4.3.1 递归取数

```
#include<stdio.h>
int count=0;
void fun(int a[],int n,int m,int i,int k,int ans[]);
//原始数组a[],从a数组的n个数当中选m个数，结果放在ans数组。
//i表示当前要选取第i个数。i从0开始。k表示上一次选取的数在a数组当中的下标。
//最开始时k==-1,表示还没选数据。
int main()
{
    int n,m,a[30],ans[30],i;
    freopen("5.out","w",stdout);
    scanf("%d%d",&n,&m);
    for(i=0;i<n;i++)
    {
        //scanf("%d",&a[i]);
        a[i]=i+1;
    }
    fun(a,n,m,0,-1,ans);
    printf("%d\n",count);
    return 0;
}
void fun(int a[],int n,int m,int i,int k,int ans[])
//原始数组a[],从a数组的n个数当中选m个数，结果放在ans数组。
//i表示当前要选取第i个数。i从0开始计数。k表示上一次选取的数在a数组当中的下标。
//最开始时k==-1,表示还没选数据。
{
    int j;
    if(i==m)
    {
        for(j=0;j<m;j++)    printf("%d ",ans[j]);
        printf("\n");
        count++;
        return ;
    }
    else
    {
        for(j=k+1;j<n-(m-i-1);j++)//从第k+1个开始，尝试选择第i个数。(注意：i从0开始计算)
        {
            ans[i]=a[j];
            fun(a,n,m,i+1,j,ans);
        }
    }
}
```

4.3.2 递归拆数

```
#include<stdio.h>
#define N 1001
int rec[N];
//start:当前拆分出的数
//pos：保存到rec数组的位置记录
//left_num：上一轮拆分后，还剩下的数
void print(int n){
    int i;
    printf("3=");
    for( i=0;i<n-1;i++){
        printf("%d+",rec[i]);

    }
    printf("%d\n",rec[n-1]);
    return;
}
void dfs(int start,int pos,int left_num){
    if(left_num==0){
        print(pos);
        return;
    }
    int i;
    for(i=start;i<=left_num;i++){
        rec[pos]=i;
        dfs(i,pos+1,left_num-i);
    }
    return;
}
int main(void) {
    int n;
    scanf("%d", &n);
    dfs(1, 0, n);

    return 0;
}
```

4.3.3 求素数之积

```
#include<stdio.h>
#include "math.h"
bool is_prime(int n) {
    int i;
    for(i = 2; i <=sqrt(n);i++){
```

```
        if(n%i==0){
            return 0;
        }
    }
    return 1;
}
int main() {
    int n = 0;
    scanf("%d", &n);
    int mul = 1;
    for(int i =1; i <=n ; i++) {
        if(is_prime(i)) {
            mul *= i;
        }
    }

    printf("%d", mul);

    return 0;
}
```

4.3.4 反转字符串

```
#include <cstring>
#include <iostream>

using namespace std;

int main() {
    string s;

    cin>>s;
    for(int i = s.size() - 1; i >= 0; i--) {
        printf("%c", s[i]);
    }

    return 0;
}
```

4.3.5 最长公共子序列

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int N = 1000;
char a[N],b[N];
```

```
int dp[N][N];
int main()
{
    int lena,lenb,i,j;
    while(scanf("%s%s",a,b)!=EOF)
    {
        memset(dp,0,sizeof(dp));
        lena=strlen(a);
        lenb=strlen(b);
        for(i=1;i<=lena;i++)
        {
            for(j=1;j<=lenb;j++)
            {
                if(a[i-1]==b[j-1])
                {
                    dp[i][j]=dp[i-1][j-1]+1;
                }
                else
                {
                    dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
                }
            }
        }
        printf("%d\n",dp[lena][lenb]);
    }
    return 0;
}
```

4.3.6 卖鸭子

```
#include<stdio.h>
#include <iostream>
using namespace std;
int sum;
int num;
int x[15];
int yazi(int n)
{
if(num+1==n)
    return 2;
else
{
    // printf("sell=%d\n",(sum+2)*2);
    x[num - n + 1] = (sum+2)*2;
    sum=(sum+1)*2;
    return (yazi(n+1)+1)*2;
}
}
```

```
int main()
{
    int k=1;
    cin >> num;
    printf("%d\n",yazi(k));
    for(int i =1 ; i <= num; i++) {
        printf("%d---%d\n", i, x[i]);
    }
    return 0;
}
```

4.3.7 进制转换

```
#include<bits/stdc++.h>
using namespace std;
int l=0,ans[500]= {0};//初始化放外面，我放在函数里面就每次递归都重新清了一遍0。。。
QAQ
void change(int x)
{
    if(x<8)
    {
        ans[l++]=x;
        for(int i=l-1; i>=0; i--)
        {
            cout<<ans[i];
        }
        return;
    }
    else
    {
        ans[l++]=x%8;
    }
    change(x/=8);
}
int main()
{
    int num;
    cin>>num;
    change(num);
    return 0;
}
```

4.3.8 角谷定理

```
#include <iostream>
using namespace std;
int main() {
```

```
    int n = 0;
    cin>>n;
    int res = 0;
    while(n != 1) {
        if(n % 2 == 0) {
            n = n/2;
            res += 1;
        } else {
            n = n*3+1;
            res += 1;
        }
    }
    cout << res + 1;
    return 0;
}
```

4.3.9杨辉三角

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


int yang(int i,int j)
{
    if(j==0||j==i)
        return 1;
    else
        return yang(i-1,j)+yang(i-1,j-1);
}
int main()
{
    int n;
    while(scanf("%d",&n)&&n!=0)
    {
        int i,j;
        for(i=0; i<n; i++)
        {
            for(j=0; j<=i; j++)
            {
                if(j!=i)
                    printf("%d ",yang(i,j));
                else
                    printf("%d",yang(i,j));
            }
            printf("\n");
        }
        printf("\n");
    }
```

```
    return 0;
}
```

4.3.10 质因数分解

```
#include<stdio.h>
#include "math.h"
bool is_prime(int n) {
    int i;
    for(i = 2; i <=sqrt(n);i++){
        if(n%i==0){
            return 0;
        }
    }
    return 1;
}
int main() {
    int n = 0;
    scanf("%d", &n);
    int mul = 1;
    for(int i =2; i < n ; i++) {
        if(is_prime(i) && n % i == 0) {
             printf("%d\n", i);
        }
    }



    return 0;
}
```

4.3.11 全排列

```
#include <iostream>
#include <string>
using namespace std;

void permute1(string prefix, string str)
{
  if(str.length() == 0)
    cout << prefix << endl;
  else
  {
    for(int i = 0; i < str.length(); i++)
      permute1(prefix+str[i],
str.substr(0,i)+str.substr(i+1,str.length()));
  }
```

```
}

void permute1(string s)
{
  permute1("",s);
}

int main()
{
  //method1, unable to remove duplicate permutations.
  string s;
  cin >> s;
  permute1(s);
}
```

4.3.12 特殊性质的数

```
#include<cstdio>
using namespace std;
int main(){
    int n,cnt=1,i,f[1010];
    f[0]=f[1]=1;
    scanf("%d",&n);
    for(i=2;i<=n;i++){
        if(i%2==0){
            f[i]=f[i-1]+f[i/2];
        }else{
            f[i]=f[i-1];
        }
    }
    printf("%d\n",f[n]);
}
```

4.3.13 放盘子 第二类斯特林数

```
#include<stdio.h>
int num[3][51]={0};
int backtrace(unsigned long a,int b)
{
    int i,j,n=0;
    unsigned long flag;
    for(i=0;i<=50;i++) num[1][i]=0;
        for(i=0;i<=50;i++) num[2][i]=0;
num[1][1]=1;num[2][1]=1;
for(flag=2;flag<=a;flag++){
    for(j=1;j<=flag&&j<=b;j++){
        num[2][j] =(num[1][j-1] + (j*num[1][j]));
```

```
    }//第flag行的计算
    n++;

    if(flag==a||a==1) return 2;
    flag++;
    for(j=1;j<=flag&&j<=b;j++){
        num[1][j] =(num[2][j-1] + (j*num[2][j]));
    }

    if(flag==a||a==1) return 1;
    if(n>=1) for(j=1;j<=flag&&j<=b;j++){
        num[1][j]%=10000;n=0;}
}


return 0;
}

int main()
{
    int cas,c,n,i,b;
    unsigned long a;
    // scanf("%d",&cas);
    // for(i=1;i<=cas;i++)
    // {
    scanf("%d %d",&a,&b);
    n=backtrace(a,b);
    printf("%d\n",num[n][b]%10000);
    //
    return 0;
}
```

4.3.14 无序划分

```
#include<iostream>
#include<stdio.h>
using namespace std;
int dp[4505][4505];
int solve(int n,int m)
{
int i,j;
for(i=1;i<=n;++i)
{
   dp[i][0]=0;
   for(j=1;j<=m;++j)
   {
    dp[0][j]=0;
    if(i>=j)
```

```
      dp[i][j]=dp[j-1][j]+1;
     else
     {
      dp[i][j]=dp[i-1][j]+dp[i][j-i];
      if(dp[i][j]>=1000000007)
       dp[i][j]-=1000000007;
     }
    }
}
return dp[n][m];
}
int main()
{
int n,m;
scanf("%d %d",&n,&m);
printf("%d\n",solve(n,m));
return 0;
}
```

4.3.15 回文数

```
#include <iostream>
using namespace std;
const int BASE = 10;
int palindrom(int n)
{
    if(n == 1 || n == 2)
        return BASE - 1;
    else {
        if(n % 2 == 1)
            return palindrom(n-1) * BASE;
        else
            return palindrom(n-2) * BASE;
    }
}
// 递推的计算回文数函数，参数为10进制位数
int palindrom2(int n)
{
    if(n == 1 || n == 2)
        return BASE - 1;
    int p1= BASE - 1, p2 = BASE - 1, i, temp;
    i = 2;
    while(i < n) {
        i++;
        if(i % 2 == 1) {
            p1 = p2;
            p2 = p1 * BASE;
        } else {
```

```
            temp = p1;
            p1 = p2;
            p2 = temp * BASE;
        }
    }
    return p2;
}
int main()
{
    int n, sum;
    cin >> n;
    sum = 0;
    for(int i=1; i<=n; i++)
        sum += palindrom2(i);
    cout << sum << endl;
    return 0;
}
```

5.3.1 素数筛选问题

简单

5.3.2 纸币换硬币

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main(){
  int n = 0;
  scanf("%d", &n);

  int result = 0;
  int num = n * 100;
  for (int i = 1; i < num; i++) {
    for (int j = 1 ; j < num; j++) {
      for (int k = 1 ; k < num; k++) {
        if ((i + 2*j + 5*k) == num) {
          result += 1;
        }
      }
    }
  }
  cout << result << endl;
  return 0;
}
```

5.3.3 勾股数问题

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main(){
  int num = 0;
  scanf("%d", &num);
  int result = 0;
  for (int i = 1; i < num; i++) {
    for (int j = i ; j < num; j++) {
      for (int k = j ; k < num; k++) {
        if ((i*i + j*j) == k*k) {
          printf("%d,%d,%d\n", i,j,k);
        }
      }
    }
  }
  return 0;
}
```

5.3.4 生理周期问题

```
#include <iostream>
#include<cstdio>
using namespace std;
#define N 21252
int main()
{
    int p,e,i,d,n,k;
    n=0;
    while(cin >> p >> e >> i >> d && p!=-1)
    {
        n++;
        for(k=d+1;(k-p)%23;k++);
        for( ;(k-e)%28;k+=23);
        for( ;(k-i)%33;k+=23*28);
        cout << "Case " << n << ": the next triple peak occurs in " << k-d
<< endl;
    }
    return 0;
}
```

5.3.5 构造比例数

简单

5.3.6 自守数

```
#include <stdio.h>
int main()
{
    //巧妙的绕过了 计算平方的过程，因为太大的数字平方超出了范围。
  long mul,number,k,a,b;
  printf("将输出200000以内的自守数：\n");
  for(number=1; number<200000; number++)
  {
    for(mul=number,k=1; (mul/=10)>0; k*=10) ; //k得到的值为mul或是number的权
数,用于后面数字的截取
        a=k*10;
        mul=0;
        b=10;
        while(k>0)
        {
          mul=(mul+(number%(k*10))*(number%b-number%(b/10)))%a;
          k/=10;
          b*=10;
        }
    if(number==mul)
        printf("%ld  \n",number);
  }
}
```

5.3.7 谁是窃贼

```
public class WhoIsTheThief {
    public static void main(String[] args) {

        for (int i = 0; i < 4; i++) {
            // 第一次循环赋甲为贼  第二次赋乙为贼...
            boolean[] isthief = { false, false, false, false };
            isthief[i] = true;
            //题目说每个人的话要么全为真    要么全为假
            //所以有两种可能    所以只要符合一句话全为真或全为假就可以
            //所以一句话表示为全为真  和  全为假两种形式    用或连接    足以符合题目要
求
            //最后还得符合四句话    所以 将他们用户&&连接    必须符合题目的四个条件
            if ((((!isthief[1] && isthief[3]) || (isthief[1] &&
!isthief[3])) && // 甲说：“乙没有偷，是丁偷的。”
                    ((!isthief[1] && isthief[2]) || (isthief[1] &&
!isthief[2])) && // 乙说：“我没有偷，是丙偷的。”
                    ((!isthief[0] && isthief[1]) || (isthief[0] &&
!isthief[1])) && // 丙说：“甲没有偷，是乙偷的。”
                    (!isthief[3] || isthief[3])) { // 丁说：“我没有偷”
```

```
                switch (i) {
                case 0:
                    System.out.println("甲是贼");
                    break;
                case 1:
                    System.out.println("乙是贼");
                    break;
                case 2:
                    System.out.println("丙是贼");
                    break;
                case 3:
                    System.out.println("丁是贼");
                    break;
                default:
                    break;
                }
            }
        }
    }
};
```

5.3.8 独特的数

简单

5.3.9 握手问题

```
#include <iostream>
#include <cstdio>
using namespace std;
int main() {
  int n = 0;
  int num = 0;
  scanf("%d", &n);
  for(int i = 0 ; i < n; i++) {
    scanf("%d", &num);
    int result = 0;
    cout<<num*(num-1)/2<<endl;
  }
  return 0;
}
```

5.3.10 趣味数学

```
简单
```

5.3.11 暴力枚举之绝对值

简单

5.3.12 回文数

```
#include<stdio.h>
int main(){
int n;
int i,a,b,c,d,e;
int j,o,p,q,r,s,t;
scanf("%d",&n);
for (i=10000;i<=99999;i++){
    a=i/10000;
    b=(i%10000)/1000;
    c=(i%1000)/100;
    d=(i%100)/10;
    e=i%10;
    if (a==e&&b==d&&a+b+c+d+e==n){
        printf("%d\n",i);
    }
}
for (j=100000;i<=999999;j++){
    o=j/100000;
    p=(j%100000)/10000;
    q=(j%10000)/1000;
    r=(j%1000)/100;
    s=(j%100)/10;
    t=j%10;
    if (o==t&&p==s&&q==r&&o+p+q+r+s+t==n){
        printf("%d\n",j);
    }
}
    return 0;
}
```

5.3.13 逆序对数

设A[1...n]是一个包含n个不同数的数组。如果在i < j的情况下，有A[i] > A[j]，则（i,j）就称为A中的一个**逆序对**

```
#include <cstdio>

int main() {
    long long MOD = 1e9 + 7, DP[200] = {0}, per[200] = {1, 1};
    for (int i = 2; i < 101; i++) {
```

```
        DP[i] = (per[i - 1] * i * (i - 1) / 2 % MOD + i * DP[i - 1] % MOD)
% MOD;
        per[i] = per[i - 1] * i % MOD;
    }
    int N, A[200];
    while (scanf("%d", &N) != EOF) {
        for (int i = 0; i < N; i++)
            scanf("%d", &A[i]);
        int vis[200] = {0};
        long long ans = 0, cnt = 0;
        for (int i = 0; i < N; i++) {
            for (int j = 1; j < A[i]; j++)
                if (!vis[j]) {
                    ans = (((ans + cnt * per[N - i - 1]) % MOD + DP[N - i -
1]) % MOD);
                    cnt++;
                }
            vis[A[i]] = 1;
        }
        printf("%lld\n", ans);
    }
    return 0;
}
```

5.3.14 放牧

```
///给你n个点 枚举三点求最小三角形面积
//利用向量点乘得到三角形的面积
# include <algorithm>
# include <stdio.h>
# include <string.h>
# include <iostream>
# include <math.h>
# include <string>
using namespace std;
int main()
{
    int t,n,i,j,k;
    double x[110],y[110],minn,s;
    while(~scanf("%d",&t))
    {
        while(t--)
        {
            scanf("%d",&n);
            for(i=0; i<n; i++)
                scanf("%lf %lf",&x[i],&y[i]);
            minn=999999999;
            int flag=0;
```

```
            if(n<=2)
                printf("Impossible\n");
            else
            {

                for(i=0; i<n; i++)
                {
                    for(j=i+1; j<n; j++)
                    {
                        for(k=j+1; k<n; k++)
                        {

 if(x[i]==x[j]&&x[j]==x[k]||y[i]==y[j]&&y[j]==y[k])
                                s=0;
                            ///  s=fabs((x[i]-x[k])*(y[j]-y[k])-(x[j]-x[k])*
(y[i]-y[k]))/2.0;///已知三点算面积
                                s=(1/2.0)*fabs(x[i]*y[j]+x[j]*y[k]+x[k]*y[i]-
x[i]*y[k]-x[j]*y[i]-x[k]*y[j]);
                            if(s>=0.005)
                            {
                                minn=min(s,minn);
                                flag=1;
                            }
                        }
                    }
                }
                if(flag)
                    printf("%.2lf\n",minn);
                else
                    printf("Impossible\n");
            }
        }
    }
    return 0;
}
```

5.3.15 餐厅点餐

```
#include<cstdio>
#include<string>
#include<iostream>
#include<algorithm>
using namespace std;
typedef long long LL;
const int maxn=1e5+10;
int T,n[5],a[5][10],f[3][111],t[3],l,r;

int main()
```

```
{
    scanf("%d",&T);
    while(T--)
    {
        for (int i=0;i<5;i++) scanf("%d",&n[i]);
        for (int i=0;i<5;i++)
        {
            for (int j=0;j<n[i];j++) scanf("%d",&a[i][j]);
        }
        t[0]=t[1]=t[2]=0;
        for (int i=0;i<n[3];i++) f[0][t[0]++]=a[3][i];
        for (int i=0;i<n[3];i++) for (int j=i+1;j<n[3];j++) f[0]
[t[0]++]=a[3][i]+a[3][j];
        for (int i=0;i<n[4];i++) f[1][t[1]++]=a[4][i];
        for (int i=0;i<n[4];i++) for (int j=i+1;j<n[4];j++) f[1]
[t[1]++]=a[4][i]+a[4][j];
        for (int i=0;i<n[2];i++) f[2][t[2]++]=a[2][i];
        for (int i=0;i<n[0];i++) for (int j=0;j<n[1];j++) f[2][t[2]++]=a[0]
[i]+a[1][j];
        scanf("%d%d",&l,&r);
        int ans=0;
        for (int i=0;i<t[0];i++)
            for (int j=0;j<t[1];j++)
                for (int k=0;k<t[2];k++)
                    if (f[0][i]+f[1][j]+f[2][k]>=l&&f[0][i]+f[1][j]+f[2][k]
<=r) ans++;
        printf("%d\n",ans);
    }
    return 0;
}
```

6.3.1 报数问题

```
#include <iostream>
#include <cstdio>
using namespace std;
int main() {
    int n = 0;
    scanf("%d", &n);

    string name[n + 1];
    bool flag[n + 1];

    for(int i = 1; i <= n; i++) {
        cin>>name[i];
        flag[i] = 0;
    }
    int W = 0;
```

```
    int S = 0;
    scanf("%d,%d", &W, &S);
    int people = 0;
    while(people != n) {
        int num = 0;
        int i = 0;
        bool first = 0;
        while(num != S) {
            i++;
            int pos = i % n + 1;
            if(first == 0&&pos != W) {
                continue;
            } else if(first == 0 && pos == W){
                first = 1;
            }
            if(flag[pos] == 0) {
                num += 1;
            }
            if(num == S && flag[pos] == 0) {
                people += 1;
                flag[pos] = 1;
                if(pos + 1 <= n) {
                    W = pos + 1;
                } else {
                    W = (pos + 1) % 5;
                }
                cout << name[pos] <<endl;
            }
        }
    }
    return 0;
}
```

6.3.2 无限次幂

```
#include<iostream>
#include<cmath>
using namespace std;
void is_zero(int n){
    int m = (sqrt(8*n+1)-1)/2;
    int re = n - m*(m+1)/2;
    if(n == 1||re == 1){
        cout<<1<<endl;
    }
    else{
        cout<<0<<endl;
    }
}
```

```
int main(){
    int t = 0;
    cin >> t;
    int x;
    for(int i = 1;i<=t;++i){
        cin >> x;
        is_zero(x);
    }
    return 0;
}
```

6.3.3 金币工资

```
#include <stdio.h>
int main()
{
    int n,i,j,k,sum;
    while(scanf("%d",&n)!=EOF)
    {
        sum=0; j=1; k=1;
        for(i=1;i<=n;i++){
            sum+=k;
            if(i==j){
                k++;  //k表示当天可获得的金币数
                j+=k; //j表示金币能增加时需要跨过的天数
            }
        }
        printf("%d\n",sum);
    }
    return 0;
}
```

6.3.4 进制转换

```
#include<iostream>
using namespace std;
int main(){
int N;
int aa[20];
cin>>N;
int temp;
temp = N;
int num;
int i=0;
while(temp !=0)
{
num = temp%2;
```

```
aa[i] = num;
i++;
temp = temp/2;
}
for(num=i-1;num>=0;num--)
{
cout<<aa[num];
}
cout<<endl;
return 0;
}
```

6.3.5 卡片魔术

```
#include <stdio.h>
int main()
{
    int n,i,t1,t2;
    scanf("%d",&n);
    while(n--)
    {
        int sum=2;  //直接让sum=2代表卡片2的位置。
        int m;
        scanf("%d",&m);
        for(i=1;i<=m;i++)
        {
            scanf("%d%d",&t1,&t2);
            if(t2==sum)  //  相等的话就交换
                sum=t1;
            else    if(t1==sum)
                sum=t2;  //   同上
        }
        printf("%d\n",sum);  //换完后就得到最后的位置。
    }
}
```

6.3.6 木棍上的蚂蚁

```
#include<cstdio>
#include<algorithm>
using namespace std;

const int maxn = 10000+5;
const char dirName[][10] = {"L", "Turing", "R"};
int order[maxn];        //输入的第i只蚂蚁是终态中的左数第order[i]只蚂蚁
struct Ant
{
```

```
    int id;     //输入顺序
    int p;      //位置
    int d;      //朝向。-1:左；0:转身中；1：右
    bool operator < (const Ant &a) const
    {
        return p < a.p;
    }
}before[maxn], after[maxn];

int main()
{
    int K;
    scanf("%d", &K);
    for(int ki = 1; ki <= K; ki++)
    {
        int i;
        int L, T, n;
        scanf("%d%d%d", &L, &T, &n);
        for(i = 0; i < n; i++)
        {
            int p, d;
            char c;
            scanf("%d %c", &p, &c);
            d = (c=='L') ? -1 : 1;
            before[i].id = i; before[i].p = p; before[i].d = d;
            after[i].id = 0; after[i].p = p+T*d; after[i].d = d;    //此处
id还是未知的，用0代替
        }
        //计算次序数组
        sort(before, before+n);
        for(i = 0; i < n; i++)
            order[before[i].id] = i;
        //计算终态，保持原态或其它态
        sort(after, after+n);
        for(i = 0; i < n-1; i++)
            if(after[i].p == after[i+1].p) after[i].d = after[i+1].d = 0;
        //输出结果
        printf("Case #%d:\n", ki);
        for(i = 0; i < n; i++)
        {
            int a = order[i];//恢复输入次序
            if(after[a].p < 0 || after[a].p > L) printf("Fell off\n");
            else printf("%d %s\n", after[a].p, dirName[after[a].d+1]);
        }
        printf("\n");
    }
    return 0;
}
```

6.3.7 串取数字

```
#include <iostream>
#include <cstdio>
using namespace std;
int main(){
    int t;
    scanf("%d",&t);
    while(t--){
        int n;
        scanf("%d",&n);
        int a = 1;//a表示当前串s的数字的个数
        while(n > a){//如果第n个位置不在当前串中
            n -= a;//则将n减去当前串所包含的数字的个数
            a++;//计算下一个串的数字的个数
        }
        //执行到这里的时候,说明第n个位置在当前串s中
        n %= 9;//因为每一个都是在串1~9之间循环
        if(n == 0){//用于处理一下n==9的时候
            n = 9;
        }
        printf("%d\n",n);
    }
    return 0;
}
```

6.3.8 多连块覆盖问题

```
#include"algorithm"
#include"iostream"
#include"cstring"
#include"cstdlib"
#include"cstdio"
#include"string"
#include"vector"
#include"queue"
#include"cmath"
#include"map"
using namespace std;
typedef long long LL ;
#define memset(x,y) memset(x,y,sizeof(x))
#define memcpy(x,y) memcpy(x,y,sizeof(x))
#define FK(x) cout<<"["<<x<<"]\n"
#define bigfor(T)  for(int qq=1;qq<= T ;qq++)
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
struct Box {
```

```
    int x,y;
    Box(int xx,int yy):x(xx),y(yy) {};
    Box() {};
} into[123];

int n,m;
int num2,num;
int erear;
char maps[20][20],box[20][20],sox[20][20],vis[20][20];
int dir[4][2]= {{0,1},{1,0},{-1,0},{0,-1}};

void init() {
    erear=0;
    memset(vis,0);
}
void DFS(int x,int y,int tox,int toy) {
    if(!vis[x][y]) {
        vis[x][y]=1;
        into[erear++]=Box(tox,toy);//记录box的覆盖坐标
        for(int i=0; i<4; i++) {
            int xx=x+dir[i][0];
            int yy=y+dir[i][1];
            if(xx<0||yy<0||xx>=m||yy>=m)continue;
            if(box[xx][yy]!='*')continue;
            if(vis[xx][yy])continue;
            DFS(xx,yy,tox+dir[i][0],toy+dir[i][1]);
        }
    }
}

int main() {
    while(~scanf("%d%d",&n,&m)) {
        if(!n&&!m)break;
        num=0;
        num2=0;
        for(int i=0; i<n; i++) {
            scanf("%s",maps[i]);
            for(int j=0; j<n; j++) {
                if(maps[i][j]=='*')num++;
            }
        }
        for(int i=0; i<m; i++) {
            scanf("%s",box[i]);
            for(int j=0; j<m; j++) {
                if(box[i][j]=='*')num2++;
            }
        }
        if(num%num2) {
            puts("0");
```

```
                continue;
            }
            int stx,sty;
            int flag=1;
            init();
            for(int i=0; i<m; i++) {
                for(int j=0; j<m; j++) {
                    if(box[i][j]=='*') {
                        DFS(i,j,0,0);
                        flag=0;
                    }
                    if(!flag)break;
                }
                if(!flag)break;
            }
            int tot=0;
            flag=0;
            for(int i=0; i<n; i++) {
                for(int j=0; j<n; j++) {
                    if(maps[i][j]=='*') {
                        for(int k=0; k<erear; k++) {
                            if(maps[i+into[k].x][j+into[k].y]!='*') {//模拟覆
盖。
                                flag=1;
                                break;
                            }
                            maps[i+into[k].x][j+into[k].y]='.';
                            tot++;
                        }
                        if(flag)break;
                    }
                    if(flag)break;
                }
            }
            if(tot==num)puts("1");
            else puts("0");
        }
        return 0;
    }
```

6.3.9 括号表达式

```
#include<stdio.h>
#include<string.h>
using namespace std;
int main()
{
```

```
    char s[42];
    int p[22],w[22];
    int t,book[42]={0};
    scanf("%d",&t);
    while(t--)
    {
        int n,i,j,m=0;
        memset(book,0,sizeof(book));
        scanf("%d",&n);
        p[0]=0;
        for(i=1;i<=n;i++)
        {
            scanf("%d",&p[i]);
            int temp=p[i]-p[i-1];
           while(temp--)
            {
                s[m++]='(';
            }
            s[m++]=')';
        }
        int mm=1;
        for(i=0;i<m;i++)
        {
            if(s[i]==')')
             {
                for(j=i-1;j>=0;j--)
                 {
                    if(s[j]=='('&&!book[j])
                       {
                        w[mm++]=(i-j+1)/2;
                        book[j]=1;
                        break;
                       }
                 }
             }
        }
        for(i=1;i<mm;i++)
           printf("%d ",w[i]);
        printf("\n");
    }
    return 0;
}
```

6.3.10 假币问题

```
#include<iostream>
using namespace std;
int main()
```

```
{
     int n,k;
     cin>>n>>k;
     bool*coin=new bool[n+1];
     int *qCoin=new int[n+1];
     int *wCoin=new int[n+1];
    for (int i = 1; i < n+1; i++)
    {
        coin[i]=false;
        qCoin[i]=0;
        wCoin[i]=0;
    }
    int comareCoinNum,nEqual=0;
    for (int i = 0; i < k; i++)
    {
        cin>>comareCoinNum;
        int *temp=new int[2*comareCoinNum];
        for (int i = 0; i < 2*comareCoinNum; i++)
        {
            cin>>temp[i];
        }
        char c;
        cin>>c;
        if(c=='=')
        {
            for (int i = 0; i < 2*comareCoinNum; i++)
                coin[temp[i]]=true;
        }
        else if(c=='<')
        {
            for (int i = 0; i < comareCoinNum; i++)
                qCoin[temp[i]]++;
            for (int i = comareCoinNum; i < 2*comareCoinNum; i++)
                wCoin[temp[i]]++;
            nEqual++;
        }
        else
        {
            for (int i = 0; i < comareCoinNum; i++)
                wCoin[temp[i]]++;
            for (int i = comareCoinNum; i < 2*comareCoinNum; i++)
                qCoin[temp[i]]++;
            nEqual++;
        }
    }
    int fCoin=0,t;
    for (int i = 1; i < n+1; i++)
    {
        if((coin[i]==false) && (nEqual==qCoin[i]||nEqual==wCoin[i]))
```

```
            {
                    fCoin++;
                    t=i;
            }
        }
        if(fCoin==1)
            cout<<t<<endl;
        else
            cout<<0<<endl;
        return 0;
    }
```

6.3.11 会议安排

```
#include <cstdio>
#include <iostream>
#include <string.h>
#include <fstream>
#include <stdlib.h>
using namespace std;
int n,q;
int m;
int date[102];
int t;
int main()
{
    //ifstream cin("input.txt");
    while(cin>>n>>q,n)
    {
        int maxd=0;
        int maxn=-0x3f3f3f3f;
        int tt=0;
        memset(date,0,sizeof(date));
        while(n--)
        {
            cin>>m;
            while(m--)
            {
                cin>>t;
                if(maxd<t) maxd=t;
                date[t]++;
            }
        }
        int ok=0;
        for(int i=1;i<=maxd;i++)
        {
            if(date[i]>=q&&maxn<date[i])
```

```
            {
                maxn=date[i];
                tt=i;
                ok=1;
            }
        }
        if(ok) cout<<tt<<endl;
        else cout<<0<<endl;
    }
    return 0;
}
```

6.3.12 取火柴游戏（尼姆博弈）

有任意堆物品，每堆物品的个数是任意的，双方轮流从中取物品，每一次只能从一堆物品中取部分或全部物品，最少取一件，取到最后一件物品的人获胜。

```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
int n,ans,a;
int main() {
    while(scanf("%d",&n)==1) {
        ans=0;
        while(n-->0) {
            scanf("%d",&a);
            ans^=a;
        }
        printf("%s\n",ans==0?"No":"Yes");
    }
    return 0;
}
```

6.3.13 取石子游戏（威佐夫博弈）

有两堆各若干的物品，两人轮流从其中一堆取至少一件物品，至多不限，或从两堆中同时取相同件物品，规定最后取完者胜利。

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int a,b;
    while(cin>>a>>b)
    {
```

```
        if(a>b)
            swap(a,b);
        double k=b-a;
        double x=(1+sqrt(5.0))/2;
        int t=(k*x+k);
        if(t==b)
            cout<<0<<endl;
        else
            cout<<1<<endl;
    }
    return 0;
}
```

巴什博弈

只有一堆n个物品，两个人轮流从中取物，规定每次最少取一个，最多取m个，最后取光者为胜

Tang和Jiang轮流写数字，Tang先写，每次写的数x满足1<=x<=k，Jiang每次写的数y满足1<=y-x<=k，谁先写到不小于n的数算输

```
#include<iostream>
using namespace std;
int main()
{
    int n,k;
    while(~scanf("%d%d",&n,&k)&&n&&k)
    {
        if((n-1)%(k+1)==0)//第n-1个肯定后手拿，先手必败
            printf("Jiang\n");
        else
            printf("Tang\n");
    }
    return 0;
}
```

斐波那契博弈

有一堆物品，两人轮流取物品，先手最少取一个，至多无上限，但不能把物品取完，之后每次取的物品数不能超过上一次取的物品数的二倍且至少为一件，取走最后一件物品的人获胜。

先手胜当且仅当n不是斐波那契数

```
#include <iostream>
#include <string.h>
#include <stdio.h>
using namespace std;
const int N = 55;
int f[N];
```

```
void Init()
{
    f[0] = f[1] = 1;
    for(int i=2;i<N;i++)
        f[i] = f[i-1] + f[i-2];
}
int main()
{
    Init();
    int n;
    while(cin>>n)
    {
        if(n == 0) break;
        bool flag = 0;
        for(int i=0;i<N;i++)
        {
            if(f[i] == n)
            {
                flag = 1;
                break;
            }
        }
        if(flag) puts("Second win");
        else     puts("First win");
    }
    return 0;
}
```

6.3.14 伪造的美元

```
#include<iostream>
#include<cmath>
using namespace std;

int main(void)
{
    int cases;
    cin>>cases;
    for(int c=1;c<=cases;c++)
    {
        char left[3][6],right[3][6],status[3][6];
        int time['L'+1]={0};  //标记各个字母被怀疑的次数
        bool zero['L'+1]={false};  //标记绝对为真币的字母（令天枰平衡的所有字母）
        for(int k=0;k<3;k++)
            cin>>left[k]>>right[k]>>status[k];
        for(int i=0;i<3;i++)
        {
            switch(status[i][0])  //检查天枰状态
```

```
            {
                case 'u':     //up，天枰左重右轻
                    {
                        for(int j=0;left[i][j];j++)
                        {
                            time[ left[i][j] ]++;  //左重
                            time[ right[i][j] ]--;  //右轻
                        }
                        break;
                    }
                case 'd':     //down，天枰左轻右重
                    {
                        for(int j=0;left[i][j];j++)
                        {
                            time[ left[i][j] ]--;  //左轻
                            time[ right[i][j] ]++;  //右重
                        }
                        break;
                    }
                case 'e':     //down，天枰平衡
                    {
                        for(int j=0;left[i][j];j++)
                        {
                            zero[ left[i][j] ]=true;   //绝对真币
                            zero[ right[i][j] ]=true;   //绝对真币
                        }
                        break;
                    }
            }
        }
        int max=-1;  //查找被怀疑程度最高的硬币（假币）
        char alpha;
        for(int j='A';j<='L';j++)
        {
            if(zero[j])  //绝对真币
                continue;

            if(max<=abs(time[j]))
            {
                max=abs(time[j]);
                alpha=j;
            }
        }
        cout<<alpha<<" is the counterfeit coin and it is ";
        if(time[alpha]>0)
            cout<<"heavy."<<endl;
        else
            cout<<"light."<<endl;
    }
```

```
    return 0;
}
```

6.3.15 HTML浏览器

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string s;
    int i,cnt=0;
    while(cin>>s){
        if(s=="<br>"){
            cout<<endl;
            cnt=0;
        }else if(s=="<hr>"){
            if(cnt)
                cout<<endl;
            for(int i=0;i<80;++i)
                cout<<"-";
            cout<<endl;
            cnt=0;
        }else{
            if(cnt+s.length()+(cnt==0?0:1)>80){
                cout<<endl<<s;
                cnt=s.length();
            }else{
                if(cnt){
                    cout<<" "<<s;
                    cnt+=s.length()+1;
                }else{
                    cout<<s;
                    cnt+=s.length();
                }
            }
        }
    }
    return 0;
}
```

7.3.1 数组二分求和

```
#include <iostream>
#include <string>
using namespace std;
int add_array(int left, int right, int a[]) {
```

```
    if(right == left) {
        return a[left];
    }
    else {
        int mid = (left + right) >> 1;
        return add_array(left, mid, a) + add_array(mid+1, right, a);
    }
}
int main()
{
    int n = 0;
    cin >> n;
    int a[n];
    for(int i = 0; i < n; i++) {
        cin >> a[i];
    }
    cout << add_array(0, n-1, a);
    return 0;
}
```

7.3.2 子序列最大值

```
#include <iostream>
#include <string>
using namespace std;

int getMaxNum(int a,int b,int c){
    if (a > b&&a > c){
        return a;
    }
    if (b > a&&b > c){
        return b;
    }
    return c;
}
int maxSumRec(int data[], int left, int right){
    if (right - left == 1){
        //如果当前序列只有一个元素
        return data[left];
    }
    int center = (left + right) / 2;//计算当前序列的分裂点
    int maxLeftSum = maxSumRec(data,left,center);
    int maxRightSum = maxSumRec(data,center,right);
    //计算左边界最大子序列和
    int leftBonderSum = 0;
    int maxLeftBonderSum = data[center-1];
    for (int i = center - 1; i >= left; i--){
        leftBonderSum += data[i];
```

```
        if (maxLeftBonderSum < leftBonderSum){
            maxLeftBonderSum = leftBonderSum;
        }
    }
    //计算右边界最大子序列和
    int rightBonderSum = 0;
    int maxRightBonderSum = data[center];
    for (int i = center; i < right; i++){
        rightBonderSum += data[i];
        if (maxRightBonderSum < rightBonderSum){
            maxRightBonderSum = rightBonderSum;
        }
    }
    //返回当前序列最大子序列和
    return getMaxNum(maxLeftBonderSum + maxRightBonderSum, maxLeftSum,
maxRightSum);
}

int main()
{
    int n = 0;
    cin >> n;
    int a[n];
    for(int i = 0; i < n; i++) {
        cin >> a[i];
    }
    cout << maxSumRec(a, 0 ,n);
    return 0;
}
```

7.3.3 棋盘覆盖

```
#include <stdio.h>

#define BOARD_SIZE 8
int board[BOARD_SIZE][BOARD_SIZE];

// c1, r1: 棋盘左上角的行号和列号
// c2, r2: 特殊方格的行号和列号
// size = 2 ^ k
void chessboard(int r1, int c1, int r2, int c2, int size)
{
    if(1 == size) return;
    int half_size;
    static int domino_num = 1;
    int d = domino_num++;
    half_size = size / 2;
```

```
    if(r2 < r1 + half_size && c2 < c1 + half_size) //特殊方格在左上角子棋盘
    {
       chessboard(r1, c1, r2, c2, half_size);
    }
    else   // 不在此棋盘，将此棋盘右下角设为相应的骨牌号
    {
       board[r1 + half_size - 1][c1 + half_size - 1] = d;
       chessboard(r1, c1, r1 + half_size - 1, c1 + half_size - 1,
half_size);
    }

    if(r2 < r1 + half_size && c2 >= c1 + half_size) //特殊方格在右上角子棋盘
    {
       chessboard(r1, c1 + half_size, r2, c2, half_size);
    }
    else  // 不在此棋盘，将此棋盘左下角设为相应的骨牌号
    {
       board[r1 + half_size - 1][c1 + half_size] = d;
       chessboard(r1, c1 + half_size, r1 + half_size - 1, c1 + half_size,
half_size);
    }

    if(r2 >= r1 + half_size && c2 < c1 + half_size) //特殊方格在左下角子棋盘
    {
       chessboard(r1 + half_size, c1, r2, c2, half_size);
    }
    else  // 不在此棋盘，将此棋盘右上角设为相应的骨牌号
    {
       board[r1 + half_size][c1 + half_size - 1] = d;
       chessboard(r1 + half_size, c1, r1 + half_size, c1 + half_size - 1,
half_size);
    }

    if(r2 >= r1 + half_size && c2 >= c1 + half_size) //特殊方格在左上角子棋盘
    {
       chessboard(r1 + half_size, c1 + half_size, r2, c2, half_size);
    }
    else   // 不在此棋盘，将此棋盘左上角设为相应的骨牌号
    {
       board[r1 + half_size][c1 + half_size] = d;
       chessboard(r1 + half_size, c1 + half_size, r1 + half_size, c1 +
half_size, half_size);
    }
}

int main()
{
    int i, j;
    board[2][2] = 0;
```

```
    chessboard(0, 0, 2, 2, BOARD_SIZE);
    for(i = 0; i < BOARD_SIZE; i++)
    {
        for(j = 0; j < BOARD_SIZE; j++)
        {
           printf("%-4d", board[i][j]);
        }
        printf("\n");
    }
}
```

7.3.4 最接近点对问题

```
#include <ctime>
#include <cmath>
#include <iostream>
#include <algorithm>

using namespace std;

// 分治法求解最近点对问题
// @`13
// 2017年4月21日
// 参考 : http://blog.csdn.net/to_baidu/article/details/50315607
// 参考 :
http://www.cnblogs.com/king1302217/archive/2010/07/08/1773413.html

#define INFINITE_DISTANCE 65535    // 无限大距离
#define COORDINATE_RANGE 100.0    // 横纵坐标范围为[-100,100]

#ifndef Closest_pair

typedef struct Point
{// 二维坐标上的点Point
    double x;
    double y;
}Point;

double Distance(Point a, Point b)
{//平面上任意两点对之间的距离公式计算
    return sqrt((a.x - b.x)*(a.x - b.x) + (a.y - b.y)*(a.y - b.y));
}

bool compareX(Point a, Point b)
{//自定义排序规则：依照结构体中的x成员变量升序排序
    return a.x < b.x;
}
```

```
bool compareY(Point a, Point b)
{//自定义排序规则：依照结构体中的x成员变量升序排序
    return a.y < b.y;
}

float ClosestPair(Point points[], int length, Point &a, Point &b)
{// 求出最近点对记录，并将两点记录再a、b中
    double distance;                    //记录集合points中最近两点距离
    double d1, d2;                      //记录分割后两个子集中各自最小点对距离
    int i = 0, j = 0, k = 0, x = 0;     //用于控制for循环的循环变量
    Point a1, b1, a2, b2;               //保存分割后两个子集中最小点对

    if (length < 2)
        return INFINITE_DISTANCE;     //若子集长度小于2，定义为最大距离，表示不可达
    else if (length == 2)
    {//若子集长度等于2，直接返回该两点的距离
        a = points[0];
        b = points[1];
        distance = Distance(points[0], points[1]);
    }
    else
    {//子集长度大于3，进行分治求解
        Point *pts1 = new Point[length];      //开辟两个子集
        Point *pts2 = new Point[length];

        sort(points, points + length, compareX);     //调用algorithm库中的
sort函数对points进行排序，compareX为自定义的排序规则
        double mid = points[(length - 1) / 2].x;     //排完序后的中间下标值，即
中位数

        for (i = 0; i < length / 2; i++)
            pts1[i] = points[i];
        for (int j = 0, i = length / 2; i < length; i++)
            pts2[j++] = points[i];

        d1 = ClosestPair(pts1, length / 2, a1, b1);              //分治求解左
半部分子集的最近点
        d2 = ClosestPair(pts2, length - length / 2, a2, b2);     //分治求解右
半部分子集的最近点

        if (d1 < d2) { distance = d1; a = a1; b = b1; }             //记录最
近点，最近距离
        else { distance = d2; a = a2; b = b2; }

        //merge - 进行子集合解合并
        //求解跨分割线并在δ×2δ区间内的最近点对
        Point *pts3 = new Point[length];
```

```
        for (i = 0, k = 0; i < length; i++)                    //取得中线2δ宽度的所有点对共k个
            if (abs(points[i].x - mid) <= distance)
                pts3[k++] = points[i];
        sort(pts3, pts3 + k, compareY);                        // 以y排序矩形阵内的点集合
        for (i = 0; i < k; i++)
        {
            if (pts3[j].x - mid >= 0)                          //只判断左侧部分的点
                continue;
            x = 0;
            for (j = i + 1; j <= i + 6 + x && j < k; j++)    //只需与有序的领接的的6个点进行比较
            {
                if (pts3[j].x - mid < 0)
                {//  假如i点是位于mid左边则只需判断在mid右边的j点即可
                    x++;
                    continue;
                }
                if (Distance(pts3[i], pts3[j]) < distance)
                {//如果跨分割线的两点距离小于已知最小距离，则记录该距离和两点
                    distance = Distance(pts3[i], pts3[j]);
                    a = pts3[i];
                    b = pts3[j];
                }
            }
        }
    }
    return distance;
}

int main()
{
    int num;             //随机生成的点对个数
    Point a, b;             //最近点对
    double diatance;     //点对距离
    cin >> num;
    Point *points = new Point[num];
    for (int i = 0; i < num; i++)
        cin>> points[i].x >> points[i].y ;
    diatance = ClosestPair(points, num, a, b);
    cout << endl << endl << "按横坐标排序后的点对:" << endl;
    for (int i = 0; i < num; i++)
        cout << "(" << points[i].x << "," << points[i].y << ")" << endl;

    cout << endl << "最近点对为：" << "(" << a.x << "," << a.y << ")和" << "(" << b.x << "," << b.y << ")" << endl << "最近点对距离为：" << diatance << endl;
```

```
}

#endif // !Closest_pair
```

7.3.5 第k小元素问题

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 7

void getData(int [], int);
void result_output(int []);

int selectmink1(int a[], int low, int high, int k);
int selectmink2(int a[], int low, int high, int k);
int split(int a[], int low, int high);

int main(void)
{
  int a[N], k, r;

  getData(a, N); /* 获得数据放入数组a中 */

  printf("datas: \n");
  result_output(a);

  scanf("%d", &k);
  if(k >= 0 && k <= N-1) {
      r = selectmink1(a, 0, N - 1, k);
      printf("result=%d\n", r);
      r = selectmink2(a, 0, N - 1, k);
      printf("result=%d\n", r);
  } else
      printf("input error: k=%d\n", k);

  return 0;
}

int selectmink1(int a[], int low, int high, int k)
{
  int middle;

  middle = split(a, low, high);
  if(middle == k)
      return a[k];
  else if(middle < k)
```

```
        return selectmink1(a, middle+1, high, k);
    else /* if(middle > k) */
        return selectmink1(a, low, middle-1, k);
}

int selectmink2(int a[], int low, int high, int k)
{
    int middle;

    for(;;) {
        middle = split(a, low, high);
        if(middle == k)
            return a[k];
        else if(middle < k)
            low = middle+1;
        else /* if(middle > k) */
            high = middle-1;
    }
}

int split(int a[], int low, int high)
{
    int part_element = a[low];

    for (;;) {
      while (low < high && part_element <= a[high])
        high--;
      if (low >= high) break;
      a[low++] = a[high];

      while (low < high && a[low] <= part_element)
        low++;
      if (low >= high) break;
      a[high--] = a[low];
    }

    a[high] = part_element;
    return high;
}

void getData(int d[], int n)
{
    time_t t;
    srand((unsigned) time(&t));  /* 设置随机数起始值 */

    int i;
    for(i=0; i < n; i++)
        d[i] = rand() % 100; /* 获得0-99之间的整数值 */
}
```

```
void result_output(int a[])
{
    int i;
    for (i = 0; i < N; i++)
      printf("%d ", a[i]);
    printf("\n");
}
```

7.3.6 循环赛日程表问题

```
#include<stdio.h>
#include<math.h>
#define N 50
void GameTable(int k,int array[][N]);
void print(int k,int array[][N]);          //输出二维数组
int main()
{
    int k;
    int array[N][N];
    printf("\t\t*****************************************\n");
    printf("\t\t**\t\t循环赛日程表           **\n");
    printf("\t\t*****************************************\n\n");
    printf("设参赛选手的人数为n（n=2^k），请输入k 的值：");
    do
    {
         scanf("%d",&k);
        if(k!=0)
        {
            GameTable(k,array);
            print(k,array);
        }
        else
          printf("您输入的数据有误,请重新输入");
    }while(k!=0);

}
void GameTable(int k,int array[][N])//数组下标从1开始
{
    int i,j,s,t;
    int n=1;
    for(i=1;i<=k;i++)
        n*=2;                          //求总人数
    for(i=1;i<=n;i++)
        array[1][i]=i;                    //第一行排1-8
    int m=1;                             //用来控制每一次填表时i行j列的起始填充位置
    for(s=1;s<=k;s++)                    //s指对称赋值的总循环次数，即分成几大步进行
制作日程表
```

```
    {
        n=n/2;
        for(t=1;t<=n;t++)                //t指明内部对称赋值的循环次数
            for(i=m+1;i<=2*m;i++)
                for(j=m+1;j<=2*m;j++)
                {
                    array[i][j+(t-1)*m*2]=array[i-m][j+(t-1)*m*2-m];
//右上角等于左上角的值
                    array[i][j+(t-1)*m*2-m]=array[i-m][j+(t-1)*m*2];
//左下角等于右上角的值
                }
        m*=2;
    }

}
void print(int k,int array[][N])
{
    int i,j;
    int num=pow(2,k);
    printf("%d人的循环赛日程表如下\n",num);
    for(i=1;i<=num;i++)                                  //输出二维数组
    {
        for(j=1;j<=num;j++)
        {
            printf("%d\t",array[i][j]);
        }
         printf("\n");
    }
}
```

7.3.7 找假币问题

```
#include <iostream>
#include <cstdlib>

using namespace std;

const int MAXNUM = 30;

int falseCoin(int weight[], int lhs, int rhs)
{
    if (lhs == rhs)
        return lhs + 1;
    //如果只剩下两个银币，则较轻的那个便是假币
    else if (lhs == (rhs - 1))
    {
        return weight[lhs] < weight[rhs] ? lhs + 1 : rhs + 1;
    }
```

```
    int lsum = 0, rsum = 0;

    //如果偶数个银币，则比较两等份
    if ((rhs - lhs + 1) % 2 == 0)
    {
        for (int i = lhs; i < (lhs + (rhs - lhs + 1) / 2); i++)
        {
            lsum += weight[i];
        }

        for (int j = lhs + (rhs - lhs + 1) / 2; j <= rhs; j++)
        {
            rsum += weight[j];
        }

        //左右两份等重，则无假币
        if (lsum == rsum)
            return -1;
        else
            return (lsum < rsum) ? falseCoin(weight, lhs, lhs + (rhs - lhs)
/ 2) : falseCoin(weight, lhs + (rhs - lhs) / 2 + 1, rhs);
    }

    //如果奇数个银币，则比较除中间银币外的两等份
    else if ((rhs - lhs + 1) % 2 != 0)
    {
        for (int i = lhs; i < (lhs + (rhs - lhs) / 2); i++)
        {
            lsum += weight[i];
        }

        for (int j = (lhs + (rhs - lhs) / 2 + 1); j <= rhs; j++)
        {
            rsum += weight[j];
        }

        //左右两份等重，则无假币
        if (lsum == rsum && weight[lhs] == weight[lhs + (rhs - lhs) / 2])
            return -1;

        //如果两份等重，中间银币较轻，则中间银币为假币
        else if (lsum == rsum && weight[lhs] > weight[lhs + (rhs - lhs) /
2])
            return lhs + (rhs - lhs) / 2 + 1;

        //否则，返回较轻那份中的假币
        else
```

```
            return (lsum < rsum) ? falseCoin(weight, lhs, lhs + (rhs - lhs)
/ 2 - 1) : falseCoin(weight, lhs + (rhs - lhs) / 2 + 1, rhs);
    }
}

int main()
{
    int weight[MAXNUM];

    int n;
    while (cin >> n)
    {
        for (int i = 0; i < n; i++)
            cin >> weight[i];

        int falsePos = falseCoin(weight, 0, n - 1);

        if (falsePos != -1)
            cout << "第" << falsePos << "个银币为假币! " << endl;
        else
            cout << "无假币!" << endl;

    }//while

    system("pause");
    return 0;
}
```

7.3.8 n阶分形

```
#include <iostream>
#include <cstring>
using namespace std;
const int MAXN=730;
const int N[8]={0,1,3,9,27,81,243,729};
int n,i,j,map[MAXN][MAXN];
void set(int n,int x,int y);
int main()
{
    while (cin>>n && n!=-1){
        memset(map,0,sizeof(map));
        set(n,1,1);
        for (i=1;i<=N[n];++i){
            for(j=1;j<=N[n];++j)
                if (map[i][j]) cout<<'X';
                    else cout<<' ';
            cout<<endl;
        }
```

```
        cout<<"-\n";
    }
    return 0;
}

void set(int n,int x,int y){
    if (n==1){
        map[x][y]=1;
        return;
    }
    set(n-1,x,y);
    set(n-1,x+2*N[n-1],y);
    set(n-1,x+N[n-1],y+N[n-1]);
    set(n-1,x,y+2*N[n-1]);
    set(n-1,x+2*N[n-1],y+2*N[n-1]);
    return;
}
```

7.3.9 m叉树问题

```
//#include "stdafx.h"
#include"stdio.h"
#include<iostream>
#include<string>
using namespace std;
typedef long long ll;
const int maxn=15;
int m;
ll c[21];
string s1,s2;
void init(){
   c[0]=1;
   for(int i=1;i<=20;i++)
       c[i]=c[i-1]*i;
}

ll cal(int n){
    return c[m]/(c[m-n]*c[n]);
}

ll func(string a,string b){//a[0]和b[b.length()-1]是一样的
    a=a.substr(1,a.length());
    b=b.substr(0,b.length()-1);
    if(a==b)return cal(a.length());//如果a和b相等，那么其字符一定是在同一层上
    int j=0;int n=0;
    ll ans=1;
    for(int i=0;j<a.length()&&i<b.length();i++){
            if(a[j]==b[i]){
```

```
                ans*=func(a.substr(j,i-j+1),b.substr(j,i-j+1));//分割成各个子
树
                j=i+1;n++;}//n统计子树的个数

        }
         ans*=cal(n);

     return ans;
}

int main(){
    init();
    while(cin>>m){
        if(m==0)break;
            cin>>s1>>s2;
            cout<<func(s1,s2)<<endl;
        }

    return 0;
}
```

7.3.10 电话查重

```
#include <iostream>
#include <stdio.h>
using namespace std;
#define MAXN 10000000
int a[MAXN+1]={0};
int Map(char c) //映射关系
{
    if('0'<=c && c<='9')
        return c-'0';
    else if(c=='A' || c=='B' || c=='C')
        return 2;
    else if(c=='D' || c=='E' || c=='F')
        return 3;
    else if(c=='G' || c=='H' || c=='I')
        return 4;
    else if(c=='J' || c=='K' || c=='L')
        return 5;
    else if(c=='M' || c=='N' || c=='O')
        return 6;
    else if(c=='P' || c=='R' || c=='S')
        return 7;
    else if(c=='T' || c=='U' || c=='V')
        return 8;
    else if(c=='W' || c=='X' || c=='Y')
        return 9;
```

```
    else
        return -1;
}
int main()
{
    char s[111],c;
    int i,j,n;
    scanf("%d%c",&n,&c);
    for(i=1;i<=n;i++){  //输入n个数
        scanf("%s",s);
        int tel = 0;
        for(j=0;s[j];j++){
            int t = Map(s[j]);
            if(t==-1) continue;
            tel = tel*10+t;
        }
        a[tel]++;   //次数加1
    }
    int f=false; //有无重复
    for(i=0;i<MAXN;i++) //输出
        if(a[i]>1){
            f=true;
            printf("%03d-%04d %d\n",i/10000,i%10000,a[i]);
        }
    if(!f)   cout<<"No duplicates."<<endl;
    return 0;
}
```

7.3.11 树的有效对

```
#include<iostream>
#include<map>
#include<string>
#include<cstring>
#include<vector>
#include<algorithm>
#include<set>
#include<sstream>
#include<cstdio>
#include<cmath>
#include<climits>
using namespace std;
const int maxn=1e4+7;
const int inf=0x3f3f3f3f;
typedef long long ll;
const int mod=1e9+7;
int n,k,allnode;
int head[maxn*2];
```

```
int num;
int dp[maxn];
int size[maxn];
int Focus,M;
ll dist[maxn];
int deep[maxn];
bool vis[maxn];
ll ans;
struct Edge
{
    int u,v,w,next;
}edge[maxn<<2];
void addEdge(int u,int v,int w)
{
    edge[num].u=u;
    edge[num].v=v;
    edge[num].w=w;
    edge[num].next=head[u];
    head[u]=num++;
}
void init()
{
    memset(head,-1,sizeof(head));
    memset(dist,0,sizeof(dist));
    memset(vis,0,sizeof(vis));
    num=0;
}
void getFocus(int u,int pre)
{
    size[u]=1;
    dp[u]=0;
    for(int i=head[u];i!=-1;i=edge[i].next)
    {
        int v=edge[i].v;
        if(v==pre||vis[v]) continue;
        getFocus(v,u);
        size[u]+=size[v];
        dp[u]=max(dp[u],size[v]);
    }
    dp[u]=max(dp[u],allnode-size[u]);
    if(M>dp[u])
    {
        M=dp[u];
        Focus=u;
    }
}
void dfs(int u,int pre)
{
    deep[++deep[0]]=dist[u];
```

```
    for(int i=head[u];i!=-1;i=edge[i].next)
    {
        int v=edge[i].v,w=edge[i].w;
        if(v==pre||vis[v]) continue;
        dist[v]=dist[u]+w;
        dfs(v,u);
    }
}
int cal(int x,int now)
{
    dist[x]=now,deep[0]=0;
    dfs(x,0);
    sort(deep+1,deep+deep[0]+1);
    int ans=0;
    for(int l=1,r=deep[0];l<r;)
    {
        if(deep[l]+deep[r]<=k)
        {
            ans+=r-l;
            l++;
        }
        else r--;
    }
    return ans;
}
void solve(int x)
{
    vis[x]=1;
    ans+=cal(x,0);
    for(int i=head[x];i!=-1;i=edge[i].next)
    {
        int v=edge[i].v;
        if(vis[v]) continue;
        ans-=cal(v,edge[i].w);
        allnode=size[v];
        Focus=0,M=1e9;
        getFocus(v,x);
        solve(Focus);
    }
}
int main()
{
    while(scanf("%d%d",&n,&k)!=EOF&&(n+k))
    {
        init();
        for(int i=1,u,v,w;i<n;i++)
        {
            scanf("%d%d%d",&u,&v,&w);
            addEdge(u,v,w);
```

```
            addEdge(v,u,w);
        }
        Focus=ans=0;
        allnode=n,M=1e9;
        getFocus(1,0);
        solve(Focus);
        printf("%lld\n",ans);
    }
    return 0;
}
```

7.3.12 回文串交换

```
#include<iostream>
using namespace std;

int main()
{

    string s;
    cin >> s;
    int n = s.length();
    int end = n - 1;     //字符串最后一个字符
    int cnt = 0;         //交换次数
    int oddNum = 0;         //判断是否已经有一个单独的奇个数的字符了

    for (int i = 0; i < end; i++)//从第一个字符到倒数第二个字符遍历
    {
        for (int j = end; j >= i; j--)//从最后一个开始，到第i个字符，寻找与s[i]相同的字符
        {
            if (i == j)        //如果没找到
            {
                if (n % 2 == 0 || oddNum == 1)  //不可能的两种情况
                {
                    cout << "Impossible";
                    return 0;
                }
                oddNum = 1;              //找到一个字符出现的次数为奇数
                cnt += n / 2 - i;     //将次字符交换到中间位置的次数
            }
            else if (s[i] == s[j])     //如果找到了，将s[j]交换到s[end]位置
            {
                for (int k = j; k < end; k++)     //交换相邻两个位置的字符
                {
                    swap(s[k], s[k+1]);
                    cnt++;
                }
```

```
                end--;                  //末尾递减
                break;                  //开始从i+1处重复操作
            }
        }
    }

    cout << cnt;

    return 0;
}
```

7.3.13 史密斯数

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
#include <queue>
#include <cmath>
using namespace std;

int n;

int cacl(int x)
{
    int sum = 0;
    while (x)
    {
        sum += x % 10;
        x /= 10;
    }
    return sum;
}

bool isprime(int x)
{
    int m = sqrt(x + 0.5);
    for (int i = 2; i <= m; i++)
    {
        if (x%i == 0)  return false;
    }
    return true;
}

int solve(int x)
{
    if (isprime(x))
        return cacl(x);
```

```
    else
    {
        int m = sqrt(x + 0.5);
        for (int i = m; i >1;i--)
        if (x%i == 0)
            return solve(i) + solve(x / i);
    }
}

int main()
{
    //freopen("D:\\input.txt", "r", stdin);
    while (~scanf("%d", &n) && n)
    {
        for (int i = n + 1;; i++)
        {
            if (!isprime(i) && cacl(i) == solve(i))
            {
                printf("%d\n", i);
                break;
            }
        }
    }
    return 0;
}
```

7.3.14 矩阵乘积

```
#include<cstdio>
#include<iostream>
using namespace std;
int main() {
    int a[2][3],b[3][2];
    int c[2][2];
    while(~scanf("%d%d%d",&a[0][0],&a[0][1],&a[0][2])) {
        for(int i=1; i<2; i++) {
            for(int j=0; j<3; j++) {
                scanf("%d",&a[i][j]);
            }
        }
        for(int i=0; i<3; i++) {
            for(int j=0; j<2; j++) {
                scanf("%d",&b[i][j]);
            }
        }
        for(int i=0; i<2; i++) {
            for(int j=0; j<2; j++) {
                int t=0;
```

```
                for(int k=0; k<3; k++) {
                    t+=a[i][k]*b[k][j];
                }
                c[i][j]=t;
            }
        }
        for(int i=0; i<2; i++) {
            printf("%d %d \n",c[i][0],c[i][1]);
        }
    }
    return 0;
}
```

7.3.15 士兵排队问题

依次求出y，x上的中位数

```
#include <iostream>
#include <cstring>
#include <cmath>
#include <ctime>
#include <cstdlib>
#define NUM 10001
using namespace std;

int com(const void *a,const void *b)
{
    return *((int*)a)-*((int*)b);//从小到大
}

int main() {
    int N,X[NUM],Y[NUM];//存放x和y坐标
    cin>>N;//表示士兵数
    for ( int i = 0; i<N;i++){
        cin>>X[i]>>Y[i];
    }//输入操作
        //士兵移动x坐标或者是y坐标移动一个单位，在poj上提交可以不用这个功能；
    for( int i = 0 ; i<N;i++){
        int dirction =0+ rand()%(int)(1-0+1);//移动的坐标
        int move = (-1)+rand()%(int)(1-(-1)+1);//移动方向

        if(dirction == 0){
            int x = X[i] + move;
            int j = 0;
            for(j = 0; j<N;j++){
                if(x!=X[j]&&Y[i]!=Y[j])//有相同的坐标就不需移动
                    continue ;
                else
```

```
                    break;
            }
            if(j == N)
                X[i] = x;
        }
            //移动x坐标
        else if(dirction == 1){
            int y = Y[i] + move;
            int j = 0;
            for(j = 0; j<N;j++){
                if(y!=Y[j] &&X[i]!=X[j])//有相同的坐标就不需移动
                    continue ;
                else
                    break;
            }
            if(j == N)
                Y[i] = y;
        }//移动y坐标

    }
    qsort(Y, N, sizeof(int), com);
    qsort(X, N, sizeof(int), com);
    for(int i = 0;i<N;i++){
        X[i] -=i;
    }
    qsort(X, N, sizeof(int), com);
    int sum  = 0;
    for(int i = 0; i<N;i++){//移动的最小步数
        sum +=abs(X[i]-X[N/2])+abs(Y[i]-Y[N/2]);
    }
    cout<<sum<< endl;
    return 0;
}
```

8.3.1 小船过河问题

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {
    int n;
    cin >> n;
    int a[n];
    for(int i =0; i < n ; i++) {
        cin >> a[i];
    }
    int sum = 0;
```

```
    //每四个人局部计算最优解
    while (n>3) {
        sum += min(a[1] + a[0] + a[n - 1] + a[1], a[n - 1] + a[0] + a[n -
2] + a[0]);
        n -= 2;
    }
    if (n==3) {
        sum += a[0] + a[1] + a[2];
    }
    else if (n == 2) {
        sum += a[1];
    }
    else {
        sum += a[0];
    }
    cout << sum;
    return 0;
}
```

8.3.2 纪念品分组

```
#include<iostream>
#include<algorithm>
using namespace std;

int main()
{
    int w,n,a[1001],i=0,j=0,sum=0;
    cin>>w;//每组纪念品价格的上限
    cin>>n;//一共多少件纪念品
    for(i;i<n;i++)
        cin>>a[i];
    sort(a,a+n);
    while(j<=n)
    {
        if(j == n)
        {
            sum=sum+1;
            break;
        }
        else
        {
            if(a[j]+a[n-1]<=w)
            {
                sum=sum+1;
                j++;
                n--;
            }
```

```
            else
            {
                sum=sum+1;
                n--;
            }
        }
    }
    cout<<sum<<endl<<endl;
    return 1;
}
```

8.3.3 数列极差问题

```
#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>
#define maxn 50005
using namespace std;
int amax[maxn],amin[maxn];
int cmp(int a,int b){
    return a>b;
}
int main()
{
    int n;
    while(~scanf("%d",&n))
    {
        if(n==0) break;
        for(int i=0;i<n;i++)
        {
            scanf("%d",&amax[i]);
            amin[i]=amax[i];
        }
        sort(amin,amin+n,cmp);
        sort(amax,amax+n);
        for(int i=1;i<n;i++)
        {
            amax[i]=amax[i]*amax[i-1]+1;
            sort(amax+i,amax+n);
        }
        for(int i=1;i<n;i++)
            amin[i]=amin[i]*amin[i-1]+1;
        printf("%d\n",amax[n-1]-amin[n-1]);
    }
}
```

8.3.4 函数求底问题

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

int main(void) {
    double n , p;
    while(cin >> n >> p) {
        double tmp = pow(p, 1 / n); // p开n次方
        int k = floor(tmp + 0.5);   // 四舍五入（+0.5后向下取整）
        cout << k << endl;
    }
    return 0;
}
```

8.3.5 开心的金明(背包问题)

```
#include<cstdio>//调用 scanf 和 printf 库
#include<cstring>//调用 memset 库，不过今次用不上
int n,m;//表示 总钱数和希望购买物品的个数
int v[30],q[30],f[30010];//先不说待活你就知道
int main()
{
    scanf("%d %d",&n,&m);
    for(int i=1;i<=m;i++)scanf("%d %d",&v[i],&q[i]);
    for(int i=1;i<=m;i++)//循环从1到m种 i++
    {
        for(int j=n;j>=v[i];j--)//双重循环 n到v[i]，注意是i--
        {
            int x=f[j-v[i]]+v[i]*q[i];
            if(x>f[j])f[j]=x;
        }
    }
    printf("%d",f[n]);//输出
    return 0;
}
```

8.3.6 小明坐车问题

```
#include<iostream>
#include<iomanip>
#include<cmath>
#include<algorithm>
```

```
#include<cstring>
#include<cstdio>
#include<cstdlib>
using namespace std;
int n,a[15],f[105];
int main()
{
    for(int i=1;i<=10;i++) scanf("%d",&a[i]);
    scanf("%d",&n);
    memset(f,0x3f,sizeof(f));
    f[0]=0;
    for(int i=1;i<=10;i++)
    for(int j=1;j<=i;j++)
    {
        f[i]=min(f[i],f[i-j]+a[j]);
    }
    for(int i=11;i<=n;i++)
    {
        for(int j=1;j<=10;j++)
        {
            f[i]=min(f[i],f[i-j]+a[j]);
        }
    }
    printf("%d",f[n]);
}
```

8.3.7 田忌赛马

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

bool cmp(int a,int b)
{
    return a>b;
}

int main()
{
    int n,i,tjfast,tjslow,kingfast,kingslow,win,lose;
    int a[1005],b[1005],sum;
    while(scanf("%d",&n)==1&&n!=0)
    {
        for(i=0;i<n;i++)
        {
            scanf("%d",&a[i]);
        }
        for(i=0;i<n;i++)
```

```
        {
            scanf("%d",&b[i]);
        }
        sort(a,a+n,cmp);  //从大到小排序
        sort(b,b+n,cmp);
        tjfast=0,tjslow=n-1;kingfast=0,kingslow=n-1;
        win=0,lose=0;
        for(i=0;i<n;i++)     //进行n轮比较
        {
            if(a[tjfast]>b[kingfast])   //最快的比较
            {
                win++;
                tjfast++;
                kingfast++;
            }
            else if(a[tjslow]>b[kingslow])  //最慢的比较
            {
                win++;
                tjslow--;
                kingslow--;
            }
            else
            {
                if(a[tjslow]<b[kingfast])   //田忌最慢的马与齐王最快的马比较
                {
                    lose++;
                    tjslow--;
                    kingfast++;
                }
            }
        }
        sum=(win-lose)*200;
        printf("%d\n",sum);
    }
    return 0;
}
```

8.3.8 装箱问题

```
#include<stdio.h>
int main()
{
    int a1, a2, a3, a4, a5, a6;
    int count;
    int b1, b2=0, b3;
    scanf("%d%d%d%d%d%d", &a1, &a2, &a3, &a4, &a5, &a6);
    count = a6 + a5 + a4;
    if (a3 % 4 == 0)
```

```
        count += (a3 / 4);
    else
        count += (a3 / 4) + 1;
    b3 = 4 - (a3 % 4);
    if (b3 == 1)
        b2 = a4 * 5 + 5;
    else if (b3 == 2)
        b2 = a4 * 5 + 3;
    else if (b3 == 3)
        b2 = a4 * 5 + 1;
    if (a2 <= b2)
    {
        b1 = a5 * 9 + (a4 * 20 + (4 - (a3 % 4)) * 9 - 4 * a2);
        if (a1 <= b1)
            ;
        else
            if ((a1 - b1) % 36 != 0)
                count += ((a1 - b1) / 36 + 1);
            else
                count += ((a1 - b1) / 36);
    }
    else
    {
        if ((a2 - b2) / 9 != 0)
            count += ((a2 - b2) / 9 + 1);
        else
            count += (a2 - b2) / 9;
        b1 = a5 * 9 + 4;
    }
    printf("%d", count);
    return 0;
}
```

8.3.9 删数问题

```
#include <stdio.h>
#include <string.h>
int main()
{
    char n[101];    //将输入的数字存储为字符串形式
    int s,len,i;    //s：删去的数字数    len：输入数字的长度    i:计数变量
    while(~scanf("%s %d",n,&s))    //获取原始数字和将要删除的数字
    {
    while(s>0)        //s=0时，删数结束。while循环执行逐个删数
    {
        i = 0;           //从i=0开始遍历，寻找第一个递减数列。
        len = strlen(n);       //储存n的位数
```

```
        while(i<len&&n[i]<=n[i+1])    //当i<len并且n[i]<=n[i+1]时，不构成递减数
列，i++
            i++;
        while(i<len)       //构成递减数列时，删除n[i],之后字符依次前移
        {
         n[i]=n[i+1];
         i++;
        }           //i=len-1时，字符迁移完毕。
        s--;            //继续删除数字
    }
//删除数字结束后，即s=0时
//有时会出现以下情况，删除过程中有可能会使原来包含零的数字串变成若干个以0开始的序列，如
80056，第一次删除8后，变成0056，高位的0是无效的，程序中应该处理掉。
        i=0;
        len = strlen(n);
        while(n[i]=='0'&&i<len)     //从i=0开始遍历，直到不为0的那一位
            i++;
        if(i==len)                  //如果i=len时还没有出现不为零的位，则该数为零。
            printf("0\n");
        else                         //找到不为0的那一位，当作首位，输出。
        {
            for(i=i;i<len;i++)
                printf("%c",n[i]);
            printf("\n");
        }

    }
    return 0;
 }
```

8.3.10 移动纸牌问题

```
#include <iostream>
using namespace std;
int main()
{
int a,p=0,js=0; cin >>a;int q[a];
for (int y=0;y<a;y++){cin >>q[y]; p+=q[y];} p/=a;
for (int y=0;y<a;y++)q[y]-=p;
for (int y=0;y<a;y++) {if (q[y]==0)continue; q[y+1]+=q[y]; js++; }
cout <<js;
return 0;
}
```

8.3.11 组合正整数

```
#include<iostream>
```

```cpp
#include<cmath>
#include<algorithm>
using namespace std;
struct num
{
    long long x,y;//x表示这个数字，y表示这个数字的长度
}v[25];
int comp(num a,num b)
{
    return a.x*pow(10,b.y)+b.x>b.x*pow(10,a.y)+a.x;//每两个数进行组合，找出最大的数字排列顺序
}
int main()
{
    int n,t,c=0;
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        cin>>v[i].x;
        t=v[i].x;
        c=0;
        while(t)//计算这个数字的长度
        {
            c++;
            t/=10;
        }
        v[i].y=c;
    }
    sort(v+1,v+1+n,comp);
    for(int i=1;i<=n;i++)
        cout<<v[i].x;
    return 0;
}
```

8.3.12 活动安排问题

```cpp
#include <iostream>
#include <stdlib.h>
#include <algorithm>
#include <math.h>
#include <string.h>
using namespace std;

struct action{
    int s;
    int f;
    int index;
};
```

```
bool cmp(const action &a, const action &b)
{
    if (a.f<=b.f) return true;
    return false;
}


void GreedySelector(int n, action a[], bool b[])
{
    b[1] = true;
    int preEnd = 1;
    for(int i=2; i<=n; i++)
        if (a[i].s>=a[preEnd].f)
        {
            b[i] = true;
            preEnd = i;
        }
}

int main()
{
    int n;
    while(cin>>n && n)
    {
        action a[1000];
        bool b[1000];
        memset(b,false,sizeof(b));
        for(int i=1; i<=n; i++)
        {
            cin>>a[i].s>>a[i].f;
            a[i].index = i;
        }
        sort(a, a+n+1, cmp);
        GreedySelector(n, a, b);
        for(int i=1; i<=n;i++)
            if(b[i]) cout<<a[i].index<<" ";
        cout<<endl;
    }
    return 0;
}
```

8.3.13 多人接水问题 1

```
#include <iostream>
#include <stdio.h>
#include <map>
#include <utility>
```

```
using namespace std;

int main(int argc, char const *argv[])
{
    int number, temp, i;
    multimap<int, int> waite;

    scanf("%d", &number);

    for(i = 1; i <= number; i++) {//根据接水时间自动排序
        scanf("%d", &temp);
        waite.insert(make_pair(temp, i));
    }

    double sum = 0.0;
    multimap<int, int>::iterator iter = waite.begin();

    i = number - 1;
    for (; iter != waite.end(); iter++) {
        //打印排队顺序
        if (i > 0)
            printf("%d ", iter->second);
        else
            printf("%d\n", iter->second);
        sum += (iter->first) * i;
        i--;
    }
    sum /= (double)number;

    printf("%.2lf\n", sum);

    return 0;
}
```

8.3.14 多人接水问题 2

```
#include<iostream>
#include<algorithm>
using namespace std;
const int maxn=1000;
int a[maxn];//接水的人
int b[maxn];//等待接水的时间
int main()
{
    int n,m,i,sum=0;
    cin>>n>>m;
    for(i=0;i<n;i++)
```

```
    cin>>a[i];
    sort(a,a+n);//接水的时间从小到大排序
    for(i=0;i<n;i++)
    b[i]=a[i];
    for(i=m;i<n;i++)
    b[i]=b[i-m]+a[i];//每个人花费的时间=等待的时间+接水的时间
    for(i=0;i<n;i++)
    sum+=b[i];//所有人的接水时间之和
    cout<<sum<<endl;
    return 0;
}
```

8.3.15 搬桌子问题

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main()
{
int i, j,T,S,B=0;
int move[200],max[200];
int N;
int form, to;
cin >> T;
S = T;
while (2) {
    for (i = 0; i < 200; i++) {
        move[i] = 0;
    }
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        scanf("%d%d", &form, &to);
        form = (form - 1) / 2;
        to = (to - 1) / 2;
        if (form > to) {
            int temp = form;
            form = to;
            to = temp;
        }
    for (j = (form - 1); j < to; j++) {
        move[j]++;
    }
}
max[B] = 0;
for (i = 0; i < 200; i++) {
    if (max[B] < move[i])
    max[B] = move[i];
}
```

```
B++;
if (T == 1)
    break;
T--;
}
for (i = 0; i < S; i++)
    cout << (max[i]*10 )<< endl;
return 0;
}
```

9.3.1排列问题(全排列)

```
#include<iostream>
using namespace std;
int x[100];
bool place(int k) {
    for(int i=1;i<k;i++) {
        if(x[k]==x[i])
            return false;
    }
    return true;
}
void queue(int n) {
    int k=1;
    while(k>=1) {
        x[k]++;
        while(x[k]<=n&&!place(k))
            x[k]++;
        if(x[k]<=n) {
            if(k==n) {
                for(int i=1;i<=n;i++)
                    cout<<x[i]<<' ';
                cout<<endl;
            }
            else if(k<n)
                k++;
        }
        else {
            x[k]=0;
            k--;
        }
    }
}
int main() {
    int n;
    while(cin>>n)
        queue(n);
    return 0;
```

```
}
```

9.3.2 低逐位整除数

```
#include<stdio.h>
int main()
{ int i,j,n,r,t,a[100]; long s=0;
  printf("  逐位整除n位，请确定n：");
  scanf("%d",&n);
  printf("  所求%d位最高位为3的右逐位整除数：\n",n);
  for(j=1;j<=100;j++) a[j]=1;
  t=0;a[1]=0;i=1;
  while(a[1]<1)
   { if(t==0 && i<n) i++;
     for(r=0,j=i;j>=1;j--)   // 检测i时是否整除i
       { r=r*10+a[j]; r=r%i; }
         if(r!=0)
           { a[i]=a[i]+1;t=1;    // 余数r!=0时a[i]增1,t=1
         while(a[i]>9 && i>1)
{ a[i]=1;i--;      // 回溯
  a[i]=a[i]+1;
}
       }
    else t=0;                 // 余数r=0时,t=0
if(t==0 && i==n)
       { if(a[n]==3)
{s++;printf("  %ld: ",s);
            for(j=n;j>=1;j--)
                printf("%d",a[j]);
            printf("\n");
}
             a[i]=a[i]+1;
       }
}
  if(s>0)
     printf("  最高位为3的%d位右逐位整除数共%ld个.",n,s);
  else
     printf("  未找到n位右逐位整除数.",n);

    return 0;
  }
```

9.3.3 子集问题

```
#include<iostream>
using namespace std;
int map[]={1,1,2,3};  //一个有序的数组
```

```
int trem[100]={0};  //统计每个元素出现此数的数组  数组下标代表数据 元素代表出现的次数
int v=4;            //元素个数
int path[100];      //辅助数组
int m=0;
void dfs(int k){
    if(k==v){
    for(int i=0;i<m;i++)
        cout<<path[i]<<" ";
    cout<<endl;

    }
    else{

        dfs(k+trem[map[k]]);  //不选  这个数开始递归

        for(int i=1;i<=trem[map[k]];i++){  //选1个到n个
            path[m++]=map[k];
            dfs(k+trem[map[k]]);
        }

        m=m-trem[map[k]];   //回溯
    }
}
void init(){   //初始化trem数组
    for(int i=0;i<v;i++)
        trem[map[i]]++;
}
int main(){
    init();
    dfs(0);
    return 0;
}
```

9.3.4 旅行商问题

```
# include <iostream>
using namespace std;
const int NoEdge=-1;
const int MAX=20;

int G[MAX][MAX];
int ans[MAX],x[MAX];
int bestc,cc;

void init(int n) {
 int i,j,len;
 memset(G,NoEdge,sizeof(G));
 while( cin>>i>>j ) {
```

```
  if( i==0 && j==0 ) break;
  cin>>len;
  G[i][j]=len;
  G[j][i]=len;
 }
 for(i=1;i<=n;i++)  x[i]=i;
 bestc=0x7fffffff;
 cc=0;
}

void Swap( int& i,int& j) {
 int t=i;
 i=j;
 j=t;
}

void Traveling(int i,int n){
 int j;
 if(i==n+1) {
  if(G[ x[n-1] ][ x[n] ] != NoEdge && G[ x[n] ][1] != NoEdge &&
  (cc + G[ x[n] ][1] < bestc ) ) {
   for(j=1;j<=n;j++)  ans[j] = x[j];
      bestc = cc + G[ x[n] ][ 1 ];
  }
 }
 else {
  for(j=i;j<=n;j++) {
   if( G[ x[i-1] ][ x[j] ] != NoEdge && (cc + G[ x[i-1] ][ x[j] ] < bestc)
) {
    Swap( x[i],x[j] );
    cc += G[ x[i-1] ][ x[i] ];
    Traveling(i+1,n);
    cc -= G[ x[i-1] ][ x[i] ];
    Swap( x[i],x[j] );
   }
  }
 }
}

void print(int n) {
 cout<<"最小的旅行费用为：<<bestc<<endl;
 cout<<"最佳路径是：";
 for(int i=1;i<=n;i++)
  cout<<ans[i]<<"->";
 cout<<ans[1]<<endl;
}

int main() {
 int n;
```

```
 while( cin>>n&&n ) {
  init(n);
  Traveling(2,n);
  print(n);
 }
 return 1;
}
```

9.3.5 两组均分问题

转化为“求子集问题”即可

9.3.6 组合数问题

```
#include <iostream>
#include <stack>
using namespace std;
int jiechen(int n) {
    int mul = 1;
    while(n != 0) {
        mul *= n;
        n--;
    }
    return mul;
}
int main()
{
    int n,m;
    cin>>n >>m;
    cout <<jiechen(n) / (jiechen(m) * jiechen(n-m));
    return 0;
}
```

9.3.7 运动员最佳配对问题

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

#define N 100
int n;
int P[N][N],Q[N][N];
int x[N];
int opt[N];
int tempValue=0,maxValue=0;
```

```
void compute(){
    tempValue = 0;
    for(int i=1;i<=n;i++){
        tempValue += P[i][x[i]]*Q[x[i]][i];
    }
    if(tempValue>maxValue){
        maxValue = tempValue;
        for(int i=1;i<=n;i++){
            opt[i] = x[i];
        }
    }
}

void traceback(int t){
    int i,j,temp;
    if(t>n){
        compute();
    }
    for(i=t;i<=n;i++){
        temp = x[i];
        x[i] = x[t];
        x[t] = temp;
        traceback(t+1);
        temp = x[i];
        x[i] = x[t];
        x[t] = temp;
    }
}
int main(){
    scanf("%d",&n);

    for(int i=1;i<=n;i++){
        x[i] = i;
    }

    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            scanf("%d",&P[i][j]);
        }
    }
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            scanf("%d",&Q[i][j]);
        }
    }
    traceback(1);
    printf("%d\n",maxValue);
    for(int i=1;i<=n;i++){
        printf("%d------%d\n",i,opt[i]);
```

```
    }
    return 0;
}
```

9.3.8 任务最佳调度问题

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
using namespace std;
int a[30],s[30];
int n,k,minn=0x7ffff;
void search(int,int);
int main() {
    scanf("%d%d",&n,&k);
    for(int i=1; i<=n; i++) //输入每个时间
        scanf("%d",&a[i]);
    search(1,0);//加入第一个时间，现在花费的时间是0；
    cout<<minn;
    return 0;
}
void search(int x,int y) {
    if(y>=minn)return;//剪枝 已经比最小的大了就不用再放了，舍弃这种方法
    if(x>n) { //放完了进行判断
        if(y<minn)minn=y;
        return;
    }
    for(int i=1; i<=k; i++) {
        if(s[i]+a[x]<=minn) { //剪枝
            s[i]+=a[x];
            search(x+1,max(y,s[i]));
            s[i]-=a[x];//回溯
        }
    }
    return;
}
```

9.3.9 迷宫问题

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
int dir[4][2] = {1,0,-1,0,0,1,0,-1};
char map[1000][1000];//地图
int vis[1000][1000];//用来标记是否走过
```

```
int sx,sy,px,py,t,n,m,flag,f;
struct step{
    int x,y,s;
};

void bfs(){
    queue<step> q;
    step p;
    p.x = sx;
    p.y = sy;
    p.s = 0;
    vis[sx][sy] = 1;
    q.push(p);//起点入队

    while(!q.empty()){//起点入队后开始循环
        step p = q.front();

        if(p.x==px && p.y==py){
            flag =1;
            if(p.s<=t){
                f=1;
                printf("YES\n");
                break;
            }
        }

        step v;
        for(int i=0 ;i<4 ;i++){
            v.x = p.x + dir[i][0];
            v.y = p.y + dir[i][1];
            if(map[v.x][v.y]=='*'){//路不通(有墙壁)
                continue;
            }
            if(v.x>=0 && v.x<n && v.y>=0 && v.y<m && vis[v.x][v.y]==0){
                v.s = p.s + 1;
                q.push(v);
                vis[v.x][v.y] = 1;
            }
        }
        q.pop();
    }
    if(flag==0||f==0){ //1.flag==0说明无法到达2.flag! =0, f==0, 说明可以到达但是
时间不够
        printf("NO\n");
    }
}

int main(){
    while(cin>>m>>n>>t){
```

```
        if(n==0&&m==0&&t==0){
            break;
        }
        memset(map,0,sizeof(map));
        memset(vis,0,sizeof(vis));
        for(int i=0 ;i<n ;i++){
            for(int j=0 ;j<m ;j++){
                cin>>map[i][j];
                if(map[i][j]=='S'){
                    sx = i;
                    sy = j;
                }
                if(map[i][j]=='P'){
                    px = i;
                    py = j;
                }
            }
        }
        flag = 0;
        f = 0;
        bfs();
    }
}
```

9.3.10 背包问题

```
#include <stdio.h>
#include <iostream>
using namespace std;
int bag[1000];
int dongtaiguihua(int s,int n)
{
    if(s==0)
    {
        return 1;
    }
    else if(s<0 || (s>0 && n==0))
    {
        return 0;//东西还没装完,已经没有剩余物件了;
    }
    else if(dongtaiguihua(s-bag[n],n-1))
    {
        return 1;
    }
    else
    {
        return dongtaiguihua(s,n-1);
    }
```

```
}
int main()
{
    int s,n;
    while(cin>>s>>n)
    {
        for(int i=1;i<=n;i++)
            cin>>bag[i];
        if(dongtaiguihua(s,n)==0)
        {
            cout<<"NO";
        }
        else
        {
            cout<<"YES";
        }
        cout<<endl;
    }
    return 0;
}
```

9.3.11 翻币问题

```
#include<iostream>
#include<cstdio>
using namespace std;
const int maxn=200;
int a[maxn*maxn]={0},s[maxn*maxn][2];
int f[maxn*maxn];
int n,len=0;

bool aka(int i,int h)
{
    if(i>s[h][0]) return false;
    if(5-i>s[h][1]) return false;
    if(a[s[h][0]-i+5-i]==1) return false;
    return true;
}
void ooo(int m)
{
    if(m==1) return;
    len++;
    ooo(f[m]);
}
void lil()
{
    int h=0,t=1;
    f[1]=0; a[n]=1;
```

```
    s[1][0]=n; s[1][1]=0;
    do
    {
        h++;
        for(int i=0;i<=5;i++)
        {
            if(aka(i,h))
            {
                t++;
                a[s[h][0]-i+5-i]=1;
                s[t][0]=s[h][0]-i;
                s[t][1]=s[h][1]+i;
                s[t][1]=s[t][1]-(5-i);
                s[t][0]=s[t][0]+5-i;
                f[t]=h;
            }
            if(a[0]==1)
            {
                ooo(t);
                printf("%d",len);
                return;
            }
        }
    }while(h<=t);
}
int main()
{
    scanf("%d",&n);
    lil();
    return 0;
}
```

9.3.12 最长滑雪问题

```
#include<iostream>
using namespace std;
const int Max = 105;

int row, col;
int map[Max][Max];                              //  记录图各点的高度。
int dp[Max][Max];                               //  记录以各点为起点的最长下降路径
的长度。

int dfs(int r, int c)
{
    if(dp[r][c] != 0)
        return dp[r][c];       //  若dp[r][c]不为0，则表示它已被访问。
    int max = 1;
```

```
    if(r + 1 <= row && map[r][c] > map[r + 1][c])
    {
        int len = dfs(r + 1, c) + 1;
        if(len > max)
            max = len;
    }
    if(r - 1 >= 1 && map[r][c] > map[r - 1][c])
    {
        int len = dfs(r - 1, c) + 1;
        if(len > max)
            max = len;
    }
    if(c + 1 <= col && map[r][c] > map[r][c + 1])
    {
        int len = dfs(r, c + 1) + 1;
        if(len > max)
            max = len;
    }
    if(c - 1 >= 1 && map[r][c] > map[r][c - 1])
    {
        int len = dfs(r, c - 1) + 1;
        if(len > max)
            max = len;
    }
    return map[r][c] = max;
}

int main()
{
    int i, j;
    cin >> row >> col;
    for(i = 1; i <= row; i ++)
        for(j = 1; j <= col; j ++)
            cin >> map[i][j];
        int ans = 0;
        memset(dp, 0, sizeof(dp));
        for(i = 1; i <= row; i ++)
            for(j = 1; j <= col; j ++)
            {
                dp[i][j] = dfs(i, j); //用记忆化搜索求出dp[i][j]，同时也求出其路
径上的dp[x][y]。
                if(dp[i][j] > ans)
                    ans = dp[i][j];
            }
            cout << ans << endl;
            return 0;
}
```

9.3.13 流水线作业调度问题

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int m1,m2;
};
struct node time[2][20000];
int cmp1(const void *p1,const void *p2)
{
    return ( (*(struct node *)p1).m1>(*(struct node *)p2).m1? 1:-1);
}
int cmp2(const void *p1,const void *p2)
{
    return ( (*(struct node *)p1).m2>(*(struct node *)p2).m2? -1:1 );
}
int main()
{
    int n,i,a,b,q,p,time1,time2;
    while(scanf("%d",&n)!=EOF&&n)
    {
        q=p=0;
        time1=time2=0;
        for(i=1;i<=n;i++)
        {
            scanf("%d%d",&a,&b);
            if(b>=a)
                {
                    time[0][q].m1=a;
                    time[0][q++].m2=b;
                }
                else{
                    time[1][p].m1=a;
                    time[1][p++].m2=b;
                }
        }
        qsort(time[0],q,sizeof(time[0][0]),cmp1);//m1  降序
        qsort(time[1],p,sizeof(time[1][0]),cmp2);//m2 升序
        for(i=0;i<q;i++)
        {
           time1+=time[0][i].m1;
           if(time2<time1)
                time2=time1;
           time2+=time[0][i].m2;
        }
        for(i=0;i<p;i++)
        {
```

```
            time1+=time[1][i].m1;
            if(time2<time1)
                time2=time1;
            time2+=time[1][i].m2;
        }
        printf("%d\n",time2);
    }
    return 0;
}
```

9.3.14 组合三角形问题

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

int n,count=0,half,sum=0;
int **a;


void backtrack(int t)
{
    if(count>half || (t*(t+1)/2-count>half)) return;
    int i,j;
    if(t==n){
        sum++;
        for(i=0;i<n;i++)
        {
            for(j=0;j<n-i;j++)
            {
                if(a[i][j])
                    printf("-");
                else
                    printf("+");
            }
            printf("\n");
        }
        printf("======================\n");
    }
    else{
        for(i=0;i<2;i++){
            a[0][t]=i;
            count=count+i;

            for(j=1;j<=t;j++){
                a[j][t-j]=a[j-1][t-j]^a[j-1][t-j+1];
                count+=a[j][t-j];
            }
```

```
            backtrack(t+1);

            for(j=1;j<=t;j++){
                count-=a[j][t-j];
            }
            count=count-i;
        }
    }
}


int main()
{
    int i,j,k;
    scanf("%d",&n);
    if((n*(n+1)/2)%2==1)
    {
        printf("result is 0.\n");
        return 0;
    }

    half=n*(n+1)/4;
    a=new int*[n];

    for(i=0;i<n;i++)
        a[i]=new int [n];

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            a[i][j]=0;
    }

    backtrack(0);

    printf("the result is %d.\n",sum);

    return 0;
}
```

9.3.15 情侣排队问题

```
无
```

10.3.1 自然数倒数求和

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

```
double fun1(int n)
{
int i;
double sum = 0;
for (i = 1; i <= n; i++)
sum += 1.0/i;
return sum;
}
int main(void)
{
int n;
printf("请输入一个整数: ");
scanf("%d",&n);
printf("前%d个自然数的倒数之和 = %lf\n",n,fun1(n));
}
```

10.3.2 今夕是何日

```
#include <stdio.h>
int main() {
    int a[12]={31,28,31,30,31,30,31,31,30,31,30,31};
    int years,months,days,i,j=0,flag = 1;//j为累加容器，flag为标记//
    printf("请输入年月日来判断\n");
    scanf("%d%d%d",&years,&months,&days);
    if(years%4==0&&years%100!=0||years%400==0) {//判断是否为闰年
        a[1] = 29;
    }
    if(months>12||days>31) {//如遇到输入错误，则把标记便为假，就屏蔽了天数的输出//
        flag = 0;
    }
    for(i=0;i<months-1;i++) {
        j+=a[i];
    }
    if(flag) {
        printf("这是第%d天",j+days);
    }
    else {
        printf("FUCK YOU");
    }
}
```

10.3.3 计算e值

```
#include<stdio.h>
int main(){
int i=1,j=1;
float e=1.0,k;
do{
j=i*j;
k=1.0/j;
e=e+k;
i++;
}while(k>1e-4);//判断误差是否小于给定的误差限E＝0.0001
printf("%f\n",e);
return 0;
}
```

10.3.4 自数

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<string>
#include<cmath>
#include<ctime>
#include<cctype>
#include<iomanip>
#include<algorithm>
#include<queue>
#include<stack>
#include<map>
#include<set>
#include<vector>
using namespace std;
int s[5005],n,k,num,a[10000005];
bool flag[10000005];
inline void scan(int &v)//读入优化
{
v=0;char c=0;int p=1;
while(c<'0'||c>'9'){if(c=='-')p=-1;c=getchar();}
while(c>='0'&& c<='9'){v=(v<<3)+(v<<1)+c-'0';c=getchar();}
v*=p;
}
void d(int x)
{
int ans=x;
while(x)
ans+=x%10,x/=10;
flag[ans]=true;
```

```
}
int main()
{
scanf("%d%d",&n,&k);
for(int i=1;i<=k;i++)scan(s[i]);
for(int i=1;i<=n;i++)
{
if(!flag[i])//没被筛
num++,a[num]=i;
d(i);//筛
}
cout<<num<<endl;
for(int i=1;i<=k;i++)
printf("%d ",a[s[i]]);
return 0;
}
```

10.3.5 火星人(下一个排列)

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{
    int n,m,k;
    vector<int> vec;
    cin>>n>>m;
    for(int i=0;i<n;++i)
    {
        cin>>k;
        vec.push_back(k);
    }
    for(int i=0;i<m;++i)
    {
        next_permutation(vec.begin(),vec.end());
    }
    for(int i=0;i<n;++i)
    {
        cout<<vec[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

10.3.6 整数平方后9位

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    int n;
    scanf("%d", &n);
    if(n<9) printf("0\n");
    else if(n == 9) printf("8\n");
    else
    {
        printf("72");
        for(int i=11; i<=n; i++)
        {
            printf("0");
        }
        printf("\n");
    }
    return 0;
}
```

10.3.7 构造等式(poj 1950)

```
#include <iostream>
#include <cstdio>

using namespace std;

char ch[100];
int n,sum;

int getNumLen(int i){
    int k = i;
    while(ch[i] == '.' && i < n) i++;
    return i-k+1;
}

void isRight(){
    int i = 1;
    int ans = 1;
    while(i < n){
        if (ch[i] == '.'){
```

```
            while(ch[i] == '.' && i < n){
                i++;
                ans*=10;
                if (i > 9) ans*= 10;
                ans+=i;
            }
            continue;
        }
        if (ch[i] == '+'){
            i++;
            int p = i;
            while(ch[i] == '.' && i < n){
                i++;
                p*=10;
                if (i > 9) p*= 10;
                p+=i;
            }
            ans += p;
            continue;
        }

        if (ch[i] == '-'){
            i++;
            int p = i;
            while(ch[i] == '.' && i < n){
                i++;
                p*=10;
                if (i > 9) p*= 10;
                p+=i;
            }
            ans -= p;
            continue;
        }

    }
    if (ans == 0) {
        if (sum < 20){
        for (int i = 1; i < n; i++)
            cout << i << " " << ch[i] << " ";
        cout << n << endl;
    }
        sum++;
    }
}

void dfs(int x){
    if (x == n) {
        isRight();
        return ;
```

```
    }

    ch[x] = '+';
    dfs(x+1);
    ch[x] = '-';
    dfs(x+1);
    ch[x] = '.';
    dfs(x+1);
}

int main(){
    scanf("%d", &n);
    sum = 0;
    dfs(1);
    printf("%d\n", sum);
}
```

10.3.8 构造回文字符串

将原串s反转，形成一个新串s1，然后求取s和s1的最长公共子序列就是原串中已经存在的回文串的长度。

```
#include<iostream>
using namespace std;
int dp[1100][1100];

int main()
{
    int t;
    string s,s1;
    cin>>t;
    while(t--)
    {
        memset(dp,0,sizeof(dp));
        s1.clear();
        cin>>s;
        int n=s.length();
        for(int i=n-1; i>=0; i--)
            s1+=s[i];
        for(int i=0; i<n; i++)
            for(int j=0; j<n; j++)
            {
                if(s[i]==s1[j])
                    dp[i+1][j+1]=dp[i][j]+1;
                else
                    dp[i+1][j+1]=max(dp[i][j+1],dp[i+1][j]);
            }
        cout<<n-dp[n][n]<<endl;
```

```
    }
    return 0;
}
```

10.3.9 开灯问题

```
#include<stdio.h>
int main() {
    int n,k;
    static int data[100];
    scanf("%d %d",&n,&k);
    for(int i = 1;i <= n;i++)//灯的数量
    {
        for(int j = 1;j <= k; j++)//人的个数
        {
            //灯是人的倍数时，改变数组的值。
            if(i % j == 0){
                if(data[i] == 0)
                    data[i] = 1;
                else data[i] = 0;
            }
        }
    }
    for(int i = 1; i <= n; i++)
        if(data[i] == 1)
        printf("%d ",i);
}
```

10.3.10 “1”的个数

```
#include <iostream>
using namespace std;
int numberOf1(int n)
{
    int count=0;
    int tag=1;
    while(tag)
    {
        if(tag&n)
            count++;
        tag=tag<<1;
    }
    return count;
}

int main()
{
```

```
    int i;
    while(cin>>i)
    {
        cout<<numberOf1(i)<<endl;
    }
    return 0;
}
```

10.3.11 小明的烦恼

```
#include<stdio.h>
int main()
{
    int n;
    char s[11],i;
    scanf("%d",&n);
    while(n--)
    {
        scanf("%s",&s);
        for(i=0; i<11; i++)
        {
            if(s[i]=='a'||s[i]=='b'||s[i]=='c')
                printf("2");
            if(s[i]=='d'||s[i]=='e'||s[i]=='f')
                printf("3");
            if(s[i]=='g'||s[i]=='i'||s[i]=='h')
                printf("4");
            if(s[i]=='j'||s[i]=='k'||s[i]=='l')
                printf("5");
            if(s[i]=='m'||s[i]=='n'||s[i]=='o')
                printf("6");
            if(s[i]=='p'||s[i]=='q'||s[i]=='r'||s[i]=='s')
                printf("7");
            if(s[i]=='t'||s[i]=='u'||s[i]=='v')
                printf("8");
            if(s[i]=='w'||s[i]=='x'||s[i]=='y'||s[i]=='z')
                printf("9");
        }
        printf("\n");
    }
    return 0;
}
```

10.3.12 乒乓球赛

```
2^n - k?
```

10.3.13 自然数拆分问题

```
#include<stdio.h>
#define N 1001
int rec[N];
//start:当前拆分出的数
//pos：保存到rec数组的位置记录
//left_num：上一轮拆分后，还剩下的数
void print(int n){
    int i;
    printf("3=");
    for( i=0;i<n-1;i++){
        printf("%d+",rec[i]);

    }
    printf("%d\n",rec[n-1]);
    return;
}
void dfs(int start,int pos,int left_num){
    if(left_num==0){
        print(pos);
        return;
    }
    int i;
    for(i=start;i<=left_num;i++){
        rec[pos]=i;
        dfs(i,pos+1,left_num-i);
    }
    return;
}
int main(void) {
    int n;
    scanf("%d", &n);
    dfs(1, 0, n);

    return 0;
}
```

10.3.14 集卡片赢大奖

收集一种卡片的概率为1，然后再买一袋即可收集2种的概率(n-1)/n, 所以期望为n/(n-1)依次类推，得到所有的期望为: f[n_] := Sum[n/k, {k, 1, n}] 上式可以优化，利用高数学的基数，Ln(n)=(1+1/2+1/3+....+1/n),可以精简为 n*(Ln(n)+0.5772156649) 其中常数为欧拉常数

```c
#include<stdio.h>
#include<math.h>
int main()
{
    double ans;
    int n;
    while(~scanf("%d",&n))
    {
        if(n<10000)
        {
            ans=0.0;
            for(int i=1;i<=n;i++)
                ans+=1.0*n/i;
            printf("%lld\n",(long long)(ans+0.5));
        }
    else{
            printf("%lld\n",(long long)(n*(log(n+1)+0.5772156649)));
        }
    }
    return 0;
}
```

10.3.15 括号匹配问题

```c
#include<iostream>
#include<algorithm>
#include<memory.h>
#include<string.h>
using std::cin;
using std::cout;
using std::endl;
using std::sort;

#define N 100
struct Stack {
    char date[N];
    int top;
} stack;
//栈初始化
void init(Stack *stack) {
    stack->top=-1;
    memset(stack->date,0,sizeof(char)*N);//数据清零
}

bool isempty(Stack*stack) { //判断是否为空
return stack->top==-1;
}
bool isfull(Stack*stack)//判断栈溢出
```

```
{
return stack->top==N-1;
}
int gettop(Stack*stack)//获取栈顶
{
return stack->date[stack->top];
}
void push(Stack*stack,char a)//压栈
{
    if(isfull(stack)){
        return;
    }else{
        stack->top++;
        stack->date[stack->top]=a;
    }

}
void pop(Stack*stack)//吐
{
if(isempty(stack)){
    return;
}else{
    stack->top--;
}
}
bool isleft(char a){
    return (a=='(')||(a=='[');
}
bool isright(char a){
    return (a==']')||(a==')');
}
bool juge(char a,char b){
    return (a=='('&&b==')')||(a=='['&&b==']');
}
int main(){
    Stack stack;
    init(&stack);
    char str[100];
    cin>>str;
    int len=strlen(str);
    for(int i=0;i<len;i++){
        if(isleft(str[i])){
            push(&stack,str[i]);
        }else {
        if(isright(str[i])){
        if(!juge(stack.date[stack.top],str[i])){
            cout<<"NO";
            return 0;
        }
```

```
            pop(&stack);
            }
            }
        }
        cout<<"YES";
        return 0;
}
```

11.3.1 最少硬币问题

```
#include <iostream>
using namespace std;
#define LL long long int
#define INF 0x3f3f3f3f
int dp[20020];
int main(){
    int i, j, k, n, m;
    cin >> n;

    int coins[n];  //硬币个数
    int T[n];      //硬币面值

    for(i = 0; i<n; i++)
        cin >> T[i] >> coins[i];
    cin >> m;
    for(i=1;i<=m;i++) dp[i]=INF;     //赋极大值,表示不可达
    for(i=0;i<n;i++)     //遍历硬币种类
        for(j=1;j<=coins[i];j++)     //遍历硬币数量
            for(k=m; k>=T[i]; k--)  //此处较难理解
                                    //只能是由m到T[i]而不能相反
                                    //试想,初始态dp[k-T[i]]应当=INF,dp[0]=0
                                    //如果能组成m元的硬币,那么应当存在一条0->m的
路径,此时
                                    //dp[m]就是需要的硬币数量
                                    //否则,dp[m]将不能链接到dp[0],从而
dp[m]=INF输出-1
                dp[k] = min(dp[k], dp[k-T[i]]+1);

    cout << (dp[m]<m?dp[m]:-1) << endl;
    return 0;
}
```

11.3.2 编辑距离问题

```
#include<iostream>
using namespace std;
```

```
const int maxn = 100;
string a, b;
int len_a, len_b;
int dp[maxn][maxn];          //dp[i][j]表示长度为i的字符串编辑为长度为j的字符串所需的最短编辑距离

int Min(int x, int y, int z)
{
    int temp = min(x, y);
    return min(temp, z);
}

int DP()
{
    //先处理边界情况
    for(int i = 0;i <= len_a;i++)
    {
        dp[i][0] = i;
    }
    for(int j = 0;j <= len_b;j++)
    {
        dp[0][j] = j;
    }

    //再来常规情况
    for(int i = 1;i <= len_a;i++)
    {
        for(int j = 1;j <= len_b;j++)
        {
            if(a[i - 1] == b[j - 1])         //因为string中是从0位开始存的，所以这里减个一
                dp[i][j] = dp[i - 1][j - 1];
            else
            {
                //dp[i - 1][j - 1]是修改a[i - 1]那个字符，使两个相等
                //dp[i - 1][j]是删掉a[i - 1]那个字符
                //dp[i][j - 1]是在a[i - 1]后还加一个字-符
                dp[i][j] = Min(dp[i - 1][j - 1], dp[i - 1][j], dp[i][j - 1]) + 1;
            }
        }
    }
    //打印dp数组
    cout << "dp数组: " << endl;
    for(int i = 0;i <= len_a;i++)
    {
        for(int j = 0;j <= len_b;j++)
        {
            cout << dp[i][j] << " ";
```

```
        }
        cout << endl;
    }
    return dp[len_a][len_b];
}

int main()
{
    cin >> a >> b;
    len_a = a.length();
    len_b = b.length();
    cout << DP() << endl;
    return 0;
}
```

11.3.3 石子合并问题

```
#include<iostream>
#include <cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
#define INF 0x3f3f3f3f

int dp[105][105];
int sum[110];
int ans=0,cnt=0;

int dfs(int l,int r){/*求最大*/
    if(dp[l][r]!=-1){
        return dp[l][r];
    }
    if(r-l==1)return sum[r]-sum[l-1];
    if(r==l) return 0;
    int &res=dp[l][r];
    res=-INF;
    for(int k=l;k<r;k++){
        res=max(res,dfs(l,k)+dfs(k+1,r));
    }
    res+=sum[r]-sum[l-1];
    return res;
}

int dfs2(int l,int r){/*求最小*/
   if(dp[l][r]!=-1) return dp[l][r];
   if(r-l==1) return sum[r]-sum[l-1];
   if(r==l) return 0;
```

```
    dp[l][r]=INF;
    for( int k=l; k<r; k++ ){
        dp[l][r]=min(dp[l][r],dfs2(l,k)+dfs2(k+1,r));
    }
    dp[l][r]+=sum[r]-sum[l-1];
    return dp[l][r];
}

int main(){
    int n;
    memset(sum,0,sizeof(sum));
    while(~scanf("%d",&n)&&n){
        for( int i=1; i<=n; i++ ){
            int t;
            cin>>t;
            sum[i]=sum[i-1]+t;
        }
        memset(dp,-1,sizeof(dp));
        ans=dfs2(1,n);
        memset(dp,-1,sizeof(dp));
        cnt=dfs(1,n);
        cout<<ans<<" "<<cnt<<endl;
    }
    return 0;
}
```

11.3.4 最小m段和问题

```
#include <stdio.h>
#define MAX(a,b) a>b?a:b

int a[100];
int dp[1000][1000];

int main()
{
    int n,m,maxvalue=0;
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(int i=1;i<=n;i++)
    {
        dp[i][1]=dp[i-1][1]+a[i];
    }
    for(int i=1;i<=n;i++)
    {
```

```
        for(int j=2;j<=m;j++)
        {
            int minvalue=99999;
            for(int k=1;k<i;k++){
                int temp=MAX(dp[i][1]-dp[k][1],dp[k][j-1]);
                if(temp<minvalue)
                {
                    minvalue=temp;
                }
            }
            dp[i][j]=minvalue;
        }
     }
     printf("%d\n",dp[n][m]);

    return 0;
}
```

11.3.5 最大长方体问题

```
#include "iostream"
using namespace std;
const int maxn=20;//定义常量指定数组的大小
/*动态规划:
d[0]=c[0];
当i>0时,d[i]=max{d[i-1]+c[i],c[i]},
其中d[i]表示数组c前(i+1)个元素的最大子段和*/
int maxSum1D(int c[],int p)//求一维数组的最大子段和
{
        int sum=0,d=0;
        for(int i=0;i<p;i++)
        {
                if(d>0)
                        d=d+c[i];
                else
                        d=c[i];
                if(sum<d)
                         sum=d;
        }
        return sum;
}
/*求出最大一面子墙的和*/
int maxSum2D(int b[][maxn],int n,int p)//求二维矩阵的最大子矩阵和
{
        int c[maxn];
        int max,sum=0;
        for(int i=0;i<n;i++)
        {
```

```
                for(int tempk=0;tempk<p;tempk++)
                        c[tempk]=0;

                for(int j=i;j<n;j++)
                {
                        for(int k=0;k<p;k++)
                                c[k]+=b[j][k];
                        max=maxSum1D(c,p);
                        if(max>sum) sum=max;
                }
        }
        return sum;
}
/*将数组a想象成一面面的墙，共有m面墙,墙宽n,高p
从第i面墙开始，依次选择相邻的0(即只有自身一面墙),1,2,3,4直到第m-1面墙,
其中i为0,1,2,3,4...,m-1*/
int maxSum3D(int a[][maxn][maxn],int m,int n,int p)//求出三维矩阵的最大长方体
{
        int b[maxn][maxn];
        int max,sum=0;

        for(int i=0;i<m;i++)
        {
                for(int tempn=0;tempn<n;tempn++)
                        for(int tempk=0;tempk<p;tempk++)
                                b[tempn][tempk]=0;


                for(int j=i;j<m;j++)
                {
                        for(int k=0;k<n;k++)
                        {
                                for(int l=0;l<p;l++)
                                {
                                        b[k][l]+=a[j][k][l];
                                }
                         }//该双重循环用于将选择的墙的压缩为一面墙，即成为二维数组

                        max=maxSum2D(b,n,p);
                        if(max>sum) sum=max;
                }
        }
        return sum;
}
int main(int argc, char* argv[])
{
        int a[maxn][maxn][maxn];
        int m,n,p;
        cin>>m>>n>>p;
```

```
        for(int i=0;i<m;i++)
                for(int j=0;j<n;j++)
                        for(int k=0;k<p;k++)
                                cin>>a[i][j][k];
        cout<<maxSum3D(a,m,n,p)<<endl;
        return 0;
}
```

11.3.6 最大k乘积问题

```
#include<stdio.h>
int cal(char* num,int i,int j){
    int value=0;
    while(j>=i){
        value=value*10+(num[i]-'0');
        i++;
    }
    return value;
}
int main(){
    int n,k;                //n代表整数有n位，k指分成多少段
    scanf("%d%d",&n,&k);
    getchar();
    char num[n+1];          //接收单个字符
    int m[n+1][n+1];        //m[i][j] 前i个数字，分成j段
    int max,value;

    for(int i=1;i<n+1;i++){
        scanf("%c",&num[i]);
    }
    //数组打底
    m[1][1]=num[1]-'0';
    for(int i=2;i<n+1;i++)
        m[i][1]=m[i-1][1]*10+(num[i]-'0');    //前i个数字，分成一段
    for(int i=2;i<=k;i++){    //分成多少段
        max=-1;
        for(int j=i;j<n+1;j++){    //j从开始遍历到最后一个数，因为i<j时(前3个数，分成4段)，无意义，所以直接令j=i
            for(int d=1;d<=j-1;d++){
                value=m[d][i-1]*cal(num,d+1,j);
                if(value>max)
                    max=value;
            }
            m[j][i]=max;
        }
    }
    printf("最大k乘积为：%d\n",m[n][k]);
    return 0;
```

```
}
```

11.3.7 最少费用购物问题

```
#include<stdio.h>
#include<string.h>
#define maxb 6 //所购商品种类数(0<=B<=5)
#define maxs 100//优惠商品种类数(0<=S<=99)
int purch[maxb][2];//存放预购商品数据
int offer[maxs][maxb];//存放优惠商品价数据
int product[maxb];
int num[1000];
int cost[maxb][maxb][maxb][maxb][maxb];
int b,s;//b件商品，s种组合
//初始化
void init()
{
int i,j,k;
int code,t,p;//code表编号，t表示优惠商品组合
memset(offer,0,sizeof(offer));
    memset(purch,0,sizeof(purch));
    memset(product,0,sizeof(product));
printf("请输入商品件数:");
scanf("%d",&b);
for(i=1;i<=b;i++)
{
scanf("%d %d %d",&code,&purch[i][0],&purch[i][1]);//code表示商品的编码；存放购买该种商品的总数；该商品的正常单价
num[code]=i;//商品的编号num[2]=1,num[8]=2
}
   printf("请输入优惠组合数目:");
   scanf("%d",&s);
   for(i=1;i<=s;i++)
   {
   scanf("%d",&t);
   for(j=1;j<=t;j++)
   {
   scanf("%d %d",&code,&p);
   offer[i][num[code]]=p;//第i种优惠组合中编号为code的商品的数量
   }
    scanf("%d",&offer[i][0]);//利用0下标存第i种优惠组合的总价格
   }
}
void mincost()
{
int i,min;
int A,B,C,D,E;
min=0;
```

```
    for(i=1;i<=b;i++)
{
min=min+(product[i]*purch[i][1]);//正常单价购买
}
//遍历s种优惠组合
   for(i=1;i<=s;i++)
   {    //五种产品减去采用优惠组合的数量
   A=product[1]-offer[i][1];
   B=product[2]-offer[i][2];
       C=product[3]-offer[i][3];
       D=product[4]-offer[i][4];
       E=product[5]-offer[i][5];
 if(A>=0&&B>=0&&C>=0&&D>=0&&E>=0&&min>(cost[A][B][C][D][E]+offer[i][0]))
 {
    min=cost[A][B][C][D][E]+offer[i][0];
 }
}
   cost[product[1]][product[2]][product[3]][product[4]][product[5]]=min;
}
void dp(int i)
{
int j;
if(i>b){
mincost();
return;
}
for(j=0;j<=purch[i][0];j++)
{
product[i]=j;
dp(i+1);
}
}
int main(int argc, char *argv[])
{
int A,B,C,D,E;
init();
dp(1);
A=purch[1][0];
B=purch[2][0];
C=purch[3][0];
D=purch[4][0];
E=purch[5][0];
printf("%d\n",cost[A][B][C][D][E]);
return 0;
}
```

11.3.8 最优时间表问题

```
#include<iostream>
#include<map>
using namespace std;
typedef long long ll;
ll dp[100002],n,m,s[100002]={0};
map<ll,int>T;
struct node{
    ll u;ll t;
}a[100002];
int main()
{
    cin>>n>>m;
    for(int i=0;i<m;i++){
        scanf("%lld%lld",&a[i].u,&a[i].t);
        T[a[i].u*100000+(s[a[i].u]++)]=a[i].t;//记录以a[i].u开始的每一个维持时
间
    }
    memset(dp,0x3f,sizeof(dp));
    dp[n+1]=0;
    for(int i=n;i>=1;i--){
        if(s[i]==0){dp[i]=dp[i+1];}
        else{
            for(int j=0;j<s[i];j++)
                dp[i]=min(dp[i+T[i*100000+j]]+T[i*100000+j],dp[i]);
        }
    }
    cout<<dp[1]<<endl;
    return 0;
}
```

11.3.9 矩形嵌套问题

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cstdlib>
using namespace std;
#define maxn 1000+10//矩形的最大个数
struct Rect
{
    int r, c;
}rect[maxn];
int n;
int G[maxn][maxn];//邻接矩阵G
int d[maxn];//dp[i]为从节点i出发的最长路长度
int dp(int i)//dp(i)求从节点i出发的最长路长度
{
```

```
    int& ans = d[i];
    if(ans > 0) return ans;  //已经求过的直接返回就好
    ans = 1;
    for(int j = 1; j <= n; j++)
        if(G[i][j])
            ans = max(ans, dp(j) + 1);//下一步只能走到相邻点,记忆化搜索。递推木写粗来。。。
    return ans;
}

void print_ans(int i)
{
    printf("%d ", i);
    for(int j = 1; j <= n; j++)
        if(G[i][j] && (d[i] == d[j] + 1)) //满足条件
    {
        print_ans(j);
        break;
    }
}
int main()
{
    int t;
    scanf("%d", &t);
    while(t--)
    {
        memset(G, 0, sizeof(G));
        memset(d, 0, sizeof(d));
        cin >> n;
        for(int i = 1; i <= n; i++)
        {
            cin >> rect[i].r >> rect[i].c;
            if(rect[i].r < rect[i].c)
            {
                int t = rect[i].r;
                rect[i].r = rect[i].c;
                rect[i].c = t;
            }
        }
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= n; j++)
            {
                if(rect[i].r < rect[j].r && rect[i].c < rect[j].c)  G[i][j] = 1;
            }
        for(int i = 1; i <= n; i++) d[i] = dp(i);
            int mm = 0;
            for(int i = 1; i <= n; i++)
                if(d[i] > mm)
```

```
                    mm = d[i];
            cout << mm << endl;
    }

    return 0;
}
```

11.3.10 导弹拦截问题

```
#include<stdio.h>

int main()
{
    int n,m,i,j,max,h[21],dp[21];
    scanf("%d",&n);
    while(n--)
    {
        max=0;
        scanf("%d",&m);
        for(i=0;i<m;i++)
        {
            scanf("%d",&h[i]);
            dp[i]=1;
        }

        for(i=m-2;i>=0;i--)
        {
            for(j=i+1;j<m;j++)
            {
                if(h[i]>h[j] && dp[i]<dp[j]+1)
                    dp[i]=dp[j]+1;
            }
            if(dp[i]>=max) max=dp[i];
        }
        printf("%d\n",max);
    }
    return 0;
}
```

11.3.11 C小加问题

```
#include<stdio.h>
#include<algorithm>
struct boom
{
    int weight,hight;
};
```

```
using namespace std;
boom a[5050];
bool cmp(boom a,boom b)  //从小到大 排序
{
    if(a.weight<b.weight)
        return true;
    if(a.weight==b.weight&&a.hight<b.hight)
       return true;
    else
      return  false;
}
int main()
{
    int t,n,i,j,m;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d",&n);
        for(i=0;i<n;i++)
            scanf("%d%d",&a[i].weight,&a[i].hight);
        sort(a,a+n,cmp);
        int q,sum=0;
        for(i=0;i<n;i++)  //重量已经有了顺序
        {
            if(a[i].hight!=0)   //  如果 这个  木棒 没有被标记的话
            {
                sum++;   //     +1
                q=a[i].hight;        //赋予临时变量
                for(j=i+1;j<n;j++)  // 看下一个
                {
                    if(a[j].hight>=q)   //下一个  大于等于的话
                    {
                        q=a[j].hight;  //更新 临时变量
                        a[j].hight=0; // 并且标记
                    }
                }
            }
        }
        printf("%d\n",sum);
    }
    return 0;
}
```

11.3.12 完全背包问题

```
#include <cstdio>
#include <iostream>
#include <map>
```

```cpp
#include <set>
#include <vector>
#include <cmath>
#include <string>
#include <cstring>
#include <algorithm>
using namespace std;
int n, v, N;
int c[2005], w[2005];
int dp[50005];
int main(){
    scanf("%d",&N);
    while(N--){

        scanf("%d%d",&n,&v);
        for(int i = 0; i <= 50005; i++) dp[i] = -1000000;

        dp[0] = 0; // 注意这里
        for(int i = 0; i < n; i++){
            scanf("%d%d",&c[i],&w[i]);
        }

        for(int i = 0; i < n; i++){
            for(int j = c[i]; j <= v; j++){
                dp[j] = max(dp[j], dp[j-c[i]]+w[i]);
            }
        }

        if(dp[v] < 0)
            puts("NO");
        else
            printf("%d\n",dp[v]);
    }
    return 0;
}

/*
完全背包问题是指每种物品都有无限件
#include <iostream>
#define V 500
using namespace std;
int weight[20 + 1];
int value[20 + 1];
int f[V + 1];
int max(int a, int b) {
    return a > b ? a : b;
}
int main() {
    int n, m;
```

```
    cout << "请输入物品个数:";
    cin >> n;
    cout << "请分别输入" << n << "个物品的重量和价值:" << endl;
    for (int i = 1; i <= n; i++) {
        cin >> weight[i] >> value[i];
    }
    cout << "请输入背包容量:";
    cin >> m;
    for (int i = 1; i <= n; i++) {
        for (int j = weight[i]; j <= m; j++) {
            f[j] = max(f[j], f[j - weight[i]] + value[i]);
        }
    }
    cout << "背包能放的最大价值为:" << f[m] << endl;
}
*/
```

11.3.13 分邮票问题

```
#include<stdio.h>
#include<string.h>
using namespace std;
int val[500001];
int main()
{
int n,m,i,j,sum;
int a[1001];
scanf("%d",&m);
while(m--)
{
memset(val,0,sizeof(val));
scanf("%d",&n);
sum=0;
for(i=0;i<n;++i)
{
scanf("%d",&a[i]);
sum+=a[i];
}
for(i=0;i<n;++i)
for(j=sum/2;j>=a[i];--j)
if(val[j]<val[j-a[i]]+a[i])
val[j]=val[j-a[i]]+a[i];
printf("%d\n",sum-2*val[sum/2]);
}
return 0;
}
```

11.3.14 排列问题

```
#include <stdio.h>

int main()
{
    int i,f[60];
    int n;
    f[1]=f[2]=1;    f[3]=2;
    for(i=4;i<56;i++)    f[i]=f[i-1]+f[i-3]+1;
    while(scanf("%d",&n)!=EOF)    printf("%d\n",f[n]);

    return 0;
}
```

11.3.15 完全覆盖问题

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
long long a[15][15];
long long dp[15][1<<12];
int n,m;
bool judge1(int s){//判断的标准是必须连续两格为1
    int i;
    for(i=0;i<m;){
        if(s & (1<<i)){
            if(i==m-1)return 0;
            else if(s & (1<<(i+1)))i+=2;
            else return 0;
        }
        else i++;
    }
    return 1;
}
bool judge2(int s,int ss){//判断第i-1行的s情况与i行的情况是否兼容
    int i;
    for(i=0;i<m;){
        if(s & (1<<i)){
            if(ss & (1<<i)){
                if(i==m-1 ||!(s &(1<<i+1))||!(ss &(1<<i+1)))return 0;
                else i+=2;
            }
            else i++;
        }
```

```
        else{
            if(ss &(1<<i))i++;
            else return 0;
        }
    }
    return 1;
}
void solve(){
    int s,ss,i;
    memset(dp,0,sizeof(dp));
    if(n<m){//为了减少情况数量，使小的为列数
        int temp;
        temp=n;n=m;m=temp;
    }
    int maxx=(1<<m)-1;
    for(s=0;s<=maxx;s++){//第一行每一种可行的情况
        if(judge1(s)){
            dp[1][s]=1;
        }
    }
    for(i=2;i<=n;i++){
        for(s=0;s<=maxx;s++){
            for(ss=0;ss<=maxx;ss++){
                if(judge2(s,ss)){//判断第i-1行与第i行情况是否兼容
                    dp[i][ss]+=dp[i-1][s];
                }
            }
        }
    }
    a[n][m]=a[m][n]=dp[n][maxx];
    cout<<a[n][m]<<endl;
}
int main(){
    int i,j;
    memset(a,-1,sizeof(a));
    while(scanf("%d%d",&n,&m)!=EOF&&n&&m){
        if(a[n][m]!=-1){//不为-1则代表以及得出答案
            cout<<a[n][m]<<endl;
            continue;
        }
        if(n&1 && m&1){//如果长和宽都为奇数，则方案数为0
            a[n][m]=a[m][n]=0;
            cout<<a[n][m]<<endl;
            continue;
        }
        solve();
    }
    return 0;
}
```