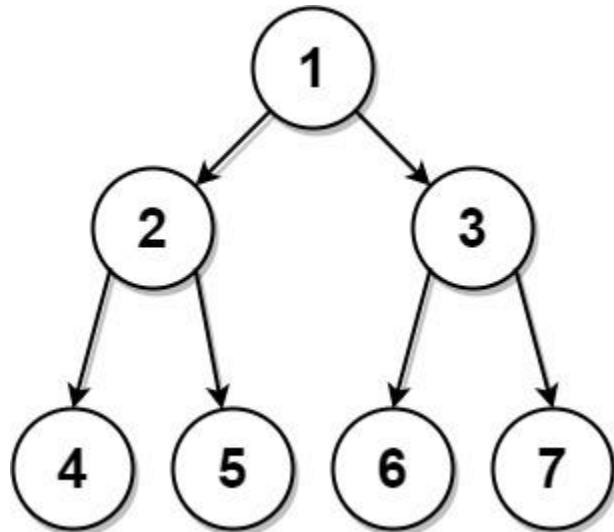
Medium

220094Add to ListShare

Given two integer arrays, `preorder` and `postorder` where `preorder` is the preorder traversal of a binary tree of **distinct** values and `postorder` is the postorder traversal of the same tree, reconstruct and return *the binary tree*.

If there exist multiple answers, you can **return any** of them.

**Example 1:**



**Input:** preorder = [1,2,4,5,3,6,7], postorder = [4,5,2,6,7,3,1]

**Output:** [1,2,3,4,5,6,7]

**Example 2:**

**Input:** preorder = [1], postorder = [1]

**Output:** [1]

**Constraints:**

1 <= `preorder.length` <= 30

```
1 <= preorder[i] <= preorder.length
```

All the values of `preorder` are **unique**.

```
postorder.length == preorder.length
```

```
1 <= postorder[i] <= postorder.length
```

All the values of `postorder` are **unique**.

It is guaranteed that `preorder` and `postorder` are the preorder traversal and postorder traversal of the same binary tree.

## 890. Find and Replace Pattern

Medium

3379153Add to ListShare

Given a list of strings `words` and a string `pattern`, return a list of `words[i]` that match `pattern`. You may return the answer in **any order**.

A word matches the pattern if there exists a permutation of letters `p` so that after replacing every letter `x` in the pattern with `p(x)`, we get the desired word.

Recall that a permutation of letters is a bijection from letters to letters: every letter maps to another letter, and no two letters map to the same letter.

### Example 1:

**Input:** `words` = `["abc", "deq", "mee", "aqq", "dkd", "ccc"]`, `pattern` = `"abb"`

**Output:** `["mee", "aqq"]`

**Explanation:** `"mee"` matches the pattern because there is a permutation `{a -> m, b -> e, ...}`.

`"ccc"` does not match the pattern because `{a -> c, b -> c, ...}` is not a permutation, since `a` and `b` map to the same letter.

### Example 2:

**Input:** `words` = `["a", "b", "c"]`, `pattern` = `"a"`

**Output:** `["a", "b", "c"]`

### Constraints:

```

1 <= pattern.length <= 20

1 <= words.length <= 50

words[i].length == pattern.length

pattern and words[i] are lowercase English letters.

```

## 891. Sum of Subsequence Widths

Hard

588155Add to ListShare

The **width** of a sequence is the difference between the maximum and minimum elements in the sequence.

Given an array of integers `nums`, return *the sum of the widths of all the non-empty subsequences of* `nums`. Since the answer may be very large, return it **modulo**  $10^9 + 7$ .

A **subsequence** is a sequence that can be derived from an array by deleting some or no elements without changing the order of the remaining elements. For example, `[3, 6, 2, 7]` is a subsequence of the array `[0, 3, 1, 6, 2, 2, 7]`.

### Example 1:

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

**Output:** 6

**Explanation:** The subsequences are `[1]`, `[2]`, `[3]`, `[2,1]`, `[2,3]`, `[1,3]`, `[2,1,3]`.

The corresponding widths are `0`, `0`, `0`, `1`, `1`, `2`, `2`.

The sum of these widths is 6.

### Example 2:

**Input:** `nums = [2]`

**Output:** 0

### Constraints:

```
1 <= nums.length <= 105
```

```
1 <= nums[i] <= 105
```

## 892. Surface Area of 3D Shapes

Easy

454637Add to ListShare

You are given an  $n \times n$  grid where you have placed some  $1 \times 1 \times 1$  cubes. Each value  $v = \text{grid}[i][j]$  represents a tower of  $v$  cubes placed on top of cell  $(i, j)$ .

After placing these cubes, you have decided to glue any directly adjacent cubes to each other, forming several irregular 3D shapes.

Return *the total surface area of the resulting shapes*.

**Note:** The bottom face of each shape counts toward its surface area.

**Example 1:**

1	2
3	4

**Input:** `grid = [[1,2],[3,4]]`

**Output:** 34

**Example 2:**

1	1	1
1	0	1
1	1	1

**Input:** `grid = [[1,1,1],[1,0,1],[1,1,1]]`

**Output:** 32

**Example 3:**

2	2	2
2	1	2
2	2	2

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

**Output:** 46

**Constraints:**

n == grid.length == grid[i].length

1 <= n <= 50

0 <= grid[i][j] <= 50

## 893. Groups of Special-Equivalent Strings

Medium

4601420Add to ListShare

You are given an array of strings of the same length `words`.

In one **move**, you can swap any two even indexed characters or any two odd indexed characters of a string `words[i]`.

Two strings `words[i]` and `words[j]` are **special-equivalent** if after any number of moves, `words[i] == words[j]`.

For example, `words[i] = "zzxy"` and `words[j] = "xyzz"` are **special-equivalent** because we may make the moves "zzxy" -> "xzzy" -> "xyzz".

A **group of special-equivalent strings** from `words` is a non-empty subset of `words` such that:

Every pair of strings in the group are special equivalent, and

The group is the largest size possible (i.e., there is not a string `words[i]` not in the group such that `words[i]` is special-equivalent to every string in the group).

Return *the number of groups of special-equivalent strings* from `words`.

### Example 1:

**Input:** `words = ["abcd", "cdab", "cbad", "xyzz", "zzxy", "zzyx"]`

**Output:** 3

#### Explanation:

One group is `["abcd", "cdab", "cbad"]`, since they are all pairwise special equivalent, and none of the other strings is all pairwise special equivalent to these.

The other two groups are `["xyzz", "zzxy"]` and `["zzyx"]`.

Note that in particular, `"zzxy"` is not special equivalent to `"zzyx"`.

### Example 2:

**Input:** `words = ["abc", "acb", "bac", "bca", "cab", "cba"]`

**Output:** 3

#### Constraints:

`1 <= words.length <= 1000`

`1 <= words[i].length <= 20`

`words[i]` consist of lowercase English letters.

All the strings are of the same length.

## 894. All Possible Full Binary Trees

Medium

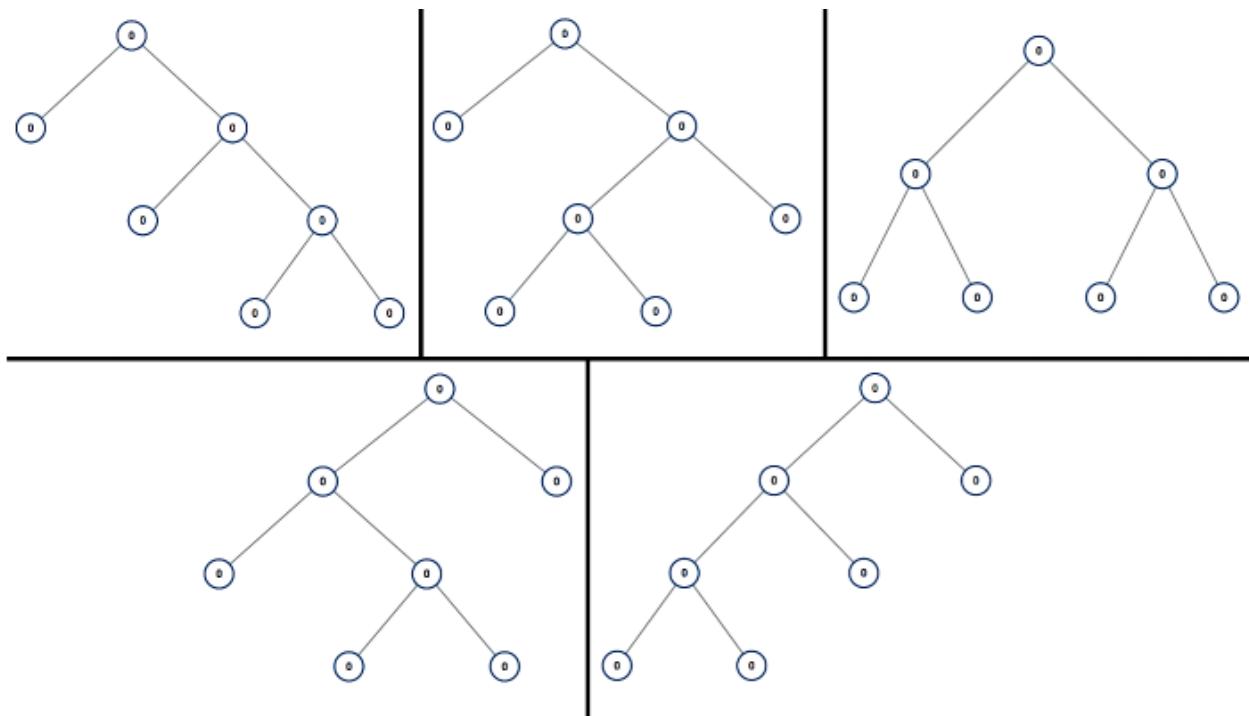
3104217Add to ListShare

Given an integer `n`, return *a list of all possible full binary trees* with `n` nodes. Each node of each tree in the answer must have `Node.val == 0`.

Each element of the answer is the root node of one possible tree. You may return the final list of trees in **any order**.

A **full binary tree** is a binary tree where each node has exactly 0 or 2 children.

**Example 1:**



**Input:** n = 7

**Output:**

```
[[0,0,0,null,null,0,0,null,null,0,0],[0,0,0,null,null,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0]]
```

**Example 2:**

**Input:** n = 3

**Output:** [[0,0,0]]

**Constraints:**

1 <= n <= 20

## 895. Maximum Frequency Stack

Hard

397257Add to ListShare

Design a stack-like data structure to push elements to the stack and pop the most frequent element from the stack.

Implement the `FreqStack` class:

`FreqStack()` constructs an empty frequency stack.

`void push(int val)` pushes an integer `val` onto the top of the stack.

`int pop()` removes and returns the most frequent element in the stack.

If there is a tie for the most frequent element, the element closest to the stack's top is removed and returned.

### Example 1:

#### Input

```
["FreqStack", "push", "push", "push", "push", "push", "push", "pop", "pop", "pop"]
[[], [5], [7], [5], [7], [4], [5], [], [], [], []]
```

#### Output

```
[null, null, null, null, null, null, null, 5, 7, 5, 4]
```

#### Explanation

```
FreqStack freqStack = new FreqStack();
freqStack.push(5); // The stack is [5]
freqStack.push(7); // The stack is [5,7]
freqStack.push(5); // The stack is [5,7,5]
freqStack.push(7); // The stack is [5,7,5,7]
freqStack.push(4); // The stack is [5,7,5,7,4]
freqStack.push(5); // The stack is [5,7,5,7,4,5]
freqStack.pop(); // return 5, as 5 is the most frequent. The stack becomes
[5,7,5,7,4].
```

```
freqStack.pop(); // return 7, as 5 and 7 is the most frequent, but 7 is closest to
the top. The stack becomes [5,7,5,4].
```

```
freqStack.pop(); // return 5, as 5 is the most frequent. The stack becomes [5,7,4].
```

```
freqStack.pop(); // return 4, as 4, 5 and 7 is the most frequent, but 4 is closest
to the top. The stack becomes [5,7].
```

### Constraints:

`0 <= val <= 109`

At most  $2 * 10^4$  calls will be made to `push` and `pop`.

It is guaranteed that there will be at least one element in the stack before calling `pop`.

## 896. Monotonic Array

Easy

174059Add to ListShare

An array is **monotonic** if it is either monotone increasing or monotone decreasing.

An array `nums` is monotone increasing if for all  $i \leq j$ ,  $nums[i] \leq nums[j]$ . An array `nums` is monotone decreasing if for all  $i \leq j$ ,  $nums[i] \geq nums[j]$ .

Given an integer array `nums`, return `true` if the given array is monotonic, or `false` otherwise.

### Example 1:

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

**Output:** `true`

### Example 2:

**Input:** `nums = [6,5,4,4]`

**Output:** `true`

### Example 3:

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

**Output:** `false`

**Constraints:**

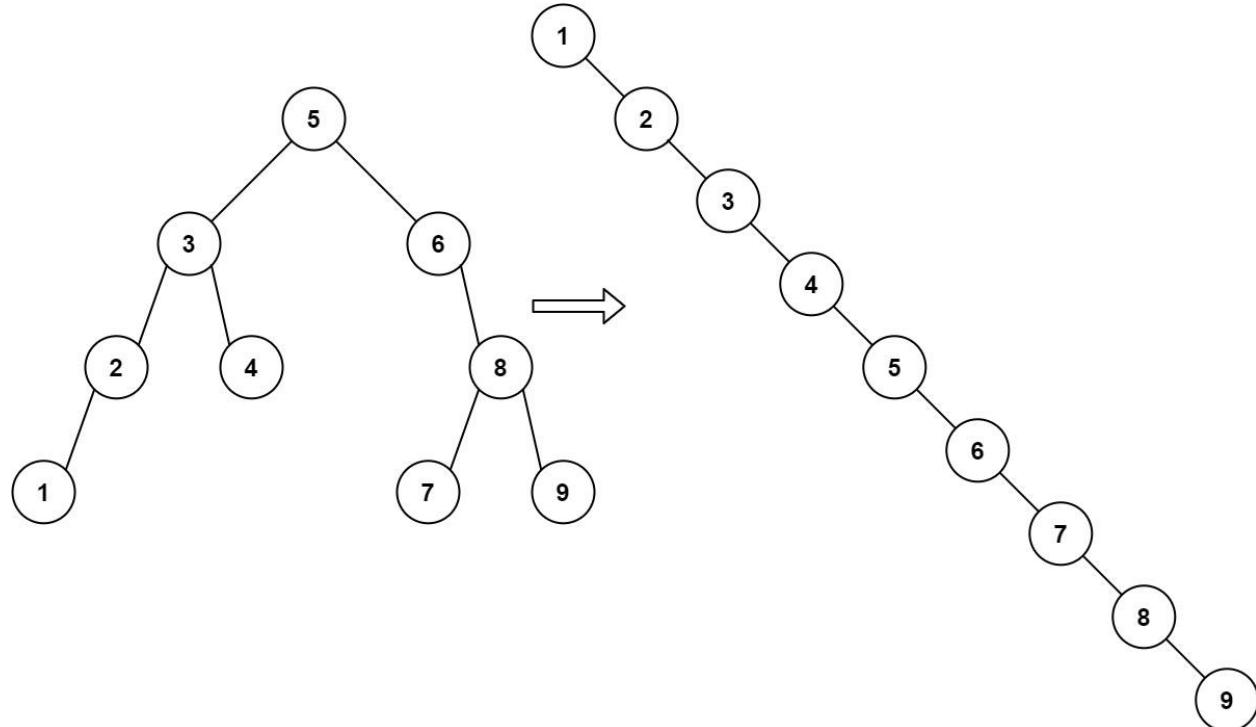
```
1 <= nums.length <= 105
```

```
-105 <= nums[i] <= 105
```

**897. Increasing Order Search Tree****Easy**

3552636 Add to List Share

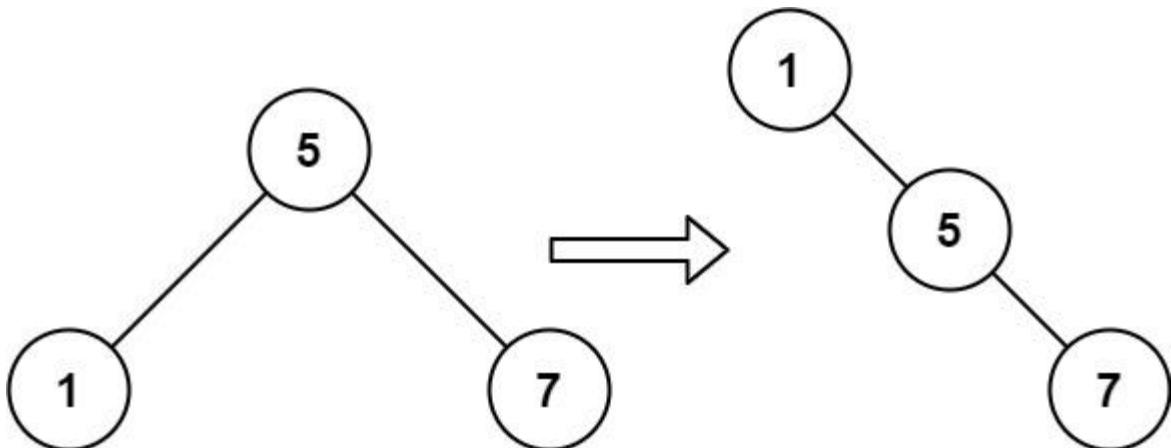
Given the `root` of a binary search tree, rearrange the tree in **in-order** so that the leftmost node in the tree is now the root of the tree, and every node has no left child and only one right child.

**Example 1:**

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

**Output:** `[1,null,2,null,3,null,4,null,5,null,6,null,7,null,8,null,9]`

**Example 2:**



**Input:** root = [5,1,7]

**Output:** [1,null,5,null,7]

#### Constraints:

The number of nodes in the given tree will be in the range [1, 100].

0 <= Node.val <= 1000

## 898. Bitwise ORs of Subarrays

Medium

1121188Add to ListShare

We have an array `arr` of non-negative integers.

For every (contiguous) subarray `sub = [arr[i], arr[i + 1], ..., arr[j]]` (with  $i \leq j$ ), we take the bitwise OR of all the elements in `sub`, obtaining a result `arr[i] | arr[i + 1] | ... | arr[j]`.

Return the number of possible results. Results that occur more than once are only counted once in the final answer

#### Example 1:

**Input:** arr = [0]

**Output:** 1

**Explanation:** There is only one possible result: 0.

#### Example 2:

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

**Output:** 3

**Explanation:** The possible subarrays are [1], [1], [2], [1, 1], [1, 2], [1, 1, 2].

These yield the results 1, 1, 2, 1, 3, 3.

There are 3 unique values, so the answer is 3.

**Example 3:**

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

**Output:** 6

**Explanation:** The possible results are 1, 2, 3, 4, 6, and 7.

**Constraints:**

1 <= nums.length <= 5 \* 10<sup>4</sup>

0 <= nums[i] <= 10<sup>9</sup>

## 899. Orderly Queue

Hard

612363Add to ListShare

You are given a string `s` and an integer `k`. You can choose one of the first `k` letters of `s` and append it at the end of the string..

Return the lexicographically smallest string you could have after applying the mentioned step any number of moves.

**Example 1:**

**Input:** s = "cba", k = 1

**Output:** "acb"

**Explanation:**

In the first move, we move the 1<sup>st</sup> character 'c' to the end, obtaining the string "bac".

In the second move, we move the 1<sup>st</sup> character 'b' to the end, obtaining the final result "acb".

**Example 2:**

**Input:** `s = "baaca", k = 3`

**Output:** `"aaabc"`

**Explanation:**

In the first move, we move the 1<sup>st</sup> character 'b' to the end, obtaining the string "aacab".

In the second move, we move the 3<sup>rd</sup> character 'c' to the end, obtaining the final result "aaabc".

**Constraints:**

`1 <= k <= s.length <= 1000`

`s` consist of lowercase English letters.

**900. RLE Iterator****Medium**

596159Add to ListShare

We can use run-length encoding (i.e., **RLE**) to encode a sequence of integers. In a run-length encoded array of even length `encoding` (**0-indexed**), for all even `i`, `encoding[i]` tells us the number of times that the non-negative integer value `encoding[i + 1]` is repeated in the sequence.

For example, the sequence `arr = [8, 8, 8, 5, 5]` can be encoded to be `encoding = [3, 8, 2, 5]`. `encoding = [3, 8, 0, 9, 2, 5]` and `encoding = [2, 8, 1, 8, 2, 5]` are also valid **RLE** of `arr`.

Given a run-length encoded array, design an iterator that iterates through it.

Implement the `RLEIterator` class:

`RLEIterator(int[] encoded)` Initializes the object with the encoded array `encoded`.

`int next(int n)` Exhausts the next `n` elements and returns the last element exhausted in this way. If there is no element left to exhaust, return `-1` instead.

**Example 1:****Input**

```
["RLEIterator", "next", "next", "next", "next"]
[[[3, 8, 0, 9, 2, 5]], [2], [1], [1], [2]]
```

#### Output

```
[null, 8, 8, 5, -1]
```

#### Explanation

RLEIterator rLEIterator = new RLEIterator([3, 8, 0, 9, 2, 5]); // This maps to the sequence [8,8,8,5,5].

rLEIterator.next(2); // exhausts 2 terms of the sequence, returning 8. The remaining sequence is now [8, 5, 5].

rLEIterator.next(1); // exhausts 1 term of the sequence, returning 8. The remaining sequence is now [5, 5].

rLEIterator.next(1); // exhausts 1 term of the sequence, returning 5. The remaining sequence is now [5].

rLEIterator.next(2); // exhausts 2 terms, returning -1. This is because the first term exhausted was 5,

but the second term did not exist. Since the last term exhausted does not exist, we return -1.

#### Constraints:

`2 <= encoding.length <= 1000`

`encoding.length` is even.

`0 <= encoding[i] <= 109`

`1 <= n <= 109`

At most 1000 calls will be made to `next`.

## 901. Online Stock Span

#### Medium

3370236 Add to List Share

Design an algorithm that collects daily price quotes for some stock and returns **the span** of that stock's price for the current day.

The **span** of the stock's price today is defined as the maximum number of consecutive days (starting from today and going backward) for which the stock price was less than or equal to today's price.

For example, if the price of a stock over the next 7 days were [100, 80, 60, 70, 60, 75, 85], then the stock spans would be [1, 1, 1, 2, 1, 4, 6].

Implement the `StockSpanner` class:

`StockSpanner()` Initializes the object of the class.

`int next(int price)` Returns the **span** of the stock's price given that today's price is `price`.

### Example 1:

#### Input

```
["StockSpanner", "next", "next", "next", "next", "next", "next", "next"]
[[], [100], [80], [60], [70], [60], [75], [85]]
```

#### Output

```
[null, 1, 1, 1, 2, 1, 4, 6]
```

#### Explanation

```
StockSpanner stockSpanner = new StockSpanner();
stockSpanner.next(100); // return 1
stockSpanner.next(80); // return 1
stockSpanner.next(60); // return 1
stockSpanner.next(70); // return 2
stockSpanner.next(60); // return 1
stockSpanner.next(75); // return 4, because the last 4 prices (including today's
                     // price of 75) were less than or equal to today's price.
stockSpanner.next(85); // return 6
```

#### Constraints:

`1 <= price <= 105`

At most  $10^4$  calls will be made to `next`.

## 902. Numbers At Most N Given Digit Set

Hard

113292Add to ListShare

Given an array of `digits` which is sorted in **non-decreasing** order. You can write numbers using each `digits[i]` as many times as we want. For example, if `digits = ['1', '3', '5']`, we may write numbers such as `'13'`, `'551'`, and `'1351315'`.

Return *the number of positive integers that can be generated* that are less than or equal to a given integer `n`.

### Example 1:

**Input:** `digits = ["1", "3", "5", "7"]`, `n = 100`

**Output:** 20

**Explanation:**

The 20 numbers that can be written are:

`1, 3, 5, 7, 11, 13, 15, 17, 31, 33, 35, 37, 51, 53, 55, 57, 71, 73, 75, 77.`

### Example 2:

**Input:** `digits = ["1", "4", "9"]`, `n = 1000000000`

**Output:** 29523

**Explanation:**

We can write 3 one digit numbers, 9 two digit numbers, 27 three digit numbers, 81 four digit numbers, 243 five digit numbers, 729 six digit numbers, 2187 seven digit numbers, 6561 eight digit numbers, and 19683 nine digit numbers.

In total, this is 29523 integers that can be written using the `digits` array.

### Example 3:

**Input:** `digits = ["7"]`, `n = 8`

**Output:** 1

**Constraints:**

`1 <= digits.length <= 9`  
`digits[i].length == 1`  
`digits[i]` is a digit from '1' to '9'.

All the values in `digits` are **unique**.

`digits` is sorted in **non-decreasing** order.

`1 <= n <= 109`

## 904. Fruit Into Baskets

**Medium**

1611109Add to ListShare

You are visiting a farm that has a single row of fruit trees arranged from left to right. The trees are represented by an integer array `fruits` where `fruits[i]` is the **type** of fruit the `ith` tree produces.

You want to collect as much fruit as possible. However, the owner has some strict rules that you must follow:

You only have **two** baskets, and each basket can only hold a **single type** of fruit. There is no limit on the amount of fruit each basket can hold.

Starting from any tree of your choice, you must pick **exactly one fruit** from **every** tree (including the start tree) while moving to the right. The picked fruits must fit in one of your baskets.

Once you reach a tree with fruit that cannot fit in your baskets, you must stop.

Given the integer array `fruits`, return the **maximum** number of fruits you can pick.

### Example 1:

**Input:** `fruits = [1,2,1]`

**Output:** 3

**Explanation:** We can pick from all 3 trees.

### Example 2:

**Input:** fruits = [0,1,2,2]

**Output:** 3

**Explanation:** We can pick from trees [1,2,2].

If we had started at the first tree, we would only pick from trees [0,1].

**Example 3:**

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

**Output:** 4

**Explanation:** We can pick from trees [2,3,2,2].

If we had started at the first tree, we would only pick from trees [1,2].

**Constraints:**

1 <= fruits.length <=  $10^5$

0 <= fruits[i] < fruits.length

## 905. Sort Array By Parity

**Easy**

3859124Add to ListShare

Given an integer array `nums`, move all the even integers at the beginning of the array followed by all the odd integers.

Return **any array** that satisfies this condition.

**Example 1:**

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

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

**Explanation:** The outputs [4,2,3,1], [2,4,1,3], and [4,2,1,3] would also be accepted.

**Example 2:**

**Input:** nums = [0]

**Output:** [0]

**Constraints:**

```
1 <= nums.length <= 5000
```

```
0 <= nums[i] <= 5000
```

**906. Super Palindromes****Hard**

313389Add to ListShare

Let's say a positive integer is a **super-palindrome** if it is a palindrome, and it is also the square of a palindrome.

Given two positive integers `left` and `right` represented as strings, return *the number of **super-palindromes** integers in the inclusive range `[left, right]`*.

**Example 1:**

**Input:** `left = "4", right = "1000"`

**Output:** 4

**Explanation:** 4, 9, 121, and 484 are superpalindromes.

Note that 676 is not a superpalindrome:  $26 * 26 = 676$ , but 26 is not a palindrome.

**Example 2:**

**Input:** `left = "1", right = "2"`

**Output:** 1

**Constraints:**

```
1 <= left.length, right.length <= 18
```

`left` and `right` consist of only digits.

`left` and `right` cannot have leading zeros.

`left` and `right` represent integers in the range `[1, 1018 - 1]`.

`left` is less than or equal to `right`.

## 907. Sum of Subarray Minimums

Medium

4253281Add to ListShare

Given an array of integers `arr`, find the sum of `min(b)`, where `b` ranges over every (contiguous) subarray of `arr`. Since the answer may be large, return the answer **modulo**  $10^9 + 7$ .

**Example 1:**

**Input:** `arr = [3,1,2,4]`

**Output:** 17

**Explanation:**

Subarrays are `[3]`, `[1]`, `[2]`, `[4]`, `[3,1]`, `[1,2]`, `[2,4]`, `[3,1,2]`, `[1,2,4]`, `[3,1,2,4]`.

Minimums are 3, 1, 2, 4, 1, 1, 2, 1, 1, 1.

Sum is 17.

**Example 2:**

**Input:** `arr = [11,81,94,43,3]`

**Output:** 444

**Constraints:**

`1 <= arr.length <= 3 * 104`

`1 <= arr[i] <= 3 * 104`

## 908. Smallest Range I

Easy

491711Add to ListShare

You are given an integer array `nums` and an integer `k`.

In one operation, you can choose any index `i` where `0 <= i < nums.length` and change `nums[i]` to `nums[i] + x` where `x` is an integer from the range `[-k, k]`. You can apply this operation **at most once** for each index `i`.

The **score** of `nums` is the difference between the maximum and minimum elements in `nums`.

Return the minimum **score** of `nums` after applying the mentioned operation at most once for each index in it.

**Example 1:**

**Input:** `nums = [1]`, `k = 0`

**Output:** 0

**Explanation:** The score is `max(nums) - min(nums) = 1 - 1 = 0`.

**Example 2:**

**Input:** `nums = [0,10]`, `k = 2`

**Output:** 6

**Explanation:** Change `nums` to be `[2, 8]`. The score is `max(nums) - min(nums) = 8 - 2 = 6`.

**Example 3:**

**Input:** `nums = [1,3,6]`, `k = 3`

**Output:** 0

**Explanation:** Change `nums` to be `[4, 4, 4]`. The score is `max(nums) - min(nums) = 4 - 4 = 0`.

**Constraints:**

`1 <= nums.length <= 104`

`0 <= nums[i] <= 104`

`0 <= k <= 104`

## 909. Snakes and Ladders

Medium

972254Add to ListShare

You are given an `n x n` integer matrix `board` where the cells are labeled from 1 to `n2` in a **Boustrophedon style** starting from the bottom left of the board (i.e. `board[n - 1][0]`) and alternating direction each row.

You start on square 1 of the board. In each move, starting from square `curr`, do the following:

Choose a destination square `next` with a label in the range `[curr + 1, min(curr + 6, n2)]`.

This choice simulates the result of a standard **6-sided die roll**: i.e., there are always at most 6 destinations, regardless of the size of the board.

If `next` has a snake or ladder, you **must** move to the destination of that snake or ladder. Otherwise, you move to `next`.

The game ends when you reach the square  $n^2$ .

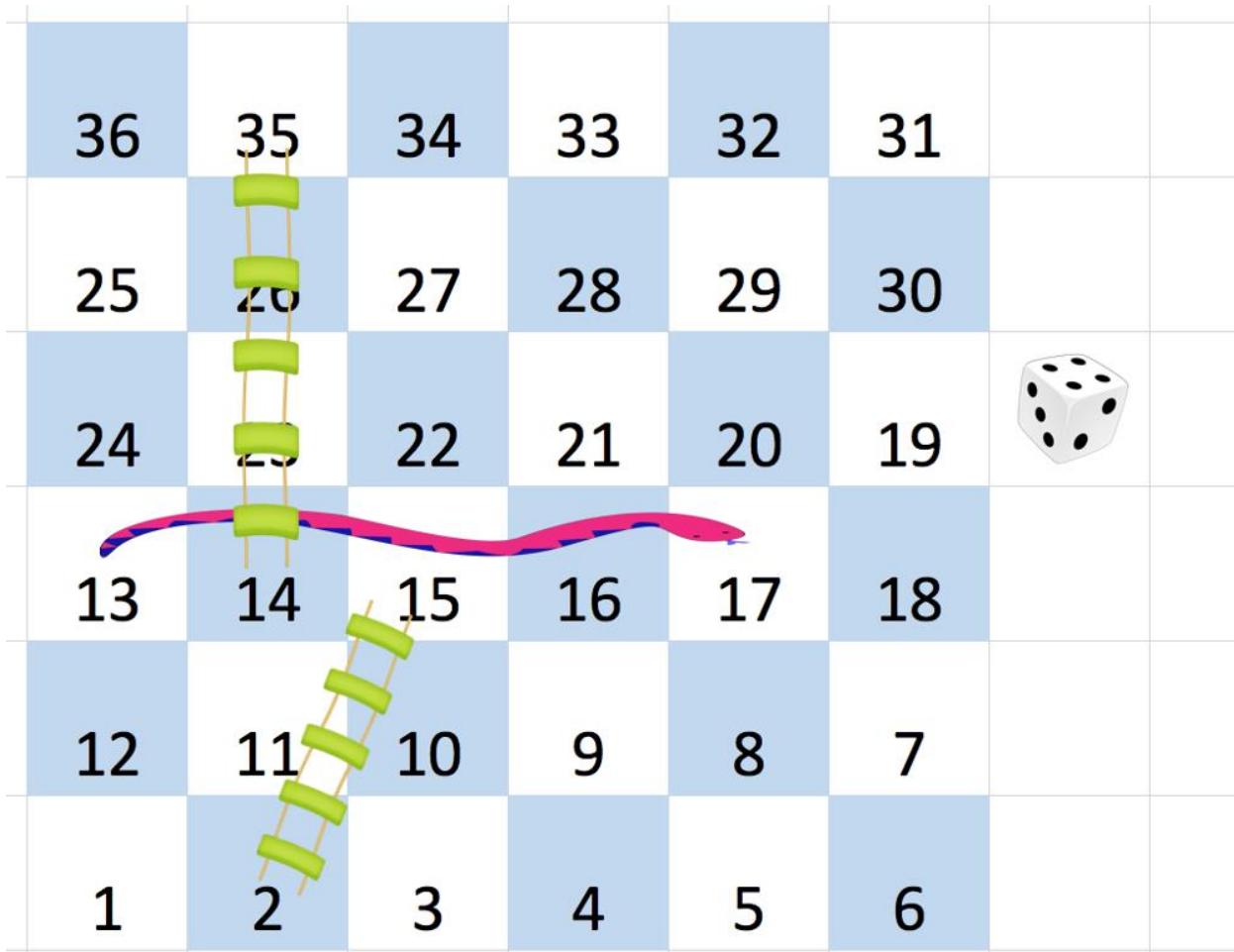
A board square on row `r` and column `c` has a snake or ladder if `board[r][c] != -1`. The destination of that snake or ladder is `board[r][c]`. Squares `1` and  $n^2$  do not have a snake or ladder.

Note that you only take a snake or ladder at most once per move. If the destination to a snake or ladder is the start of another snake or ladder, you do **not** follow the subsequent snake or ladder.

For example, suppose the board is `[[ -1, 4], [-1, 3]]`, and on the first move, your destination square is `2`. You follow the ladder to square `3`, but do **not** follow the subsequent ladder to `4`.

Return *the least number of moves required to reach the square  $n^2$ . If it is not possible to reach the square, return `-1`.*

### Example 1:



**Input:** board = `[[ -1, -1, -1, -1, -1, -1], [-1, -1, -1, -1, -1, -1], [-1, -1, -1, -1, -1, -1], [-1, 35, -1, -1, 13, -1], [-1, -1, -1, -1, -1, -1], [-1, 15, -1, -1, -1, -1]]`

**Output:** 4

**Explanation:**

In the beginning, you start at square 1 (at row 5, column 0).

You decide to move to square 2 and must take the ladder to square 15.

You then decide to move to square 17 and must take the snake to square 13.

You then decide to move to square 14 and must take the ladder to square 35.

You then decide to move to square 36, ending the game.

This is the lowest possible number of moves to reach the last square, so return 4.

**Example 2:**

**Input:** board = `[[ -1, -1], [-1, 3]]`

**Output:** 1

**Constraints:**

```
n == board.length == board[i].length
2 <= n <= 20
grid[i][j] is either -1 or in the range [1, n^2].
```

The squares labeled 1 and  $n^2$  do not have any ladders or snakes.

## 910. Smallest Range II

Medium

1298382Add to ListShare

You are given an integer array `nums` and an integer `k`.

For each index `i` where  $0 \leq i < \text{nums.length}$ , change `nums[i]` to be either `nums[i] + k` or `nums[i] - k`.

The **score** of `nums` is the difference between the maximum and minimum elements in `nums`.

Return *the minimum score of `nums` after changing the values at each index*.

**Example 1:**

**Input:** `nums = [1]`, `k = 0`

**Output:** 0

**Explanation:** The score is `max(nums) - min(nums) = 1 - 1 = 0`.

**Example 2:**

**Input:** `nums = [0,10]`, `k = 2`

**Output:** 6

**Explanation:** Change `nums` to be `[2, 8]`. The score is `max(nums) - min(nums) = 8 - 2 = 6`.

**Example 3:**

**Input:** `nums = [1,3,6]`, `k = 3`

**Output:** 3

**Explanation:** Change `nums` to be `[4, 6, 3]`. The score is `max(nums) - min(nums) = 6 - 3 = 3`.

**Constraints:**

`1 <= nums.length <= 104`

`0 <= nums[i] <= 104`

`0 <= k <= 104`

## 911. Online Election

Medium

794558Add to ListShare

You are given two integer arrays `persons` and `times`. In an election, the `ith` vote was cast for `persons[i]` at time `times[i]`.

For each query at a time `t`, find the person that was leading the election at time `t`. Votes cast at time `t` will count towards our query. In the case of a tie, the most recent vote (among tied candidates) wins.

Implement the `TopVotedCandidate` class:

`TopVotedCandidate(int[] persons, int[] times)` Initializes the object with the `persons` and `times` arrays.

`int q(int t)` Returns the number of the person that was leading the election at time `t` according to the mentioned rules.

**Example 1:**

**Input**

```
["TopVotedCandidate", "q", "q", "q", "q", "q", "q", "q"]
[[[0, 1, 1, 0, 0, 1, 0], [0, 5, 10, 15, 20, 25, 30]], [3], [12], [25], [15], [24], [8]]
```

**Output**

```
[null, 0, 1, 1, 0, 0, 1]
```

### Explanation

```
TopVotedCandidate topVotedCandidate = new TopVotedCandidate([0, 1, 1, 0, 0, 1, 0],  
[0, 5, 10, 15, 20, 25, 30]);  
  
topVotedCandidate.q(3); // return 0, At time 3, the votes are [0], and 0 is leading.  
  
topVotedCandidate.q(12); // return 1, At time 12, the votes are [0,1,1], and 1 is  
leading.  
  
topVotedCandidate.q(25); // return 1, At time 25, the votes are [0,1,1,0,0,1], and 1  
is leading (as ties go to the most recent vote.)  
  
topVotedCandidate.q(15); // return 0  
  
topVotedCandidate.q(24); // return 0  
  
topVotedCandidate.q(8); // return 1
```

### Constraints:

```
1 <= persons.length <= 5000  
  
times.length == persons.length  
  
0 <= persons[i] < persons.length  
  
0 <= times[i] <= 109  
  
times is sorted in a strictly increasing order.  
  
times[0] <= t <= 109
```

At most  $10^4$  calls will be made to `q`.

## 912. Sort an Array

Medium

2770571Add to ListShare

Given an array of integers `nums`, sort the array in ascending order and return it.

You must solve the problem **without using any built-in** functions in  $O(n \log(n))$  time complexity and with the smallest space complexity possible.

### Example 1:

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

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

**Explanation:** After sorting the array, the positions of some numbers are not changed (for example, 2 and 3), while the positions of other numbers are changed (for example, 1 and 5).

**Example 2:**

**Input:** nums = [5,1,1,2,0,0]

**Output:** [0,0,1,1,2,5]

**Explanation:** Note that the values of nums are not necessarily unique.

**Constraints:**

```
1 <= nums.length <= 5 * 104
-5 * 104 <= nums[i] <= 5 * 104
```

## 913. Cat and Mouse

Hard

718127Add to ListShare

A game on an **undirected** graph is played by two players, Mouse and Cat, who alternate turns.

The graph is given as follows: `graph[a]` is a list of all nodes `b` such that `ab` is an edge of the graph.

The mouse starts at node `1` and goes first, the cat starts at node `2` and goes second, and there is a hole at node `0`.

During each player's turn, they **must** travel along one edge of the graph that meets where they are. For example, if the Mouse is at node `1`, it **must** travel to any node in `graph[1]`.

Additionally, it is not allowed for the Cat to travel to the Hole (node `0`.)

Then, the game can end in three ways:

If ever the Cat occupies the same node as the Mouse, the Cat wins.

If ever the Mouse reaches the Hole, the Mouse wins.

If ever a position is repeated (i.e., the players are in the same position as a previous turn, and it is the same player's turn to move), the game is a draw.

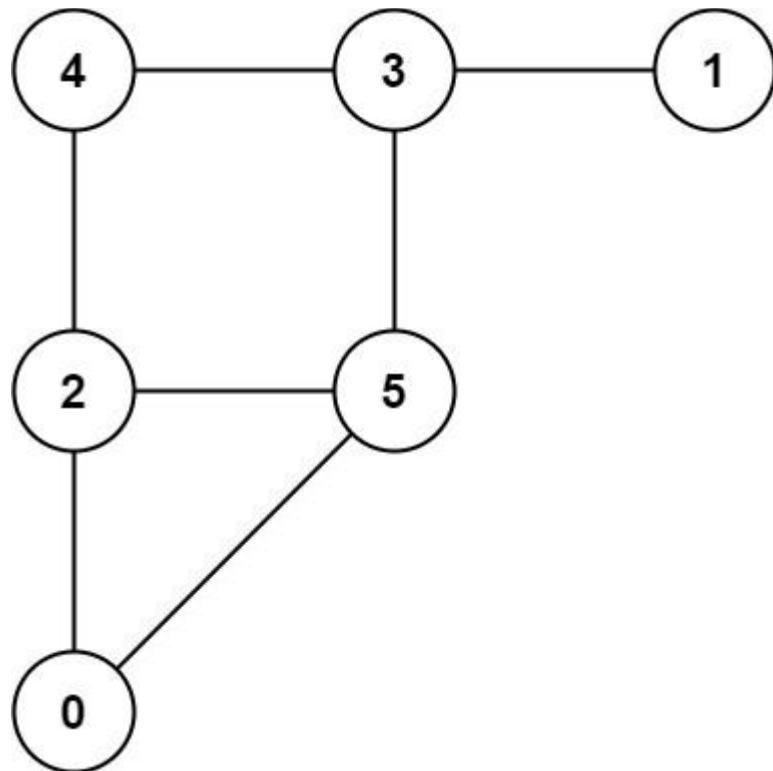
Given a `graph`, and assuming both players play optimally, return

`1` if the mouse wins the game,

`2` if the cat wins the game, or

`0` if the game is a draw.

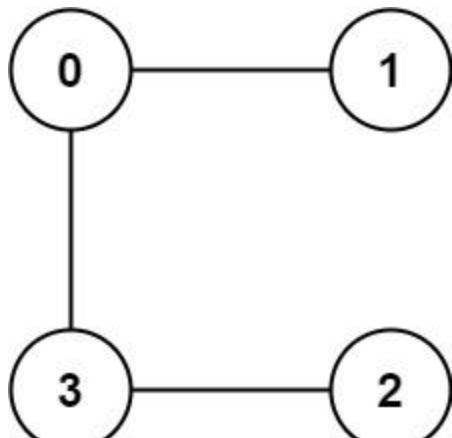
**Example 1:**



**Input:** `graph = [[2,5],[3],[0,4,5],[1,4,5],[2,3],[0,2,3]]`

**Output:** `0`

**Example 2:**



**Input:** graph = [[1,3],[0],[3],[0,2]]

**Output:** 1

**Constraints:**

3 <= graph.length <= 50

1 <= graph[i].length < graph.length

0 <= graph[i][j] < graph.length

graph[i][j] != i

graph[i] is unique.

The mouse and the cat can always move.

## 914. X of a Kind in a Deck of Cards

**Easy**

1420345Add to ListShare

In a deck of cards, each card has an integer written on it.

Return `true` if and only if you can choose  $x \geq 2$  such that it is possible to split the entire deck into 1 or more groups of cards, where:

Each group has exactly  $x$  cards.

All the cards in each group have the same integer.

**Example 1:**

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

**Output:** true

**Explanation:** Possible partition [1,1],[2,2],[3,3],[4,4].

**Example 2:**

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

**Output:** false

**Explanation:** No possible partition.

**Constraints:**

1 <= deck.length <= 10<sup>4</sup>

0 <= deck[i] < 10<sup>4</sup>

## 915. Partition Array into Disjoint Intervals

Medium

132167Add to ListShare

Given an integer array `nums`, partition it into two (contiguous) subarrays `left` and `right` so that:

Every element in `left` is less than or equal to every element in `right`.

`left` and `right` are non-empty.

`left` has the smallest possible size.

Return the length of `left` after such a partitioning.

Test cases are generated such that partitioning exists.

**Example 1:**

**Input:** nums = [5,0,3,8,6]

**Output:** 3

**Explanation:** `left` = [5,0,3], `right` = [8,6]

**Example 2:**

**Input:** nums = [1,1,1,0,6,12]

**Output:** 4

**Explanation:** left = [1,1,1,0], right = [6,12]

### Constraints:

2  $\leq$  nums.length  $\leq$  10<sup>5</sup>

0  $\leq$  nums[i]  $\leq$  10<sup>6</sup>

There is at least one valid answer for the given input.

## 916. Word Subsets

Medium

2440201Add to ListShare

You are given two string arrays words1 and words2.

A string  $b$  is a **subset** of string  $a$  if every letter in  $b$  occurs in  $a$  including multiplicity.

For example, "wrr" is a subset of "warrior" but is not a subset of "world".

A string  $a$  from words1 is **universal** if for every string  $b$  in words2,  $b$  is a subset of  $a$ .

Return an array of all the **universal** strings in words1. You may return the answer in **any order**.

### Example 1:

**Input:** words1 = ["amazon", "apple", "facebook", "google", "leetcode"], words2 = ["e", "o"]

**Output:** ["facebook", "google", "leetcode"]

### Example 2:

**Input:** words1 = ["amazon", "apple", "facebook", "google", "leetcode"], words2 = ["l", "e"]

**Output:** ["apple", "google", "leetcode"]

### Constraints:

1  $\leq$  words1.length, words2.length  $\leq$  10<sup>4</sup>

```
1 <= words1[i].length, words2[i].length <= 10
```

`words1[i]` and `words2[i]` consist only of lowercase English letters.

All the strings of `words1` are **unique**.

## 917. Reverse Only Letters

Easy

160755Add to ListShare

Given a string `s`, reverse the string according to the following rules:

All the characters that are not English letters remain in the same position.

All the English letters (lowercase or uppercase) should be reversed.

Return `s` *after reversing it*.

**Example 1:**

**Input:** `s = "ab-cd"`

**Output:** `"dc-ba"`

**Example 2:**

**Input:** `s = "a-bC-dEf-ghIj"`

**Output:** `"j-Ih-gfE-dCba"`

**Example 3:**

**Input:** `s = "Test1ng-Leet=code-Q!"`

**Output:** `"Qedo1ct-eeLg=ntse-T!"`

**Constraints:**

```
1 <= s.length <= 100
```

`s` consists of characters with ASCII values in the range `[33, 122]`.

`s` does not contain `'\'` or `'\\'`.

## 918. Maximum Sum Circular Subarray

### Medium

3753173Add to ListShare

Given a **circular integer array** `nums` of length `n`, return *the maximum possible sum of a non-empty subarray* of `nums`.

A **circular array** means the end of the array connects to the beginning of the array. Formally, the next element of `nums[i]` is `nums[(i + 1) % n]` and the previous element of `nums[i]` is `nums[(i - 1 + n) % n]`.

A **subarray** may only include each element of the fixed buffer `nums` at most once. Formally, for a subarray `nums[i], nums[i + 1], ..., nums[j]`, there does not exist `i <= k1, k2 <= j` with `k1 % n == k2 % n`.

#### Example 1:

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

**Output:** 3

**Explanation:** Subarray [3] has maximum sum 3.

#### Example 2:

**Input:** `nums = [5, -3, 5]`

**Output:** 10

**Explanation:** Subarray [5,5] has maximum sum  $5 + 5 = 10$ .

#### Example 3:

**Input:** `nums = [-3, -2, -3]`

**Output:** -2

**Explanation:** Subarray [-2] has maximum sum -2.

#### Constraints:

```

n == nums.length
1 <= n <= 3 * 104
-3 * 104 <= nums[i] <= 3 * 104

```

## 919. Complete Binary Tree Inserter

**Medium**

86282Add to ListShare

A **complete binary tree** is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

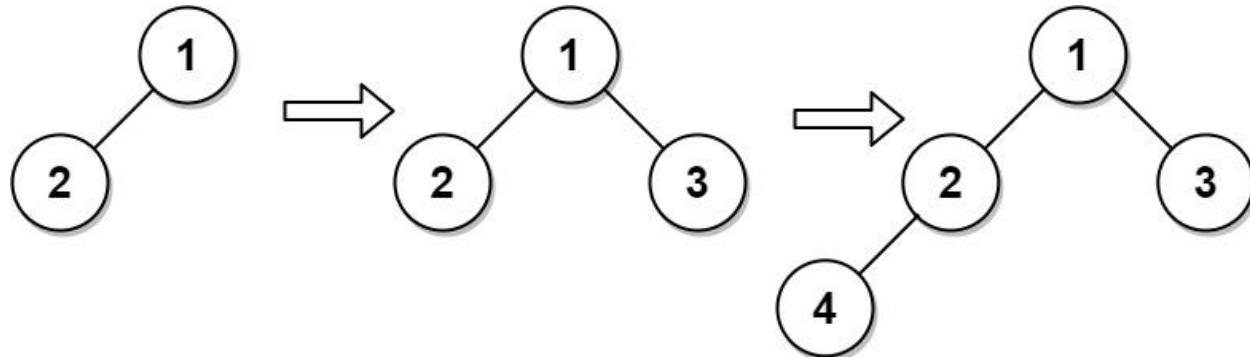
Design an algorithm to insert a new node to a complete binary tree keeping it complete after the insertion.

Implement the `CBTInserter` class:

`CBTInserter(TreeNode root)` Initializes the data structure with the `root` of the complete binary tree.

`int insert(int v)` Inserts a `TreeNode` into the tree with value `Node.val == val` so that the tree remains complete, and returns the value of the parent of the inserted `TreeNode`.

`TreeNode get_root()` Returns the root node of the tree.

**Example 1:****Input**

```
["CBTInserter", "insert", "insert", "get_root"]
[[[1, 2]], [3], [4], []]
```

**Output**

```
[null, 1, 2, [1, 2, 3, 4]]
```

**Explanation**

```
CBTInserter cBTInserter = new CBTInserter([1, 2]);
```

```
cBTInserter.insert(3); // return 1
```

```
cBTInserter.insert(4); // return 2
cBTInserter.get_root(); // return [1, 2, 3, 4]
```

### Constraints:

The number of nodes in the tree will be in the range  $[1, 1000]$ .

$0 \leq \text{Node.val} \leq 5000$

`root` is a complete binary tree.

$0 \leq \text{val} \leq 5000$

At most  $10^4$  calls will be made to `insert` and `get_root`.

## 920. Number of Music Playlists

Hard

82281Add to ListShare

Your music player contains  $n$  different songs. You want to listen to  $goal$  songs (not necessarily different) during your trip. To avoid boredom, you will create a playlist so that:

Every song is played **at least once**.

A song can only be played again only if  $k$  other songs have been played.

Given  $n$ ,  $goal$ , and  $k$ , return *the number of possible playlists that you can create*. Since the answer can be very large, return it **modulo**  $10^9 + 7$ .

### Example 1:

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

**Output:** 6

**Explanation:** There are 6 possible playlists:  $[1, 2, 3]$ ,  $[1, 3, 2]$ ,  $[2, 1, 3]$ ,  $[2, 3, 1]$ ,  $[3, 1, 2]$ , and  $[3, 2, 1]$ .

### Example 2:

**Input:**  $n = 2$ ,  $goal = 3$ ,  $k = 0$

**Output:** 6

**Explanation:** There are 6 possible playlists: [1, 1, 2], [1, 2, 1], [2, 1, 1], [2, 2, 1], [2, 1, 2], and [1, 2, 2].

**Example 3:**

**Input:** n = 2, goal = 3, k = 1

**Output:** 2

**Explanation:** There are 2 possible playlists: [1, 2, 1] and [2, 1, 2].

**Constraints:**

0 <= k < n <= goal <= 100

## 921. Minimum Add to Make Parentheses Valid

Medium

2964162Add to ListShare

A parentheses string is valid if and only if:

It is the empty string,

It can be written as `AB` (`A` concatenated with `B`), where `A` and `B` are valid strings, or

It can be written as `(A)`, where `A` is a valid string.

You are given a parentheses string `s`. In one move, you can insert a parenthesis at any position of the string.

For example, if `s = "(() )"`, you can insert an opening parenthesis to be `"(())"` or a closing parenthesis to be `"(()) )"`.

Return *the minimum number of moves required to make `s` valid.*

**Example 1:**

**Input:** s = "())"

**Output:** 1

**Example 2:**

**Input:** s = "(((("

**Output:** 3

**Constraints:**

`1 <= s.length <= 1000`

`s[i]` is either `'('` or `')'`.

## 922. Sort Array By Parity II

Easy

200879Add to ListShare

Given an array of integers `nums`, half of the integers in `nums` are **odd**, and the other half are **even**.

Sort the array so that whenever `nums[i]` is odd, `i` is **odd**, and whenever `nums[i]` is even, `i` is **even**.

Return *any answer array that satisfies this condition*.

**Example 1:**

**Input:** `nums = [4,2,5,7]`

**Output:** `[4,5,2,7]`

**Explanation:** `[4,7,2,5]`, `[2,5,4,7]`, `[2,7,4,5]` would also have been accepted.

**Example 2:**

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

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

**Constraints:**

`2 <= nums.length <= 2 * 104`

`nums.length` is even.

Half of the integers in `nums` are even.

`0 <= nums[i] <= 1000`

## 923. 3Sum With Multiplicity

**Medium**

2371284Add to ListShare

Given an integer array `arr`, and an integer `target`, return the number of tuples `i, j, k` such that `i < j < k` and `arr[i] + arr[j] + arr[k] == target`.

As the answer can be very large, return it **modulo**  $10^9 + 7$ .

**Example 1:**

**Input:** `arr = [1,1,2,2,3,3,4,4,5,5]`, `target = 8`

**Output:** 20

**Explanation:**

Enumerating by the values (`arr[i]`, `arr[j]`, `arr[k]`):

`(1, 2, 5)` occurs 8 times;

`(1, 3, 4)` occurs 8 times;

`(2, 2, 4)` occurs 2 times;

`(2, 3, 3)` occurs 2 times.

**Example 2:**

**Input:** `arr = [1,1,2,2,2,2]`, `target = 5`

**Output:** 12

**Explanation:**

`arr[i] = 1, arr[j] = arr[k] = 2` occurs 12 times:

We choose one 1 from `[1,1]` in 2 ways,

and two 2s from `[2,2,2,2]` in 6 ways.

**Example 3:**

**Input:** `arr = [2,1,3]`, `target = 6`

**Output:** 1

**Explanation:** `(1, 2, 3)` occurred one time in the array so we return 1.

**Constraints:**

```

3 <= arr.length <= 3000

0 <= arr[i] <= 100

0 <= target <= 300

```

## 924. Minimize Malware Spread

Hard

744463Add to ListShare

You are given a network of  $n$  nodes represented as an  $n \times n$  adjacency matrix  $\text{graph}$ , where the  $i^{\text{th}}$  node is directly connected to the  $j^{\text{th}}$  node if  $\text{graph}[i][j] == 1$ .

Some nodes  $\text{initial}$  are initially infected by malware. Whenever two nodes are directly connected, and at least one of those two nodes is infected by malware, both nodes will be infected by malware. This spread of malware will continue until no more nodes can be infected in this manner.

Suppose  $M(\text{initial})$  is the final number of nodes infected with malware in the entire network after the spread of malware stops. We will remove **exactly one node** from  $\text{initial}$ .

Return the node that, if removed, would minimize  $M(\text{initial})$ . If multiple nodes could be removed to minimize  $M(\text{initial})$ , return such a node with **the smallest index**.

Note that if a node was removed from the  $\text{initial}$  list of infected nodes, it might still be infected later due to the malware spread.

### Example 1:

**Input:**  $\text{graph} = [[1,1,0],[1,1,0],[0,0,1]]$ ,  $\text{initial} = [0,1]$

**Output:** 0

### Example 2:

**Input:**  $\text{graph} = [[1,0,0],[0,1,0],[0,0,1]]$ ,  $\text{initial} = [0,2]$

**Output:** 0

### Example 3:

**Input:**  $\text{graph} = [[1,1,1],[1,1,1],[1,1,1]]$ ,  $\text{initial} = [1,2]$

**Output:** 1

### Constraints:

```

n == graph.length

n == graph[i].length

2 <= n <= 300

graph[i][j] is 0 or 1.

graph[i][j] == graph[j][i]

graph[i][i] == 1

1 <= initial.length <= n

0 <= initial[i] <= n - 1

```

All the integers in `initial` are **unique**.

## 925. Long Pressed Name

**Easy**

1862271Add to ListShare

Your friend is typing his `name` into a keyboard. Sometimes, when typing a character `c`, the key might get *long pressed*, and the character will be typed 1 or more times.

You examine the `typed` characters of the keyboard. Return `True` if it is possible that it was your friends name, with some characters (possibly none) being long pressed.

**Example 1:**

**Input:** `name = "alex", typed = "aaleex"`

**Output:** `true`

**Explanation:** 'a' and 'e' in 'alex' were long pressed.

**Example 2:**

**Input:** `name = "saeed", typed = "ssaaedd"`

**Output:** `false`

**Explanation:** 'e' must have been pressed twice, but it was not in the typed output.

**Constraints:**

`1 <= name.length, typed.length <= 1000`

`name` and `typed` consist of only lowercase English letters.

## 926. Flip String to Monotone Increasing

Medium

225497Add to ListShare

A binary string is monotone increasing if it consists of some number of `0`'s (possibly none), followed by some number of `1`'s (also possibly none).

You are given a binary string `s`. You can flip `s[i]` changing it from `0` to `1` or from `1` to `0`.

Return *the minimum number of flips to make s monotone increasing*.

### Example 1:

**Input:** `s = "00110"`

**Output:** `1`

**Explanation:** We flip the last digit to get `00111`.

### Example 2:

**Input:** `s = "010110"`

**Output:** `2`

**Explanation:** We flip to get `011111`, or alternatively `000111`.

### Example 3:

**Input:** `s = "00011000"`

**Output:** `2`

**Explanation:** We flip to get `00000000`.

### Constraints:

`1 <= s.length <= 105`

`s[i]` is either `'0'` or `'1'`.

Accepted

107,386

Submissions

180,294

## 927. Three Equal Parts

**Hard**

722110Add to ListShare

You are given an array `arr` which consists of only zeros and ones, divide the array into **three non-empty parts** such that all of these parts represent the same binary value.

If it is possible, return any `[i, j]` with  $i + 1 < j$ , such that:

`arr[0], arr[1], ..., arr[i]` is the first part,

`arr[i + 1], arr[i + 2], ..., arr[j - 1]` is the second part, and

`arr[j], arr[j + 1], ..., arr[arr.length - 1]` is the third part.

All three parts have equal binary values.

If it is not possible, return `[-1, -1]`.

Note that the entire part is used when considering what binary value it represents. For example, `[1, 1, 0]` represents 6 in decimal, not 3. Also, leading zeros **are allowed**, so `[0, 1, 1]` and `[1, 1]` represent the same value.

### Example 1:

**Input:** `arr = [1, 0, 1, 0, 1]`

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

### Example 2:

**Input:** `arr = [1, 1, 0, 1, 1]`

**Output:** `[-1, -1]`

### Example 3:

**Input:** `arr = [1, 1, 0, 0, 1]`

**Output:** `[0, 2]`

### Constraints:

```
3 <= arr.length <= 3 * 104
arr[i] is 0 or 1
```

## 928. Minimize Malware Spread II

Hard

51776Add to ListShare

You are given a network of  $n$  nodes represented as an  $n \times n$  adjacency matrix  $graph$ , where the  $i^{\text{th}}$  node is directly connected to the  $j^{\text{th}}$  node if  $graph[i][j] == 1$ .

Some nodes  $initial$  are initially infected by malware. Whenever two nodes are directly connected, and at least one of those two nodes is infected by malware, both nodes will be infected by malware. This spread of malware will continue until no more nodes can be infected in this manner.

Suppose  $M(initial)$  is the final number of nodes infected with malware in the entire network after the spread of malware stops.

We will remove **exactly one node** from  $initial$ , **completely removing it and any connections from this node to any other node**.

Return the node that, if removed, would minimize  $M(initial)$ . If multiple nodes could be removed to minimize  $M(initial)$ , return such a node with **the smallest index**.

### Example 1:

**Input:**  $graph = [[1,1,0],[1,1,0],[0,0,1]]$ ,  $initial = [0,1]$

**Output:** 0

### Example 2:

**Input:**  $graph = [[1,1,0],[1,1,1],[0,1,1]]$ ,  $initial = [0,1]$

**Output:** 1

### Example 3:

**Input:**  $graph = [[1,1,0,0],[1,1,1,0],[0,1,1,1],[0,0,1,1]]$ ,  $initial = [0,1]$

**Output:** 1

### Constraints:

```
n == graph.length
```

```

n == graph[i].length

2 <= n <= 300

graph[i][j] is 0 or 1.

graph[i][j] == graph[j][i]

graph[i][i] == 1

1 <= initial.length < n

0 <= initial[i] <= n - 1

```

All the integers in `initial` are **unique**.

## 929. Unique Email Addresses

Easy

1969258Add to ListShare

Every **valid email** consists of a **local name** and a **domain name**, separated by the '@' sign. Besides lowercase letters, the email may contain one or more '.' or '+'.

For example, in "alice@leetcode.com", "alice" is the **local name**, and "leetcode.com" is the **domain name**.

If you add periods '.' between some characters in the **local name** part of an email address, mail sent there will be forwarded to the same address without dots in the local name. Note that this rule **does not apply to domain names**.

For example, "alice.z@leetcode.com" and "alicez@leetcode.com" forward to the same email address.

If you add a plus '+' in the **local name**, everything after the first plus sign **will be ignored**. This allows certain emails to be filtered. Note that this rule **does not apply to domain names**.

For example, "m.y+name@email.com" will be forwarded to "my@email.com".

It is possible to use both of these rules at the same time.

Given an array of strings `emails` where we send one email to each `emails[i]`, return *the number of different addresses that actually receive mails*.

**Example 1:**

**Input:** emails =  
["test.email+alex@leetcode.com", "test.e.mail+bob.cathy@leetcode.com", "testemail+david@lee.tcode.com"]

**Output:** 2

**Explanation:** "testemail@leetcode.com" and "testemail@lee.tcode.com" actually receive mails.

**Example 2:**

**Input:** emails = ["a@leetcode.com", "b@leetcode.com", "c@leetcode.com"]

**Output:** 3

**Constraints:**

1 <= emails.length <= 100

1 <= emails[i].length <= 100

emails[i] consist of lowercase English letters, '+' '.', and '@'.

Each emails[i] contains exactly one '@' character.

All local and domain names are non-empty.

Local names do not start with a '+' character.

Domain names end with the ".com" suffix.

## 930. Binary Subarrays With Sum

Medium

162354Add to ListShare

Given a binary array `nums` and an integer `goal`, return *the number of non-empty subarrays* with a sum `goal`.

A **subarray** is a contiguous part of the array.

**Example 1:**

**Input:** nums = [1,0,1,0,1], goal = 2

**Output:** 4

**Explanation:** The 4 subarrays are bolded and underlined below:

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

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

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

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

**Example 2:**

**Input:** nums = [0,0,0,0,0], goal = 0

**Output:** 15

**Constraints:**

1 <= nums.length <= 3 \* 10<sup>4</sup>

nums[i] is either 0 or 1.

0 <= goal <= nums.length

## 931. Minimum Falling Path Sum

Medium

304198Add to ListShare

Given an  $n \times n$  array of integers `matrix`, return the **minimum sum** of any **falling path** through `matrix`.

A **falling path** starts at any element in the first row and chooses the element in the next row that is either directly below or diagonally left/right. Specifically, the next element from position `(row, col)` will be `(row + 1, col - 1)`, `(row + 1, col)`, or `(row + 1, col + 1)`.

**Example 1:**

2	1	3
6	5	4
7	8	9

2	1	3
6	5	4
7	8	9

2	1	3
6	5	4
7	8	9

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

**Output:** 13

**Explanation:** There are two falling paths with a minimum sum as shown.

**Example 2:**

-19	57
-40	-5

-19	57
-40	-5

**Input:** matrix = [[-19,57],[-40,-5]]

**Output:** -59

**Explanation:** The falling path with a minimum sum is shown.

#### Constraints:

```
n == matrix.length == matrix[i].length
1 <= n <= 100
-100 <= matrix[i][j] <= 100
```

## 932. Beautiful Array

Medium

8721287Add to ListShare

An array `nums` of length `n` is **beautiful** if:

`nums` is a permutation of the integers in the range `[1, n]`.

For every `0 <= i < j < n`, there is no index `k` with `i < k < j` where `2 * nums[k] == nums[i] + nums[j]`.

Given the integer `n`, return *any beautiful array* `nums` of length `n`. There will be at least one valid answer for the given `n`.

**Example 1:**

**Input:** `n = 4`

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

**Example 2:**

**Input:** `n = 5`

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

**Constraints:**

`1 <= n <= 1000`

## 933. Number of Recent Calls

**Easy**

8282544Add to ListShare

You have a `RecentCounter` class which counts the number of recent requests within a certain time frame.

Implement the `RecentCounter` class:

`RecentCounter()` Initializes the counter with zero recent requests.

`int ping(int t)` Adds a new request at time `t`, where `t` represents some time in milliseconds, and returns the number of requests that has happened in the past `3000` milliseconds (including the new request). Specifically, return the number of requests that have happened in the inclusive range `[t - 3000, t]`.

It is **guaranteed** that every call to `ping` uses a strictly larger value of `t` than the previous call.

**Example 1:**

**Input**

`["RecentCounter", "ping", "ping", "ping", "ping"]`

```
[], [1], [100], [3001], [3002]]
```

#### Output

```
[null, 1, 2, 3, 3]
```

#### Explanation

```
RecentCounter recentCounter = new RecentCounter();

recentCounter.ping(1);      // requests = [1], range is [-2999,1], return 1
recentCounter.ping(100);    // requests = [1, 100], range is [-2900,100], return 2
recentCounter.ping(3001);   // requests = [1, 100, 3001], range is [1,3001], return 3
recentCounter.ping(3002);   // requests = [1, 100, 3001, 3002], range is [2,3002],
                           return 3
```

#### Constraints:

$1 \leq t \leq 10^9$

Each test case will call `ping` with **strictly increasing** values of `t`.

At most  $10^4$  calls will be made to `ping`.

## 934. Shortest Bridge

#### Medium

3175140Add to ListShare

You are given an  $n \times n$  binary matrix `grid` where `1` represents land and `0` represents water.

An **island** is a 4-directionally connected group of `1`'s not connected to any other `1`'s. There are **exactly two islands** in `grid`.

You may change `0`'s to `1`'s to connect the two islands to form **one island**.

Return *the smallest number of 0's you must flip to connect the two islands*.

#### Example 1:

**Input:** `grid = [[0,1],[1,0]]`

**Output:** `1`

**Example 2:**

```
Input: grid = [[0,1,0],[0,0,0],[0,0,1]]
```

```
Output: 2
```

**Example 3:**

```
Input: grid = [[1,1,1,1,1],[1,0,0,0,1],[1,0,1,0,1],[1,0,0,0,1],[1,1,1,1,1]]
```

```
Output: 1
```

**Constraints:**

`n == grid.length == grid[i].length`

`2 <= n <= 100`

`grid[i][j]` is either `0` or `1`.

There are exactly two islands in `grid`.

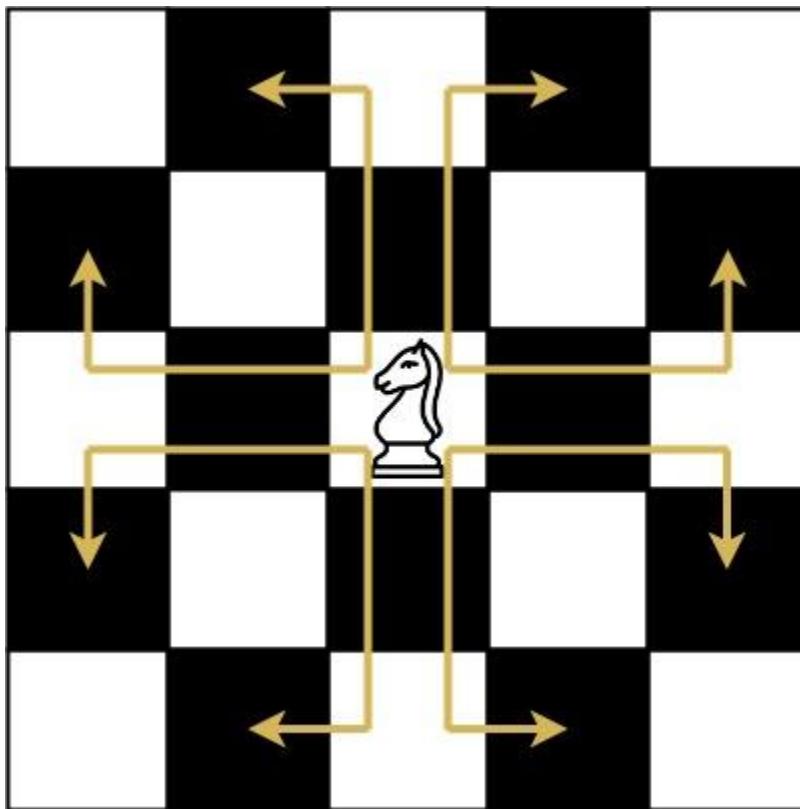
**935. Knight Dialer**

Medium

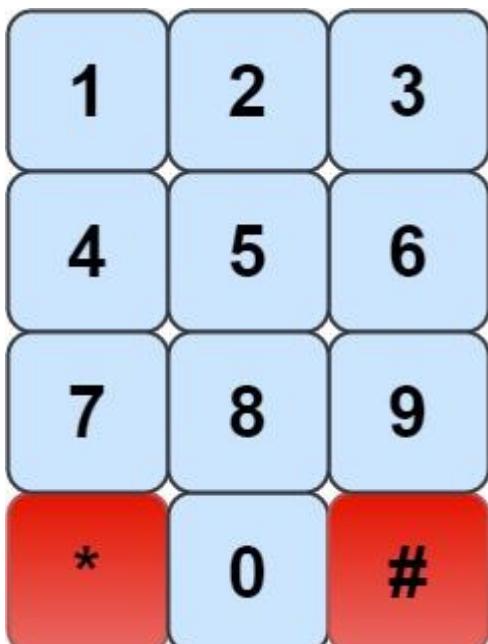
1757370Add to ListShare

The chess knight has a **unique movement**, it may move two squares vertically and one square horizontally, or two squares horizontally and one square vertically (with both forming the shape of an **L**). The possible movements of chess knight are shown in this diagram:

A chess knight can move as indicated in the chess diagram below:



We have a chess knight and a phone pad as shown below, the knight **can only stand on a numeric cell** (i.e. blue cell).



Given an integer  $n$ , return how many distinct phone numbers of length  $n$  we can dial.

You are allowed to place the knight **on any numeric cell** initially and then you should perform  $n - 1$  jumps to dial a number of length  $n$ . All jumps should be **valid** knight jumps.

As the answer may be very large, **return the answer modulo  $10^9 + 7$ .**

**Example 1:**

**Input:**  $n = 1$

**Output:** 10

**Explanation:** We need to dial a number of length 1, so placing the knight over any numeric cell of the 10 cells is sufficient.

**Example 2:**

**Input:**  $n = 2$

**Output:** 20

**Explanation:** All the valid numbers we can dial are [04, 06, 16, 18, 27, 29, 34, 38, 40, 43, 49, 60, 61, 67, 72, 76, 81, 83, 92, 94]

**Example 3:**

**Input:**  $n = 3131$

**Output:** 136006598

**Explanation:** Please take care of the mod.

**Constraints:**

$1 \leq n \leq 5000$

## 936. Stamping The Sequence

**Hard**

1410210 Add to List Share

You are given two strings `stamp` and `target`. Initially, there is a string `s` of length `target.length` with all `s[i] == '?'`.

In one turn, you can place `stamp` over `s` and replace every letter in the `s` with the corresponding letter from `stamp`.

For example, if `stamp = "abc"` and `target = "abcba"`, then `s` is `"?????"` initially. In one turn you can:

place `stamp` at index 0 of `s` to obtain `"abc??"`,

place `stamp` at index `1` of `s` to obtain "`?abc?`", or

place `stamp` at index `2` of `s` to obtain "`??abc`".

Note that `stamp` must be fully contained in the boundaries of `s` in order to stamp (i.e., you cannot place `stamp` at index `3` of `s`).

We want to convert `s` to `target` using **at most** `10 * target.length` turns.

Return *an array of the index of the left-most letter being stamped at each turn*. If we cannot obtain `target` from `s` within `10 * target.length` turns, return an empty array.

### Example 1:

**Input:** `stamp = "abc", target = "ababc"`

**Output:** `[0,2]`

**Explanation:** Initially `s = "?????"`.

- Place stamp at index `0` to get "`abc??`".
- Place stamp at index `2` to get "`ababc`".

`[1,0,2]` would also be accepted as an answer, as well as some other answers.

### Example 2:

**Input:** `stamp = "abca", target = "aabcaca"`

**Output:** `[3,0,1]`

**Explanation:** Initially `s = "???????"`.

- Place stamp at index `3` to get "`???abca`".
- Place stamp at index `0` to get "`abcabca`".
- Place stamp at index `1` to get "`aabcaca`".

### Constraints:

`1 <= stamp.length <= target.length <= 1000`

`stamp` and `target` consist of lowercase English letters.

## 937. Reorder Data in Log Files

Medium

18714173Add to ListShare

You are given an array of `logs`. Each log is a space-delimited string of words, where the first word is the **identifier**.

There are two types of logs:

**Letter-logs:** All words (except the identifier) consist of lowercase English letters.

**Digit-logs:** All words (except the identifier) consist of digits.

Reorder these logs so that:

The **letter-logs** come before all **digit-logs**.

The **letter-logs** are sorted lexicographically by their contents. If their contents are the same, then sort them lexicographically by their identifiers.

The **digit-logs** maintain their relative ordering.

Return *the final order of the logs*.

### Example 1:

**Input:** `logs = ["dig1 8 1 5 1", "let1 art can", "dig2 3 6", "let2 own kit dig", "let3 art zero"]`

**Output:** `["let1 art can", "let3 art zero", "let2 own kit dig", "dig1 8 1 5 1", "dig2 3 6"]`

### Explanation:

The letter-log contents are all different, so their ordering is "art can", "art zero", "own kit dig".

The digit-logs have a relative order of "dig1 8 1 5 1", "dig2 3 6".

### Example 2:

**Input:** `logs = ["a1 9 2 3 1", "g1 act car", "zo4 4 7", "ab1 off key dog", "a8 act zoo"]`

**Output:** `["g1 act car", "a8 act zoo", "ab1 off key dog", "a1 9 2 3 1", "zo4 4 7"]`

### Constraints:

```
1 <= logs.length <= 100
3 <= logs[i].length <= 100
```

All the tokens of `logs[i]` are separated by a **single** space.

`logs[i]` is guaranteed to have an identifier and at least one word after the identifier.

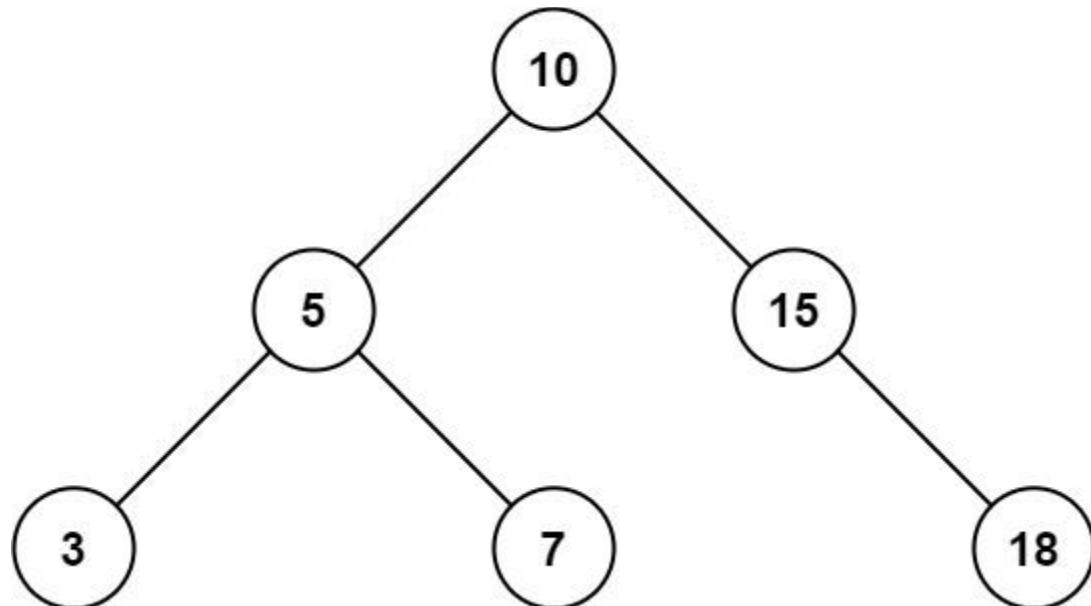
## 938. Range Sum of BST

Easy

463337Add to ListShare

Given the `root` node of a binary search tree and two integers `low` and `high`, return *the sum of values of all nodes with a value in the **inclusive** range* `[low, high]`.

**Example 1:**

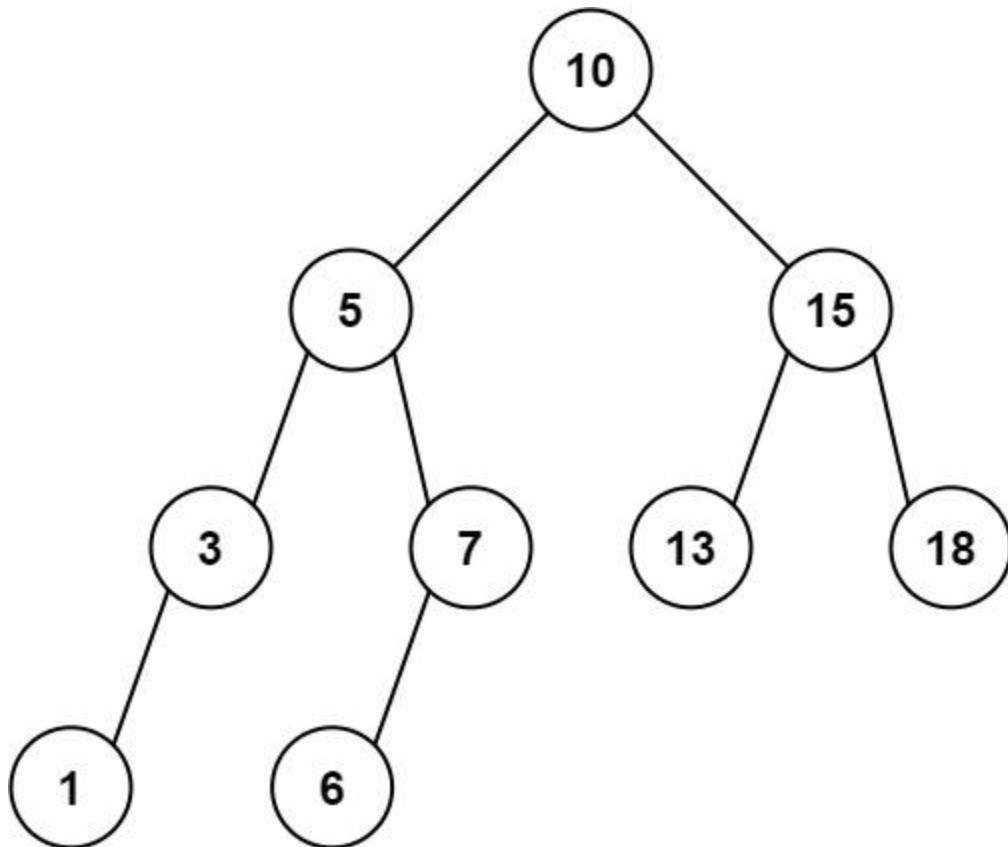


**Input:** `root = [10,5,15,3,7,null,18], low = 7, high = 15`

**Output:** 32

**Explanation:** Nodes 7, 10, and 15 are in the range `[7, 15]`.  $7 + 10 + 15 = 32$ .

**Example 2:**



**Input:** root = [10,5,15,3,7,13,18,1,null,6], low = 6, high = 10

**Output:** 23

**Explanation:** Nodes 6, 7, and 10 are in the range [6, 10].  $6 + 7 + 10 = 23$ .

#### Constraints:

The number of nodes in the tree is in the range  $[1, 2 * 10^4]$ .

$1 \leq \text{Node.val} \leq 10^5$

$1 \leq \text{low} \leq \text{high} \leq 10^5$

All `Node.val` are **unique**.

## 939. Minimum Area Rectangle

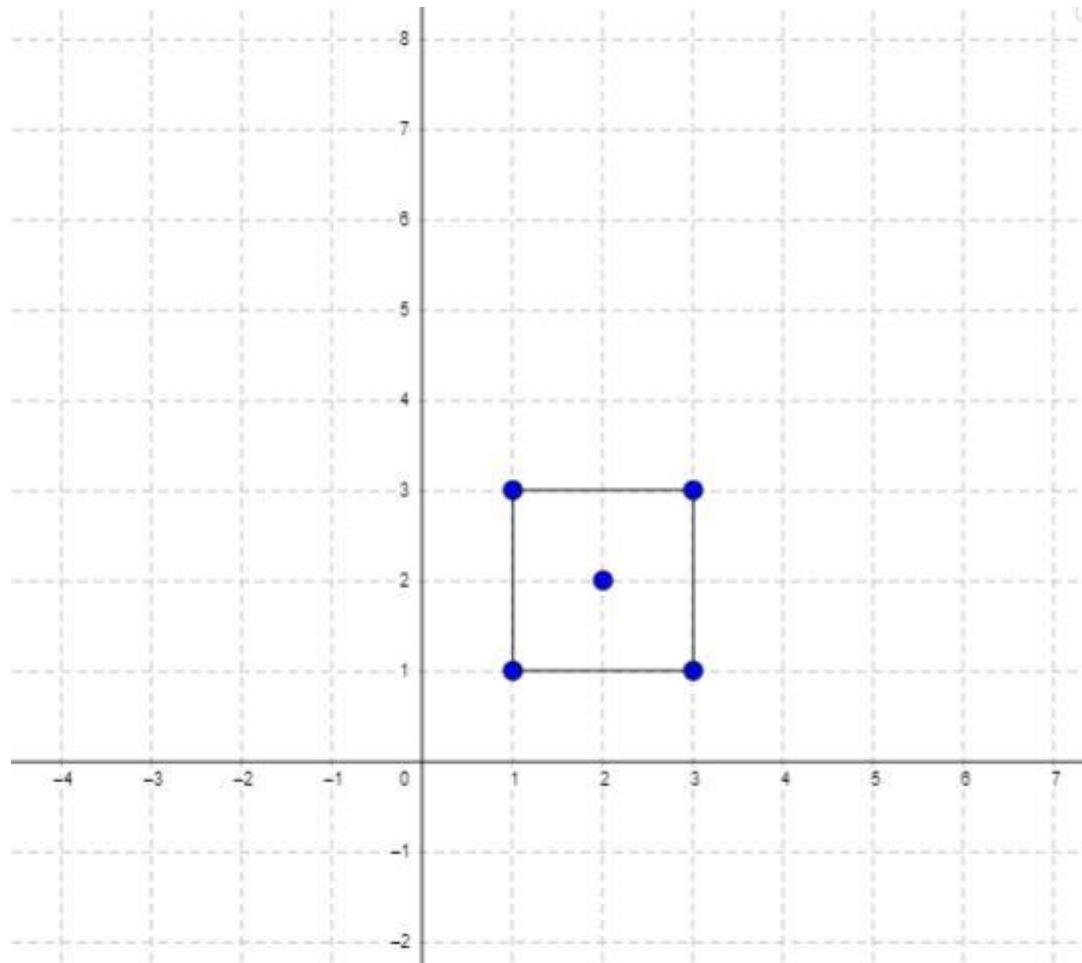
Medium

1624247Add to ListShare

You are given an array of points in the **X-Y** plane `points` where `points[i] = [xi, yi]`.

Return the minimum area of a rectangle formed from these points, with sides parallel to the X and Y axes. If there is not any such rectangle, return 0.

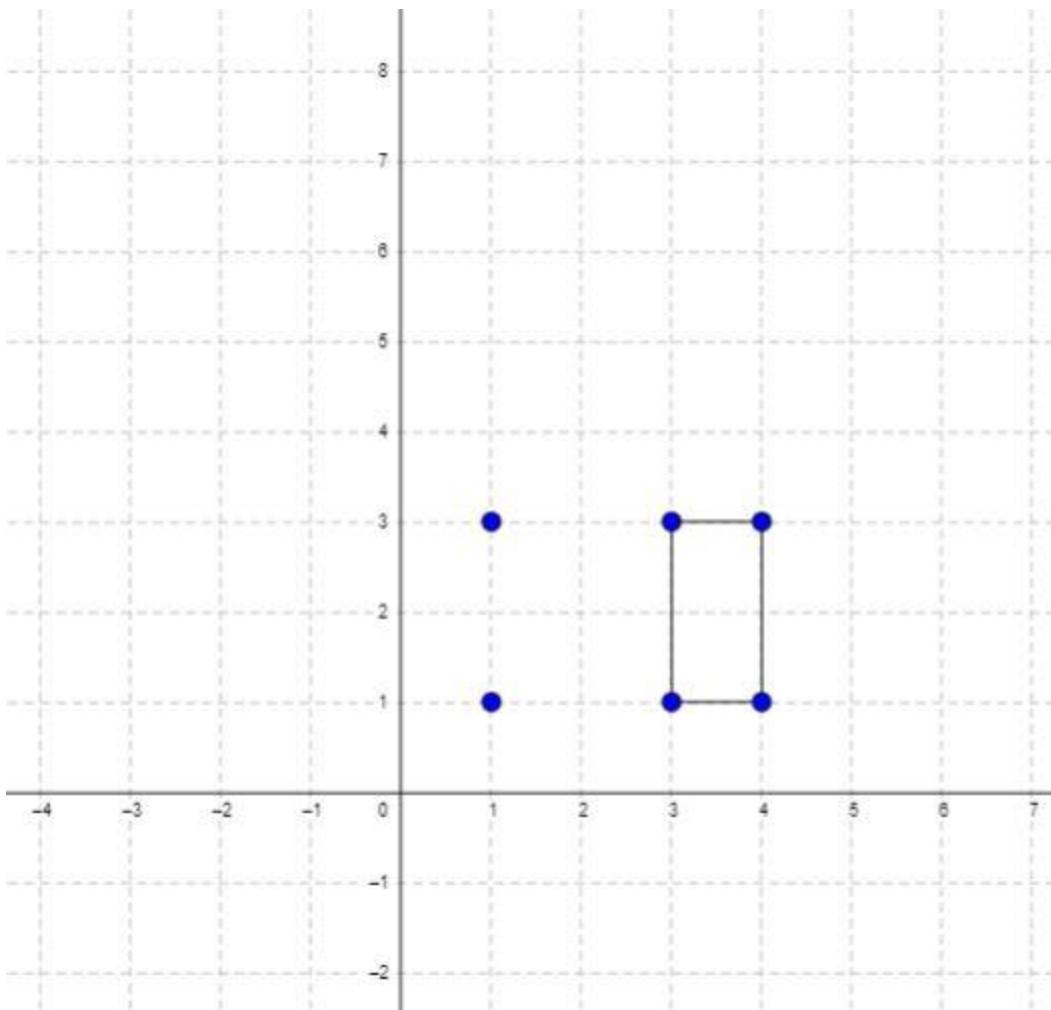
**Example 1:**



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

**Output:** 4

**Example 2:**



**Constraints:**

```
1 <= points.length <= 500
points[i].length == 2
0 <= xi, yi <= 4 * 104
```

All the given points are **unique**.

## 940. Distinct Subsequences II

Hard

129728Add to ListShare

Given a string  $s$ , return *the number of distinct non-empty subsequences* of  $s$ . Since the answer may be very large, return it **modulo**  $10^9 + 7$ .

A **subsequence** of a string is a new string that is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (i.e., "ace" is a subsequence of "abcde" while "aec" is not.

### Example 1:

**Input:**  $s = \text{"abc"}$

**Output:** 7

**Explanation:** The 7 distinct subsequences are "a", "b", "c", "ab", "ac", "bc", and "abc".

### Example 2:

**Input:**  $s = \text{"aba"}$

**Output:** 6

**Explanation:** The 6 distinct subsequences are "a", "b", "ab", "aa", "ba", and "aba".

### Example 3:

**Input:**  $s = \text{"aaa"}$

**Output:** 3

**Explanation:** The 3 distinct subsequences are "a", "aa" and "aaa".

### Constraints:

$1 \leq s.\text{length} \leq 2000$

$s$  consists of lowercase English letters.

## 941. Valid Mountain Array

Easy

2350150Add to ListShare

Given an array of integers  $\text{arr}$ , return *true* if and only if it is a valid mountain array.

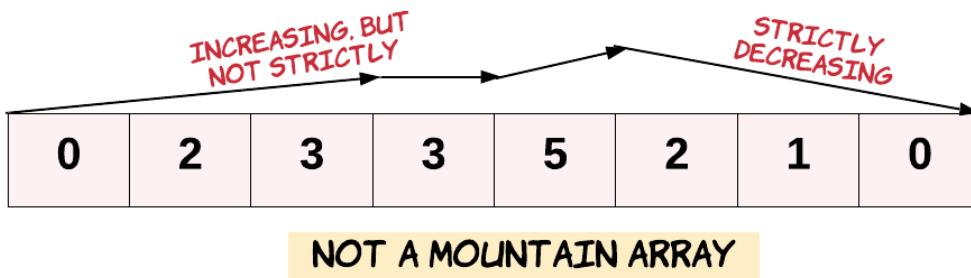
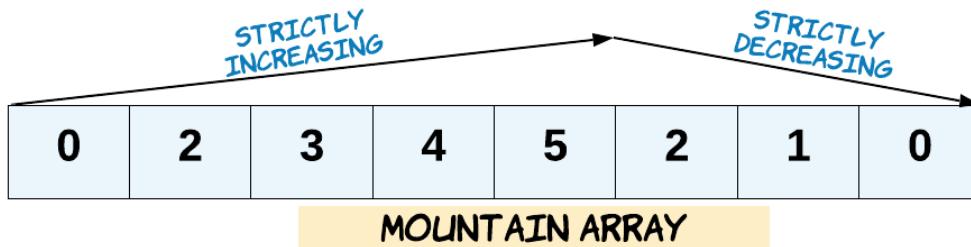
Recall that  $\text{arr}$  is a mountain array if and only if:

$\text{arr.length} \geq 3$

There exists some  $i$  with  $0 < i < \text{arr.length} - 1$  such that:

$\text{arr}[0] < \text{arr}[1] < \dots < \text{arr}[i - 1] < \text{arr}[i]$

$\text{arr}[i] > \text{arr}[i + 1] > \dots > \text{arr}[\text{arr.length} - 1]$



### Example 1:

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

**Output:** false

### Example 2:

**Input:** arr = [3,5,5]

**Output:** false

### Example 3:

**Input:** arr = [0,3,2,1]

**Output:** true

### Constraints:

$1 \leq \text{arr.length} \leq 10^4$

```
0 <= arr[i] <= 104
```

## 941. Valid Mountain Array

Easy

2350150 Add to List Share

Given an array of integers `arr`, return `true` if and only if it is a valid mountain array.

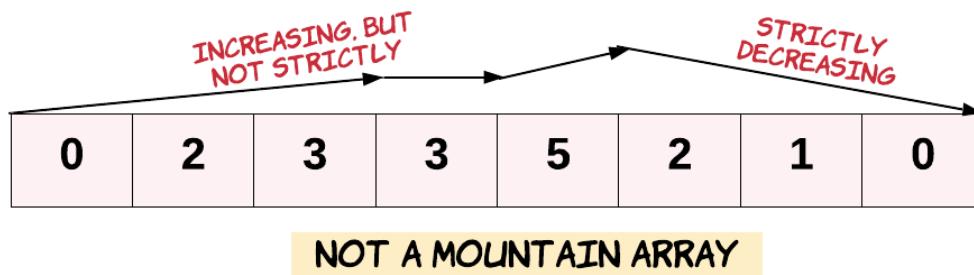
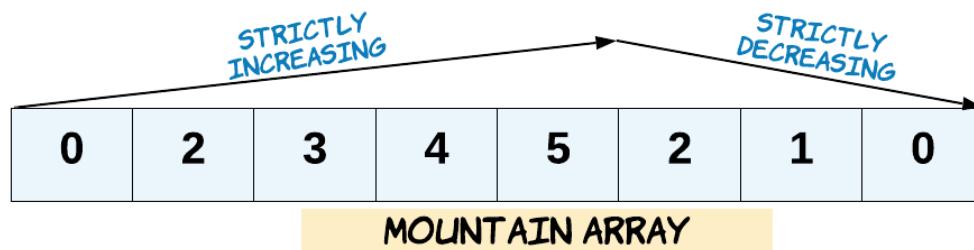
Recall that `arr` is a mountain array if and only if:

```
arr.length >= 3
```

There exists some `i` with  $0 < i < \text{arr.length} - 1$  such that:

```
arr[0] < arr[1] < ... < arr[i - 1] < arr[i]
```

```
arr[i] > arr[i + 1] > ... > arr[\text{arr.length} - 1]
```



**Example 1:**

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

**Output:** false

**Example 2:**

**Input:** arr = [3,5,5]

**Output:** false

**Example 3:****Input:** arr = [0,3,2,1]**Output:** true**Constraints:** $1 \leq \text{arr.length} \leq 10^4$  $0 \leq \text{arr}[i] \leq 10^4$ **943. Find the Shortest Superstring****Hard**

1167133Add to ListShare

Given an array of strings `words`, return *the smallest string that contains each string in words as a substring*. If there are multiple valid strings of the smallest length, return **any of them**.You may assume that no string in `words` is a substring of another string in `words`.**Example 1:****Input:** words = ["alex", "loves", "leetcode"]**Output:** "alexlovesleetcode"**Explanation:** All permutations of "alex", "loves", "leetcode" would also be accepted.**Example 2:****Input:** words = ["catg", "ctaagt", "gcta", "ttca", "atgcatc"]**Output:** "gctaagttcatgcac"**Constraints:** $1 \leq \text{words.length} \leq 12$  $1 \leq \text{words}[i].length \leq 20$ `words[i]` consists of lowercase English letters.All the strings of `words` are **unique**.

## 944. Delete Columns to Make Sorted

Easy

4052077Add to ListShare

You are given an array of `n` strings `strs`, all of the same length.

The strings can be arranged such that there is one on each line, making a grid. For example, `strs = ["abc", "bce", "cae"]` can be arranged as:

abc

bce

cae

You want to **delete** the columns that are **not sorted lexicographically**. In the above example (0-indexed), columns 0 ('a', 'b', 'c') and 2 ('c', 'e', 'e') are sorted while column 1 ('b', 'c', 'a') is not, so you would delete column 1.

Return *the number of columns that you will delete*.

### Example 1:

**Input:** `strs = ["cba", "daf", "ghi"]`

**Output:** 1

**Explanation:** The grid looks as follows:

cba

daf

ghi

Columns 0 and 2 are sorted, but column 1 is not, so you only need to delete 1 column.

### Example 2:

**Input:** `strs = ["a", "b"]`

**Output:** 0

**Explanation:** The grid looks as follows:

a

b

Column 0 is the only column and is sorted, so you will not delete any columns.

**Example 3:**

**Input:** `strs = ["zyx", "wvu", "tsr"]`

**Output:** 3

**Explanation:** The grid looks as follows:

zyx

wvu

tsr

All 3 columns are not sorted, so you will delete all 3.

**Constraints:**

`n == strs.length`

`1 <= n <= 100`

`1 <= strs[i].length <= 1000`

`strs[i]` consists of lowercase English letters.

## 945. Minimum Increment to Make Array Unique

**Medium**

124147Add to ListShare

You are given an integer array `nums`. In one move, you can pick an index `i` where `0 <= i < nums.length` and increment `nums[i]` by 1.

Return *the minimum number of moves to make every value in `nums` unique*.

The test cases are generated so that the answer fits in a 32-bit integer.

**Example 1:**

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

**Output:** 1

**Explanation:** After 1 move, the array could be [1, 2, 3].

**Example 2:**

**Input:** nums = [3,2,1,2,1,7]

**Output:** 6

**Explanation:** After 6 moves, the array could be [3, 4, 1, 2, 5, 7].

It can be shown with 5 or less moves that it is impossible for the array to have all unique values.

### Constraints:

1 <= nums.length <= 10<sup>5</sup>

0 <= nums[i] <= 10<sup>5</sup>

## 946. Validate Stack Sequences

Medium

379164Add to ListShare

Given two integer arrays `pushed` and `popped` each with distinct values, return `true` if this could have been the result of a sequence of push and pop operations on an initially empty stack, or `false` otherwise.

### Example 1:

**Input:** pushed = [1,2,3,4,5], popped = [4,5,3,2,1]

**Output:** true

**Explanation:** We might do the following sequence:

push(1), push(2), push(3), push(4),  
 pop() -> 4,  
 push(5),  
 pop() -> 5, pop() -> 3, pop() -> 2, pop() -> 1

### Example 2:

**Input:** pushed = [1,2,3,4,5], popped = [4,3,5,1,2]

**Output:** false

**Explanation:** 1 cannot be popped before 2.

**Constraints:**

`1 <= pushed.length <= 1000`

`0 <= pushed[i] <= 1000`

All the elements of `pushed` are **unique**.

`popped.length == pushed.length`

`popped` is a permutation of `pushed`.

## 947. Most Stones Removed with Same Row or Column

Medium

2923529Add to ListShare

On a 2D plane, we place `n` stones at some integer coordinate points. Each coordinate point may have at most one stone.

A stone can be removed if it shares either **the same row or the same column** as another stone that has not been removed.

Given an array `stones` of length `n` where `stones[i] = [xi, yi]` represents the location of the *i<sup>th</sup>* stone, return *the largest possible number of stones that can be removed*.

### Example 1:

**Input:** `stones = [[0,0],[0,1],[1,0],[1,2],[2,1],[2,2]]`

**Output:** 5

**Explanation:** One way to remove 5 stones is as follows:

1. Remove stone [2,2] because it shares the same row as [2,1].
2. Remove stone [2,1] because it shares the same column as [0,1].
3. Remove stone [1,2] because it shares the same row as [1,0].
4. Remove stone [1,0] because it shares the same column as [0,0].
5. Remove stone [0,1] because it shares the same row as [0,0].

Stone [0,0] cannot be removed since it does not share a row/column with another stone still on the plane.

**Example 2:**

**Input:** stones = [[0,0],[0,2],[1,1],[2,0],[2,2]]

**Output:** 3

**Explanation:** One way to make 3 moves is as follows:

1. Remove stone [2,2] because it shares the same row as [2,0].
2. Remove stone [2,0] because it shares the same column as [0,0].
3. Remove stone [0,2] because it shares the same row as [0,0].

Stones [0,0] and [1,1] cannot be removed since they do not share a row/column with another stone still on the plane.

**Example 3:**

**Input:** stones = [[0,0]]

**Output:** 0

**Explanation:** [0,0] is the only stone on the plane, so you cannot remove it.

**Constraints:**

1 <= stones.length <= 1000

0 <=  $x_i, y_i \leq 10^4$

No two stones are at the same coordinate point.

## 948. Bag of Tokens

Medium

2034406Add to ListShare

You have an initial **power** of `power`, an initial **score** of `0`, and a bag of `tokens` where `tokens[i]` is the value of the  $i^{\text{th}}$  token (0-indexed).

Your goal is to maximize your total **score** by potentially playing each token in one of two ways:

If your current **power** is at least `tokens[i]`, you may play the  $i^{\text{th}}$  token face up, losing `tokens[i]` **power** and gaining `1` **score**.

If your current **score** is at least `1`, you may play the  $i^{\text{th}}$  token face down, gaining `tokens[i]` **power** and losing `1` **score**.

Each token may be played **at most** once and **in any order**. You do **not** have to play all the tokens.

Return *the largest possible score* you can achieve after playing any number of tokens.

**Example 1:**

**Input:** tokens = [100], power = 50

**Output:** 0

**Explanation:** Playing the only token in the bag is impossible because you either have too little power or too little score.

**Example 2:**

**Input:** tokens = [100,200], power = 150

**Output:** 1

**Explanation:** Play the 0<sup>th</sup> token (100) face up, your power becomes 50 and score becomes 1.

There is no need to play the 1<sup>st</sup> token since you cannot play it face up to add to your score.

**Example 3:**

**Input:** tokens = [100,200,300,400], power = 200

**Output:** 2

**Explanation:** Play the tokens in this order to get a score of 2:

1. Play the 0<sup>th</sup> token (100) face up, your power becomes 100 and score becomes 1.
2. Play the 3<sup>rd</sup> token (400) face down, your power becomes 500 and score becomes 0.
3. Play the 1<sup>st</sup> token (200) face up, your power becomes 300 and score becomes 1.
4. Play the 2<sup>nd</sup> token (300) face up, your power becomes 0 and score becomes 2.

**Constraints:**

0 <= tokens.length <= 1000

0 <= tokens[i], power < 10<sup>4</sup>

## 949. Largest Time for Given Digits

**Medium**

590943Add to ListShare

Given an array `arr` of 4 digits, find the latest 24-hour time that can be made using each digit **exactly once**.

24-hour times are formatted as "`HH:MM`", where `HH` is between `00` and `23`, and `MM` is between `00` and `59`. The earliest 24-hour time is `00:00`, and the latest is `23:59`.

Return *the latest 24-hour time in "`HH:MM`" format*. If no valid time can be made, return an empty string.

**Example 1:**

**Input:** `arr = [1,2,3,4]`

**Output:** `"23:41"`

**Explanation:** The valid 24-hour times are `"12:34"`, `"12:43"`, `"13:24"`, `"13:42"`, `"14:23"`, `"14:32"`, `"21:34"`, `"21:43"`, `"23:14"`, and `"23:41"`. Of these times, `"23:41"` is the latest.

**Example 2:**

**Input:** `arr = [5,5,5,5]`

**Output:** `""`

**Explanation:** There are no valid 24-hour times as `"55:55"` is not valid.

**Constraints:**

`arr.length == 4`

`0 <= arr[i] <= 9`

**950. Reveal Cards In Increasing Order****Medium**

2049294Add to ListShare

You are given an integer array `deck`. There is a deck of cards where every card has a unique integer. The integer on the `ith` card is `deck[i]`.

You can order the deck in any order you want. Initially, all the cards start face down (unrevealed) in one deck.

You will do the following steps repeatedly until all cards are revealed:

Take the top card of the deck, reveal it, and take it out of the deck.

If there are still cards in the deck then put the next top card of the deck at the bottom of the deck.

If there are still unrevealed cards, go back to step 1. Otherwise, stop.

Return *an ordering of the deck that would reveal the cards in increasing order*.

**Note** that the first entry in the answer is considered to be the top of the deck.

### Example 1:

**Input:** deck = [17,13,11,2,3,5,7]

**Output:** [2,13,3,11,5,17,7]

#### Explanation:

We get the deck in the order [17,13,11,2,3,5,7] (this order does not matter), and reorder it.

After reordering, the deck starts as [2,13,3,11,5,17,7], where 2 is the top of the deck.

We reveal 2, and move 13 to the bottom. The deck is now [3,11,5,17,7,13].

We reveal 3, and move 11 to the bottom. The deck is now [5,17,7,13,11].

We reveal 5, and move 17 to the bottom. The deck is now [7,13,11,17].

We reveal 7, and move 13 to the bottom. The deck is now [11,17,13].

We reveal 11, and move 17 to the bottom. The deck is now [13,17].

We reveal 13, and move 17 to the bottom. The deck is now [17].

We reveal 17.

Since all the cards revealed are in increasing order, the answer is correct.

### Example 2:

**Input:** deck = [1,1000]

**Output:** [1,1000]

#### Constraints:

```
1 <= deck.length <= 1000
```

```
1 <= deck[i] <= 106
```

All the values of `deck` are **unique**.

## 951. Flip Equivalent Binary Trees

Medium

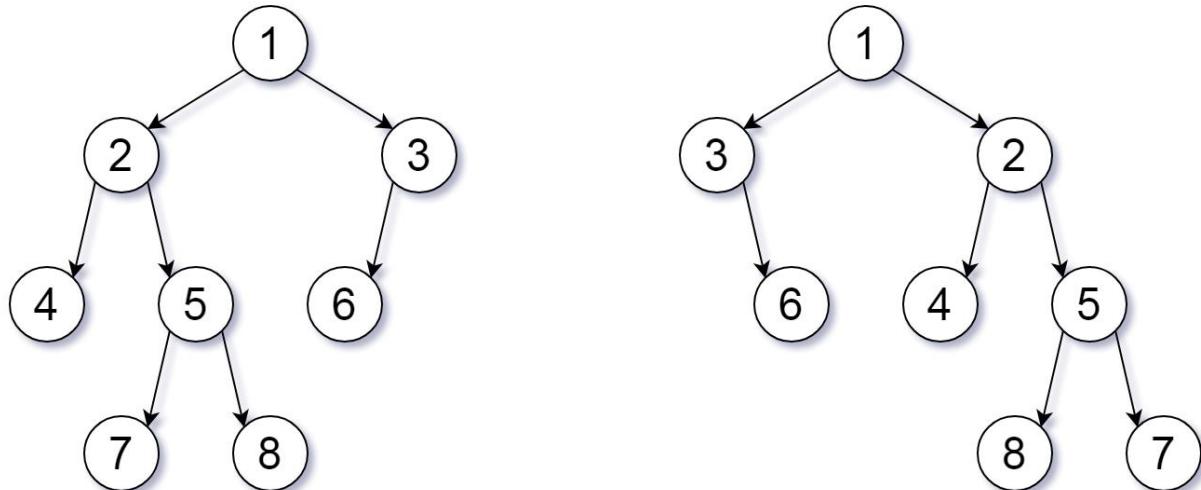
184180Add to ListShare

For a binary tree **T**, we can define a **flip operation** as follows: choose any node, and swap the left and right child subtrees.

A binary tree **X** is *flip equivalent* to a binary tree **Y** if and only if we can make **X** equal to **Y** after some number of flip operations.

Given the roots of two binary trees `root1` and `root2`, return `true` if the two trees are flip equivalent or `false` otherwise.

**Example 1:**



**Input:** `root1 = [1,2,3,4,5,6,null,null,null,7,8]`, `root2 = [1,3,2,null,6,4,5,null,null,null,8,7]`

**Output:** `true`

**Explanation:** We flipped at nodes with values 1, 3, and 5.

**Example 2:**

**Input:** `root1 = []`, `root2 = []`

**Output:** true

**Example 3:**

**Input:** root1 = [], root2 = [1]

**Output:** false

**Constraints:**

The number of nodes in each tree is in the range [0, 100].

Each tree will have **unique node values** in the range [0, 99].

## 952. Largest Component Size by Common Factor

Hard

137387Add to ListShare

You are given an integer array of unique positive integers `nums`. Consider the following graph:

There are `nums.length` nodes, labeled `nums[0]` to `nums[nums.length - 1]`,

There is an undirected edge between `nums[i]` and `nums[j]` if `nums[i]` and `nums[j]` share a common factor greater than 1.

Return *the size of the largest connected component in the graph*.

**Example 1:**



**Input:** `nums = [4,6,15,35]`

**Output:** 4

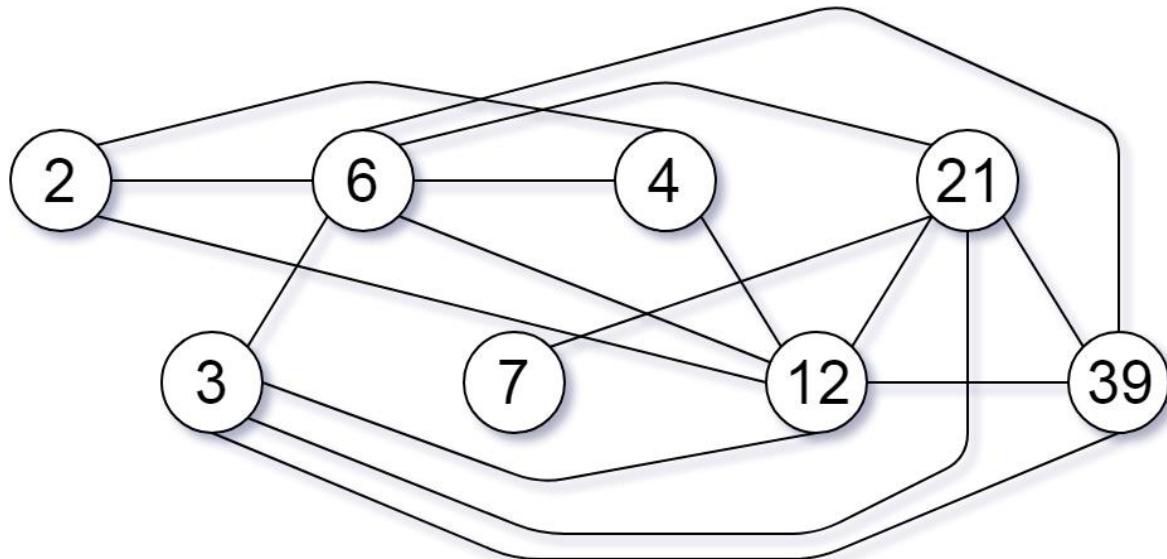
**Example 2:**



**Input:** nums = [20,50,9,63]

**Output:** 2

**Example 3:**



**Input:** nums = [2,3,6,7,4,12,21,39]

**Output:** 8

**Constraints:**

1 <= nums.length <= 2 \* 10<sup>4</sup>

1 <= nums[i] <= 10<sup>5</sup>

All the values of nums are **unique**.

## 953. Verifying an Alien Dictionary

Easy

3092986Add to ListShare

In an alien language, surprisingly, they also use English lowercase letters, but possibly in a different `order`. The `order` of the alphabet is some permutation of lowercase letters.

Given a sequence of `words` written in the alien language, and the `order` of the alphabet, return `true` if and only if the given `words` are sorted lexicographically in this alien language.

**Example 1:**

**Input:** `words = ["hello", "leetcode"]`, `order = "hlabcd...xyz"`

**Output:** `true`

**Explanation:** As 'h' comes before 'l' in this language, then the sequence is sorted.

**Example 2:**

**Input:** `words = ["word", "world", "row"]`, `order = "worldab...xyz"`

**Output:** `false`

**Explanation:** As 'd' comes after 'l' in this language, then `words[0] > words[1]`, hence the sequence is unsorted.

**Example 3:**

**Input:** `words = ["apple", "app"]`, `order = "abcde...xyz"`

**Output:** `false`

**Explanation:** The first three characters "app" match, and the second string is shorter (in size.) According to lexicographical rules "apple" > "app", because 'l' > 'Ø', where 'Ø' is defined as the blank character which is less than any other character ([More info](#)).

**Constraints:**

`1 <= words.length <= 100`

`1 <= words[i].length <= 20`

`order.length == 26`

All characters in `words[i]` and `order` are English lowercase letters.

## 954. Array of Doubled Pairs

Medium

1260130Add to ListShare

Given an integer array of even length `arr`, return `true` if it is possible to reorder `arr` such that  $arr[2 * i + 1] = 2 * arr[2 * i]$  for every  $0 \leq i < \text{len}(arr) / 2$ , or `false` otherwise.

**Example 1:**

**Input:** `arr = [3,1,3,6]`

**Output:** `false`

**Example 2:**

**Input:** `arr = [2,1,2,6]`

**Output:** `false`

**Example 3:**

**Input:** `arr = [4,-2,2,-4]`

**Output:** `true`

**Explanation:** We can take two groups,  $[-2, -4]$  and  $[2, 4]$  to form  $[-2, -4, 2, 4]$  or  $[2, 4, -2, -4]$ .

**Constraints:**

$2 \leq \text{arr.length} \leq 3 * 10^4$

`arr.length` is even.

$-10^5 \leq \text{arr}[i] \leq 10^5$

## 955. Delete Columns to Make Sorted II

Medium

51876Add to ListShare

You are given an array of `n` strings `strs`, all of the same length.

We may choose any deletion indices, and we delete all the characters in those indices for each string.

For example, if we have `strs = ["abcdef", "uvwxyz"]` and deletion indices `{0, 2, 3}`, then the final array after deletions is `["bef", "vyz"]`.

Suppose we chose a set of deletion indices `answer` such that after deletions, the final array has its elements in **lexicographic** order (i.e., `strs[0] <= strs[1] <= strs[2] <= ... <= strs[n - 1]`). Return the *minimum possible value of* `answer.length`.

### Example 1:

**Input:** `strs = ["ca", "bb", "ac"]`

**Output:** 1

**Explanation:**

After deleting the first column, `strs = ["a", "b", "c"]`.

Now `strs` is in lexicographic order (i.e. `strs[0] <= strs[1] <= strs[2]`).

We require at least 1 deletion since initially `strs` was not in lexicographic order, so the answer is 1.

### Example 2:

**Input:** `strs = ["xc", "yb", "za"]`

**Output:** 0

**Explanation:**

`strs` is already in lexicographic order, so we do not need to delete anything.

Note that the rows of `strs` are not necessarily in lexicographic order:

i.e., it is NOT necessarily true that `(strs[0][0] <= strs[0][1] <= ...)`

### Example 3:

**Input:** `strs = ["zyx", "wvu", "tsr"]`

**Output:** 3

**Explanation:** We have to delete every column.

### Constraints:

```

n == strs.length
1 <= n <= 100
1 <= strs[i].length <= 100

```

`strs[i]` consists of lowercase English letters.

## 956. Tallest Billboard

Hard

77727Add to ListShare

You are installing a billboard and want it to have the largest height. The billboard will have two steel supports, one on each side. Each steel support must be an equal height.

You are given a collection of `rods` that can be welded together. For example, if you have rods of lengths `1`, `2`, and `3`, you can weld them together to make a support of length `6`.

Return *the largest possible height of your billboard installation*. If you cannot support the billboard, return `0`.

### Example 1:

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

**Output:** 6

**Explanation:** We have two disjoint subsets  $\{1,2,3\}$  and  $\{6\}$ , which have the same sum = 6.

### Example 2:

**Input:** rods = [1,2,3,4,5,6]

**Output:** 10

**Explanation:** We have two disjoint subsets  $\{2,3,5\}$  and  $\{4,6\}$ , which have the same sum = 10.

### Example 3:

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

**Output:** 0

**Explanation:** The billboard cannot be supported, so we return 0.

### Constraints:

`1 <= rods.length <= 20`

`1 <= rods[i] <= 1000`

```
sum(rods[i]) <= 5000
```

## 957. Prison Cells After N Days

Medium

13231606Add to ListShare

There are 8 prison cells in a row and each cell is either occupied or vacant.

Each day, whether the cell is occupied or vacant changes according to the following rules:

If a cell has two adjacent neighbors that are both occupied or both vacant, then the cell becomes occupied.

Otherwise, it becomes vacant.

**Note** that because the prison is a row, the first and the last cells in the row can't have two adjacent neighbors.

You are given an integer array `cells` where `cells[i] == 1` if the `ith` cell is occupied and `cells[i] == 0` if the `ith` cell is vacant, and you are given an integer `n`.

Return the state of the prison after `n` days (i.e., `n` such changes described above).

### Example 1:

**Input:** `cells = [0,1,0,1,1,0,0,1], n = 7`

**Output:** `[0,0,1,1,0,0,0,0]`

**Explanation:** The following table summarizes the state of the prison on each day:

Day 0: `[0, 1, 0, 1, 1, 0, 0, 1]`

Day 1: `[0, 1, 1, 0, 0, 0, 0, 0]`

Day 2: `[0, 0, 0, 0, 1, 1, 0, 0]`

Day 3: `[0, 1, 1, 0, 0, 1, 0, 0]`

Day 4: `[0, 0, 0, 0, 0, 1, 0, 0]`

Day 5: `[0, 1, 1, 1, 0, 1, 0, 0]`

Day 6: `[0, 0, 1, 0, 1, 1, 0, 0]`

Day 7: `[0, 0, 1, 1, 0, 0, 0, 0]`

### Example 2:

**Input:** cells = [1,0,0,1,0,0,1,0], n = 1000000000

**Output:** [0,0,1,1,1,1,1,0]

**Constraints:**

cells.length == 8

cells[i] is either 0 or 1.

1 <= n <= 10<sup>9</sup>

## 958. Check Completeness of a Binary Tree

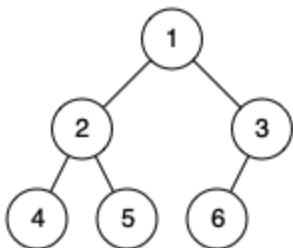
Medium

221231Add to ListShare

Given the `root` of a binary tree, determine if it is a *complete binary tree*.

In a complete binary tree, every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible. It can have between 1 and  $2^h$  nodes inclusive at the last level `h`.

**Example 1:**

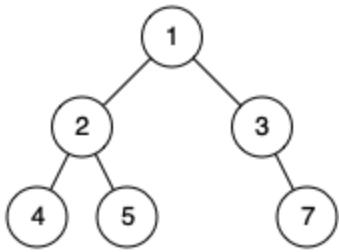


**Input:** root = [1,2,3,4,5,6]

**Output:** true

**Explanation:** Every level before the last is full (ie. levels with node-values {1} and {2, 3}), and all nodes in the last level ({4, 5, 6}) are as far left as possible.

**Example 2:**



**Input:** root = [1,2,3,4,5,null,7]

**Output:** false

**Explanation:** The node with value 7 isn't as far left as possible.

#### Constraints:

The number of nodes in the tree is in the range [1, 100].

`1 <= Node.val <= 1000`

## 959. Regions Cut By Slashes

Medium

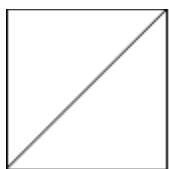
2439466Add to ListShare

An  $n \times n$  grid is composed of  $1 \times 1$  squares where each  $1 \times 1$  square consists of a `'/'`, `'\\'`, or blank space `' '`. These characters divide the square into contiguous regions.

Given the grid `grid` represented as a string array, return *the number of regions*.

Note that backslash characters are escaped, so a `'\\'` is represented as `'\\\'`.

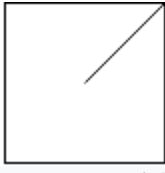
#### Example 1:



**Input:** grid = [" /","/ "]

**Output:** 2

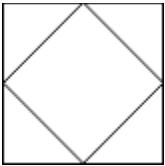
#### Example 2:



**Input:** grid = [" /", " \"]

**Output:** 1

**Example 3:**



**Input:** grid = ["/\\\", \"\\\/\""]

**Output:** 5

**Explanation:** Recall that because \ characters are escaped, "\\/" refers to \/, and "/\\\" refers to \\.\\\".

**Constraints:**

`n == grid.length == grid[i].length`

`1 <= n <= 30`

`grid[i][j]` is either `'/'`, `'\\'`, or `' '`.

## 960. Delete Columns to Make Sorted III

Hard

47111Add to ListShare

You are given an array of `n` strings `strs`, all of the same length.

We may choose any deletion indices, and we delete all the characters in those indices for each string.

For example, if we have `strs = ["abcdef", "uvwxyz"]` and deletion indices `{0, 2, 3}`, then the final array after deletions is `["bef", "vyz"]`.

Suppose we chose a set of deletion indices `answer` such that after deletions, the final array has **every string (row) in lexicographic order**. (i.e., `(strs[0][0] <= strs[0][1] <= ... <= strs[0][strs[0].length - 1])`, and `(strs[1][0] <= strs[1][1] <= ... <= strs[1][strs[1].length - 1])`, and so on). Return *the minimum possible value of answer.length*.

**Example 1:**

**Input:** strs = ["babca", "bbazb"]

**Output:** 3

**Explanation:** After deleting columns 0, 1, and 4, the final array is strs = ["bc", "az"].

Both these rows are individually in lexicographic order (ie.  $\text{strs}[0][0] \leq \text{strs}[0][1]$  and  $\text{strs}[1][0] \leq \text{strs}[1][1]$ ).

Note that  $\text{strs}[0] > \text{strs}[1]$  - the array strs is not necessarily in lexicographic order.

**Example 2:**

**Input:** strs = ["edcba"]

**Output:** 4

**Explanation:** If we delete less than 4 columns, the only row will not be lexicographically sorted.

**Example 3:**

**Input:** strs = ["ghi", "def", "abc"]

**Output:** 0

**Explanation:** All rows are already lexicographically sorted.

**Constraints:**

`n == strs.length`

`1 <= n <= 100`

`1 <= strs[i].length <= 100`

`strs[i]` consists of lowercase English letters.

## 961. N-Repeated Element in Size 2N Array

Easy

1016307Add to ListShare

You are given an integer array `nums` with the following properties:

`nums.length == 2 * n.`

`nums` contains  $n + 1$  **unique** elements.

Exactly one element of `nums` is repeated  $n$  times.

Return *the element that is repeated  $n$  times*.

### Example 1:

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

**Output:** 3

### Example 2:

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

**Output:** 2

### Example 3:

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

**Output:** 5

### Constraints:

$2 \leq n \leq 5000$

`nums.length == 2 * n`

$0 \leq \text{nums}[i] \leq 10^4$

`nums` contains  $n + 1$  **unique** elements and one of them is repeated exactly  $n$  times.

## 962. Maximum Width Ramp

Medium

131339Add to ListShare

A **ramp** in an integer array `nums` is a pair  $(i, j)$  for which  $i < j$  and `nums[i] <= nums[j]`.

The **width** of such a ramp is  $j - i$ .

Given an integer array `nums`, return *the maximum width of a ramp* in `nums`. If there is no **ramp** in `nums`, return 0.

**Example 1:**

**Input:** `nums` = [6,0,8,2,1,5]

**Output:** 4

**Explanation:** The maximum width ramp is achieved at  $(i, j) = (1, 5)$ :  $nums[1] = 0$  and  $nums[5] = 5$ .

**Example 2:**

**Input:** `nums` = [9,8,1,0,1,9,4,0,4,1]

**Output:** 7

**Explanation:** The maximum width ramp is achieved at  $(i, j) = (2, 9)$ :  $nums[2] = 1$  and  $nums[9] = 1$ .

**Constraints:**

`2 <= nums.length <= 5 * 104`

`0 <= nums[i] <= 5 * 104`

## 963. Minimum Area Rectangle II

Medium

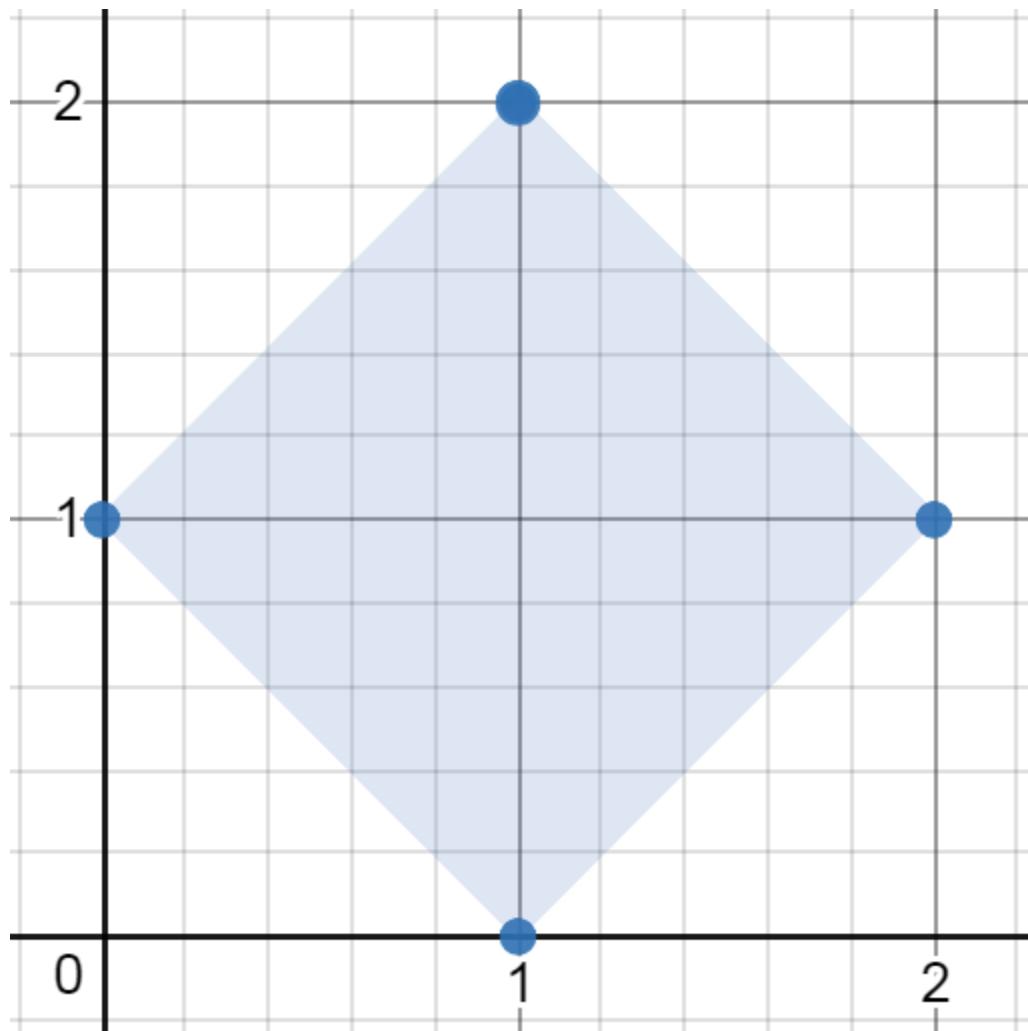
327424Add to ListShare

You are given an array of points in the **X-Y** plane `points` where `points[i] = [xi, yi]`.

Return *the minimum area of any rectangle formed from these points, with sides **not necessarily parallel** to the X and Y axes*. If there is not any such rectangle, return 0.

Answers within  $10^{-5}$  of the actual answer will be accepted.

**Example 1:**

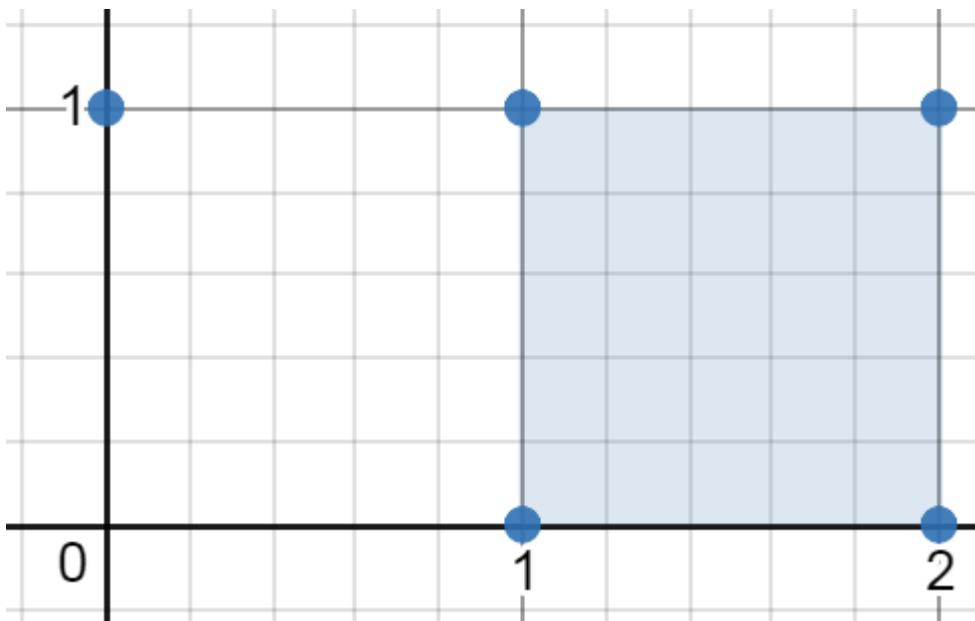


**Input:** points = [[1,2],[2,1],[1,0],[0,1]]

**Output:** 2.00000

**Explanation:** The minimum area rectangle occurs at [1,2],[2,1],[1,0],[0,1], with an area of 2.

**Example 2:**

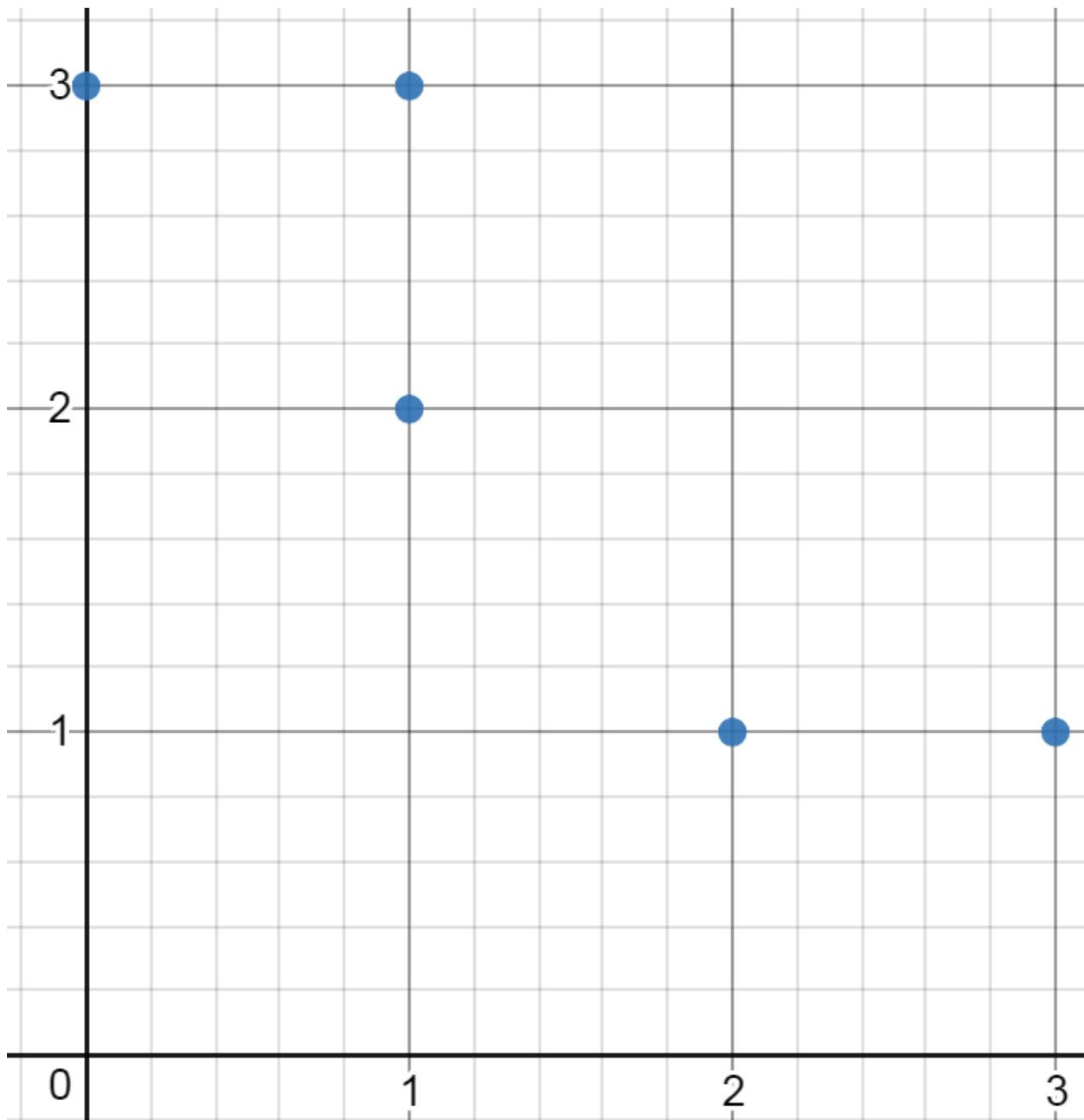


**Input:** points = [[0,1],[2,1],[1,1],[1,0],[2,0]]

**Output:** 1.00000

**Explanation:** The minimum area rectangle occurs at [1,0],[1,1],[2,1],[2,0], with an area of 1.

**Example 3:**

**Constraints:**

```
1 <= points.length <= 50
```

```
points[i].length == 2
0 <= xi, yi <= 4 * 104
```

All the given points are **unique**.

## 964. Least Operators to Express Number

Hard

27466Add to ListShare

Given a single positive integer  $x$ , we will write an expression of the form  $x \text{ (op1)} x \text{ (op2)} x \text{ (op3)} \dots$  where each operator  $\text{op1}$ ,  $\text{op2}$ , etc. is either addition, subtraction, multiplication, or division (+, -, \*, or /). For example, with  $x = 3$ , we might write  $3 * 3 / 3 + 3 - 3$  which is a value of 3.

When writing such an expression, we adhere to the following conventions:

The division operator (/) returns rational numbers.

There are no parentheses placed anywhere.

We use the usual order of operations: multiplication and division happen before addition and subtraction.

It is not allowed to use the unary negation operator (-). For example, " $x - x$ " is a valid expression as it only uses subtraction, but " $-x + x$ " is not because it uses negation.

We would like to write an expression with the least number of operators such that the expression equals the given `target`. Return the least number of operators used.

### Example 1:

**Input:**  $x = 3$ , `target` = 19

**Output:** 5

**Explanation:**  $3 * 3 + 3 * 3 + 3 / 3$ .

The expression contains 5 operations.

### Example 2:

**Input:**  $x = 5$ , `target` = 501

**Output:** 8

**Explanation:**  $5 * 5 * 5 * 5 - 5 * 5 * 5 + 5 / 5$ .

The expression contains 8 operations.

**Example 3:**

**Input:** `x = 100, target = 1000000000`

**Output:** 3

**Explanation:** `100 * 100 * 100 * 100.`

The expression contains 3 operations.

**Constraints:**

`2 <= x <= 100`

`1 <= target <= 2 * 108`

## 965. Univalued Binary Tree

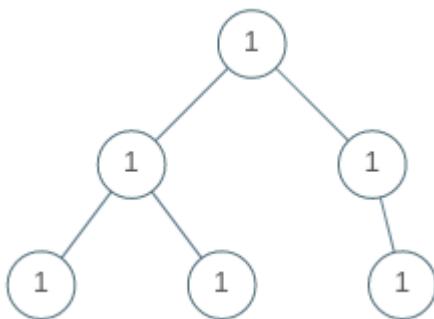
Easy

146257Add to ListShare

A binary tree is **uni-valued** if every node in the tree has the same value.

Given the `root` of a binary tree, return `true` if the given tree is **uni-valued**, or `false` otherwise.

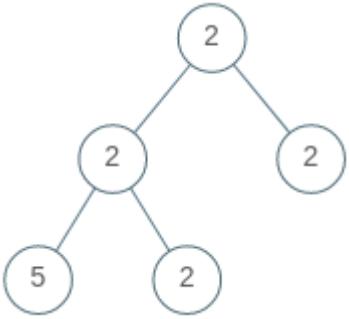
**Example 1:**



**Input:** `root = [1,1,1,1,1,null,1]`

**Output:** `true`

**Example 2:**



**Input:** root = [2,2,2,5,2]

**Output:** false

### Constraints:

The number of nodes in the tree is in the range [1, 100].

0 <= Node.val < 100

## 966. Vowel Spellchecker

Medium

364754Add to ListShare

Given a `wordlist`, we want to implement a spellchecker that converts a query word into a correct word.

For a given `query` word, the spell checker handles two categories of spelling mistakes:

Capitalization: If the query matches a word in the wordlist (**case-insensitive**), then the query word is returned with the same case as the case in the wordlist.

Example: `wordlist = ["yellow"], query = "Yellow": correct = "yellow"`

Example: `wordlist = ["Yellow"], query = "yellow": correct = "Yellow"`

Example: `wordlist = ["yellow"], query = "yellow": correct = "yellow"`

Vowel Errors: If after replacing the vowels ('a', 'e', 'i', 'o', 'u') of the query word with any vowel individually, it matches a word in the wordlist (**case-insensitive**), then the query word is returned with the same case as the match in the wordlist.

Example: `wordlist = ["Yellow"], query = "yellow": correct = "Yellow"`

Example: `wordlist = ["Yellow"], query = "yeellow": correct = "" (no match)`

Example: `wordlist = ["YellOw"], query = "yllw": correct = ""` (no match)

In addition, the spell checker operates under the following precedence rules:

When the query exactly matches a word in the wordlist (**case-sensitive**), you should return the same word back.

When the query matches a word up to capitalization, you should return the first such match in the wordlist.

When the query matches a word up to vowel errors, you should return the first such match in the wordlist.

If the query has no matches in the wordlist, you should return the empty string.

Given some `queries`, return a list of words `answer`, where `answer[i]` is the correct word for `query = queries[i]`.

### Example 1:

**Input:** `wordlist = ["KiTe", "kite", "hare", "Hare"], queries = ["kite", "Kite", "KiTe", "Hare", "HARE", "Hear", "hear", "keti", "keet", "keto"]`

**Output:** `["kite", "KiTe", "KiTe", "Hare", "hare", "", "", "KiTe", "", "KiTe"]`

### Example 2:

**Input:** `wordlist = ["yellow"], queries = ["YellowW"]`

**Output:** `["yellow"]`

### Constraints:

`1 <= wordlist.length, queries.length <= 5000`

`1 <= wordlist[i].length, queries[i].length <= 7`

`wordlist[i]` and `queries[i]` consist only of only English letters.

## 967. Numbers With Same Consecutive Differences

Medium

2500184Add to ListShare

Return all **non-negative** integers of length `n` such that the absolute difference between every two consecutive digits is `k`.

Note that **every** number in the answer **must not** have leading zeros. For example, `01` has one leading zero and is invalid.

You may return the answer in **any order**.

### Example 1:

**Input:** `n = 3, k = 7`

**Output:** `[181, 292, 707, 818, 929]`

**Explanation:** Note that `070` is not a valid number, because it has leading zeroes.

### Example 2:

**Input:** `n = 2, k = 1`

**Output:** `[10, 12, 21, 23, 32, 34, 43, 45, 54, 56, 65, 67, 76, 78, 87, 89, 98]`

### Constraints:

`2 <= n <= 9`

`0 <= k <= 9`

## 968. Binary Tree Cameras

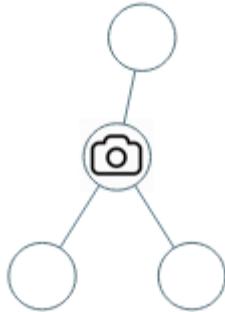
Hard

445856Add to ListShare

You are given the `root` of a binary tree. We install cameras on the tree nodes where each camera at a node can monitor its parent, itself, and its immediate children.

Return *the minimum number of cameras needed to monitor all nodes of the tree*.

### Example 1:

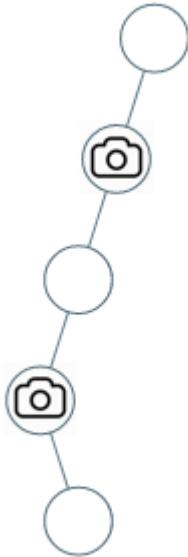


**Input:** root = [0,0,null,0,0]

**Output:** 1

**Explanation:** One camera is enough to monitor all nodes if placed as shown.

**Example 2:**



**Input:** root = [0,0,null,0,null,0,null,null,0]

**Output:** 2

**Explanation:** At least two cameras are needed to monitor all nodes of the tree. The above image shows one of the valid configurations of camera placement.

**Constraints:**

The number of nodes in the tree is in the range [1, 1000].

Node.val == 0

## 969. Pancake Sorting

Medium

12021309Add to ListShare

Given an array of integers `arr`, sort the array by performing a series of **pancake flips**.

In one pancake flip we do the following steps:

Choose an integer `k` where `1 <= k <= arr.length`.

Reverse the sub-array `arr[0...k-1]` (**0-indexed**).

For example, if `arr = [3, 2, 1, 4]` and we performed a pancake flip choosing `k = 3`, we reverse the sub-array `[3, 2, 1]`, so `arr = [1, 2, 3, 4]` after the pancake flip at `k = 3`.

Return *an array of the k-values corresponding to a sequence of pancake flips that sort arr*. Any valid answer that sorts the array within `10 * arr.length` flips will be judged as correct.

### Example 1:

**Input:** `arr = [3, 2, 4, 1]`

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

#### Explanation:

We perform 4 pancake flips, with `k` values 4, 2, 4, and 3.

Starting state: `arr = [3, 2, 4, 1]`

After 1st flip (`k = 4`): `arr = [1, 4, 2, 3]`

After 2nd flip (`k = 2`): `arr = [4, 1, 2, 3]`

After 3rd flip (`k = 4`): `arr = [3, 2, 1, 4]`

After 4th flip (`k = 3`): `arr = [1, 2, 3, 4]`, which is sorted.

### Example 2:

**Input:** `arr = [1, 2, 3]`

**Output:** `[]`

**Explanation:** The input is already sorted, so there is no need to flip anything.

Note that other answers, such as `[3, 3]`, would also be accepted.

**Constraints:**

```
1 <= arr.length <= 100
```

```
1 <= arr[i] <= arr.length
```

All integers in `arr` are unique (i.e. `arr` is a permutation of the integers from `1` to `arr.length`).

## 970. Powerful Integers

Medium

28267Add to ListShare

Given three integers `x`, `y`, and `bound`, return a list of all the **powerful integers** that have a value less than or equal to `bound`.

An integer is **powerful** if it can be represented as  $x^i + y^j$  for some integers `i >= 0` and `j >= 0`.

You may return the answer in **any order**. In your answer, each value should occur **at most once**.

### Example 1:

**Input:** `x = 2, y = 3, bound = 10`

**Output:** `[2,3,4,5,7,9,10]`

### Explanation:

$2 = 2^0 + 3^0$

$3 = 2^1 + 3^0$

$4 = 2^0 + 3^1$

$5 = 2^1 + 3^1$

$7 = 2^2 + 3^1$

$9 = 2^3 + 3^0$

$10 = 2^0 + 3^2$

### Example 2:

**Input:** `x = 3, y = 5, bound = 15`

**Output:** `[2,4,6,8,10,14]`

**Constraints:**

```
1 <= x, y <= 100
```

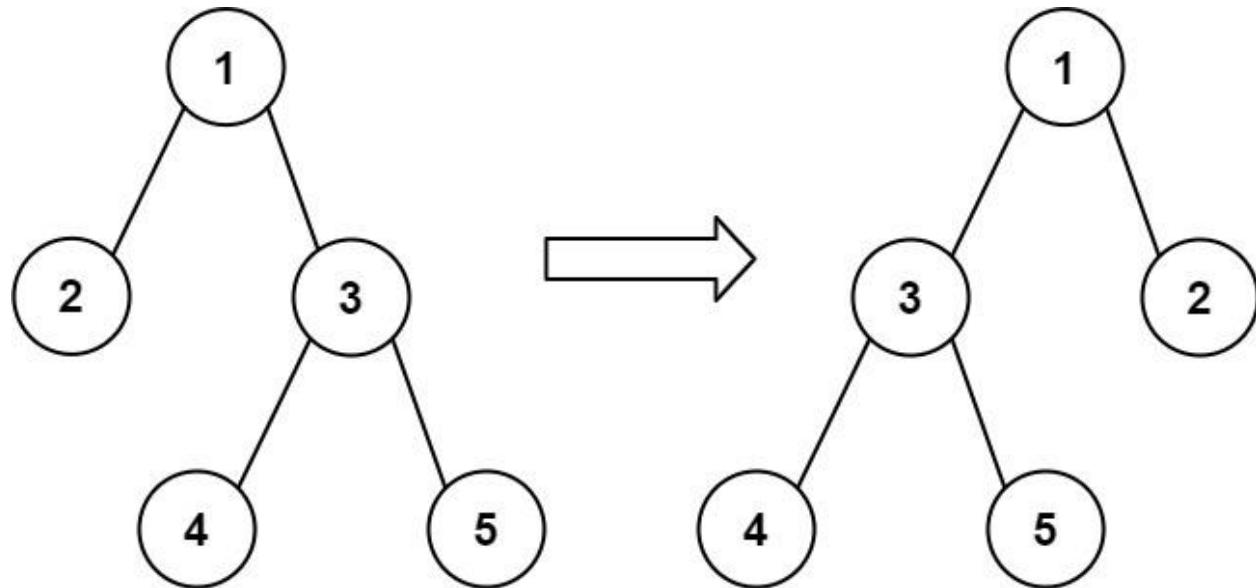
```
0 <= bound <= 106
```

**971. Flip Binary Tree To Match Preorder Traversal****Medium**

785250Add to ListShare

You are given the `root` of a binary tree with `n` nodes, where each node is uniquely assigned a value from 1 to `n`. You are also given a sequence of `n` values `voyage`, which is the **desired pre-order traversal** of the binary tree.

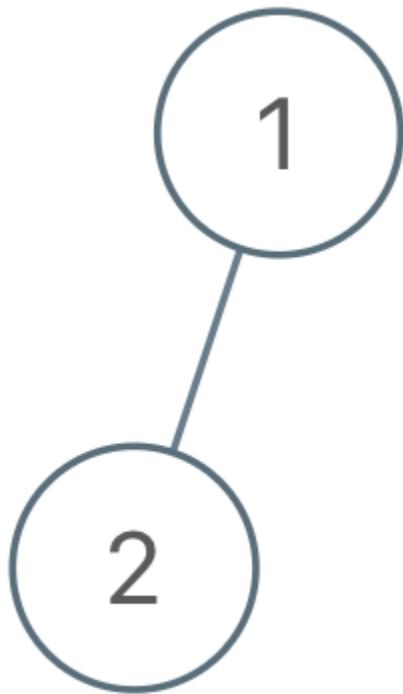
Any node in the binary tree can be **flipped** by swapping its left and right subtrees. For example, flipping node 1 will have the following effect:



Flip the **smallest** number of nodes so that the **pre-order traversal** of the tree **matches** `voyage`.

Return a *list of the values of all **flipped** nodes*. You may return the answer in **any order**. If it is **impossible** to flip the nodes in the tree to make the pre-order traversal match `voyage`, return the list `[-1]`.

**Example 1:**

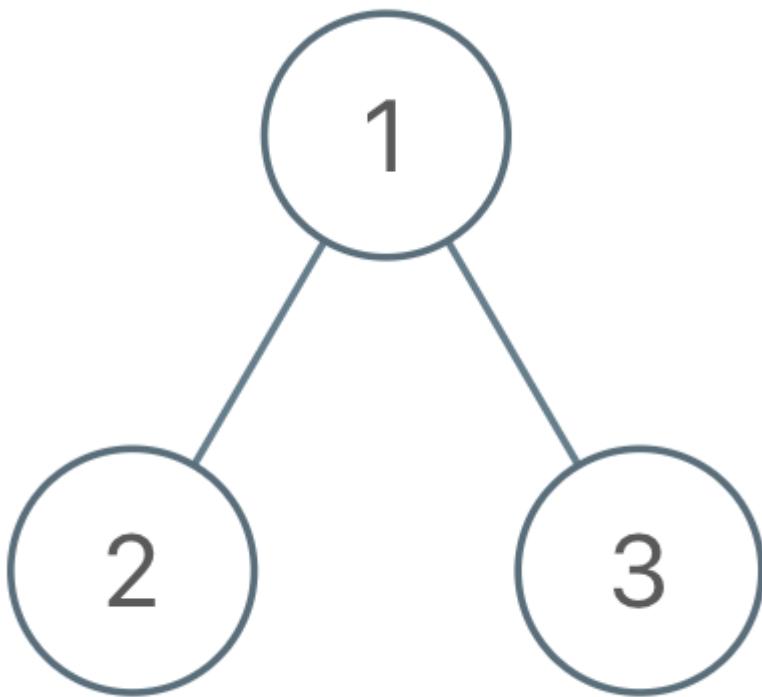


**Input:** root = [1,2], voyage = [2,1]

**Output:** [-1]

**Explanation:** It is impossible to flip the nodes such that the pre-order traversal matches voyage.

**Example 2:**

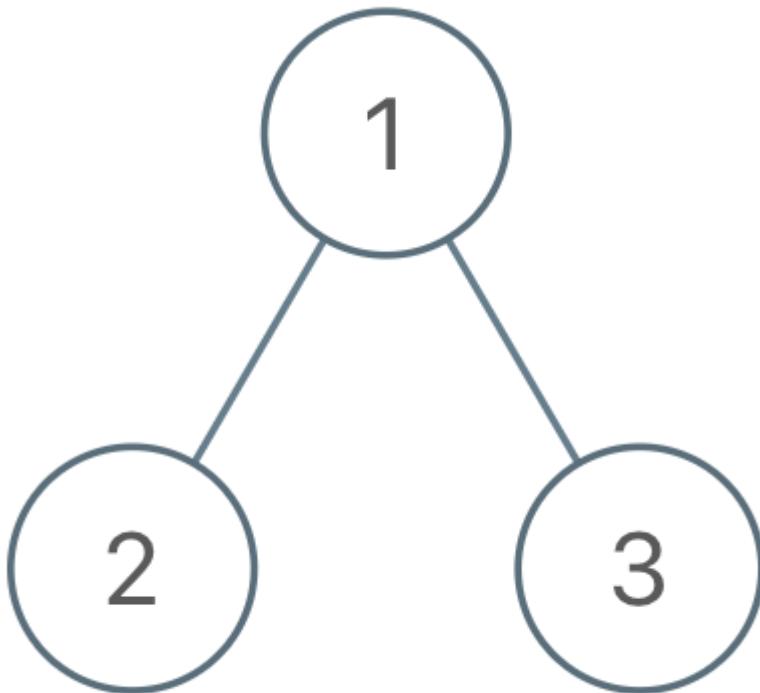


**Input:** root = [1,2,3], voyage = [1,3,2]

**Output:** [1]

**Explanation:** Flipping node 1 swaps nodes 2 and 3, so the pre-order traversal matches voyage.

**Example 3:**



**Input:** root = [1,2,3], voyage = [1,2,3]

**Output:** []

**Explanation:** The tree's pre-order traversal already matches voyage, so no nodes need to be flipped.

#### Constraints:

The number of nodes in the tree is n.

`n == voyage.length`

`1 <= n <= 100`

`1 <= Node.val, voyage[i] <= n`

All the values in the tree are **unique**.

All the values in `voyage` are **unique**.

#### 972. Equal Rational Numbers

## Hard

73195Add to ListShare

Given two strings  $s$  and  $t$ , each of which represents a non-negative rational number, return `true` if and only if they represent the same number. The strings may use parentheses to denote the repeating part of the rational number.

A **rational number** can be represented using up to three parts: `<IntegerPart>`, `<NonRepeatingPart>`, and a `<RepeatingPart>`. The number will be represented in one of the following three ways:

`<IntegerPart>`

For example, `12`, `0`, and `123`.

`<IntegerPart><.><NonRepeatingPart>`

For example, `0.5`, `1.`, `2.12`, and `123.0001`.

`<IntegerPart><.><NonRepeatingPart><(><RepeatingPart><)>`

For example, `0.1(6)`, `1.(9)`, `123.00(1212)`.

The repeating portion of a decimal expansion is conventionally denoted within a pair of round brackets. For example:

$1/6 = 0.16666666\dots = 0.1(6) = 0.1666(6) = 0.166(66).$

### Example 1:

**Input:**  $s = "0.(52)"$ ,  $t = "0.5(25)"$

**Output:** `true`

**Explanation:** Because `"0.(52)"` represents  $0.52525252\dots$ , and `"0.5(25)"` represents  $0.52525252525\dots$ , the strings represent the same number.

### Example 2:

**Input:**  $s = "0.1666(6)"$ ,  $t = "0.166(66)"$

**Output:** `true`

### Example 3:

**Input:**  $s = "0.9(9)"$ ,  $t = "1."$

**Output:** true

**Explanation:** "0.9(9)" represents 0.99999999... repeated forever, which equals 1. [See [this link for an explanation.](#)]

"1." represents the number 1, which is formed correctly: (IntegerPart) = "1" and (NonRepeatingPart) = "".

### Constraints:

Each part consists only of digits.

The `<IntegerPart>` does not have leading zeros (except for the zero itself).

```
1 <= <IntegerPart>.length <= 4
0 <= <NonRepeatingPart>.length <= 4
1 <= <RepeatingPart>.length <= 4
```

## 973. K Closest Points to Origin

Medium

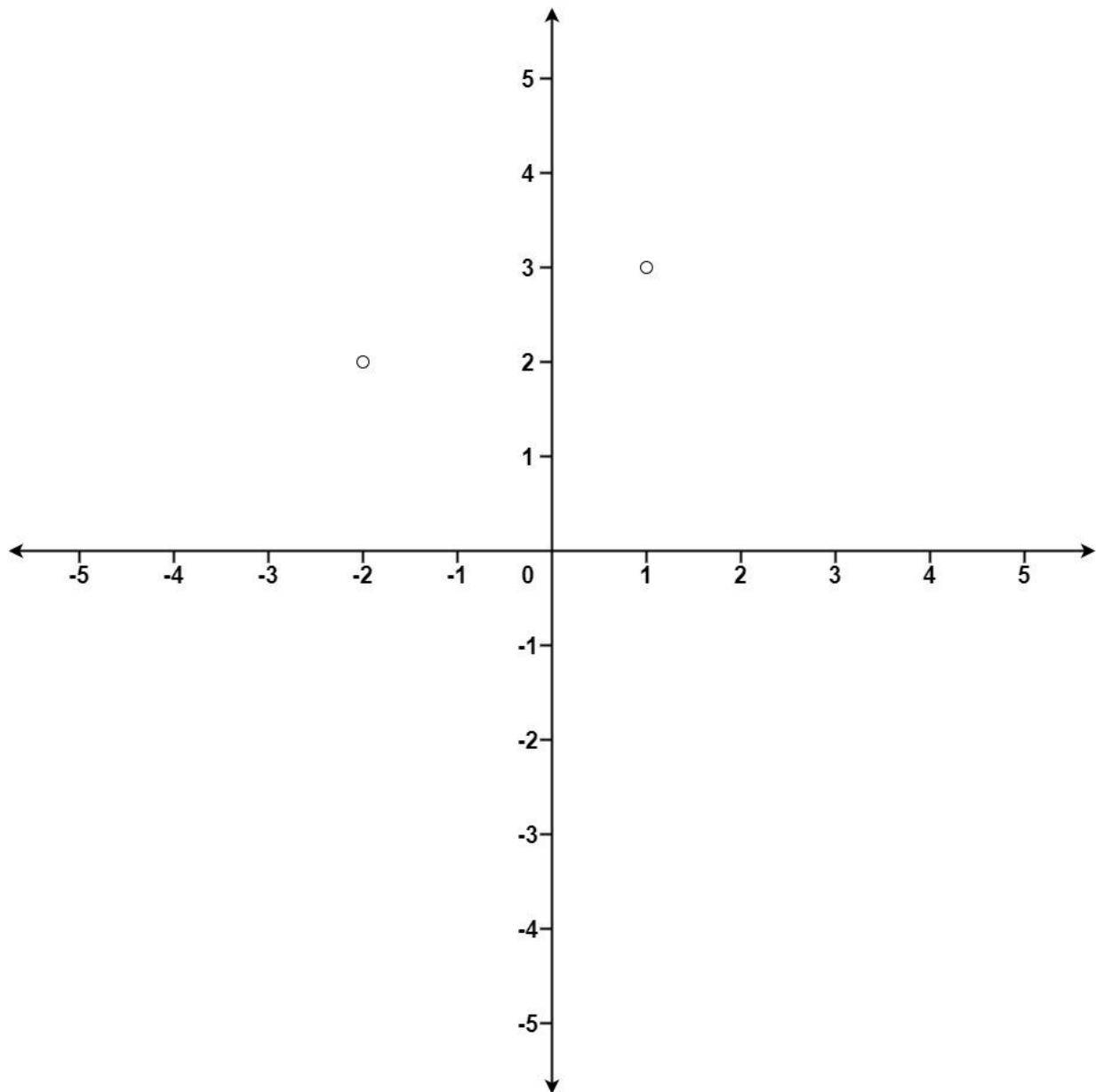
6554238Add to ListShare

Given an array of `points` where `points[i] = [xi, yi]` represents a point on the **X-Y** plane and an integer `k`, return the `k` closest points to the origin `(0, 0)`.

The distance between two points on the **X-Y** plane is the Euclidean distance (i.e.,  $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ ).

You may return the answer in **any order**. The answer is **guaranteed** to be **unique** (except for the order that it is in).

### Example 1:



**Input:** points = [[1,3], [-2,2]], k = 1

**Output:** [[-2,2]]

**Explanation:**

The distance between (1, 3) and the origin is  $\sqrt{10}$ .

The distance between (-2, 2) and the origin is  $\sqrt{8}$ .

Since  $\sqrt{8} < \sqrt{10}$ , (-2, 2) is closer to the origin.

We only want the closest  $k = 1$  points from the origin, so the answer is just [[-2,2]].

**Example 2:**

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

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

**Explanation:** The answer [[-2,4],[3,3]] would also be accepted.

**Constraints:**

$1 \leq k \leq \text{points.length} \leq 10^4$

$-10^4 \leq x_i, y_i \leq 10^4$

**974. Subarray Sums Divisible by K**

Medium

3576146Add to ListShare

Given an integer array `nums` and an integer `k`, return *the number of non-empty subarrays* that have a sum divisible by `k`.

A **subarray** is a **contiguous** part of an array.

**Example 1:**

**Input:** nums = [4,5,0,-2,-3,1], k = 5

**Output:** 7

**Explanation:** There are 7 subarrays with a sum divisible by  $k = 5$ :

[4, 5, 0, -2, -3, 1], [5], [5, 0], [5, 0, -2, -3], [0], [0, -2, -3], [-2, -3]

**Example 2:**

**Input:** nums = [5], k = 9

**Output:** 0

**Constraints:**

$1 \leq \text{nums.length} \leq 3 * 10^4$

$-10^4 \leq \text{nums}[i] \leq 10^4$

`2 <= k <= 104`

## 975. Odd Even Jump

Hard

1587422 Add to List Share

You are given an integer array `arr`. From some starting index, you can make a series of jumps. The (1<sup>st</sup>, 3<sup>rd</sup>, 5<sup>th</sup>, ...) jumps in the series are called **odd-numbered jumps**, and the (2<sup>nd</sup>, 4<sup>th</sup>, 6<sup>th</sup>, ...) jumps in the series are called **even-numbered jumps**. Note that the **jumps** are numbered, not the indices.

You may jump forward from index `i` to index `j` (with `i < j`) in the following way:

During **odd-numbered jumps** (i.e., jumps 1, 3, 5, ...), you jump to the index `j` such that `arr[i] <= arr[j]` and `arr[j]` is the smallest possible value. If there are multiple such indices `j`, you can only jump to the **smallest** such index `j`.

During **even-numbered jumps** (i.e., jumps 2, 4, 6, ...), you jump to the index `j` such that `arr[i] >= arr[j]` and `arr[j]` is the largest possible value. If there are multiple such indices `j`, you can only jump to the **smallest** such index `j`.

It may be the case that for some index `i`, there are no legal jumps.

A starting index is **good** if, starting from that index, you can reach the end of the array (index `arr.length - 1`) by jumping some number of times (possibly 0 or more than once).

Return *the number of good starting indices*.

### Example 1:

**Input:** `arr = [10,13,12,14,15]`

**Output:** 2

#### Explanation:

From starting index `i = 0`, we can make our 1st jump to `i = 2` (since `arr[2]` is the smallest among `arr[1], arr[2], arr[3], arr[4]` that is greater or equal to `arr[0]`), then we cannot jump any more.

From starting index `i = 1` and `i = 2`, we can make our 1st jump to `i = 3`, then we cannot jump any more.

From starting index `i = 3`, we can make our 1st jump to `i = 4`, so we have reached the end.

From starting index `i = 4`, we have reached the end already.

In total, there are 2 different starting indices  $i = 3$  and  $i = 4$ , where we can reach the end with some number of jumps.

**Example 2:**

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

**Output:** 3

**Explanation:**

From starting index  $i = 0$ , we make jumps to  $i = 1, i = 2, i = 3$ :

During our 1st jump (odd-numbered), we first jump to  $i = 1$  because  $arr[1]$  is the smallest value in  $[arr[1], arr[2], arr[3], arr[4]]$  that is greater than or equal to  $arr[0]$ .

During our 2nd jump (even-numbered), we jump from  $i = 1$  to  $i = 2$  because  $arr[2]$  is the largest value in  $[arr[2], arr[3], arr[4]]$  that is less than or equal to  $arr[1]$ .  $arr[3]$  is also the largest value, but 2 is a smaller index, so we can only jump to  $i = 2$  and not  $i = 3$

During our 3rd jump (odd-numbered), we jump from  $i = 2$  to  $i = 3$  because  $arr[3]$  is the smallest value in  $[arr[3], arr[4]]$  that is greater than or equal to  $arr[2]$ .

We can't jump from  $i = 3$  to  $i = 4$ , so the starting index  $i = 0$  is not good.

In a similar manner, we can deduce that:

From starting index  $i = 1$ , we jump to  $i = 4$ , so we reach the end.

From starting index  $i = 2$ , we jump to  $i = 3$ , and then we can't jump anymore.

From starting index  $i = 3$ , we jump to  $i = 4$ , so we reach the end.

From starting index  $i = 4$ , we are already at the end.

In total, there are 3 different starting indices  $i = 1, i = 3$ , and  $i = 4$ , where we can reach the end with some

number of jumps.

**Example 3:**

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

**Output:** 3

**Explanation:** We can reach the end from starting indices 1, 2, and 4.

**Constraints:**

```
1 <= arr.length <= 2 * 104
```

```
0 <= arr[i] < 105
```

**976. Largest Perimeter Triangle****Easy**

1386205Add to ListShare

Given an integer array `nums`, return *the largest perimeter of a triangle with a non-zero area, formed from three of these lengths*. If it is impossible to form any triangle of a non-zero area, return `0`.

**Example 1:**

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

**Output:** 5

**Example 2:**

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

**Output:** 0

**Constraints:**

```
3 <= nums.length <= 104
```

```
1 <= nums[i] <= 106
```

**977. Squares of a Sorted Array****Easy**

6468163Add to ListShare

Given an integer array `nums` sorted in **non-decreasing** order, return *an array of **the squares of each number** sorted in non-decreasing order*.

**Example 1:**

**Input:** `nums = [-4,-1,0,3,10]`

**Output:** `[0,1,9,16,100]`

**Explanation:** After squaring, the array becomes `[16,1,0,9,100]`.

After sorting, it becomes `[0,1,9,16,100]`.

**Example 2:**

**Input:** `nums = [-7,-3,2,3,11]`

**Output:** `[4,9,9,49,121]`

**Constraints:**

`1 <= nums.length <= 104`

`-104 <= nums[i] <= 104`

`nums` is sorted in **non-decreasing** order.

## 978. Longest Turbulent Subarray

Medium

1454177Add to ListShare

Given an integer array `arr`, return the length of a maximum size turbulent subarray of `arr`.

A subarray is **turbulent** if the comparison sign flips between each adjacent pair of elements in the subarray.

More formally, a subarray `[arr[i], arr[i + 1], ..., arr[j]]` of `arr` is said to be turbulent if and only if:

For `i <= k < j`:

`arr[k] > arr[k + 1]` when `k` is odd, and

`arr[k] < arr[k + 1]` when `k` is even.

Or, for `i <= k < j`:

`arr[k] > arr[k + 1]` when `k` is even, and

`arr[k] < arr[k + 1]` when `k` is odd.

**Example 1:**

**Input:** arr = [9,4,2,10,7,8,8,1,9]

**Output:** 5

**Explanation:** arr[1] > arr[2] < arr[3] > arr[4] < arr[5]

**Example 2:**

**Input:** arr = [4,8,12,16]

**Output:** 2

**Example 3:**

**Input:** arr = [100]

**Output:** 1

**Constraints:**

1 <= arr.length <= 4 \* 10<sup>4</sup>

0 <= arr[i] <= 10<sup>9</sup>

## 979. Distribute Coins in Binary Tree

Medium

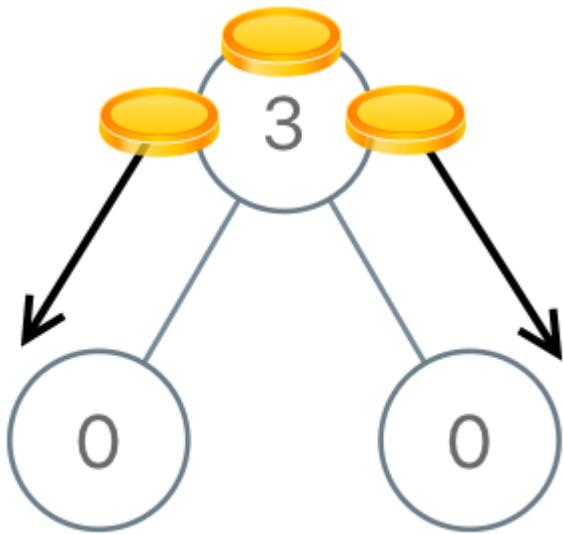
4157139Add to ListShare

You are given the `root` of a binary tree with `n` nodes where each `node` in the tree has `node.val` coins. There are `n` coins in total throughout the whole tree.

In one move, we may choose two adjacent nodes and move one coin from one node to another. A move may be from parent to child, or from child to parent.

Return *the **minimum** number of moves required to make every node have **exactly** one coin.*

**Example 1:**

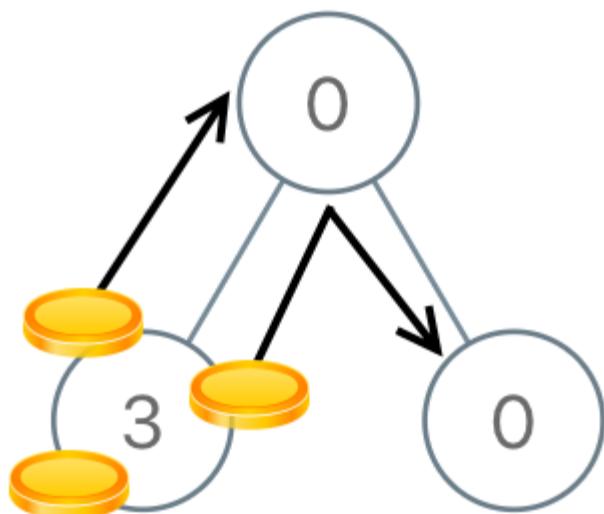


**Input:** root = [3,0,0]

**Output:** 2

**Explanation:** From the root of the tree, we move one coin to its left child, and one coin to its right child.

**Example 2:**



**Input:** root = [0,3,0]

**Output:** 3

**Explanation:** From the left child of the root, we move two coins to the root [taking two moves]. Then, we move one coin from the root of the tree to the right child.

### Constraints:

The number of nodes in the tree is  $n$ .

$1 \leq n \leq 100$

$0 \leq \text{Node.val} \leq n$

The sum of all `Node.val` is  $n$ .

## 980. Unique Paths III

Hard

3306144Add to ListShare

You are given an  $m \times n$  integer array `grid` where `grid[i][j]` could be:

`1` representing the starting square. There is exactly one starting square.

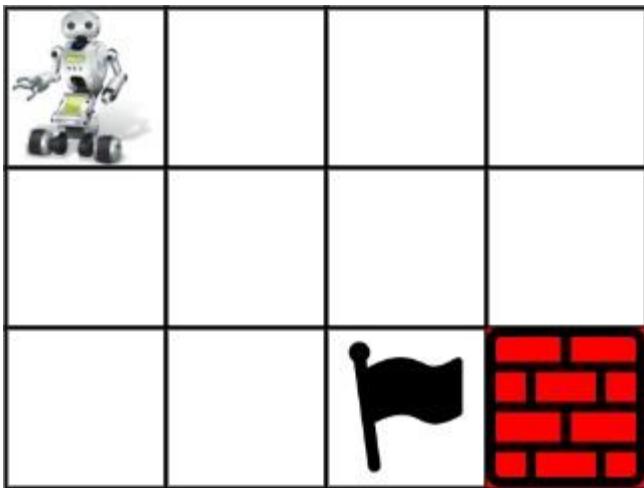
`2` representing the ending square. There is exactly one ending square.

`0` representing empty squares we can walk over.

`-1` representing obstacles that we cannot walk over.

Return the number of 4-directional walks from the starting square to the ending square, that walk over every non-obstacle square exactly once.

### Example 1:



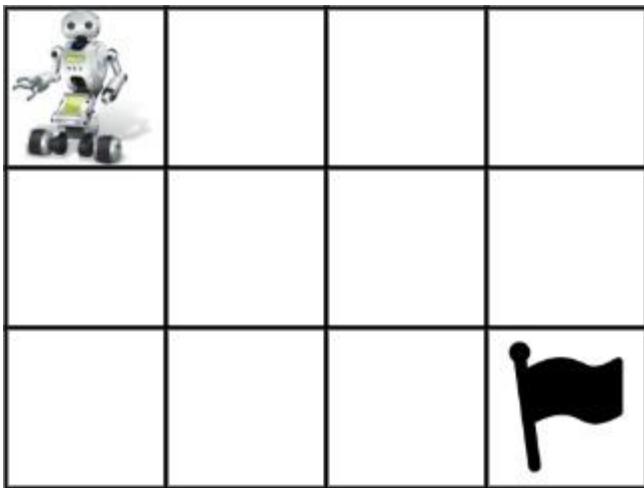
**Input:** grid = [[1,0,0,0],[0,0,0,0],[0,0,2,-1]]

**Output:** 2

**Explanation:** We have the following two paths:

1. (0,0),(0,1),(0,2),(0,3),(1,3),(1,2),(1,1),(1,0),(2,0),(2,1),(2,2)
2. (0,0),(1,0),(2,0),(2,1),(1,1),(0,1),(0,2),(0,3),(1,3),(1,2),(2,2)

**Example 2:**



**Input:** grid = [[1,0,0,0],[0,0,0,0],[0,0,0,2]]

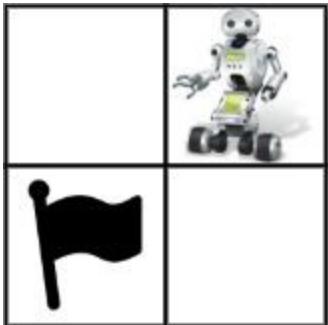
**Output:** 4

**Explanation:** We have the following four paths:

1. (0,0),(0,1),(0,2),(0,3),(1,3),(1,2),(1,1),(1,0),(2,0),(2,1),(2,2),(2,3)
2. (0,0),(0,1),(1,1),(1,0),(2,0),(2,1),(2,2),(1,2),(0,2),(0,3),(1,3),(2,3)
3. (0,0),(1,0),(2,0),(2,1),(2,2),(1,2),(1,1),(0,1),(0,2),(0,3),(1,3),(2,3)

4.  $(0,0), (1,0), (2,0), (2,1), (1,1), (0,1), (0,2), (0,3), (1,3), (1,2), (2,2), (2,3)$

**Example 3:**



**Input:** `grid = [[0,1],[2,0]]`

**Output:** 0

**Explanation:** There is no path that walks over every empty square exactly once.

Note that the starting and ending square can be anywhere in the grid.

**Constraints:**

```
m == grid.length
n == grid[i].length
1 <= m, n <= 20
1 <= m * n <= 20
-1 <= grid[i][j] <= 2
```

There is exactly one starting cell and one ending cell.

## 981. Time Based Key-Value Store

Medium

2551265 Add to List Share

Design a time-based key-value data structure that can store multiple values for the same key at different time stamps and retrieve the key's value at a certain timestamp.

Implement the `TimeMap` class:

`TimeMap()` Initializes the object of the data structure.

`void set(String key, String value, int timestamp)` Stores the key `key` with the value `value` at the given time `timestamp`.

`String get(String key, int timestamp)` Returns a value such that `set` was called previously, with `timestamp_prev <= timestamp`. If there are multiple such values, it returns the value associated with the largest `timestamp_prev`. If there are no values, it returns `""`.

### Example 1:

#### Input

```
["TimeMap", "set", "get", "get", "set", "get", "get"]
[[], ["foo", "bar", 1], ["foo", 1], ["foo", 3], ["foo", "bar2", 4], ["foo", 4],
["foo", 5]]
```

#### Output

```
null, null, "bar", "bar", null, "bar2", "bar2"]
```

#### Explanation

```
TimeMap timeMap = new TimeMap();

timeMap.set("foo", "bar", 1); // store the key "foo" and value "bar" along with
                           // timestamp = 1.

timeMap.get("foo", 1);       // return "bar"

timeMap.get("foo", 3);       // return "bar", since there is no value corresponding
                           // to foo at timestamp 3 and timestamp 2, then the only value is at timestamp 1 is
                           // "bar".

timeMap.set("foo", "bar2", 4); // store the key "foo" and value "bar2" along with
                           // timestamp = 4.

timeMap.get("foo", 4);       // return "bar2"

timeMap.get("foo", 5);       // return "bar2"
```

#### Constraints:

`1 <= key.length, value.length <= 100`

`key` and `value` consist of lowercase English letters and digits.

`1 <= timestamp <= 107`

All the timestamps `timestamp` of `set` are strictly increasing.

At most  $2 * 10^5$  calls will be made to `set` and `get`.

## 982. Triples with Bitwise AND Equal To Zero

Hard

303194Add to ListShare

Given an integer array `nums`, return *the number of AND triples*.

An **AND triple** is a triple of indices `(i, j, k)` such that:

`0 <= i < nums.length`

`0 <= j < nums.length`

`0 <= k < nums.length`

`nums[i] & nums[j] & nums[k] == 0`, where `&` represents the bitwise-AND operator.

### Example 1:

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

**Output:** 12

**Explanation:** We could choose the following `i, j, k` triples:

`(i=0, j=0, k=1) : 2 & 2 & 1`

`(i=0, j=1, k=0) : 2 & 1 & 2`

`(i=0, j=1, k=1) : 2 & 1 & 1`

`(i=0, j=1, k=2) : 2 & 1 & 3`

`(i=0, j=2, k=1) : 2 & 3 & 1`

`(i=1, j=0, k=0) : 1 & 2 & 2`

`(i=1, j=0, k=1) : 1 & 2 & 1`

`(i=1, j=0, k=2) : 1 & 2 & 3`

`(i=1, j=1, k=0) : 1 & 1 & 2`

```
(i=1, j=2, k=0) : 1 & 3 & 2
(i=2, j=0, k=1) : 3 & 2 & 1
(i=2, j=1, k=0) : 3 & 1 & 2
```

**Example 2:**

**Input:** nums = [0,0,0]

**Output:** 27

**Constraints:**

```
1 <= nums.length <= 1000
```

```
0 <= nums[i] < 216
```

## 983. Minimum Cost For Tickets

Medium

514887Add to ListShare

You have planned some train traveling one year in advance. The days of the year in which you will travel are given as an integer array `days`. Each day is an integer from 1 to 365.

Train tickets are sold in **three different ways**:

a **1-day** pass is sold for `costs[0]` dollars,

a **7-day** pass is sold for `costs[1]` dollars, and

a **30-day** pass is sold for `costs[2]` dollars.

The passes allow that many days of consecutive travel.

For example, if we get a **7-day** pass on day 2, then we can travel for 7 days: 2, 3, 4, 5, 6, 7, and 8.

Return *the minimum number of dollars you need to travel every day in the given list of days*.

**Example 1:**

**Input:** days = [1,4,6,7,8,20], costs = [2,7,15]

**Output:** 11

**Explanation:** For example, here is one way to buy passes that lets you travel your travel plan:

On day 1, you bought a 1-day pass for  $\text{costs}[0] = \$2$ , which covered day 1.

On day 3, you bought a 7-day pass for  $\text{costs}[1] = \$7$ , which covered days 3, 4, ..., 9.

On day 20, you bought a 1-day pass for  $\text{costs}[0] = \$2$ , which covered day 20.

In total, you spent  $\$11$  and covered all the days of your travel.

### Example 2:

**Input:**  $\text{days} = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 30, 31]$ ,  $\text{costs} = [2, 7, 15]$

**Output:** 17

**Explanation:** For example, here is one way to buy passes that lets you travel your travel plan:

On day 1, you bought a 30-day pass for  $\text{costs}[2] = \$15$  which covered days 1, 2, ..., 30.

On day 31, you bought a 1-day pass for  $\text{costs}[0] = \$2$  which covered day 31.

In total, you spent  $\$17$  and covered all the days of your travel.

### Constraints:

$1 \leq \text{days.length} \leq 365$

$1 \leq \text{days}[i] \leq 365$

$\text{days}$  is in strictly increasing order.

$\text{costs.length} == 3$

$1 \leq \text{costs}[i] \leq 1000$

## 984. String Without AAA or BBB

Medium

576342Add to ListShare

Given two integers  $a$  and  $b$ , return **any** string  $s$  such that:

$s$  has length  $a + b$  and contains exactly  $a$  ' $a$ ' letters, and exactly  $b$  ' $b$ ' letters,

The substring ' $aaa$ ' does not occur in  $s$ , and

The substring 'bbb' does not occur in `s`.

**Example 1:**

**Input:** `a = 1, b = 2`

**Output:** "abb"

**Explanation:** "abb", "bab" and "bba" are all correct answers.

**Example 2:**

**Input:** `a = 4, b = 1`

**Output:** "aabaa"

**Constraints:**

`0 <= a, b <= 100`

It is guaranteed such an `s` exists for the given `a` and `b`.

## 985. Sum of Even Numbers After Queries

Medium

1857310Add to ListShare

You are given an integer array `nums` and an array `queries` where `queries[i] = [vali, indexi]`.

For each query `i`, first, apply `nums[indexi] = nums[indexi] + vali`, then print the sum of the even values of `nums`.

Return an integer array `answer` where `answer[i]` is the answer to the `ith` query.

**Example 1:**

**Input:** `nums = [1,2,3,4], queries = [[1,0],[-3,1],[-4,0],[2,3]]`

**Output:** [8,6,2,4]

**Explanation:** At the beginning, the array is [1,2,3,4].

After adding 1 to `nums[0]`, the array is [2,2,3,4], and the sum of even values is  $2 + 2 + 4 = 8$ .

After adding -3 to `nums[1]`, the array is `[2, -1, 3, 4]`, and the sum of even values is  $2 + 4 = 6$ .

After adding -4 to `nums[0]`, the array is `[-2, -1, 3, 4]`, and the sum of even values is  $-2 + 4 = 2$ .

After adding 2 to `nums[3]`, the array is `[-2, -1, 3, 6]`, and the sum of even values is  $-2 + 6 = 4$ .

### Example 2:

**Input:** `nums = [1], queries = [[4, 0]]`

**Output:** `[0]`

### Constraints:

```
1 <= nums.length <= 104
-104 <= nums[i] <= 104
1 <= queries.length <= 104
-104 <= vali <= 104
0 <= indexi < nums.length
```

## 986. Interval List Intersections

Medium

470294Add to ListShare

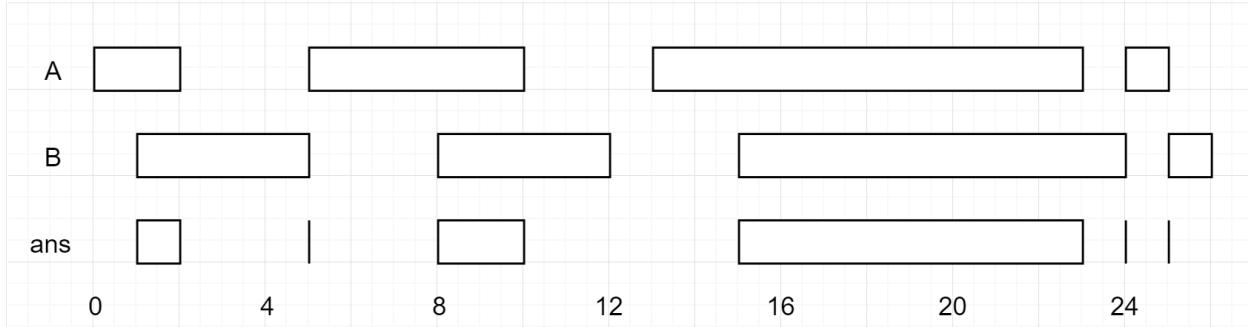
You are given two lists of closed intervals, `firstList` and `secondList`, where `firstList[i] = [starti, endi]` and `secondList[j] = [startj, endj]`. Each list of intervals is pairwise **disjoint** and in **sorted order**.

Return *the intersection of these two interval lists*.

A **closed interval** `[a, b]` (with `a <= b`) denotes the set of real numbers `x` with `a <= x <= b`.

The **intersection** of two closed intervals is a set of real numbers that are either empty or represented as a closed interval. For example, the intersection of `[1, 3]` and `[2, 4]` is `[2, 3]`.

### Example 1:



**Input:** firstList = [[0,2],[5,10],[13,23],[24,25]], secondList = [[1,5],[8,12],[15,24],[25,26]]

**Output:** [[1,2],[5,5],[8,10],[15,23],[24,24],[25,25]]

### Example 2:

**Input:** firstList = [[1,3],[5,9]], secondList = []

**Output:** []

### Constraints:

```
0 <= firstList.length, secondList.length <= 1000
firstList.length + secondList.length >= 1
0 <= starti < endi <= 109
endi < starti+1
0 <= startj < endj <= 109
endj < startj+1
```

## 987. Vertical Order Traversal of a Binary Tree

Hard

52023959Add to ListShare

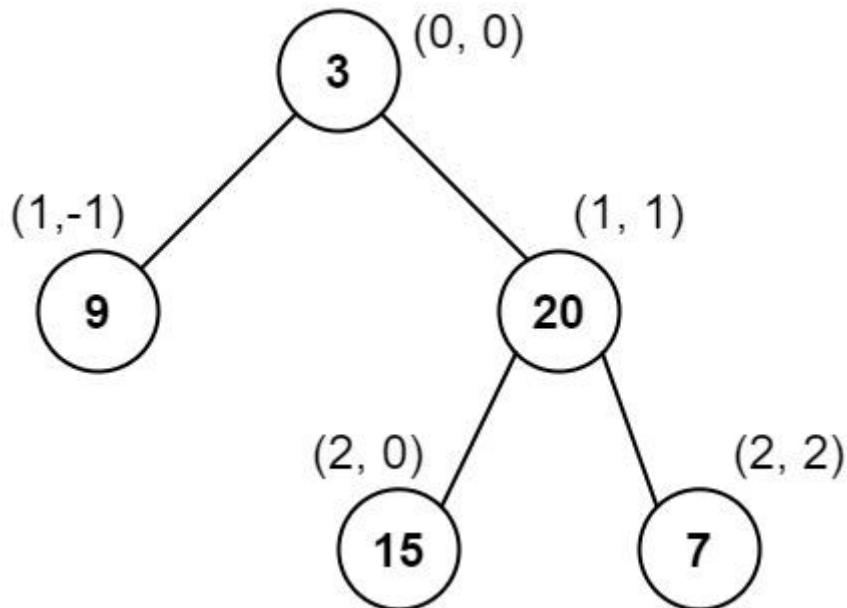
Given the `root` of a binary tree, calculate the **vertical order traversal** of the binary tree.

For each node at position `(row, col)`, its left and right children will be at positions `(row + 1, col - 1)` and `(row + 1, col + 1)` respectively. The root of the tree is at `(0, 0)`.

The **vertical order traversal** of a binary tree is a list of top-to-bottom orderings for each column index starting from the leftmost column and ending on the rightmost column. There may be multiple nodes in the same row and same column. In such a case, sort these nodes by their values.

Return *the vertical order traversal* of the binary tree.

**Example 1:**



**Input:** root = [3,9,20,null,null,15,7]

**Output:** [[9],[3,15],[20],[7]]

**Explanation:**

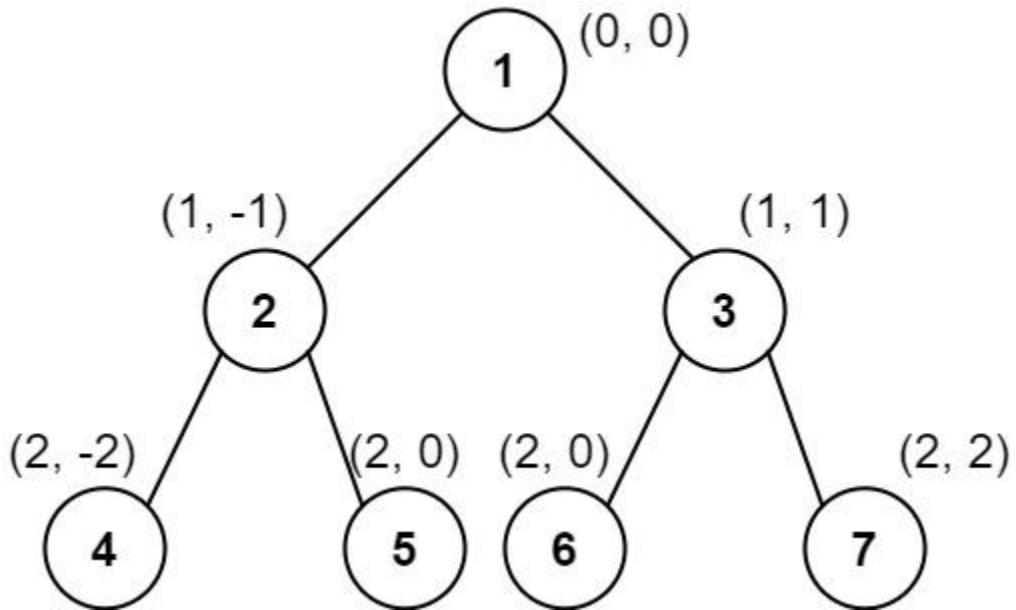
Column -1: Only node 9 is in this column.

Column 0: Nodes 3 and 15 are in this column in that order from top to bottom.

Column 1: Only node 20 is in this column.

Column 2: Only node 7 is in this column.

**Example 2:**



**Input:** root = [1,2,3,4,5,6,7]

**Output:** [[4],[2],[1,5,6],[3],[7]]

**Explanation:**

Column -2: Only node 4 is in this column.

Column -1: Only node 2 is in this column.

Column 0: Nodes 1, 5, and 6 are in this column.

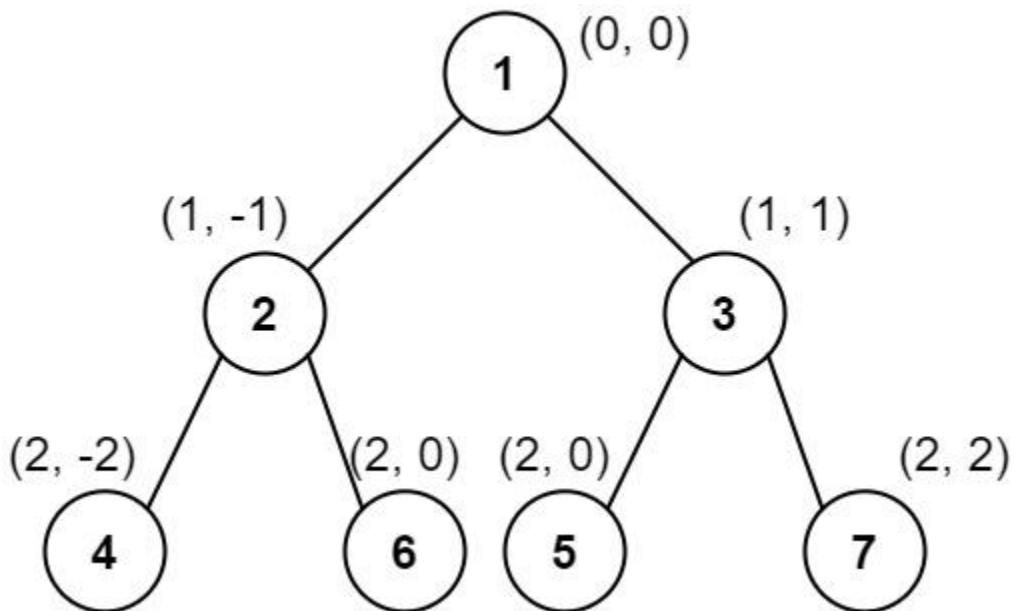
1 is at the top, so it comes first.

5 and 6 are at the same position (2, 0), so we order them by their value, 5 before 6.

Column 1: Only node 3 is in this column.

Column 2: Only node 7 is in this column.

**Example 3:**



**Input:** root = [1,2,3,4,6,5,7]

**Output:** [[4],[2],[1,5,6],[3],[7]]

**Explanation:**

This case is the exact same as example 2, but with nodes 5 and 6 swapped.

Note that the solution remains the same since 5 and 6 are in the same location and should be ordered by their values.

**Constraints:**

The number of nodes in the tree is in the range [1, 1000].

0 <= Node.val <= 1000

## 988. Smallest String Starting From Leaf

Medium

1283189Add to ListShare

You are given the `root` of a binary tree where each node has a value in the range [0, 25] representing the letters 'a' to 'z'.

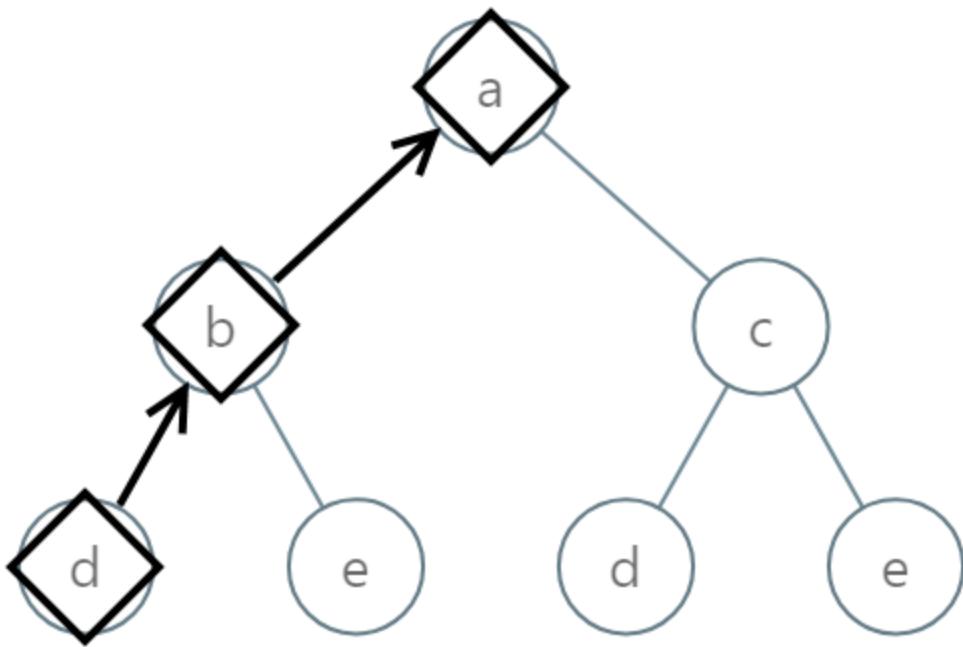
Return the **lexicographically smallest** string that starts at a leaf of this tree and ends at the root.

As a reminder, any shorter prefix of a string is **lexicographically smaller**.

For example, "ab" is lexicographically smaller than "aba".

A leaf of a node is a node that has no children.

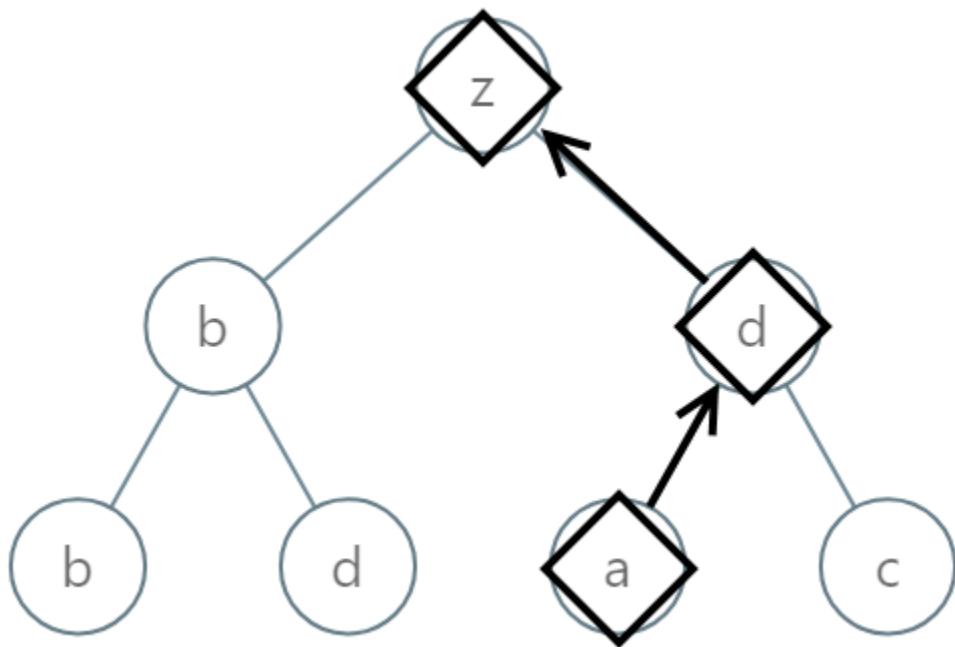
**Example 1:**



**Input:** root = [0,1,2,3,4,3,4]

**Output:** "dba"

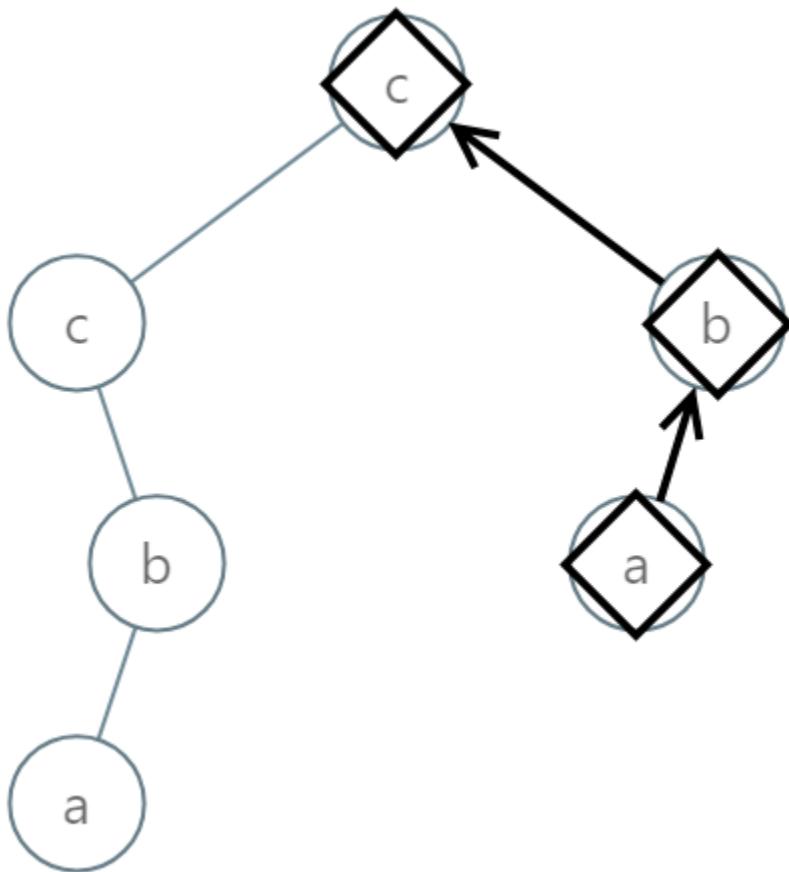
**Example 2:**



**Input:** root = [25,1,3,1,3,0,2]

**Output:** "adz"

**Example 3:**



**Input:** root = [2,2,1,null,1,0,null,0]

**Output:** "abc"

**Constraints:**

The number of nodes in the tree is in the range [1, 8500].

0 <= Node.val <= 25

## 989. Add to Array-Form of Integer

Easy

1538161Add to ListShare

The **array-form** of an integer `num` is an array representing its digits in left to right order.

For example, for `num = 1321`, the array form is [1, 3, 2, 1].

Given `num`, the **array-form** of an integer, and an integer `k`, return *the array-form of the integer `num` + `k`*.

**Example 1:**

**Input:** `num = [1,2,0,0]`, `k = 34`

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

**Explanation:**  $1200 + 34 = 1234$

**Example 2:**

**Input:** `num = [2,7,4]`, `k = 181`

**Output:** `[4,5,5]`

**Explanation:**  $274 + 181 = 455$

**Example 3:**

**Input:** `num = [2,1,5]`, `k = 806`

**Output:** `[1,0,2,1]`

**Explanation:**  $215 + 806 = 1021$

**Constraints:**

`1 <= num.length <= 104`

`0 <= num[i] <= 9`

`num` does not contain any leading zeros except for the zero itself.

`1 <= k <= 104`

## 990. Satisfiability of Equality Equations

Medium

264438Add to ListShare

You are given an array of strings `equations` that represent relationships between variables where each string `equations[i]` is of length 4 and takes one of two different forms: "`xi==yi`" or "`xi!=yi`". Here, `xi` and `yi` are lowercase letters (not necessarily different) that represent one-letter variable names.

Return `true` if it is possible to assign integers to variable names so as to satisfy all the given equations, or `false` otherwise.

**Example 1:**

**Input:** `equations = ["a==b","b!=a"]`

**Output:** `false`

**Explanation:** If we assign say,  $a = 1$  and  $b = 1$ , then the first equation is satisfied, but not the second.

There is no way to assign the variables to satisfy both equations.

**Example 2:**

**Input:** `equations = ["b==a","a==b"]`

**Output:** `true`

**Explanation:** We could assign  $a = 1$  and  $b = 1$  to satisfy both equations.

**Constraints:**

`1 <= equations.length <= 500`

`equations[i].length == 4`

`equations[i][0]` is a lowercase letter.

`equations[i][1]` is either `'='` or `'!='`.

`equations[i][2]` is `'='`.

`equations[i][3]` is a lowercase letter.

## 991. Broken Calculator

Medium

2396197 Add to List Share

There is a broken calculator that has the integer `startValue` on its display initially. In one operation, you can:

multiply the number on display by `2`, or

subtract 1 from the number on display.

Given two integers `startValue` and `target`, return *the minimum number of operations needed to display target on the calculator*.

**Example 1:**

**Input:** `startValue = 2, target = 3`

**Output:** 2

**Explanation:** Use double operation and then decrement operation {2 -> 4 -> 3}.

**Example 2:**

**Input:** `startValue = 5, target = 8`

**Output:** 2

**Explanation:** Use decrement and then double {5 -> 4 -> 8}.

**Example 3:**

**Input:** `startValue = 3, target = 10`

**Output:** 3

**Explanation:** Use double, decrement and double {3 -> 6 -> 5 -> 10}.

**Constraints:**

`1 <= startValue, target <= 109`

## 992. Subarrays with K Different Integers

Hard

359053Add to ListShare

Given an integer array `nums` and an integer `k`, return *the number of good subarrays of* `nums`.

A **good array** is an array where the number of different integers in that array is exactly `k`.

For example, `[1, 2, 3, 1, 2]` has 3 different integers: 1, 2, and 3.

A **subarray** is a **contiguous** part of an array.

**Example 1:**

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

**Output:** 7

**Explanation:** Subarrays formed with exactly 2 different integers: [1,2], [2,1], [1,2], [2,3], [1,2,1], [2,1,2], [1,2,1,2]

**Example 2:**

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

**Output:** 3

**Explanation:** Subarrays formed with exactly 3 different integers: [1,2,1,3], [2,1,3], [1,3,4].

**Constraints:**

1 <= nums.length <= 2 \* 10<sup>4</sup>

1 <= nums[i], k <= nums.length

**992. Subarrays with K Different Integers****Hard 994. Rotting Oranges**

Medium

8406309 Add to List Share

You are given an  $m \times n$  grid where each cell can have one of three values:

0 representing an empty cell,

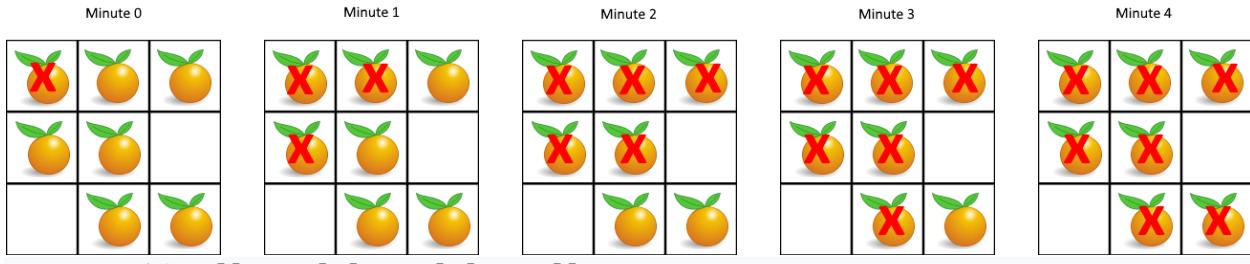
1 representing a fresh orange, or

2 representing a rotten orange.

Every minute, any fresh orange that is **4-directionally adjacent** to a rotten orange becomes rotten.

Return the *minimum number of minutes* that must elapse until no cell has a fresh orange. If this is impossible, return -1.

**Example 1:**



**Input:** grid = [[2,1,1],[1,1,0],[0,1,1]]

**Output:** 4

### Example 2:

**Input:** grid = [[2,1,1],[0,1,1],[1,0,1]]

**Output:** -1

**Explanation:** The orange in the bottom left corner (row 2, column 0) is never rotten, because rotting only happens 4-directionally.

### Example 3:

**Input:** grid = [[0,2]]

**Output:** 0

**Explanation:** Since there are already no fresh oranges at minute 0, the answer is just 0.

### Constraints:

```

m == grid.length
n == grid[i].length
1 <= m, n <= 10
grid[i][j] is 0, 1, or 2.

```

359053Add to ListShare

Given an integer array `nums` and an integer `k`, return the number of **good subarrays** of `nums`.

A **good array** is an array where the number of different integers in that array is exactly `k`.

For example, `[1, 2, 3, 1, 2]` has 3 different integers: `1, 2, and 3`.

A **subarray** is a **contiguous** part of an array.

**Example 1:**

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

**Output:** 7

**Explanation:** Subarrays formed with exactly 2 different integers: `[1,2]`, `[2,1]`, `[1,2]`, `[2,3]`, `[1,2,1]`, `[2,1,2]`, `[1,2,1,2]`

**Example 2:**

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

**Output:** 3

**Explanation:** Subarrays formed with exactly 3 different integers: `[1,2,1,3]`, `[2,1,3]`, `[1,3,4]`.

**Constraints:**

`1 <= nums.length <= 2 * 104`

`1 <= nums[i], k <= nums.length`

## 995. Minimum Number of K Consecutive Bit Flips

Hard

93752Add to ListShare

You are given a binary array `nums` and an integer `k`.

A **k-bit flip** is choosing a **subarray** of length `k` from `nums` and simultaneously changing every `0` in the subarray to `1`, and every `1` in the subarray to `0`.

Return *the minimum number of k-bit flips required so that there is no 0 in the array*. If it is not possible, return `-1`.

A **subarray** is a **contiguous** part of an array.

**Example 1:**

**Input:** `nums = [0,1,0], k = 1`

**Output:** 2

**Explanation:** Flip `nums[0]`, then flip `nums[2]`.

**Example 2:**

**Input:** `nums = [1,1,0]`, `k = 2`

**Output:** -1

**Explanation:** No matter how we flip subarrays of size 2, we cannot make the array become `[1,1,1]`.

**Example 3:**

**Input:** `nums = [0,0,0,1,0,1,1,0]`, `k = 3`

**Output:** 3

**Explanation:**

Flip `nums[0]`,`nums[1]`,`nums[2]`: `nums` becomes `[1,1,1,1,0,1,1,0]`

Flip `nums[4]`,`nums[5]`,`nums[6]`: `nums` becomes `[1,1,1,1,1,0,0,0]`

Flip `nums[5]`,`nums[6]`,`nums[7]`: `nums` becomes `[1,1,1,1,1,1,1,1]`

**Constraints:**

`1 <= nums.length <= 105`

`1 <= k <= nums.length`

## 996. Number of Squareful Arrays

Hard

77032Add to ListShare

An array is **squareful** if the sum of every pair of adjacent elements is a **perfect square**.

Given an integer array `nums`, return *the number of permutations of `nums` that are **squareful***.

Two permutations `perm1` and `perm2` are different if there is some index `i` such that `perm1[i] != perm2[i]`.

**Example 1:**

**Input:** `nums = [1,17,8]`

**Output:** 2

**Explanation:** [1,8,17] and [17,8,1] are the valid permutations.

**Example 2:**

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

**Output:** 1

**Constraints:**

1 <= nums.length <= 12

0 <= nums[i] <= 10<sup>9</sup>

## 997. Find the Town Judge

Easy

3928291Add to ListShare

In a town, there are  $n$  people labeled from 1 to  $n$ . There is a rumor that one of these people is secretly the town judge.

If the town judge exists, then:

The town judge trusts nobody.

Everybody (except for the town judge) trusts the town judge.

There is exactly one person that satisfies properties **1** and **2**.

You are given an array `trust` where `trust[i] = [ai, bi]` representing that the person labeled  $a_i$  trusts the person labeled  $b_i$ .

Return *the label of the town judge if the town judge exists and can be identified, or return -1 otherwise.*

**Example 1:**

**Input:** n = 2, trust = [[1,2]]

**Output:** 2

**Example 2:**

**Input:** n = 3, trust = [[1,3],[2,3]]

**Output:** 3

**Example 3:**

**Input:** n = 3, trust = [[1,3],[2,3],[3,1]]

**Output:** -1

**Constraints:**

```
1 <= n <= 1000
0 <= trust.length <= 104
trust[i].length == 2
```

All the pairs of `trust` are **unique**.

```
ai != bi
```

```
1 <= ai, bi <= n
```

## 998. Maximum Binary Tree II

Medium

417698Add to ListShare

A **maximum tree** is a tree where every node has a value greater than any other value in its subtree.

You are given the `root` of a maximum binary tree and an integer `val`.

Just as in the [previous problem](#), the given tree was constructed from a list `a` (`root = Construct(a)`) recursively with the following `Construct(a)` routine:

If `a` is empty, return `null`.

Otherwise, let `a[i]` be the largest element of `a`. Create a `root` node with the value `a[i]`.

The left child of `root` will be `Construct([a[0], a[1], ..., a[i - 1]])`.

The right child of `root` will be `Construct([a[i + 1], a[i + 2], ..., a[a.length - 1]])`.

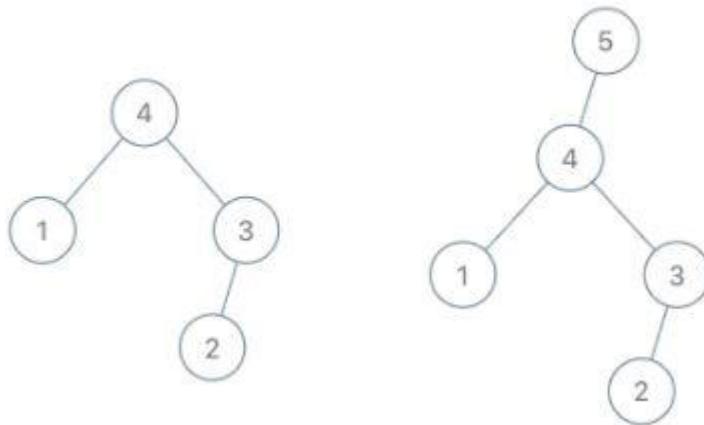
Return `root`.

Note that we were not given `a` directly, only a root node `root = Construct(a)`.

Suppose `b` is a copy of `a` with the value `val` appended to it. It is guaranteed that `b` has unique values.

Return `Construct(b)`.

**Example 1:**

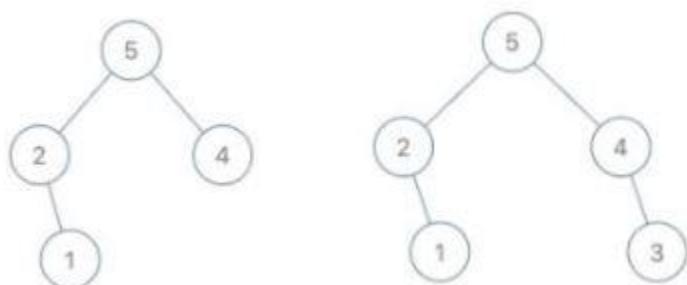


**Input:** `root = [4,1,3,null,null,2]`, `val = 5`

**Output:** `[5,4,null,1,3,null,null,2]`

**Explanation:** `a = [1,4,2,3]`, `b = [1,4,2,3,5]`

**Example 2:**

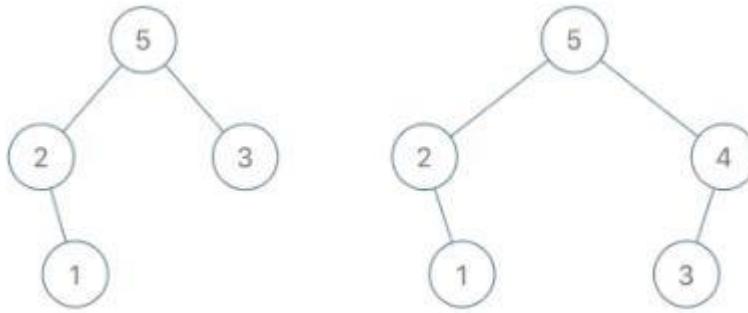


**Input:** `root = [5,2,4,null,1]`, `val = 3`

**Output:** `[5,2,4,null,1,null,3]`

**Explanation:** `a = [2,1,5,4]`, `b = [2,1,5,4,3]`

**Example 3:**



**Input:** root = [5,2,3,null,1], val = 4

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

**Explanation:** a = [2,1,5,3], b = [2,1,5,3,4]

### Constraints:

The number of nodes in the tree is in the range [1, 100].

1 <= Node.val <= 100

All the values of the tree are **unique**.

1 <= val <= 100

## 999. Available Captures for Rook

Easy

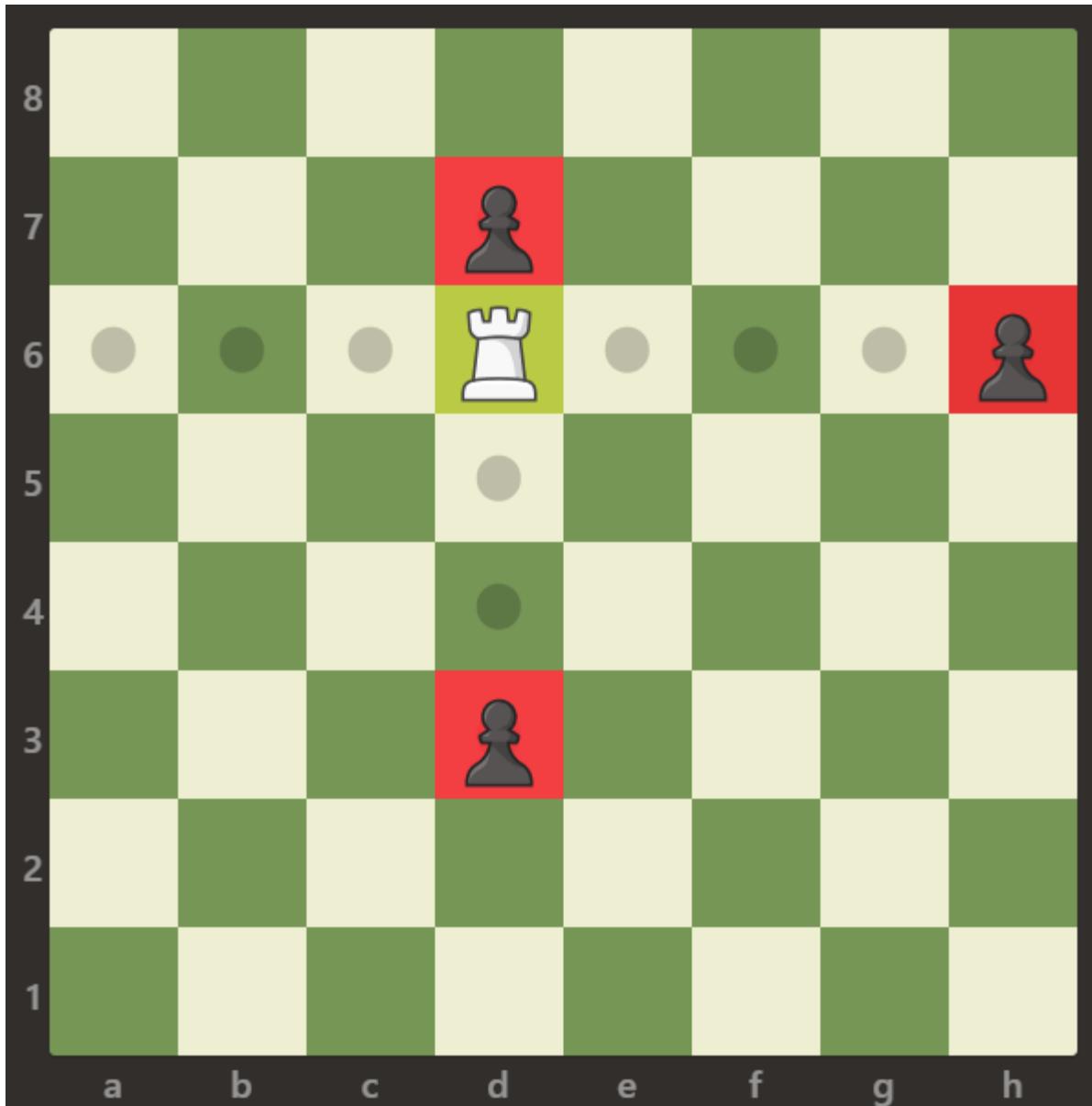
495583Add to ListShare

On an 8 x 8 chessboard, there is **exactly one** white rook 'R' and some number of white bishops 'B', black pawns 'p', and empty squares '.'.

When the rook moves, it chooses one of four cardinal directions (north, east, south, or west), then moves in that direction until it chooses to stop, reaches the edge of the board, captures a black pawn, or is blocked by a white bishop. A rook is considered **attacking** a pawn if the rook can capture the pawn on the rook's turn. The **number of available captures** for the white rook is the number of pawns that the rook is **attacking**.

Return *the number of available captures* for the white rook.

### Example 1:



**Input:** board =  
`[[".",".",".",".",".",".",".","."],[".",".",".","p",".",".",".","."],[".",".",".",".","R",".",".","p"],[".",".",".",".","."],[".",".",".",".","."],[".",".",".",".","."],[".",".","p",".",".","."],[".",".","p",".",".","."],[".",".","p",".",".","."],[".",".","p",".",".","."]]`

**Output:** 3

**Explanation:** In this example, the rook is attacking all the pawns.

**Example 2:**

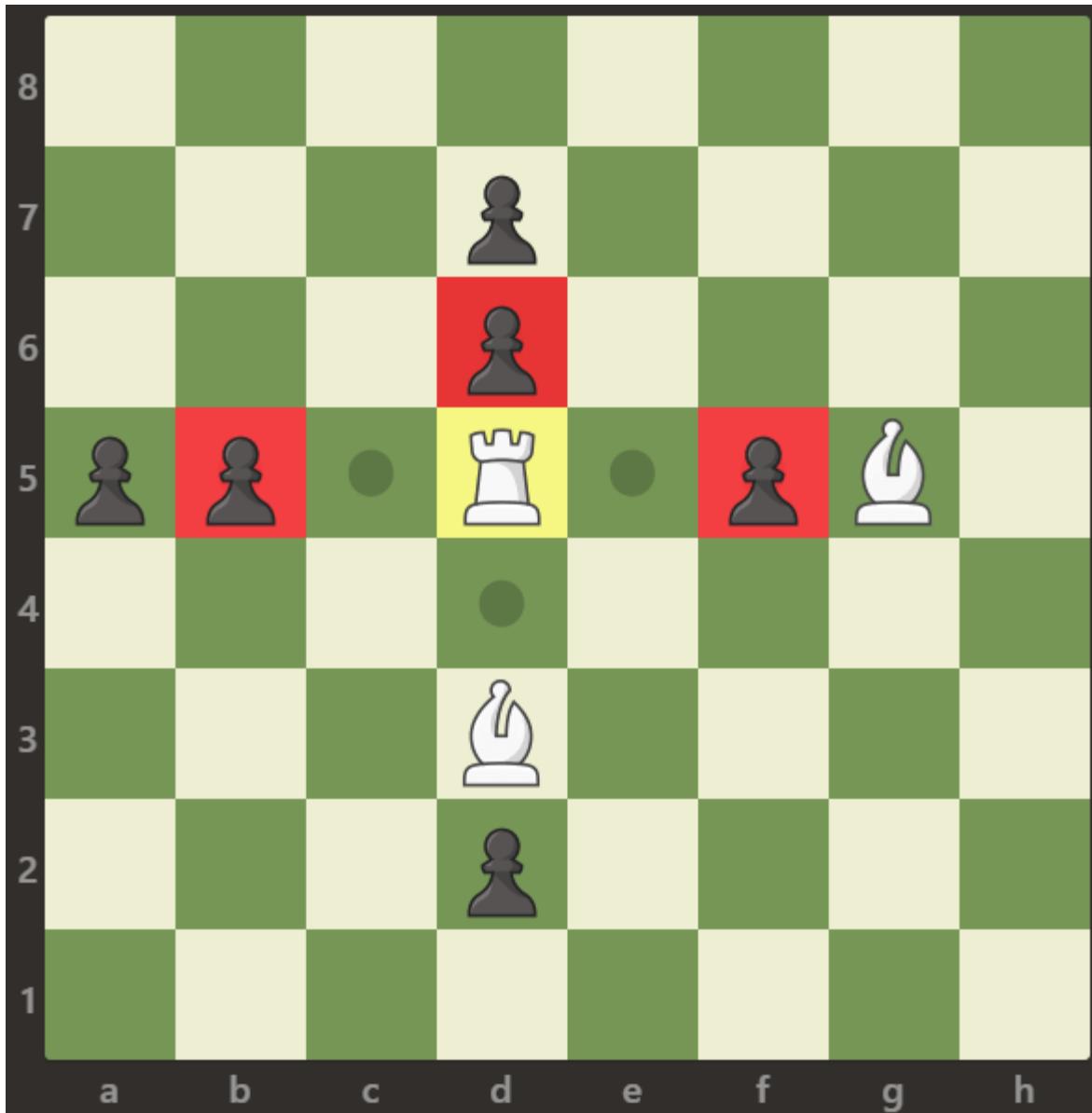


**Input:** board =  
[[".", ".", ".", ".", ".", ".", "."], [".", "p", "p", "p", "p", "p", "."], [".", "p", "p", "B", "R", "B", "p", "."], [".", "p", "p", "p", "p", "p", "p", "."]]

Output: 0

**Explanation:** The bishops are blocking the rook from attacking any of the pawns.

### Example 3:



**Input:** board =

```
[[".",".",".",".",".",".","."],[".",".",".","p",".",".","."],[".",".",".","p",".",".","p","."],[p,"p",".",R,"p","B","."],[".",".",B,".",".","."],[".",".",p,".",".","."],[".",".",p,".",".","."],[".",".",p,".",".","."]]
```

**Output:** 3

**Explanation:** The rook is attacking the pawns at positions b5, d6, and f5.

**Constraints:**

```
board.length == 8
```

```
board[i].length == 8
```

```
board[i][j] is either 'R', '.', 'B', or 'p'
```

There is exactly one cell with `board[i][j] == 'R'`

## 1000. Minimum Cost to Merge Stones

Hard

188692Add to ListShare

There are `n` piles of `stones` arranged in a row. The `ith` pile has `stones[i]` stones.

A move consists of merging exactly `k` **consecutive** piles into one pile, and the cost of this move is equal to the total number of stones in these `k` piles.

Return *the minimum cost to merge all piles of stones into one pile*. If it is impossible, return `-1`.

### Example 1:

**Input:** `stones = [3,2,4,1], k = 2`

**Output:** `20`

**Explanation:** We start with `[3, 2, 4, 1]`.

We merge `[3, 2]` for a cost of `5`, and we are left with `[5, 4, 1]`.

We merge `[4, 1]` for a cost of `5`, and we are left with `[5, 5]`.

We merge `[5, 5]` for a cost of `10`, and we are left with `[10]`.

The total cost was `20`, and this is the minimum possible.

### Example 2:

**Input:** `stones = [3,2,4,1], k = 3`

**Output:** `-1`

**Explanation:** After any merge operation, there are `2` piles left, and we can't merge anymore. So the task is impossible.

### Example 3:

**Input:** `stones = [3,5,1,2,6], k = 3`

**Output:** `25`

**Explanation:** We start with `[3, 5, 1, 2, 6]`.

We merge  $[5, 1, 2]$  for a cost of 8, and we are left with  $[3, 8, 6]$ .

We merge  $[3, 8, 6]$  for a cost of 17, and we are left with  $[17]$ .

The total cost was 25, and this is the minimum possible.

### Constraints:

```

n == stones.length

1 <= n <= 30

1 <= stones[i] <= 100

2 <= k <= 30

```

## 1001. Grid Illumination

Hard

489122Add to ListShare

There is a 2D grid of size  $n \times n$  where each cell of this grid has a lamp that is initially **turned off**.

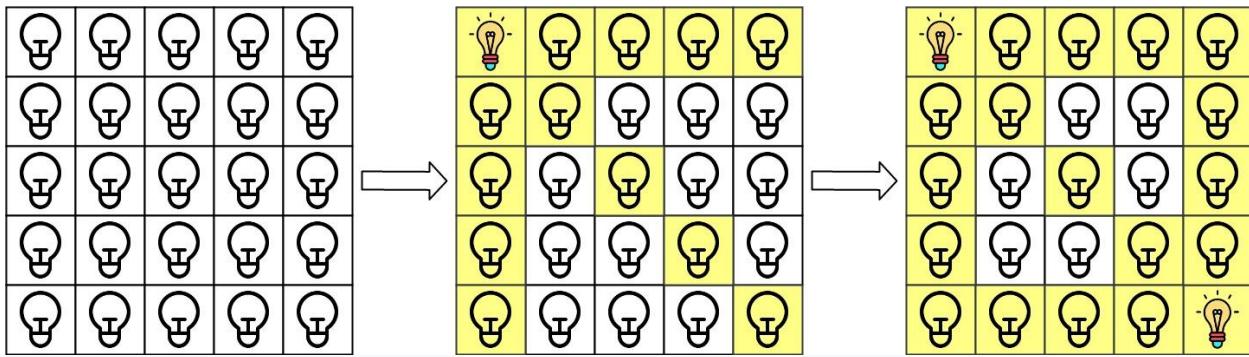
You are given a 2D array of lamp positions `lamps`, where  $\text{lamps}[i] = [\text{row}_i, \text{col}_i]$  indicates that the lamp at  $\text{grid}[\text{row}_i][\text{col}_i]$  is **turned on**. Even if the same lamp is listed more than once, it is turned on.

When a lamp is turned on, it **illuminates its cell** and **all other cells** in the same **row, column, or diagonal**.

You are also given another 2D array `queries`, where  $\text{queries}[j] = [\text{row}_j, \text{col}_j]$ . For the  $j^{\text{th}}$  query, determine whether  $\text{grid}[\text{row}_j][\text{col}_j]$  is illuminated or not. After answering the  $j^{\text{th}}$  query, **turn off** the lamp at  $\text{grid}[\text{row}_j][\text{col}_j]$  and its **8 adjacent lamps** if they exist. A lamp is adjacent if its cell shares either a side or corner with  $\text{grid}[\text{row}_j][\text{col}_j]$ .

Return an array of integers `ans`, where  $\text{ans}[j]$  should be 1 if the cell in the  $j^{\text{th}}$  query was illuminated, or 0 if the lamp was not.

### Example 1:

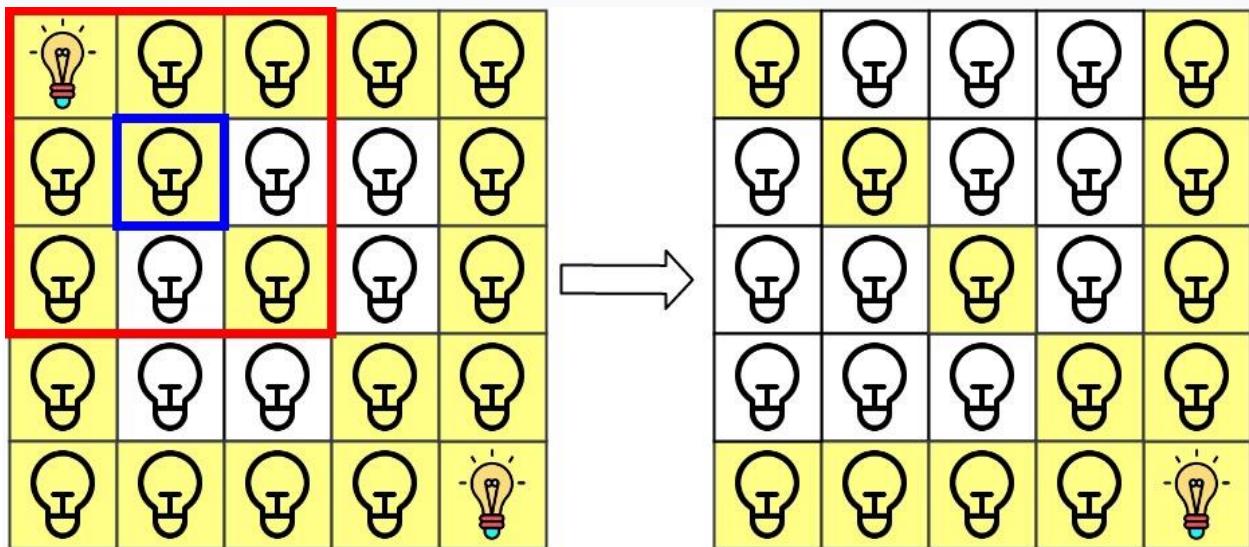


**Input:**  $n = 5$ ,  $\text{lamps} = [[0,0], [4,4]]$ ,  $\text{queries} = [[1,1], [1,0]]$

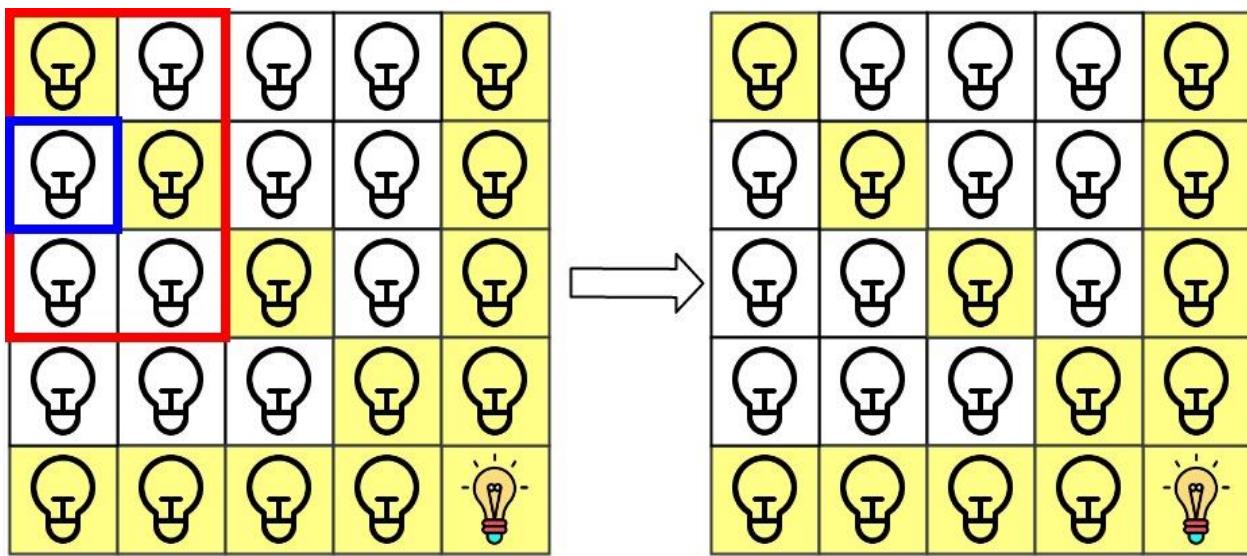
**Output:**  $[1,0]$

**Explanation:** We have the initial grid with all lamps turned off. In the above picture we see the grid after turning on the lamp at  $\text{grid}[0][0]$  then turning on the lamp at  $\text{grid}[4][4]$ .

The 0<sup>th</sup> query asks if the lamp at  $\text{grid}[1][1]$  is illuminated or not (the blue square). It is illuminated, so set  $\text{ans}[0] = 1$ . Then, we turn off all lamps in the red square.



The 1<sup>st</sup> query asks if the lamp at  $\text{grid}[1][0]$  is illuminated or not (the blue square). It is not illuminated, so set  $\text{ans}[1] = 0$ . Then, we turn off all lamps in the red rectangle.



### Example 2:

**Input:**  $n = 5$ ,  $\text{lamps} = [[0,0],[4,4]]$ ,  $\text{queries} = [[1,1],[1,1]]$

**Output:**  $[1,1]$

### Example 3:

**Input:**  $n = 5$ ,  $\text{lamps} = [[0,0],[0,4]]$ ,  $\text{queries} = [[0,4],[0,1],[1,4]]$

**Output:**  $[1,1,0]$

### Constraints:

- $1 \leq n \leq 10^9$
- $0 \leq \text{lamps.length} \leq 20000$
- $0 \leq \text{queries.length} \leq 20000$
- $\text{lamps}[i].length == 2$
- $0 \leq \text{row}_i, \text{col}_i < n$
- $\text{queries}[j].length == 2$
- $0 \leq \text{row}_j, \text{col}_j < n$

## 1002. Find Common Characters

Easy

2615214Add to ListShare

Given a string array `words`, return an array of all characters that show up in all strings within the `words` (including duplicates). You may return the answer in **any order**.

**Example 1:**

**Input:** words = ["bella", "label", "roller"]

**Output:** ["e", "l", "l"]

**Example 2:**

**Input:** words = ["cool", "lock", "cook"]

**Output:** ["c", "o"]

**Constraints:**

- $1 \leq \text{words.length} \leq 100$
- $1 \leq \text{words[i].length} \leq 100$
- `words[i]` consists of lowercase English letters.

**1003. Check If Word Is Valid After Substitutions**

Medium

696450Add to ListShare

Given a string `s`, determine if it is **valid**.

A string `s` is **valid** if, starting with an empty string `t = ""`, you can **transform** `t` **into** `s` after performing the following operation **any number of times**:

- Insert string `"abc"` into any position in `t`. More formally, `t` becomes `tleft + "abc" + tright`, where `t == tleft + tright`. Note that `tleft` and `tright` may be **empty**.

Return `true` if `s` is a **valid** string, otherwise, return `false`.

**Example 1:**

**Input:** s = "aabcbc"

**Output:** true

**Explanation:**

`"" -> "abc" -> "aabcbc"`

Thus, "aabcbc" is valid.

**Example 2:**

**Input:** s = "abcabcbabcc"

**Output:** true

**Explanation:**

"" -> abc -> abcabc -> abcababc -> abcabcababcc"

Thus, "abcabcbabcc" is valid.

**Example 3:**

**Input:** s = "abccba"

**Output:** false

**Explanation:** It is impossible to get "abccba" using the operation.

**Constraints:**

- $1 \leq s.length \leq 2 * 10^4$
- s consists of letters 'a', 'b', and 'c'

## 1004. Max Consecutive Ones III

Medium

524465Add to ListShare

Given a binary array `nums` and an integer `k`, return *the maximum number of consecutive 1's in the array if you can flip at most `k` 0's*.

**Example 1:**

**Input:** nums = [1,1,1,0,0,0,1,1,1,1,0], k = 2

**Output:** 6

**Explanation:** [1,1,1,0,0,1,1,1,1,1,1]

Bolded numbers were flipped from 0 to 1. The longest subarray is underlined.

**Example 2:**

**Input:** nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

**Output:** 10

**Explanation:** [0,0,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1]

Bolded numbers were flipped from 0 to 1. The longest subarray is underlined.

### Constraints:

- $1 \leq \text{nums.length} \leq 10^5$
- $\text{nums}[i]$  is either 0 or 1.
- $0 \leq k \leq \text{nums.length}$

## 1005. Maximize Sum Of Array After K Negations

Easy

113287Add to ListShare

Given an integer array `nums` and an integer `k`, modify the array in the following way:

- choose an index `i` and replace `nums[i]` with `-nums[i]`.

You should apply this process exactly `k` times. You may choose the same index `i` multiple times.

Return *the largest possible sum of the array after modifying it in this way*.

### Example 1:

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

**Output:** 5

**Explanation:** Choose index 1 and `nums` becomes `[4,-2,3]`.

### Example 2:

**Input:** `nums = [3,-1,0,2]`, `k = 3`

**Output:** 6

**Explanation:** Choose indices (1, 2, 2) and `nums` becomes `[3,1,0,2]`.

### Example 3:

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

**Output:** 13

**Explanation:** Choose indices (1, 4) and `nums` becomes `[2,3,-1,5,4]`.

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^4$
- $-100 \leq \text{nums}[i] \leq 100$
- $1 \leq k \leq 10^4$

**1006. Clumsy Factorial****Medium**

234261Add to ListShare

The **factorial** of a positive integer  $n$  is the product of all positive integers less than or equal to  $n$ .

- For example,  $\text{factorial}(10) = 10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1$ .

We make a **clumsy factorial** using the integers in decreasing order by swapping out the multiply operations for a fixed rotation of operations with multiply ' $*$ ', divide ' $/$ ', add ' $+$ ', and subtract ' $-$ ' in this order.

- For example,  $\text{clumsy}(10) = 10 * 9 / 8 + 7 - 6 * 5 / 4 + 3 - 2 * 1$ .

However, these operations are still applied using the usual order of operations of arithmetic. We do all multiplication and division steps before any addition or subtraction steps, and multiplication and division steps are processed left to right.

Additionally, the division that we use is floor division such that  $10 * 9 / 8 = 90 / 8 = 11$ .

Given an integer  $n$ , return *the clumsy factorial of  $n$* .

**Example 1:****Input:**  $n = 4$ **Output:** 7**Explanation:**  $7 = 4 * 3 / 2 + 1$ **Example 2:****Input:**  $n = 10$ **Output:** 12**Explanation:**  $12 = 10 * 9 / 8 + 7 - 6 * 5 / 4 + 3 - 2 * 1$ **Constraints:**

- $1 \leq n \leq 10^4$

## 1007. Minimum Domino Rotations For Equal Row

Medium

2587239Add to ListShare

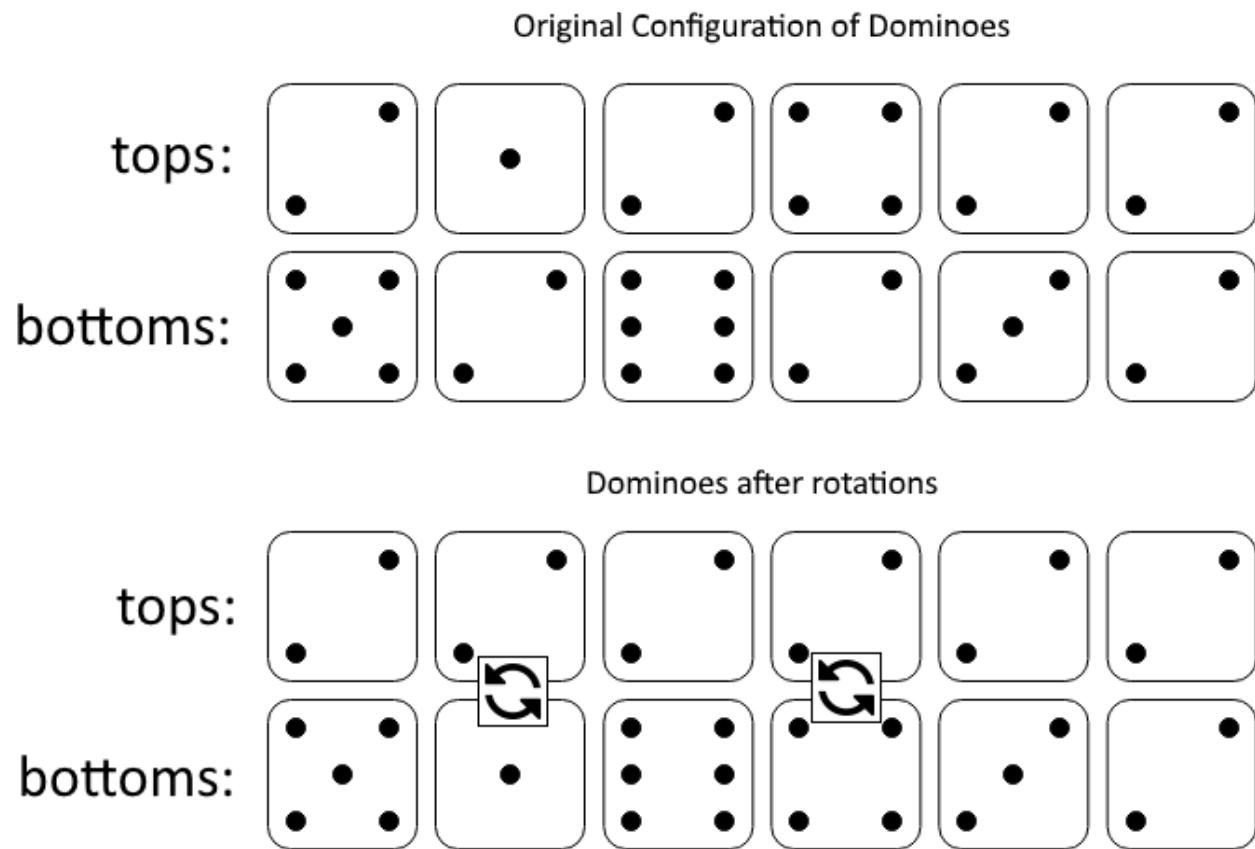
In a row of dominoes, `tops[i]` and `bottoms[i]` represent the top and bottom halves of the  $i^{\text{th}}$  domino. (A domino is a tile with two numbers from 1 to 6 - one on each half of the tile.)

We may rotate the  $i^{\text{th}}$  domino, so that `tops[i]` and `bottoms[i]` swap values.

Return the minimum number of rotations so that all the values in `tops` are the same, or all the values in `bottoms` are the same.

If it cannot be done, return `-1`.

**Example 1:**



**Input:** `tops = [2,1,2,4,2,2]`, `bottoms = [5,2,6,2,3,2]`

**Output:** 2

**Explanation:**

The first figure represents the dominoes as given by tops and bottoms: before we do any rotations.

If we rotate the second and fourth dominoes, we can make every value in the top row equal to 2, as indicated by the second figure.

**Example 2:**

**Input:** tops = [3,5,1,2,3], bottoms = [3,6,3,3,4]

**Output:** -1

**Explanation:**

In this case, it is not possible to rotate the dominoes to make one row of values equal.

**Constraints:**

- $2 \leq \text{tops.length} \leq 2 * 10^4$
- $\text{bottoms.length} == \text{tops.length}$
- $1 \leq \text{tops}[i], \text{bottoms}[i] \leq 6$

## 1008. Construct Binary Search Tree from Preorder Traversal

Medium

449564Add to ListShare

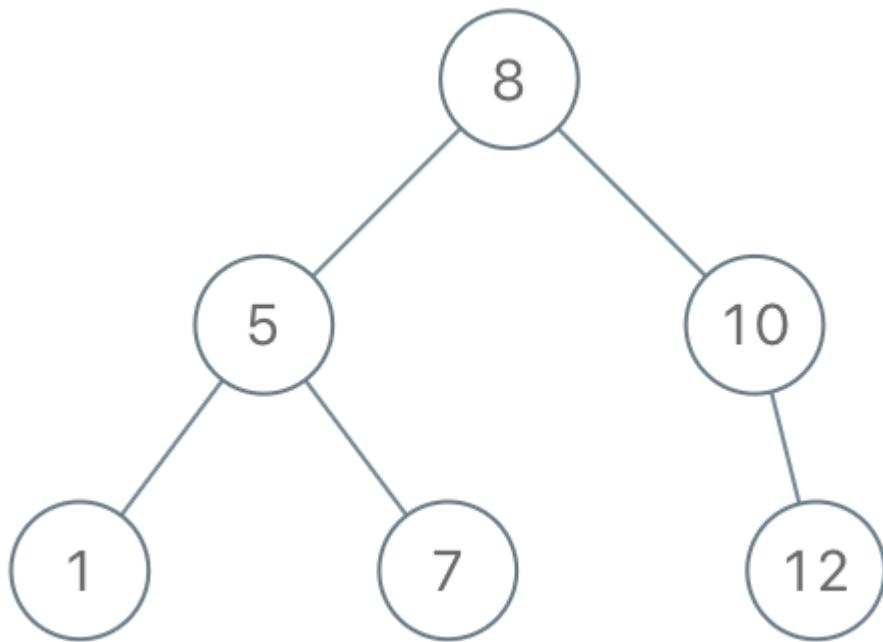
Given an array of integers preorder, which represents the **preorder traversal** of a BST (i.e., **binary search tree**), construct the tree and return *its root*.

It is **guaranteed** that there is always possible to find a binary search tree with the given requirements for the given test cases.

A **binary search tree** is a binary tree where for every node, any descendant of `Node.left` has a value **strictly less than** `Node.val`, and any descendant of `Node.right` has a value **strictly greater than** `Node.val`.

A **preorder traversal** of a binary tree displays the value of the node first, then traverses `Node.left`, then traverses `Node.right`.

**Example 1:**



**Input:** preorder = [8,5,1,7,10,12]

**Output:** [8,5,10,1,7,null,12]

**Example 2:**

**Input:** preorder = [1,3]

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

**Constraints:**

- $1 \leq \text{preorder.length} \leq 100$
- $1 \leq \text{preorder}[i] \leq 1000$
- All the values of `preorder` are **unique**.

## 1009. Complement of Base 10 Integer

Easy

161982Add to ListShare

The **complement** of an integer is the integer you get when you flip all the 0's to 1's and all the 1's to 0's in its binary representation.

- For example, The integer 5 is "101" in binary and its **complement** is "010" which is the integer 2.

Given an integer `n`, return *its complement*.

**Example 1:**

**Input:** `n = 5`

**Output:** `2`

**Explanation:** `5` is "101" in binary, with complement "010" in binary, which is `2` in base-10.

**Example 2:**

**Input:** `n = 7`

**Output:** `0`

**Explanation:** `7` is "111" in binary, with complement "000" in binary, which is `0` in base-10.

**Example 3:**

**Input:** `n = 10`

**Output:** `5`

**Explanation:** `10` is "1010" in binary, with complement "0101" in binary, which is `5` in base-10.

**Constraints:**

- `0 <= n < 109`

## 1010. Pairs of Songs With Total Durations Divisible by 60

Medium

3477133Add to ListShare

You are given a list of songs where the `ith` song has a duration of `time[i]` seconds.

Return *the number of pairs of songs for which their total duration in seconds is divisible by 60*.

Formally, we want the number of indices `i, j` such that `i < j` with `(time[i] + time[j]) % 60 == 0`.

**Example 1:**

**Input:** time = [30,20,150,100,40]

**Output:** 3

**Explanation:** Three pairs have a total duration divisible by 60:

(time[0] = 30, time[2] = 150): total duration 180

(time[1] = 20, time[3] = 100): total duration 120

(time[1] = 20, time[4] = 40): total duration 60

### Example 2:

**Input:** time = [60,60,60]

**Output:** 3

**Explanation:** All three pairs have a total duration of 120, which is divisible by 60.

### Constraints:

- $1 \leq \text{time.length} \leq 6 * 10^4$
- $1 \leq \text{time}[i] \leq 500$

## 1011. Capacity To Ship Packages Within D Days

Medium

5182112Add to ListShare

A conveyor belt has packages that must be shipped from one port to another within `days` days.

The  $i^{\text{th}}$  package on the conveyor belt has a weight of `weights[i]`. Each day, we load the ship with packages on the conveyor belt (in the order given by `weights`). We may not load more weight than the maximum weight capacity of the ship.

Return the least weight capacity of the ship that will result in all the packages on the conveyor belt being shipped within `days` days.

### Example 1:

**Input:** weights = [1,2,3,4,5,6,7,8,9,10], days = 5

**Output:** 15

**Explanation:** A ship capacity of 15 is the minimum to ship all the packages in 5 days like this:

1st day: 1, 2, 3, 4, 5

2nd day: 6, 7

3rd day: 8

4th day: 9

5th day: 10

Note that the cargo must be shipped in the order given, so using a ship of capacity 14 and splitting the packages into parts like (2, 3, 4, 5), (1, 6, 7), (8), (9), (10) is not allowed.

### Example 2:

**Input:** weights = [3,2,2,4,1,4], days = 3

**Output:** 6

**Explanation:** A ship capacity of 6 is the minimum to ship all the packages in 3 days like this:

1st day: 3, 2

2nd day: 2, 4

3rd day: 1, 4

### Example 3:

**Input:** weights = [1,2,3,1,1], days = 4

**Output:** 3

**Explanation:**

1st day: 1

2nd day: 2

3rd day: 3

4th day: 1, 1

### Constraints:

- $1 \leq \text{days} \leq \text{weights.length} \leq 5 * 10^4$
- $1 \leq \text{weights}[i] \leq 500$

## 1012. Numbers With Repeated Digits

Hard

51968Add to ListShare

Given an integer `n`, return *the number of positive integers in the range `[1, n]` that have **at least one** repeated digit.*

**Example 1:**

**Input:** `n = 20`

**Output:** 1

**Explanation:** The only positive number ( $\leq 20$ ) with at least 1 repeated digit is 11.

**Example 2:**

**Input:** `n = 100`

**Output:** 10

**Explanation:** The positive numbers ( $\leq 100$ ) with atleast 1 repeated digit are 11, 22, 33, 44, 55, 66, 77, 88, 99, and 100.

**Example 3:**

**Input:** `n = 1000`

**Output:** 262

**Constraints:**

- $1 \leq n \leq 10^9$

## 1013. Partition Array Into Three Parts With Equal Sum

Easy

1297128Add to ListShare

Given an array of integers `arr`, return `true` if we can partition the array into three **non-empty** parts with equal sums.

Formally, we can partition the array if we can find indexes  $i + 1 < j$  with `(arr[0] + arr[1] + ... + arr[i] == arr[i + 1] + arr[i + 2] + ... + arr[j - 1] == arr[j] + arr[j + 1] + ... + arr[arr.length - 1])`

**Example 1:**

**Input:** arr = [0,2,1,-6,6,-7,9,1,2,0,1]

**Output:** true

**Explanation:**  $0 + 2 + 1 = -6 + 6 - 7 + 9 + 1 = 2 + 0 + 1$

**Example 2:**

**Input:** arr = [0,2,1,-6,6,7,9,-1,2,0,1]

**Output:** false

**Example 3:**

**Input:** arr = [3,3,6,5,-2,2,5,1,-9,4]

**Output:** true

**Explanation:**  $3 + 3 = 6 = 5 - 2 + 2 + 5 + 1 - 9 + 4$

**Constraints:**

- $3 \leq \text{arr.length} \leq 5 * 10^4$
- $-10^4 \leq \text{arr}[i] \leq 10^4$

**1014. Best Sightseeing Pair**

Medium

202144Add to ListShare

You are given an integer array `values` where `values[i]` represents the value of the  $i^{\text{th}}$  sightseeing spot. Two sightseeing spots  $i$  and  $j$  have a **distance**  $j - i$  between them.

The score of a pair  $(i < j)$  of sightseeing spots is  $\text{values}[i] + \text{values}[j] + i - j$ : the sum of the values of the sightseeing spots, minus the distance between them.

Return *the maximum score of a pair of sightseeing spots*.

**Example 1:**

**Input:** values = [8,1,5,2,6]

**Output:** 11

**Explanation:**  $i = 0, j = 2, \text{values}[i] + \text{values}[j] + i - j = 8 + 5 + 0 - 2 = 11$

**Example 2:****Input:** values = [1,2]**Output:** 2**Constraints:**

- $2 \leq \text{values.length} \leq 5 * 10^4$
- $1 \leq \text{values}[i] \leq 1000$

**1015. Smallest Integer Divisible by K****Medium**

1008824Add to ListShare

Given a positive integer  $k$ , you need to find the **length** of the **smallest** positive integer  $n$  such that  $n$  is divisible by  $k$ , and  $n$  only contains the digit 1.

Return *the length of n*. If there is no such  $n$ , return -1.

**Note:**  $n$  may not fit in a 64-bit signed integer.

**Example 1:****Input:**  $k = 1$ **Output:** 1

**Explanation:** The smallest answer is  $n = 1$ , which has length 1.

**Example 2:****Input:**  $k = 2$ **Output:** -1

**Explanation:** There is no such positive integer  $n$  divisible by 2.

**Example 3:****Input:**  $k = 3$ **Output:** 3

**Explanation:** The smallest answer is  $n = 111$ , which has length 3.

**Constraints:**

- $1 \leq k \leq 10^5$

**1016. Binary String With Substrings Representing 1 To N****Medium**

275482Add to ListShare

Given a binary string  $s$  and a positive integer  $n$ , return `true` if the binary representation of all the integers in the range  $[1, n]$  are **substrings** of  $s$ , or `false` otherwise.

A **substring** is a contiguous sequence of characters within a string.

**Example 1:**

**Input:**  $s = "0110"$ ,  $n = 3$

**Output:** `true`

**Example 2:**

**Input:**  $s = "0110"$ ,  $n = 4$

**Output:** `false`

**Constraints:**

- $1 \leq s.length \leq 1000$
- $s[i]$  is either '0' or '1'.
- $1 \leq n \leq 10^9$

**1017. Convert to Base -2****Medium**

394252Add to ListShare

Given an integer  $n$ , return a binary string representing its representation in base  $-2$ .

**Note** that the returned string should not have leading zeros unless the string is `"0"`.

**Example 1:**

**Input:**  $n = 2$

**Output:** "110"

**Explanation:**  $(-2)^2 + (-2)^1 = 2$

**Example 2:**

**Input:**  $n = 3$

**Output:** "111"

**Explanation:**  $(-2)^2 + (-2)^1 + (-2)^0 = 3$

**Example 3:**

**Input:**  $n = 4$

**Output:** "100"

**Explanation:**  $(-2)^2 = 4$

**Constraints:**

- $0 \leq n \leq 10^9$

## 1018. Binary Prefix Divisible By 5

**Easy**

568149Add to ListShare

You are given a binary array `nums` (**0-indexed**).

We define  $x_i$  as the number whose binary representation is the subarray `nums[0..i]` (from most-significant-bit to least-significant-bit).

- For example, if `nums = [1, 0, 1]`, then  $x_0 = 1$ ,  $x_1 = 2$ , and  $x_2 = 5$ .

Return an array of booleans `answer` where `answer[i]` is true if  $x_i$  is divisible by 5.

**Example 1:**

**Input:** `nums = [0, 1, 1]`

**Output:** `[true, false, false]`

**Explanation:** The input numbers in binary are 0, 01, 011; which are 0, 1, and 3 in base-10.

Only the first number is divisible by 5, so `answer[0]` is true.

**Example 2:****Input:** nums = [1,1,1]**Output:** [false, false, false]**Constraints:**

- $1 \leq \text{nums.length} \leq 10^5$
- $\text{nums}[i]$  is either 0 or 1.

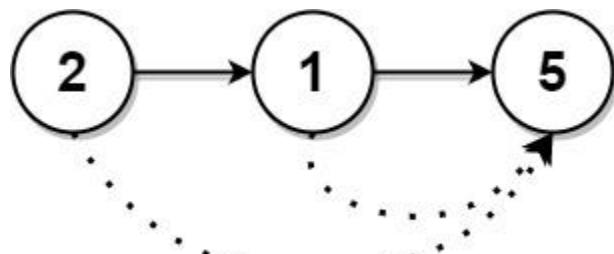
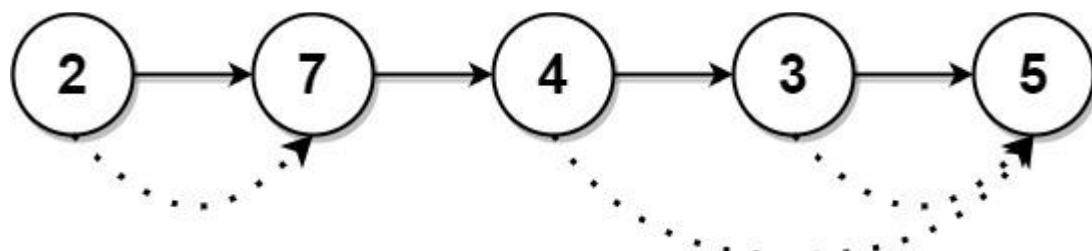
**1019. Next Greater Node In Linked List****Medium**

2486101Add to ListShare

You are given the `head` of a linked list with `n` nodes.

For each node in the list, find the value of the **next greater node**. That is, for each node, find the value of the first node that is next to it and has a **strictly larger** value than it.

Return an integer array `answer` where `answer[i]` is the value of the next greater node of the `ith` node (**1-indexed**). If the `ith` node does not have a next greater node, set `answer[i] = 0`.

**Example 1:****Input:** head = [2,1,5]**Output:** [5,5,0]**Example 2:**

**Input:** head = [2,7,4,3,5]

**Output:** [7,0,5,5,0]

**Constraints:**

- The number of nodes in the list is `n`.
- $1 \leq n \leq 10^4$
- $1 \leq \text{Node.val} \leq 10^9$

## 1020. Number of Enclaves

Medium

171136Add to ListShare

You are given an `m x n` binary matrix `grid`, where `0` represents a sea cell and `1` represents a land cell.

A **move** consists of walking from one land cell to another adjacent (**4-directionally**) land cell or walking off the boundary of the `grid`.

Return *the number of land cells in `grid` for which we cannot walk off the boundary of the `grid` in any number of **moves***.

**Example 1:**

0	0	0	0
1	0	1	0
0	1	1	0
0	0	0	0

**Input:** `grid = [[0,0,0,0],[1,0,1,0],[0,1,1,0],[0,0,0,0]]`

**Output:** 3

**Explanation:** There are three 1s that are enclosed by 0s, and one 1 that is not enclosed because its on the boundary.

**Example 2:**

0	1	1	0
0	0	1	0
0	0	1	0
0	0	0	0

**Input:** grid = [[0,1,1,0],[0,0,1,0],[0,0,1,0],[0,0,0,0]]

**Output:** 0

**Explanation:** All 1s are either on the boundary or can reach the boundary.

**Constraints:**

- $m == \text{grid.length}$
- $n == \text{grid[i].length}$
- $1 \leq m, n \leq 500$
- $\text{grid[i][j]}$  is either 0 or 1.

## 1021. Remove Outermost Parentheses

Easy

15581267Add to ListShare

A valid parentheses string is either empty "", " $(\text{A})$ ", or  $\text{A} + \text{B}$ , where  $\text{A}$  and  $\text{B}$  are valid parentheses strings, and  $+$  represents string concatenation.

- For example, "", "()", "(())()", and "(((())))" are all valid parentheses strings.

A valid parentheses string  $s$  is primitive if it is nonempty, and there does not exist a way to split it into  $s = A + B$ , with  $A$  and  $B$  nonempty valid parentheses strings.

Given a valid parentheses string  $s$ , consider its primitive decomposition:  $s = P_1 + P_2 + \dots + P_k$ , where  $P_i$  are primitive valid parentheses strings.

Return  $s$  after removing the outermost parentheses of every primitive string in the primitive decomposition of  $s$ .

### Example 1:

**Input:**  $s = "((())())()$ "

**Output:**  $"()()()$ "

**Explanation:**

The input string is  $"((())())()$ ", with primitive decomposition  $"((())())" + "()"$ .

After removing outer parentheses of each part, this is  $"()()" + "()" = "()()()$ .

### Example 2:

**Input:**  $s = "((())())((())())()$ "

**Output:**  $"()()()()()$ "

**Explanation:**

The input string is  $"((())())((())())()$ ", with primitive decomposition  $"((())())" + "()" + "((())())"$ .

After removing outer parentheses of each part, this is  $"()()" + "()" + "()()()" = "()()()()()$ .

### Example 3:

**Input:**  $s = "()()$ "

**Output:**  $"$ "

**Explanation:**

The input string is  $"()()$ , with primitive decomposition  $"()" + "()"$ .

After removing outer parentheses of each part, this is  $"" + "()" = ""$ .

### Constraints:

- $1 \leq s.length \leq 10^5$
- $s[i]$  is either '(' or ')'.  
•  $s$  is a valid parentheses string.

## 1022. Sum of Root To Leaf Binary Numbers

Easy

2805163 Add to List Share

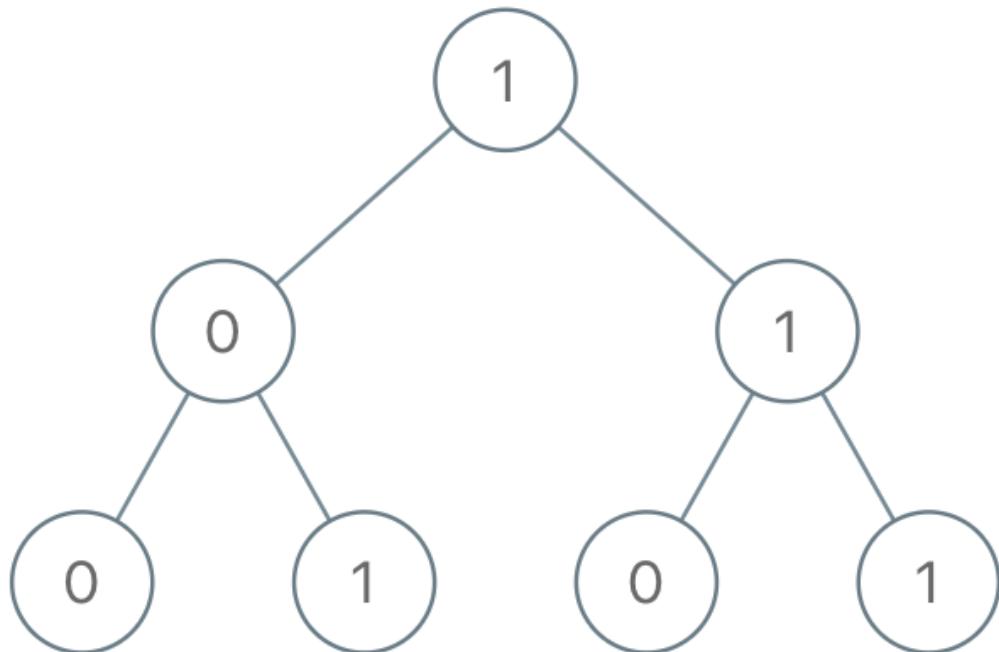
You are given the `root` of a binary tree where each node has a value `0` or `1`. Each root-to-leaf path represents a binary number starting with the most significant bit.

- For example, if the path is `0 -> 1 -> 1 -> 0 -> 1`, then this could represent `01101` in binary, which is `13`.

For all leaves in the tree, consider the numbers represented by the path from the `root` to that leaf. Return *the sum of these numbers*.

The test cases are generated so that the answer fits in a **32-bits** integer.

**Example 1:**



**Input:** `root = [1,0,1,0,1,0,1]`

**Output:** 22

**Explanation:**  $(100) + (101) + (110) + (111) = 4 + 5 + 6 + 7 = 22$

**Example 2:**

**Input:** root = [0]

**Output:** 0

**Constraints:**

- The number of nodes in the tree is in the range  $[1, 1000]$ .
- `Node.val` is 0 or 1.

## 1023. Camelcase Matching

Medium

648240Add to ListShare

Given an array of strings `queries` and a string `pattern`, return a boolean array `answer` where `answer[i]` is `true` if `queries[i]` matches `pattern`, and `false` otherwise.

A query word `queries[i]` matches `pattern` if you can insert lowercase English letters `pattern` so that it equals the query. You may insert each character at any position and you may not insert any characters.

**Example 1:**

**Input:** `queries = ["FooBar", "FooBarTest", "FootBall", "FrameBuffer", "ForceFeedBack"]`, `pattern = "FB"`

**Output:** [true, false, true, true, false]

**Explanation:** "FooBar" can be generated like this "F" + "oo" + "B" + "ar".

"FootBall" can be generated like this "F" + "oot" + "B" + "all".

"FrameBuffer" can be generated like this "F" + "rame" + "B" + "uffer".

**Example 2:**

**Input:** `queries = ["FooBar", "FooBarTest", "FootBall", "FrameBuffer", "ForceFeedBack"]`, `pattern = "FoBa"`

**Output:** [true, false, true, false, false]

**Explanation:** "FooBar" can be generated like this "Fo" + "o" + "Ba" + "r".

"FootBall" can be generated like this "Fo" + "ot" + "Ba" + "ll".

**Example 3:**

**Input:** queries = ["FooBar", "FooBarTest", "FootBall", "FrameBuffer", "ForceFeedBack"], pattern = "FoBaT"

**Output:** [false, true, false, false, false]

**Explanation:** "FooBarTest" can be generated like this "Fo" + "o" + "Ba" + "r" + "T" + "est".

**Constraints:**

- $1 \leq \text{pattern.length}, \text{queries.length} \leq 100$
- $1 \leq \text{queries[i].length} \leq 100$
- `queries[i]` and `pattern` consist of English letters.

## 1024. Video Stitching

Medium

137048Add to ListShare

You are given a series of video clips from a sporting event that lasted `time` seconds. These video clips can be overlapping with each other and have varying lengths.

Each video clip is described by an array `clips` where `clips[i] = [starti, endi]` indicates that the *i*th clip started at `starti` and ended at `endi`.

We can cut these clips into segments freely.

- For example, a clip `[0, 7]` can be cut into segments `[0, 1] + [1, 3] + [3, 7]`.

Return *the minimum number of clips needed so that we can cut the clips into segments that cover the entire sporting event* `[0, time]`. If the task is impossible, return `-1`.

**Example 1:**

**Input:** clips = [[0,2],[4,6],[8,10],[1,9],[1,5],[5,9]], time = 10

**Output:** 3

**Explanation:** We take the clips `[0,2]`, `[8,10]`, `[1,9]`; a total of 3 clips.

Then, we can reconstruct the sporting event as follows:

We cut `[1,9]` into segments `[1,2] + [2,8] + [8,9]`.

Now we have segments  $[0,2] + [2,8] + [8,10]$  which cover the sporting event  $[0, 10]$ .

**Example 2:**

**Input:** `clips = [[0,1],[1,2]]`, `time = 5`

**Output:** `-1`

**Explanation:** We cannot cover  $[0,5]$  with only  $[0,1]$  and  $[1,2]$ .

**Example 3:**

**Input:** `clips = [[0,1],[6,8],[0,2],[5,6],[0,4],[0,3],[6,7],[1,3],[4,7],[1,4],[2,5],[2,6],[3,4],[4,5],[5,7],[6,9]]`, `time = 9`

**Output:** `3`

**Explanation:** We can take clips  $[0,4]$ ,  $[4,7]$ , and  $[6,9]$ .

**Constraints:**

- `1 <= clips.length <= 100`
- `0 <= starti <= endi <= 100`
- `1 <= time <= 100`

## 1025. Divisor Game

**Easy**

15603452 Add to List Share

Alice and Bob take turns playing a game, with Alice starting first.

Initially, there is a number `n` on the chalkboard. On each player's turn, that player makes a move consisting of:

- Choosing any `x` with `0 < x < n` and `n % x == 0`.
- Replacing the number `n` on the chalkboard with `n - x`.

Also, if a player cannot make a move, they lose the game.

Return `true` if and only if Alice wins the game, assuming both players play optimally.

**Example 1:**

**Input:** `n = 2`

**Output:** true

**Explanation:** Alice chooses 1, and Bob has no more moves.

**Example 2:**

**Input:** n = 3

**Output:** false

**Explanation:** Alice chooses 1, Bob chooses 1, and Alice has no more moves.

**Constraints:**

- $1 \leq n \leq 1000$

## 1026. Maximum Difference Between Node and Ancestor

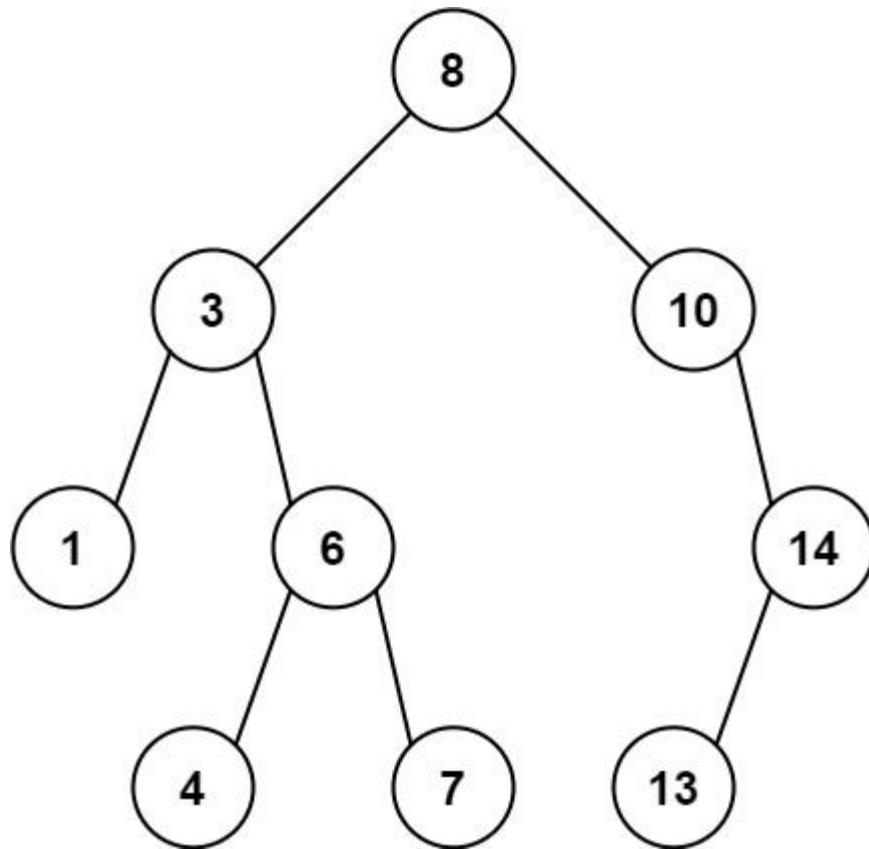
Medium

260868Add to ListShare

Given the `root` of a binary tree, find the maximum value `v` for which there exist **different** nodes `a` and `b` where `v = |a.val - b.val|` and `a` is an ancestor of `b`.

A node `a` is an ancestor of `b` if either: any child of `a` is equal to `b` or any child of `a` is an ancestor of `b`.

**Example 1:**



**Input:** root = [8,3,10,1,6,null,14,null,null,4,7,13]

**Output:** 7

**Explanation:** We have various ancestor-node differences, some of which are given below :

$$|8 - 3| = 5$$

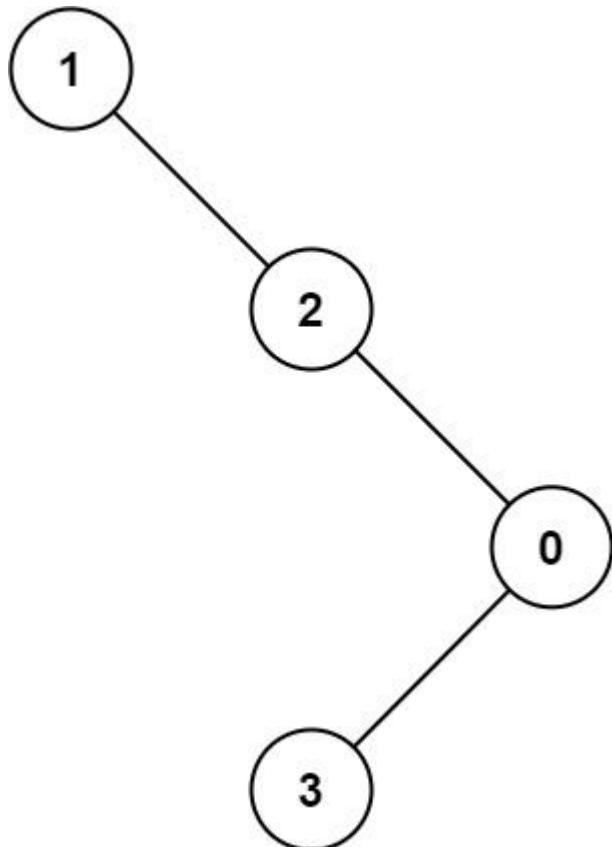
$$|3 - 7| = 4$$

$$|8 - 1| = 7$$

$$|10 - 13| = 3$$

Among all possible differences, the maximum value of 7 is obtained by  $|8 - 1| = 7$ .

**Example 2:**



**Input:** root = [1,null,2,null,0,3]

**Output:** 3

**Constraints:**

- The number of nodes in the tree is in the range [2, 5000].
- $0 \leq \text{Node.val} \leq 10^5$

## 1027. Longest Arithmetic Subsequence

Medium

2475115Add to ListShare

Given an array `nums` of integers, return the **length** of the longest arithmetic subsequence in `nums`.

Recall that a *subsequence* of an array `nums` is a list `nums[i1], nums[i2], ..., nums[ik]` with  $0 \leq i_1 < i_2 < \dots < i_k \leq \text{nums.length} - 1$ , and that a sequence `seq` is *arithmetic* if `seq[i+1] - seq[i]` are all the same value (for  $0 \leq i < \text{seq.length} - 1$ ).

**Example 1:**

**Input:** nums = [3,6,9,12]

**Output:** 4

**Explanation:**

The whole array is an arithmetic sequence with steps of length = 3.

**Example 2:**

**Input:** nums = [9,4,7,2,10]

**Output:** 3

**Explanation:**

The longest arithmetic subsequence is [4,7,10].

**Example 3:**

**Input:** nums = [20,1,15,3,10,5,8]

**Output:** 4

**Explanation:**

The longest arithmetic subsequence is [20,15,10,5].

**Constraints:**

- $2 \leq \text{nums.length} \leq 1000$
- $0 \leq \text{nums}[i] \leq 500$

## 1028. Recover a Tree From Preorder Traversal

Hard

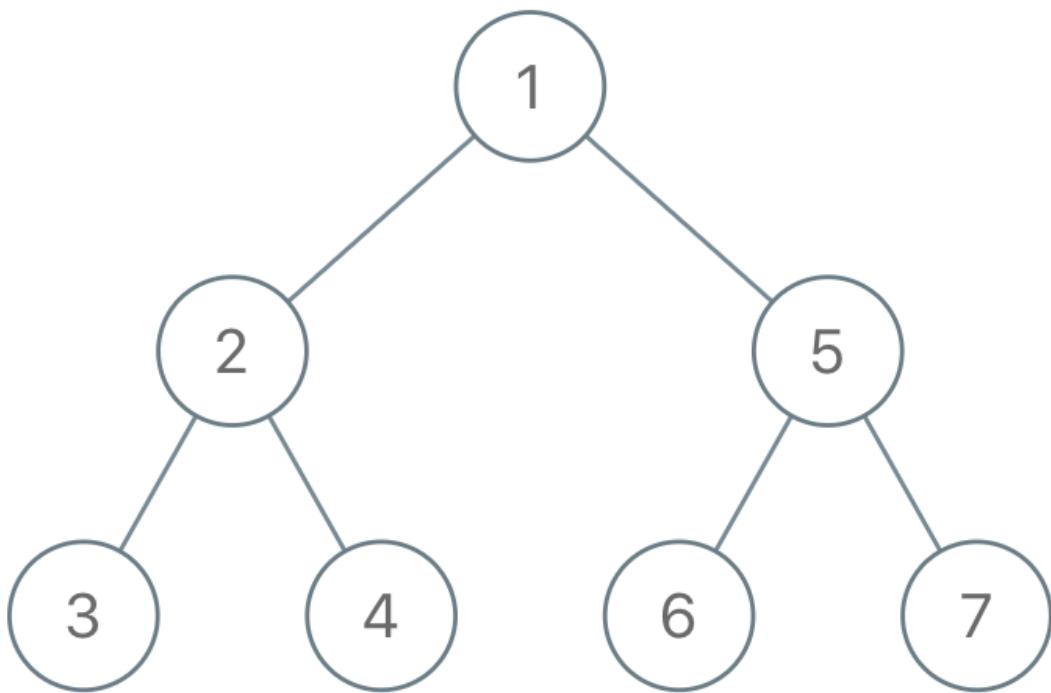
122236Add to ListShare

We run a preorder depth-first search (DFS) on the `root` of a binary tree.

At each node in this traversal, we output `D` dashes (where `D` is the depth of this node), then we output the value of this node. If the depth of a node is `D`, the depth of its immediate child is `D + 1`. The depth of the `root` node is `0`.

If a node has only one child, that child is guaranteed to be **the left child**.

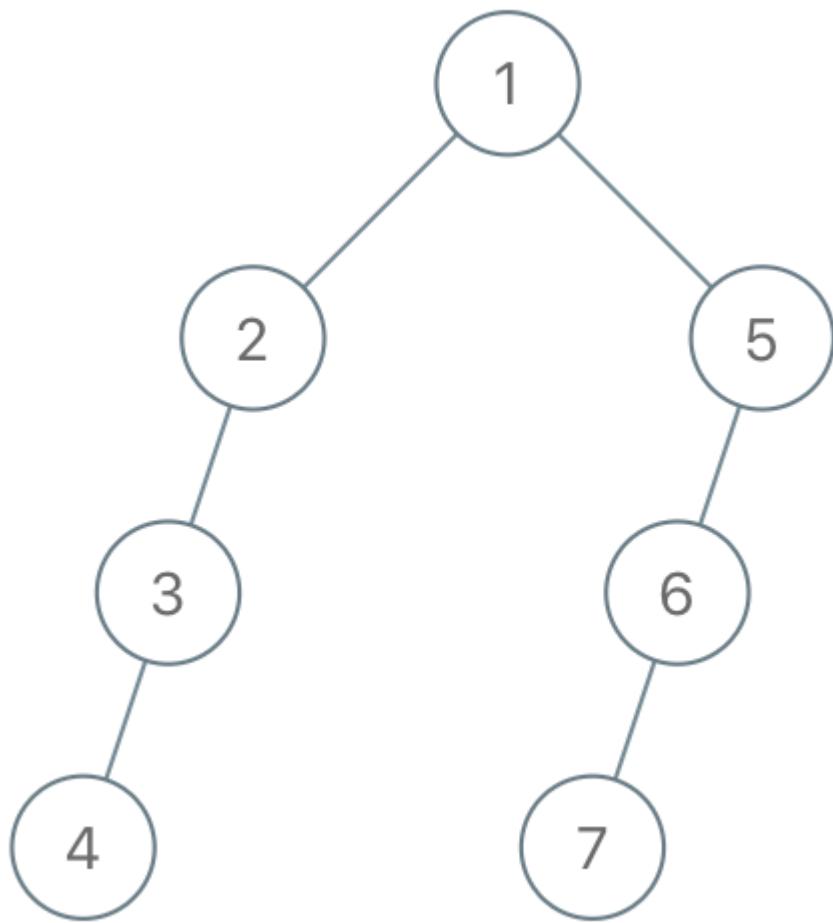
Given the output `traversal` of this traversal, recover the tree and return *its root*.

**Example 1:**

**Input:** traversal = "1-2--3--4-5--6--7"

**Output:** [1,2,5,3,4,6,7]

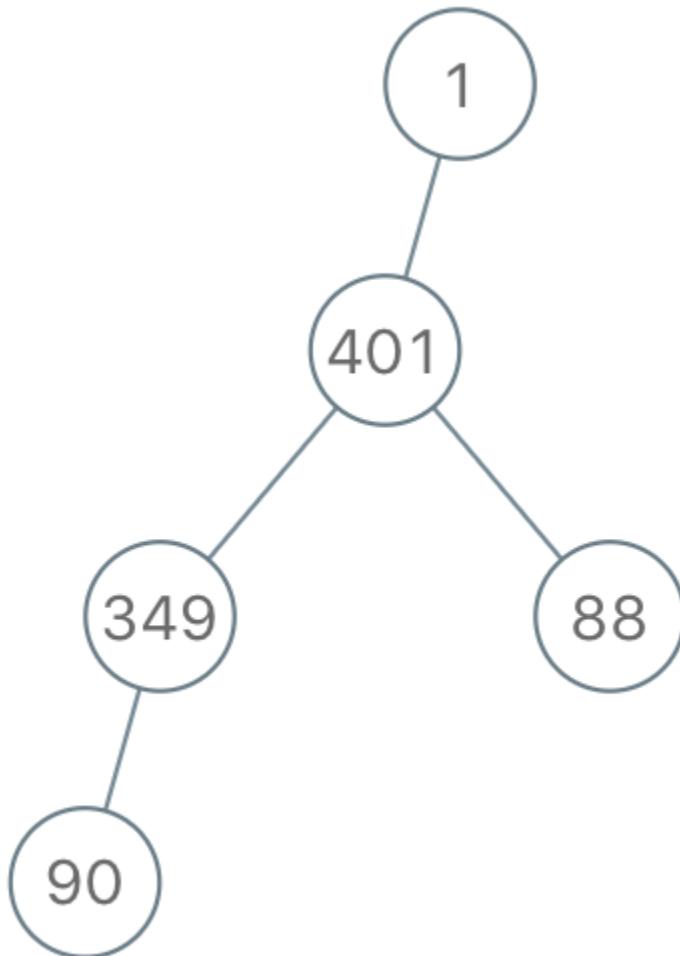
**Example 2:**



**Input:** traversal = "1-2--3---4-5---6---7"

**Output:** [1,2,5,3,null,6,null,4,null,7]

**Example 3:**



**Input:** traversal = "1-401--349---90--88"

**Output:** [1,401,null,349,88,90]

#### Constraints:

- The number of nodes in the original tree is in the range [1, 1000].
- $1 \leq \text{Node.val} \leq 10^9$

## 1029. Two City Scheduling

Medium

3914292Add to ListShare

A company is planning to interview  $2n$  people. Given the array `costs` where  $\text{costs}[i] = [\text{aCost}_i, \text{bCost}_i]$ , the cost of flying the  $i^{\text{th}}$  person to city  $a$  is  $\text{aCost}_i$ , and the cost of flying the  $i^{\text{th}}$  person to city  $b$  is  $\text{bCost}_i$ .

Return the minimum cost to fly every person to a city such that exactly  $n$  people arrive in each city.

### Example 1:

**Input:** `costs = [[10,20],[30,200],[400,50],[30,20]]`

**Output:** 110

#### Explanation:

The first person goes to city A for a cost of 10.

The second person goes to city A for a cost of 30.

The third person goes to city B for a cost of 50.

The fourth person goes to city B for a cost of 20.

The total minimum cost is  $10 + 30 + 50 + 20 = 110$  to have half the people interviewing in each city.

### Example 2:

**Input:** `costs = [[259,770],[448,54],[926,667],[184,139],[840,118],[577,469]]`

**Output:** 1859

### Example 3:

**Input:** `costs = [[515,563],[451,713],[537,709],[343,819],[855,779],[457,60],[650,359],[631,42]]`

**Output:** 3086

### Constraints:

- $2 * n == \text{costs.length}$
- $2 \leq \text{costs.length} \leq 100$
- `costs.length` is even.
- $1 \leq \text{aCost}_i, \text{bCost}_i \leq 1000$

## 1030. Matrix Cells in Distance Order

Easy

550246Add to ListShare

You are given four integers `row`, `cols`, `rCenter`, and `cCenter`. There is a `rows x cols` matrix and you are on the cell with the coordinates `(rCenter, cCenter)`.

Return the coordinates of all cells in the matrix, sorted by their **distance** from `(rCenter, cCenter)` from the smallest distance to the largest distance. You may return the answer in **any order** that satisfies this condition.

The **distance** between two cells `(r1, c1)` and `(r2, c2)` is  $|r_1 - r_2| + |c_1 - c_2|$ .

### Example 1:

**Input:** `rows = 1, cols = 2, rCenter = 0, cCenter = 0`

**Output:** `[[0,0],[0,1]]`

**Explanation:** The distances from `(0, 0)` to other cells are: `[0,1]`

### Example 2:

**Input:** `rows = 2, cols = 2, rCenter = 0, cCenter = 1`

**Output:** `[[0,1],[0,0],[1,1],[1,0]]`

**Explanation:** The distances from `(0, 1)` to other cells are: `[0,1,1,2]`

The answer `[[0,1],[1,1],[0,0],[1,0]]` would also be accepted as correct.

### Example 3:

**Input:** `rows = 2, cols = 3, rCenter = 1, cCenter = 2`

**Output:** `[[1,2],[0,2],[1,1],[0,1],[1,0],[0,0]]`

**Explanation:** The distances from `(1, 2)` to other cells are: `[0,1,1,2,2,3]`

There are other answers that would also be accepted as correct, such as `[[1,2],[1,1],[0,2],[1,0],[0,1],[0,0]]`.

### Constraints:

- $1 \leq \text{rows}, \text{cols} \leq 100$
- $0 \leq \text{rCenter} < \text{rows}$
- $0 \leq \text{cCenter} < \text{cols}$

## 1031. Maximum Sum of Two Non-Overlapping Subarrays

Medium

202273Add to ListShare

Given an integer array `nums` and two integers `firstLen` and `secondLen`, return *the maximum sum of elements in two non-overlapping subarrays with lengths `firstLen` and `secondLen`*.

The array with length `firstLen` could occur before or after the array with length `secondLen`, but they have to be non-overlapping.

A **subarray** is a **contiguous** part of an array.

**Example 1:**

**Input:** `nums = [0,6,5,2,2,5,1,9,4]`, `firstLen = 1`, `secondLen = 2`

**Output:** 20

**Explanation:** One choice of subarrays is [9] with length 1, and [6,5] with length 2.

**Example 2:**

**Input:** `nums = [3,8,1,3,2,1,8,9,0]`, `firstLen = 3`, `secondLen = 2`

**Output:** 29

**Explanation:** One choice of subarrays is [3,8,1] with length 3, and [8,9] with length 2.

**Example 3:**

**Input:** `nums = [2,1,5,6,0,9,5,0,3,8]`, `firstLen = 4`, `secondLen = 3`

**Output:** 31

**Explanation:** One choice of subarrays is [5,6,0,9] with length 4, and [0,3,8] with length 3.

**Constraints:**

- $1 \leq \text{firstLen}, \text{secondLen} \leq 1000$
- $2 \leq \text{firstLen} + \text{secondLen} \leq 1000$
- $\text{firstLen} + \text{secondLen} \leq \text{nums.length} \leq 1000$
- $0 \leq \text{nums}[i] \leq 1000$

## 1032. Stream of Characters

**Hard**

1615174Add to ListShare

Design an algorithm that accepts a stream of characters and checks if a suffix of these characters is a string of a given array of strings `words`.

For example, if `words = ["abc", "xyz"]` and the stream added the four characters (one by one) '`a`', '`x`', '`y`', and '`z`', your algorithm should detect that the suffix "`xyz`" of the characters "`axyz`" matches "`xyz`" from `words`.

Implement the `StreamChecker` class:

- `StreamChecker(String[] words)` Initializes the object with the strings array `words`.
- `boolean query(char letter)` Accepts a new character from the stream and returns `true` if any non-empty suffix from the stream forms a word that is in `words`.

**Example 1:****Input**

```
["StreamChecker", "query", "query"]
[[["cd", "f", "kl"]], ["a"], ["b"], ["c"], ["d"], ["e"], ["f"], ["g"], ["h"], ["i"], ["j"], ["k"], ["l"]]
```

**Output**

```
[null, false, false, false, true, false, true, false, false, false, false, false, true]
```

**Explanation**

```
StreamChecker streamChecker = new StreamChecker(["cd", "f", "kl"]);
streamChecker.query("a"); // return False
streamChecker.query("b"); // return False
streamChecker.query("c"); // return False
streamChecker.query("d"); // return True, because 'cd' is in the wordlist
streamChecker.query("e"); // return False
streamChecker.query("f"); // return True, because 'f' is in the wordlist
streamChecker.query("g"); // return False
```

```
streamChecker.query("h"); // return False
streamChecker.query("i"); // return False
streamChecker.query("j"); // return False
streamChecker.query("k"); // return False
streamChecker.query("l"); // return True, because 'kl' is in the wordlist
```

### Constraints:

- $1 \leq \text{words.length} \leq 2000$
- $1 \leq \text{words[i].length} \leq 200$
- `words[i]` consists of lowercase English letters.
- `letter` is a lowercase English letter.
- At most  $4 * 10^4$  calls will be made to `query`.

## 1033. Moving Stones Until Consecutive

Medium

167608Add to ListShare

There are three stones in different positions on the X-axis. You are given three integers `a`, `b`, and `c`, the positions of the stones.

In one move, you pick up a stone at an endpoint (i.e., either the lowest or highest position stone), and move it to an unoccupied position between those endpoints. Formally, let's say the stones are currently at positions `x`, `y`, and `z` with  $x < y < z$ . You pick up the stone at either position `x` or position `z`, and move that stone to an integer position `k`, with  $x < k < z$  and  $k \neq y$ .

The game ends when you cannot make any more moves (i.e., the stones are in three consecutive positions).

Return an integer array `answer` of length 2 where:

- `answer[0]` is the minimum number of moves you can play, and
- `answer[1]` is the maximum number of moves you can play.

### Example 1:

**Input:** `a = 1, b = 2, c = 5`

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

**Explanation:** Move the stone from 5 to 3, or move the stone from 5 to 4 to 3.

**Example 2:**

**Input:** a = 4, b = 3, c = 2

**Output:** [0,0]

**Explanation:** We cannot make any moves.

**Example 3:**

**Input:** a = 3, b = 5, c = 1

**Output:** [1,2]

**Explanation:** Move the stone from 1 to 4; or move the stone from 1 to 2 to 4.

**Constraints:**

- $1 \leq a, b, c \leq 100$
- a, b, and c have different values.

## 1034. Coloring A Border

Medium

525692Add to ListShare

You are given an  $m \times n$  integer matrix `grid`, and three integers `row`, `col`, and `color`. Each value in the grid represents the color of the grid square at that location.

Two squares belong to the same **connected component** if they have the same color and are next to each other in any of the 4 directions.

The **border of a connected component** is all the squares in the connected component that are either **4-directionally** adjacent to a square not in the component, or on the boundary of the grid (the first or last row or column).

You should color the **border** of the **connected component** that contains the square `grid[row][col]` with `color`.

Return *the final grid*.

**Example 1:**

**Input:** `grid = [[1,1],[1,2]]`, `row = 0`, `col = 0`, `color = 3`

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

**Example 2:**

**Input:** grid = [[1,2,2],[2,3,2]], row = 0, col = 1, color = 3

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

**Example 3:**

**Input:** grid = [[1,1,1],[1,1,1],[1,1,1]], row = 1, col = 1, color = 2

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

**Constraints:**

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 50`
- `1 <= grid[i][j], color <= 1000`
- `0 <= row < m`
- `0 <= col < n`

**1035. Uncrossed Lines**

Medium

185128Add to ListShare

You are given two integer arrays `nums1` and `nums2`. We write the integers of `nums1` and `nums2` (in the order they are given) on two separate horizontal lines.

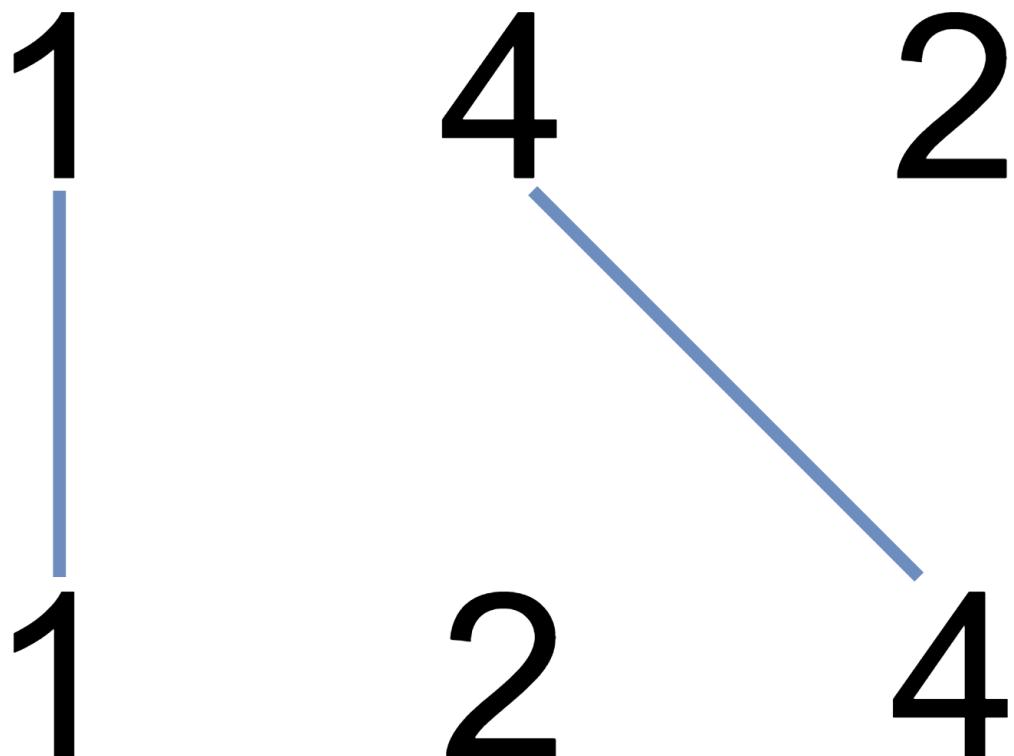
We may draw connecting lines: a straight line connecting two numbers `nums1[i]` and `nums2[j]` such that:

- `nums1[i] == nums2[j]`, and
- the line we draw does not intersect any other connecting (non-horizontal) line.

Note that a connecting line cannot intersect even at the endpoints (i.e., each number can only belong to one connecting line).

Return *the maximum number of connecting lines we can draw in this way*.

**Example 1:**



**Input:** `nums1 = [1,4,2], nums2 = [1,2,4]`

**Output:** 2

**Explanation:** We can draw 2 uncrossed lines as in the diagram.

We cannot draw 3 uncrossed lines, because the line from  $\text{nums1}[1] = 4$  to  $\text{nums2}[2] = 4$  will intersect the line from  $\text{nums1}[2] = 2$  to  $\text{nums2}[1] = 2$ .

**Example 2:**

**Input:** `nums1 = [2,5,1,2,5], nums2 = [10,5,2,1,5,2]`

**Output:** 3

**Example 3:**

**Input:** `nums1 = [1,3,7,1,7,5], nums2 = [1,9,2,5,1]`

**Output:** 2

**Constraints:**

- $1 \leq \text{nums1.length}, \text{nums2.length} \leq 500$
- $1 \leq \text{nums1}[i], \text{nums2}[j] \leq 2000$

## 1036. Escape a Large Maze

Hard

518152Add to ListShare

There is a 1 million by 1 million grid on an XY-plane, and the coordinates of each grid square are  $(x, y)$ .

We start at the  $\text{source} = [s_x, s_y]$  square and want to reach the  $\text{target} = [t_x, t_y]$  square. There is also an array of `blocked` squares, where each  $\text{blocked}[i] = [x_i, y_i]$  represents a blocked square with coordinates  $(x_i, y_i)$ .

Each move, we can walk one square north, east, south, or west if the square is **not** in the array of `blocked` squares. We are also not allowed to walk outside of the grid.

Return `true` if and only if it is possible to reach the `target` square from the `source` square through a sequence of valid moves.

### Example 1:

**Input:** `blocked = [[0,1],[1,0]]`, `source = [0,0]`, `target = [0,2]`

**Output:** `false`

**Explanation:** The target square is inaccessible starting from the source square because we cannot move.

We cannot move north or east because those squares are blocked.

We cannot move south or west because we cannot go outside of the grid.

### Example 2:

**Input:** `blocked = []`, `source = [0,0]`, `target = [999999,999999]`

**Output:** `true`

**Explanation:** Because there are no blocked cells, it is possible to reach the target square.

### Constraints:

- $0 \leq \text{blocked.length} \leq 200$
- $\text{blocked}[i].length == 2$

- $0 \leq x_i, y_i < 10^6$
- `source.length == target.length == 2`
- $0 \leq s_x, s_y, t_x, t_y < 10^6$
- `source != target`
- It is guaranteed that `source` and `target` are not blocked.

## 1037. Valid Boomerang

Easy

268399Add to ListShare

Given an array `points` where `points[i] = [xi, yi]` represents a point on the **X-Y** plane, return `true` if these points are a **boomerang**.

A **boomerang** is a set of three points that are **all distinct** and **not in a straight line**.

**Example 1:**

**Input:** `points = [[1,1],[2,3],[3,2]]`

**Output:** `true`

**Example 2:**

**Input:** `points = [[1,1],[2,2],[3,3]]`

**Output:** `false`

**Constraints:**

- `points.length == 3`
- `points[i].length == 2`
- $0 \leq x_i, y_i \leq 100$

## 1038. Binary Search Tree to Greater Sum Tree

Medium

2969140Add to ListShare

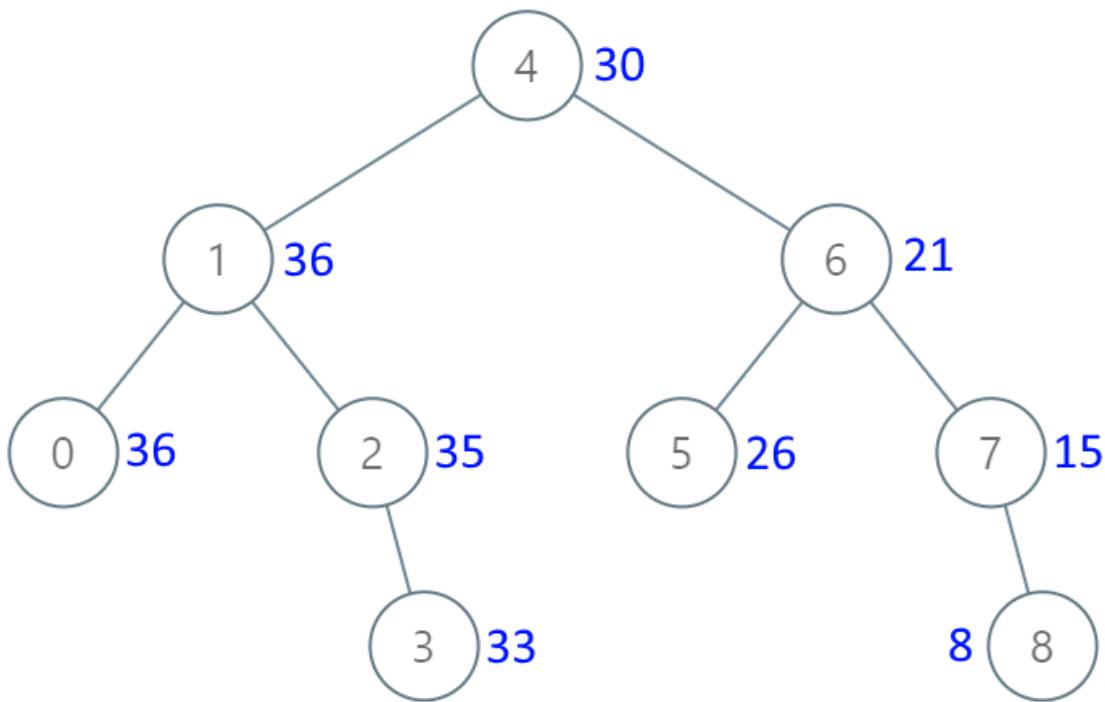
Given the `root` of a Binary Search Tree (BST), convert it to a Greater Tree such that every key of the original BST is changed to the original key plus the sum of all keys greater than the original key in BST.

As a reminder, a *binary search tree* is a tree that satisfies these constraints:

- The left subtree of a node contains only nodes with keys **less than** the node's key.
- The right subtree of a node contains only nodes with keys **greater than** the node's key.

- Both the left and right subtrees must also be binary search trees.

**Example 1:**



**Input:** root = [4,1,6,0,2,5,7,null,null,null,3,null,null,null,8]

**Output:** [30,36,21,36,35,26,15,null,null,null,33,null,null,null,8]

**Example 2:**

**Input:** root = [0,null,1]

**Output:** [1,null,1]

**Constraints:**

- The number of nodes in the tree is in the range [1, 100].
- $0 \leq \text{Node.val} \leq 100$
- All the values in the tree are **unique**.

## 1039. Minimum Score Triangulation of Polygon

Medium

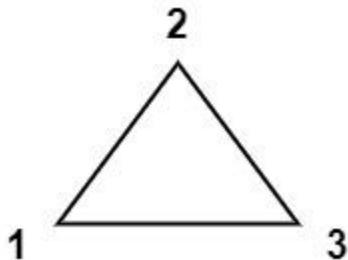
1277126Add to ListShare

You have a convex  $n$ -sided polygon where each vertex has an integer value. You are given an integer array `values` where `values[i]` is the value of the  $i^{\text{th}}$  vertex (i.e., **clockwise order**).

You will **triangulate** the polygon into  $n - 2$  triangles. For each triangle, the value of that triangle is the product of the values of its vertices, and the total score of the triangulation is the sum of these values over all  $n - 2$  triangles in the triangulation.

Return *the smallest possible total score that you can achieve with some triangulation of the polygon.*

**Example 1:**

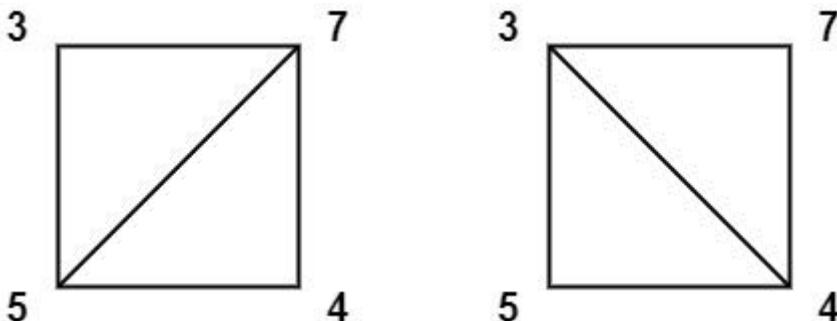


**Input:** `values = [1,2,3]`

**Output:** 6

**Explanation:** The polygon is already triangulated, and the score of the only triangle is 6.

**Example 2:**



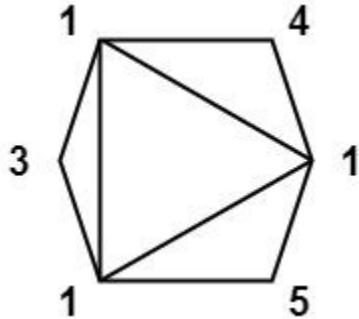
**Input:** `values = [3,7,4,5]`

**Output:** 144

**Explanation:** There are two triangulations, with possible scores:  $3*7*5 + 4*5*7 = 245$ , or  $3*4*5 + 3*4*7 = 144$ .

The minimum score is 144.

**Example 3:**



**Input:** values = [1,3,1,4,1,5]

**Output:** 13

**Explanation:** The minimum score triangulation has score  $1*1*3 + 1*1*4 + 1*1*5 + 1*1*1 = 13$ .

**Constraints:**

- `n == values.length`
- `3 <= n <= 50`
- `1 <= values[i] <= 100`

## 1040. Moving Stones Until Consecutive II

Medium

316530Add to ListShare

There are some stones in different positions on the X-axis. You are given an integer array `stones`, the positions of the stones.

Call a stone an **endpoint stone** if it has the smallest or largest position. In one move, you pick up an **endpoint stone** and move it to an unoccupied position so that it is no longer an **endpoint stone**.

- In particular, if the stones are at say, `stones = [1, 2, 5]`, you cannot move the endpoint stone at position 5, since moving it to any position (such as 0, or 3) will still keep that stone as an endpoint stone.

The game ends when you cannot make any more moves (i.e., the stones are in three consecutive positions).

Return an integer array `answer` of length 2 where:

- `answer[0]` is the minimum number of moves you can play, and

- `answer[1]` is the maximum number of moves you can play.

### Example 1:

**Input:** stones = [7,4,9]

**Output:** [1,2]

**Explanation:** We can move 4  $\rightarrow$  8 for one move to finish the game.

Or, we can move 9  $\rightarrow$  5, 4  $\rightarrow$  6 for two moves to finish the game.

### Example 2:

**Input:** stones = [6,5,4,3,10]

**Output:** [2,3]

**Explanation:** We can move 3  $\rightarrow$  8 then 10  $\rightarrow$  7 to finish the game.

Or, we can move 3  $\rightarrow$  7, 4  $\rightarrow$  8, 5  $\rightarrow$  9 to finish the game.

Notice we cannot move 10  $\rightarrow$  2 to finish the game, because that would be an illegal move.

### Constraints:

- $3 \leq \text{stones.length} \leq 10^4$
- $1 \leq \text{stones}[i] \leq 10^9$
- All the values of `stones` are **unique**.

## 1041. Robot Bounded In Circle

Medium

3253639Add to ListShare

On an infinite plane, a robot initially stands at  $(0, 0)$  and faces north. Note that:

- The **north direction** is the positive direction of the y-axis.
- The **south direction** is the negative direction of the y-axis.
- The **east direction** is the positive direction of the x-axis.
- The **west direction** is the negative direction of the x-axis.

The robot can receive one of three instructions:

- `"G"`: go straight 1 unit.
- `"L"`: turn 90 degrees to the left (i.e., anti-clockwise direction).

- "R": turn 90 degrees to the right (i.e., clockwise direction).

The robot performs the `instructions` given in order, and repeats them forever.

Return `true` if and only if there exists a circle in the plane such that the robot never leaves the circle.

### Example 1:

**Input:** `instructions = "GGLLGG"`

**Output:** `true`

**Explanation:** The robot is initially at  $(0, 0)$  facing the north direction.

"G": move one step. Position:  $(0, 1)$ . Direction: North.

"G": move one step. Position:  $(0, 2)$ . Direction: North.

"L": turn 90 degrees anti-clockwise. Position:  $(0, 2)$ . Direction: West.

"L": turn 90 degrees anti-clockwise. Position:  $(0, 2)$ . Direction: South.

"G": move one step. Position:  $(0, 1)$ . Direction: South.

"G": move one step. Position:  $(0, 0)$ . Direction: South.

Repeating the instructions, the robot goes into the cycle:  $(0, 0) \rightarrow (0, 1) \rightarrow (0, 2) \rightarrow (0, 1) \rightarrow (0, 0)$ .

Based on that, we return `true`.

### Example 2:

**Input:** `instructions = "GG"`

**Output:** `false`

**Explanation:** The robot is initially at  $(0, 0)$  facing the north direction.

"G": move one step. Position:  $(0, 1)$ . Direction: North.

"G": move one step. Position:  $(0, 2)$ . Direction: North.

Repeating the instructions, keeps advancing in the north direction and does not go into cycles.

Based on that, we return `false`.

### Example 3:

**Input:** `instructions = "GL"`

**Output:** true

**Explanation:** The robot is initially at (0, 0) facing the north direction.

"G": move one step. Position: (0, 1). Direction: North.

"L": turn 90 degrees anti-clockwise. Position: (0, 1). Direction: West.

"G": move one step. Position: (-1, 1). Direction: West.

"L": turn 90 degrees anti-clockwise. Position: (-1, 1). Direction: South.

"G": move one step. Position: (-1, 0). Direction: South.

"L": turn 90 degrees anti-clockwise. Position: (-1, 0). Direction: East.

"G": move one step. Position: (0, 0). Direction: East.

"L": turn 90 degrees anti-clockwise. Position: (0, 0). Direction: North.

Repeating the instructions, the robot goes into the cycle: (0, 0) --> (0, 1) --> (-1, 1) --> (-1, 0) --> (0, 0).

Based on that, we return true.

### Constraints:

- `1 <= instructions.length <= 100`
- `instructions[i]` is 'G', 'L' or, 'R'.

## 1042. Flower Planting With No Adjacent

### Medium

1003673Add to ListShare

You have `n` gardens, labeled from `1` to `n`, and an array `paths` where `paths[i] = [xi, yi]` describes a bidirectional path between garden `xi` to garden `yi`. In each garden, you want to plant one of 4 types of flowers.

All gardens have **at most 3** paths coming into or leaving it.

Your task is to choose a flower type for each garden such that, for any two gardens connected by a path, they have different types of flowers.

Return **any such a choice as an array** `answer`, where `answer[i]` is the type of flower planted in the `(i+1)th` garden. The flower types are denoted 1, 2, 3, or 4. It is guaranteed an answer exists.

### Example 1:

**Input:** n = 3, paths = [[1,2],[2,3],[3,1]]

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

**Explanation:**

Gardens 1 and 2 have different types.

Gardens 2 and 3 have different types.

Gardens 3 and 1 have different types.

Hence, [1,2,3] is a valid answer. Other valid answers include [1,2,4], [1,4,2], and [3,2,1].

**Example 2:**

**Input:** n = 4, paths = [[1,2],[3,4]]

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

**Example 3:**

**Input:** n = 4, paths = [[1,2],[2,3],[3,4],[4,1],[1,3],[2,4]]

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

**Constraints:**

- $1 \leq n \leq 10^4$
- $0 \leq \text{paths.length} \leq 2 * 10^4$
- $\text{paths}[i].length == 2$
- $1 \leq x_i, y_i \leq n$
- $x_i \neq y_i$
- Every garden has **at most 3** paths coming into or leaving it.

## 1043. Partition Array for Maximum Sum

**Medium**

2724206Add to ListShare

Given an integer array `arr`, partition the array into (contiguous) subarrays of length **at most** `k`. After partitioning, each subarray has their values changed to become the maximum value of that subarray.

Return *the largest sum of the given array after partitioning*. Test cases are generated so that the answer fits in a **32-bit** integer.

**Example 1:**

**Input:** arr = [1,15,7,9,2,5,10], k = 3

**Output:** 84

**Explanation:** arr becomes [15,15,15,9,10,10,10]

**Example 2:**

**Input:** arr = [1,4,1,5,7,3,6,1,9,9,3], k = 4

**Output:** 83

**Example 3:**

**Input:** arr = [1], k = 1

**Output:** 1

**Constraints:**

- $1 \leq \text{arr.length} \leq 500$
- $0 \leq \text{arr}[i] \leq 10^9$
- $1 \leq k \leq \text{arr.length}$

## 1044. Longest Duplicate Substring

Hard

1867356Add to ListShare

Given a string  $s$ , consider all *duplicated substrings*: (contiguous) substrings of  $s$  that occur 2 or more times. The occurrences may overlap.

Return **any** duplicated substring that has the longest possible length. If  $s$  does not have a duplicated substring, the answer is `""`.

**Example 1:**

**Input:** s = "banana"

**Output:** "ana"

**Example 2:**

**Input:** s = "abcd"

**Output:** ""

**Constraints:**

- $2 \leq s.length \leq 3 * 10^4$
- $s$  consists of lowercase English letters.

**1046. Last Stone Weight****Easy**

376873Add to ListShare

You are given an array of integers `stones` where `stones[i]` is the weight of the  $i^{\text{th}}$  stone.

We are playing a game with the stones. On each turn, we choose the **heaviest two stones** and smash them together. Suppose the heaviest two stones have weights  $x$  and  $y$  with  $x \leq y$ . The result of this smash is:

- If  $x == y$ , both stones are destroyed, and
- If  $x != y$ , the stone of weight  $x$  is destroyed, and the stone of weight  $y$  has new weight  $y - x$ .

At the end of the game, there is **at most one** stone left.

Return *the weight of the last remaining stone*. If there are no stones left, return 0.

**Example 1:**

**Input:** stones = [2,7,4,1,8,1]

**Output:** 1

**Explanation:**

We combine 7 and 8 to get 1 so the array converts to [2,4,1,1,1] then,  
 we combine 2 and 4 to get 2 so the array converts to [2,1,1,1] then,  
 we combine 2 and 1 to get 1 so the array converts to [1,1,1] then,  
 we combine 1 and 1 to get 0 so the array converts to [1] then that's the value of the last stone.

**Example 2:**

**Input:** stones = [1]

**Output:** 1

**Constraints:**

- $1 \leq \text{stones.length} \leq 30$
- $1 \leq \text{stones}[i] \leq 1000$

**1047. Remove All Adjacent Duplicates In String****Easy**

3974173Add to ListShare

You are given a string  $s$  consisting of lowercase English letters. A **duplicate removal** consists of choosing two **adjacent** and **equal** letters and removing them.

We repeatedly make **duplicate removals** on  $s$  until we no longer can.

Return *the final string after all such duplicate removals have been made*. It can be proven that the answer is **unique**.

**Example 1:****Input:**  $s = \text{"abbaca"}$ **Output:**  $\text{"ca"}$ **Explanation:**

For example, in "abbaca" we could remove "bb" since the letters are adjacent and equal, and this is the only possible move. The result of this move is that the string is "aaca", of which only "aa" is possible, so the final string is "ca".

**Example 2:****Input:**  $s = \text{"azxxzy"}$ **Output:**  $\text{"ay"}$ **Constraints:**

- $1 \leq s.length \leq 10^5$
- $s$  consists of lowercase English letters.

**1048. Longest String Chain****Medium**

5304209Add to ListShare

You are given an array of  $\text{words}$  where each word consists of lowercase English letters.

`worda` is a **predecessor** of `wordb` if and only if we can insert **exactly one** letter anywhere in `worda` **without changing the order of the other characters** to make it equal to `wordb`.

- For example, "abc" is a **predecessor** of "abac", while "cba" is not a **predecessor** of "bcad".

A **word chain** is a sequence of words  $[word_1, word_2, \dots, word_k]$  with  $k \geq 1$ , where `word1` is a **predecessor** of `word2`, `word2` is a **predecessor** of `word3`, and so on. A single word is trivially a **word chain** with  $k == 1$ .

Return the **length** of the **longest possible word chain** with words chosen from the given list of words.

### Example 1:

**Input:** words = ["a", "b", "ba", "bca", "bda", "bdca"]

**Output:** 4

**Explanation:** One of the longest word chains is ["a", "ba", "bda", "bdca"].

### Example 2:

**Input:** words = ["xbc", "pcxbcf", "xb", "cxbc", "pcxbc"]

**Output:** 5

**Explanation:** All the words can be put in a word chain ["xb", "xbc", "cxbc", "pcbc", "pcbcf"].

### Example 3:

**Input:** words = ["abcd", "dbqca"]

**Output:** 1

**Explanation:** The trivial word chain ["abcd"] is one of the longest word chains.

["abcd", "dbqca"] is not a valid word chain because the ordering of the letters is changed.

### Constraints:

- $1 \leq \text{words.length} \leq 1000$
- $1 \leq \text{words}[i].length \leq 16$
- `words[i]` only consists of lowercase English letters.

## 1048. Longest String Chain

Medium

5304209Add to ListShare

You are given an array of `words` where each word consists of lowercase English letters.

`worda` is a **predecessor** of `wordb` if and only if we can insert **exactly one** letter anywhere in `worda` **without changing the order of the other characters** to make it equal to `wordb`.

- For example, "abc" is a **predecessor** of "abac", while "cba" is not a **predecessor** of "bcad".

A **word chain** is a sequence of words  $[word_1, word_2, \dots, word_k]$  with  $k \geq 1$ , where `word1` is a **predecessor** of `word2`, `word2` is a **predecessor** of `word3`, and so on. A single word is trivially a **word chain** with  $k == 1$ .

Return the **length** of the **longest possible word chain** with words chosen from the given list of `words`.

### Example 1:

**Input:** `words` = ["a", "b", "ba", "bca", "bda", "bdca"]

**Output:** 4

**Explanation:** One of the longest word chains is ["a", "ba", "bda", "bdca"].

### Example 2:

**Input:** `words` = ["xbc", "pcxbcf", "xb", "cxbc", "pcxbc"]

**Output:** 5

**Explanation:** All the words can be put in a word chain ["xb", "xbc", "cxbc", "pcbc", "pcbcf"].

### Example 3:

**Input:** `words` = ["abcd", "dbqca"]

**Output:** 1

**Explanation:** The trivial word chain ["abcd"] is one of the longest word chains.

["abcd", "dbqca"] is not a valid word chain because the ordering of the letters is changed.

**Constraints:**

- $1 \leq \text{words.length} \leq 1000$
- $1 \leq \text{words}[i].length \leq 16$
- `words[i]` only consists of lowercase English letters.

**1050. Actors and Directors Who Cooperated At Least Three Times****Easy**

30333Add to ListShare

SQL Schema

Table: `ActorDirector`

Column Name	Type
<code>actor_id</code>	int
<code>director_id</code>	int
<code>timestamp</code>	int

`timestamp` is the primary key column for this table.

Write a SQL query for a report that provides the pairs `(actor_id, director_id)` where the actor has cooperated with the director at least three times.

Return the result table in **any order**.

The query result format is in the following example.

**Example 1:****Input:**

ActorDirector table:

+	-----	-----	-----
---	-------	-------	-------

actor_id	director_id	timestamp
1	1	0
1	1	1
1	1	2
1	2	3
1	2	4
2	1	5
2	1	6

**Output:**

actor_id	director_id
1	1

**Explanation:** The only pair is (1, 1) where they cooperated exactly 3 times.

## 1051. Height Checker

**Easy**

63456Add to ListShare

A school is trying to take an annual photo of all the students. The students are asked to stand in a single file line in **non-decreasing order** by height. Let this ordering be represented by the integer array `expected` where `expected[i]` is the expected height of the `ith` student in line.

You are given an integer array `heights` representing the **current order** that the students are standing in. Each `heights[i]` is the height of the `ith` student in line (**0-indexed**).

Return *the number of indices* where `heights[i] != expected[i]`.

**Example 1:**

**Input:** `heights = [1,1,4,2,1,3]`

**Output:** 3

**Explanation:**

heights: [1,1,4,2,1,3]

expected: [1,1,1,2,3,4]

Indices 2, 4, and 5 do not match.

**Example 2:**

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

**Output:** 5

**Explanation:**

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

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

All indices do not match.

**Example 3:**

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

**Output:** 0

**Explanation:**

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

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

All indices match.

**Constraints:**

- $1 \leq \text{heights.length} \leq 100$
- $1 \leq \text{heights}[i] \leq 100$

## 1052. Grumpy Bookstore Owner

Medium

1310109Add to ListShare

There is a bookstore owner that has a store open for  $n$  minutes. Every minute, some number of customers enter the store. You are given an integer array `customers` of

length `n` where `customers[i]` is the number of the customer that enters the store at the start of the `ith` minute and all those customers leave after the end of that minute.

On some minutes, the bookstore owner is grumpy. You are given a binary array `grumpy` where `grumpy[i]` is `1` if the bookstore owner is grumpy during the `ith` minute, and is `0` otherwise.

When the bookstore owner is grumpy, the customers of that minute are not satisfied, otherwise, they are satisfied.

The bookstore owner knows a secret technique to keep themselves not grumpy for `minutes` consecutive minutes, but can only use it once.

Return *the maximum number of customers that can be satisfied throughout the day*.

### Example 1:

**Input:** `customers = [1,0,1,2,1,1,7,5]`, `grumpy = [0,1,0,1,0,1,0,1]`, `minutes = 3`

**Output:** `16`

**Explanation:** The bookstore owner keeps themselves not grumpy for the last 3 minutes.

The maximum number of customers that can be satisfied =  $1 + 1 + 1 + 1 + 7 + 5 = 16$ .

### Example 2:

**Input:** `customers = [1]`, `grumpy = [0]`, `minutes = 1`

**Output:** `1`

### Constraints:

- `n == customers.length == grumpy.length`
- `1 <= minutes <= n <= 2 * 104`
- `0 <= customers[i] <= 1000`
- `grumpy[i]` is either `0` or `1`.

## 1053. Previous Permutation With One Swap

Medium

31127Add to ListShare

Given an array of positive integers `arr` (not necessarily distinct), return *the lexicographically largest permutation that is smaller than arr*, that can be **made with exactly one swap** (A swap exchanges the positions of two numbers `arr[i]` and `arr[j]`). If it cannot be done, then return the same array.

**Example 1:****Input:** arr = [3,2,1]**Output:** [3,1,2]**Explanation:** Swapping 2 and 1.**Example 2:****Input:** arr = [1,1,5]**Output:** [1,1,5]**Explanation:** This is already the smallest permutation.**Example 3:****Input:** arr = [1,9,4,6,7]**Output:** [1,7,4,6,9]**Explanation:** Swapping 9 and 7.**Constraints:**

- $1 \leq \text{arr.length} \leq 10^4$
- $1 \leq \text{arr}[i] \leq 10^4$

**1054. Distant Barcodes****Medium**

95741Add to ListShare

In a warehouse, there is a row of barcodes, where the  $i^{\text{th}}$  barcode is `barcodes[i]`.

Rearrange the barcodes so that no two adjacent barcodes are equal. You may return any answer, and it is guaranteed an answer exists.

**Example 1:****Input:** barcodes = [1,1,1,2,2,2]**Output:** [2,1,2,1,2,1]**Example 2:**

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

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

**Constraints:**

- $1 \leq \text{barcodes.length} \leq 10000$
- $1 \leq \text{barcodes}[i] \leq 10000$

## 1071. Greatest Common Divisor of Strings

Easy

1434287Add to ListShare

For two strings  $s$  and  $t$ , we say " $t$  divides  $s$ " if and only if  $s = t + \dots + t$  (i.e.,  $t$  is concatenated with itself one or more times).

Given two strings  $\text{str1}$  and  $\text{str2}$ , return *the largest string  $x$  such that  $x$  divides both  $\text{str1}$  and  $\text{str2}$* .

**Example 1:**

**Input:** str1 = "ABCABC", str2 = "ABC"

**Output:** "ABC"

**Example 2:**

**Input:** str1 = "ABABAB", str2 = "ABAB"

**Output:** "AB"

**Example 3:**

**Input:** str1 = "LEET", str2 = "CODE"

**Output:** ""

**Constraints:**

- $1 \leq \text{str1.length}, \text{str2.length} \leq 1000$
- $\text{str1}$  and  $\text{str2}$  consist of English uppercase letters.

## 1072. Flip Columns For Maximum Number of Equal Rows

Medium

59543Add to ListShare

You are given an  $m \times n$  binary matrix `matrix`.

You can choose any number of columns in the matrix and flip every cell in that column (i.e., Change the value of the cell from `0` to `1` or vice versa).

Return *the maximum number of rows that have all values equal after some number of flips*.

### Example 1:

**Input:** `matrix = [[0,1],[1,1]]`

**Output:** `1`

**Explanation:** After flipping no values, 1 row has all values equal.

### Example 2:

**Input:** `matrix = [[0,1],[1,0]]`

**Output:** `2`

**Explanation:** After flipping values in the first column, both rows have equal values.

### Example 3:

**Input:** `matrix = [[0,0,0],[0,0,1],[1,1,0]]`

**Output:** `2`

**Explanation:** After flipping values in the first two columns, the last two rows have equal values.

### Constraints:

- $m == \text{matrix.length}$
- $n == \text{matrix}[i].length$
- $1 \leq m, n \leq 300$
- $\text{matrix}[i][j]$  is either `0` or `1`.

## 1073. Adding Two Negabinary Numbers

Medium

24794Add to ListShare

Given two numbers `arr1` and `arr2` in base **-2**, return the result of adding them together.

Each number is given in *array format*: as an array of 0s and 1s, from most significant bit to least significant bit. For example, `arr = [1,1,0,1]` represents the number  $(-2)^3 + (-2)^2 + (-2)^0 = -3$ . A number `arr` in *array format* is also guaranteed to have no leading zeros: either `arr == [0]` or `arr[0] == 1`.

Return the result of adding `arr1` and `arr2` in the same format: as an array of 0s and 1s with no leading zeros.

### Example 1:

**Input:** `arr1 = [1,1,1,1,1]`, `arr2 = [1,0,1]`

**Output:** `[1,0,0,0,0]`

**Explanation:** `arr1` represents 11, `arr2` represents 5, the output represents 16.

### Example 2:

**Input:** `arr1 = [0]`, `arr2 = [0]`

**Output:** `[0]`

### Example 3:

**Input:** `arr1 = [0]`, `arr2 = [1]`

**Output:** `[1]`

### Constraints:

- $1 \leq \text{arr1.length}, \text{arr2.length} \leq 1000$
- `arr1[i]` and `arr2[i]` are 0 or 1
- `arr1` and `arr2` have no leading zeros

## 1074. Number of Submatrices That Sum to Target

Hard

276261Add to ListShare

Given a `matrix` and a `target`, return the number of non-empty submatrices that sum to `target`.

A submatrix  $x_1, y_1, x_2, y_2$  is the set of all cells `matrix[x][y]` with  $x_1 \leq x \leq x_2$  and  $y_1 \leq y \leq y_2$ .

Two submatrices  $(x_1, y_1, x_2, y_2)$  and  $(x_1', y_1', x_2', y_2')$  are different if they have some coordinate that is different: for example, if  $x_1 \neq x_1'$ .

**Example 1:**

0	1	0
1	1	1
0	1	0

**Input:** matrix = [[0,1,0],[1,1,1],[0,1,0]], target = 0

**Output:** 4

**Explanation:** The four 1x1 submatrices that only contain 0.

**Example 2:**

**Input:** matrix = [[1,-1],[-1,1]], target = 0

**Output:** 5

**Explanation:** The two 1x2 submatrices, plus the two 2x1 submatrices, plus the 2x2 submatrix.

**Example 3:**

**Input:** matrix = [[904]], target = 0

**Output:** 0

**Constraints:**

- $1 \leq \text{matrix.length} \leq 100$
- $1 \leq \text{matrix[0].length} \leq 100$
- $-1000 \leq \text{matrix[i]} \leq 1000$
- $-10^8 \leq \text{target} \leq 10^8$

**1078. Occurrences After Bigram**

Easy

363308Add to ListShare

Given two strings `first` and `second`, consider occurrences in some text of the form "`first second third`", where `second` comes immediately after `first`, and `third` comes immediately after `second`.

Return *an array of all the words* `third` *for each occurrence of* "`first second third`".

### Example 1:

**Input:** `text = "alice is a good girl she is a good student", first = "a", second = "good"`

**Output:** `["girl", "student"]`

### Example 2:

**Input:** `text = "we will we will rock you", first = "we", second = "will"`

**Output:** `["we", "rock"]`

### Constraints:

- `1 <= text.length <= 1000`
- `text` consists of lowercase English letters and spaces.
- All the words in `text` are separated by **a single space**.
- `1 <= first.length, second.length <= 10`
- `first` and `second` consist of lowercase English letters.

## 1079. Letter Tile Possibilities

Medium

192153Add to ListShare

You have `n` tiles, where each tile has one letter `tiles[i]` printed on it.

Return *the number of possible non-empty sequences of letters* you can make using the letters printed on those tiles.

### Example 1:

**Input:** `tiles = "AAB"`

**Output:** `8`

**Explanation:** The possible sequences are "A", "B", "AA", "AB", "BA", "AAB", "ABA", "BAA".

**Example 2:**

**Input:** tiles = "AAABBC"

**Output:** 188

**Example 3:**

**Input:** tiles = "V"

**Output:** 1

**Constraints:**

- `1 <= tiles.length <= 7`
- `tiles` consists of uppercase English letters.

## 1080. Insufficient Nodes in Root to Leaf Paths

Medium

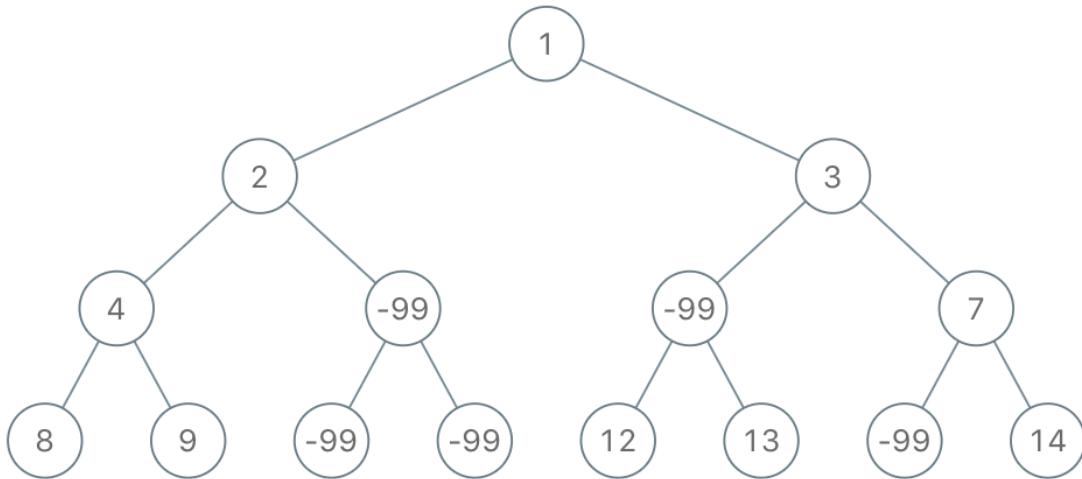
502612Add to ListShare

Given the `root` of a binary tree and an integer `limit`, delete all **insufficient nodes** in the tree simultaneously, and return *the root of the resulting binary tree*.

A node is **insufficient** if every root to **leaf** path intersecting this node has a sum strictly less than `limit`.

A **leaf** is a node with no children.

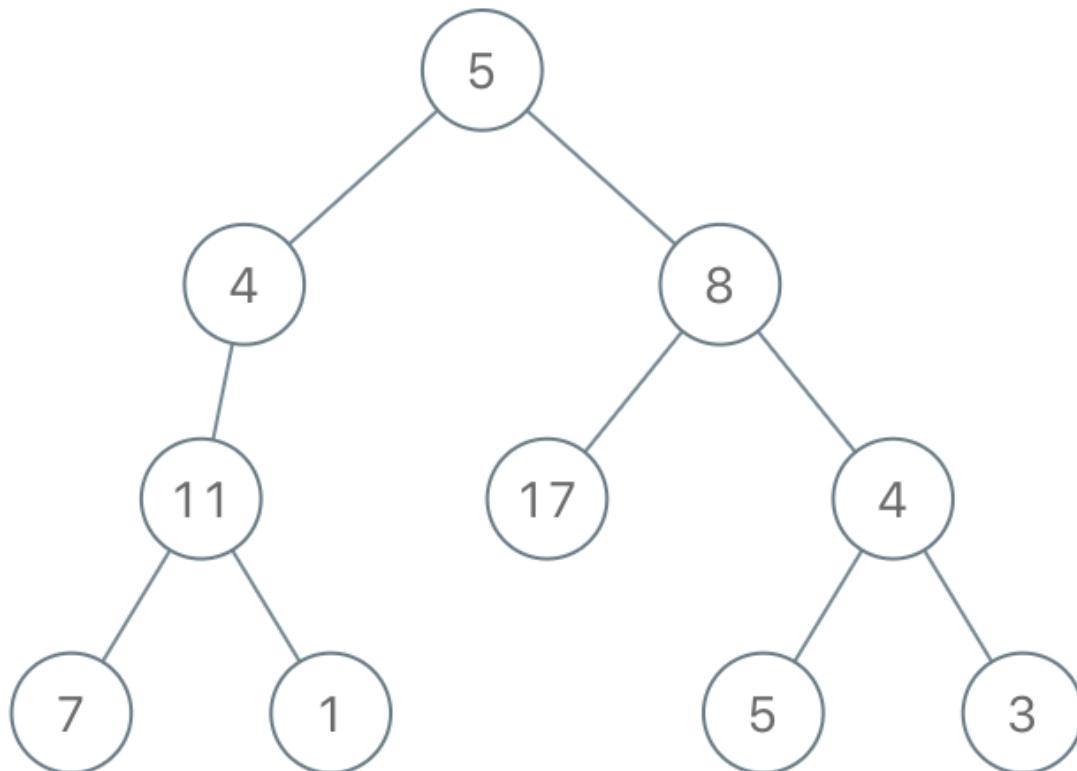
**Example 1:**



**Input:** root = [1,2,3,4,-99,-99,7,8,9,-99,-99,12,13,-99,14], limit = 1

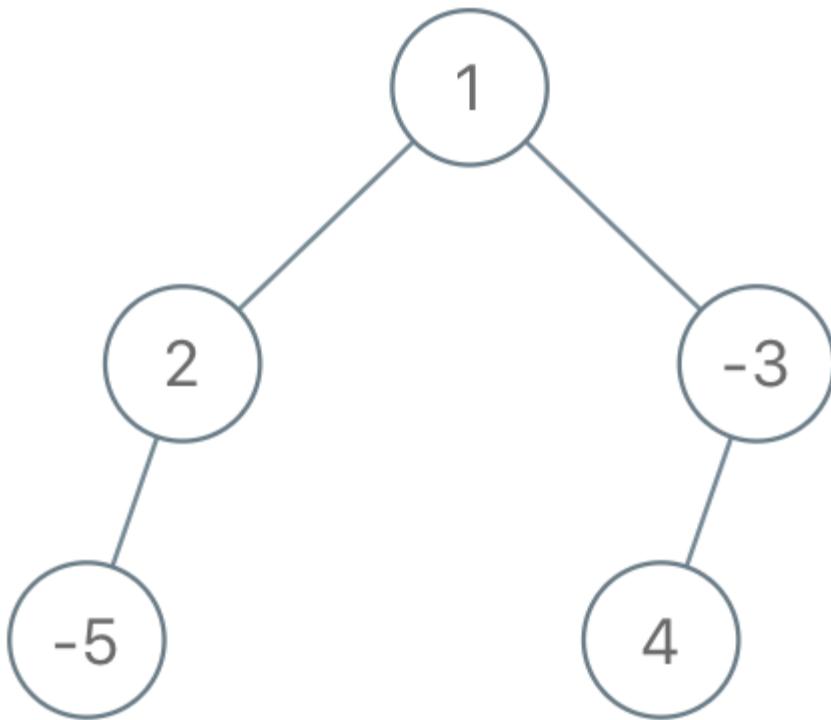
**Output:** [1,2,3,4,null,null,7,8,9,null,14]

**Example 2:**



**Input:** root = [5,4,8,11,null,17,4,7,1,null,null,5,3], limit = 22

**Output:** [5,4,8,11,null,17,4,7,null,null,null,5]

**Example 3:**

**Input:** root = [1,2,-3,-5,null,4,null], limit = -1

**Output:** [1,null,-3,4]

**Constraints:**

- The number of nodes in the tree is in the range [1, 5000].
- $-10^5 \leq \text{Node.val} \leq 10^5$
- $-10^9 \leq \text{limit} \leq 10^9$

**1081. Smallest Subsequence of Distinct Characters**

Medium

1912155Add to ListShare

Given a string  $s$ , return the lexicographically smallest subsequence of  $s$  that contains all the distinct characters of  $s$  exactly once.

**Example 1:**

**Input:** s = "bcabc"

**Output:** "abc"

**Example 2:**

**Input:** s = "cbacdcdbc"

**Output:** "acdb"

**Constraints:**

- $1 \leq s.length \leq 1000$
- s consists of lowercase English letters.

## 1084. Sales Analysis III

Easy

36882Add to ListShare

SQL Schema

Table: Product

Column Name	Type
product_id	int
product_name	varchar
unit_price	int

product\_id is the primary key of this table.

Each row of this table indicates the name and the price of each product.

Table: Sales

Column Name	Type

seller_id	int	
product_id	int	
buyer_id	int	
sale_date	date	
quantity	int	
price	int	
+-----+-----+		

This table has no primary key, it can have repeated rows.

product\_id is a foreign key to the Product table.

Each row of this table contains some information about one sale.

Write an SQL query that reports the **products** that were **only** sold in the first quarter of **2019**. That is, between **2019-01-01** and **2019-03-31** inclusive.

Return the result table in **any order**.

The query result format is in the following example.

### Example 1:

**Input:**

Product table:

+-----+-----+-----+		
product_id	product_name	unit_price
+-----+-----+-----+		
1	S8	1000
2	G4	800
3	iPhone	1400
+-----+-----+-----+		

Sales table:

seller_id	product_id	buyer_id	sale_date	quantity	price
1	1	1	2019-01-21	2	2000
1	2	2	2019-02-17	1	800
2	2	3	2019-06-02	1	800
3	3	4	2019-05-13	2	2800

**Output:**

product_id	product_name
1	S8

**Explanation:**

The product with id 1 was only sold in the spring of 2019.

The product with id 2 was sold in the spring of 2019 but was also sold after the spring of 2019.

The product with id 3 was sold after spring 2019.

We return only product 1 as it is the product that was only sold in the spring of 2019.

## 1090. Largest Values From Labels

Medium

330579Add to ListShare

There is a set of `n` items. You are given two integer arrays `values` and `labels` where the value and the label of the  $i^{\text{th}}$  element are `values[i]` and `labels[i]` respectively. You are also given two integers `numWanted` and `useLimit`.

Choose a subset `s` of the `n` elements such that:

- The size of the subset `s` is **less than or equal to** `numWanted`.
- There are **at most** `useLimit` items with the same label in `s`.

The **score** of a subset is the sum of the values in the subset.

Return *the maximum score of a subset s*.

**Example 1:**

**Input:** values = [5,4,3,2,1], labels = [1,1,2,2,3], numWanted = 3, useLimit = 1

**Output:** 9

**Explanation:** The subset chosen is the first, third, and fifth items.

**Example 2:**

**Input:** values = [5,4,3,2,1], labels = [1,3,3,3,2], numWanted = 3, useLimit = 2

**Output:** 12

**Explanation:** The subset chosen is the first, second, and third items.

**Example 3:**

**Input:** values = [9,8,8,7,6], labels = [0,0,0,1,1], numWanted = 3, useLimit = 1

**Output:** 16

**Explanation:** The subset chosen is the first and fourth items.

**Constraints:**

- $n == \text{values.length} == \text{labels.length}$
- $1 \leq n \leq 2 * 10^4$
- $0 \leq \text{values}[i], \text{labels}[i] \leq 2 * 10^4$
- $1 \leq \text{numWanted}, \text{useLimit} \leq n$

## 1091. Shortest Path in Binary Matrix

Medium

3961167 Add to List Share

Given an  $n \times n$  binary matrix `grid`, return *the length of the shortest clear path in the matrix*. If there is no clear path, return `-1`.

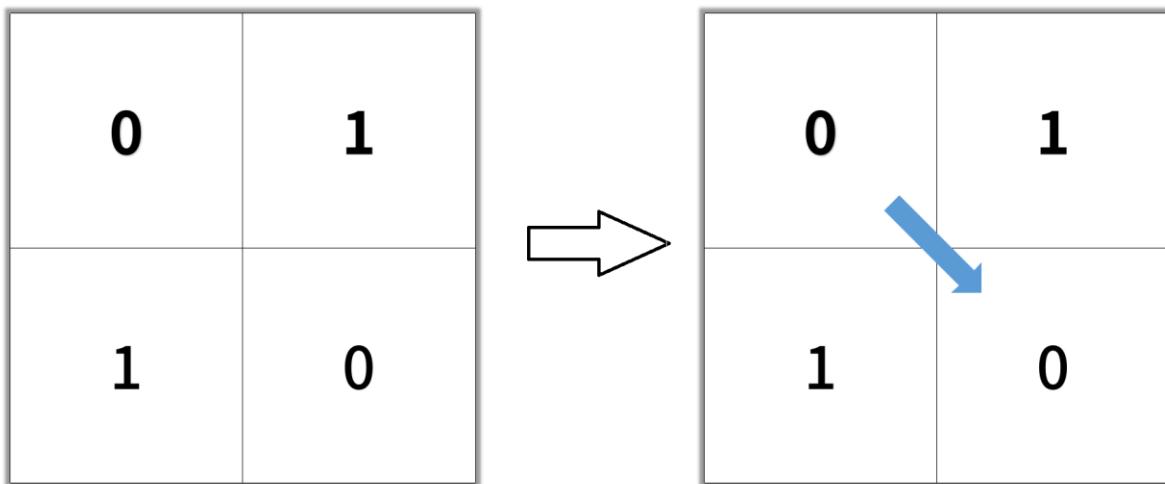
A **clear path** in a binary matrix is a path from the **top-left** cell (i.e.,  $(0, 0)$ ) to the **bottom-right** cell (i.e.,  $(n - 1, n - 1)$ ) such that:

- All the visited cells of the path are `0`.

- All the adjacent cells of the path are **8-directionally** connected (i.e., they are different and they share an edge or a corner).

The **length of a clear path** is the number of visited cells of this path.

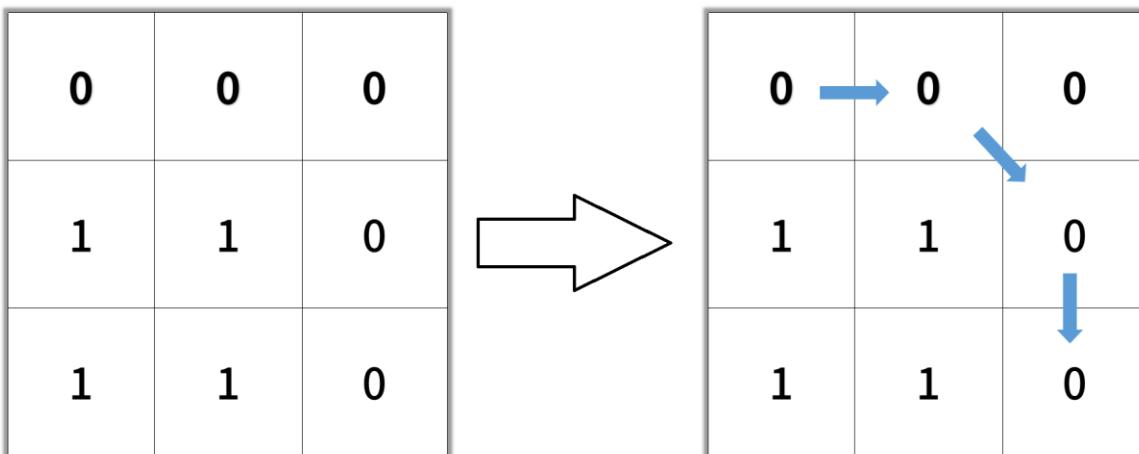
**Example 1:**



**Input:** grid = [[0,1],[1,0]]

**Output:** 2

**Example 2:**



**Input:** grid = [[0,0,0],[1,1,0],[1,1,0]]

**Output:** 4

**Example 3:**

**Input:** grid = [[1,0,0],[1,1,0],[1,1,0]]

**Output:** -1

**Constraints:**

- `n == grid.length`
- `n == grid[i].length`
- `1 <= n <= 100`
- `grid[i][j]` is 0 or 1

## 1092. Shortest Common Supersequence

Hard

308249Add to ListShare

Given two strings `str1` and `str2`, return *the shortest string that has both str1 and str2 as subsequences*. If there are multiple valid strings, return **any** of them.

A string `s` is a **subsequence** of string `t` if deleting some number of characters from `t` (possibly 0) results in the string `s`.

**Example 1:**

**Input:** str1 = "abac", str2 = "cab"

**Output:** "cabac"

**Explanation:**

`str1 = "abac"` is a subsequence of `"cabac"` because we can delete the first "c".

`str2 = "cab"` is a subsequence of `"cabac"` because we can delete the last "ac".

The answer provided is the shortest such string that satisfies these properties.

**Example 2:**

**Input:** str1 = "aaaaaaaa", str2 = "aaaaaaaa"

**Output:** "aaaaaaaa"

## Constraints:

- $1 \leq \text{str1.length}, \text{str2.length} \leq 1000$
  - `str1` and `str2` consist of lowercase English letters.

## 1093. Statistics from a Large Sample

Medium

## 6468Add to ListShare

You are given a large sample of integers in the range  $[0, 255]$ . Since the sample is so large, it is represented by an array `count` where `count[k]` is the **number of times** that `k` appears in the sample.

Calculate the following statistics:

- **minimum**: The minimum element in the sample.
  - **maximum**: The maximum element in the sample.
  - **mean**: The average of the sample, calculated as the total sum of all elements divided by the total number of elements.
  - **median**:
    - If the sample has an odd number of elements, then the **median** is the middle element once the sample is sorted.
    - If the sample has an even number of elements, then the **median** is the average of the two middle elements once the sample is sorted.
  - **mode**: The number that appears the most in the sample. It is guaranteed to be **unique**.

Return the statistics of the sample as an array of floating-point numbers [minimum, maximum, mean, median, mode]. Answers within  $10^{-5}$  of the actual answer will be accepted.

### Example 1:

**Output:** [1.00000,3.00000,2.37500,2.50000,3.00000]

**Explanation:** The sample represented by count is [1,2,2,2,3,3,3,3,3].

The minimum and maximum are 1 and 3 respectively.

The mean is  $(1+2+2+3+3+3+3) / 8 = 19 / 8 = 2.375$ .

Since the size of the sample is even, the median is the average of the two middle elements 2 and 3, which is 2.5.

The mode is 3 as it appears the most in the sample.

## Example 2:

Output: [1.00000,4.00000,2.18182,2.00000,1.00000]

**Explanation:** The sample represented by count is [1,1,1,1,2,2,2,3,3,4,4].

The minimum and maximum are 1 and 4 respectively.

The mean is  $(1+1+1+1+2+2+2+3+3+4+4) / 11 = 24 / 11 = 2.18181818\dots$  (for display purposes, the output shows the rounded number 2.18182).

Since the size of the sample is odd, the median is the middle element 2.

The mode is 1 as it appears the most in the sample.

## Constraints:

- `count.length == 256`
  - $0 \leq \text{count}[i] \leq 10^9$
  - $1 \leq \text{sum}(\text{count}) \leq 10^9$
  - The mode of the sample that `count` represents is **unique**.

## 1094. Car Pooling

## Medium

358577Add to listShare

There is a car with `capacity` empty seats. The vehicle only drives east (i.e., it cannot turn around and drive west).

You are given the integer `capacity` and an array `trips` where `trips[i] = [numPassengersi, fromi, toi]` indicates that the  $i^{\text{th}}$  trip has `numPassengersi` passengers and the locations to pick them up and drop them off are `fromi` and `toi` respectively. The locations are given as the number of kilometers due east from the car's initial location.

Return `true` if it is possible to pick up and drop off all passengers for all the given trips, or `false` otherwise.

**Example 1:**

**Input:** `trips = [[2,1,5],[3,3,7]]`, `capacity = 4`

**Output:** `false`

**Example 2:**

**Input:** `trips = [[2,1,5],[3,3,7]]`, `capacity = 5`

**Output:** `true`

**Constraints:**

- $1 \leq \text{trips.length} \leq 1000$
- $\text{trips}[i].length == 3$
- $1 \leq \text{numPassengers}_i \leq 100$
- $0 \leq \text{from}_i < \text{to}_i \leq 1000$
- $1 \leq \text{capacity} \leq 10^5$

## 1095. Find in Mountain Array

Hard

153064Add to ListShare

(This problem is an **interactive problem**.)

You may recall that an array `arr` is a **mountain array** if and only if:

- `arr.length >= 3`
- There exists some `i` with  $0 < i < \text{arr.length} - 1$  such that:
  - $\text{arr}[0] < \text{arr}[1] < \dots < \text{arr}[i - 1] < \text{arr}[i]$
  - $\text{arr}[i] > \text{arr}[i + 1] > \dots > \text{arr}[\text{arr.length} - 1]$

Given a mountain array `mountainArr`, return the **minimum** `index` such that `mountainArr.get(index) == target`. If such an `index` does not exist, return `-1`.

**You cannot access the mountain array directly.** You may only access the array using a `MountainArray` interface:

- `MountainArray.get(k)` returns the element of the array at index `k` (0-indexed).
- `MountainArray.length()` returns the length of the array.

Submissions making more than 100 calls to `MountainArray.get` will be judged *Wrong Answer*. Also, any solutions that attempt to circumvent the judge will result in disqualification.

**Example 1:**

**Input:** `array = [1,2,3,4,5,3,1]`, `target = 3`

**Output:** 2

**Explanation:** 3 exists in the array, at `index=2` and `index=5`. Return the minimum index, which is 2.

**Example 2:**

**Input:** `array = [0,1,2,4,2,1]`, `target = 3`

**Output:** -1

**Explanation:** 3 does not exist in the array, so we return -1.

**Constraints:**

- `3 <= mountain_arr.length() <= 104`
- `0 <= target <= 109`
- `0 <= mountain_arr.get(index) <= 109`

## 1096. Brace Expansion II

**Hard**

407255Add to ListShare

Under the grammar given below, strings can represent a set of lowercase words.

Let `R(expr)` denote the set of words the expression represents.

The grammar can best be understood through simple examples:

- Single letters represent a singleton set containing that word.
  - `R("a") = {"a"}`
  - `R("w") = {"w"}`
- When we take a comma-delimited list of two or more expressions, we take the union of possibilities.
  - `R("{a,b,c}") = {"a", "b", "c"}`
  - `R("{{a,b},{b,c}}") = {"a", "b", "c"}` (notice the final set only contains each word at most once)
- When we concatenate two expressions, we take the set of possible concatenations between two words where the first word comes from the first expression and the second word comes from the second expression.

- o  $R(\{a,b\}\{c,d\}) = \{ac, ad, bc, bd\}$
- o  $R(\{a,b,c\}\{d,e\}f\{g,h\}) = \{abdfg, abdfh, abefg, abefh, acdfg, acdfh, acefg, acefh\}$

Formally, the three rules for our grammar:

- For every lowercase letter  $x$ , we have  $R(x) = \{x\}$ .
- For expressions  $e_1, e_2, \dots, e_k$  with  $k \geq 2$ , we have  $R(\{e_1, e_2, \dots\}) = R(e_1) \cup R(e_2) \cup \dots$
- For expressions  $e_1$  and  $e_2$ , we have  $R(e_1 + e_2) = \{a + b \text{ for } (a, b) \text{ in } R(e_1) \times R(e_2)\}$ , where  $+$  denotes concatenation, and  $\times$  denotes the cartesian product.

Given an expression representing a set of words under the given grammar, return *the sorted list of words that the expression represents*.

### Example 1:

**Input:** expression = "{a,b}{c,{d,e}}"

**Output:** ["ac", "ad", "ae", "bc", "bd", "be"]

### Example 2:

**Input:** expression = "{{a,z},a{b,c},{ab,z}}"

**Output:** ["a", "ab", "ac", "z"]

**Explanation:** Each distinct word is written only once in the final answer.

### Constraints:

- $1 \leq \text{expression.length} \leq 60$
- $\text{expression}[i]$  consists of '`,`', '`}`', '`,`', '`'` or lowercase English letters.
- The given `expression` represents a set of words based on the grammar given in the description.

## 1103. Distribute Candies to People

Easy

759175Add to ListShare

We distribute some number of `candies`, to a row of `n = num_people` people in the following way:

We then give 1 candy to the first person, 2 candies to the second person, and so on until we give `n` candies to the last person.

Then, we go back to the start of the row, giving  $n + 1$  candies to the first person,  $n + 2$  candies to the second person, and so on until we give  $2 * n$  candies to the last person.

This process repeats (with us giving one more candy each time, and moving to the start of the row after we reach the end) until we run out of candies. The last person will receive all of our remaining candies (not necessarily one more than the previous gift).

Return an array (of length `num_people` and sum `candies`) that represents the final distribution of candies.

### Example 1:

**Input:** `candies = 7, num_people = 4`

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

**Explanation:**

On the first turn, `ans[0] += 1`, and the array is `[1,0,0,0]`.

On the second turn, `ans[1] += 2`, and the array is `[1,2,0,0]`.

On the third turn, `ans[2] += 3`, and the array is `[1,2,3,0]`.

On the fourth turn, `ans[3] += 1` (because there is only one candy left), and the final array is `[1,2,3,1]`.

### Example 2:

**Input:** `candies = 10, num_people = 3`

**Output:** `[5,2,3]`

**Explanation:**

On the first turn, `ans[0] += 1`, and the array is `[1,0,0]`.

On the second turn, `ans[1] += 2`, and the array is `[1,2,0]`.

On the third turn, `ans[2] += 3`, and the array is `[1,2,3]`.

On the fourth turn, `ans[0] += 4`, and the final array is `[5,2,3]`.

### Constraints:

- $1 \leq \text{candies} \leq 10^9$
- $1 \leq \text{num\_people} \leq 1000$

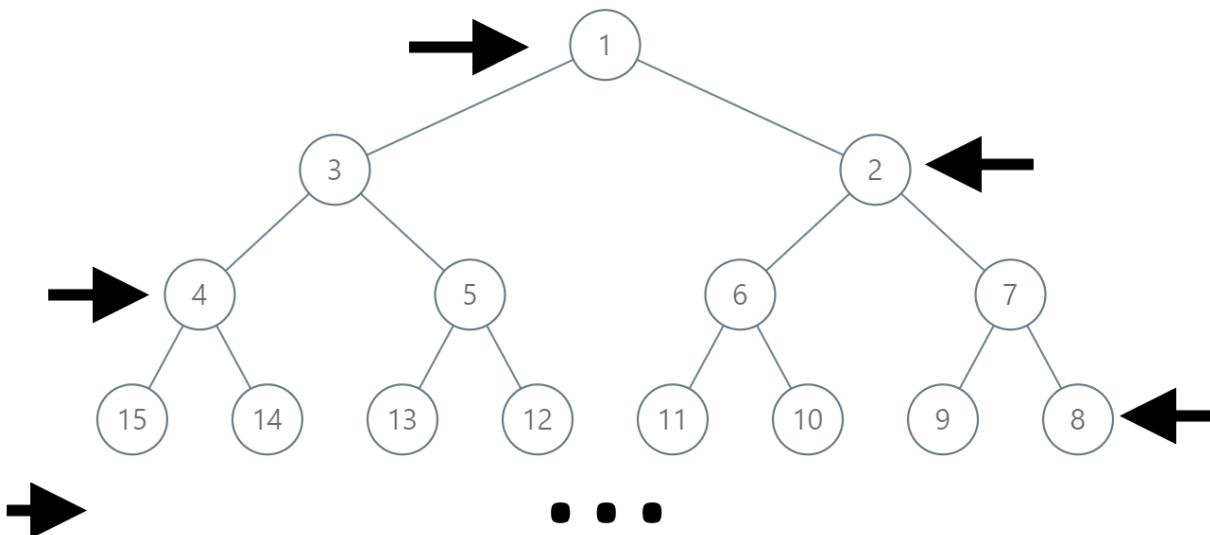
## 1104. Path In Zigzag Labelled Binary Tree

Medium

1148286Add to ListShare

In an infinite binary tree where every node has two children, the nodes are labelled in row order.

In the odd numbered rows (ie., the first, third, fifth,...), the labelling is left to right, while in the even numbered rows (second, fourth, sixth,...), the labelling is right to left.



Given the `label` of a node in this tree, return the labels in the path from the root of the tree to the node with that `label`.

**Example 1:**

**Input:** label = 14

**Output:** [1,3,4,14]

**Example 2:**

**Input:** label = 26

**Output:** [1,2,6,10,26]

**Constraints:**

- $1 \leq \text{label} \leq 10^6$

## 1105. Filling Bookcase Shelves

Medium

141688Add to ListShare

You are given an array `books` where `books[i] = [thicknessi, heighti]` indicates the thickness and height of the  $i^{\text{th}}$  book. You are also given an integer `shelfWidth`.

We want to place these books in order onto bookcase shelves that have a total width `shelfWidth`.

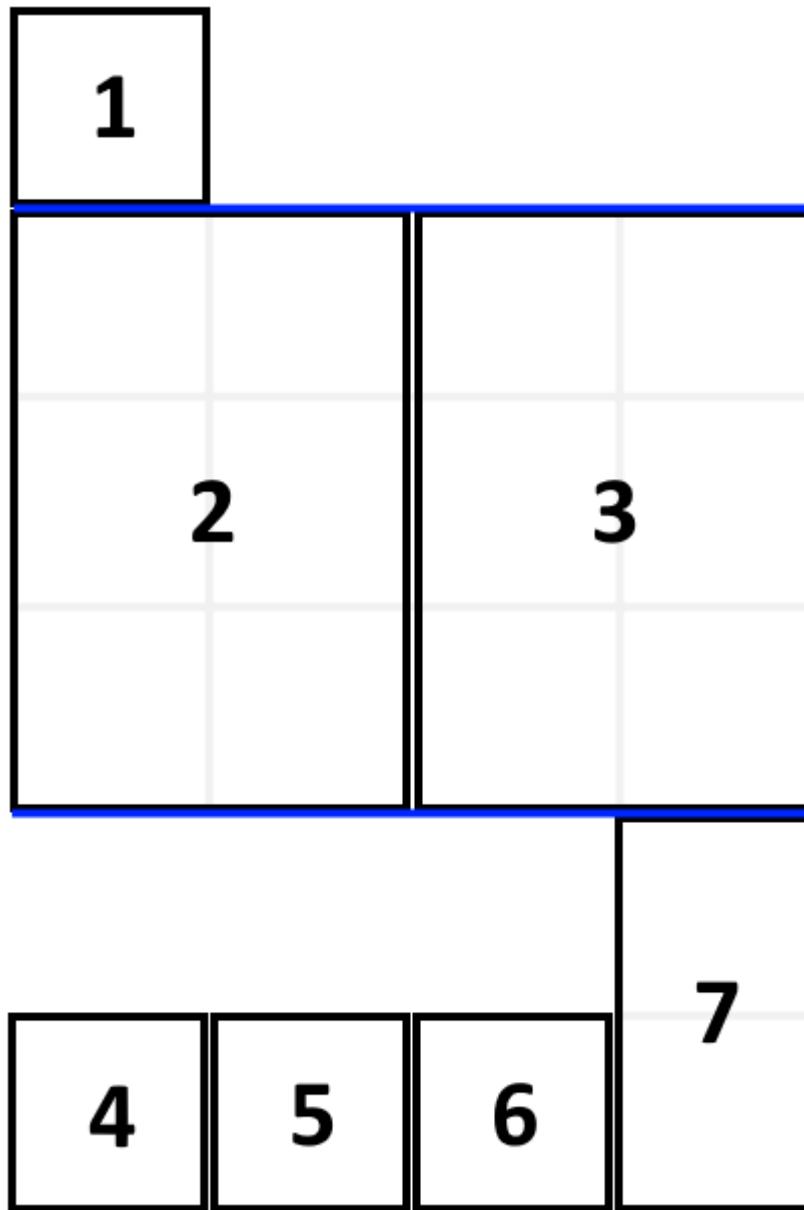
We choose some of the books to place on this shelf such that the sum of their thickness is less than or equal to `shelfWidth`, then build another level of the shelf of the bookcase so that the total height of the bookcase has increased by the maximum height of the books we just put down. We repeat this process until there are no more books to place.

Note that at each step of the above process, the order of the books we place is the same order as the given sequence of books.

- For example, if we have an ordered list of 5 books, we might place the first and second book onto the first shelf, the third book on the second shelf, and the fourth and fifth book on the last shelf.

Return *the minimum possible height that the total bookshelf can be after placing shelves in this manner*.

**Example 1:**



**Input:** books = [[1,1],[2,3],[2,3],[1,1],[1,1],[1,1],[1,2]], shelf\_width = 4

**Output:** 6

**Explanation:**

The sum of the heights of the 3 shelves is  $1 + 3 + 2 = 6$ .

Notice that book number 2 does not have to be on the first shelf.

**Example 2:**

**Input:** books = [[1,3],[2,4],[3,2]], shelfWidth = 6

**Output:** 4

### Constraints:

- $1 \leq \text{books.length} \leq 1000$
- $1 \leq \text{thickness}_i \leq \text{shelfWidth} \leq 1000$
- $1 \leq \text{height}_i \leq 1000$

## 1106. Parsing A Boolean Expression

Hard

76742Add to ListShare

A **boolean expression** is an expression that evaluates to either `true` or `false`. It can be in one of the following shapes:

- '`t`' that evaluates to `true`.
- '`f`' that evaluates to `false`.
- '`! (subExpr)`' that evaluates to **the logical NOT** of the inner expression `subExpr`.
- '`& (subExpr1, subExpr2, ..., subExprn)`' that evaluates to **the logical AND** of the inner expressions `subExpr1, subExpr2, ..., subExprn` where  $n \geq 1$ .
- '`| (subExpr1, subExpr2, ..., subExprn)`' that evaluates to **the logical OR** of the inner expressions `subExpr1, subExpr2, ..., subExprn` where  $n \geq 1$ .

Given a string `expression` that represents a **boolean expression**, return *the evaluation of that expression*.

It is **guaranteed** that the given expression is valid and follows the given rules.

### Example 1:

**Input:** `expression = "&(|(f))"`

**Output:** `false`

#### Explanation:

First, evaluate `|(f) --> f`. The expression is now `"&(f)"`.

Then, evaluate `&(f) --> f`. The expression is now `"f"`.

Finally, return `false`.

### Example 2:

**Input:** `expression = "|(f,f,f,t)"`

**Output:** `true`

**Explanation:** The evaluation of (false OR false OR false OR true) is true.

**Example 3:**

**Input:** expression = "!(&(f,t))"

**Output:** true

**Explanation:**

First, evaluate  $\&(f,t)$   $\rightarrow$  (false AND true)  $\rightarrow$  false  $\rightarrow$  f. The expression is now  $!(f)$ .

Then, evaluate  $!(f)$   $\rightarrow$  NOT false  $\rightarrow$  true. We return true.

**Constraints:**

- $1 \leq \text{expression.length} \leq 2 * 10^4$
- $\text{expression}[i]$  is one following characters: '(', ')', '&', '|', '!', 't', 'f', and ','.

## 1108. Defanging an IP Address

Easy

13241567Add to ListShare

Given a valid (IPv4) IP address, return a defanged version of that IP address.

A *defanged IP address* replaces every period ". " with "[ . ]".

**Example 1:**

**Input:** address = "1.1.1.1"

**Output:** "1[.]1[.]1[.]1"

**Example 2:**

**Input:** address = "255.100.50.0"

**Output:** "255[.]100[.]50[.]0"

**Constraints:**

- The given address is a valid IPv4 address.

## 1109. Corporate Flight Bookings

Medium

1200145Add to ListShare

There are  $n$  flights that are labeled from 1 to  $n$ .

You are given an array of flight bookings `bookings`, where `bookings[i] = [firsti, lasti, seatsi]` represents a booking for flights `firsti` through `lasti` (inclusive) with `seatsi` seats reserved for **each flight** in the range.

Return an array `answer` of length  $n$ , where `answer[i]` is the total number of seats reserved for flight  $i$ .

### Example 1:

**Input:** `bookings = [[1,2,10],[2,3,20],[2,5,25]]`,  $n = 5$

**Output:** `[10,55,45,25,25]`

**Explanation:**

Flight labels: 1 2 3 4 5

Booking 1 reserved: 10 10

Booking 2 reserved: 20 20

Booking 3 reserved: 25 25 25 25

Total seats: 10 55 45 25 25

Hence, `answer = [10,55,45,25,25]`

### Example 2:

**Input:** `bookings = [[1,2,10],[2,2,15]]`,  $n = 2$

**Output:** `[10,25]`

**Explanation:**

Flight labels: 1 2

Booking 1 reserved: 10 10

Booking 2 reserved: 15

Total seats: 10 25

Hence, `answer = [10,25]`

**Constraints:**

- $1 \leq n \leq 2 * 10^4$
- $1 \leq \text{bookings.length} \leq 2 * 10^4$
- $\text{bookings}[i].length == 3$
- $1 \leq \text{first}_i \leq \text{last}_i \leq n$
- $1 \leq \text{seats}_i \leq 10^4$

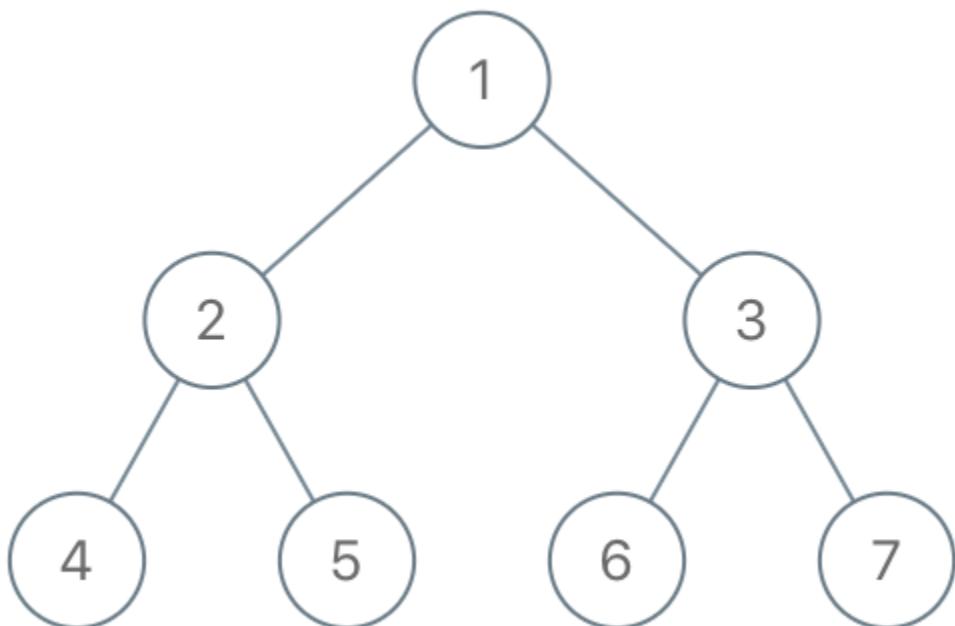
**1110. Delete Nodes And Return Forest****Medium**

312596 Add to List Share

Given the `root` of a binary tree, each node in the tree has a distinct value.

After deleting all nodes with a value in `to_delete`, we are left with a forest (a disjoint union of trees).

Return the roots of the trees in the remaining forest. You may return the result in any order.

**Example 1:**

**Input:** `root = [1,2,3,4,5,6,7]`, `to_delete = [3,5]`

**Output:** `[[1,2,null,4],[6],[7]]`

**Example 2:**

**Input:** `root = [1,2,4,null,3], to_delete = [3]`

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

**Constraints:**

- The number of nodes in the given tree is at most 1000.
- Each node has a distinct value between 1 and 1000.
- `to_delete.length <= 1000`
- `to_delete` contains distinct values between 1 and 1000.

## 1111. Maximum Nesting Depth of Two Valid Parentheses Strings

**Medium**

3331396Add to ListShare

A string is a *valid parentheses string* (denoted VPS) if and only if it consists of `"("` and `")"` characters only, and:

- It is the empty string, or
- It can be written as `AB` (`A` concatenated with `B`), where `A` and `B` are VPS's, or
- It can be written as `(A)`, where `A` is a VPS.

We can similarly define the *nesting depth* `depth(S)` of any VPS `S` as follows:

- `depth("") = 0`
- `depth(A + B) = max(depth(A), depth(B))`, where `A` and `B` are VPS's
- `depth("(" + A + ")") = 1 + depth(A)`, where `A` is a VPS.

For example, `""`, `"()()`, and `"()((())())"` are VPS's (with nesting depths 0, 1, and 2), and `"()("` and `"(())"` are not VPS's.

Given a VPS `seq`, split it into two disjoint subsequences `A` and `B`, such that `A` and `B` are VPS's (and `A.length + B.length = seq.length`).

Now choose **any** such `A` and `B` such that `max(depth(A), depth(B))` is the minimum possible value.

Return an `answer` array (of length `seq.length`) that encodes such a choice of `A` and `B`: `answer[i] = 0` if `seq[i]` is part of `A`, else `answer[i] = 1`. Note that even though multiple answers may exist, you may return any of them.

**Example 1:****Input:** seq = "((())")**Output:** [0,1,1,1,1,0]**Example 2:****Input:** seq = "()(())()"**Output:** [0,0,0,1,1,0,1,1]**Constraints:**

- `1 <= seq.size <= 10000`

**1114. Print in Order****Easy**

1094180Add to ListShare

Suppose we have a class:

```
public class Foo {
    public void first() { print("first"); }
    public void second() { print("second"); }
    public void third() { print("third"); }
}
```

The same instance of `Foo` will be passed to three different threads. Thread A will call `first()`, thread B will call `second()`, and thread C will call `third()`. Design a mechanism and modify the program to ensure that `second()` is executed after `first()`, and `third()` is executed after `second()`.

**Note:**

We do not know how the threads will be scheduled in the operating system, even though the numbers in the input seem to imply the ordering. The input format you see is mainly to ensure our tests' comprehensiveness.

**Example 1:**

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

**Output:** "firstsecondthird"

**Explanation:** There are three threads being fired asynchronously. The input [1,2,3] means thread A calls first(), thread B calls second(), and thread C calls third(). "firstsecondthird" is the correct output.

**Example 2:**

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

**Output:** "firstsecondthird"

**Explanation:** The input [1,3,2] means thread A calls first(), thread B calls third(), and thread C calls second(). "firstsecondthird" is the correct output.

**Constraints:**

- `nums` is a permutation of [1, 2, 3].

## 1115. Print FooBar Alternately

Medium

51841Add to ListShare

Suppose you are given the following code:

```
class FooBar {
    public void foo() {
        for (int i = 0; i < n; i++) {
            print("foo");
        }
    }

    public void bar() {
        for (int i = 0; i < n; i++) {
            print("bar");
        }
    }
}
```

```
}
```

The same instance of `FooBar` will be passed to two different threads:

- thread **A** will call `foo()`, while
- thread **B** will call `bar()`.

Modify the given program to output "foobar" `n` times.

### Example 1:

**Input:** `n = 1`

**Output:** "foobar"

**Explanation:** There are two threads being fired asynchronously. One of them calls `foo()`, while the other calls `bar()`.

"foobar" is being output 1 time.

### Example 2:

**Input:** `n = 2`

**Output:** "foobarfoobar"

**Explanation:** "foobar" is being output 2 times.

### Constraints:

- `1 <= n <= 1000`

## 1116. Print Zero Even Odd

Medium

370257Add to ListShare

You have a function `printNumber` that can be called with an integer parameter and prints it to the console.

- For example, calling `printNumber(7)` prints `7` to the console.

You are given an instance of the class `ZeroEvenOdd` that has three functions: `zero`, `even`, and `odd`. The same instance of `ZeroEvenOdd` will be passed to three different threads:

- **Thread A:** calls `zero()` that should only output 0's.

- **Thread B:** calls `even()` that should only output even numbers.
- **Thread C:** calls `odd()` that should only output odd numbers.

Modify the given class to output the series "010203040506..." where the length of the series must be `2n`.

Implement the `ZeroEvenOdd` class:

- `ZeroEvenOdd(int n)` Initializes the object with the number `n` that represents the numbers that should be printed.
- `void zero(printNumber)` Calls `printNumber` to output one zero.
- `void even(printNumber)` Calls `printNumber` to output one even number.
- `void odd(printNumber)` Calls `printNumber` to output one odd number.

### Example 1:

**Input:** `n = 2`

**Output:** "0102"

**Explanation:** There are three threads being fired asynchronously.

One of them calls `zero()`, the other calls `even()`, and the last one calls `odd()`.

"0102" is the correct output.

### Example 2:

**Input:** `n = 5`

**Output:** "0102030405"

### Constraints:

- `1 <= n <= 1000`

## 1117. Building H2O

Medium

37898Add to ListShare

There are two kinds of threads: `oxygen` and `hydrogen`. Your goal is to group these threads to form water molecules.

There is a barrier where each thread has to wait until a complete molecule can be formed. Hydrogen and oxygen threads will be given `releaseHydrogen` and `releaseOxygen` methods respectively,

which will allow them to pass the barrier. These threads should pass the barrier in groups of three, and they must immediately bond with each other to form a water molecule. You must guarantee that all the threads from one molecule bond before any other threads from the next molecule do.

In other words:

- If an oxygen thread arrives at the barrier when no hydrogen threads are present, it must wait for two hydrogen threads.
- If a hydrogen thread arrives at the barrier when no other threads are present, it must wait for an oxygen thread and another hydrogen thread.

We do not have to worry about matching the threads up explicitly; the threads do not necessarily know which other threads they are paired up with. The key is that threads pass the barriers in complete sets; thus, if we examine the sequence of threads that bind and divide them into groups of three, each group should contain one oxygen and two hydrogen threads.

Write synchronization code for oxygen and hydrogen molecules that enforces these constraints.

### Example 1:

**Input:** water = "HOH"

**Output:** "HHO"

**Explanation:** "HOH" and "OHH" are also valid answers.

### Example 2:

**Input:** water = "OOHHHH"

**Output:** "HHOHHO"

**Explanation:** "HOHHHO", "OHHHHO", "HHOHOH", "HOHHOH", "OHHHOH", "HHOOHH", "HOHOHH" and "OHHOHH" are also valid answers.

### Constraints:

- $3 * n == \text{water.length}$
- $1 \leq n \leq 20$
- `water[i]` is either 'H' or 'O'.
- There will be exactly  $2 * n$  'H' in `water`.
- There will be exactly  $n$  'O' in `water`.

## 1122. Relative Sort Array

**Easy**

1895110Add to ListShare

Given two arrays `arr1` and `arr2`, the elements of `arr2` are distinct, and all elements in `arr2` are also in `arr1`.

Sort the elements of `arr1` such that the relative ordering of items in `arr1` are the same as in `arr2`. Elements that do not appear in `arr2` should be placed at the end of `arr1` in **ascending** order.

**Example 1:**

**Input:** `arr1 = [2,3,1,3,2,4,6,7,9,2,19]`, `arr2 = [2,1,4,3,9,6]`

**Output:** `[2,2,2,1,4,3,3,9,6,7,19]`

**Example 2:**

**Input:** `arr1 = [28,6,22,8,44,17]`, `arr2 = [22,28,8,6]`

**Output:** `[22,28,8,6,17,44]`

**Constraints:**

- `1 <= arr1.length, arr2.length <= 1000`
- `0 <= arr1[i], arr2[i] <= 1000`
- All the elements of `arr2` are **distinct**.
- Each `arr2[i]` is in `arr1`.

**1123. Lowest Common Ancestor of Deepest Leaves****Medium**

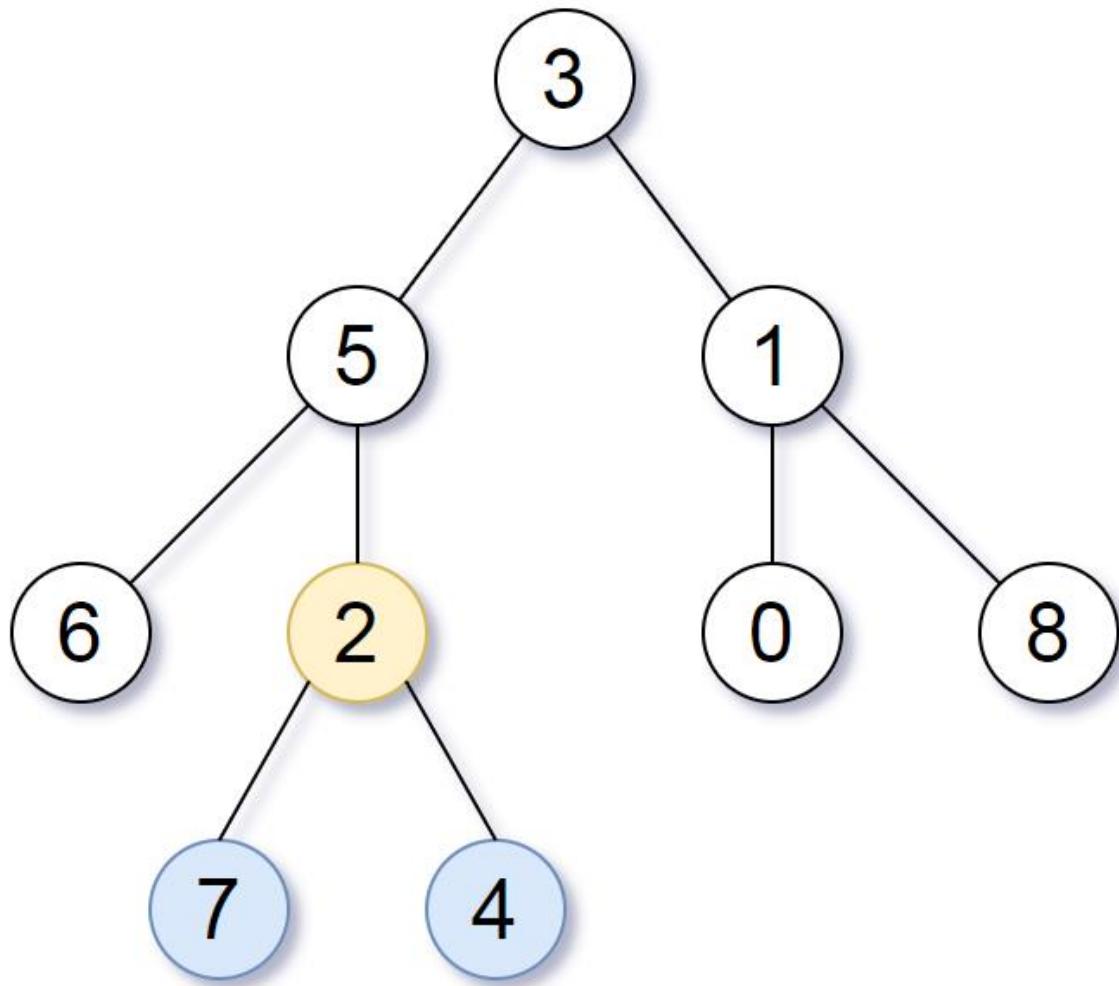
1480760Add to ListShare

Given the `root` of a binary tree, return *the lowest common ancestor of its deepest leaves*.

Recall that:

- The node of a binary tree is a leaf if and only if it has no children
- The depth of the root of the tree is `0`. if the depth of a node is `d`, the depth of each of its children is `d + 1`.
- The lowest common ancestor of a set `S` of nodes, is the node `A` with the largest depth such that every node in `S` is in the subtree with root `A`.

**Example 1:**



**Input:** root = [3,5,1,6,2,0,8,null,null,7,4]

**Output:** [2,7,4]

**Explanation:** We return the node with value 2, colored in yellow in the diagram.

The nodes coloured in blue are the deepest leaf-nodes of the tree.

Note that nodes 6, 0, and 8 are also leaf nodes, but the depth of them is 2, but the depth of nodes 7 and 4 is 3.

### Example 2:

**Input:** root = [1]

**Output:** [1]

**Explanation:** The root is the deepest node in the tree, and it's the lca of itself.

### Example 3:

**Input:** root = [0,1,3,null,2]

**Output:** [2]

**Explanation:** The deepest leaf node in the tree is 2, the lca of one node is itself.

**Constraints:**

- The number of nodes in the tree will be in the range [1, 1000].
- $0 \leq \text{Node.val} \leq 1000$
- The values of the nodes in the tree are **unique**.

## 1124. Longest Well-Performing Interval

**Medium**

110598Add to ListShare

We are given `hours`, a list of the number of hours worked per day for a given employee.

A day is considered to be a *tiring day* if and only if the number of hours worked is (strictly) greater than 8.

A *well-performing interval* is an interval of days for which the number of tiring days is strictly larger than the number of non-tiring days.

Return the length of the longest well-performing interval.

**Example 1:**

**Input:** hours = [9,9,6,0,6,6,9]

**Output:** 3

**Explanation:** The longest well-performing interval is [9,9,6].

**Example 2:**

**Input:** hours = [6,6,6]

**Output:** 0

**Constraints:**

- $1 \leq \text{hours.length} \leq 10^4$
- $0 \leq \text{hours}[i] \leq 16$

## 1125. Smallest Sufficient Team

Hard

81915Add to ListShare

In a project, you have a list of required skills `req_skills`, and a list of people. The  $i^{\text{th}}$  person `people[i]` contains a list of skills that the person has.

Consider a sufficient team: a set of people such that for every required skill in `req_skills`, there is at least one person in the team who has that skill. We can represent these teams by the index of each person.

- For example, `team = [0, 1, 3]` represents the people with skills `people[0]`, `people[1]`, and `people[3]`.

Return *any sufficient team of the smallest possible size, represented by the index of each person*. You may return the answer in **any order**.

It is **guaranteed** an answer exists.

### Example 1:

**Input:** `req_skills = ["java", "nodejs", "reactjs"]`, `people = [[["java"]], [["nodejs"]], [["nodejs", "reactjs"]]]`

**Output:** `[0,2]`

### Example 2:

**Input:** `req_skills = ["algorithms", "math", "java", "reactjs", "csharp", "aws"]`, `people = [[[["algorithms"]], [math], [java]], [[["algorithms"], [math], [reactjs]], [java, csharp, aws]], [[["reactjs"], [csharp]], [csharp, math], [aws, java]]]]`

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

### Constraints:

- $1 \leq \text{req_skills.length} \leq 16$
- $1 \leq \text{req_skills}[i].length \leq 16$
- `req_skills[i]` consists of lowercase English letters.
- All the strings of `req_skills` are **unique**.
- $1 \leq \text{people.length} \leq 60$
- $0 \leq \text{people}[i].length \leq 16$

- $1 \leq \text{people}[i][j].length \leq 16$
- `people[i][j]` consists of lowercase English letters.
- All the strings of `people[i]` are **unique**.
- Every skill in `people[i]` is a skill in `req_skills`.
- It is guaranteed a sufficient team exists.

## 1128. Number of Equivalent Domino Pairs

Easy

530264Add to ListShare

Given a list of `dominoes`, `dominoes[i] = [a, b]` is **equivalent to** `dominoes[j] = [c, d]` if and only if either  $(a == c \text{ and } b == d)$ , or  $(a == d \text{ and } b == c)$  - that is, one domino can be rotated to be equal to another domino.

Return the number of pairs  $(i, j)$  for which  $0 \leq i < j < \text{dominoes.length}$ , and `dominoes[i]` is **equivalent to** `dominoes[j]`.

**Example 1:**

**Input:** `dominoes = [[1,2],[2,1],[3,4],[5,6]]`

**Output:** 1

**Example 2:**

**Input:** `dominoes = [[1,2],[1,2],[1,1],[1,2],[2,2]]`

**Output:** 3

**Constraints:**

- $1 \leq \text{dominoes.length} \leq 4 * 10^4$
- `dominoes[i].length == 2`
- $1 \leq \text{dominoes}[i][j] \leq 9$

## 1129. Shortest Path with Alternating Colors

Medium

155677Add to ListShare

You are given an integer `n`, the number of nodes in a directed graph where the nodes are labeled from `0` to `n - 1`. Each edge is red or blue in this graph, and there could be self-edges and parallel edges.

You are given two arrays `redEdges` and `blueEdges` where:

- `redEdges[i] = [ai, bi]` indicates that there is a directed red edge from node `ai` to node `bi` in the graph, and
- `blueEdges[j] = [uj, vj]` indicates that there is a directed blue edge from node `uj` to node `vj` in the graph.

Return an array `answer` of length `n`, where each `answer[x]` is the length of the shortest path from node `0` to node `x` such that the edge colors alternate along the path, or `-1` if such a path does not exist.

### Example 1:

**Input:** `n = 3, redEdges = [[0,1],[1,2]], blueEdges = []`

**Output:** `[0,1,-1]`

### Example 2:

**Input:** `n = 3, redEdges = [[0,1]], blueEdges = [[2,1]]`

**Output:** `[0,1,-1]`

### Constraints:

- `1 <= n <= 100`
- `0 <= redEdges.length, blueEdges.length <= 400`
- `redEdges[i].length == blueEdges[j].length == 2`
- `0 <= ai, bi, uj, vj < n`

## 1130. Minimum Cost Tree From Leaf Values

Medium

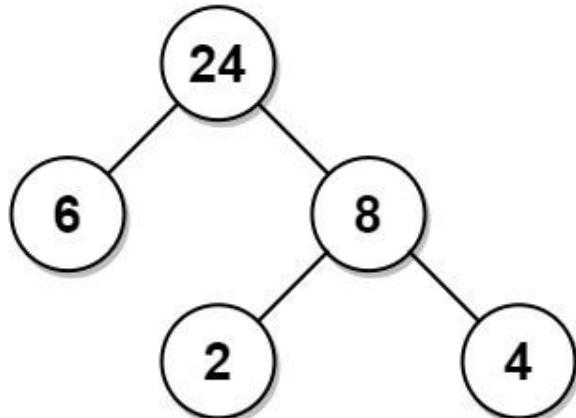
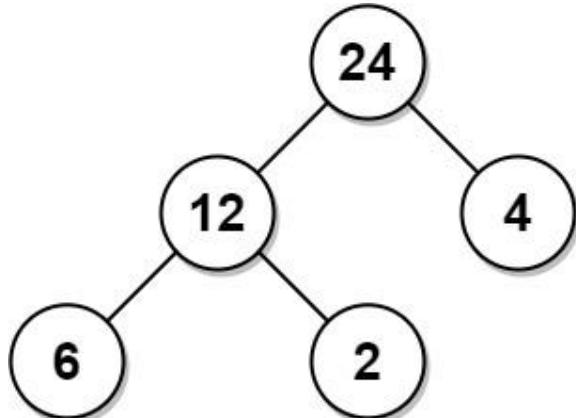
3530237 Add to List Share

Given an array `arr` of positive integers, consider all binary trees such that:

- Each node has either `0` or `2` children;
- The values of `arr` correspond to the values of each **leaf** in an in-order traversal of the tree.
- The value of each non-leaf node is equal to the product of the largest leaf value in its left and right subtree, respectively.

Among all possible binary trees considered, return *the smallest possible sum of the values of each non-leaf node*. It is guaranteed this sum fits into a **32-bit** integer.

A node is a **leaf** if and only if it has zero children.

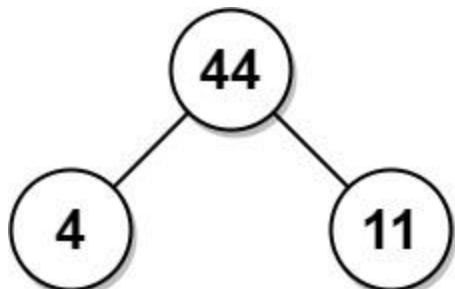
**Example 1:**

**Input:** arr = [6,2,4]

**Output:** 32

**Explanation:** There are two possible trees shown.

The first has a non-leaf node sum 36, and the second has non-leaf node sum 32.

**Example 2:**

**Input:** arr = [4,11]

**Output:** 44

**Constraints:**

- $2 \leq \text{arr.length} \leq 40$
- $1 \leq \text{arr}[i] \leq 15$
- It is guaranteed that the answer fits into a **32-bit** signed integer (i.e., it is less than  $2^{31}$ ).

### 1131. Maximum of Absolute Value Expression

Medium

512358Add to ListShare

Given two arrays of integers with equal lengths, return the maximum value of:

```
|arr1[i] - arr1[j]| + |arr2[i] - arr2[j]| + |i - j|
```

where the maximum is taken over all  $0 \leq i, j < \text{arr1.length}$ .

**Example 1:**

**Input:** arr1 = [1,2,3,4], arr2 = [-1,4,5,6]

**Output:** 13

**Example 2:**

**Input:** arr1 = [1,-2,-5,0,10], arr2 = [0,-2,-1,-7,-4]

**Output:** 20

**Constraints:**

- $2 \leq \text{arr1.length} == \text{arr2.length} \leq 40000$
- $-10^6 \leq \text{arr1}[i], \text{arr2}[i] \leq 10^6$

## 1137. N-th Tribonacci Number

Easy

2275125Add to ListShare

The Tribonacci sequence  $T_n$  is defined as follows:

$T_0 = 0, T_1 = 1, T_2 = 1$ , and  $T_{n+3} = T_n + T_{n+1} + T_{n+2}$  for  $n \geq 0$ .

Given  $n$ , return the value of  $T_n$ .

**Example 1:**

**Input:** n = 4

**Output:** 4

**Explanation:**

$T_3 = 0 + 1 + 1 = 2$

$T_4 = 1 + 1 + 2 = 4$

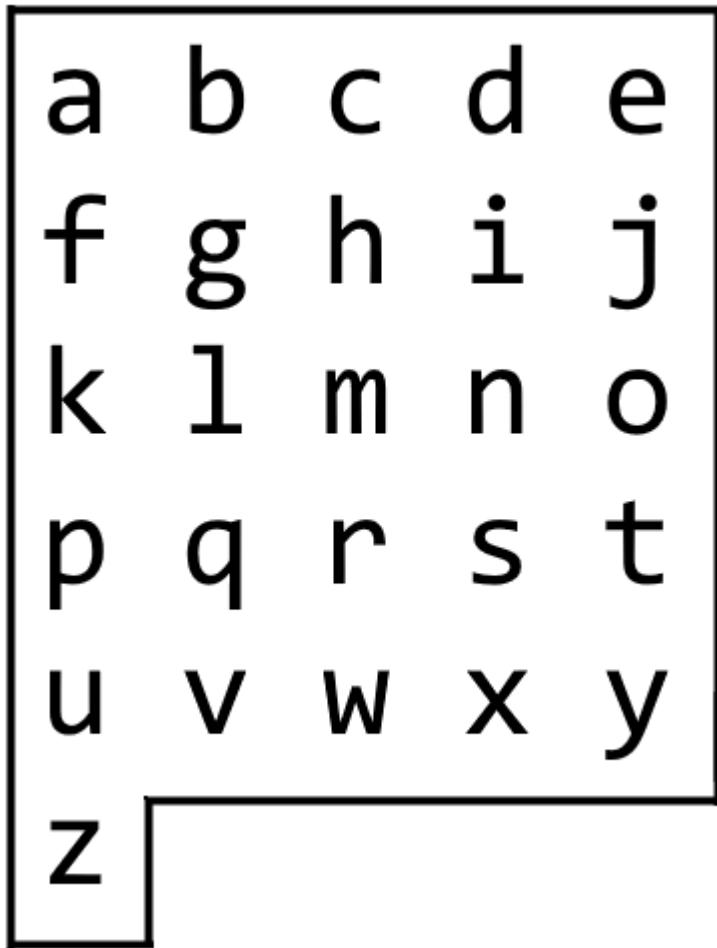
**Example 2:****Input:** n = 25**Output:** 1389537**Constraints:**

- $0 \leq n \leq 37$
- The answer is guaranteed to fit within a 32-bit integer, ie. `answer`  $\leq 2^{31} - 1$ .

**1138. Alphabet Board Path****Medium**

714146Add to ListShare

On an alphabet board, we start at position `(0, 0)`, corresponding to character `board[0][0]`.Here, `board = ["abcde", "fghij", "klmno", "pqrst", "uvwxyz"]`, as shown in the diagram below.



We may make the following moves:

- 'U' moves our position up one row, if the position exists on the board;
- 'D' moves our position down one row, if the position exists on the board;
- 'L' moves our position left one column, if the position exists on the board;
- 'R' moves our position right one column, if the position exists on the board;
- '!' adds the character `board[r][c]` at our current position `(r, c)` to the answer.

(Here, the only positions that exist on the board are positions with letters on them.)

Return a sequence of moves that makes our answer equal to `target` in the minimum number of moves. You may return any path that does so.

**Example 1:**

**Input:** `target = "leet"`

**Output:** "DDR!UURRR!!DDD!"

**Example 2:**

**Input:** target = "code"

**Output:** "RR!DDRR!UUL!R!"

**Constraints:**

- `1 <= target.length <= 100`
- `target` consists only of English lowercase letters.

## 1139. Largest 1-Bordered Square

Medium

57088Add to ListShare

Given a 2D `grid` of `0`s and `1`s, return the number of elements in the largest **square** subgrid that has all `1`s on its **border**, or `0` if such a subgrid doesn't exist in the `grid`.

**Example 1:**

**Input:** `grid = [[1,1,1],[1,0,1],[1,1,1]]`

**Output:** 9

**Example 2:**

**Input:** `grid = [[1,1,0,0]]`

**Output:** 1

**Constraints:**

- `1 <= grid.length <= 100`
- `1 <= grid[0].length <= 100`
- `grid[i][j]` is `0` or `1`

## 1140. Stone Game II

Medium

1503274Add to ListShare

Alice and Bob continue their games with piles of stones. There are a number of piles **arranged in a row**, and each pile has a positive integer number of stones `piles[i]`. The objective of the game is to end with the most stones.

Alice and Bob take turns, with Alice starting first. Initially, `M = 1`.

On each player's turn, that player can take **all the stones** in the **first  $X$**  remaining piles, where  $1 \leq X \leq 2M$ . Then, we set `M = max(M, X)`.

The game continues until all the stones have been taken.

Assuming Alice and Bob play optimally, return the maximum number of stones Alice can get.

### Example 1:

**Input:** `piles = [2,7,9,4,4]`

**Output:** `10`

**Explanation:** If Alice takes one pile at the beginning, Bob takes two piles, then Alice takes 2 piles again. Alice can get  $2 + 4 + 4 = 10$  piles in total. If Alice takes two piles at the beginning, then Bob can take all three piles left. In this case, Alice get  $2 + 7 = 9$  piles in total. So we return 10 since it's larger.

### Example 2:

**Input:** `piles = [1,2,3,4,5,100]`

**Output:** `104`

### Constraints:

- $1 \leq \text{piles.length} \leq 100$
- $1 \leq \text{piles}[i] \leq 10^4$

## 1141. User Activity for the Past 30 Days I

Easy

217336Add to ListShare

SQL Schema

Table: `Activity`

Column Name	Type
user_id	int
session_id	int
activity_date	date
activity_type	enum

There is no primary key for this table, it may have duplicate rows.

The activity\_type column is an ENUM of type ('open\_session', 'end\_session', 'scroll\_down', 'send\_message').

The table shows the user activities for a social media website.

Note that each session belongs to exactly one user.

Write an SQL query to find the daily active user count for a period of 30 days ending 2019-07-27 inclusively. A user was active on someday if they made at least one activity on that day.

Return the result table in **any order**.

The query result format is in the following example.

### Example 1:

**Input:**

Activity table:

user_id	session_id	activity_date	activity_type
1	1	2019-07-20	open_session
1	1	2019-07-20	scroll_down
1	1	2019-07-20	end_session
2	4	2019-07-20	open_session

2	4	2019-07-21	send_message
2	4	2019-07-21	end_session
3	2	2019-07-21	open_session
3	2	2019-07-21	send_message
3	2	2019-07-21	end_session
4	3	2019-06-25	open_session
4	3	2019-06-25	end_session

**Output:**

+-----+-----+
day   active_users
+-----+-----+
2019-07-20   2
2019-07-21   2
+-----+-----+

**Explanation:** Note that we do not care about days with zero active users.

### 1143. Longest Common Subsequence

Medium

834996Add to ListShare

Given two strings `text1` and `text2`, return *the length of their longest **common subsequence***. If there is no **common subsequence**, return 0.

A **subsequence** of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

- For example, "ace" is a subsequence of "abcde".

A **common subsequence** of two strings is a subsequence that is common to both strings.

**Example 1:**

**Input:** `text1 = "abcde", text2 = "ace"`

**Output:** 3

**Explanation:** The longest common subsequence is "ace" and its length is 3.

**Example 2:**

**Input:** text1 = "abc", text2 = "abc"

**Output:** 3

**Explanation:** The longest common subsequence is "abc" and its length is 3.

**Example 3:**

**Input:** text1 = "abc", text2 = "def"

**Output:** 0

**Explanation:** There is no such common subsequence, so the result is 0.

**Constraints:**

- $1 \leq \text{text1.length}, \text{text2.length} \leq 1000$
- `text1` and `text2` consist of only lowercase English characters.

## 1144. Decrease Elements To Make Array Zigzag

Medium

339151Add to ListShare

Given an array `nums` of integers, a *move* consists of choosing any element and **decreasing it by 1**.

An array `A` is a *zigzag array* if either:

- Every even-indexed element is greater than adjacent elements, ie.  $A[0] > A[1] < A[2] > A[3] < A[4] > \dots$
- OR, every odd-indexed element is greater than adjacent elements, ie.  $A[0] < A[1] > A[2] < A[3] > A[4] < \dots$

Return the minimum number of moves to transform the given array `nums` into a zigzag array.

**Example 1:**

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

**Output:** 2

**Explanation:** We can decrease 2 to 0 or 3 to 1.

**Example 2:**

**Input:** nums = [9,6,1,6,2]

**Output:** 4

**Constraints:**

- $1 \leq \text{nums.length} \leq 1000$
- $1 \leq \text{nums}[i] \leq 1000$

**1145. Binary Tree Coloring Game**

Medium

1090190Add to ListShare

Two players play a turn based game on a binary tree. We are given the `root` of this binary tree, and the number of nodes `n` in the tree. `n` is odd, and each node has a distinct value from `1` to `n`.

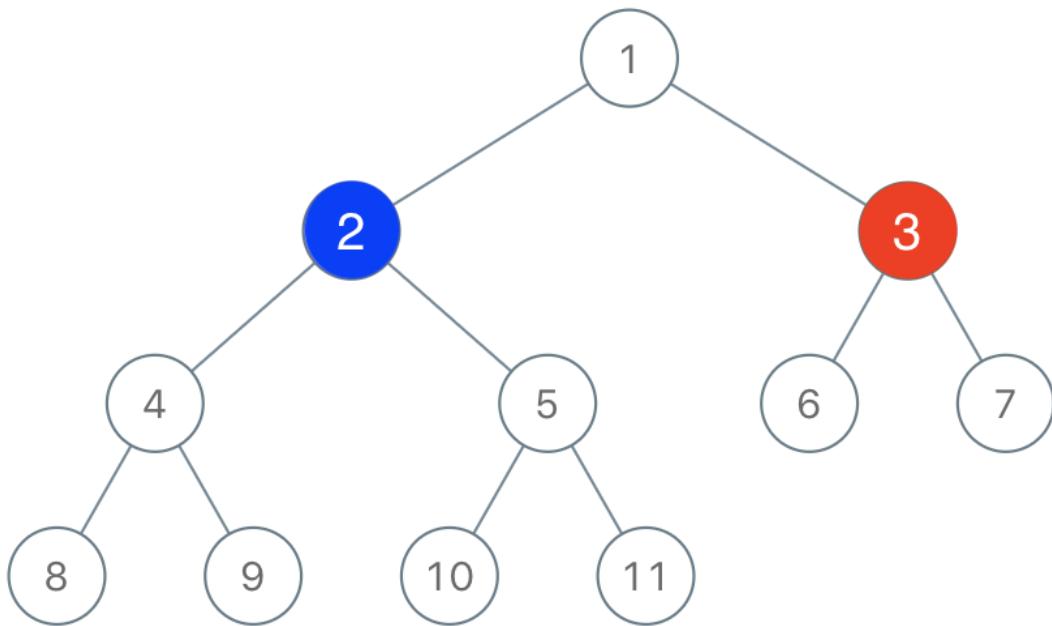
Initially, the first player names a value `x` with  $1 \leq x \leq n$ , and the second player names a value `y` with  $1 \leq y \leq n$  and  $y \neq x$ . The first player colors the node with value `x` red, and the second player colors the node with value `y` blue.

Then, the players take turns starting with the first player. In each turn, that player chooses a node of their color (red if player 1, blue if player 2) and colors an **uncolored** neighbor of the chosen node (either the left child, right child, or parent of the chosen node.)

If (and only if) a player cannot choose such a node in this way, they must pass their turn. If both players pass their turn, the game ends, and the winner is the player that colored more nodes.

You are the second player. If it is possible to choose such a `y` to ensure you win the game, return `true`. If it is not possible, return `false`.

**Example 1:**



**Input:** root = [1,2,3,4,5,6,7,8,9,10,11], n = 11, x = 3

**Output:** true

**Explanation:** The second player can choose the node with value 2.

**Example 2:**

**Input:** root = [1,2,3], n = 3, x = 1

**Output:** false

**Constraints:**

- The number of nodes in the tree is `n`.
- `1 <= x <= n <= 100`
- `n` is odd.
- `1 <= Node.val <= n`
- All the values of the tree are **unique**.

## 1146. Snapshot Array

Medium

1941281 Add to List Share

Implement a SnapshotArray that supports the following interface:

- `SnapshotArray(int length)` initializes an array-like data structure with the given length. **Initially, each element equals 0.**
- `void set(index, val)` sets the element at the given `index` to be equal to `val`.
- `int snap()` takes a snapshot of the array and returns the `snap_id`: the total number of times we called `snap()` minus 1.
- `int get(index, snap_id)` returns the value at the given `index`, at the time we took the snapshot with the given `snap_id`

### Example 1:

**Input:** `["SnapshotArray", "set", "snap", "set", "get"]`

`[[3], [0, 5], [], [0, 6], [0, 0]]`

**Output:** `[null, null, 0, null, 5]`

#### Explanation:

```
SnapshotArray snapshotArr = new SnapshotArray(3); // set the length to be 3
snapshotArr.set(0,5); // Set array[0] = 5
snapshotArr.snap(); // Take a snapshot, return snap_id = 0
snapshotArr.set(0,6);
snapshotArr.get(0,0); // Get the value of array[0] with snap_id = 0, return 5
```

#### Constraints:

- $1 \leq \text{length} \leq 5 * 10^4$
- $0 \leq \text{index} < \text{length}$
- $0 \leq \text{val} \leq 10^9$
- $0 \leq \text{snap_id} < (\text{the total number of times we call snap}())$
- At most  $5 * 10^4$  calls will be made to `set`, `snap`, and `get`.

## 1148. Article Views I

Easy

24324Add to ListShare

SQL Schema

Table: `Views`

Column Name	Type
article_id	int
author_id	int
viewer_id	int
view_date	date

There is no primary key for this table, it may have duplicate rows.

Each row of this table indicates that some viewer viewed an article (written by some author) on some date.

Note that equal author\_id and viewer\_id indicate the same person.

Write an SQL query to find all the authors that viewed at least one of their own articles.

Return the result table sorted by `id` in ascending order.

The query result format is in the following example.

### Example 1:

**Input:**

Views table:

article_id	author_id	viewer_id	view_date
1	3	5	2019-08-01
1	3	6	2019-08-02
2	7	7	2019-08-01
2	7	6	2019-08-02

```

| 4      | 7      | 1      | 2019-07-22 |
| 3      | 4      | 4      | 2019-07-21 |
| 3      | 4      | 4      | 2019-07-21 |
+-----+-----+-----+

```

**Output:**

```

+-----+
| id  |
+-----+
| 4   |
| 7   |
+-----+

```

## 1154. Day of the Year

**Easy**

281342Add to ListShare

Given a string `date` representing a [Gregorian calendar](#) date formatted as `YYYY-MM-DD`, return *the day number of the year*.

**Example 1:**

**Input:** `date = "2019-01-09"`

**Output:** 9

**Explanation:** Given date is the 9th day of the year in 2019.

**Example 2:**

**Input:** `date = "2019-02-10"`

**Output:** 41

**Constraints:**

- `date.length == 10`
- `date[4] == date[7] == '-'`, and all other `date[i]`'s are digits
- `date` represents a calendar date between Jan 1<sup>st</sup>, 1900 and Dec 31<sup>th</sup>, 2019.

## 1155. Number of Dice Rolls With Target Sum

Medium

231591Add to ListShare

You have  $n$  dice and each die has  $k$  faces numbered from 1 to  $k$ .

Given three integers  $n$ ,  $k$ , and  $target$ , return *the number of possible ways (out of the  $k^n$  total ways) to roll the dice so the sum of the face-up numbers equals  $target$* . Since the answer may be too large, return it **modulo  $10^9 + 7$** .

**Example 1:**

**Input:**  $n = 1$ ,  $k = 6$ ,  $target = 3$

**Output:** 1

**Explanation:** You throw one die with 6 faces.

There is only one way to get a sum of 3.

**Example 2:**

**Input:**  $n = 2$ ,  $k = 6$ ,  $target = 7$

**Output:** 6

**Explanation:** You throw two dice, each with 6 faces.

There are 6 ways to get a sum of 7: 1+6, 2+5, 3+4, 4+3, 5+2, 6+1.

**Example 3:**

**Input:**  $n = 30$ ,  $k = 30$ ,  $target = 500$

**Output:** 222616187

**Explanation:** The answer must be returned modulo  $10^9 + 7$ .

**Constraints:**

- $1 \leq n, k \leq 30$
- $1 \leq target \leq 1000$

## 1156. Swap For Longest Repeated Character Substring

Medium

80172Add to ListShare

You are given a string `text`. You can swap two of the characters in the `text`.

Return *the length of the longest substring with repeated characters*.

**Example 1:**

**Input:** `text = "ababa"`

**Output:** 3

**Explanation:** We can swap the first 'b' with the last 'a', or the last 'b' with the first 'a'. Then, the longest repeated character substring is "aaa" with length 3.

**Example 2:**

**Input:** `text = "aaabaaa"`

**Output:** 6

**Explanation:** Swap 'b' with the last 'a' (or the first 'a'), and we get longest repeated character substring "aaaaaa" with length 6.

**Example 3:**

**Input:** `text = "aaaaa"`

**Output:** 5

**Explanation:** No need to swap, longest repeated character substring is "aaaaa" with length is 5.

**Constraints:**

- $1 \leq \text{text.length} \leq 2 * 10^4$
- `text` consist of lowercase English characters only.

## 1157. Online Majority Element In Subarray

Hard

50550Add to ListShare

Design a data structure that efficiently finds the **majority element** of a given subarray.

The **majority element** of a subarray is an element that occurs `threshold` times or more in the subarray.

Implementing the `MajorityChecker` class:

- `MajorityChecker(int[] arr)` Initializes the instance of the class with the given array `arr`.
- `int query(int left, int right, int threshold)` returns the element in the subarray `arr[left...right]` that occurs at least `threshold` times, or `-1` if no such element exists.

### Example 1:

#### Input

```
[ "MajorityChecker", "query", "query", "query" ]
[[[1, 1, 2, 2, 1, 1]], [0, 5, 4], [0, 3, 3], [2, 3, 2]]
```

#### Output

```
[null, 1, -1, 2]
```

#### Explanation

```
MajorityChecker majorityChecker = new MajorityChecker([1, 1, 2, 2, 1, 1]);
majorityChecker.query(0, 5, 4); // return 1
majorityChecker.query(0, 3, 3); // return -1
majorityChecker.query(2, 3, 2); // return 2
```

#### Constraints:

- $1 \leq \text{arr.length} \leq 2 * 10^4$
- $1 \leq \text{arr}[i] \leq 2 * 10^4$
- $0 \leq \text{left} \leq \text{right} < \text{arr.length}$
- $\text{threshold} \leq \text{right} - \text{left} + 1$
- $2 * \text{threshold} > \text{right} - \text{left} + 1$
- At most  $10^4$  calls will be made to `query`.

## 1158. Market Analysis I

Medium

31251Add to ListShare

SQL Schema

Table: `Users`

Column Name	Type
<code>user_id</code>	<code>int</code>
<code>join_date</code>	<code>date</code>
<code>favorite_brand</code>	<code>varchar</code>

`user_id` is the primary key of this table.

This table has the info of the users of an online shopping website where users can sell and buy items.

Table: `Orders`

Column Name	Type
<code>order_id</code>	<code>int</code>
<code>order_date</code>	<code>date</code>
<code>item_id</code>	<code>int</code>
<code>buyer_id</code>	<code>int</code>
<code>seller_id</code>	<code>int</code>

`order_id` is the primary key of this table.

`item_id` is a foreign key to the `Items` table.

`buyer_id` and `seller_id` are foreign keys to the `Users` table.

Table: `Items`

```

+-----+-----+
| Column Name | Type   |
+-----+-----+
| item_id     | int    |
| item_brand  | varchar |
+-----+-----+
item_id is the primary key of this table.

```

Write an SQL query to find for each user, the join date and the number of orders they made as a buyer in [2019](#).

Return the result table in **any order**.

The query result format is in the following example.

### Example 1:

#### Input:

Users table:

```

+-----+-----+-----+
| user_id | join_date | favorite_brand |
+-----+-----+-----+
| 1       | 2018-01-01 | Lenovo        |
| 2       | 2018-02-09 | Samsung       |
| 3       | 2018-01-19 | LG            |
| 4       | 2018-05-21 | HP            |
+-----+-----+-----+

```

Orders table:

```

+-----+-----+-----+-----+-----+
| order_id | order_date | item_id | buyer_id | seller_id |
+-----+-----+-----+-----+-----+

```

1	2019-08-01	4	1	2	
2	2018-08-02	2	1	3	
3	2019-08-03	3	2	3	
4	2018-08-04	1	4	2	
5	2018-08-04	1	3	4	
6	2019-08-05	2	2	4	

Items table:

item_id	item_brand
1	Samsung
2	Lenovo
3	LG
4	HP

Output:

buyer_id	join_date	orders_in_2019
1	2018-01-01	1
2	2018-02-09	2
3	2018-01-19	0
4	2018-05-21	0

## 1160. Find Words That Can Be Formed by Characters

Easy

1149134Add to ListShare

You are given an array of strings `words` and a string `chars`.

A string is **good** if it can be formed by characters from `chars` (each character can only be used once).

Return *the sum of lengths of all good strings in words*.

**Example 1:**

**Input:** `words` = ["cat", "bt", "hat", "tree"], `chars` = "atach"

**Output:** 6

**Explanation:** The strings that can be formed are "cat" and "hat" so the answer is  $3 + 3 = 6$ .

**Example 2:**

**Input:** `words` = ["hello", "world", "leetcode"], `chars` = "welldonehoneyr"

**Output:** 10

**Explanation:** The strings that can be formed are "hello" and "world" so the answer is  $5 + 5 = 10$ .

**Constraints:**

- $1 \leq \text{words.length} \leq 1000$
- $1 \leq \text{words[i].length}, \text{chars.length} \leq 100$
- `words[i]` and `chars` consist of lowercase English letters.

## 1161. Maximum Level Sum of a Binary Tree

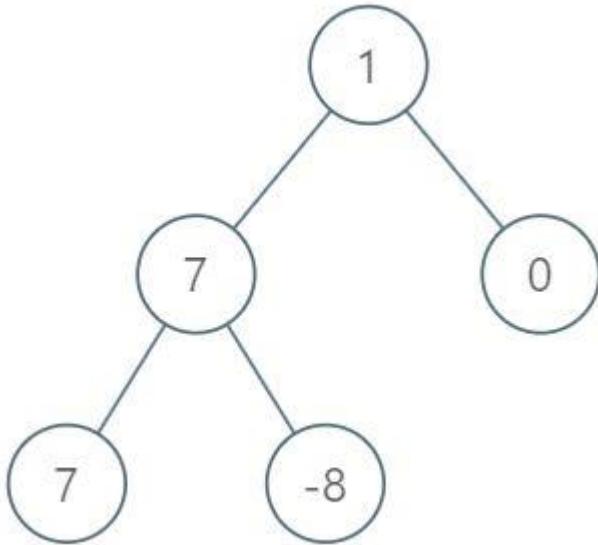
Medium

165263Add to ListShare

Given the `root` of a binary tree, the level of its root is `1`, the level of its children is `2`, and so on.

Return the **smallest** level `x` such that the sum of all the values of nodes at level `x` is **maximal**.

**Example 1:**



**Input:** root = [1,7,0,7,-8,null,null]

**Output:** 2

**Explanation:**

Level 1 sum = 1.

Level 2 sum =  $7 + 0 = 7$ .

Level 3 sum =  $7 + -8 = -1$ .

So we return the level with the maximum sum which is level 2.

**Example 2:**

**Input:** root = [989,null,10250,98693,-89388,null,null,null,-32127]

**Output:** 2

**Constraints:**

- The number of nodes in the tree is in the range  $[1, 10^4]$ .
- $-10^5 \leq \text{Node.val} \leq 10^5$

## 1162. As Far from Land as Possible

Medium

223566Add to ListShare

Given an  $n \times n$  grid containing only values 0 and 1, where 0 represents water and 1 represents land, find a water cell such that its distance to the nearest land cell is maximized, and return the distance. If no land or water exists in the grid, return -1.

The distance used in this problem is the Manhattan distance: the distance between two cells  $(x_0, y_0)$  and  $(x_1, y_1)$  is  $|x_0 - x_1| + |y_0 - y_1|$ .

**Example 1:**

1	0	1
0	0	0
1	0	1

**Input:** grid = [[1,0,1],[0,0,0],[1,0,1]]

**Output:** 2

**Explanation:** The cell (1, 1) is as far as possible from all the land with distance 2.

**Example 2:**

1	0	0
0	0	0
0	0	0

**Input:** grid = [[1,0,0],[0,0,0],[0,0,0]]

**Output:** 4

**Explanation:** The cell (2, 2) is as far as possible from all the land with distance 4.

**Constraints:**

- $n == \text{grid.length}$
- $n == \text{grid[i].length}$
- $1 \leq n \leq 100$
- $\text{grid[i][j]}$  is 0 or 1

## 1163. Last Substring in Lexicographical Order

Hard

488424Add to ListShare

Given a string  $s$ , return the last substring of  $s$  in lexicographical order.

**Example 1:****Input:** s = "abab"**Output:** "bab"**Explanation:** The substrings are ["a", "ab", "aba", "abab", "b", "ba", "bab"]. The lexicographically maximum substring is "bab".**Example 2:****Input:** s = "leetcode"**Output:** "tcode"**Constraints:**

- `1 <= s.length <= 4 * 105`
- `s` contains only lowercase English letters.

**1169. Invalid Transactions****Medium**

3541735Add to ListShare

A transaction is possibly invalid if:

- the amount exceeds \$1000, or;
- if it occurs within (and including) 60 minutes of another transaction with the **same name** in a **different city**.

You are given an array of strings `transaction` where `transactions[i]` consists of comma-separated values representing the name, time (in minutes), amount, and city of the transaction.

Return a list of `transactions` that are possibly invalid. You may return the answer in **any order**.

**Example 1:****Input:** transactions = ["alice,20,800,mtv", "alice,50,100,beijing"]**Output:** ["alice,20,800,mtv", "alice,50,100,beijing"]**Explanation:** The first transaction is invalid because the second transaction occurs within a difference of 60 minutes, have the same name and is in a different city. Similarly the second one is invalid too.

**Example 2:**

**Input:** transactions = ["alice,20,800,mtv", "alice,50,1200,mtv"]

**Output:** ["alice,50,1200,mtv"]

**Example 3:**

**Input:** transactions = ["alice,20,800,mtv", "bob,50,1200,mtv"]

**Output:** ["bob,50,1200,mtv"]

**Constraints:**

- `transactions.length <= 1000`
- Each `transactions[i]` takes the form "`{name},{time},{amount},{city}`"
- Each `{name}` and `{city}` consist of lowercase English letters, and have lengths between 1 and 10.
- Each `{time}` consist of digits, and represent an integer between 0 and 1000.
- Each `{amount}` consist of digits, and represent an integer between 0 and 2000.

## 1170. Compare Strings by Frequency of the Smallest Character

Medium

548920Add to ListShare

Let the function `f(s)` be the **frequency of the lexicographically smallest character** in a non-empty string `s`. For example, if `s = "dcce"` then `f(s) = 2` because the lexicographically smallest character is '`c`', which has a frequency of 2.

You are given an array of strings `words` and another array of query strings `queries`. For each query `queries[i]`, count the **number of words** in `words` such that `f(queries[i]) < f(W)` for each `W` in `words`.

Return an integer array `answer`, where each `answer[i]` is the answer to the `ith` query.

**Example 1:**

**Input:** queries = ["cbd"], words = ["zaaaz"]

**Output:** [1]

**Explanation:** On the first query we have `f("cbd") = 1`, `f("zaaaz") = 3` so `f("cbd") < f("zaaaz")`.

**Example 2:**

**Input:** queries = ["bbb", "cc"], words = ["a", "aa", "aaa", "aaaa"]

**Output:** [1,2]

**Explanation:** On the first query only  $f("bbb") < f("aaaa")$ . On the second query both  $f("aaa")$  and  $f("aaaa")$  are both  $> f("cc")$ .

### Constraints:

- $1 \leq \text{queries.length} \leq 2000$
- $1 \leq \text{words.length} \leq 2000$
- $1 \leq \text{queries[i].length}, \text{words[i].length} \leq 10$
- $\text{queries[i][j]}, \text{words[i][j]}$  consist of lowercase English letters.

## 1171. Remove Zero Sum Consecutive Nodes from Linked List

Medium

164677Add to ListShare

Given the `head` of a linked list, we repeatedly delete consecutive sequences of nodes that sum to `0` until there are no such sequences.

After doing so, return the head of the final linked list. You may return any such answer.

(Note that in the examples below, all sequences are serializations of `ListNode` objects.)

### Example 1:

**Input:** head = [1,2,-3,3,1]

**Output:** [3,1]

**Note:** The answer [1,2,1] would also be accepted.

### Example 2:

**Input:** head = [1,2,3,-3,4]

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

### Example 3:

**Input:** head = [1,2,3,-3,-2]

**Output:** [1]

**Constraints:**

- The given linked list will contain between 1 and 1000 nodes.
- Each node in the linked list has  $-1000 \leq \text{node.val} \leq 1000$ .

**1172. Dinner Plate Stacks****Hard**

36952Add to ListShare

You have an infinite number of stacks arranged in a row and numbered (left to right) from 0, each of the stacks has the same maximum capacity.

Implement the `DinnerPlates` class:

- `DinnerPlates(int capacity)` Initializes the object with the maximum capacity of the stacks `capacity`.
- `void push(int val)` Pushes the given integer `val` into the leftmost stack with a size less than `capacity`.
- `int pop()` Returns the value at the top of the rightmost non-empty stack and removes it from that stack, and returns `-1` if all the stacks are empty.
- `int popAtStack(int index)` Returns the value at the top of the stack with the given `index` `index` and removes it from that stack or returns `-1` if the stack with that given index is empty.

**Example 1:****Input**

```
["DinnerPlates", "push", "push", "push", "push", "push", "popAtStack", "push",
 "push", "popAtStack", "popAtStack", "pop", "pop", "pop", "pop", "pop"]
```

```
[[2], [1], [2], [3], [4], [5], [0], [20], [21], [0], [2], [], [], [], [], []]
```

**Output**

```
[null, null, null, null, null, null, 2, null, null, 20, 21, 5, 4, 3, 1, -1]
```

**Explanation:**

```
DinnerPlates D = DinnerPlates(2); // Initialize with capacity = 2
D.push(1);
D.push(2);
D.push(3);
```

```
D.push(4);

D.push(5);      // The stacks are now: 2 4
                1 3 5
                — — —

D.popAtStack(0); // Returns 2. The stacks are now:      4
                1 3 5
                — — —

D.push(20);      // The stacks are now: 20 4
                1 3 5
                — — —

D.push(21);      // The stacks are now: 20 4 21
                1 3 5
                — — —

D.popAtStack(0); // Returns 20. The stacks are now:      4 21
                1 3 5
                — — —

D.popAtStack(2); // Returns 21. The stacks are now:      4
                1 3 5
                — — —

D.pop();         // Returns 5. The stacks are now:      4
                1 3
                — —

D.pop();         // Returns 4. The stacks are now:      1 3
                — —

D.pop();         // Returns 3. The stacks are now:      1
```

```

D.pop()           // Returns 1.  There are no stacks.
D.pop()           // Returns -1.  There are still no stacks.

```

### Constraints:

- $1 \leq \text{capacity} \leq 2 * 10^4$
- $1 \leq \text{val} \leq 2 * 10^4$
- $0 \leq \text{index} \leq 10^5$
- At most  $2 * 10^5$  calls will be made to `push`, `pop`, and `popAtStack`.

## 1175. Prime Arrangements

Easy

281390Add to ListShare

Return the number of permutations of 1 to `n` so that prime numbers are at prime indices (1-indexed.)

(Recall that an integer is prime if and only if it is greater than 1, and cannot be written as a product of two positive integers both smaller than it.)

Since the answer may be large, return the answer **modulo  $10^9 + 7$** .

### Example 1:

**Input:** `n = 5`

**Output:** 12

**Explanation:** For example `[1,2,5,4,3]` is a valid permutation, but `[5,2,3,4,1]` is not because the prime number 5 is at index 1.

### Example 2:

**Input:** `n = 100`

**Output:** 682289015

### Constraints:

- $1 \leq n \leq 100$

## 1177. Can Make Palindrome from Substring

Medium

606239Add to ListShare

You are given a string `s` and array `queries` where `queries[i] = [lefti, righti, ki]`. We may rearrange the substring `s[lefti...righti]` for each query and then choose up to `ki` of them to replace with any lowercase English letter.

If the substring is possible to be a palindrome string after the operations above, the result of the query is `true`. Otherwise, the result is `false`.

Return a boolean array `answer` where `answer[i]` is the result of the `ith` query `queries[i]`.

Note that each letter is counted individually for replacement, so if, for example `s[lefti...righti] = "aaa"`, and `ki = 2`, we can only replace two of the letters. Also, note that no query modifies the initial string `s`.

**Example :**

**Input:** `s = "abcda"`, `queries = [[3,3,0],[1,2,0],[0,3,1],[0,3,2],[0,4,1]]`

**Output:** `[true, false, false, true, true]`

**Explanation:**

`queries[0]`: substring = "d", is palindrome.

`queries[1]`: substring = "bc", is not palindrome.

`queries[2]`: substring = "abcd", is not palindrome after replacing only 1 character.

`queries[3]`: substring = "abcd", could be changed to "abba" which is palindrome. Also this can be changed to "baab" first rearrange it "bacd" then replace "cd" with "ab".

`queries[4]`: substring = "abcda", could be changed to "abcba" which is palindrome.

**Example 2:**

**Input:** `s = "lyb"`, `queries = [[0,1,0],[2,2,1]]`

**Output:** `[false, true]`

**Constraints:**

- `1 <= s.length, queries.length <= 105`
- `0 <= lefti <= righti < s.length`

- $0 \leq k_i \leq s.length$
- $s$  consists of lowercase English letters.

## 1178. Number of Valid Words for Each Puzzle

Hard

114276Add to ListShare

With respect to a given `puzzle` string, a `word` is *valid* if both the following conditions are satisfied:

- `word` contains the first letter of `puzzle`.
- For each letter in `word`, that letter is in `puzzle`.
  - For example, if the puzzle is "abcdefg", then valid words are "faced", "cabbage", and "baggage", while
  - invalid words are "beefed" (does not include 'a') and "based" (includes 's' which is not in the puzzle).

Return an array `answer`, where `answer[i]` is the number of words in the given word list `words` that is valid with respect to the puzzle `puzzles[i]`.

### Example 1:

**Input:** `words = ["aaaa", "asas", "able", "ability", "actt", "actor", "access"]`, `puzzles = ["aboveyz", "abrodyz", "bsolute", "absoryz", "actresz", "gaswxyz"]`

**Output:** `[1,1,3,2,4,0]`

**Explanation:**

```
1 valid word for "aboveyz" : "aaaa"
1 valid word for "abrodyz" : "aaaa"
3 valid words for "bsolute" : "aaaa", "asas", "able"
2 valid words for "absoryz" : "aaaa", "asas"
4 valid words for "actresz" : "aaaa", "asas", "actt", "access"
```

There are no valid words for "gaswxyz" cause none of the words in the list contains letter 'g'.

### Example 2:

**Input:** `words = ["apple", "pleas", "please"]`, `puzzles = ["aelwxyz", "aelpxyz", "aelpsxy", "saelpxy", "xaelpsy"]`

**Output:** `[0,1,3,2,0]`

**Constraints:**

- $1 \leq \text{words.length} \leq 10^5$
- $4 \leq \text{words[i].length} \leq 50$
- $1 \leq \text{puzzles.length} \leq 10^4$
- $\text{puzzles[i].length} == 7$
- $\text{words[i]}$  and  $\text{puzzles[i]}$  consist of lowercase English letters.
- Each  $\text{puzzles[i]}$  does not contain repeated characters.

**1179. Reformat Department Table****Easy**

509408Add to ListShare

SQL Schema

Table: Department

Column Name	Type
<code>id</code>	<code>int</code>
<code>revenue</code>	<code>int</code>
<code>month</code>	<code>varchar</code>

`(id, month)` is the primary key of this table.

The table has information about the revenue of each department per month.

The month has values in  
["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"].Write an SQL query to reformat the table such that there is a department id column and a revenue column **for each month**.Return the result table in **any order**.

The query result format is in the following example.

**Example 1:****Input:**

Department table:

id	revenue	month
1	8000	Jan
2	9000	Jan
3	10000	Feb
1	7000	Feb
1	6000	Mar

**Output:**

id	Jan_Revenue	Feb_Revenue	Mar_Revenue	...	Dec_Revenue
1	8000	7000	6000	...	null
2	9000	null	null	...	null
3	null	10000	null	...	null

**Explanation:** The revenue from Apr to Dec is null.

Note that the result table has 13 columns (1 for the department id + 12 for the months).

**1184. Distance Between Bus Stops****Easy**

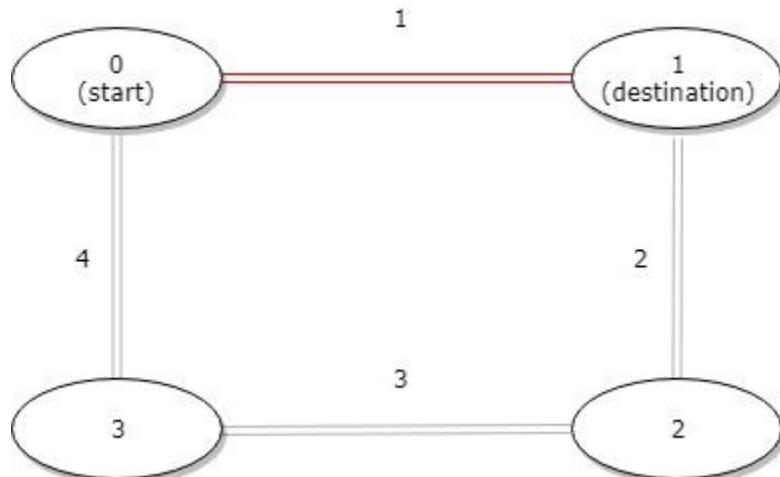
58260Add to ListShare

A bus has  $n$  stops numbered from  $0$  to  $n - 1$  that form a circle. We know the distance between all pairs of neighboring stops where  $\text{distance}[i]$  is the distance between the stops number  $i$  and  $(i + 1) \% n$ .

The bus goes along both directions i.e. clockwise and counterclockwise.

Return the shortest distance between the given `start` and `destination` stops.

**Example 1:**

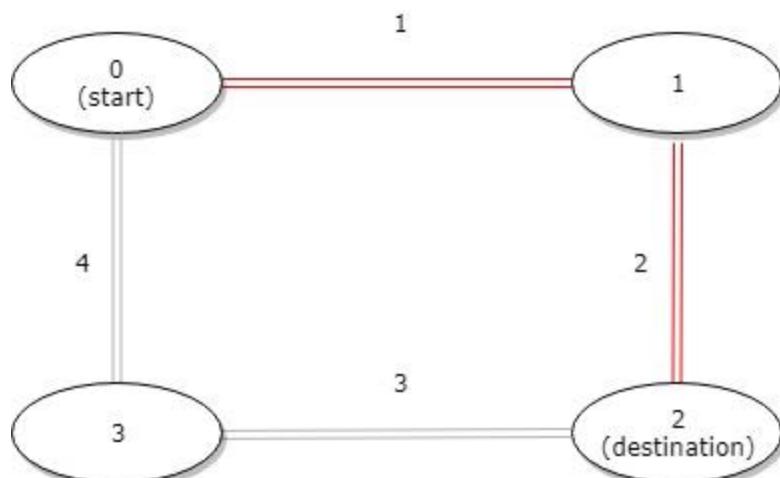


**Input:** `distance = [1,2,3,4]`, `start = 0`, `destination = 1`

**Output:** 1

**Explanation:** Distance between 0 and 1 is 1 or 9, minimum is 1.

**Example 2:**

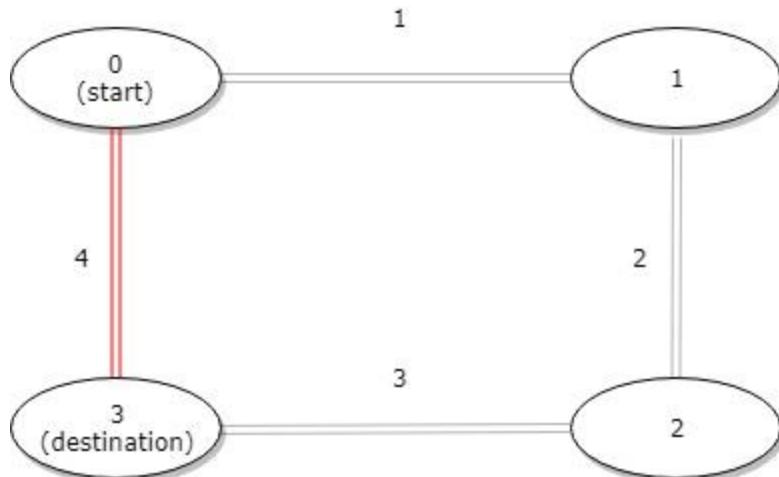


**Input:** `distance = [1,2,3,4]`, `start = 0`, `destination = 2`

**Output:** 3

**Explanation:** Distance between 0 and 2 is 3 or 7, minimum is 3.

**Example 3:**



**Input:** distance = [1,2,3,4], start = 0, destination = 3

**Output:** 4

**Explanation:** Distance between 0 and 3 is 6 or 4, minimum is 4.

**Constraints:**

- $1 \leq n \leq 10^4$
- $\text{distance.length} == n$
- $0 \leq \text{start}, \text{destination} < n$
- $0 \leq \text{distance}[i] \leq 10^4$

## 1185. Day of the Week

Easy

2702052Add to ListShare

Given a date, return the corresponding day of the week for that date.

The input is given as three integers representing the `day`, `month` and `year` respectively.

Return the answer as one of the following values `{"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"}`.

**Example 1:**

**Input:** day = 31, month = 8, year = 2019

**Output:** "Saturday"

**Example 2:**

**Input:** day = 18, month = 7, year = 1999

**Output:** "Sunday"

**Example 3:**

**Input:** day = 15, month = 8, year = 1993

**Output:** "Sunday"

**Constraints:**

- The given dates are valid dates between the years 1971 and 2100.

## 1186. Maximum Subarray Sum with One Deletion

Medium

141247Add to ListShare

Given an array of integers, return the maximum sum for a **non-empty** subarray (contiguous elements) with at most one element deletion. In other words, you want to choose a subarray and optionally delete one element from it so that there is still at least one element left and the sum of the remaining elements is maximum possible.

Note that the subarray needs to be **non-empty** after deleting one element.

**Example 1:**

**Input:** arr = [1, -2, 0, 3]

**Output:** 4

**Explanation:** Because we can choose [1, -2, 0, 3] and drop -2, thus the subarray [1, 0, 3] becomes the maximum value.

**Example 2:**

**Input:** arr = [1, -2, -2, 3]

**Output:** 3

**Explanation:** We just choose [3] and it's the maximum sum.

**Example 3:****Input:** arr = [-1, -1, -1, -1]**Output:** -1**Explanation:** The final subarray needs to be non-empty. You can't choose [-1] and delete -1 from it, then get an empty subarray to make the sum equals to 0.**Constraints:**

- $1 \leq \text{arr.length} \leq 10^5$
- $-10^4 \leq \text{arr}[i] \leq 10^4$

**1187. Make Array Strictly Increasing****Hard**

73116Add to ListShare

Given two integer arrays `arr1` and `arr2`, return the minimum number of operations (possibly zero) needed to make `arr1` strictly increasing.In one operation, you can choose two indices  $0 \leq i < \text{arr1.length}$  and  $0 \leq j < \text{arr2.length}$  and do the assignment `arr1[i] = arr2[j]`.If there is no way to make `arr1` strictly increasing, return -1.**Example 1:****Input:** arr1 = [1, 5, 3, 6, 7], arr2 = [1, 3, 2, 4]**Output:** 1**Explanation:** Replace 5 with 2, then `arr1 = [1, 2, 3, 6, 7]`.**Example 2:****Input:** arr1 = [1, 5, 3, 6, 7], arr2 = [4, 3, 1]**Output:** 2**Explanation:** Replace 5 with 3 and then replace 3 with 4. `arr1 = [1, 3, 4, 6, 7]`.**Example 3:****Input:** arr1 = [1, 5, 3, 6, 7], arr2 = [1, 6, 3, 3]**Output:** -1

**Explanation:** You can't make arr1 strictly increasing.

**Constraints:**

- $1 \leq \text{arr1.length}, \text{arr2.length} \leq 2000$
- $0 \leq \text{arr1[i]}, \text{arr2[i]} \leq 10^9$

## 1189. Maximum Number of Balloons

Easy

115472Add to ListShare

Given a string `text`, you want to use the characters of `text` to form as many instances of the word "**balloon**" as possible.

You can use each character in `text` **at most once**. Return the maximum number of instances that can be formed.

**Example 1:**

nlaebolko

**Input:** text = "nlaebolko"

**Output:** 1

**Example 2:**

loonbalxballpoon

**Input:** text = "loonbalxballpoon"

**Output:** 2

**Example 3:**

**Input:** text = "leetcode"

**Output:** 0

**Constraints:**

- `1 <= text.length <= 104`
- `text` consists of lower case English letters only.

**1190. Reverse Substrings Between Each Pair of Parentheses****Medium**

146139Add to ListShare

You are given a string `s` that consists of lower case English letters and brackets.

Reverse the strings in each pair of matching parentheses, starting from the innermost one.

Your result should **not** contain any brackets.

**Example 1:**

**Input:** `s = "(abcd)"`

**Output:** "dcba"

**Example 2:**

**Input:** `s = "(u(love)i)"`

**Output:** "iloveu"

**Explanation:** The substring "love" is reversed first, then the whole string is reversed.

**Example 3:**

**Input:** `s = "(ed(et(oc))el)"`

**Output:** "leetcode"

**Explanation:** First, we reverse the substring "oc", then "etco", and finally, the whole string.

**Constraints:**

- `1 <= s.length <= 2000`
- `s` only contains lower case English characters and parentheses.
- It is guaranteed that all parentheses are balanced.

**1191. K-Concatenation Maximum Sum**

**Medium**

108492Add to ListShare

Given an integer array `arr` and an integer `k`, modify the array by repeating it `k` times.For example, if `arr = [1, 2]` and `k = 3` then the modified array will be `[1, 2, 1, 2, 1, 2]`.

Return the maximum sub-array sum in the modified array. Note that the length of the sub-array can be 0 and its sum in that case is 0.

As the answer can be very large, return the answer **modulo**  $10^9 + 7$ .**Example 1:****Input:** `arr = [1,2]`, `k = 3`**Output:** 9**Example 2:****Input:** `arr = [1,-2,1]`, `k = 5`**Output:** 2**Example 3:****Input:** `arr = [-1,-2]`, `k = 7`**Output:** 0**Constraints:**

- $1 \leq \text{arr.length} \leq 10^5$
- $1 \leq k \leq 10^5$
- $-10^4 \leq \text{arr}[i] \leq 10^4$

**1192. Critical Connections in a Network****Hard**

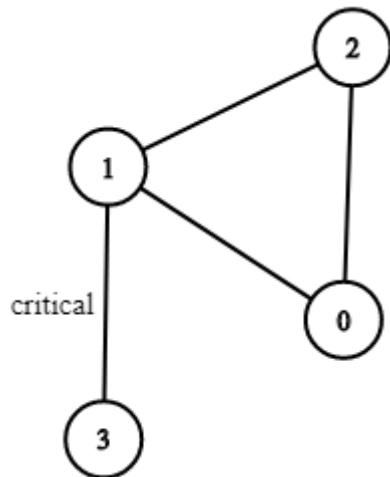
4908166Add to ListShare

There are `n` servers numbered from 0 to `n - 1` connected by undirected server-to-server `connections` forming a network where `connections[i] = [ai, bi]` represents a connection between servers `ai` and `bi`. Any server can reach other servers directly or indirectly through the network.

A *critical connection* is a connection that, if removed, will make some servers unable to reach some other server.

Return all critical connections in the network in any order.

**Example 1:**



**Input:**  $n = 4$ ,  $\text{connections} = [[0,1],[1,2],[2,0],[1,3]]$

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

**Explanation:**  $[[3,1]]$  is also accepted.

**Example 2:**

**Input:**  $n = 2$ ,  $\text{connections} = [[0,1]]$

**Output:**  $[[0,1]]$

**Constraints:**

- $2 \leq n \leq 10^5$
- $n - 1 \leq \text{connections.length} \leq 10^5$
- $0 \leq a_i, b_i \leq n - 1$
- $a_i \neq b_i$
- There are no repeated connections.

## 1195. Fizz Buzz Multithreaded

Medium

422316Add to ListShare

You have the four functions:

- `printFizz` that prints the word "fizz" to the console,
- `printBuzz` that prints the word "buzz" to the console,
- `printFizzBuzz` that prints the word "fizzbuzz" to the console, and
- `printNumber` that prints a given integer to the console.

You are given an instance of the class `FizzBuzz` that has four functions: `fizz`, `buzz`, `fizzbuzz` and `number`. The same instance of `FizzBuzz` will be passed to four different threads:

- **Thread A:** calls `fizz()` that should output the word "fizz".
- **Thread B:** calls `buzz()` that should output the word "buzz".
- **Thread C:** calls `fizzbuzz()` that should output the word "fizzbuzz".
- **Thread D:** calls `number()` that should only output the integers.

Modify the given class to output the series `[1, 2, "fizz", 4, "buzz", ...]` where the `ith` token (**1-indexed**) of the series is:

- "fizzbuzz" if `i` is divisible by 3 and 5,
- "fizz" if `i` is divisible by 3 and not 5,
- "buzz" if `i` is divisible by 5 and not 3, or
- `i` if `i` is not divisible by 3 or 5.

Implement the `FizzBuzz` class:

- `FizzBuzz(int n)` Initializes the object with the number `n` that represents the length of the sequence that should be printed.
- `void fizz(printFizz)` Calls `printFizz` to output "fizz".
- `void buzz(printBuzz)` Calls `printBuzz` to output "buzz".
- `void fizzbuzz(printFizzBuzz)` Calls `printFizzBuzz` to output "fizzbuzz".
- `void number(printNumber)` Calls `printnumber` to output the numbers.

### Example 1:

**Input:** `n = 15`

**Output:** `[1,2,"fizz",4,"buzz","fizz",7,8,"fizz","buzz",11,"fizz",13,14,"fizzbuzz"]`

### Example 2:

**Input:** `n = 5`

**Output:** `[1,2,"fizz",4,"buzz"]`

**Constraints:**

- $1 \leq n \leq 50$

**1200. Minimum Absolute Difference****Easy**

172161Add to ListShare

Given an array of **distinct** integers `arr`, find all pairs of elements with the minimum absolute difference of any two elements.

Return a list of pairs in ascending order (with respect to pairs), each pair `[a, b]` follows

- `a, b` are from `arr`
- $a < b$
- $b - a$  equals to the minimum absolute difference of any two elements in `arr`

**Example 1:****Input:** `arr = [4,2,1,3]`**Output:** `[[1,2],[2,3],[3,4]]`

**Explanation:** The minimum absolute difference is 1. List all pairs with difference equal to 1 in ascending order.

**Example 2:****Input:** `arr = [1,3,6,10,15]`**Output:** `[[1,3]]`**Example 3:****Input:** `arr = [3,8,-10,23,19,-4,-14,27]`**Output:** `[[ -14, -10], [19, 23], [23, 27]]`**Constraints:**

- $2 \leq \text{arr.length} \leq 10^5$
- $-10^6 \leq \text{arr}[i] \leq 10^6$

## 1201. Ugly Number III

Medium

918438Add to ListShare

An **ugly number** is a positive integer that is divisible by `a`, `b`, or `c`.

Given four integers `n`, `a`, `b`, and `c`, return the `nth` **ugly number**.

**Example 1:**

**Input:** `n = 3, a = 2, b = 3, c = 5`

**Output:** 4

**Explanation:** The ugly numbers are 2, 3, 4, 5, 6, 8, 9, 10... The 3<sup>rd</sup> is 4.

**Example 2:**

**Input:** `n = 4, a = 2, b = 3, c = 4`

**Output:** 6

**Explanation:** The ugly numbers are 2, 3, 4, 6, 8, 9, 10, 12... The 4<sup>th</sup> is 6.

**Example 3:**

**Input:** `n = 5, a = 2, b = 11, c = 13`

**Output:** 10

**Explanation:** The ugly numbers are 2, 4, 6, 8, 10, 11, 12, 13... The 5<sup>th</sup> is 10.

**Constraints:**

- $1 \leq n, a, b, c \leq 10^9$
- $1 \leq a * b * c \leq 10^{18}$
- It is guaranteed that the result will be in range  $[1, 2 * 10^9]$ .

## 1202. Smallest String With Swaps

Medium

3081104Add to ListShare

You are given a string `s`, and an array of pairs of indices in the string `pairs` where `pairs[i] = [a, b]` indicates 2 indices(0-indexed) of the string.

You can swap the characters at any pair of indices in the given `pairs` **any number of times**.

Return the lexicographically smallest string that `s` can be changed to after using the swaps.

**Example 1:**

**Input:** `s = "dcab"`, `pairs = [[0,3],[1,2]]`

**Output:** `"bacd"`

**Explanation:**

Swap `s[0]` and `s[3]`, `s = "bcad"`

Swap `s[1]` and `s[2]`, `s = "bacd"`

**Example 2:**

**Input:** `s = "dcab"`, `pairs = [[0,3],[1,2],[0,2]]`

**Output:** `"abcd"`

**Explanation:**

Swap `s[0]` and `s[3]`, `s = "bcad"`

Swap `s[0]` and `s[2]`, `s = "acbd"`

Swap `s[1]` and `s[2]`, `s = "abcd"`

**Example 3:**

**Input:** `s = "cba"`, `pairs = [[0,1],[1,2]]`

**Output:** `"abc"`

**Explanation:**

Swap `s[0]` and `s[1]`, `s = "bca"`

Swap `s[1]` and `s[2]`, `s = "bac"`

Swap `s[0]` and `s[1]`, `s = "abc"`

**Constraints:**

- $1 \leq s.length \leq 10^5$
- $0 \leq pairs.length \leq 10^5$
- $0 \leq pairs[i][0], pairs[i][1] < s.length$
- `s` only contains lower case English letters.

## 1203. Sort Items by Groups Respecting Dependencies

Hard

69294Add to ListShare

There are  $n$  items each belonging to zero or one of  $m$  groups where `group[i]` is the group that the  $i$ -th item belongs to and it's equal to  $-1$  if the  $i$ -th item belongs to no group. The items and the groups are zero indexed. A group can have no item belonging to it.

Return a sorted list of the items such that:

- The items that belong to the same group are next to each other in the sorted list.
- There are some relations between these items where `beforeItems[i]` is a list containing all the items that should come before the  $i$ -th item in the sorted array (to the left of the  $i$ -th item).

Return any solution if there is more than one solution and return an **empty list** if there is no solution.

**Example 1:**

Item	Group	Before
0	-1	
1	-1	6
2	1	5
3	0	6
4	0	3, 6
5	1	
6	0	
7	-1	

**Input:**  $n = 8$ ,  $m = 2$ , `group = [-1, -1, 1, 0, 0, 1, 0, -1]`, `beforeItems = [[], [6], [5], [6], [3, 6], [], [], []]`

**Output:** `[6, 3, 4, 1, 5, 2, 0, 7]`

**Example 2:**

**Input:**  $n = 8$ ,  $m = 2$ , `group = [-1, -1, 1, 0, 0, 1, 0, -1]`, `beforeItems = [[], [6], [5], [6], [3], [], [4], []]`

**Output:** `[]`

**Explanation:** This is the same as example 1 except that 4 needs to be before 6 in the sorted list.

### Constraints:

- $1 \leq m \leq n \leq 3 * 10^4$
- `group.length == beforeItems.length == n`
- $-1 \leq group[i] \leq m - 1$
- $0 \leq beforeItems[i].length \leq n - 1$
- $0 \leq beforeItems[i][j] \leq n - 1$
- $i \neq beforeItems[i][j]$
- `beforeItems[i]` does not contain duplicates elements.

## 1206. Design Skiplist

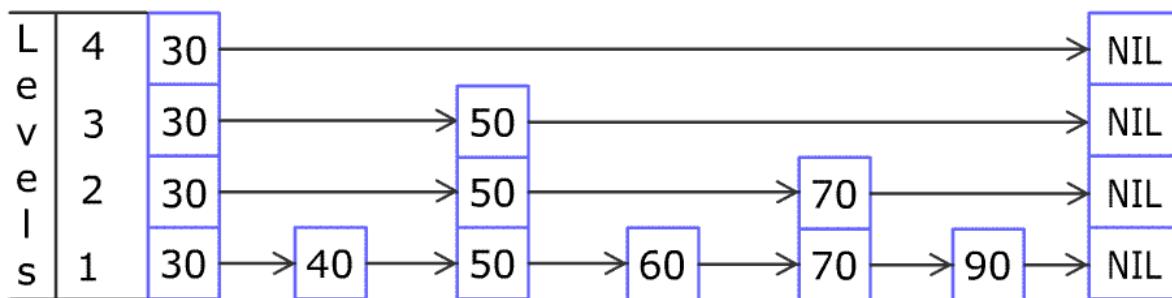
Hard

49060Add to ListShare

Design a **Skiplist** without using any built-in libraries.

A **skiplist** is a data structure that takes  $O(\log(n))$  time to add, erase and search. Comparing with treap and red-black tree which has the same function and performance, the code length of Skiplist can be comparatively short and the idea behind Skiplists is just simple linked lists.

For example, we have a Skiplist containing `[30, 40, 50, 60, 70, 90]` and we want to add `80` and `45` into it. The Skiplist works this way:



Artyom Kalinin [CC BY-SA 3.0], via [Wikimedia Commons](#)

You can see there are many layers in the Skiplist. Each layer is a sorted linked list. With the help of the top layers, add, erase and search can be faster than  $O(n)$ . It can be proven that the average time complexity for each operation is  $O(\log(n))$  and space complexity is  $O(n)$ .

See more about Skiplist: [https://en.wikipedia.org/wiki/Skip\\_list](https://en.wikipedia.org/wiki/Skip_list)

Implement the `Skiplist` class:

- `Skiplist()` Initializes the object of the skiplist.
- `bool search(int target)` Returns `true` if the integer `target` exists in the Skiplist or `false` otherwise.

- `void add(int num)` Inserts the value `num` into the SkipList.
- `bool erase(int num)` Removes the value `num` from the Skiplist and returns `true`. If `num` does not exist in the Skiplist, do nothing and return `false`. If there exist multiple `num` values, removing any one of them is fine.

Note that duplicates may exist in the Skiplist, your code needs to handle this situation.

### Example 1:

#### Input

```
["Skiplist", "add", "add", "add", "search", "add", "search", "erase", "erase",
 "search"]
[[[], [1], [2], [3], [0], [4], [1], [0], [1], [1]]]
```

#### Output

```
[null, null, null, null, false, null, true, false, true, false]
```

#### Explanation

```
Skiplist skiplist = new Skiplist();
skiplist.add(1);
skiplist.add(2);
skiplist.add(3);
skiplist.search(0); // return False
skiplist.add(4);
skiplist.search(1); // return True
skiplist.erase(0); // return False, 0 is not in skiplist.
skiplist.erase(1); // return True
skiplist.search(1); // return False, 1 has already been erased.
```

#### Constraints:

- $0 \leq num, target \leq 2 * 10^4$
- At most  $5 * 10^4$  calls will be made to `search`, `add`, and `erase`.

## 1207. Unique Number of Occurrences

Easy

198547Add to ListShare

Given an array of integers `arr`, return `true` if the number of occurrences of each value in the array is **unique**, or `false` otherwise.

**Example 1:**

**Input:** `arr = [1,2,2,1,1,3]`

**Output:** `true`

**Explanation:** The value 1 has 3 occurrences, 2 has 2 and 3 has 1. No two values have the same number of occurrences.

**Example 2:**

**Input:** `arr = [1,2]`

**Output:** `false`

**Example 3:**

**Input:** `arr = [-3,0,1,-3,1,1,1,-3,10,0]`

**Output:** `true`

**Constraints:**

- $1 \leq \text{arr.length} \leq 1000$
- $-1000 \leq \text{arr}[i] \leq 1000$

## 1208. Get Equal Substrings Within Budget

Medium

76750Add to ListShare

You are given two strings `s` and `t` of the same length and an integer `maxCost`.

You want to change `s` to `t`. Changing the  $i^{\text{th}}$  character of `s` to  $i^{\text{th}}$  character of `t` costs  $|s[i] - t[i]|$  (i.e., the absolute difference between the ASCII values of the characters).

Return the maximum length of a substring of `s` that can be changed to be the same as the corresponding substring of `t` with a cost less than or equal to `maxCost`. If there is no substring from `s` that can be changed to its corresponding substring from `t`, return 0.

**Example 1:****Input:** s = "abcd", t = "bcdf", maxCost = 3**Output:** 3**Explanation:** "abc" of s can change to "bcd".

That costs 3, so the maximum length is 3.

**Example 2:****Input:** s = "abcd", t = "cdef", maxCost = 3**Output:** 1**Explanation:** Each character in s costs 2 to change to character in t, so the maximum length is 1.**Example 3:****Input:** s = "abcd", t = "acde", maxCost = 0**Output:** 1**Explanation:** You cannot make any change, so the maximum length is 1.**Constraints:**

- $1 \leq s.length \leq 10^5$
- $t.length == s.length$
- $0 \leq \text{maxCost} \leq 10^6$
- s and t consist of only lowercase English letters.

**1209. Remove All Adjacent Duplicates in String II****Medium**

442884Add to ListShare

You are given a string `s` and an integer `k`, a `k` **duplicate removal** consists of choosing `k` adjacent and equal letters from `s` and removing them, causing the left and the right side of the deleted substring to concatenate together.

We repeatedly make `k` **duplicate removals** on `s` until we no longer can.

Return *the final string after all such duplicate removals have been made*. It is guaranteed that the answer is **unique**.

**Example 1:****Input:** s = "abcd", k = 2**Output:** "abcd"**Explanation:** There's nothing to delete.**Example 2:****Input:** s = "deeedbbcccbdaa", k = 3**Output:** "aa"**Explanation:**

First delete "eee" and "ccc", get "ddbbbdaa"

Then delete "bbb", get "dddaa"

Finally delete "ddd", get "aa"

**Example 3:****Input:** s = "pbabcggttciiippooaais", k = 2**Output:** "ps"**Constraints:**

- $1 \leq s.length \leq 10^5$
- $2 \leq k \leq 10^4$
- s only contains lowercase English letters.

**1210. Minimum Moves to Reach Target with Rotations****Hard**

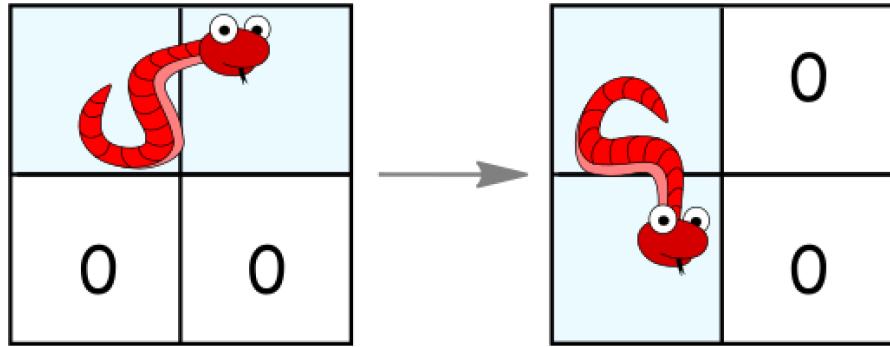
22262Add to ListShare

In an  $n \times n$  grid, there is a snake that spans 2 cells and starts moving from the top left corner at  $(0, 0)$  and  $(0, 1)$ . The grid has empty cells represented by zeros and blocked cells represented by ones. The snake wants to reach the lower right corner at  $(n-1, n-2)$  and  $(n-1, n-1)$ .

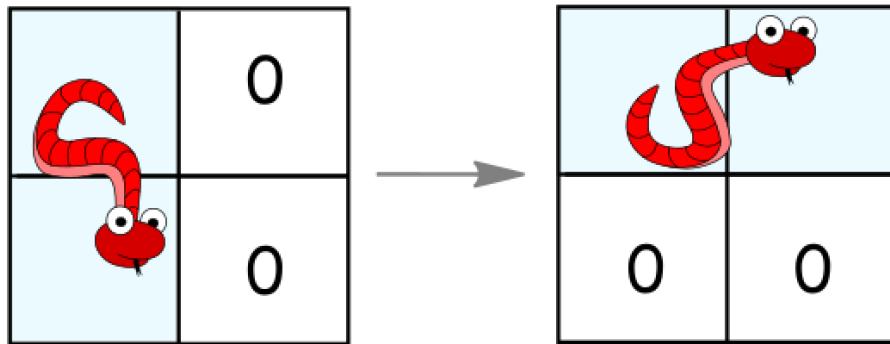
In one move the snake can:

- Move one cell to the right if there are no blocked cells there. This move keeps the horizontal/vertical position of the snake as it is.

- Move down one cell if there are no blocked cells there. This move keeps the horizontal/vertical position of the snake as it is.
- Rotate clockwise if it's in a horizontal position and the two cells under it are both empty. In that case the snake moves from  $(r, c)$  and  $(r, c+1)$  to  $(r, c)$  and  $(r+1, c)$ .



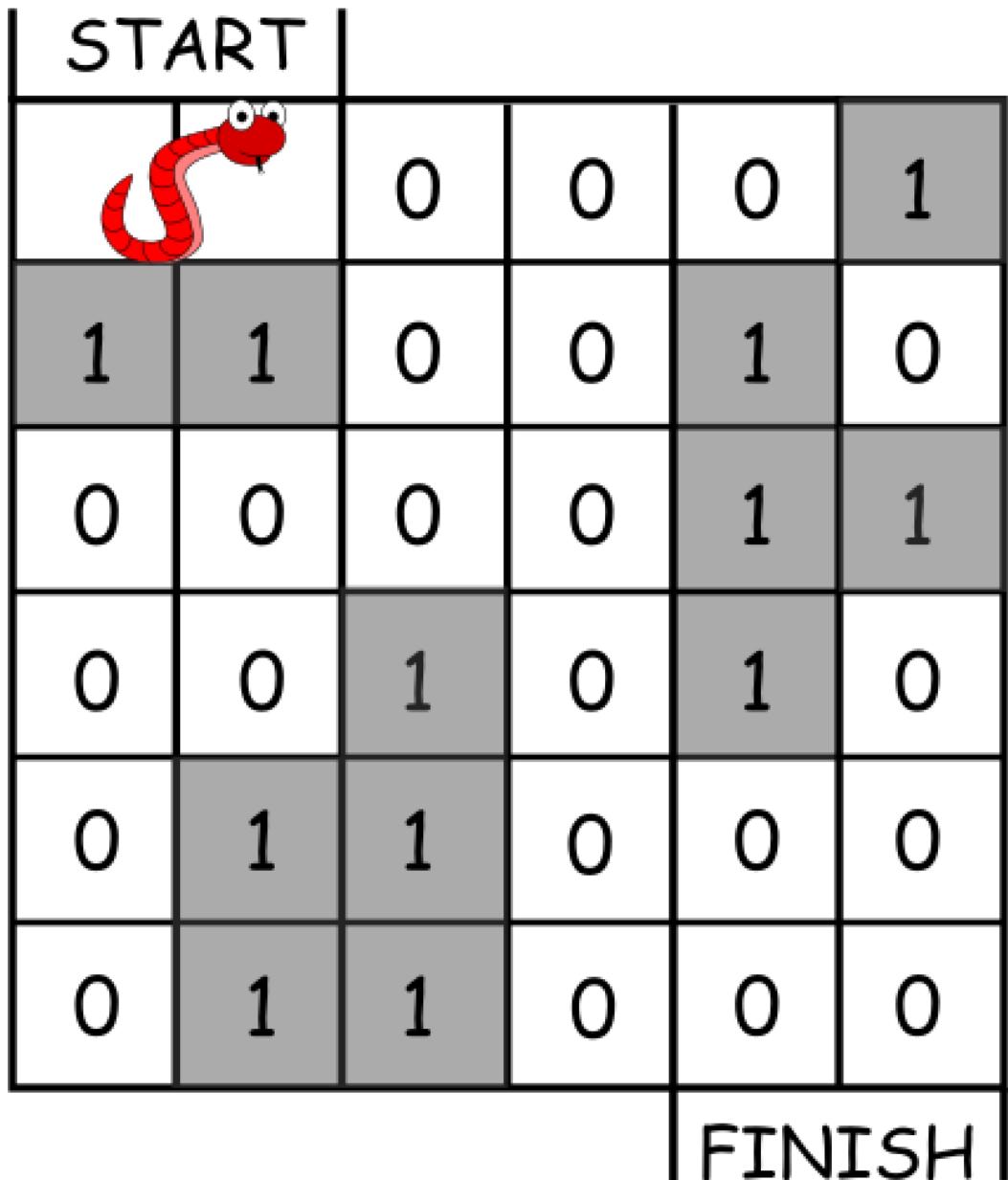
- Rotate counterclockwise if it's in a vertical position and the two cells to its right are both empty. In that case the snake moves from  $(r, c)$  and  $(r+1, c)$  to  $(r, c)$  and  $(r, c+1)$ .



Return the minimum number of moves to reach the target.

If there is no way to reach the target, return  $-1$ .

**Example 1:**



```
Input: grid = [[0,0,0,0,0,1],  
               [1,1,0,0,1,0],  
               [0,0,0,0,1,1],  
               [0,0,1,0,1,0],  
               [0,1,1,0,0,0],
```

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

**Output:** 11

**Explanation:**

One possible solution is [right, right, rotate clockwise, right, down, down, down, down, rotate counterclockwise, right, down].

**Example 2:**

```
Input: grid = [[0,0,1,1,1,1],  
              [0,0,0,0,1,1],  
              [1,1,0,0,0,1],  
              [1,1,1,0,0,1],  
              [1,1,1,0,0,1],  
              [1,1,1,0,0,0]]
```

**Output:** 9

**Constraints:**

- $2 \leq n \leq 100$
- $0 \leq \text{grid}[i][j] \leq 1$
- It is guaranteed that the snake starts at empty cells.

## 1217. Minimum Cost to Move Chips to The Same Position

Easy

1747226Add to ListShare

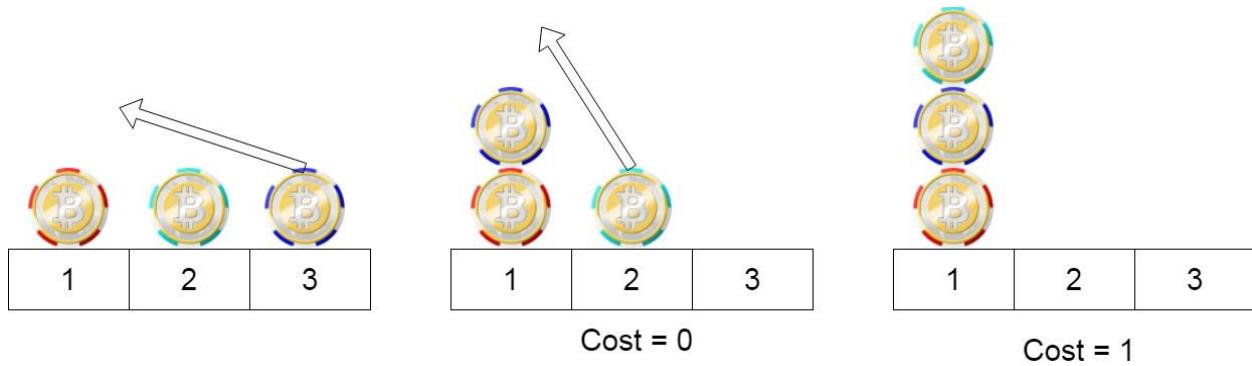
We have  $n$  chips, where the position of the  $i^{\text{th}}$  chip is  $\text{position}[i]$ .

We need to move all the chips to **the same position**. In one step, we can change the position of the  $i^{\text{th}}$  chip from  $\text{position}[i]$  to:

- $\text{position}[i] + 2$  or  $\text{position}[i] - 2$  with  $\text{cost} = 0$ .
- $\text{position}[i] + 1$  or  $\text{position}[i] - 1$  with  $\text{cost} = 1$ .

Return *the minimum cost* needed to move all the chips to the same position.

**Example 1:**



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

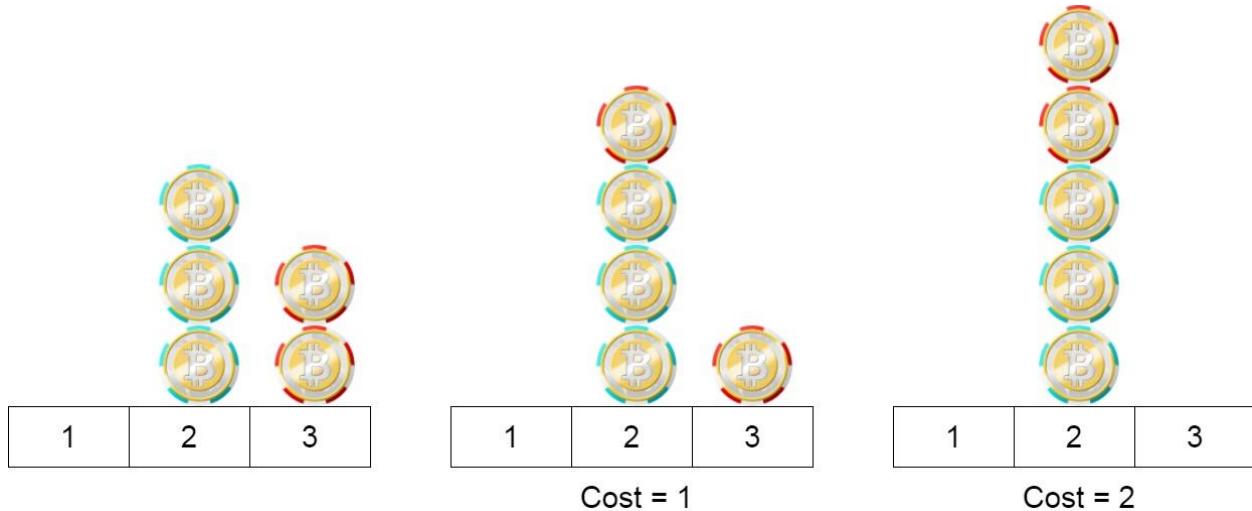
**Output:** 1

**Explanation:** First step: Move the chip at position 3 to position 1 with cost = 0.

Second step: Move the chip at position 2 to position 1 with cost = 1.

Total cost is 1.

### Example 2:



**Input:** position = [2,2,2,3,3]

**Output:** 2

**Explanation:** We can move the two chips at position 3 to position 2. Each move has cost = 1. The total cost = 2.

### Example 3:

**Input:** position = [1,1000000000]

**Output:** 1

**Constraints:**

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

**1218. Longest Arithmetic Subsequence of Given Difference****Medium**

122245Add to ListShare

Given an integer array `arr` and an integer `difference`, return the length of the longest subsequence in `arr` which is an arithmetic sequence such that the difference between adjacent elements in the subsequence equals `difference`.

A **subsequence** is a sequence that can be derived from `arr` by deleting some or no elements without changing the order of the remaining elements.

**Example 1:**

**Input:** `arr = [1,2,3,4]`, `difference = 1`

**Output:** 4

**Explanation:** The longest arithmetic subsequence is `[1,2,3,4]`.

**Example 2:**

**Input:** `arr = [1,3,5,7]`, `difference = 1`

**Output:** 1

**Explanation:** The longest arithmetic subsequence is any single element.

**Example 3:**

**Input:** `arr = [1,5,7,8,5,3,4,2,1]`, `difference = -2`

**Output:** 4

**Explanation:** The longest arithmetic subsequence is `[7,5,3,1]`.

**Constraints:**

- $1 \leq \text{arr.length} \leq 10^5$
- $-10^4 \leq \text{arr}[i], \text{difference} \leq 10^4$

**1219. Path with Maximum Gold**

**Medium**

218756Add to ListShare

In a gold mine `grid` of size  $m \times n$ , each cell in this mine has an integer representing the amount of gold in that cell, `0` if it is empty.

Return the maximum amount of gold you can collect under the conditions:

- Every time you are located in a cell you will collect all the gold in that cell.
- From your position, you can walk one step to the left, right, up, or down.
- You can't visit the same cell more than once.
- Never visit a cell with `0` gold.
- You can start and stop collecting gold from **any** position in the grid that has some gold.

**Example 1:**

**Input:** `grid = [[0,6,0],[5,8,7],[0,9,0]]`

**Output:** 24

**Explanation:**

```
[[0,6,0],  
 [5,8,7],  
 [0,9,0]]
```

Path to get the maximum gold, `9 -> 8 -> 7`.

**Example 2:**

**Input:** `grid = [[1,0,7],[2,0,6],[3,4,5],[0,3,0],[9,0,20]]`

**Output:** 28

**Explanation:**

```
[[1,0,7],  
 [2,0,6],  
 [3,4,5],  
 [0,3,0],  
 [9,0,20]]
```

Path to get the maximum gold, `1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7`.

**Constraints:**

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 15`
- `0 <= grid[i][j] <= 100`
- There are at most **25** cells containing gold.

**1220. Count Vowels Permutation****Hard**

2375160Add to ListShare

Given an integer `n`, your task is to count how many strings of length `n` can be formed under the following rules:

- Each character is a lower case vowel ('a', 'e', 'i', 'o', 'u')
- Each vowel 'a' may only be followed by an 'e'.
- Each vowel 'e' may only be followed by an 'a' or an 'i'.
- Each vowel 'i' **may not** be followed by another 'i'.
- Each vowel 'o' may only be followed by an 'i' or a 'u'.
- Each vowel 'u' may only be followed by an 'a'.

Since the answer may be too large, return it modulo  $10^9 + 7$ .

**Example 1:****Input:** `n = 1`**Output:** 5**Explanation:** All possible strings are: "a", "e", "i" , "o" and "u".**Example 2:****Input:** `n = 2`**Output:** 10**Explanation:** All possible strings are: "ae", "ea", "ei", "ia", "ie", "io", "iu", "oi", "ou" and "ua".**Example 3:****Input:** `n = 5`

**Output:** 68

**Constraints:**

- $1 \leq n \leq 2 * 10^4$

## 1221. Split a String in Balanced Strings

Easy

1975801Add to ListShare

**Balanced** strings are those that have an equal quantity of '`L`' and '`R`' characters.

Given a **balanced** string `s`, split it into some number of substrings such that:

- Each substring is balanced.

Return *the maximum number of balanced strings you can obtain*.

**Example 1:**

**Input:** `s = "RLRRRLRLRL"`

**Output:** 4

**Explanation:** `s` can be split into "RL", "RRLL", "RL", "RL", each substring contains same number of 'L' and 'R'.

**Example 2:**

**Input:** `s = "RLRRRLLRLRL"`

**Output:** 2

**Explanation:** `s` can be split into "RL", "RRRLLRLRL", each substring contains same number of 'L' and 'R'.

Note that `s` cannot be split into "RL", "RR", "RL", "LR", "LL", because the 2<sup>nd</sup> and 5<sup>th</sup> substrings are not balanced.

**Example 3:**

**Input:** `s = "LLLLRRRR"`

**Output:** 1

**Explanation:** `s` can be split into "LLLLRRRR".

**Constraints:**

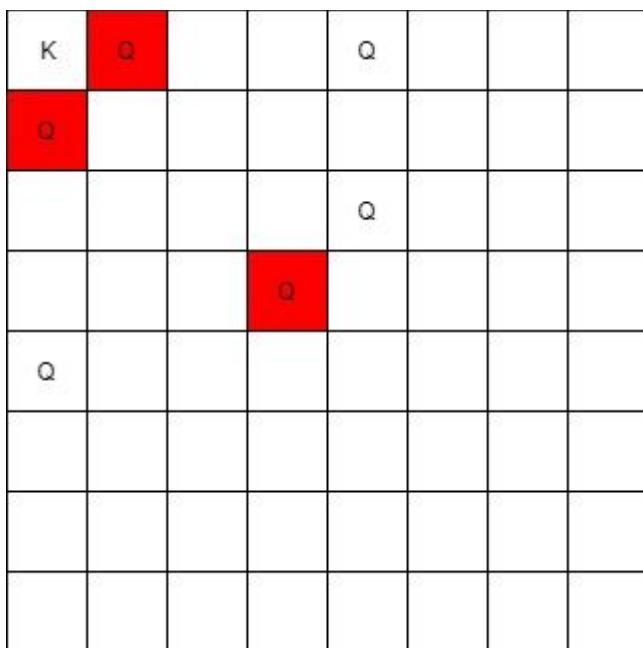
- $2 \leq s.length \leq 1000$
- $s[i]$  is either 'L' or 'R'.
- $s$  is a **balanced** string.

**1222. Queens That Can Attack the King****Medium**

753134Add to ListShare

On an **8x8** chessboard, there can be multiple Black Queens and one White King.

Given an array of integer coordinates `queens` that represents the positions of the Black Queens, and a pair of coordinates `king` that represent the position of the White King, return the coordinates of all the queens (in any order) that can attack the King.

**Example 1:**

**Input:** `queens = [[0,1],[1,0],[4,0],[0,4],[3,3],[2,4]]`, `king = [0,0]`

**Output:** `[[0,1],[1,0],[3,3]]`

**Explanation:**

The queen at `[0,1]` can attack the king cause they're in the same row.

The queen at [1,0] can attack the king cause they're in the same column.

The queen at [3,3] can attack the king cause they're in the same diagonal.

The queen at [0,4] can't attack the king cause it's blocked by the queen at [0,1].

The queen at [4,0] can't attack the king cause it's blocked by the queen at [1,0].

The queen at [2,4] can't attack the king cause it's not in the same row/column/diagonal as the king.

**Example 2:**

Q							
	Q						
		Q					
			K	Q	Q		
				Q	Q		

**Input:** queens = [[0,0],[1,1],[2,2],[3,4],[3,5],[4,4],[4,5]], king = [3,3]

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

**Example 3:**

Q	Q		Q	Q	Q	Q	Q
Q	Q	Q		Q		Q	Q
	Q	Q	Q			Q	Q
				K			Q
Q		Q	Q		Q	Q	
Q	Q	Q		Q		Q	
	Q		Q	Q			
				Q			Q

**Input:** queens =  
`[[5,6],[7,7],[2,1],[0,7],[1,6],[5,1],[3,7],[0,3],[4,0],[1,2],[6,3],[5,0],[0,4],[2,2],[1,1],[6,4],[5,4],[0,0],[2,6],[4,5],[5,2],[1,4],[7,5],[2,3],[0,5],[4,2],[1,0],[2,7],[0,1],[4,6],[6,1],[0,6],[4,3],[1,7]]`, king = [3,4]

**Output:** [[2,3],[1,4],[1,6],[3,7],[4,3],[5,4],[4,5]]

### Constraints:

- `1 <= queens.length <= 63`
- `queens[i].length == 2`
- `0 <= queens[i][j] < 8`
- `king.length == 2`
- `0 <= king[0], king[1] < 8`
- At most one piece is allowed in a cell.

## 1223. Dice Roll Simulation

Hard

780180Add to ListShare

A die simulator generates a random number from 1 to 6 for each roll. You introduced a constraint to the generator such that it cannot roll the number `i` more than `rollMax[i]` (**1-indexed**) consecutive times.

Given an array of integers `rollMax` and an integer `n`, return *the number of distinct sequences that can be obtained with exact n rolls*. Since the answer may be too large, return it **modulo**  $10^9 + 7$ .

Two sequences are considered different if at least one element differs from each other.

**Example 1:****Input:** n = 2, rollMax = [1,1,2,2,2,3]**Output:** 34

**Explanation:** There will be 2 rolls of die, if there are no constraints on the die, there are  $6 * 6 = 36$  possible combinations. In this case, looking at rollMax array, the numbers 1 and 2 appear at most once consecutively, therefore sequences (1,1) and (2,2) cannot occur, so the final answer is  $36 - 2 = 34$ .

**Example 2:****Input:** n = 2, rollMax = [1,1,1,1,1,1]**Output:** 30**Example 3:****Input:** n = 3, rollMax = [1,1,1,2,2,3]**Output:** 181**Constraints:**

- $1 \leq n \leq 5000$
- $\text{rollMax.length} == 6$
- $1 \leq \text{rollMax}[i] \leq 15$

**1224. Maximum Equal Frequency****Hard**

41749Add to ListShare

Given an array `nums` of positive integers, return the longest possible length of an array prefix of `nums`, such that it is possible to remove **exactly one** element from this prefix so that every number that has appeared in it will have the same number of occurrences.

If after removing one element there are no remaining elements, it's still considered that every appeared number has the same number of occurrences (0).

**Example 1:****Input:** nums = [2,2,1,1,5,3,3,5]**Output:** 7

**Explanation:** For the subarray `[2,2,1,1,5,3,3]` of length 7, if we remove `nums[4] = 5`, we will get `[2,2,1,1,3,3]`, so that each number will appear exactly twice.

**Example 2:**

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

**Output:** 13

**Constraints:**

- `2 <= nums.length <= 105`
- `1 <= nums[i] <= 105`

## 1226. The Dining Philosophers

Medium

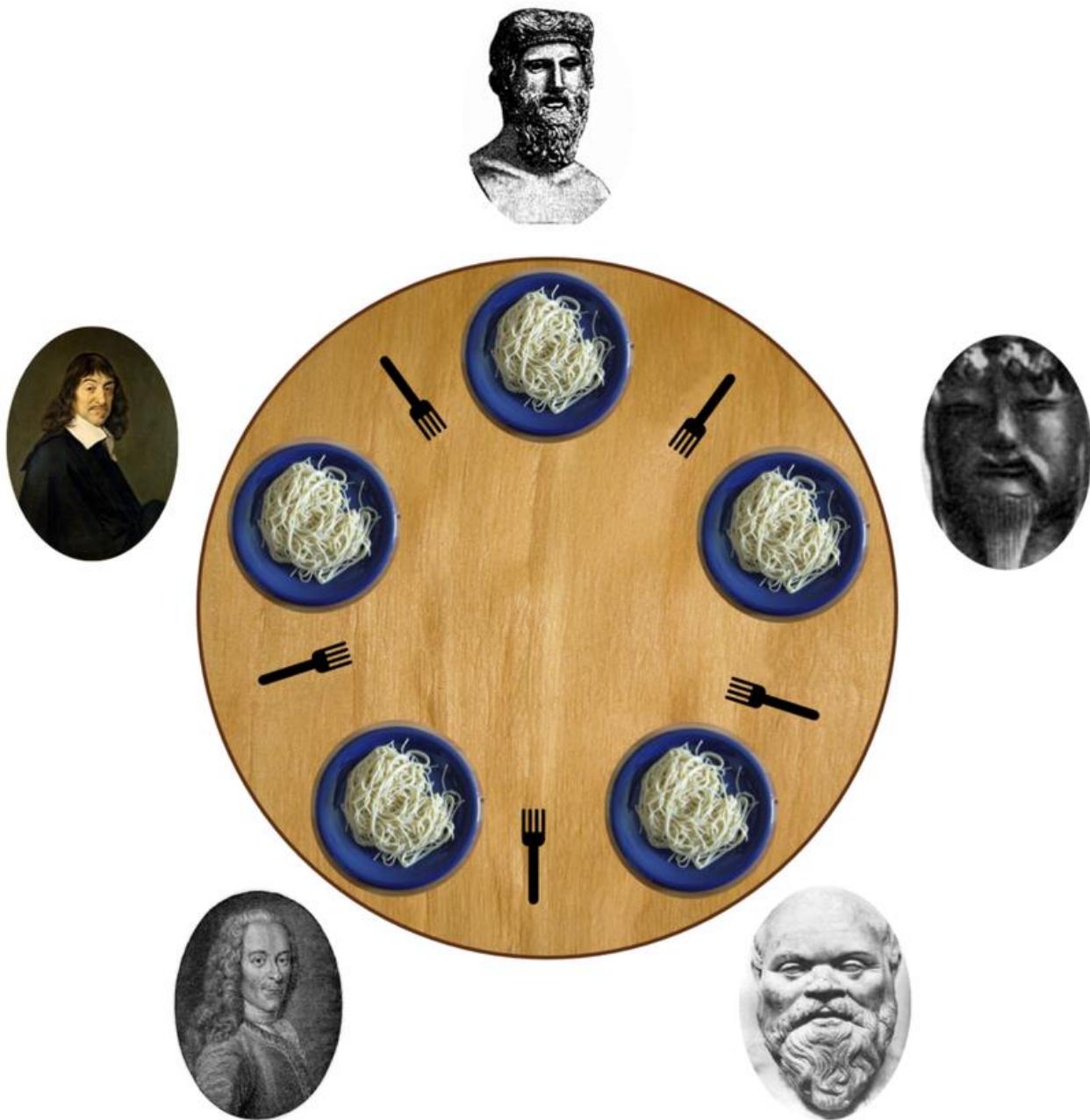
221238Add to ListShare

Five silent philosophers sit at a round table with bowls of spaghetti. Forks are placed between each pair of adjacent philosophers.

Each philosopher must alternately think and eat. However, a philosopher can only eat spaghetti when they have both left and right forks. Each fork can be held by only one philosopher and so a philosopher can use the fork only if it is not being used by another philosopher. After an individual philosopher finishes eating, they need to put down both forks so that the forks become available to others. A philosopher can take the fork on their right or the one on their left as they become available, but cannot start eating before getting both forks.

Eating is not limited by the remaining amounts of spaghetti or stomach space; an infinite supply and an infinite demand are assumed.

Design a discipline of behaviour (a concurrent algorithm) such that no philosopher will starve; *i.e.*, each can forever continue to alternate between eating and thinking, assuming that no philosopher can know when others may want to eat or think.



*The problem statement and the image above are taken from [wikipedia.org](https://en.wikipedia.org)*

The philosophers' ids are numbered from **0** to **4** in a **clockwise** order. Implement the function `void wantsToEat (philosopher, pickLeftFork, pickRightFork, eat, putLeftFork, putRightFork)` where:

- `philosopher` is the id of the philosopher who wants to eat.
- `pickLeftFork` and `pickRightFork` are functions you can call to pick the corresponding forks of that philosopher.

- `eat` is a function you can call to let the philosopher eat once he has picked both forks.
- `putLeftFork` and `putRightFork` are functions you can call to put down the corresponding forks of that philosopher.
- The philosophers are assumed to be thinking as long as they are not asking to eat (the function is not being called with their number).

Five threads, each representing a philosopher, will simultaneously use one object of your class to simulate the process. The function may be called for the same philosopher more than once, even before the last call ends.

### Example 1:

**Input:** `n = 1`

**Output:**

```
[[4,2,1],[4,1,1],[0,1,1],[2,2,1],[2,1,1],[2,0,3],[2,1,2],[2,2,2],[4,0,3],[4,1,2],[0,2,1],[4,2,2],[3,2,1],[3,1,1],[0,0,3],[0,1,2],[0,2,2],[1,2,1],[1,1,1],[3,0,3],[3,1,2],[3,2,2],[1,0,3],[1,1,2],[1,2,2]]
```

**Explanation:**

`n` is the number of times each philosopher will call the function.

The output array describes the calls you made to the functions controlling the forks and the `eat` function, its format is:

`output[i] = [a, b, c]` (three integers)

- `a` is the id of a philosopher.
- `b` specifies the fork: {1 : left, 2 : right}.
- `c` specifies the operation: {1 : pick, 2 : put, 3 : eat}.

### Constraints:

- `1 <= n <= 60`

## 1227. Airplane Seat Assignment Probability

**Medium**

430745Add to ListShare

`n` passengers board an airplane with exactly `n` seats. The first passenger has lost the ticket and picks a seat randomly. But after that, the rest of the passengers will:

- Take their own seat if it is still available, and

- Pick other seats randomly when they find their seat occupied

Return *the probability that the  $n^{\text{th}}$  person gets his own seat.*

### Example 1:

**Input:**  $n = 1$

**Output:** 1.00000

**Explanation:** The first person can only get the first seat.

### Example 2:

**Input:**  $n = 2$

**Output:** 0.50000

**Explanation:** The second person has a probability of 0.5 to get the second seat (when first person gets the first seat).

### Constraints:

- $1 \leq n \leq 10^5$

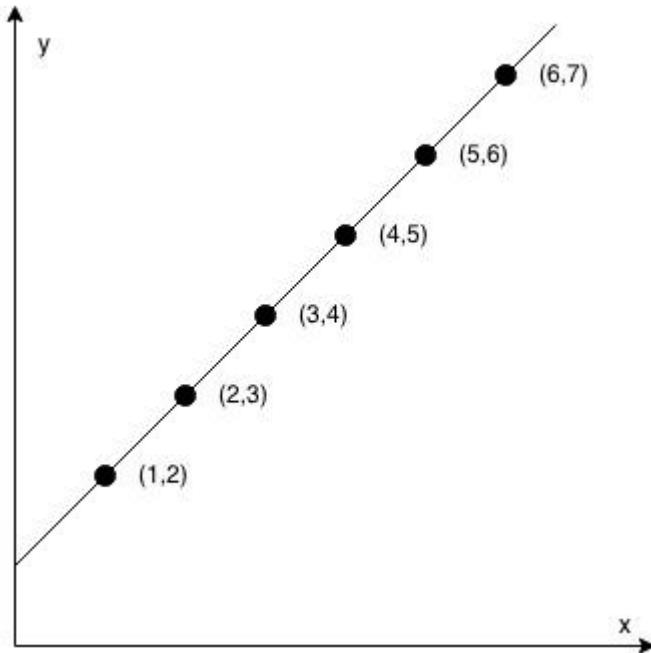
## 1232. Check If It Is a Straight Line

**Easy**

1082161Add to ListShare

You are given an array `coordinates`,  $\text{coordinates}[i] = [x, y]$ , where  $[x, y]$  represents the coordinate of a point. Check if these points make a straight line in the XY plane.

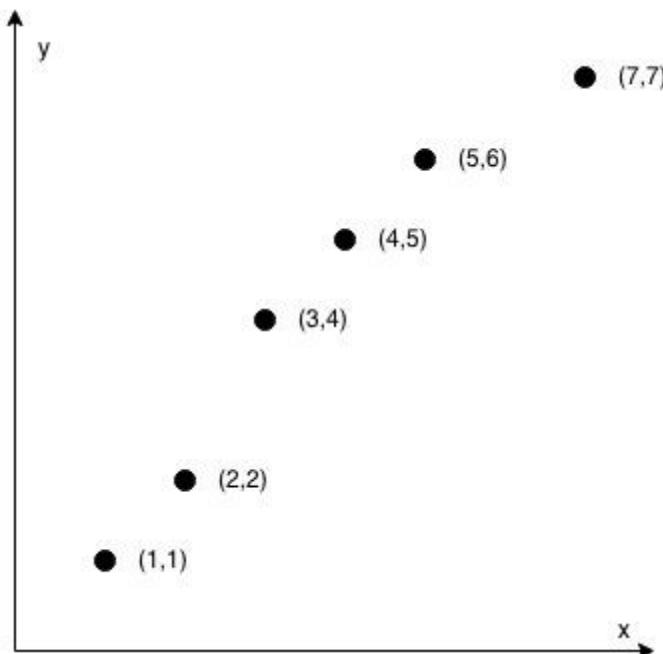
### Example 1:



**Input:** coordinates = [[1,2],[2,3],[3,4],[4,5],[5,6],[6,7]]

**Output:** true

**Example 2:**



**Input:** coordinates = [[1,1],[2,2],[3,4],[4,5],[5,6],[7,7]]

**Output:** false

**Constraints:**

- $2 \leq \text{coordinates.length} \leq 1000$
- $\text{coordinates}[i].length == 2$
- $-10^4 \leq \text{coordinates}[i][0], \text{coordinates}[i][1] \leq 10^4$
- $\text{coordinates}$  contains no duplicate point.

**1233. Remove Sub-Folders from the Filesystem****Medium**

64988Add to ListShare

Given a list of folders `folder`, return the *folders after removing all **sub-folders** in those folders*. You may return the answer in **any order**.

If a `folder[i]` is located within another `folder[j]`, it is called a **sub-folder** of it.

The format of a path is one or more concatenated strings of the form: `' / '` followed by one or more lowercase English letters.

- For example, `"/leetcode"` and `"/leetcode/problems"` are valid paths while an empty string and `"/"` are not.

**Example 1:**

**Input:** `folder = ["/a", "/a/b", "/c/d", "/c/d/e", "/c/f"]`

**Output:** `["/a", "/c/d", "/c/f"]`

**Explanation:** Folders `"/a/b"` is a subfolder of `"/a"` and `"/c/d/e"` is inside of folder `"/c/d"` in our filesystem.

**Example 2:**

**Input:** `folder = ["/a", "/a/b/c", "/a/b/d"]`

**Output:** `["/a"]`

**Explanation:** Folders `"/a/b/c"` and `"/a/b/d"` will be removed because they are subfolders of `"/a"`.

**Example 3:**

**Input:** `folder = ["/a/b/c", "/a/b/ca", "/a/b/d"]`

**Output:** `["/a/b/c", "/a/b/ca", "/a/b/d"]`

**Constraints:**

- $1 \leq \text{folder.length} \leq 4 * 10^4$
- $2 \leq \text{folder[i].length} \leq 100$
- `folder[i]` contains only lowercase letters and `'/'`.
- `folder[i]` always starts with the character `'/'`.
- Each folder name is **unique**.

**1234. Replace the Substring for Balanced String****Medium**

884167Add to ListShare

You are given a string `s` of length `n` containing only four kinds of characters: `'Q'`, `'W'`, `'E'`, and `'R'`.A string is said to be **balanced** if each of its characters appears  $n / 4$  times where `n` is the length of the string.Return *the minimum length of the substring that can be replaced with **any** other string of the same length to make `s` balanced*. If `s` is already **balanced**, return 0.**Example 1:****Input:** `s = "QWER"`**Output:** 0**Explanation:** `s` is already balanced.**Example 2:****Input:** `s = "QQWE"`**Output:** 1**Explanation:** We need to replace a 'Q' to 'R', so that "RQWE" (or "QRWE") is balanced.**Example 3:****Input:** `s = "QQQW"`**Output:** 2**Explanation:** We can replace the first "QQ" to "ER".

**Constraints:**

- $n == s.length$
- $4 \leq n \leq 10^5$
- $n$  is a multiple of 4.
- $s$  contains only 'Q', 'W', 'E', and 'R'.

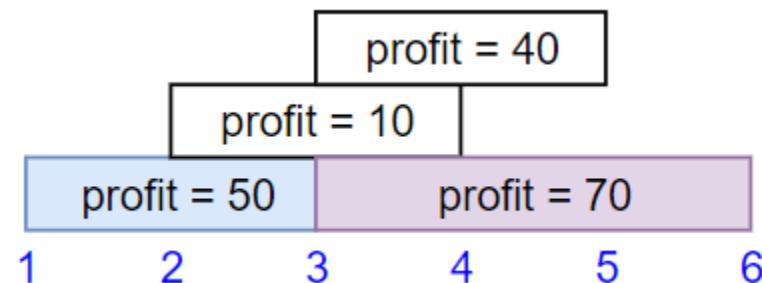
**1235. Maximum Profit in Job Scheduling****Hard**

379539Add to ListShare

We have  $n$  jobs, where every job is scheduled to be done from `startTime[i]` to `endTime[i]`, obtaining a profit of `profit[i]`.

You're given the `startTime`, `endTime` and `profit` arrays, return the maximum profit you can take such that there are no two jobs in the subset with overlapping time range.

If you choose a job that ends at time  $x$  you will be able to start another job that starts at time  $x$ .

**Example 1:**

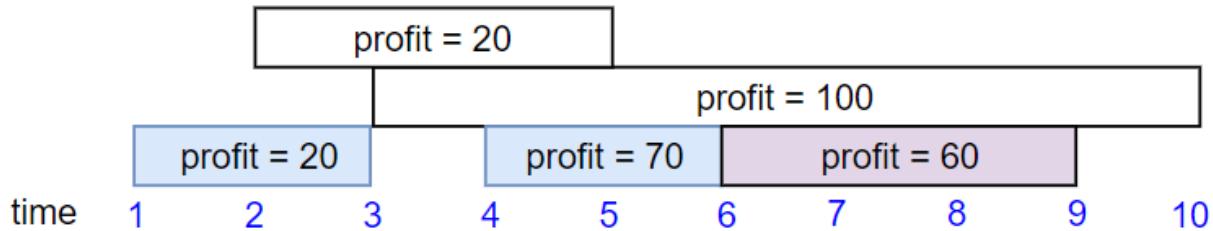
**Input:** `startTime = [1,2,3,3]`, `endTime = [3,4,5,6]`, `profit = [50,10,40,70]`

**Output:** 120

**Explanation:** The subset chosen is the first and fourth job.

Time range  $[1-3] + [3-6]$ , we get profit of  $120 = 50 + 70$ .

**Example 2:**



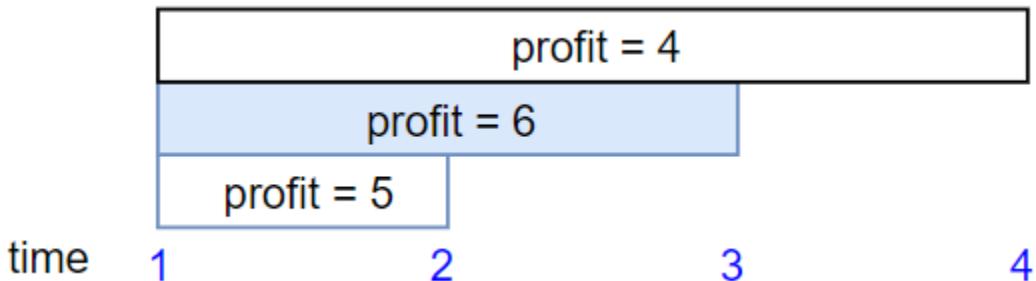
**Input:** startTime = [1,2,3,4,6], endTime = [3,5,10,6,9], profit = [20,20,100,70,60]

**Output:** 150

**Explanation:** The subset chosen is the first, fourth and fifth job.

Profit obtained  $150 = 20 + 70 + 60$ .

**Example 3:**



**Input:** startTime = [1,1,1], endTime = [2,3,4], profit = [5,6,4]

**Output:** 6

**Constraints:**

- $1 \leq \text{startTime.length} == \text{endTime.length} == \text{profit.length} \leq 5 * 10^4$
- $1 \leq \text{startTime}[i] < \text{endTime}[i] \leq 10^9$
- $1 \leq \text{profit}[i] \leq 10^4$

## 1237. Find Positive Integer Solution for a Given Equation

Medium

3911242Add to ListShare

Given a callable function `f(x, y)` **with a hidden formula** and a value `z`, reverse engineer the formula and return *all positive integer pairs x and y where  $f(x, y) == z$* . You may return the pairs in any order.

While the exact formula is hidden, the function is monotonically increasing, i.e.:

- $f(x, y) < f(x + 1, y)$
- $f(x, y) < f(x, y + 1)$

The function interface is defined like this:

```
interface CustomFunction {
public:
    // Returns some positive integer f(x, y) for two positive integers x and y based on
    // a formula.
    int f(int x, int y);
};
```

We will judge your solution as follows:

- The judge has a list of 9 hidden implementations of `CustomFunction`, along with a way to generate an **answer key** of all valid pairs for a specific  $z$ .
- The judge will receive two inputs: a `function_id` (to determine which implementation to test your code with), and the target  $z$ .
- The judge will call your `findSolution` and compare your results with the **answer key**.
- If your results match the **answer key**, your solution will be **Accepted**.

### Example 1:

**Input:** `function_id = 1, z = 5`

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

**Explanation:** The hidden formula for `function_id = 1` is  $f(x, y) = x + y$ .

The following positive integer values of  $x$  and  $y$  make  $f(x, y)$  equal to 5:

$x=1, y=4 \rightarrow f(1, 4) = 1 + 4 = 5$ .

$x=2, y=3 \rightarrow f(2, 3) = 2 + 3 = 5$ .

$x=3, y=2 \rightarrow f(3, 2) = 3 + 2 = 5$ .

$x=4, y=1 \rightarrow f(4, 1) = 4 + 1 = 5$ .

### Example 2:

**Input:** `function_id = 2, z = 5`

**Output:** `[[1,5],[5,1]]`

**Explanation:** The hidden formula for `function_id = 2` is  $f(x, y) = x * y$ .

The following positive integer values of  $x$  and  $y$  make  $f(x, y)$  equal to 5:

$x=1, y=5 \rightarrow f(1, 5) = 1 * 5 = 5$ .

$x=5, y=1 \rightarrow f(5, 1) = 5 * 1 = 5$ .

### Constraints:

- $1 \leq \text{function\_id} \leq 9$
- $1 \leq z \leq 100$
- It is guaranteed that the solutions of  $f(x, y) == z$  will be in the range  $1 \leq x, y \leq 1000$ .
- It is also guaranteed that  $f(x, y)$  will fit in 32 bit signed integer if  $1 \leq x, y \leq 1000$ .

## 1238. Circular Permutation in Binary Representation

Medium

295169Add to ListShare

Given 2 integers  $n$  and `start`. Your task is return **any** permutation  $p$  of  $(0, 1, 2, \dots, 2^n - 1)$  such that :

- $p[0] = \text{start}$
- $p[i]$  and  $p[i+1]$  differ by only one bit in their binary representation.
- $p[0]$  and  $p[2^n - 1]$  must also differ by only one bit in their binary representation.

### Example 1:

**Input:**  $n = 2$ , `start = 3`

**Output:** `[3,2,0,1]`

**Explanation:** The binary representation of the permutation is  $(11, 10, 00, 01)$ .

All the adjacent element differ by one bit. Another valid permutation is `[3,1,0,2]`

### Example 2:

**Input:**  $n = 3$ , `start = 2`

**Output:** `[2,6,7,5,4,0,1,3]`

**Explanation:** The binary representation of the permutation is  $(010, 110, 111, 101, 100, 000, 001, 011)$ .

**Constraints:**

- $1 \leq n \leq 16$
- $0 \leq \text{start} < 2^n$

**1239. Maximum Length of a Concatenated String with Unique Characters****Medium**

2043162 Add to List Share

You are given an array of strings `arr`. A string `s` is formed by the **concatenation** of a **subsequence** of `arr` that has **unique characters**.

Return *the maximum possible length of `s`*.

A **subsequence** is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements.

**Example 1:**

**Input:** `arr = ["un", "iq", "ue"]`

**Output:** 4

**Explanation:** All the valid concatenations are:

- ""
- "un"
- "iq"
- "ue"
- "uniq" ("un" + "iq")
- "ique" ("iq" + "ue")

Maximum length is 4.

**Example 2:**

**Input:** `arr = ["cha", "r", "act", "ers"]`

**Output:** 6

**Explanation:** Possible longest valid concatenations are "chaers" ("cha" + "ers") and "acters" ("act" + "ers").

**Example 3:**

**Input:** arr = ["abcdefghijklmnopqrstuvwxyz"]

**Output:** 26

**Explanation:** The only string in arr has all 26 characters.

**Constraints:**

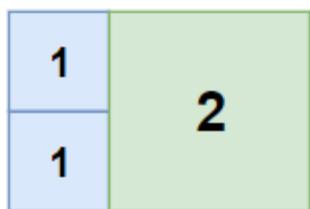
- $1 \leq \text{arr.length} \leq 16$
- $1 \leq \text{arr}[i].length \leq 26$
- $\text{arr}[i]$  contains only lowercase English letters.

**1240. Tiling a Rectangle with the Fewest Squares**

Hard

556518Add to ListShare

Given a rectangle of size  $n \times m$ , return the minimum number of integer-sided squares that tile the rectangle.

**Example 1:**

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

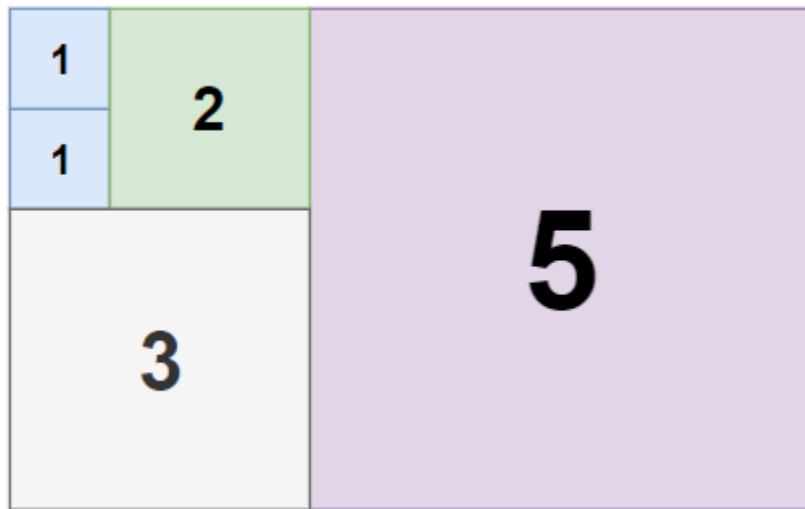
**Output:** 3

**Explanation:** 3 squares are necessary to cover the rectangle.

2 (squares of 1x1)

1 (square of 2x2)

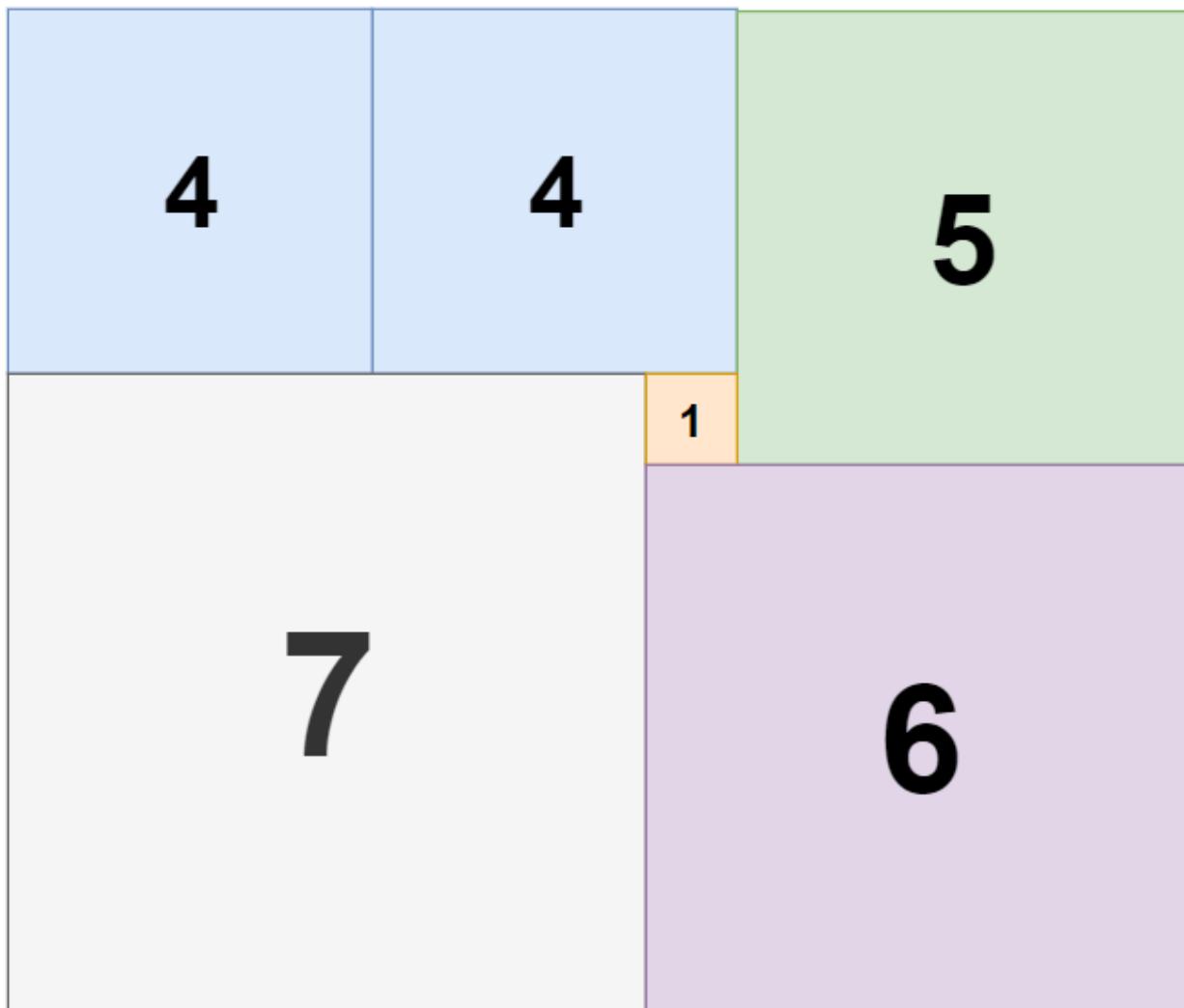
**Example 2:**



**Input:**  $n = 5, m = 8$

**Output:** 5

**Example 3:**



**Input:** n = 11, m = 13

**Output:** 6

**Constraints:**

- $1 \leq n, m \leq 13$

## 1247. Minimum Swaps to Make Strings Equal

Medium

977218Add to ListShare

You are given two strings `s1` and `s2` of equal length consisting of letters "x" and "y" only. Your task is to make these two strings equal to each other. You can swap any two characters that belong to **different** strings, which means: swap `s1[i]` and `s2[j]`.

Return the minimum number of swaps required to make `s1` and `s2` equal, or return `-1` if it is impossible to do so.

**Example 1:**

**Input:** `s1 = "xx"`, `s2 = "yy"`

**Output:** 1

**Explanation:** Swap `s1[0]` and `s2[1]`, `s1 = "yx"`, `s2 = "yx"`.

**Example 2:**

**Input:** `s1 = "xy"`, `s2 = "yx"`

**Output:** 2

**Explanation:** Swap `s1[0]` and `s2[0]`, `s1 = "yy"`, `s2 = "xx"`.

Swap `s1[0]` and `s2[1]`, `s1 = "xy"`, `s2 = "xy"`.

Note that you cannot swap `s1[0]` and `s1[1]` to make `s1` equal to "yx", cause we can only swap chars in different strings.

**Example 3:**

**Input:** `s1 = "xx"`, `s2 = "xy"`

**Output:** -1

**Constraints:**

- `1 <= s1.length, s2.length <= 1000`
- `s1, s2` only contain 'x' or 'y'.

## 1248. Count Number of Nice Subarrays

Medium

217951Add to ListShare

Given an array of integers `nums` and an integer `k`. A continuous subarray is called **nice** if there are `k` odd numbers on it.

Return the number of **nice** sub-arrays.

**Example 1:****Input:** nums = [1,1,2,1,1], k = 3**Output:** 2**Explanation:** The only sub-arrays with 3 odd numbers are [1,1,2,1] and [1,2,1,1].**Example 2:****Input:** nums = [2,4,6], k = 1**Output:** 0**Explanation:** There is no odd numbers in the array.**Example 3:****Input:** nums = [2,2,2,1,2,2,1,2,2,2], k = 2**Output:** 16**Constraints:**

- $1 \leq \text{nums.length} \leq 50000$
- $1 \leq \text{nums}[i] \leq 10^5$
- $1 \leq k \leq \text{nums.length}$

**1249. Minimum Remove to Make Valid Parentheses****Medium**

529499Add to ListShare

Given a string  $s$  of ' ( ' , ' ) ' and lowercase English characters.Your task is to remove the minimum number of parentheses ( ' ( ' or ' ) ' , in any positions ) so that the resulting *parentheses string* is valid and return **any** valid string.Formally, a *parentheses string* is valid if and only if:

- It is the empty string, contains only lowercase characters, or
- It can be written as  $\text{AB}$  ( $\text{A}$  concatenated with  $\text{B}$ ), where  $\text{A}$  and  $\text{B}$  are valid strings, or
- It can be written as  $(\text{A})$ , where  $\text{A}$  is a valid string.

**Example 1:****Input:** s = "lee(t(c)o)de"

**Output:** "lee(t(c)o)de"

**Explanation:** "lee(t(co)de)" , "lee(t(c)ode)" would also be accepted.

**Example 2:**

**Input:** s = "a)b(c)d"

**Output:** "ab(c)d"

**Example 3:**

**Input:** s = "))()("

**Output:** ""

**Explanation:** An empty string is also valid.

**Constraints:**

- $1 \leq s.length \leq 10^5$
- $s[i]$  is either '(', ')', or lowercase English letter.

## 1250. Check If It Is a Good Array

**Hard**

319315Add to ListShare

Given an array `nums` of positive integers. Your task is to select some subset of `nums`, multiply each element by an integer and add all these numbers. The array is said to be **good** if you can obtain a sum of `1` from the array by any possible subset and multiplicand.

Return `True` if the array is **good** otherwise return `False`.

**Example 1:**

**Input:** `nums` = [12,5,7,23]

**Output:** `true`

**Explanation:** Pick numbers 5 and 7.

$5*3 + 7*(-2) = 1$

**Example 2:**

**Input:** `nums` = [29,6,10]

**Output:** true

**Explanation:** Pick numbers 29, 6 and 10.

$$29*1 + 6*(-3) + 10*(-1) = 1$$

**Example 3:**

**Input:** nums = [3,6]

**Output:** false

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^5$
- $1 \leq \text{nums}[i] \leq 10^9$

## 1252. Cells with Odd Values in a Matrix

Easy

8601235Add to ListShare

There is an  $m \times n$  matrix that is initialized to all 0's. There is also a 2D array `indices` where each `indices[i] = [ri, ci]` represents a **0-indexed location** to perform some increment operations on the matrix.

For each location `indices[i]`, do **both** of the following:

1. Increment **all** the cells on row `ri`.
2. Increment **all** the cells on column `ci`.

Given  $m$ ,  $n$ , and `indices`, return *the number of odd-valued cells* in the matrix after applying the increment to all locations in `indices`.

**Example 1:**

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 3 & 1 \\ 1 & 3 & 1 \end{bmatrix}$$

**Input:** m = 2, n = 3, indices = [[0,1],[1,1]]

**Output:** 6

**Explanation:** Initial matrix =  $[[0,0,0],[0,0,0]]$ .

After applying first increment it becomes  $[[1,2,1],[0,1,0]]$ .

The final matrix is  $[[1,3,1],[1,3,1]]$ , which contains 6 odd numbers.

**Example 2:**

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

**Input:**  $m = 2, n = 2, \text{indices} = [[1,1],[0,0]]$

**Output:** 0

**Explanation:** Final matrix =  $[[2,2],[2,2]]$ . There are no odd numbers in the final matrix.

**Constraints:**

- $1 \leq m, n \leq 50$
- $1 \leq \text{indices.length} \leq 100$
- $0 \leq r_i < m$
- $0 \leq c_i < n$

## 1253. Reconstruct a 2-Row Binary Matrix

Medium

36226Add to ListShare

Given the following details of a matrix with  $n$  columns and 2 rows :

- The matrix is a binary matrix, which means each element in the matrix can be 0 or 1.
- The sum of elements of the 0-th(upper) row is given as `upper`.
- The sum of elements of the 1-st(lower) row is given as `lower`.
- The sum of elements in the  $i$ -th column(0-indexed) is `colsum[i]`, where `colsum` is given as an integer array with length `n`.

Your task is to reconstruct the matrix with `upper`, `lower` and `colsum`.

Return it as a 2-D integer array.

If there are more than one valid solution, any of them will be accepted.

If no valid solution exists, return an empty 2-D array.

**Example 1:**

**Input:** upper = 2, lower = 1, colsum = [1,1,1]

**Output:** [[1,1,0],[0,0,1]]

**Explanation:** [[1,0,1],[0,1,0]], and [[0,1,1],[1,0,0]] are also correct answers.

**Example 2:**

**Input:** upper = 2, lower = 3, colsum = [2,2,1,1]

**Output:** []

**Example 3:**

**Input:** upper = 5, lower = 5, colsum = [2,1,2,0,1,0,1,2,0,1]

**Output:** [[1,1,1,0,1,0,0,1,0,0],[1,0,1,0,0,0,1,1,0,1]]

**Constraints:**

- $1 \leq \text{colsum.length} \leq 10^5$
- $0 \leq \text{upper}, \text{lower} \leq \text{colsum.length}$
- $0 \leq \text{colsum}[i] \leq 2$

## 1254. Number of Closed Islands

Medium

244355Add to ListShare

Given a 2D grid consists of 0s (land) and 1s (water). An *island* is a maximal 4-directionally connected group of 0s and a *closed island* is an island **totally** (all left, top, right, bottom) surrounded by 1s.

Return the number of *closed islands*.

**Example 1:**

1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	0
1	0	1	0	1	1	1	0
1	0	0	0	0	1	0	1
1	1	1	1	1	1	1	0

**Input:** grid =  
`[[1,1,1,1,1,1,1,0],[1,0,0,0,0,1,1,0],[1,0,1,0,1,1,1,0],[1,0,0,0,0,1,0,1],[1,1,1,1,1,1,1,0]]`

**Output:** 2

**Explanation:**

Islands in gray are closed because they are completely surrounded by water (group of 1s).

**Example 2:**

0	0	1	0	0
0	1	0	1	0
0	1	1	1	0

**Input:** grid = `[[0,0,1,0,0],[0,1,0,1,0],[0,1,1,1,0]]`

**Output:** 1

**Example 3:**

**Input:** grid = `[[1,1,1,1,1,1,1],[1,0,0,0,0,0,1],[1,0,1,1,1,0,1],`

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

**Output:** 2

### Constraints:

- $1 \leq \text{grid.length}, \text{grid[0].length} \leq 100$
- $0 \leq \text{grid[i][j]} \leq 1$

## 1255. Maximum Score Words Formed by Letters

Hard

82945Add to ListShare

Given a list of `words`, list of single `letters` (might be repeating) and `score` of every character.

Return the maximum score of **any** valid set of words formed by using the given letters (`words[i]` cannot be used two or more times).

It is not necessary to use all characters in `letters` and each letter can only be used once. Score of letters '`'a'`', '`'b'`', '`'c'`', ..., '`'z'`' is given by `score[0]`, `score[1]`, ..., `score[25]` respectively.

### Example 1:

**Input:** `words = ["dog", "cat", "dad", "good"]`, `letters = ["a", "a", "c", "d", "d", "d", "g", "o", "o"]`, `score = [1,0,9,5,0,0,3,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0]`

**Output:** 23

### Explanation:

Score `a=1, c=9, d=5, g=3, o=2`

Given letters, we can form the words "dad" (5+1+5) and "good" (3+2+2+5) with a score of 23.

Words "dad" and "dog" only get a score of 21.

### Example 2:

**Input:** words = ["xxxz", "ax", "bx", "cx"], letters = ["z", "a", "b", "c", "x", "x", "x"], score = [4,4,4,0,5,0,10]

**Output:** 27

### Explanation:

Score a=4, b=4, c=4, x=5, z=10

Given letters, we can form the words "ax" (4+5), "bx" (4+5) and "cx" (4+5) with a score of 27.

Word "xxxx" only get a score of 25.

### Example 3:

**Input:** words = ["leetcode"], letters = ["l", "e", "t", "c", "o", "d"], score = [0,0,1,1,1,0,0,0,0,0,1,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0]

Output: 0

### Explanation:

Letter "e" can only be used once.

## Constraints:

- `1 <= words.length <= 14`
  - `1 <= words[i].length <= 15`
  - `1 <= letters.length <= 100`
  - `letters[i].length == 1`
  - `score.length == 26`
  - `0 <= score[i] <= 10`
  - `words[i], letters[i] contains only lower case English letters.`

## 1260. Shift 2D Grid

## Easy

1483311Add to ListShare

Given a 2D grid of size  $m \times n$  and an integer  $k$ . You need to shift the grid  $k$  times.

In one shift operation:

- Element at `grid[i][j]` moves to `grid[i][j + 1]`.
  - Element at `grid[i][n - 1]` moves to `grid[i + 1][0]`.
  - Element at `grid[m - 1][n - 1]` moves to `grid[0][0]`.

Return the *2D grid* after applying shift operation  $k$  times.

**Example 1:**

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow \begin{bmatrix} 9 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

**Input:** grid = [[1,2,3],[4,5,6],[7,8,9]], k = 1

**Output:** [[9,1,2],[3,4,5],[6,7,8]]

**Example 2:**

$$\begin{bmatrix} 3 & 8 & 1 & 9 \\ 19 & 7 & 2 & 5 \\ 4 & 6 & 11 & 10 \\ 12 & 0 & 21 & 13 \end{bmatrix} \rightarrow \begin{bmatrix} 13 & 3 & 8 & 1 \\ 9 & 19 & 7 & 2 \\ 5 & 4 & 6 & 11 \\ 10 & 12 & 0 & 21 \end{bmatrix} \rightarrow \begin{bmatrix} 21 & 13 & 3 & 8 \\ 1 & 9 & 19 & 7 \\ 2 & 5 & 4 & 6 \\ 11 & 10 & 12 & 0 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 0 & 21 & 13 & 3 \\ 8 & 1 & 9 & 19 \\ 7 & 2 & 5 & 4 \\ 6 & 11 & 10 & 12 \end{bmatrix} \rightarrow \begin{bmatrix} 12 & 0 & 21 & 13 \\ 3 & 8 & 1 & 9 \\ 19 & 7 & 2 & 5 \\ 4 & 6 & 11 & 10 \end{bmatrix}$$

**Input:** grid = [[3,8,1,9],[19,7,2,5],[4,6,11,10],[12,0,21,13]], k = 4

**Output:** [[12,0,21,13],[3,8,1,9],[19,7,2,5],[4,6,11,10]]

**Example 3:**

**Input:** grid = [[1,2,3],[4,5,6],[7,8,9]], k = 9

**Output:** [[1,2,3],[4,5,6],[7,8,9]]

**Constraints:**

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m <= 50`
- `1 <= n <= 50`
- `-1000 <= grid[i][j] <= 1000`
- `0 <= k <= 100`

**1261. Find Elements in a Contaminated Binary Tree****Medium**

74687 Add to List Share

Given a binary tree with the following rules:

1. `root.val == 0`
2. If `treeNode.val == x` and `treeNode.left != null`, then `treeNode.left.val == 2 * x + 1`
3. If `treeNode.val == x` and `treeNode.right != null`, then `treeNode.right.val == 2 * x + 2`

Now the binary tree is contaminated, which means all `treeNode.val` have been changed to `-1`.Implement the `FindElements` class:

- `FindElements(TreeNode* root)` Initializes the object with a contaminated binary tree and recovers it.
- `bool find(int target)` Returns `true` if the `target` value exists in the recovered binary tree.

**Example 1:****Input**

["FindElements", "find", "find"]

[[[-1, null, -1]], [1], [2]]

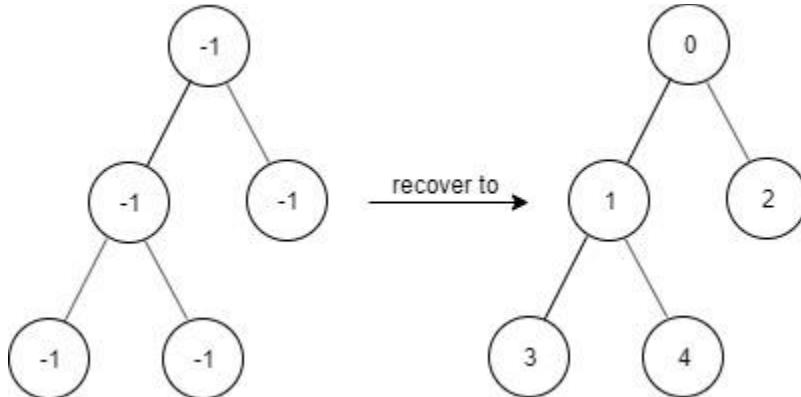
**Output**

[null, false, true]

**Explanation**

```
FindElements findElements = new FindElements([-1,null,-1]);
findElements.find(1); // return False
findElements.find(2); // return True
```

**Example 2:**



**Input**

```
["FindElements","find","find","find"]
[[[-1,-1,-1,-1,-1]],[1],[3],[5]]
```

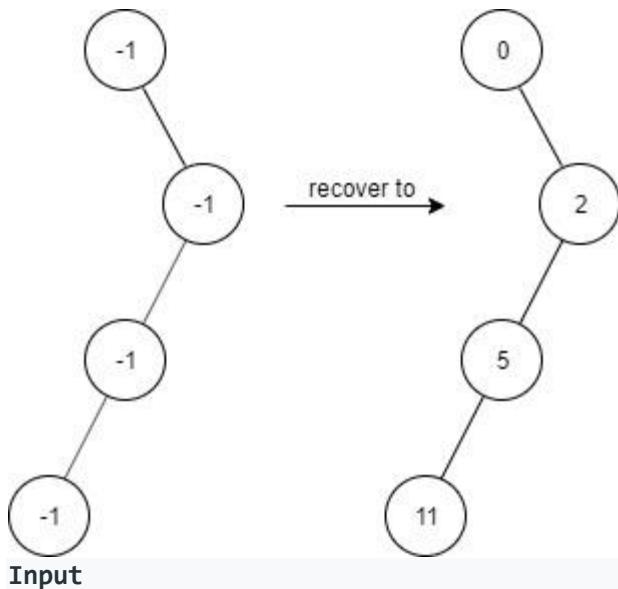
**Output**

[null, true, true, false]

**Explanation**

```
FindElements findElements = new FindElements([-1,-1,-1,-1,-1]);
findElements.find(1); // return True
findElements.find(3); // return True
findElements.find(5); // return False
```

**Example 3:**

**Input**

```

["FindElements", "find", "find", "find", "find"]
[[[-1, null, -1, -1, null, -1]], [2], [3], [4], [5]]
  
```

**Output**

```
[null, true, false, false, true]
```

**Explanation**

```

FindElements findElements = new FindElements([-1, null, -1, -1, null, -1]);
findElements.find(2); // return True
findElements.find(3); // return False
findElements.find(4); // return False
findElements.find(5); // return True
  
```

**Constraints:**

- `TreeNode.val == -1`
- The height of the binary tree is less than or equal to `20`
- The total number of nodes is between `[1, 104]`
- Total calls of `find()` is between `[1, 104]`
- `0 <= target <= 106`

**1262. Greatest Sum Divisible by Three****Medium**

138134Add to ListShare

Given an integer array `nums`, return *the maximum possible sum of elements of the array such that it is divisible by three*.

**Example 1:**

**Input:** `nums = [3,6,5,1,8]`

**Output:** 18

**Explanation:** Pick numbers 3, 6, 1 and 8 their sum is 18 (maximum sum divisible by 3).

**Example 2:**

**Input:** `nums = [4]`

**Output:** 0

**Explanation:** Since 4 is not divisible by 3, do not pick any number.

**Example 3:**

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

**Output:** 12

**Explanation:** Pick numbers 1, 3, 4 and 4 their sum is 12 (maximum sum divisible by 3).

**Constraints:**

- `1 <= nums.length <= 4 * 10^4`
- `1 <= nums[i] <= 10^4`

## 1263. Minimum Moves to Move a Box to Their Target Location

**Hard**

70050Add to ListShare

A storekeeper is a game in which the player pushes boxes around in a warehouse trying to get them to target locations.

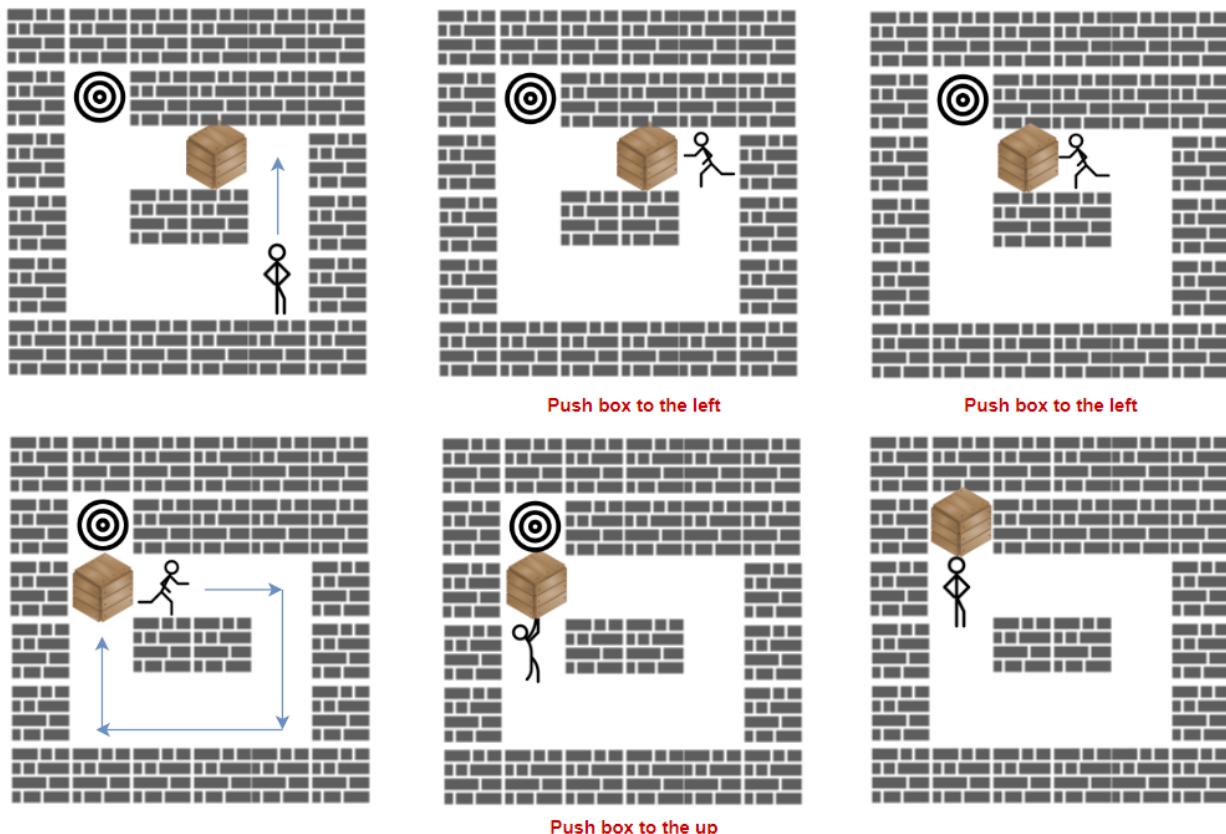
The game is represented by an `m x n` grid of characters `grid` where each element is a wall, floor, or box.

Your task is to move the box '`'B'`' to the target position '`'T'`' under the following rules:

- The character '`S`' represents the player. The player can move up, down, left, right in `grid` if it is a floor (empty cell).
- The character '`.`' represents the floor which means a free cell to walk.
- The character '`#`' represents the wall which means an obstacle (impossible to walk there).
- There is only one box '`B`' and one target cell '`T`' in the `grid`.
- The box can be moved to an adjacent free cell by standing next to the box and then moving in the direction of the box. This is a **push**.
- The player cannot walk through the box.

Return the *minimum number of pushes* to move the box to the target. If there is no way to reach the target, return `-1`.

**Example 1:**



```
Input: grid = [[ "#", "#", "#", "#", "#", "#"],  
                [ "#", "T", "#", "#", "#", "#"],  
                [ "#", ".", ".", "B", ".", "#"],  
                [ "#", ".", "#", "#", ".", "#"],  
                [ "#", ".", ".", ".", "S", "#"],
```

```
[ "#", "#", "#", "#", "#", "#"] ]
```

**Output:** 3

**Explanation:** We return only the number of times the box is pushed.

**Example 2:**

```
Input: grid = [[ "#", "#", "#", "#", "#", "#"],  
               [ "#", "T", "#", "#", "#", "#"],  
               [ "#", ".", ".", "B", ".", "#"],  
               [ "#", "#", "#", "#", ".", "#"],  
               [ "#", ".", ".", ".", "S", "#"],  
               [ "#", "#", "#", "#", "#", "#"] ]
```

**Output:** -1

**Example 3:**

```
Input: grid = [[ "#", "#", "#", "#", "#", "#"],  
               [ "#", "T", ".", ".", "#", "#"],  
               [ "#", ".", "#", "B", ".", "#"],  
               [ "#", ".", ".", ".", ".", "#"],  
               [ "#", ".", ".", ".", "S", "#"],  
               [ "#", "#", "#", "#", "#", "#"] ]
```

**Output:** 5

**Explanation:** push the box down, left, left, up and up.

**Constraints:**

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 20`
- `grid` contains only characters `'.'`, `'#'`, `'S'`, `'T'`, or `'B'`.
- There is only one character `'S'`, `'B'`, and `'T'` in the `grid`.

## 1266. Minimum Time Visiting All Points

**Easy**

1308168Add to ListShare

On a 2D plane, there are `n` points with integer coordinates `points[i] = [xi, yi]`.Return the **minimum time** in seconds to visit all the points in the order given by `points`.

You can move according to these rules:

- In `1` second, you can either:
  - move vertically by one unit,
  - move horizontally by one unit, or
  - move diagonally `sqrt(2)` units (in other words, move one unit vertically then one unit horizontally in `1` second).
- You have to visit the points in the same order as they appear in the array.
- You are allowed to pass through points that appear later in the order, but these do not count as visits.

**Example 1:**



**Input:** points = [[1,1],[3,4],[-1,0]]

**Output:** 7

**Explanation:** One optimal path is [1,1] -> [2,2] -> [3,3] -> [3,4] -> [2,3] -> [1,2] -> [0,1] -> [-1,0]

Time from [1,1] to [3,4] = 3 seconds

Time from [3,4] to [-1,0] = 4 seconds

Total time = 7 seconds

### Example 2:

**Input:** points = [[3,2],[-2,2]]

**Output:** 5

**Constraints:**

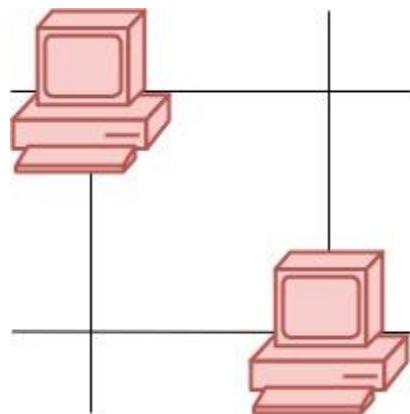
- `points.length == n`
- `1 <= n <= 100`
- `points[i].length == 2`
- `-1000 <= points[i][0], points[i][1] <= 1000`

**1267. Count Servers that Communicate****Medium**

101575Add to ListShare

You are given a map of a server center, represented as a `m * n` integer matrix `grid`, where 1 means that on that cell there is a server and 0 means that it is no server. Two servers are said to communicate if they are on the same row or on the same column.

Return the number of servers that communicate with any other server.

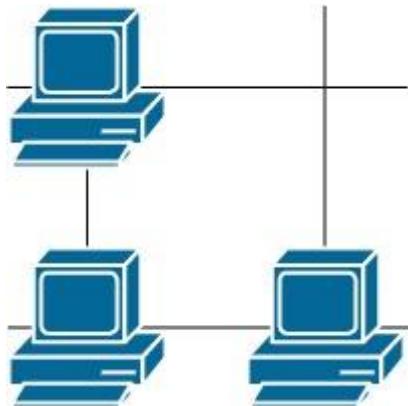
**Example 1:**

**Input:** `grid = [[1,0],[0,1]]`

**Output:** 0

**Explanation:** No servers can communicate with others.

**Example 2:**

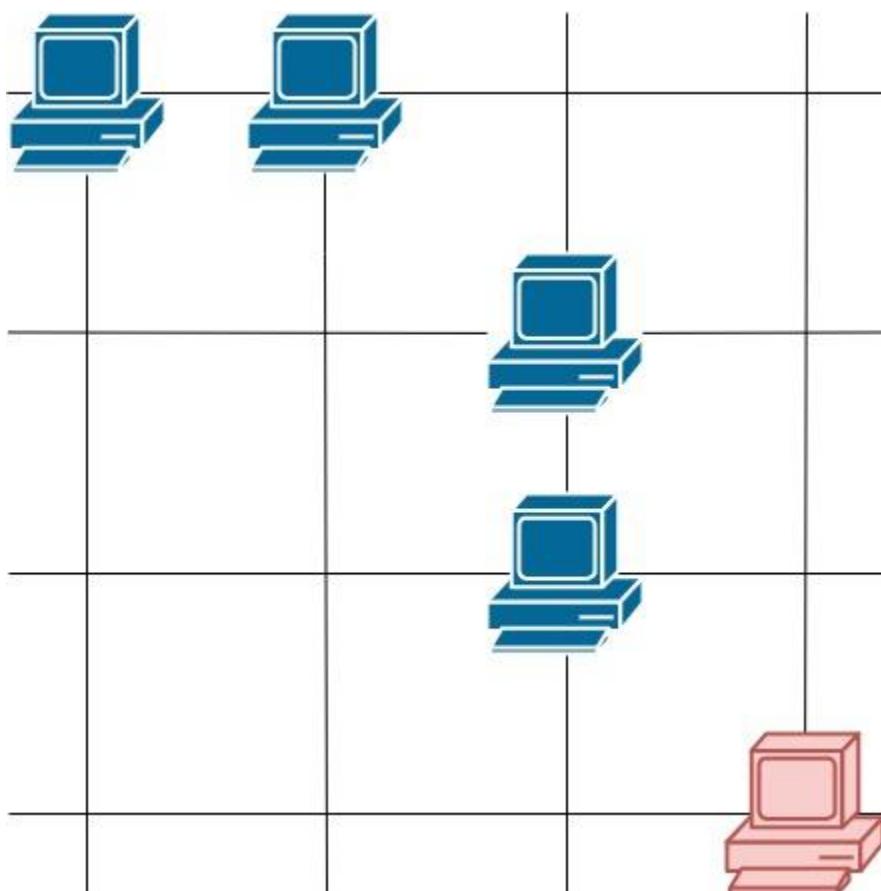


**Input:** grid = [[1,0],[1,1]]

**Output:** 3

**Explanation:** All three servers can communicate with at least one other server.

**Example 3:**



**Input:** grid = [[1,1,0,0],[0,0,1,0],[0,0,1,0],[0,0,0,1]]

**Output:** 4

**Explanation:** The two servers in the first row can communicate with each other. The two servers in the third column can communicate with each other. The server at right bottom corner can't communicate with any other server.

### Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m <= 250`
- `1 <= n <= 250`
- `grid[i][j] == 0 or 1`

## 1269. Number of Ways to Stay in the Same Place After Some Steps

Hard

63434Add to ListShare

You have a pointer at index `0` in an array of size `arrLen`. At each step, you can move 1 position to the left, 1 position to the right in the array, or stay in the same place (The pointer should not be placed outside the array at any time).

Given two integers `steps` and `arrLen`, return the number of ways such that your pointer still at index `0` after **exactly** `steps` steps. Since the answer may be too large, return it **modulo**  $10^9 + 7$ .

### Example 1:

**Input:** `steps = 3, arrLen = 2`

**Output:** `4`

**Explanation:** There are 4 differents ways to stay at index `0` after 3 steps.

Right, Left, Stay

Stay, Right, Left

Right, Stay, Left

Stay, Stay, Stay

### Example 2:

**Input:** `steps = 2, arrLen = 4`

**Output:** `2`

**Explanation:** There are 2 differents ways to stay at index `0` after 2 steps

Right, Left

Stay, Stay

**Example 3:**

**Input:** steps = 4, arrLen = 2

**Output:** 8

**Constraints:**

- $1 \leq \text{steps} \leq 500$
- $1 \leq \text{arrLen} \leq 10^6$

## 1275. Find Winner on a Tic Tac Toe Game

Easy

1051270Add to ListShare

**Tic-tac-toe** is played by two players **A** and **B** on a  $3 \times 3$  grid. The rules of Tic-Tac-Toe are:

- Players take turns placing characters into empty squares ' '.
- The first player **A** always places 'X' characters, while the second player **B** always places 'O' characters.
- 'X' and 'O' characters are always placed into empty squares, never on filled ones.
- The game ends when there are **three** of the same (non-empty) character filling any row, column, or diagonal.
- The game also ends if all squares are non-empty.
- No more moves can be played if the game is over.

Given a 2D integer array **moves** where  $\text{moves}[i] = [\text{row}_i, \text{col}_i]$  indicates that the  $i^{\text{th}}$  move will be played on  $\text{grid}[\text{row}_i][\text{col}_i]$ . return the winner of the game if it exists (**A** or **B**). In case the game ends in a draw return "Draw". If there are still movements to play return "Pending".

You can assume that **moves** is valid (i.e., it follows the rules of **Tic-Tac-Toe**), the grid is initially empty, and **A** will play first.

**Example 1:**

X		
	X	
O	O	X

**Input:** moves = [[0,0],[2,0],[1,1],[2,1],[2,2]]

**Output:** "A"

**Explanation:** A wins, they always play first.

**Example 2:**

X	X	O
X	O	
O		

**Input:** moves = [[0,0],[1,1],[0,1],[0,2],[1,0],[2,0]]

**Output:** "B"

**Explanation:** B wins.

**Example 3:**

X	X	O
O	O	X
X	O	X

**Input:** moves = [[0,0],[1,1],[2,0],[1,0],[1,2],[2,1],[0,1],[0,2],[2,2]]

**Output:** "Draw"

**Explanation:** The game ends in a draw since there are no moves to make.

### Constraints:

- $1 \leq \text{moves.length} \leq 9$
- $\text{moves}[i].length == 2$
- $0 \leq \text{row}_i, \text{col}_i \leq 2$
- There are no repeated elements on moves.
- moves follow the rules of tic tac toe.

## 1276. Number of Burgers with No Waste of Ingredients

Medium

248200Add to ListShare

Given two integers `tomatoSlices` and `cheeseSlices`. The ingredients of different burgers are as follows:

- **Jumbo Burger:** 4 tomato slices and 1 cheese slice.
- **Small Burger:** 2 Tomato slices and 1 cheese slice.

Return `[total_jumbo, total_small]` so that the number of remaining `tomatoSlices` equal to 0 and the number of remaining `cheeseSlices` equal to 0. If it is not possible to make the remaining `tomatoSlices` and `cheeseSlices` equal to 0 return `[]`.

### Example 1:

**Input:** `tomatoSlices = 16, cheeseSlices = 7`

**Output:** [1,6]

**Explanation:** To make one jumbo burger and 6 small burgers we need  $4*1 + 2*6 = 16$  tomato and  $1 + 6 = 7$  cheese.

There will be no remaining ingredients.

**Example 2:**

**Input:** tomatoSlices = 17, cheeseSlices = 4

**Output:** []

**Explanation:** There will be no way to use all ingredients to make small and jumbo burgers.

**Example 3:**

**Input:** tomatoSlices = 4, cheeseSlices = 17

**Output:** []

**Explanation:** Making 1 jumbo burger there will be 16 cheese remaining and making 2 small burgers there will be 15 cheese remaining.

**Constraints:**

- $0 \leq \text{tomatoSlices}, \text{cheeseSlices} \leq 10^7$

## 1277. Count Square Submatrices with All Ones

Medium

381766Add to ListShare

Given a  $m * n$  matrix of ones and zeros, return how many **square** submatrices have all ones.

**Example 1:**

**Input:** matrix =

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

**Output:** 15

**Explanation:**

There are 10 squares of side 1.

There are 4 squares of side 2.

There is 1 square of side 3.

Total number of squares =  $10 + 4 + 1 = 15$ .

**Example 2:**

**Input:** matrix =

[

[1,0,1],

[1,1,0],

[1,1,0]

]

**Output:** 7

**Explanation:**

There are 6 squares of side 1.

There is 1 square of side 2.

Total number of squares =  $6 + 1 = 7$ .

**Constraints:**

- $1 \leq \text{arr.length} \leq 300$
- $1 \leq \text{arr}[0].length \leq 300$
- $0 \leq \text{arr}[i][j] \leq 1$

## 1278. Palindrome Partitioning III

Hard

86412Add to ListShare

You are given a string  $s$  containing lowercase letters and an integer  $k$ . You need to :

- First, change some characters of  $s$  to other lowercase English letters.
- Then divide  $s$  into  $k$  non-empty disjoint substrings such that each substring is a palindrome.

Return the minimal number of characters that you need to change to divide the string.

**Example 1:**

**Input:** s = "abc", k = 2

**Output:** 1

**Explanation:** You can split the string into "ab" and "c", and change 1 character in "ab" to make it palindrome.

**Example 2:**

**Input:** s = "aabbc", k = 3

**Output:** 0

**Explanation:** You can split the string into "aa", "bb" and "c", all of them are palindrome.

**Example 3:**

**Input:** s = "leetcode", k = 8

**Output:** 0

**Constraints:**

- $1 \leq k \leq s.length \leq 100$ .
- s only contains lowercase English letters.

## 1281. Subtract the Product and Sum of Digits of an Integer

Easy

1580198Add to ListShare

Given an integer number `n`, return the difference between the product of its digits and the sum of its digits.

**Example 1:**

**Input:** n = 234

**Output:** 15

**Explanation:**

Product of digits =  $2 * 3 * 4 = 24$

Sum of digits =  $2 + 3 + 4 = 9$

Result =  $24 - 9 = 15$

### Example 2:

**Input:**  $n = 4421$

**Output:**  $21$

### Explanation:

Product of digits =  $4 * 4 * 2 * 1 = 32$

Sum of digits =  $4 + 4 + 2 + 1 = 11$

Result =  $32 - 11 = 21$

### Constraints:

- $1 \leq n \leq 10^5$

## 1282. Group the People Given the Group Size They Belong To

Medium

1238492Add to ListShare

There are  $n$  people that are split into some unknown number of groups. Each person is labeled with a **unique ID** from  $0$  to  $n - 1$ .

You are given an integer array `groupSizes`, where `groupSizes[i]` is the size of the group that person  $i$  is in. For example, if `groupSizes[1] = 3`, then person  $1$  must be in a group of size  $3$ .

Return a *list of groups* such that each person  $i$  is in a group of size `groupSizes[i]`.

Each person should appear in **exactly one group**, and every person must be in a group. If there are multiple answers, **return any of them**. It is **guaranteed** that there will be **at least one** valid solution for the given input.

### Example 1:

**Input:** `groupSizes = [3,3,3,3,3,1,3]`

**Output:** `[[5],[0,1,2],[3,4,6]]`

### Explanation:

The first group is [5]. The size is 1, and groupSizes[5] = 1.

The second group is [0,1,2]. The size is 3, and groupSizes[0] = groupSizes[1] = groupSizes[2] = 3.

The third group is [3,4,6]. The size is 3, and groupSizes[3] = groupSizes[4] = groupSizes[6] = 3.

Other possible solutions are [[2,1,6],[5],[0,4,3]] and [[5],[0,6,2],[4,3,1]].

### Example 2:

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

**Output:** [[1],[0,5],[2,3,4]]

### Constraints:

- groupSizes.length == n
- 1 <= n <= 500
- 1 <= groupSizes[i] <= n

## 1283. Find the Smallest Divisor Given a Threshold

### Medium

1693159Add to ListShare

Given an array of integers `nums` and an integer `threshold`, we will choose a positive integer `divisor`, divide all the array by it, and sum the division's result. Find the **smallest** divisor such that the result mentioned above is less than or equal to `threshold`.

Each result of the division is rounded to the nearest integer greater than or equal to that element. (For example:  $7/3 = 3$  and  $10/2 = 5$ ).

The test cases are generated so that there will be an answer.

### Example 1:

**Input:** nums = [1,2,5,9], threshold = 6

**Output:** 5

**Explanation:** We can get a sum to 17 ( $1+2+5+9$ ) if the divisor is 1.

If the divisor is 4 we can get a sum of 7 ( $1+1+2+3$ ) and if the divisor is 5 the sum will be 5 ( $1+1+1+2$ ).

### Example 2:

**Input:** nums = [44,22,33,11,1], threshold = 5

**Output:** 44

**Constraints:**

- $1 \leq \text{nums.length} \leq 5 * 10^4$
- $1 \leq \text{nums}[i] \leq 10^6$
- $\text{nums.length} \leq \text{threshold} \leq 10^6$

## 1284. Minimum Number of Flips to Convert Binary Matrix to Zero Matrix

Hard

74376Add to ListShare

Given a  $m \times n$  binary matrix `mat`. In one step, you can choose one cell and flip it and all the four neighbors of it if they exist (Flip is changing `1` to `0` and `0` to `1`). A pair of cells are called neighbors if they share one edge.

Return the *minimum number of steps* required to convert `mat` to a zero matrix or `-1` if you cannot.

A **binary matrix** is a matrix with all cells equal to `0` or `1` only.

A **zero matrix** is a matrix with all cells equal to `0`.

**Example 1:**

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

**Input:** mat = [[0,0],[0,1]]

**Output:** 3

**Explanation:** One possible solution is to flip  $(1, 0)$  then  $(0, 1)$  and finally  $(1, 1)$  as shown.

**Example 2:**

**Input:** mat = [[0]]

**Output:** 0

**Explanation:** Given matrix is a zero matrix. We do not need to change it.

**Example 3:**

**Input:** mat = [[1,0,0],[1,0,0]]

**Output:** -1

**Explanation:** Given matrix cannot be a zero matrix.

**Constraints:**

- m == mat.length
- n == mat[i].length
- 1 <= m, n <= 3
- mat[i][j] is either 0 or 1.

**1286. Iterator for Combination**

Medium

122297Add to ListShare

Design the `CombinationIterator` class:

- `CombinationIterator(string characters, int combinationLength)` Initializes the object with a string `characters` of **sorted distinct** lowercase English letters and a number `combinationLength` as arguments.
- `next()` Returns the next combination of length `combinationLength` in **lexicographical order**.
- `hasNext()` Returns `true` if and only if there exists a next combination.

**Example 1:**

**Input**

```
["CombinationIterator", "next", "hasNext", "next", "hasNext", "next", "hasNext"]
```

```
[[["abc", 2], [], [], [], [], [], []]]
```

**Output**

```
[null, "ab", true, "ac", true, "bc", false]
```

**Explanation**

```
CombinationIterator itr = new CombinationIterator("abc", 2);
```

```

itr.next();    // return "ab"
itr.hasNext(); // return True
itr.next();    // return "ac"
itr.hasNext(); // return True
itr.next();    // return "bc"
itr.hasNext(); // return False

```

### Constraints:

- $1 \leq \text{combinationLength} \leq \text{characters.length} \leq 15$
- All the characters of `characters` are **unique**.
- At most  $10^4$  calls will be made to `next` and `hasNext`.
- It is guaranteed that all calls of the function `next` are valid.

## 1287. Element Appearing More Than 25% In Sorted Array

Easy

71739Add to ListShare

Given an integer array **sorted** in non-decreasing order, there is exactly one integer in the array that occurs more than 25% of the time, return that integer.

### Example 1:

**Input:** arr = [1,2,2,6,6,6,6,7,10]

**Output:** 6

### Example 2:

**Input:** arr = [1,1]

**Output:** 1

### Constraints:

- $1 \leq \text{arr.length} \leq 10^4$
- $0 \leq \text{arr[i]} \leq 10^5$

## 1288. Remove Covered Intervals

**Medium**

197048Add to ListShare

Given an array `intervals` where `intervals[i] = [li, ri]` represent the interval  $[l_i, r_i]$ , remove all intervals that are covered by another interval in the list.

The interval  $[a, b]$  is covered by the interval  $[c, d]$  if and only if  $c \leq a$  and  $b \leq d$ .

Return *the number of remaining intervals*.

**Example 1:**

**Input:** `intervals = [[1,4],[3,6],[2,8]]`

**Output:** 2

**Explanation:** Interval  $[3,6]$  is covered by  $[2,8]$ , therefore it is removed.

**Example 2:**

**Input:** `intervals = [[1,4],[2,3]]`

**Output:** 1

**Constraints:**

- $1 \leq \text{intervals.length} \leq 1000$
- $\text{intervals}[i].length == 2$
- $0 \leq l_i < r_i \leq 10^5$
- All the given intervals are **unique**.

**1289. Minimum Falling Path Sum II****Hard**

109575Add to ListShare

Given an  $n \times n$  integer matrix `grid`, return *the minimum sum of a falling path with non-zero shifts*.

A **falling path with non-zero shifts** is a choice of exactly one element from each row of `grid` such that no two elements chosen in adjacent rows are in the same column.

**Example 1:**

1	2	3
4	5	6
7	8	9

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

**Output:** 13

**Explanation:**

The possible falling paths are:

[1,5,9], [1,5,7], [1,6,7], [1,6,8],  
 [2,4,8], [2,4,9], [2,6,7], [2,6,8],  
 [3,4,8], [3,4,9], [3,5,7], [3,5,9]

The falling path with the smallest sum is [1,5,7], so the answer is 13.

**Example 2:**

**Input:** grid = [[7]]

**Output:** 7

**Constraints:**

- `n == grid.length == grid[i].length`
- `1 <= n <= 200`
- `-99 <= grid[i][j] <= 99`

## 1290. Convert Binary Number in a Linked List to Integer

Easy

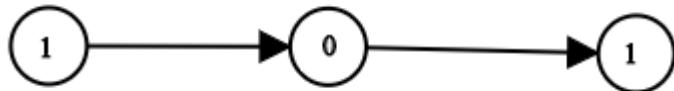
3072139Add to ListShare

Given `head` which is a reference node to a singly-linked list. The value of each node in the linked list is either `0` or `1`. The linked list holds the binary representation of a number.

Return the *decimal value* of the number in the linked list.

The **most significant bit** is at the head of the linked list.

**Example 1:**



**Input:** head = [1,0,1]

**Output:** 5

**Explanation:** (101) in base 2 = (5) in base 10

**Example 2:**

**Input:** head = [0]

**Output:** 0

**Constraints:**

- The Linked List is not empty.
- Number of nodes will not exceed 30.
- Each node's value is either 0 or 1.

## 1291. Sequential Digits

Medium

1772107Add to ListShare

An integer has *sequential digits* if and only if each digit in the number is one more than the previous digit.

Return a **sorted** list of all the integers in the range `[low, high]` inclusive that have sequential digits.

**Example 1:**

**Input:** low = 100, high = 300

**Output:** [123,234]

**Example 2:**

**Input:** low = 1000, high = 13000

**Output:** [1234,2345,3456,4567,5678,6789,12345]

**Constraints:**

- $10 \leq \text{low} \leq \text{high} \leq 10^9$

## 1292. Maximum Side Length of a Square with Sum Less than or Equal to Threshold

**Medium**

91376Add to ListShare

Given a  $m \times n$  matrix `mat` and an integer `threshold`, return the maximum side-length of a square with a sum less than or equal to `threshold` or return 0 if there is no such square.

**Example 1:**

1	1	3	2	4	3	2
1	1	3	2	4	3	2
1	1	3	2	4	3	2

**Input:** mat = [[1,1,3,2,4,3,2],[1,1,3,2,4,3,2],[1,1,3,2,4,3,2]], threshold = 4

**Output:** 2

**Explanation:** The maximum side length of square with sum less than 4 is 2 as shown.

**Example 2:**

**Input:** mat = [[2,2,2,2,2],[2,2,2,2,2],[2,2,2,2,2],[2,2,2,2,2],[2,2,2,2,2]], threshold = 1

**Output:** 0

**Constraints:**

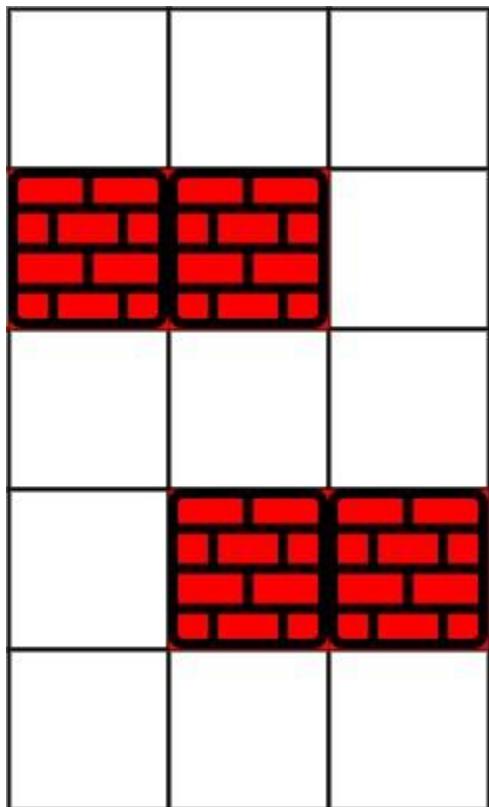
- $m == \text{mat.length}$
- $n == \text{mat[i].length}$
- $1 \leq m, n \leq 300$
- $0 \leq \text{mat[i][j]} \leq 10^4$
- $0 \leq \text{threshold} \leq 10^5$

**1293. Shortest Path in a Grid with Obstacles Elimination****Hard**

280452 Add to List Share

You are given an  $m \times n$  integer matrix `grid` where each cell is either `0` (empty) or `1` (obstacle). You can move up, down, left, or right from and to an empty cell in **one step**.

Return *the minimum number of steps* to walk from the upper left corner  $(0, 0)$  to the lower right corner  $(m - 1, n - 1)$  given that you can eliminate **at most** `k` obstacles. If it is not possible to find such walk return `-1`.

**Example 1:**

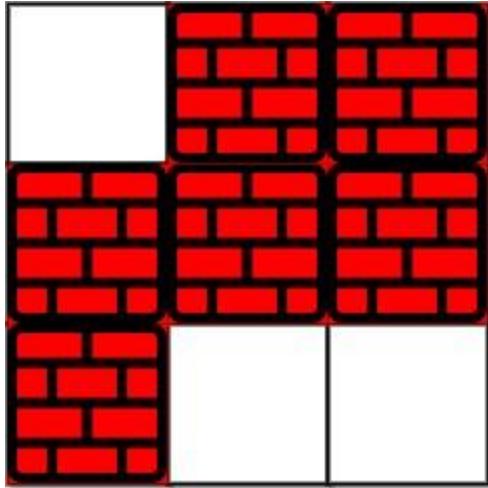
**Input:** `grid = [[0,0,0],[1,1,0],[0,0,0],[0,1,1],[0,0,0]]`, `k = 1`

**Output:** 6

**Explanation:**

The shortest path without eliminating any obstacle is 10.

The shortest path with one obstacle elimination at position (3,2) is 6. Such path is (0,0) -> (0,1) -> (0,2) -> (1,2) -> (2,2) -> (3,2) -> (4,2).

**Example 2:**

**Input:** grid = [[0,1,1],[1,1,1],[1,0,0]], k = 1

**Output:** -1

**Explanation:** We need to eliminate at least two obstacles to find such a walk.

**Constraints:**

- $m == \text{grid.length}$
- $n == \text{grid[i].length}$
- $1 \leq m, n \leq 40$
- $1 \leq k \leq m * n$
- $\text{grid}[i][j]$  is either 0 **or** 1.
- $\text{grid}[0][0] == \text{grid}[m - 1][n - 1] == 0$

## 1295. Find Numbers with Even Number of Digits

Easy

1559105 Add to List Share

Given an array `nums` of integers, return how many of them contain an **even number** of digits.

**Example 1:**

**Input:** nums = [12,345,2,6,7896]

**Output:** 2

**Explanation:**

12 contains 2 digits (even number of digits).

345 contains 3 digits (odd number of digits).

2 contains 1 digit (odd number of digits).

6 contains 1 digit (odd number of digits).

7896 contains 4 digits (even number of digits).

Therefore only 12 and 7896 contain an even number of digits.

**Example 2:**

**Input:** nums = [555,901,482,1771]

**Output:** 1

**Explanation:**

Only 1771 contains an even number of digits.

**Constraints:**

- $1 \leq \text{nums.length} \leq 500$
- $1 \leq \text{nums}[i] \leq 10^5$

## 1296. Divide Array in Sets of K Consecutive Numbers

Medium

130887Add to ListShare

Given an array of integers `nums` and a positive integer `k`, check whether it is possible to divide this array into sets of `k` consecutive numbers.

Return `true` if it is possible. Otherwise, return `false`.

**Example 1:**

**Input:** nums = [1,2,3,3,4,4,5,6], k = 4

**Output:** true

**Explanation:** Array can be divided into [1,2,3,4] and [3,4,5,6].

**Example 2:**

**Input:** nums = [3,2,1,2,3,4,3,4,5,9,10,11], k = 3

**Output:** true

**Explanation:** Array can be divided into [1,2,3] , [2,3,4] , [3,4,5] and [9,10,11].

**Example 3:**

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

**Output:** false

**Explanation:** Each array should be divided in subarrays of size 3.

**Constraints:**

- $1 \leq k \leq \text{nums.length} \leq 10^5$
- $1 \leq \text{nums}[i] \leq 10^9$

## 1297. Maximum Number of Occurrences of a Substring

Medium

729326Add to ListShare

Given a string `s`, return the maximum number of occurrences of **any** substring under the following rules:

- The number of unique characters in the substring must be less than or equal to `maxLetters`.
- The substring size must be between `minSize` and `maxSize` inclusive.

**Example 1:**

**Input:** s = "aababcaab", `maxLetters` = 2, `minSize` = 3, `maxSize` = 4

**Output:** 2

**Explanation:** Substring "aab" has 2 occurrences in the original string.

It satisfies the conditions, 2 unique letters and size 3 (between `minSize` and `maxSize`).

**Example 2:**

**Input:** `s = "aaaa", maxLetters = 1, minSize = 3, maxSize = 3`

**Output:** 2

**Explanation:** Substring "aaa" occur 2 times in the string. It can overlap.

### Constraints:

- `1 <= s.length <= 105`
- `1 <= maxLetters <= 26`
- `1 <= minSize <= maxSize <= min(26, s.length)`
- `s` consists of only lowercase English letters.

## 1298. Maximum Candies You Can Get from Boxes

Hard

213120Add to ListShare

You have `n` boxes labeled from `0` to `n - 1`. You are given four arrays: `status`, `candies`, `keys`, and `containedBoxes` where:

- `status[i]` is `1` if the `ith` box is open and `0` if the `ith` box is closed,
- `candies[i]` is the number of candies in the `ith` box,
- `keys[i]` is a list of the labels of the boxes you can open after opening the `ith` box.
- `containedBoxes[i]` is a list of the boxes you found inside the `ith` box.

You are given an integer array `initialBoxes` that contains the labels of the boxes you initially have. You can take all the candies in **any open box** and you can use the keys in it to open new boxes and you also can use the boxes you find in it.

Return *the maximum number of candies you can get following the rules above*.

### Example 1:

**Input:** `status = [1,0,1,0], candies = [7,5,4,100], keys = [[],[],[1],[]], containedBoxes = [[1,2],[3],[],[]], initialBoxes = [0]`

**Output:** 16

**Explanation:** You will be initially given box `0`. You will find 7 candies in it and boxes `1` and `2`.

Box `1` is closed and you do not have a key for it so you will open box `2`. You will find 4 candies and a key to box `1` in box `2`.

In box `1`, you will find 5 candies and box `3` but you will not find a key to box `3` so box `3` will remain closed.

Total number of candies collected =  $7 + 4 + 5 = 16$  candy.

**Example 2:**

**Input:** status = [1,0,0,0,0,0], candies = [1,1,1,1,1,1], keys = [[1,2,3,4,5],[],[],[],[]], containedBoxes = [[1,2,3,4,5],[],[],[],[],[]], initialBoxes = [0]

**Output:** 6

**Explanation:** You have initially box 0. Opening it you can find boxes 1,2,3,4 and 5 and their keys.

The total number of candies will be 6.

**Constraints:**

- $n == \text{status.length} == \text{candies.length} == \text{keys.length} == \text{containedBoxes.length}$
- $1 \leq n \leq 1000$
- $\text{status}[i]$  is either 0 or 1.
- $1 \leq \text{candies}[i] \leq 1000$
- $0 \leq \text{keys}[i].length \leq n$
- $0 \leq \text{keys}[i][j] < n$
- All values of  $\text{keys}[i]$  are **unique**.
- $0 \leq \text{containedBoxes}[i].length \leq n$
- $0 \leq \text{containedBoxes}[i][j] < n$
- All values of  $\text{containedBoxes}[i]$  are unique.
- Each box is contained in one box at most.
- $0 \leq \text{initialBoxes.length} \leq n$
- $0 \leq \text{initialBoxes}[i] < n$

## 1299. Replace Elements with Greatest Element on Right Side

Easy

1481171Add to ListShare

Given an array  $\text{arr}$ , replace every element in that array with the greatest element among the elements to its right, and replace the last element with  $-1$ .

After doing so, return the array.

**Example 1:**

**Input:** arr = [17,18,5,4,6,1]

**Output:** [18,6,6,6,1,-1]

**Explanation:**

- index 0 --> the greatest element to the right of index 0 is index 1 (18).
- index 1 --> the greatest element to the right of index 1 is index 4 (6).
- index 2 --> the greatest element to the right of index 2 is index 4 (6).
- index 3 --> the greatest element to the right of index 3 is index 4 (6).
- index 4 --> the greatest element to the right of index 4 is index 5 (1).
- index 5 --> there are no elements to the right of index 5, so we put -1.

**Example 2:**

**Input:** arr = [400]

**Output:** [-1]

**Explanation:** There are no elements to the right of index 0.

**Constraints:**

- $1 \leq \text{arr.length} \leq 10^4$
- $1 \leq \text{arr}[i] \leq 10^5$

## 1300. Sum of Mutated Array Closest to Target

Medium

881108Add to ListShare

Given an integer array `arr` and a target value `target`, return the integer `value` such that when we change all the integers larger than `value` in the given array to be equal to `value`, the sum of the array gets as close as possible (in absolute difference) to `target`.

In case of a tie, return the minimum such integer.

Notice that the answer is not necessarily a number from `arr`.

**Example 1:**

**Input:** arr = [4,9,3], target = 10

**Output:** 3

**Explanation:** When using 3 arr converts to [3, 3, 3] which sums 9 and that's the optimal answer.

**Example 2:**

**Input:** arr = [2,3,5], target = 10

**Output:** 5

**Example 3:**

**Input:** arr = [60864,25176,27249,21296,20204], target = 56803

**Output:** 11361

**Constraints:**

- $1 \leq \text{arr.length} \leq 10^4$
- $1 \leq \text{arr}[i], \text{target} \leq 10^5$

**1301. Number of Paths with Max Score**

Hard

38919Add to ListShare

You are given a square `board` of characters. You can move on the board starting at the bottom right square marked with the character '`S`'.

You need to reach the top left square marked with the character '`E`'. The rest of the squares are labeled either with a numeric character `1, 2, ..., 9` or with an obstacle '`X`'. In one move you can go up, left or up-left (diagonally) only if there is no obstacle there.

Return a list of two integers: the first integer is the maximum sum of numeric characters you can collect, and the second is the number of such paths that you can take to get that maximum sum, **taken modulo  $10^9 + 7$** .

In case there is no path, return `[0, 0]`.

**Example 1:**

**Input:** board = ["E23", "2X2", "12S"]

**Output:** [7,1]

**Example 2:**

**Input:** board = ["E12", "1X1", "21S"]

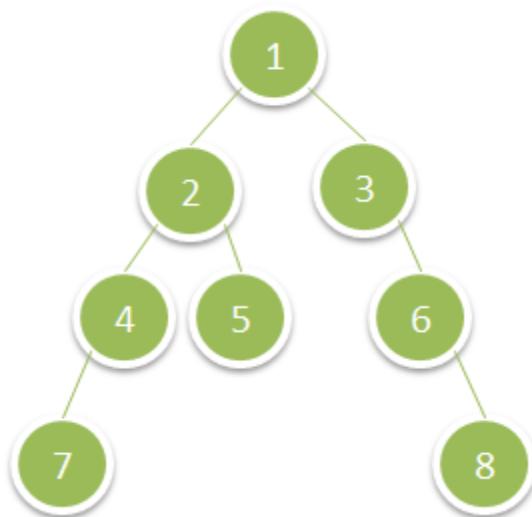
**Output:** [4,2]

**Example 3:****Input:** board = ["E11", "XXX", "11S"]**Output:** [0,0]**Constraints:**

- $2 \leq \text{board.length} == \text{board}[i].length \leq 100$

**1302. Deepest Leaves Sum****Medium**

3693100Add to ListShare

Given the `root` of a binary tree, return *the sum of values of its deepest leaves*.**Example 1:****Input:** root = [1,2,3,4,5,null,6,7,null,null,null,8]**Output:** 15**Example 2:****Input:** root = [6,7,8,2,7,1,3,9,null,1,4,null,null,null,5]**Output:** 19**Constraints:**

- The number of nodes in the tree is in the range `[1, 104]`.
- `1 <= Node.val <= 100`

## 1304. Find N Unique Integers Sum up to Zero

Easy

1384509Add to ListShare

Given an integer `n`, return **any** array containing `n` **unique** integers such that they add up to `0`.

**Example 1:**

**Input:** `n = 5`

**Output:** `[-7, -1, 1, 3, 4]`

**Explanation:** These arrays also are accepted `[-5, -1, 1, 2, 3]` , `[-3, -1, 2, -2, 4]`.

**Example 2:**

**Input:** `n = 3`

**Output:** `[-1, 0, 1]`

**Example 3:**

**Input:** `n = 1`

**Output:** `[0]`

**Constraints:**

- `1 <= n <= 1000`

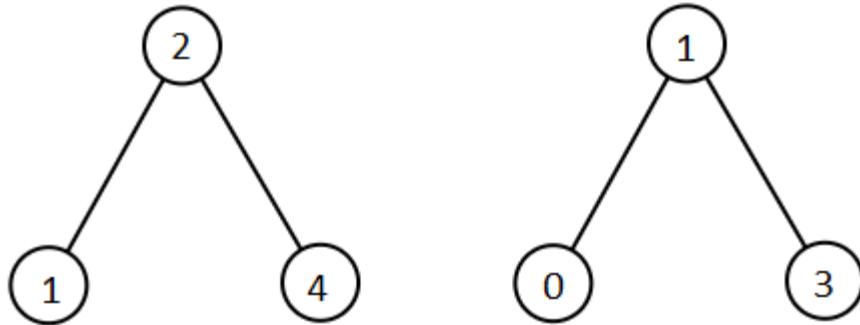
## 1305. All Elements in Two Binary Search Trees

Medium

248672Add to ListShare

Given two binary search trees `root1` and `root2`, return a list containing all the integers from both trees sorted in **ascending** order.

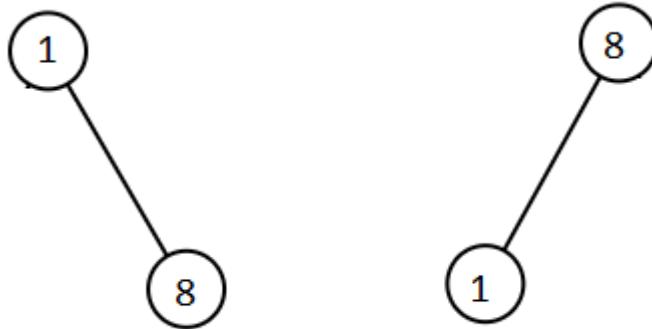
**Example 1:**



**Input:** root1 = [2,1,4], root2 = [1,0,3]

**Output:** [0,1,1,2,3,4]

**Example 2:**



**Input:** root1 = [1,null,8], root2 = [8,1]

**Output:** [1,1,8,8]

**Constraints:**

- The number of nodes in each tree is in the range [0, 5000].
- $-10^5 \leq \text{Node.val} \leq 10^5$

## 1306. Jump Game III

Medium

318474Add to ListShare

Given an array of non-negative integers `arr`, you are initially positioned at `start` index of the array. When you are at index `i`, you can jump to `i + arr[i]` or `i - arr[i]`, check if you can reach to **any** index with value 0.

Notice that you can not jump outside of the array at any time.

**Example 1:****Input:** arr = [4,2,3,0,3,1,2], start = 5**Output:** true**Explanation:**

All possible ways to reach at index 3 with value 0 are:

index 5 -&gt; index 4 -&gt; index 1 -&gt; index 3

index 5 -&gt; index 6 -&gt; index 4 -&gt; index 1 -&gt; index 3

**Example 2:****Input:** arr = [4,2,3,0,3,1,2], start = 0**Output:** true**Explanation:**

One possible way to reach at index 3 with value 0 is:

index 0 -&gt; index 4 -&gt; index 1 -&gt; index 3

**Example 3:****Input:** arr = [3,0,2,1,2], start = 2**Output:** false**Explanation:** There is no way to reach at index 1 with value 0.**Constraints:**

- $1 \leq \text{arr.length} \leq 5 * 10^4$
- $0 \leq \text{arr}[i] < \text{arr.length}$
- $0 \leq \text{start} < \text{arr.length}$

**1307. Verbal Arithmetic Puzzle****Hard**

36196Add to ListShare

Given an equation, represented by `words` on the left side and the `result` on the right side.

You need to check if the equation is solvable under the following rules:

- Each character is decoded as one digit (0 - 9).
- No two characters can map to the same digit.
- Each `words[i]` and `result` are decoded as one number **without** leading zeros.
- Sum of numbers on the left side (`words`) will equal to the number on the right side (`result`).

Return `true` if the equation is solvable, otherwise return `false`.

### Example 1:

**Input:** `words = ["SEND", "MORE"]`, `result = "MONEY"`

**Output:** `true`

**Explanation:** Map 'S' -> 9, 'E' -> 5, 'N' -> 6, 'D' -> 7, 'M' -> 1, 'O' -> 0, 'R' -> 8, 'Y' -> 2'

Such that: "SEND" + "MORE" = "MONEY" , 9567 + 1085 = 10652

### Example 2:

**Input:** `words = ["SIX", "SEVEN", "SEVEN"]`, `result = "TWENTY"`

**Output:** `true`

**Explanation:** Map 'S' -> 6, 'I' -> 5, 'X' -> 0, 'E' -> 8, 'V' -> 7, 'N' -> 2, 'T' -> 1, 'W' -> 3', 'Y' -> 4

Such that: "SIX" + "SEVEN" + "SEVEN" = "TWENTY" , 650 + 68782 + 68782 = 138214

### Example 3:

**Input:** `words = ["LEET", "CODE"]`, `result = "POINT"`

**Output:** `false`

**Explanation:** There is no possible mapping to satisfy the equation, so we return `false`.

Note that two different characters cannot map to the same digit.

### Constraints:

- `2 <= words.length <= 5`
- `1 <= words[i].length, result.length <= 7`
- `words[i], result` contain only uppercase English letters.
- The number of different characters used in the expression is at most 10.

## 1309. Decrypt String from Alphabet to Integer Mapping

**Easy**

115683Add to ListShare

You are given a string `s` formed by digits and `'#'`. We want to map `s` to English lowercase characters as follows:

- Characters (`'a'` to `'i'`) are represented by (`'1'` to `'9'`) respectively.
- Characters (`'j'` to `'z'`) are represented by (`'10#'` to `'26#'`) respectively.

Return *the string formed after mapping*.

The test cases are generated so that a unique mapping will always exist.

**Example 1:**

**Input:** `s = "10#11#12"`

**Output:** `"jkab"`

**Explanation:** `"j" -> "10#"` , `"k" -> "11#"` , `"a" -> "1"` , `"b" -> "2"`.

**Example 2:**

**Input:** `s = "1326#"`

**Output:** `"acz"`

**Constraints:**

- `1 <= s.length <= 1000`
- `s` consists of digits and the `'#'` letter.
- `s` will be a valid string such that mapping is always possible.

**1310. XOR Queries of a Subarray****Medium**

109634Add to ListShare

You are given an array `arr` of positive integers. You are also given the array `queries` where `queries[i] = [lefti, righti]`.

For each query `i` compute the **XOR** of elements from `lefti` to `righti` (that is, `arr[lefti] XOR arr[lefti + 1] XOR ... XOR arr[righti]`).

Return an array `answer` where `answer[i]` is the answer to the `ith` query.

**Example 1:**

**Input:** arr = [1,3,4,8], queries = [[0,1],[1,2],[0,3],[3,3]]

**Output:** [2,7,14,8]

**Explanation:**

The binary representation of the elements in the array are:

1 = 0001

3 = 0011

4 = 0100

8 = 1000

The XOR values for queries are:

[0,1] = 1 xor 3 = 2

[1,2] = 3 xor 4 = 7

[0,3] = 1 xor 3 xor 4 xor 8 = 14

[3,3] = 8

**Example 2:**

**Input:** arr = [4,8,2,10], queries = [[2,3],[1,3],[0,0],[0,3]]

**Output:** [8,0,4,4]

**Constraints:**

- $1 \leq \text{arr.length}, \text{queries.length} \leq 3 * 10^4$
- $1 \leq \text{arr}[i] \leq 10^9$
- $\text{queries}[i].length == 2$
- $0 \leq \text{left}_i \leq \text{right}_i < \text{arr.length}$

**1311. Get Watched Videos by Your Friends**

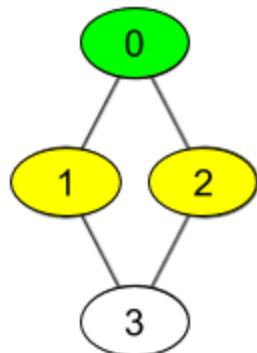
Medium

244325Add to ListShare

There are  $n$  people, each person has a unique  $id$  between 0 and  $n-1$ . Given the arrays `watchedVideos` and `friends`, where `watchedVideos[i]` and `friends[i]` contain the list of watched videos and the list of friends respectively for the person with  $id = i$ .

Level **1** of videos are all watched videos by your friends, level **2** of videos are all watched videos by the friends of your friends and so on. In general, the level  $k$  of videos are all watched videos by people with the shortest path **exactly** equal to  $k$  with you. Given your `id` and the `level` of videos, return the list of videos ordered by their frequencies (increasing). For videos with the same frequency order them alphabetically from least to greatest.

### Example 1:



**Input:** `watchedVideos = [["A", "B"], ["C"], ["B", "C"], ["D"]]`, `friends = [[1,2], [0,3], [0,3], [1,2]]`, `id = 0`, `level = 1`

**Output:** `["B", "C"]`

#### Explanation:

You have  $id = 0$  (green color in the figure) and your friends are (yellow color in the figure):

Person with  $id = 1 \rightarrow$  `watchedVideos = ["C"]`

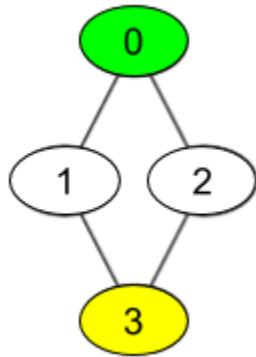
Person with  $id = 2 \rightarrow$  `watchedVideos = ["B", "C"]`

The frequencies of `watchedVideos` by your friends are:

`B`  $\rightarrow$  1

`C`  $\rightarrow$  2

### Example 2:



**Input:** watchedVideos = [["A","B"],["C"],[["B","C"],[["D"]]], friends = [[1,2],[0,3],[0,3],[1,2]]], id = 0, level = 2

**Output:** ["D"]

**Explanation:**

You have id = 0 (green color in the figure) and the only friend of your friends is the person with id = 3 (yellow color in the figure).

**Constraints:**

- `n == watchedVideos.length == friends.length`
- `2 <= n <= 100`
- `1 <= watchedVideos[i].length <= 100`
- `1 <= watchedVideos[i][j].length <= 8`
- `0 <= friends[i].length < n`
- `0 <= friends[i][j] < n`
- `0 <= id < n`
- `1 <= level < n`
- if `friends[i]` contains `j`, then `friends[j]` contains `i`

## 1312. Minimum Insertion Steps to Make a String Palindrome

Hard

253233Add to ListShare

Given a string `s`. In one step you can insert any character at any index of the string.

Return *the minimum number of steps* to make `s` palindrome.

A **Palindrome String** is one that reads the same backward as well as forward.

**Example 1:**

**Input:** `s = "zzazz"`

**Output:** 0

**Explanation:** The string "zzazz" is already palindrome we don't need any insertions.

**Example 2:**

**Input:** s = "mbadm"

**Output:** 2

**Explanation:** String can be "mbdadbm" or "mdbabdm".

**Example 3:**

**Input:** s = "leetcode"

**Output:** 5

**Explanation:** Inserting 5 characters the string becomes "leetcodocoteel".

**Constraints:**

- $1 \leq s.length \leq 500$
- s consists of lowercase English letters.

### 1313. Decompress Run-Length Encoded List

Easy

8801168Add to ListShare

We are given a list `nums` of integers representing a list compressed with run-length encoding.

Consider each adjacent pair of elements `[freq, val] = [nums[2*i], nums[2*i+1]]` (with  $i \geq 0$ ). For each such pair, there are `freq` elements with value `val` concatenated in a sublist. Concatenate all the sublists from left to right to generate the decompressed list.

Return the decompressed list.

**Example 1:**

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

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

**Explanation:** The first pair [1,2] means we have freq = 1 and val = 2 so we generate the array [2].

The second pair [3,4] means we have freq = 3 and val = 4 so we generate [4,4,4].

At the end the concatenation `[2] + [4,4,4]` is `[2,4,4,4]`.

**Example 2:**

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

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

**Constraints:**

- `2 <= nums.length <= 100`
- `nums.length % 2 == 0`
- `1 <= nums[i] <= 100`

## 1314. Matrix Block Sum

Medium

1902301Add to ListShare

Given a `m x n` matrix `mat` and an integer `k`, return a matrix `answer` where each `answer[i][j]` is the sum of all elements `mat[r][c]` for:

- `i - k <= r <= i + k`,
- `j - k <= c <= j + k`, and
- `(r, c)` is a valid position in the matrix.

**Example 1:**

**Input:** `mat = [[1,2,3],[4,5,6],[7,8,9]]`, `k = 1`

**Output:** `[[12,21,16],[27,45,33],[24,39,28]]`

**Example 2:**

**Input:** `mat = [[1,2,3],[4,5,6],[7,8,9]]`, `k = 2`

**Output:** `[[45,45,45],[45,45,45],[45,45,45]]`

**Constraints:**

- `m == mat.length`
- `n == mat[i].length`
- `1 <= m, n, k <= 100`
- `1 <= mat[i][j] <= 100`

## 1315. Sum of Nodes with Even-Valued Grandparent

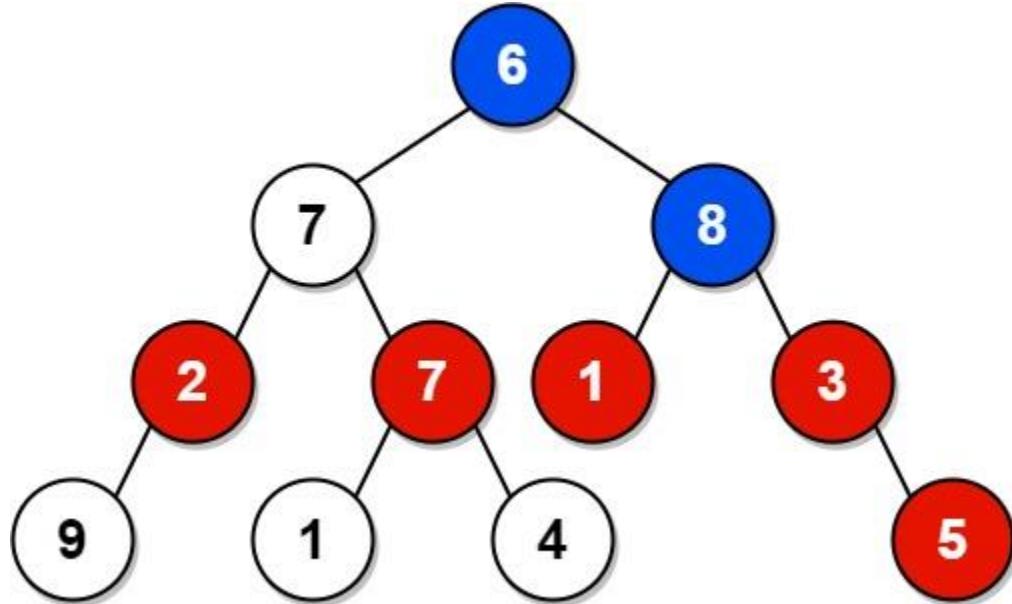
Medium

207762Add to ListShare

Given the `root` of a binary tree, return *the sum of values of nodes with an even-valued grandparent*. If there are no nodes with an **even-valued grandparent**, return `0`.

A **grandparent** of a node is the parent of its parent if it exists.

**Example 1:**



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

**Output:** 18

**Explanation:** The red nodes are the nodes with even-value grandparent while the blue nodes are the even-value grandparents.

**Example 2:**



**Input:** root = [1]

**Output:** 0

**Constraints:**

- The number of nodes in the tree is in the range  $[1, 10^4]$ .
- $1 \leq \text{Node.val} \leq 100$

## 1316. Distinct Echo Substrings

Hard

238184Add to ListShare

Return the number of **distinct** non-empty substrings of `text` that can be written as the concatenation of some string with itself (i.e. it can be written as  $a + a$  where  $a$  is some string).

**Example 1:**

**Input:** `text = "abcabcabc"`

**Output:** 3

**Explanation:** The 3 substrings are "abcabc", "bcabca" and "cabcab".

**Example 2:**

**Input:** `text = "leetcodeleetcode"`

**Output:** 2

**Explanation:** The 2 substrings are "ee" and "leetcodeleetcode".

**Constraints:**

- $1 \leq \text{text.length} \leq 2000$
- `text` has only lowercase English letters.

## 1317. Convert Integer to the Sum of Two No-Zero Integers

Easy

298240Add to ListShare

**No-Zero integer** is a positive integer that **does not contain any 0** in its decimal representation.

Given an integer `n`, return a list of two integers  $[A, B]$  where:

- `A` and `B` are **No-Zero integers**.
- $A + B = n$

The test cases are generated so that there is at least one valid solution. If there are many valid solutions you can return any of them.

**Example 1:****Input:**  $n = 2$ **Output:**  $[1,1]$ **Explanation:**  $A = 1$ ,  $B = 1$ .  $A + B = n$  and both  $A$  and  $B$  do not contain any 0 in their decimal representation.**Example 2:****Input:**  $n = 11$ **Output:**  $[2,9]$ **Constraints:**

- $2 \leq n \leq 10^4$

**1318. Minimum Flips to Make a OR b Equal to c****Medium**

52340Add to ListShare

Given 3 positive numbers  $a$ ,  $b$  and  $c$ . Return the minimum flips required in some bits of  $a$  and  $b$  to make  $(a \text{ OR } b == c)$ . (bitwise OR operation).Flip operation consists of change **any** single bit 1 to 0 or change the bit 0 to 1 in their binary representation.**Example 1:**

0010 -> a	0001 -> a
0110 -> b	0100 -> b
-----	-----
0101 -> c	0101 -> c

**Input:**  $a = 2$ ,  $b = 6$ ,  $c = 5$ **Output:** 3**Explanation:** After flips  $a = 1$ ,  $b = 4$ ,  $c = 5$  such that  $(a \text{ OR } b == c)$ **Example 2:**

**Input:** a = 4, b = 2, c = 7

**Output:** 1

**Example 3:**

**Input:** a = 1, b = 2, c = 3

**Output:** 0

**Constraints:**

- $1 \leq a \leq 10^9$
- $1 \leq b \leq 10^9$
- $1 \leq c \leq 10^9$

## 1319. Number of Operations to Make Network Connected

Medium

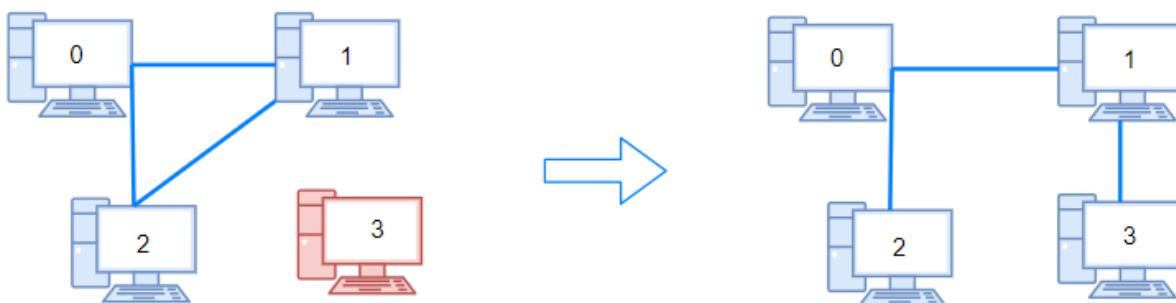
271434Add to ListShare

There are  $n$  computers numbered from 0 to  $n - 1$  connected by ethernet cables `connections` forming a network where `connections[i] = [ai, bi]` represents a connection between computers  $a_i$  and  $b_i$ . Any computer can reach any other computer directly or indirectly through the network.

You are given an initial computer network `connections`. You can extract certain cables between two directly connected computers, and place them between any pair of disconnected computers to make them directly connected.

Return *the minimum number of times you need to do this in order to make all the computers connected*. If it is not possible, return `-1`.

**Example 1:**

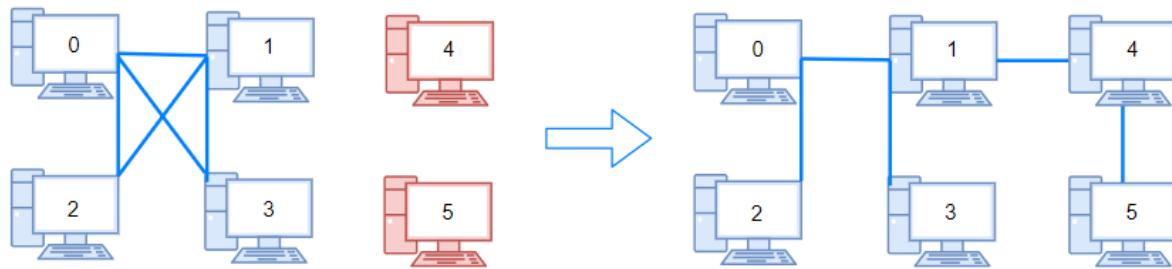


**Input:** n = 4, `connections` = [[0,1],[0,2],[1,2]]

**Output:** 1

**Explanation:** Remove cable between computer 1 and 2 and place between computers 1 and 3.

**Example 2:**



**Input:**  $n = 6$ ,  $\text{connections} = [[0,1],[0,2],[0,3],[1,2],[1,3]]$

**Output:** 2

**Example 3:**

**Input:**  $n = 6$ ,  $\text{connections} = [[0,1],[0,2],[0,3],[1,2]]$

**Output:** -1

**Explanation:** There are not enough cables.

**Constraints:**

- $1 \leq n \leq 10^5$
- $1 \leq \text{connections.length} \leq \min(n * (n - 1) / 2, 10^5)$
- $\text{connections}[i].length == 2$
- $0 \leq a_i, b_i < n$
- $a_i \neq b_i$
- There are no repeated connections.
- No two computers are connected by more than one cable.

## 1320. Minimum Distance to Type a Word Using Two Fingers

Hard

85131Add to ListShare

A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	X
Y	Z				

You have a keyboard layout as shown above in the **X-Y** plane, where each English uppercase letter is located at some coordinate.

- For example, the letter '**A**' is located at coordinate  $(0, 0)$ , the letter '**B**' is located at coordinate  $(0, 1)$ , the letter '**P**' is located at coordinate  $(2, 3)$  and the letter '**Z**' is located at coordinate  $(4, 1)$ .

Given the string `word`, return *the minimum total **distance** to type such string using only two fingers*.

The **distance** between coordinates  $(x_1, y_1)$  and  $(x_2, y_2)$  is  $|x_1 - x_2| + |y_1 - y_2|$ .

**Note** that the initial positions of your two fingers are considered free so do not count towards your total distance, also your two fingers do not have to start at the first letter or the first two letters.

### Example 1:

**Input:** word = "CAKE"

**Output:** 3

**Explanation:** Using two fingers, one optimal way to type "CAKE" is:

Finger 1 on letter 'C' -> cost = 0

Finger 1 on letter 'A' -> cost = Distance from letter 'C' to letter 'A' = 2

Finger 2 on letter 'K' -> cost = 0

Finger 2 on letter 'E' -> cost = Distance from letter 'K' to letter 'E' = 1

Total distance = 3

### Example 2:

**Input:** word = "HAPPY"

**Output:** 6

**Explanation:** Using two fingers, one optimal way to type "HAPPY" is:

Finger 1 on letter 'H' -> cost = 0

Finger 1 on letter 'A' -> cost = Distance from letter 'H' to letter 'A' = 2

Finger 2 on letter 'P' -> cost = 0

Finger 2 on letter 'P' -> cost = Distance from letter 'P' to letter 'P' = 0

Finger 1 on letter 'Y' -> cost = Distance from letter 'A' to letter 'Y' = 4

Total distance = 6

### Constraints:

- `2 <= word.length <= 300`
- `word` consists of uppercase English letters.

## 1323. Maximum 69 Number

**Easy**

1181129Add to ListShare

You are given a positive integer `num` consisting only of digits 6 and 9.

Return *the maximum number you can get by changing **at most** one digit (6 becomes 9, and 9 becomes 6).*

### Example 1:

**Input:** num = 9669

**Output:** 9969

### Explanation:

Changing the first digit results in 6669.

Changing the second digit results in 9969.

Changing the third digit results in 9699.

Changing the fourth digit results in 9666.

The maximum number is 9969.

**Example 2:****Input:** num = 9996**Output:** 9999**Explanation:** Changing the last digit 6 to 9 results in the maximum number.**Example 3:****Input:** num = 9999**Output:** 9999**Explanation:** It is better not to apply any change.**Constraints:**

- $1 \leq \text{num} \leq 10^4$
- num consists of only 6 and 9 digits.

**1324. Print Words Vertically****Medium**

51196Add to ListShare

Given a string s. Return all the words vertically in the same order in which they appear in s.

Words are returned as a list of strings, complete with spaces when necessary. (Trailing spaces are not allowed).

Each word would be put on only one column and that in one column there will be only one word.

**Example 1:****Input:** s = "HOW ARE YOU"**Output:** ["HAY", "ORO", "WEU"]**Explanation:** Each word is printed vertically.

"HAY"

"ORO"

"WEU"

**Example 2:****Input:** s = "TO BE OR NOT TO BE"

**Output:** ["TBONTB", "OEROOE", " T"]

**Explanation:** Trailing spaces is not allowed.

"TBONTB"

"OEROOE"

" T"

**Example 3:**

**Input:** s = "CONTEST IS COMING"

**Output:** ["CIC", "OSO", "N M", "T I", "E N", "S G", "T"]

**Constraints:**

- $1 \leq s.length \leq 200$
- $s$  contains only upper case English letters.
- It's guaranteed that there is only one space between 2 words

## 1325. Delete Leaves With a Given Value

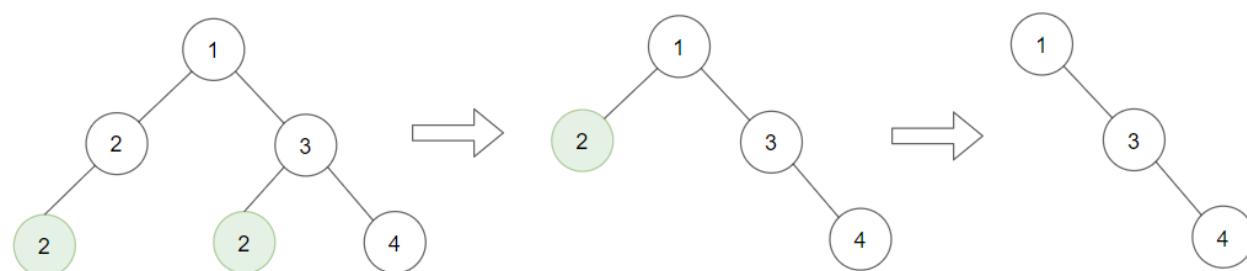
Medium

171031Add to ListShare

Given a binary tree `root` and an integer `target`, delete all the **leaf nodes** with value `target`.

Note that once you delete a leaf node with value `target`, if its parent node becomes a leaf node and has the value `target`, it should also be deleted (you need to continue doing that until you cannot).

**Example 1:**



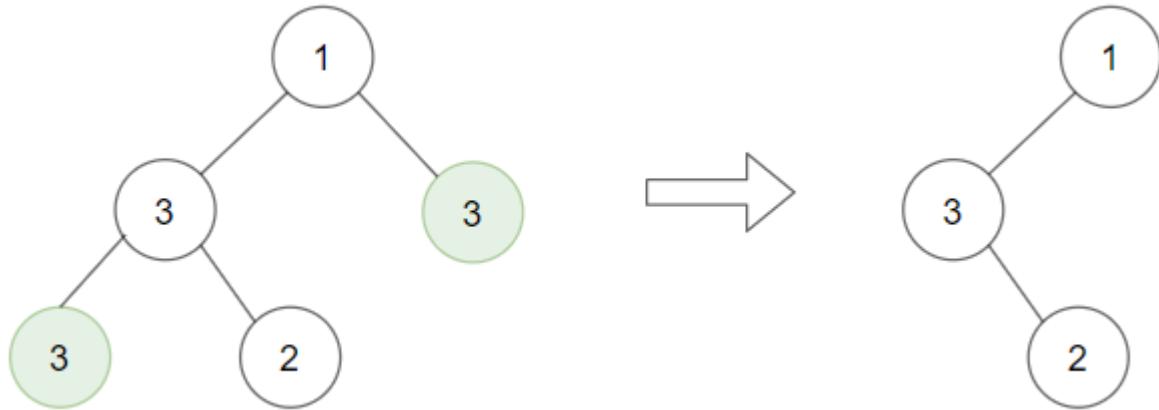
**Input:** root = [1,2,3,2,null,2,4], target = 2

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

**Explanation:** Leaf nodes in green with value (target = 2) are removed (Picture in left).

After removing, new nodes become leaf nodes with value (target = 2) (Picture in center).

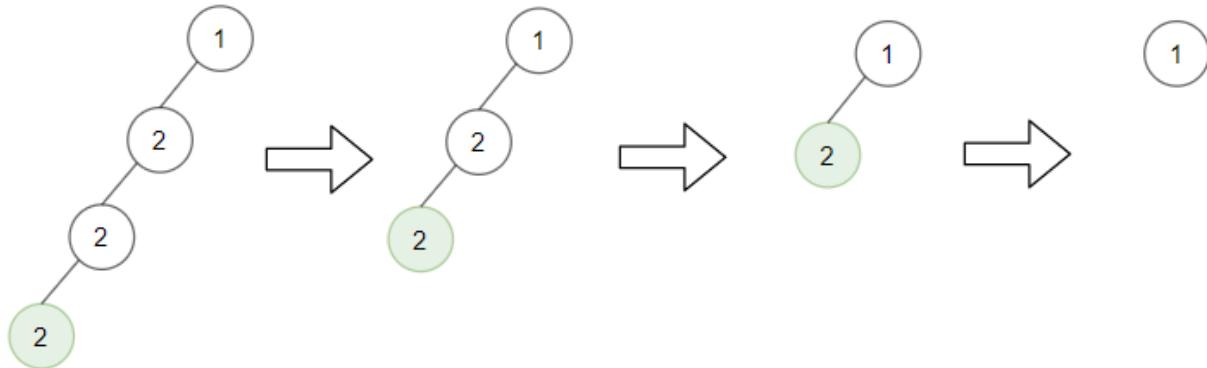
**Example 2:**



**Input:** root = [1,3,3,3,2], target = 3

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

**Example 3:**



**Input:** root = [1,2,null,2,null,2], target = 2

**Output:** [1]

**Explanation:** Leaf nodes in green with value (target = 2) are removed at each step.

**Constraints:**

- The number of nodes in the tree is in the range `[1, 3000]`.
- `1 <= Node.val, target <= 1000`

## 1326. Minimum Number of Taps to Open to Water a Garden

Hard

1744118Add to ListShare

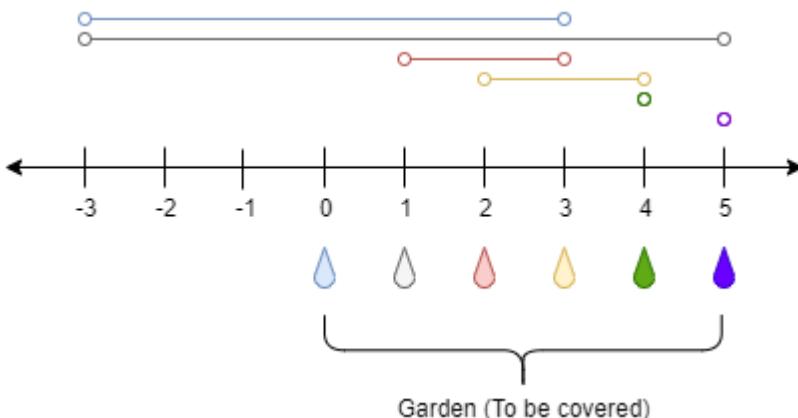
There is a one-dimensional garden on the x-axis. The garden starts at the point `0` and ends at the point `n`. (i.e The length of the garden is `n`).

There are `n + 1` taps located at points `[0, 1, ..., n]` in the garden.

Given an integer `n` and an integer array `ranges` of length `n + 1` where `ranges[i]` (0-indexed) means the `i`-th tap can water the area `[i - ranges[i], i + ranges[i]]` if it was open.

Return *the minimum number of taps* that should be open to water the whole garden, If the garden cannot be watered return `-1`.

### Example 1:



**Input:** `n = 5, ranges = [3,4,1,1,0,0]`

**Output:** `1`

**Explanation:** The tap at point `0` can cover the interval `[-3,3]`

The tap at point `1` can cover the interval `[-3,5]`

The tap at point `2` can cover the interval `[1,3]`

The tap at point `3` can cover the interval `[2,4]`

The tap at point `4` can cover the interval `[4,4]`

The tap at point `5` can cover the interval `[5,5]`

Opening Only the second tap will water the whole garden [0,5]

**Example 2:**

**Input:** n = 3, ranges = [0,0,0,0]

**Output:** -1

**Explanation:** Even if you activate all the four taps you cannot water the whole garden.

**Constraints:**

- $1 \leq n \leq 10^4$
- $\text{ranges.length} == n + 1$
- $0 \leq \text{ranges}[i] \leq 100$

## 1328. Break a Palindrome

Medium

852461Add to ListShare

Given a palindromic string of lowercase English letters `palindrome`, replace **exactly one** character with any lowercase English letter so that the resulting string is **not** a palindrome and that it is the **lexicographically smallest** one possible.

Return *the resulting string*. If there is no way to replace a character to make it not a palindrome, return an **empty string**.

A string `a` is lexicographically smaller than a string `b` (of the same length) if in the first position where `a` and `b` differ, `a` has a character strictly smaller than the corresponding character in `b`. For example, "abcc" is lexicographically smaller than "abcd" because the first position they differ is at the fourth character, and '`c`' is smaller than '`d`'.

**Example 1:**

**Input:** `palindrome` = "abccba"

**Output:** "aaccba"

**Explanation:** There are many ways to make "abccba" not a palindrome, such as "zbccba", "aaccba", and "abacba".

Of all the ways, "aaccba" is the lexicographically smallest.

**Example 2:**

**Input:** palindrome = "a"

**Output:** ""

**Explanation:** There is no way to replace a single character to make "a" not a palindrome, so return an empty string.

**Constraints:**

- $1 \leq \text{palindrome.length} \leq 1000$
- `palindrome` consists of only lowercase English letters.

## 1329. Sort the Matrix Diagonally

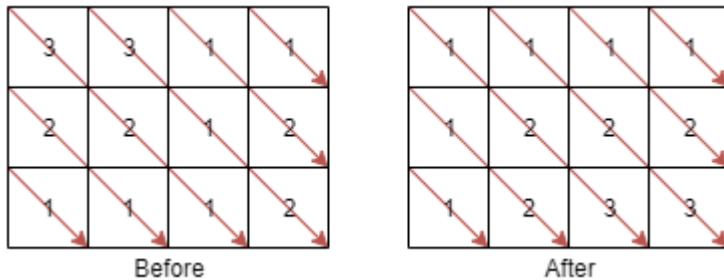
Medium

2839209Add to ListShare

A **matrix diagonal** is a diagonal line of cells starting from some cell in either the topmost row or leftmost column and going in the bottom-right direction until reaching the matrix's end. For example, the **matrix diagonal** starting from `mat[2][0]`, where `mat` is a  $6 \times 3$  matrix, includes cells `mat[2][0]`, `mat[3][1]`, and `mat[4][2]`.

Given an  $m \times n$  matrix `mat` of integers, sort each **matrix diagonal** in ascending order and return *the resulting matrix*.

**Example 1:**



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

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

**Example 2:**

**Input:** mat =  
`[[11,25,66,1,69,7],[23,55,17,45,15,52],[75,31,36,44,58,8],[22,27,33,25,68,4],[84,28,14,11,5,50]]`

**Output:**

```
[[5,17,4,1,52,7],[11,11,25,45,8,69],[14,23,25,44,58,15],[22,27,31,36,50,66],[84,28,75,33,55,68]]
```

**Constraints:**

- $m == \text{mat.length}$
- $n == \text{mat}[i].length$
- $1 \leq m, n \leq 100$
- $1 \leq \text{mat}[i][j] \leq 100$

**1330. Reverse Subarray To Maximize Array Value****Hard**

37335Add to ListShare

You are given an integer array `nums`. The *value* of this array is defined as the sum of  $|\text{nums}[i] - \text{nums}[i + 1]|$  for all  $0 \leq i < \text{nums.length} - 1$ .

You are allowed to select any subarray of the given array and reverse it. You can perform this operation **only once**.

Find maximum possible value of the final array.

**Example 1:****Input:** `nums = [2,3,1,5,4]`**Output:** 10

**Explanation:** By reversing the subarray `[3,1,5]` the array becomes `[2,5,1,3,4]` whose value is 10.

**Example 2:****Input:** `nums = [2,4,9,24,2,1,10]`**Output:** 68**Constraints:**

- $1 \leq \text{nums.length} \leq 3 * 10^4$
- $-10^5 \leq \text{nums}[i] \leq 10^5$

**1331. Rank Transform of an Array**

**Easy**

115061Add to ListShare

Given an array of integers `arr`, replace each element with its rank.

The rank represents how large the element is. The rank has the following rules:

- Rank is an integer starting from 1.
- The larger the element, the larger the rank. If two elements are equal, their rank must be the same.
- Rank should be as small as possible.

**Example 1:****Input:** arr = [40,10,20,30]**Output:** [4,1,2,3]**Explanation:** 40 is the largest element. 10 is the smallest. 20 is the second smallest. 30 is the third smallest.**Example 2:****Input:** arr = [100,100,100]**Output:** [1,1,1]**Explanation:** Same elements share the same rank.**Example 3:****Input:** arr = [37,12,28,9,100,56,80,5,12]**Output:** [5,3,4,2,8,6,7,1,3]**Constraints:**

- $0 \leq \text{arr.length} \leq 10^5$
- $-10^9 \leq \text{arr}[i] \leq 10^9$

**1332. Remove Palindromic Subsequences****Easy**

13761502Add to ListShare

You are given a string `s` consisting **only** of letters '`a`' and '`b`'. In a single step you can remove one **palindromic subsequence** from `s`.

Return *the minimum number of steps to make the given string empty*.

A string is a **subsequence** of a given string if it is generated by deleting some characters of a given string without changing its order. Note that a subsequence does **not** necessarily need to be contiguous.

A string is called **palindrome** if it reads the same backward as well as forward.

### Example 1:

**Input:** `s = "ababa"`

**Output:** 1

**Explanation:** `s` is already a palindrome, so its entirety can be removed in a single step.

### Example 2:

**Input:** `s = "abb"`

**Output:** 2

**Explanation:** `"abb" -> "bb" -> "".`

Remove palindromic subsequence "a" then "bb".

### Example 3:

**Input:** `s = "baabb"`

**Output:** 2

**Explanation:** `"baabb" -> "b" -> "".`

Remove palindromic subsequence "baab" then "b".

### Constraints:

- `1 <= s.length <= 1000`
- `s[i]` is either 'a' or 'b'.

## 1334. Find the City With the Smallest Number of Neighbors at a Threshold Distance

Medium

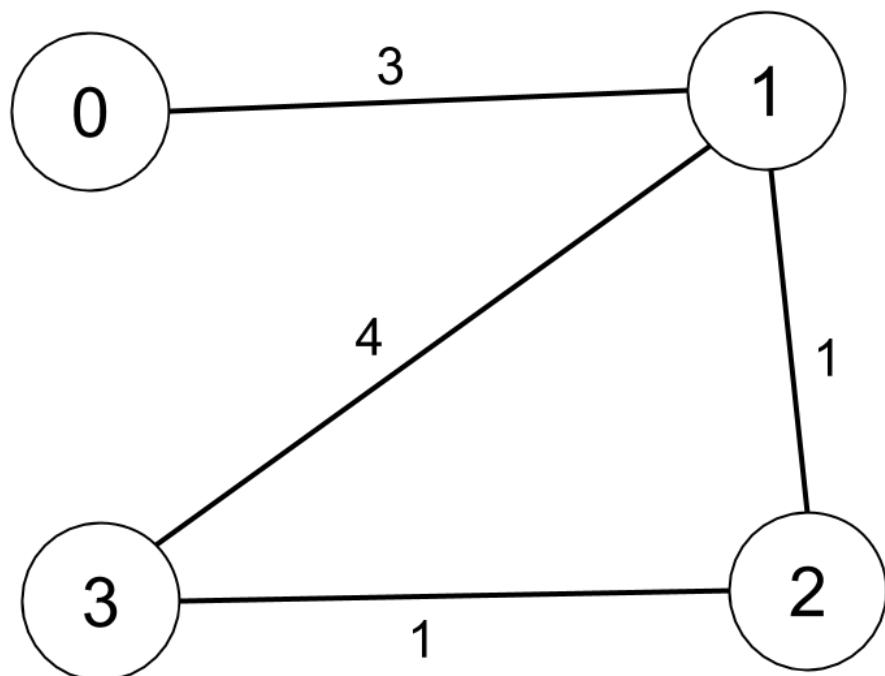
150063Add to ListShare

There are  $n$  cities numbered from 0 to  $n-1$ . Given the array `edges` where `edges[i] = [fromi, toi, weighti]` represents a bidirectional and weighted edge between cities `fromi` and `toi`, and given the integer `distanceThreshold`.

Return the city with the smallest number of cities that are reachable through some path and whose distance is **at most** `distanceThreshold`. If there are multiple such cities, return the city with the greatest number.

Notice that the distance of a path connecting cities  $i$  and  $j$  is equal to the sum of the edges' weights along that path.

**Example 1:**



**Input:**  $n = 4$ , `edges` =  $[[0,1,3],[1,2,1],[1,3,4],[2,3,1]]$ , `distanceThreshold` = 4

**Output:** 3

**Explanation:** The figure above describes the graph.

The neighboring cities at a `distanceThreshold` = 4 for each city are:

City 0 -> [City 1, City 2]

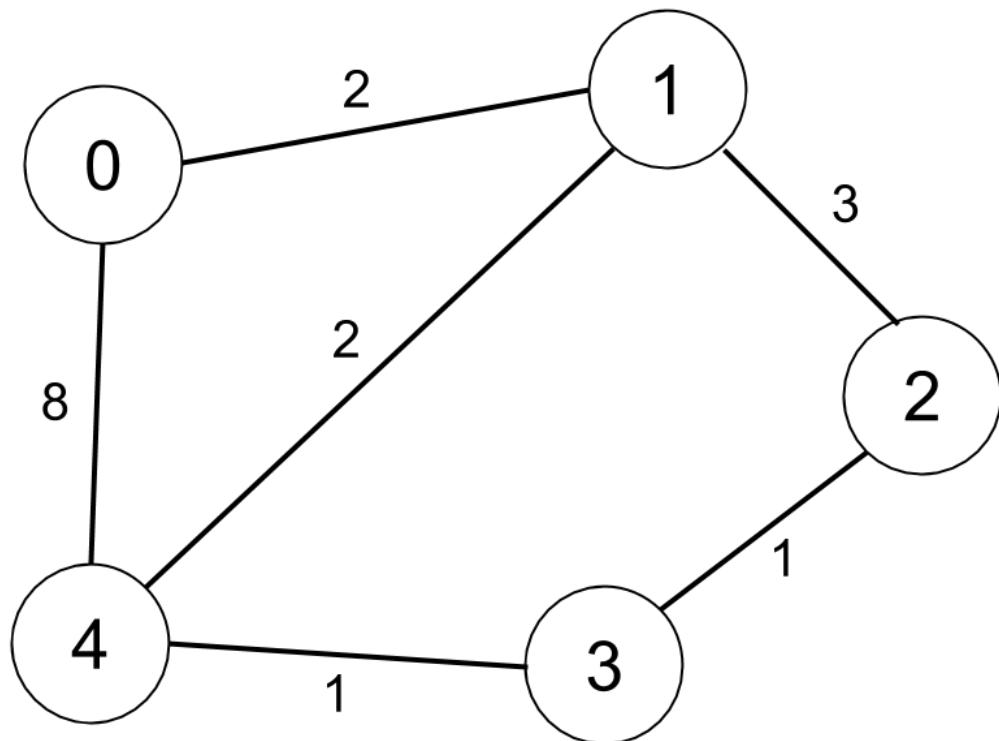
City 1 -> [City 0, City 2, City 3]

City 2 -> [City 0, City 1, City 3]

City 3 -> [City 1, City 2]

Cities 0 and 3 have 2 neighboring cities at a `distanceThreshold = 4`, but we have to return city 3 since it has the greatest number.

**Example 2:**



**Input:** `n = 5, edges = [[0,1,2],[0,4,8],[1,2,3],[1,4,2],[2,3,1],[3,4,1]]`, `distanceThreshold = 2`

**Output:** 0

**Explanation:** The figure above describes the graph.

The neighboring cities at a `distanceThreshold = 2` for each city are:

City 0 -> [City 1]

```
City 1 -> [City 0, City 4]
```

```
City 2 -> [City 3, City 4]
```

```
City 3 -> [City 2, City 4]
```

```
City 4 -> [City 1, City 2, City 3]
```

```
The city 0 has 1 neighboring city at a distanceThreshold = 2.
```

### Constraints:

- $2 \leq n \leq 100$
- $1 \leq \text{edges.length} \leq n * (n - 1) / 2$
- $\text{edges}[i].length == 3$
- $0 \leq \text{from}_i < \text{to}_i < n$
- $1 \leq \text{weight}_i, \text{distanceThreshold} \leq 10^4$
- All pairs  $(\text{from}_i, \text{to}_i)$  are distinct.

## 1335. Minimum Difficulty of a Job Schedule

Hard

1439165Add to ListShare

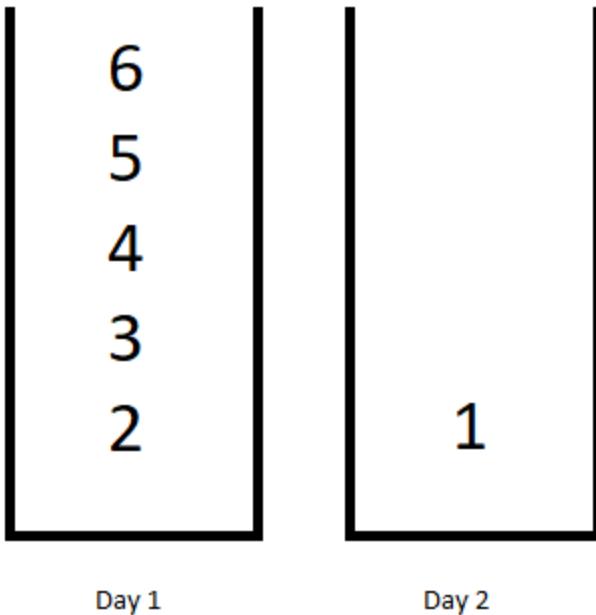
You want to schedule a list of jobs in  $d$  days. Jobs are dependent (i.e To work on the  $i^{\text{th}}$  job, you have to finish all the jobs  $j$  where  $0 \leq j < i$ ).

You have to finish **at least** one task every day. The difficulty of a job schedule is the sum of difficulties of each day of the  $d$  days. The difficulty of a day is the maximum difficulty of a job done on that day.

You are given an integer array `jobDifficulty` and an integer  $d$ . The difficulty of the  $i^{\text{th}}$  job is `jobDifficulty[i]`.

Return *the minimum difficulty of a job schedule*. If you cannot find a schedule for the jobs return `-1`.

### Example 1:



**Input:** jobDifficulty = [6,5,4,3,2,1], d = 2

**Output:** 7

**Explanation:** First day you can finish the first 5 jobs, total difficulty = 6.

Second day you can finish the last job, total difficulty = 1.

The difficulty of the schedule =  $6 + 1 = 7$

**Example 2:**

**Input:** jobDifficulty = [9,9,9], d = 4

**Output:** -1

**Explanation:** If you finish a job per day you will still have a free day. you cannot find a schedule for the given jobs.

**Example 3:**

**Input:** jobDifficulty = [1,1,1], d = 3

**Output:** 3

**Explanation:** The schedule is one job per day. total difficulty will be 3.

**Constraints:**

- $1 \leq \text{jobDifficulty.length} \leq 300$
- $0 \leq \text{jobDifficulty}[i] \leq 1000$
- $1 \leq d \leq 10$

## 1337. The K Weakest Rows in a Matrix

Easy

2783162Add to ListShare

You are given an  $m \times n$  binary matrix  $\text{mat}$  of 1's (representing soldiers) and 0's (representing civilians). The soldiers are positioned **in front** of the civilians. That is, all the 1's will appear to the **left** of all the 0's in each row.

A row  $i$  is **weaker** than a row  $j$  if one of the following is true:

- The number of soldiers in row  $i$  is less than the number of soldiers in row  $j$ .
- Both rows have the same number of soldiers and  $i < j$ .

Return *the indices of the  $k$  weakest rows in the matrix ordered from weakest to strongest*.

**Example 1:**

**Input:**  $\text{mat} =$

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

$k = 3$

**Output:** [2,0,3]

**Explanation:**

The number of soldiers in each row is:

- Row 0: 2
- Row 1: 4
- Row 2: 1
- Row 3: 2

- Row 4: 5

The rows ordered from weakest to strongest are [2,0,3,1,4].

**Example 2:**

**Input:** mat =  
 [[1,0,0,0],  
 [1,1,1,1],  
 [1,0,0,0],  
 [1,0,0,0]],  
 k = 2

**Output:** [0,2]

**Explanation:**

The number of soldiers in each row is:

- Row 0: 1
- Row 1: 4
- Row 2: 1
- Row 3: 1

The rows ordered from weakest to strongest are [0,2,3,1].

**Constraints:**

- `m == mat.length`
- `n == mat[i].length`
- `2 <= n, m <= 100`
- `1 <= k <= m`
- `matrix[i][j]` is either 0 or 1.

## 1338. Reduce Array Size to The Half

**Medium**

2771134Add to ListShare

You are given an integer array `arr`. You can choose a set of integers and remove all the occurrences of these integers in the array.

Return the minimum size of the set so that **at least** half of the integers of the array are removed.

**Example 1:****Input:** arr = [3,3,3,3,5,5,5,2,2,7]**Output:** 2**Explanation:** Choosing {3,7} will make the new array [5,5,5,2,2] which has size 5 (i.e equal to half of the size of the old array).

Possible sets of size 2 are {3,5},{3,2},{5,2}.

Choosing set {2,7} is not possible as it will make the new array [3,3,3,3,5,5,5] which has a size greater than half of the size of the old array.

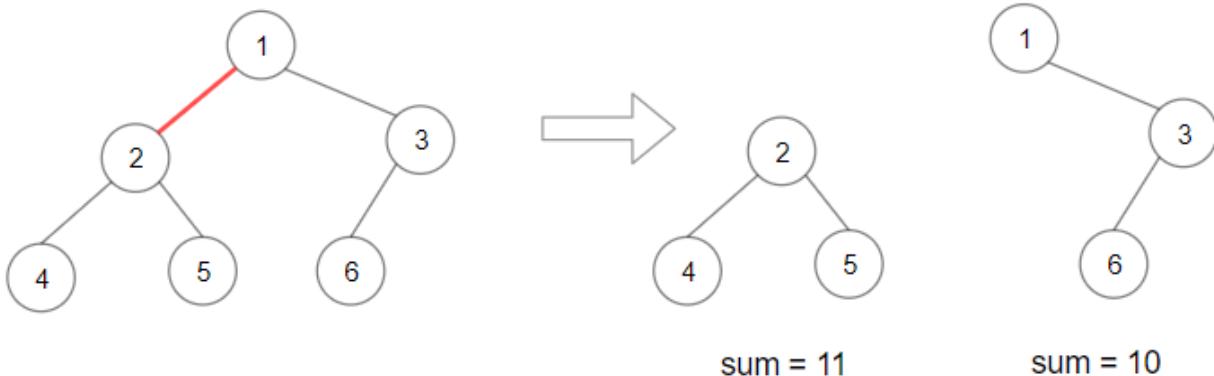
**Example 2:****Input:** arr = [7,7,7,7,7,7]**Output:** 1**Explanation:** The only possible set you can choose is {7}. This will make the new array empty.**Constraints:**

- $2 \leq \text{arr.length} \leq 10^5$
- arr.length is even.
- $1 \leq \text{arr}[i] \leq 10^5$

**1339. Maximum Product of Splitted Binary Tree****Medium**

145764Add to ListShare

Given the `root` of a binary tree, split the binary tree into two subtrees by removing one edge such that the product of the sums of the subtrees is maximized.Return the maximum product of the sums of the two subtrees. Since the answer may be too large, return it **modulo**  $10^9 + 7$ .**Note** that you need to maximize the answer before taking the mod and not after taking it.**Example 1:**

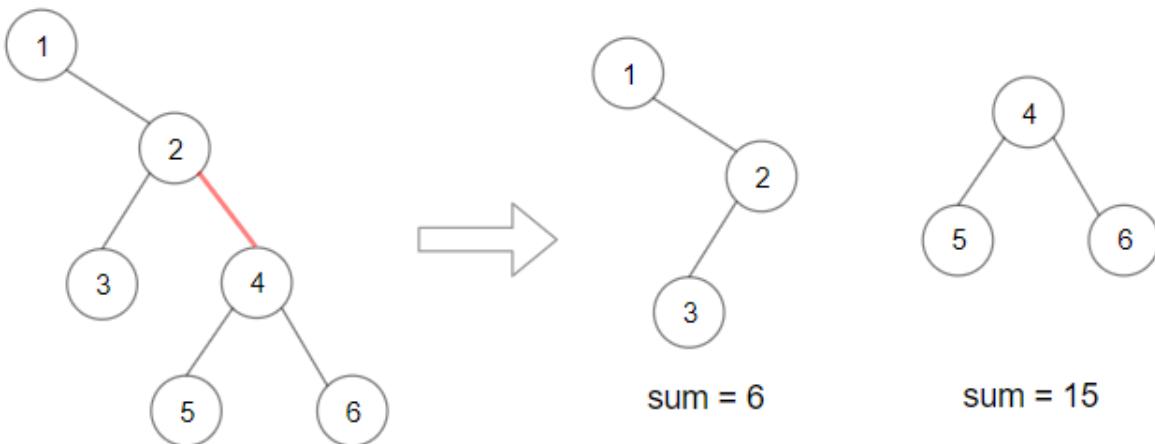


**Input:** root = [1,2,3,4,5,6]

**Output:** 110

**Explanation:** Remove the red edge and get 2 binary trees with sum 11 and 10. Their product is 110 (11\*10)

**Example 2:**



**Input:** root = [1,null,2,3,4,null,null,5,6]

**Output:** 90

**Explanation:** Remove the red edge and get 2 binary trees with sum 15 and 6. Their product is 90 (15\*6)

**Constraints:**

- The number of nodes in the tree is in the range  $[2, 5 * 10^4]$ .
- $1 \leq \text{Node.val} \leq 10^4$

## 1340. Jump Game V

Hard

81928Add to ListShare

Given an array of integers `arr` and an integer `d`. In one step you can jump from index `i` to index:

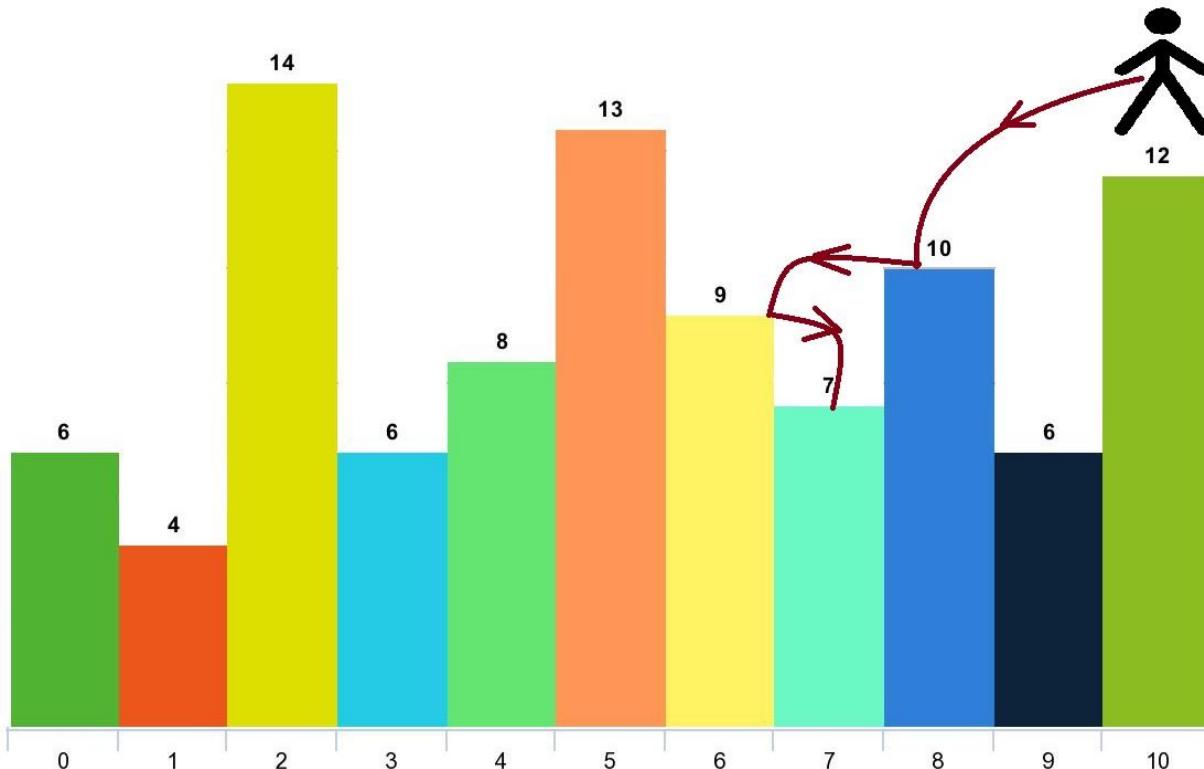
- `i + x` where: `i + x < arr.length` and `0 < x <= d`.
- `i - x` where: `i - x >= 0` and `0 < x <= d`.

In addition, you can only jump from index `i` to index `j` if `arr[i] > arr[j]` and `arr[i] > arr[k]` for all indices `k` between `i` and `j` (More formally `min(i, j) < k < max(i, j)`).

You can choose any index of the array and start jumping. Return *the maximum number of indices* you can visit.

Notice that you can not jump outside of the array at any time.

**Example 1:**



**Input:** `arr = [6,4,14,6,8,13,9,7,10,6,12], d = 2`

**Output:** 4

**Explanation:** You can start at index 10. You can jump 10 --> 8 --> 6 --> 7 as shown.

Note that if you start at index 6 you can only jump to index 7. You cannot jump to index 5 because  $13 > 9$ . You cannot jump to index 4 because index 5 is between index 4 and 6 and  $13 > 9$ .

Similarly You cannot jump from index 3 to index 2 or index 1.

**Example 2:**

**Input:** arr = [3,3,3,3,3], d = 3

**Output:** 1

**Explanation:** You can start at any index. You always cannot jump to any index.

**Example 3:**

**Input:** arr = [7,6,5,4,3,2,1], d = 1

**Output:** 7

**Explanation:** Start at index 0. You can visit all the indicies.

**Constraints:**

- $1 \leq \text{arr.length} \leq 1000$
- $1 \leq \text{arr}[i] \leq 10^5$
- $1 \leq d \leq \text{arr.length}$

## 1342. Number of Steps to Reduce a Number to Zero

Easy

2751136Add to ListShare

Given an integer `num`, return *the number of steps to reduce it to zero*.

In one step, if the current number is even, you have to divide it by 2, otherwise, you have to subtract 1 from it.

**Example 1:**

**Input:** num = 14

**Output:** 6

**Explanation:**

Step 1) 14 is even; divide by 2 and obtain 7.  
 Step 2) 7 is odd; subtract 1 and obtain 6.  
 Step 3) 6 is even; divide by 2 and obtain 3.  
 Step 4) 3 is odd; subtract 1 and obtain 2.  
 Step 5) 2 is even; divide by 2 and obtain 1.  
 Step 6) 1 is odd; subtract 1 and obtain 0.

**Example 2:**

**Input:** num = 8

**Output:** 4

**Explanation:**

Step 1) 8 is even; divide by 2 and obtain 4.  
 Step 2) 4 is even; divide by 2 and obtain 2.  
 Step 3) 2 is even; divide by 2 and obtain 1.  
 Step 4) 1 is odd; subtract 1 and obtain 0.

**Example 3:**

**Input:** num = 123

**Output:** 12

**Constraints:**

- $0 \leq \text{num} \leq 10^6$

### 1343. Number of Sub-arrays of Size K and Average Greater than or Equal to Threshold

Medium

80169Add to ListShare

Given an array of integers `arr` and two integers `k` and `threshold`, return *the number of sub-arrays of size `k` and average greater than or equal to `threshold`*.

**Example 1:**

**Input:** arr = [2,2,2,2,5,5,5,8], k = 3, threshold = 4

**Output:** 3

**Explanation:** Sub-arrays [2,5,5],[5,5,5] and [5,5,8] have averages 4, 5 and 6 respectively. All other sub-arrays of size 3 have averages less than 4 (the threshold).

**Example 2:**

**Input:** arr = [11,13,17,23,29,31,7,5,2,3], k = 3, threshold = 5

**Output:** 6

**Explanation:** The first 6 sub-arrays of size 3 have averages greater than 5. Note that averages are not integers.

**Constraints:**

- $1 \leq \text{arr.length} \leq 10^5$
- $1 \leq \text{arr}[i] \leq 10^4$
- $1 \leq k \leq \text{arr.length}$
- $0 \leq \text{threshold} \leq 10^4$

## 1344. Angle Between Hands of a Clock

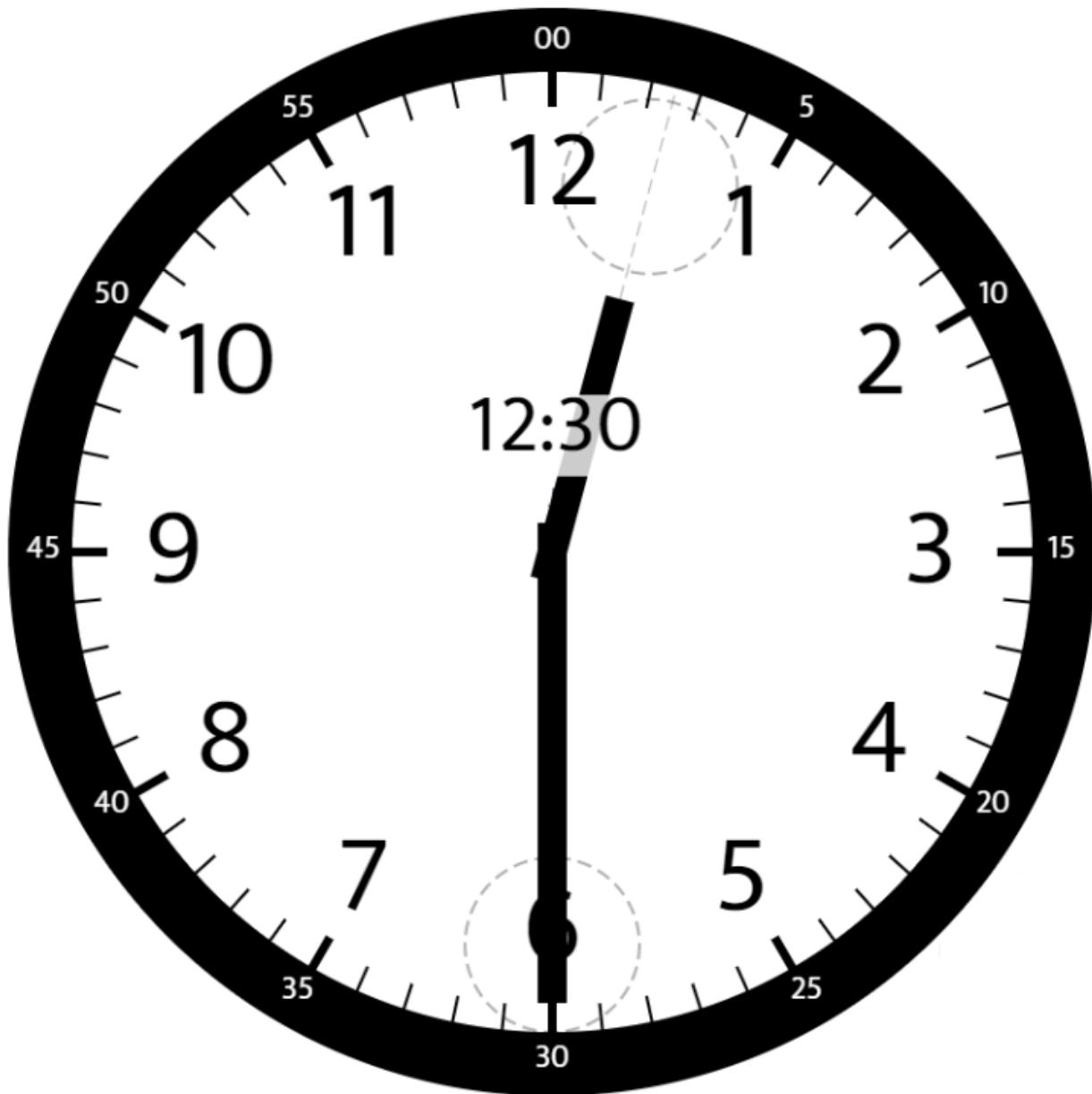
Medium

1052217 Add to List Share

Given two numbers, `hour` and `minutes`, return the smaller angle (in degrees) formed between the `hour` and the `minute` hand.

Answers within  $10^{-5}$  of the actual value will be accepted as correct.

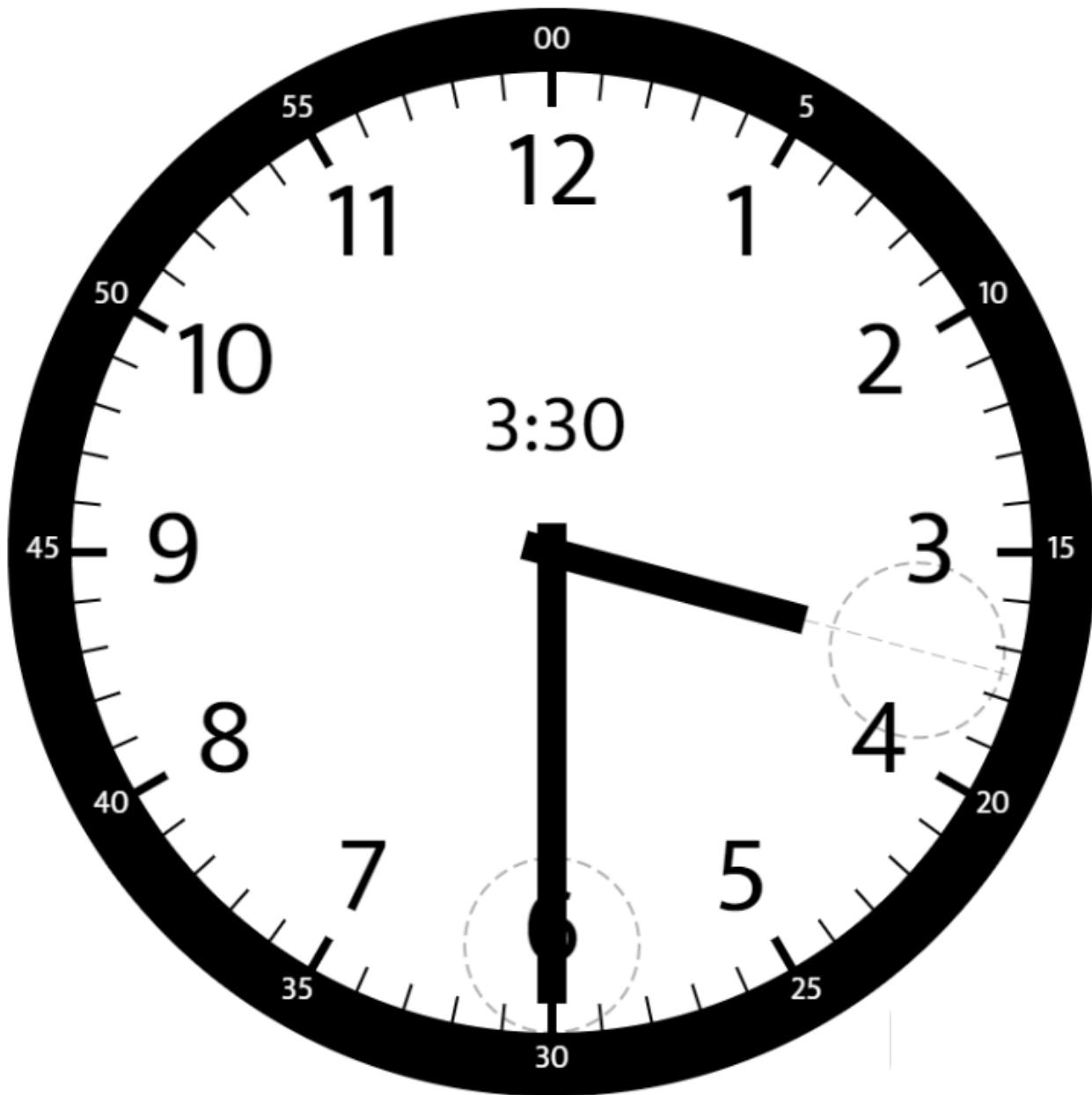
**Example 1:**



**Input:** hour = 12, minutes = 30

**Output:** 165

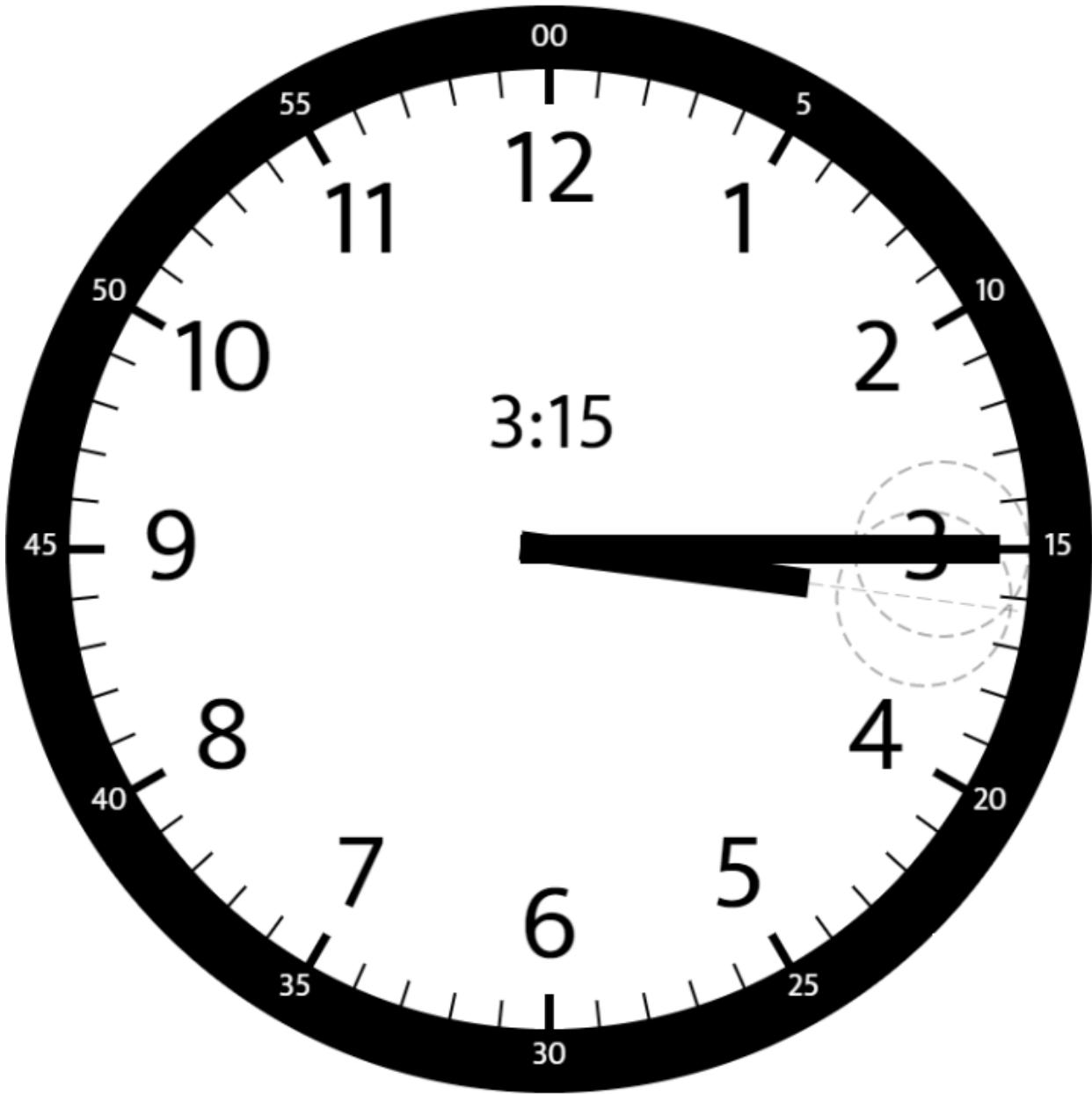
**Example 2:**



**Input:** hour = 3, minutes = 30

**Output:** 75

**Example 3:**



**Input:** hour = 3, minutes = 15

**Output:** 7.5

**Constraints:**

- $1 \leq \text{hour} \leq 12$
- $0 \leq \text{minutes} \leq 59$

**1345. Jump Game IV**

**Hard**

208182Add to ListShare

Given an array of integers `arr`, you are initially positioned at the first index of the array.In one step you can jump from index `i` to index:

- `i + 1` where:  $i + 1 < arr.length$ .
- `i - 1` where:  $i - 1 \geq 0$ .
- `j` where:  $arr[i] == arr[j]$  and  $i \neq j$ .

Return *the minimum number of steps* to reach the **last index** of the array.

Notice that you can not jump outside of the array at any time.

**Example 1:****Input:** `arr = [100, -23, -23, 404, 100, 23, 23, 23, 3, 404]`**Output:** 3**Explanation:** You need three jumps from index 0  $\rightarrow$  4  $\rightarrow$  3  $\rightarrow$  9. Note that index 9 is the last index of the array.**Example 2:****Input:** `arr = [7]`**Output:** 0**Explanation:** Start index is the last index. You do not need to jump.**Example 3:****Input:** `arr = [7, 6, 9, 6, 9, 6, 9, 7]`**Output:** 1**Explanation:** You can jump directly from index 0 to index 7 which is last index of the array.**Constraints:**

- $1 \leq arr.length \leq 5 * 10^4$
- $-10^8 \leq arr[i] \leq 10^8$

**1346. Check If N and Its Double Exist**

**Easy**

1239145Add to ListShare

Given an array `arr` of integers, check if there exist two indices `i` and `j` such that :

- `i != j`
- `0 <= i, j < arr.length`
- `arr[i] == 2 * arr[j]`

**Example 1:****Input:** `arr = [10,2,5,3]`**Output:** `true`**Explanation:** For  $i = 0$  and  $j = 2$ ,  $arr[i] == 10 == 2 * 5 == 2 * arr[j]$ **Example 2:****Input:** `arr = [3,1,7,11]`**Output:** `false`**Explanation:** There is no  $i$  and  $j$  that satisfy the conditions.**Constraints:**

- $2 \leq arr.length \leq 500$
- $-10^3 \leq arr[i] \leq 10^3$

**1347. Minimum Number of Steps to Make Two Strings Anagram****Medium**

145968Add to ListShare

You are given two strings of the same length `s` and `t`. In one step you can choose **any character** of `t` and replace it with **another character**.Return *the minimum number of steps* to make `t` an anagram of `s`.An **Anagram** of a string is a string that contains the same characters with a different (or the same) ordering.**Example 1:**

**Input:** s = "bab", t = "aba"

**Output:** 1

**Explanation:** Replace the first 'a' in t with b, t = "bba" which is anagram of s.

**Example 2:**

**Input:** s = "leetcode", t = "practice"

**Output:** 5

**Explanation:** Replace 'p', 'r', 'a', 'i' and 'c' from t with proper characters to make t anagram of s.

**Example 3:**

**Input:** s = "anagram", t = "mangaar"

**Output:** 0

**Explanation:** "anagram" and "mangaar" are anagrams.

**Constraints:**

- $1 \leq s.length \leq 5 * 10^4$
- $s.length == t.length$
- $s$  and  $t$  consist of lowercase English letters only.

## 1348. Tweet Counts Per Frequency

Medium

128228Add to ListShare

A social media company is trying to monitor activity on their site by analyzing the number of tweets that occur in select periods of time. These periods can be partitioned into smaller **time chunks** based on a certain frequency (every **minute**, **hour**, or **day**).

For example, the period `[10, 10000]` (in **seconds**) would be partitioned into the following **time chunks** with these frequencies:

- Every **minute** (60-second chunks): `[10, 69], [70, 129], [130, 189], ..., [9970, 10000]`
- Every **hour** (3600-second chunks): `[10, 3609], [3610, 7209], [7210, 10000]`
- Every **day** (86400-second chunks): `[10, 10000]`

Notice that the last chunk may be shorter than the specified frequency's chunk size and will always end with the end time of the period (`10000` in the above example).

Design and implement an API to help the company with their analysis.

Implement the `TweetCounts` class:

- `TweetCounts()` Initializes the `TweetCounts` object.
- `void recordTweet(String tweetName, int time)` Stores the `tweetName` at the recorded time (in seconds).
- `List<Integer> getTweetCountsPerFrequency(String freq, String tweetName, int startTime, int endTime)` Returns a list of integers representing the number of tweets with `tweetName` in each time chunk for the given period of time `[startTime, endTime]` (in seconds) and frequency `freq`.
  - `freq` is one of "minute", "hour", or "day" representing a frequency of every minute, hour, or day respectively.

### Example:

#### Input

```
["TweetCounts", "recordTweet", "recordTweet", "recordTweet", "getTweetCountsPerFrequency", "getTweetCountsPerFrequency", "recordTweet", "getTweetCountsPerFrequency"]
[[[], ["tweet3", 0], ["tweet3", 60], ["tweet3", 10], ["minute", "tweet3", 0, 59], ["minute", "tweet3", 0, 60], ["tweet3", 120], ["hour", "tweet3", 0, 210]]
```

#### Output

```
[null, null, null, null, [2], [2, 1], null, [4]]
```

#### Explanation

```
TweetCounts tweetCounts = new TweetCounts();

tweetCounts.recordTweet("tweet3", 0); // New tweet
"tweet3" at time 0

tweetCounts.recordTweet("tweet3", 60); // New tweet
"tweet3" at time 60

tweetCounts.recordTweet("tweet3", 10); // New tweet
"tweet3" at time 10

tweetCounts.getTweetCountsPerFrequency("minute", "tweet3", 0, 59); // return [2];
chunk [0,59] had 2 tweets

tweetCounts.getTweetCountsPerFrequency("minute", "tweet3", 0, 60); // return [2,1];
chunk [0,59] had 2 tweets, chunk [60,60] had 1 tweet
```

```

tweetCounts.recordTweet("tweet3", 120);           // New tweet
"tweet3" at time 120

tweetCounts.getTweetCountsPerFrequency("hour", "tweet3", 0, 210); // return [4];
chunk [0,210] had 4 tweets

```

### Constraints:

- $0 \leq \text{time}, \text{startTime}, \text{endTime} \leq 10^9$
- $0 \leq \text{endTime} - \text{startTime} \leq 10^4$
- There will be at most  $10^4$  calls in **total** to `recordTweet` and `getTweetCountsPerFrequency`.

## 1349. Maximum Students Taking Exam

### Hard

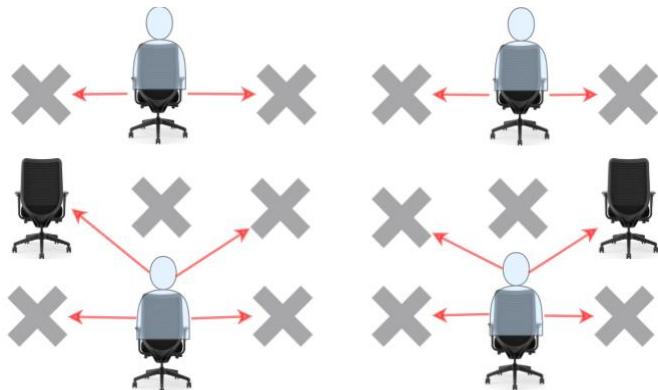
64312Add to ListShare

Given a  $m * n$  matrix `seats` that represent seats distributions in a classroom. If a seat is broken, it is denoted by '#' character otherwise it is denoted by '.' character.

Students can see the answers of those sitting next to the left, right, upper left and upper right, but he cannot see the answers of the student sitting directly in front or behind him. Return the **maximum** number of students that can take the exam together without any cheating being possible..

Students must be placed in seats in good condition.

### Example 1:



```

Input: seats = [["#",".","#","#",".","#"],
                [".","#","#","#","#","."],
                ["#",".","#","#",".","#"]]]

```

**Output:** 4

**Explanation:** Teacher can place 4 students in available seats so they don't cheat on the exam.

**Example 2:**

**Input:** seats = [[".","#"],  
["#", "#"],  
["#", "."],  
["#", "#"],  
[".", "#"]]

**Output:** 3

**Explanation:** Place all students in available seats.

**Example 3:**

**Input:** seats = [[ "#", ".", ".", ".", "#"],  
[".", "#", ".", "#", "."],  
[".", ".", "#", ".", "."],  
[".", "#", ".", "#", "."],  
["#", ".", ".", ".", "#"]]

**Output:** 10

**Explanation:** Place students in available seats in column 1, 3 and 5.

**Constraints:**

- `seats` contains only characters `'.'` and `'#'`.
- `m == seats.length`
- `n == seats[i].length`
- `1 <= m <= 8`
- `1 <= n <= 8`

## 1351. Count Negative Numbers in a Sorted Matrix

Easy

270282Add to ListShare

Given a  $m \times n$  matrix `grid` which is sorted in non-increasing order both row-wise and column-wise, return *the number of negative numbers in `grid`*.

**Example 1:**

**Input:** `grid = [[4,3,2,-1],[3,2,1,-1],[1,1,-1,-2],[-1,-1,-2,-3]]`

**Output:** 8

**Explanation:** There are 8 negative numbers in the matrix.

**Example 2:**

**Input:** `grid = [[3,2],[1,0]]`

**Output:** 0

**Constraints:**

- $m == \text{grid.length}$
- $n == \text{grid[i].length}$
- $1 \leq m, n \leq 100$
- $-100 \leq \text{grid[i][j]} \leq 100$

## 1352. Product of the Last K Numbers

Medium

107156Add to ListShare

Design an algorithm that accepts a stream of integers and retrieves the product of the last  $k$  integers of the stream.

Implement the `ProductOfNumbers` class:

- `ProductOfNumbers()` Initializes the object with an empty stream.
- `void add(int num)` Appends the integer `num` to the stream.
- `int getProduct(int k)` Returns the product of the last  $k$  numbers in the current list. You can assume that always the current list has at least  $k$  numbers.

The test cases are generated so that, at any time, the product of any contiguous sequence of numbers will fit into a single 32-bit integer without overflowing.

**Example:****Input**

```
["ProductOfNumbers", "add", "add", "add", "add", "add", "getProduct", "getProduct", "getProduct", "add", "getProduct"]
[[[],[3],[0],[2],[5],[4],[2],[3],[4],[8],[2]]]
```

**Output**

```
[null,null,null,null,null,null,20,40,0,null,32]
```

**Explanation**

```
ProductOfNumbers productOfNumbers = new ProductOfNumbers();

productOfNumbers.add(3);           // [3]
productOfNumbers.add(0);          // [3,0]
productOfNumbers.add(2);          // [3,0,2]
productOfNumbers.add(5);          // [3,0,2,5]
productOfNumbers.add(4);          // [3,0,2,5,4]

productOfNumbers.getProduct(2); // return 20. The product of the last 2 numbers is 5
* 4 = 20

productOfNumbers.getProduct(3); // return 40. The product of the last 3 numbers is 2
* 5 * 4 = 40

productOfNumbers.getProduct(4); // return 0. The product of the last 4 numbers is 0 *
2 * 5 * 4 = 0

productOfNumbers.add(8);          // [3,0,2,5,4,8]

productOfNumbers.getProduct(2); // return 32. The product of the last 2 numbers is 4
* 8 = 32
```

**Constraints:**

- $0 \leq \text{num} \leq 100$
- $1 \leq k \leq 4 * 10^4$
- At most  $4 * 10^4$  calls will be made to `add` and `getProduct`.
- The product of the stream at any point in time will fit in a **32-bit** integer.

### 1353. Maximum Number of Events That Can Be Attended

Medium

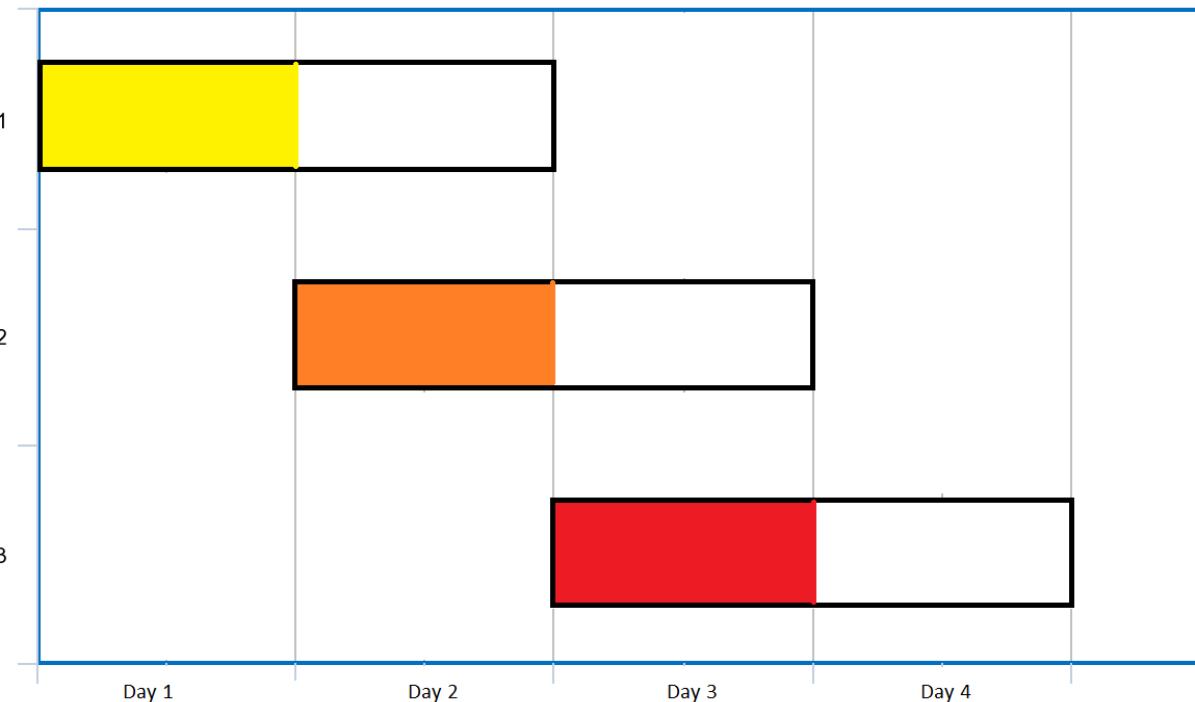
2112284Add to ListShare

You are given an array of `events` where `events[i] = [startDayi, endDayi]`. Every event `i` starts at `startDayi` and ends at `endDayi`.

You can attend an event `i` at any day `d` where `startTimei <= d <= endTimei`. You can only attend one event at any time `d`.

Return *the maximum number of events you can attend*.

**Example 1:**



**Input:** `events = [[1,2],[2,3],[3,4]]`

**Output:** 3

**Explanation:** You can attend all the three events.

One way to attend them all is as shown.

Attend the first event on day 1.

Attend the second event on day 2.

Attend the third event on day 3.

**Example 2:**

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

**Output:** 4

**Constraints:**

- $1 \leq \text{events.length} \leq 10^5$
- $\text{events}[i].length == 2$
- $1 \leq \text{startDay}_i \leq \text{endDay}_i \leq 10^5$

## 1354. Construct Target Array With Multiple Sums

Hard

1924155Add to ListShare

You are given an array `target` of  $n$  integers. From a starting array `arr` consisting of  $n$  1's, you may perform the following procedure :

- let  $x$  be the sum of all elements currently in your array.
- choose index  $i$ , such that  $0 \leq i < n$  and set the value of `arr` at index  $i$  to  $x$ .
- You may repeat this procedure as many times as needed.

Return `true` if it is possible to construct the `target` array from `arr`, otherwise, return `false`.

**Example 1:**

**Input:** target = [9,3,5]

**Output:** true

**Explanation:** Start with `arr` = [1, 1, 1]

[1, 1, 1], sum = 3 choose index 1

[1, 3, 1], sum = 5 choose index 2

[1, 3, 5], sum = 9 choose index 0

[9, 3, 5] Done

**Example 2:**

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

**Output:** false

**Explanation:** Impossible to create target array from [1,1,1,1].

**Example 3:**

**Input:** target = [8,5]

**Output:** true

**Constraints:**

- $n == \text{target.length}$
- $1 \leq n \leq 5 * 10^4$
- $1 \leq \text{target}[i] \leq 10^9$

## 1356. Sort Integers by The Number of 1 Bits

Easy

127756Add to ListShare

You are given an integer array `arr`. Sort the integers in the array in ascending order by the number of 1's in their binary representation and in case of two or more integers have the same number of 1's you have to sort them in ascending order.

Return *the array after sorting it.*

**Example 1:**

**Input:** arr = [0,1,2,3,4,5,6,7,8]

**Output:** [0,1,2,4,8,3,5,6,7]

**Explanation:** [0] is the only integer with 0 bits.

[1,2,4,8] all have 1 bit.

[3,5,6] have 2 bits.

[7] has 3 bits.

The sorted array by bits is [0,1,2,4,8,3,5,6,7]

**Example 2:**

**Input:** arr = [1024,512,256,128,64,32,16,8,4,2,1]

**Output:** [1,2,4,8,16,32,64,128,256,512,1024]

**Explanation:** All integers have 1 bit in the binary representation, you should just sort them in ascending order.

### Constraints:

- $1 \leq \text{arr.length} \leq 500$
- $0 \leq \text{arr}[i] \leq 10^4$

## 1357. Apply Discount Every n Orders

Medium

130178Add to ListShare

There is a supermarket that is frequented by many customers. The products sold at the supermarket are represented as two parallel integer arrays `products` and `prices`, where the  $i^{\text{th}}$  product has an ID of `products[i]` and a price of `prices[i]`.

When a customer is paying, their bill is represented as two parallel integer arrays `product` and `amount`, where the  $j^{\text{th}}$  product they purchased has an ID of `product[j]`, and `amount[j]` is how much of the product they bought. Their subtotal is calculated as the sum of each `amount[j] * (price of the  $j^{\text{th}}$  product)`.

The supermarket decided to have a sale. Every  $n^{\text{th}}$  customer paying for their groceries will be given a **percentage discount**. The discount amount is given by `discount`, where they will be given `discount` percent off their subtotal. More formally, if their subtotal is `bill`, then they would actually pay `bill * ((100 - discount) / 100)`.

Implement the `Cashier` class:

- `Cashier(int n, int discount, int[] products, int[] prices)` Initializes the object with `n`, the `discount`, and the `products` and their `prices`.
- `double getBill(int[] product, int[] amount)` Returns the final total of the bill with the discount applied (if any). Answers within  $10^{-5}$  of the actual value will be accepted.

### Example 1:

Input

```
["Cashier","getBill","getBill","getBill","getBill","getBill","getBill","getBill"]
[[3,50,[1,2,3,4,5,6,7],[100,200,300,400,300,200,100]],[[1,2],[1,2]],[[3,7],[10,10]],[[1,2,3,4,5,6,7],[1,1,1,1,1,1,1]],[[4],[10]],[[7,3],[10,10]],[[7,5,3,1,6,4,2],[10,10,10,9,9,9,7]],[[2,3,5],[5,3,2]]]
```

**Output**

```
[null,500.0,4000.0,800.0,4000.0,4000.0,7350.0,2500.0]
```

**Explanation**

```
Cashier cashier = new Cashier(3,50,[1,2,3,4,5,6,7],[100,200,300,400,300,200,100]);  
  
cashier.getBill([1,2],[1,2]); // return 500.0. 1st customer, no discount.  
  
 // bill = 1 * 100 + 2 * 200 = 500.  
  
cashier.getBill([3,7],[10,10]); // return 4000.0. 2nd customer, no discount.  
  
 // bill = 10 * 300 + 10 * 100 = 4000.  
  
cashier.getBill([1,2,3,4,5,6,7],[1,1,1,1,1,1,1]); // return 800.0. 3rd customer, 50% discount.  
  
 // Original bill = 1600  
  
 // Actual bill = 1600 * ((100 - 50) / 100) = 800.  
  
cashier.getBill([4],[10]); // return 4000.0. 4th customer, no discount.  
  
cashier.getBill([7,3],[10,10]); // return 4000.0. 5th customer, no discount.  
  
cashier.getBill([7,5,3,1,6,4,2],[10,10,10,9,9,9,7]); // return 7350.0. 6th customer, 50% discount.  
  
 // Original bill = 14700, but with  
 // Actual bill = 14700 * ((100 - 50) / 100) = 7350.  
  
cashier.getBill([2,3,5],[5,3,2]); // return 2500.0. 6th customer, no discount.
```

**Constraints:**

- $1 \leq n \leq 10^4$
- $0 \leq \text{discount} \leq 100$
- $1 \leq \text{products.length} \leq 200$

- `prices.length == products.length`
- `1 <= products[i] <= 200`
- `1 <= prices[i] <= 1000`
- The elements in `products` are **unique**.
- `1 <= product.length <= products.length`
- `amount.length == product.length`
- `product[j]` exists in `products`.
- `1 <= amount[j] <= 1000`
- The elements of `product` are **unique**.
- At most `1000` calls will be made to `getBill`.
- Answers within `10-5` of the actual value will be accepted.

## 1358. Number of Substrings Containing All Three Characters

Medium

177429Add to ListShare

Given a string `s` consisting only of characters `a`, `b` and `c`.

Return the number of substrings containing **at least** one occurrence of all these characters `a`, `b` and `c`.

### Example 1:

**Input:** `s = "abcabc"`

**Output:** `10`

**Explanation:** The substrings containing at least one occurrence of the characters `a`, `b` and `c` are `"abc"`, `"abca"`, `"abcab"`, `"abcabc"`, `"bca"`, `"bcab"`, `"bcabc"`, `"cab"`, `"cabc"` and `"abc"` (again).

### Example 2:

**Input:** `s = "aaacb"`

**Output:** `3`

**Explanation:** The substrings containing at least one occurrence of the characters `a`, `b` and `c` are `"aaacb"`, `"aacb"` and `"acb"`.

### Example 3:

**Input:** `s = "abc"`

**Output:** `1`

**Constraints:**

- $3 \leq s.length \leq 5 \times 10^4$
- $s$  only consists of  $a, b$  or  $c$  characters.

**1359. Count All Valid Pickup and Delivery Options****Hard**

1634148Add to ListShare

Given  $n$  orders, each order consist in pickup and delivery services.Count all valid pickup/delivery possible sequences such that delivery( $i$ ) is always after of pickup( $i$ ).Since the answer may be too large, return it modulo  $10^9 + 7$ .**Example 1:****Input:**  $n = 1$ **Output:** 1**Explanation:** Unique order (P1, D1), Delivery 1 always is after of Pickup 1.**Example 2:****Input:**  $n = 2$ **Output:** 6**Explanation:** All possible orders:

(P1,P2,D1,D2), (P1,P2,D2,D1), (P1,D1,P2,D2), (P2,P1,D1,D2), (P2,P1,D2,D1) and (P2,D2,P1,D1).

This is an invalid order (P1,D2,P2,D1) because Pickup 2 is after of Delivery 2.

**Example 3:****Input:**  $n = 3$ **Output:** 90**Constraints:**

- $1 \leq n \leq 500$

## 1360. Number of Days Between Two Dates

Easy

234954Add to ListShare

Write a program to count the number of days between two dates.

The two dates are given as strings, their format is `YYYY-MM-DD` as shown in the examples.

**Example 1:**

**Input:** `date1 = "2019-06-29", date2 = "2019-06-30"`

**Output:** `1`

**Example 2:**

**Input:** `date1 = "2020-01-15", date2 = "2019-12-31"`

**Output:** `15`

**Constraints:**

- The given dates are valid dates between the years `1971` and `2100`.

## 1361. Validate Binary Tree Nodes

Medium

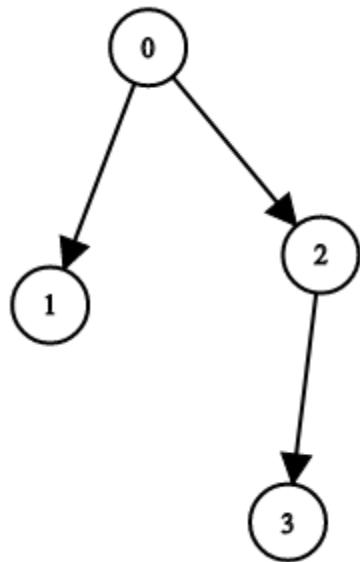
933282Add to ListShare

You have `n` binary tree nodes numbered from `0` to `n - 1` where node `i` has two children `leftChild[i]` and `rightChild[i]`, return `true` if and only if **all** the given nodes form **exactly one** valid binary tree.

If node `i` has no left child then `leftChild[i]` will equal `-1`, similarly for the right child.

Note that the nodes have no values and that we only use the node numbers in this problem.

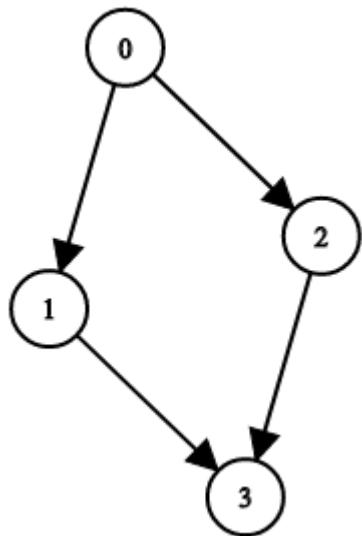
**Example 1:**



**Input:**  $n = 4$ ,  $\text{leftChild} = [1, -1, 3, -1]$ ,  $\text{rightChild} = [2, -1, -1, -1]$

**Output:** true

**Example 2:**



**Input:**  $n = 4$ ,  $\text{leftChild} = [1, -1, 3, -1]$ ,  $\text{rightChild} = [2, 3, -1, -1]$

**Output:** false

**Example 3:**



**Input:** n = 2, leftChild = [1,0], rightChild = [-1,-1]

**Output:** false

#### Constraints:

- $n == \text{leftChild.length} == \text{rightChild.length}$
- $1 \leq n \leq 10^4$
- $-1 \leq \text{leftChild}[i], \text{rightChild}[i] \leq n - 1$

## 1362. Closest Divisors

Medium

22089Add to ListShare

Given an integer `num`, find the closest two integers in absolute difference whose product equals `num + 1` or `num + 2`.

Return the two integers in any order.

#### Example 1:

**Input:** num = 8

**Output:** [3,3]

**Explanation:** For  $\text{num} + 1 = 9$ , the closest divisors are 3 & 3, for  $\text{num} + 2 = 10$ , the closest divisors are 2 & 5, hence 3 & 3 is chosen.

#### Example 2:

**Input:** num = 123

**Output:** [5,25]

#### Example 3:

**Input:** num = 999

**Output:** [40,25]

**Constraints:**

- $1 \leq \text{num} \leq 10^9$

### 1363. Largest Multiple of Three

Hard

46662Add to ListShare

Given an array of digits `digits`, return the largest multiple of **three** that can be formed by concatenating some of the given digits in **any order**. If there is no answer return an empty string.

Since the answer may not fit in an integer data type, return the answer as a string. Note that the returning answer must not contain unnecessary leading zeros.

**Example 1:**

**Input:** digits = [8,1,9]

**Output:** "981"

**Example 2:**

**Input:** digits = [8,6,7,1,0]

**Output:** "8760"

**Example 3:**

**Input:** digits = [1]

**Output:** ""

**Constraints:**

- $1 \leq \text{digits.length} \leq 10^4$
- $0 \leq \text{digits}[i] \leq 9$

### 1365. How Many Numbers Are Smaller Than the Current Number

Easy

383575Add to ListShare

Given the array `nums`, for each `nums[i]` find out how many numbers in the array are smaller than it. That is, for each `nums[i]` you have to count the number of valid `j`'s such that `j != i` and `nums[j] < nums[i]`.

Return the answer in an array.

### Example 1:

**Input:** `nums = [8,1,2,2,3]`

**Output:** `[4,0,1,1,3]`

#### Explanation:

For `nums[0]=8` there exist four smaller numbers than it (1, 2, 2 and 3).

For `nums[1]=1` does not exist any smaller number than it.

For `nums[2]=2` there exist one smaller number than it (1).

For `nums[3]=2` there exist one smaller number than it (1).

For `nums[4]=3` there exist three smaller numbers than it (1, 2 and 2).

### Example 2:

**Input:** `nums = [6,5,4,8]`

**Output:** `[2,1,0,3]`

### Example 3:

**Input:** `nums = [7,7,7,7]`

**Output:** `[0,0,0,0]`

### Constraints:

- `2 <= nums.length <= 500`
- `0 <= nums[i] <= 100`

## 1366. Rank Teams by Votes

Medium

967102Add to ListShare

In a special ranking system, each voter gives a rank from highest to lowest to all teams participated in the competition.

The ordering of teams is decided by who received the most position-one votes. If two or more teams tie in the first position, we consider the second position to resolve the conflict, if they tie again, we continue this process until the ties are resolved. If two or more teams are still tied after considering all positions, we rank them alphabetically based on their team letter.

Given an array of strings `votes` which is the votes of all voters in the ranking systems. Sort all teams according to the ranking system described above.

Return a *string of all teams sorted* by the ranking system.

### Example 1:

**Input:** votes = ["ABC", "ACB", "ABC", "ACB", "ACB"]

**Output:** "ACB"

**Explanation:** Team A was ranked first place by 5 voters. No other team was voted as first place so team A is the first team.

Team B was ranked second by 2 voters and was ranked third by 3 voters.

Team C was ranked second by 3 voters and was ranked third by 2 voters.

As most of the voters ranked C second, team C is the second team and team B is the third.

### Example 2:

**Input:** votes = ["WXYZ", "XYZW"]

**Output:** "XWYZ"

**Explanation:** X is the winner due to tie-breaking rule. X has same votes as W for the first position but X has one vote as second position while W doesn't have any votes as second position.

### Example 3:

**Input:** votes = ["ZMNAGUEDSJYLBOPHRQICWFXTVK"]

**Output:** "ZMNAGUEDSJYLBOPHRQICWFXTVK"

**Explanation:** Only one voter so his votes are used for the ranking.

**Constraints:**

- `1 <= votes.length <= 1000`
- `1 <= votes[i].length <= 26`
- `votes[i].length == votes[j].length` for  $0 \leq i, j < votes.length$ .
- `votes[i][j]` is an English **uppercase** letter.
- All characters of `votes[i]` are unique.
- All the characters that occur in `votes[0]` **also occur** in `votes[j]` where  $1 \leq j < votes.length$ .

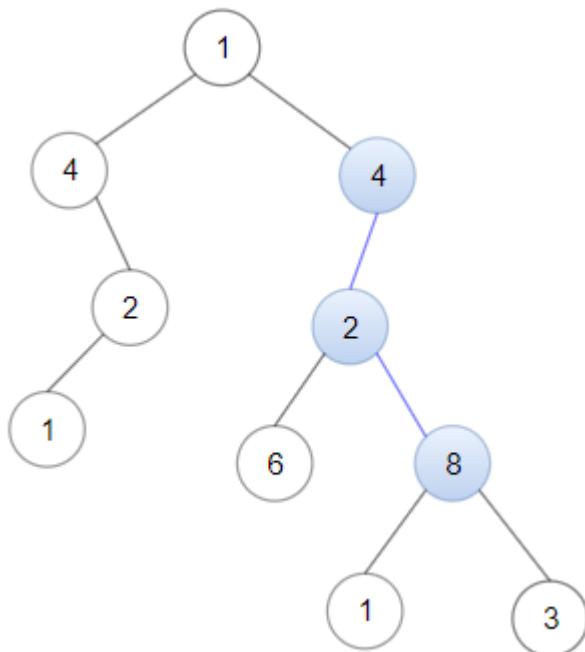
**1367. Linked List in Binary Tree****Medium**

170857Add to ListShare

Given a binary tree `root` and a linked list with `head` as the first node.

Return True if all the elements in the linked list starting from the `head` correspond to some *downward path* connected in the binary tree otherwise return False.

In this context downward path means a path that starts at some node and goes downwards.

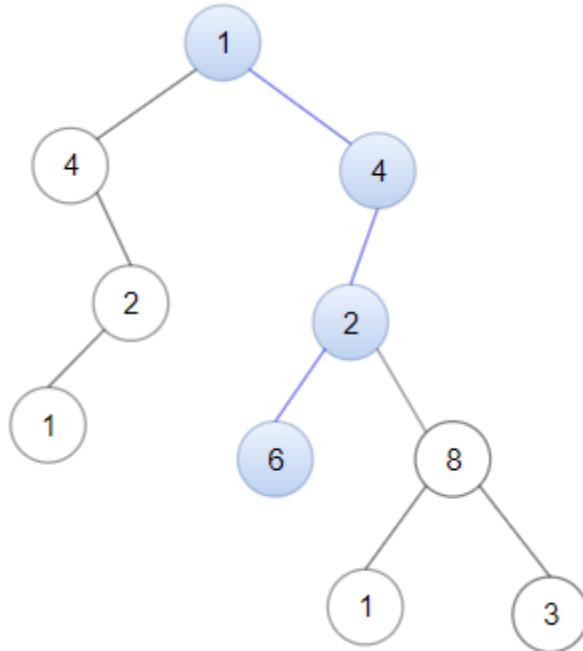
**Example 1:**

**Input:** `head = [4,2,8]`, `root = [1,4,4,null,2,2,null,1,null,6,8,null,null,null,1,3]`

**Output:** true

**Explanation:** Nodes in blue form a subpath in the binary Tree.

**Example 2:**



**Input:** head = [1,4,2,6], root = [1,4,4,null,2,2,null,1,null,6,8,null,null,null,1,3]

**Output:** true

**Example 3:**

**Input:** head = [1,4,2,6,8], root = [1,4,4,null,2,2,null,1,null,6,8,null,null,null,1,3]

**Output:** false

**Explanation:** There is no path in the binary tree that contains all the elements of the linked list from head.

**Constraints:**

- The number of nodes in the tree will be in the range [1, 2500].
- The number of nodes in the list will be in the range [1, 100].
- $1 \leq \text{Node.val} \leq 100$  for each node in the linked list and binary tree.

## 1368. Minimum Cost to Make at Least One Valid Path in a Grid

**Hard**

131112Add to ListShare

Given an  $m \times n$  grid. Each cell of the grid has a sign pointing to the next cell you should visit if you are currently in this cell. The sign of `grid[i][j]` can be:

- 1 which means go to the cell to the right. (i.e go from `grid[i][j]` to `grid[i][j + 1]`)
- 2 which means go to the cell to the left. (i.e go from `grid[i][j]` to `grid[i][j - 1]`)
- 3 which means go to the lower cell. (i.e go from `grid[i][j]` to `grid[i + 1][j]`)
- 4 which means go to the upper cell. (i.e go from `grid[i][j]` to `grid[i - 1][j]`)

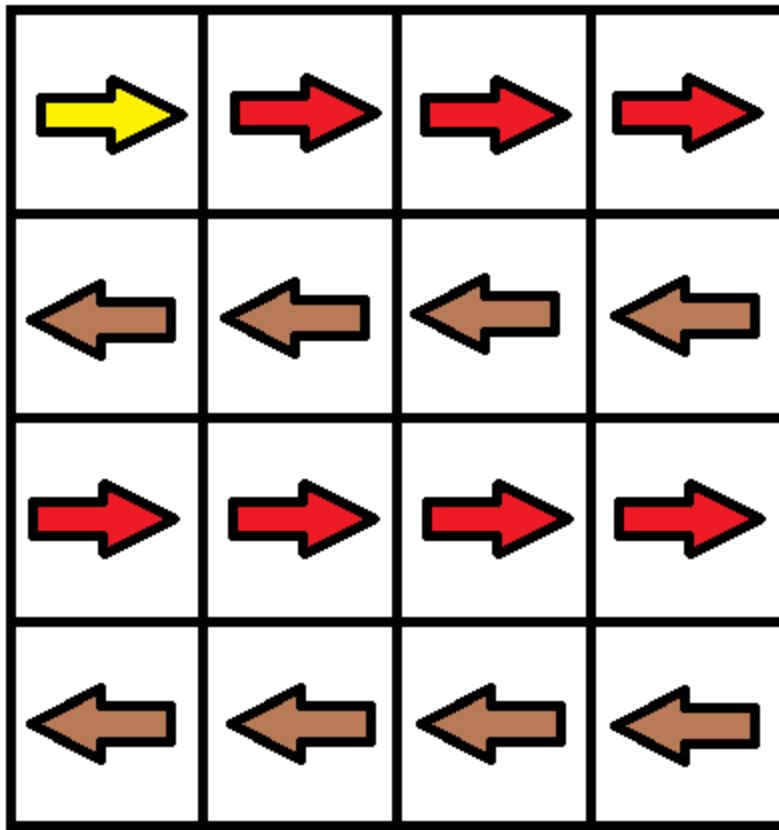
Notice that there could be some signs on the cells of the grid that point outside the grid.

You will initially start at the upper left cell  $(0, 0)$ . A valid path in the grid is a path that starts from the upper left cell  $(0, 0)$  and ends at the bottom-right cell  $(m - 1, n - 1)$  following the signs on the grid. The valid path does not have to be the shortest.

You can modify the sign on a cell with `cost = 1`. You can modify the sign on a cell **one time only**.

Return *the minimum cost to make the grid have at least one valid path*.

**Example 1:**



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

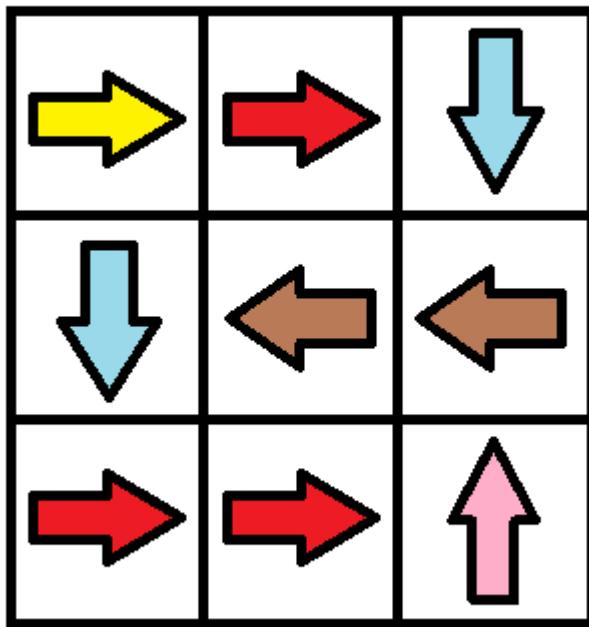
**Output:** 3

**Explanation:** You will start at point (0, 0).

The path to (3, 3) is as follows. (0, 0) --> (0, 1) --> (0, 2) --> (0, 3) change the arrow to down with cost = 1 --> (1, 3) --> (1, 2) --> (1, 1) --> (1, 0) change the arrow to down with cost = 1 --> (2, 0) --> (2, 1) --> (2, 2) --> (2, 3) change the arrow to down with cost = 1 --> (3, 3)

The total cost = 3.

**Example 2:**

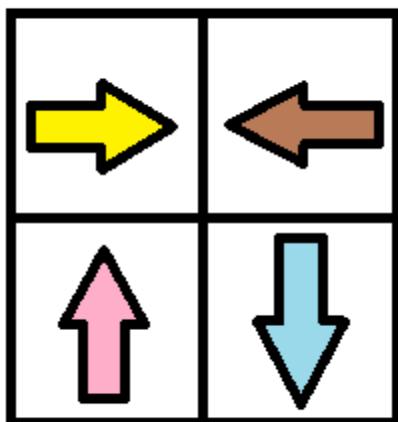


**Input:** grid = [[1,1,3],[3,2,2],[1,1,4]]

**Output:** 0

**Explanation:** You can follow the path from (0, 0) to (2, 2).

**Example 3:**



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

**Output:** 1

**Constraints:**

- $m == \text{grid.length}$
- $n == \text{grid}[i].length$
- $1 \leq m, n \leq 100$
- $1 \leq \text{grid}[i][j] \leq 4$

## 1370. Increasing Decreasing String

Easy

578689Add to ListShare

You are given a string  $s$ . Reorder the string using the following algorithm:

1. Pick the **smallest** character from  $s$  and **append** it to the result.
2. Pick the **smallest** character from  $s$  which is greater than the last appended character to the result and **append** it.
3. Repeat step 2 until you cannot pick more characters.
4. Pick the **largest** character from  $s$  and **append** it to the result.
5. Pick the **largest** character from  $s$  which is smaller than the last appended character to the result and **append** it.
6. Repeat step 5 until you cannot pick more characters.
7. Repeat the steps from 1 to 6 until you pick all characters from  $s$ .

In each step, If the smallest or the largest character appears more than once you can choose any occurrence and append it to the result.

Return *the result string after sorting  $s$  with this algorithm.*

**Example 1:**

**Input:**  $s = \text{"aaaabbbbcccc"}$

**Output:** "abccbaabccba"

**Explanation:** After steps 1, 2 and 3 of the first iteration,  $\text{result} = \text{"abc"}$

After steps 4, 5 and 6 of the first iteration,  $\text{result} = \text{"abccba"}$

First iteration is done. Now  $s = \text{"aabbcc"}$  and we go back to step 1

After steps 1, 2 and 3 of the second iteration,  $\text{result} = \text{"abccbaabc"}$

After steps 4, 5 and 6 of the second iteration,  $\text{result} = \text{"abccbaabccba"}$

**Example 2:****Input:** s = "rat"**Output:** "art"**Explanation:** The word "rat" becomes "art" after re-ordering it with the mentioned algorithm.**Constraints:**

- $1 \leq s.length \leq 500$
- s consists of only lowercase English letters.

**1371. Find the Longest Substring Containing Vowels in Even Counts****Medium**

126446Add to ListShare

Given the string s, return the size of the longest substring containing each vowel an even number of times. That is, 'a', 'e', 'i', 'o', and 'u' must appear an even number of times.

**Example 1:****Input:** s = "eleetminicoworoep"**Output:** 13**Explanation:** The longest substring is "leetminicowor" which contains two each of the vowels: e, i and o and zero of the vowels: a and u.**Example 2:****Input:** s = "leetcodeisgreat"**Output:** 5**Explanation:** The longest substring is "leetc" which contains two e's.**Example 3:****Input:** s = "bcbcbc"**Output:** 6**Explanation:** In this case, the given string "bcbcbc" is the longest because all vowels: a, e, i, o and u appear zero times.

**Constraints:**

- $1 \leq s.length \leq 5 \times 10^5$
- $s$  contains only lowercase English letters.

**1372. Longest ZigZag Path in a Binary Tree****Medium**

133023Add to ListShare

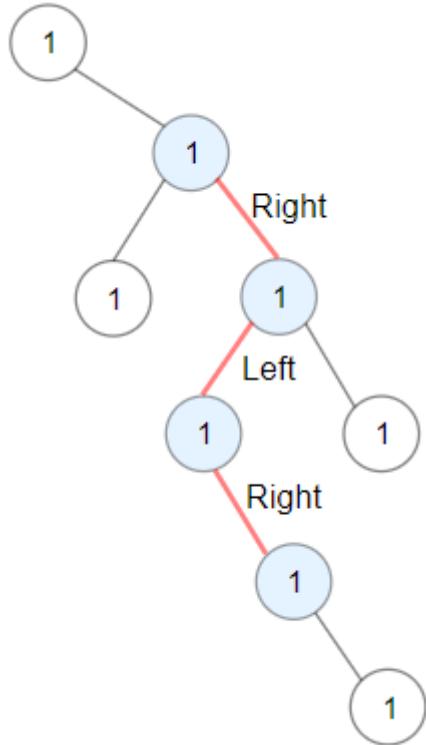
You are given the `root` of a binary tree.

A ZigZag path for a binary tree is defined as follow:

- Choose **any** node in the binary tree and a direction (right or left).
- If the current direction is right, move to the right child of the current node; otherwise, move to the left child.
- Change the direction from right to left or from left to right.
- Repeat the second and third steps until you can't move in the tree.

Zigzag length is defined as the number of nodes visited - 1. (A single node has a length of 0).

Return *the longest ZigZag path contained in that tree*.**Example 1:**

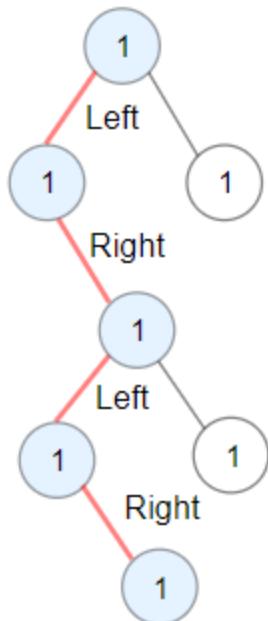


**Input:** root = [1,null,1,1,1,null,null,1,1,null,1,null,null,null,1,null,1]

**Output:** 3

**Explanation:** Longest ZigZag path in blue nodes (right  $\rightarrow$  left  $\rightarrow$  right).

**Example 2:**



**Input:** root = [1,1,1,null,1,null,null,1,1,null,1]

**Output:** 4

**Explanation:** Longest ZigZag path in blue nodes (left  $\rightarrow$  right  $\rightarrow$  left  $\rightarrow$  right).

**Example 3:**

**Input:** root = [1]

**Output:** 0

**Constraints:**

- The number of nodes in the tree is in the range  $[1, 5 * 10^4]$ .
- $1 \leq \text{Node.val} \leq 100$

## 1373. Maximum Sum BST in Binary Tree

Hard

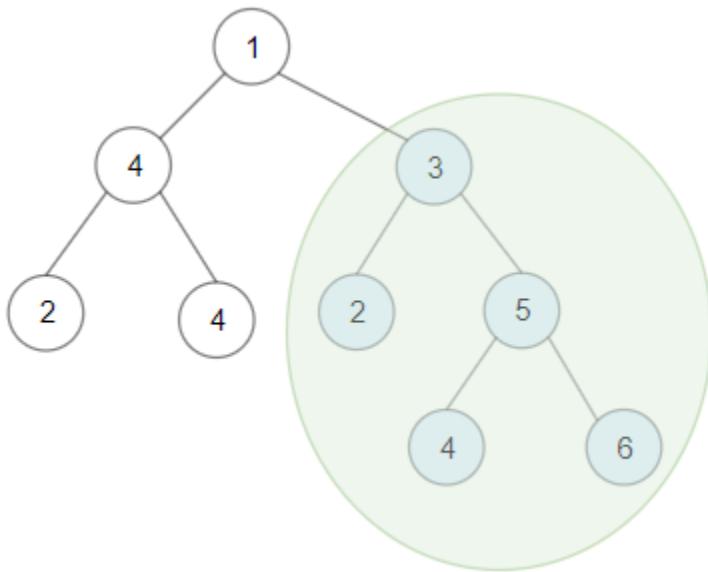
1700144Add to ListShare

Given a **binary tree** `root`, return the maximum sum of all keys of **any** sub-tree which is also a Binary Search Tree (BST).

Assume a BST is defined as follows:

- The left subtree of a node contains only nodes with keys **less than** the node's key.
- The right subtree of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

**Example 1:**

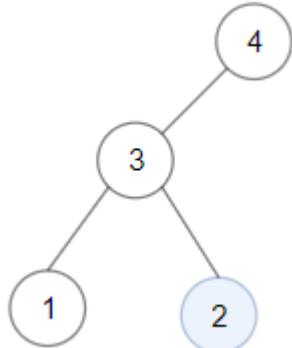


**Input:** root = [1,4,3,2,4,2,5,null,null,null,null,null,null,4,6]

**Output:** 20

**Explanation:** Maximum sum in a valid Binary search tree is obtained in root node with key equal to 3.

**Example 2:**



**Input:** root = [4,3,null,1,2]

**Output:** 2

**Explanation:** Maximum sum in a valid Binary search tree is obtained in a single root node with key equal to 2.

**Example 3:**

**Input:** root = [-4,-2,-5]

**Output:** 0

**Explanation:** All values are negatives. Return an empty BST.

**Constraints:**

- The number of nodes in the tree is in the range  $[1, 4 * 10^4]$ .
- $-4 * 10^4 \leq \text{Node.val} \leq 4 * 10^4$

## 1374. Generate a String With Characters That Have Odd Counts

Easy

3351074Add to ListShare

Given an integer  $n$ , return a string with  $n$  characters such that each character in such string occurs **an odd number of times**.

The returned string must contain only lowercase English letters. If there are multiples valid strings, return **any** of them.

**Example 1:**

**Input:** n = 4

**Output:** "pppz"

**Explanation:** "pppz" is a valid string since the character 'p' occurs three times and the character 'z' occurs once. Note that there are many other valid strings such as "ohhh" and "love".

**Example 2:**

**Input:** n = 2

**Output:** "xy"

**Explanation:** "xy" is a valid string since the characters 'x' and 'y' occur once. Note that there are many other valid strings such as "ag" and "ur".

**Example 3:**

**Input:** n = 7

**Output:** "holasss"

**Constraints:**

- $1 \leq n \leq 500$

## 1375. Number of Times Binary String Is Prefix-Aligned

Medium

775116Add to ListShare

You have a **1-indexed** binary string of length  $n$  where all the bits are `0` initially. We will flip all the bits of this binary string (i.e., change them from `0` to `1`) one by one. You are given a **1-indexed** integer array `flips` where `flips[i]` indicates that the bit at index  $i$  will be flipped in the  $i^{\text{th}}$  step.

A binary string is **prefix-aligned** if, after the  $i^{\text{th}}$  step, all the bits in the **inclusive** range  $[1, i]$  are ones and all the other bits are zeros.

Return *the number of times the binary string is prefix-aligned during the flipping process*.

### Example 1:

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

**Output:** 2

**Explanation:** The binary string is initially "00000".

After applying step 1: The string becomes "00100", which is not prefix-aligned.

After applying step 2: The string becomes "01100", which is not prefix-aligned.

After applying step 3: The string becomes "01110", which is not prefix-aligned.

After applying step 4: The string becomes "11110", which is prefix-aligned.

After applying step 5: The string becomes "11111", which is prefix-aligned.

We can see that the string was prefix-aligned 2 times, so we return 2.

### Example 2:

**Input:** `flips = [4,1,2,3]`

**Output:** 1

**Explanation:** The binary string is initially "0000".

After applying step 1: The string becomes "0001", which is not prefix-aligned.

After applying step 2: The string becomes "1001", which is not prefix-aligned.

After applying step 3: The string becomes "1101", which is not prefix-aligned.

After applying step 4: The string becomes "1111", which is prefix-aligned.

We can see that the string was prefix-aligned 1 time, so we return 1.

### Constraints:

- $n == \text{flips.length}$
- $1 \leq n \leq 5 * 10^4$
- $\text{flips}$  is a permutation of the integers in the range  $[1, n]$ .

## 1376. Time Needed to Inform All Employees

Medium

2302149 Add to List Share

A company has  $n$  employees with a unique ID for each employee from  $0$  to  $n - 1$ . The head of the company is the one with `headID`.

Each employee has one direct manager given in the `manager` array where `manager[i]` is the direct manager of the  $i$ -th employee, `manager[headID] = -1`. Also, it is guaranteed that the subordination relationships have a tree structure.

The head of the company wants to inform all the company employees of an urgent piece of news. He will inform his direct subordinates, and they will inform their subordinates, and so on until all employees know about the urgent news.

The  $i$ -th employee needs `informTime[i]` minutes to inform all of his direct subordinates (i.e., After `informTime[i]` minutes, all his direct subordinates can start spreading the news).

Return *the number of minutes* needed to inform all the employees about the urgent news.

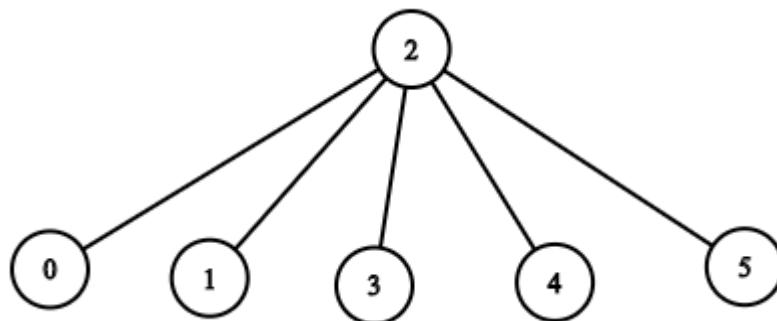
### Example 1:

**Input:**  $n = 1$ ,  $\text{headID} = 0$ ,  $\text{manager} = [-1]$ ,  $\text{informTime} = [0]$

**Output:** 0

**Explanation:** The head of the company is the only employee in the company.

### Example 2:



**Input:** `n = 6, headID = 2, manager = [2,2,-1,2,2,2], informTime = [0,0,1,0,0,0]`

**Output:** 1

**Explanation:** The head of the company with `id = 2` is the direct manager of all the employees in the company and needs 1 minute to inform them all.

The tree structure of the employees in the company is shown.

#### Constraints:

- `1 <= n <= 105`
- `0 <= headID < n`
- `manager.length == n`
- `0 <= manager[i] < n`
- `manager[headID] == -1`
- `informTime.length == n`
- `0 <= informTime[i] <= 1000`
- `informTime[i] == 0` if employee `i` has no subordinates.
- It is **guaranteed** that all the employees can be informed.

## 1376. Time Needed to Inform All Employees

Medium

2302149 Add to List Share

A company has `n` employees with a unique ID for each employee from `0` to `n - 1`. The head of the company is the one with `headID`.

Each employee has one direct manager given in the `manager` array where `manager[i]` is the direct manager of the `i-th` employee, `manager[headID] = -1`. Also, it is guaranteed that the subordination relationships have a tree structure.

The head of the company wants to inform all the company employees of an urgent piece of news. He will inform his direct subordinates, and they will inform their subordinates, and so on until all employees know about the urgent news.

The  $i$ -th employee needs `informTime[i]` minutes to inform all of his direct subordinates (i.e., After `informTime[i]` minutes, all his direct subordinates can start spreading the news).

Return *the number of minutes* needed to inform all the employees about the urgent news.

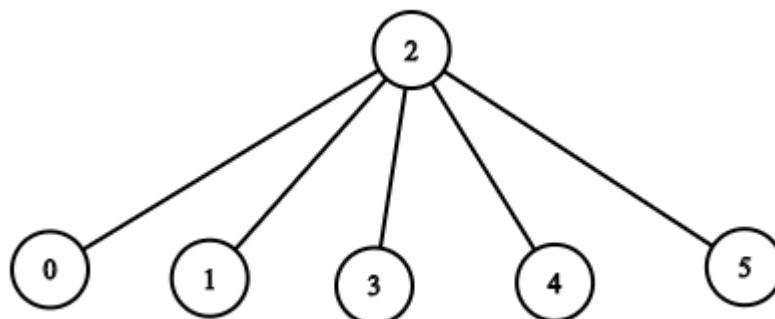
**Example 1:**

**Input:** `n = 1, headID = 0, manager = [-1], informTime = [0]`

**Output:** 0

**Explanation:** The head of the company is the only employee in the company.

**Example 2:**



**Input:** `n = 6, headID = 2, manager = [2,2,-1,2,2,2], informTime = [0,0,1,0,0,0]`

**Output:** 1

**Explanation:** The head of the company with `id = 2` is the direct manager of all the employees in the company and needs 1 minute to inform them all.

The tree structure of the employees in the company is shown.

**Constraints:**

- $1 \leq n \leq 10^5$
- $0 \leq \text{headID} < n$
- `manager.length == n`
- $0 \leq \text{manager}[i] < n$
- `manager[headID] == -1`
- `informTime.length == n`
- $0 \leq \text{informTime}[i] \leq 1000$
- `informTime[i] == 0` if employee  $i$  has no subordinates.
- It is **guaranteed** that all the employees can be informed.

## 1379. Find a Corresponding Node of a Binary Tree in a Clone of That Tree

Easy

13171664Add to ListShare

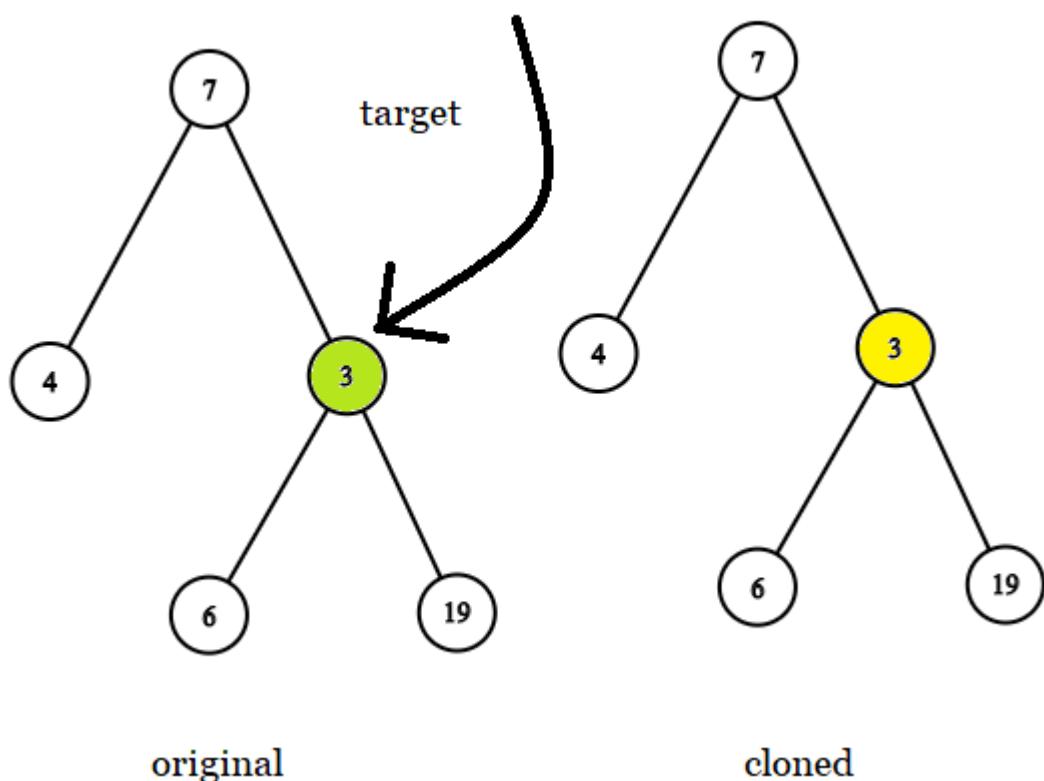
Given two binary trees `original` and `cloned` and given a reference to a node `target` in the original tree.

The cloned tree is a **copy of** the original tree.

Return a reference to the same node in the cloned tree.

**Note** that you are **not allowed** to change any of the two trees or the `target` node and the answer **must be** a reference to a node in the `cloned` tree.

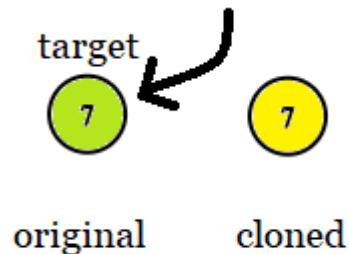
### Example 1:



**Input:** tree = [7,4,3,null,null,6,19], target = 3

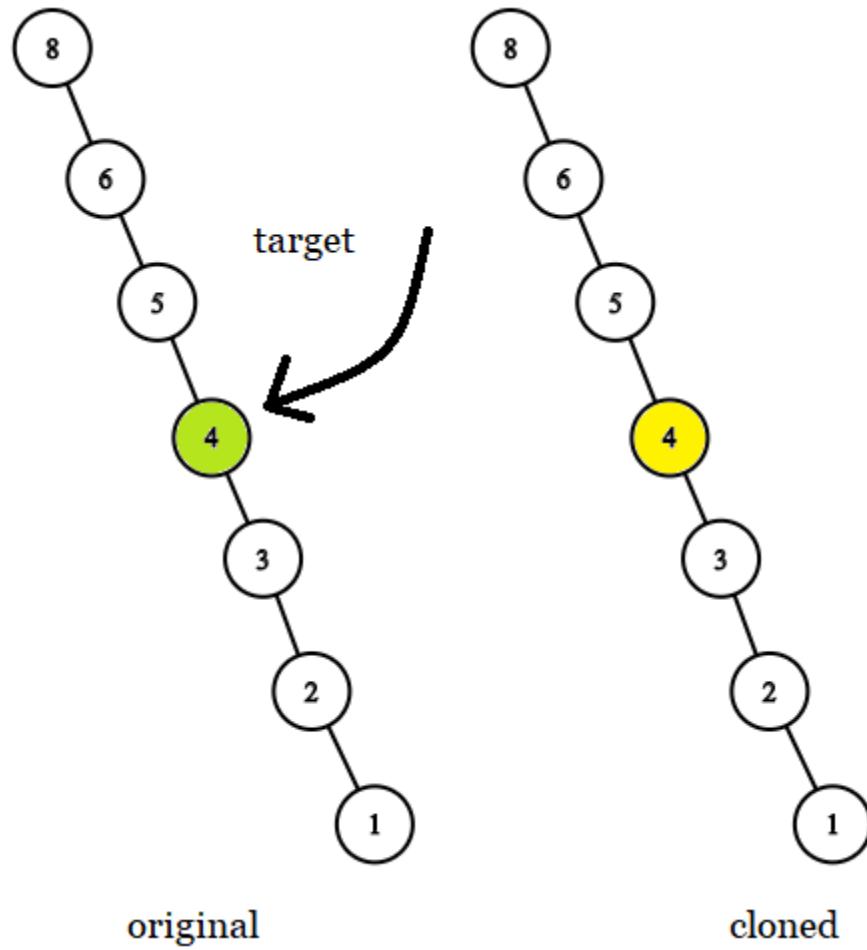
**Output:** 3

**Explanation:** In all examples the original and cloned trees are shown. The target node is a green node from the original tree. The answer is the yellow node from the cloned tree.

**Example 2:**

**Input:** tree = [7], target = 7

**Output:** 7

**Example 3:**

**Input:** tree = [8,null,6,null,5,null,4,null,3,null,2,null,1], target = 4

**Output:** 4

**Constraints:**

- The number of nodes in the `tree` is in the range  $[1, 10^4]$ .
- The values of the nodes of the `tree` are unique.
- `target` node is a node from the `original` tree and is not `null`.

**1380. Lucky Numbers in a Matrix****Easy**

117669Add to ListShare

Given an  $m \times n$  matrix of **distinct** numbers, return *all lucky numbers in the matrix in any order*.

A **lucky number** is an element of the matrix such that it is the minimum element in its row and maximum in its column.

**Example 1:****Input:** matrix = [[3,7,8],[9,11,13],[15,16,17]]**Output:** [15]

**Explanation:** 15 is the only lucky number since it is the minimum in its row and the maximum in its column.

**Example 2:****Input:** matrix = [[1,10,4,2],[9,3,8,7],[15,16,17,12]]**Output:** [12]

**Explanation:** 12 is the only lucky number since it is the minimum in its row and the maximum in its column.

**Example 3:****Input:** matrix = [[7,8],[1,2]]**Output:** [7]

**Explanation:** 7 is the only lucky number since it is the minimum in its row and the maximum in its column.

**Constraints:**

- $m == \text{mat.length}$
- $n == \text{mat[i].length}$

- $1 \leq n, m \leq 50$
- $1 \leq \text{matrix}[i][j] \leq 10^5$ .
- All elements in the matrix are distinct.

## 1381. Design a Stack With Increment Operation

Medium

136774Add to ListShare

Design a stack which supports the following operations.

Implement the `CustomStack` class:

- `CustomStack(int maxSize)` Initializes the object with `maxSize` which is the maximum number of elements in the stack or do nothing if the stack reached the `maxSize`.
- `void push(int x)` Adds `x` to the top of the stack if the stack hasn't reached the `maxSize`.
- `int pop()` Pops and returns the top of stack or `-1` if the stack is empty.
- `void inc(int k, int val)` Increments the bottom `k` elements of the stack by `val`. If there are less than `k` elements in the stack, just increment all the elements in the stack.

### Example 1:

#### Input

```
["CustomStack","push","push","pop","push","push","push","increment","increment","pop",
"pop","pop","pop"]

[[3],[1],[2],[[]],[2],[3],[4],[5,100],[2,100],[[],[],[],[]]]
```

#### Output

```
[null,null,null,2,null,null,null,null,103,202,201,-1]
```

#### Explanation

```
CustomStack customStack = new CustomStack(3); // Stack is Empty []

customStack.push(1); // stack becomes [1]

customStack.push(2); // stack becomes [1, 2]

customStack.pop(); // return 2 --> Return top of the stack
2, stack becomes [1]

customStack.push(2); // stack becomes [1, 2]

customStack.push(3); // stack becomes [1, 2, 3]

customStack.push(4); // stack still [1, 2, 3], Don't add
another elements as size is 4
```

```

customStack.increment(5, 100);           // stack becomes [101, 102, 103]
customStack.increment(2, 100);           // stack becomes [201, 202, 103]
customStack.pop();                     // return 103 --> Return top of the
stack 103, stack becomes [201, 202]
customStack.pop();                     // return 202 --> Return top of the
stack 102, stack becomes [201]
customStack.pop();                     // return 201 --> Return top of the
stack 101, stack becomes []
customStack.pop();                     // return -1 --> Stack is empty return
-1.

```

### Constraints:

- $1 \leq \text{maxSize} \leq 1000$
- $1 \leq x \leq 1000$
- $1 \leq k \leq 1000$
- $0 \leq \text{val} \leq 100$
- At most 1000 calls will be made to each method of `increment`, `push` and `pop` each separately.

## 1382. Balance a Binary Search Tree

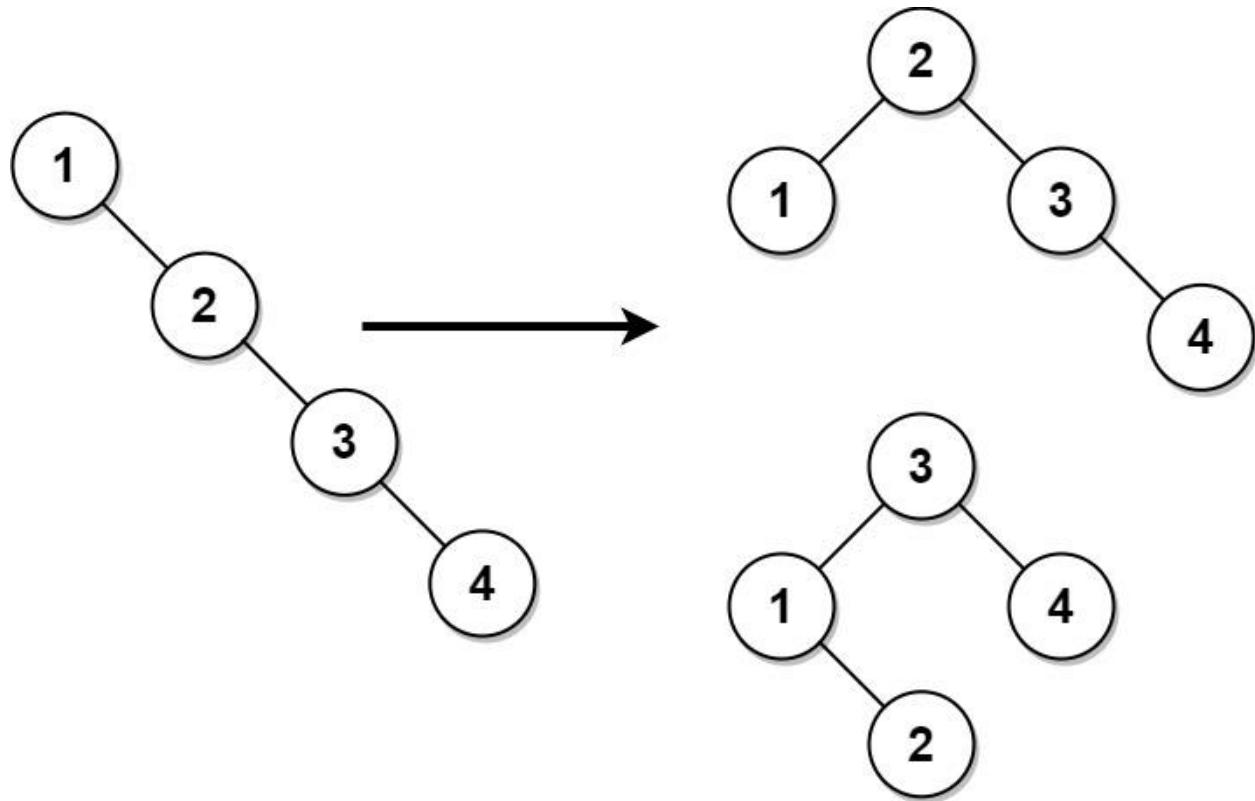
Medium

221360Add to ListShare

Given the `root` of a binary search tree, return a **balanced** binary search tree with the same node values. If there is more than one answer, return **any of them**.

A binary search tree is **balanced** if the depth of the two subtrees of every node never differs by more than 1.

### Example 1:

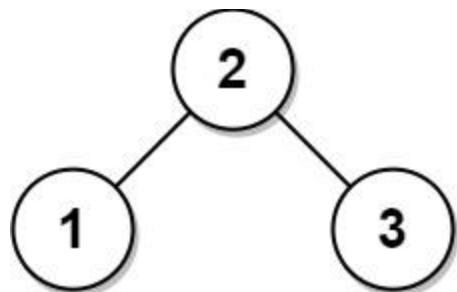


**Input:** root = [1,null,2,null,3,null,4,null,null]

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

**Explanation:** This is not the only correct answer, [3,1,4,null,2] is also correct.

**Example 2:**



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

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

**Constraints:**

- The number of nodes in the tree is in the range  $[1, 10^4]$ .
- $1 \leq \text{Node.val} \leq 10^5$

### 1383. Maximum Performance of a Team

Hard

260068Add to ListShare

You are given two integers `n` and `k` and two integer arrays `speed` and `efficiency` both of length `n`. There are `n` engineers numbered from 1 to `n`. `speed[i]` and `efficiency[i]` represent the speed and efficiency of the `ith` engineer respectively.

Choose **at most** `k` different engineers out of the `n` engineers to form a team with the maximum **performance**.

The performance of a team is the sum of their engineers' speeds multiplied by the minimum efficiency among their engineers.

Return *the maximum performance of this team*. Since the answer can be a huge number, return it **modulo**  $10^9 + 7$ .

#### Example 1:

**Input:** `n = 6, speed = [2,10,3,1,5,8], efficiency = [5,4,3,9,7,2], k = 2`

**Output:** 60

**Explanation:**

We have the maximum performance of the team by selecting engineer 2 (with speed=10 and efficiency=4) and engineer 5 (with speed=5 and efficiency=7). That is,  $\text{performance} = (10 + 5) * \min(4, 7) = 60$ .

#### Example 2:

**Input:** `n = 6, speed = [2,10,3,1,5,8], efficiency = [5,4,3,9,7,2], k = 3`

**Output:** 68

**Explanation:**

This is the same example as the first but `k = 3`. We can select engineer 1, engineer 2 and engineer 5 to get the maximum performance of the team. That is,  $\text{performance} = (2 + 10 + 5) * \min(5, 4, 7) = 68$ .

#### Example 3:

**Input:** `n = 6, speed = [2,10,3,1,5,8], efficiency = [5,4,3,9,7,2], k = 4`

**Output:** 72

**Constraints:**

- $1 \leq k \leq n \leq 10^5$
- $\text{speed.length} == n$
- $\text{efficiency.length} == n$
- $1 \leq \text{speed}[i] \leq 10^5$
- $1 \leq \text{efficiency}[i] \leq 10^8$

**1385. Find the Distance Value Between Two Arrays****Easy**

5642140 Add to List Share

Given two integer arrays `arr1` and `arr2`, and the integer `d`, return the distance value between the two arrays.

The distance value is defined as the number of elements `arr1[i]` such that there is not any element `arr2[j]` where  $|arr1[i]-arr2[j]| \leq d$ .

**Example 1:**

**Input:** `arr1 = [4,5,8]`, `arr2 = [10,9,1,8]`, `d = 2`

**Output:** 2

**Explanation:**

For  $arr1[0]=4$  we have:

$$|4-10|=6 > d=2$$

$$|4-9|=5 > d=2$$

$$|4-1|=3 > d=2$$

$$|4-8|=4 > d=2$$

For  $arr1[1]=5$  we have:

$$|5-10|=5 > d=2$$

$$|5-9|=4 > d=2$$

$$|5-1|=4 > d=2$$

$$|5-8|=3 > d=2$$

For  $arr1[2]=8$  we have:

$$|8-10|=2 \leq d=2$$

$|8-9|=1 \leq d=2$

$|8-1|=7 > d=2$

$|8-8|=0 \leq d=2$

**Example 2:**

**Input:** arr1 = [1,4,2,3], arr2 = [-4,-3,6,10,20,30], d = 3

**Output:** 2

**Example 3:**

**Input:** arr1 = [2,1,100,3], arr2 = [-5,-2,10,-3,7], d = 6

**Output:** 1

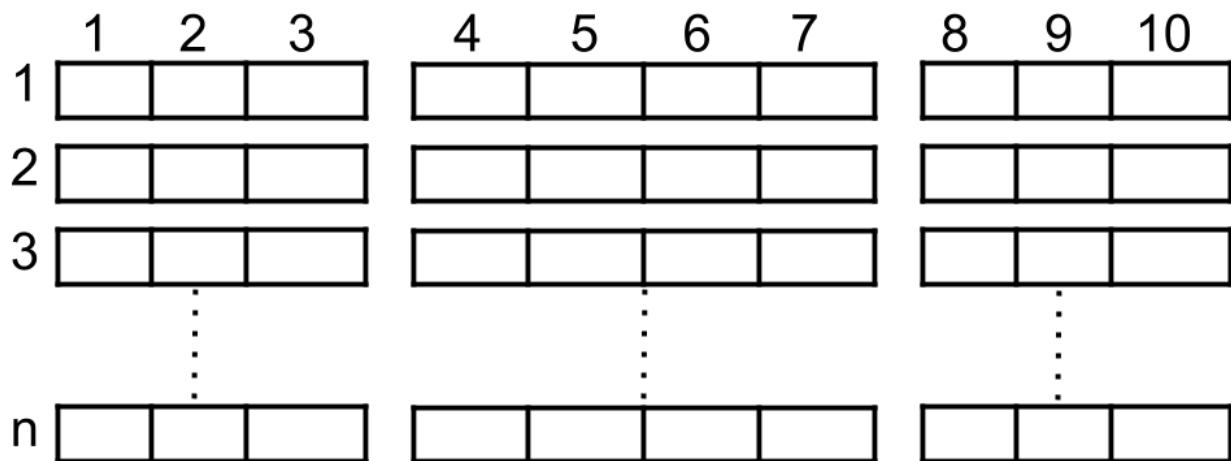
**Constraints:**

- $1 \leq \text{arr1.length}, \text{arr2.length} \leq 500$
- $-1000 \leq \text{arr1}[i], \text{arr2}[j] \leq 1000$
- $0 \leq d \leq 100$

## 1386. Cinema Seat Allocation

Medium

615315 Add to List Share

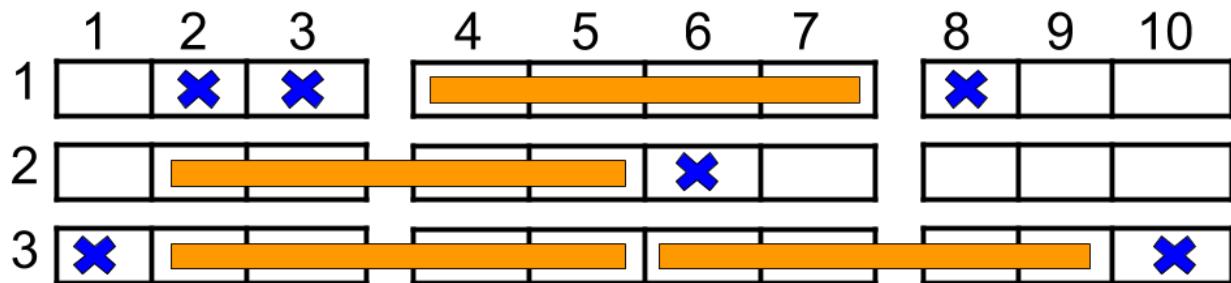


A cinema has  $n$  rows of seats, numbered from 1 to  $n$  and there are ten seats in each row, labelled from 1 to 10 as shown in the figure above.

Given the array `reservedSeats` containing the numbers of seats already reserved, for example, `reservedSeats[i] = [3, 8]` means the seat located in row **3** and labelled with **8** is already reserved.

*Return the maximum number of four-person groups you can assign on the cinema seats.* A four-person group occupies four adjacent seats **in one single row**. Seats across an aisle (such as [3,3] and [3,4]) are not considered to be adjacent, but there is an exceptional case on which an aisle split a four-person group, in that case, the aisle split a four-person group in the middle, which means to have two people on each side.

#### Example 1:



**Input:** `n = 3, reservedSeats = [[1,2],[1,3],[1,8],[2,6],[3,1],[3,10]]`

**Output:** 4

**Explanation:** The figure above shows the optimal allocation for four groups, where seats mark with blue are already reserved and contiguous seats mark with orange are for one group.

#### Example 2:

**Input:** `n = 2, reservedSeats = [[2,1],[1,8],[2,6]]`

**Output:** 2

#### Example 3:

**Input:** `n = 4, reservedSeats = [[4,3],[1,4],[4,6],[1,7]]`

**Output:** 4

#### Constraints:

- `1 <= n <= 10^9`
- `1 <= reservedSeats.length <= min(10*n, 10^4)`
- `reservedSeats[i].length == 2`

- $1 \leq \text{reservedSeats}[i][0] \leq n$
- $1 \leq \text{reservedSeats}[i][1] \leq 10$
- All  $\text{reservedSeats}[i]$  are distinct.

## 1387. Sort Integers by The Power Value

Medium

105298Add to ListShare

The power of an integer  $x$  is defined as the number of steps needed to transform  $x$  into  $1$  using the following steps:

- if  $x$  is even then  $x = x / 2$
- if  $x$  is odd then  $x = 3 * x + 1$

For example, the power of  $x = 3$  is  $7$  because  $3$  needs  $7$  steps to become  $1$  ( $3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$ ).

Given three integers  $lo$ ,  $hi$  and  $k$ . The task is to sort all integers in the interval  $[lo, hi]$  by the power value in **ascending order**, if two or more integers have **the same** power value sort them by **ascending order**.

Return the  $k^{\text{th}}$  integer in the range  $[lo, hi]$  sorted by the power value.

Notice that for any integer  $x$  ( $lo \leq x \leq hi$ ) it is **guaranteed** that  $x$  will transform into  $1$  using these steps and that the power of  $x$  is will **fit** in a 32-bit signed integer.

### Example 1:

**Input:**  $lo = 12$ ,  $hi = 15$ ,  $k = 2$

**Output:**  $13$

**Explanation:** The power of  $12$  is  $9$  ( $12 \rightarrow 6 \rightarrow 3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$ )

The power of  $13$  is  $9$

The power of  $14$  is  $17$

The power of  $15$  is  $17$

The interval sorted by the power value  $[12, 13, 14, 15]$ . For  $k = 2$  answer is the second element which is  $13$ .

Notice that  $12$  and  $13$  have the same power value and we sorted them in ascending order. Same for  $14$  and  $15$ .

### Example 2:

**Input:** lo = 7, hi = 11, k = 4

**Output:** 7

**Explanation:** The power array corresponding to the interval [7, 8, 9, 10, 11] is [16, 3, 19, 6, 14].

The interval sorted by power is [8, 10, 11, 7, 9].

The fourth number in the sorted array is 7.

### Constraints:

- $1 \leq lo \leq hi \leq 1000$
- $1 \leq k \leq hi - lo + 1$

## 1388. Pizza With 3n Slices

Hard

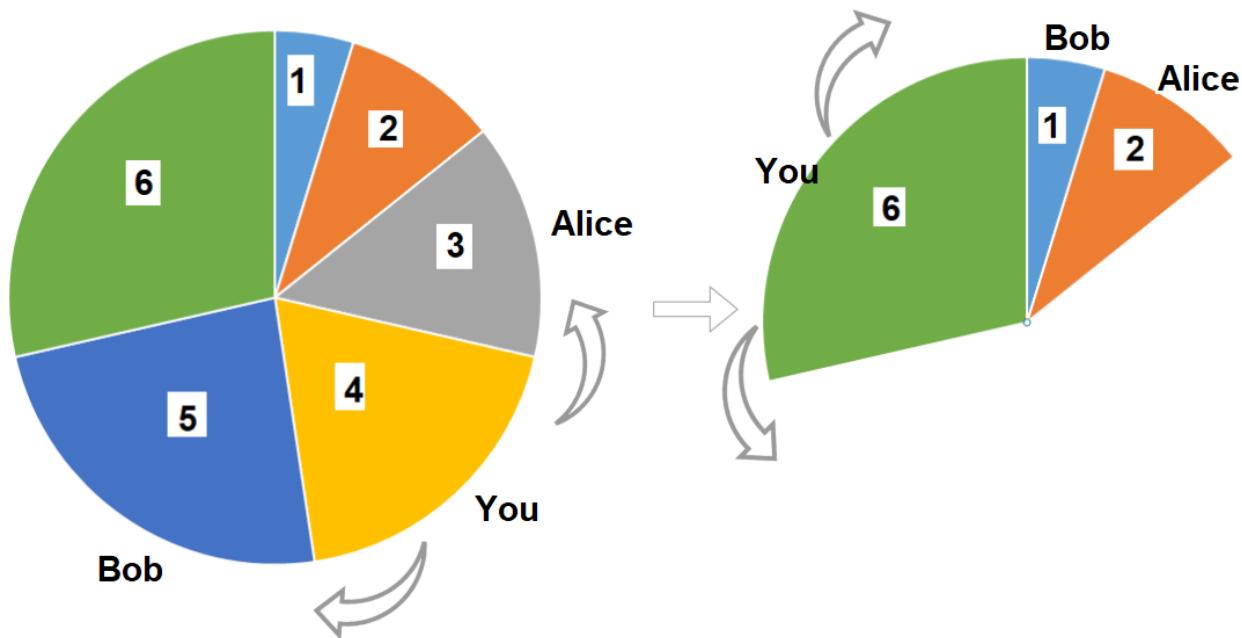
69812Add to ListShare

There is a pizza with  $3n$  slices of varying size, you and your friends will take slices of pizza as follows:

- You will pick **any** pizza slice.
- Your friend Alice will pick the next slice in the anti-clockwise direction of your pick.
- Your friend Bob will pick the next slice in the clockwise direction of your pick.
- Repeat until there are no more slices of pizzas.

Given an integer array `slices` that represent the sizes of the pizza slices in a clockwise direction, return *the maximum possible sum of slice sizes that you can pick*.

### Example 1:

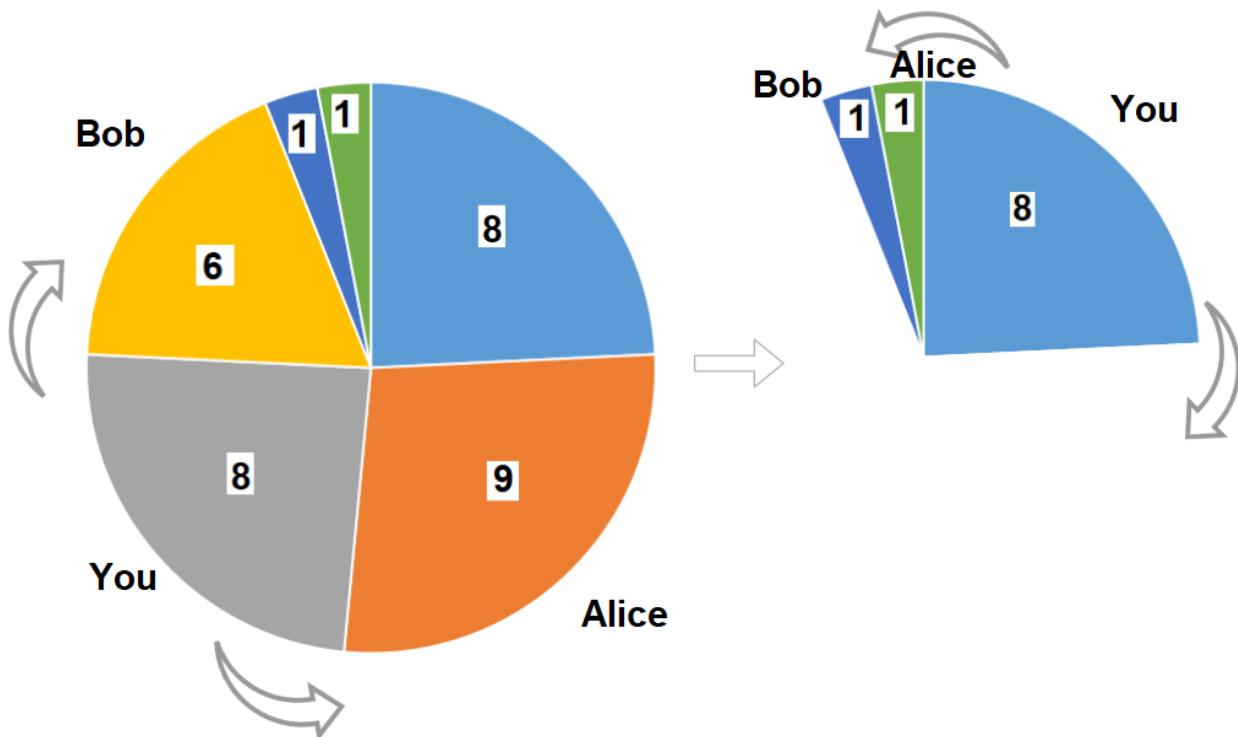


**Input:** slices = [1,2,3,4,5,6]

**Output:** 10

**Explanation:** Pick pizza slice of size 4, Alice and Bob will pick slices with size 3 and 5 respectively. Then Pick slices with size 6, finally Alice and Bob will pick slice of size 2 and 1 respectively. Total = 4 + 6.

**Example 2:**



**Input:** slices = [8,9,8,6,1,1]

**Output:** 16

**Explanation:** Pick pizza slice of size 8 in each turn. If you pick slice with size 9 your partners will pick slices of size 8.

#### Constraints:

- $3 * n == \text{slices.length}$
- $1 \leq \text{slices.length} \leq 500$
- $1 \leq \text{slices}[i] \leq 1000$

### 1389. Create Target Array in the Given Order

Easy

12681351 Add to List Share

Given two arrays of integers `nums` and `index`. Your task is to create `target` array under the following rules:

- Initially `target` array is empty.
- From left to right read `nums[i]` and `index[i]`, insert at index `index[i]` the value `nums[i]` in `target` array.
- Repeat the previous step until there are no elements to read in `nums` and `index`.

Return the *target* array.

It is guaranteed that the insertion operations will be valid.

**Example 1:**

**Input:** `nums = [0,1,2,3,4], index = [0,1,2,2,1]`

**Output:** `[0,4,1,3,2]`

**Explanation:**

nums	index	target
0	0	[0]
1	1	[0,1]
2	2	[0,1,2]
3	2	[0,1,3,2]
4	1	[0,4,1,3,2]

**Example 2:**

**Input:** `nums = [1,2,3,4,0], index = [0,1,2,3,0]`

**Output:** `[0,1,2,3,4]`

**Explanation:**

nums	index	target
1	0	[1]
2	1	[1,2]
3	2	[1,2,3]
4	3	[1,2,3,4]
0	0	[0,1,2,3,4]

**Example 3:**

**Input:** `nums = [1], index = [0]`

**Output:** `[1]`

**Constraints:**

- $1 \leq \text{nums.length}, \text{index.length} \leq 100$
- $\text{nums.length} == \text{index.length}$
- $0 \leq \text{nums}[i] \leq 100$
- $0 \leq \text{index}[i] \leq i$

**1390. Four Divisors****Medium**

260158Add to ListShare

Given an integer array `nums`, return *the sum of divisors of the integers in that array that have exactly four divisors*. If there is no such integer in the array, return `0`.

**Example 1:****Input:** `nums = [21,4,7]`**Output:** `32`**Explanation:**`21` has 4 divisors: `1, 3, 7, 21``4` has 3 divisors: `1, 2, 4``7` has 2 divisors: `1, 7`The answer is the sum of divisors of `21` only.**Example 2:****Input:** `nums = [21,21]`**Output:** `64`**Example 3:****Input:** `nums = [1,2,3,4,5]`**Output:** `0`**Constraints:**

- $1 \leq \text{nums.length} \leq 10^4$
- $1 \leq \text{nums}[i] \leq 10^5$

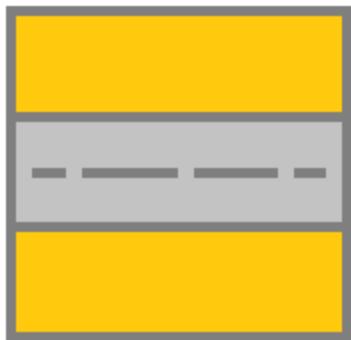
## 1391. Check if There is a Valid Path in a Grid

Medium

632266Add to ListShare

You are given an  $m \times n$  grid. Each cell of `grid` represents a street. The street of `grid[i][j]` can be:

- 1 which means a street connecting the left cell and the right cell.
- 2 which means a street connecting the upper cell and the lower cell.
- 3 which means a street connecting the left cell and the lower cell.
- 4 which means a street connecting the right cell and the lower cell.
- 5 which means a street connecting the left cell and the upper cell.
- 6 which means a street connecting the right cell and the upper cell.



Street 1



Street 2



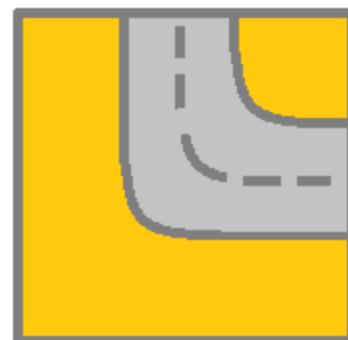
Street 3



Street 4



Street 5



Street 6

You will initially start at the street of the upper-left cell  $(0, 0)$ . A valid path in the grid is a path that starts from the upper left cell  $(0, 0)$  and ends at the bottom-right cell  $(m - 1, n - 1)$ . **The path should only follow the streets.**

**Notice** that you are **not allowed** to change any street.

Return `true` if there is a valid path in the grid or `false` otherwise.

**Example 1:**

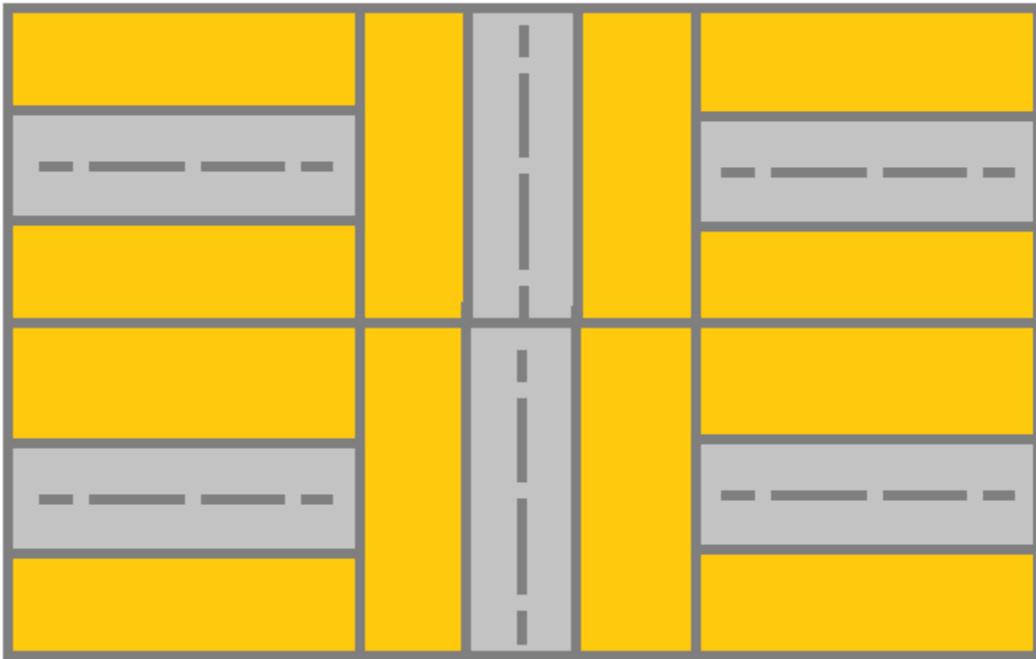


**Input:** `grid = [[2,4,3],[6,5,2]]`

**Output:** `true`

**Explanation:** As shown you can start at cell  $(0, 0)$  and visit all the cells of the grid to reach  $(m - 1, n - 1)$ .

**Example 2:**



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

**Output:** false

**Explanation:** As shown you the street at cell (0, 0) is not connected with any street of any other cell and you will get stuck at cell (0, 0)

**Example 3:**

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

**Output:** false

**Explanation:** You will get stuck at cell (0, 1) and you cannot reach cell (0, 2).

**Constraints:**

- $m == \text{grid.length}$
- $n == \text{grid}[i].length$
- $1 \leq m, n \leq 300$
- $1 \leq \text{grid}[i][j] \leq 6$

## 1392. Longest Happy Prefix

Hard

82827Add to ListShare

A string is called a **happy prefix** if is a **non-empty** prefix which is also a suffix (excluding itself).

Given a string `s`, return *the longest happy prefix* of `s`. Return an empty string `""` if no such prefix exists.

**Example 1:**

**Input:** `s = "level"`

**Output:** `"l"`

**Explanation:** `s` contains 4 prefix excluding itself ("l", "le", "lev", "leve"), and suffix ("l", "el", "vel", "evel"). The largest prefix which is also suffix is given by "l".

**Example 2:**

**Input:** `s = "ababab"`

**Output:** `"abab"`

**Explanation:** "abab" is the largest prefix which is also suffix. They can overlap in the original string.

**Constraints:**

- `1 <= s.length <= 105`
- `s` contains only lowercase English letters.

## 1393. Capital Gain/Loss

Medium

42232Add to ListShare

SQL Schema

Table: `Stocks`

Column Name	Type
<code>stock_name</code>	<code>varchar</code>
<code>operation</code>	<code>enum</code>

operation_day	int	
price	int	
-----+-----+		

(stock\_name, operation\_day) is the primary key for this table.

The operation column is an ENUM of type ('Sell', 'Buy')

Each row of this table indicates that the stock which has stock\_name had an operation on the day operation\_day with the price.

It is guaranteed that each 'Sell' operation for a stock has a corresponding 'Buy' operation in a previous day. It is also guaranteed that each 'Buy' operation for a stock has a corresponding 'Sell' operation in an upcoming day.

Write an SQL query to report the **Capital gain/loss** for each stock.

The **Capital gain/loss** of a stock is the total gain or loss after buying and selling the stock one or many times.

Return the result table in **any order**.

The query result format is in the following example.

### Example 1:

**Input:**

Stocks table:

stock_name	operation	operation_day	price	
-----+-----+-----+-----+				
Leetcode	Buy	1	1000	
Corona Masks	Buy	2	10	
Leetcode	Sell	5	9000	
Handbags	Buy	17	30000	
Corona Masks	Sell	3	1010	
Corona Masks	Buy	4	1000	

Corona Masks	Sell	5	500	
Corona Masks	Buy	6	1000	
Handbags	Sell	29	7000	
Corona Masks	Sell	10	10000	

**Output:**

stock_name	capital_gain_loss
Corona Masks	9500
Leetcode	8000
Handbags	-23000

**Explanation:**

Leetcode stock was bought at day 1 for 1000\$ and was sold at day 5 for 9000\$. Capital gain =  $9000 - 1000 = 8000$$ .

Handbags stock was bought at day 17 for 30000\$ and was sold at day 29 for 7000\$. Capital loss =  $7000 - 30000 = -23000$$ .

Corona Masks stock was bought at day 1 for 10\$ and was sold at day 3 for 1010\$. It was bought again at day 4 for 1000\$ and was sold at day 5 for 500\$. At last, it was bought at day 6 for 1000\$ and was sold at day 10 for 10000\$. Capital gain/loss is the sum of capital gains/losses for each ('Buy' --> 'Sell') operation =  $(1010 - 10) + (500 - 1000) + (10000 - 1000) = 1000 - 500 + 9000 = 9500$$ .

## 1394. Find Lucky Integer in an Array

Easy

77022Add to ListShare

Given an array of integers `arr`, a **lucky integer** is an integer that has a frequency in the array equal to its value.

Return *the largest lucky integer in the array*. If there is no **lucky integer** return `-1`.

**Example 1:**

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

**Output:** 2

**Explanation:** The only lucky number in the array is 2 because frequency[2] == 2.

**Example 2:**

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

**Output:** 3

**Explanation:** 1, 2 and 3 are all lucky numbers, return the largest of them.

**Example 3:**

**Input:** arr = [2,2,2,3,3]

**Output:** -1

**Explanation:** There are no lucky numbers in the array.

**Constraints:**

- $1 \leq \text{arr.length} \leq 500$
- $1 \leq \text{arr}[i] \leq 500$

## 1395. Count Number of Teams

Medium

2198169Add to ListShare

There are  $n$  soldiers standing in a line. Each soldier is assigned a **unique** `rating` value.

You have to form a team of 3 soldiers amongst them under the following rules:

- Choose 3 soldiers with index  $(i, j, k)$  with rating  $(\text{rating}[i], \text{rating}[j], \text{rating}[k])$ .
- A team is valid if:  $(\text{rating}[i] < \text{rating}[j] < \text{rating}[k])$  or  $(\text{rating}[i] > \text{rating}[j] > \text{rating}[k])$  where  $(0 \leq i < j < k < n)$ .

Return the number of teams you can form given the conditions. (soldiers can be part of multiple teams).

**Example 1:**

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

**Output:** 3

**Explanation:** We can form three teams given the conditions. (2,3,4), (5,4,1), (5,3,1).

**Example 2:**

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

**Output:** 0

**Explanation:** We can't form any team given the conditions.

**Example 3:**

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

**Output:** 4

**Constraints:**

- `n == rating.length`
- `3 <= n <= 1000`
- `1 <= rating[i] <= 105`
- All the integers in `rating` are **unique**.

## 1396. Design Underground System

Medium

2232110Add to ListShare

An underground railway system is keeping track of customer travel times between different stations. They are using this data to calculate the average time it takes to travel from one station to another.

Implement the `UndergroundSystem` class:

- `void checkIn(int id, string stationName, int t)`
  - A customer with a card ID equal to `id`, checks in at the station `stationName` at time `t`.
  - A customer can only be checked into one place at a time.
- `void checkOut(int id, string stationName, int t)`
  - A customer with a card ID equal to `id`, checks out from the station `stationName` at time `t`.
- `double getAverageTime(string startStation, string endStation)`
  - Returns the average time it takes to travel from `startStation` to `endStation`.
  - The average time is computed from all the previous traveling times from `startStation` to `endStation` that happened **directly**, meaning a check in at `startStation` followed by a check out from `endStation`.

- The time it takes to travel from `startStation` to `endStation` **may be different** from the time it takes to travel from `endStation` to `startStation`.
- There will be at least one customer that has traveled from `startStation` to `endStation` before `getAverageTime` is called.

You may assume all calls to the `checkIn` and `checkOut` methods are consistent. If a customer checks in at time  $t_1$  then checks out at time  $t_2$ , then  $t_1 < t_2$ . All events happen in chronological order.

### Example 1:

#### Input

```
["UndergroundSystem", "checkIn", "checkIn", "checkIn", "checkOut", "checkOut", "checkOut", "getAverageTime", "getAverageTime", "checkIn", "getAverageTime", "checkOut", "getAverageTime"]
[[[], [45, "Leyton", 3], [32, "Paradise", 8], [27, "Leyton", 10], [45, "Waterloo", 15], [27, "Waterloo", 20], [32, "Cambridge", 22], ["Paradise", "Cambridge"], ["Leyton", "Waterloo"], [10, "Leyton", 24], ["Leyton", "Waterloo"], [10, "Waterloo", 38], ["Leyton", "Waterloo"]]
```

#### Output

```
[null, null, null, null, null, null, 14.00000, 11.00000, null, 11.00000, null, 12.00000]
```

#### Explanation

```
UndergroundSystem undergroundSystem = new UndergroundSystem();

undergroundSystem.checkIn(45, "Leyton", 3);

undergroundSystem.checkIn(32, "Paradise", 8);

undergroundSystem.checkIn(27, "Leyton", 10);

undergroundSystem.checkOut(45, "Waterloo", 15); // Customer 45 "Leyton" -> "Waterloo" in 15-3 = 12

undergroundSystem.checkOut(27, "Waterloo", 20); // Customer 27 "Leyton" -> "Waterloo" in 20-10 = 10

undergroundSystem.checkOut(32, "Cambridge", 22); // Customer 32 "Paradise" -> "Cambridge" in 22-8 = 14

undergroundSystem.getAverageTime("Paradise", "Cambridge"); // return 14.00000. One trip "Paradise" -> "Cambridge", (14) / 1 = 14
```

```

undergroundSystem.getAverageTime("Leyton", "Waterloo"); // return 11.00000. Two
trips "Leyton" -> "Waterloo", (10 + 12) / 2 = 11

undergroundSystem.checkIn(10, "Leyton", 24);

undergroundSystem.getAverageTime("Leyton", "Waterloo"); // return 11.00000

undergroundSystem.checkOut(10, "Waterloo", 38); // Customer 10 "Leyton" ->
"Waterloo" in 38-24 = 14

undergroundSystem.getAverageTime("Leyton", "Waterloo"); // return 12.00000. Three
trips "Leyton" -> "Waterloo", (10 + 12 + 14) / 3 = 12

```

### Example 2:

#### Input

```

["UndergroundSystem", "checkIn", "checkOut", "getAverageTime", "checkIn", "checkOut", "getA
verageTime", "checkIn", "checkOut", "getAverageTime"]

[[[],[10,"Leyton",3],[10,"Paradise",8],[["Leyton","Paradise"],[5,"Leyton",10],[5,"Parad
ise",16],[["Leyton","Paradise"],[2,"Leyton",21],[2,"Paradise",30],[["Leyton","Paradise"
]]]

```

#### Output

```
[null,null,null,5.00000,null,null,5.50000,null,null,6.66667]
```

#### Explanation

```

UndergroundSystem undergroundSystem = new UndergroundSystem();

undergroundSystem.checkIn(10, "Leyton", 3);

undergroundSystem.checkOut(10, "Paradise", 8); // Customer 10 "Leyton" -> "Paradise"
in 8-3 = 5

undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 5.00000, (5) / 1 =
5

undergroundSystem.checkIn(5, "Leyton", 10);

undergroundSystem.checkOut(5, "Paradise", 16); // Customer 5 "Leyton" -> "Paradise"
in 16-10 = 6

undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 5.50000, (5 + 6) /
2 = 5.5

undergroundSystem.checkIn(2, "Leyton", 21);

```

```

undergroundSystem.checkOut(2, "Paradise", 30); // Customer 2 "Leyton" -> "Paradise"
in 30-21 = 9

undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 6.66667, (5 + 6 +
9) / 3 = 6.66667

```

### Constraints:

- $1 \leq id, t \leq 10^6$
- $1 \leq stationName.length, startStation.length, endStation.length \leq 10$
- All strings consist of uppercase and lowercase English letters and digits.
- There will be at most  $2 * 10^4$  calls **in total** to `checkIn`, `checkOut`, and `getAverageTime`.
- Answers within  $10^{-5}$  of the actual value will be accepted.

## 1397. Find All Good Strings

Hard

354115Add to ListShare

Given the strings `s1` and `s2` of size `n` and the string `evil`, return *the number of good strings*.

A **good** string has size `n`, it is alphabetically greater than or equal to `s1`, it is alphabetically smaller than or equal to `s2`, and it does not contain the string `evil` as a substring. Since the answer can be a huge number, return this **modulo**  $10^9 + 7$ .

### Example 1:

**Input:** `n = 2, s1 = "aa", s2 = "da", evil = "b"`

**Output:** 51

**Explanation:** There are 25 good strings starting with 'a': "aa", "ac", "ad", ..., "az". Then there are 25 good strings starting with 'c': "ca", "cc", "cd", ..., "cz" and finally there is one good string starting with 'd': "da".

### Example 2:

**Input:** `n = 8, s1 = "leetcode", s2 = "leetgoes", evil = "leet"`

**Output:** 0

**Explanation:** All strings greater than or equal to `s1` and smaller than or equal to `s2` start with the prefix "leet", therefore, there is not any good string.

### Example 3:

**Input:** `n = 2, s1 = "gx", s2 = "gz", evil = "x"`

**Output:** 2

**Constraints:**

- `s1.length == n`
- `s2.length == n`
- `s1 <= s2`
- `1 <= n <= 500`
- `1 <= evil.length <= 50`
- All strings consist of lowercase English letters.

## 1399. Count Largest Group

**Easy**

313720Add to ListShare

You are given an integer `n`.

Each number from `1` to `n` is grouped according to the sum of its digits.

Return *the number of groups that have the largest size*.

**Example 1:**

**Input:** `n = 13`

**Output:** 4

**Explanation:** There are 9 groups in total, they are grouped according sum of its digits of numbers from 1 to 13:

`[1,10], [2,11], [3,12], [4,13], [5], [6], [7], [8], [9]`.

There are 4 groups with largest size.

**Example 2:**

**Input:** `n = 2`

**Output:** 2

**Explanation:** There are 2 groups `[1], [2]` of size 1.

**Constraints:**

- $1 \leq n \leq 10^4$

## 1400. Construct K Palindrome Strings

Medium

81877Add to ListShare

Given a string  $s$  and an integer  $k$ , return `true` if you can use all the characters in  $s$  to construct  $k$  palindrome strings or `false` otherwise.

### Example 1:

**Input:**  $s = \text{"annabelle"}$ ,  $k = 2$

**Output:** `true`

**Explanation:** You can construct two palindromes using all characters in  $s$ .

Some possible constructions "anna" + "elble", "anbna" + "elle", "anellena" + "b"

### Example 2:

**Input:**  $s = \text{"leetcode"}$ ,  $k = 3$

**Output:** `false`

**Explanation:** It is impossible to construct 3 palindromes using all the characters of  $s$ .

### Example 3:

**Input:**  $s = \text{"true"}$ ,  $k = 4$

**Output:** `true`

**Explanation:** The only possible solution is to put each character in a separate string.

### Constraints:

- $1 \leq s.length \leq 10^5$
- $s$  consists of lowercase English letters.
- $1 \leq k \leq 10^5$

## 1401. Circle and Rectangle Overlapping

Medium

25659Add to ListShare

You are given a circle represented as `(radius, xCenter, yCenter)` and an axis-aligned rectangle represented as `(x1, y1, x2, y2)`, where `(x1, y1)` are the coordinates of the bottom-left corner, and `(x2, y2)` are the coordinates of the top-right corner of the rectangle.

Return `true` if the circle and rectangle are overlapped otherwise return `false`. In other words, check if there is **any** point `(xi, yi)` that belongs to the circle and the rectangle at the same time.

### Example 1:



**Input:** radius = 1, xCenter = 0, yCenter = 0, x1 = 1, y1 = -1, x2 = 3, y2 = 1

**Output:** true

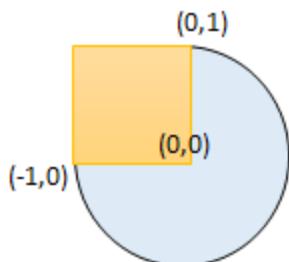
**Explanation:** Circle and rectangle share the point (1,0).

### Example 2:

**Input:** radius = 1, xCenter = 1, yCenter = 1, x1 = 1, y1 = -3, x2 = 2, y2 = -1

**Output:** false

### Example 3:



**Input:** radius = 1, xCenter = 0, yCenter = 0, x1 = -1, y1 = 0, x2 = 0, y2 = 1

**Output:** true

### Constraints:

- $1 \leq \text{radius} \leq 2000$
- $-10^4 \leq \text{xCenter}, \text{yCenter} \leq 10^4$
- $-10^4 \leq \text{x1} < \text{x2} \leq 10^4$
- $-10^4 \leq \text{y1} < \text{y2} \leq 10^4$

## 1402. Reducing Dishes

Hard

1217154 Add to List Share

A chef has collected data on the `satisfaction` level of his `n` dishes. Chef can cook any dish in 1 unit of time.

**Like-time coefficient** of a dish is defined as the time taken to cook that dish including previous dishes multiplied by its satisfaction level i.e. `time[i] * satisfaction[i]`.

Return *the maximum sum of like-time coefficient that the chef can obtain after dishes preparation.*

Dishes can be prepared in **any** order and the chef can discard some dishes to get this maximum value.

### Example 1:

**Input:** `satisfaction = [-1, -8, 0, 5, -9]`

**Output:** 14

**Explanation:** After Removing the second and last dish, the maximum total like-time coefficient will be equal to  $(-1*1 + 0*2 + 5*3 = 14)$ .

Each dish is prepared in one unit of time.

### Example 2:

**Input:** `satisfaction = [4, 3, 2]`

**Output:** 20

**Explanation:** Dishes can be prepared in any order,  $(2*1 + 3*2 + 4*3 = 20)$

### Example 3:

**Input:** `satisfaction = [-1, -4, -5]`

**Output:** 0

**Explanation:** People do not like the dishes. No dish is prepared.

**Constraints:**

- `n == satisfaction.length`
- `1 <= n <= 500`
- `-1000 <= satisfaction[i] <= 1000`

**1403. Minimum Subsequence in Non-Increasing Order****Easy**

415420Add to ListShare

Given the array `nums`, obtain a subsequence of the array whose sum of elements is **strictly greater** than the sum of the non included elements in such subsequence.

If there are multiple solutions, return the subsequence with **minimum size** and if there still exist multiple solutions, return the subsequence with the **maximum total sum** of all its elements. A subsequence of an array can be obtained by erasing some (possibly zero) elements from the array.

Note that the solution with the given constraints is guaranteed to be **unique**. Also return the answer sorted in **non-increasing** order.

**Example 1:**

**Input:** `nums = [4,3,10,9,8]`

**Output:** `[10,9]`

**Explanation:** The subsequences `[10,9]` and `[10,8]` are minimal such that the sum of their elements is strictly greater than the sum of elements not included. However, the subsequence `[10,9]` has the maximum total sum of its elements.

**Example 2:**

**Input:** `nums = [4,4,7,6,7]`

**Output:** `[7,7,6]`

**Explanation:** The subsequence `[7,7]` has the sum of its elements equal to 14 which is not strictly greater than the sum of elements not included ( $14 = 4 + 4 + 6$ ). Therefore, the subsequence `[7,6,7]` is the minimal satisfying the conditions. Note the subsequence has to be returned in non-decreasing order.

**Constraints:**

- `1 <= nums.length <= 500`
- `1 <= nums[i] <= 100`

## 1404. Number of Steps to Reduce a Number in Binary Representation to One

Medium

58049 Add to List Share

Given the binary representation of an integer as a string `s`, return *the number of steps to reduce it to 1 under the following rules:*

- If the current number is even, you have to divide it by 2.
- If the current number is odd, you have to add 1 to it.

It is guaranteed that you can always reach one for all test cases.

### Example 1:

**Input:** `s = "1101"`

**Output:** 6

**Explanation:** "1101" corresponds to number 13 in their decimal representation.

Step 1) 13 is odd, add 1 and obtain 14.

Step 2) 14 is even, divide by 2 and obtain 7.

Step 3) 7 is odd, add 1 and obtain 8.

Step 4) 8 is even, divide by 2 and obtain 4.

Step 5) 4 is even, divide by 2 and obtain 2.

Step 6) 2 is even, divide by 2 and obtain 1.

### Example 2:

**Input:** `s = "10"`

**Output:** 1

**Explanation:** "10" corresponds to number 2 in their decimal representation.

Step 1) 2 is even, divide by 2 and obtain 1.

### Example 3:

**Input:** `s = "1"`

**Output:** 0

**Constraints:**

- $1 \leq s.length \leq 500$
- $s$  consists of characters '0' or '1'
- $s[0] == '1'$

**1405. Longest Happy String****Medium**

1452208Add to ListShare

A string  $s$  is called **happy** if it satisfies the following conditions:

- $s$  only contains the letters 'a', 'b', and 'c'.
- $s$  does not contain any of "aaa", "bbb", or "ccc" as a substring.
- $s$  contains **at most**  $a$  occurrences of the letter 'a'.
- $s$  contains **at most**  $b$  occurrences of the letter 'b'.
- $s$  contains **at most**  $c$  occurrences of the letter 'c'.

Given three integers  $a$ ,  $b$ , and  $c$ , return the **longest possible happy string**. If there are multiple longest happy strings, return *any of them*. If there is no such string, return the empty string "".

A **substring** is a contiguous sequence of characters within a string.

**Example 1:****Input:**  $a = 1$ ,  $b = 1$ ,  $c = 7$ **Output:** "ccaccbcc"**Explanation:** "ccbccacc" would also be a correct answer.**Example 2:****Input:**  $a = 7$ ,  $b = 1$ ,  $c = 0$ **Output:** "aabaa"**Explanation:** It is the only correct answer in this case.**Constraints:**

- $0 \leq a, b, c \leq 100$
- $a + b + c > 0$

**1406. Stone Game III**

**Hard**

110826Add to ListShare

Alice and Bob continue their games with piles of stones. There are several stones **arranged in a row**, and each stone has an associated value which is an integer given in the array `stoneValue`.

Alice and Bob take turns, with Alice starting first. On each player's turn, that player can take `1`, `2`, or `3` stones from the **first** remaining stones in the row.

The score of each player is the sum of the values of the stones taken. The score of each player is `0` initially.

The objective of the game is to end with the highest score, and the winner is the player with the highest score and there could be a tie. The game continues until all the stones have been taken.

Assume Alice and Bob **play optimally**.

Return `"Alice"` if Alice will win, `"Bob"` if Bob will win, or `"Tie"` if they will end the game with the same score.

**Example 1:**

**Input:** `values = [1,2,3,7]`

**Output:** `"Bob"`

**Explanation:** Alice will always lose. Her best move will be to take three piles and the score become 6. Now the score of Bob is 7 and Bob wins.

**Example 2:**

**Input:** `values = [1,2,3,-9]`

**Output:** `"Alice"`

**Explanation:** Alice must choose all the three piles at the first move to win and leave Bob with negative score.

If Alice chooses one pile her score will be 1 and the next move Bob's score becomes 5. In the next move, Alice will take the pile with value = `-9` and lose.

If Alice chooses two piles her score will be 3 and the next move Bob's score becomes 3. In the next move, Alice will take the pile with value = `-9` and also lose.

Remember that both play optimally so here Alice will choose the scenario that makes her win.

**Example 3:**

**Input:** `values = [1,2,3,6]`

**Output:** "Tie"

**Explanation:** Alice cannot win this game. She can end the game in a draw if she decided to choose all the first three piles, otherwise she will lose.

**Constraints:**

- $1 \leq \text{stoneValue.length} \leq 5 * 10^4$
- $-1000 \leq \text{stoneValue}[i] \leq 1000$

## 1407. Top Travellers

Easy

30232Add to ListShare

SQL Schema

Table: `Users`

Column Name	Type
<code>id</code>	<code>int</code>
<code>name</code>	<code>varchar</code>

`id` is the primary key for this table.

`name` is the name of the user.

Table: `Rides`

Column Name	Type
<code>id</code>	<code>int</code>
<code>user_id</code>	<code>int</code>

distance	int
id	is the primary key for this table.
user_id	is the id of the user who traveled the distance "distance".

Write an SQL query to report the distance traveled by each user.

Return the result table ordered by `travelled_distance` in **descending order**, if two or more users traveled the same distance, order them by their `name` in **ascending order**.

The query result format is in the following example.

### Example 1:

**Input:**

Users table:

id	name
1	Alice
2	Bob
3	Alex
4	Donald
7	Lee
13	Jonathan
19	Elvis

Rides table:

id	user_id	distance
----	---------	----------

1	1	120	
2	2	317	
3	3	222	
4	7	100	
5	13	312	
6	19	50	
7	7	120	
8	19	400	
9	7	230	

**Output:**

name	travelled_distance	
Elvis	450	
Lee	450	
Bob	317	
Jonathan	312	
Alex	222	
Alice	120	
Donald	0	

**Explanation:**

Elvis and Lee traveled 450 miles, Elvis is the top traveler as his name is alphabetically smaller than Lee.

Bob, Jonathan, Alex, and Alice have only one ride and we just order them by the total distances of the ride.

Donald did not have any rides, the distance traveled by him is 0.

## 1408. String Matching in an Array

Easy

61277Add to ListShare

Given an array of string `words`. Return all strings in `words` which is substring of another word in **any** order.

String `words[i]` is substring of `words[j]`, if can be obtained removing some characters to left and/or right side of `words[j]`.

### Example 1:

**Input:** `words = ["mass", "as", "hero", "superhero"]`

**Output:** `["as", "hero"]`

**Explanation:** "as" is substring of "mass" and "hero" is substring of "superhero".

`["hero", "as"]` is also a valid answer.

### Example 2:

**Input:** `words = ["leetcode", "et", "code"]`

**Output:** `["et", "code"]`

**Explanation:** "et", "code" are substring of "leetcode".

### Example 3:

**Input:** `words = ["blue", "green", "bu"]`

**Output:** `[]`

### Constraints:

- `1 <= words.length <= 100`
- `1 <= words[i].length <= 30`
- `words[i]` contains only lowercase English letters.
- It's **guaranteed** that `words[i]` will be unique.

## 1409. Queries on a Permutation With Key

Medium

371556Add to ListShare

Given the array `queries` of positive integers between `1` and `m`, you have to process all `queries[i]` (from `i=0` to `i=queries.length-1`) according to the following rules:

- In the beginning, you have the permutation `P=[1,2,3,...,m]`.
- For the current `i`, find the position of `queries[i]` in the permutation `P` (**indexing from 0**) and then move this at the beginning of the permutation `P`. Notice that the position of `queries[i]` in `P` is the result for `queries[i]`.

Return an array containing the result for the given `queries`.

### Example 1:

**Input:** `queries = [3,1,2,1]`, `m = 5`

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

**Explanation:** The queries are processed as follow:

For `i=0`: `queries[i]=3`, `P=[1,2,3,4,5]`, position of `3` in `P` is `2`, then we move `3` to the beginning of `P` resulting in `P=[3,1,2,4,5]`.

For `i=1`: `queries[i]=1`, `P=[3,1,2,4,5]`, position of `1` in `P` is `1`, then we move `1` to the beginning of `P` resulting in `P=[1,3,2,4,5]`.

For `i=2`: `queries[i]=2`, `P=[1,3,2,4,5]`, position of `2` in `P` is `2`, then we move `2` to the beginning of `P` resulting in `P=[2,1,3,4,5]`.

For `i=3`: `queries[i]=1`, `P=[2,1,3,4,5]`, position of `1` in `P` is `1`, then we move `1` to the beginning of `P` resulting in `P=[1,2,3,4,5]`.

Therefore, the array containing the result is `[2,1,2,1]`.

### Example 2:

**Input:** `queries = [4,1,2,2]`, `m = 4`

**Output:** `[3,1,2,0]`

### Example 3:

**Input:** `queries = [7,5,5,8,3]`, `m = 8`

**Output:** `[6,5,0,7,5]`

### Constraints:

- `1 <= m <= 10^3`

- `1 <= queries.length <= m`
- `1 <= queries[i] <= m`

## 1410. HTML Entity Parser

Medium

145281Add to ListShare

**HTML entity parser** is the parser that takes HTML code as input and replace all the entities of the special characters by the characters itself.

The special characters and their entities for HTML are:

- **Quotation Mark:** the entity is `&quot;` and symbol character is `"`.
- **Single Quote Mark:** the entity is `&apos;` and symbol character is `'`.
- **Ampersand:** the entity is `&amp;` and symbol character is `&`.
- **Greater Than Sign:** the entity is `&gt;` and symbol character is `>`.
- **Less Than Sign:** the entity is `&lt;` and symbol character is `<`.
- **Slash:** the entity is `&frasl;` and symbol character is `/`.

Given the input `text` string to the HTML parser, you have to implement the entity parser.

Return *the text after replacing the entities by the special characters*.

### Example 1:

**Input:** `text = "&amp; is an HTML entity but &ambassador; is not."`

**Output:** `"& is an HTML entity but &ambassador; is not."`

**Explanation:** The parser will replace the `&amp;` entity by `&`

### Example 2:

**Input:** `text = "and I quote: &quot;...&quot;"`

**Output:** `"and I quote: \"...\""`

### Constraints:

- `1 <= text.length <= 105`
- The string may contain any possible characters out of all the 256 ASCII characters.

## 1411. Number of Ways to Paint N × 3 Grid

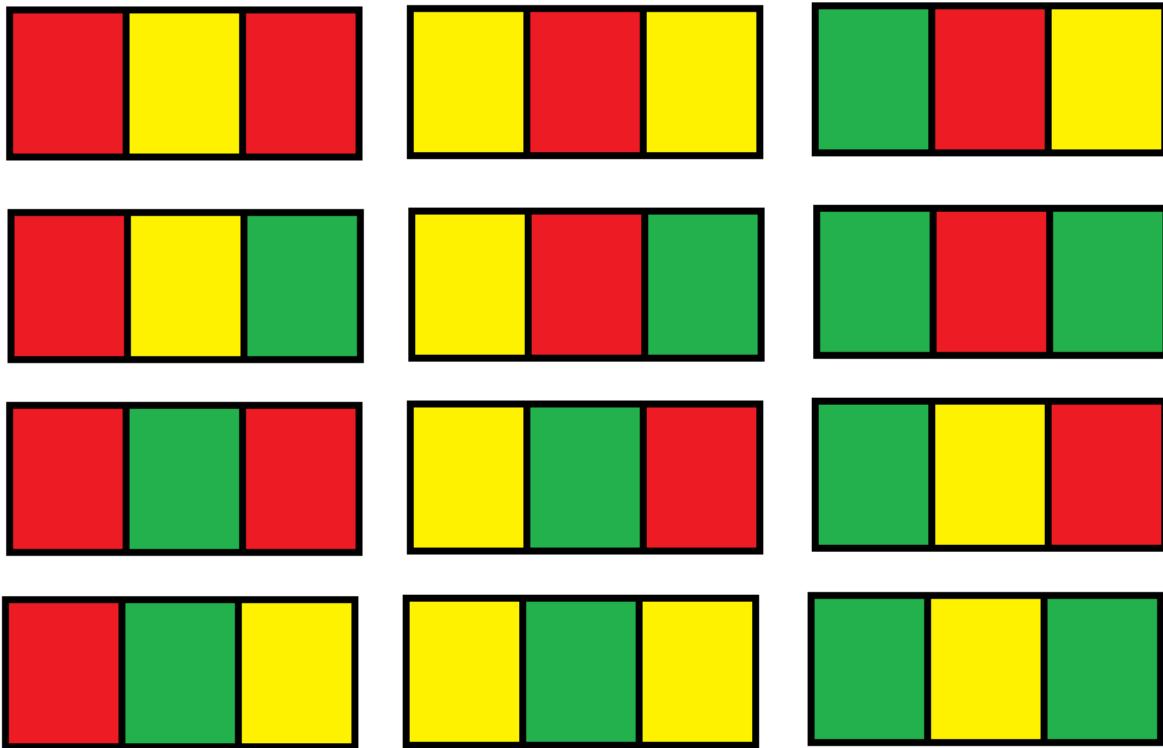
Hard

87645Add to ListShare

You have a grid of size  $n \times 3$  and you want to paint each cell of the grid with exactly one of the three colors: **Red**, **Yellow**, or **Green** while making sure that no two adjacent cells have the same color (i.e., no two cells that share vertical or horizontal sides have the same color).

Given  $n$  the number of rows of the grid, return *the number of ways* you can paint this grid. As the answer may grow large, the answer **must be** computed modulo  $10^9 + 7$ .

**Example 1:**



**Input:**  $n = 1$

**Output:** 12

**Explanation:** There are 12 possible ways to paint the grid as shown.

**Example 2:**

**Input:**  $n = 5000$

**Output:** 30228214

**Constraints:**

- `n == grid.length`
- `1 <= n <= 5000`

## 1413. Minimum Value to Get Positive Step by Step Sum

Easy

1153243Add to ListShare

Given an array of integers `nums`, you start with an initial **positive** value `startValue`.

In each iteration, you calculate the step by step sum of `startValue` plus elements in `nums` (from left to right).

Return the minimum **positive** value of `startValue` such that the step by step sum is never less than 1.

### Example 1:

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

**Output:** 5

**Explanation:** If you choose `startValue = 4`, in the third iteration your step by step sum is less than 1.

**step by step sum**

`startValue = 4 | startValue = 5 | nums`

`(4 -3 ) = 1 | (5 -3 ) = 2 | -3`

`(1 +2 ) = 3 | (2 +2 ) = 4 | 2`

`(3 -3 ) = 0 | (4 -3 ) = 1 | -3`

`(0 +4 ) = 4 | (1 +4 ) = 5 | 4`

`(4 +2 ) = 6 | (5 +2 ) = 7 | 2`

### Example 2:

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

**Output:** 1

**Explanation:** Minimum start value should be positive.

### Example 3:

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

**Output:** 5

**Constraints:**

- $1 \leq \text{nums.length} \leq 100$
- $-100 \leq \text{nums}[i] \leq 100$

**1414. Find the Minimum Number of Fibonacci Numbers Whose Sum Is K****Medium**

79555Add to ListShare

Given an integer  $k$ , return the minimum number of Fibonacci numbers whose sum is equal to  $k$ . The same Fibonacci number can be used multiple times.

The Fibonacci numbers are defined as:

- $F_1 = 1$
- $F_2 = 1$
- $F_n = F_{n-1} + F_{n-2}$  for  $n > 2$ .

It is guaranteed that for the given constraints we can always find such Fibonacci numbers that sum up to  $k$ .

**Example 1:****Input:**  $k = 7$ **Output:** 2**Explanation:** The Fibonacci numbers are: 1, 1, 2, 3, 5, 8, 13, ...For  $k = 7$  we can use  $2 + 5 = 7$ .**Example 2:****Input:**  $k = 10$ **Output:** 2**Explanation:** For  $k = 10$  we can use  $2 + 8 = 10$ .**Example 3:****Input:**  $k = 19$ **Output:** 3**Explanation:** For  $k = 19$  we can use  $1 + 5 + 13 = 19$ .

**Constraints:**

- $1 \leq k \leq 10^9$

**1415. The k-th Lexicographical String of All Happy Strings of Length n****Medium**

75619Add to ListShare

A **happy string** is a string that:

- consists only of letters of the set `['a', 'b', 'c']`.
- $s[i] \neq s[i + 1]$  for all values of  $i$  from  $1$  to  $s.length - 1$  (string is 1-indexed).

For example, strings `"abc"`, `"ac"`, `"b"` and `"abcbabcbcb"` are all happy strings and strings `"aa"`, `"baa"` and `"ababbc"` are not happy strings.

Given two integers  $n$  and  $k$ , consider a list of all happy strings of length  $n$  sorted in lexicographical order.

Return *the kth string* of this list or return an **empty string** if there are less than  $k$  happy strings of length  $n$ .

**Example 1:****Input:**  $n = 1, k = 3$ **Output:** `"c"`

**Explanation:** The list `["a", "b", "c"]` contains all happy strings of length 1. The third string is `"c"`.

**Example 2:****Input:**  $n = 1, k = 4$ **Output:** `""`

**Explanation:** There are only 3 happy strings of length 1.

**Example 3:****Input:**  $n = 3, k = 9$ **Output:** `"cab"`

**Explanation:** There are 12 different happy string of length 3 ["aba", "abc", "aca", "acb", "bab", "bac", "bca", "bcb", "cab", "cac", "cba", "cbc"]. You will find the 9<sup>th</sup> string = "cab"

### Constraints:

- $1 \leq n \leq 10$
- $1 \leq k \leq 100$

## 1416. Restore The Array

Hard

42414Add to ListShare

A program was supposed to print an array of integers. The program forgot to print whitespaces and the array is printed as a string of digits `s` and all we know is that all integers in the array were in the range `[1, k]` and there are no leading zeros in the array.

Given the string `s` and the integer `k`, return *the number of the possible arrays that can be printed as s using the mentioned program*. Since the answer may be very large, return it **modulo  $10^9 + 7$** .

### Example 1:

**Input:** `s = "1000", k = 10000`

**Output:** 1

**Explanation:** The only possible array is [1000]

### Example 2:

**Input:** `s = "1000", k = 10`

**Output:** 0

**Explanation:** There cannot be an array that was printed this way and has all integer  $\geq 1$  and  $\leq 10$ .

### Example 3:

**Input:** `s = "1317", k = 2000`

**Output:** 8

**Explanation:** Possible arrays are  
`[1317], [131, 7], [13, 17], [1, 317], [13, 1, 7], [1, 31, 7], [1, 3, 17], [1, 3, 1, 7]`

**Constraints:**

- $1 \leq s.length \leq 10^5$
- $s$  consists of only digits and does not contain leading zeros.
- $1 \leq k \leq 10^9$

**1417. Reformat The String****Easy**

44081Add to ListShare

You are given an alphanumeric string  $s$ . (**Alphanumeric string** is a string consisting of lowercase English letters and digits).

You have to find a permutation of the string where no letter is followed by another letter and no digit is followed by another digit. That is, no two adjacent characters have the same type.

Return *the reformatted string* or return **an empty string** if it is impossible to reformat the string.

**Example 1:**

**Input:**  $s = "a0b1c2"$

**Output:**  $"0a1b2c"$

**Explanation:** No two adjacent characters have the same type in  $"0a1b2c"$ .  $"a0b1c2"$ ,  $"0a1b2c"$ ,  $"0c2a1b"$  are also valid permutations.

**Example 2:**

**Input:**  $s = "leetcode"$

**Output:**  $"$

**Explanation:** "leetcode" has only characters so we cannot separate them by digits.

**Example 3:**

**Input:**  $s = "1229857369"$

**Output:**  $"$

**Explanation:** "1229857369" has only digits so we cannot separate them by characters.

**Constraints:**

- $1 \leq s.length \leq 500$

- `s` consists of only lowercase English letters and/or digits.

## 1418. Display Table of Food Orders in a Restaurant

Medium

240381Add to ListShare

Given the array `orders`, which represents the orders that customers have done in a restaurant. More specifically `orders[i]=[customerNamei,tableNumberi,foodItemi]` where `customerNamei` is the name of the customer, `tableNumberi` is the table customer sit at, and `foodItemi` is the item customer orders.

*Return the restaurant's "**display table**". The "**display table**" is a table whose row entries denote how many of each food item each table ordered. The first column is the table number and the remaining columns correspond to each food item in alphabetical order. The first row should be a header whose first column is "Table", followed by the names of the food items. Note that the customer names are not part of the table. Additionally, the rows should be sorted in numerically increasing order.*

### Example 1:

**Input:** `orders = [["David","3","Ceviche"],["Corina","10","Beef Burrito"],["David","3","Fried Chicken"],["Carla","5","Water"],["Carla","5","Ceviche"],["Rous","3","Ceviche"]]`

**Output:** `[["Table","Beef Burrito","Ceviche","Fried Chicken","Water"],[["3","0","2","1","0"],[["5","0","1","0","1"],[["10","1","0","0","0"]]`

### Explanation:

The displaying table looks like:

**Table,Beef Burrito,Ceviche,Fried Chicken,Water**

3	,0	,2	,1	,0
5	,0	,1	,0	,1
10	,1	,0	,0	,0

For the table 3: David orders "Ceviche" and "Fried Chicken", and Rous orders "Ceviche".

For the table 5: Carla orders "Water" and "Ceviche".

For the table 10: Corina orders "Beef Burrito".

### Example 2:

**Input:** orders = [["James", "12", "Fried Chicken"], ["Ratesh", "12", "Fried Chicken"], ["Amadeus", "12", "Fried Chicken"], ["Adam", "1", "Canadian Waffles"], ["Brianna", "1", "Canadian Waffles"]]

**Output:** [["Table", "Canadian Waffles", "Fried Chicken"], ["1", "2", "0"], ["12", "0", "3"]]

**Explanation:**

For the table 1: Adam and Brianna order "Canadian Waffles".

For the table 12: James, Ratesh and Amadeus order "Fried Chicken".

**Example 3:**

**Input:** orders = [["Laura", "2", "Bean Burrito"], ["Jhon", "2", "Beef Burrito"], ["Melissa", "2", "Soda"]]

**Output:** [["Table", "Bean Burrito", "Beef Burrito", "Soda"], ["2", "1", "1", "1"]]

**Constraints:**

- $1 \leq \text{orders.length} \leq 5 * 10^4$
- $\text{orders[i].length} == 3$
- $1 \leq \text{customerName}_i.\text{length}, \text{foodItem}_i.\text{length} \leq 20$
- $\text{customerName}_i$  and  $\text{foodItem}_i$  consist of lowercase and uppercase English letters and the space character.
- $\text{tableNumber}_i$  is a valid integer between 1 and 500.

## 1419. Minimum Number of Frogs Croaking

Medium

73650Add to ListShare

You are given the string `croakOfFrogs`, which represents a combination of the string `"croak"` from different frogs, that is, multiple frogs can croak at the same time, so multiple `"croak"` are mixed.

*Return the minimum number of different frogs to finish all the croaks in the given string.*

A valid `"croak"` means a frog is printing five letters `'c'`, `'r'`, `'o'`, `'a'`, and `'k'` **sequentially**. The frogs have to print all five letters to finish a croak. If the given string is not a combination of a valid `"croak"` return `-1`.

**Example 1:**

**Input:** croakOfFrogs = "croakcroak"

**Output:** 1

**Explanation:** One frog yelling "croak" twice.

**Example 2:**

**Input:** croakOfFrogs = "crcoakroak"

**Output:** 2

**Explanation:** The minimum number of frogs is two.

The first frog could yell "crcoakroak".

The second frog could yell later "crcoakroak".

**Example 3:**

**Input:** croakOfFrogs = "croakcrook"

**Output:** -1

**Explanation:** The given string is an invalid combination of "croak" from different frogs.

**Constraints:**

- $1 \leq \text{croakOfFrogs.length} \leq 10^5$
- `croakOfFrogs` is either 'c', 'r', 'o', 'a', or 'k'.

## 1420. Build Array Where You Can Find The Maximum Exactly K Comparisons

Hard

45010Add to ListShare

You are given three integers `n`, `m` and `k`. Consider the following algorithm to find the maximum element of an array of positive integers:

```

maximum_value = -1
maximum_index = -1
search_cost = 0
n = arr.length
for (i = 0; i < n; i++) {
    if (maximum_value < arr[i]) {
        maximum_value = arr[i]
        maximum_index = i
        search_cost = search_cost + 1
    }
}
return maximum_index

```

You should build the array `arr` which has the following properties:

- `arr` has exactly `n` integers.
- `1 <= arr[i] <= m` where  $(0 <= i < n)$ .
- After applying the mentioned algorithm to `arr`, the value `search_cost` is equal to `k`.

Return *the number of ways* to build the array `arr` under the mentioned conditions. As the answer may grow large, the answer **must be** computed modulo  $10^9 + 7$ .

### Example 1:

**Input:** `n = 2, m = 3, k = 1`

**Output:** `6`

**Explanation:** The possible arrays are `[1, 1], [2, 1], [2, 2], [3, 1], [3, 2] [3, 3]`

### Example 2:

**Input:** `n = 5, m = 2, k = 3`

**Output:** `0`

**Explanation:** There are no possible arrays that satisfy the mentioned conditions.

### Example 3:

**Input:** n = 9, m = 1, k = 1

**Output:** 1

**Explanation:** The only possible array is [1, 1, 1, 1, 1, 1, 1, 1, 1]

**Constraints:**

- $1 \leq n \leq 50$
- $1 \leq m \leq 100$
- $0 \leq k \leq n$

## 1422. Maximum Score After Splitting a String

Easy

53331Add to ListShare

Given a string `s` of zeros and ones, *return the maximum score after splitting the string into two **non-empty** substrings* (i.e. **left** substring and **right** substring).

The score after splitting a string is the number of **zeros** in the **left** substring plus the number of **ones** in the **right** substring.

**Example 1:**

**Input:** s = "011101"

**Output:** 5

**Explanation:**

All possible ways of splitting s into two non-empty substrings are:

left = "0" and right = "11101", score = 1 + 4 = 5

left = "01" and right = "1101", score = 1 + 3 = 4

left = "011" and right = "101", score = 1 + 2 = 3

left = "0111" and right = "01", score = 1 + 1 = 2

left = "01110" and right = "1", score = 2 + 1 = 3

**Example 2:**

**Input:** s = "00111"

**Output:** 5

**Explanation:** When `left = "00"` and `right = "111"`, we get the maximum score =  $2 + 3 = 5$

**Example 3:**

**Input:** `s = "1111"`

**Output:** 3

**Constraints:**

- $2 \leq s.length \leq 500$
- The string `s` consists of characters '0' and '1' only.

## 1423. Maximum Points You Can Obtain from Cards

Medium

4723174Add to ListShare

There are several cards **arranged in a row**, and each card has an associated number of points. The points are given in the integer array `cardPoints`.

In one step, you can take one card from the beginning or from the end of the row. You have to take exactly `k` cards.

Your score is the sum of the points of the cards you have taken.

Given the integer array `cardPoints` and the integer `k`, return the *maximum score* you can obtain.

**Example 1:**

**Input:** `cardPoints = [1,2,3,4,5,6,1], k = 3`

**Output:** 12

**Explanation:** After the first step, your score will always be 1. However, choosing the rightmost card first will maximize your total score. The optimal strategy is to take the three cards on the right, giving a final score of  $1 + 6 + 5 = 12$ .

**Example 2:**

**Input:** `cardPoints = [2,2,2], k = 2`

**Output:** 4

**Explanation:** Regardless of which two cards you take, your score will always be 4.

**Example 3:**

**Input:** cardPoints = [9,7,7,9,7,7,9], k = 7

**Output:** 55

**Explanation:** You have to take all the cards. Your score is the sum of points of all cards.

**Constraints:**

- $1 \leq \text{cardPoints.length} \leq 10^5$
- $1 \leq \text{cardPoints}[i] \leq 10^4$
- $1 \leq k \leq \text{cardPoints.length}$

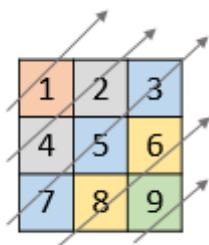
## 1424. Diagonal Traverse II

Medium

92980Add to ListShare

Given a 2D integer array `nums`, return *all elements of `nums` in diagonal order as shown in the below images.*

**Example 1:**



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

**Output:** [1,4,2,7,5,3,8,6,9]

**Example 2:**

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

**Input:** nums = [[1,2,3,4,5],[6,7],[8],[9,10,11],[12,13,14,15,16]]

**Output:** [1,6,2,8,7,3,9,4,12,10,5,13,11,14,15,16]

### Constraints:

- $1 \leq \text{nums.length} \leq 10^5$
- $1 \leq \text{nums[i].length} \leq 10^5$
- $1 \leq \text{sum}(\text{nums[i].length}) \leq 10^5$
- $1 \leq \text{nums[i][j]} \leq 10^5$

## 1425. Constrained Subsequence Sum

Hard

96539Add to ListShare

Given an integer array `nums` and an integer `k`, return the maximum sum of a **non-empty** subsequence of that array such that for every two **consecutive** integers in the subsequence, `nums[i]` and `nums[j]`, where  $i < j$ , the condition  $j - i \leq k$  is satisfied.

A *subsequence* of an array is obtained by deleting some number of elements (can be zero) from the array, leaving the remaining elements in their original order.

### Example 1:

**Input:** nums = [10,2,-10,5,20], k = 2

**Output:** 37

**Explanation:** The subsequence is [10, 2, 5, 20].

### Example 2:

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

**Output:** -1

**Explanation:** The subsequence must be non-empty, so we choose the largest number.

**Example 3:**

**Input:** `nums = [10, -2, -10, -5, 20]`, `k = 2`

**Output:** 23

**Explanation:** The subsequence is `[10, -2, -5, 20]`.

**Constraints:**

- `1 <= k <= nums.length <= 105`
- `-104 <= nums[i] <= 104`

## 1431. Kids With the Greatest Number of Candies

**Easy**

1956300Add to ListShare

There are `n` kids with candies. You are given an integer array `candies`, where each `candies[i]` represents the number of candies the `ith` kid has, and an integer `extraCandies`, denoting the number of extra candies that you have.

Return a boolean array `result` of length `n`, where `result[i]` is true if, after giving the `ith` kid all the `extraCandies`, they will have the **greatest** number of candies among all the kids, or false otherwise.

Note that **multiple** kids can have the **greatest** number of candies.

**Example 1:**

**Input:** `candies = [2,3,5,1,3]`, `extraCandies = 3`

**Output:** `[true,true,true,false,true]`

**Explanation:** If you give all `extraCandies` to:

- Kid 1, they will have  $2 + 3 = 5$  candies, which is the greatest among the kids.
- Kid 2, they will have  $3 + 3 = 6$  candies, which is the greatest among the kids.
- Kid 3, they will have  $5 + 3 = 8$  candies, which is the greatest among the kids.
- Kid 4, they will have  $1 + 3 = 4$  candies, which is not the greatest among the kids.
- Kid 5, they will have  $3 + 3 = 6$  candies, which is the greatest among the kids.

**Example 2:**

**Input:** candies = [4,2,1,1,2], extraCandies = 1

**Output:** [true, false, false, false, false]

**Explanation:** There is only 1 extra candy.

Kid 1 will always have the greatest number of candies, even if a different kid is given the extra candy.

**Example 3:**

**Input:** candies = [12,1,12], extraCandies = 10

**Output:** [true, false, true]

**Constraints:**

- $n == \text{candies.length}$
- $2 \leq n \leq 100$
- $1 \leq \text{candies}[i] \leq 100$
- $1 \leq \text{extraCandies} \leq 50$

## 1432. Max Difference You Can Get From Changing an Integer

Medium

157205Add to ListShare

You are given an integer  $\text{num}$ . You will apply the following steps exactly **two** times:

- Pick a digit  $x$  ( $0 \leq x \leq 9$ ).
- Pick another digit  $y$  ( $0 \leq y \leq 9$ ). The digit  $y$  can be equal to  $x$ .
- Replace all the occurrences of  $x$  in the decimal representation of  $\text{num}$  by  $y$ .
- The new integer **cannot** have any leading zeros, also the new integer **cannot** be 0.

Let  $a$  and  $b$  be the results of applying the operations to  $\text{num}$  the first and second times, respectively.

Return *the max difference* between  $a$  and  $b$ .

**Example 1:**

**Input:** num = 555

**Output:** 888

**Explanation:** The first time pick  $x = 5$  and  $y = 9$  and store the new integer in  $a$ .

The second time pick  $x = 5$  and  $y = 1$  and store the new integer in  $b$ .

We have now  $a = 999$  and  $b = 111$  and max difference = 888

### Example 2:

**Input:** num = 9

**Output:** 8

**Explanation:** The first time pick  $x = 9$  and  $y = 9$  and store the new integer in  $a$ .

The second time pick  $x = 9$  and  $y = 1$  and store the new integer in  $b$ .

We have now  $a = 9$  and  $b = 1$  and max difference = 8

### Constraints:

- $1 \leq \text{num} \leq 10^8$

## 1432. Max Difference You Can Get From Changing an Integer

Medium

157205Add to ListShare

You are given an integer  $\text{num}$ . You will apply the following steps exactly **two** times:

- Pick a digit  $x$  ( $0 \leq x \leq 9$ ).
- Pick another digit  $y$  ( $0 \leq y \leq 9$ ). The digit  $y$  can be equal to  $x$ .
- Replace all the occurrences of  $x$  in the decimal representation of  $\text{num}$  by  $y$ .
- The new integer **cannot** have any leading zeros, also the new integer **cannot** be 0.

Let  $a$  and  $b$  be the results of applying the operations to  $\text{num}$  the first and second times, respectively.

Return *the max difference* between  $a$  and  $b$ .

### Example 1:

**Input:** num = 555

**Output:** 888

**Explanation:** The first time pick  $x = 5$  and  $y = 9$  and store the new integer in  $a$ .

The second time pick  $x = 5$  and  $y = 1$  and store the new integer in  $b$ .

We have now  $a = 999$  and  $b = 111$  and max difference = 888

**Example 2:****Input:** num = 9**Output:** 8**Explanation:** The first time pick x = 9 and y = 9 and store the new integer in a.

The second time pick x = 9 and y = 1 and store the new integer in b.

We have now a = 9 and b = 1 and max difference = 8

**Constraints:**

- $1 \leq \text{num} \leq 10^8$

**1434. Number of Ways to Wear Different Hats to Each Other****Hard**

6958Add to ListShare

There are  $n$  people and  $40$  types of hats labeled from  $1$  to  $40$ .Given a 2D integer array `hats`, where `hats[i]` is a list of all hats preferred by the  $i^{\text{th}}$  person.Return *the number of ways that the  $n$  people wear different hats to each other.*Since the answer may be too large, return it modulo  $10^9 + 7$ .**Example 1:****Input:** hats = [[3,4],[4,5],[5]]**Output:** 1**Explanation:** There is only one way to choose hats given the conditions.

First person choose hat 3, Second person choose hat 4 and last one hat 5.

**Example 2:****Input:** hats = [[3,5,1],[3,5]]**Output:** 4**Explanation:** There are 4 ways to choose hats:

(3,5), (5,3), (1,3) and (1,5)

**Example 3:**

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

**Output:** 24

**Explanation:** Each person can choose hats labeled from 1 to 4.

Number of Permutations of (1,2,3,4) = 24.

**Constraints:**

- n == hats.length
- 1 <= n <= 10
- 1 <= hats[i].length <= 40
- 1 <= hats[i][j] <= 40
- hats[i] contains a list of **unique** integers.

**1436. Destination City**

**Easy**

111160Add to ListShare

You are given the array paths, where paths[i] = [cityA<sub>i</sub>, cityB<sub>i</sub>] means there exists a direct path going from cityA<sub>i</sub> to cityB<sub>i</sub>. Return the destination city, that is, the city without any path outgoing to another city.

It is guaranteed that the graph of paths forms a line without any loop, therefore, there will be exactly one destination city.

**Example 1:**

**Input:** paths = [["London", "New York"], ["New York", "Lima"], ["Lima", "Sao Paulo"]]

**Output:** "Sao Paulo"

**Explanation:** Starting at "London" city you will reach "Sao Paulo" city which is the destination city. Your trip consist of: "London" -> "New York" -> "Lima" -> "Sao Paulo".

**Example 2:**

**Input:** paths = [["B", "C"], ["D", "B"], ["C", "A"]]

**Output:** "A"

**Explanation:** All possible trips are:

"D" -> "B" -> "C" -> "A".

"B" -> "C" -> "A".

"C" -> "A".

"A".

Clearly the destination city is "A".

**Example 3:**

**Input:** paths = [["A", "Z"]]

**Output:** "Z"

**Constraints:**

- $1 \leq \text{paths.length} \leq 100$
- $\text{paths[i].length} == 2$
- $1 \leq \text{cityA}_i.\text{length}, \text{cityB}_i.\text{length} \leq 10$
- $\text{cityA}_i \neq \text{cityB}_i$
- All strings consist of lowercase and uppercase English letters and the space character.

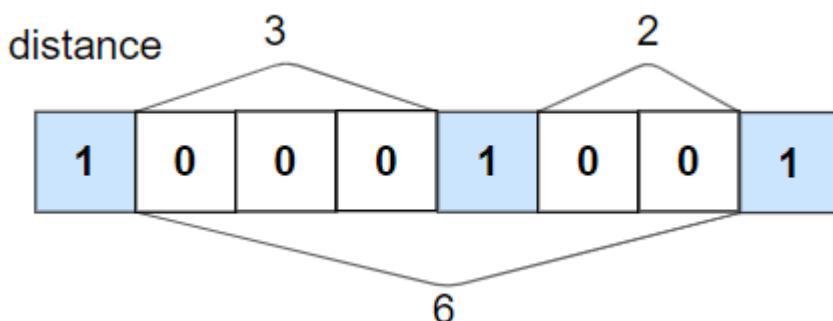
## 1437. Check If All 1's Are at Least Length K Places Away

Easy

459198Add to ListShare

Given a binary array `nums` and an integer `k`, return `true` if all `1`'s are at least `k` places away from each other, otherwise return `false`.

**Example 1:**

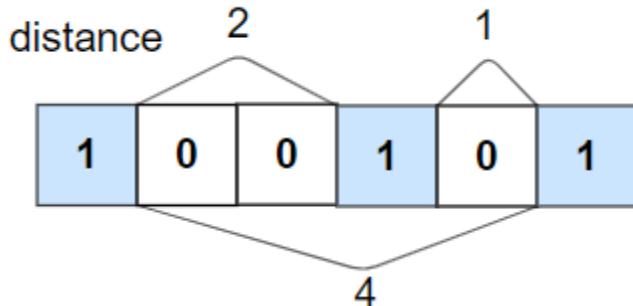


**Input:** nums = [1,0,0,0,1,0,0,1], k = 2

**Output:** true

**Explanation:** Each of the 1s are at least 2 places away from each other.

**Example 2:**



**Input:** `nums = [1,0,0,1,0,1]`, `k = 2`

**Output:** `false`

**Explanation:** The second 1 and third 1 are only one apart from each other.

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^5$
- $0 \leq k \leq \text{nums.length}$
- `nums[i]` is 0 or 1

## 1438. Longest Continuous Subarray With Absolute Diff Less Than or Equal to Limit

Medium

2580110Add to ListShare

Given an array of integers `nums` and an integer `limit`, return the size of the longest **non-empty** subarray such that the absolute difference between any two elements of this subarray is less than or equal to `limit`.

**Example 1:**

**Input:** `nums = [8,2,4,7]`, `limit = 4`

**Output:** `2`

**Explanation:** All subarrays are:

`[8]` with maximum absolute diff  $|8-8| = 0 \leq 4$ .

`[8,2]` with maximum absolute diff  $|8-2| = 6 > 4$ .

[8,2,4] with maximum absolute diff  $|8-2| = 6 > 4$ .  
 [8,2,4,7] with maximum absolute diff  $|8-2| = 6 > 4$ .  
 [2] with maximum absolute diff  $|2-2| = 0 \leq 4$ .  
 [2,4] with maximum absolute diff  $|2-4| = 2 \leq 4$ .  
 [2,4,7] with maximum absolute diff  $|2-7| = 5 > 4$ .  
 [4] with maximum absolute diff  $|4-4| = 0 \leq 4$ .  
 [4,7] with maximum absolute diff  $|4-7| = 3 \leq 4$ .  
 [7] with maximum absolute diff  $|7-7| = 0 \leq 4$ .

Therefore, the size of the longest subarray is 2.

### Example 2:

**Input:** `nums = [10,1,2,4,7,2]`, `limit = 5`

**Output:** 4

**Explanation:** The subarray `[2,4,7,2]` is the longest since the maximum absolute diff is  $|2-7| = 5 \leq 5$ .

### Example 3:

**Input:** `nums = [4,2,2,2,4,4,2,2]`, `limit = 0`

**Output:** 3

### Constraints:

- $1 \leq \text{nums.length} \leq 10^5$
- $1 \leq \text{nums}[i] \leq 10^9$
- $0 \leq \text{limit} \leq 10^9$

## 1439. Find the Kth Smallest Sum of a Matrix With Sorted Rows

Hard

95814Add to ListShare

You are given an `m x n` matrix `mat` that has its rows sorted in non-decreasing order and an integer `k`.

You are allowed to choose **exactly one element** from each row to form an array.

Return *the `kth` smallest array sum among all possible arrays*.

**Example 1:****Input:** mat = [[1,3,11],[2,4,6]], k = 5**Output:** 7**Explanation:** Choosing one element from each row, the first k smallest sum are:

[1,2], [1,4], [3,2], [3,4], [1,6]. Where the 5th sum is 7.

**Example 2:****Input:** mat = [[1,3,11],[2,4,6]], k = 9**Output:** 17**Example 3:****Input:** mat = [[1,10,10],[1,4,5],[2,3,6]], k = 7**Output:** 9**Explanation:** Choosing one element from each row, the first k smallest sum are:

[1,1,2], [1,1,3], [1,4,2], [1,4,3], [1,1,6], [1,5,2], [1,5,3]. Where the 7th sum is 9.

**Constraints:**

- $m == \text{mat.length}$
- $n == \text{mat.length}[i]$
- $1 \leq m, n \leq 40$
- $1 \leq \text{mat}[i][j] \leq 5000$
- $1 \leq k \leq \min(200, n^m)$
- $\text{mat}[i]$  is a non-decreasing array.

**1441. Build an Array With Stack Operations****Medium**

445Add to ListShare

You are given an integer array `target` and an integer `n`.

You have an empty stack with the two following operations:

- "`Push`": pushes an integer to the top of the stack.
- "`Pop`": removes the integer on the top of the stack.

You also have a stream of the integers in the range `[1, n]`.

Use the two stack operations to make the numbers in the stack (from the bottom to the top) equal to `target`. You should follow the following rules:

- If the stream of the integers is not empty, pick the next integer from the stream and push it to the top of the stack.
- If the stack is not empty, pop the integer at the top of the stack.
- If, at any moment, the elements in the stack (from the bottom to the top) are equal to `target`, do not read new integers from the stream and do not do more operations on the stack.

Return *the stack operations needed to build* `target` following the mentioned rules. If there are multiple valid answers, return **any of them**.

### Example 1:

**Input:** `target = [1,3]`, `n = 3`

**Output:** `["Push", "Push", "Pop", "Push"]`

**Explanation:** Initially the stack `s` is empty. The last element is the top of the stack.

Read 1 from the stream and push it to the stack. `s = [1]`.

Read 2 from the stream and push it to the stack. `s = [1,2]`.

Pop the integer on the top of the stack. `s = [1]`.

Read 3 from the stream and push it to the stack. `s = [1,3]`.

### Example 2:

**Input:** `target = [1,2,3]`, `n = 3`

**Output:** `["Push", "Push", "Push"]`

**Explanation:** Initially the stack `s` is empty. The last element is the top of the stack.

Read 1 from the stream and push it to the stack. `s = [1]`.

Read 2 from the stream and push it to the stack. `s = [1,2]`.

Read 3 from the stream and push it to the stack. `s = [1,2,3]`.

### Example 3:

**Input:** target = [1,2], n = 4

**Output:** ["Push", "Push"]

**Explanation:** Initially the stack s is empty. The last element is the top of the stack.

Read 1 from the stream and push it to the stack. s = [1].

Read 2 from the stream and push it to the stack. s = [1,2].

Since the stack (from the bottom to the top) is equal to target, we stop the stack operations.

The answers that read integer 3 from the stream are not accepted.

### Constraints:

- `1 <= target.length <= 100`
- `1 <= n <= 100`
- `1 <= target[i] <= n`
- `target` is strictly increasing.

## 1442. Count Triplets That Can Form Two Arrays of Equal XOR

Medium

104848Add to ListShare

Given an array of integers `arr`.

We want to select three indices `i`, `j` and `k` where `(0 <= i < j <= k < arr.length)`.

Let's define `a` and `b` as follows:

- `a = arr[i] ^ arr[i + 1] ^ ... ^ arr[j - 1]`
- `b = arr[j] ^ arr[j + 1] ^ ... ^ arr[k]`

Note that `^` denotes the **bitwise-xor** operation.

Return *the number of triplets* `(i, j and k)` Where `a == b`.

### Example 1:

**Input:** arr = [2,3,1,6,7]

**Output:** 4

**Explanation:** The triplets are (0,1,2), (0,2,2), (2,3,4) and (2,4,4)

**Example 2:**

**Input:** arr = [1,1,1,1,1]

**Output:** 10

**Constraints:**

- $1 \leq \text{arr.length} \leq 300$
- $1 \leq \text{arr}[i] \leq 10^8$

## 1443. Minimum Time to Collect All Apples in a Tree

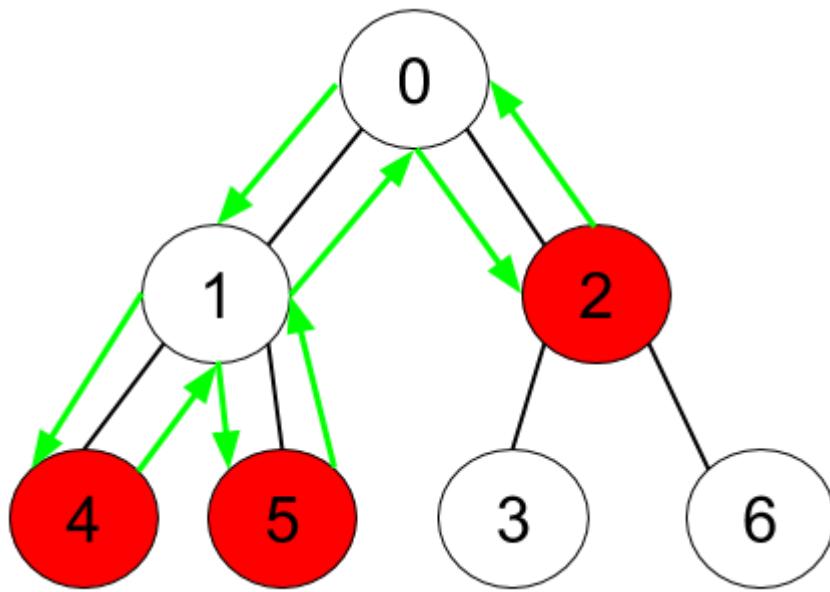
Medium

111692Add to ListShare

Given an undirected tree consisting of  $n$  vertices numbered from 0 to  $n-1$ , which has some apples in their vertices. You spend 1 second to walk over one edge of the tree. *Return the minimum time in seconds you have to spend to collect all apples in the tree, starting at vertex 0 and coming back to this vertex.*

The edges of the undirected tree are given in the array `edges`, where `edges[i] = [ai, bi]` means that exists an edge connecting the vertices `ai` and `bi`. Additionally, there is a boolean array `hasApple`, where `hasApple[i] = true` means that vertex `i` has an apple; otherwise, it does not have any apple.

**Example 1:**

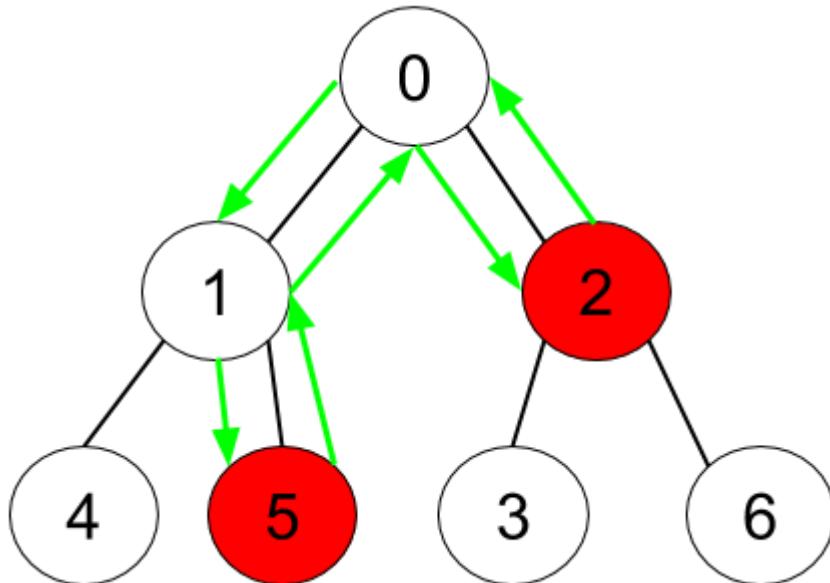


**Input:**  $n = 7$ ,  $\text{edges} = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]]$ ,  $\text{hasApple} = [\text{false},\text{false},\text{true},\text{false},\text{true},\text{true},\text{false}]$

**Output:** 8

**Explanation:** The figure above represents the given tree where red vertices have an apple. One optimal path to collect all apples is shown by the green arrows.

**Example 2:**



**Input:**  $n = 7$ ,  $\text{edges} = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]]$ ,  $\text{hasApple} = [\text{false},\text{false},\text{true},\text{false},\text{true},\text{true},\text{false}]$

**Output:** 6

**Explanation:** The figure above represents the given tree where red vertices have an apple. One optimal path to collect all apples is shown by the green arrows.

**Example 3:**

**Input:** `n = 7, edges = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], hasApple = [false, false, false, false, false, false, false]`

**Output:** `0`

**Constraints:**

- `1 <= n <= 105`
- `edges.length == n - 1`
- `edges[i].length == 2`
- `0 <= ai < bi <= n - 1`
- `fromi < toi`
- `hasApple.length == n`

## 1444. Number of Ways of Cutting a Pizza

Hard

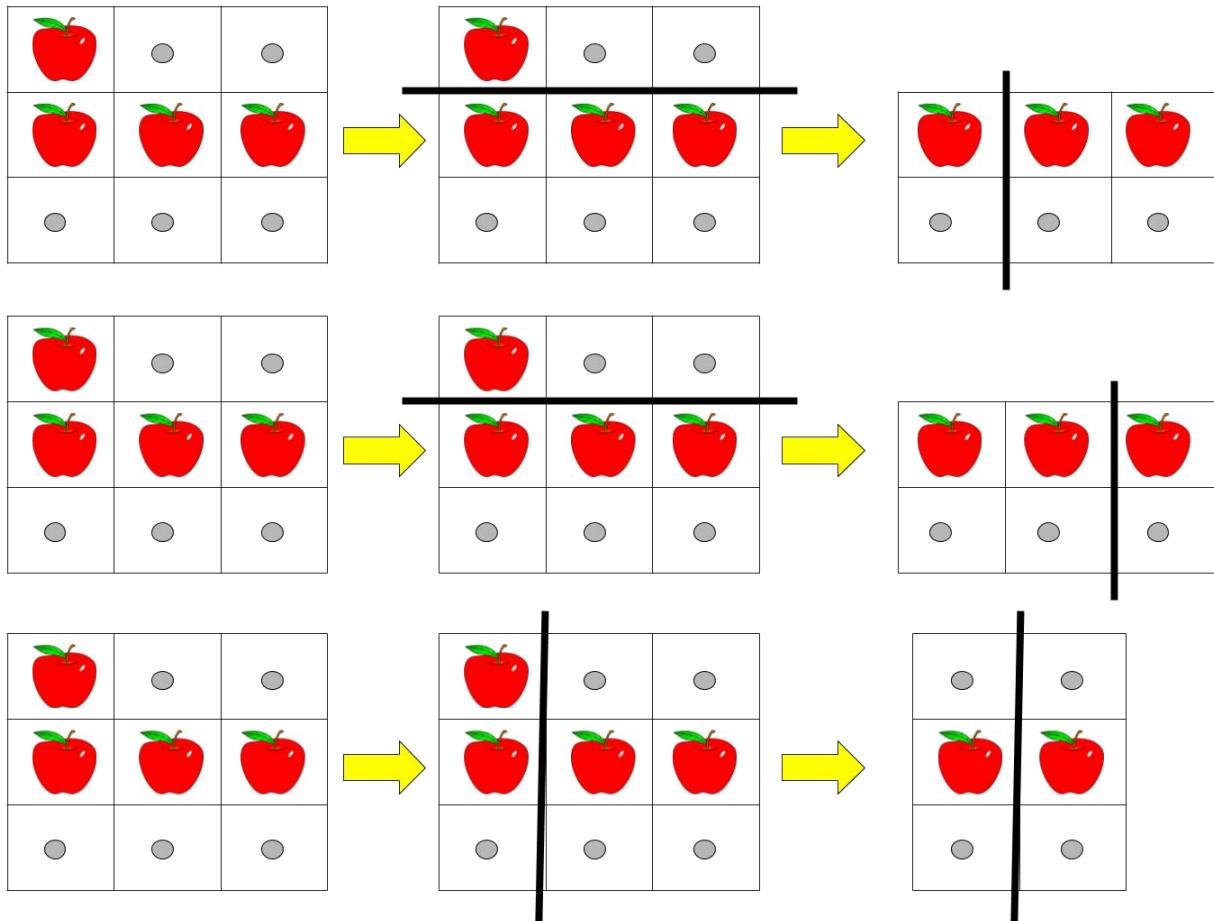
52120Add to ListShare

Given a rectangular pizza represented as a `rows x cols` matrix containing the following characters: '`A`' (an apple) and '`.`' (empty cell) and given the integer `k`. You have to cut the pizza into `k` pieces using `k-1` cuts.

For each cut you choose the direction: vertical or horizontal, then you choose a cut position at the cell boundary and cut the pizza into two pieces. If you cut the pizza vertically, give the left part of the pizza to a person. If you cut the pizza horizontally, give the upper part of the pizza to a person. Give the last piece of pizza to the last person.

*Return the number of ways of cutting the pizza such that each piece contains **at least** one apple. Since the answer can be a huge number, return this modulo  $10^9 + 7$ .*

**Example 1:**



**Input:** pizza = ["A..", "AAA", "..."], k = 3

**Output:** 3

**Explanation:** The figure above shows the three ways to cut the pizza. Note that pieces must contain at least one apple.

**Example 2:**

**Input:** pizza = ["A..", "AA.", "..."], k = 3

**Output:** 1

**Example 3:**

**Input:** pizza = ["A..", "A..", "..."], k = 1

**Output:** 1

**Constraints:**

- `1 <= rows, cols <= 50`
- `rows == pizza.length`
- `cols == pizza[i].length`
- `1 <= k <= 10`
- `pizza` consists of characters '`A`' and '`.`' only.

## 1446. Consecutive Characters

Easy

135627Add to ListShare

The **power** of the string is the maximum length of a non-empty substring that contains only one unique character.

Given a string `s`, return *the power of s*.

**Example 1:**

**Input:** `s = "leetcode"`

**Output:** 2

**Explanation:** The substring "ee" is of length 2 with the character 'e' only.

**Example 2:**

**Input:** `s = "abbccddddeeeeedcba"`

**Output:** 5

**Explanation:** The substring "eeeeee" is of length 5 with the character 'e' only.

**Constraints:**

- `1 <= s.length <= 500`
- `s` consists of only lowercase English letters.

## 1447. Simplified Fractions

Medium

28936Add to ListShare

Given an integer `n`, return a list of all **simplified fractions** between `0` and `1` (exclusive) such that the denominator is less-than-or-equal-to `n`. You can return the answer in **any order**.

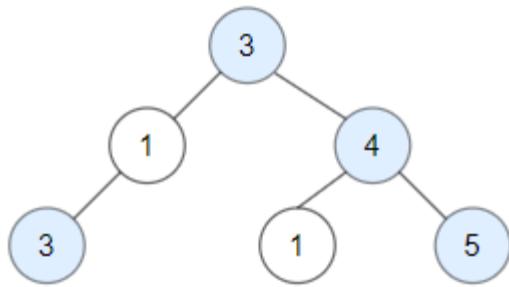
**Example 1:****Input:** n = 2**Output:** ["1/2"]**Explanation:** "1/2" is the only unique fraction with a denominator less-than-or-equal-to 2.**Example 2:****Input:** n = 3**Output:** ["1/2", "1/3", "2/3"]**Example 3:****Input:** n = 4**Output:** ["1/2", "1/3", "1/4", "2/3", "3/4"]**Explanation:** "2/4" is not a simplified fraction because it can be simplified to "1/2".**Constraints:**

- $1 \leq n \leq 100$

**1448. Count Good Nodes in Binary Tree****Medium**

4201118Add to ListShare

Given a binary tree `root`, a node  $X$  in the tree is named **good** if in the path from root to  $X$  there are no nodes with a value *greater than*  $X$ .Return the number of **good** nodes in the binary tree.**Example 1:**



**Input:** root = [3,1,4,3,null,1,5]

**Output:** 4

**Explanation:** Nodes in blue are **good**.

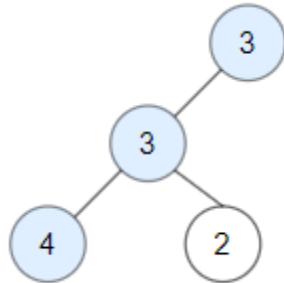
Root Node (3) is always a good node.

Node 4 -> (3,4) is the maximum value in the path starting from the root.

Node 5 -> (3,4,5) is the maximum value in the path

Node 3 -> (3,1,3) is the maximum value in the path.

### Example 2:



**Input:** root = [3,3,null,4,2]

**Output:** 3

**Explanation:** Node 2 -> (3, 3, 2) is not good, because "3" is higher than it.

### Example 3:

**Input:** root = [1]

**Output:** 1

**Explanation:** Root is considered as **good**.

**Constraints:**

- The number of nodes in the binary tree is in the range  $[1, 10^5]$ .
- Each node's value is between  $[-10^4, 10^4]$ .

**1449. Form Largest Integer With Digits That Add up to Target****Hard**

56111Add to ListShare

Given an array of integers `cost` and an integer `target`, return *the maximum integer you can paint under the following rules:*

- The cost of painting a digit  $(i + 1)$  is given by `cost[i]` (**0-indexed**).
- The total cost used must be equal to `target`.
- The integer does not have `0` digits.

Since the answer may be very large, return it as a string. If there is no way to paint any integer given the condition, return `"0"`.

**Example 1:**

**Input:** `cost = [4,3,2,5,6,7,2,5,5]`, `target = 9`

**Output:** `"7772"`

**Explanation:** The cost to paint the digit '7' is 2, and the digit '2' is 3. Then  $\text{cost}("7772") = 2*3 + 3*1 = 9$ . You could also paint "977", but "7772" is the largest number.

**Digit      cost**

1	->	4
2	->	3
3	->	2
4	->	5
5	->	6
6	->	7
7	->	2
8	->	5
9	->	5

**Example 2:**

**Input:** cost = [7,6,5,5,5,6,8,7,8], target = 12

**Output:** "85"

**Explanation:** The cost to paint the digit '8' is 7, and the digit '5' is 5. Then cost("85") = 7 + 5 = 12.

**Example 3:**

**Input:** cost = [2,4,6,2,4,6,4,4,4], target = 5

**Output:** "0"

**Explanation:** It is impossible to paint any integer with total cost equal to target.

**Constraints:**

- cost.length == 9
- 1 <= cost[i], target <= 5000

## 1450. Number of Students Doing Homework at a Given Time

Easy

659138Add to ListShare

Given two integer arrays startTime and endTime and given an integer queryTime.

The *i*th student started doing their homework at the time startTime[i] and finished it at time endTime[i].

Return *the number of students* doing their homework at time queryTime. More formally, return the number of students where queryTime lays in the interval [startTime[i], endTime[i]] inclusive.

**Example 1:**

**Input:** startTime = [1,2,3], endTime = [3,2,7], queryTime = 4

**Output:** 1

**Explanation:** We have 3 students where:

The first student started doing homework at time 1 and finished at time 3 and wasn't doing anything at time 4.

The second student started doing homework at time 2 and finished at time 2 and also wasn't doing anything at time 4.

The third student started doing homework at time 3 and finished at time 7 and was the only student doing homework at time 4.

**Example 2:**

**Input:** startTime = [4], endTime = [4], queryTime = 4

**Output:** 1

**Explanation:** The only student was doing their homework at the queryTime.

**Constraints:**

- startTime.length == endTime.length
- 1 <= startTime.length <= 100
- 1 <= startTime[i] <= endTime[i] <= 1000
- 1 <= queryTime <= 1000

## 1451. Rearrange Words in a Sentence

Medium

55366Add to ListShare

Given a sentence `text` (A *sentence* is a string of space-separated words) in the following format:

- First letter is in upper case.
- Each word in `text` are separated by a single space.

Your task is to rearrange the words in `text` such that all words are rearranged in an increasing order of their lengths. If two words have the same length, arrange them in their original order.

Return the new text following the format shown above.

**Example 1:**

**Input:** text = "Leetcode is cool"

**Output:** "Is cool leetcode"

**Explanation:** There are 3 words, "Leetcode" of length 8, "is" of length 2 and "cool" of length 4.

Output is ordered by length and the new first word starts with capital letter.

**Example 2:**

**Input:** text = "Keep calm and code on"

**Output:** "On and keep calm code"

**Explanation:** Output is ordered as follows:

"On" 2 letters.

"and" 3 letters.

"keep" 4 letters in case of tie order by position in original text.

"calm" 4 letters.

"code" 4 letters.

**Example 3:**

**Input:** text = "To be or not to be"

**Output:** "To be or to be not"

**Constraints:**

- `text` begins with a capital letter and then contains lowercase letters and single space between words.
- $1 \leq \text{text.length} \leq 10^5$

## 1452. People Whose List of Favorite Companies Is Not a Subset of Another List

Medium

279203Add to ListShare

Given the array `favoriteCompanies` where `favoriteCompanies[i]` is the list of favorites companies for the `i`th person (**indexed from 0**).

*Return the indices of people whose list of favorite companies is not a **subset** of any other list of favorites companies. You must return the indices in increasing order.*

**Example 1:**

**Input:** `favoriteCompanies = [[["leetcode", "google", "facebook"], ["google", "microsoft"], ["google", "facebook"], ["google"], ["amazon"]]]`

**Output:** `[0,1,4]`

**Explanation:**

Person with index=2 has `favoriteCompanies[2]=["google", "facebook"]` which is a subset of `favoriteCompanies[0]=["leetcode", "google", "facebook"]` corresponding to the person with index 0.

Person with index=3 has `favoriteCompanies[3]=["google"]` which is a subset of `favoriteCompanies[0]=["leetcode", "google", "facebook"]` and `favoriteCompanies[1]=["google", "microsoft"]`.

Other lists of favorite companies are not a subset of another list, therefore, the answer is `[0,1,4]`.

**Example 2:**

**Input:** `favoriteCompanies = [[["leetcode", "google", "facebook"], ["leetcode", "amazon"], ["facebook", "google"]]]`

**Output:** `[0,1]`

**Explanation:** In this case `favoriteCompanies[2]=["facebook", "google"]` is a subset of `favoriteCompanies[0]=["leetcode", "google", "facebook"]`, therefore, the answer is `[0,1]`.

**Example 3:**

**Input:** `favoriteCompanies = [[["leetcode"], ["google"], ["facebook"], ["amazon"]]]`

**Output:** `[0,1,2,3]`

**Constraints:**

- `1 <= favoriteCompanies.length <= 100`
- `1 <= favoriteCompanies[i].length <= 500`
- `1 <= favoriteCompanies[i][j].length <= 20`
- All strings in `favoriteCompanies[i]` are **distinct**.
- All lists of favorite companies are **distinct**, that is, If we sort alphabetically each list then `favoriteCompanies[i] != favoriteCompanies[j]`.
- All strings consist of lowercase English letters only.

## 1453. Maximum Number of Darts Inside of a Circular Dartboard

Hard

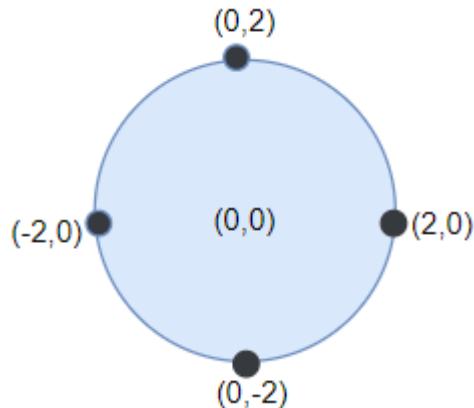
120252 Add to List Share

Alice is throwing `n` darts on a very large wall. You are given an array `darts` where `darts[i] = [xi, yi]` is the position of the `ith` dart that Alice threw on the wall.

Bob knows the positions of the `n` darts on the wall. He wants to place a dartboard of radius `r` on the wall so that the maximum number of darts that Alice throws lies on the dartboard.

Given the integer  $r$ , return the maximum number of darts that can lie on the dartboard.

**Example 1:**

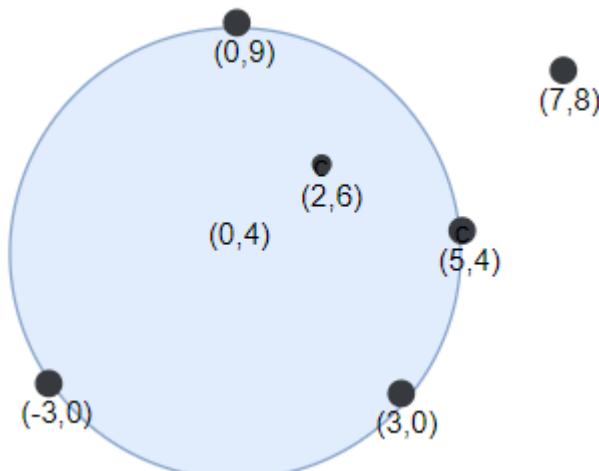


**Input:** darts = `[-2,0],[2,0],[0,2],[0,-2]`,  $r = 2$

**Output:** 4

**Explanation:** Circle dartboard with center in  $(0,0)$  and radius = 2 contain all points.

**Example 2:**



**Input:** darts = `[-3,0],[3,0],[2,6],[5,4],[0,9],[7,8]`,  $r = 5$

**Output:** 5

**Explanation:** Circle dartboard with center in  $(0,4)$  and radius = 5 contain all points except the point  $(7,8)$ .

**Constraints:**

- $1 \leq \text{darts.length} \leq 100$
- $\text{darts}[i].length == 2$
- $-10^4 \leq x_i, y_i \leq 10^4$
- $1 \leq r \leq 5000$

## 1455. Check If a Word Occurs As a Prefix of Any Word in a Sentence

### Easy

60229 Add to List Share

Given a `sentence` that consists of some words separated by a **single space**, and a `searchWord`, check if `searchWord` is a prefix of any word in `sentence`.

Return *the index of the word in sentence (1-indexed) where `searchWord` is a prefix of this word*. If `searchWord` is a prefix of more than one word, return the index of the first word (**minimum index**). If there is no such word return `-1`.

A **prefix** of a string `s` is any leading contiguous substring of `s`.

### Example 1:

**Input:** `sentence = "i love eating burger", searchWord = "burg"`

**Output:** `4`

**Explanation:** "burg" is prefix of "burger" which is the 4th word in the sentence.

### Example 2:

**Input:** `sentence = "this problem is an easy problem", searchWord = "pro"`

**Output:** `2`

**Explanation:** "pro" is prefix of "problem" which is the 2nd and the 6th word in the sentence, but we return 2 as it's the minimal index.

### Example 3:

**Input:** `sentence = "i am tired", searchWord = "you"`

**Output:** `-1`

**Explanation:** "you" is not a prefix of any word in the sentence.

### Constraints:

- $1 \leq \text{sentence.length} \leq 100$

- `1 <= searchWord.length <= 10`
- `sentence` consists of lowercase English letters and spaces.
- `searchWord` consists of lowercase English letters.

## 1456. Maximum Number of Vowels in a Substring of Given Length

Medium

87243Add to ListShare

Given a string `s` and an integer `k`, return *the maximum number of vowel letters in any substring of s with length k*.

**Vowel letters** in English are '`a`', '`e`', '`i`', '`o`', and '`u`'.

### Example 1:

**Input:** `s = "abciiidef", k = 3`

**Output:** 3

**Explanation:** The substring "iii" contains 3 vowel letters.

### Example 2:

**Input:** `s = "aeiou", k = 2`

**Output:** 2

**Explanation:** Any substring of length 2 contains 2 vowels.

### Example 3:

**Input:** `s = "leetcode", k = 3`

**Output:** 2

**Explanation:** "lee", "eet" and "ode" contain 2 vowels.

### Constraints:

- `1 <= s.length <= 105`
- `s` consists of lowercase English letters.
- `1 <= k <= s.length`

## 1457. Pseudo-Palindromic Paths in a Binary Tree

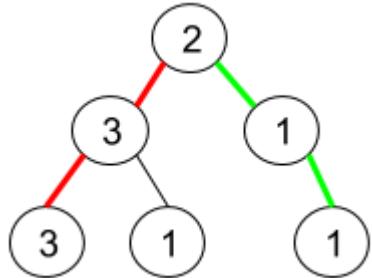
Medium

225077Add to ListShare

Given a binary tree where node values are digits from 1 to 9. A path in the binary tree is said to be **pseudo-palindromic** if at least one permutation of the node values in the path is a palindrome.

*Return the number of **pseudo-palindromic** paths going from the root node to leaf nodes.*

**Example 1:**

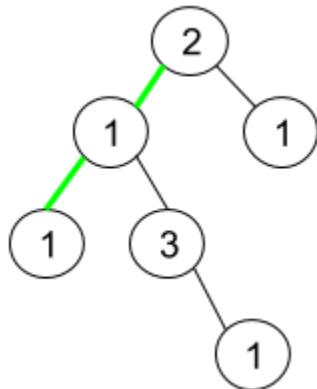


**Input:** root = [2,3,1,3,1,null,1]

**Output:** 2

**Explanation:** The figure above represents the given binary tree. There are three paths going from the root node to leaf nodes: the red path [2,3,3], the green path [2,1,1], and the path [2,3,1]. Among these paths only red path and green path are pseudo-palindromic paths since the red path [2,3,3] can be rearranged in [3,2,3] (palindrome) and the green path [2,1,1] can be rearranged in [1,2,1] (palindrome).

**Example 2:**



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

**Output:** 1

**Explanation:** The figure above represents the given binary tree. There are three paths going from the root node to leaf nodes: the green path [2,1,1], the path [2,1,3,1],

and the path `[2,1]`. Among these paths only the green path is pseudo-palindromic since `[2,1,1]` can be rearranged in `[1,2,1]` (palindrome).

**Example 3:**

**Input:** `root = [9]`

**Output:** `1`

**Constraints:**

- The number of nodes in the tree is in the range `[1, 105]`.
- `1 <= Node.val <= 9`

## 1458. Max Dot Product of Two Subsequences

Hard

66014Add to ListShare

Given two arrays `nums1` and `nums2`.

Return the maximum dot product between **non-empty** subsequences of `nums1` and `nums2` with the same length.

A subsequence of a array is a new array which is formed from the original array by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (ie, `[2,3,5]` is a subsequence of `[1,2,3,4,5]` while `[1,5,3]` is not).

**Example 1:**

**Input:** `nums1 = [2,1,-2,5], nums2 = [3,0,-6]`

**Output:** `18`

**Explanation:** Take subsequence `[2,-2]` from `nums1` and subsequence `[3,-6]` from `nums2`.

Their dot product is  $(2*3 + (-2)*(-6)) = 18$ .

**Example 2:**

**Input:** `nums1 = [3,-2], nums2 = [2,-6,7]`

**Output:** `21`

**Explanation:** Take subsequence `[3]` from `nums1` and subsequence `[7]` from `nums2`.

Their dot product is  $(3*7) = 21$ .

**Example 3:**

**Input:** `nums1 = [-1,-1], nums2 = [1,1]`

**Output:** `-1`

**Explanation:** Take subsequence `[-1]` from `nums1` and subsequence `[1]` from `nums2`.

Their dot product is `-1`.

**Constraints:**

- `1 <= nums1.length, nums2.length <= 500`
- `-1000 <= nums1[i], nums2[i] <= 1000`

**1460. Make Two Arrays Equal by Reversing Subarrays**

**Easy**

826120Add to ListShare

You are given two integer arrays of equal length `target` and `arr`. In one step, you can select any **non-empty subarray** of `arr` and reverse it. You are allowed to make any number of steps.

Return `true` if you can make `arr` equal to `target` or `false` otherwise.

**Example 1:**

**Input:** `target = [1,2,3,4], arr = [2,4,1,3]`

**Output:** `true`

**Explanation:** You can follow the next steps to convert `arr` to `target`:

1- Reverse subarray `[2,4,1]`, `arr` becomes `[1,4,2,3]`

2- Reverse subarray `[4,2]`, `arr` becomes `[1,2,4,3]`

3- Reverse subarray `[4,3]`, `arr` becomes `[1,2,3,4]`

There are multiple ways to convert `arr` to `target`, this is not the only way to do so.

**Example 2:**

**Input:** `target = [7], arr = [7]`

**Output:** `true`

**Explanation:** `arr` is equal to `target` without any reverses.

**Example 3:**

**Input:** target = [3,7,9], arr = [3,7,11]

**Output:** false

**Explanation:** arr does not have value 9 and it can never be converted to target.

**Constraints:**

- target.length == arr.length
- 1 <= target.length <= 1000
- 1 <= target[i] <= 1000
- 1 <= arr[i] <= 1000

**1461. Check If a String Contains All Binary Codes of Size K**

Medium

192990Add to ListShare

Given a binary string `s` and an integer `k`, return `true` if every binary code of length `k` is a substring of `s`. Otherwise, return `false`.

**Example 1:**

**Input:** s = "00110110", k = 2

**Output:** true

**Explanation:** The binary codes of length 2 are "00", "01", "10" and "11". They can be all found as substrings at indices 0, 1, 3 and 2 respectively.

**Example 2:**

**Input:** s = "0110", k = 1

**Output:** true

**Explanation:** The binary codes of length 1 are "0" and "1", it is clear that both exist as a substring.

**Example 3:**

**Input:** s = "0110", k = 2

**Output:** false

**Explanation:** The binary code "00" is of length 2 and does not exist in the array.

**Constraints:**

- $1 \leq s.length \leq 5 * 10^5$
- $s[i]$  is either '0' or '1'.
- $1 \leq k \leq 20$

**1462. Course Schedule IV****Medium**

100452Add to ListShare

There are a total of  $numCourses$  courses you have to take, labeled from  $0$  to  $numCourses - 1$ . You are given an array  $prerequisites$  where  $prerequisites[i] = [a_i, b_i]$  indicates that you **must** take course  $a_i$  first if you want to take course  $b_i$ .

- For example, the pair  $[0, 1]$  indicates that you have to take course  $0$  before you can take course  $1$ .

Prerequisites can also be **indirect**. If course  $a$  is a prerequisite of course  $b$ , and course  $b$  is a prerequisite of course  $c$ , then course  $a$  is a prerequisite of course  $c$ .

You are also given an array  $queries$  where  $queries[j] = [u_j, v_j]$ . For the  $j^{\text{th}}$  query, you should answer whether course  $u_j$  is a prerequisite of course  $v_j$  or not.

Return a boolean array  $answer$ , where  $answer[j]$  is the answer to the  $j^{\text{th}}$  query.

**Example 1:**

**Input:**  $numCourses = 2$ ,  $prerequisites = [[1,0]]$ ,  $queries = [[0,1],[1,0]]$

**Output:** [false,true]

**Explanation:** The pair  $[1, 0]$  indicates that you have to take course 1 before you can take course 0.

Course 0 is not a prerequisite of course 1, but the opposite is true.

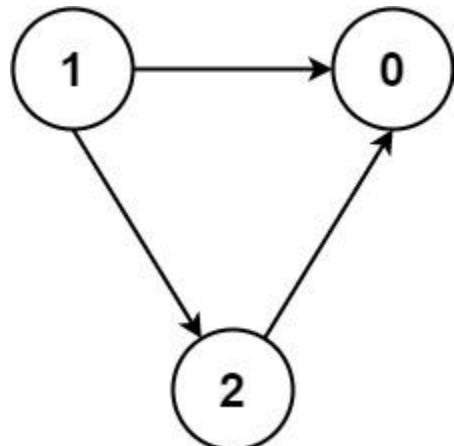
**Example 2:**

**Input:**  $numCourses = 2$ ,  $prerequisites = []$ ,  $queries = [[1,0],[0,1]]$

**Output:** [false,false]

**Explanation:** There are no prerequisites, and each course is independent.

**Example 3:**



**Input:** numCourses = 3, prerequisites = [[1,2],[1,0],[2,0]], queries = [[1,0],[1,2]]

**Output:** [true,true]

**Constraints:**

- $2 \leq \text{numCourses} \leq 100$
- $0 \leq \text{prerequisites.length} \leq (\text{numCourses} * (\text{numCourses} - 1)) / 2$
- $\text{prerequisites}[i].length == 2$
- $0 \leq a_i, b_i \leq n - 1$
- $a_i \neq b_i$
- All the pairs  $[a_i, b_i]$  are **unique**.
- The prerequisites graph has no cycles.
- $1 \leq \text{queries.length} \leq 10^4$
- $0 \leq u_i, v_i \leq n - 1$
- $u_i \neq v_i$

## 1463. Cherry Pickup II

Hard

249625 Add to List Share

You are given a  $\text{rows} \times \text{cols}$  matrix `grid` representing a field of cherries where `grid[i][j]` represents the number of cherries that you can collect from the  $(i, j)$  cell.

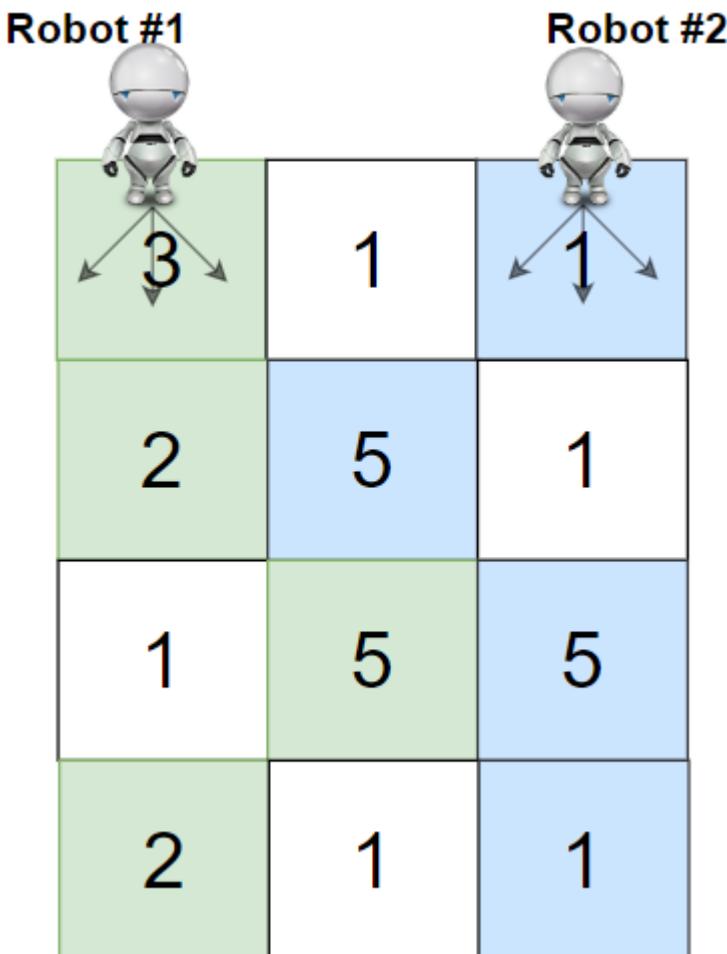
You have two robots that can collect cherries for you:

- **Robot #1** is located at the **top-left corner**  $(0, 0)$ , and
- **Robot #2** is located at the **top-right corner**  $(0, \text{cols} - 1)$ .

Return the maximum number of cherries collection using both robots by following the rules below:

- From a cell  $(i, j)$ , robots can move to cell  $(i + 1, j - 1)$ ,  $(i + 1, j)$ , or  $(i + 1, j + 1)$ .
- When any robot passes through a cell, it picks up all cherries, and the cell becomes an empty cell.
- When both robots stay in the same cell, only one takes the cherries.
- Both robots cannot move outside of the grid at any moment.
- Both robots should reach the bottom row in `grid`.

**Example 1:**



**Input:** `grid = [[3,1,1],[2,5,1],[1,5,5],[2,1,1]]`

**Output:** 24

**Explanation:** Path of robot #1 and #2 are described in color green and blue respectively.

Cherries taken by Robot #1,  $(3 + 2 + 5 + 2) = 12$ .

Cherries taken by Robot #2,  $(1 + 5 + 5 + 1) = 12$ .

Total of cherries:  $12 + 12 = 24$ .

**Example 2:**

**Robot #1**



1	0	0	0	0	0	1
2	0	0	0	0	3	0
2	0	9	0	0	0	0
0	3	0	5	4	0	0
1	0	2	3	0	0	6

**Robot #2**



**Input:** grid =  
`[[1,0,0,0,0,0,1],[2,0,0,0,0,3,0],[2,0,9,0,0,0,0],[0,3,0,5,4,0,0],[1,0,2,3,0,0,6]]`

**Output:** 28

**Explanation:** Path of robot #1 and #2 are described in color green and blue respectively.

Cherries taken by Robot #1,  $(1 + 9 + 5 + 2) = 17$ .

Cherries taken by Robot #2,  $(1 + 3 + 4 + 3) = 11$ .

Total of cherries:  $17 + 11 = 28$ .

**Constraints:**

- `rows == grid.length`
- `cols == grid[i].length`
- `2 <= rows, cols <= 70`
- `0 <= grid[i][j] <= 100`

**1464. Maximum Product of Two Elements in an Array****Easy**

1216169Add to ListShare

Given the array of integers `nums`, you will choose two different indices `i` and `j` of that array. *Return the maximum value of  $(nums[i]-1) * (nums[j]-1)$ .*

**Example 1:****Input:** `nums = [3,4,5,2]`**Output:** 12

**Explanation:** If you choose the indices `i=1` and `j=2` (indexed from 0), you will get the maximum value, that is,  $(nums[1]-1)*(nums[2]-1) = (4-1)*(5-1) = 3*4 = 12$ .

**Example 2:****Input:** `nums = [1,5,4,5]`**Output:** 16

**Explanation:** Choosing the indices `i=1` and `j=3` (indexed from 0), you will get the maximum value of  $(5-1)*(5-1) = 16$ .

**Example 3:****Input:** `nums = [3,7]`**Output:** 12**Constraints:**

- `2 <= nums.length <= 500`
- `1 <= nums[i] <= 10^3`

**1465. Maximum Area of a Piece of Cake After Horizontal and Vertical Cuts****Medium**

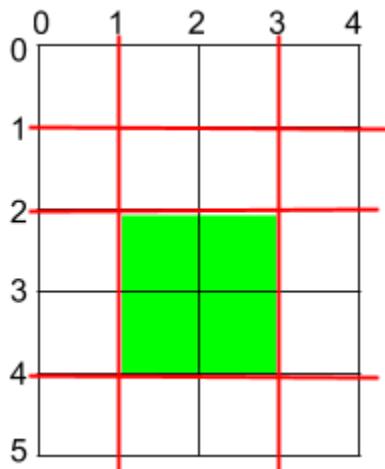
2431330Add to ListShare

You are given a rectangular cake of size  $h \times w$  and two arrays of integers `horizontalCuts` and `verticalCuts` where:

- `horizontalCuts[i]` is the distance from the top of the rectangular cake to the  $i^{\text{th}}$  horizontal cut and similarly, and
- `verticalCuts[j]` is the distance from the left of the rectangular cake to the  $j^{\text{th}}$  vertical cut.

Return *the maximum area of a piece of cake after you cut at each horizontal and vertical position provided in the arrays `horizontalCuts` and `verticalCuts`*. Since the answer can be a large number, return this **modulo**  $10^9 + 7$ .

**Example 1:**

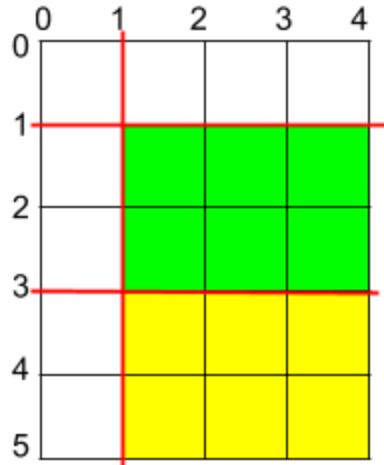


**Input:**  $h = 5$ ,  $w = 4$ , `horizontalCuts` = [1,2,4], `verticalCuts` = [1,3]

**Output:** 4

**Explanation:** The figure above represents the given rectangular cake. Red lines are the horizontal and vertical cuts. After you cut the cake, the green piece of cake has the maximum area.

**Example 2:**



**Input:**  $h = 5$ ,  $w = 4$ , `horizontalCuts = [3,1]`, `verticalCuts = [1]`

**Output:** 6

**Explanation:** The figure above represents the given rectangular cake. Red lines are the horizontal and vertical cuts. After you cut the cake, the green and yellow pieces of cake have the maximum area.

**Example 3:**

**Input:**  $h = 5$ ,  $w = 4$ , `horizontalCuts = [3]`, `verticalCuts = [3]`

**Output:** 9

**Constraints:**

- $2 \leq h, w \leq 10^9$
- $1 \leq \text{horizontalCuts.length} \leq \min(h - 1, 10^5)$
- $1 \leq \text{verticalCuts.length} \leq \min(w - 1, 10^5)$
- $1 \leq \text{horizontalCuts}[i] < h$
- $1 \leq \text{verticalCuts}[i] < w$
- All the elements in `horizontalCuts` are distinct.
- All the elements in `verticalCuts` are distinct.

## 1466. Reorder Routes to Make All Paths Lead to the City Zero

Medium

184148Add to ListShare

There are  $n$  cities numbered from 0 to  $n - 1$  and  $n - 1$  roads such that there is only one way to travel between two different cities (this network form a tree). Last year, The ministry of transport decided to orient the roads in one direction because they are too narrow.

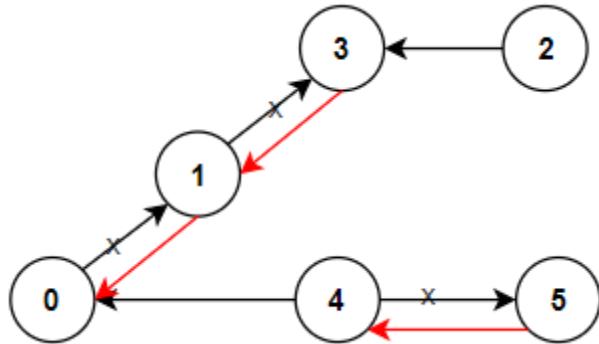
Roads are represented by `connections` where `connections[i] = [ai, bi]` represents a road from city  $a_i$  to city  $b_i$ .

This year, there will be a big event in the capital (city 0), and many people want to travel to this city.

Your task consists of reorienting some roads such that each city can visit the city 0. Return the **minimum** number of edges changed.

It's **guaranteed** that each city can reach city 0 after reorder.

**Example 1:**

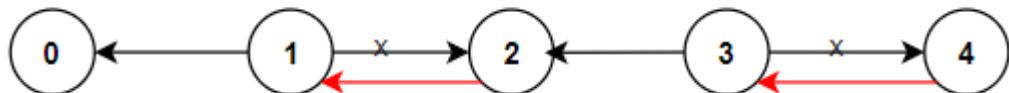


**Input:**  $n = 6$ ,  $\text{connections} = [[0,1],[1,3],[2,3],[4,0],[4,5]]$

**Output:** 3

**Explanation:** Change the direction of edges show in red such that each node can reach the node 0 (capital).

**Example 2:**



**Input:**  $n = 5$ ,  $\text{connections} = [[1,0],[1,2],[3,2],[3,4]]$

**Output:** 2

**Explanation:** Change the direction of edges show in red such that each node can reach the node 0 (capital).

**Example 3:**

**Input:**  $n = 3$ ,  $\text{connections} = [[1,0],[2,0]]$

**Output:** 0

**Constraints:**

- $2 \leq n \leq 5 * 10^4$
- $\text{connections.length} == n - 1$
- $\text{connections}[i].length == 2$
- $0 \leq a_i, b_i \leq n - 1$
- $a_i \neq b_i$

**1467. Probability of a Two Boxes Having The Same Number of Distinct Balls****Hard**

216154Add to ListShare

Given  $2n$  balls of  $k$  distinct colors. You will be given an integer array `balls` of size  $k$  where `balls[i]` is the number of balls of color  $i$ .

All the balls will be **shuffled uniformly at random**, then we will distribute the first  $n$  balls to the first box and the remaining  $n$  balls to the other box (Please read the explanation of the second example carefully).

Please note that the two boxes are considered different. For example, if we have two balls of colors `a` and `b`, and two boxes `[]` and `()`, then the distribution `[a] (b)` is considered different than the distribution `[b] (a)` (Please read the explanation of the first example carefully).

Return *the probability* that the two boxes have the same number of distinct balls. Answers within  $10^{-5}$  of the actual value will be accepted as correct.

**Example 1:****Input:** `balls = [1,1]`**Output:** `1.00000`**Explanation:** Only 2 ways to divide the balls equally:

- A ball of color 1 to box 1 and a ball of color 2 to box 2
- A ball of color 2 to box 1 and a ball of color 1 to box 2

In both ways, the number of distinct colors in each box is equal. The probability is  $2/2 = 1$

**Example 2:****Input:** `balls = [2,1,1]`**Output:** `0.66667`**Explanation:** We have the set of balls `[1, 1, 2, 3]`

This set of balls will be shuffled randomly and we may have one of the 12 distinct shuffles with equal probability (i.e. 1/12):

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

After that, we add the first two balls to the first box and the second two balls to the second box.

We can see that 8 of these 12 possible random distributions have the same number of distinct colors of balls in each box.

Probability is 8/12 = 0.66667

### Example 3:

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

**Output:** 0.60000

**Explanation:** The set of balls is [1, 2, 2, 3, 4, 4]. It is hard to display all the 180 possible random shuffles of this set but it is easy to check that 108 of them will have the same number of distinct colors in each box.

Probability = 108 / 180 = 0.6

### Constraints:

- $1 \leq \text{balls.length} \leq 8$
- $1 \leq \text{balls}[i] \leq 6$
- $\text{sum}(\text{balls})$  is even.

## 1470. Shuffle the Array

Easy

2915195Add to ListShare

Given the array `nums` consisting of  $2n$  elements in the form  $[x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n]$ .

Return the array in the form  $[x_1, y_1, x_2, y_2, \dots, x_n, y_n]$ .

### Example 1:

**Input:** nums = [2,5,1,3,4,7], n = 3

**Output:** [2,3,5,4,1,7]

**Explanation:** Since  $x_1=2$ ,  $x_2=5$ ,  $x_3=1$ ,  $y_1=3$ ,  $y_2=4$ ,  $y_3=7$  then the answer is [2,3,5,4,1,7].

**Example 2:**

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

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

**Example 3:**

**Input:** `nums = [1,1,2,2], n = 2`

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

**Constraints:**

- $1 \leq n \leq 500$
- $\text{nums.length} == 2n$
- $1 \leq \text{nums}[i] \leq 10^3$

## 1471. The k Strongest Values in an Array

Medium

481104Add to ListShare

Given an array of integers `arr` and an integer `k`.

A value `arr[i]` is said to be stronger than a value `arr[j]` if  $|arr[i] - m| > |arr[j] - m|$  where `m` is the **median** of the array.

If  $|arr[i] - m| == |arr[j] - m|$ , then `arr[i]` is said to be stronger than `arr[j]` if `arr[i] > arr[j]`.

Return a *list of the strongest `k` values* in the array. return the answer **in any arbitrary order**.

**Median** is the middle value in an ordered integer list. More formally, if the length of the list is `n`, the median is the element in position  $((n - 1) / 2)$  in the sorted list (**0-indexed**).

- For `arr = [6, -3, 7, 2, 11], n = 5` and the median is obtained by sorting the array `arr = [-3, 2, 6, 7, 11]` and the median is `arr[m]` where `m = ((5 - 1) / 2) = 2`. The median is 6.
- For `arr = [-7, 22, 17, 3], n = 4` and the median is obtained by sorting the array `arr = [-7, 3, 17, 22]` and the median is `arr[m]` where `m = ((4 - 1) / 2) = 1`. The median is 3.

**Example 1:**

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

**Output:** [5,1]

**Explanation:** Median is 3, the elements of the array sorted by the strongest are [5,1,4,2,3]. The strongest 2 elements are [5, 1]. [1, 5] is also **accepted** answer.

Please note that although  $|5 - 3| = |1 - 3|$  but 5 is stronger than 1 because  $5 > 1$ .

**Example 2:**

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

**Output:** [5,5]

**Explanation:** Median is 3, the elements of the array sorted by the strongest are [5,5,1,1,3]. The strongest 2 elements are [5, 5].

**Example 3:**

**Input:** arr = [6,7,11,7,6,8], k = 5

**Output:** [11,8,6,6,7]

**Explanation:** Median is 7, the elements of the array sorted by the strongest are [11,8,6,6,7,7].

Any permutation of [11,8,6,6,7] is **accepted**.

**Constraints:**

- $1 \leq \text{arr.length} \leq 10^5$
- $-10^5 \leq \text{arr}[i] \leq 10^5$
- $1 \leq k \leq \text{arr.length}$

## 1472. Design Browser History

Medium

1653118Add to ListShare

You have a **browser** of one tab where you start on the `homepage` and you can visit another `url`, get back in the history number of `steps` or move forward in the history number of `steps`.

Implement the `BrowserHistory` class:

- `BrowserHistory(string homepage)` Initializes the object with the `homepage` of the browser.
- `void visit(string url)` Visits `url` from the current page. It clears up all the forward history.

- `string back(int steps)` Move `steps` back in history. If you can only return `x` steps in the history and `steps > x`, you will return only `x` steps. Return the current `url` after moving back in history **at most** `steps`.
- `string forward(int steps)` Move `steps` forward in history. If you can only forward `x` steps in the history and `steps > x`, you will forward only `x` steps. Return the current `url` after forwarding in history **at most** `steps`.

### Example:

#### Input:

```
["BrowserHistory","visit","visit","visit","back","back","forward","visit","forward","back","back"]
[[["leetcode.com"], ["google.com"], ["facebook.com"], ["youtube.com"], [1], [1], [1], ["linkedin.com"], [2], [2], [7]]]
```

#### Output:

```
[null,null,null,null,"facebook.com","google.com","facebook.com",null,"linkedin.com","google.com","leetcode.com"]
```

### Explanation:

```
BrowserHistory browserHistory = new BrowserHistory("leetcode.com");

browserHistory.visit("google.com");           // You are in "leetcode.com". Visit "google.com"

browserHistory.visit("facebook.com");         // You are in "google.com". Visit "facebook.com"

browserHistory.visit("youtube.com");           // You are in "facebook.com". Visit "youtube.com"

browserHistory.back(1);                      // You are in "youtube.com", move back to "facebook.com" return "facebook.com"

browserHistory.back(1);                      // You are in "facebook.com", move back to "google.com" return "google.com"

browserHistory.forward(1);                   // You are in "google.com", move forward to "facebook.com" return "facebook.com"

browserHistory.visit("linkedin.com");         // You are in "facebook.com". Visit "linkedin.com"
```

```

browserHistory.forward(2);           // You are in "linkedin.com", you cannot
move forward any steps.

browserHistory.back(2);             // You are in "linkedin.com", move back two
steps to "facebook.com" then to "google.com". return "google.com"

browserHistory.back(7);             // You are in "google.com", you can move
back only one step to "leetcode.com". return "leetcode.com"

```

### Constraints:

- `1 <= homepage.length <= 20`
- `1 <= url.length <= 20`
- `1 <= steps <= 100`
- `homepage` and `url` consist of '.' or lower case English letters.
- At most 5000 calls will be made to `visit`, `back`, and `forward`.

## 1473. Paint House III

### Hard

1822134Add to ListShare

There is a row of `m` houses in a small city, each house must be painted with one of the `n` colors (labeled from `1` to `n`), some houses that have been painted last summer should not be painted again.

A neighborhood is a maximal group of continuous houses that are painted with the same color.

- For example: `houses = [1,2,2,3,3,2,1,1]` contains 5 neighborhoods `[[1], [2,2], [3,3], [2], [1,1]]`.

Given an array `houses`, an `m x n` matrix `cost` and an integer `target` where:

- `houses[i]`: is the color of the house `i`, and `0` if the house is not painted yet.
- `cost[i][j]`: is the cost of paint the house `i` with the color `j + 1`.

Return the minimum cost of painting all the remaining houses in such a way that there are exactly `target` neighborhoods. If it is not possible, return `-1`.

### Example 1:

**Input:** `houses = [0,0,0,0,0]`, `cost = [[1,10],[10,1],[10,1],[1,10],[5,1]]`, `m = 5`, `n = 2`, `target = 3`

**Output:** 9

**Explanation:** Paint houses of this way `[1,2,2,1,1]`

This array contains target = 3 neighborhoods, [{1}, {2,2}, {1,1}].

Cost of paint all houses (1 + 1 + 1 + 1 + 5) = 9.

**Example 2:**

**Input:** houses = [0,2,1,2,0], cost = [[1,10],[10,1],[10,1],[1,10],[5,1]], m = 5, n = 2, target = 3

**Output:** 11

**Explanation:** Some houses are already painted, Paint the houses of this way [2,2,1,2,2]

This array contains target = 3 neighborhoods, [{2,2}, {1}, {2,2}].

Cost of paint the first and last house (10 + 1) = 11.

**Example 3:**

**Input:** houses = [3,1,2,3], cost = [[1,1,1],[1,1,1],[1,1,1],[1,1,1]], m = 4, n = 3, target = 3

**Output:** -1

**Explanation:** Houses are already painted with a total of 4 neighborhoods [{3},{1},{2},{3}] different of target = 3.

**Constraints:**

- m == houses.length == cost.length
- n == cost[i].length
- 1 <= m <= 100
- 1 <= n <= 20
- 1 <= target <= m
- 0 <= houses[i] <= n
- 1 <= cost[i][j] <= 10<sup>4</sup>

## 1475. Final Prices With a Special Discount in a Shop

Easy

133880 Add to List Share

You are given an integer array `prices` where `prices[i]` is the price of the  $i^{\text{th}}$  item in a shop.

There is a special discount for items in the shop. If you buy the  $i^{\text{th}}$  item, then you will receive a discount equivalent to `prices[j]` where  $j$  is the minimum index such that  $j > i$  and `prices[j] <= prices[i]`. Otherwise, you will not receive any discount at all.

Return an integer array `answer` where `answer[i]` is the final price you will pay for the  $i^{\text{th}}$  item of the shop, considering the special discount.

**Example 1:**

**Input:** `prices = [8,4,6,2,3]`

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

**Explanation:**

For item 0 with `price[0]=8` you will receive a discount equivalent to `prices[1]=4`, therefore, the final price you will pay is  $8 - 4 = 4$ .

For item 1 with `price[1]=4` you will receive a discount equivalent to `prices[3]=2`, therefore, the final price you will pay is  $4 - 2 = 2$ .

For item 2 with `price[2]=6` you will receive a discount equivalent to `prices[3]=2`, therefore, the final price you will pay is  $6 - 2 = 4$ .

For items 3 and 4 you will not receive any discount at all.

**Example 2:**

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

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

**Explanation:** In this case, for all items, you will not receive any discount at all.

**Example 3:**

**Input:** `prices = [10,1,1,6]`

**Output:** `[9,0,1,6]`

**Constraints:**

- $1 \leq \text{prices.length} \leq 500$
- $1 \leq \text{prices}[i] \leq 1000$

## 1476. Subrectangle Queries

Medium

4391150Add to ListShare

Implement the class `SubrectangleQueries` which receives a `rows x cols` rectangle as a matrix of integers in the constructor and supports two methods:

1. `updateSubrectangle(int row1, int col1, int row2, int col2, int newValue)`
  - Updates all values with `newValue` in the subrectangle whose upper left coordinate is `(row1, col1)` and bottom right coordinate is `(row2, col2)`.
2. `getValue(int row, int col)`
  - Returns the current value of the coordinate `(row, col)` from the rectangle.

### Example 1:

#### Input

```
["SubrectangleQueries","getValue","updateSubrectangle","getValue","getValue","updateSubrectangle","getValue","getValue"]
[[[[1,2,1],[4,3,4],[3,2,1],[1,1,1]]],[0,2],[0,0,3,2,5],[0,2],[3,1],[3,0,3,2,10],[3,1],[0,2]]
```

#### Output

```
[null,1,null,5,5,null,10,5]
```

#### Explanation

```
SubrectangleQueries subrectangleQueries = new
SubrectangleQueries([[1,2,1],[4,3,4],[3,2,1],[1,1,1]]);

// The initial rectangle (4x3) looks like:
// 1 2 1
// 4 3 4
// 3 2 1
// 1 1 1

subrectangleQueries.getValue(0, 2); // return 1

subrectangleQueries.updateSubrectangle(0, 0, 3, 2, 5);

// After this update the rectangle looks like:
// 5 5 5
// 5 5 5
// 5 5 5
```

```

// 5 5 5

subrectangleQueries.getValue(0, 2); // return 5

subrectangleQueries.getValue(3, 1); // return 5

subrectangleQueries.updateSubrectangle(3, 0, 3, 2, 10);

// After this update the rectangle looks like:

// 5 5 5

// 5 5 5

// 5 5 5

// 10 10 10

subrectangleQueries.getValue(3, 1); // return 10

subrectangleQueries.getValue(0, 2); // return 5

```

### Example 2:

#### Input

```

["SubrectangleQueries","getValue","updateSubrectangle","getValue","getValue","updateSubrectangle","getValue"]

[[[[1,1,1],[2,2,2],[3,3,3]]],[0,0],[0,0,2,2,100],[0,0],[2,2],[1,1,2,2,20],[2,2]]

```

#### Output

```
[null,1,null,100,100,null,20]
```

#### Explanation

```

SubrectangleQueries subrectangleQueries = new
SubrectangleQueries([[1,1,1],[2,2,2],[3,3,3]]);

subrectangleQueries.getValue(0, 0); // return 1

subrectangleQueries.updateSubrectangle(0, 0, 2, 2, 100);

subrectangleQueries.getValue(0, 0); // return 100

subrectangleQueries.getValue(2, 2); // return 100

subrectangleQueries.updateSubrectangle(1, 1, 2, 2, 20);

subrectangleQueries.getValue(2, 2); // return 20

```

**Constraints:**

- There will be at most 500 operations considering both methods: `updateSubrectangle` and `getValue`.
- $1 \leq \text{rows}, \text{cols} \leq 100$
- $\text{rows} == \text{rectangle.length}$
- $\text{cols} == \text{rectangle[i].length}$
- $0 \leq \text{row1} \leq \text{row2} < \text{rows}$
- $0 \leq \text{col1} \leq \text{col2} < \text{cols}$
- $1 \leq \text{newValue}, \text{rectangle[i][j]} \leq 10^9$
- $0 \leq \text{row} < \text{rows}$
- $0 \leq \text{col} < \text{cols}$

**1477. Find Two Non-overlapping Sub-arrays Each With Target Sum****Medium**

138570Add to ListShare

You are given an array of integers `arr` and an integer `target`.

You have to find **two non-overlapping sub-arrays** of `arr` each with a sum equal `target`. There can be multiple answers so you have to find an answer where the sum of the lengths of the two sub-arrays is **minimum**.

Return *the minimum sum of the lengths* of the two required sub-arrays, or return `-1` if you cannot find such two sub-arrays.

**Example 1:****Input:** arr = [3,2,2,4,3], target = 3**Output:** 2

**Explanation:** Only two sub-arrays have sum = 3 ([3] and [3]). The sum of their lengths is 2.

**Example 2:****Input:** arr = [7,3,4,7], target = 7**Output:** 2

**Explanation:** Although we have three non-overlapping sub-arrays of sum = 7 ([7], [3,4] and [7]), but we will choose the first and third sub-arrays as the sum of their lengths is 2.

**Example 3:****Input:** arr = [4,3,2,6,2,3,4], target = 6

**Output:** -1

**Explanation:** We have only one sub-array of sum = 6.

**Constraints:**

- $1 \leq \text{arr.length} \leq 10^5$
- $1 \leq \text{arr}[i] \leq 1000$
- $1 \leq \text{target} \leq 10^8$

## 1478. Allocate Mailboxes

Hard

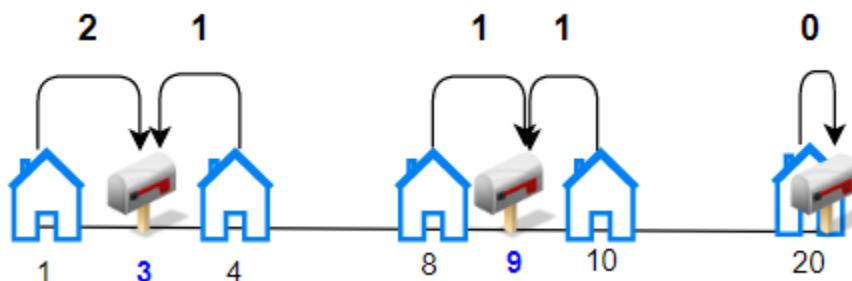
85613Add to ListShare

Given the array `houses` where `houses[i]` is the location of the  $i^{\text{th}}$  house along a street and an integer `k`, allocate `k` mailboxes in the street.

Return *the minimum total distance between each house and its nearest mailbox*.

The test cases are generated so that the answer fits in a 32-bit integer.

**Example 1:**



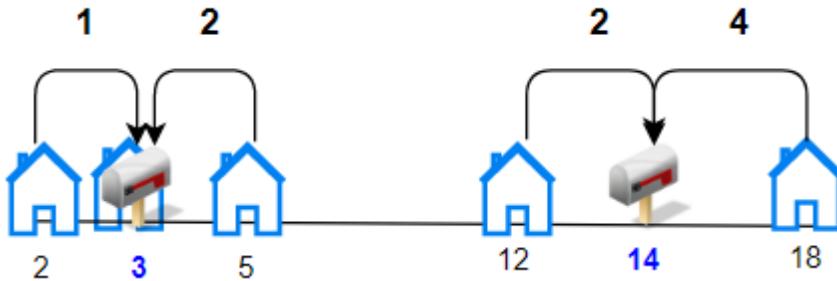
**Input:** `houses = [1,4,8,10,20]`, `k = 3`

**Output:** 5

**Explanation:** Allocate mailboxes in position 3, 9 and 20.

Minimum total distance from each houses to nearest mailboxes is  $|3-1| + |4-3| + |9-8| + |10-9| + |20-20| = 5$

**Example 2:**



**Input:** houses = [2,3,5,12,18], k = 2

**Output:** 9

**Explanation:** Allocate mailboxes in position 3 and 14.

Minimum total distance from each houses to nearest mailboxes is  $|2-3| + |3-3| + |5-3| + |12-14| + |18-14| = 9$ .

### Constraints:

- $1 \leq k \leq \text{houses.length} \leq 100$
- $1 \leq \text{houses}[i] \leq 10^4$
- All the integers of houses are **unique**.

## 1480. Running Sum of 1d Array

**Easy**

4773252Add to ListShare

Given an array `nums`. We define a running sum of an array as `runningSum[i] = sum(nums[0]...nums[i])`.

Return the running sum of `nums`.

### Example 1:

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

**Output:** [1,3,6,10]

**Explanation:** Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].

### Example 2:

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

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

**Explanation:** Running sum is obtained as follows: [1, 1+1, 1+1+1, 1+1+1+1, 1+1+1+1+1].

**Example 3:**

**Input:** nums = [3,1,2,10,1]

**Output:** [3,4,6,16,17]

**Constraints:**

- $1 \leq \text{nums.length} \leq 1000$
- $-10^6 \leq \text{nums}[i] \leq 10^6$

## 1481. Least Number of Unique Integers after K Removals

Medium

106687Add to ListShare

Given an array of integers `arr` and an integer `k`. Find the *least number of unique integers* after removing **exactly** `k` elements.

**Example 1:**

**Input:** arr = [5,5,4], k = 1

**Output:** 1

**Explanation:** Remove the single 4, only 5 is left.

**Example 2:**

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

**Output:** 2

**Explanation:** Remove 4, 2 and either one of the two 1s or three 3s. 1 and 3 will be left.

**Constraints:**

- $1 \leq \text{arr.length} \leq 10^5$
- $1 \leq \text{arr}[i] \leq 10^9$
- $0 \leq k \leq \text{arr.length}$

## 1482. Minimum Number of Days to Make m Bouquets

**Medium**

201951Add to ListShare

You are given an integer array `bloomDay`, an integer `m` and an integer `k`.

You want to make `m` bouquets. To make a bouquet, you need to use `k` adjacent flowers from the garden.

The garden consists of `n` flowers, the `ith` flower will bloom in the `bloomDay[i]` and then can be used in **exactly one** bouquet.

Return *the minimum number of days you need to wait to be able to make `m` bouquets from the garden*. If it is impossible to make `m` bouquets return `-1`.

**Example 1:**

**Input:** `bloomDay = [1,10,3,10,2]`, `m = 3`, `k = 1`

**Output:** 3

**Explanation:** Let us see what happened in the first three days. `x` means flower bloomed and `_` means flower did not bloom in the garden.

We need 3 bouquets each should contain 1 flower.

After day 1: `[x, _, _, _, _]` // we can only make one bouquet.

After day 2: `[x, _, _, _, x]` // we can only make two bouquets.

After day 3: `[x, _, x, _, x]` // we can make 3 bouquets. The answer is 3.

**Example 2:**

**Input:** `bloomDay = [1,10,3,10,2]`, `m = 3`, `k = 2`

**Output:** -1

**Explanation:** We need 3 bouquets each has 2 flowers, that means we need 6 flowers. We only have 5 flowers so it is impossible to get the needed bouquets and we return -1.

**Example 3:**

**Input:** `bloomDay = [7,7,7,7,12,7,7]`, `m = 2`, `k = 3`

**Output:** 12

**Explanation:** We need 2 bouquets each should have 3 flowers.

Here is the garden after the 7 and 12 days:

After day 7: [x, x, x, x, \_, x, x]

We can make one bouquet of the first three flowers that bloomed. We cannot make another bouquet from the last three flowers that bloomed because they are not adjacent.

After day 12: [x, x, x, x, x, x, x]

It is obvious that we can make two bouquets in different ways.

### Constraints:

- `bloomDay.length == n`
- `1 <= n <= 105`
- `1 <= bloomDay[i] <= 109`
- `1 <= m <= 106`
- `1 <= k <= n`

## 1483. Kth Ancestor of a Tree Node

Hard

130389Add to ListShare

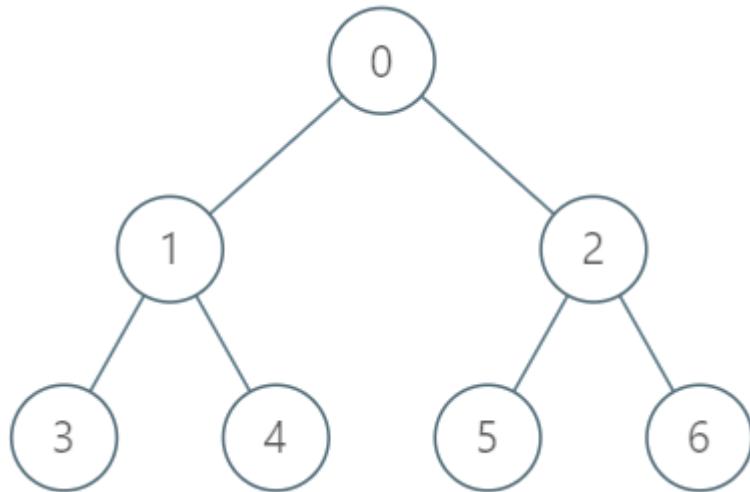
You are given a tree with `n` nodes numbered from `0` to `n - 1` in the form of a parent array `parent` where `parent[i]` is the parent of `ith` node. The root of the tree is node `0`. Find the `kth` ancestor of a given node.

The `kth` ancestor of a tree node is the `kth` node in the path from that node to the root node.

Implement the `TreeAncestor` class:

- `TreeAncestor(int n, int[] parent)` initializes the object with the number of nodes in the tree and the parent array.
- `int getKthAncestor(int node, int k)` return the `kth` ancestor of the given node `node`. If there is no such ancestor, return `-1`.

### Example 1:

**Input**

```
["TreeAncestor", "getKthAncestor", "getKthAncestor", "getKthAncestor"]
[[7, [-1, 0, 0, 1, 1, 2, 2]], [3, 1], [5, 2], [6, 3]]
```

**Output**

```
[null, 1, 0, -1]
```

**Explanation**

```
TreeAncestor treeAncestor = new TreeAncestor(7, [-1, 0, 0, 1, 1, 2, 2]);
treeAncestor.getKthAncestor(3, 1); // returns 1 which is the parent of 3
treeAncestor.getKthAncestor(5, 2); // returns 0 which is the grandparent of 5
treeAncestor.getKthAncestor(6, 3); // returns -1 because there is no such ancestor
```

**Constraints:**

- $1 \leq k \leq n \leq 5 * 10^4$
- $\text{parent.length} == n$
- $\text{parent}[0] == -1$
- $0 \leq \text{parent}[i] < n$  for all  $0 < i < n$
- $0 \leq \text{node} < n$
- There will be at most  $5 * 10^4$  queries.

## 1484. Group Sold Products By The Date

Easy

64651Add to ListShare

SQL Schema

Table Activities:

Column Name	Type
sell_date	date
product	varchar

There is no primary key for this table, it may contain duplicates.

Each row of this table contains the product name and the date it was sold in a market.

Write an SQL query to find for each date the number of different products sold and their names.

The sold products names for each date should be sorted lexicographically.

Return the result table ordered by `sell_date`.

The query result format is in the following example.

### Example 1:

**Input:**

Activities table:

sell_date	product
2020-05-30	Headphone
2020-06-01	Pencil

2020-06-02	Mask	
2020-05-30	Basketball	
2020-06-01	Bible	
2020-06-02	Mask	
2020-05-30	T-Shirt	

**Output:**

sell_date	num_sold	products
2020-05-30	3	Basketball,Headphone,T-shirt
2020-06-01	2	Bible,Pencil
2020-06-02	1	Mask

**Explanation:**

For 2020-05-30, Sold items were (Headphone, Basketball, T-shirt), we sort them lexicographically and separate them by a comma.

For 2020-06-01, Sold items were (Pencil, Bible), we sort them lexicographically and separate them by a comma.

For 2020-06-02, the Sold item is (Mask), we just return it.

## 1486. XOR Operation in an Array

**Easy**

968304Add to ListShare

You are given an integer `n` and an integer `start`.

Define an array `nums` where `nums[i] = start + 2 * i` (**0-indexed**) and `n == nums.length`.

Return *the bitwise XOR of all elements of* `nums`.

**Example 1:**

**Input:** `n = 5, start = 0`

**Output:** 8

**Explanation:** Array `nums` is equal to `[0, 2, 4, 6, 8]` where  $(0 \wedge 2 \wedge 4 \wedge 6 \wedge 8) = 8$ .

Where " $\wedge$ " corresponds to bitwise XOR operator.

**Example 2:**

**Input:** `n = 4, start = 3`

**Output:** 8

**Explanation:** Array `nums` is equal to `[3, 5, 7, 9]` where  $(3 \wedge 5 \wedge 7 \wedge 9) = 8$ .

**Constraints:**

- $1 \leq n \leq 1000$
- $0 \leq \text{start} \leq 1000$
- $n == \text{nums.length}$

## 1487. Making File Names Unique

Medium

367599Add to ListShare

Given an array of strings `names` of size `n`. You will create `n` folders in your file system **such that**, at the `ith` minute, you will create a folder with the name `names[i]`.

Since two files **cannot** have the same name, if you enter a folder name that was previously used, the system will have a suffix addition to its name in the form of `(k)`, where, `k` is the **smallest positive integer** such that the obtained name remains unique.

Return *an array of strings of length n* where `ans[i]` is the actual name the system will assign to the `ith` folder when you create it.

**Example 1:**

**Input:** `names = ["pes", "fifa", "gta", "pes(2019)"]`

**Output:** `["pes", "fifa", "gta", "pes(2019)"]`

**Explanation:** Let's see how the file system creates folder names:

`"pes"` --> not assigned before, remains `"pes"`

`"fifa"` --> not assigned before, remains `"fifa"`

`"gta"` --> not assigned before, remains `"gta"`

```
"pes(2019)" --> not assigned before, remains "pes(2019)"
```

### Example 2:

**Input:** names = ["gta", "gta(1)", "gta", "avalon"]

**Output:** ["gta", "gta(1)", "gta(2)", "avalon"]

**Explanation:** Let's see how the file system creates folder names:

"gta" --> not assigned before, remains "gta"

"gta(1)" --> not assigned before, remains "gta(1)"

"gta" --> the name is reserved, system adds (k), since "gta(1)" is also reserved, systems put k = 2. it becomes "gta(2)"

"avalon" --> not assigned before, remains "avalon"

### Example 3:

**Input:** names = ["onepiece", "onepiece(1)", "onepiece(2)", "onepiece(3)", "onepiece"]

**Output:** ["onepiece", "onepiece(1)", "onepiece(2)", "onepiece(3)", "onepiece(4)"]

**Explanation:** When the last folder is created, the smallest positive valid k is 4, and it becomes "onepiece(4)".

### Constraints:

- $1 \leq \text{names.length} \leq 5 * 10^4$
- $1 \leq \text{names[i].length} \leq 20$
- `names[i]` consists of lowercase English letters, digits, and/or round brackets.

## 1488. Avoid Flood in The City

### Medium

1176228Add to ListShare

Your country has an infinite number of lakes. Initially, all the lakes are empty, but when it rains over the `nth` lake, the `nth` lake becomes full of water. If it rains over a lake that is **full of water**, there will be a **flood**. Your goal is to avoid floods in any lake.

Given an integer array `rains` where:

- `rains[i] > 0` means there will be rains over the `rains[i]` lake.
- `rains[i] == 0` means there are no rains this day and you can choose **one lake** this day and **dry it**.

Return an array `ans` where:

- `ans.length == rains.length`
- `ans[i] == -1` if `rains[i] > 0`.
- `ans[i]` is the lake you choose to dry in the `i`th day if `rains[i] == 0`.

If there are multiple valid answers return **any** of them. If it is impossible to avoid flood return **an empty array**.

Notice that if you chose to dry a full lake, it becomes empty, but if you chose to dry an empty lake, nothing changes.

### Example 1:

**Input:** `rains = [1,2,3,4]`

**Output:** `[-1,-1,-1,-1]`

**Explanation:** After the first day full lakes are `[1]`

After the second day full lakes are `[1,2]`

After the third day full lakes are `[1,2,3]`

After the fourth day full lakes are `[1,2,3,4]`

There's no day to dry any lake and there is no flood in any lake.

### Example 2:

**Input:** `rains = [1,2,0,0,2,1]`

**Output:** `[-1,-1,2,1,-1,-1]`

**Explanation:** After the first day full lakes are `[1]`

After the second day full lakes are `[1,2]`

After the third day, we dry lake 2. Full lakes are `[1]`

After the fourth day, we dry lake 1. There is no full lakes.

After the fifth day, full lakes are `[2]`.

After the sixth day, full lakes are `[1,2]`.

It is easy that this scenario is flood-free. `[-1,-1,1,2,-1,-1]` is another acceptable scenario.

### Example 3:

**Input:** rains = [1,2,0,1,2]

**Output:** []

**Explanation:** After the second day, full lakes are [1,2]. We have to dry one lake in the third day.

After that, it will rain over lakes [1,2]. It's easy to prove that no matter which lake you choose to dry in the 3rd day, the other one will flood.

### Constraints:

- $1 \leq \text{rains.length} \leq 10^5$
- $0 \leq \text{rains}[i] \leq 10^9$

## 1489. Find Critical and Pseudo-Critical Edges in Minimum Spanning Tree

### Hard

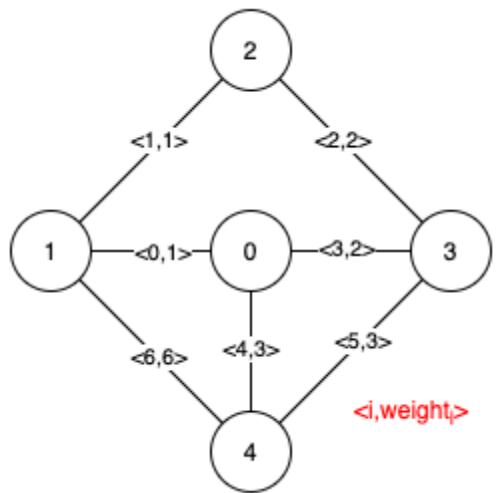
57147Add to ListShare

Given a weighted undirected connected graph with  $n$  vertices numbered from 0 to  $n - 1$ , and an array `edges` where `edges[i] = [ai, bi, weighti]` represents a bidirectional and weighted edge between nodes  $a_i$  and  $b_i$ . A minimum spanning tree (MST) is a subset of the graph's edges that connects all vertices without cycles and with the minimum possible total edge weight.

Find *all the critical and pseudo-critical edges in the given graph's minimum spanning tree (MST)*. An MST edge whose deletion from the graph would cause the MST weight to increase is called a *critical edge*. On the other hand, a pseudo-critical edge is that which can appear in some MSTs but not all.

Note that you can return the indices of the edges in any order.

### Example 1:

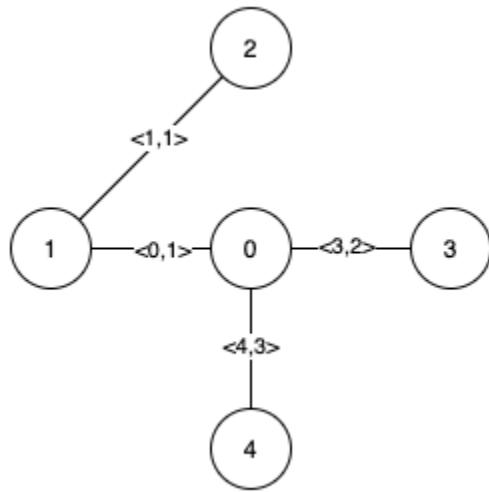
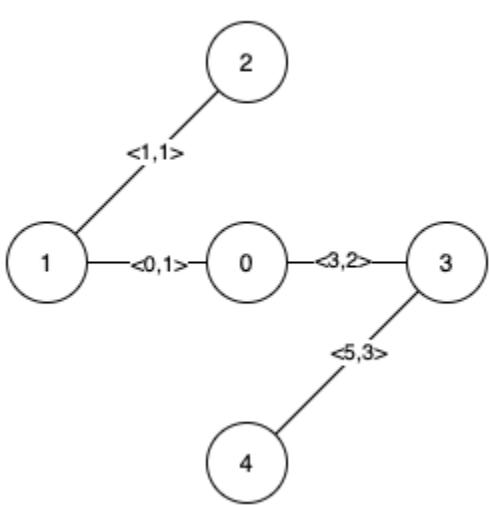
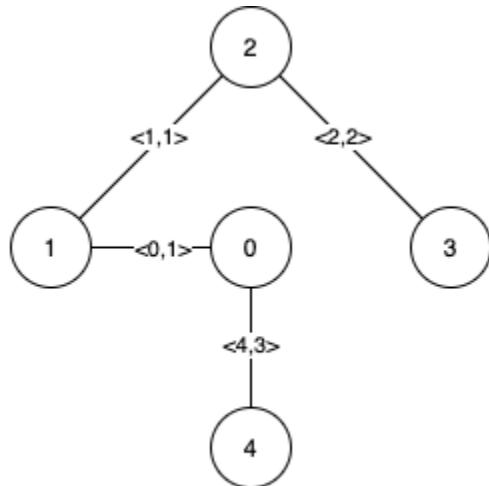
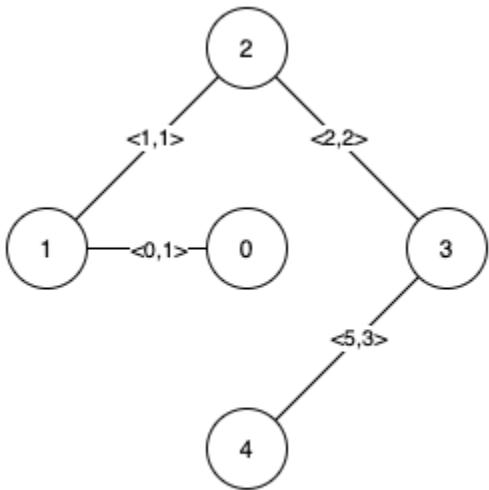


**Input:** `n = 5, edges = [[0,1,1],[1,2,1],[2,3,2],[0,3,2],[0,4,3],[3,4,3],[1,4,6]]`

**Output:** `[[0,1],[2,3,4,5]]`

**Explanation:** The figure above describes the graph.

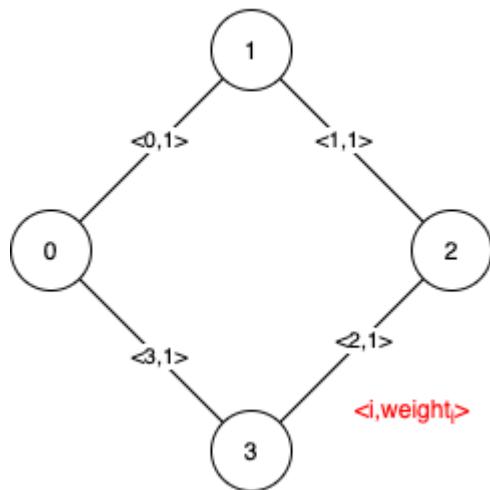
The following figure shows all the possible MSTs:



Notice that the two edges 0 and 1 appear in all MSTs, therefore they are critical edges, so we return them in the first list of the output.

The edges 2, 3, 4, and 5 are only part of some MSTs, therefore they are considered pseudo-critical edges. We add them to the second list of the output.

**Example 2:**



**Input:** `n = 4, edges = [[0,1,1],[1,2,1],[2,3,1],[0,3,1]]`

**Output:** `[[], [0,1,2,3]]`

**Explanation:** We can observe that since all 4 edges have equal weight, choosing any 3 edges from the given 4 will yield an MST. Therefore all 4 edges are pseudo-critical.

#### Constraints:

- `2 <= n <= 100`
- `1 <= edges.length <= min(200, n * (n - 1) / 2)`
- `edges[i].length == 3`
- `0 <= ai < bi < n`
- `1 <= weighti <= 1000`
- All pairs  $(a_i, b_i)$  are **distinct**.

## 1491. Average Salary Excluding the Minimum and Maximum Salary

Easy

914111Add to ListShare

You are given an array of **unique** integers `salary` where `salary[i]` is the salary of the  $i^{\text{th}}$  employee.

Return *the average salary of employees excluding the minimum and maximum salary*. Answers within  $10^{-5}$  of the actual answer will be accepted.

#### Example 1:

**Input:** `salary = [4000,3000,1000,2000]`

**Output:** 2500.00000

**Explanation:** Minimum salary and maximum salary are 1000 and 4000 respectively.

Average salary excluding minimum and maximum salary is  $(2000+3000) / 2 = 2500$

**Example 2:**

**Input:** salary = [1000,2000,3000]

**Output:** 2000.00000

**Explanation:** Minimum salary and maximum salary are 1000 and 3000 respectively.

Average salary excluding minimum and maximum salary is  $(2000) / 1 = 2000$

**Constraints:**

- $3 \leq \text{salary.length} \leq 100$
- $1000 \leq \text{salary}[i] \leq 10^6$
- All the integers of `salary` are **unique**.

## 1492. The kth Factor of n

Medium

869228Add to ListShare

You are given two positive integers `n` and `k`. A factor of an integer `n` is defined as an integer `i` where `n % i == 0`.

Consider a list of all factors of `n` sorted in **ascending order**, return *the `kth` factor* in this list or return `-1` if `n` has less than `k` factors.

**Example 1:**

**Input:** `n = 12, k = 3`

**Output:** 3

**Explanation:** Factors list is [1, 2, 3, 4, 6, 12], the 3<sup>rd</sup> factor is 3.

**Example 2:**

**Input:** `n = 7, k = 2`

**Output:** 7

**Explanation:** Factors list is [1, 7], the 2<sup>nd</sup> factor is 7.

**Example 3:**

**Input:** n = 4, k = 4

**Output:** -1

**Explanation:** Factors list is [1, 2, 4], there is only 3 factors. We should return -1.

**Constraints:**

- 1 <= k <= n <= 1000

**1493. Longest Subarray of 1's After Deleting One Element**

Medium

112922Add to ListShare

Given a binary array `nums`, you should delete one element from it.

Return *the size of the longest non-empty subarray containing only 1's in the resulting array*. Return 0 if there is no such subarray.

**Example 1:**

**Input:** nums = [1,1,0,1]

**Output:** 3

**Explanation:** After deleting the number in position 2, [1,1,1] contains 3 numbers with value of 1's.

**Example 2:**

**Input:** nums = [0,1,1,1,0,1,1,0,1]

**Output:** 5

**Explanation:** After deleting the number in position 4, [0,1,1,1,1,1,0,1] longest subarray with value of 1's is [1,1,1,1,1].

**Example 3:**

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

**Output:** 2

**Explanation:** You must delete one element.

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^5$
- $\text{nums}[i]$  is either 0 or 1.

## 1494. Parallel Courses II

Hard

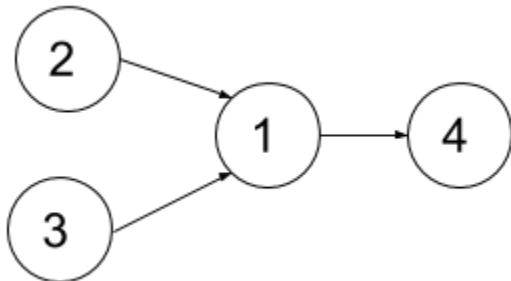
70759 Add to List Share

You are given an integer  $n$ , which indicates that there are  $n$  courses labeled from 1 to  $n$ . You are also given an array  $\text{relations}$  where  $\text{relations}[i] = [\text{prevCourse}_i, \text{nextCourse}_i]$ , representing a prerequisite relationship between course  $\text{prevCourse}_i$  and course  $\text{nextCourse}_i$ : course  $\text{prevCourse}_i$  has to be taken before course  $\text{nextCourse}_i$ . Also, you are given the integer  $k$ .

In one semester, you can take **at most**  $k$  courses as long as you have taken all the prerequisites in the **previous** semesters for the courses you are taking.

Return *the minimum number of semesters needed to take all courses*. The testcases will be generated such that it is possible to take every course.

**Example 1:**



**Input:**  $n = 4$ ,  $\text{relations} = [[2,1], [3,1], [1,4]]$ ,  $k = 2$

**Output:** 3

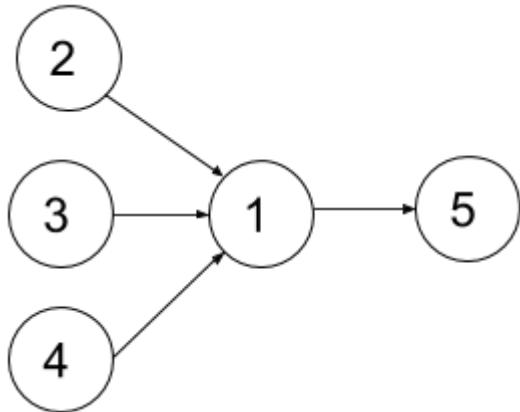
**Explanation:** The figure above represents the given graph.

In the first semester, you can take courses 2 and 3.

In the second semester, you can take course 1.

In the third semester, you can take course 4.

**Example 2:**



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

**Output:** 4

**Explanation:** The figure above represents the given graph.

In the first semester, you can only take courses 2 and 3 since you cannot take more than two per semester.

In the second semester, you can take course 4.

In the third semester, you can take course 1.

In the fourth semester, you can take course 5.

### Constraints:

- $1 \leq n \leq 15$
- $1 \leq k \leq n$
- $0 \leq \text{relations.length} \leq n * (n-1) / 2$
- $\text{relations}[i].length == 2$
- $1 \leq \text{prevCourse}_i, \text{nextCourse}_i \leq n$
- $\text{prevCourse}_i \neq \text{nextCourse}_i$
- All the pairs  $[\text{prevCourse}_i, \text{nextCourse}_i]$  are **unique**.
- The given graph is a directed acyclic graph.

## 1496. Path Crossing

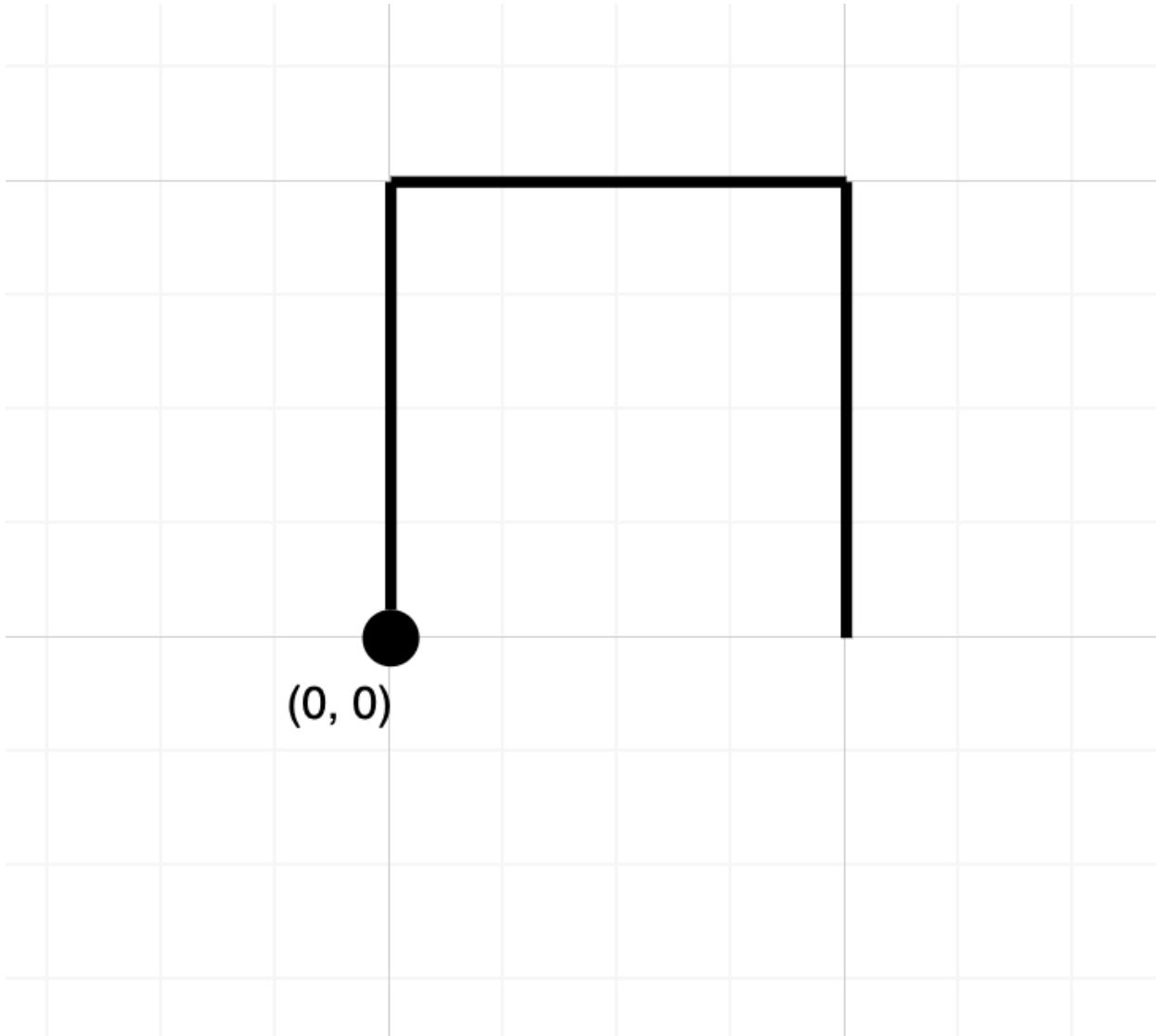
Easy

53710Add to ListShare

Given a string `path`, where `path[i] = 'N', 'S', 'E' or 'W'`, each representing moving one unit north, south, east, or west, respectively. You start at the origin  $(0, 0)$  on a 2D plane and walk on the path specified by `path`.

Return `true` if the path crosses itself at any point, that is, if at any time you are on a location you have previously visited. Return `false` otherwise.

**Example 1:**

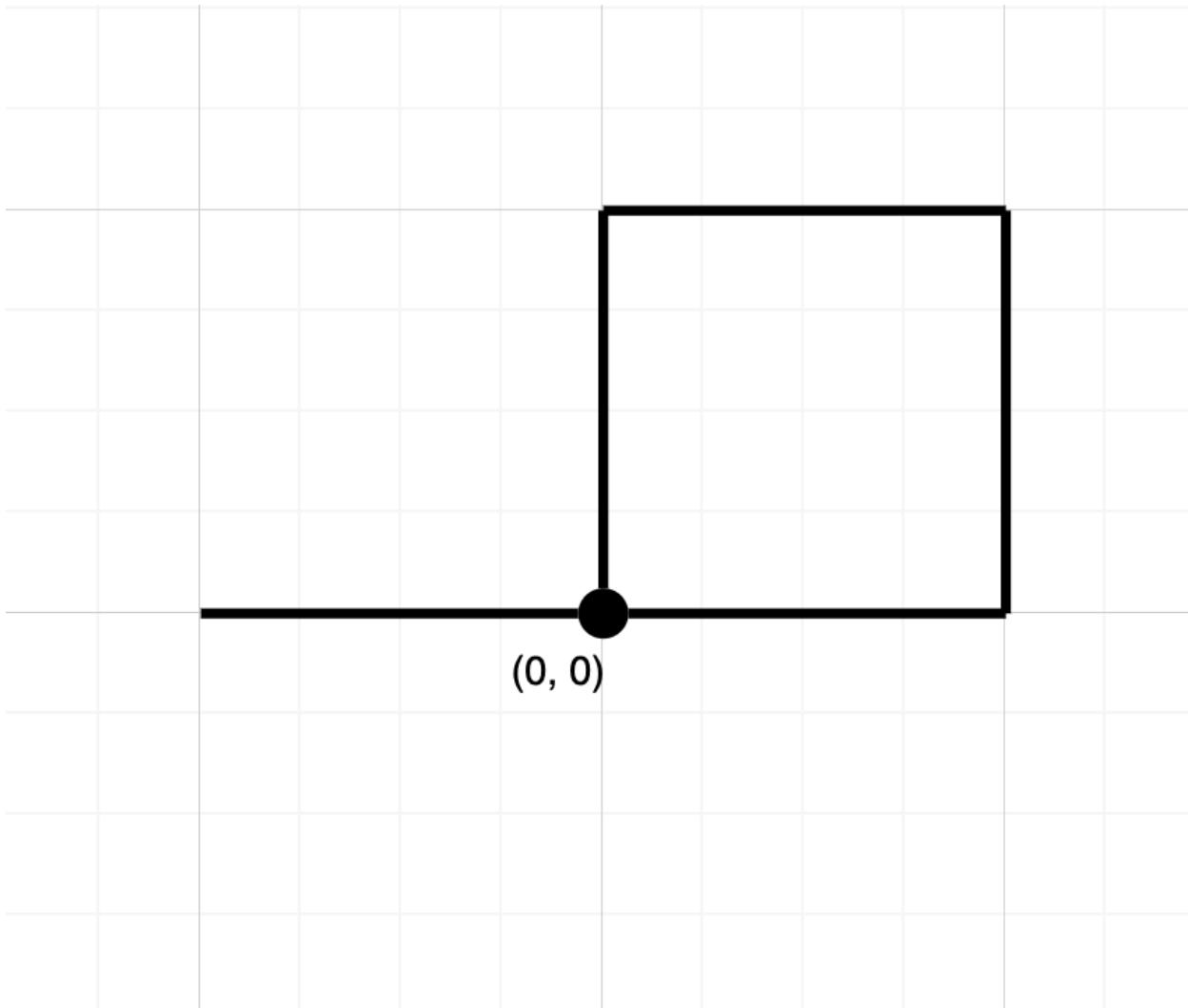


**Input:** path = "NES"

**Output:** false

**Explanation:** Notice that the path doesn't cross any point more than once.

**Example 2:**



Input: path = "NESWW"

Output: true

**Explanation:** Notice that the path visits the origin twice.

## Constraints:

- $1 \leq \text{path.length} \leq 10^4$
  - `path[i]` is either 'N', 'S', 'E', or 'W'.

## 1497. Check If Array Pairs Are Divisible by k

## Medium

120671Add to ListShare

Given an array of integers `arr` of even length `n` and an integer `k`.

We want to divide the array into exactly  $n / 2$  pairs such that the sum of each pair is divisible by  $k$ .

Return `true` If you can find a way to do that or `false` otherwise.

### Example 1:

**Input:** arr = [1,2,3,4,5,10,6,7,8,9], k = 5

**Output:** true

**Explanation:** Pairs are (1,9),(2,8),(3,7),(4,6) and (5,10).

### Example 2:

**Input:** arr = [1,2,3,4,5,6], k = 7

**Output:** true

**Explanation:** Pairs are (1,6),(2,5) and(3,4).

### Example 3:

**Input:** arr = [1,2,3,4,5,6], k = 10

**Output:** false

**Explanation:** You can try all possible pairs to see that there is no way to divide arr into 3 pairs each with sum divisible by 10.

### Constraints:

- `arr.length == n`
- $1 \leq n \leq 10^5$
- `n` is even.
- $-10^9 \leq arr[i] \leq 10^9$
- $1 \leq k \leq 10^5$

## 1498. Number of Subsequences That Satisfy the Given Sum Condition

Medium

1550131Add to ListShare

You are given an array of integers `nums` and an integer `target`.

Return the number of **non-empty** subsequences of `nums` such that the sum of the minimum and maximum element on it is less or equal to `target`. Since the answer may be too large, return it **modulo**  $10^9 + 7$ .

**Example 1:**

**Input:** nums = [3,5,6,7], target = 9

**Output:** 4

**Explanation:** There are 4 subsequences that satisfy the condition.

[3] -> Min value + max value  $\leq$  target ( $3 + 3 \leq 9$ )

[3,5] -> ( $3 + 5 \leq 9$ )

[3,5,6] -> ( $3 + 6 \leq 9$ )

[3,6] -> ( $3 + 6 \leq 9$ )

**Example 2:**

**Input:** nums = [3,3,6,8], target = 10

**Output:** 6

**Explanation:** There are 6 subsequences that satisfy the condition. (nums can have repeated numbers).

[3] , [3] , [3,3] , [3,6] , [3,6] , [3,3,6]

**Example 3:**

**Input:** nums = [2,3,3,4,6,7], target = 12

**Output:** 61

**Explanation:** There are 63 non-empty subsequences, two of them do not satisfy the condition ([6,7], [7]).

Number of valid subsequences ( $63 - 2 = 61$ ).

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^5$
- $1 \leq \text{nums}[i] \leq 10^6$
- $1 \leq \text{target} \leq 10^6$

**1499. Max Value of Equation**

Hard

100336Add to ListShare

You are given an array `points` containing the coordinates of points on a 2D plane, sorted by the  $x$ -values, where `points[i] = [xi, yi]` such that  $x<sub>i</sub> < x<sub>j</sub>$  for all  $1 \leq i < j \leq \text{points.length}$ . You are also given an integer `k`.

Return the maximum value of the equation  $y<sub>i</sub> + y<sub>j</sub> + |x<sub>i</sub> - x<sub>j</sub>|$  where  $|x<sub>i</sub> - x<sub>j</sub>| \leq k$  and  $1 \leq i < j \leq \text{points.length}$ .

It is guaranteed that there exists at least one pair of points that satisfy the constraint  $|x<sub>i</sub> - x<sub>j</sub>| \leq k$ .

### Example 1:

**Input:** `points = [[1,3],[2,0],[5,10],[6,-10]]`, `k = 1`

**Output:** 4

**Explanation:** The first two points satisfy the condition  $|x<sub>i</sub> - x<sub>j</sub>| \leq 1$  and if we calculate the equation we get  $3 + 0 + |1 - 2| = 4$ . Third and fourth points also satisfy the condition and give a value of  $10 + -10 + |5 - 6| = 1$ .

No other pairs satisfy the condition, so we return the max of 4 and 1.

### Example 2:

**Input:** `points = [[0,0],[3,0],[9,2]]`, `k = 3`

**Output:** 3

**Explanation:** Only the first two points have an absolute difference of 3 or less in the  $x$ -values, and give the value of  $0 + 0 + |0 - 3| = 3$ .

### Constraints:

- $2 \leq \text{points.length} \leq 10^5$
- `points[i].length == 2`
- $-10^8 \leq x_i, y_i \leq 10^8$
- $0 \leq k \leq 2 * 10^8$
- $x_i < x_j$  for all  $1 \leq i < j \leq \text{points.length}$
- $x_i$  form a strictly increasing sequence.

## 1502. Can Make Arithmetic Progression From Sequence

Easy

8375Add to ListShare

A sequence of numbers is called an **arithmetic progression** if the difference between any two consecutive elements is the same.

Given an array of numbers `arr`, return `true` if the array can be rearranged to form an **arithmetic progression**. Otherwise, return `false`.

**Example 1:**

**Input:** `arr = [3,5,1]`

**Output:** `true`

**Explanation:** We can reorder the elements as `[1,3,5]` or `[5,3,1]` with differences 2 and -2 respectively, between each consecutive elements.

**Example 2:**

**Input:** `arr = [1,2,4]`

**Output:** `false`

**Explanation:** There is no way to reorder the elements to obtain an arithmetic progression.

**Constraints:**

- $2 \leq \text{arr.length} \leq 1000$
- $-10^6 \leq \text{arr}[i] \leq 10^6$

## 1503. Last Moment Before All Ants Fall Out of a Plank

Medium

388188Add to ListShare

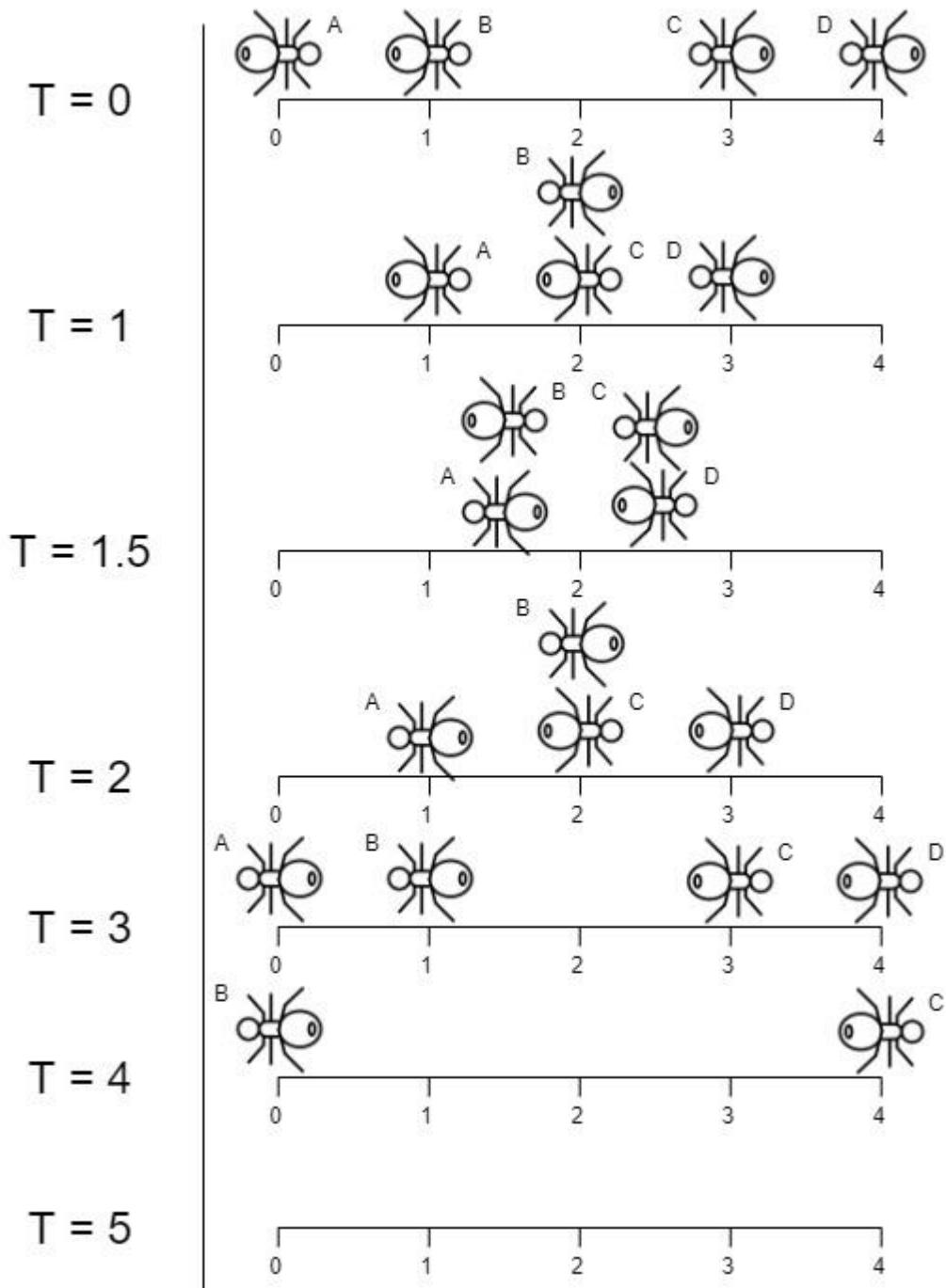
We have a wooden plank of the length `n` **units**. Some ants are walking on the plank, each ant moves with a speed of **1 unit per second**. Some of the ants move to the **left**, the other move to the **right**.

When two ants moving in two **different** directions meet at some point, they change their directions and continue moving again. Assume changing directions does not take any additional time.

When an ant reaches **one end** of the plank at a time `t`, it falls out of the plank immediately.

Given an integer `n` and two integer arrays `left` and `right`, the positions of the ants moving to the left and the right, return *the moment when the last ant(s) fall out of the plank*.

**Example 1:**



**Input:**  $n = 4$ ,  $\text{left} = [4, 3]$ ,  $\text{right} = [0, 1]$

**Output:** 4

**Explanation:** In the image above:

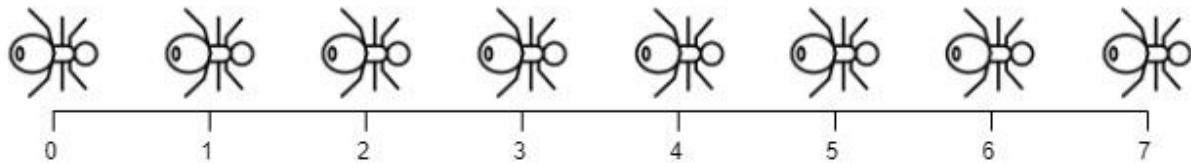
- The ant at index 0 is named A and going to the right.
- The ant at index 1 is named B and going to the right.

-The ant at index 3 is named C and going to the left.

-The ant at index 4 is named D and going to the left.

The last moment when an ant was on the plank is  $t = 4$  seconds. After that, it falls immediately out of the plank. (i.e., We can say that at  $t = 4.0000000001$ , there are no ants on the plank).

### Example 2:

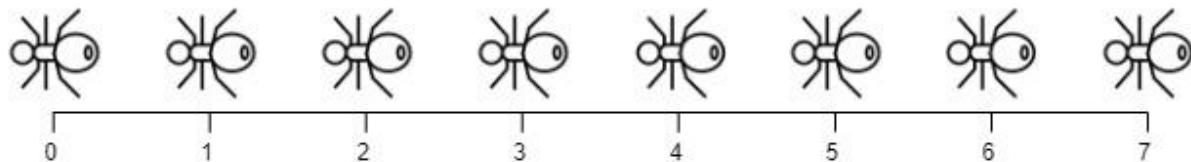


**Input:**  $n = 7$ ,  $\text{left} = []$ ,  $\text{right} = [0,1,2,3,4,5,6,7]$

**Output:** 7

**Explanation:** All ants are going to the right, the ant at index 0 needs 7 seconds to fall.

### Example 3:



**Input:**  $n = 7$ ,  $\text{left} = [0,1,2,3,4,5,6,7]$ ,  $\text{right} = []$

**Output:** 7

**Explanation:** All ants are going to the left, the ant at index 7 needs 7 seconds to fall.

### Constraints:

- $1 \leq n \leq 10^4$
- $0 \leq \text{left.length} \leq n + 1$
- $0 \leq \text{left}[i] \leq n$
- $0 \leq \text{right.length} \leq n + 1$
- $0 \leq \text{right}[i] \leq n$
- $1 \leq \text{left.length} + \text{right.length} \leq n + 1$
- All values of `left` and `right` are unique, and each value can appear **only in one** of the two arrays.

## 1504. Count Submatrices With All Ones

Medium

1697136Add to ListShare

Given an  $m \times n$  binary matrix  $\text{mat}$ , return the number of **submatrices** that have all ones.

**Example 1:**

1	0	1
1	1	0
1	1	0

**Input:**  $\text{mat} = [[1,0,1],[1,1,0],[1,1,0]]$

**Output:** 13

**Explanation:**

There are 6 rectangles of side 1x1.

There are 2 rectangles of side 1x2.

There are 3 rectangles of side 2x1.

There is 1 rectangle of side 2x2.

There is 1 rectangle of side 3x1.

Total number of rectangles =  $6 + 2 + 3 + 1 + 1 = 13$ .

**Example 2:**

0	1	1	0
0	1	1	1
1	1	1	0

**Input:** mat = [[0,1,1,0],[0,1,1,1],[1,1,1,0]]

**Output:** 24

**Explanation:**

There are 8 rectangles of side 1x1.

There are 5 rectangles of side 1x2.

There are 2 rectangles of side 1x3.

There are 4 rectangles of side 2x1.

There are 2 rectangles of side 2x2.

There are 2 rectangles of side 3x1.

There is 1 rectangle of side 3x2.

Total number of rectangles = 8 + 5 + 2 + 4 + 2 + 2 + 1 = 24.

**Constraints:**

- $1 \leq m, n \leq 150$
- `mat[i][j]` is either 0 or 1.

## 1505. Minimum Possible Integer After at Most K Adjacent Swaps On Digits

**Hard**

37522Add to ListShare

You are given a string `num` representing **the digits** of a very large integer and an integer `k`. You are allowed to swap any two adjacent digits of the integer **at most** `k` times.

Return *the minimum integer you can obtain also as a string*.

**Example 1:**

4321 → 3421 → 3412 → 3142 → 1342

**Input:** num = "4321", k = 4

**Output:** "1342"

**Explanation:** The steps to obtain the minimum integer from 4321 with 4 adjacent swaps are shown.

**Example 2:**

**Input:** num = "100", k = 1

**Output:** "010"

**Explanation:** It's ok for the output to have leading zeros, but the input is guaranteed not to have any leading zeros.

**Example 3:**

**Input:** num = "36789", k = 1000

**Output:** "36789"

**Explanation:** We can keep the number without any swaps.

**Constraints:**

- $1 \leq \text{num.length} \leq 3 * 10^4$
- num consists of only **digits** and does not contain **leading zeros**.
- $1 \leq k \leq 10^9$

**1507. Reformat Date**

Easy

297365Add to ListShare

Given a date string in the form Day Month Year, where:

- Day is in the set {"1st", "2nd", "3rd", "4th", ..., "30th", "31st"}.
- Month is in the set {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"}.
- Year is in the range [1900, 2100].

Convert the date string to the format `YYYY-MM-DD`, where:

- `YYYY` denotes the 4 digit year.
- `MM` denotes the 2 digit month.
- `DD` denotes the 2 digit day.

**Example 1:**

**Input:** date = "20th Oct 2052"

**Output:** "2052-10-20"

**Example 2:**

**Input:** date = "6th Jun 1933"

**Output:** "1933-06-06"

**Example 3:**

**Input:** date = "26th May 1960"

**Output:** "1960-05-26"

**Constraints:**

- The given dates are guaranteed to be valid, so no error handling is necessary.

## 1508. Range Sum of Sorted Subarray Sums

Medium

716126Add to ListShare

You are given the array `nums` consisting of `n` positive integers. You computed the sum of all non-empty continuous subarrays from the array and then sorted them in non-decreasing order, creating a new array of `n * (n + 1) / 2` numbers.

*Return the sum of the numbers from index `left` to index `right` (**indexed from 1**), inclusive, in the new array. Since the answer can be a huge number return it modulo  $10^9 + 7$ .*

**Example 1:**

**Input:** `nums` = [1,2,3,4], `n` = 4, `left` = 1, `right` = 5

**Output:** 13

**Explanation:** All subarray sums are 1, 3, 6, 10, 2, 5, 9, 3, 7, 4. After sorting them in non-decreasing order we have the new array [1, 2, 3, 3, 4, 5, 6, 7, 9, 10]. The sum of the numbers from index  $le = 1$  to  $ri = 5$  is  $1 + 2 + 3 + 3 + 4 = 13$ .

**Example 2:**

**Input:** `nums = [1,2,3,4], n = 4, left = 3, right = 4`

**Output:** 6

**Explanation:** The given array is the same as example 1. We have the new array [1, 2, 3, 3, 4, 5, 6, 7, 9, 10]. The sum of the numbers from index  $le = 3$  to  $ri = 4$  is  $3 + 3 = 6$ .

**Example 3:**

**Input:** `nums = [1,2,3,4], n = 4, left = 1, right = 10`

**Output:** 50

**Constraints:**

- $n == \text{nums.length}$
- $1 \leq \text{nums.length} \leq 1000$
- $1 \leq \text{nums}[i] \leq 100$
- $1 \leq \text{left} \leq \text{right} \leq n * (n + 1) / 2$

## 1509. Minimum Difference Between Largest and Smallest Value in Three Moves

Medium

1272153Add to ListShare

You are given an integer array `nums`. In one move, you can choose one element of `nums` and change it by **any value**.

Return *the minimum difference between the largest and smallest value of `nums` after performing **at most three moves***.

**Example 1:**

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

**Output:** 0

**Explanation:** Change the array [5,3,2,4] to [2,2,2,2].

The difference between the maximum and minimum is  $2-2 = 0$ .

**Example 2:**

**Input:** `nums = [1,5,0,10,14]`

**Output:** `1`

**Explanation:** Change the array `[1,5,0,10,14]` to `[1,1,0,1,1]`.

The difference between the maximum and minimum is  $1-0 = 1$ .

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^5$
- $-10^9 \leq \text{nums}[i] \leq 10^9$

## 1510. Stone Game IV

Hard

135661Add to ListShare

Alice and Bob take turns playing a game, with Alice starting first.

Initially, there are `n` stones in a pile. On each player's turn, that player makes a *move* consisting of removing **any** non-zero **square number** of stones in the pile.

Also, if a player cannot make a move, he/she loses the game.

Given a positive integer `n`, return `true` if and only if Alice wins the game otherwise return `false`, assuming both players play optimally.

**Example 1:**

**Input:** `n = 1`

**Output:** `true`

**Explanation:** Alice can remove 1 stone winning the game because Bob doesn't have any moves.

**Example 2:**

**Input:** `n = 2`

**Output:** `false`

**Explanation:** Alice can only remove 1 stone, after that Bob removes the last one winning the game (2 -> 1 -> 0).

**Example 3:**

**Input:** n = 4

**Output:** true

**Explanation:** n is already a perfect square, Alice can win with one move, removing 4 stones (4 -> 0).

**Constraints:**

- $1 \leq n \leq 10^5$

## 1512. Number of Good Pairs

**Easy**

3044158Add to ListShare

Given an array of integers `nums`, return *the number of good pairs*.

A pair  $(i, j)$  is called *good* if  $\text{nums}[i] == \text{nums}[j]$  and  $i < j$ .

**Example 1:**

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

**Output:** 4

**Explanation:** There are 4 good pairs (0,3), (0,4), (3,4), (2,5) 0-indexed.

**Example 2:**

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

**Output:** 6

**Explanation:** Each pair in the array are *good*.

**Example 3:**

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

**Output:** 0

**Constraints:**

- $1 \leq \text{nums.length} \leq 100$
- $1 \leq \text{nums}[i] \leq 100$

**1513. Number of Substrings With Only 1s****Medium**

61828Add to ListShare

Given a binary string  $s$ , return the number of substrings with all characters 1's. Since the answer may be too large, return it modulo  $10^9 + 7$ .

**Example 1:****Input:**  $s = "0110111"$ **Output:** 9**Explanation:** There are 9 substring in total with only 1's characters. $"1" \rightarrow 5 \text{ times.}$  $"11" \rightarrow 3 \text{ times.}$  $"111" \rightarrow 1 \text{ time.}$ **Example 2:****Input:**  $s = "101"$ **Output:** 2**Explanation:** Substring "1" is shown 2 times in  $s$ .**Example 3:****Input:**  $s = "111111"$ **Output:** 21**Explanation:** Each substring contains only 1's characters.**Constraints:**

- $1 \leq s.length \leq 10^5$
- $s[i]$  is either '0' or '1'.

## 1514. Path with Maximum Probability

Medium

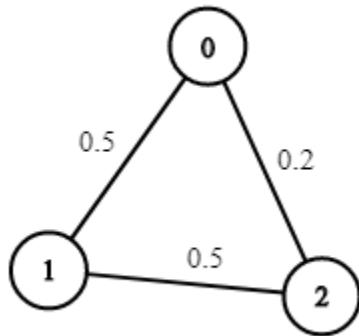
146628Add to ListShare

You are given an undirected weighted graph of `n` nodes (0-indexed), represented by an edge list where `edges[i] = [a, b]` is an undirected edge connecting the nodes `a` and `b` with a probability of success of traversing that edge `succProb[i]`.

Given two nodes `start` and `end`, find the path with the maximum probability of success to go from `start` to `end` and return its success probability.

If there is no path from `start` to `end`, **return 0**. Your answer will be accepted if it differs from the correct answer by at most **1e-5**.

**Example 1:**

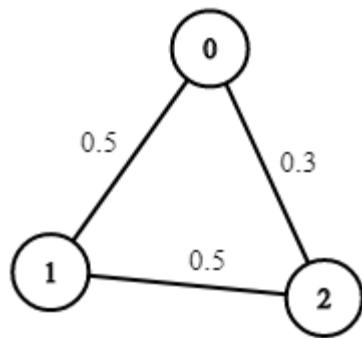


**Input:** `n = 3, edges = [[0,1],[1,2],[0,2]], succProb = [0.5,0.5,0.2], start = 0, end = 2`

**Output:** `0.25000`

**Explanation:** There are two paths from start to end, one having a probability of success = `0.2` and the other has  $0.5 * 0.5 = 0.25$ .

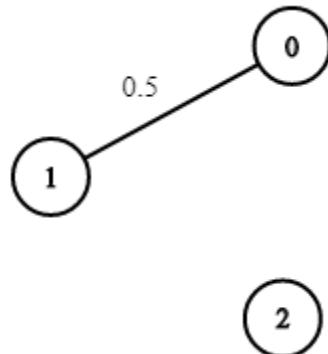
**Example 2:**



**Input:** n = 3, edges = [[0,1],[1,2],[0,2]], succProb = [0.5,0.5,0.3], start = 0, end = 2

**Output:** 0.30000

**Example 3:**



**Input:** n = 3, edges = [[0,1]], succProb = [0.5], start = 0, end = 2

**Output:** 0.00000

**Explanation:** There is no path between 0 and 2.

**Constraints:**

- $2 \leq n \leq 10^4$
- $0 \leq \text{start}, \text{end} < n$
- $\text{start} \neq \text{end}$
- $0 \leq a, b < n$
- $a \neq b$
- $0 \leq \text{succProb.length} == \text{edges.length} \leq 2 \times 10^4$
- $0 \leq \text{succProb}[i] \leq 1$
- There is at most one edge between every two nodes.

## 1515. Best Position for a Service Centre

Hard

186220Add to ListShare

A delivery company wants to build a new service center in a new city. The company knows the positions of all the customers in this city on a 2D-Map and wants to build the new center in a position such that **the sum of the euclidean distances to all customers is minimum**.

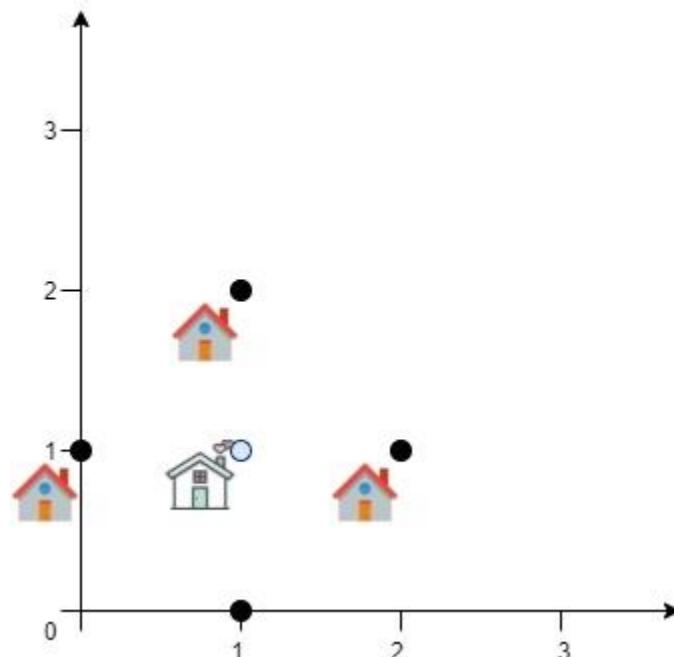
Given an array `positions` where `positions[i] = [xi, yi]` is the position of the *i*th customer on the map, return *the minimum sum of the euclidean distances to all customers*.

In other words, you need to choose the position of the service center `[xcentre, ycentre]` such that the following formula is minimized:

$$\sum_{i=0}^{n-1} \sqrt{(x_{\text{centre}} - x_i)^2 + (y_{\text{centre}} - y_i)^2}$$

Answers within  $10^{-5}$  of the actual value will be accepted.

### Example 1:

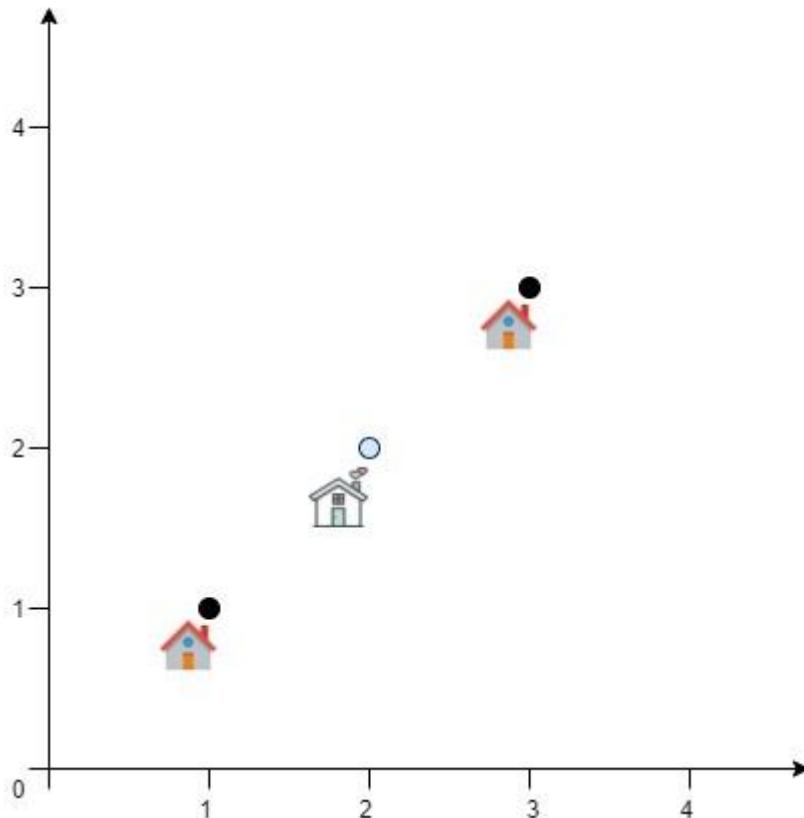


**Input:** `positions = [[0,1],[1,0],[1,2],[2,1]]`

**Output:** `4.00000`

**Explanation:** As shown, you can see that choosing  $[x_{\text{centre}}, y_{\text{centre}}] = [1, 1]$  will make the distance to each customer = 1, the sum of all distances is 4 which is the minimum possible we can achieve.

**Example 2:**



**Input:** positions = [[1,1],[3,3]]

**Output:** 2.82843

**Explanation:** The minimum possible sum of distances =  $\sqrt{2} + \sqrt{2} = 2.82843$

**Constraints:**

- $1 \leq \text{positions.length} \leq 50$
- $\text{positions}[i].length == 2$
- $0 \leq x_i, y_i \leq 100$

## 1518. Water Bottles

Easy

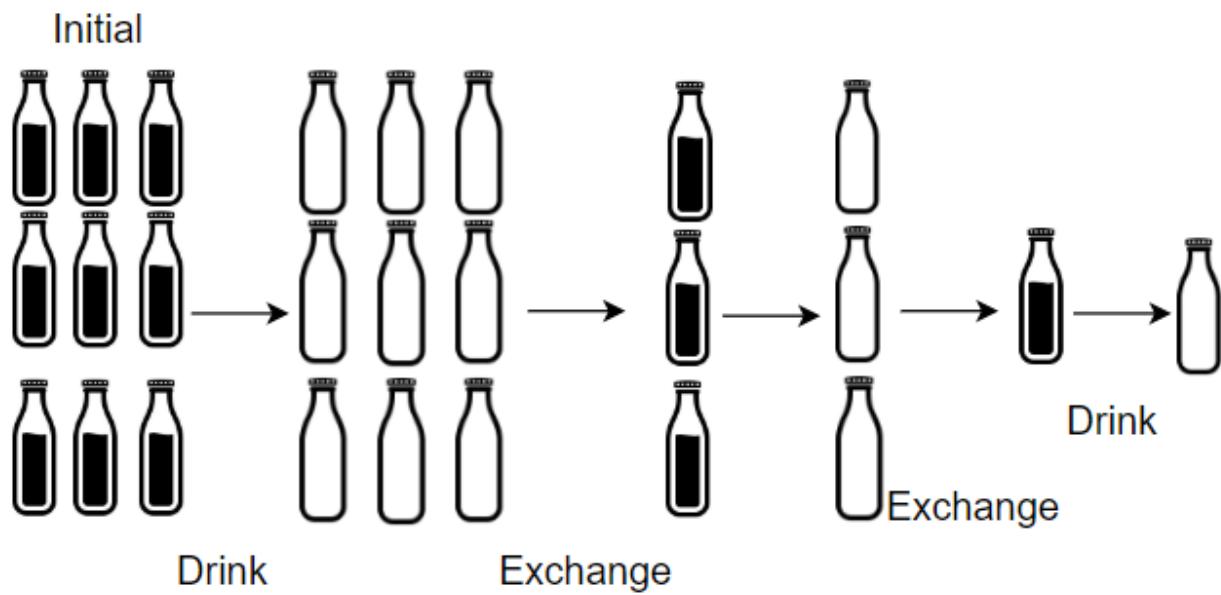
75457 Add to List Share

There are `numBottles` water bottles that are initially full of water. You can exchange `numExchange` empty water bottles from the market with one full water bottle.

The operation of drinking a full water bottle turns it into an empty bottle.

Given the two integers `numBottles` and `numExchange`, return *the maximum number of water bottles you can drink*.

**Example 1:**



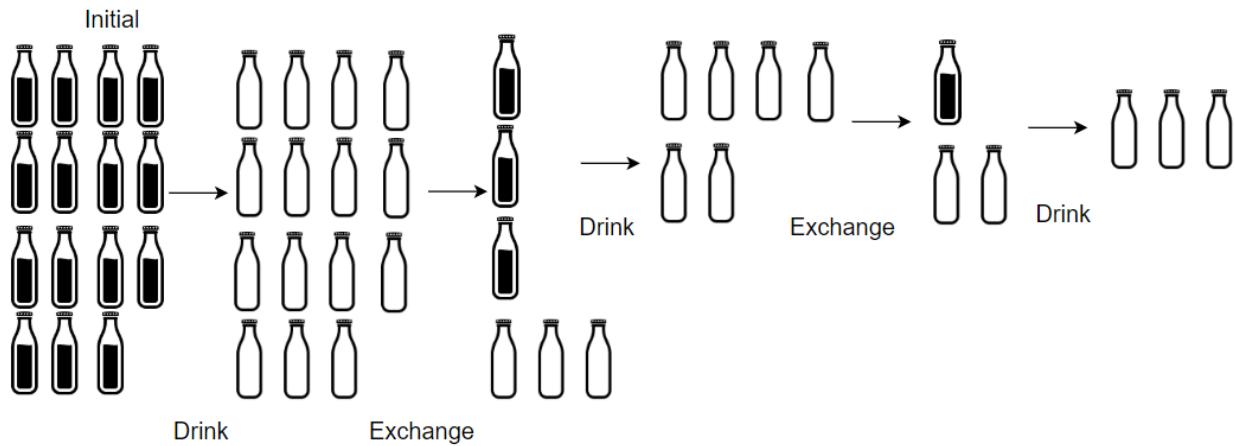
**Input:** `numBottles` = 9, `numExchange` = 3

**Output:** 13

**Explanation:** You can exchange 3 empty bottles to get 1 full water bottle.

Number of water bottles you can drink:  $9 + 3 + 1 = 13$ .

**Example 2:**



**Input:** numBottles = 15, numExchange = 4

**Output:** 19

**Explanation:** You can exchange 4 empty bottles to get 1 full water bottle.

Number of water bottles you can drink:  $15 + 3 + 1 = 19$ .

### Constraints:

- $1 \leq \text{numBottles} \leq 100$
- $2 \leq \text{numExchange} \leq 100$

## 1519. Number of Nodes in the Sub-Tree With the Same Label

Medium

477414Add to ListShare

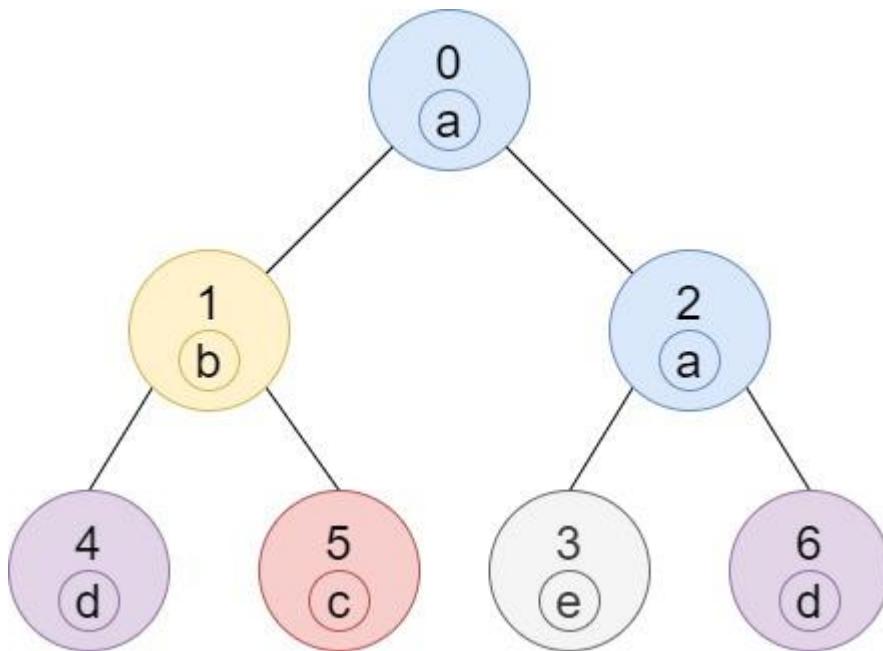
You are given a tree (i.e. a connected, undirected graph that has no cycles) consisting of  $n$  nodes numbered from 0 to  $n - 1$  and exactly  $n - 1$  edges. The **root** of the tree is the node 0, and each node of the tree has a **label** which is a lower-case character given in the string `labels` (i.e. The node with the number  $i$  has the label `labels[i]`).

The `edges` array is given on the form `edges[i] = [ai, bi]`, which means there is an edge between nodes  $a_i$  and  $b_i$  in the tree.

Return an array of size  $n$  where `ans[i]` is the number of nodes in the subtree of the  $i^{\text{th}}$  node which have the same label as node  $i$ .

A subtree of a tree  $T$  is the tree consisting of a node in  $T$  and all of its descendant nodes.

### Example 1:



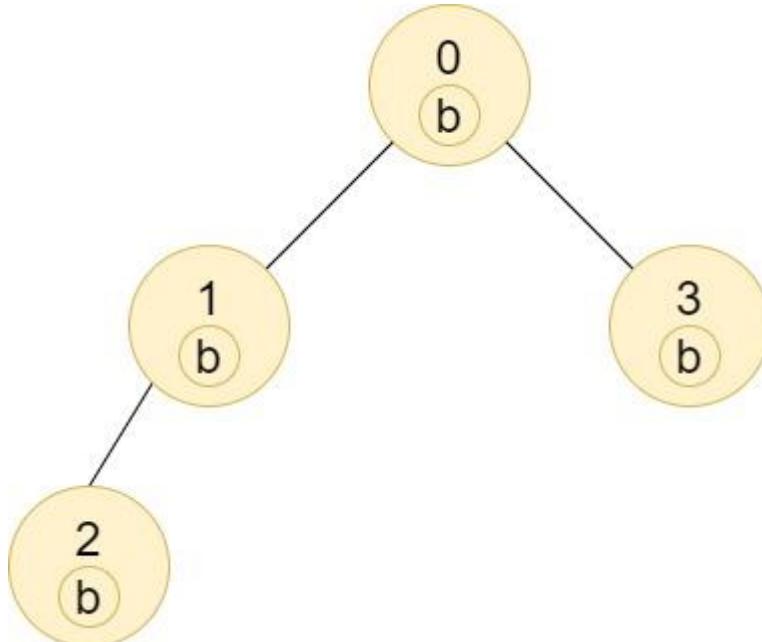
**Input:** n = 7, edges = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], labels = "abaedcd"

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

**Explanation:** Node 0 has label 'a' and its sub-tree has node 2 with label 'a' as well, thus the answer is 2. Notice that any node is part of its sub-tree.

Node 1 has a label 'b'. The sub-tree of node 1 contains nodes 1,4 and 5, as nodes 4 and 5 have different labels than node 1, the answer is just 1 (the node itself).

**Example 2:**



**Input:** n = 4, edges = [[0,1],[1,2],[0,3]], labels = "bbbb"

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

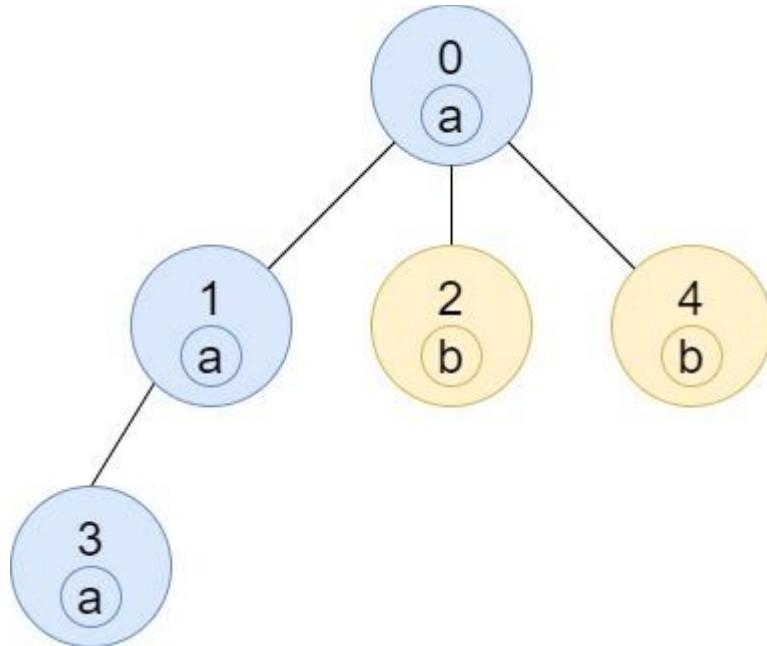
**Explanation:** The sub-tree of node 2 contains only node 2, so the answer is 1.

The sub-tree of node 3 contains only node 3, so the answer is 1.

The sub-tree of node 1 contains nodes 1 and 2, both have label 'b', thus the answer is 2.

The sub-tree of node 0 contains nodes 0, 1, 2 and 3, all with label 'b', thus the answer is 4.

**Example 3:**



**Input:** n = 5, edges = [[0,1],[0,2],[1,3],[0,4]], labels = "aabab"

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

**Constraints:**

- $1 \leq n \leq 10^5$
- $\text{edges.length} == n - 1$
- $\text{edges}[i].length == 2$
- $0 \leq a_i, b_i < n$
- $a_i \neq b_i$
- $\text{labels.length} == n$
- $\text{labels}$  is consisting of only of lowercase English letters.

## 1520. Maximum Number of Non-Overlapping Substrings

**Hard**

60362Add to ListShare

Given a string  $s$  of lowercase letters, you need to find the maximum number of **non-empty** substrings of  $s$  that meet the following conditions:

1. The substrings do not overlap, that is for any two substrings  $s[i..j]$  and  $s[x..y]$ , either  $j < x$  or  $i > y$  is true.
2. A substring that contains a certain character  $c$  must also contain all occurrences of  $c$ .

Find the maximum number of substrings that meet the above conditions. If there are multiple solutions with the same number of substrings, return the one with minimum total length. It can be shown that there exists a unique solution of minimum total length.

Notice that you can return the substrings in **any** order.

**Example 1:**

**Input:**  $s = \text{"adefaddaccc"}$

**Output:**  $[\text{"e", "f", "ccc"}]$

**Explanation:** The following are all the possible substrings that meet the conditions:

[

  "adefaddaccc"

  "adefadda",

  "ef",

  "e",

  "f",

  "ccc",

]

If we choose the first string, we cannot choose anything else and we'd get only 1. If we choose "adefadda", we are left with "ccc" which is the only one that doesn't overlap, thus obtaining 2 substrings. Notice also, that it's not optimal to choose "ef" since it can be split into two. Therefore, the optimal way is to choose  $[\text{"e", "f", "ccc"}]$  which gives us 3 substrings. No other solution of the same number of substrings exist.

**Example 2:**

**Input:**  $s = \text{"abbaccd"}$

**Output:** ["d", "bb", "cc"]

**Explanation:** Notice that while the set of substrings ["d", "abba", "cc"] also has length 3, it's considered incorrect since it has larger total length.

### Constraints:

- $1 \leq s.length \leq 10^5$
- $s$  contains only lowercase English letters.

## 1521. Find a Value of a Mysterious Function Closest to Target

Hard

30115Add to ListShare

```
func(arr, l, r) {
    if (r < l) {
        return -1000000000
    }
    ans = arr[l]
    for (i = l + 1; i <= r; i++) {
        ans = ans & arr[i]
    }
    return ans
}
```

Winston was given the above mysterious function `func`. He has an integer array `arr` and an integer `target` and he wants to find the values `l` and `r` that make the value  $|func(arr, l, r) - target|$  minimum possible.

Return *the minimum possible value* of  $|func(arr, l, r) - target|$ .

Notice that `func` should be called with the values `l` and `r` where  $0 \leq l, r < arr.length$ .

### Example 1:

**Input:** arr = [9,12,3,7,15], target = 5

**Output:** 2

**Explanation:** Calling func with all the pairs of  $[l,r] = [[0,0],[1,1],[2,2],[3,3],[4,4],[0,1],[1,2],[2,3],[3,4],[0,2],[1,3],[2,4],[0,3],[1,4],[0,4]]$ , Winston got the following results  $[9,12,3,7,15,8,0,3,7,0,0,3,0,0,0]$ . The value closest to 5 is 7 and 3, thus the minimum difference is 2.

### Example 2:

**Input:** arr = [1000000,1000000,1000000], target = 1

**Output:** 999999

**Explanation:** Winston called the func with all possible values of  $[l,r]$  and he always got 1000000, thus the min difference is 999999.

### Example 3:

**Input:** arr = [1,2,4,8,16], target = 0

**Output:** 0

### Constraints:

- $1 \leq \text{arr.length} \leq 10^5$
- $1 \leq \text{arr}[i] \leq 10^6$
- $0 \leq \text{target} \leq 10^7$

## 1523. Count Odd Numbers in an Interval Range

Easy

105378Add to ListShare

Given two non-negative integers `low` and `high`. Return the *count of odd numbers between `low` and `high` (inclusive)*.

### Example 1:

**Input:** low = 3, high = 7

**Output:** 3

**Explanation:** The odd numbers between 3 and 7 are [3,5,7].

### Example 2:

**Input:** low = 8, high = 10

**Output:** 1

**Explanation:** The odd numbers between 8 and 10 are [9].

**Constraints:**

- $0 \leq \text{low} \leq \text{high} \leq 10^9$

## 1524. Number of Sub-arrays With Odd Sum

Medium

92344Add to ListShare

Given an array of integers `arr`, return *the number of subarrays with an **odd** sum*.

Since the answer can be very large, return it modulo  $10^9 + 7$ .

**Example 1:**

**Input:** `arr = [1,3,5]`

**Output:** 4

**Explanation:** All subarrays are `[[1],[1,3],[1,3,5],[3],[3,5],[5]]`

All sub-arrays sum are `[1,4,9,3,8,5]`.

Odd sums are `[1,9,3,5]` so the answer is 4.

**Example 2:**

**Input:** `arr = [2,4,6]`

**Output:** 0

**Explanation:** All subarrays are `[[2],[2,4],[2,4,6],[4],[4,6],[6]]`

All sub-arrays sum are `[2,6,12,4,10,6]`.

All sub-arrays have even sum and the answer is 0.

**Example 3:**

**Input:** `arr = [1,2,3,4,5,6,7]`

**Output:** 16

**Constraints:**

- $1 \leq \text{arr.length} \leq 10^5$
- $1 \leq \text{arr}[i] \leq 100$

## 1525. Number of Good Ways to Split a String

Medium

160637Add to ListShare

You are given a string  $s$ .

A split is called **good** if you can split  $s$  into two non-empty strings  $s_{\text{left}}$  and  $s_{\text{right}}$  where their concatenation is equal to  $s$  (i.e.,  $s_{\text{left}} + s_{\text{right}} = s$ ) and the number of distinct letters in  $s_{\text{left}}$  and  $s_{\text{right}}$  is the same.

Return the number of **good splits** you can make in  $s$ .

### Example 1:

**Input:**  $s = \text{"aacaba"}$

**Output:** 2

**Explanation:** There are 5 ways to split "aacaba" and 2 of them are good.

("a", "acaba") Left string and right string contains 1 and 3 different letters respectively.

("aa", "caba") Left string and right string contains 1 and 3 different letters respectively.

("aac", "aba") Left string and right string contains 2 and 2 different letters respectively (good split).

("aaca", "ba") Left string and right string contains 2 and 2 different letters respectively (good split).

("aacab", "a") Left string and right string contains 3 and 1 different letters respectively.

### Example 2:

**Input:**  $s = \text{"abcd"}$

**Output:** 1

**Explanation:** Split the string as follows ("ab", "cd").

### Constraints:

- $1 \leq s.length \leq 10^5$
- $s$  consists of only lowercase English letters.

## 1526. Minimum Number of Increments on Subarrays to Form a Target Array

Hard

107555Add to ListShare

You are given an integer array `target`. You have an integer array `initial` of the same size as `target` with all elements initially zeros.

In one operation you can choose **any** subarray from `initial` and increment each value by one.

Return *the minimum number of operations to form a target array from initial*.

The test cases are generated so that the answer fits in a 32-bit integer.

### Example 1:

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

**Output:** 3

**Explanation:** We need at least 3 operations to form the target array from the initial array.

[0,0,0,0,0] increment 1 from index 0 to 4 (inclusive).

[1,1,1,1,1] increment 1 from index 1 to 3 (inclusive).

[1,2,2,2,1] increment 1 at index 2.

[1,2,3,2,1] target array is formed.

### Example 2:

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

**Output:** 4

**Explanation:** [0,0,0,0] -> [1,1,1,1] -> [1,1,1,2] -> [2,1,1,2] -> [3,1,1,2]

### Example 3:

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

**Output:** 7

**Explanation:** [0,0,0,0,0] -> [1,1,1,1,1] -> [2,1,1,1,1] -> [3,1,1,1,1] -> [3,1,2,2,2] -> [3,1,3,3,2] -> [3,1,4,4,2] -> [3,1,5,4,2].

### Constraints:

- $1 \leq \text{target.length} \leq 10^5$
- $1 \leq \text{target}[i] \leq 10^5$

### 1527. Patients With a Condition

- Easy

- 215258Add to ListShare

- SQL Schema

- Table: Patients

Column Name	Type
patient_id	int
patient_name	varchar
conditions	varchar

- patient\_id is the primary key for this table.
- 'conditions' contains 0 or more code separated by spaces.
- This table contains information of the patients in the hospital.

- 

- Write an SQL query to report the patient\_id, patient\_name all conditions of patients who have Type I Diabetes. Type I Diabetes always starts with DIAB1 prefix

- Return the result table in **any order**.

- The query result format is in the following example.

- 

- **Example 1:**

- **Input:**

- Patients table:

patient_id	patient_name	conditions
1	Daniel	YFEV COUGH
2	Alice	
3	Bob	DIAB100 MYOP
4	George	ACNE DIAB100
5	Alain	DIAB201

- **Output:**

patient_id	patient_name	conditions
3	Bob	DIAB100 MYOP
4	George	ACNE DIAB100

- **Explanation:** Bob and George both have a condition that starts with DIAB1.

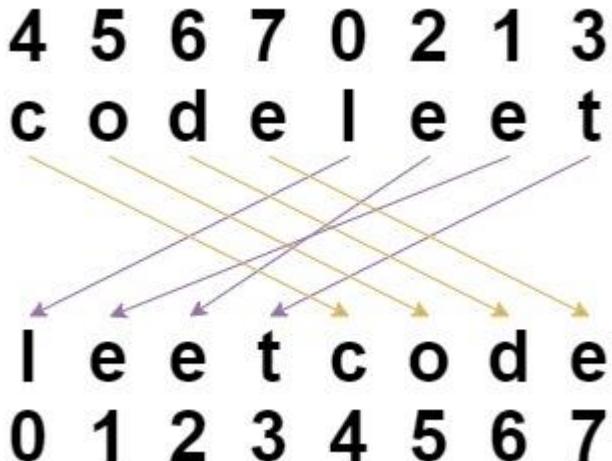
### 1528. Shuffle String

**Easy**

1793304Add to ListShare

You are given a string `s` and an integer array `indices` of the **same length**. The string `s` will be shuffled such that the character at the `ith` position moves to `indices[i]` in the shuffled string.

Return *the shuffled string*.

**Example 1:**

**Input:** `s = "codeleet", indices = [4,5,6,7,0,2,1,3]`

**Output:** "leetcode"

**Explanation:** As shown, "codeleet" becomes "leetcode" after shuffling.

**Example 2:**

**Input:** `s = "abc", indices = [0,1,2]`

**Output:** "abc"

**Explanation:** After shuffling, each character remains in its position.

**Constraints:**

- `s.length == indices.length == n`
- `1 <= n <= 100`
- `s` consists of only lowercase English letters.
- `0 <= indices[i] < n`
- All values of `indices` are **unique**.

## 1529. Minimum Suffix Flips

Medium

73033Add to ListShare

You are given a **0-indexed** binary string `target` of length `n`. You have another binary string `s` of length `n` that is initially set to all zeros. You want to make `s` equal to `target`.

In one operation, you can pick an index `i` where `0 <= i < n` and flip all bits in the **inclusive** range `[i, n - 1]`. Flip means changing '`0`' to '`1`' and '`1`' to '`0`'.

Return *the minimum number of operations needed to make `s` equal to `target`*.

### Example 1:

**Input:** `target = "10111"`

**Output:** 3

**Explanation:** Initially, `s = "00000"`.

Choose index `i = 2`: "00000" -> "00111"

Choose index `i = 0`: "00111" -> "11000"

Choose index `i = 1`: "11000" -> "10111"

We need at least 3 flip operations to form target.

### Example 2:

**Input:** `target = "101"`

**Output:** 3

**Explanation:** Initially, `s = "000"`.

Choose index `i = 0`: "000" -> "111"

Choose index `i = 1`: "111" -> "100"

Choose index `i = 2`: "100" -> "101"

We need at least 3 flip operations to form target.

### Example 3:

**Input:** `target = "00000"`

**Output:** 0

**Explanation:** We do not need any operations since the initial `s` already equals target.

**Constraints:**

- `n == target.length`
- $1 \leq n \leq 10^5$
- `target[i]` is either '0' or '1'.

## 1530. Number of Good Leaf Nodes Pairs

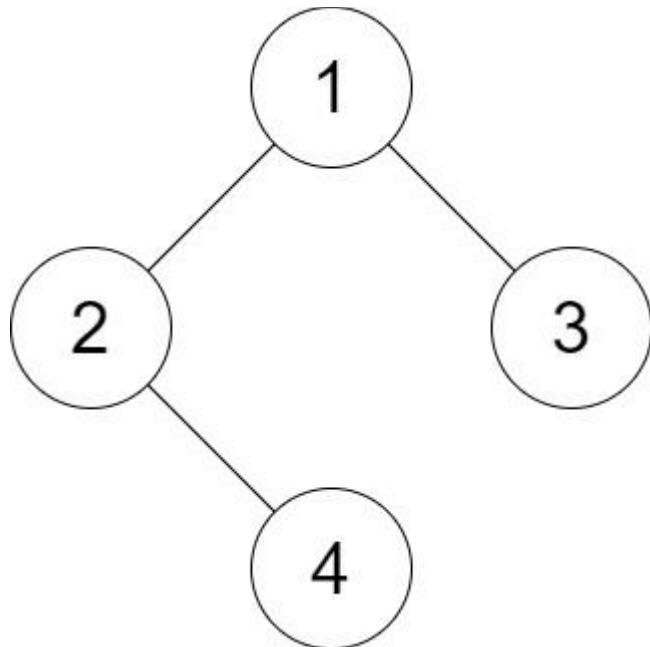
Medium

132732Add to ListShare

You are given the `root` of a binary tree and an integer `distance`. A pair of two different **leaf** nodes of a binary tree is said to be good if the length of **the shortest path** between them is less than or equal to `distance`.

Return *the number of good leaf node pairs* in the tree.

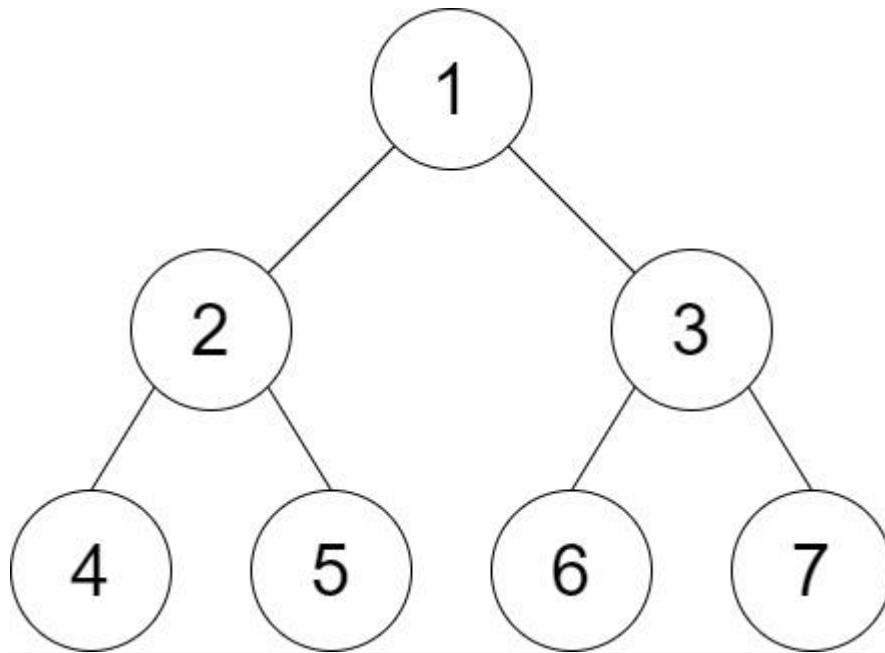
**Example 1:**



**Input:** `root = [1,2,3,null,4]`, `distance = 3`

**Output:** 1

**Explanation:** The leaf nodes of the tree are 3 and 4 and the length of the shortest path between them is 3. This is the only good pair.

**Example 2:**

**Input:** root = [1,2,3,4,5,6,7], distance = 3

**Output:** 2

**Explanation:** The good pairs are [4,5] and [6,7] with shortest path = 2. The pair [4,6] is not good because the length of the shortest path between them is 4.

**Example 3:**

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

**Output:** 1

**Explanation:** The only good pair is [2,5].

**Constraints:**

- The number of nodes in the tree is in the range [1,  $2^{10}$ ].
- $1 \leq \text{Node.val} \leq 100$
- $1 \leq \text{distance} \leq 10$

**1531. String Compression II**

Hard

51853Add to ListShare

Run-length encoding is a string compression method that works by replacing consecutive identical characters (repeated 2 or more times) with the concatenation of the character and the number

marking the count of the characters (length of the run). For example, to compress the string "aabccc" we replace "aa" by "a2" and replace "ccc" by "c3". Thus the compressed string becomes "a2bc3".

Notice that in this problem, we are not adding '1' after single characters.

Given a string `s` and an integer `k`. You need to delete **at most** `k` characters from `s` such that the run-length encoded version of `s` has minimum length.

Find the *minimum length of the run-length encoded version of s after deleting at most k characters*.

### Example 1:

**Input:** `s = "aabcccd", k = 2`

**Output:** 4

**Explanation:** Compressing `s` without deleting anything will give us "a3bc3d" of length 6. Deleting any of the characters 'a' or 'c' would at most decrease the length of the compressed string to 5, for instance delete 2 'a' then we will have `s = "abcccd"` which compressed is abc3d. Therefore, the optimal way is to delete 'b' and 'd', then the compressed version of `s` will be "a3c3" of length 4.

### Example 2:

**Input:** `s = "aabbaa", k = 2`

**Output:** 2

**Explanation:** If we delete both 'b' characters, the resulting compressed string would be "a4" of length 2.

### Example 3:

**Input:** `s = "aaaaaaaaaa", k = 0`

**Output:** 3

**Explanation:** Since `k` is zero, we cannot delete anything. The compressed string is "a11" of length 3.

### Constraints:

- `1 <= s.length <= 100`
- `0 <= k <= s.length`
- `s` contains only lowercase English letters.

## 1534. Count Good Triplets

Easy

501984Add to ListShare

Given an array of integers `arr`, and three integers `a`, `b` and `c`. You need to find the number of good triplets.

A triplet `(arr[i], arr[j], arr[k])` is **good** if the following conditions are true:

- $0 \leq i < j < k < arr.length$
- $|arr[i] - arr[j]| \leq a$
- $|arr[j] - arr[k]| \leq b$
- $|arr[i] - arr[k]| \leq c$

Where  $|x|$  denotes the absolute value of `x`.

Return *the number of good triplets*.

### Example 1:

**Input:** `arr = [3,0,1,1,9,7]`, `a = 7`, `b = 2`, `c = 3`

**Output:** 4

**Explanation:** There are 4 good triplets: `[(3,0,1), (3,0,1), (3,1,1), (0,1,1)]`.

### Example 2:

**Input:** `arr = [1,1,2,2,3]`, `a = 0`, `b = 0`, `c = 1`

**Output:** 0

**Explanation:** No triplet satisfies all conditions.

### Constraints:

- $3 \leq arr.length \leq 100$
- $0 \leq arr[i] \leq 1000$
- $0 \leq a, b, c \leq 1000$

## 1535. Find the Winner of an Array Game

Medium

50326Add to ListShare

Given an integer array `arr` of **distinct** integers and an integer `k`.

A game will be played between the first two elements of the array (i.e. `arr[0]` and `arr[1]`). In each round of the game, we compare `arr[0]` with `arr[1]`, the larger integer wins and remains at position 0, and the smaller integer moves to the end of the array. The game ends when an integer wins `k` consecutive rounds.

Return *the integer which will win the game*.

It is **guaranteed** that there will be a winner of the game.

### Example 1:

**Input:** `arr = [2,1,3,5,4,6,7], k = 2`

**Output:** 5

**Explanation:** Let's see the rounds of the game:

Round	arr	winner	win_count
1	[2,1,3,5,4,6,7]	2	1
2	[2,3,5,4,6,7,1]	3	1
3	[3,5,4,6,7,1,2]	5	1
4	[5,4,6,7,1,2,3]	5	2

So we can see that 4 rounds will be played and 5 is the winner because it wins 2 consecutive games.

### Example 2:

**Input:** `arr = [3,2,1], k = 10`

**Output:** 3

**Explanation:** 3 will win the first 10 rounds consecutively.

### Constraints:

- $2 \leq \text{arr.length} \leq 10^5$
- $1 \leq \text{arr}[i] \leq 10^6$
- `arr` contains **distinct** integers.
- $1 \leq k \leq 10^9$

## 1536. Minimum Swaps to Arrange a Binary Grid

## Medium

47561Add to ListShare

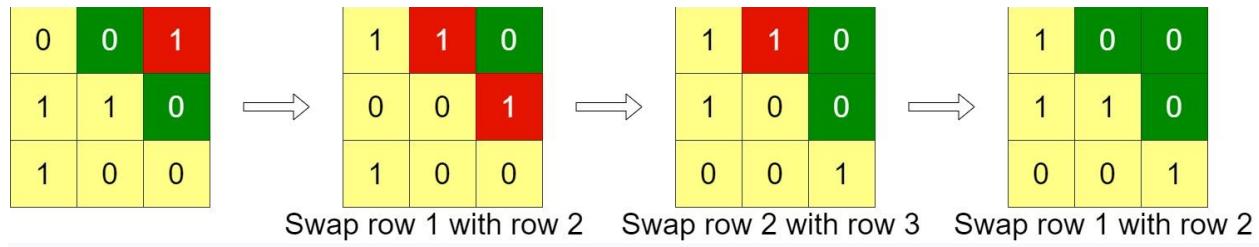
Given an  $n \times n$  binary grid, in one step you can choose two **adjacent rows** of the grid and swap them.

A grid is said to be **valid** if all the cells above the main diagonal are **zeros**.

Return *the minimum number of steps* needed to make the grid valid, or **-1** if the grid cannot be valid.

The main diagonal of a grid is the diagonal that starts at cell  $(1, 1)$  and ends at cell  $(n, n)$ .

## Example 1:



Input: grid = [[0,0,1],[1,1,0],[1,0,0]]

Output: 3

## Example 2:

0	1	1	0
0	1	1	0
0	1	1	0
0	1	1	0

Input: grid = [[0,1,1,0],[0,1,1,0],[0,1,1,0],[0,1,1,0]]

Output: -1

**Explanation:** All rows are similar, swaps have no effect on the grid.

**Example 3:**

1	0	0
1	1	0
1	1	1

**Input:** grid = [[1,0,0],[1,1,0],[1,1,1]]

**Output:** 0

**Constraints:**

- `n == grid.length == grid[i].length`
- `1 <= n <= 200`
- `grid[i][j]` is either 0 or 1

## 1537. Get the Maximum Score

Hard

70238Add to ListShare

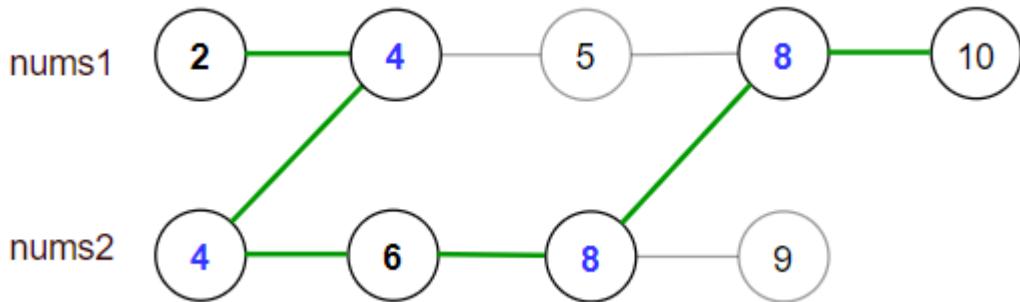
You are given two **sorted** arrays of distinct integers `nums1` and `nums2`.

A **valid path** is defined as follows:

- Choose array `nums1` or `nums2` to traverse (from index-0).
- Traverse the current array from left to right.
- If you are reading any value that is present in `nums1` and `nums2` you are allowed to change your path to the other array. (Only one repeated value is considered in the valid path).

The **score** is defined as the sum of unique values in a valid path.

Return *the maximum score you can obtain of all possible valid paths*. Since the answer may be too large, return it modulo  $10^9 + 7$ .

**Example 1:**

**Input:** nums1 = [2,4,5,8,10], nums2 = [4,6,8,9]

**Output:** 30

**Explanation:** Valid paths:

[2,4,5,8,10], [2,4,5,8,9], [2,4,6,8,9], [2,4,6,8,10], (starting from nums1)

[4,6,8,9], [4,5,8,10], [4,5,8,9], [4,6,8,10] (starting from nums2)

The maximum is obtained with the path in green [2,4,6,8,10].

**Example 2:**

**Input:** nums1 = [1,3,5,7,9], nums2 = [3,5,100]

**Output:** 109

**Explanation:** Maximum sum is obtained with the path [1,3,5,100].

**Example 3:**

**Input:** nums1 = [1,2,3,4,5], nums2 = [6,7,8,9,10]

**Output:** 40

**Explanation:** There are no common elements between nums1 and nums2.

Maximum sum is obtained with the path [6,7,8,9,10].

**Constraints:**

- $1 \leq \text{nums1.length}, \text{nums2.length} \leq 10^5$
- $1 \leq \text{nums1}[i], \text{nums2}[i] \leq 10^7$
- $\text{nums1}$  and  $\text{nums2}$  are strictly increasing.

**1539. Kth Missing Positive Number**

**Easy**

3136224Add to ListShare

Given an array `arr` of positive integers sorted in a **strictly increasing order**, and an integer `k`.Return *the  $k^{\text{th}}$  positive integer that is missing* from this array.**Example 1:****Input:** arr = [2,3,4,7,11], k = 5**Output:** 9**Explanation:** The missing positive integers are [1,5,6,8,9,10,12,13,...]. The  $5^{\text{th}}$  missing positive integer is 9.**Example 2:****Input:** arr = [1,2,3,4], k = 2**Output:** 6**Explanation:** The missing positive integers are [5,6,7,...]. The  $2^{\text{nd}}$  missing positive integer is 6.**Constraints:**

- $1 \leq \text{arr.length} \leq 1000$
- $1 \leq \text{arr}[i] \leq 1000$
- $1 \leq k \leq 1000$
- $\text{arr}[i] < \text{arr}[j]$  for  $1 \leq i < j \leq \text{arr.length}$

**1540. Can Convert String in K Moves****Medium**

296245Add to ListShare

Given two strings `s` and `t`, your goal is to convert `s` into `t` in `k` moves or less.During the  $i^{\text{th}}$  ( $1 \leq i \leq k$ ) move you can:

- Choose any index `j` (1-indexed) from `s`, such that  $1 \leq j \leq s.length$  and `j` has not been chosen in any previous move, and shift the character at that index `i` times.
- Do nothing.

Shifting a character means replacing it by the next letter in the alphabet (wrapping around so that 'z' becomes 'a'). Shifting a character by  $i$  means applying the shift operations  $i$  times.

Remember that any index  $j$  can be picked at most once.

Return `true` if it's possible to convert  $s$  into  $t$  in no more than  $k$  moves, otherwise return `false`.

### Example 1:

**Input:**  $s = \text{"input"}$ ,  $t = \text{"ouput"}$ ,  $k = 9$

**Output:** `true`

**Explanation:** In the 6th move, we shift 'i' 6 times to get 'o'. And in the 7th move we shift 'n' to get 'u'.

### Example 2:

**Input:**  $s = \text{"abc"}$ ,  $t = \text{"bcd"}$ ,  $k = 10$

**Output:** `false`

**Explanation:** We need to shift each character in  $s$  one time to convert it into  $t$ . We can shift 'a' to 'b' during the 1st move. However, there is no way to shift the other characters in the remaining moves to obtain  $t$  from  $s$ .

### Example 3:

**Input:**  $s = \text{"aab"}$ ,  $t = \text{"bbb"}$ ,  $k = 27$

**Output:** `true`

**Explanation:** In the 1st move, we shift the first 'a' 1 time to get 'b'. In the 27th move, we shift the second 'a' 27 times to get 'b'.

### Constraints:

- $1 \leq s.length, t.length \leq 10^5$
- $0 \leq k \leq 10^9$
- $s, t$  contain only lowercase English letters.

## 1541. Minimum Insertions to Balance a Parentheses String

Medium

756162Add to ListShare

Given a parentheses string  $s$  containing only the characters '(' and ')'. A parentheses string is **balanced** if:

- Any left parenthesis '`(`' must have a corresponding two consecutive right parenthesis '`) )`'.
- Left parenthesis '`(`' must go before the corresponding two consecutive right parenthesis '`) )`'.

In other words, we treat '`(`' as an opening parenthesis and '`) )`' as a closing parenthesis.

- For example, "`( )`", "`( ) ( ( ) ) )`" and "`( ( ) ( ) ) )`" are balanced, "`) ( )`", "`( ) )`" and "`( ( ) )`" are not balanced.

You can insert the characters '`(`' and '`)`' at any position of the string to balance it if needed.

Return *the minimum number of insertions* needed to make `s` balanced.

### Example 1:

**Input:** `s = "((()))"`

**Output:** 1

**Explanation:** The second '`(`' has two matching '`)`'s, but the first '`(`' has only '`)`' matching. We need to add one more '`)`' at the end of the string to be "`((()))`" which is balanced.

### Example 2:

**Input:** `s = "())"`

**Output:** 0

**Explanation:** The string is already balanced.

### Example 3:

**Input:** `s = "))())()`

**Output:** 3

**Explanation:** Add '`(`' to match the first '`)`', Add '`)`' to match the last '`(`'.

### Constraints:

- `1 <= s.length <= 105`
- `s` consists of '`(`' and '`)`' only.

## 1542. Find Longest Awesome Substring

**Hard**

59911Add to ListShare

You are given a string `s`. An **awesome** substring is a non-empty substring of `s` such that we can make any number of swaps in order to make it a palindrome.

Return *the length of the maximum length awesome substring* of `s`.

**Example 1:**

**Input:** `s = "3242415"`

**Output:** 5

**Explanation:** "24241" is the longest awesome substring, we can form the palindrome "24142" with some swaps.

**Example 2:**

**Input:** `s = "12345678"`

**Output:** 1

**Example 3:**

**Input:** `s = "213123"`

**Output:** 6

**Explanation:** "213123" is the longest awesome substring, we can form the palindrome "231132" with some swaps.

**Constraints:**

- `1 <= s.length <= 105`
- `s` consists only of digits.

**1544. Make The String Great****Easy**

82954Add to ListShare

Given a string `s` of lower and upper case English letters.

A good string is a string which doesn't have **two adjacent characters** `s[i]` and `s[i + 1]` where:

- `0 <= i <= s.length - 2`
- `s[i]` is a lower-case letter and `s[i + 1]` is the same letter but in upper-case or **vice-versa**.

To make the string good, you can choose **two adjacent** characters that make the string bad and remove them. You can keep doing this until the string becomes good.

Return *the string* after making it good. The answer is guaranteed to be unique under the given constraints.

**Notice** that an empty string is also good.

### Example 1:

**Input:** s = "leEeetcode"

**Output:** "leetcode"

**Explanation:** In the first step, either you choose  $i = 1$  or  $i = 2$ , both will result "leEeetcode" to be reduced to "leetcode".

### Example 2:

**Input:** s = "abBAcc"

**Output:** ""

**Explanation:** We have many possible scenarios, and all lead to the same answer. For example:

"abBAcc"  $\rightarrow$  "aAcc"  $\rightarrow$  "cC"  $\rightarrow$  ""

"abBAcc"  $\rightarrow$  "abBA"  $\rightarrow$  "aA"  $\rightarrow$  ""

### Example 3:

**Input:** s = "s"

**Output:** "s"

### Constraints:

- $1 \leq s.length \leq 100$
- s contains only lower and upper case English letters.

## 1545. Find Kth Bit in Nth Binary String

Medium

60843Add to ListShare

Given two positive integers  $n$  and  $k$ , the binary string  $s_n$  is formed as follows:

- $S_1 = "0"$
- $S_i = S_{i-1} + "1" + \text{reverse}(\text{invert}(S_{i-1}))$  for  $i > 1$

Where `+` denotes the concatenation operation, `reverse(x)` returns the reversed string `x`, and `invert(x)` inverts all the bits in `x` (`0` changes to `1` and `1` changes to `0`).

For example, the first four strings in the above sequence are:

- $S_1 = "0"$
- $S_2 = "011"$
- $S_3 = "0111001"$
- $S_4 = "011100110110001"$

Return the  $k^{\text{th}}$  bit in  $S_n$ . It is guaranteed that  $k$  is valid for the given  $n$ .

### Example 1:

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

**Output:** "0"

**Explanation:**  $S_3$  is "0111001".

The 1<sup>st</sup> bit is "0".

### Example 2:

**Input:**  $n = 4, k = 11$

**Output:** "1"

**Explanation:**  $S_4$  is "011100110110001".

The 11<sup>th</sup> bit is "1".

### Constraints:

- $1 \leq n \leq 20$
- $1 \leq k \leq 2^n - 1$

## 1546. Maximum Number of Non-Overlapping Subarrays With Sum Equals Target

Medium

85421Add to ListShare

Given an array `nums` and an integer `target`, return the maximum number of **non-empty non-overlapping** subarrays such that the sum of values in each subarray is equal to `target`.

**Example 1:****Input:** nums = [1,1,1,1,1], target = 2**Output:** 2**Explanation:** There are 2 non-overlapping subarrays [1,1,1,1,1] with sum equals to target(2).**Example 2:****Input:** nums = [-1,3,5,1,4,2,-9], target = 6**Output:** 2**Explanation:** There are 3 subarrays with sum equal to 6.

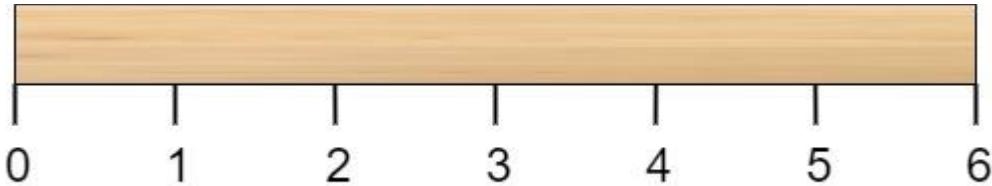
([5,1], [4,2], [3,5,1,4,2,-9]) but only the first 2 are non-overlapping.

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^5$
- $-10^4 \leq \text{nums}[i] \leq 10^4$
- $0 \leq \text{target} \leq 10^6$

**1547. Minimum Cost to Cut a Stick****Hard**

204637Add to ListShare

Given a wooden stick of length  $n$  units. The stick is labelled from 0 to  $n$ . For example, a stick of length 6 is labelled as follows:Given an integer array  $\text{cuts}$  where  $\text{cuts}[i]$  denotes a position you should perform a cut at.

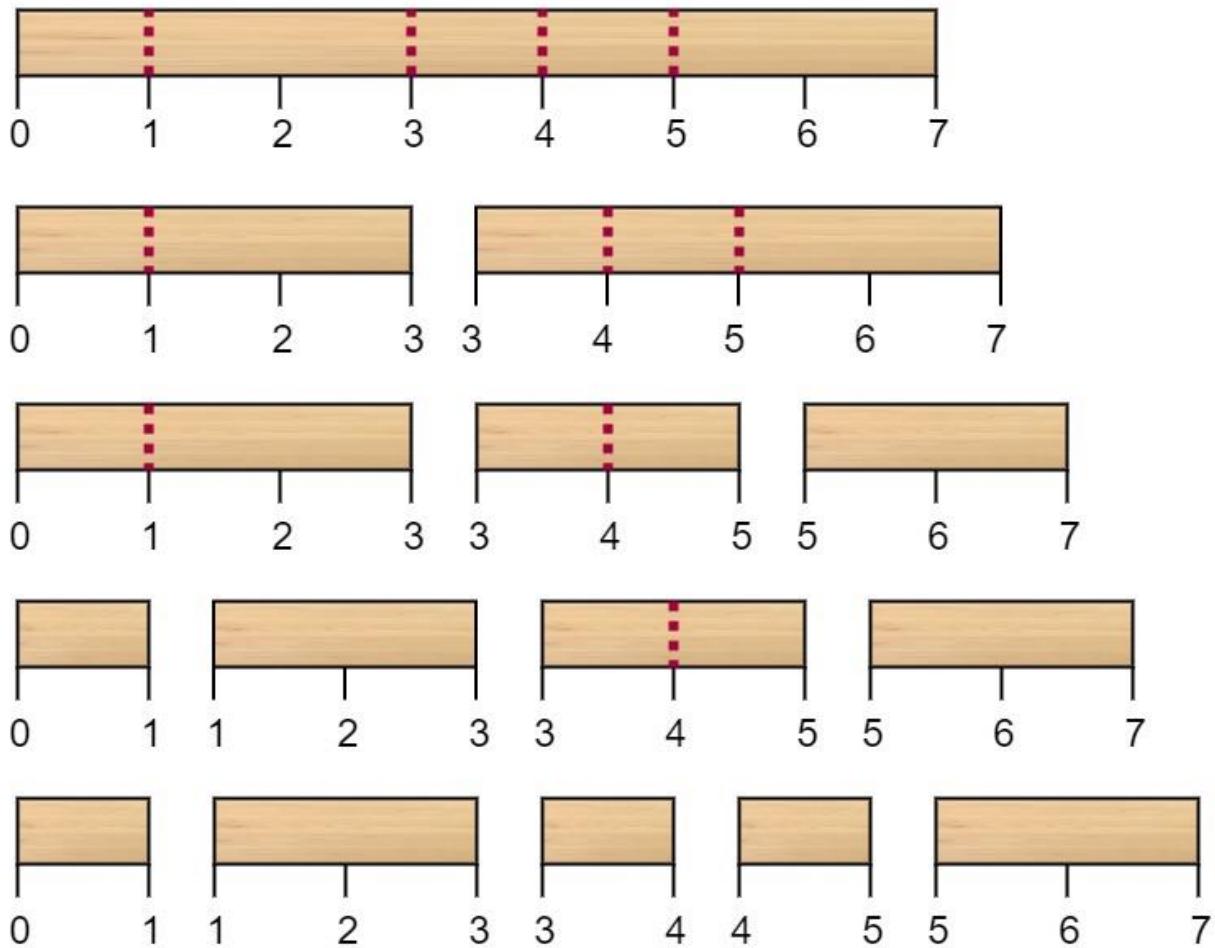
You should perform the cuts in order, you can change the order of the cuts as you wish.

The cost of one cut is the length of the stick to be cut, the total cost is the sum of costs of all cuts. When you cut a stick, it will be split into two smaller sticks (i.e. the sum of their lengths is the length of the stick before the cut). Please refer to the first example for a better explanation.

Return *the minimum total cost* of the cuts.

**Example 1:**

`cuts = [3, 5, 1, 4] (Optimal Ordering)`

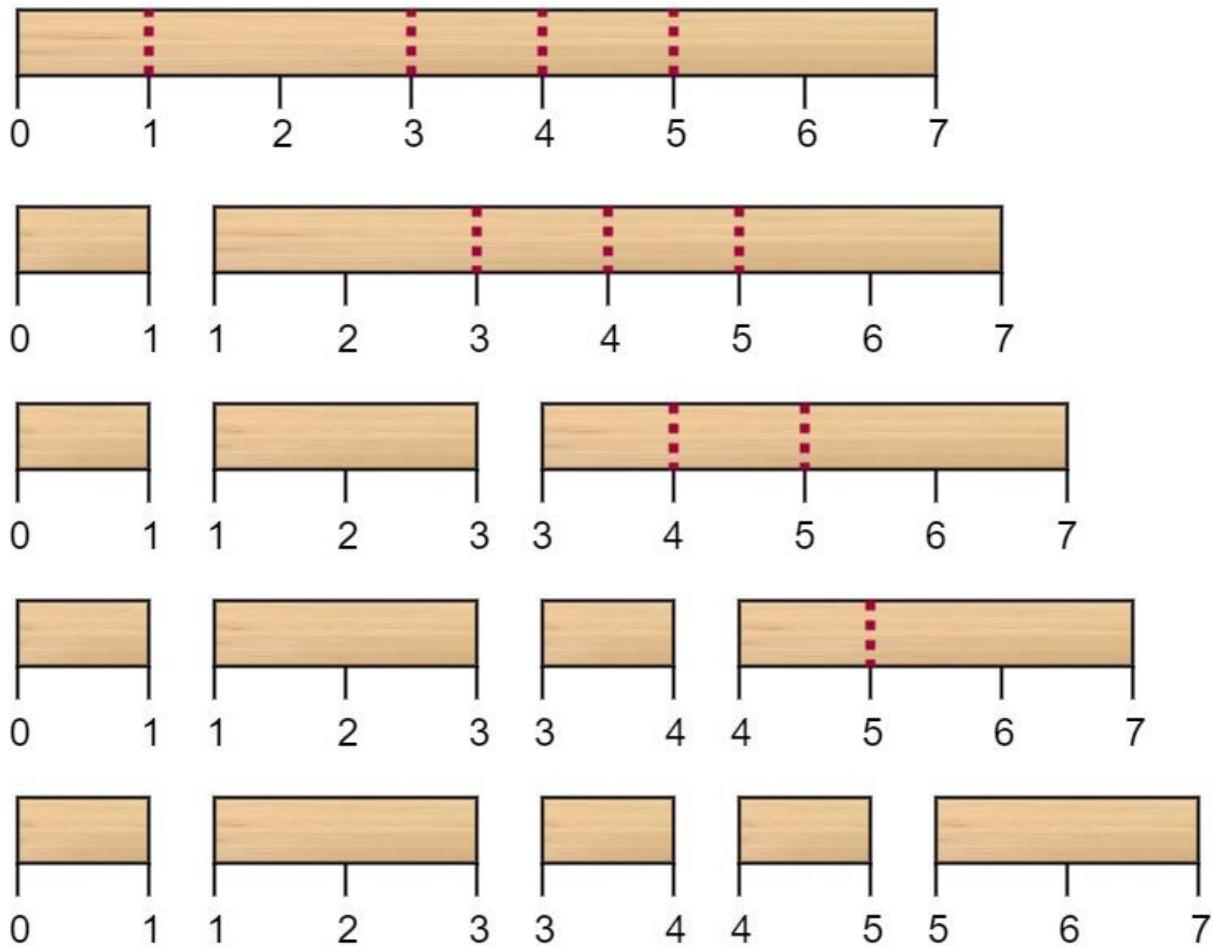


**Input:**  $n = 7$ ,  $\text{cuts} = [1, 3, 4, 5]$

**Output:** 16

**Explanation:** Using cuts order = [1, 3, 4, 5] as in the input leads to the following scenario:

$\text{cuts} = [1, 3, 4, 5]$



The first cut is done to a rod of length 7 so the cost is 7. The second cut is done to a rod of length 6 (i.e. the second part of the first cut), the third is done to a rod of length 4 and the last cut is to a rod of length 3. The total cost is  $7 + 6 + 4 + 3 = 20$ .

Rearranging the cuts to be  $[3, 5, 1, 4]$  for example will lead to a scenario with total cost = 16 (as shown in the example photo  $7 + 4 + 3 + 2 = 16$ ).

### Example 2:

**Input:**  $n = 9$ ,  $\text{cuts} = [5, 6, 1, 4, 2]$

**Output:** 22

**Explanation:** If you try the given cuts ordering the cost will be 25.

There are much ordering with total cost  $\leq 25$ , for example, the order  $[4, 6, 5, 2, 1]$  has total cost = 22 which is the minimum possible.

**Constraints:**

- $2 \leq n \leq 10^6$
- $1 \leq \text{cuts.length} \leq \min(n - 1, 100)$
- $1 \leq \text{cuts}[i] \leq n - 1$
- All the integers in `cuts` array are **distinct**.

**1550. Three Consecutive Odds****Easy**

43049Add to ListShare

Given an integer array `arr`, return `true` if there are three consecutive odd numbers in the array. Otherwise, return `false`.

**Example 1:**

**Input:** `arr = [2,6,4,1]`

**Output:** `false`

**Explanation:** There are no three consecutive odds.

**Example 2:**

**Input:** `arr = [1,2,34,3,4,5,7,23,12]`

**Output:** `true`

**Explanation:** `[5,7,23]` are three consecutive odds.

**Constraints:**

- $1 \leq \text{arr.length} \leq 1000$
- $1 \leq \text{arr}[i] \leq 1000$

**1551. Minimum Operations to Make Array Equal****Medium**

987149Add to ListShare

You have an array `arr` of length `n` where `arr[i] = (2 * i) + 1` for all valid values of `i` (i.e.,  $0 \leq i < n$ ).

In one operation, you can select two indices `x` and `y` where  $0 \leq x, y < n$  and subtract 1 from `arr[x]` and add 1 to `arr[y]` (i.e., perform `arr[x] -= 1` and `arr[y] += 1`). The goal is to make all the elements of the array **equal**. It is **guaranteed** that all the elements of the array can be made equal using some operations.

Given an integer `n`, the length of the array, return *the minimum number of operations* needed to make all the elements of `arr` equal.

**Example 1:**

**Input:** `n = 3`

**Output:** `2`

**Explanation:** `arr = [1, 3, 5]`

First operation choose  $x = 2$  and  $y = 0$ , this leads `arr` to be `[2, 3, 4]`

In the second operation choose  $x = 2$  and  $y = 0$  again, thus `arr = [3, 3, 3]`.

**Example 2:**

**Input:** `n = 6`

**Output:** `9`

**Constraints:**

- $1 \leq n \leq 10^4$

## 1552. Magnetic Force Between Two Balls

Medium

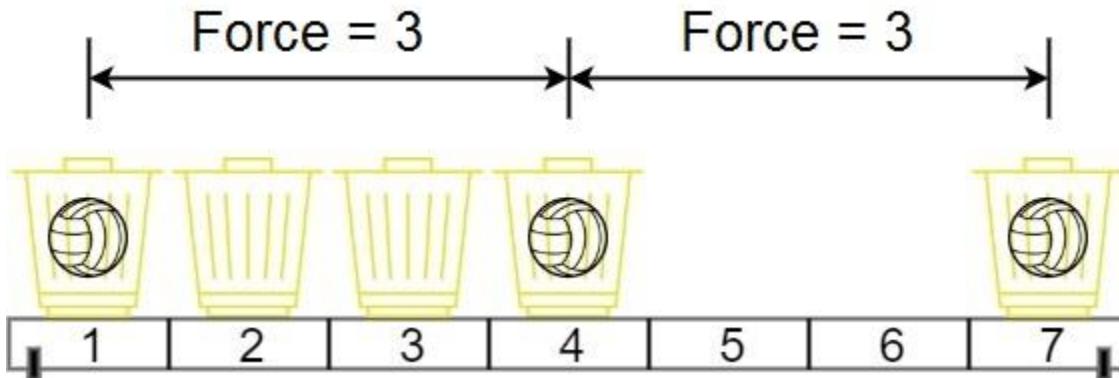
143788Add to ListShare

In the universe Earth C-137, Rick discovered a special form of magnetic force between two balls if they are put in his new invented basket. Rick has `n` empty baskets, the  $i^{th}$  basket is at `position[i]`, Morty has `m` balls and needs to distribute the balls into the baskets such that the **minimum magnetic force** between any two balls is **maximum**.

Rick stated that magnetic force between two different balls at positions `x` and `y` is  $|x - y|$ .

Given the integer array `position` and the integer `m`. Return *the required force*.

**Example 1:**



**Input:** position = [1,2,3,4,7], m = 3

**Output:** 3

**Explanation:** Distributing the 3 balls into baskets 1, 4 and 7 will make the magnetic force between ball pairs [3, 3, 6]. The minimum magnetic force is 3. We cannot achieve a larger minimum magnetic force than 3.

**Example 2:**

**Input:** position = [5,4,3,2,1,1000000000], m = 2

**Output:** 999999999

**Explanation:** We can use baskets 1 and 1000000000.

**Constraints:**

- `n == position.length`
- `2 <= n <= 105`
- `1 <= position[i] <= 109`
- All integers in `position` are **distinct**.
- `2 <= m <= position.length`

## 1553. Minimum Number of Days to Eat N Oranges

Hard

77547 Add to List Share

There are `n` oranges in the kitchen and you decided to eat some of these oranges every day as follows:

- Eat one orange.
- If the number of remaining oranges `n` is divisible by 2 then you can eat `n / 2` oranges.
- If the number of remaining oranges `n` is divisible by 3 then you can eat `2 * (n / 3)` oranges.

You can only choose one of the actions per day.

Given the integer `n`, return *the minimum number of days to eat `n` oranges*.

**Example 1:**

**Input:** `n = 10`

**Output:** 4

**Explanation:** You have 10 oranges.

Day 1: Eat 1 orange,  $10 - 1 = 9$ .

Day 2: Eat 6 oranges,  $9 - 2*(9/3) = 9 - 6 = 3$ . (Since 9 is divisible by 3)

Day 3: Eat 2 oranges,  $3 - 2*(3/3) = 3 - 2 = 1$ .

Day 4: Eat the last orange  $1 - 1 = 0$ .

You need at least 4 days to eat the 10 oranges.

**Example 2:**

**Input:** `n = 6`

**Output:** 3

**Explanation:** You have 6 oranges.

Day 1: Eat 3 oranges,  $6 - 6/2 = 6 - 3 = 3$ . (Since 6 is divisible by 2).

Day 2: Eat 2 oranges,  $3 - 2*(3/3) = 3 - 2 = 1$ . (Since 3 is divisible by 3)

Day 3: Eat the last orange  $1 - 1 = 0$ .

You need at least 3 days to eat the 6 oranges.

**Constraints:**

- $1 \leq n \leq 2 * 10^9$

## 1556. Thousand Separator

**Easy**

3616Add to ListShare

Given an integer `n`, add a dot (".") as the thousands separator and return it in string format.

**Example 1:****Input:** n = 987**Output:** "987"**Example 2:****Input:** n = 1234**Output:** "1.234"**Constraints:**

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

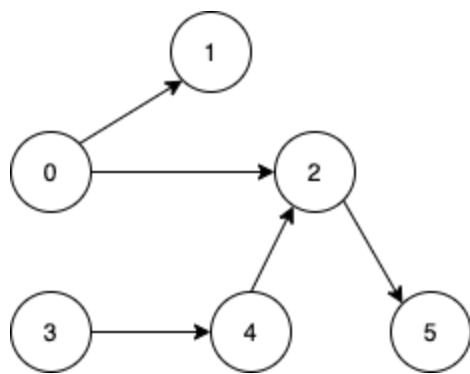
**1557. Minimum Number of Vertices to Reach All Nodes****Medium**

184366Add to ListShare

Given a **directed acyclic graph**, with  $n$  vertices numbered from  $0$  to  $n-1$ , and an array  $\text{edges}$  where  $\text{edges}[i] = [\text{from}_i, \text{to}_i]$  represents a directed edge from node  $\text{from}_i$  to node  $\text{to}_i$ .

Find the *smallest set of vertices from which all nodes in the graph are reachable*. It's guaranteed that a unique solution exists.

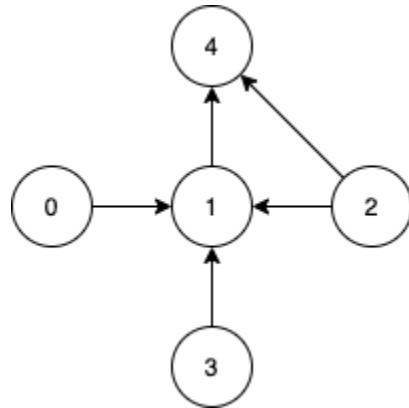
Notice that you can return the vertices in any order.

**Example 1:****Input:** n = 6, edges = [[0,1],[0,2],[2,5],[3,4],[4,2]]

**Output:** [0,3]

**Explanation:** It's not possible to reach all the nodes from a single vertex. From 0 we can reach [0,1,2,5]. From 3 we can reach [3,4,2,5]. So we output [0,3].

**Example 2:**



**Input:** n = 5, edges = [[0,1],[2,1],[3,1],[1,4],[2,4]]

**Output:** [0,2,3]

**Explanation:** Notice that vertices 0, 3 and 2 are not reachable from any other node, so we must include them. Also any of these vertices can reach nodes 1 and 4.

**Constraints:**

- $2 \leq n \leq 10^5$
- $1 \leq \text{edges.length} \leq \min(10^5, n * (n - 1) / 2)$
- $\text{edges}[i].length == 2$
- $0 \leq \text{from}_i, \text{to}_i < n$
- All pairs  $(\text{from}_i, \text{to}_i)$  are distinct.

## 1558. Minimum Numbers of Function Calls to Make Target Array

Medium

48125Add to ListShare

You are given an integer array `nums`. You have an integer array `arr` of the same length with all values set to 0 initially. You also have the following `modify` function:

```

func modify(arr, op, idx){
    //add by 1 index idx
    if (op == 0) {
        arr[idx] = arr[idx] + 1
    }
    //multiply by 2 all elements
    if (op == 1) {
        for(i = 0; i < arr.length; i++) {
            arr[i] = arr[i] * 2
        }
    }
}

```

You want to use the modify function to convert `arr` to `nums` using the minimum number of calls.

Return *the minimum number of function calls to make* `nums` *from* `arr`.

The test cases are generated so that the answer fits in a **32-bit** signed integer.

### Example 1:

**Input:** `nums` = [1,5]

**Output:** 5

**Explanation:** Increment by 1 (second element): [0, 0] to get [0, 1] (1 operation).

Double all the elements: [0, 1] -> [0, 2] -> [0, 4] (2 operations).

Increment by 1 (both elements) [0, 4] -> [1, 4] -> [1, 5] (2 operations).

Total of operations:  $1 + 2 + 2 = 5$ .

### Example 2:

**Input:** `nums` = [2,2]

**Output:** 3

**Explanation:** Increment by 1 (both elements) [0, 0] -> [0, 1] -> [1, 1] (2 operations).

Double all the elements: [1, 1] -> [2, 2] (1 operation).

Total of operations:  $2 + 1 = 3$ .

**Example 3:**

**Input:** `nums = [4,2,5]`

**Output:** 6

**Explanation:** (initial)`[0,0,0]` -> `[1,0,0]` -> `[1,0,1]` -> `[2,0,2]` -> `[2,1,2]` -> `[4,2,4]` -> `[4,2,5]`(`nums`).

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^5$
- $0 \leq \text{nums}[i] \leq 10^9$

**1559. Detect Cycles in 2D Grid**

Medium

77122Add to ListShare

Given a 2D array of characters `grid` of size `m x n`, you need to find if there exists any cycle consisting of the **same value** in `grid`.

A cycle is a path of **length 4 or more** in the grid that starts and ends at the same cell. From a given cell, you can move to one of the cells adjacent to it - in one of the four directions (up, down, left, or right), if it has the **same value** of the current cell.

Also, you cannot move to the cell that you visited in your last move. For example, the cycle `(1, 1)` -> `(1, 2)` -> `(1, 1)` is invalid because from `(1, 2)` we visited `(1, 1)` which was the last visited cell.

Return `true` if any cycle of the same value exists in `grid`, otherwise, return `false`.

**Example 1:**

a	a	a	a
a	b	b	a
a	b	b	a
a	a	a	a

**Input:** `grid =`

`[["a","a","a","a"], ["a","b","b","a"], ["a","b","b","a"], ["a","a","a","a"]]`

**Output:** true

**Explanation:** There are two valid cycles shown in different colors in the image below:

a	a	a	a
a	b	b	a
a	b	b	a
a	a	a	a

**Example 2:**

c	c	c	a
c	d	c	c
c	c	e	c
f	c	c	c

**Input:** grid =  
`[[ "c", "c", "c", "a" ], [ "c", "d", "c", "c" ], [ "c", "c", "e", "c" ], [ "f", "c", "c", "c" ]]`

**Output:** true

**Explanation:** There is only one valid cycle highlighted in the image below:

c	c	c	a
c	d	c	c
c	c	e	c
f	c	c	c

**Example 3:**

a	b	b
b	z	b
b	b	a

**Input:** grid = [["a", "b", "b"], ["b", "z", "b"], ["b", "b", "a"]]

**Output:** false

**Constraints:**

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 500`
- `grid` consists only of lowercase English letters.

## 1560. Most Visited Sector in a Circular Track

Easy

233481Add to ListShare

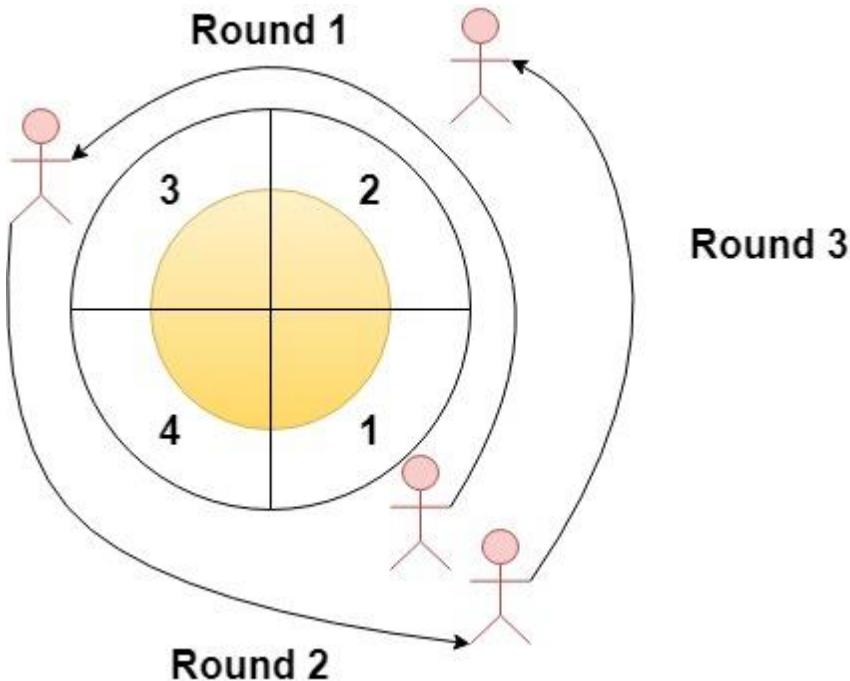
Given an integer `n` and an integer array `rounds`. We have a circular track which consists of `n` sectors labeled from `1` to `n`. A marathon will be held on this track, the marathon consists of `m` rounds.

The `ith` round starts at sector `rounds[i - 1]` and ends at sector `rounds[i]`. For example, round `1` starts at sector `rounds[0]` and ends at sector `rounds[1]`

Return *an array of the most visited sectors* sorted in **ascending** order.

Notice that you circulate the track in ascending order of sector numbers in the counter-clockwise direction (See the first example).

**Example 1:**



**Input:**  $n = 4$ ,  $\text{rounds} = [1, 3, 1, 2]$

**Output:** [1,2]

**Explanation:** The marathon starts at sector 1. The order of the visited sectors is as follows:

1  $\rightarrow$  2  $\rightarrow$  3 (end of round 1)  $\rightarrow$  4  $\rightarrow$  1 (end of round 2)  $\rightarrow$  2 (end of round 3 and the marathon)

We can see that both sectors 1 and 2 are visited twice and they are the most visited sectors. Sectors 3 and 4 are visited only once.

**Example 2:**

**Input:**  $n = 2$ ,  $\text{rounds} = [2, 1, 2, 1, 2, 1, 2, 1, 2]$

**Output:** [2]

**Example 3:**

**Input:**  $n = 7$ ,  $\text{rounds} = [1, 3, 5, 7]$

**Output:** [1,2,3,4,5,6,7]

**Constraints:**

- $2 \leq n \leq 100$
- $1 \leq m \leq 100$

- `rounds.length == m + 1`
- `1 <= rounds[i] <= n`
- `rounds[i] != rounds[i + 1]` for  $0 \leq i < m$

## 1561. Maximum Number of Coins You Can Get

Medium

812104Add to ListShare

There are  $3n$  piles of coins of varying size, you and your friends will take piles of coins as follows:

- In each step, you will choose **any 3** piles of coins (not necessarily consecutive).
- Of your choice, Alice will pick the pile with the maximum number of coins.
- You will pick the next pile with the maximum number of coins.
- Your friend Bob will pick the last pile.
- Repeat until there are no more piles of coins.

Given an array of integers `piles` where `piles[i]` is the number of coins in the  $i^{\text{th}}$  pile.

Return the maximum number of coins that you can have.

### Example 1:

**Input:** `piles = [2,4,1,2,7,8]`

**Output:** 9

**Explanation:** Choose the triplet (2, 7, 8), Alice Pick the pile with 8 coins, you the pile with 7 coins and Bob the last one.

Choose the triplet (1, 2, 4), Alice Pick the pile with 4 coins, you the pile with 2 coins and Bob the last one.

The maximum number of coins which you can have are:  $7 + 2 = 9$ .

On the other hand if we choose this arrangement (1, 2, 8), (2, 4, 7) you only get  $2 + 4 = 6$  coins which is not optimal.

### Example 2:

**Input:** `piles = [2,4,5]`

**Output:** 4

### Example 3:

**Input:** `piles = [9,8,7,6,5,1,2,3,4]`

**Output:** 18

**Constraints:**

- $3 \leq \text{piles.length} \leq 10^5$
- $\text{piles.length} \% 3 == 0$
- $1 \leq \text{piles}[i] \leq 10^4$

**1562. Find Latest Group of Size M****Medium**

534112Add to ListShare

Given an array `arr` that represents a permutation of numbers from `1` to `n`.

You have a binary string of size `n` that initially has all its bits set to zero. At each step `i` (assuming both the binary string and `arr` are 1-indexed) from `1` to `n`, the bit at position `arr[i]` is set to `1`.

You are also given an integer `m`. Find the latest step at which there exists a group of ones of length `m`. A group of ones is a contiguous substring of `1`'s such that it cannot be extended in either direction.

Return *the latest step at which there exists a group of ones of length **exactly** m. If no such group exists, return -1*.

**Example 1:****Input:** `arr = [3,5,1,2,4]`, `m = 1`**Output:** 4**Explanation:**Step 1: "00100", groups: ["1"]Step 2: "00101", groups: ["1", "1"]Step 3: "10101", groups: ["1", "1", "1"]Step 4: "11101", groups: ["111", "1"]Step 5: "11111", groups: ["11111"]

The latest step at which there exists a group of size 1 is step 4.

**Example 2:****Input:** `arr = [3,1,5,4,2]`, `m = 2`**Output:** -1

**Explanation:**

Step 1: "00100", groups: ["1"]

Step 2: "10100", groups: ["1", "1"]

Step 3: "10101", groups: ["1", "1", "1"]

Step 4: "10111", groups: ["1", "111"]

Step 5: "11111", groups: ["11111"]

No group of size 2 exists during any step.

**Constraints:**

- $n == \text{arr.length}$
- $1 \leq m \leq n \leq 10^5$
- $1 \leq \text{arr}[i] \leq n$
- All integers in `arr` are **distinct**.

**1563. Stone Game V****Hard**

45263Add to ListShare

There are several stones **arranged in a row**, and each stone has an associated value which is an integer given in the array `stoneValue`.

In each round of the game, Alice divides the row into **two non-empty rows** (i.e. left row and right row), then Bob calculates the value of each row which is the sum of the values of all the stones in this row. Bob throws away the row which has the maximum value, and Alice's score increases by the value of the remaining row. If the value of the two rows are equal, Bob lets Alice decide which row will be thrown away. The next round starts with the remaining row.

The game ends when there is only **one stone remaining**. Alice's is initially **zero**.

Return *the maximum score that Alice can obtain*.

**Example 1:**

**Input:** `stoneValue = [6,2,3,4,5,5]`

**Output:** 18

**Explanation:** In the first round, Alice divides the row to [6,2,3], [4,5,5]. The left row has the value 11 and the right row has value 14. Bob throws away the right row and Alice's score is now 11.

In the second round Alice divides the row to [6], [2,3]. This time Bob throws away the left row and Alice's score becomes 16 (11 + 5).

The last round Alice has only one choice to divide the row which is [2], [3]. Bob throws away the right row and Alice's score is now 18 (16 + 2). The game ends because only one stone is remaining in the row.

### Example 2:

**Input:** stoneValue = [7,7,7,7,7,7,7]

**Output:** 28

### Example 3:

**Input:** stoneValue = [4]

**Output:** 0

### Constraints:

- $1 \leq \text{stoneValue.length} \leq 500$
- $1 \leq \text{stoneValue}[i] \leq 10^6$

## 1566. Detect Pattern of Length M Repeated K or More Times

Easy

52094Add to ListShare

Given an array of positive integers `arr`, find a pattern of length `m` that is repeated `k` or more times.

A **pattern** is a subarray (consecutive sub-sequence) that consists of one or more values, repeated multiple times **consecutively** without overlapping. A pattern is defined by its length and the number of repetitions.

Return `true` if there exists a pattern of length `m` that is repeated `k` or more times, otherwise return `false`.

### Example 1:

**Input:** arr = [1,2,4,4,4,4], m = 1, k = 3

**Output:** true

**Explanation:** The pattern (4) of length 1 is repeated 4 consecutive times. Notice that pattern can be repeated k or more times but not less.

**Example 2:**

**Input:** arr = [1,2,1,2,1,1,1,3], m = 2, k = 2

**Output:** true

**Explanation:** The pattern (1,2) of length 2 is repeated 2 consecutive times. Another valid pattern (2,1) is also repeated 2 times.

**Example 3:**

**Input:** arr = [1,2,1,2,1,3], m = 2, k = 3

**Output:** false

**Explanation:** The pattern (1,2) is of length 2 but is repeated only 2 times. There is no pattern of length 2 that is repeated 3 or more times.

**Constraints:**

- $2 \leq \text{arr.length} \leq 100$
- $1 \leq \text{arr}[i] \leq 100$
- $1 \leq m \leq 100$
- $2 \leq k \leq 100$

## 1567. Maximum Length of Subarray With Positive Product

Medium

184348Add to ListShare

Given an array of integers `nums`, find the maximum length of a subarray where the product of all its elements is positive.

A subarray of an array is a consecutive sequence of zero or more values taken out of that array.

Return *the maximum length of a subarray with positive product*.

**Example 1:**

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

**Output:** 4

**Explanation:** The array `nums` already has a positive product of 24.

**Example 2:**

**Input:** `nums = [0,1,-2,-3,-4]`

**Output:** 3

**Explanation:** The longest subarray with positive product is `[1,-2,-3]` which has a product of 6.

Notice that we cannot include 0 in the subarray since that'll make the product 0 which is not positive.

**Example 3:**

**Input:** `nums = [-1,-2,-3,0,1]`

**Output:** 2

**Explanation:** The longest subarray with positive product is `[-1,-2]` or `[-2,-3]`.

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^5$
- $-10^9 \leq \text{nums}[i] \leq 10^9$

## 1568. Minimum Number of Days to Disconnect Island

Hard

489134Add to ListShare

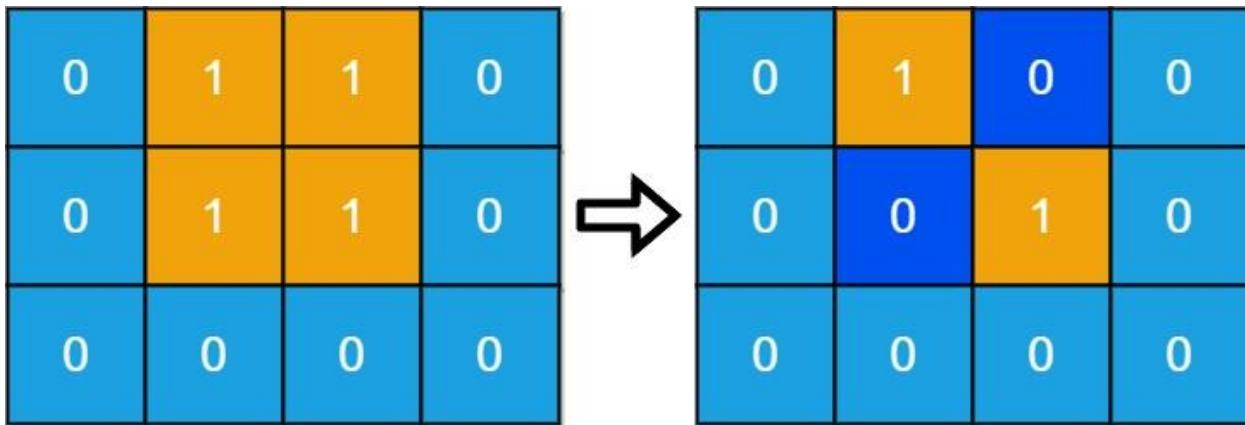
You are given an  $m \times n$  binary grid `grid` where `1` represents land and `0` represents water. An **island** is a maximal **4-directionally** (horizontal or vertical) connected group of `1`'s.

The grid is said to be **connected** if we have **exactly one island**, otherwise is said **disconnected**.

In one day, we are allowed to change **any** single land cell `(1)` into a water cell `(0)`.

Return *the minimum number of days to disconnect the grid*.

**Example 1:**



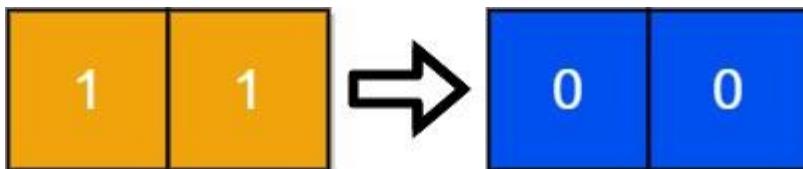
**Input:** grid = [[0,1,1,0],[0,1,1,0],[0,0,0,0]]

**Output:** 2

**Explanation:** We need at least 2 days to get a disconnected grid.

Change land grid[1][1] and grid[0][2] to water and get 2 disconnected island.

**Example 2:**



**Input:** grid = [[1,1]]

**Output:** 2

**Explanation:** Grid of full water is also disconnected ([[1,1]] -> [[0,0]]), 0 islands.

**Constraints:**

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 30`
- `grid[i][j]` is either 0 or 1.

## 1569. Number of Ways to Reorder Array to Get Same BST

Hard

47950Add to ListShare

Given an array `nums` that represents a permutation of integers from 1 to n. We are going to construct a binary search tree (BST) by inserting the elements of `nums` in order into an initially empty

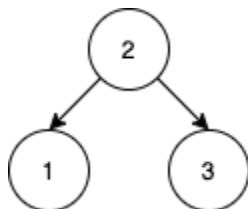
BST. Find the number of different ways to reorder `nums` so that the constructed BST is identical to that formed from the original array `nums`.

- For example, given `nums = [2, 1, 3]`, we will have 2 as the root, 1 as a left child, and 3 as a right child. The array `[2, 3, 1]` also yields the same BST but `[3, 2, 1]` yields a different BST.

Return *the number of ways to reorder* `nums` *such that the BST formed is identical to the original BST formed from* `nums`.

Since the answer may be very large, **return it modulo**  $10^9 + 7$ .

### Example 1:

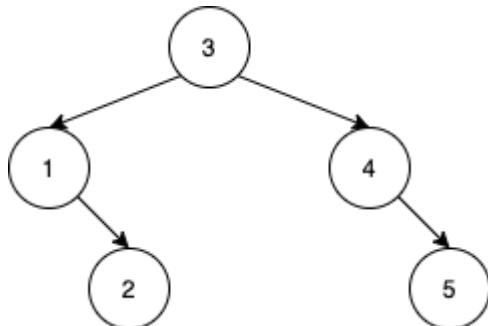


**Input:** `nums = [2,1,3]`

**Output:** 1

**Explanation:** We can reorder `nums` to be `[2,3,1]` which will yield the same BST. There are no other ways to reorder `nums` which will yield the same BST.

### Example 2:



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

**Output:** 5

**Explanation:** The following 5 arrays will yield the same BST:

`[3,1,2,4,5]`

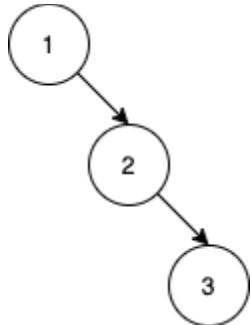
`[3,1,4,2,5]`

`[3,1,4,5,2]`

[3,4,1,2,5]

[3,4,1,5,2]

**Example 3:**



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

**Output:** 0

**Explanation:** There are no other orderings of `nums` that will yield the same BST.

**Constraints:**

- `1 <= nums.length <= 1000`
- `1 <= nums[i] <= nums.length`
- All integers in `nums` are **distinct**.

## 1572. Matrix Diagonal Sum

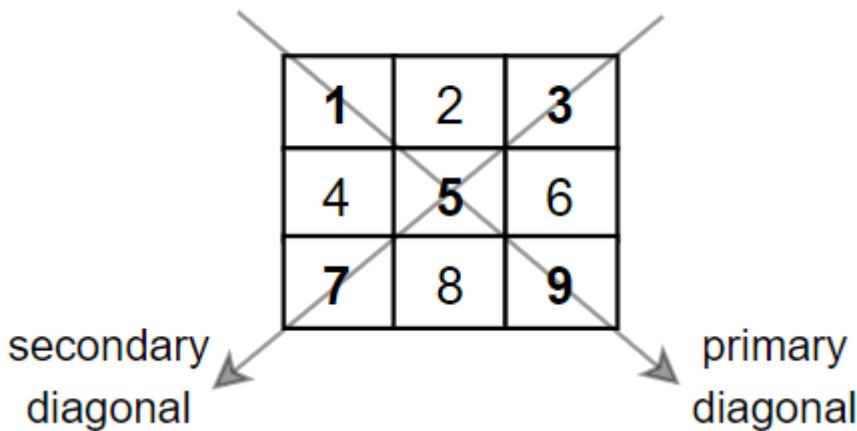
Easy

155823Add to ListShare

Given a square matrix `mat`, return the sum of the matrix diagonals.

Only include the sum of all the elements on the primary diagonal and all the elements on the secondary diagonal that are not part of the primary diagonal.

**Example 1:**



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

[4,5,6],

[7,8,9]]

**Output:** 25

**Explanation:** Diagonals sum: 1 + 5 + 9 + 3 + 7 = 25

Notice that element mat[1][1] = 5 is counted only once.

### Example 2:

**Input:** mat = [[1,1,1,1],

[1,1,1,1],

[1,1,1,1],

[1,1,1,1]]

**Output:** 8

### Example 3:

**Input:** mat = [[5]]

**Output:** 5

### Constraints:

- $n == \text{mat.length} == \text{mat[i].length}$
- $1 \leq n \leq 100$
- $1 \leq \text{mat[i][j]} \leq 100$

## 1573. Number of Ways to Split a String

Medium

51963Add to ListShare

Given a binary string  $s$ , you can split  $s$  into 3 **non-empty** strings  $s_1, s_2$ , and  $s_3$  where  $s_1 + s_2 + s_3 = s$ .

Return the number of ways  $s$  can be split such that the number of ones is the same in  $s_1, s_2$ , and  $s_3$ . Since the answer may be too large, return it **modulo**  $10^9 + 7$ .

### Example 1:

**Input:**  $s = "10101"$

**Output:** 4

**Explanation:** There are four ways to split  $s$  in 3 parts where each part contain the same number of letters '1'.

"1|010|1"

"1|01|01"

"10|10|1"

"10|1|01"

### Example 2:

**Input:**  $s = "1001"$

**Output:** 0

### Example 3:

**Input:**  $s = "0000"$

**Output:** 3

**Explanation:** There are three ways to split  $s$  in 3 parts.

"0|0|00"

"0|00|0"

"00|0|0"

### Constraints:

- $3 \leq s.length \leq 10^5$
- $s[i]$  is either '0' or '1'.

## 1574. Shortest Subarray to be Removed to Make Array Sorted

Medium

121844Add to ListShare

Given an integer array `arr`, remove a subarray (can be empty) from `arr` such that the remaining elements in `arr` are **non-decreasing**.

Return *the length of the shortest subarray to remove*.

A **subarray** is a contiguous subsequence of the array.

### Example 1:

**Input:** `arr = [1,2,3,10,4,2,3,5]`

**Output:** 3

**Explanation:** The shortest subarray we can remove is `[10,4,2]` of length 3. The remaining elements after that will be `[1,2,3,3,5]` which are sorted.

Another correct solution is to remove the subarray `[3,10,4]`.

### Example 2:

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

**Output:** 4

**Explanation:** Since the array is strictly decreasing, we can only keep a single element. Therefore we need to remove a subarray of length 4, either `[5,4,3,2]` or `[4,3,2,1]`.

### Example 3:

**Input:** `arr = [1,2,3]`

**Output:** 0

**Explanation:** The array is already non-decreasing. We do not need to remove any elements.

### Constraints:

- $1 \leq arr.length \leq 10^5$

- $0 \leq \text{arr}[i] \leq 10^9$

## 1575. Count All Possible Routes

Hard

42424Add to ListShare

You are given an array of **distinct** positive integers `locations` where `locations[i]` represents the position of city `i`. You are also given integers `start`, `finish` and `fuel` representing the starting city, ending city, and the initial amount of fuel you have, respectively.

At each step, if you are at city `i`, you can pick any city `j` such that `j != i` and  $0 \leq j < \text{locations.length}$  and move to city `j`. Moving from city `i` to city `j` reduces the amount of fuel you have by  $|\text{locations}[i] - \text{locations}[j]|$ . Please notice that  $|x|$  denotes the absolute value of `x`.

Notice that `fuel` **cannot** become negative at any point in time, and that you are **allowed** to visit any city more than once (including `start` and `finish`).

Return *the count of all possible routes from `start` to `finish`*. Since the answer may be too large, return it modulo  $10^9 + 7$ .

### Example 1:

**Input:** `locations = [2,3,6,8,4]`, `start = 1`, `finish = 3`, `fuel = 5`

**Output:** 4

**Explanation:** The following are all possible routes, each uses 5 units of fuel:

`1 -> 3`

`1 -> 2 -> 3`

`1 -> 4 -> 3`

`1 -> 4 -> 2 -> 3`

### Example 2:

**Input:** `locations = [4,3,1]`, `start = 1`, `finish = 0`, `fuel = 6`

**Output:** 5

**Explanation:** The following are all possible routes:

`1 -> 0, used fuel = 1`

`1 -> 2 -> 0, used fuel = 5`

```
1 -> 2 -> 1 -> 0, used fuel = 5
1 -> 0 -> 1 -> 0, used fuel = 3
1 -> 0 -> 1 -> 0 -> 1 -> 0, used fuel = 5
```

### Example 3:

**Input:** locations = [5,2,1], start = 0, finish = 2, fuel = 3

**Output:** 0

**Explanation:** It is impossible to get from 0 to 2 using only 3 units of fuel since the shortest route needs 4 units of fuel.

### Constraints:

- $2 \leq \text{locations.length} \leq 100$
- $1 \leq \text{locations}[i] \leq 10^9$
- All integers in locations are **distinct**.
- $0 \leq \text{start}, \text{finish} < \text{locations.length}$
- $1 \leq \text{fuel} \leq 200$

## 1576. Replace All ?'s to Avoid Consecutive Repeating Characters

Easy

430148Add to ListShare

Given a string  $s$  containing only lowercase English letters and the ' $?$ ' character, convert **all** the ' $?$ ' characters into lowercase letters such that the final string does not contain any **consecutive repeating** characters. You **cannot** modify the non ' $?$ ' characters.

It is **guaranteed** that there are no consecutive repeating characters in the given string **except** for ' $?$ '.

Return *the final string after all the conversions (possibly zero) have been made*. If there is more than one solution, return **any of them**. It can be shown that an answer is always possible with the given constraints.

### Example 1:

**Input:** s = "?zs"

**Output:** "azs"

**Explanation:** There are 25 solutions for this problem. From "azs" to "yzs", all are valid. Only "z" is an invalid modification as the string will consist of consecutive repeating characters in "zzs".

### Example 2:

**Input:** s = "ubv?w"

**Output:** "ubvaw"

**Explanation:** There are 24 solutions for this problem. Only "v" and "w" are invalid modifications as the strings will consist of consecutive repeating characters in "ubvw" and "ubvw".

### Constraints:

- $1 \leq s.length \leq 100$
- s consist of lowercase English letters and '?'.

## 1577. Number of Ways Where Square of Number Is Equal to Product of Two Numbers

Medium

27248Add to ListShare

Given two arrays of integers `nums1` and `nums2`, return the number of triplets formed (type 1 and type 2) under the following rules:

- Type 1: Triplet (i, j, k) if `nums1[i]² == nums2[j] * nums2[k]` where  $0 \leq i < \text{nums1.length}$  and  $0 \leq j < k < \text{nums2.length}$ .
- Type 2: Triplet (i, j, k) if `nums2[i]² == nums1[j] * nums1[k]` where  $0 \leq i < \text{nums2.length}$  and  $0 \leq j < k < \text{nums1.length}$ .

### Example 1:

**Input:** `nums1 = [7,4]`, `nums2 = [5,2,8,9]`

**Output:** 1

**Explanation:** Type 1: (1, 1, 2),  $\text{nums1}[1]^2 = \text{nums2}[1] * \text{nums2}[2]$ . ( $4^2 = 2 * 8$ ).

### Example 2:

**Input:** `nums1 = [1,1]`, `nums2 = [1,1,1]`

**Output:** 9

**Explanation:** All Triplets are valid, because  $1^2 = 1 * 1$ .

Type 1:  $(0,0,1), (0,0,2), (0,1,2), (1,0,1), (1,0,2), (1,1,2)$ .  $\text{nums1}[i]^2 = \text{nums2}[j] * \text{nums2}[k]$ .

Type 2:  $(0,0,1), (1,0,1), (2,0,1)$ .  $\text{nums2}[i]^2 = \text{nums1}[j] * \text{nums1}[k]$ .

**Example 3:**

**Input:**  $\text{nums1} = [7, 7, 8, 3]$ ,  $\text{nums2} = [1, 2, 9, 7]$

**Output:** 2

**Explanation:** There are 2 valid triplets.

Type 1:  $(3,0,2)$ .  $\text{nums1}[3]^2 = \text{nums2}[0] * \text{nums2}[2]$ .

Type 2:  $(3,0,1)$ .  $\text{nums2}[3]^2 = \text{nums1}[0] * \text{nums1}[1]$ .

**Constraints:**

- $1 \leq \text{nums1.length}, \text{nums2.length} \leq 1000$
- $1 \leq \text{nums1}[i], \text{nums2}[i] \leq 10^5$

## 1578. Minimum Time to Make Rope Colorful

Medium

129347Add to ListShare

Alice has  $n$  balloons arranged on a rope. You are given a **0-indexed** string `colors` where `colors[i]` is the color of the  $i^{\text{th}}$  balloon.

Alice wants the rope to be **colorful**. She does not want **two consecutive balloons** to be of the same color, so she asks Bob for help. Bob can remove some balloons from the rope to make it **colorful**. You are given a **0-indexed** integer array `neededTime` where `neededTime[i]` is the time (in seconds) that Bob needs to remove the  $i^{\text{th}}$  balloon from the rope.

Return *the minimum time Bob needs to make the rope colorful*.

**Example 1:**



**Input:** colors = "abaac", neededTime = [1,2,3,4,5]

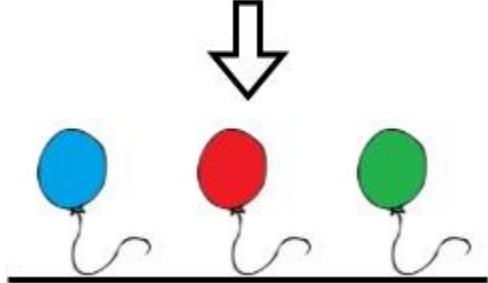
**Output:** 3

**Explanation:** In the above image, 'a' is blue, 'b' is red, and 'c' is green.

Bob can remove the blue balloon at index 2. This takes 3 seconds.

There are no longer two consecutive balloons of the same color. Total time = 3.

**Example 2:**

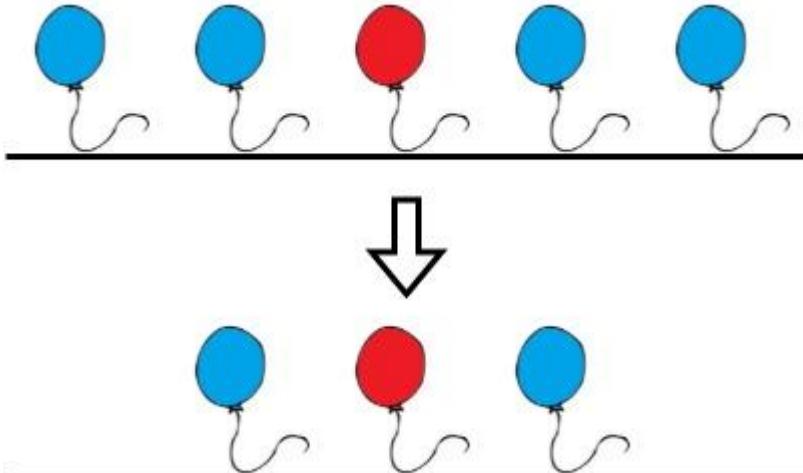


**Input:** colors = "abc", neededTime = [1,2,3]

**Output:** 0

**Explanation:** The rope is already colorful. Bob does not need to remove any balloons from the rope.

**Example 3:**



**Input:** colors = "aabaa", neededTime = [1,2,3,4,1]

**Output:** 2

**Explanation:** Bob will remove the balloons at indices 0 and 4. Each balloon takes 1 second to remove.

There are no longer two consecutive balloons of the same color. Total time = 1 + 1 = 2.

#### Constraints:

- `n == colors.length == neededTime.length`
- `1 <= n <= 105`
- `1 <= neededTime[i] <= 104`
- `colors` contains only lowercase English letters.

## 1579. Remove Max Number of Edges to Keep Graph Fully Traversable

Hard

8267Add to ListShare

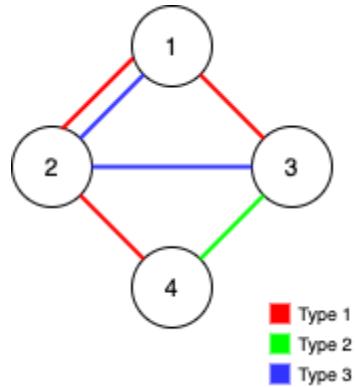
Alice and Bob have an undirected graph of `n` nodes and three types of edges:

- Type 1: Can be traversed by Alice only.
- Type 2: Can be traversed by Bob only.
- Type 3: Can be traversed by both Alice and Bob.

Given an array `edges` where `edges[i] = [typei, ui, vi]` represents a bidirectional edge of type `typei` between nodes `ui` and `vi`, find the maximum number of edges you can remove so that after removing the edges, the graph can still be fully traversed by both Alice and Bob. The graph is fully traversed by Alice and Bob if starting from any node, they can reach all other nodes.

Return the maximum number of edges you can remove, or return `-1` if Alice and Bob cannot fully traverse the graph.

**Example 1:**

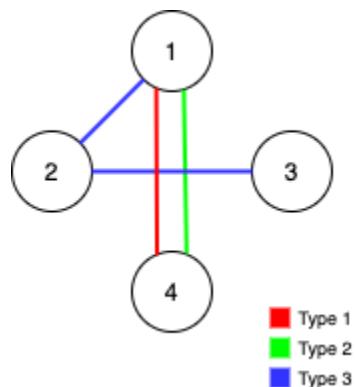


**Input:** `n = 4, edges = [[3,1,2],[3,2,3],[1,1,3],[1,2,4],[1,1,2],[2,3,4]]`

**Output:** `2`

**Explanation:** If we remove the 2 edges `[1,1,2]` and `[1,1,3]`. The graph will still be fully traversable by Alice and Bob. Removing any additional edge will not make it so. So the maximum number of edges we can remove is `2`.

**Example 2:**

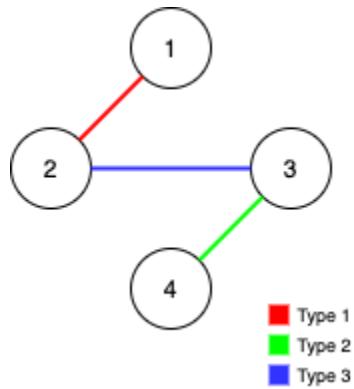


**Input:** `n = 4, edges = [[3,1,2],[3,2,3],[1,1,4],[2,1,4]]`

**Output:** `0`

**Explanation:** Notice that removing any edge will not make the graph fully traversable by Alice and Bob.

**Example 3:**



**Input:** `n = 4, edges = [[3,2,3],[1,1,2],[2,3,4]]`

**Output:** `-1`

**Explanation:** In the current graph, Alice cannot reach node 4 from the other nodes. Likewise, Bob cannot reach 1. Therefore it's impossible to make the graph fully traversable.

### Constraints:

- `1 <= n <= 105`
- `1 <= edges.length <= min(105, 3 * n * (n - 1) / 2)`
- `edges[i].length == 3`
- `1 <= typei <= 3`
- `1 <= ui < vi <= n`
- All tuples `(typei, ui, vi)` are distinct.

## 1581. Customer Who Visited but Did Not Make Any Transactions

- **Easy**

- 32564Add to ListShare

- SQL Schema

- Table: `Visits`

- | Column Name              | Type             |
|--------------------------|------------------|
| <code>visit_id</code>    | <code>int</code> |
| <code>customer_id</code> | <code>int</code> |

- `visit_id` is the primary key for this table.

- This table contains information about the customers who visited the mall.

- 

- Table: `Transactions`

- | Column Name | Type |
|-------------|------|
|-------------|------|

- ```

+-----+-----+
transaction_id	int
visit_id	int
amount	int
+-----+-----+

```

transaction\_id is the primary key for this table.
- This table contains information about the transactions made during the visit\_id.
- 
- Write an SQL query to find the IDs of the users who visited without making any transactions and the number of times they made these types of visits.
- Return the result table sorted in **any order**.
- The query result format is in the following example.
- 
- Example 1:**
- Input:**
- Visits

| visit_id | customer_id |
|----------|-------------|
| 1        | 23          |
| 2        | 9           |
| 4        | 30          |
| 5        | 54          |
| 6        | 96          |
| 7        | 54          |
| 8        | 54          |

- Transactions

| transaction_id | visit_id | amount |
|----------------|----------|--------|
| 2              | 5        | 310    |
| 3              | 5        | 300    |
| 9              | 5        | 200    |
| 12             | 1        | 910    |
| 13             | 2        | 970    |

- Output:**

| customer_id | count_no_trans |
|-------------|----------------|
| 54          | 2              |
| 30          | 1              |
| 96          | 1              |

- Explanation:**
- Customer with id = 23 visited the mall once and made one transaction during the visit with id = 12.
- Customer with id = 9 visited the mall once and made one transaction during the visit with id = 13.

- Customer with `id = 30` visited the mall once and did not make any transactions.
- Customer with `id = 54` visited the mall three times. During 2 visits they did not make any transactions, and during one visit they made 3 transactions.
- Customer with `id = 96` visited the mall once and did not make any transactions.
- As we can see, users with IDs 30 and 96 visited the mall one time without making any transactions. Also, user 54 visited the mall twice and did not make any transactions.

## 1582. Special Positions in a Binary Matrix

Easy

51619Add to ListShare

Given an `m x n` binary matrix `mat`, return *the number of special positions in mat*.

A position `(i, j)` is called **special** if `mat[i][j] == 1` and all other elements in row `i` and column `j` are `0` (rows and columns are **0-indexed**).

**Example 1:**

|   |   |   |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |

**Input:** `mat = [[1,0,0],[0,0,1],[1,0,0]]`

**Output:** 1

**Explanation:** `(1, 2)` is a special position because `mat[1][2] == 1` and all other elements in row 1 and column 2 are `0`.

**Example 2:**

|   |   |   |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

**Input:** mat = [[1,0,0],[0,1,0],[0,0,1]]

**Output:** 3

**Explanation:** (0, 0), (1, 1) and (2, 2) are special positions.

### Constraints:

- $m == \text{mat.length}$
- $n == \text{mat[i].length}$
- $1 \leq m, n \leq 100$
- $\text{mat[i][j]}$  is either 0 or 1.

## 1583. Count Unhappy Friends

Medium

197660Add to ListShare

You are given a list of preferences for  $n$  friends, where  $n$  is always **even**.

For each person  $i$ ,  $\text{preferences}[i]$  contains a list of friends **sorted** in the **order of preference**. In other words, a friend earlier in the list is more preferred than a friend later in the list. Friends in each list are denoted by integers from 0 to  $n-1$ .

All the friends are divided into pairs. The pairings are given in a list  $\text{pairs}$ , where  $\text{pairs}[i] = [x_i, y_i]$  denotes  $x_i$  is paired with  $y_i$  and  $y_i$  is paired with  $x_i$ .

However, this pairing may cause some of the friends to be unhappy. A friend  $x$  is unhappy if  $x$  is paired with  $y$  and there exists a friend  $u$  who is paired with  $v$  but:

- $x$  prefers  $u$  over  $y$ , and
- $u$  prefers  $x$  over  $v$ .

Return *the number of unhappy friends*.

**Example 1:**

**Input:** `n = 4, preferences = [[1, 2, 3], [3, 2, 0], [3, 1, 0], [1, 2, 0]], pairs = [[0, 1], [2, 3]]`

**Output:** 2

**Explanation:**

Friend 1 is unhappy because:

- 1 is paired with 0 but prefers 3 over 0, and
- 3 prefers 1 over 2.

Friend 3 is unhappy because:

- 3 is paired with 2 but prefers 1 over 2, and
- 1 prefers 3 over 0.

Friends 0 and 2 are happy.

**Example 2:**

**Input:** `n = 2, preferences = [[1], [0]], pairs = [[1, 0]]`

**Output:** 0

**Explanation:** Both friends 0 and 1 are happy.

**Example 3:**

**Input:** `n = 4, preferences = [[1, 3, 2], [2, 3, 0], [1, 3, 0], [0, 2, 1]], pairs = [[1, 3], [0, 2]]`

**Output:** 4

**Constraints:**

- `2 <= n <= 500`
- `n` is even.
- `preferences.length == n`
- `preferences[i].length == n - 1`
- `0 <= preferences[i][j] <= n - 1`
- `preferences[i]` does not contain `i`.

- All values in `preferences[i]` are unique.
- `pairs.length == n/2`
- `pairs[i].length == 2`
- `xi != yi`
- `0 <= xi, yi <= n - 1`
- Each person is contained in **exactly one** pair.

## 1584. Min Cost to Connect All Points

Medium

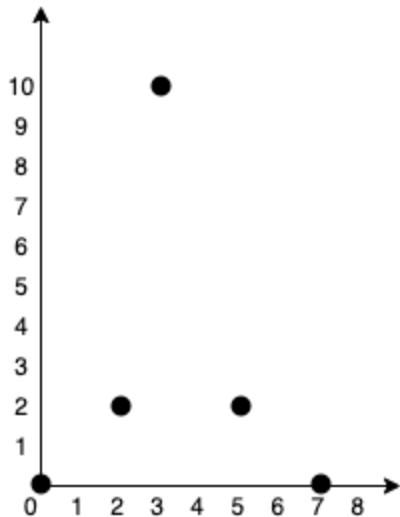
296979Add to ListShare

You are given an array `points` representing integer coordinates of some points on a 2D-plane, where `points[i] = [xi, yi]`.

The cost of connecting two points `[xi, yi]` and `[xj, yj]` is the **manhattan distance** between them:  $|x_i - x_j| + |y_i - y_j|$ , where  $|val|$  denotes the absolute value of `val`.

Return *the minimum cost to make all points connected*. All points are connected if there is **exactly one** simple path between any two points.

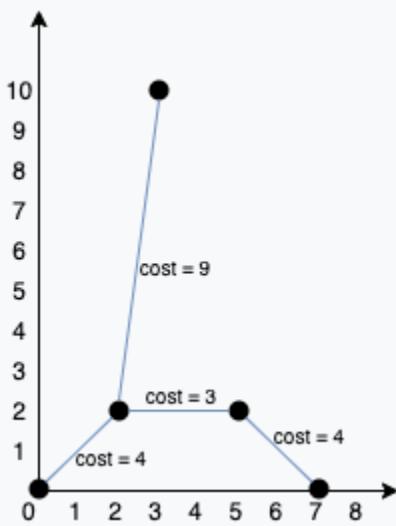
### Example 1:



**Input:** `points = [[0,0],[2,2],[3,10],[5,2],[7,0]]`

**Output:** 20

**Explanation:**



We can connect the points as shown above to get the minimum cost of 20.

Notice that there is a unique path between every pair of points.

### Example 2:

**Input:** points = [[3,12],[-2,5],[-4,1]]

**Output:** 18

### Constraints:

- $1 \leq \text{points.length} \leq 1000$
- $-10^6 \leq x_i, y_i \leq 10^6$
- All pairs  $(x_i, y_i)$  are distinct.

## 1585. Check If String Is Transformable With Substring Sort Operations

Hard

3647Add to ListShare

Given two strings  $s$  and  $t$ , transform string  $s$  into string  $t$  using the following operation any number of times:

- Choose a **non-empty** substring in  $s$  and sort it in place so the characters are in **ascending order**.
  - For example, applying the operation on the underlined substring in "14234" results in "12344".

Return `true` if it is possible to transform  $s$  into  $t$ . Otherwise, return `false`.

A **substring** is a contiguous sequence of characters within a string.

**Example 1:**

**Input:** s = "84532", t = "34852"

**Output:** true

**Explanation:** You can transform s into t using the following sort operations:

"84532" (from index 2 to 3) -> "84352"

"84352" (from index 0 to 2) -> "34852"

**Example 2:**

**Input:** s = "34521", t = "23415"

**Output:** true

**Explanation:** You can transform s into t using the following sort operations:

"34521" -> "23451"

"23451" -> "23415"

**Example 3:**

**Input:** s = "12345", t = "12435"

**Output:** false

**Constraints:**

- s.length == t.length
- 1 <= s.length <= 10<sup>5</sup>
- s and t consist of only digits.

## 1587. Bank Account Summary II

Easy

2204Add to ListShare

SQL Schema

Table: `Users`

| Column Name | Type    |
|-------------|---------|
| account     | int     |
| name        | varchar |

account is the primary key for this table.

Each row of this table contains the account number of each user in the bank.

There will be no two users having the same name in the table.

Table: Transactions

| Column Name   | Type |
|---------------|------|
| trans_id      | int  |
| account       | int  |
| amount        | int  |
| transacted_on | date |

trans\_id is the primary key for this table.

Each row of this table contains all changes made to all accounts.

amount is positive if the user received money and negative if they transferred money.

All accounts start with a balance of 0.

Write an SQL query to report the name and balance of users with a balance higher than 10000. The balance of an account is equal to the sum of the amounts of all transactions involving that account.

Return the result table in **any order**.

The query result format is in the following example.

**Example 1:**

**Input:**

Users table:

| account | name    |
|---------|---------|
| 900001  | Alice   |
| 900002  | Bob     |
| 900003  | Charlie |

Transactions table:

| trans_id | account | amount | transacted_on |
|----------|---------|--------|---------------|
| 1        | 900001  | 7000   | 2020-08-01    |
| 2        | 900001  | 7000   | 2020-09-01    |
| 3        | 900001  | -3000  | 2020-09-02    |
| 4        | 900002  | 1000   | 2020-09-12    |
| 5        | 900003  | 6000   | 2020-08-07    |
| 6        | 900003  | 6000   | 2020-09-07    |
| 7        | 900003  | -4000  | 2020-09-11    |

**Output:**

| name | balance |
|------|---------|
|      |         |

|             |       |  |
|-------------|-------|--|
| ----- ----- |       |  |
| Alice       | 11000 |  |
| ----- ----- |       |  |

**Explanation:**

Alice's balance is  $(7000 + 7000 - 3000) = 11000$ .

Bob's balance is 1000.

Charlie's balance is  $(6000 + 6000 - 4000) = 8000$ .

## 1588. Sum of All Odd Length Subarrays

Easy

2534195Add to ListShare

Given an array of positive integers `arr`, return the sum of all possible **odd-length subarrays** of `arr`.

A **subarray** is a contiguous subsequence of the array.

**Example 1:**

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

**Output:** 58

**Explanation:** The odd-length subarrays of arr and their sums are:

[1] = 1

[4] = 4

[2] = 2

[5] = 5

[3] = 3

[1,4,2] = 7

[4,2,5] = 11

[2,5,3] = 10

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

If we add all these together we get  $1 + 4 + 2 + 5 + 3 + 7 + 11 + 10 + 15 = 58$

**Example 2:**

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

**Output:** 3

**Explanation:** There are only 2 subarrays of odd length, [1] and [2]. Their sum is 3.

**Example 3:**

**Input:** arr = [10,11,12]

**Output:** 66

**Constraints:**

- $1 \leq \text{arr.length} \leq 100$
- $1 \leq \text{arr}[i] \leq 1000$

## 1589. Maximum Sum Obtained of Any Permutation

Medium

56527Add to ListShare

We have an array of integers, `nums`, and an array of `requests` where `requests[i] = [starti, endi]`. The  $i^{\text{th}}$  request asks for the sum of `nums[starti] + nums[starti + 1] + ... + nums[endi - 1] + nums[endi]`. Both `starti` and `endi` are 0-indexed.

Return the maximum total sum of all requests **among all permutations** of `nums`.

Since the answer may be too large, return it **modulo**  $10^9 + 7$ .

**Example 1:**

**Input:** `nums = [1,2,3,4,5]`, `requests = [[1,3],[0,1]]`

**Output:** 19

**Explanation:** One permutation of `nums` is `[2,1,3,4,5]` with the following result:

`requests[0]`  $\rightarrow$  `nums[1] + nums[2] + nums[3] = 1 + 3 + 4 = 8`

`requests[1]`  $\rightarrow$  `nums[0] + nums[1] = 2 + 1 = 3`

Total sum:  $8 + 3 = 11$ .

A permutation with a higher total sum is `[3,5,4,2,1]` with the following result:

```
requests[0] -> nums[1] + nums[2] + nums[3] = 5 + 4 + 2 = 11
```

```
requests[1] -> nums[0] + nums[1] = 3 + 5 = 8
```

Total sum: 11 + 8 = 19, which is the best that you can do.

### Example 2:

**Input:** nums = [1,2,3,4,5,6], requests = [[0,1]]

**Output:** 11

**Explanation:** A permutation with the max total sum is [6,5,4,3,2,1] with request sums [11].

### Example 3:

**Input:** nums = [1,2,3,4,5,10], requests = [[0,2],[1,3],[1,1]]

**Output:** 47

**Explanation:** A permutation with the max total sum is [4,10,5,3,2,1] with request sums [19,18,10].

### Constraints:

- $n == \text{nums.length}$
- $1 \leq n \leq 10^5$
- $0 \leq \text{nums}[i] \leq 10^5$
- $1 \leq \text{requests.length} \leq 10^5$
- $\text{requests}[i].length == 2$
- $0 \leq \text{start}_i \leq \text{end}_i < n$

## 1590. Make Sum Divisible by P

Medium

106145Add to ListShare

Given an array of positive integers `nums`, remove the **smallest** subarray (possibly **empty**) such that the **sum** of the remaining elements is divisible by `p`. It is **not** allowed to remove the whole array.

Return the *length* of the smallest subarray that you need to remove, or `-1` if it's impossible.

A **subarray** is defined as a contiguous block of elements in the array.

### Example 1:

**Input:** nums = [3,1,4,2], p = 6

**Output:** 1

**Explanation:** The sum of the elements in `nums` is 10, which is not divisible by 6. We can remove the subarray [4], and the sum of the remaining elements is 6, which is divisible by 6.

**Example 2:**

**Input:** `nums` = [6,3,5,2], `p` = 9

**Output:** 2

**Explanation:** We cannot remove a single element to get a sum divisible by 9. The best way is to remove the subarray [5,2], leaving us with [6,3] with sum 9.

**Example 3:**

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

**Output:** 0

**Explanation:** Here the sum is 6. which is already divisible by 3. Thus we do not need to remove anything.

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^5$
- $1 \leq \text{nums}[i] \leq 10^9$
- $1 \leq p \leq 10^9$

## 1591. Strange Printer II

Hard

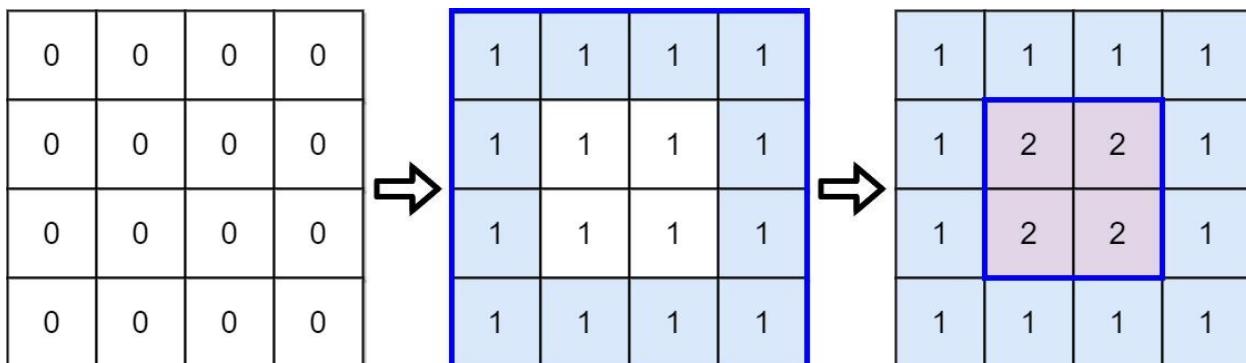
42315Add to ListShare

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

- On each turn, the printer will print a solid rectangular pattern of a single color on the grid. This will cover up the existing colors in the rectangle.
- Once the printer has used a color for the above operation, **the same color cannot be used again.**

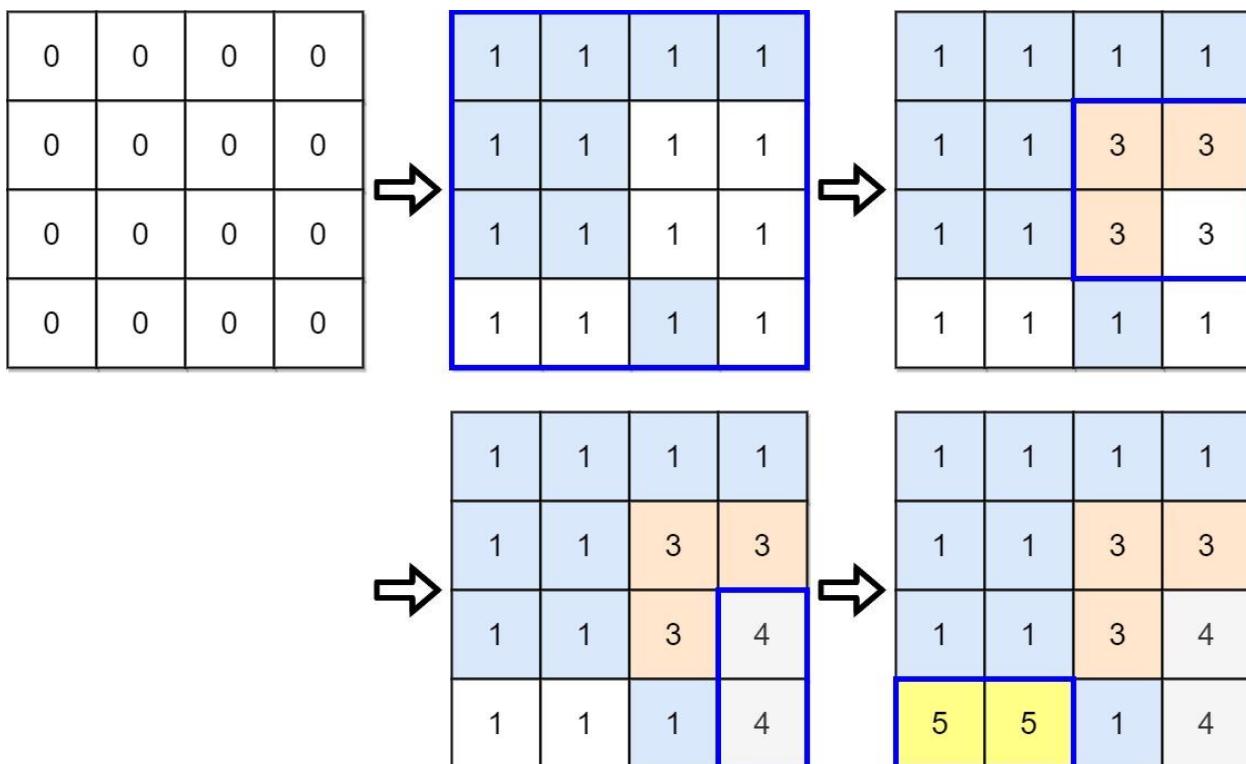
You are given a  $m \times n$  matrix `targetGrid`, where `targetGrid[row][col]` is the color in the position  $(row, col)$  of the grid.

Return `true` if it is possible to print the matrix `targetGrid`, otherwise, return `false`.

**Example 1:**

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

**Output:** true

**Example 2:**

**Input:** targetGrid = [[1,1,1,1],[1,1,3,3],[1,1,3,4],[5,5,1,4]]

**Output:** true

**Example 3:**

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

**Output:** false

**Explanation:** It is impossible to form targetGrid because it is not allowed to print the same color in different turns.

### Constraints:

- `m == targetGrid.length`
- `n == targetGrid[i].length`
- `1 <= m, n <= 60`
- `1 <= targetGrid[row][col] <= 60`

## 1592. Rearrange Spaces Between Words

Easy

317274Add to ListShare

You are given a string `text` of words that are placed among some number of spaces. Each word consists of one or more lowercase English letters and are separated by at least one space. It's guaranteed that `text` **contains at least one word**.

Rearrange the spaces so that there is an **equal** number of spaces between every pair of adjacent words and that number is **maximized**. If you cannot redistribute all the spaces equally, place the **extra spaces at the end**, meaning the returned string should be the same length as `text`.

Return *the string after rearranging the spaces*.

### Example 1:

**Input:** `text = " this is a sentence "`

**Output:** `"this is a sentence"`

**Explanation:** There are a total of 9 spaces and 4 words. We can evenly divide the 9 spaces between the words:  $9 / (4-1) = 3$  spaces.

### Example 2:

**Input:** `text = " practice makes perfect "`

**Output:** `"practice makes perfect "`

**Explanation:** There are a total of 7 spaces and 3 words.  $7 / (3-1) = 3$  spaces plus 1 extra space. We place this extra space at the end of the string.

### Constraints:

- `1 <= text.length <= 100`
- `text` consists of lowercase English letters and `' '`.
- `text` contains at least one word.

## 1593. Split a String Into the Max Number of Unique Substrings

Medium

64425Add to ListShare

Given a string `s`, return *the maximum number of unique substrings that the given string can be split into*.

You can split string `s` into any list of **non-empty substrings**, where the concatenation of the substrings forms the original string. However, you must split the substrings such that all of them are **unique**.

A **substring** is a contiguous sequence of characters within a string.

### Example 1:

**Input:** `s = "ababccc"`

**Output:** 5

**Explanation:** One way to split maximally is `['a', 'b', 'ab', 'c', 'cc']`. Splitting like `['a', 'b', 'a', 'b', 'c', 'cc']` is not valid as you have 'a' and 'b' multiple times.

### Example 2:

**Input:** `s = "aba"`

**Output:** 2

**Explanation:** One way to split maximally is `['a', 'ba']`.

### Example 3:

**Input:** `s = "aa"`

**Output:** 1

**Explanation:** It is impossible to split the string any further.

### Constraints:

- `1 <= s.length <= 16`

- `s` contains only lower case English letters.

## 1594. Maximum Non Negative Product in a Matrix

Medium

63230Add to ListShare

You are given a  $m \times n$  matrix `grid`. Initially, you are located at the top-left corner  $(0, 0)$ , and in each step, you can only **move right or down** in the matrix.

Among all possible paths starting from the top-left corner  $(0, 0)$  and ending in the bottom-right corner  $(m - 1, n - 1)$ , find the path with the **maximum non-negative product**. The product of a path is the product of all integers in the grid cells visited along the path.

Return the *maximum non-negative product modulo  $10^9 + 7$* . If the maximum product is **negative**, return `-1`.

Notice that the modulo is performed after getting the maximum product.

**Example 1:**

|    |    |    |
|----|----|----|
| -1 | -2 | -3 |
| -2 | -3 | -3 |
| -3 | -3 | -2 |

**Input:** `grid = [[-1,-2,-3],[-2,-3,-3],[-3,-3,-2]]`

**Output:** `-1`

**Explanation:** It is not possible to get non-negative product in the path from  $(0, 0)$  to  $(2, 2)$ , so return `-1`.

**Example 2:**

|   |    |   |
|---|----|---|
| 1 | -2 | 1 |
| 1 | -2 | 1 |
| 3 | -4 | 1 |

**Input:** grid = [[1,-2,1],[1,-2,1],[3,-4,1]]

**Output:** 8

**Explanation:** Maximum non-negative product is shown (1 \* 1 \* -2 \* -4 \* 1 = 8).

**Example 3:**

|   |    |
|---|----|
| 1 | 3  |
| 0 | -4 |

**Input:** grid = [[1,3],[0,-4]]

**Output:** 0

**Explanation:** Maximum non-negative product is shown (1 \* 0 \* -4 = 0).

**Constraints:**

- $m == \text{grid.length}$
- $n == \text{grid}[i].length$
- $1 \leq m, n \leq 15$
- $-4 \leq \text{grid}[i][j] \leq 4$

## 1595. Minimum Cost to Connect Two Groups of Points

Hard

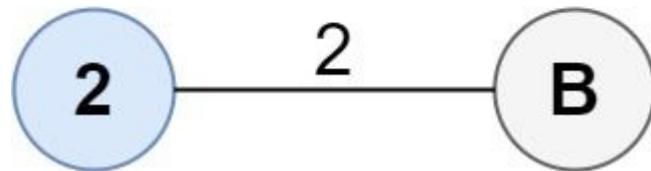
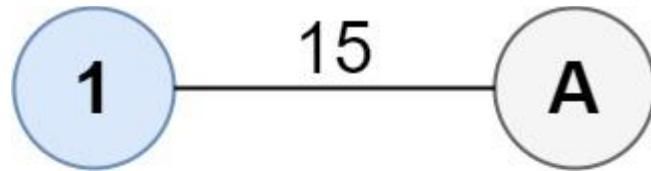
36913Add to ListShare

You are given two groups of points where the first group has `size1` points, the second group has `size2` points, and `size1 >= size2`.

The `cost` of the connection between any two points are given in an `size1 x size2` matrix where `cost[i][j]` is the cost of connecting point `i` of the first group and point `j` of the second group. The groups are connected if **each point in both groups is connected to one or more points in the opposite group**. In other words, each point in the first group must be connected to at least one point in the second group, and each point in the second group must be connected to at least one point in the first group.

Return *the minimum cost it takes to connect the two groups*.

**Example 1:**



**Input:** `cost = [[15, 96], [36, 2]]`

**Output:** 17

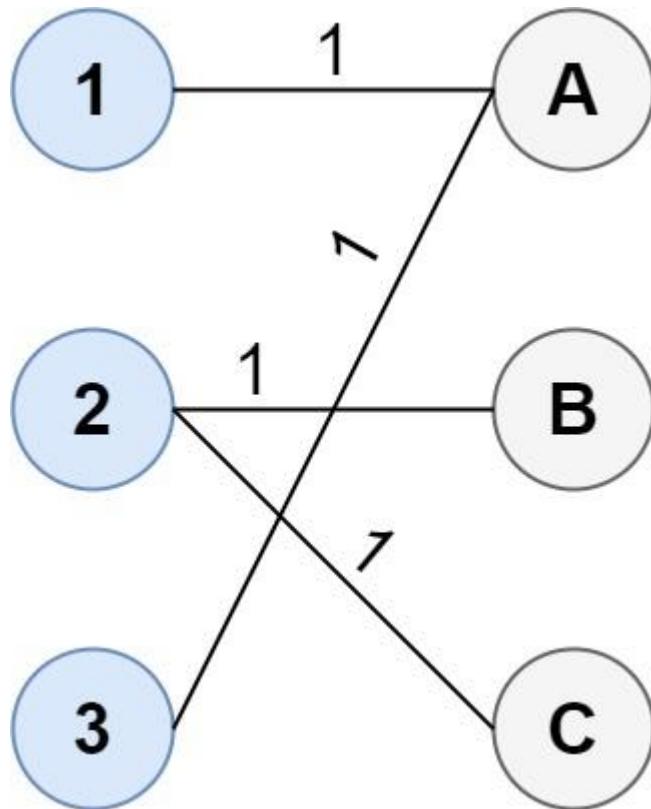
**Explanation:** The optimal way of connecting the groups is:

1--A

2--B

This results in a total cost of 17.

**Example 2:**



**Input:** cost = [[1, 3, 5], [4, 1, 1], [1, 5, 3]]

**Output:** 4

**Explanation:** The optimal way of connecting the groups is:

1--A

2--B

2--C

3--A

This results in a total cost of 4.

Note that there are multiple points connected to point 2 in the first group and point A in the second group. This does not matter as there is no limit to the number of points that can be connected. We only care about the minimum total cost.

**Example 3:**

**Input:** cost = [[2, 5, 1], [3, 4, 7], [8, 1, 2], [6, 2, 4], [3, 8, 8]]

**Output:** 10

### Constraints:

- $size_1 == cost.length$
- $size_2 == cost[i].length$
- $1 \leq size_1, size_2 \leq 12$
- $size_1 \geq size_2$
- $0 \leq cost[i][j] \leq 100$

## 1598. Crawler Log Folder

Easy

63944Add to ListShare

The Leetcode file system keeps a log each time some user performs a *change folder* operation.

The operations are described below:

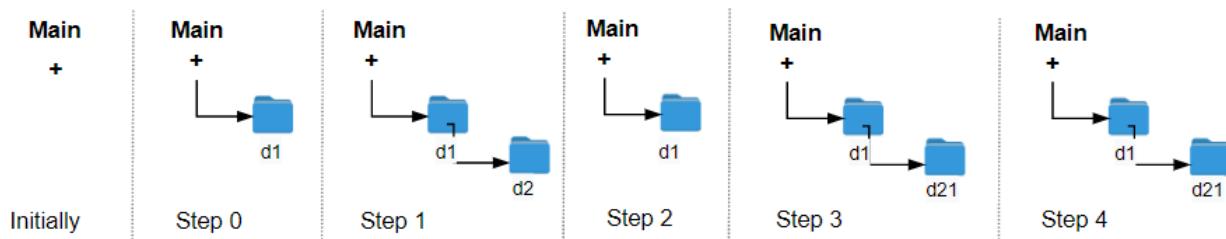
- `"../"` : Move to the parent folder of the current folder. (If you are already in the main folder, **remain in the same folder**).
- `". /"` : Remain in the same folder.
- `"x /"` : Move to the child folder named `x` (This folder is **guaranteed to always exist**).

You are given a list of strings `logs` where `logs[i]` is the operation performed by the user at the  $i^{\text{th}}$  step.

The file system starts in the main folder, then the operations in `logs` are performed.

Return the *minimum number of operations needed to go back to the main folder after the change folder operations*.

### Example 1:

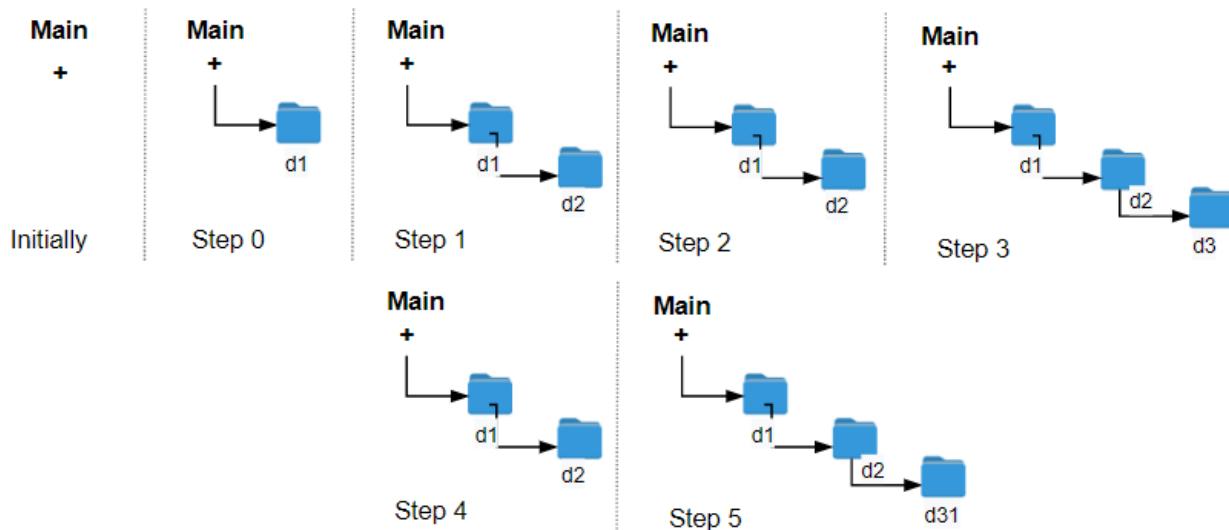


**Input:** `logs = ["d1/", "d2/", "../", "d21/", "./"]`

**Output:** 2

**Explanation:** Use this change folder operation `../` 2 times and go back to the main folder.

### Example 2:



**Input:** logs = ["d1/", "d2/", "./", "d3/", "../", "d31/"]

**Output:** 3

**Example 3:**

**Input:** logs = ["d1/", "../", "../", "../"]

**Output:** 0

**Constraints:**

- $1 \leq \text{logs.length} \leq 10^3$
- $2 \leq \text{logs}[i].length \leq 10$
- $\text{logs}[i]$  contains lowercase English letters, digits, '.', and '/'.
- $\text{logs}[i]$  follows the format described in the statement.
- Folder names consist of lowercase English letters and digits.

## 1599. Maximum Profit of Operating a Centennial Wheel

Medium

79221Add to ListShare

You are the operator of a Centennial Wheel that has **four gondolas**, and each gondola has room for **up to four people**. You have the ability to rotate the gondolas **counterclockwise**, which costs you **runningCost** dollars.

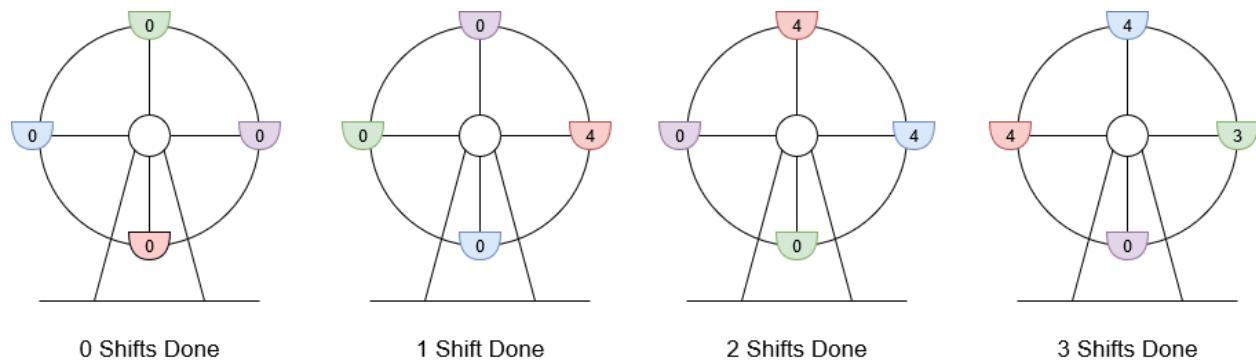
You are given an array `customers` of length `n` where `customers[i]` is the number of new customers arriving just before the `ith` rotation (0-indexed). This means you **must rotate the wheel `i` times before the `customers[i]` customers arrive**. You cannot make customers wait if

**there is room in the gondola.** Each customer pays `boardingCost` dollars when they board on the gondola closest to the ground and will exit once that gondola reaches the ground again.

You can stop the wheel at any time, including **before serving all customers**. If you decide to stop serving customers, **all subsequent rotations are free** in order to get all the customers down safely. Note that if there are currently more than four customers waiting at the wheel, only four will board the gondola, and the rest will wait **for the next rotation**.

Return *the minimum number of rotations you need to perform to maximize your profit*. If there is **no scenario** where the profit is positive, return `-1`.

### Example 1:



**Input:** `customers = [8,3]`, `boardingCost = 5`, `runningCost = 6`

**Output:** 3

**Explanation:** The numbers written on the gondolas are the number of people currently there.

1. 8 customers arrive, 4 board and 4 wait for the next gondola, the wheel rotates. Current profit is  $4 * \$5 - 1 * \$6 = \$14$ .
2. 3 customers arrive, the 4 waiting board the wheel and the other 3 wait, the wheel rotates. Current profit is  $8 * \$5 - 2 * \$6 = \$28$ .
3. The final 3 customers board the gondola, the wheel rotates. Current profit is  $11 * \$5 - 3 * \$6 = \$37$ .

The highest profit was \$37 after rotating the wheel 3 times.

### Example 2:

**Input:** `customers = [10,9,6]`, `boardingCost = 6`, `runningCost = 4`

**Output:** 7

**Explanation:**

1. 10 customers arrive, 4 board and 6 wait for the next gondola, the wheel rotates. Current profit is  $4 * \$6 - 1 * \$4 = \$20$ .
2. 9 customers arrive, 4 board and 11 wait (2 originally waiting, 9 newly waiting), the wheel rotates. Current profit is  $8 * \$6 - 2 * \$4 = \$40$ .
3. The final 6 customers arrive, 4 board and 13 wait, the wheel rotates. Current profit is  $12 * \$6 - 3 * \$4 = \$60$ .
4. 4 board and 9 wait, the wheel rotates. Current profit is  $16 * \$6 - 4 * \$4 = \$80$ .
5. 4 board and 5 wait, the wheel rotates. Current profit is  $20 * \$6 - 5 * \$4 = \$100$ .
6. 4 board and 1 waits, the wheel rotates. Current profit is  $24 * \$6 - 6 * \$4 = \$120$ .
7. 1 boards, the wheel rotates. Current profit is  $25 * \$6 - 7 * \$4 = \$122$ .

The highest profit was \$122 after rotating the wheel 7 times.

**Example 3:**

**Input:** customers = [3,4,0,5,1], boardingCost = 1, runningCost = 92

**Output:** -1

**Explanation:**

1. 3 customers arrive, 3 board and 0 wait, the wheel rotates. Current profit is  $3 * \$1 - 1 * \$92 = -\$89$ .
2. 4 customers arrive, 4 board and 0 wait, the wheel rotates. Current profit is  $7 * \$1 - 2 * \$92 = -\$177$ .
3. 0 customers arrive, 0 board and 0 wait, the wheel rotates. Current profit is  $7 * \$1 - 3 * \$92 = -\$269$ .
4. 5 customers arrive, 4 board and 1 waits, the wheel rotates. Current profit is  $11 * \$1 - 4 * \$92 = -\$357$ .
5. 1 customer arrives, 2 board and 0 wait, the wheel rotates. Current profit is  $13 * \$1 - 5 * \$92 = -\$447$ .

The profit was never positive, so return -1.

**Constraints:**

- $n == \text{customers.length}$
- $1 \leq n \leq 10^5$
- $0 \leq \text{customers}[i] \leq 50$

- `1 <= boardingCost, runningCost <= 100`

## 1600. Throne Inheritance

Medium

206260Add to ListShare

A kingdom consists of a king, his children, his grandchildren, and so on. Every once in a while, someone in the family dies or a child is born.

The kingdom has a well-defined order of inheritance that consists of the king as the first member. Let's define the recursive function `Successor(x, curOrder)`, which given a person `x` and the inheritance order so far, returns who should be the next person after `x` in the order of inheritance.

`Successor(x, curOrder):`

```
if x has no children or all of x's children are in curOrder:
    if x is the king return null
    else return Successor(x's parent, curOrder)
else return x's oldest child who's not in curOrder
```

For example, assume we have a kingdom that consists of the king, his children Alice and Bob (Alice is older than Bob), and finally Alice's son Jack.

1. In the beginning, `curOrder` will be `["king"]`.
2. Calling `Successor(king, curOrder)` will return Alice, so we append to `curOrder` to get `["king", "Alice"]`.
3. Calling `Successor(Alice, curOrder)` will return Jack, so we append to `curOrder` to get `["king", "Alice", "Jack"]`.
4. Calling `Successor(Jack, curOrder)` will return Bob, so we append to `curOrder` to get `["king", "Alice", "Jack", "Bob"]`.
5. Calling `Successor(Bob, curOrder)` will return `null`. Thus the order of inheritance will be `["king", "Alice", "Jack", "Bob"]`.

Using the above function, we can always obtain a unique order of inheritance.

Implement the `ThroneInheritance` class:

- `ThroneInheritance(string kingName)` Initializes an object of the `ThroneInheritance` class. The name of the king is given as part of the constructor.
- `void birth(string parentName, string childName)` Indicates that `parentName` gave birth to `childName`.
- `void death(string name)` Indicates the death of `name`. The death of the person doesn't affect the `Successor` function nor the current inheritance order. You can treat it as just marking the person as dead.

- `string[] getInheritanceOrder()` Returns a list representing the current order of inheritance **excluding** dead people.

### Example 1:

#### Input

```
["ThroneInheritance", "birth", "birth", "birth", "birth", "birth", "birth", "birth", "getInheritanceOrder", "death", "getInheritanceOrder"]
[[["king"], ["king", "andy"], ["king", "bob"], ["king", "catherine"], ["andy", "matthew"], ["bob", "alex"], ["bob", "asha"], [null], ["bob"], [null]]]
```

#### Output

```
[null, null, null, null, null, null, ["king", "andy", "matthew", "bob", "alex", "asha", "catherine"], null, ["king", "andy", "matthew", "alex", "asha", "catherine"]]
```

#### Explanation

```
ThroneInheritance t= new ThroneInheritance("king"); // order: king
t.birth("king", "andy"); // order: king > andy
t.birth("king", "bob"); // order: king > andy > bob
t.birth("king", "catherine"); // order: king > andy > bob > catherine
t.birth("andy", "matthew"); // order: king > andy > matthew > bob > catherine
t.birth("bob", "alex"); // order: king > andy > matthew > bob > alex > catherine
t.birth("bob", "asha"); // order: king > andy > matthew > bob > alex > asha > catherine
t.getInheritanceOrder(); // return ["king", "andy", "matthew", "bob", "alex", "asha", "catherine"]
t.death("bob"); // order: king > andy > matthew > bob > alex > asha > catherine
t.getInheritanceOrder(); // return ["king", "andy", "matthew", "alex", "asha", "catherine"]
```

#### Constraints:

- `1 <= kingName.length, parentName.length, childName.length, name.length <= 15`

- `kingName`, `parentName`, `childName`, and `name` consist of lowercase English letters only.
- All arguments `childName` and `kingName` are **distinct**.
- All `name` arguments of `death` will be passed to either the constructor or as `childName` to `birth` first.
- For each call to `birth(parentName, childName)`, it is guaranteed that `parentName` is alive.
- At most  $10^5$  calls will be made to `birth` and `death`.
- At most 10 calls will be made to `getInheritanceOrder`.