40. Combination Sum II

Solve

Medium Topics Companies

Given a collection of candidate numbers (`candidates`) and a target number (`target`), find all unique combinations in `candidates` where the candidate numbers sum to `target` .

Each number in `candidates` may only be used **once** in the combination.

Note: The solution set must not contain duplicate combinations.

Example 1:

```
Input: candidates = [10,1,2,7,6,1,5], target = 8
Output:
[
[1,1,6],
[1,2,5],
[1,7],
[2,6]
]
```

Example 2:

```
Input: candidates = [2,5,2,1,2], target = 5
Output:
[
[1,2,2],
[5]
]
```

Constraints:

- `1 <= candidates.length <= 100`
- `1 <= candidates[i] <= 50`
- `1 <= target <= 30`

Code

Java Auto

```java
class Solution {
    public List<List<Integer>> combinationSum2(int[] candidates, int target) {
      List<List<Integer>> ret = new ArrayList<>();

      Arrays.sort(candidates);
      helper(candidates, ret, new ArrayList<Integer>(), 0, target);
      return ret;
    }
    public void helper(int[] candidates, List<List<Integer>> ret, List<Integer> curList, int start, int target) {
      if(target<0)
        return;
      if(target==0) {
        ret.add(new ArrayList<Integer>(curList));
      }

      for(int i=start; i<candidates.length; i++)       {
        if (candidates[i]>target) break; //shotcut

        if (i>start && candidates[i]==candidates[i-1]) //avoid dups, as this is combination
          continue;

        curList.add(candidates[i]);
        //as you can't repeat the same, so increase i as start
        helper(candidates, ret, curList, i+1, target-candidates[i]);
        curList.remove(curList.size()-1);
      }

    }
}
```

Saved to local Ln

Testcase Test Result

41. First Missing Positive

Solv

Hard Topics Companies Hint

Given an unsorted integer array `nums`, return the smallest missing positive integer.

You must implement an algorithm that runs in `O(n)` time and uses `O(1)` auxiliary space.

Example 1:

```
Input: nums = [1,2,0]
Output: 3
Explanation: The numbers in the range [1,2] are all in the array.
```

Example 2:

```
Input: nums = [3,4,-1,1]
Output: 2
Explanation: 1 is in the array but 2 is missing.
```

Example 3:

```
Input: nums = [7,8,9,11,12]
Output: 1
Explanation: The smallest positive integer 1 is missing.
```

Constraints:

- `1 <= nums.length <= 10^5`
- `-2^31 <= nums[i] <= 2^31 - 1`

Seen this question in a real interview before? 1/4

Yes No

Accepted 971.1K Submissions 2.6M Acceptance Rate 37.3%

15.3K 108

Code

Java Auto

```java
class Solution {
    public int firstMissingPositive(int[] nums) {
      int n = nums.length;

      //idea: similar to bucket sort o(n)
//     Put each number in its right place.

//     For example:

//     When we find 5, then swap it with A[4].

//     At last, the first place where its number is not right, return the place + 1.
      for(int i=0; i<n; i++){
        while(nums[i]>0 && nums[i]<=n && nums[nums[i]-1] != nums[i]) {

          //swap
          int tmp=nums[nums[i]-1];
          nums[nums[i]-1]=nums[i];
          nums[i]=tmp;
        }

      }
      for(int i=0;i<n; i++) {
        if(nums[i]!=i+1)
          return i+1;
      }
      return n+1;

    }
}
```

Saved to local Ln

Testcase Test Result

42. Trapping Rain Water

Hard Topics Companies

Given `n` non-negative integers representing an elevation map where the width of each bar is `1`, compute how much water it can trap raining.

Example 1:


```
Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
Explanation: The above elevation map (black section) is represented by array
[0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.
```

Example 2:

```
Input: height = [4,2,0,3,2,5]
Output: 9
```

Constraints:

- `n == height.length`
- `1 <= n <= 2 * 10^4`
- `0 <= height[i] <= 10^5`

/> Code

ava Auto

```java
class Solution {
//     //stack
// 下面这种解法是用 stack 来做的，博主一开始都没有注意到这道题的 tag 还有 stack，所以以后在总结的时候还是要多多留意一下标签啊。其实用的方法博主感觉更容易理解，思路是，遍历高度，如果此时栈为空，或者当前高度小于等于栈顶高度，则把当前高度的坐标压入栈，注意这里不直接把高度而是把坐标压入栈，这样方便在后来算水平距离。当遇到比栈顶高度大的时候，就说明有可能会有坑存在，可以装雨水。此时栈里至少有一个高度，如果只话，那么不能形成坑，直接跳过，如果多余一个的话，那么此时把栈顶元素取出来当作坑，新的栈顶元素就是左边界，当前高度是右边界，只要取二者较小坑的高度，长度就是右边界坐标减去左边界坐标再减1，二者相乘就是盛水量啦
    public int sol1(int[] height) {
      Stack<Integer> stack = new Stack<>();
      int ret = 0;
      for(int i=0; i<height.length; i++){
        while(!stack.empty() && height[stack.peek()]<height[i]) {
          int bot=height[stack.pop()];
          if(stack.empty()) break;
          int left=stack.peek();
          int wid=i-left-1;
          int hei=Math.min(height[i], height[left])-bot;

          ret+=wid*hei;
        }
        stack.push(i);
      }

      return ret;
    }

// 这个 DP 算法需要遍历两遍数组，第一遍在 dp[i] 中存入i位置左边的最大值，然后开始第二遍遍历数组，第二次遍历时找右边最大值，然后和左边最较取其中的较小值，然后跟当前值 A[i] 相比，如果大于当前值，则将差值存入结果，
    public int sol2(int[] height) {
        int res = 0, mx = 0, n = height.length;
        int[] dp = new int[n];
        //find left max
        for (int i = 0; i < n; ++i) {
            dp[i] = mx;
            mx = Math.max(mx, height[i]);
        }
        mx = 0;
        //find right max
        for (int i = n - 1; i >= 0; --i) {
            dp[i] = Math.min(dp[i], mx);
            mx = Math.max(mx, height[i]);
            if (dp[i] - height[i] > 0) res += dp[i] - height[i];
        }
        return res;
    }

    //two pointer
// 这个算法需要 left 和 right 两个指针分别指向数组的首尾位置，从两边向中间扫描，在当前两指针确定的范围内，先比较两头找出较小值，如果较
```

aved to local L

Testcase Test Result

Code

ava Auto

```
    //two pointer
// 这个算法需要 left 和 right 两个指针分别指向数组的首尾位置，从两边向中间扫描，在当前两指针确定的范围内，先比较两头找出较小值，如果较小值是
left 指向的值，则从左向右扫描，如果较小值是 right 指向的值，则从右向左扫描，若遇到的值比当较小值小，则将差值存入结果，如遇到的值大，则更
新的窗口范围，以此类推直至 left 和 right 指针重合
    public int sol3(int[] height) {

      int left = 0, right = height.length - 1;
      int ans = 0;
      int left_max = 0, right_max = 0;
      while (left < right) {
          if (height[left] < height[right]) {
              if (height[left] >= left_max )
                left_max = height[left];
              else
                ans += left_max - height[left];
              ++left;
          }
          else {
              if (height[right] >= right_max)
                right_max = height[right];
              else
                ans += right_max - height[right];
              --right;
          }
      }
      return ans;
       // int res = 0, l = 0, r = height.length - 1;
       //  while (l < r) {
       //      int mn = Math.min(height[l], height[r]);
       //      if (height[l] == mn) {
       //          ++l;
       //          while (l < r && height[l] < mn) {
       //              res += mn - height[l++];
       //          }
       //      } else {
       //          --r;
       //          while (l < r && height[r] < mn) {
       //              res += mn - height[r--];
       //          }
       //      }
       //  }
       //  return res;
    }

    public int trap(int[] height) {
      // return sol1(height);
      // return sol2(height);
      return sol3(height);
```

aved to local

Testcase Test Result

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied Very Satisfied

43. Multiply Strings

Solv

Medium Topics Companies

Given two non-negative integers `num1` and `num2` represented as strings, return the product of `num1` and `num2`, also represented as a string.

Note: You must not use any built-in BigInteger library or convert the inputs to integer directly.

Example 1:

```
Input: num1 = "2", num2 = "3"
Output: "6"
```

Example 2:

```
Input: num1 = "123", num2 = "456"
Output: "56088"
```

Constraints:

- `1 <= num1.length, num2.length <= 200`
- `num1` and `num2` consist of digits only.
- Both `num1` and `num2` do not contain any leading zero, except the number `0` itself.

Seen this question in a real interview before? 1/4

Yes No

Accepted 720.2K Submissions 1.8M Acceptance Rate 39.6%

Topics

Companies

6.7K 72

Code

Java Auto

```java
class Solution {
    public String multiply(String num1, String num2) {
      int[] mul = new int[num1.length()+num2.length()];
      int n = mul.length;
      for(int i=num1.length()-1; i>=0; i--) {
        for(int j=num2.length()-1;j>=0; j--) {
          //System.out.println(""+i+" "+j);
          mul[i+j+1] += (num1.charAt(i)-'0') * (num2.charAt(j)-'0');

        }//for num2
      }//for num1
      //System.out.println(Arrays.toString(mul));
      for(int i=mul.length-1; i>0; i--) {
        if(mul[i]>=10) {
          int t=mul[i];
          mul[i]=t%10;
          mul[i-1]+=t/10;
        }

      }//for carry
      //System.out.println(Arrays.toString(mul));
      StringBuilder sb = new StringBuilder();

      for(int i=0; i<mul.length; i++) {
        if(mul[i]!=0 || sb.length()>0) {
          sb.append(mul[i]);

        }

      }
      if(sb.length()==0)
        return("0");
      return sb.toString();
    }
}
```

Saved to local

Testcase Test Result

44. Wildcard Matching

Solv

Hard Topics Companies

Given an input string (`s`) and a pattern (`p`), implement wildcard pattern matching with support for `'?'` and `'*'` where:

- `'?'` Matches any single character.
- `'*'` Matches any sequence of characters (including the empty sequence).

The matching should cover the **entire** input string (not partial).

Example 1:

```
Input: s = "aa", p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
```

Example 2:

```
Input: s = "aa", p = "*"
Output: true
Explanation: '*' matches any sequence.
```

Example 3:

```
Input: s = "cb", p = "?a"
Output: false
Explanation: '?' matches 'c', but the second letter is 'a', which does not match 'b'.
```

Constraints:

- `0 <= s.length, p.length <= 2000`
- `s` contains only lowercase English letters.
- `p` contains only lowercase English letters, `'?'` or `'*'`.

Code

Java Auto

```java
class Solution {
    public boolean isMatch(String s, String p) {
      int lens=s.length();
      int lenp=p.length();
      boolean[][] dp = new boolean[lens+1][lenp+1];
      dp[0][0]=true;
      for(int i=0; i<lenp; i++){
        if(p.charAt(i)!='*') break;
        dp[0][i+1]=true;
      }
      for(int i=0; i<lens; i++){
        for(int j=0; j<lenp; j++) {
          if(p.charAt(j)=='*') { //*
            dp[i+1][j+1]=dp[i+1][j] || dp[i][j+1];
          }else {
            dp[i+1][j+1]=dp[i][j] && (s.charAt(i)==p.charAt(j) || p.charAt(j)=='?');
          }//not *
        }//j
      }//for i

      return dp[lens][lenp];
    }
}
```

Saving...

Testcase Test Result

45. Jump Game II

Solv

Medium Topics Companies

You are given a **0-indexed** array of integers `nums` of length `n`. You are initially positioned at `nums[0]`.

Each element `nums[i]` represents the maximum length of a forward jump from index `i`. In other words, if you are at `nums[i]`, you c jump to any `nums[i + j]` where:

- `0 <= j <= nums[i]` and
- `i + j < n`

Return *the minimum number of jumps to reach* `nums[n - 1]`. The test cases are generated such that you can reach `nums[n - 1]`.

Example 1:

```
Input: nums = [2,3,1,1,4]
Output: 2
Explanation: The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0
1, then 3 steps to the last index.
```

Example 2:

```
Input: nums = [2,3,0,1,4]
Output: 2
```

Constraints:

- `1 <= nums.length <= 10^4`
- `0 <= nums[i] <= 1000`
- It's guaranteed that you can reach `nums[n - 1]`.

Seen this question in a real interview before? 1/4

13.7K 91

Problem List Run Submit 0

Code

Java Auto

```java
class Solution {
    public int jump(int[] nums) {
      int left=0; int right=0; int ret=0;
      while(right<nums.length-1){
        ret++;
        int cur=right;
        while(left<=cur) {
          right=Math.max(right, left+nums[left]);
          left++;
        }
        //System.out.println(""+left+" "+right);
      }//while
      return ret;
    }
}
```

Saved to local

Testcase Test Result

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied Very Satisfied

Code

Java Auto

```
17
18        helper1(nums, ret, new ArrayList<Integer>());
19
20        return ret;
21      }
22
23
24    public void helper2(int[] nums, List<List<Integer>> ret,  int start ) {
25       if( start >= nums.length )  {
26         List<Integer> l = new ArrayList<>();
27         //no better way to convert int[] to List<Integer>
28         for(int n : nums) {
29           l.add(n);
30         }
31         ret.add(l);
32         return;
33       }
34       for(int i=start; i<nums.length; i++) {
35         swap(nums, i, start);
36         helper2(nums, ret, start+1);
37         swap(nums, i, start);
38       }
39     }
40     public List<List<Integer>> permute2(int[] nums) {
41       List<List<Integer>> ret = new ArrayList<>()        ;
42
43       helper2(nums, ret,  0);
44
45       return ret;
46     }
47
48     public void swap(int[] nums, int i, int j) {
49       int tmp = nums[i];
50       nums[i]=nums[j];
51       nums[j]=tmp;
52     }
53
54     public List<List<Integer>> permute(int[] nums) {
55       return permute2(nums);
56       // return permute1(nums);
57     }
58
59  }
```

Saved to local

Testcase Test Result

46. Permutations

Solv

Medium Topics 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]]
```

Constraints:

- `1 <= nums.length <= 6`
- `-10 <= nums[i] <= 10`
- All the integers of `nums` are **unique**.

Seen this question in a real interview before? 1/4

Yes No

Accepted 1.9M Submissions 2.4M Acceptance Rate 77.5%

Topics

18.2K 79

Code

Java Auto

```java
class Solution {
    public void helper1(int[] nums, List<List<Integer>> ret, List<Integer> curList ) {
      if(curList.size() == nums.length )  {
        ret.add(new ArrayList<Integer>(curList));
        return;
      }
      for(int i=0; i<nums.length; i++) {
        if(curList.contains(nums[i]))
          continue;
        curList.add(nums[i]);
        helper1(nums,ret,curList);
        curList.remove(curList.size()-1);
      }
    }
    public List<List<Integer>> permute1(int[] nums) {
      List<List<Integer>> ret = new ArrayList<>()        ;

      helper1(nums, ret, new ArrayList<Integer>());

      return ret;
    }


   public void helper2(int[] nums, List<List<Integer>> ret,  int start ) {
      if( start >= nums.length )  {
        List<Integer> l = new ArrayList<>();
        //no better way to convert int[] to List<Integer>
        for(int n : nums) {
          l.add(n);
        }
        ret.add(l);
        return;
      }
      for(int i=start; i<nums.length; i++) {
        swap(nums, i, start);
        helper2(nums, ret, start+1);
        swap(nums, i, start);
      }
    }
    public List<List<Integer>> permute2(int[] nums) {
      List<List<Integer>> ret = new ArrayList<>()        ;

      helper2(nums, ret,  0);

      return ret;
    }

    public void swap(int[] nums, int i, int j) {
```

Saved to local

Testcase Test Result

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied Very Satisfied

47. Permutations II

Solv

Medium Topics Companies

Given a collection of numbers, `nums`, that might contain duplicates, return *all possible unique permutations* ***in any order***.

Example 1:

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

Example 2:

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

Constraints:

- `1 <= nums.length <= 8`
- `-10 <= nums[i] <= 10`

Seen this question in a real interview before? 1/4

Yes No

Accepted 850.5K Submissions 1.5M Acceptance Rate 58.3%

Topics

Companies

8.2K 21

Code

Java
Auto

```java
class Solution {

    public void helper1(int[] nums, List<List<Integer>> ret, List<Integer> curList, boolean[] used ) {
      if(curList.size() == nums.length )  {
        ret.add(new ArrayList<Integer>(curList));
        return;
      }
      for(int i=0; i<nums.length; i++) {
        if(used[i] || (i>0) && nums[i]==nums[i-1] && !used[i-1])
          continue;
        curList.add(nums[i]);
        used[i]=true;
        helper1(nums,ret,curList,used);
        curList.remove(curList.size()-1);
        used[i]=false;
      }
    }
    public List<List<Integer>> permuteUnique(int[] nums) {
      List<List<Integer>> ret = new ArrayList<>()        ;
      boolean[] used = new boolean[nums.length];
      Arrays.sort(nums);
      helper1(nums, ret, new ArrayList<Integer>(), used);

      return ret;
    }

}
```

Saved to local

Ln

Testcase
Test Result

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied
Very Satisfied

48. Rotate Image

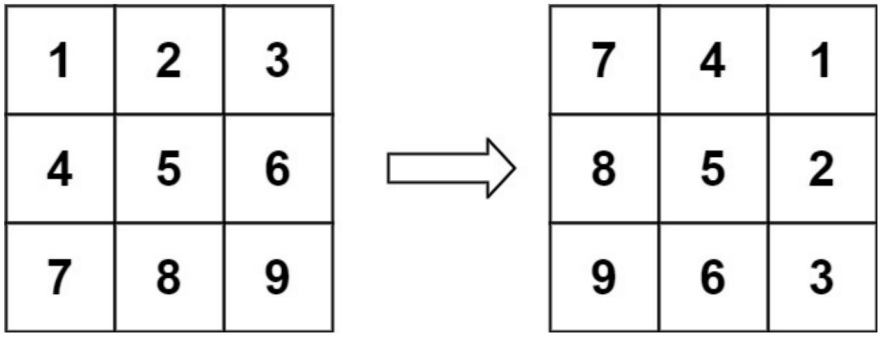
Solv

Medium Topics Companies

You are given an `n x n` 2D `matrix` representing an image, rotate the image by **90** degrees (clockwise).

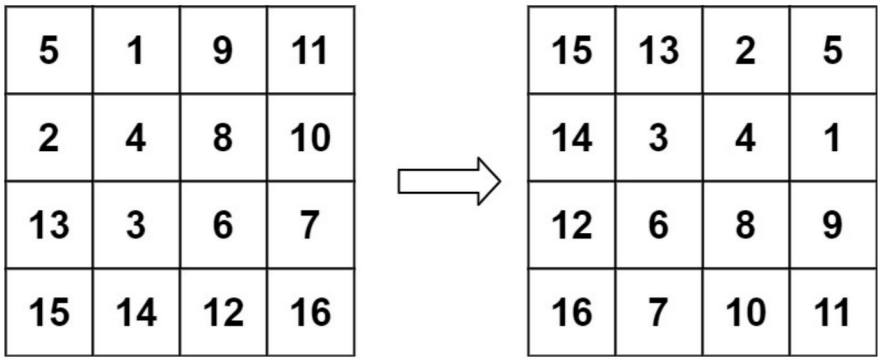
You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D ma and do the rotation.

Example 1:




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

Example 2:




```
Input: matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,14,12,16]]
Output: [[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]
```

16.5K 96

Problem List Run Submit 0

Code

Java Auto

```java
class Solution {
    public void print(int[][] m) {
      for(int i=0; i<m.length; i++) {
        System.out.println(Arrays.toString(m[i]));
      }
    }

    //streight forward
    public void sol1(int[][] matrix) {
        int n = matrix.length;
        for (int i = 0; i < n / 2; ++i) {
            for (int j = i; j < n - 1 - i; ++j) {
                //start from [i,i] to the 2nd last col
                int tmp = matrix[i][j];
                matrix[i][j] = matrix[n - 1 - j][i];
                matrix[n - 1 - j][i] = matrix[n - 1 - i][n - 1 - j];
                matrix[n - 1 - i][n - 1 - j] = matrix[j][n - 1 - i];
                matrix[j][n - 1 - i] = tmp;
            }
        }
    }

//     public void sol2(int[][] matrix) {
//       int n = matrix.length;
//       for(int i=0; i<n;i++) {
//         for(int j=i; j<n; j++) {

//           int tmp = matrix[i][j];
//           matrix[i][j]=matrix[j][i];
//           matrix[j][i]=tmp;

//         }//for j

//       }//for i
//       //print(matrix);
//       for(int i=0; i<n; i++) {
//         int left=0;
//         int right=n-1;
//         while(left<right) {
//           int tmp=matrix[i][left];
//           matrix[i][left]=matrix[i][right];
//           matrix[i][right]=tmp;
//           left++;
//           right--;
//         }

//       }//for i
//     }
```

Saved to local Ln

Testcase Test Result

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied Very Satisfied

49. Group Anagrams

Solv

Medium Topics Companies

Given an array of strings `strs`, group **the anagrams** together. You can return the answer in **any order**.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Example 1:

```
Input: strs = ["eat","tea","tan","ate","nat","bat"]
Output: [["bat"],["nat","tan"],["ate","eat","tea"]]
```

Example 2:

```
Input: strs = [""]
Output: [[""]]
```

Example 3:

```
Input: strs = ["a"]
Output: [["a"]]
```

Constraints:

- `1 <= strs.length <= 10^4`
- `0 <= strs[i].length <= 100`
- `strs[i]` consists of lowercase English letters.

Seen this question in a real interview before? 1/4

Yes No

Accepted 2.3M Submissions 3.5M Acceptance Rate 67.2%

17.6K 89

Code

Java Auto

```java
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
      Map<String, List<String>> m=new HashMap<>();
      for( String s : strs) {
        char[] chars=s.toCharArray();
        Arrays.sort(chars);
        String key= new String(chars);
        m.computeIfAbsent(key, k->new ArrayList<String>());
        m.get(key).add(s);

      }//for each s

      return new ArrayList(m.values());
    }
}
```

Saved to local

Testcase Test Result

50. Pow(x, n)

Solv

Medium Topics 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:

- $-100.0 < x < 100.0$
- $-2^{31} <= n <= 2^{31}-1$
- `n` is an integer.
- Either `x` is not zero or $n > 0$.
- $-10^4 <= x^n <= 10^4$

Seen this question in a real interview before? 1/4

Yes No

9K 201

Code

Java
Auto

```java
class Solution {
    public double myPow(double x, int n) {
      int i=n;
      double ret = 1.0;
      while(i!=0) {
        if(i%2!=0) {
          ret=ret*x;
        }
        x=x*x;
        i=i/2;

      }
      if(n<0) ret=1/ret;
      return ret;

    }
}
```

Saved to local

Ln

Testcase
Test Result

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied
Very Satisfied

51. N-Queens

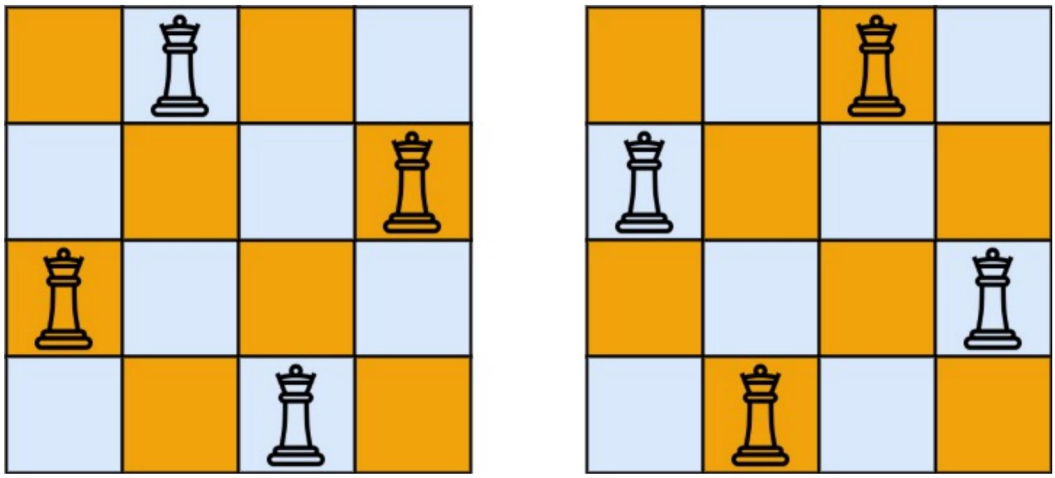
Hard Topics Companies

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:



```
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:

Code

Java Auto

```java
class Solution {
    public List<List<String>> solveNQueens(int n) {
      char[][] board=new char[n][n];
      List<List<String>> ret = new ArrayList<>();
      for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
          board[i][j]='.';
        }
      }
      backtrack(ret, board, 0);

      return ret;
    }
    public void backtrack( List<List<String>> ret, char[][] board, int row ) {
      if(row==board.length){
        ret.add(createStrList(board));
        return;
      }

      for(int col=0; col<board.length; col++) {
        if(isValid(board, row, col)) {
          board[row][col]='Q';
          backtrack(ret, board, row+1);
          board[row][col]='.';
        }
      }
    }

    public boolean isValid( char[][] board, int r, int c) {
      for(int i=0; i<board.length; i++) {
        for(int j=0; j<board.length; j++ ) {
          if(board[i][j]=='Q' && (r+c==i+j || r-c==i-j || r==i || c==j ) )
            return false;
        }
      }

      return true;
    }

    public List<String> createStrList(char[][] board){
      List<String> ret = new ArrayList<>();
      for(int i=0; i<board.length; i++)
        ret.add( new String(board[i]) );

      return ret;
    }
}
```

Saved to local

Testcase Test Result

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied Very Satisfied

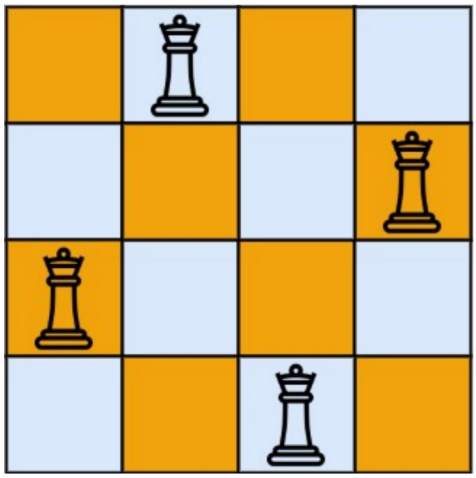
52. N-Queens II

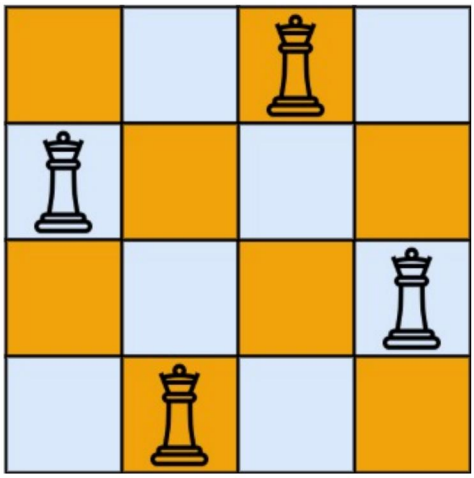
Hard Topics Companies

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 *the number of distinct solutions to the **n-queens puzzle***.

Example 1:





```
Input: n = 4
Output: 2
Explanation: There are two distinct solutions to the 4-queens puzzle as shown.
```

Example 2:

```
Input: n = 1
Output: 1
```

Constraints:

- `1 <= n <= 9`

Code

Java Auto

```java
class Solution {
    public int totalNQueens(int n) {
      char[][] board=new char[n][n];
      int ret=0;
      for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
          board[i][j]='.';
        }
      }
      ret = backtrack(ret, board, 0);

      return ret;

    }


    public int backtrack( int ret, char[][] board, int row ) {
      if(row==board.length){

        return ret+1;
      }

      for(int col=0; col<board.length; col++) {
        if(isValid(board, row, col)) {
          board[row][col]='Q';
          ret = backtrack(ret, board, row+1);
          board[row][col]='.';
        }
      }
      return ret;
    }

    public boolean isValid( char[][] board, int r, int c) {
      for(int i=0; i<board.length; i++) {
        for(int j=0; j<board.length; j++ ) {
          if(board[i][j]=='Q' && (r+c==i+j || r-c==i-j || r==i || c==j ) )
            return false;
        }
      }

      return true;
    }

}
```

Saved to local Ln

Testcase Test Result

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied Very Satisfied

53. Maximum Subarray

Sc

Medium Topics Companies

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

Example 1:

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

Example 2:

```
Input: nums = [1]
Output: 1
Explanation: The subarray [1] has the largest sum 1.
```

Example 3:

```
Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.
```

Constraints:

- `1 <= nums.length <= 10^5`
- `-10^4 <= nums[i] <= 10^4`

Follow up: If you have figured out the `O(n)` solution, try coding another solution using the **divide and conquer** approach, which is n
subtle.

Seen this question in a real interview before? 1/4

32.5K 159

Code

Java Auto

```java
class Solution {
    public int sol1(int[] nums) {
      int sum = nums[0];
      int ret = sum;
      for(int i=1; i<nums.length; i++) {
        sum=Math.max(sum+nums[i], nums[i]);
        ret=Math.max(ret, sum);
      }
      return ret;

    }
    public int maxSubArray(int[] nums) {
      return sol1(nums);
    }
}
```

Saved to local

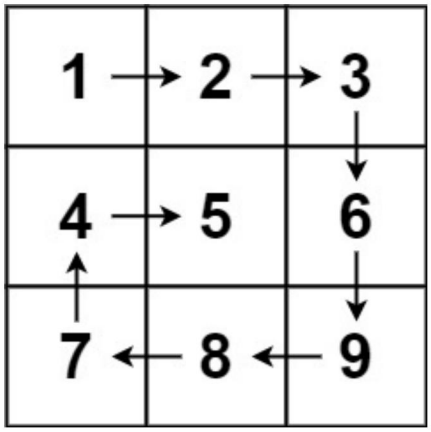
Testcase Test Result

54. Spiral Matrix

Medium Topics Companies Hint

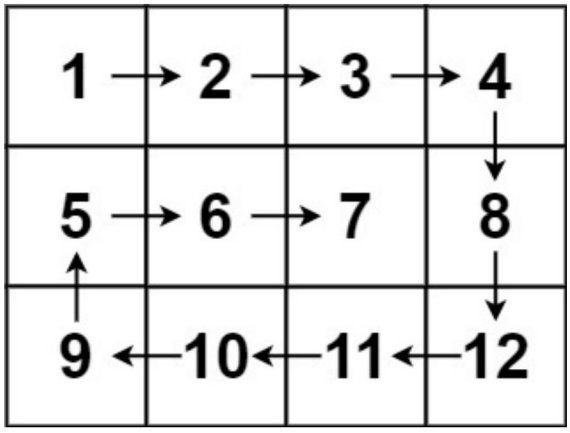
Given an `m x n` `matrix`, return *all elements of the* `matrix` *in spiral order.*

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:




```
Input: matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
```

Code

Java Auto

```java
class Solution {
    public List<Integer> spiralOrder(int[][] matrix) {
      List<Integer> ret = new ArrayList<>();
      int m = matrix.length;
      if(m==0) return ret;
      int n = matrix[0].length;
      if(n==0) return ret;
      int left=0;
      int right=n-1;
      int top=0;
      int buttom=m-1;

      while(true) {
        for(int i=left; i<=right; i++) ret.add(matrix[top][i]);
        if(++top>buttom) break;

        for(int i=top; i<=buttom; i++) ret.add(matrix[i][right]);
        if(--right<left) break;

        for(int i=right; i>=left; i--) ret.add(matrix[buttom][i]);
        if(--buttom<top) break;

        for(int i=buttom; i>=top; i--) ret.add(matrix[i][left]);
        if(++left>right) break;
      }
      return ret;
    }
}
```

Saved to local

Testcase Test Result

55. Jump Game

Medium Topics Companies

You are given an integer array `nums`. You are initially positioned at the array's **first index**, and each element in the array represents yo maximum jump length at that position.

Return `true` *if you can reach the last index, or* `false` *otherwise.*

Example 1:

```
Input: nums = [2,3,1,1,4]
Output: true
Explanation: Jump 1 step from index 0 to 1, then 3 steps to the last index.
```

Example 2:

```
Input: nums = [3,2,1,0,4]
Output: false
Explanation: You will always arrive at index 3 no matter what. Its maximum jump length is 0, whi
makes it impossible to reach the last index.
```

Constraints:

- `1 <= nums.length <= 10^4`
- `0 <= nums[i] <= 10^5`

Seen this question in a real interview before? 1/4

Yes No

Accepted 1.6M Submissions 4.2M Acceptance Rate 38.6%

Topics

18.2K 117

Code

Java
Auto

```java
class Solution {
    public boolean canJump(int[] nums) {
      int maxpos=0;
      int i;
      for(i=0; i<nums.length; i++){
        if(i>maxpos) break;
        maxpos=Math.max(maxpos, i+nums[i]);
      }
      return i>=nums.length;

    }
}
```

Saved to local

Testcase
Test Result

56. Merge Intervals

Medium Topics Companies

Given an array of `intervals` where `intervals[i] = [start_i, end_i]`, merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input.*

Example 1:

```
Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].
```

Example 2:

```
Input: intervals = [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered overlapping.
```

Constraints:

- `1 <= intervals.length <= 10^4`
- `intervals[i].length == 2`
- `0 <= start_i <= end_i <= 10^4`

Seen this question in a real interview before? 1/4

Yes No

Accepted 2.1M Submissions 4.5M Acceptance Rate 46.7%

Topics

Companies

21.1K 63

Code

Java Auto

```java
  public int[][] merge(int[][] intervals) {
    if (intervals.length <= 1)
      return intervals;

    // Sort by ascending starting point
    Arrays.sort(intervals, (i1, i2) -> Integer.compare(i1[0], i2[0]));

    List<int[]> result = new ArrayList<>();
    int[] newInterval = intervals[0];
    result.add(newInterval);

    for (int[] interval : intervals) {
      if (interval[0] <= newInterval[1]) // Overlapping intervals, move the end if needed -- in-place change because newInterval is still the ref to the last element of the list
        newInterval[1] = Math.max(newInterval[1], interval[1]);
      else {
        // Disjoint intervals, add the new interval to the list
        result.add(interval);
        newInterval = interval; //newInterval is still the ref to the last element of the list


      }
    }

    return result.toArray(new int[result.size()][]);
  }
}

// class Solution {
//     public List<Interval> merge(List<Interval> intervals) {
//       Collections.sort(intervals, (x,y)->{ return x.start-y.start;});
//       //Collections.sort(intervals, (x,y)->x.start-y.start); //works too
//       List<Interval> ret = new ArrayList<>();
//       if(intervals.size()==0)
//         return ret;
//       ret.add(intervals.get(0));
//       for(int i=1; i<intervals.size(); i++){
//         if(ret.get(ret.size()-1).end<intervals.get(i).start) {
//           ret.add(intervals.get(i));
//         }else {
//           ret.get(ret.size()-1).end = Math.max(ret.get(ret.size()-1).end, intervals.get(i).end);
//         }
//       }


//       return ret;
//     }
// }
```

Saved to local

Testcase Test Result

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied Very Satisfied

57. Insert Interval

Medium Topics Companies

You are given an array of non-overlapping intervals `intervals` where `intervals[i] = [start_i, end_i]` represent the start and the e the `i^th` interval and `intervals` is sorted in ascending order by `start_i`. You are also given an interval `newInterval = [start, end]` represents the start and end of another interval.

Insert `newInterval` into `intervals` such that `intervals` is still sorted in ascending order by `start_i` and `intervals` still does not h any overlapping intervals (merge overlapping intervals if necessary).

Return `intervals` *after the insertion*.

Example 1:

```
Input: intervals = [[1,3],[6,9]], newInterval = [2,5]
Output: [[1,5],[6,9]]
```

Example 2:

```
Input: intervals = [[1,2],[3,5],[6,7],[8,10],[12,16]], newInterval = [4,8]
Output: [[1,2],[3,10],[12,16]]
Explanation: Because the new interval [4,8] overlaps with [3,5],[6,7],[8,10].
```

Constraints:

- `0 <= intervals.length <= 10^4`
- `intervals[i].length == 2`
- `0 <= start_i <= end_i <= 10^5`
- `intervals` is sorted by `start_i` in **ascending** order.
- `newInterval.length == 2`
- `0 <= start <= end <= 10^5`

Code

Java Auto

```java
class Solution {
    public int[][] insert(int[][] intervals, int[] newInterval) {
        int start = newInterval[0];
        int end = newInterval[1];
        List<int[]> list = new ArrayList<>();
        for (int[]interval : intervals) {
            int curStart = interval[0];
            int curEnd = interval[1];
            if (curEnd < start) {
                list.add(new int[]{curStart, curEnd});
            } else if (curStart > end) {
                list.add(new int[]{start, end});
                start = curStart;
                end = curEnd;
            } else {
                start = Math.min(start, curStart);
                end = Math.max(end, curEnd);
            }
        }
        list.add(new int[]{start, end});
        int[][] res = new int[list.size()][2];
        for (int i = 0; i < list.size(); i++) {
            res[i][0] = list.get(i)[0];
            res[i][1] = list.get(i)[1];
        }
        return res;
    }
}

  // public List<Interval> insert(List<Interval> intervals, Interval newInterval) {
  //     List<Interval> result = new LinkedList<>();
  //     int i = 0;
  //     // add all the intervals ending before newInterval starts
  //     while (i < intervals.size() && intervals.get(i).end < newInterval.start)
  //         result.add(intervals.get(i++));
  //     // merge all overlapping intervals to one considering newInterval
  //     while (i < intervals.size() && intervals.get(i).start <= newInterval.end) {
  //         newInterval = new Interval( // we could mutate newInterval here also
  //                 Math.min(newInterval.start, intervals.get(i).start),
  //                 Math.max(newInterval.end, intervals.get(i).end));
  //         i++;
  //     }
  //     result.add(newInterval); // add the union of intervals we got
  //     // add all the rest
  //     while (i < intervals.size()) result.add(intervals.get(i++));
  //     return result;
  // }
```

Saved to local

Testcase Test Result

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied Very Satisfied

Description Editorial Solutions Submissions

58. Length of Last Word

Easy Topics Companies

Given a string `s` consisting of words and spaces, return *the length of the **last** word in the string.*

A **word** is a maximal substring consisting of non-space characters only.

Example 1:

```
Input: s = "Hello World"
Output: 5
Explanation: The last word is "World" with length 5.
```

Example 2:

```
Input: s = "   fly me   to   the moon  "
Output: 4
Explanation: The last word is "moon" with length 4.
```

Example 3:

```
Input: s = "luffy is still joyboy"
Output: 6
Explanation: The last word is "joyboy" with length 6.
```

Constraints:

- `1 <= s.length <= 10^4`
- `s` consists of only English letters and spaces `' '`.
- There will be at least one word in `s`.

Seen this question in a real interview before? 1/4

Yes No

4.3K 136

Problem List Run Submit 0

Code

Java Auto

```java
class Solution {
    public int lengthOfLastWord(String s) {
        int len = 0, tail = s.length() - 1;
        while (tail >= 0 && s.charAt(tail) == ' ') tail--;
        while (tail >= 0 && s.charAt(tail) != ' ') {
            len++;
            tail--;
        }
        return len;
    }


}
```

Saved to local

Testcase Test Result

How satisfied are you with the new feature - Dynamic Layout?

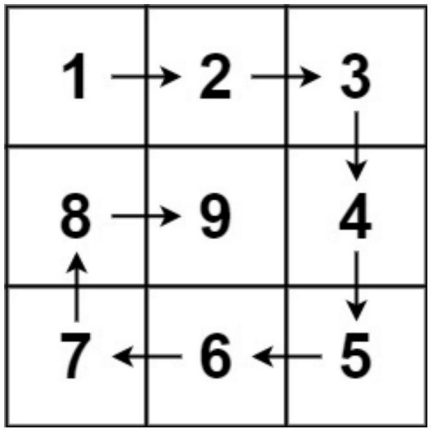
Very Dissatisfied Very Satisfied

59. Spiral Matrix II

Medium Topics Companies

Given a positive integer `n`, generate an `n x n` `matrix` filled with elements from `1` to n^2 in spiral order.

Example 1:




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

Example 2:

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

Constraints:

- `1 <= n <= 20`

Seen this question in a real interview before? 1/4

Yes No

6.1K 50

How satisfied are you with the new feature - Dynamic Layout?

Very Dissatisfied Very Satisfied

Code

Java Auto

```java
class Solution {
    public int[][] generateMatrix(int n) {
      int[][] ret=new int[n][n];
      int left=0;
      int right=n-1;
      int top=0;
      int buttom=n-1;
      int num=1;
      while(true) {
        for(int i=left; i<=right; i++) ret[top][i]=num++;
        if(num>n*n || ++top > buttom) break;
        for(int i=top; i<=buttom; i++) ret[i][right]=num++;
        if(num>n*n || --right<left) break;
        for(int i=right; i>=left; i--) ret[buttom][i]=num++;

        if(num>n*n || --buttom<top) break;

        for(int i=buttom; i>=top; i--) ret[i][left]=num++;
        if(num>n*n || ++left>right) break;
      }
      return ret;
    }
}
```

Saved to local Ln

Testcase Test Result