The Jumping Frog

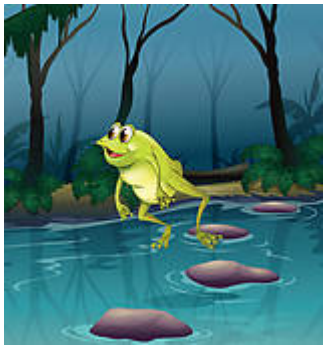
Attempted by: **26** / Accuracy: **71%** / Maximum Score: **50** / 0 Votes

Tag(s): Dynamic Programming, Medium-Hard

PROBLEM EDITORIAL

There was a small pond besides a road in Babylon . There were N consecutive stones in that small pond . Each stone had a no 1 , 2 , or 3 written on it .

At the farther end of the pond lived a frog with his family . In order to search for food the frog family crossed the pond by jumping on stones as shown in the image .



A Professor passing by the road observed the frog's motion and found that each Frog performed a sequence of steps to reach the end. Each step was lead in the positive direction, that is, if current position of the frog was stone k, the next jump will take frog to one of the stones k+1 through N stones ,inclusive . Jumps made by the frog can be short/long . However, longer jumps are costly as jump of length l required work of about pow(l,2) .

Additionally, Frog's jumped on the stones in a cyclic order 1,2,3, 1,2,3 and so on . ... That is, Frog started on the 1st stone with no 1, made a jump to a stone with no 2 on it , from there to a stone with no 3 , and so on, always repeating no's 1, 2, and 3 in a cycle.

```
Notes :

* The 1st stone has a no 1 written on it by default .

* There wouldn't be any such cases in which the frog's cant cross the pond .
```

Professor was impressed by the clever Frogs and thus Transformed the situation into a computing problem . He wanted his student's to print the minimum cost to reach the last stone .

Input

- You would be a given string S with value of each stone .

Output

- Print Minimal cost of reaching the end Mod 1e9+7 . (Following an Optimal Strategy)

Constraints

```
1<=L<=4000
```

SAMPLE INPUT

```
12223
```

SAMPLE OUTPUT

```
8
```

Explanation

```
Sample Case

Length = 5

stone 1 ,2 ,3 ,4 ,5

Optimal way -> stone 1 -> stone 3 -> stone 5

Soln :-)

since jump lengths are  l1 = 2 , length l2 = 2

thus,

pow(l1,2)+ pow(l2,2) ----> l1 * 2 + l2 * 2 = 8
```

Time Limit:	2.0 sec(s) for each input file.
Memory Limit:	256 MB
Source Limit:	1024 KB
Marking Scheme:	Marks are awarded when all the testcases pass.

Allowed Languages: C, C++, C++14, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java, Java 8, JavaScript(Rhino), JavaScript(Node.js), Julia, Kotlin, Lisp, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, R(RScript), Racket, Ruby, Rust, Scala, Swift, Visual Basic

CODE EDITOR

Enter your code or Upload your code as file. Save C (gcc 5.4.0)

```
1  /*
2  // Sample code to perform I/O:
3
4  scanf("%s", name);              // Reading input from STDIN
5  printf("Hi, %s.\n", name);      // Writing output to STDOUT
6
7  // Warning: Printing unwanted or ill-formatted data to output will cause the test c
8  */
9
10 // Write your code here
11
```

1:1

Provide custom input

Press Ctrl-space for autocomplete suggestions.

COMPILE & TEST SUBMIT

Your Rating: Like 0 Share Tweet