# “Week 1 - 100 Days DSA Challenge”

## WEEK 1 OF MY 100-DAY DSA CHALLENGE COMPLETED!

I’m excited to share that I’ve successfully completed the first 7 days of my 100-day Data Structures and Algorithms (DSA) challenge on LeetCode. Each day, I’ve tackled a new problem using Java, and it’s been an incredible learning experience so far. Here are the problems I’ve solved this week:

## Problem 1:Palindrome Number(Day-1)

Code

Java Auto

```java
class Solution {
    public boolean isPalindrome(int x) {
        int n=x;
        int reveNumber=0;
        int rem=0;
        if(n<0){
            return false;
        }

        while(n!=0){
            rem=n%10;
            reveNumber=reveNumber*10 + rem;
            n=n/10;
        }

        if(x==reveNumber){
           return true;
        }else{
            return false;
        }
    }
}
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

x =
121

Output

true

Expected

true

Description | Accepted | Editorial

All Submissions

Accepted

submitted at Sep 0

Editorial Solution

Runtime

4 ms | Beats 100.00%

Analyze Complexity

# “Week 1 - 100 Days DSA Challenge”

## Problem 2:Power of two(Day-2)

Problem List

Code

Java Auto

```java
class Solution {

    public boolean isPowerOfTwo(int n) {
        if(n<1){
            return false;
        }else if(n==1){
            return true;
        }else if((n & n-1)==0){
            return true;
        }
        else{
            return false;
        }
    }
}
```

Saved

Description | Accepted | Editorial

All Submissions

Accepted

submitted at Sep 0

Editorial | Solution

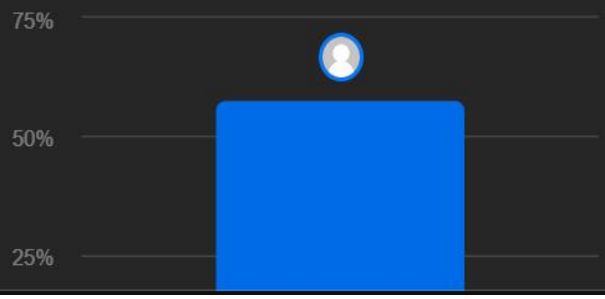
Runtime

1 ms | Beats 42.19%

Analyze Complexity

Memory

40.58 MB | Beats 80.73%




Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

n =
1

Output

true

Expected

true

Contribute a testcase

# “Week 1 - 100 Days DSA Challenge”

## Problem 3:Fibonacci Number(Day-3)

Code

Java | Auto

```java
class Solution {
    public int fib(int n) {
        int a=0;
        int b=1;
        int temp=0;

            if(n==0){
                return a;
            }
            else if(n==1){
                return b;
            }
            for(int i=0; i<=n-2; i++){
                temp=a+b;
                a=b;
                b=temp;
            }
            return b;
    }
}
```

Description | Accepted | Editorial

All Submissions

Accepted

submitted at Sep 0

Editorial | Solution

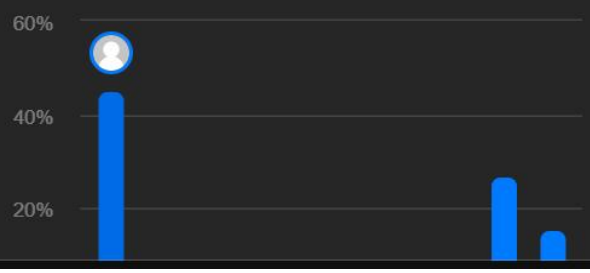
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

40.01 MB | Beats 81.39%




Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

n =
2

Output

1

Expected

1

Contribute a testcase

# “Week 1 - 100 Days DSA Challenge”

Problem List

Code

Java | Auto

```java
class Solution {
    public int tribonacci(int n) {
        int fstNum=0;
        int secNum=1;
        int thrdNum=fstNum + secNum;
        int temp;

        if(n==fstNum){
            return fstNum;

        }
         else if(n==secNum){
            return secNum;
        }
        else if(n==thrdNum){
            return thrdNum;
        }

        for(int i=fstNum;i<=n-3;i++){
            temp=thrdNum + secNum + fstNum;

            fstNum=secNum;
```

Saved

```java
        for(int i=fstNum;i<=n-3;i++){
            temp=thrdNum + secNum + fstNum;

            fstNum=secNum;
            secNum=thrdNum;
            thrdNum=temp;
        }
        return thrdNum;
    }
}
```

Description | Accepted | Editorial | Solution

All Submissions

Accepted

submitted at Sep 0

Editorial | Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

39.74 MB | Beats 98.61%

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

n =

4

Output

4

Expected

4

# “Week 1 - 100 Days DSA Challenge”

## Problem 5:Water Bottels(Day-5)

There are `numBottles` water bottles that are initially full of water. You can exchange `numExchange` empty water bottles from the market with one full water bottle.

The operation of drinking a full water bottle turns it into an empty bottle.

Given the two integers `numBottles` and `numExchange`, return *the **maximum** number of water bottles you can drink.*

Code

Java Auto

```java
class Solution {
    public int numWaterBottles(int numBottles, int numExchange) {

            int drinkedBottle=numBottles;
        while(numBottles>=numExchange){
            drinkedBottle = drinkedBottle+numBottles/numExchange;
            numBottles=(numBottles / numExchange) + (numBottles % numExchange);
        }
        return drinkedBottle;


    }
}
```

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

numBottles =
9

numExchange =
3

Output

13

Expected

13

Description | Accepted | Editorial

All Submissions

Accepted
submitted at Sep 0

Editorial Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

40.09 MB | Beats 88.87%

# “Week 1 - 100 Days DSA Challenge”

## Problem 6: Two Sum(Day-6)

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Problem List

Code

Java | Auto

```java
class Solution {
    public int[] twoSum(int[] nums, int target) {

        int result[]=new int[2];

        for(int i=0;i<nums.length;i++){

            for(int j=i+1;j<nums.length;j++){

                if(nums[i] + nums[j] == target){

                    result[0]=i;
                    result[1]=j;
                    return result;

                }
            }
        }

        return result;
    }
    public static void main(String[]arg){
```

Saved

Description | Accepted | Editorial | Solution

All Submissions

Accepted

submitted at Sep 0

Editorial | Solution

Runtime

45 ms | Beats 30.06%

Analyze Complexity

Memory

44.91 MB | Beats 32.00%

40%

20%

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

nums =

[2,7,11,15]

target =

9

Output

[0,1]

Expected

[0,1]

# “Week 1 - 100 Days DSA Challenge”

## Problem 7: Add to Array-Form of Integer(Day-7)

Easy | Topics | Companies

The **array-form** of an integer `num` is an array representing its digits in left to right order.

- For example, for `num = 1321`, the array form is `[1,3,2,1]`.

Given `num`, the **array-form** of an integer, and an integer `k`, return *the **array-form** of the integer* `num + k`.

Problem List

Code

Java | Auto

```java
class Solution {
    public List<Integer> addToArrayForm(int[] num, int k){
        List<Integer> list =new ArrayList<>();
            int p=num.length-1;
            int sum=0;
            int carry=0;

        while(p>=0 || k>0){
            int numVal=0;
            if(p>=0){
                numVal=num[p];
            }
            int d=k%10;
            sum=d + carry + numVal;
            int digit=sum%10;
            carry=sum/10;
            list.add(digit);
            k=k/10;
            p--;
        }
         if(carry>0){
            list.add(carry);
```


```java
                p--;
            }
             if(carry>0){
                list.add(carry);
             }
```


```java
             Collections.reverse(list);

            return list;



        }

}
```

Description | Accepted | Editorial

All Submissions

Accepted

submitted at Sep 0

Editorial | Solution

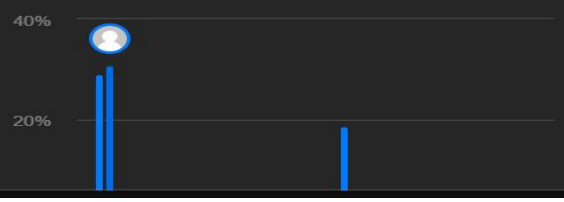
Runtime

3 ms | Beats 69.29%

Analyze Complexity

Memory

45.72 MB | Beats 41.45%




Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

num =

[1,2,0,0]

k =

34

Output

[1,2,3,4]

Expected

[1,2,3,4]