算法

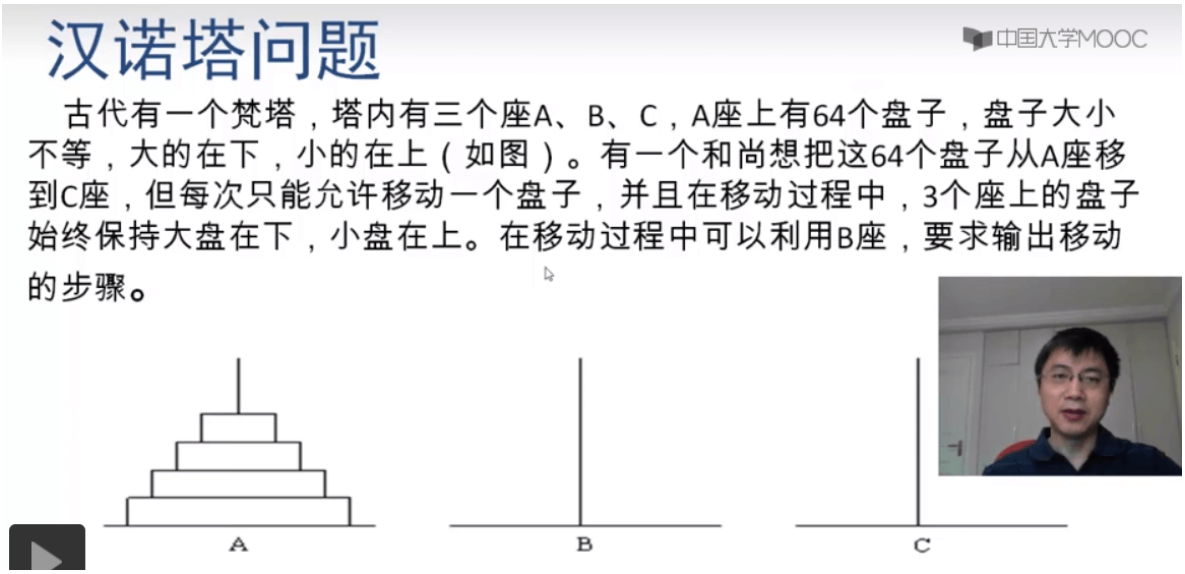
第二章 递归

递归说明

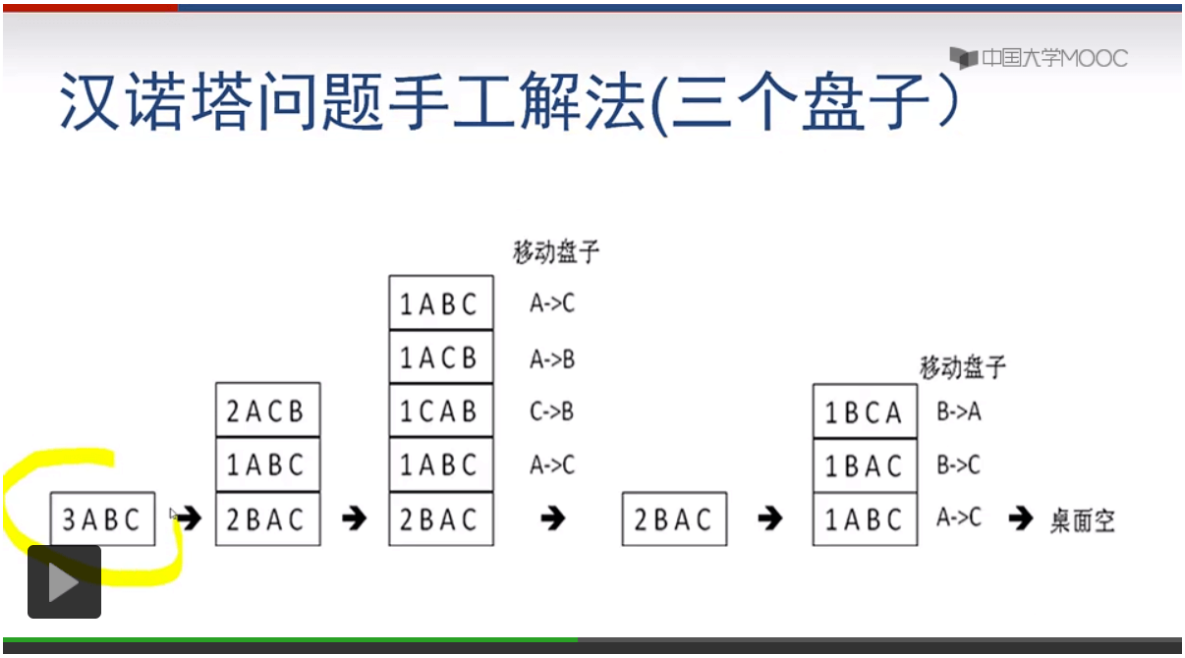
递归有许多重复计算，适合重复量少的计算，如果重复的数量很大，需要用动态规划。递归分类是很重要的思想。递归需要有终止条件，不能让程序一直递归下去。

汉诺塔问题

题目说明



解法说明



c++ code

```
#include<iostream>
using namespace std;
void Hanoi(int n, char src, char mid, char dest);

int  main(){
    int n;
    cin >> n;
    Hanoi(n, 'A', 'B', 'C');
    system("pause");
    return 0;
}
void Hanoi(int n, char src, char mid, char dest){
    if (n == 1){
        cout << src << "->" << dest<<endl;
        return;
    }
    Hanoi(n - 1, src, dest, mid);//先将n-1个 src移动到mid
    cout << src << "->" << dest << endl;
    Hanoi(n - 1, mid, src, dest);//再将n-1个从mid 移动带dest
    return;
}
```

N皇后问题

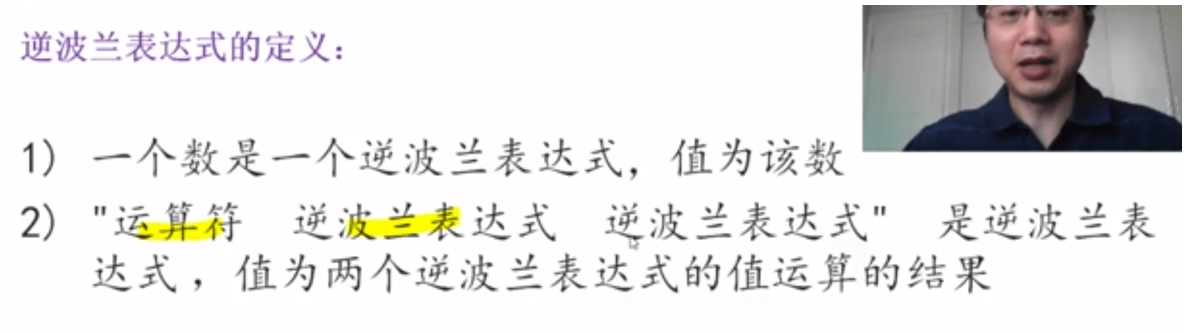
说明：N皇后 N行N列 N个皇后不能在同一行 同一列 同意对角线

```
#include<iostream>
#include<vector>
#include<math.h>
using namespace std;
int N;


void N_help(vector<vector<int>> &res, vector<int>&t,int k){
    int i;
    if (k == N){
        res.push_back(t);
        return;
    }
    for (i = 0; i < N; ++i){//逐个尝试
        int j;
        for (j = 0; j < k; ++j){//和已经摆好的K-1个皇后比较
            if (t[j] == i || abs(t[j] - i) == abs(k - j))break;

        }
        if (j == k){
            t[k] = i;
            N_help(res,t, k + 1);
        }
    }

}
int main(){
    vector<vector<int>> res;

    cin >> N;
    vector<int>t(N, 0);
```

```
    N_help(res, t,0);
    for (int i = 0; i < res.size(); ++i){
        for (int j = 0; j < N; ++j){

            cout << res[i][j]+1 << " ";
        }
        cout << endl;
    }
    system("pause");
    return 0;

}
```

逆波兰表达式

定义



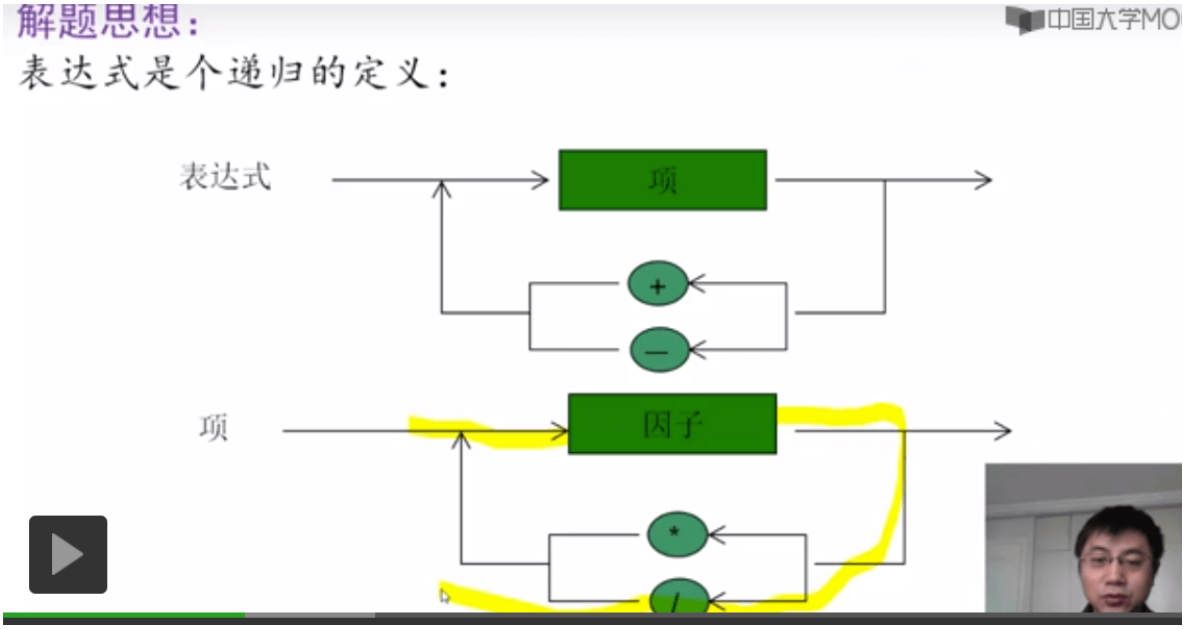
code C++

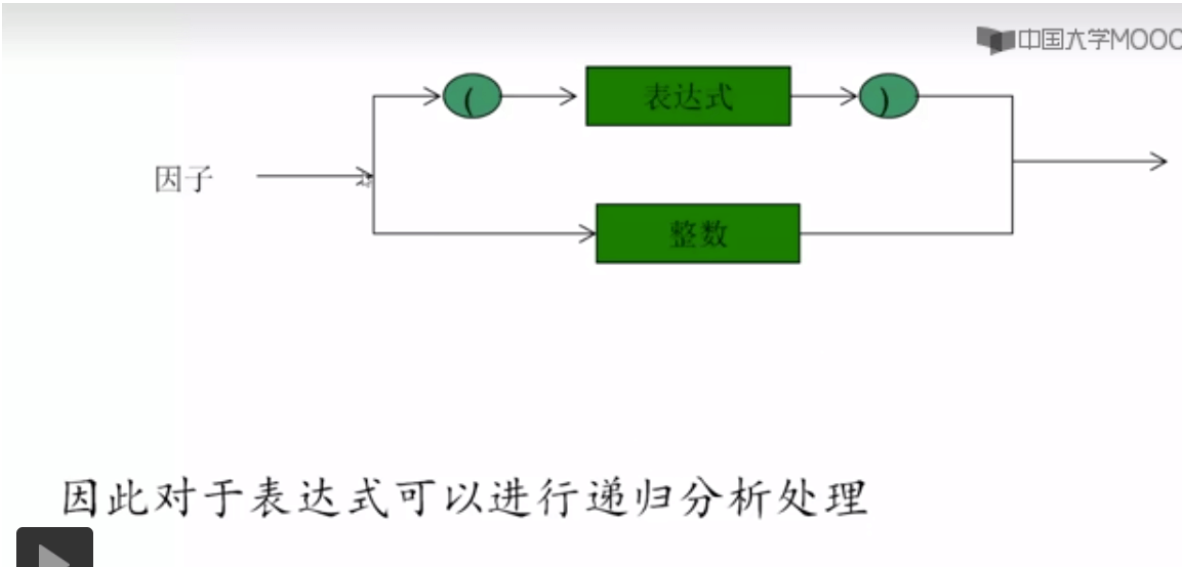
```
#include<iostream>
#include<cstdio>
#include<cstdlib>
using namespace std;
double exp(){
    char s[20];
    cin >> s;
    switch (s[0]){
    case '+':return exp() + exp();
    case '-':return exp() - exp();
    case '*':return exp() * exp();
    case '/':return exp() / exp();
    default:return atof(s);
    break;
    }
}
int main(){
    printf("%lf", exp());
    system("pause");
    return 0;
}
```

表达式求值

加减乘除括号

逻辑图





因此对于表达式可以进行递归分析处理

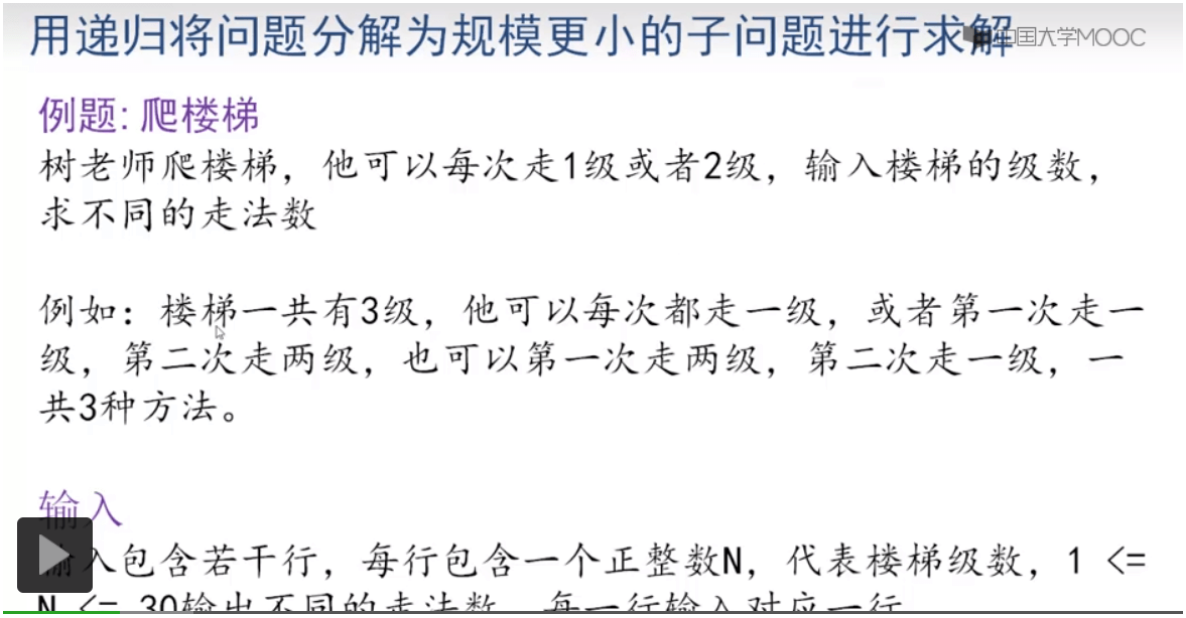
c++code

```
#include<iostream>
#include<cstring>
#include<cstdlib>
using namespace std;
int factor_value();//因子
int term_value();//项
int expression_value();//表达式

int main(){
    cout << expression_value() << endl;
    system("pause");
    return 0;
}
int factor_value(){
    char op = cin.peek();
    int result = 0;
    if (op == '('){
        cin.get();
        result = expression_value();
```

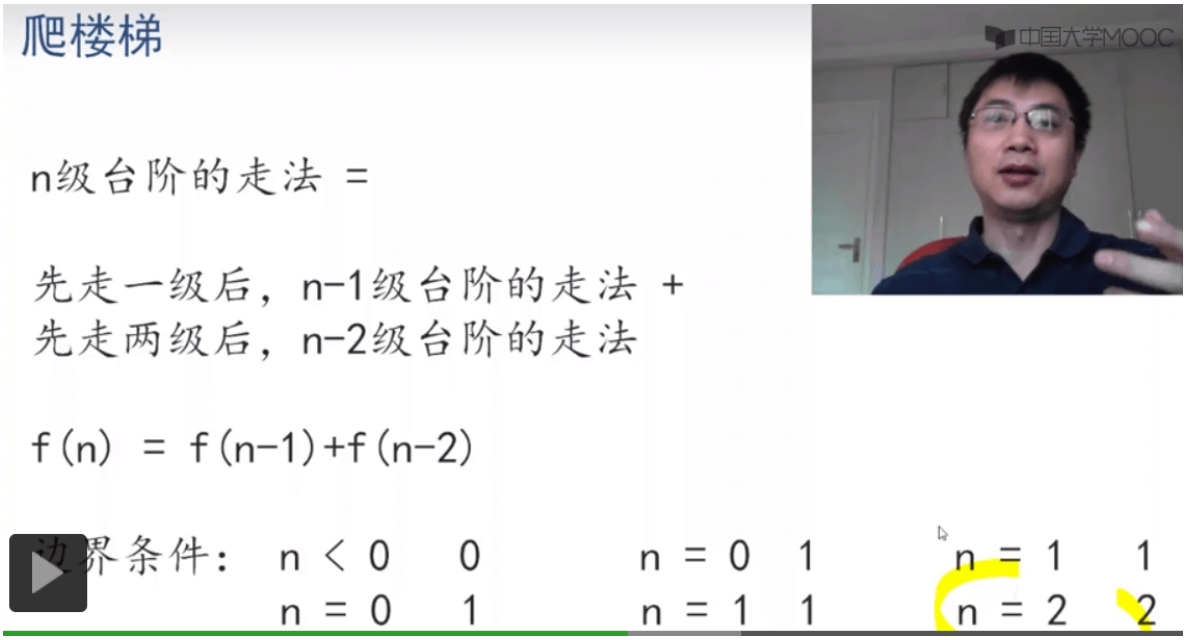
```
        cin.get();
    }
    else while (isdigit(op)){
        result = 10 * result + op - '0';
        cin.get();
        op = cin.peek();

    }
    return result;


}
int term_value(){
    int result = factor_value();
    while (1){
        char op = cin.peek();
        if (op == '*' || op == '/'){
            cin.get();
            int value = factor_value();
            if (op == '*')result *= value;
            else result /= value;
        }
        break;
    }
    return result;

}
int expression_value(){
    int result = term_value();
    while (1){
        char op = cin.peek();
        if (op == '+' || op == '-'){
            cin.get();
            int value = term_value();
            if (op == '+')result += value;
            else result += value;
        }
        break;
    }
    return result;
}
```

爬楼梯问题

题目



题解



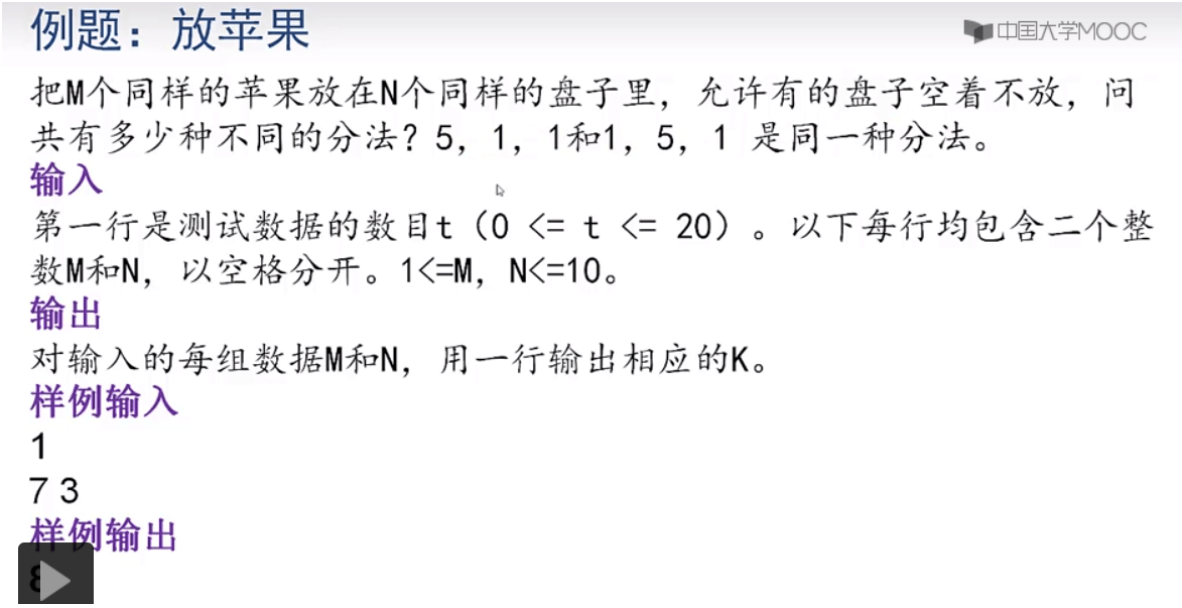
边界条件的意思就是：递归的终止条件 递归一定要有终止条件 不能一直递归下去

c++ code

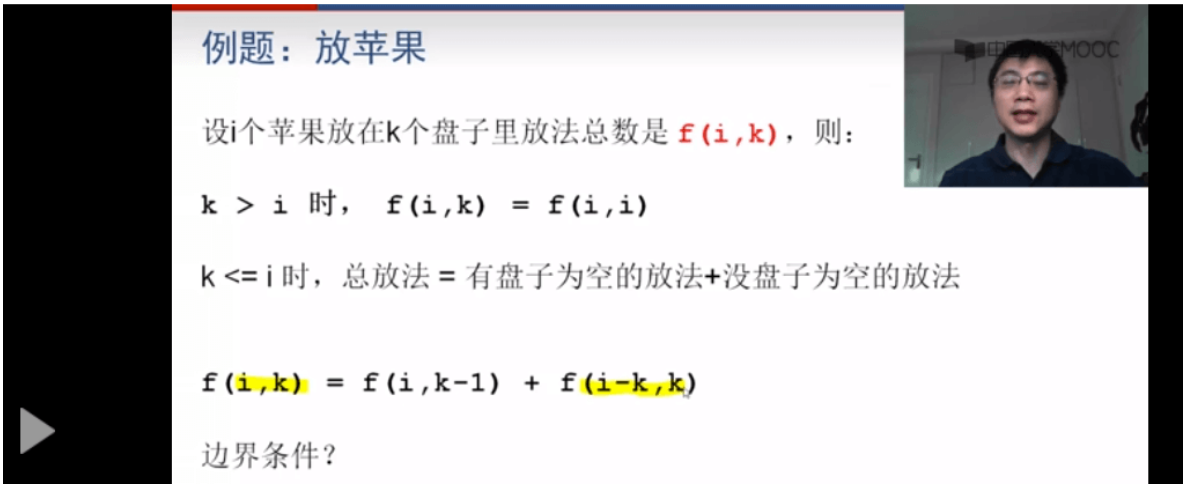
```
#include<iostream>
using namespace std;
int louti(int n);
int main(){
    int N;
    while (cin >> N){
        cout << louti(N) << endl;
    }
    system("pause");
    return 0;
}
int louti(int n){
    if (n == 1)return 1;
    if (n == 2)return 2;
    return louti(n - 1) + louti(n - 2);
}
```

放苹果

题目



解释



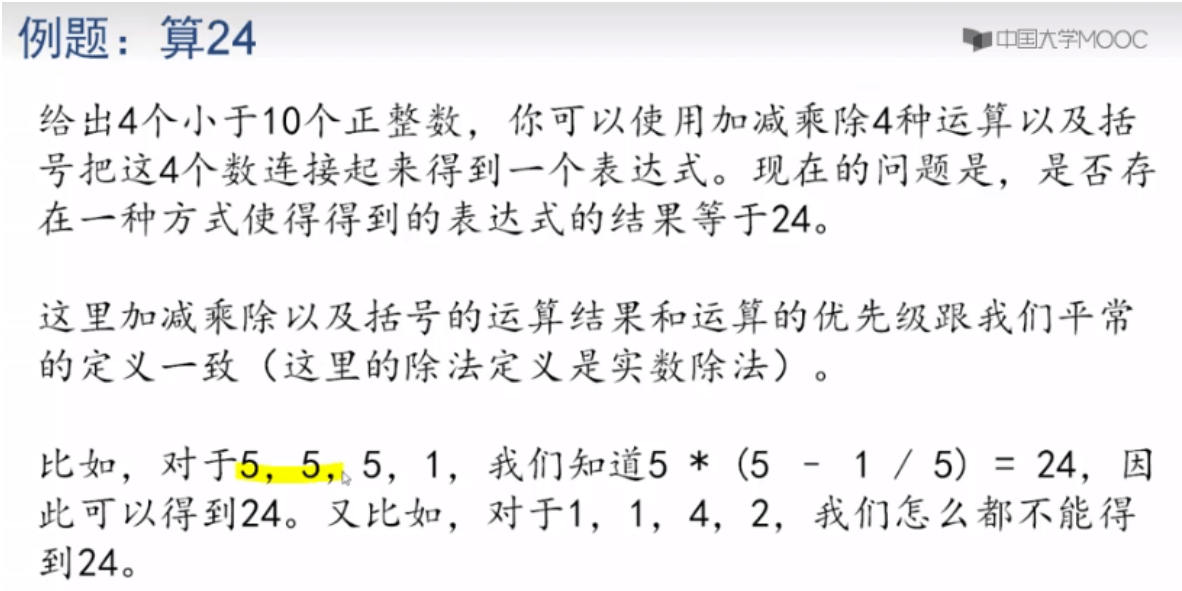
c++code

```
#include<iostream>
using namespace std;
int f(int m, int n){
    if (n > m)return f(m, m);
    if (m == 0)return 1;
    if (n == 0)return 0;
    return f(m, n - 1) + f(m - n, n);
    //有盘子为空和没有盘子为空
    //没有盘子为空代表，先每个盘子放一个。剩下m-n在放
}
void main(){
    int t, m,n;//m是苹果数目，n是盘子数目
    cin >> t;
    while (t--){
        cin >> m >> n;
        cout << f(m, n) << endl;
```

```
    }
    system("pause");
}
```

算24

题目



解释

c++ code

```
#include<iostream>
#include<cmath>
using namespace std;
#define EPS 1e-7
double a[5];
bool isZero(double x){
    return fabs(x) < EPS;
}
bool count24(double a[], int n){
    if (n == 1)return isZero(a[0] - 24);
    double b[5];
    for (int i = 0; i < n - 1; ++i){
        for (int j = i + 1; j < n; ++j){
            int m = 0;//还剩下m个数 m=n-2
```

```
            for (int t = 0; t < n; ++t){
                if (t != i&&t != j)
                    b[m++] = a[t];//把其余的数放b里面
            }
            b[m] = a[i] + a[j];
            if (count24(b, m + 1))return true;
            b[m] = a[i] - a[j];
            if (count24(b, m + 1))return true;
            b[m] = a[j] - a[i];
            if (count24(b, m + 1))return true;
            b[m] = a[i] * a[j];
            if (count24(b, m + 1))return true;
            if (!isZero(a[i])){
                b[m] = a[j] / a[i];
                if (count24(b, m + 1))return true;
            }
            if (!isZero(a[j])){
                b[m] = a[i] / a[j];
                if (count24(b, m + 1))return true;
            }
        }
    }
    return false;

}
void main(){
    int t = 4;
    for (int i = 0; i < t;++i){
        cin >> a[i];
    }
    cout<<count24(a, 4);
    system("pause");
}
```