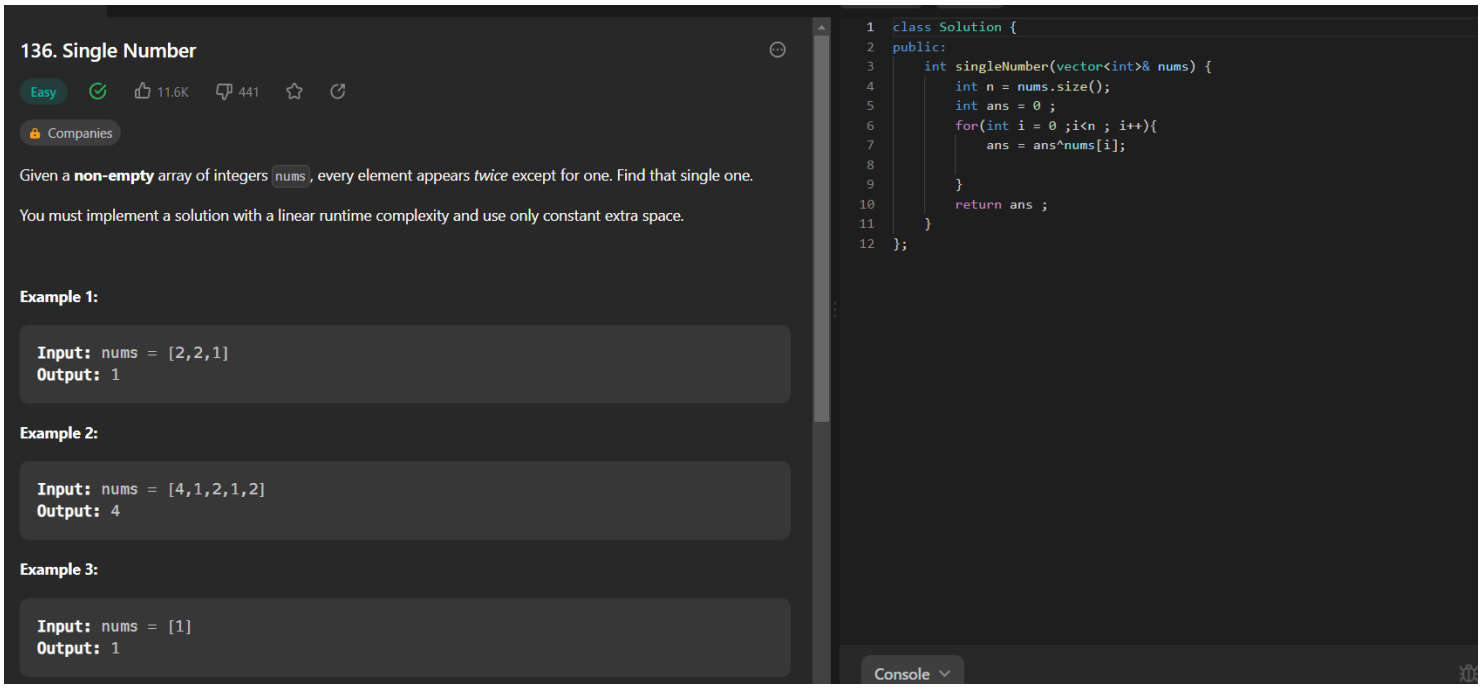
DSA Questions

Array

Q . Find Unique Number (single Number)



136. Single Number

Easy 11.6K 441

Companies

Given a **non-empty** array of integers `nums`, every element appears *twice* except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:

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

Example 2:

```
Input: nums = [4,1,2,1,2]
Output: 4
```

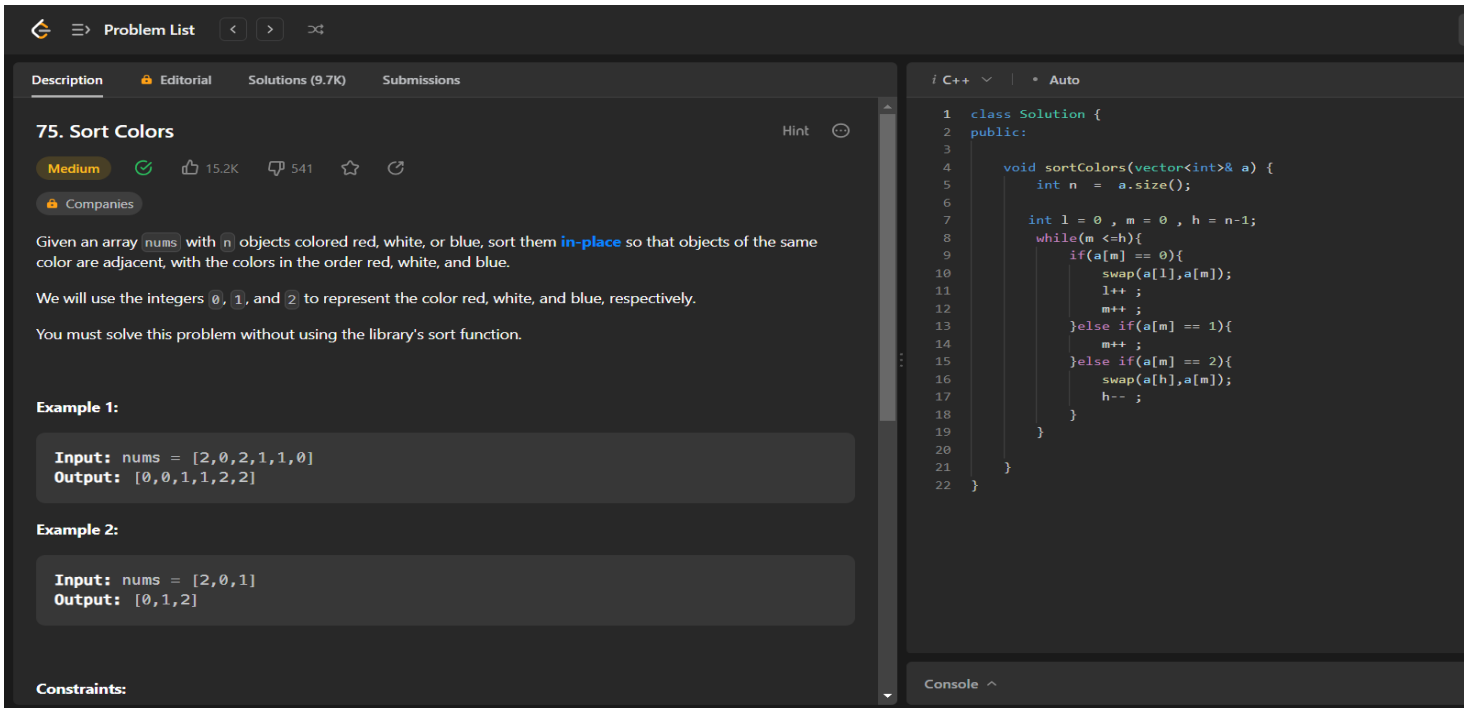
Example 3:

```
Input: nums = [1]
Output: 1
```

```cpp
class Solution {
public:
    int singleNumber(vector<int>& nums) {
        int n = nums.size();
        int ans = 0 ;
        for(int i = 0 ;i<n ; i++){
            ans = ans^nums[i];

        }
        return ans ;
    }
};
```

Console

Q. Sort Colors



Problem List

Description Editorial Solutions (9.7K) Submissions

75. Sort Colors

Hint

Medium 15.2K 541

Companies

Given an array `nums` with `n` objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers `0`, `1`, and `2` to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:

```
Input: nums = [2,0,2,1,1,0]
Output: [0,0,1,1,2,2]
```

Example 2:

```
Input: nums = [2,0,1]
Output: [0,1,2]
```

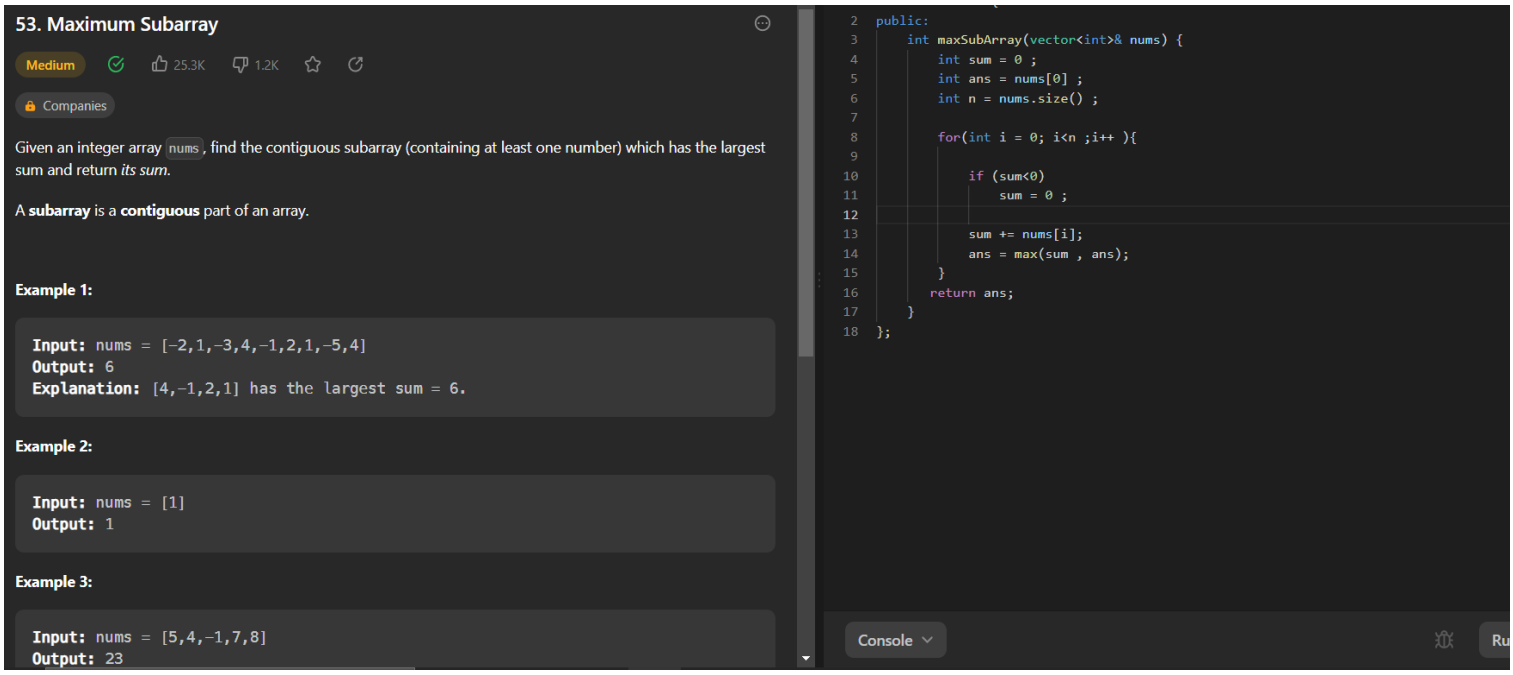
Constraints:

C++ Auto

```cpp
class Solution {
public:

    void sortColors(vector<int>& a) {
        int n  =  a.size();

       int l = 0 , m = 0 , h = n-1;
        while(m <=h){
            if(a[m] == 0){
                swap(a[l],a[m]);
                l++ ;
                m++ ;
            }else if(a[m] == 1){
                m++ ;
            }else if(a[m] == 2){
                swap(a[h],a[m]);
                h-- ;
            }
        }

    }
}
```

Console

Q. Maximum sum of Subarray (Kadane's Algorithm)



53. Maximum Subarray

Medium 25.3K 1.2K

Companies

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum and return *its sum*.

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

Example 1:

```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
```

Example 2:

```
Input: nums = [1]
Output: 1
```

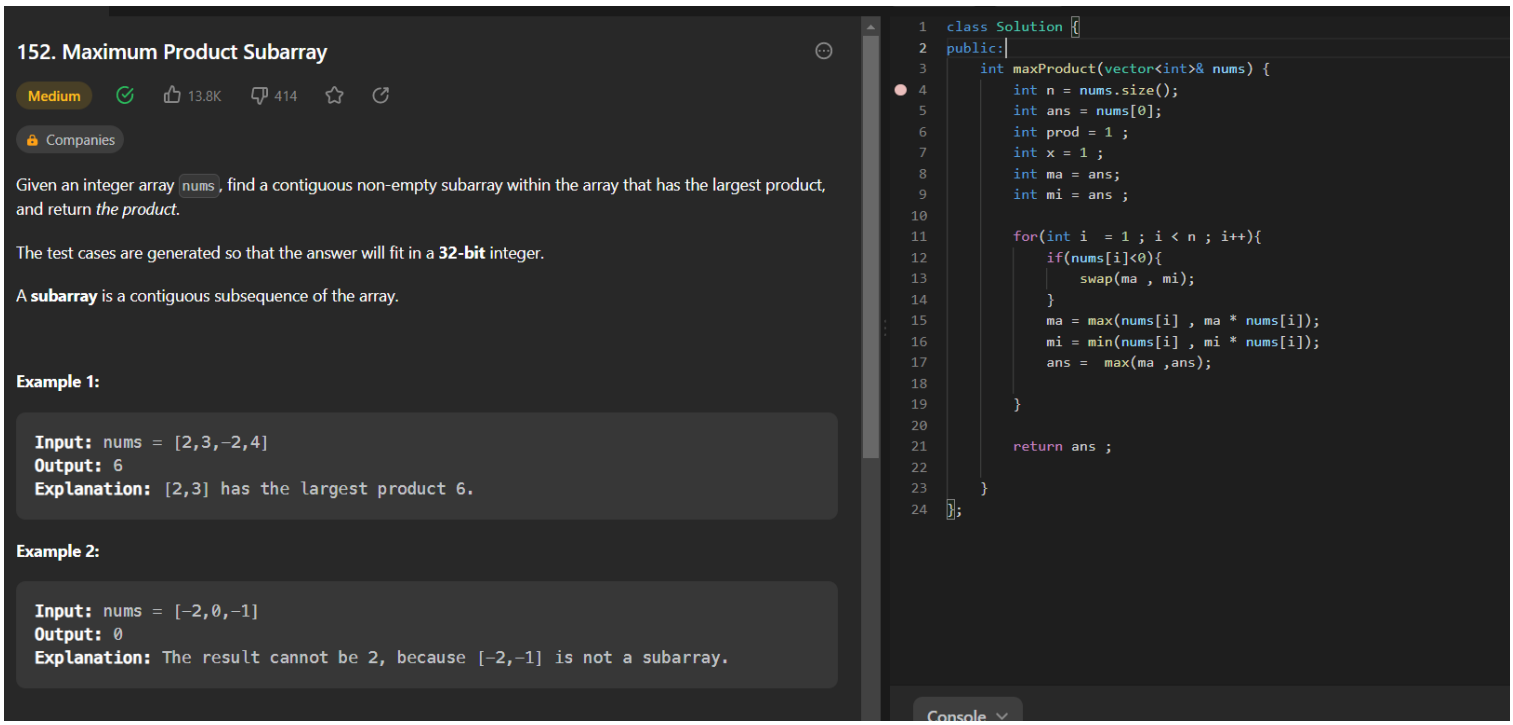
Example 3:

```
Input: nums = [5,4,-1,7,8]
Output: 23
```

```cpp
public:
    int maxSubArray(vector<int>& nums) {
        int sum = 0 ;
        int ans = nums[0] ;
        int n = nums.size() ;

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

            if (sum<0)
                sum = 0 ;

            sum += nums[i];
            ans = max(sum , ans);
        }
       return ans;
    }
};
```

Console

Q. Maximum Product Subarray



152. Maximum Product Subarray

Medium 13.8K 414

Companies

Given an integer array `nums`, find a contiguous non-empty subarray within the array that has the largest product, and return *the product*.

The test cases are generated so that the answer will fit in a **32-bit** integer.

A **subarray** is a contiguous subsequence of the array.

Example 1:

```
Input: nums = [2,3,-2,4]
Output: 6
Explanation: [2,3] has the largest product 6.
```

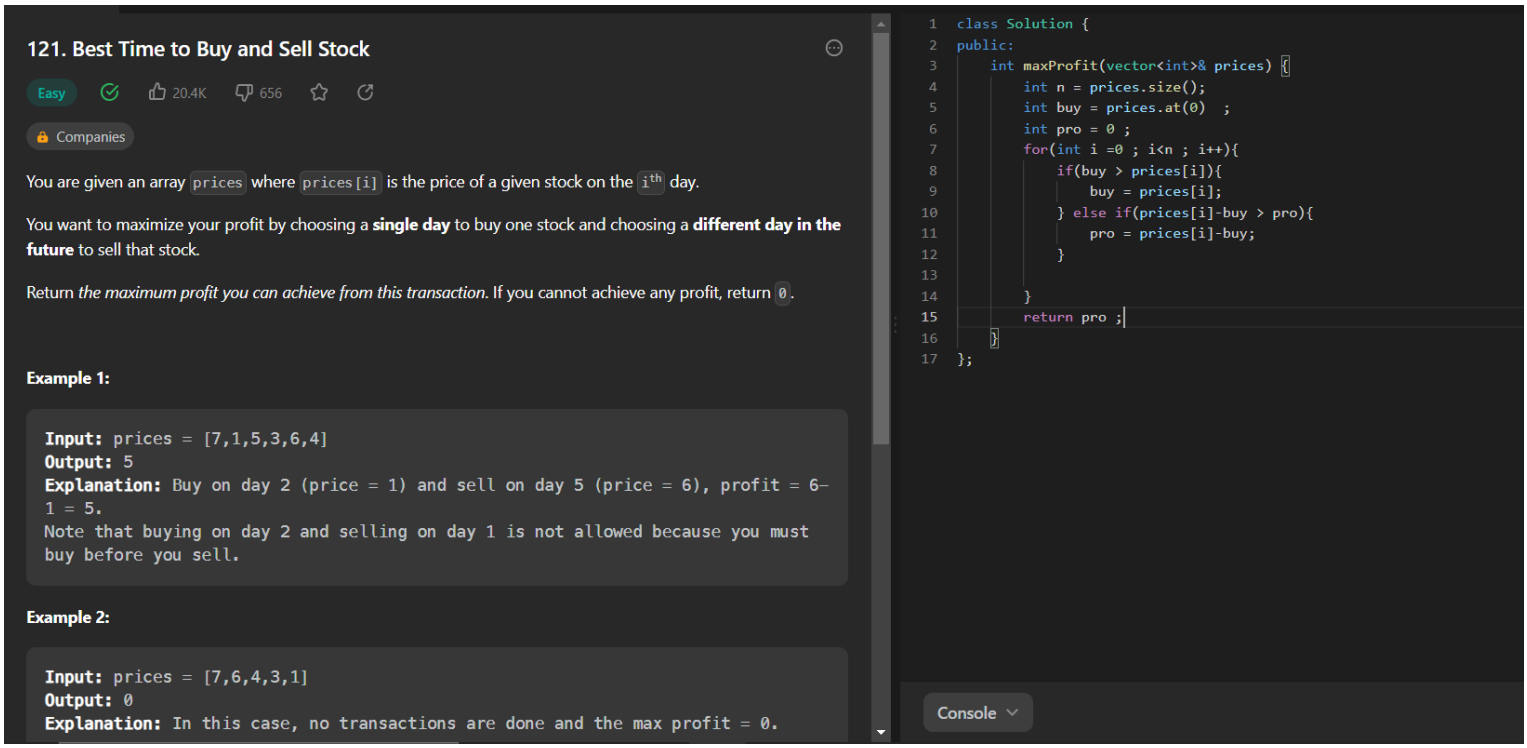
Example 2:

```
Input: nums = [-2,0,-1]
Output: 0
Explanation: The result cannot be 2, because [-2,-1] is not a subarray.
```

```cpp
class Solution {
public:
    int maxProduct(vector<int>& nums) {
        int n = nums.size();
        int ans = nums[0];
        int prod = 1 ;
        int x = 1 ;
        int ma = ans;
        int mi = ans ;

        for(int i  = 1 ; i < n ; i++){
            if(nums[i]<0){
                swap(ma , mi);
            }
            ma = max(nums[i] , ma * nums[i]);
            mi = min(nums[i] , mi * nums[i]);
            ans =  max(ma ,ans);

        }

        return ans ;

    }
};
```

Console

Q. Best Time to Buy and Sell Stock I



121. Best Time to Buy and Sell Stock

Easy 20.4K 656

Companies

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

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return `0`.

Example 1:

```
Input: prices = [7,1,5,3,6,4]
Output: 5
Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.
```

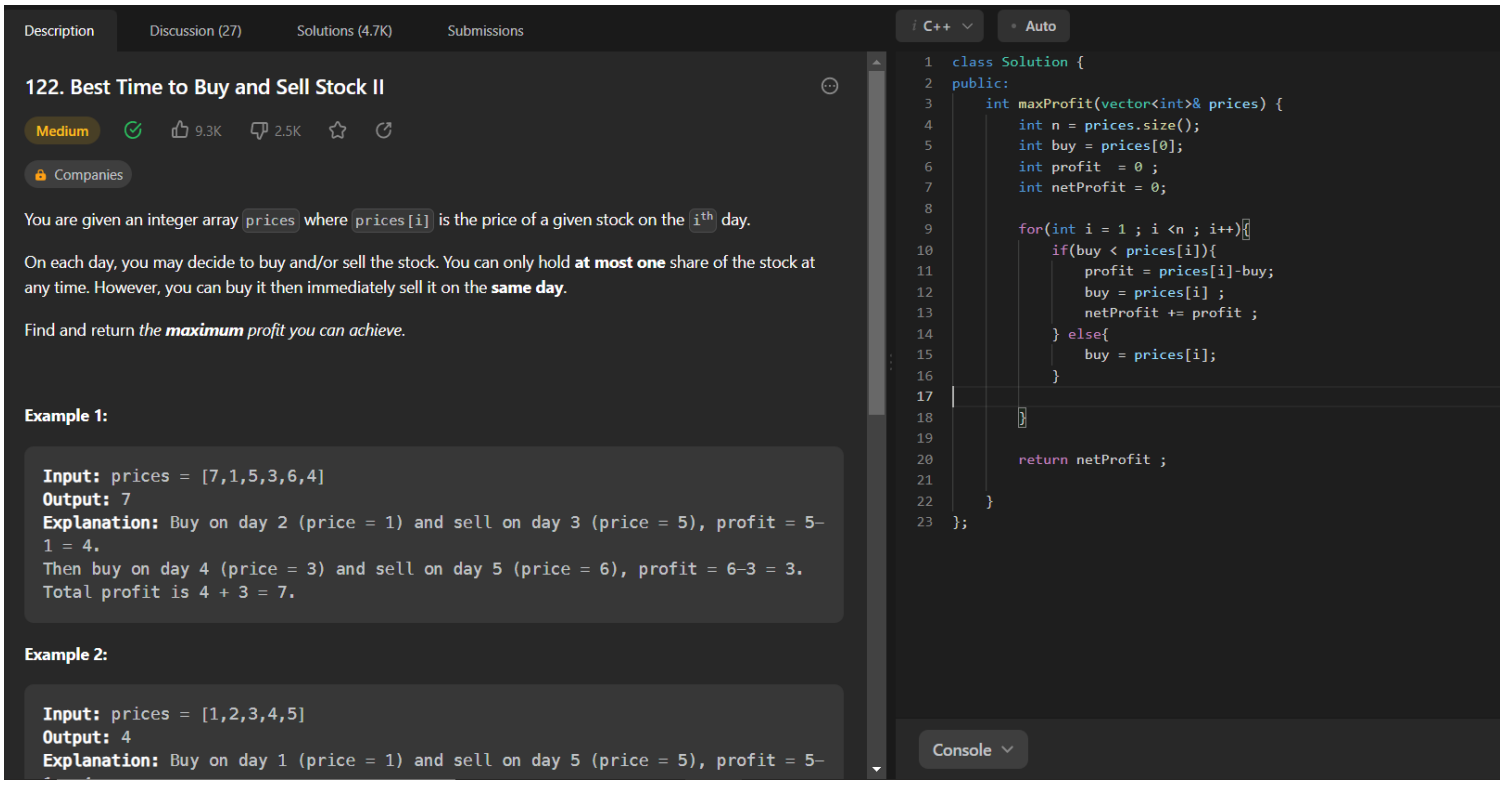
Example 2:

```
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transactions are done and the max profit = 0.
```

```cpp
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int n = prices.size();
        int buy = prices.at(0)  ;
        int pro = 0 ;
        for(int i =0 ; i<n ; i++){
            if(buy > prices[i]){
                buy = prices[i];
            } else if(prices[i]-buy > pro){
                pro = prices[i]-buy;
            }

        }
        return pro ;
    }
};
```

Console

Q. Best Time to Buy and Sell Stock II



Description | Discussion (27) | Solutions (4.7K) | Submissions

C++ | Auto

122. Best Time to Buy and Sell Stock II

Medium 9.3K 2.5K

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-
```

```cpp
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int n = prices.size();
        int buy = prices[0];
        int profit  = 0 ;
        int netProfit = 0;

        for(int i = 1 ; i <n ; i++){
            if(buy < prices[i]){
                profit = prices[i]-buy;
                buy = prices[i] ;
                netProfit += profit ;
            } else{
                buy = prices[i];
            }

        }

        return netProfit ;

    }
};
```

Console

Q. Container with Most Water

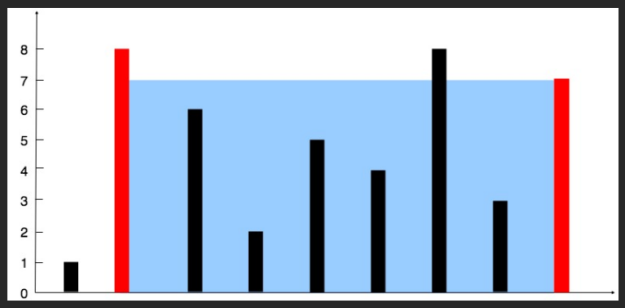
You are given an integer array `height` of length `n`. There are `n` vertical lines drawn such that the two endpoints of the `i^th` line are `(i, 0)` and `(i, height[i])`.

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return *the maximum amount of water a container can store.*

Notice that you may not slant the container.

Example 1:




```
Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49
Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.
```

```cpp
class Solution {
public:
    int maxArea(vector<int>& height) {
        int l = height.size() -1 ;
        int area  ;
        int b = 0;
        int maxA = 0 ;

        int i =0 ;

        while(i<l){
            b = min (height[i], height[l]);
                area = (l-i) * b ;
            maxA = max (maxA , area);

            if(height[i]<height[l]){
                i++ ;
            }else{
                l-- ;
            }
        }
        return maxA;
    }
};
```

Console Run Subr

Q. Set Matrix Zero

73. Set Matrix Zeroes

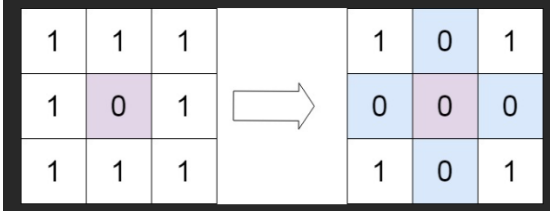
Medium 9.3K 555

Companies

Given an `m x n` integer matrix `matrix`, if an element is `0`, set its entire row and column to `0`'s.

You must do it in place.

Example 1:




```
Input: matrix = [[1,1,1],[1,0,1],[1,1,1]]
Output: [[1,0,1],[0,0,0],[1,0,1]]
```

Example 2:

```cpp
public:
    void setZeroes(vector<vector<int>>& matrix) {
        int r = matrix.size();
        int c = matrix[0].size();
        set<int> row ;
        set<int> column ;

        for(int i = 0; i < r; i++){
            for(int j = 0; j < c; j++){
                if (matrix[i][j] == 0){
                    row.insert(i);
                    column.insert(j);
                }

            }
        }

        for(auto i : row){
            for(int j = 0; j < c; j++){
                matrix[i][j]= 0;
            }
        }
        for(auto i : column){
            for(int j = 0; j < r; j++){
                matrix[j][i]= 0;
            }
        }
    }
};
```

Q. Pascal’s Triangle I

118. Pascal's Triangle

Easy 8K 266

Companies

Given an integer `numRows`, return the first numRows of **Pascal's triangle**.

In **Pascal's triangle**, each number is the sum of the two numbers directly above it as shown:

Example 1:

```
Input: numRows = 5
Output: [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]
```

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

        for(int i = 0 ; i<numRows ; i++){
            ans[i]= vector<int>(i+1);
            for (int j = 0 ; j<i+1; j++){
                if(j>0 && j < i){
                    ans[i][j] = ans[i-1][j-1] + ans[i-1][j];
                }else{
                    ans[i][j] = 1;
                }
            }
        }
        return ans ;
    }
};
```

Q. Pascal’s Triangle II

119. Pascal's Triangle II

Easy 3.2K 281

Companies

Given an integer `rowIndex`, return the `rowIndex`th (**0-indexed**) row of the **Pascal's triangle**.

In **Pascal's triangle**, each number is the sum of the two numbers directly above it as shown:

Example 1:

```
Input: rowIndex = 3
Output: [1,3,3,1]
```

```java
class Solution {
public:
    vector<int> getRow(int rowIndex) {
        vector<vector<int>> ans (rowIndex+1) ;

        for(int i = 0 ; i<=rowIndex ; i++){
            ans[i]= vector<int>(i+1);
            for (int j = 0 ; j<i+1; j++){
                if(j>0 && j < i){
                    ans[i][j] = ans[i-1][j-1] + ans[i-1][j];
                }else{
                    ans[i][j] = 1;
                }
            }
        }

        return ans[rowIndex] ;
    }
};
```

Q . Spiral Matrix

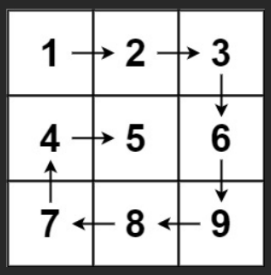
54. Spiral Matrix

Medium 9.1K 925

Companies

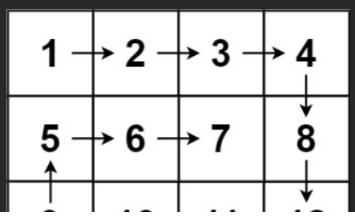
Given an `m x n` `matrix`, return *all elements of the* `matrix` *in spiral order.*

Example 1:




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

Example 2:




```cpp
public:
    vector<int> spiralOrder(vector<vector<int>>& matrix) {
        int row = matrix.size();
        int col = matrix[0].size();
        vector<int> ans ;

        int l = 0 , r = col-1 , t = 0 , b = row -1 , d = 0;

        while (l<=r && t<=b){
            switch(d){
                case 0 :
                    for(int i = l ; i<= r ; i++){
                        ans.push_back(matrix[t][i]);
                    }
                    t++ ;
                    d = 1;
                    break;
                case 1 :
                    for(int i = t ; i<= b ; i++){
                        ans.push_back(matrix[i][r]);
                    }
                    r--;
                    d = 2;
                    break;
                case 2 :
                    for(int i = r ; i>= l ; i--){
                        ans.push_back(matrix[b][i]);
                    }
                    b--;
                    d = 3;
                    break;
                case 3 :
                    for(int i = b ; i>= t ; i--){
                        ans.push_back(matrix[i][l]);
                    }
                    l++;
                    d = 0 ;
                    break ;
            }
        }
        return ans ;
```

Console Run

Strings

Searching And Sorting

Q. Merge Sort

```
void merge(int arr[], int l, int m, int r)
{
   // Your code here
   int len1 = m - l + 1 ;
   int len2 = r-m ;
   int first[len1] ;
   int second [len2] ;

   int k = l ;
   //   cout<<"first " ;
   for(int i = 0 ; i<len1 ; i++){
      first[i] = arr[k++] ;
      //  cout<<first[i]<<" " ;
   }
   //  cout<<endl ;
   k = m + 1 ;
   //  cout<<"second " ;
   for(int i = 0 ; i<len2 ; i++){
      second[i] = arr[k++] ;
      //  cout<<second[i]<<" " ;
   }
   //  cout<<endl ;

   //  merge two sorted array
   int index1 = 0 ;
   int index2 = 0 ;
   k=l ;
   while(index1 < len1 && index2 <len2){
      if(first[index1]<=second[index2]){
         arr[k++] =  first[index1++] ;
      }else{
         arr[k++] =  second[index2++] ;
      }
   }

   while(index1 < len1){
      arr[k++] =  first[index1++] ;
```

```
    }
    while(index2 < len2){
        arr[k++] =  second[index2++] ;
    }
}

void mergeSort(int arr[], int l, int r)
{
    //code here
    if(l>=r){
        return ;
    }
    int mid = l + (r-l)/2 ;
    // cout<<l <<r <<endl;

    mergeSort(arr , l , mid) ;
    mergeSort(arr , mid+1 , r) ;

    merge(arr , l , mid , r) ;

    }
```

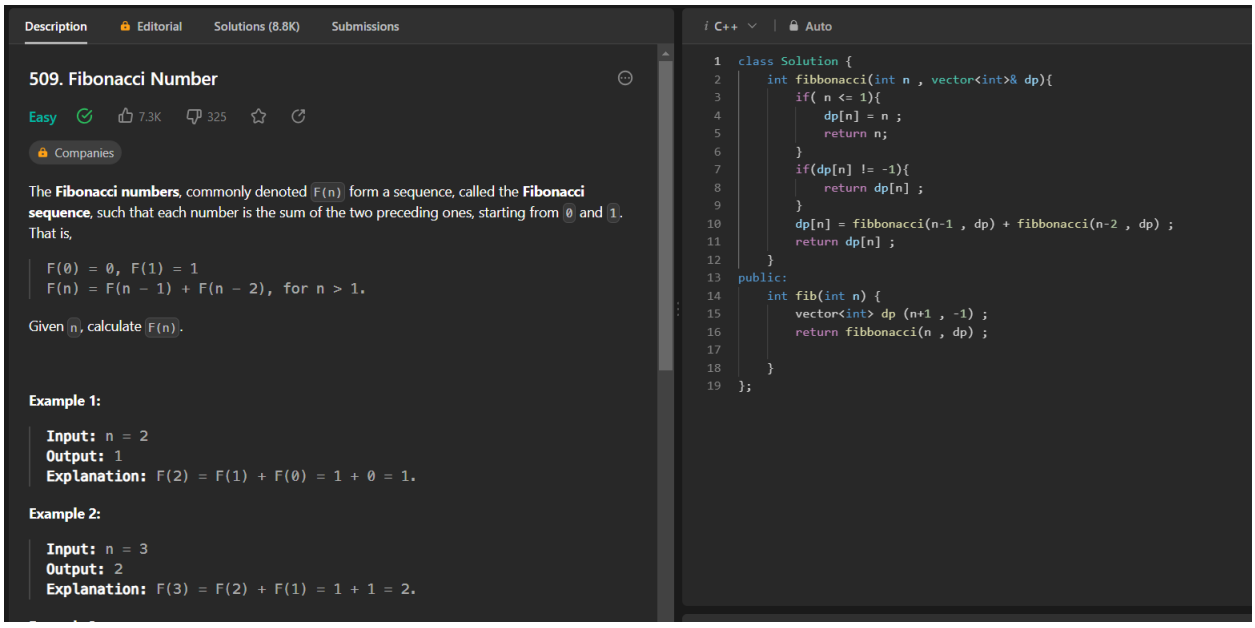
Q. Quick sort

```
int partition (int arr[], int low, int high)
{
    // Your code here

    int pivot = arr[low] ;

    int cnt = 0 ;
    for(int i = low+1  ; i<=high ; i++){
        if(pivot >= arr[i]){
            cnt++ ;
        }
    }
    int pivotindex = low + cnt ;

    swap(arr[pivotindex] , arr[low]);

    //  correct  left or right part

    int i = low , j = high ;
```

```
    while(i<pivotindex && j>pivotindex){
        while(arr[i]<pivot){
            i++ ;
        }
        while(arr[j]> pivot){
            j-- ;
        }

        if(i<pivotindex && j>pivotindex){
            swap(arr[i++] , arr[j--]) ;
        }
    }

    return pivotindex ;
}

void quickSort(int arr[], int low, int high)
{
    // code here
    if(low>= high)
        return ;

    int p = partition(arr , low , high);

    quickSort(arr , low , p-1) ;

    quickSort(arr , p+1 , high) ;

}
```

Recursion

Q. Fibonacci Number (509)



Description Editorial Solutions (8.8K) Submissions

509. Fibonacci Number

Easy 7.3K 325

Companies

The **Fibonacci numbers**, commonly denoted `F(n)` form a sequence, called the **Fibonacci sequence**, such that each number is the sum of the two preceding ones, starting from `0` and `1`. That is,

```
F(0) = 0, F(1) = 1
F(n) = F(n - 1) + F(n - 2), for n > 1.
```

Given `n`, calculate `F(n)`.

Example 1:

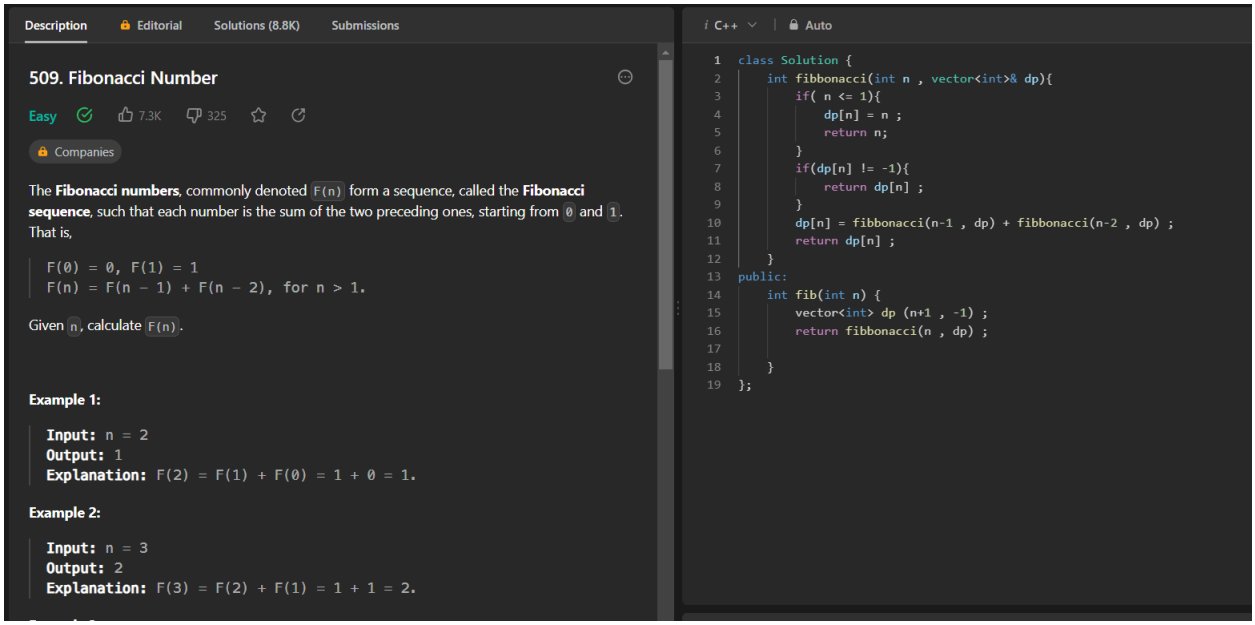
```
Input: n = 2
Output: 1
Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.
```

Example 2:

```
Input: n = 3
Output: 2
Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.
```

C++ Auto

```cpp
class Solution {
    int fibbonacci(int n , vector<int>& dp){
        if( n <= 1){
            dp[n] = n ;
            return n;
        }
        if(dp[n] != -1){
            return dp[n] ;
        }
        dp[n] = fibbonacci(n-1 , dp) + fibbonacci(n-2 , dp) ;
        return dp[n] ;
    }
public:
    int fib(int n) {
        vector<int> dp (n+1 , -1) ;
        return fibbonacci(n , dp) ;

    }
};
```



Description Editorial Solutions (8.8K) Submissions

509. Fibonacci Number

Easy 7.3K 325

Companies

The **Fibonacci numbers**, commonly denoted `F(n)` form a sequence, called the **Fibonacci sequence**, such that each number is the sum of the two preceding ones, starting from `0` and `1`. That is,

```
F(0) = 0, F(1) = 1
F(n) = F(n - 1) + F(n - 2), for n > 1.
```

Given `n`, calculate `F(n)`.

Example 1:

```
Input: n = 2
Output: 1
Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.
```

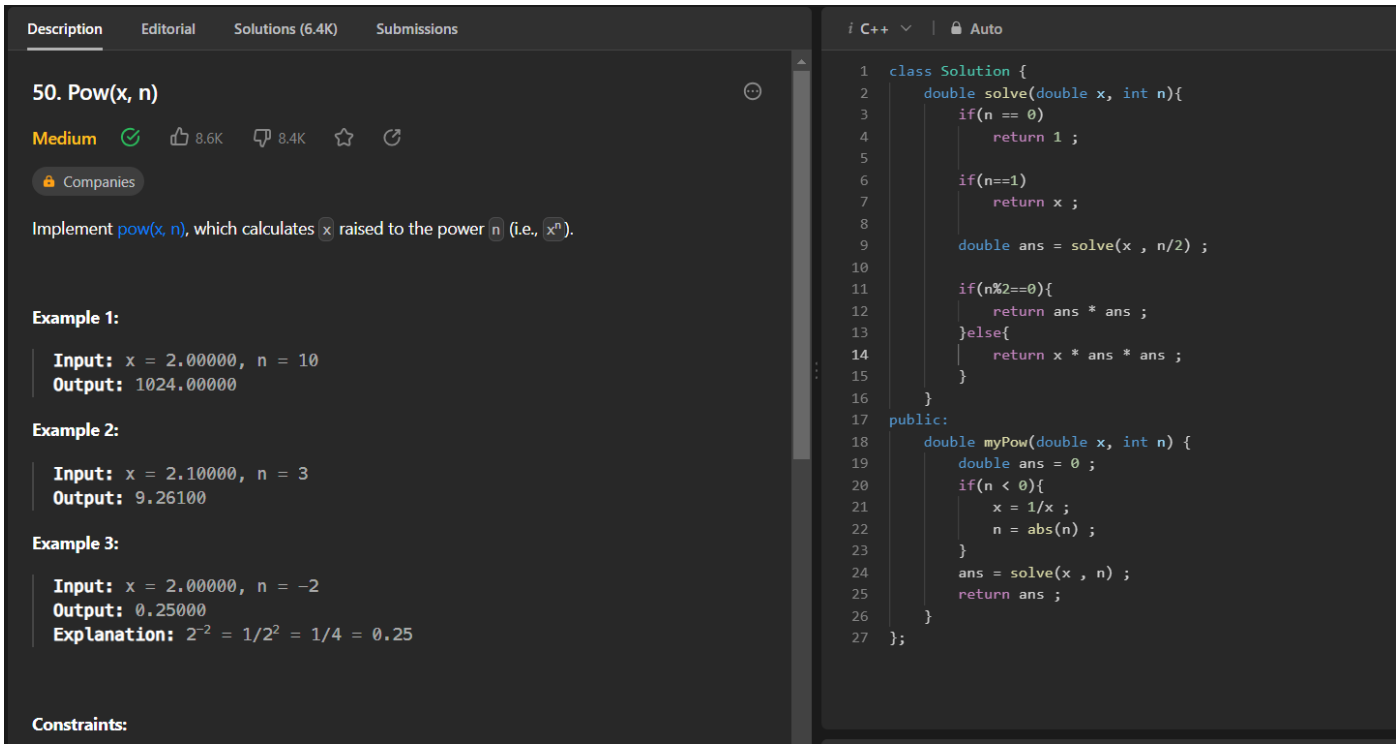
Example 2:

```
Input: n = 3
Output: 2
Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.
```

C++ Auto

```cpp
class Solution {
    int fibbonacci(int n , vector<int>& dp){
        if( n <= 1){
            dp[n] = n ;
            return n;
        }
        if(dp[n] != -1){
            return dp[n] ;
        }
        dp[n] = fibbonacci(n-1 , dp) + fibbonacci(n-2 , dp) ;
        return dp[n] ;
    }
public:
    int fib(int n) {
        vector<int> dp (n+1 , -1) ;
        return fibbonacci(n , dp) ;

    }
};
```

Q. Pow(x, n)



Description Editorial Solutions (6.4K) Submissions

50. Pow(x, n)

Medium 8.6K 8.4K

Companies

Implement pow(x, n), which calculates `x` raised to the power `n` (i.e., x^n).

Example 1:

```
Input: x = 2.00000, n = 10
Output: 1024.00000
```

Example 2:

```
Input: x = 2.10000, n = 3
Output: 9.26100
```

Example 3:

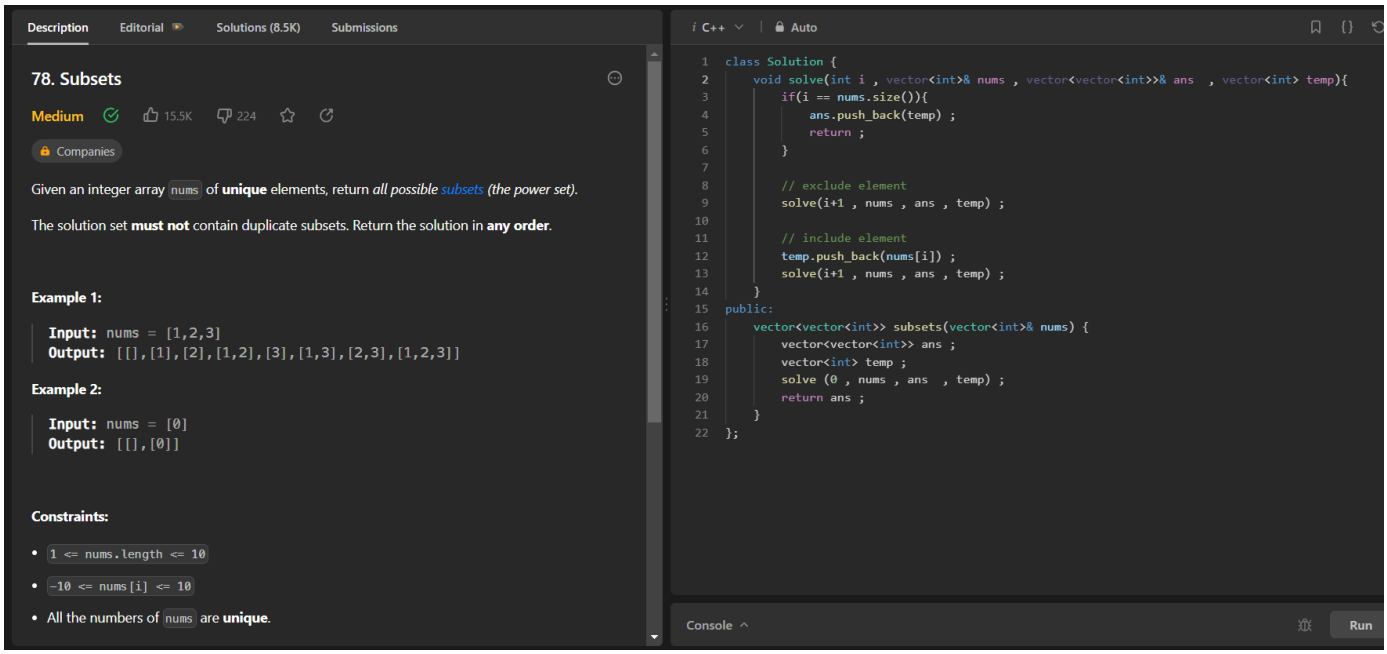
```
Input: x = 2.00000, n = -2
Output: 0.25000
Explanation: 2^-2 = 1/2^2 = 1/4 = 0.25
```

Constraints:

C++ | Auto

```cpp
class Solution {
    double solve(double x, int n){
        if(n == 0)
            return 1 ;

        if(n==1)
            return x ;

        double ans = solve(x , n/2) ;

        if(n%2==0){
            return ans * ans ;
        }else{
            return x * ans * ans ;
        }
    }
public:
    double myPow(double x, int n) {
        double ans = 0 ;
        if(n < 0){
            x = 1/x ;
            n = abs(n) ;
        }
        ans = solve(x , n) ;
        return ans ;
    }
};
```

Q. Subsets



Description Editorial Solutions (8.5K) Submissions

78. Subsets

Medium 15.5K 224

Companies

Given an integer array `nums` of **unique** elements, return *all possible subsets (the power set)*.

The solution set **must not** contain duplicate subsets. Return the solution in **any order**.

Example 1:

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

Example 2:

```
Input: nums = [0]
Output: [[],[0]]
```

Constraints:

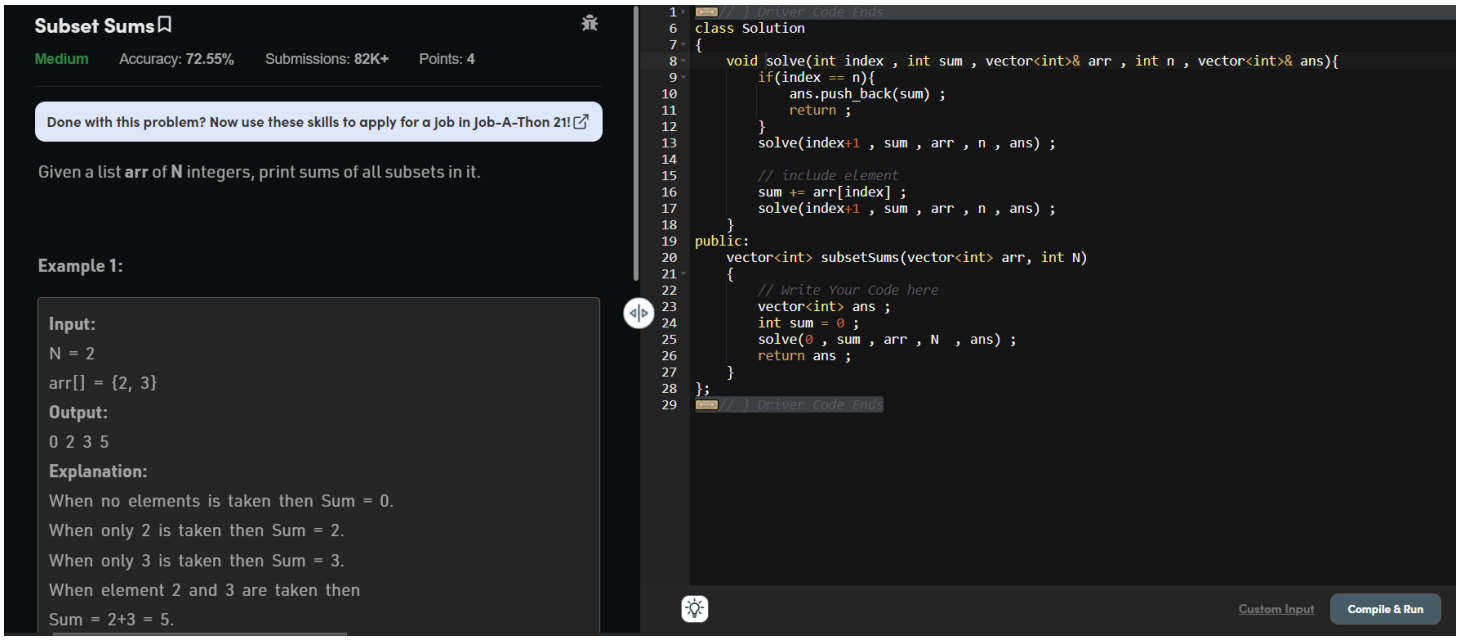
- `1 <= nums.length <= 10`
- `-10 <= nums[i] <= 10`
- All the numbers of `nums` are **unique**.

C++ | Auto

```cpp
class Solution {
    void solve(int i , vector<int>& nums , vector<vector<int>>& ans  , vector<int> temp){
        if(i == nums.size()){
            ans.push_back(temp) ;
            return ;
        }

        // exclude element
        solve(i+1 , nums , ans , temp) ;

        // include element
        temp.push_back(nums[i]) ;
        solve(i+1 , nums , ans , temp) ;
    }
public:
    vector<vector<int>> subsets(vector<int>& nums) {
        vector<vector<int>> ans ;
        vector<int> temp ;
        solve (0 , nums , ans  , temp) ;
        return ans ;
    }
};
```

Console Run

Q. Subset Sums



Subset Sums

Medium Accuracy: 72.55% Submissions: 82K+ Points: 4

Done with this problem? Now use these skills to apply for a job in Job-A-Thon 21!

Given a list **arr** of **N** integers, print sums of all subsets in it.

Example 1:

Input:
N = 2
arr[] = {2, 3}
Output:
0 2 3 5
Explanation:
When no elements is taken then Sum = 0.
When only 2 is taken then Sum = 2.
When only 3 is taken then Sum = 3.
When element 2 and 3 are taken then
Sum = 2+3 = 5.

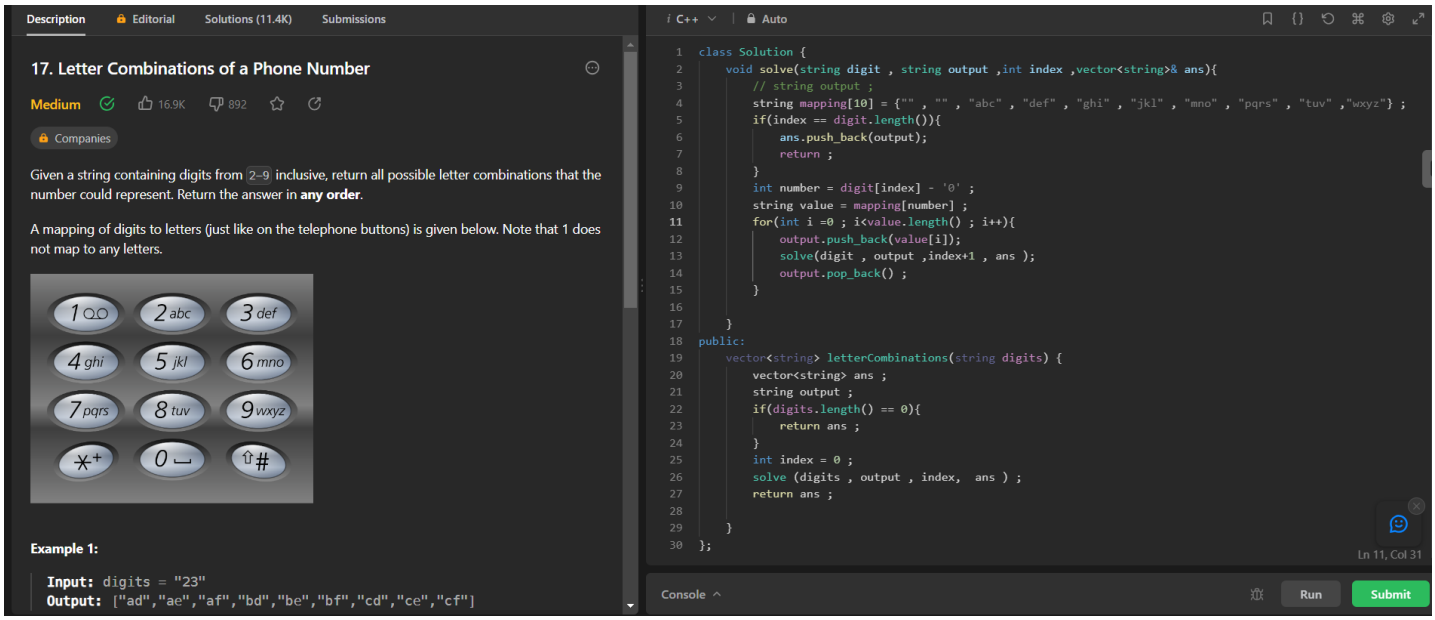
```
// } Driver Code Ends
class Solution
{
    void solve(int index , int sum , vector<int>& arr , int n , vector<int>& ans){
        if(index == n){
            ans.push_back(sum) ;
            return ;
        }
        solve(index+1 , sum , arr , n , ans) ;

        // include element
        sum += arr[index] ;
        solve(index+1 , sum , arr , n , ans) ;
    }
public:
    vector<int> subsetSums(vector<int> arr, int N)
    {
        // Write Your Code here
        vector<int> ans ;
        int sum = 0 ;
        solve(0 , sum , arr , N  , ans) ;
        return ans ;
    }
};
// } Driver Code Ends
```

Custom Input | Compile & Run

End

BackTracking

Q. Letter Combinations of a Phone Number



Description | Editorial | Solutions (11.4K) | Submissions

17. Letter Combinations of a Phone Number

Medium 16.9K 892

Companies

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example 1:

```
Input: digits = "23"
Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]
```

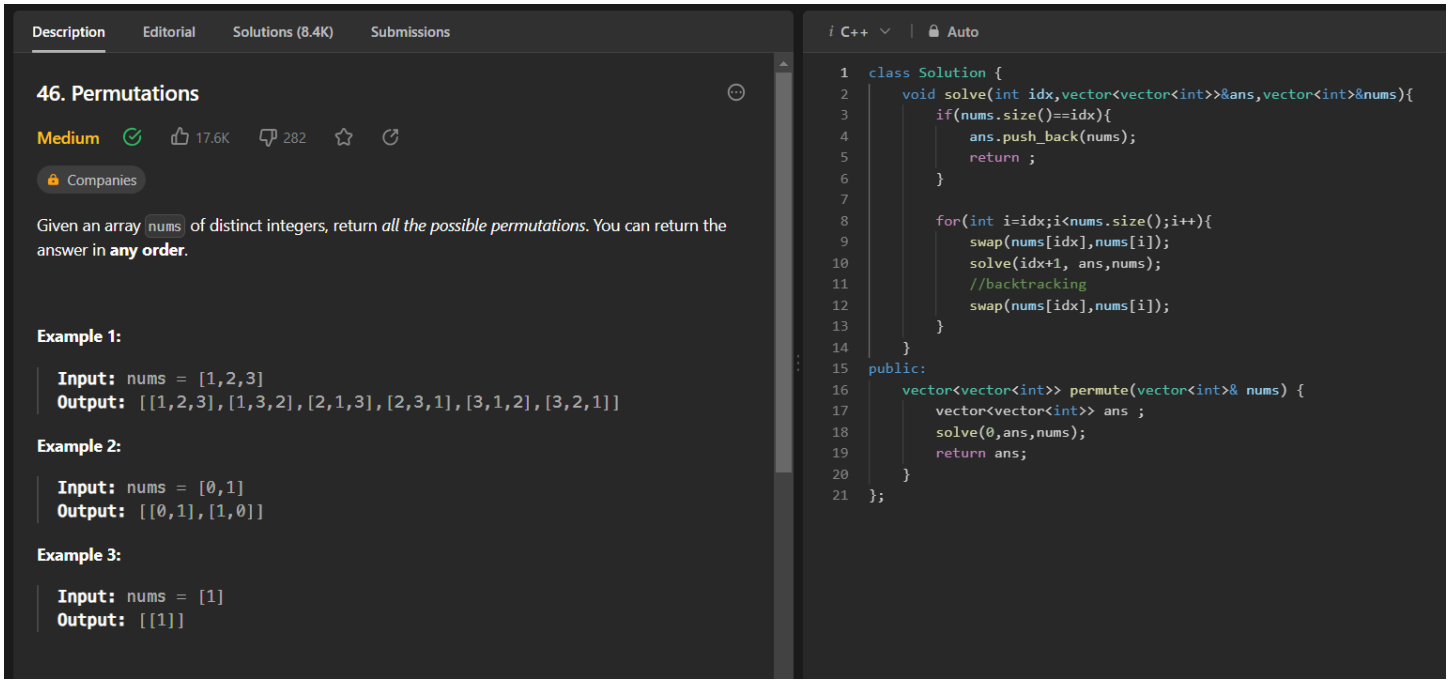
C++ | Auto

```cpp
class Solution {
    void solve(string digit , string output ,int index ,vector<string>& ans){
        // string output ;
        string mapping[10] = {"" , "" , "abc" , "def" , "ghi" , "jkl" , "mno" , "pqrs" , "tuv" ,"wxyz"} ;
        if(index == digit.length()){
            ans.push_back(output);
            return ;
        }
        int number = digit[index] - '0' ;
        string value = mapping[number] ;
        for(int i =0 ; i<value.length() ; i++){
            output.push_back(value[i]);
            solve(digit , output ,index+1 , ans );
            output.pop_back() ;
        }

    }
public:
    vector<string> letterCombinations(string digits) {
        vector<string> ans ;
        string output ;
        if(digits.length() == 0){
            return ans ;
        }
        int index = 0 ;
        solve (digits , output , index,  ans ) ;
        return ans ;

    }
};
```

Ln 11, Col 31

Console | Run | Submit

Q. Permutations



Description | Editorial | Solutions (8.4K) | Submissions

46. Permutations

Medium 17.6K 282

Companies

Given an array `nums` of distinct integers, return *all the possible permutations*. You can return the answer in **any order**.

Example 1:

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

Example 2:

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

Example 3:

```
Input: nums = [1]
Output: [[1]]
```

C++ | Auto

```cpp
class Solution {
    void solve(int idx,vector<vector<int>>&ans,vector<int>&nums){
        if(nums.size()==idx){
            ans.push_back(nums);
            return ;
        }

        for(int i=idx;i<nums.size();i++){
            swap(nums[idx],nums[i]);
            solve(idx+1, ans,nums);
            //backtracking
            swap(nums[idx],nums[i]);
        }
    }
public:
    vector<vector<int>> permute(vector<int>& nums) {
        vector<vector<int>> ans ;
        solve(0,ans,nums);
        return ans;
    }
};
```

Q. Rat In a Maze

Rat In a Maze

Last Updated: 11 Jul, 2023

Hard 0/120 6 upvotes

Problem Statement

Suggest Edit

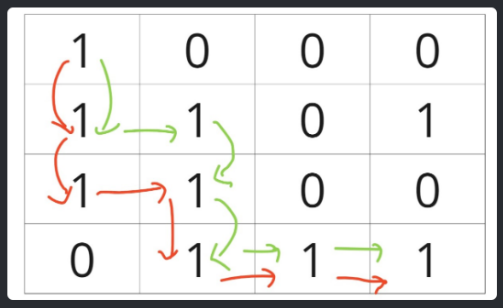
You are given a ***N*N*** maze with a rat placed at ***'mat[0][0]'***. Find all paths that rat can follow to reach its destination i.e. mat[N-1][N-1]. The directions in which the rat can move are 'U'(up), 'D'(down), 'L' (left), 'R' (right).

In the given maze, each cell can have a value of either 0 or 1. Cells with a value of 0 are considered blocked, which means the rat cannot enter or traverse through them. On the other hand, cells with a value of 1 are open, indicating that the rat is allowed to enter and move through those cells.

Example:

```
mat:{{1, 0, 0, 0},
     {1, 1, 0, 1},
     {1, 1, 0, 0},
     {0, 1, 1, 1}}

All possible paths are:
```

```
mat:{{1, 0, 0, 0},
     {1, 1, 0, 1},
     {1, 1, 0, 0},
     {0, 1, 1, 1}}

All possible paths are:
DDRDRR (in red)
DRDDRR (in green)
```




```
void findPathHelper(int i, int j, vector < vector < int >> & a, int n,
    vector < string > & ans, string move, vector < vector < int >> & vis) {
    if (i == n - 1 && j == n - 1) {
      ans.push_back(move);
      return;
    }

    // downward
    if (i + 1 < n && !vis[i + 1][j] && a[i + 1][j] == 1) {
      vis[i][j] = 1;
      findPathHelper(i + 1, j, a, n, ans, move + 'D', vis);
      vis[i][j] = 0;
    }

    // left
    if (j - 1 >= 0 && !vis[i][j - 1] && a[i][j - 1] == 1) {
      vis[i][j] = 1;
      findPathHelper(i, j - 1, a, n, ans, move + 'L', vis);
      vis[i][j] = 0;
    }

    // right
    if (j + 1 < n && !vis[i][j + 1] && a[i][j + 1] == 1) {
      vis[i][j] = 1;
      findPathHelper(i, j + 1, a, n, ans, move + 'R', vis);
      vis[i][j] = 0;
    }
```

```
    // right
    if (j + 1 < n && !vis[i][j + 1] && a[i][j + 1] == 1) {
      vis[i][j] = 1;
      findPathHelper(i, j + 1, a, n, ans, move + 'R', vis);
      vis[i][j] = 0;
    }

    // upward
    if (i - 1 >= 0 && !vis[i - 1][j] && a[i - 1][j] == 1) {
      vis[i][j] = 1;
      findPathHelper(i - 1, j, a, n, ans, move + 'U', vis);
      vis[i][j] = 0;
    }

  }

vector<string> ratMaze(vector<vector<int>> &mat) {
    // Write your code here.
    vector<string> ans ;
    int n = mat.size() ;

    vector < vector < int >> vis(n, vector < int > (n, 0));

    if (mat[0][0] == 1) findPathHelper(0, 0, mat, n, ans, "", vis);

    return ans;
}
```

Q . N-Queen

Description | Editorial | Solutions (4.4K) | Submissions

51. N-Queens

Hard | 11.1K | 234

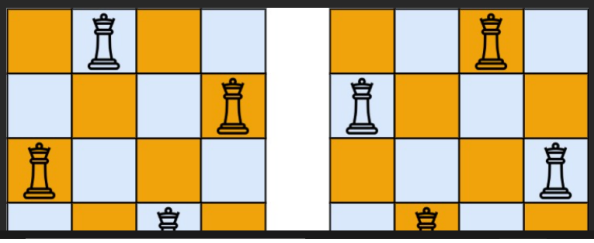
Companies

The **n-queens** puzzle is the problem of placing `n` queens on an `n x n` chessboard such that no two queens attack each other.

Given an integer `n`, return *all distinct solutions to the* ***n-queens puzzle***. You may return the answer in **any order**.

Each solution contains a distinct board configuration of the n-queens' placement, where `'Q'` and `'.'` both indicate a queen and an empty space, respectively.

Example 1:

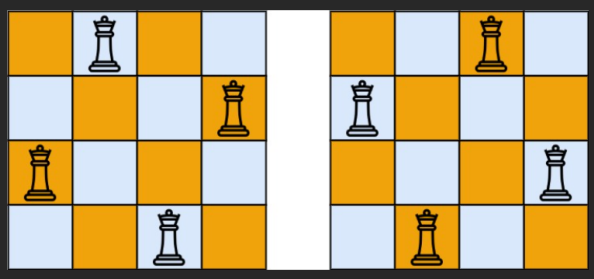


```cpp
class Solution {
    bool issafe(int row , int col , vector<vector<char>>& board , int n){
        int x = row ;
        int y = col ;
        while(x >= 0){
            if(board[x][y] == 'Q'){
                return false ;
            }
            x-- ;
        }
        x = row ;
        y = col ;
        while(x >= 0 && y>= 0){
            if(board[x][y] == 'Q'){
                return false ;
            }
            x-- ;
            y-- ;
        }
        x = row ;
        y = col ;
        while(x >= 0 && y < n){
            if(board[x][y] == 'Q'){
                return false ;
            }
            x-- ;
            y++ ;
        }
        return true ;
    }
    void addsolution (vector<vector<char>>& board , vector<vector<string>>& ans , int n){
```

Console | Run

Description | Editorial | Solutions (4.4K) | Submissions

Example 1:



```
Input: n = 4
Output: [[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]
Explanation: There exist two distinct solutions to the 4-queens puzzle as shown above
```

Example 2:

```
Input: n = 1
Output: [["Q"]]
```

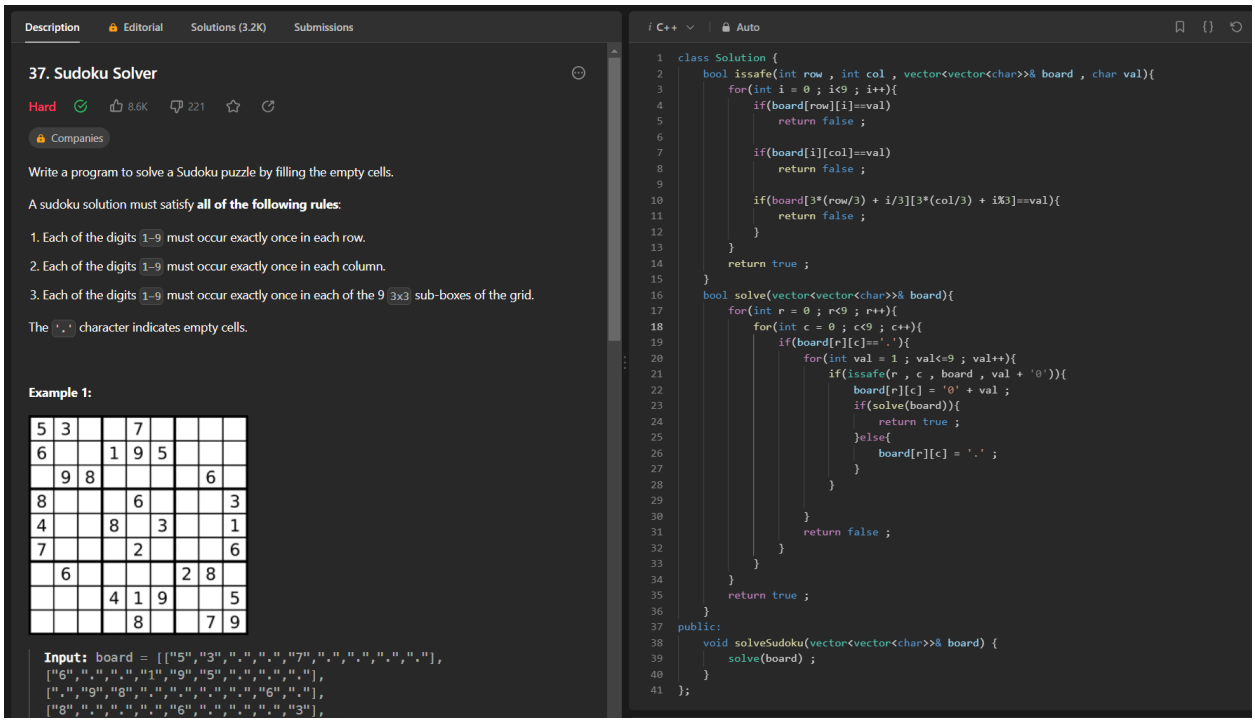
Constraints:

- `1 <= n <= 9`

```cpp
    void addsolution (vector<vector<char>>& board , vector<vector<string>>& ans , int n){
        vector<string> sol ;
        for(int i = 0 ; i<n ; i++){
            string temp ;
            for(int j = 0 ; j<n ; j++){
                temp.push_back(board[i][j]) ;
            }
            sol.push_back(temp) ;
        }
        ans.push_back(sol) ;
    }
    void solve(int row , vector<vector<char>>& board , vector<vector<string>>& ans , int n){
        if(row == n ){
            addsolution(board , ans , n) ;
            return ;
        }
        for(int col = 0 ; col< n ; col++){
            if(issafe(row , col  , board , n)){
                board[row][col] = 'Q' ;
                solve(row+1 , board , ans , n);
                board[row][col] = '.' ;
            }
        }
    }
public:
    vector<vector<string>> solveNQueens(int n) {
        vector<vector<char>> board (n , vector<char> (n , '.')) ;
        vector<vector<string>> ans ;
        int row = 0 ;
        solve(row , board , ans , n) ;
        return ans ;
```

Console | Run

Q. Sudoku Solver



Description | Editorial | Solutions (3.2K) | Submissions

37. Sudoku Solver

Hard | 8.6K | 221

Companies

Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy **all of the following rules**:

1. Each of the digits `1-9` must occur exactly once in each row.
2. Each of the digits `1-9` must occur exactly once in each column.
3. Each of the digits `1-9` must occur exactly once in each of the 9 `3x3` sub-boxes of the grid.

The `'.'` character indicates empty cells.

Example 1:

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

Input: board = [["5","3",".",".","7",".",".",".","."],
["6",".",".","1","9","5",".",".","."],
[".","9","8",".",".",".",".","6","."],
["8",".",".",".","6",".",".",".","3"],

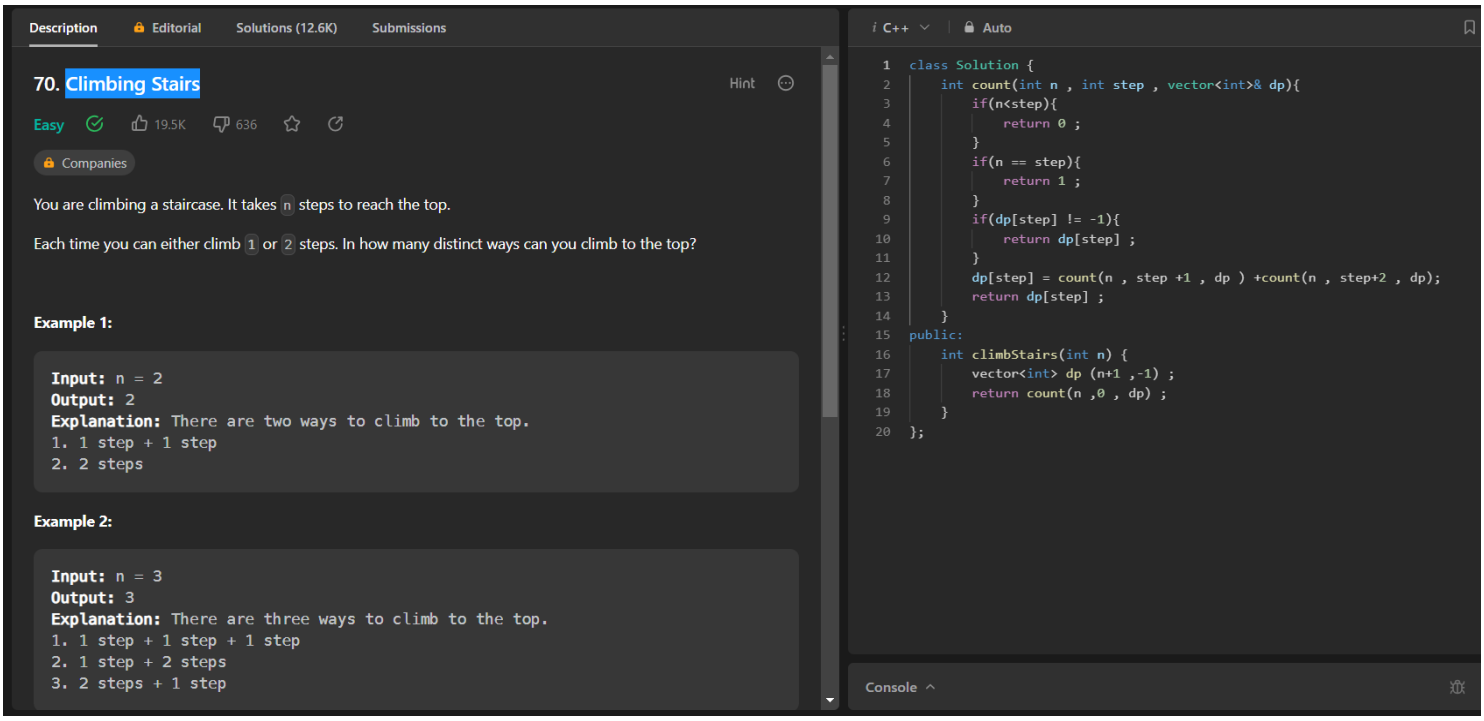
C++ | Auto

```cpp
class Solution {
    bool issafe(int row , int col , vector<vector<char>>& board , char val){
        for(int i = 0 ; i<9 ; i++){
            if(board[row][i]==val)
                return false ;

            if(board[i][col]==val)
                return false ;

            if(board[3*(row/3) + i/3][3*(col/3) + i%3]==val){
                return false ;
            }
        }
        return true ;
    }
    bool solve(vector<vector<char>>& board){
        for(int r = 0 ; r<9 ; r++){
            for(int c = 0 ; c<9 ; c++){
                if(board[r][c]=='.'){
                    for(int val = 1 ; val<=9 ; val++){
                        if(issafe(r , c , board , val + '0')){
                            board[r][c] = '0' + val ;
                            if(solve(board)){
                                return true ;
                            }else{
                                board[r][c] = '.' ;
                            }
                        }

                    }
                    return false ;
                }
            }
        }
        return true ;
    }
public:
    void solveSudoku(vector<vector<char>>& board) {
        solve(board) ;
    }
};
```

End

DP

Q. Climbing Stairs



Description | Editorial | Solutions (12.6K) | Submissions

70. Climbing Stairs Hint

Easy 19.5K 636

Companies

You are climbing a staircase. It takes `n` steps to reach the top.

Each time you can either climb `1` or `2` steps. In how many distinct ways can you climb to the top?

Example 1:

```
Input: n = 2
Output: 2
Explanation: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps
```

Example 2:

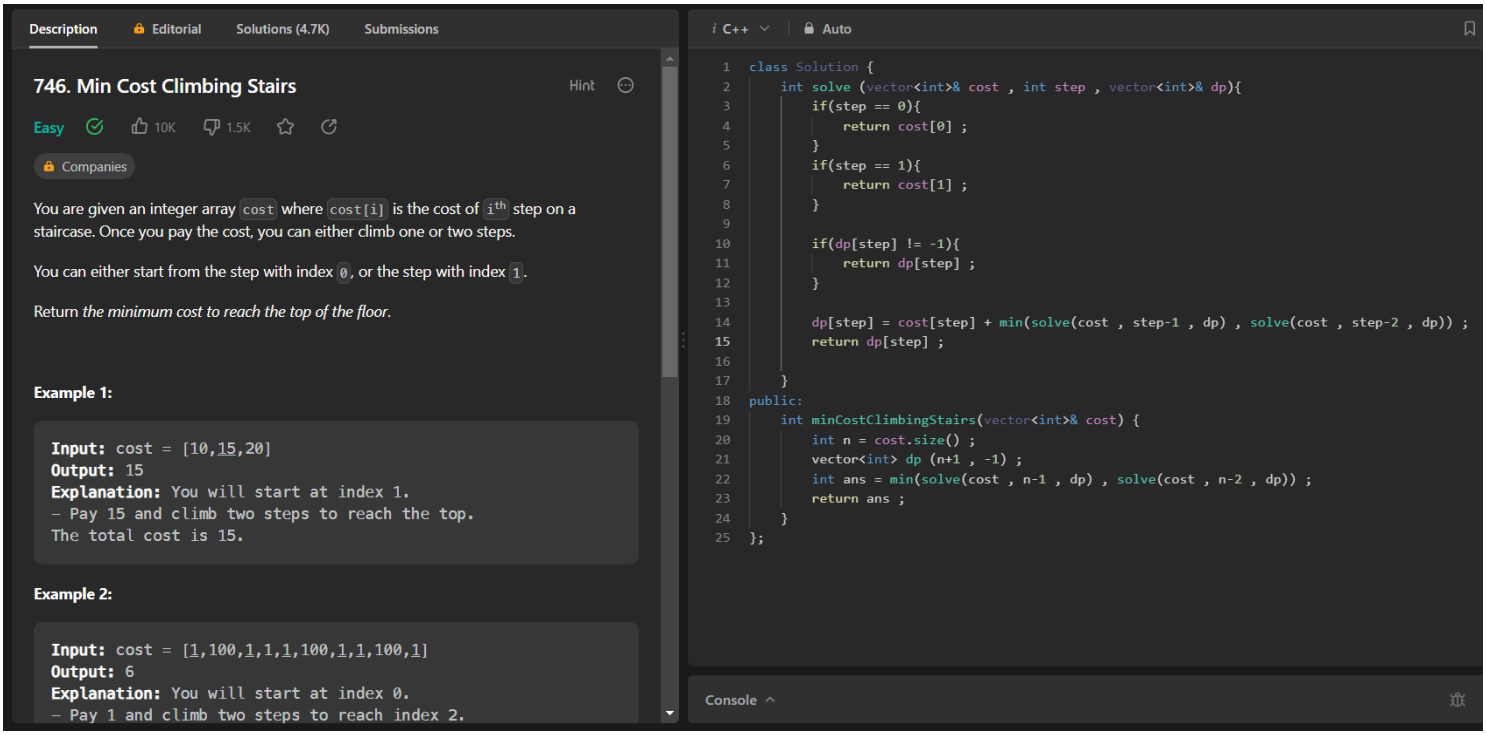
```
Input: n = 3
Output: 3
Explanation: There are three ways to climb to the top.
1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step
```

C++ | Auto

```cpp
class Solution {
    int count(int n , int step , vector<int>& dp){
        if(n<step){
            return 0 ;
        }
        if(n == step){
            return 1 ;
        }
        if(dp[step] != -1){
            return dp[step] ;
        }
        dp[step] = count(n , step +1 , dp ) +count(n , step+2 , dp);
        return dp[step] ;
    }
public:
    int climbStairs(int n) {
        vector<int> dp (n+1 ,-1) ;
        return count(n ,0 , dp) ;
    }
};
```

Console

Q. Min Cost Climbing Stairs



Description | Editorial | Solutions (4.7K) | Submissions

746. Min Cost Climbing Stairs Hint

Easy 10K 1.5K

Companies

You are given an integer array `cost` where `cost[i]` is the cost of i^{th} step on a staircase. Once you pay the cost, you can either climb one or two steps.

You can either start from the step with index `0`, or the step with index `1`.

Return *the minimum cost to reach the top of the floor.*

Example 1:

```
Input: cost = [10,15,20]
Output: 15
Explanation: You will start at index 1.
- Pay 15 and climb two steps to reach the top.
The total cost is 15.
```

Example 2:

```
Input: cost = [1,100,1,1,1,100,1,1,100,1]
Output: 6
Explanation: You will start at index 0.
- Pay 1 and climb two steps to reach index 2.
```

C++ | Auto

```cpp
class Solution {
    int solve (vector<int>& cost , int step , vector<int>& dp){
        if(step == 0){
            return cost[0] ;
        }
        if(step == 1){
            return cost[1] ;
        }

        if(dp[step] != -1){
            return dp[step] ;
        }

        dp[step] = cost[step] + min(solve(cost , step-1 , dp) , solve(cost , step-2 , dp)) ;
        return dp[step] ;

    }
public:
    int minCostClimbingStairs(vector<int>& cost) {
        int n = cost.size() ;
        vector<int> dp (n+1 , -1) ;
        int ans = min(solve(cost , n-1 , dp) , solve(cost , n-2 , dp)) ;
        return ans ;
    }
};
```

Console

Q.

End

Linked List

Q. Reverse Linked List

Problem List

Description | Editorial | Solutions (10.3K) | Submissions

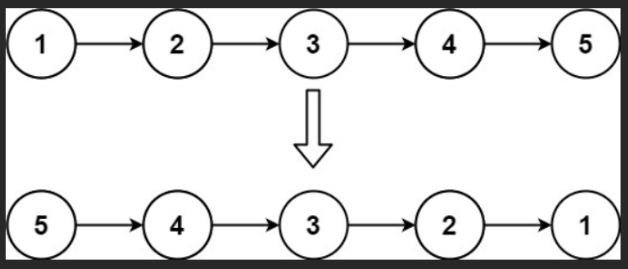
206. Reverse Linked List

Easy | 18.2K | 337

Companies

Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.

Example 1:



Input: head = [1,2,3,4,5]
Output: [5,4,3,2,1]

C++ | Auto

```cpp
*     ListNode *next;
*     ListNode() : val(0), next(nullptr) {}
*     ListNode(int x) : val(x), next(nullptr) {}
*     ListNode(int x, ListNode *next) : val(x), next(next) {}
* };
*/
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        // listNode * temp = NULL ;
        // if(head->next== NULL ){
        //     return head ;
        // }

        // temp = reverseList(head->next)
        // temp->next
        ListNode * prev = NULL ;
        ListNode * curr = head ;
        ListNode * forward = NULL ;

        while(curr != NULL){
            forward = curr->next ;
            curr->next = prev ;
            prev = curr ;
            curr = forward ;
        }
        return prev ;


    }
};
```

Q. Two Sums

Description | Discussion (67) | Solutions (9.9K) | Submissions

2. Add Two Numbers

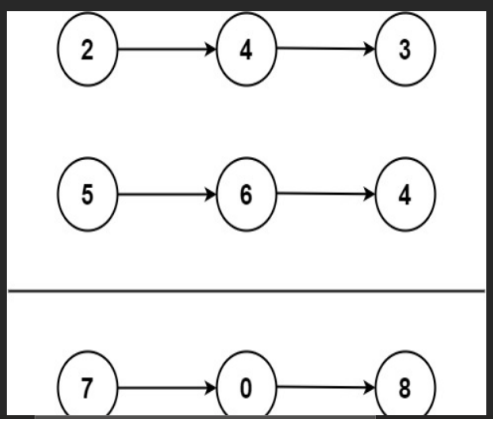
Medium | 21.8K | 4.3K

Companies

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1:



C++ | Auto

```cpp
*/
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        int carry = 0 ;
        ListNode* head = NULL;
        ListNode * tail = NULL ;
        while(l1 != NULL || l2 != NULL || carry){
            int sum = 0 ;
            if(l1 !=NULL){
                sum += l1->val;
                l1 = l1->next ;
            }if(l2 !=NULL){
                sum += l2->val;
                l2 = l2->next ;
            }
            sum += carry ;
            carry = sum/10 ;
            ListNode* temp = new ListNode(sum%10);

            if(head == NULL){
                head = temp ;
                tail = temp ;
            }else{
                tail ->next = temp ;
                tail = tail->next ;
            }
        }
        return head ;

    }
};
```

Console | Run | Submit

Q. Remove Duplicate From sorted List

Description | Discussion (5) | Solutions (5.1K) | Submissions

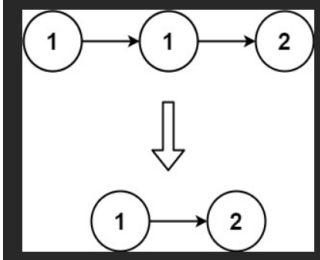
83. Remove Duplicates from Sorted List

Easy | 5.9K | 210

Companies

Given the `head` of a sorted linked list, *delete all duplicates such that each element appears only once.* Return *the linked list* ***sorted*** *as well.*

Example 1:




```
Input: head = [1,1,2]
Output: [1,2]
```

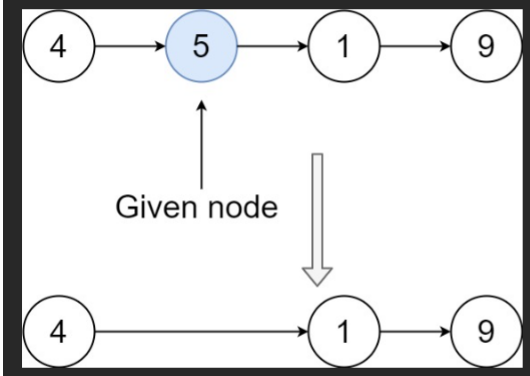
C++ | Auto

```cpp
/*
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode * temp = head ;

        if(temp == NULL)
            return head ;
        while(temp->next != NULL){
            if(temp->val == temp->next->val ){
                ListNode * a = temp->next ->next ; ;
                // ListNode * l =a->next ;
                delete temp ->next ;
                temp ->next = a ;

            }else{
                temp = temp ->next ;

            }

        }
        return head ;

    }
};
```

Q. Delete Node in a Linked List

Description | Discussion (65) | Solutions (2.8K) | Submissions

Example 1:




```
Input: head = [4,5,1,9], node = 5
Output: [4,1,9]
Explanation: You are given the second node with value 5, the linked list should
become 4 -> 1 -> 9 after calling your function.
```

C++ | Auto

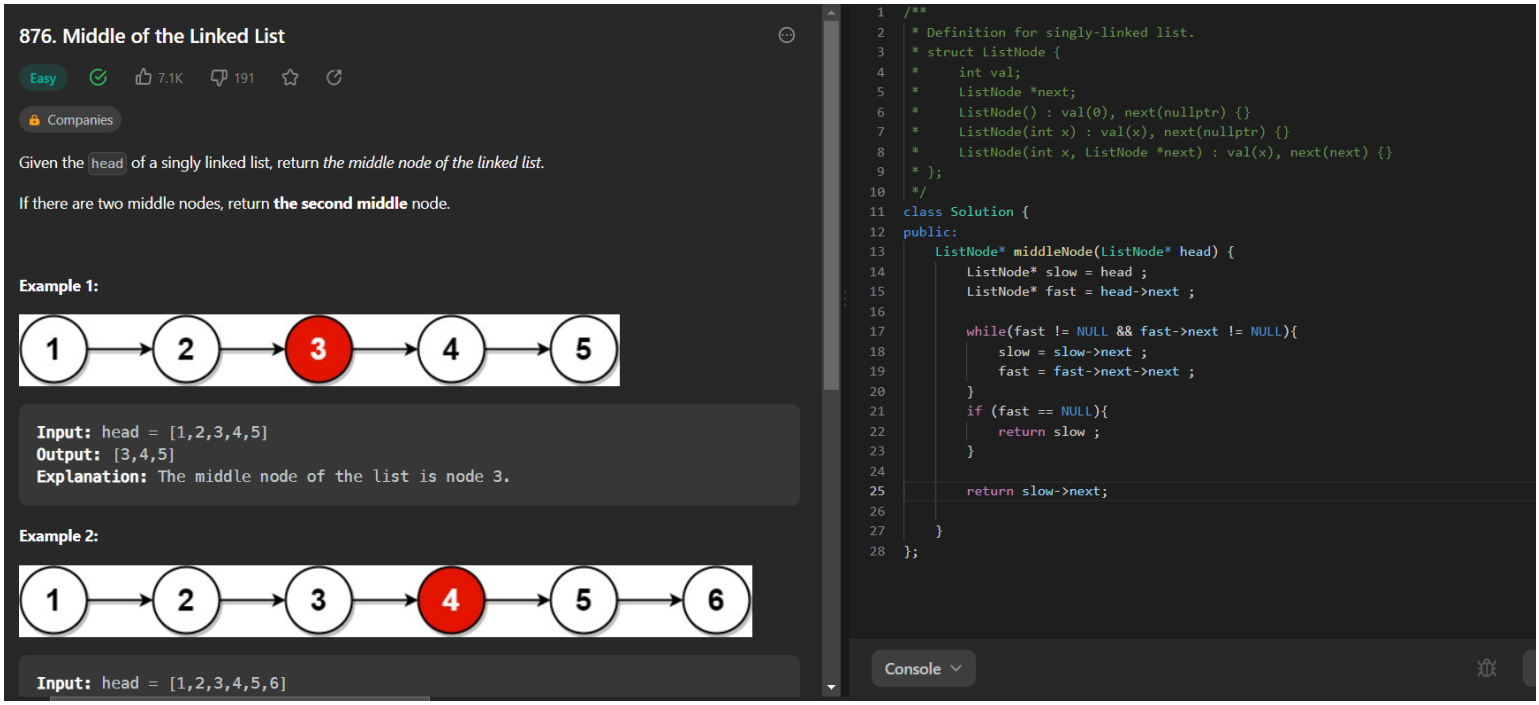
```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    void deleteNode(ListNode* node) {

        node ->val = node->next->val;
        node ->next = node->next->next;


    }
};
```

Q . Remove Nth Node From End of List

```
class Solution {
    int length(ListNode* head){
        int i=1;
        while(head->next!=NULL){
            i++;
            head=head->next;
        }
        return i;
    }
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        if(head==NULL)
        return NULL;
        if(head->next==NULL&&n==1)
        return NULL;
        int l=length(head);
        if(l==n)
        {
            head=head->next;
            return head;
        }
        l=l-n;
        int i=1;
        ListNode* curr=head;
        while(i<l){
            curr=curr->next;
            i++;
        }
        ListNode* temp=curr->next;
        curr->next=curr->next->next;
        temp->next=NULL;
        delete temp;
        return head;
    }
```

};

Q . Middle of The Linked List



876. Middle of the Linked List

Easy 7.1K 191

Companies

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]
```

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

        while(fast != NULL && fast->next != NULL){
            slow = slow->next ;
            fast = fast->next->next ;
        }
        if (fast == NULL){
            return slow ;
        }

        return slow->next;

    }
};
```

Console

Q . Linked List Cycle

Problem List

Description | Editorial | Solutions (7.9K) | Submissions

141. Linked List Cycle

Easy | 12.4K | 1K

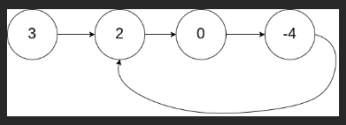
Companies

Given `head`, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to. **Note that `pos` is not passed as a parameter**.

Return `true` *if there is a cycle in the linked list*. Otherwise, return `false`.

Example 1:




```
Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).
```

C++ | Auto

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode * temp = head ;
        map<ListNode* , int> m ;
        int i = 1 ;
        while(temp != NULL){
            if(m[temp] == 0){
                m[temp] = i++ ;
                temp =  temp->next ;
            }else{
                return true ;
            }
        }
        return false ;

    }
};
```

Console

Stacks

Queue

Binary Tree

Binary Search Tree

Hashmap

Heap And Priority Queue

Tries

Graphs

Greedy

Algorithms & Other