

## Professional Development Skills Session-3

--Digit Concept  
--Prefix sum Approach  
--Practice programs

### I) Digit concept

<div style="border: 2px solid red; padding: 10px; margin-bottom: 10px;"> <math display="block">7789 \% 10 = 9</math> <math display="block">\begin{array}{r} 7789 \\ 10 \end{array} \rightarrow 778 \% 10 = 8</math> <math display="block">\begin{array}{r} 778 \\ 10 \end{array} \rightarrow 77 \% 10 = 7</math> <math display="block">\begin{array}{r} 77 \\ 10 \end{array} \rightarrow 7 \% 10 = 7</math> <math display="block">\begin{array}{r} 7 \\ 10 \end{array} \rightarrow 0</math> </div>	<p><b>Pseudocode:-</b></p> <pre> while(N &gt; 0){     // Extract the last digit of N     int lastDigit = N % 10;      SOP(lastDigit);      // Remove the last digit from N     N = N / 10; }         </pre> <p style="color: red; font-weight: bold;">Extracts the digits in reverse order</p>
--	--

### 1. Count Digits:-

#### Problem statement

You are given a number 'n'.  
Find the number of digits of 'n' that evenly divide 'n'.  
Note:  
A digit evenly divides 'n' if it leaves no remainder when dividing 'n'.  
Example:  
Input: 'n' = 336  
Output: 3

```

public static int countDigits(int n) {
    // Initialize a counter variable
    int cnt = 0;
    while (n > 0) {
        // Increment the counter for each digit
        cnt = cnt + 1;
        // Divide 'n' by 10 to remove the last digit.
        n = n / 10;
    }
    // Return the count of digits.
    return cnt;
}
        
```

**Time Complexity:**  $O(\log_{10}N + 1)$ , in the while loop we divide N by 10 until it becomes 0 which takes  $\log_{10}N$  iterations and +1 is added for the assumption N is divisible by 10.  
**Space Complexity :**  $O(1)$

#### Optimal Approach

```

static int countDigits(int n) {
    // Initialize a variable 'cnt' to
    int cnt = (int) (Math.log10(n) + 1);

    // The expression (int)(Math.log10(n) + 1)
    // calculates the number of digits in 'n'
    // and casts it to an integer.
    // Adding 1 to the result accounts
    // for the case when 'n' is a power of 10,

    // Return the count of digits in 'n'.
    return cnt;
}
        
```

**Time Complexity:**  $O(1)$   
**Space Complexity :**  $O(1)$

**NOTE:-** If the time complexity depends on the division , then time complexity will be logarithmic

## 2. Reversing a Number:-

Input Number:	Reversed Number:	<u>Pseudocode:-</u>
$7789 \% 10 = 9$	$0 \times 10 + 9 = 9$	<b>while</b> (N > 0){
$\frac{7789}{10} \rightarrow 778$ $778 \% 10 = 8$	$9 \times 10 + 8 = 98$	// Extract the last digit of N
$\frac{778}{10} \rightarrow 77$ $77 \% 10 = 7$	$98 \times 10 + 7 = 987$	<b>int</b> lastDigit = N % 10;
$\frac{77}{10} \rightarrow 7$ $7 \% 10 = 7$	$987 \times 10 + 7 = 9877$	revNum = (revNum * 10) + lastDigit;
$\frac{7}{10} \rightarrow 0$		// Remove the last digit from N
		N = N / 10;
		}

### Problem statement

There is a song concert going to happen in the city. The price of each ticket is equal to the number obtained after reversing the bits of a given 32 bits unsigned integer 'n'.

Example 1:

Input: x = 123

Output: 321

```
public static void reverseDigits(int n) {
```

```
    int revNum=0;
```

```
    while (n > 0) {
```

```
        // Extract last digit
```

```
        int lastDigit = n%10;
```

```
        revNum = (revNum * 10) + lastDigit;
```

```
        // Divide 'n' by 10 to remove the last digit.
```

```
        n = n / 10;
```

```
    }
```

```
    System.out.println(revNum);
```

```
}
```

**Time Complexity:**  $O(\log_{10} N + 1)$ , in the while loop we divide N by 10 until it becomes 0 which takes  $\log_{10} N$  iterations and +1 is added for the assumption N is divisible by 10.

**Space Complexity :**  $O(1)$

### 3. Palindrome

#### Problem statement

Check whether a given number ' $n$ ' is a palindrome number.

**Example:** Input: 'n' = 51415 Output: true

```
public static boolean palindrome(int n) {  
    int dup=n;  
    int revNum=0;  
    while (n > 0) {  
        // Extract last digit  
        int lastDigit = n%10;  
        revNum = (revNum * 10) + lastDigit;  
        // Divide 'n' by 10 to remove the last digit.  
        n = n / 10;  
    }  
    if(dup==revNum)  
        return true;  
    else  
        return false;  
}
```

**Time Complexity:**  $O(\log_{10} N + 1)$ , in the while loop we divide N by 10 until it becomes 0 which takes  $\log_{10} N$  iterations and +1 is added for the assumption N is divisible by 10.

**Space Complexity :**  $O(1)$

### 4. Armstrong Number

#### Problem statement

You are given an integer ' $n$ '. Return '*true*' if ' $n$ ' is an Armstrong number, and '*false*' otherwise.

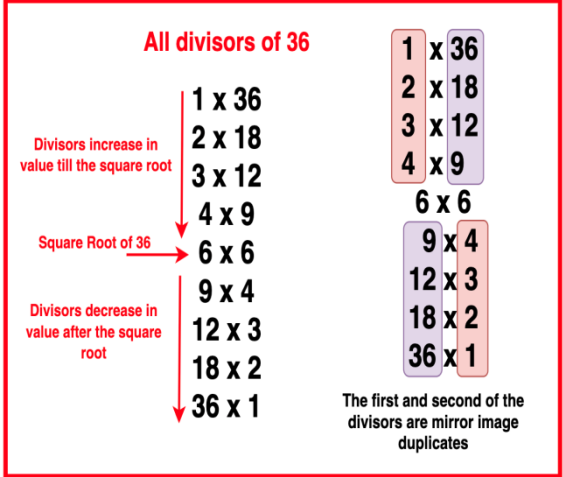
An Armstrong number is a number (with ' $k$ ' digits) such that the sum of its digits raised to ' $k$ 'th power is equal to the number itself. For example, 371 is an Armstrong number because  $3^3 + 7^3 + 1^3 = 371$ .

```
import java.lang.Math;  
public static boolean isArmstrong(int num) {  
    int k = String.valueOf(num).length();  
    int sum = 0;  
    int n = num;  
    while(n > 0){  
        int ld = n % 10;  
        sum += Math.pow(ld, k);  
        n = n / 10;  
    }  
    return sum == num ? true : false;  
}
```

**Time Complexity:**  $O(\log_{10} N + 1)$ , in the while loop we divide N by 10 until it becomes 0 which takes  $\log_{10} N$  iterations and +1 is added for the assumption N is divisible by 10.

**Space Complexity : O(1)**

### 5.print All Divisors

 <p><b>All divisors of 36</b></p> <p>Divisors increase in value till the square root</p> <p>Square Root of 36</p> <p>Divisors decrease in value after the square root</p> <p>The first and second of the divisors are mirror image duplicates</p>	<pre>for (int i = 1; i &lt;= <b>sqrtN</b>; ++i) {     if (n % i == 0) {         divisors.add(i);          if (i != n / i) {             <b>divisors.add(n / i);</b>         }     } }</pre>
<pre>public static int[] printDivisors(int n, int[] size) {     int[] divisors = new int[n];     int count = 0;      for (int i = 1; i &lt;= n; i++) {         if (n % i == 0) {             divisors[count++] = i;         }     }     size[0] = count;     return divisors; }</pre> <p><b>Time Complexity: O(N)</b> <b>Space Complexity : O(N)</b></p>	<pre>public static ArrayList&lt;Integer&gt; findDivisors(int n) {     <b>ArrayList&lt;Integer&gt; divisors = new ArrayList&lt;&gt;();</b>      // Iterate up to the square root of n     <b>int sqrtN = (int) Math.sqrt(n);</b>      for (int i = 1; i &lt;= sqrtN; ++i) {         if (n % i == 0) {             divisors.add(i);              if (i != n / i) {                 <b>divisors.add(n / i);</b>             }         }     }      return divisors; }</pre> <p><b>Time Complexity: O(N/2)</b> <b>Space Complexity : O(N)</b></p>

### 6.check for prime

#### Problem statement

A prime number is a positive integer that is divisible by exactly 2 integers, 1 and the number itself.

You are given a number 'n'.

Find out whether 'n' is prime or not.

Example :

Input: 'n' = 5    Output: YES

<pre> static boolean checkPrime(int n) {     int cnt = 0;     for (int i = 1; i &lt;= n; i++) {         if (n % i == 0) {             cnt = cnt + 1;         }     }     if (cnt == 2)         return true;     else         return false; } </pre> <p><b>Time Complexity: <math>O(N)</math></b>  <b>Space Complexity : <math>O(1)</math></b></p>	<pre> static boolean checkPrime(int n){     int cnt = 0;      for(int i = 1; i &lt;= Math.sqrt(n); i++){         if(n % i == 0){             cnt = cnt + 1;             if(n / i != i){                 cnt = cnt + 1;             }         }     }      if(cnt == 2)         return true;     else         return false; } </pre> <p><b>Time Complexity: <math>O(\sqrt{N})</math></b>  <b>Space Complexity : <math>O(1)</math></b></p>
---	--

# Prefix sum

Number of subarrays having sum equal to k

$K = 6$

The number of subarrays with sum equal to k is 3

A subarray sum is the sum of contiguous elements in a subarray

$sum(0, 3) = 9$   
 $sum(0, 1) = 4$   
 $sum(2, 3) = sum(0, 3) - sum(0, 1) = 9 - 4 = 5$

## Example :-

Num[] = {1, 2, 3, -3, 1, 1, 1, 4, 2, -3};

k=3;

HashMap

9	1
12	1
10	1
5	1
4	1
6	1+1
3	1+1
1	1
0	1

**Count=8**

## Problem statement

You are given an integer array 'arr' of size 'N' and an integer 'K'.

Your task is to find the total number of subarrays of the given array whose sum of elements is equal to k.

A subarray is defined as a contiguous block of elements in the array.

Example:

Input: 'N' = 4, 'arr' = [3, 1, 2, 4], 'K' = 6

Output: 2

Explanation: The subarrays that sum up to '6' are: [3, 1, 2], and [2, 4].

```
public static int
findAllSubarraysWithGivenSum(int arr[], int k) {
    int n = arr.length; // size of the given array.
    int cnt = 0; // Number of subarrays:

    for (int i = 0 ; i < n; i++) { // starting index i
        for (int j = i; j < n; j++) { // ending index j

            // calculate the sum of subarray [i...j]
            int sum = 0;
            for (int K = i; K <= j; K++)
                sum += arr[K];

            // Increase the count if sum == k:
            if (sum == k)
                cnt++;
        }
    }
    return cnt;
}
```

**Time Complexity:  $O(N^3)$**

**Space Complexity :  $O(1)$**

```
public static int
findAllSubarraysWithGivenSum(int arr[], int k) {
    int n = arr.length; // size of the given array.
    int cnt = 0; // Number of subarrays:

    for (int i = 0 ; i < n; i++) { // starting index i
        int sum = 0;
        for (int j = i; j < n; j++) { // ending index j

            // calculate the sum of subarray [i...j]
            sum += arr[K];

            // Increase the count if sum == k:
            if (sum == k)
                cnt++;
        }
    }
    return cnt;
}
```

**Time Complexity:  $O(N^2)$**

**Space Complexity :  $O(1)$**

```
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
public class Main {
    public static void main(String[] args) {
        int num2[] = {1,2,3,-3, 1,1,1,4,2,-3};
        int k=3;
        System.out.println(subarraySum(num2,k));
    }
}
```

```
public static int subarraySum(int[] nums, int k) {
    int ans = 0;
    int prefix = 0;
    Map<Integer, Integer> count = new HashMap<>();
}
```

```

        count.put(0, 1);
        for (final int num : nums) {
            prefix += num;
            ans += count.getDefault(prefix - k, 0);
            count.merge(prefix, 1, Integer::sum);
        }
        return ans;
    }
}

```

**Time Complexity:  $O(N)$  or  $O(N \cdot \log N)$**

**Space Complexity :  $O(N)$**

### Practice Problems

---

1. Ramesh's basic salary is input, his dearness allowance is 40% of basic salary, and house rent allowance is 20% of basic salary.
2. The distance between two cities (in km.) is input. Program to convert and print this distance in meters, feet, inches and centimeters
3. If the marks obtained by a Student in five different subjects are input, write a c , java, python program to find the aggregate marks and percentage marks obtained by the student assuming the maximum marks as 100
4. Temperature of a city in foreign degrees is input, write a c, java, python program to convert this temperature into centigrade degrees.
5. The length and breadth of a rectangle and radius of a circle is input, write a c, java, python program to calculate area , perimeter of rectangle and the area , circumference of the circle.
6. If the total selling price of 15 items and the total profit earned on them is input, write a c, java, python program to find the cost price of one item
7. If a four digit number is input, write a c, java and python program to calculate the sum of its digits
8. If a four digit number is input, write a c, java and python program to reverse the number
9. if four digit number is input, write a c, java and python program to obtain the sum of first and last digit of the given number
10. Consider the currency system in which there are notes of six denominations namely Re 1, Rs 2, Rs 5, Rs 10, Rs 50, Rs 100. If a sum of Rs N is entered as input. Write a c , java, python program to compute the smallest number of notes that will combine to give Rs N.
11. If cost price and selling price of an item is input, write a program to determine whether the seller has made profit or incurred loss. Also determine how much profit he made or loss he incurred
12. Any integer is input, write a c, java and python program to find out whether it is even or odd number.