CS263

ASSIGNMENT 3

NAME:

ARCHIT AGRAWAL

ROLL NO.:

202052307

SECTION:

A

Problem

John wants to put everything in its proper order. He follows his rule that all numbers must be arranged in ascending sequence. Unfortunately, this isn't always the case. A "violation" circumstance, according to him, is one in which a lower number appears after a greater number in the collection, violating the ascending order. Determine the total number of such violations given a set of integers.

Example:

Input: {2, 4, 1, 3, 5}

Output: 3

Explanation: Violations are - {2,1}, {4, 1} and {4, 3}:- total 3 violations

Algorithm

Recursive algorithm for the above problem. The name of the method(function) is: violations(int[] arr, int index). It returns the number of violations.

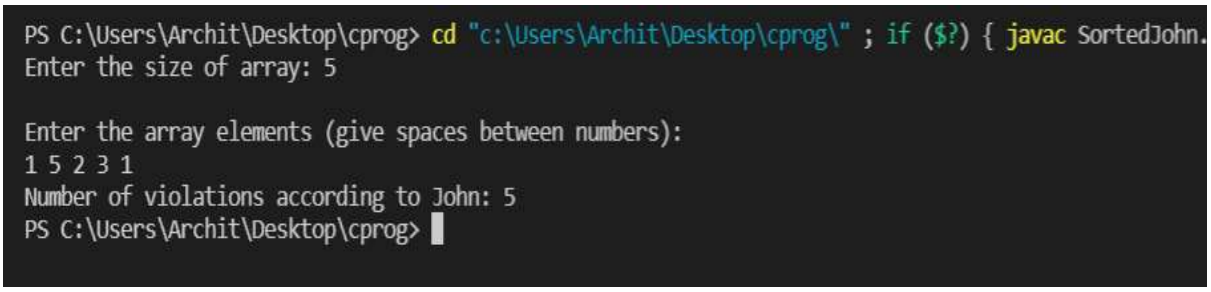
- Base Case to end recursion is when index passed is 0. In such a case, recursion will end by returning 0.
- A variable 'count' is made which is initialized to 0. It stores the number of violations for arr[index].
- When index passed is not zero, run a loop from i = 0 to i = (index – 1). If arr[i] > arr[index], it means that there is a violation in the array, therefore increment the variable count. If arr[i] is not greater than arr[index], do nothing.
- Make a recursive call to the method violations, this time the index will be decremented by 1.

This way, in each recursive call, the method will count the number of violations for arr[index] and ultimately index will reduce to 0, which will end the recursion.

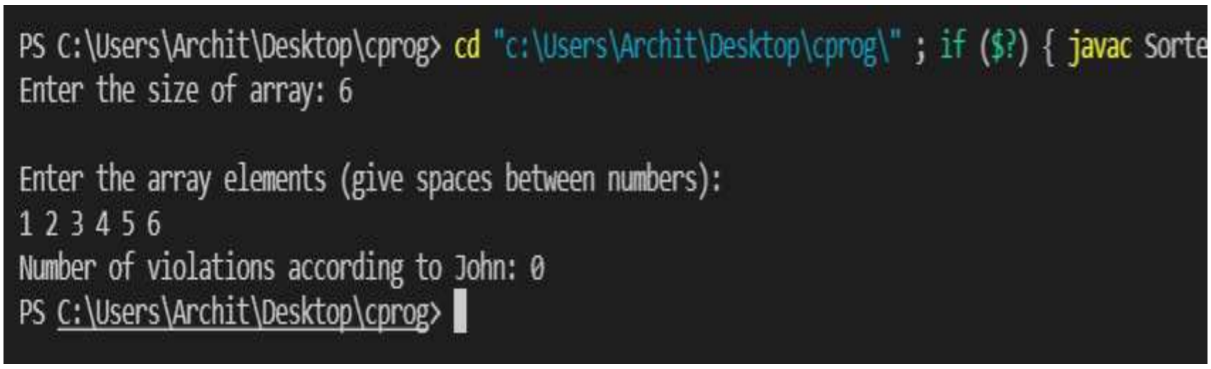
Code

```java
import java.util.*;

public class SortedJohn{

    public static int violations(int[] arr, int index){

        if(index == 0) return 0; //base case to end recursion

        int count = 0;
        for(int i = 0; i < index; i++){  //counting the number of violations
for arr[index]
            if(arr[i] > arr[index]) count++;
        }

        return count + violations(arr, index - 1); //recursice call for index
- 1
    }

    //the user just needs to pass the array
    //the underlying implementaion i.e. the index and count is passed within
it.

    public static int violations(int[] arr){
        return violations(arr, arr.length - 1);
    }

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the size of array: ");
        int n = sc.nextInt();
        System.out.println();

        System.out.println("Enter the array elements (give spaces between
numbers): ");
        int[] array = new int[n];
        for(int i = 0; i < n; i++){
            array[i] = sc.nextInt();
        }

        System.out.println("Number of violations according to John: "+
violations(array));

    }
}
}
```

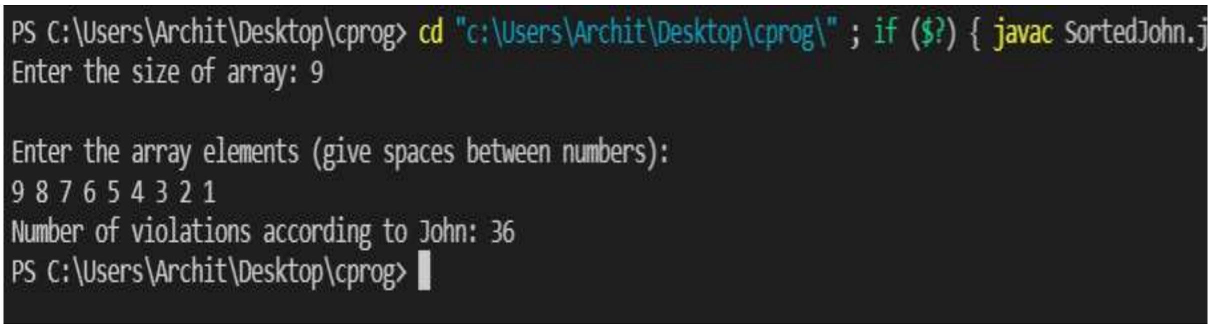
Output



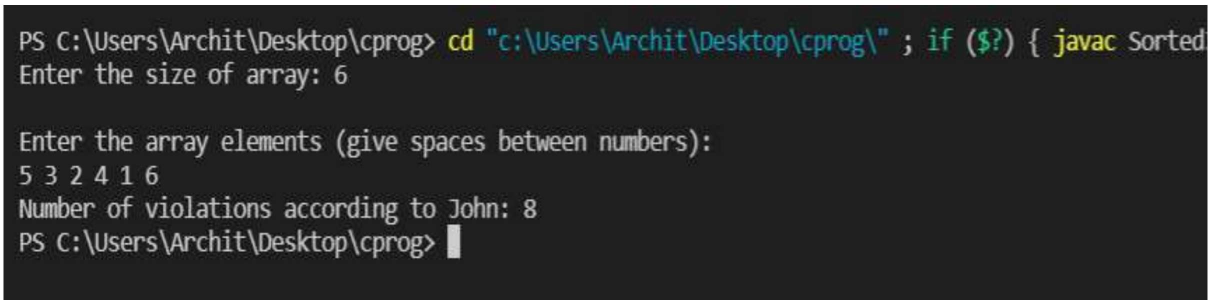
```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac SortedJohn.
Enter the size of array: 5

Enter the array elements (give spaces between numbers):
1 5 2 3 1
Number of violations according to John: 5
PS C:\Users\Archit\Desktop\cprog>
```



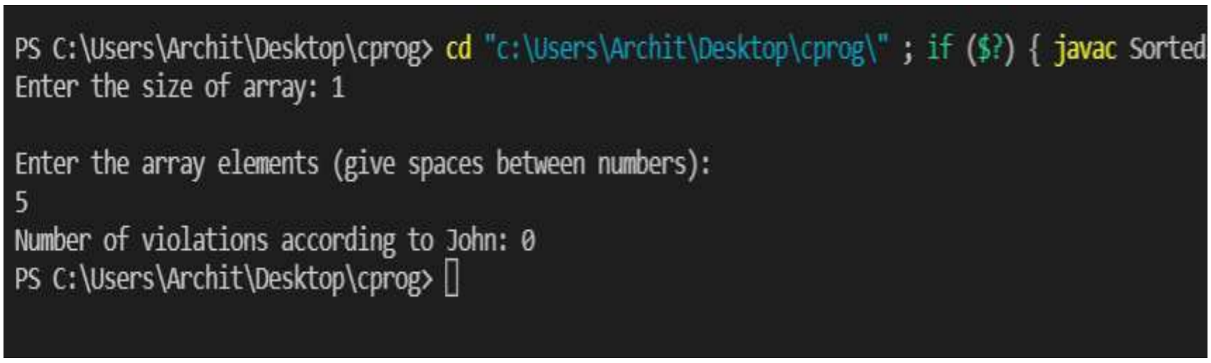
```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Sorte
Enter the size of array: 6

Enter the array elements (give spaces between numbers):
1 2 3 4 5 6
Number of violations according to John: 0
PS C:\Users\Archit\Desktop\cprog>
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac SortedJohn.j
Enter the size of array: 9

Enter the array elements (give spaces between numbers):
9 8 7 6 5 4 3 2 1
Number of violations according to John: 36
PS C:\Users\Archit\Desktop\cprog>
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Sorted
Enter the size of array: 6

Enter the array elements (give spaces between numbers):
5 3 2 4 1 6
Number of violations according to John: 8
PS C:\Users\Archit\Desktop\cprog>
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Sorted
Enter the size of array: 1

Enter the array elements (give spaces between numbers):
5
Number of violations according to John: 0
PS C:\Users\Archit\Desktop\cprog>
```

Date ___/___/___

Time Complexity

```
public static int violations(int[] arr, int index, int count){
    if (index == 0) return count;                        — 1
    for (int i=0; i<index; i++){                         — n
        if (arr[i] > arr[index]) count++;
    }
    return violations(arr, index-1, count);              — T(n-1)
}
```

n → size of arr

the reccurence relation for the method violations is

$$\boxed{T(n) = T(n-1) + n + 1} \quad — \textcircled{1}$$

Solving, it using iterative substitution

from the recurrence relation, we have

$T(n-1) = T(n-2) + (n-1) + 1$

Substituting $T(n-1)$ in ①

$T(n) = T(n-2) + (n-1) + n + 2$

Using ①, Substituting $T(n-2)$ in ①

$T(n) = T(n-3) + (n-2) + (n-1) + n + 3$

⋮

Iteratively substituting $T(n-i)$ using ① in ①

⋮

~~$T(n-k) = T$~~

$T(n) = T(n-k) + (n-k+1) + (n-k+2) + \cdots + (n-1) + n + k$

Assume $n-k=0$ i.e. $n=k$

$\therefore$ $T(n) = T(0) + 1 + 2 + \cdots + (n-1) + n + n$

We know, $T(0) = 1$

$\therefore$ $T(n) = 1 + n + (1 + 2 + \cdots + (n-1) + n)$

$T(n) = n + 1 + \frac{n(n+1)}{2}$

$T(n) = \frac{n^2 + 3n + 2}{2}$

☆ $\boxed{\therefore T(n) = O(n^2)}$ ☆

Problem

You arrive at Disney Land, where you can participate in various activities and games. Each activity or game has different costs. You can choose only one activity. You've been given a set C of n coins in various values (c1, c2, c3, . . . , cn). Try to find the fewest number of coins to pay for your activity.

Example:

Input 1: Coins = {20, 10, 5}, V = 40

Output: 2

Explanation: 2 coins of 20 are required.

Input 2: Coins = {20, 10, 5}, V = 25

Output: 2

Explanation: 1 coin of 20 and 1 of 5 are required.

Input 3: Coins = {20, 10, 5, 4, 3}, V = 27

Output: 3

Explanation: 1 coin of 20, 1 of 4 and 1 of 3 are required.

Algorithm

Coins[] is set of coins and 'value' is amount we have to make.

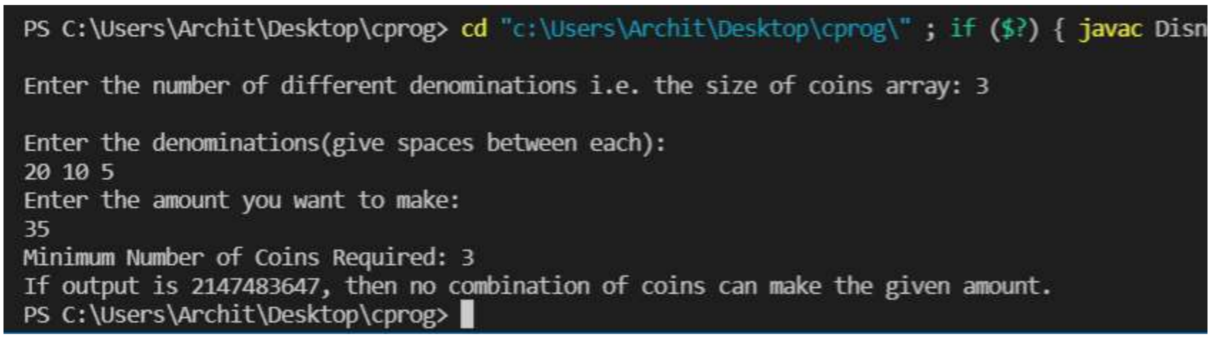
- Base case to end recursion is when value = 0
- Pick any coin, say coin[i], now our problem reduces to finding minimum number of coins to get an amount of 'value – coins[i]'. The solution, therefore, to original problem will be 1 + minimumCoins(value – c[i]).
- Pick coins from one end of array, and recurse down, in each recursive step, select other coins for the previous selected coins.
- If 'value – coins[i] < 0' then we need not to recurse further and we can end this recursive call. Else, we need to recurse until we obtain a solution i.e value == 0.

Code

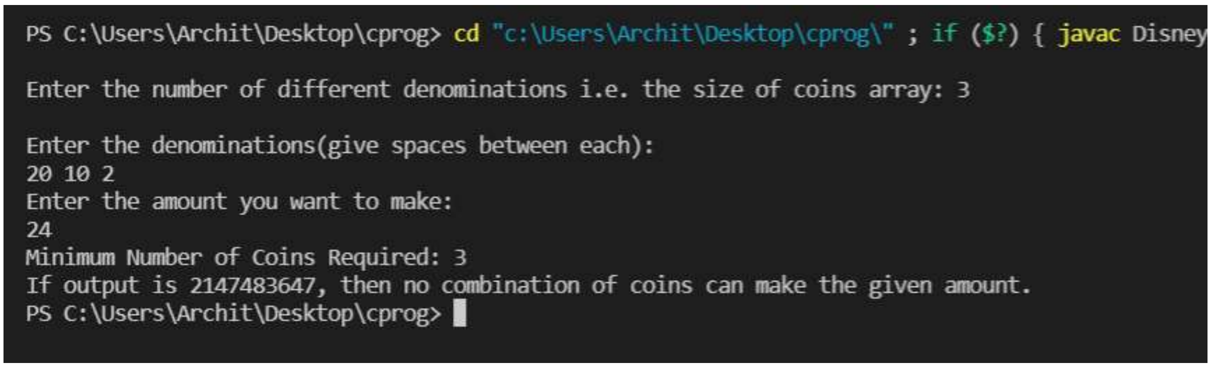
```java
import java.util.*;

class DisneyLand{

    public static int minimumCoins(int[] coins, int value){

        if(value == 0) return 0;

        int min = Integer.MAX_VALUE;

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

            if(coins[i] <= value){
                int currMin = minimumCoins(coins, value - coins[i]);
                if(currMin != Integer.MAX_VALUE && currMin + 1 < min){
                    min = currMin + 1;
                }
            }
        }

        return min;
    }

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);

        System.out.println();
        System.out.print("Enter the number of different denominations i.e. the 
size of coins array: ");
        int n = sc.nextInt();
        System.out.println();

        System.out.println("Enter the denominations(give spaces between 
each):");
        int[] coins = new int[n];

        for(int i = 0; i < n; i++){
            coins[i] = sc.nextInt();
        }
        System.out.println("Enter the amount you want to make: ");
        int value = sc.nextInt();

        System.out.println("Minimum Number of Coins Required: "
+minimumCoins(coins, value));
        System.out.println("If output is 2147483647, then no combination of 
coins can make the given amount.");
```

```
    }
}
```

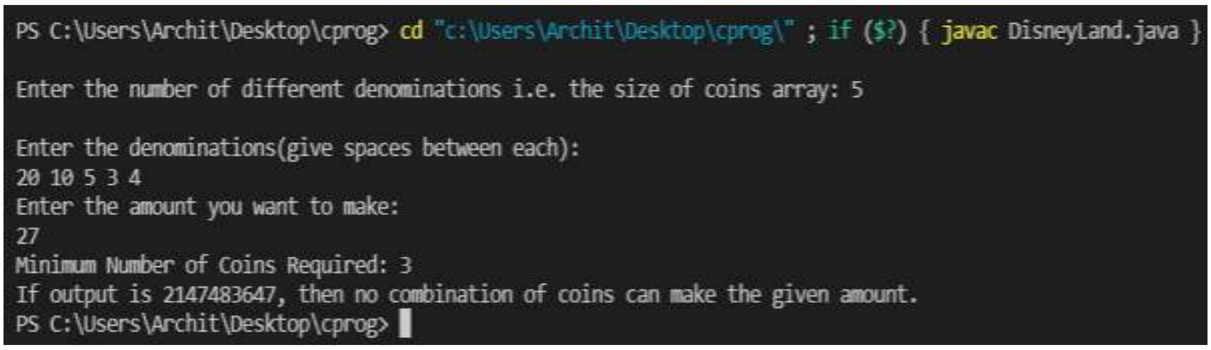
Output



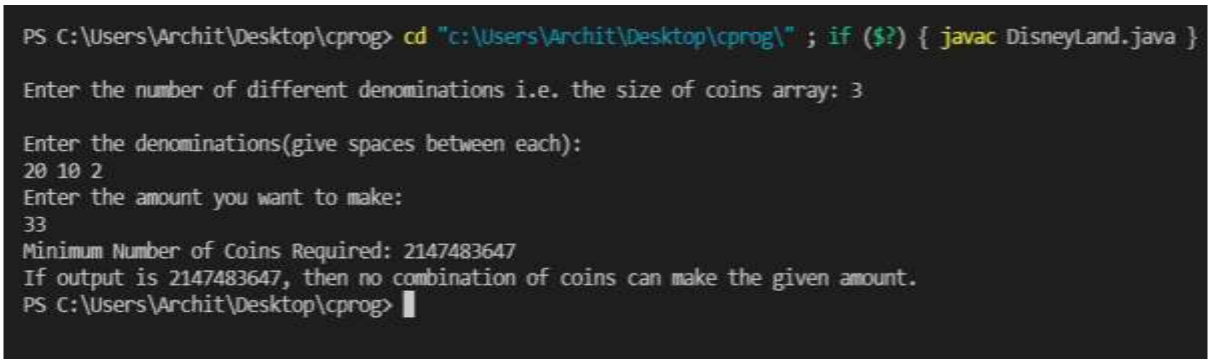
```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Disn

Enter the number of different denominations i.e. the size of coins array: 3

Enter the denominations(give spaces between each):
20 10 5
Enter the amount you want to make:
35
Minimum Number of Coins Required: 3
If output is 2147483647, then no combination of coins can make the given amount.
PS C:\Users\Archit\Desktop\cprog>
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Disney

Enter the number of different denominations i.e. the size of coins array: 3

Enter the denominations(give spaces between each):
20 10 2
Enter the amount you want to make:
24
Minimum Number of Coins Required: 3
If output is 2147483647, then no combination of coins can make the given amount.
PS C:\Users\Archit\Desktop\cprog>
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac DisneyLand.java }

Enter the number of different denominations i.e. the size of coins array: 5

Enter the denominations(give spaces between each):
20 10 5 3 4
Enter the amount you want to make:
27
Minimum Number of Coins Required: 3
If output is 2147483647, then no combination of coins can make the given amount.
PS C:\Users\Archit\Desktop\cprog>
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac DisneyLand.java }

Enter the number of different denominations i.e. the size of coins array: 3

Enter the denominations(give spaces between each):
20 10 2
Enter the amount you want to make:
33
Minimum Number of Coins Required: 2147483647
If output is 2147483647, then no combination of coins can make the given amount.
PS C:\Users\Archit\Desktop\cprog>
```

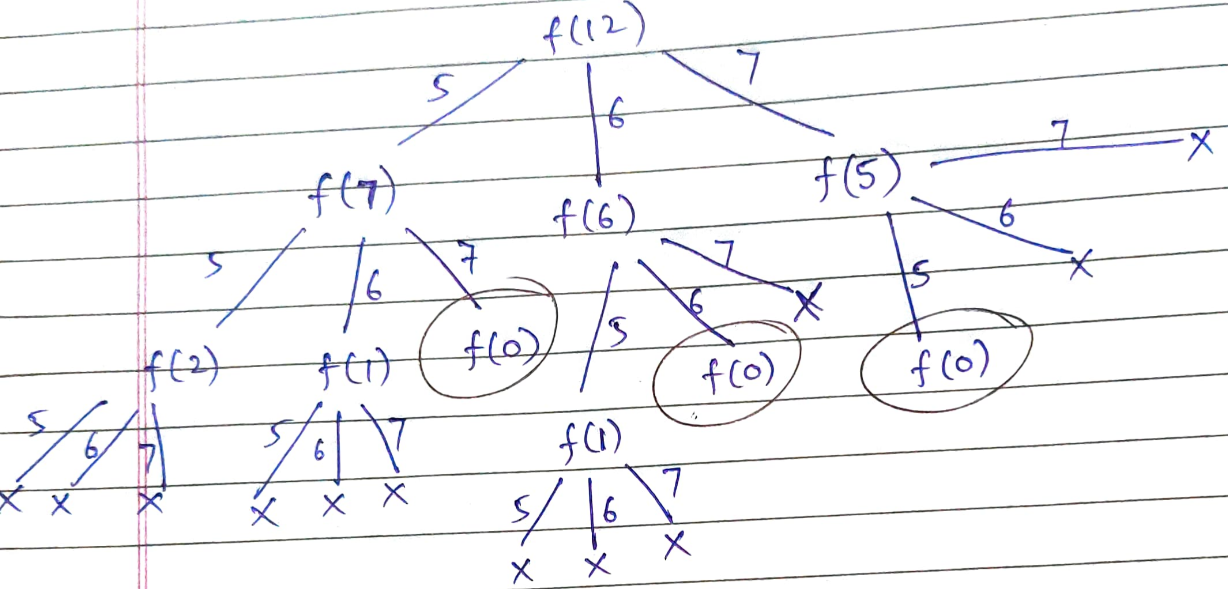
Saathi

Date __ / __ / __

Time- Complexity

Let us take an example and try to find an upper bound using recursion tree.

Lets take coins = [5,6,7] and value = 12,



Let us assume that there are n different types of denominations and the value be k. From the above example, for the upper bound, suppose coins[i] <= k during recursion, then the recursion tree would be a full tree with ~~m~~ each child having m nodes. So, we have n number of recursive calls at 1st level, n^2 at 2nd level, n^3 at third level. and n^k at kth level. The recursive calls will reach ~~[illegible]~~ maximum till kth level only.

Date ___/___/___

$\therefore T(n) = 1 + m + m^2 + \dots + m^K$

$T(n) = \dfrac{(m^K - 1)}{m - 1}$

★ $\boxed{\therefore T(n) = O(m^K)}$ ★