ASSUMPTION UNIVERSITY

Vincent Mary School of Science and Technology

Department of Computer Science

CS3201

Algorithm Design

Term Project Report

Submit to

Asst Prof.Dr.Thitipong Tanpraset

By

5918131 FENGBO ZHOU

Semester 2/2017

OUTLINES

Problem Definition

- Input
- Output
- Sample Input & Output

Problem Analysis

Problem Solution

Submission Result

Problem Definition

1495.One-two, One-two 2

Time limit:2.0 second

Memory limit: 64 MB

difficulty:348

Description:

A year ago the famous Vito Maretti woke up in the morning and realized that he was bored of robbing banks of round sums. And for the last year he has been taking from banks sums that have only digits 1 and 2 in their decimal notation. After each robbery, Vito divides the money between N members of his gang, Your task is to determine the minimal stolen sum which is a multiple of N.

Input

The input contains the number N (1 <= N <= (10^6)).

Output

Output the minimal number which is a multiple of N and whose decimal notation contains only digits 1 and 2, if it contains more than 30 digits or if there are no such numbers, then output "Impossible".

Samples

input	output
5	Impossible
8	112

Problem Analysis

This problem is to find a number M that M % N = 0 and M contains only digits 1 and 2, and M's length is not more than 30. First we need to know the property of modulations. Which is

(a*b) mod m = ((a mod m) * (b mod m)) mod m

And because of this property, we could make sure that if a mod b = m, and c mod b = m also, but c > a, and if there is a number after c that could divide b. then there must exist a number a < d < c, that d mod m = 0. By utilizing these two properties. We could make a memorization table that record the remainder from 0 to N. And the key idea is to use BFS to find the solution if there has one, by adding 1 or 2 to the end of the last numbers, then we could form a new number, and check it by using the modulation properties. If we use BFS only there would be 2^30 solutions, but because we are still using the property that if a remainder shows up, then we won't add it to the queue anymore, and that could reduce a lot of time.

Problem solution

```java
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

class State{
    String num;
    int modd;
    int size;

    public State(String n, int m, int s){
        this.num = n;
        this.modd = m;
        this.size = s;
    }
}

```

```java
public class P1495{
    ▶ Run | Debug
    public static void main(String args[]){
        Queue<State> queue = new LinkedList<>();
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        queue.add(new State("",0,0));
        String ret = "-1";
        int[] mem = new int[n+1];
        mem[0] = 1;
        int mod10 = 10 % n;
        int mod1 = 1 % n;
        int mod2 = 2 % n;
        boolean hasAns = false;

        while (!queue.isEmpty()){
```

```java
        while (!queue.isEmpty()){
            State current = queue.poll();
            if (current.size > 30){
                break;
            }
            int temp = (mod10 * current.modd) % n;
            int temp1 = (temp + mod1) % n;
            int temp2 = (temp + mod2) % n;
            if (temp1 == 0){
                ret = current.num + "1";
                hasAns = true;
                break;
            }
            if (temp2 == 0){
                ret = current.num + "2";
                hasAns = true;
                break;
            }
            if (mem[temp1] == 0){
                queue.add(new State(current.num + "1", temp1, current.size + 1));
                mem[temp1] = 1;
            }
            if (mem[temp2] == 0){
                queue.add(new State(current.num + "2", temp2, current.size + 1));
                mem[temp2] = 1;
            }
        }
```

```
58
59          if (hasAns){
60              System.out.println(ret);
61          }
62          else{
63              System.out.println("Impossible");
64          }
65
66      }
```

The State class contains three fields, the number itself, which is represented as string, and the modd (the number mod n result), the size, which is number of digit of the number.

As normally, the BFS would detect whether or not a number is the solutions, if it's not then we would either add 1 or 2 at the end of that number, and check its modulation, if it's not in the memorization table, then it would be added to the queue, otherwise we would skip it. And finally we could find the solution, or there is no such a solution.

Examples:

```
C:\Users\zfb19\Desktop\c2018>java P1495
5
Impossible

C:\Users\zfb19\Desktop\c2018>java P1495
8
112
```

Submission Result

8152778	10:01:30 28 Nov 2018	fengbo	1495. One-two, One-two 2	Java 1.8	Accepted		0.951	62 428 KB

Running time analysis

Normally with the brute force technique, the running time is 2^30;

But with the dynamic programming, calculating the result is much faster, and by using BFS,I use a look up table to track the result of the modulation. If it appears, then it won’t be added to the queue anymore, because it won’t be the minimal solution any way. At final, the running time of this program should be around O(30 * n)