

“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

1 class Solution {
2     public boolean isPalindrome(int x) {
3         int n=x;
4         int reveNumber=0;
5         int rem=0;
6         if(n<0){
7             return false;
8         }
9
10        while(n!=0){
11            rem=n%10;
12            reveNumber=reveNumber*10 + rem;
13            n=n/10;
14        }
15
16        if(x==reveNumber){
17            return true;
18        }else{
19            return false;
20        }
21    }
22 }
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

x =
121

Output

true

Expected

true

Description | Accepted x | Editorial | <

← All Submissions

Accepted submitted at Sep 0'

Editorial Solution

🕒 Runtime

4 ms | Beats 100.00% 🌱

🔗 Analyze Complexity

“Week 1 - 100 Days DSA Challenge”

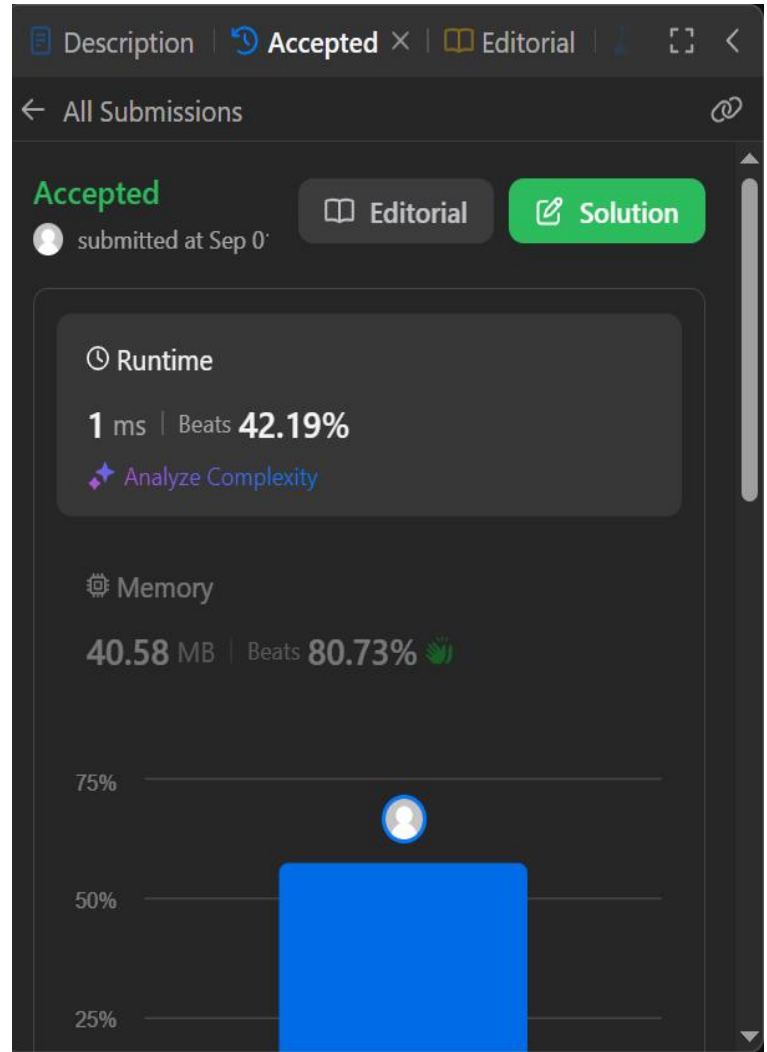
Problem 2:Power of two(Day-2)



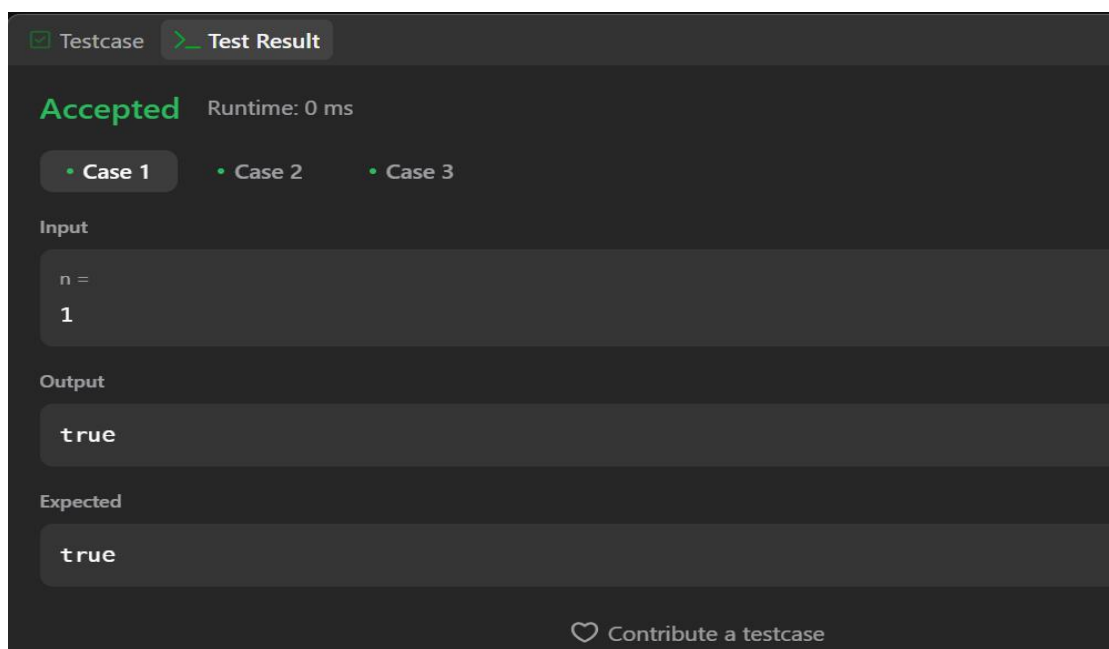
The screenshot shows a code editor with a dark theme. At the top, there's a navigation bar with a home icon, a menu icon, and the text 'Problem List'. Below this, there's a 'Code' tab with a code icon. The language is set to 'Java' and 'Auto' is checked. The code is as follows:

```
1 class Solution {
2
3     public boolean isPowerOfTwo(int n) {
4         if(n<1){
5             return false;
6         }else if(n==1){
7             return true;
8         }else if((n & n-1)==0){
9             return true;
10        }
11        else{
12            return false;
13        }
14    }
15 }
```

At the bottom left, it says 'Saved'.



The screenshot shows the submission page for the 'Power of two' problem. At the top, there's a navigation bar with 'Description', 'Accepted', 'Editorial', and a link icon. Below this, there's a 'All Submissions' section. The submission status is 'Accepted' and it was submitted at 'Sep 0'. There are buttons for 'Editorial' and 'Solution'. The 'Runtime' section shows '1 ms' and 'Beats 42.19%', with a link to 'Analyze Complexity'. The 'Memory' section shows '40.58 MB' and 'Beats 80.73%'. Below this, there's a bar chart showing the performance of the submission compared to other submissions. The chart has a y-axis with labels '25%', '50%', and '75%'. A blue bar represents the submission's performance, reaching approximately 50% on the y-axis. A user profile icon is shown above the bar.



The screenshot shows the test result page for the 'Power of two' problem. At the top, there's a navigation bar with 'Testcase' and 'Test Result'. The submission status is 'Accepted' and the runtime is '0 ms'. There are three test cases: 'Case 1', 'Case 2', and 'Case 3'. The 'Input' section shows 'n = 1'. The 'Output' section shows 'true'. The 'Expected' section shows 'true'. At the bottom, there's a link to 'Contribute a testcase'.

“Week 1 - 100 Days DSA Challenge”

Problem 3: Fibonacci Number(Day-3)

```
</> Code
Java ▾ 🔒 Auto

1 class Solution {
2     public int fib(int n) {
3         int a=0;
4         int b=1;
5         int temp=0;
6
7         if(n==0){
8             return a;
9         }
10        else if(n==1){
11            return b;
12        }
13        for(int i=0; i<=n-2; i++){
14            temp=a+b;
15            a=b;
16            b=temp;
17        }
18        return b;
19    }
20 }
```

Description Accepted × Editorial 🔗 <

← All Submissions 🔗

Accepted submitted at Sep 0'

Editorial Solution

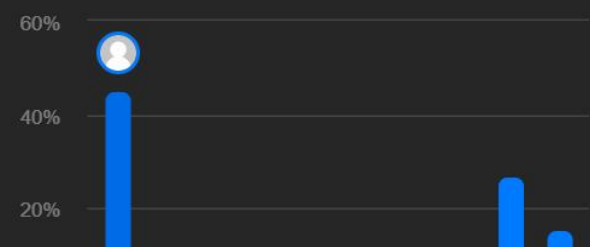
🕒 Runtime

0 ms | Beats 100.00% 🌿

🔧 Analyze Complexity

⚙️ Memory

40.01 MB | Beats 81.39% 🌿



Memory Usage (%)
60%
40%
20%

☑️ 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

```
1 class Solution {
2     public int tribonacci(int n) {
3         int fstNum=0;
4         int secNum=1;
5         int thrdNum=fstNum + secNum;
6         int temp;
7
8         if(n==fstNum){
9             return fstNum;
10        }
11        else if(n==secNum){
12            return secNum;
13        }
14        else if(n==thrdNum){
15            return thrdNum;
16        }
17
18        for(int i=fstNum;i<=n-3;i++){
19            temp=thrdNum + secNum + fstNum;
20
21            fstNum=secNum;
22        }
```

Saved

```
        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 Bottles(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

```
1 class Solution {
2     public int numWaterBottles(int numBottles, int numExchange) {
3
4         int drankedBottle=numBottles;
5         while(numBottles>=numExchange){
6             drankedBottle = drankedBottle+numBottles/numExchange;
7             numBottles=(numBottles / numExchange) + (numBottles % numExchange);
8         }
9         return drankedBottle;
10
11     }
12 }
13 }
```

☑ 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

```
1 class Solution {
2     public int[] twoSum(int[] nums, int target) {
3
4         int result[] = new int[2];
5
6         for(int i=0; i<nums.length; i++){
7
8             for(int j=i+1; j<nums.length; j++){
9
10                if(nums[i] + nums[j] == target){
11
12                    result[0]=i;
13                    result[1]=j;
14                    return result;
15                }
16            }
17        }
18
19        return result;
20    }
21
22    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

```
1 class Solution {
2     public List<Integer> addToArrayForm(int[] num, int k){
3         List<Integer> list =new ArrayList<>();
4         int p=num.length-1;
5         int sum=0;
6         int carry=0;
7
8         while(p>=0 || k>0){
9             int numVal=0;
10            if(p>=0){
11                numVal=num[p];
12            }
13            int d=k%10;
14            sum=d + carry + numVal;
15            int digit=sum%10;
16            carry=sum/10;
17            list.add(digit);
18            k=k/10;
19            p--;
20        }
21        if(carry>0){
22            list.add(carry);
23            p--;
24        }
25        if(carry>0){
26            list.add(carry);
27        }
28        Collections.reverse(list);
29        return list;
30    }
31 }
32
33 }
```

Description Accepted Editorial

All Submissions

Accepted

submitted at Sep 0

Editorial

Solution

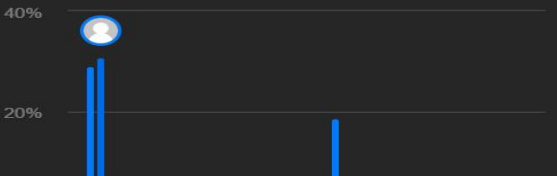
Runtime

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]