Homework 1

CSE 202
Fall 2024

Due at: 11:59 PM Monday, October 7

Background: order notation, induction, loop invariants, simple analysis of algorithms

---

Graded Problems - 20 points each

1. Rank functions according to their order (smallest to largest). Note: it is possible for functions to be of the same order.

   (a) $\sqrt{\log n}$, $\log(\sqrt{n})$, $\log(\log(n))$, $\sqrt{\sqrt{n}}$

   $\log(\log(n))$ is a logarithmic function of a logarithmic function which means it grows the slowest. $\sqrt{\sqrt{n}}$ can be transferred to $n^{1/4}$ which is a polynomial function which grows faster than logarithmic function.
   From smallest to largest: $\log(\log(n))$**,** $\sqrt{\log n}$**,** $\log(\sqrt{n})$**,**$\sqrt{\sqrt{n}}$

   (b) 2^n, $n^{\log n}$, $(\log n)^{\sqrt{n}}$

   Exponential functions grows the fastest. $n^{\log n}$ grows faster than polynomial functions. $(\log n)^{\sqrt{n}} = \sqrt{n}(\log n)$ grows slowest since it is a sub-linear function.
   From smallest to largest: $n^{\log n}$**,**$(\log n)^{\sqrt{n}}$**,** 2^n

   (c) $n!$, $2^{n \log n}$, $10^{n^{0.5}}$

   $n!$ grows the fastest, even faster than exponential functions. $10^{n^{0.5}}$ is an exponential function with a sub-linear exponent so it grows slower than $n!$. $2^{n \log n}$ have a larger exponent so it grows faster than $10^{n^{0.5}}$.
   From smallest to largest: $10^{n^{0.5}}$**,**$2^{n \log n}$**,**$n!$

   (d) $\binom{n}{d}$, n^d (assume d is a constant)

   $\binom{n}{d}$ can be considered as a polynomial function with a constant exponent. For a small d, binomial coefficient behaves like polynomial with a lower exponent.
   From smallest to largest:$\binom{n}{d}$**,** n^d

   (e) $(\log n)^{\log n}$, $n^{\sqrt{\log n}}$

   Since both parts of $(\log n)^{\log n}$ are logarithmic, it grows pretty slow.
   From smallest to largest:$(\log n)^{\log n}$**,** $n^{\sqrt{\log n}}$

2. Prove that if f, g, h are functions in n and satisfy $f = \Theta(h)$ and $g = \Theta(h)$, then $f + g = \Theta(h)$. Prove or disprove if $f_1, ..., f_n = \Theta(h)$, then $f_1 + ... + f_n = \Theta(h)$.

   Given: $f_1, ..., f_n = \Theta(h)$**,** we have $f_i = \Theta(h)$. There exist constants $c_1, c_2, ...c_n > 0$ that for all i:

   $$c_i^1 < f_i < c_i^2$$

   Prove by induction:

   $$f_1, ..., f_n = \Theta(h)$$

   Base Case: n = 2

   $$f_1 = \Theta(h) \text{ , } f_2 = \Theta(h)$$
   $$f_1 + f_2 = \Theta(h) + \Theta(h) = \Theta(h)$$

   Since 2 is a constant:

   $$f_1 + f_2 = \Theta(h)$$

   Inductive Hypothesis:
   Assume there exists a constant $k > 2$ that $f_1 + ... + f_n = \Theta(h)$.
   Inductive Step:
   We need to show that $f_1 + ... + f_k + f_{k+1} = \Theta(h)$. From the IH, we know that $f_1 + ... + f_n = \Theta(h)$. There exists constants c_{sum}^1 and c_{sum}^2 that:

   $$c_{sum}^1 \Theta(h) < f_1 + f_2 + ... + f_k < c_{sum}^2 \Theta(h)$$

   Add f_{k+1} where $f_{k+1} = \Theta(h)$ and $c_{k+1}^1 < f_{k+1} < c_{k+1}^2$:

   $$(c_{sum}^1 + c_{k+1}^1)\Theta(h) < f_1 + f_2 + ... + f_k + f_{k+1} < (c_{sum}^2 + c_{k+1}^2)\Theta(h)$$

   Since $(c_{sum}^1 + c_{k+1}^1)$ and $(c_{sum}^2 + c_{k+1}^2)$ are both constants, we can conclude that:

   $$f_1 + ... + f_k + f_{k+1} = \Theta(h)$$

   By induction, we prove that if $f_1, ..., f_n = \Theta(h)$, then $f_1 + ... + f_n = \Theta(h)$.

3. Imagine a $(n \geq 1)$. Show that if one of the small square is black, the others are white; then no matter which one in the grid is black, the white part can always be fully covered by L shaped pieces formed by 3 small squares.

   We will prove the statements by induction.
   Base Case: n =1
   When n = 1, the grid will be 2×2. If one of the grids is black, the rest three grids form a L-shaped piece. The base case holds.
   Inductive Hypothesis:
   Assume that there exists a constant $k \geq 2$ that a $2^k \times 2^k$ grid can always be fully covered by L shaped pieces formed by 3 small squares when there is only one black square.
   Inductive Step:

Consider a $2^{k+1} \times 2^{k+1}$ grid that can be divided into four $2^k \times 2^k$ quadrants. When there is one black square in the one of the quadrants, we can put a L-shaped piece in the middle of the grid which covers one grid of three quadrants that do not have the black square. Now, each quadrants only have one square covered.
According to the Inductive Hypothesis that a $2^k \times 2^k$ grid can always be fully covered by L shaped pieces formed by 3 small squares when there is only one black square. Since each $2^k \times 2^k$ quadrant has one square covered, the rest squares can all be covered by L-shaped piece. As a result, the $2^{k+1} \times 2^{k+1}$ grid can be covered by L-shaped piece when there is one black square in it.
Conclusion:
By induction, we proved that a $2^n \times 2^n$ grid with one black square can always be covered with L-shaped piece formed by small white squares.

4. You are given an array A of n elements, and you know there is an element x that appears strictly more than $n/2$ times. Design and analyze an algorithm that outputs this element x in linear time and constant space.

Algorithm 1 Find Majority Element

```
procedure MajorityElement($A$)
    candidate ← None
    count ← 0
    for all $num \in A$ do
        if $count = 0$ then
            candidate ← num
        end if
        if $num = candidate$ then
            count ← count + 1
        else
            count ← count - 1
        end if
    end for
end procedure
```

we will prove the correctness of the algorithm by inducting on the length n of the array.

Base Case: n = 1
when there's only one element in the array, we see that the algorithm sets candidate to such element and exists the loop. Since such element appear more than $\frac{1}{n}$ time, we correctly return the majority element.

Inductive Hypothesis:
Assume that the algorithm correctly finds the majority element for all array with length $\leq k$ where the majority element appears more than $\frac{k}{2}$ times

Inductive Step:

Now we show that for an array with size $k+1$, the algorithm can correctly return the majority element. By definition, we know that the majority element will appear at least $\frac{k+1}{2}+1=\frac{k+3}{2}$ times in the array. WLOG, we assume the majority element appears $\frac{k+3}{2}$ times.

Since the array only has one majority element, we consider two cases for the first element of the array:

Case 1: the First Element is Not the Majority Element:
If the first element is not the majority element than *count* must becomes 0 at some point of the iteration. We assume that count goes back to 0 after x amount of elements have been passed in the array where $x \geq 1$. This means that the majority element can appear at most $\frac{x}{2}$ time. In such case, we notice that there will be $\frac{k+3}{2}-\frac{x}{2}$ majority elements left in the remaining $k+1-x$ array. We see that $\frac{k+3-x}{2} > \frac{k+1-x}{2}$. Thus, by the inductive hypothesis, the algorithm would still correctly identify the majority element within the remaining $k+1-x$ array.

Case 2: the First Element is the Majority Element:
There's two possible scenarios if the first element is the majority element.
Scenario 1: count reduces to 0 at some point
Under this scenario, we see that the problem effectively reduces to case 1 which we have already proven its correctness
Scenario 2: count never reduces to 0
If count never reduces to 0 through out the whole $k+1$ sized array than the *candidate* variable will never point to other element beside the first appeared majority element. Thus, the algorithm will still correctly return the majority element within the $k+1$ array. QED

Run Time Analysis:
Let the length of the array be n.

Loop Behavior:
For each element, the algorithm either updates the candidate or adjusts the count. These operations are all done in constant time.

Since the loop processes each element once and perform constant time operation at each element, the total time complexity is proportional to the number of elements in the array. Thus, the loop runs in linear time, $\mathcal{O}(n)$.

Space Complexity:
The algorithm only uses two variables: `candidate` and `count`. Since these variables do not depend on the size of the array, the space complexity is constant, $\mathcal{O}(1)$.

5. A 4×4 box is filled with integers 1,2,...,15 with the bottom-right corner initially left empty. At any time, there is exactly one empty cell. At each step, you may move to the empty cell one of its adjacent numbers (move up, down, left, right, if there is space). Question: Prove that the second configuration cannot be reached from the first configuration. [0 denotes empty cell]

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	0

(a) First Configuration

2	1	3	4
5	6	7	8
9	10	11	12
13	14	15	0

(b) Second Configuration

To solve this, we will use the concept of **inversions**. An **inversion** in a grid occurs when a higher numbered tile precedes a lower numbered tile in the reading order (from left to right, top to bottom) ignoring the empty space.

Counting Inversions:
We first count the number of inversions in both the initial and final configurations.

(a) Initial configuration: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
(b) Final configuration: 2, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

The first sequence is already sorted, so there are no inversions. Thus, the number of inversions in the initial configuration is 0. In the second sequence tile 2 precedes tile 1, which creates one inversion. Thus, the number of inversions in the final configuration is 1.

Parity of Inversion:
A fundamental property of the 15-puzzle is that the parity (evenness or oddness) of the number of inversions must be preserved after each valid move.

In the initial configuration, we have 0 inversions (even parity). In the final configuration, we have 1 inversion (odd parity).

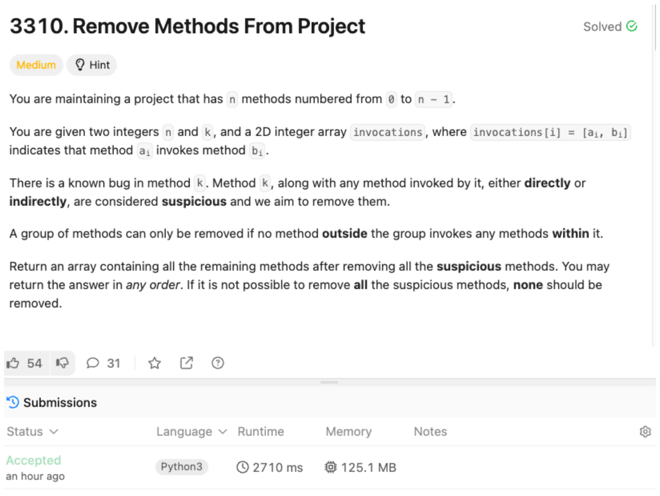
Since the number of inversions changes from even to odd, it is impossible to reach the second configuration from the first configuration through valid moves, as the parity of the number of inversions must remain consistent.

Position of the Empty Cell:
Additionally, we observe that the empty cell (denoted as 0) remains in the same position in both configurations (bottom-right corner). Since the empty cell does not change rows or columns, the relative number of inversions must remain unchanged if a solution were possible. However, the inversion count changes from 0 to 1, which is not allowed.

Conclusion:
Since the parity of the number of inversions is different in the two configurations, it is impossible to reach the second configuration from the first. Therefore, we have proved that the second configuration cannot be reached from the first configuration.

Leetcode Question:

Yuchen:



3310. Remove Methods From Project Solved

Medium · Hint

You are maintaining a project that has `n` methods numbered from `0` to `n - 1`.

You are given two integers `n` and `k`, and a 2D integer array `invocations`, where `invocations[i]` = [a_i, b_i] indicates that method a_i invokes method b_i.

There is a known bug in method `k`. Method `k`, along with any method invoked by it, either **directly** or **indirectly**, are considered **suspicious** and we aim to remove them.

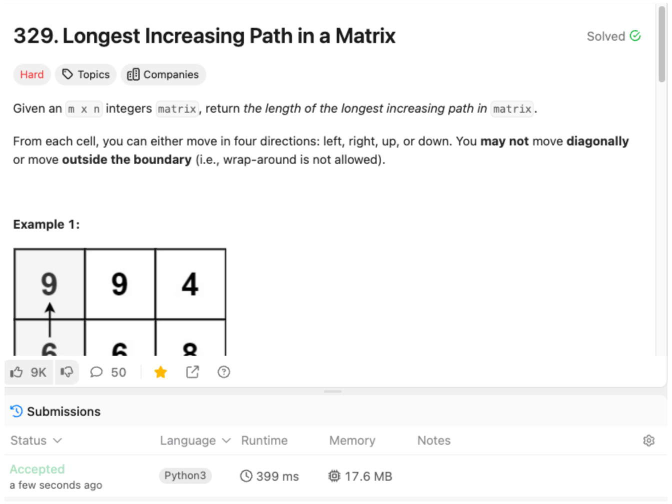
A group of methods can only be removed if no method **outside** the group invokes any methods **within** it.

Return an array containing all the remaining methods after removing all the **suspicious** methods. You may return the answer in *any order*. If it is not possible to remove **all** the suspicious methods, **none** should be removed.

54 · 31

Submissions

Status	Language	Runtime	Memory	Notes
Accepted an hour ago	Python3	2710 ms	125.1 MB	



329. Longest Increasing Path in a Matrix Solved

Hard · Topics · Companies

Given an `m x n` integers `matrix`, return *the length of the longest increasing path in* `matrix`.

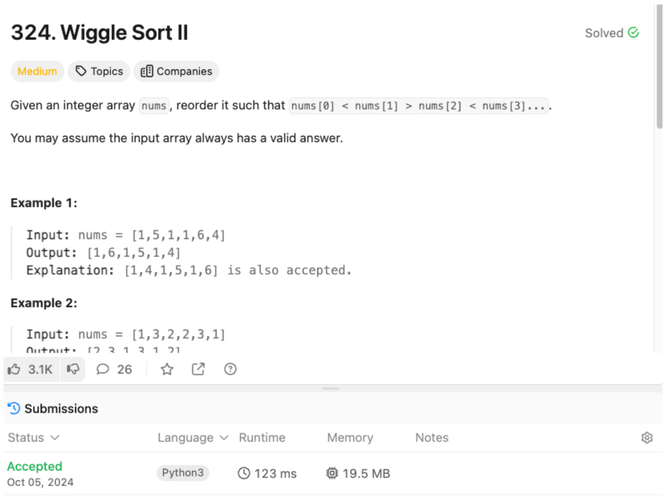
From each cell, you can either move in four directions: left, right, up, or down. You **may not** move **diagonally** or move **outside the boundary** (i.e., wrap-around is not allowed).

Example 1:

9K · 50

Submissions

Status	Language	Runtime	Memory	Notes
Accepted a few seconds ago	Python3	399 ms	17.6 MB	



324. Wiggle Sort II Solved

Medium · Topics · Companies

Given an integer array `nums`, reorder it such that `nums[0] < nums[1] > nums[2] < nums[3]...`.

You may assume the input array always has a valid answer.

Example 1:

```
Input: nums = [1,5,1,1,6,4]
Output: [1,6,1,5,1,4]
Explanation: [1,4,1,5,1,6] is also accepted.
```

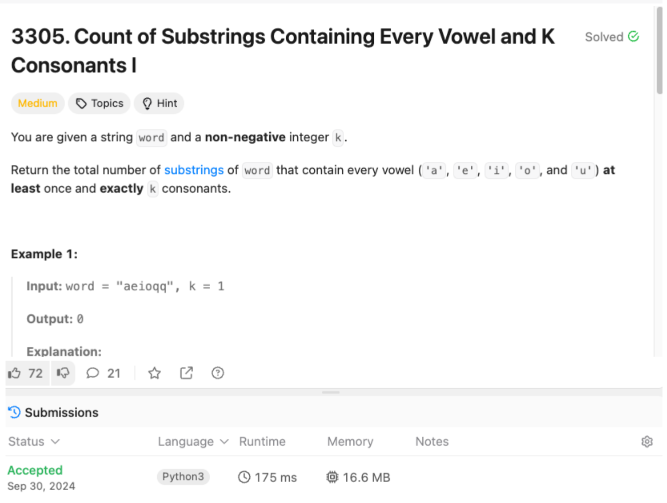
Example 2:

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

3.1K · 26

Submissions

Status	Language	Runtime	Memory	Notes
Accepted Oct 05, 2024	Python3	123 ms	19.5 MB	



3305. Count of Substrings Containing Every Vowel and K Consonants I Solved

Medium · Topics · Hint

You are given a string `word` and a **non-negative** integer `k`.

Return the total number of substrings of `word` that contain every vowel (`'a'`, `'e'`, `'i'`, `'o'`, and `'u'`) **at least** once and **exactly** `k` consonants.

Example 1:

Input: `word = "aeioqq", k = 1`

Output: `0`

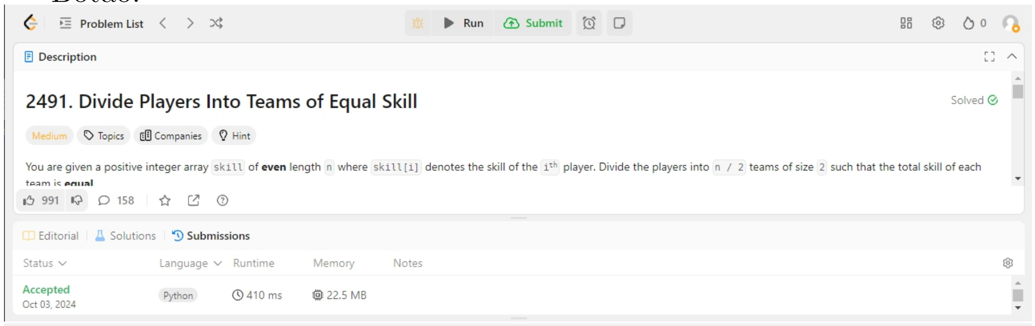
Explanation:

72 · 21

Submissions

Status	Language	Runtime	Memory	Notes
Accepted Sep 30, 2024	Python3	175 ms	16.6 MB	

Botao:



2491. Divide Players Into Teams of Equal Skill

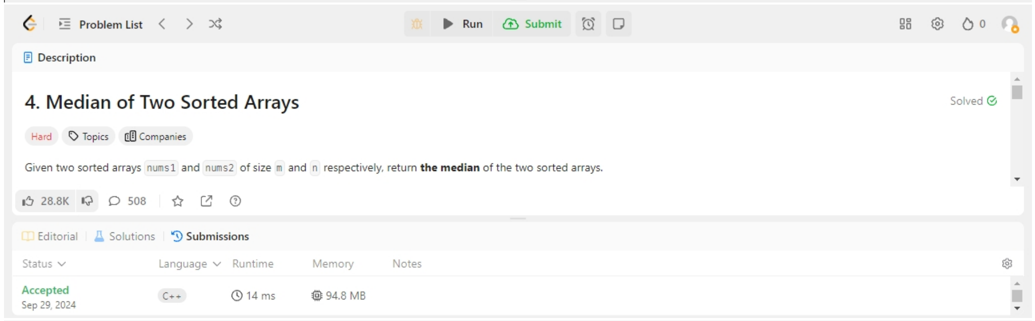
Solved

Medium · Topics · Companies · Hint

You are given a positive integer array `skill` of **even** length `n` where `skill[i]` denotes the skill of the i^{th} player. Divide the players into `n / 2` teams of size `2` such that the total skill of each team is **equal**.

991 · 158

Status	Language	Runtime	Memory	Notes
Accepted Oct 03, 2024	Python	410 ms	22.5 MB	



4. Median of Two Sorted Arrays

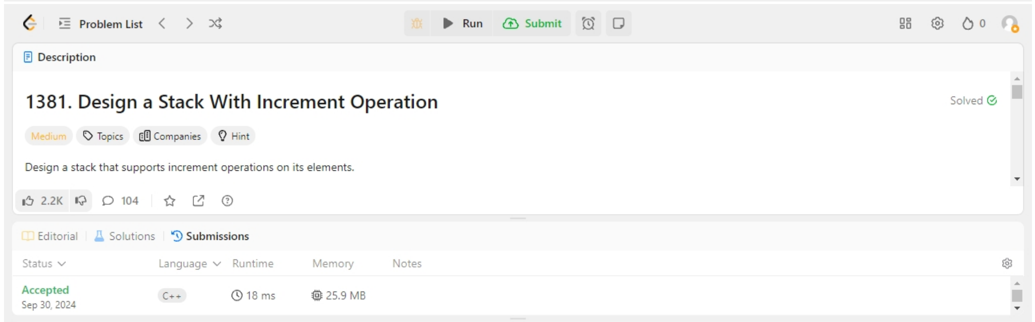
Solved

Hard · Topics · Companies

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return **the median** of the two sorted arrays.

28.8K · 508

Status	Language	Runtime	Memory	Notes
Accepted Sep 29, 2024	C++	14 ms	94.8 MB	



1381. Design a Stack With Increment Operation

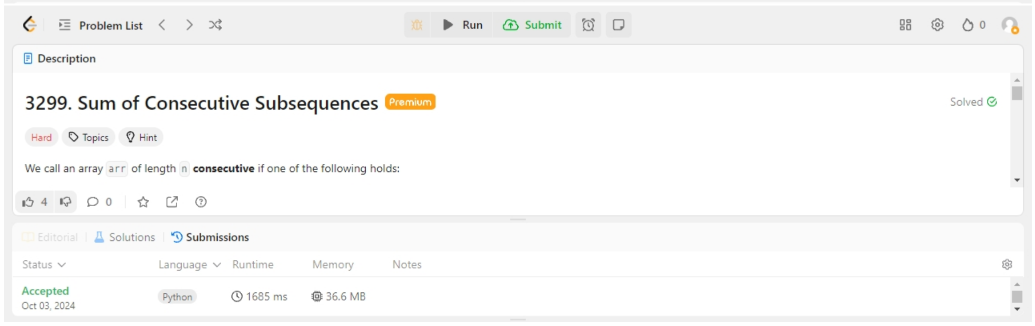
Solved

Medium · Topics · Companies · Hint

Design a stack that supports increment operations on its elements.

2.2K · 104

Status	Language	Runtime	Memory	Notes
Accepted Sep 30, 2024	C++	18 ms	25.9 MB	



3299. Sum of Consecutive Subsequences Premium

Solved

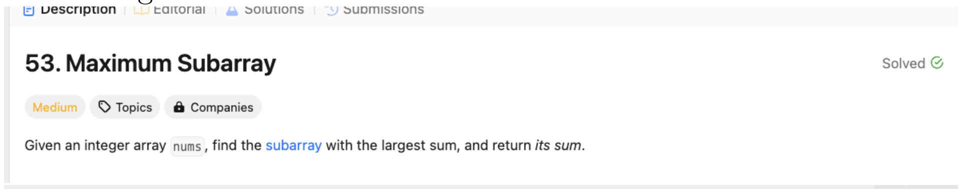
Hard · Topics · Hint

We call an array `arr` of length `n` **consecutive** if one of the following holds:

4 · 0

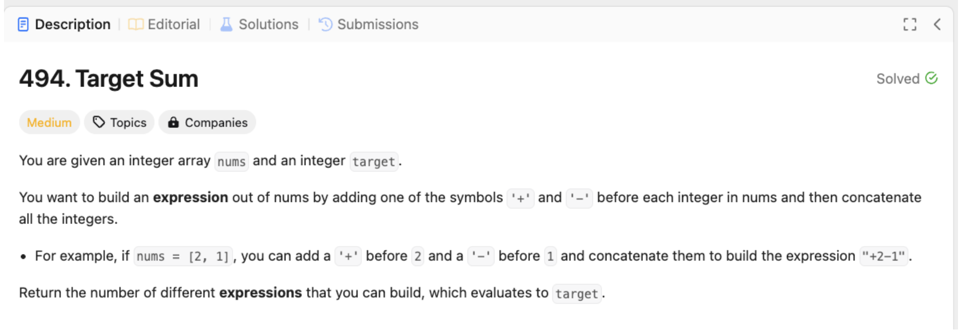
Status	Language	Runtime	Memory	Notes
Accepted Oct 03, 2024	Python	1685 ms	36.6 MB	

Yuheng:



53. Maximum Subarray Solved

Given an integer array `nums`, find the subarray with the largest sum, and return *its sum*.



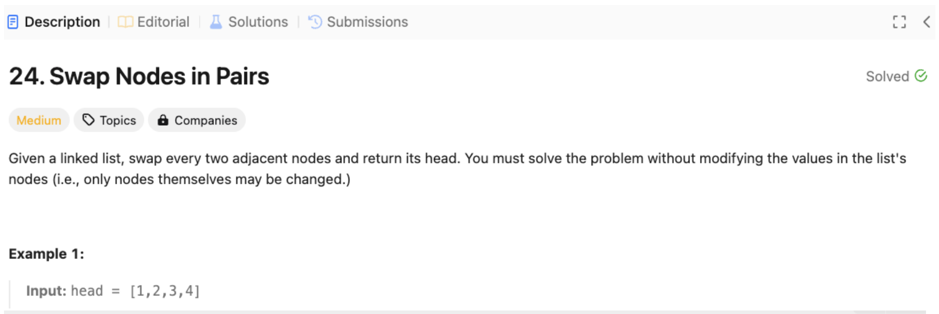
494. Target Sum Solved

You are given an integer array `nums` and an integer `target`.

You want to build an **expression** out of nums by adding one of the symbols `'+'` and `'-'` before each integer in nums and then concatenate all the integers.

- For example, if `nums = [2, 1]`, you can add a `'+'` before `2` and a `'-'` before `1` and concatenate them to build the expression `"+2-1"`.

Return the number of different **expressions** that you can build, which evaluates to `target`.

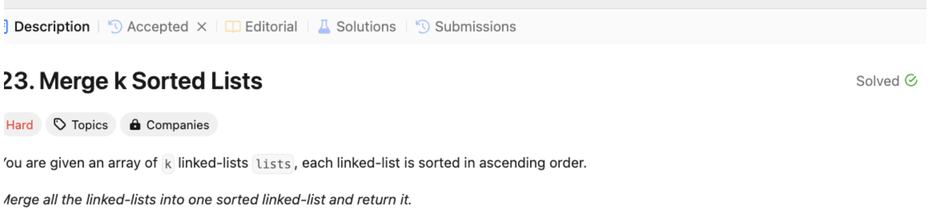


24. Swap Nodes in Pairs Solved

Given a linked list, swap every two adjacent nodes and return its head. You must solve the problem without modifying the values in the list's nodes (i.e., only nodes themselves may be changed.)

Example 1:

Input: `head = [1,2,3,4]`

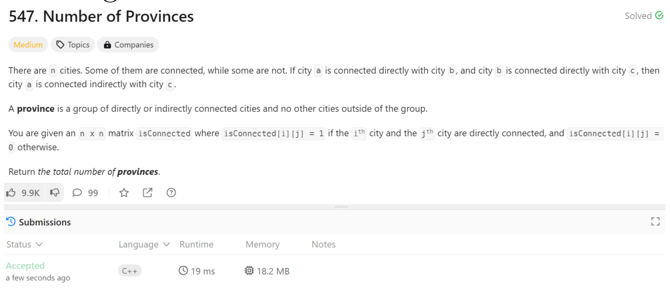

Description | Accepted | Editorial | Solutions | Submissions

23. Merge k Sorted Lists Solved

ou are given an array of `k` linked-lists `lists`, each linked-list is sorted in ascending order.

lerge all the linked-lists into one sorted linked-list and return it.

Peng Yao:


547. Number of Provinces

Solved

Medium Topics Companies

There are `n` cities. Some of them are connected, while some are not. If city `a` is connected directly with city `b`, and city `b` is connected directly with city `c`, then city `a` is connected indirectly with city `c`.

A **province** is a group of directly or indirectly connected cities and no other cities outside of the group.

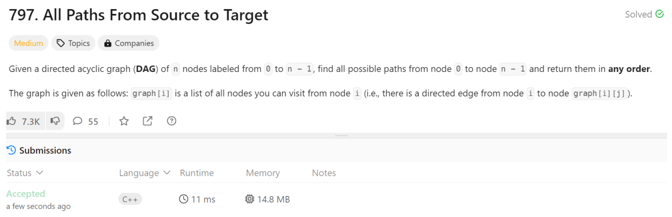
You are given an `n x n` matrix `isConnected` where `isConnected[i][j] = 1` if the ith city and the jth city are directly connected, and `isConnected[i][j] = 0` otherwise.

Return the total number of ***provinces****.*

9.9K 99

Submissions

Status	Language	Runtime	Memory	Notes
Accepted a few seconds ago	C++	19 ms	18.2 MB	


797. All Paths From Source to Target

Solved

Medium Topics Companies

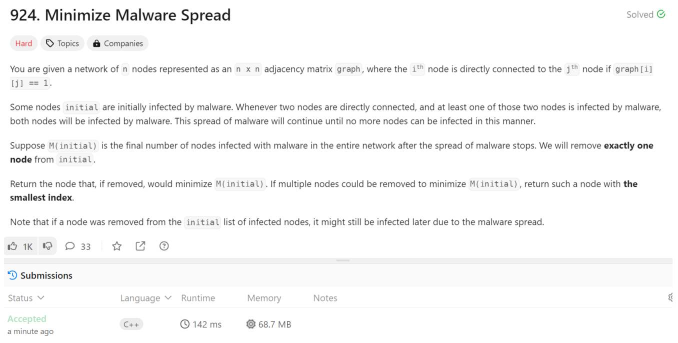
Given a directed acyclic graph (**DAG**) of `n` nodes labeled from `0` to `n - 1`, find all possible paths from node `0` to node `n - 1` and return them in **any order**.

The graph is given as follows: `graph[i]` is a list of all nodes you can visit from node `i` (i.e., there is a directed edge from node `i` to node `graph[i][j]`).

7.3K 55

Submissions

Status	Language	Runtime	Memory	Notes
Accepted a few seconds ago	C++	11 ms	14.8 MB	


924. Minimize Malware Spread

Solved

Hard Topics Companies

You are given a network of `n` nodes represented as an `n x n` adjacency matrix `graph`, where the ith node is directly connected to the jth 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`.

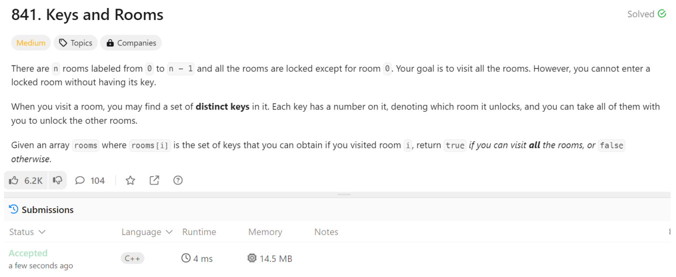
Return the node that, if removed, would minimize `M(initial)`. If multiple nodes could be removed to minimize `M(initial)`, return such a node with **the smallest index**.

Note that if a node was removed from the `initial` list of infected nodes, it might still be infected later due to the malware spread.

1K 33

Submissions

Status	Language	Runtime	Memory	Notes
Accepted a minute ago	C++	142 ms	68.7 MB	


841. Keys and Rooms

Solved

Medium Topics Companies

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.

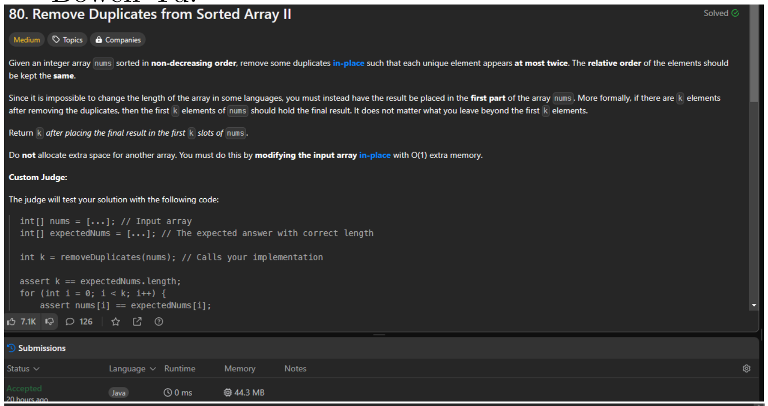
Given an array `rooms` where `rooms[i]` is the set of keys that you can obtain if you visited room `i`, return `true` *if you can visit* ***all*** *the rooms, or* `false` *otherwise.*

6.2K 104

Submissions

Status	Language	Runtime	Memory	Notes
Accepted a few seconds ago	C++	4 ms	14.5 MB	

Bowen Yu:



80. Remove Duplicates from Sorted Array II

Solved

Medium · Topics · Companies

Given an integer array `nums` sorted in **non-decreasing order**, remove some duplicates in-place such that each unique element appears **at most twice**. The **relative order** of the elements should be kept the **same**.

Since it is impossible to change the length of the array in some languages, you must instead have the result be placed in the **first part** of the array `nums`. More formally, if there are `k` elements after removing the duplicates, then the first `k` elements of `nums` should hold the final result. It does not matter what you leave beyond the first `k` elements.

Return `k` *after placing the final result in the first* `k` *slots of* `nums`.

Do **not** allocate extra space for another array. You must do this by **modifying the input array** in-place with O(1) extra memory.

Custom Judge:

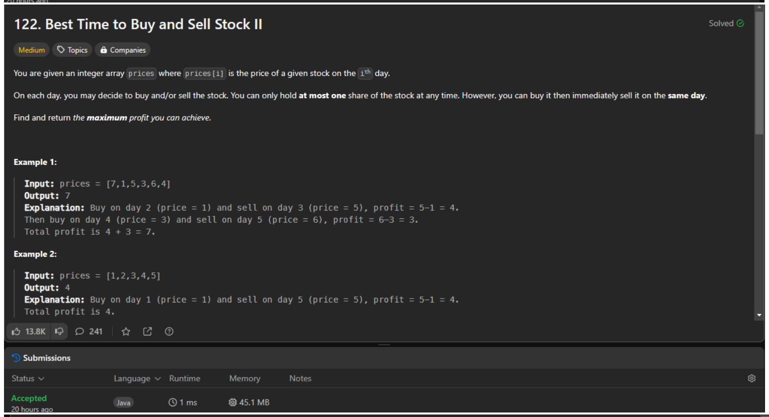
The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length

int k = removeDuplicates(nums); // Calls your implementation

assert k == expectedNums.length;
for (int i = 0; i < k; i++) {
    assert nums[i] == expectedNums[i];
```

7.1K · 126

Submissions

Status	Language	Runtime	Memory	Notes
Accepted 20 hours ago	Java	0 ms	44.3 MB	



122. Best Time to Buy and Sell Stock II

Solved

Medium · Topics · Companies

You are given an integer array `prices` where `prices[i]` is the price of a given stock on the `i`th day.

On each day, you may decide to buy and/or sell the stock. You can only hold **at most one** share of the stock at any time. However, you can buy it then immediately sell it on the **same day**.

Find and return *the **maximum** profit you can achieve.*

Example 1:

```
Input: prices = [7,1,5,3,6,4]
Output: 7
Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4.
Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.
Total profit is 4 + 3 = 7.
```

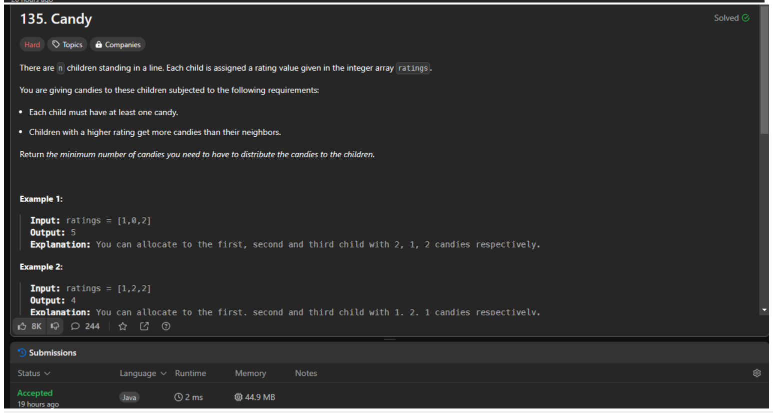
Example 2:

```
Input: prices = [1,2,3,4,5]
Output: 4
Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.
Total profit is 4.
```

13.8K · 241

Submissions

Status	Language	Runtime	Memory	Notes
Accepted 20 hours ago	Java	1 ms	45.1 MB	



135. Candy

Solved

Hard · Topics · Companies

There are `n` children standing in a line. Each child is assigned a rating value given in the integer array `ratings`.

You are giving candies to these children subjected to the following requirements:

- Each child must have at least one candy.
- Children with a higher rating get more candies than their neighbors.

Return *the minimum number of candies you need to have to distribute the candies to the children.*

Example 1:

```
Input: ratings = [1,0,2]
Output: 5
Explanation: You can allocate to the first, second and third child with 2, 1, 2 candies respectively.
```

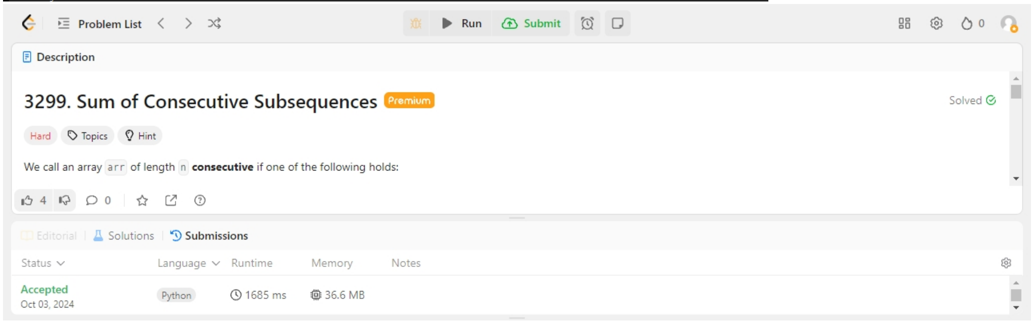
Example 2:

```
Input: ratings = [1,2,2]
Output: 4
Explanation: You can allocate to the first, second and third child with 1, 2, 1 candies respectively.
```

8K · 244

Submissions

Status	Language	Runtime	Memory	Notes
Accepted 19 hours ago	Java	2 ms	44.9 MB	



Description

3299. Sum of Consecutive Subsequences

Premium

Solved

Hard · Topics · Hint

We call an array `arr` of length `n` **consecutive** if one of the following holds:

4 · 0

Editorial · Solutions · Submissions

Status	Language	Runtime	Memory	Notes
Accepted Oct 03, 2024	Python	1685 ms	36.6 MB	