Description | Editorial | Solutions | Submissions

509. Fibonacci Number

Easy | Topics | 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.
```

Example 3:

```
Input: n = 4
Output: 3
Explanation: F(4) = F(3) + F(2) = 2 + 1 = 3.
```


Code

Java | Auto

```java
class Solution {
    public int fib(int n) {
        if(n<=0)
        return n;
        int a= 0, b=1;
        for (int i=2;i<=n;i++){
            int c = a+b;
            a=b;
            b=c;
        }
        return b;
    }
}
```

Saved | Ln 11, Col 18

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

n =
2

Output

1

Description | Editorial | Solutions | Submissions

50. Pow(x, n)

Solved

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.

Code

Java | Auto

```java
class Solution {
    public double myPow(double x, int n) {
        return Math.pow(x,n);
    }
}
```

Saved | Ln 3, Col 30

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

x =
2.00000

n =
10

Description | Editorial | Solutions | Submissions

125. Valid Palindrome

Solved

Easy | Topics | Companies

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string `s`, return `true` *if it is a* ***palindrome****, or* `false` *otherwise.*

Example 1:

> **Input:** s = "A man, a plan, a canal: Panama"
> **Output:** true
> **Explanation:** "amanaplanacanalpanama" is a palindrome.

Example 2:

> **Input:** s = "race a car"
> **Output:** false
> **Explanation:** "raceacar" is not a palindrome.

Example 3:

> **Input:** s = " "
> **Output:** true
> **Explanation:** s is an empty string "" after removing non-alphanumeric characters.
> Since an empty string reads the same forward and

Code

Java | Auto

```java
class Solution {
    public boolean isPalindrome(String s) {
        int l = 0;
        int r = s.length() - 1;

        while (l < r) {
            while (l < r && !Character.isLetterOrDigit(s.charAt(l))) l++;
            while (l < r && !Character.isLetterOrDigit(s.charAt(r))) r--;

            if (Character.toLowerCase(s.charAt(l)) !=
                Character.toLowerCase(s.charAt(r))) {
                return false;
            }

            l++;
            r--;
        }

        return true;
    }
}
```

Saved

Ln 22, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

s =
"A man, a plan, a canal: Panama"

Description | Accepted | Editorial | Solutions | Submissi

39. Combination Sum

Solved

Medium | Topics | Companies

Given an array of **distinct** integers `candidates` and a target integer `target`, return *a list of all **unique combinations** of* `candidates` *where the chosen numbers sum to* `target`. You may return the combinations in **any order**.

The **same** number may be chosen from `candidates` an **unlimited number of times**. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

The test cases are generated such that the number of unique combinations that sum up to `target` is less than `150` combinations for the given input.

Example 1:

```
Input: candidates = [2,3,6,7], target = 7
Output: [[2,2,3],[7]]
Explanation:
2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that
2 can be used multiple times.
7 is a candidate, and 7 = 7.
These are the only two combinations.
```

Example 2:

```
Input: candidates = [2,3,5], target = 8
Output: [[2,2,2,2],[2,3,3],[3,5]]
```

20.7K | 217 | 442 Online

Code

Java | Auto

```java
import java.util.*;

class Solution {
    public List<List<Integer>> combinationSum(int[] arr, int target) {
        List<List<Integer>> ans = new ArrayList<>();
        solve(arr, target, 0, new ArrayList<>(), ans);
        return ans;
    }
    void solve(int[] arr, int target, int i, List<Integer> list, List<List<Integer>> ans) {
        if (target == 0) {
            ans.add(new ArrayList<>(list));
            return;
        }
        if (target < 0 || i == arr.length) return;

        list.add(arr[i]);
        solve(arr, target - arr[i], i, list, ans);
        list.remove(list.size() - 1);
        solve(arr, target, i + 1, list, ans);
    }
```

Saved | Ln 8, Col 6

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

candidates =

[2,3,6,7]

Description | Editorial | Solutions | Submissions

22. Generate Parentheses

Medium | Topics | Companies

Given `n` pairs of parentheses, write a function to *generate all combinations of well-formed parentheses.*

Example 1:

```
Input: n = 3
Output: ["((()))","(()())","(())()","()(())","()()()"]
```

Example 2:

```
Input: n = 1
Output: ["()"]
```

Constraints:

- `1 <= n <= 8`

Seen this question in a real interview before? 1/5

Code

Java | Auto

```java
import java.util.*;
class Solution {
    public List<String> generateParenthesis(int n) {
        List<String> ans = new ArrayList<>();
        help(ans, "", n, n);
        return ans;
    }
    void help(List<String> ans, String s, int open, int close) {
        if (open == 0 && close == 0) {
            ans.add(s);
            return;
        }
        if (open > 0) {
            help(ans, s + "(", open - 1, close);
        }
        if (close > open) {
            help(ans, s + ")", open, close - 1);
        }
    }
}
```

Saved | Ln 1, Col 20

Testcase | Test Result

Accepted Runtime: 2 ms

Case 1 | Case 2

Input

n =
3

Description | Editorial | Solutions | Submissions

17. Letter Combinations of a Phone Number

Medium | Topics | 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"]
```

20.7K | 368 | 282 Online

Code

Java | Auto

```java
import java.util.*;
class Solution {
    String[] map = {
        "", "", "abc", "def", "ghi", "jkl",
        "mno", "pqrs", "tuv", "wxyz"
    };
    public List<String> letterCombinations(String digits) {
        List<String> ans = new ArrayList<>();
        if (digits.length() == 0) return ans;
        solve(digits, 0, "", ans);
        return ans;
    }
    void solve(String digits, int i, String cur, List<String> ans) {
        if (i == digits.length()) {
            ans.add(cur);
            return;
        }
        String letters = map[digits.charAt(i) - '0'];
        for (char c : letters.toCharArray()) {
            solve(digits, i + 1, cur + c, ans);
        }
```

Saved | Ln 18, Col 54

Testcase | Test Result

Accepted Runtime: 2 ms

Case 1 | Case 2

Input

digits =
"23"