



Experiment 3.1

Student Name: Nishant Kumar Mehta

UID: 21BCS3402

Branch: CSE

Section/Group: IOT - 602

Semester: 5th

Date of Performance: 20/09/23

Subject Name: Advance Programming Lab

Subject Code: 21CSP - 314

1. Aim: Implement the problems based on Dynamic Programming.

2. Objective:

- Your goal is to find the number of ways to construct an array such that consecutive positions contain different values. Specifically, we want to construct an array with n elements such that each element is between 1 and k , inclusive. We also want the first and last elements of the array to be 1 and x . Given n , and x , find the number of ways to construct such an array. Since the answer may be large, only find it modulo $10^9 + 7$.
- Christy is interning at HackerRank. One day she has to distribute some chocolates to her colleagues. She is biased towards her friends and plans to give them more than the others. One of the program managers hears of this and tells her to make sure everyone gets the same number. To make things difficult, she must equalize the number of chocolates in a series of operations. For each operation, she can give 1, 2 or 5 pieces to all but one colleague. Everyone who gets a piece in a round receives the same number of pieces. Given a starting distribution, calculate the minimum number of operations needed so that every colleague has the same number of pieces..

3. Program and output:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    static long countArray(int n, int k, int x) {
```

```
// Return the number of ways to fill in the array.
long max = 10000000007;
long[] f = new long[n + 1]; //[1,...,1]
long[] g = new long[n + 1]; //[1,...,2]
f[3] = k - 1;
g[2] = 1;
g[3] = k - 2;
for (int i = 4; i <= n; i++) {
    f[i] = (k - 1) * g[i - 1] % max;
    g[i] = (f[i - 1] + (k - 2) * g[i - 1]) % max;
}
return x == 1 ? f[n] : g[n];
}

public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int n = in.nextInt();
    int k = in.nextInt();
    int x = in.nextInt();
    long answer = countArray(n, k, x);
    System.out.println(answer);
    in.close();
}
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

Input (stdin)

[Download](#)

1 4 3 2

Your Output (stdout)

1 3

Expected Output

[Download](#)

1 3

2.

```
import java.io.BufferedReader;
import java.util.Scanner;
public class Solution {
    static int offset = 100;
    static Integer f[][] = new Integer[1111][1111];
    static int a[] = new int[1111];
    // From i down to j
    static Integer F(int i, int j) {
        if (i < j) return Integer.MAX_VALUE / 2;
        if (f[i + offset][j + offset] != null) return f[i + offset][j + offset];
        if (i == j) return 0;
        int ans = Integer.MAX_VALUE / 2;
        ans = Math.min(ans, F(i - 1, j) + 1);
        ans = Math.min(ans, F(i - 2, j) + 1);
        ans = Math.min(ans, F(i - 5, j) + 1);
        return f[i + offset][j + offset] = ans;
    }
    public static void main(String[] args) {
        Scanner cin = new Scanner(new BufferedReader(System.in));
        for (int T = cin.nextInt(); T != 0; T--) {
            int n = cin.nextInt();
            int min = Integer.MAX_VALUE;
            for (int i = 0; i < n; i++) {
                a[i] = cin.nextInt();
            }
        }
    }
}
```

```

        min = Math.min(min, a[i]);
    }
    int ans = Integer.MAX_VALUE;
    for (int i=min; i>=min-30; i--) {
        int tmp = 0;
        for (int j=0; j<n; j++) {
            tmp += F(a[j], i);
        }
        ans = Math.min(ans, tmp);
    }
    System.out.println(ans);
}
}

```

Congratulations!
You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0	Input (stdin)	Download
✓ Sample Test case 0	<pre> 1 5 2 110 3 53 361 188 665 786 898 447 562 272 123 229 629 670 848 994 54 822 46 208 17 449 302 466 832 931 778 156 39 31 777 749 436 138 289 453 276 539 901 839 811 24 420 440 46 269 786 101 443 832 661 460 281 964 278 465 247 408 622 638 440 751 739 876 889 380 330 517 919 583 356 83 959 129 875 5 750 662 106 193 494 120 653 128 84 283 593 683 44 567 321 484 318 412 712 559 792 394 77 711 977 785 146 936 914 22 942 664 36 400 857 4 82 5 520 862 10 956 498 956 991 542 523 664 378 194 76 90 753 868 837 830 932 814 616 78 103 882 452 397 899 488 149 108 723 22 323 733 330 821 41 322 715 917 986 93 111 63 535 864 931 372 47 215 539 15 294 642 897 98 391 796 939 540 257 662 562 580 747 893 401 789 215 468 58 553 561 169 616 448 385 900 173 432 115 712 </pre>	
✓ Sample Test case 1		

Learning Outcomes:

- Learnt about Dynamic Programming.
- Learnt about how to optimize a Problem using DP.
- Learnt about Dynamic approach to solve a problem.