HDOJ 2604 - Queue

——by A Code Rabbit

Description

一个长度为 L 的队列,里面排列着男人和女人,分别用 f 和 m 表示。 计算出长度为 L 的队列中,不含子串 fff 和 fmf 的队列个数,并对一个数 M 取模。

Types

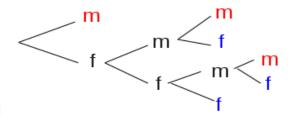
Maths:: Matrix

Analysis

这种题目很像递推题,没错它就是递推题。

考虑当前长度为L的队列,和长度小于L的队列的关系,看能不能找到递推关系。

我们可以试图在长度小于 L 且符合条件的队列后面加上几个元素,使之长度为 L ,又不含 fff 和 fmf。看图:



这是在长度小于 L 的队列

后面加元素的情况,如果加一

个m,那么必然得到的队列也符合条件,如果得到f,还要继续考虑,然后以此类推。

最后我们发现,在长度为 L-1、 L-3、 L-4 的队列后面加上 m、 ffm、 ffmm,都可以得到符合条件的队列。

所以,长度为L的符合条件的队列的数量,就等于有几条长度为L-1、L-3、L-4的队列之和。可以设长度为L的符合条件的队列数量为F(L),则得到递推公式:

$$F(L) = F(L-1) + F(L-3) + F(L-4)$$

既然有了递推公式,我们就可以构造矩阵,用矩阵乘法和快速幂来求解了。

需要构造的矩阵如下:

不过,在知道这种方法之前,我不是这样做的,囧。

我先写了一个代码暴力地求解答案,然后从答案中间发现了另外一个规律。

http://blog.csdn.net/Ra_WinDing

然后进行矩阵乘法和快速幂求解。

Solution

```
1. 暴力求解
// HDDJ 2604
// Queuing
// by A Code Rabbit
#include <iostream>
#include <string>
using namespace std;
int ans;
int n;
void Search(string str);
int main() {
    while (cin >> n) {
        ans = 0;
        Search("");
        cout << ans << endl;
```

```
}
}
void Search(string str) {
    if (str.length() >= n) {
        if (str.find("fmf") == string::npos &&
            str.find("fff") == string::npos)
        {
           ++ans;
        }
        return;
   }
   Search(str + "f");
   Search(str + "m");
}
// HD0J 2604
// Queuing
// by A Code Rabbit
2. 递推+矩阵乘法+快速幂
#include <cstdio>
struct Matrix {
   int element[5][5];
};
int l, m;
const Matrix mat_init = {
   5, 3, 2, 1, 0,
   3, 2, 1, 1, 0,
   0, 0, 0, 0, 0,
   0, 0, 0, 0, 0,
   1, 1, 1, 0, 1,
};
const Matrix mat_unit = {
   1, 0, 0, 0, 0,
   0, 1, 0, 0, 0,
   0, 0, 1, 0, 0,
```

http://blog.csdn.net/Ra_WinDing

```
0, 0, 0, 1, 0,
   0, 0, 0, 0, 1,
};
Matrix mat_one, mat_ans;
const int ans[] = {
   9, 6, 4, 2, 1,
};
void INIT();
void Multiply(Matrix& mat_a, Matrix mat_b);
int main() {
    while (scanf("%d%d", &l, &m) != EOF) {
        // INIT.
        INIT();
        // Quick Sort.
        int num = (l - 1) / 4;
        while (num) {
            if (num & 1) {
                Multiply(mat_ans, mat_one);
            }
            Multiply(mat_one, mat_one);
            num >>= 1;
        }
        // Compete and outputs.
        int ans_sum = 0;
        for (int i = 0; i < 5; ++i) {
            ans_sum += ans[i] * mat_ans.element[i][4 - ((l - 1) % 4 + 1)];
        printf("%d\n", ans_sum % m);
    }
    return 0;
void INIT() {
   mat_one = mat_init;
   mat_ans = mat_unit;
```

http://blog.csdn.net/Ra_WinDing

```
void Multiply(Matrix& mat_a, Matrix mat_b) {
    Matrix mat_c;
    for (int i = 0; i < 5; ++i) {
        for (int j = 0; j < 5; ++j) {
            mat_c.element[i][j] = 0;
            for (int k = 0; k < 5; ++k) {
                 mat_c.element[i][j] += mat_a.element[i][k] * mat_b.element[k][j];
            }
            mat_c.element[i][j] %= m;
        }
        mat_a = mat_c;
}
</pre>
```