数据结构专场题解

“Accordian” Patience

题目意思是如果左边的第三张满足移动条件就移到第三张，否则判断左边的第一张，只要有卡片的移动，那么我们就要去判断它和它右边的卡片是否可以移动。

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <vector>
#include <stack>
using namespace std;
//卡片的结构体
struct Card {
    char val;//卡片的值
    char suit;//卡片的颜色
};
Card c[52];//存储52个卡片
vector<stack<Card> >v;//开个vector里面放stack， stack里面放Card结构体
string str;

//判断是否满足移动的情况
int judge(Card x, Card y) {
    if (x.val == y.val || x.suit == y.suit)
        return 1;
    return 0;
}

//处理函数
void solve() {
    v.clear();//清空容器
    for (int i = 0; i < 52; i++) {
        stack<Card>s;
        s.push(c[i]);
        v.push_back(s);//插入最后一个位置
        Card temp;
        //printf("%d:  size: %d\n", i, v.size());
        for (int j = v.size() - 1 ; j < v.size() ; j++) {
            temp = v[j].top();//当前栈顶元素
            if (j >= 3) {//如果是当前的位置大于3，判断左边的第三个位置
                if (judge(temp, v[j-3].top())) {//判断左边的第三个是否和它匹配
                    v[j-3].push(temp);//插入到该位置
                    v[j].pop();//删除当前的j的栈顶元素
```

```
                        if (v[j].empty()) //如果此时j位置为空则删除它
                            v.erase(v.begin() + j);
                        j -= 4;//到左边的第4个位置,因为考虑左边第三个本身也是要判断的
                    }
                    else {//如果和左边第三个位置不匹配，则判断是否和左边第一个匹配
                        if (judge(temp, v[j-1].top())) {
                            v[j-1].push(temp);//将temp插入左边的第一个位置
                            v[j].pop();//删除当前的栈顶的元素
                            if (v[j].empty()) //如果为空，则就删除
                                v.erase(v.begin() + j);
                            j -= 2;//位置变为左边的第2个，因为考虑到左边第一个也要判断
                        }
                    }
                }
                else if (j > 0 && j < 3) {//如果是0-3只要判断左边第一个即可
                    if (judge(temp, v[j-1].top())) {//如果和左边的第一个匹配
                        v[j-1].push(temp);//插入到左边的第一个位置
                        v[j].pop();//删除栈顶元素
                        if (v[j].empty())
                            v.erase(v.begin() + j);
                        j -= 2;//位置变为左边的第2个，因为考虑到左边第一个也要判断
                    }
                }

            }
        }
}

//输出函数
void output() {
    int ans = v.size();//最后有几堆
    if (ans == 1)//注意1的情况
        printf("1 pile remaining:");
    else
        printf("%d piles remaining:", ans);
    for (int i = 0; i < ans; i++)
        printf(" %d", v[i].size());
    printf("\n");
}

//输入函数
int main() {
    int i;
    while (cin >> str && str != "#") {
        i = 0;
        c[i].val = str[0];
        c[i].suit = str[1];
        for (i = 1; i < 52; i++) {
            cin >> str;
            c[i].val = str[0];
            c[i].suit = str[1];
        }
        solve();
```

```
        output();
    }
}
```

约瑟夫环

建立一个循环链表模拟实现。

```
#include <stdio.h>
#include <malloc.h>

typedef struct NODE {
    int data;
    struct NODE *next;
} Node;

/*
 * initialize the circle list
 */
Node* initCircle(int n) {
    int i;
    Node *head = (Node *) malloc(n * sizeof(Node));
    for(i = 0; i < n - 1; i++) {
        head[i].data = i + 1;
        head[i].next = &head[i + 1];
    }
    head[n - 1].data = n;
    head[n - 1].next = &head[0];
    return head;
}

/*
 * travel m step and print&delete current node
 */
void Josephus(Node *head, int m) {
    Node **plink = &(head->next);
    while(*plink != head) {
        plink = &((*plink)->next);
    }
    while((*plink)->next != (*plink)) {
        for(int i = 1; i < m; i++) {
            plink = &(*plink)->next;
        }
        printf("%d ", (*plink)->data);
        *plink = (*plink)->next;
    }
    printf("%d\n", (*plink)->data);
}
```

```c
void destroy(Node *head) {
    free(head);
}

int main() {
    int n, m;
    while(scanf("%d %d", &n, &m) != EOF) {
        Node *L = initCircle(n);
        Josephus(L, m);
        destroy(L);
    }
    return 0;
}
```

签到题

这个程序是Haskell的快速排序程序，Haskell与常见的命令式语言非常不同，没有循环语句，没有全局变量，但是其实现快速排序的程序和快速排序的思想描述是非常接近的。在C语言环境下我们是不需要自己实现快速排序的，正因为如此，很多人便不会实现甚至不认识快速排序，造成了很多面试或考试时的悲剧。

逆序输出

非常简单的问题题，只要遇到一个#号就将之前的内容逆序输出即可。需要注意的是，最后一个#号之后的内容是不需要输出的。输入文件是一首歌的歌词。

```
##nwod klaw nam a tsum sdaor ynam woH##
#woH
? nam a mih llac uoy erofeB#am woh ,seY
? eerf eb ot dewolla er'yeht erofeB
tsixe elpoep emos nac sraey ynam woh ,seY
? aes eht ot dehsaw s'ti erofeB
tsixe niatnuom a nac sraey ynam woh ,seY

.dniw eht ni 'niwolb si rewsna ehT
dniw eht ni 'niwolb si dneirf ym rewsna ehT
? dennab reverof er'yeht erofeB
ylf sllab nonnac eht tsum semit ynam woh ,seY
? dnas eht ni speels ehs erofeB
lias evod etihw a tsum saes ynam #erofeB
pu kool nam a tsum semit ynam woh ,seY

.dniw eht ni 'niwolb si rewsna ehT
dniw eht ni 'niwolb si dneirf ym rewsna ehT
```

```
? ees t'nseod tsuj eh gnidneterP
daeh sih nrut nam a nac semit yn##############################.dniw eht ni 'ni
wolb si rewsna ehT
dniw eht ni 'niwolb si dneirf ym rewsna ehT
? deid evah elpoep ynam oot tahT
swonk eh llit ekat ti lliw shtaed ynam woh ,seY
? yrc elpoep raeh nac eh erofeB
evah nam eno tsum srae ynam woh ,seY
? yks eht ees nac eh # This should not be output
```

SBT

这两个题关系到一颗非常著名的树，可以查看文档: http://quant67.com/DL/stern-brocot.pdf
代码实现非常简单，不再给出。

---

gcd

对于数字全部由1组成的两个自然数，当且仅当它们的位数互素时，这两个自然数互素。

证明 我们用J_m表示数字全部由1组成的m位数。

$$J_m = 10^{m-1} + 10^{m-2} + \cdots + 10 + 1 = \frac{1}{9}(10^m - 1)$$

可以证明如下两个引理
引理1 对$m, d \in N$, 若$d|m$，则$J_d|J_m$.
事实上，由$d|m$可设

$$m = kd, \quad k \in N$$
$$J_m = J_{kd} = \frac{1}{9}(10^{kd} - 1) =$$
$$\frac{1}{9}(10^d - 1)(10^{kd-d} + 10^{kd-2d} + \cdots + 10^d + 1) =$$
$$J_d(10^{kd-d} + 10^{kd-2d} + \cdots + 10^d + 1)$$

由于$10^{kd-d} + 10^{kd-2d} + \cdots + 10^d + 1$是整数，所以

$$J_d|J_m$$

引理2 如果$m, n \in N$，且$m > n$，则

$$J_{m-n} | J_m - J - n$$

事实上

$$J_m - J_n = \frac{1}{9}(10^m - 1) - \frac{1}{9}(10^n - 1) =$$
$$\frac{1}{9}(10^{m-n} - 1) \cdot 10^n = J_{m-n} \cdot 10^n$$

因此

$$J_{m-n} | J_m - J - n$$

现在设J_m, J_n是两个已知数，且$m > n$。

- 我们证明：若J_m和J_n互素，则m和n也互素。

事实上，若m和n不互素，则$gcd(m, n) = d > 1$。于是由引理1必有

$$J_d | J_m, J_d | J_n$$

因而J_m和J_n有大于1的公约数，矛盾。

- 我们再证明：若m和n互素，则J_m和J_n也互素。

对m和n使用辗转相除法

$$\begin{gathered}
m = nq + r, 0 < r < n \\
n = rq_1 + r_1, 0 < r_1 < r \\
r = r_1 q_2 + r_2, 0 < r_2 < r_1 \\
\cdots \cdots \\
r_{k-2} = r_{k-1} + r_k, 0 < r_k < r_{k-1} \\
r_{k-2} = r_k q_{k-1}
\end{gathered}$$

由以上一系列等式可知，r_k是r_{k-1}，r_{k-2}，$\cdots$，r, n, m的公约数。因为m和n互素，所以

$$r_k = 1$$

由引理2有 $\quad J_r | J_m - J_{nq}$
由引理1由 $\quad J_n | J_{nq}$

因此，若D是J_m和J_n的公约数，则D也是$J_m - J_{nq}$的约数。由于

$$J_m - J_{nq} = J_r \cdot 10^{nq}$$

并且D显然没有约束2和5，因此D与10^{nq}互素，因此D能整出J_r。
类似的也可以证明D整除$J_{r1}, J_{r2}, \cdots$。
由于$J_{r_k} = J_1 = 1$
因此$D = 1$。于是J_m和J_n互素。

```
#include <stdio.h>
#include <string>
#include <iostream>
using namespace std;

int gcd(int a, int b) {
    b == 0 ? a : gcd(b, a % b);
}

int main() {
    string str1, str2;
    while(cin >> str1 >> str2) {
        puts(gcd(str1.size(), str2.size()) == 1 ? "Yes" : "No");
    }
    return 0;
}
```

括号匹配

置一个栈S，若输入字符与栈顶字符配对,则弹出，否则输入字符压栈。
最后判断是否栈空即可。

```
#include <stdio.h>
#include <string>
#include <stack>
#include <iostream>
using namespace std;

int main() {
    string str;
    while(cin >> str) {
        stack<char> S;
        for(int i = 0; i < str.size(); i++) {
            if(!S.empty() &&
               ((S.top() == '(' && str[i] == ')')
                || (S.top() == '[' && str[i] == ']')
                || (S.top() == '{' && str[i] == '}')) ) {
```

```
                    S.pop();
                } else {
                    S.push(str[i]);
                }
            }
            puts(S.empty() ? "Yes" : "No");
        }
        return 0;
    }
```

陶瓷妹妹的问题

注意只能用有限多种方法把给定的$n \in N^*$表示为自然数的和，因此存在n的分解师$n = m_1 + m_2 + m_3 + \cdots + m_k$,其中$m_1 \leq m_2 \leq \cdots \leq m_k$，使得乘积$m_1 m_2 \cdots m_k$取到最大值$f(n)$。但因为$4 = 2 \times 2 = 2 + 2$，所以可以约定，在分解式中每个$m_i$都不等于4。如果$m_k > 4$，则因为$(m_k - 2) \times 2 > m_k$，所以和为n的数$m_1, m_2, \cdots, m_k - 2, 2$的乘积将更大。因此$m_i < 3, i = 1, 2, \cdots, k$。其次当n=1时，分解式是唯一的，$f(1) = 1$。当$n > 1$时，如果$m_1 = 1$则因为$m_1 + m_2 > m_1 m_2$，所以和为n的整数$m_1 + m_2, m_3, \cdots, m_k - 2, 2$的乘积将更大。因此每个$m_i$不等于1。最后，在$m_i$中不能有三个或以上为2，否则设$m_1 = m_2 = m_3 = 2$，则和为n的数$3, 3, m_4, \cdots, m_k$的乘积将更大。于是，如果$n > 1$，则$m_1, m_2, m_3, \cdots, m_k$中至多有两个2，其余都是3。
于是

$$f(1) = 1, f(3k) = 3^k, f(3k-1) = 2 \times 3^{k-1}, f(3k+1) = 4 \times 3^{k-1}, \quad k \in N^*.$$

```
#include <stdio.h>
#include <iostream>
using namespace std;
typedef unsigned long long ll;

//x^n % mod
ll mod_pow(ll x, ll n, ll mod) {
    ll res = 1;
    while(n > 0) {
        if(n & 1) res = res * x % mod;
        x = x * x % mod;
        n >>= 1;
    }
    return res;
}

const int mod = 1000000007;

int main() {
    int n;
    while(scanf("%d", &n) != EOF) {
        if(n == 1) {
```

```cpp
                printf("1\n");
                continue;
            }
            int l = (n + 1) / 3;
            if(3*l == n) {
                cout << mod_pow(3, l, mod) << endl;
            } else if(3 * l + 1 == n) {
                cout << mod_pow(3, l - 1, mod) * 4 % mod << endl;
            } else {
                cout << mod_pow(3, l - 1, mod) * 2 % mod<< endl;
            }
        }
        return 0;
    }
```