1. 求原根个数 [数论] [普及+/提高]

CF284A Cows and Primitive Roots

题目描述：

奶牛们刚学会原根的定义! 给定一个质数 p 那么模 p 意义下的一个原根 $x(1 <= x < p)$ 满足如下性质:

$x-1, x^2-1, x^3-1 \cdots x^{p-2}-1$都不能被 p 整除,但是$x^{p-1}-1$却可以。

但是奶牛们太菜了需要你的帮助,给定一个质数 ***p***,求模 ***p*** 意义下原根的个数。

关键点：

1. x^i-1 不能被p整除 $\iff$ x^i 模p不能等于 **1**，x^i-1 能被p整除 $\iff$ x^i 模 p 等于 **1**，所以只需要判断 x^i 模 p的情况。
2. 模运算的规则：

$$(a+b)\ \%\ p = (a\ \%\ p\ +\ b\ \%\ p)\ \%\ p \tag{1}$$

$$(a-b)\ \%\ p = (a\ \%\ p\ -\ b\ \%\ p)\ \%\ p \tag{2}$$

$$(a*b)\ \%\ p = (a\ \%\ p\ *\ b\ \%\ p)\ \%\ p \tag{3}$$

3.

$$(\ a^b\ )\ \%\ p\ =\ (\ (a\ \%\ p)^b\ )\ \%\ p \tag{4}$$

应用公式(3):

$$x^i\ \%\ p\ =\ (x^{i-1}\ \%\ p\ *\ x\ \%\ p)\ \%\ p \tag{5}$$

部分代码：

```
bool solve(int x)
{
    int res = 1;
    int temp = x % p;
    for (int i = 1; i < p - 1; ++i)
    {
        res = (res * temp) % p;
        if (res == 1)
            return false;
    }
    res = (res * temp) % p;
    if (res == 1)
        return true;
    return false;
}
```

2. 过河卒 [动态规划] [普及-]

P1002 [NOIP2002 普及组] 过河卒

题目描述：

棋盘上 A*A* 点有一个过河卒，需要走到目标 B*B* 点。卒行走的规则：可以向下、或者向右。同时在棋盘上 C*C* 点有一个对方的马，该马　　所在的点和所有跳跃一步可达的点称为对方马的控制点。因此称之为“马拦过河卒”。

棋盘用坐标表示，A*A* 点 (0, 0)(0,0)、B*B* 点 (n, m)(*n*,*m*)，同样马的位置坐标是需要给出的。

现在要求你计算出卒从 A*A* 点能够到达 B*B* 点的路径的条数，假设马的位置是固定不动的，并不是卒走一步马走一步。

关键点：

1. 用一个set，将马通过0步或一步能到的位置提前存储起来。
2. 状态转移方程为：

$$f[1][1] = 1 \tag{6}$$

$$f[i][j] = \max(f[i-1][j] + f[i][j-1], f[i][j]) \tag{7}$$

关键代码：

```
        for (int j = 1; j <= m; ++j)
        {
            if (s.count(pair<int, int>(i, j)) == 1)
            {
                dp[i][j] = 0;
                continue;
            }
            if (i == 0)
                dp[i][j] = dp[i][j-1];
            else
                dp[i][j] = dp[i - 1][j] + dp[i][j - 1];
        }
```

3. 铺地毯 [模拟] [普及-]

P1003 [NOIP2011 提高组] 铺地毯

题目：

为了准备一个独特的颁奖典礼，组织者在会场的一片矩形区域（可看做是平面直角坐标系的第一象限）铺上一些矩形地毯。一共有 n*n* 张地毯，编号从 1 到 n*n*。现在将这些地毯按照编号从小到大的顺序平行于坐标轴先后铺设，后铺的地毯覆盖在前面已经铺好的地毯之上。

地毯铺设完成后，组织者想知道覆盖地面某个点的最上面的那张地毯的编号。注意：在矩形地毯边界和四个顶点上的点也算被地毯覆盖。

输入共 *n*+2 行。

第一行，一个整数 *n*，表示总共有 *n* 张地毯。

接下来的 *n* 行中，第 *i*+1 行表示编号 *i* 的地毯的信息，包含四个整数 *a*, *b*, *g*, *k*，每两个整数之间用一个空格隔开，分别表示铺设地毯的左下角的坐标 (*a*,*b*) 以及地毯在 *x* 轴和 *y* 轴方向的长度。

第 *n*+2 行包含两个整数 *x* 和 *y*，表示所求的地面的点的坐标 (*x*,*y*)。

关键点：

1. 巧妙利用输入的先后关系来处理地毯的上下关系。

```
int res=-1,n, x, y;
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i)
    {
        scanf("%d%d%d%d", &arr[i].a, &arr[i].b, &arr[i].g, &arr[i].k);
    }
    scanf("%d%d", &x, &y);
    for(int i=1;i<=n;++i){
        if(x>=arr[i].a&&x<=arr[i].a+arr[i].g&&y>=arr[i].b&&y<=arr[i].b+arr[i].k)
            res=i;
    }
    printf("%d\n",res);
```

4. 三连击[枚举/暴力] [普及-]

P1008 [NOIP1998 普及组] 三连击

题目描述:

将$1,2,\ldots,9$共9个数分成3组，分别组成3个三位数，且使这3个三位数构成$1:2:3$的比例，试求出所有满足条件的3个三位数。

看代码:

```
int a,b,c;
for(a=123;a<=333;a++)
{
    b=a*2;
    c=a*3;

 if((a/100+a/10%10+a%10+b/100+b/10%10+b%10+c/100+c/10%10+c%10==1+2+3+4+5+6+7+8+9)&&((a/100)*(a/10%10)*(a%10)*(b/100)*(b/10%10)*(b%10)*(c/100)*(c/10%10)*(c%10)==(1)*(2)*(3)*(4)*(5)*(6)*(7)*(8)*(9)))
        printf("%d %d %d\n",a,b,c);
}
```

```
 for(i=192;i<=327;i++)//第一个数最小192，最大327。其实不知道的情况下简单来说是从123-329的但是算出来是最值就稍微改了下下
    {
        memset(a,0,sizeof(a));v=0;//清零

 a[i%10]=a[i/10%10]=a[i/100]=a[i*2%10]=a[i*2/10%10]=a[i*2/100]=a[i*3%10]=a[i*3/10%10]=a[i*3/100]=1;//统计数字
        for(j=1;j<=9;j++) v+=a[j];//v表示1-9这些数字是否全部齐了
        if(v==9) printf("%d %d %d\n",i,i*2,i*3);//如果齐了就输出
    }
```

5. Cantor 表 [模拟/枚举/暴力] [入门]

P1014 [NOIP1999 普及组] Cantor 表

题目描述：

现代数学的著名证明之一是 Georg Cantor 证明了有理数是可枚举的。他是用下面这一张表来证明这一命题的：

1/1 , 1/2 , 1/3 , 1/4, 1/5, …

2/1，2/2 , 2/3, 2/4, …

3/1 , 3/2，3/3, …

4/1，4/2, …

5/1, …

…

我们以 Z 字形给上表的每一项编号。第一项是 1/1，然后是 1/2，2/1，3/1，2/2，…

关键点：

算法1：模拟，按题意一个个枚举

时间复杂度O(n),可以通过本题n≤10^7

算法2：发现Z字形的每条斜线可以快速枚举，即枚举

1/1 , 1/2 , 3/1 , 1/4 , 5/1 , 1/6……找到要求的第n项所在斜线，再一个个枚举或计算得出答案

时间复杂度O(√n),可以通过n≤10^14

算法3：发现第i条斜线（即分子分母之和=i+1的所有项）中包含*i(i-1)/2+1至*(i+1)中的每一项，所以可以二分分子分母之和，再根据分　子分母之和的奇偶性直接计算第n项

时间复杂度O($\log_2 n$),可以通过n≤10^{18},加上高精可通过n≤10^{1000}

算法2代码：

```
    for (int i = 1; i <= 10000; ++i)
    {
        a[i] = i * (i - 1) / 2 + 1;
    }
    int k = lower_bound(a + 1, a + 10001, n) - a;
    if (a[k] != n)
        --k;
    if (k & 1)
    {
        cout << k - (n - a[k]) << "/" << 1 + n - a[k] << endl;
    }
    else
    {
        cout << 1 + n - a[k] << "/" << k - (n - a[k]) << endl;
```

```
    }
```

算法3代码：

```
long long l=1,r,mid,n,a;
       cin>>n;
       r=n;
       while(l<r){
           mid=(l+r)/2;
           if(mid*(mid+1)/2<n)l=mid+1;
           else r=mid;
       }
       a=n-l*(l-1)/2;
       if(l%2==0)cout<<a<<'/'<<l+1-a;
       else cout<<l+1-a<<'/'<<a;
```

6. 单词接龙 [字符串/搜索] [普及+/提高-]

P1019 [NOIP2000 提高组] 单词接龙

题目描述：

单词接龙是一个与我们经常玩的成语接龙相类似的游戏，现在我们已知一组单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”（每个单词都最多在“龙”中出现两次），在两个单词相连时，其重合部分合为一部分，例如 `beast` 和 `astonish`，如果接成一条龙则变为 `beastonish`，另外相邻的两部分不能存在包含关系，例如 `at` 和 `atide` 间不能相连。

代码：

```
int n, g[21][21], ans, c[21], len[21], b[42];
string a[21];
char s;
void dfs(int x, int cnt)
{
    bool f = false;
    for (int i = 1; i <= n; ++i)
    {
        if (g[x][i] && c[i] < 2)
        {
            f = true;
            ++c[i];
            b[cnt + 1] = i;
            dfs(i, cnt + 1);
            b[cnt + 1] = 0;
            --c[i];
        }
    }
    if (!f)
    {
        int count = 0;
        for (int i = 0; i <= cnt; ++i)
        {
            count += len[b[i]];
```

```
        }
        for (int i = 0; i < cnt; ++i)
        {
            count -= g[b[i]][b[i + 1]];
        }
        if (count > ans)
            ans = count;
    }
}
int main()
{
    cin >> n;
    for (int i = 1; i <= n; ++i)
    {
        cin >> a[i];
        len[i] = a[i].size();
    }
    cin >> s;
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= n; ++j)
        {
            string s1 = a[i], s2 = a[j];
            int k = 0;
            if (i == j)
                k = 1;
            for (; k < s1.size(); ++k)
            {
                if (s1[k] == s2[0])
                {
                    int v = k;
                    while (v < s1.size() && v - k < s2.size())
                    {
                        if (s1[v] != s2[v - k])
                            break;
                        ++v;
                    }
                    if (v == s1.size())
                    {
                        if (k == 0 || v - k == s2.size())
                            break;
                        else
                            g[i][j] = s1.size() - k;
                    }
                }
            }
        }
    }
    for (int i = 1; i <= n; ++i)
    {
        if (a[i][0] == s)
        {
            c[i] = 1;
            b[0] = i;
            dfs(i, 0);
        }
    }
    printf("%d\n", ans);
    system("pause");
    return 0;
}
```

关键点：

1. 用矩阵g[i][j]记录字符串 i 和 j 首尾相连时重合部分的长度。数组a记录字符串的输入顺序。数组b[i]记录第 i 次选的是第几个字符串。数组c[i]记录第 i 个字符串被选了几次。
2. 看清楚题目，求的是长度，而不是字符串。
3. 注意单词有可能自己和自己连接，
4. 注意数组作为参数传递的情况，
5. 回溯不需要传递数组，直接用全局数组就可以。
6. 注意长度不要重复加。
7. dfs前，注意一些变量的赋值。
8. 写代码之前想清楚怎么写。
9. 不要写错了变量。
10. 明确各个变量的意义。

7. 数的计算 [递推/递归] [普及-]

P1028 [NOIP2001 普及组] 数的计算1

题目描述：

我们要求找出具有下列性质数的个数(包含输入的正整数 n)。

先输入一个正整数 n(n≤1000),然后对此正整数按照如下方法进行处理：

1. 不作任何处理；
2. 在它的左边加上一个正整数,但该正整数不能超过原数的一半；
3. 加上数后,继续按此规则进行处理,直到不能再加正整数为止。

方法1：暴力递归：

```
#include<cstdio>
using namespace std;
int n,cnt=1;
void func(int x){
    for(int i=1;i<=x/2;i++){
        cnt++;
        func(i);
    }
}
int main(){
    scanf("%d",&n);
    func(n);
    printf("%d\n",cnt);
}
```

这个递归大概能骗过n=500，然而题目中是n<=1000 所以正解肯定不是暴力。

方法2：递推

```cpp
#include<bits/stdc++.h>//万能头文件
using namespace std;
int n;
int f[1001];//存每一位数的种类
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){ //1-n的递推
        for(int j=1;j<=i/2;j++){
            f[i]+=f[j]; //每一位叠加，递推走起
        }
        f[i]++; //加上本身
    }
    cout<<f[n];//输出n的种类
    return 0;
}
```

我们以4为例子来进行说明

4后面可以跟上1,2组成14,24

14后面跟不了,24可以跟上1组成124

再加上4本身就可以得到4的种类

即 14,24,124,4

而我们只要算出1,2的种类就可以加起来得到4的种类

总体来说，这道题是数学思想以及对递推的理解。

8. 求先序排列 [字符串/树形结构/递归] [普及-]

P1030 [NOIP2001 普及组] 求先序排列

题目描述：

给出一棵二叉树的中序与后序排列。求出它的先序排列。（约定树结点用不同的大写字母表示，长度≤8）。

关键点：

首先，一点基本常识，给你一个后序遍历，那么最后一个就是根（如ABCD，则根为D）。

因为题目求先序，意味着要不断找根。

那么我们来看这道题方法：（示例）

中序ACGDBHZKX，后序CDGAHXKZB，首先可找到主根B；

那么我们找到中序遍历中的B，由这种遍历的性质，可将中序遍历分为ACGD和HZKX两棵子树，

那么对应可找到后序遍历CDGA和HXKZ（从头找即可）

从而问题就变成求1.中序遍历ACGD，后序遍历CDGA的树 2.中序遍历HZKX，后序遍历HXKZ的树；

接着递归，按照原先方法，找到1.子根A，再分为两棵子树2.子根Z，再分为两棵子树。

就按这样一直做下去（先输出根，再递归）；

模板概括为step1:找到根并输出

step2:将中序，后序各分为左右两棵子树；

step3:递归，重复step1,2；

代码：

```
#include<cstdio>
#include<iostream>
#include<cstring>
using namespace std;
void beford(string in,string after){
    if (in.size()>0){
        char ch=after[after.size()-1];
        cout<<ch;//找根输出
        int k=in.find(ch);
        beford(in.substr(0,k),after.substr(0,k));
        beford(in.substr(k+1),after.substr(k,in.size()-k-1));//递归左右子树；
    }
}
int main(){
    string inord,aftord;
    cin>>inord;cin>>aftord;//读入
    beford(inord,aftord);cout<<endl;
    return 0;
}
```

9. 选数 [DFS] [普及-]

P1036 [NOIP2002 普及组] 选数

题目描述：

已知 n 个整数 $x1,x2,...,x*n$，以及11个整数$k(k<n)$。从n个整数中任选k个整数相加，可分别得到一系列的和。例如当$n=4,k=3$,44个整数分别为3,7,12,19时，可得全部的组合与它们的和为：

3+7+12=22

3+7+19=29

7+12+19=38

3+12+19=34。

现在，要求你计算出和为素数共有多少种。

例如上例，只有一种的和为素数：3+7+19=29

代码：

```
#include<iostream>
#include<math.h>
using namespace std;
int x[20],n,k;//依照题目所设
```

```
bool isprime(int n){//判断是否质数
    int s=sqrt(double(n));
    for(int i=2;i<=s;i++){
        if(n%i==0)return false;
    }
    return true;
}
int rule(int choose_left_num,int already_sum,int start,int end){//choose_left_num为剩余的k，already_sum为前面累加的和，start和end为全组合剩下数字的选取范围；调用递归生成全组合，在过程中逐渐把K个数相加，当选取的数个数为0时，直接返回前面的累加和是否为质数即可
    if(choose_left_num==0)return isprime(already_sum);
    int sum=0;
    for(int i=start;i<=end;i++){
        sum+=rule(choose_left_num-1,already_sum+x[i],i+1,end);
    }
    return sum;
}
int main(){
    cin>>n>>k;
    for(int i =0;i<n;i++)cin>>x[i];
    cout<<rule(k,0,0,n-1);//调用递归解决问题
}
```

其实这里的难点是：如何去重？

答案是：不降原则

10. 乒乓球 [字符串/模拟] [普及-]

P1042 [NOIP2003 普及组] 乒乓球

题目描述：

华华通过以下方式进行分析，首先将比赛每个球的胜负列成一张表，然后分别计算在1111分制和2121分制下，双方的比赛结果（截至记录末尾）。

比如现在有这么一份记录，（其中`W`表示华华获得一分，`L`表示华华对手获得一分）：

WWWWWWWWWWWWWWWWWWWWWWWLW

在1111分制下，此时比赛的结果是华华第一局11比0获胜，第二局11比0获胜，正在进行第三局，当前比分1比1。而在21分制下，此时比赛结果是华华第一局21比0获胜，正在进行第二局，比分2比1。如果一局比赛刚开始，则此时比分为0比0。直到分差大于或者等于2，才一局结束。

你的程序就是要对于一系列比赛信息的输入（*W****L*形式），输出正确的结果。

关键点：

1. 比赛必须要有一方的分数不小于11或21，并且领先2个球才能获胜，11:10的比分是不存在的

代码：

```
#include <iostream>
#include <cstdio>
using namespace std;
char a[62510];
int cnt = 0, x = 0, y = 0;
```

```
int main()
{
    while (scanf(" %c", &a[++cnt]))
    {
        if (a[cnt] == 'E')
            break;
    }
    for (int i = 1; i < cnt; ++i)
    {
        if (a[i] == 'W')
            ++x;
        else if (a[i] == 'L')
            ++y;
        if ((x >= 11 && x - 2 >= y) || (y >= 11 && y - 2 >= x))
        {
            printf("%d:%d\n", x, y);
            x = y = 0;
        }
    }
    printf("%d:%d\n", x, y); //最后要输出，即使是x=y=0，不要忘记了。
    printf("\n");
    x = y = 0;
    for (int i = 1; i < cnt; ++i)
    {
        if (a[i] == 'W')
            ++x;
        else
            ++y;
        if ((x >= 21 && x - 2 >= y) || (y >= 21 && y - 2 >= x))
        {
            printf("%d:%d\n", x, y);
            x = y = 0;
        }
    }
    printf("%d:%d\n", x, y); //最后要输出，即使是x=y=0，不要忘记了。
    return 0;
}
```

11. 栈 [dp, catalan,递推] [普及-]

P1044 [NOIP2003 普及组] 栈

题目描述：

一个操作数序列 1,2,...,n，栈 A 的深度大于 n。

现在可以进行两种操作，

1. 将一个数，从操作数序列的头端移到栈的头端（对应数据结构栈的 push 操作）
2. 将一个数，从栈的头端移到输出序列的尾端（对应数据结构栈的 pop 操作）

使用这两种操作，由一个操作数序列就可以得到一系列的输出序列。

你的程序将对给定的 nn，计算并输出由操作数序列 1,2,\ldots,n1,2,...,n 经过操作可能得到的输出序列的总数。

1. 递归/记忆化搜索

看这个数据，我总感觉dfs会超时，然后真的超了？（没试过），于是很自然的，我们就会想到记忆化搜索，这也是做这题的一种技巧吧，但无论如何，这也是最基础的

下面谈谈搜索(递归)思路：

- 既然记忆化搜索了，定义一个二维数组*f[i,j]*，用下标 *i* 表示队列里还有几个待排的数，*j* 表示栈里有 *j* 个数，*f[i,j]*表示此时的情况数
- 那么，更加自然的，只要f[i,j]*f[i,j]*有值就直接返回；
- 然后递归如何实现呢？首先，可以想到，要是数全在栈里了，就只剩1种情况了，所以：i*=0时，返回1；
- 然后，有两种情况：一种栈空，一种栈不空：在栈空时，我们不可以弹出栈里的元素，只能进入，所以队列里的数−1，栈里的数+1，即加上 *f[i*−1,*j*+1] ；另一种是栈不空，那么此时有出栈1个或者进1个再出1个 2种情况，分别加上 *f[i*−1,*j*+1] 和 *f[i,j*−1] ，便是此时的情况了，于是递归就愉快的结束了；

2. 动态规划

c[i][j]表示有i个数已经进栈，有j个数已经出栈的方法总数

那么如果现在有i-1个数进栈，j个数出栈，不难发现，再入栈一个数就能得到i个数进栈，j个数出栈的状况

如果有i个数进栈，j-1个数出栈，不难发现，再出栈一个数就能得到i个数进栈，j个数出栈的状况。

$$c[i][j] = c[i-1][j] + c[i][j-1] \tag{8}$$

i个数进栈，0个数出栈不是只有1种方法。

3. catalan

递推式1

$$f[n] = f[0] * f[n-1] + f[1] * f[n-2] + \ldots + f[n-1]f[0] (n \geq 2) \tag{9}$$

递推式2

$$f[n] = C[2n, n] - C[2n, n-1] (n = 0, 1, 2, \ldots) \tag{10}$$

代码：

```
ll c[MAX_N*2][MAX_N];
int main(){

    scanf("%d",&n);
    for(int i=1;i<=2*n;i++)
    {
        c[i][0]=c[i][i]=1;
        for(int j=1;j<i;j++)
        {
            c[i][j]=c[i-1][j]+c[i-1][j-1];
        }
    }
    printf("%lld",c[2*n][n]-c[2*n][n-1]);
    return 0;
}
```

12. 麦森数 [数论/高精] [普及/提高-]

P1045 [NOIP2003 普及组] 麦森数

形如 2^P-1 的素数称为麦森数，这时*P*一定也是个素数。但反过来不一定，即如果*P*是个素数，2^P-1不一定也是素数。到1998年底，人们已找到了37个麦森数。最大的一个是P*=3021377，它有909526位。麦森数有许多重要应用，它与完全数密切相关。

任务：从文件中输入*P*（1000<*P*<3100000），计算 2^P-1 的位数和最后500位数字（用十进制高精度数表示）

第一行：十进制高精度数2^P-1 的位数。

第2-11行：十进制高精度数2^P-1 的最后500位数字。（每行输出50位，共输出10行，不足500位时高位补0）

这道题可以分为两个模块，第一个模块为求的位数，第二个模块为求的后500位（不足补零）。我们主要来解决第一个模块：

一、求位数

首先我们知道 2^p-1 与 2^p 有着相同的位数，因为2的次方满足了最后一位不为零的要求，所以减一后位数并不会改变，那么我们可以直接求 2^p 的位数。那么怎么求位数呢？我们不妨设 $k=2^p$，根据 10^n 的位数为 $n+1$，我们只要想办法把 中的底数$k=2^p$改为10，指数加一就是位数了。由此想到用10的几次方来代替2，那么就不难想到 $10^{log_{10}2}=2$，这样便可以把 中的2代换掉， $k=2^p$ 变为了 $k=(10^{log_{10}2})^p$ 。根据乘方的原理，将p乘进去，原式便可化为我们最终想要的形式 $k=10^{p*log_{10}2}$ 了，所以位数就是$10^{p*log_{10}2}+1$。

二、求最后500位数

高精快速幂

13. 装箱问题 [dp,背包] [普及-]

P1049 [NOIP2001 普及组] 装箱问题

题目描述

有一个箱子容量为*V*（正整数, 0≤*V*≤20000），同时有*n*个物品（0<*n*≤30，每个物品有一个体积（正整数）。

要求*n*个物品中，任取若干个装入箱内，使箱子的剩余空间为最小。

关键点：

将价值设置为体积的大小，运用01背包问题即可解决。

14. 金明的预算方案 [dp] [普及+/提高]

P1064 [NOIP2006 提高组] 金明的预算方案

题目描述：

金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间金明自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他··　　说：“你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过 *n* 元钱就行”。今天一早，金明就开始做预算了，他把想买的　　物品分为两类：主件与附件，附件是从属于某个主件的，

如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有 0 个、1 个或 2 个附件。每个附件对应一个主件，附　　件 不再有从属于自己的附件。金明想买的东西很多，肯定会超过妈妈限定的 *n* 元。于是，他把每件物品规定了一个重要度，分为 5　　等：用整数 1~5 表示，第 5 等最重要。他还从因特网上查到了每件物品的价格（都是 10 元的整数倍）。他希望在不超过 *n* 元的前　　提下，使每件物品的价格与重要度的乘积的总和最大。

设第 jj 件物品的价格为 v_j，重要度为w_j，共选中了 *k* 件物品，编号依次为$j_1, j_2, \ldots, j_k$则所求的总和为：

$$v_{j_1} \times w_{j_1} + v_{j_2} \times w_{j_2} + \cdots + v_{j_k} \times w_{j_k} \tag{11}$$

请你帮助金明设计一个满足要求的购物单。

第一行有两个整数，分别表示总钱数 n*n* 和希望购买的物品个数 m*m*。

第 22 到第(*m*+1) 行，每行三个整数，第 (*i*+1) 行的整数 v_i，p_i，q_i 分别表示第 *i* 件物品的价格、重要度以及它对应的的主件。如果　　$q_i = 0$，表示该物品本身是主件。

思路：

还记得01背包的决策是什么吗?

- 不选，然后去考虑下一个
- 选，背包容量减掉那个重量，总值加上那个价值。

这个题的决策是五个，分别是：

- 不选，然后去考虑下一个
- 选且只选这个主件
- 选这个主件，并且选附件1
- 选这个主件，并且选附件2
- 选这个主件，并且选附件1和附件2.

代码：

```
#include <iostream>
#include <algorithm>
using namespace std;
int dp[61][32001], z[61][2], f1[61][2]={0}, f2[61][2]={0};
int N, M;
int main()
{
    cin >> N >> M;
    int a, b, c;
    for (int i = 1; i <= M; ++i)
    {
        cin >> a >> b >> c;
        if (c == 0)
        {
            z[i][0] = a;
```

```
            z[i][1] = b;
        }
        else
        {
            if (f1[c][0] == 0)
            {
                f1[c][0] = a;
                f1[c][1] = b;
            }
            else
            {
                f2[c][0] = a;
                f2[c][1] = b;
            }
        }
    }
    for (int i = 1; i <= M; ++i)
    {
        for (int j = 1; j <= N; ++j)
        {
            if (j >= z[i][0])
            {
                dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - z[i][0]] + z[i][0] * z[i][1]);
                if (j >= z[i][0] + f1[i][0])
                {
                    dp[i][j] = max(dp[i][j], dp[i - 1][j - z[i][0] - f1[i][0]] + z[i][0] * z[i][1] + f1[i][0] * f1[i][1]);
                }
                if (j >= z[i][0] + f2[i][0])
                {
                    dp[i][j] = max(dp[i][j], dp[i - 1][j - z[i][0] - f2[i][0]] + z[i][0] * z[i][1] + f2[i][0] * f2[i][1]);
                }
                if (j >= z[i][0] + f1[i][0] + f2[i][0])
                {
                    dp[i][j] = max(dp[i][j], dp[i - 1][j - z[i][0] - f1[i][0] - f2[i][0]] + z[i][0] * z[i][1] + f1[i][0] * f1[i][1] + f2[i][0] * f2[i][1]);
                }
            }
            else
            {
                dp[i][j] = dp[i - 1][j];
            }
        }
    }
    cout<<dp[M][N]<<endl;
    return 0;
}
```

15. 火星人 [搜索] [普及-]

P1088 [NOIP2004 普及组] 火星人

题目描述：

人类终于登上了火星的土地并且见到了神秘的火星人。人类和火星人都无法理解对方的语言，但是我们的科学家发明了一种用数字交流的方法。这种交流方法是这样的，首先，火星人把一个非常大的数字告诉人类科学家，科学家破解这个数字的含义后，再把一个很小的数字加到这个大数上面，把结果告诉火星人，作为人类的回答。

火星人用一种非常简单的方式来表示数字——掰手指。火星人只有一只手，但这只手上有成千上万的手指，这些手指排成一列，分别编号为1,2,3...。火星人的任意两根手指都能随意交换位置，他们就是通过这方法计数的。

一个火星人用一个人类的手演示了如何用手指计数。如果把五根手指——拇指、食指、中指、无名指和小指分别编号为1,2,3,4和55，当它们按正常顺序排列时，形成了5位数12345，当你交换无名指和小指的位置时，会形成5位数12354，当你把五个手指的顺序完全颠倒时，会形成54321，在所有能够形成的120个5位数中，12345最小，它表示1；12354第二小，它表示2；54321最大，它表示120。

现在你有幸成为了第一个和火星人交流的地球人。一个火星人会让你看他的手指，科学家会告诉你要加上去的很小的数。你的任务是，把火星人用手指表示的数与科学家告诉你的数相加，并根据相加的结果改变火星人手指的排列顺序。输入数据保证这个结果不会超出火星人手指能表示的范围。

代码1:

```
#include<bits/stdc++.h>
using namespace std;
int a[10005],n,m;
int main(){
    scanf("%d%d",&n,&m);
    for(int i=0;i<n;i++)scanf("%d",&a[i]);
    while(m--)    next_permutation(a,a+n);
    for(int i=0;i<n-1;i++)    printf("%d ",a[i]);    printf("%d",a[n-1]);
    return 0;
}
```

代码2:

```
#include <iostream>
#include <cstdio>
using namespace std;
inline int read();
int n, m, a[10001], b[10001], used[10001], ans;
void dfs(int cnt)
{
    if (cnt == n + 1)
    {
        ++ans;
    }
    else
    {
        for (int i = 1; i <= n; ++i)
        {
            if (!ans)
                i = a[cnt];
            if (!used[i])
            {
                used[i] = 1;
                b[cnt] = i;
                dfs(cnt + 1);
                used[i] = 0;
                if (ans == m + 1)
                    return;
```

```
            }
        }
    }
}
int main()
{
    n = read();
    m = read();
    for (int i = 1; i <= n; ++i)
    {
        a[i] = read();
    }
    dfs(1);
    for (int i = 1; i <= n; ++i)
    {
        printf("%d%c", b[i], i == n ? '\n' : ' ');
    }
    system("pause");
    return 0;
}
```

16. 合并果子 [贪心/优先队列] [普及-]

P1090 [NOIP2004 提高组] 合并果子 / [USACO06NOV] Fence Repair G

题目描述：

在一个果园里，多多已经将所有的果子打了下来，而且按果子的不同种类分成了不同的堆。多多决定把所有的果子合成一堆。

每一次合并，多多可以把两堆果子合并到一起，消耗的体力等于两堆果子的重量之和。可以看出，所有的果子经过 $n-1$ 次合并之后， 就只剩下一堆了。多多在合并果子时总共消耗的体力等于每次合并所耗体力之和。

因为还要花大力气把这些果子搬回家，所以多多在合并果子时要尽可能地节省体力。假定每个果子重量都为 1 ，并且已知果子的种类 数和每种果子的数目，你的任务是设计出合并的次序方案，使多多耗费的体力最少，并输出这个最小的体力耗费值。

例如有 33 种果子，数目依次为 1 ， 2 ， 9 。可以先将 1 、 2 堆合并，新堆数目为 3 ，耗费体力为 3 。接着，将新堆与原先的第三堆合并，又得到新的堆，数目为 12 ，耗费体力为 12 。所以多多总共耗费体力 3+12=15 。可以证明 15 为最小的体力耗费值。

```
int n;
long long res = 0;
int main()
{
    scanf("%d", &n);
    int t;
    priority_queue<int, vector<int>, greater<int> > pq;
    for (int i = 0; i < n; ++i)
    {
        scanf("%d", &t);
        pq.push(t);
    }
    while (pq.size() > 1)
    {
        int m1 = pq.top();
```

```
        pq.pop();
        int m2 = pq.top();
        pq.pop();
        int m = m1 + m2;
        res += m;
        pq.push(m);
    }
    printf("%lld\n",res);
    return 0;
}
```

17. 纪念品分组 [贪心/排序] [普及-]

P1094 [NOIP2007 普及组] 纪念品分组

题目描述：

元旦快到了，校学生会让乐乐负责新年晚会的纪念品发放工作。为使得参加晚会的同学所获得 的纪念品价值相对均衡，他要把购来的纪念品根据价格进行分组，但每组最多只能包括两件纪念品， 并且每组纪念品的价格之和不能超过一个给定的整数。为了保证在尽量短的时间内发完所有纪念品，乐乐希望分组的数目最少。

你的任务是写一个程序，找出所有分组方案中分组数最少的一种，输出最少的分组数目。

贪心算法：先对数组从小到大排序，用 `i = 1, j = n` 指针指向首尾元素； 如果 $a_i + a_j > w$，则将 a_j 单独作为一组，指针 `j--` ；如果 $a_i + a_j \leq w$， 则将 a_i 和 a_j 分为一组， 指针 `i--, j--`。 如此重复直到 “取完” 所有元素。

贪心算法证明： 对于 `a[i...j]` 问题，如果存在最优解 *S*，但是 a_i 和 a_j 的分组不符合上述贪心选择过程。会有以下几种情况：

1. 如果 $a_i + a_j > w$也不可能与其他任何a_k，i < k < j 一组, a_j 只能单独一组。这是符合贪心选择性质的。
2. 如果 $a_i + a_j \leq w$， 在最优解中，a_j并不与 a_i 一组，

- a_j 单独一组，这时候如果最优解的 a_i 仍然孤单， 那么将他俩合为一组，最优解的分组数减一，居然优于最优解，矛盾。
- a_j 单独一组， a_i 与另一个 a_k 一组，这时候，将 a_i 从 a_k 身边拆开，再与 a_j 一组，所得分组数不变，新的解 S'=S*，a_i和 a_j 组是符合贪心选择性质的，贪心选择可以得到最优解。
- a_j 与 a_k 一组，a_i 单独一组，交换 a_i 和 a_k， 不难看出 S'=S。
- a_j 与 a_k 一组， a_i 与 a_m 一组，交换 a_k 与 a_i之后， $a_i + a_j \leq w$，$a_k + a_m \leq a_k + a_j \leq w$，S'=S*，

代码：

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int n, w, a[30001], ans;
int main()
{
    scanf("%d%d", &w, &n);
    for (int i = 0; i < n; ++i)
    {
        scanf("%d", &a[i]);
    }
```

```
    sort(a, a + n);
    int i = 0, j = n - 1;
    while (i <= j)
    {
        if (a[i] + a[j] <= w)
        {
            ++ans;
            ++i;
            --j;
        }
        else
        {
            --j;
            ++ans;
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

18. 单词方阵 [字符串/搜索] [普及-]

P1101 单词方阵

题目描述：

给一n×n的字母方阵，内可能蕴含多个“`yizhong`”单词。单词在方阵中是沿着同一方向连续摆放的。摆放可沿着 8 个方向的任一方向，同一单词摆放时不再改变方向，单词与单词之间可以交叉,因此有可能共用字母。输出时，将不是单词的字母用`*`代替，以突出显示单词。例如：

```
输入：
    8                   输出：
    qyizhong            *yizhong
    gydthkjy            gy******
    nwidghji            n*i*****
    orbzsfgz            o**z****
    hhgrhwth            h***h***
    zzzzzozo            z****o**
    iwdfrgng            i*****n*
    yyyygggg            y******g
```

直接搜索,看代码：

```
#include <iostream>
#include <cstdio>
using namespace std;
int a[101][101], n;
char c[101][101], b[8] = {"yizhong"};
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
```

```
            scanf(" %c", &c[i][j]);
        }
    }
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            if (c[i][j] == 'y')
            {
                if (j + 6 < n)
                {
                    int k = 0;
                    for (; k <= 6; ++k)
                    {
                        if (c[i][j + k] != b[k])
                            break;
                    }
                    if (k == 7)
                    {
                        for (int v = 0; v <= 6; ++v)
                        {
                            a[i][j + v] = 1;
                        }
                    }
                    if (i + 6 < n)
                    {
                        int k = 0;
                        for (; k <= 6; ++k)
                        {
                            if (c[i + k][j + k] != b[k])
                                break;
                        }
                        if (k == 7)
                        {
                            for (int v = 0; v <= 6; ++v)
                            {
                                a[i + v][j + v] = 1;
                            }
                        }
                    }
                }
                if (i + 6 < n)
                {
                    int k = 0;
                    for (; k <= 6; ++k)
                    {
                        if (c[i + k][j] != b[k])
                            break;
                    }
                    if (k == 7)
                    {
                        for (int v = 0; v <= 6; ++v)
                        {
                            a[i + v][j] = 1;
                        }
                    }
                    if (j - 6 >= 0)
                    {
                        int k = 0;
                        for (; k <= 6; ++k)
                        {
```

```
                            if (c[i + k][j - k] != b[k])
                                break;
                        }
                        if (k == 7)
                        {
                            for (int v = 0; v <= 6; ++v)
                            {
                                a[i + v][j - v] = 1;
                            }
                        }
                    }
                }
                if (j - 6 >= 0)
                {
                    int k = 0;
                    for (; k <= 6; ++k)
                    {
                        if (c[i][j - k] != b[k])
                            break;
                    }
                    if (k == 7)
                    {
                        for (int v = 0; v <= 6; ++v)
                        {
                            a[i][j - v] = 1;
                        }
                    }
                    if (i - 6 >= 0)
                    {
                        int k = 0;
                        for (; k <= 6; ++k)
                        {
                            if (c[i - k][j - k] != b[k])
                                break;
                        }
                        if (k == 7)
                        {
                            for (int v = 0; v <= 6; ++v)
                            {
                                a[i - v][j - v] = 1;
                            }
                        }
                    }
                }
                if (i - 6 >= 0)
                {
                    int k = 0;
                    for (; k <= 6; ++k)
                    {
                        if (c[i - k][j] != b[k])
                            break;
                    }
                    if (k == 7)
                    {
                        for (int v = 0; v <= 6; ++v)
                        {
                            a[i - v][j] = 1;
                        }
                    }
                    if (j + 6 < n)
                    {
```

```
                        int k = 0;
                        for (; k <= 6; ++k)
                        {
                            if (c[i - k][j + k] != b[k])
                                break;
                        }
                        if (k == 7)
                        {
                            for (int v = 0; v <= 6; ++v)
                            {
                                a[i - v][j + v] = 1;
                            }
                        }
                    }
                }
            }
        }
    }
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            printf("%c", a[i][j] ? c[i][j] : '*');
        }
        printf("\n");
    }
    return 0;
}
```

19. A-B 数对 [二分查找/哈希/排序] [普及-]

P1102 A-B 数对

出题是一件痛苦的事情!

相同的题目看多了也会有审美疲劳，于是我舍弃了大家所熟悉的 A+B Problem，改用 A-B 了哈哈!

好吧，题目是这样的：给出一串数以及一个数字 C，要求计算出所有$A-B=C$ 的数对的个数（不同位置的数字一样的数对算不同的数对）。

关键点：

看lower_bound和uppper_bound的使用：

```
void solve()
{
    sort(A, A + n);
    long long res = 0;
    for (int i = 0; i < n; ++i)
    {
        int q = A[i] + c;
        res += upper_bound(A, A + n, q) - lower_bound(A, A + n, q);
    }
    printf("%lld\n",res);
}
```

20. 删数问题 [字符串/贪心] [普及-/提高]

P1106 删数问题

题目描述：

键盘输入一个高精度的正整数 N*N*（不超过 250 位），去掉其中任意 *k* 个数字后剩下的数字按原左右次序将组成一个新的非负整数。编程对给定的 *N* 和 *k*，寻找一种方案使得剩下的数字组成的新数最小。

关键点

贪心

在*0~k*之间选择第1个数，假设下标为p。

在p~k+1之间选择第2个数，假设下标为p。

在p~k+2之间选择第3个数，假设下标为p。

...

```
#include <iostream>
#include <cstdio>
#include <string>
using namespace std;
int n, k, a[251];
string num;
int main()
{
    cin >> num >> k;
    n = num.size();
    int m = n - k, p = -1;
    for (int i = 0; i < m; ++i)
    {
        int mins = 10;
        for (int j = p + 1; j <= k + i; ++j)
        {
            if (a[j] == 0 && num[j] - '0' < mins)
            {
                mins = num[j] - '0';
                p = j;
                if (mins == 0)
                    break;
            }
        }
        a[p] = 1;
    }
    int f = 0;
    for (int i = 0; i < n; ++i)
    {
        if (a[i] == 1)
        {
            if (f == 0 && num[i] != '0')
            {
                printf("%c", num[i]);
                f = 1;
```

```
            }
            else if (f == 1)
                printf("%c", num[i]);
        }
    }
    if (f == 0)
        printf("0");
    printf("\n");
    return 0;
}
```

21. 最大子段和 [dp] [普及-]

[P1115 最大子段和

题目描述：

给出一个长度为 n*n* 的序列 a*a*，选出其中连续且非空的一段使得这段和最大。

关键点：

$$f[i]=max(f[i-1]+n[i],n[i]) \tag{12}$$

f[n]的值并不一定是最终结果

所以，我们还要再用一个数从1到n再查找一次，才能找出最大数！！！

```
    for(int i=1;i<=n;i++)
    {
        ans[i]=max(ans[i-1]+n[i],n[i]);//DP
        sum=max(sum,ans[i]);//取最大值也同时进行，节约时间
    }
    cout<<sum;//直接输出
```

22. 奇怪的电梯 [bfs] [普及/提高-]

P1135 奇怪的电梯

题目描述：

呵呵，有一天我做了一个梦，梦见了一种很奇怪的电梯。大楼的每一层楼都可以停电梯，而且第*i*层楼(1≤*i*≤*N*)上有一个数字$K_i(0 \leq K_i \leq N)$。电梯只有四个按钮：开，关，上，下。上下的层数等于当前楼层上的那个数字。当然，如果不能满足要求，相应的按钮就会失灵。例如：3,3,1,2,5代表了$K_i(K_1=3,K_2=3,\ldots)$，从1楼开始。在1楼，按“上”可以到4楼，按“下”是不起作用的，因为没有-2楼。那么，从*A*楼到*B*楼至少要按几次按钮呢?

广度优先搜索

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

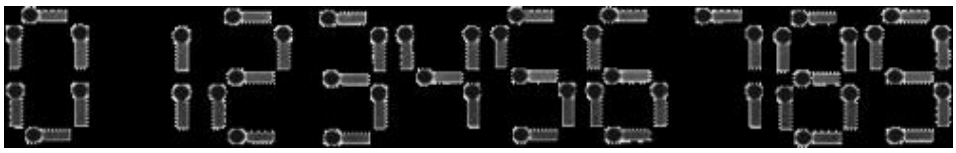
```cpp
int n, a, b, h[201], cnt[201];
queue<int> que;
int main()
{
    scanf("%d%d%d", &n, &a, &b);
    for (int i = 1; i <= n; ++i)
    {
        scanf("%d", &h[i]);
    }
    for (int i = 1; i <= n; ++i)
    {
        cnt[i] = 2147483647;
    }
    cnt[a] = 0;
    que.push(a);
    while (!que.empty())
    {
        int u = que.front();
        que.pop();
        if (u + h[u] <= n && cnt[u + h[u]] > cnt[u] + 1)
        {
            cnt[u + h[u]] = cnt[u] + 1;
            que.push(u + h[u]);
        }
        if (u - h[u] >= 1 && cnt[u - h[u]] > cnt[u] + 1)
        {
            cnt[u - h[u]] = cnt[u] + 1;
            que.push(u - h[u]);
        }
    }
    if (cnt[b] == 2147483647)
        cnt[b] = -1;
    printf("%d\n", cnt[b]);
    return 0;
}
```

23. 火柴棒等式 [搜索] [普及-]

P1149 [NOIP2008 提高组] 火柴棒等式

描述：

给你n根火柴棍，你可以拼出多少个形如“A+B=C”*的等式？等式中的*A、B、C是用火柴棍拼出的整数（若该数非零，则最高位不能是0）。用火柴棍拼数字0-9的拼法如图所示：



注意：

1. 加号与等号各自需要两根火柴棍
2. 如果A≠B，则*A*+*B*=*C*与*B*+*A*=*C*视为不同的等式(*A*,*B*,*C*>=0)
3. *n*根火柴棍必须全部用上

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
int n, a[2001] = {6, 2, 5, 5, 4, 5, 6, 3, 7, 6},res;
int main()
{
    scanf("%d", &n);
    n -= 4;
    for (int i = 10; i < 2000; ++i)
    {
        a[i] = a[i / 10] + a[i % 10];//求每个数所用的火柴棒
    }
    for (int i = 0; i < 1000; ++i)
    {
        for (int j = 0; j < 1000; ++j)
        {
            if (a[i] + a[j] + a[i + j] == n)
            {
                ++res;
            }
        }
    }
    printf("%d\n", res);
    return 0;
}
```

24.组合的输出 [搜索/递归/排列组合] [普及-]

P1157 组合的输出

题目描述：

排列与组合是常用的数学方法，其中组合就是从nn个元素中抽出rr个元素(不分顺序且r \le n)$r \le n$)，我们可以简单地将nn个元素理解为自然数1,2,…,n，从中任取rr个数。

现要求你输出所有组合。

例如n=5,r*=3，所有组合为：

123,124,125,134,135,145,234,235,245,345

模板题，上代码：

```cpp
#include <iostream>
using namespace std;
int n, k, a[1000], b[1000];
void dfs(int cnt, int s)
{
    if (cnt == k)
    {
        for (int i = 0; i < k; ++i)
        {
            printf("%3d",b[i]);
        }
        printf("\n");
    }
```

```
    else
    {
        for (int i = s; i < n; ++i)
        {
            b[cnt] = a[i];
            dfs(cnt + 1, i + 1);
        }
    }
}
int main()
{
    scanf("%d%d",&n,&k);
    for (int i = 0; i < n; ++i)
    {
        a[i] = i + 1;
    }
    dfs(0, 0);
    system("pause");
    return 0;
}
```

25. 填涂颜色 [bfs/队列] [普及-]

P1162 填涂颜色

题目描述：

由数字00组成的方阵中，有一任意形状闭合圈，闭合圈由数字1构成，围圈时只走上下左右4个方向。现要求把闭合圈内的所有空间都填写成2.例如：6×6的方阵（n=6），涂色前和涂色后的方阵如下：

```
0 0 0 0 0 0
0 0 1 1 1 1
0 1 1 0 0 1
1 1 0 0 0 1
1 0 0 0 0 1
1 1 1 1 1 1
```

```
0 0 0 0 0 0
0 0 1 1 1 1
0 1 1 2 2 1
1 1 2 2 2 1
1 2 2 2 2 1
1 1 1 1 1 1
```

看代码：

```
#include <iostream>
#include <cstdio>
#include <queue>
#include <vector>
using namespace std;
struct A
{
    int x, y;
```

```
    A(int _x, int _y)
    {
        x = _x;
        y = _y;
    }
};
int n, a[31][31], used[31][31], dx[4] = {0, 0, 1, -1}, dy[4] = {1, -1, 0, 0};
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            scanf("%d", &a[i][j]);
        }
    }
    queue<A> que;
    vector<A> v;
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            if (a[i][j] == 0)
            {
                que.push(A(i, j));
                used[i][j] = 1;
                int f = 0;
                while (!que.empty())
                {
                    A u = que.front();
                    que.pop();
                    v.push_back(u);
                    int x = u.x;
                    int y = u.y;
                    for (int k = 0; k < 4; ++k)
                    {
                        int x1 = x + dx[k];
                        int y1 = y + dy[k];
                        if (x1 >= 0 && x1 < n && y1 >= 0 && y1 < n && a[x1][y1]
== 0 && !used[x1][y1])
                        {
                            que.push(A(x1, y1));
                            used[x1][y1] = 1;
                            if (x1 == 0 || x1 == n - 1 || y1 == 0 || y1 == n - 1)
                                f = 1;
                        }
                    }
                }
                int p = 2;
                if (f)
                    p = -2;
                for (int k = 0; k < v.size(); ++k)
                {
                    A u = v[k];
                    a[u.x][u.y] = p;
                }
                v.clear();
            }
        }
    }
```

```
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            printf("%d%c", a[i][j] == -2 ? 0 : a[i][j], j == n - 1 ? '\n' : ' ');
        }
    }
    return 0;
}
```

26. 小A点菜 [动态规划/背包] [普及-]

P1164 小A点菜

题目描述:

uim由于买了一些辅（e）辅（ro）书，口袋里只剩*M*元(*M*≤10000)。

餐馆虽低端，但是菜品种类不少，有N*种*(N≤*100*)，*第*i*种卖a_i元（$a_i \leq 1000$）。由于是很低端的餐馆，所以每种菜只有一份。

小A奉行“不把钱吃光不罢休”，所以他点单一定刚好吧uim身上所有钱花完。他想知道有多少种点菜方法。

由于小A肚子太饿，所以最多只能等待1秒。

```
#include <iostream>
using namespace std;
int dp[101][10001], v[101], n, m;
int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; ++i)
    {
        cin >> v[i];
    }
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= m; ++j)
        {
            if (j > v[i])
            {
                dp[i][j] = dp[i - 1][j - v[i]] + dp[i - 1][j];
            }
             if (j == v[i])
            {
                dp[i][j] = dp[i - 1][j] + 1;
            }
            if(j<v[i])
            {
                dp[i][j] = dp[i - 1][j];
            }
        }
    }
    cout << dp[n][m] << endl;
    return 0;
```

```
}
```

27. 数列分段 Section II [二分/最小化最大值/贪心/前缀和] [普及/提高-]

P1182 数列分段 Section II

题目描述：

对于给定的一个长度为N的正整数数列 $A_{1\sim N}$，现要将其分成 M（$M\leq N$）段，并要求每段连续，且每段和的最大值最小。

```
// logu p2884 p1182
int N, M, MAX = 0, A[100000];
bool judge(long long mid)
{
    long long sum = 0;
    int cnt = 0;
    for (int i = 0; i < N; ++i)
    {
        sum += A[i];
        if (sum > mid)
        {
            sum = A[i];
            ++cnt;
        }
    }
    ++cnt; //不要漏掉了这个，
    if (cnt <= M)
        return true;
    return false;
}

int main()
{
    N = read();
    M = read();
    for (int i = 0; i < N; ++i)
    {
        A[i] = read();
        MAX = A[i] > MAX ? A[i] : MAX;
    }
    long long lb = MAX - 1, ub = 10e13+1, mid = 0;
    while (ub - lb > 1)
    {
        mid = (lb + ub) >> 1;
        if (judge(mid))
        {
            ub = mid;
        }
        else
        {
            lb = mid;
        }
    }
    printf("%lld\n", ub);
    return 0;
```

```
}
```

28. 混合牛奶 [贪心/排序] [普及-]

P1208 [USACO1.3]混合牛奶 Mixing Milk

题目描述：

由于乳制品产业利润很低，所以降低原材料（牛奶）价格就变得十分重要。帮助 Marry 乳业找到最优的牛奶采购方案。

Marry 乳业从一些奶农手中采购牛奶，并且每一位奶农为乳制品加工企业提供的价格是不同的。此外，就像每头奶牛每天只能挤出固定数量的奶，每位奶农每天能提供的牛奶数量是一定的。每天 Marry 乳业可以从奶农手中采购到小于或者等于奶农最大产量的整数数量的牛奶。

给出 Marry 乳业每天对牛奶的需求量，还有每位奶农提供的牛奶单价和产量。计算采购足够数量的牛奶所需的最小花费。

注：每天所有奶农的总产量大于 Marry 乳业的需求量。

输入格式

第一行二个整数 *n*,*m*，表示需要牛奶的总量，和提供牛奶的农民个数。

接下来 *m* 行，每行两个整数 p_i, a_i，表示第 *i* 个农民牛奶的单价，和农民 *ii* 一天最多能卖出的牛奶量。

输出格式

单独的一行包含单独的一个整数，表示 Marry 的牛奶制造公司拿到所需的牛奶所要的最小费用。

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int n;
int t;
struct A
{
    int dj = 0, w = 0;
};
A a[5001];
int m, v, ans = 0;
bool cmp(const A &a1, const A &a2)
{
    return a1.dj < a2.dj;
}
int main()
{
    scanf("%d%d", &t, &n);
    for (int i = 0; i < n; ++i)
    {
        scanf("%d%d", &a[i].dj, &a[i].w);
    }
    sort(a, a + n, cmp);
    int i = 0;
    while (t > 0 && i < n)
    {
        m = t > a[i].w ? a[i].w : t;
```

```
        ans += m * a[i].dj;
        ++i;
        t -= m;
    }
    printf("%d\n", ans);
    return 0;
}
```

29. 数字三角形 [dp] [普及-]

P1216 [USACO1.5][IOI1994]数字三角形 Number Triangles

题目描述：

观察下面的数字金字塔。

写一个程序来查找从最高点到底部任意处结束的路径，使路径经过数字的和最大。每一步可以走到左下方的点也可以到达右下方的点。

```
        7
      3   8
    8   1   0
  2   7   4   4
4   5   2   6   5
```

在上面的样例中,从7→3→8→7→5 的路径产生了最大

```
#include <iostream>
using namespace std;
int dp[1001][1001], a[1001][1001];
int n;
#define MAX(x, y) (x) > (y) ? (x) : (y)
int res = 0;
int main()
{
    cin >> n;
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= i; ++j)
        {
            cin >> a[i][j];
        }
    }
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= i; ++j)
        {
            dp[i][j] = MAX(dp[i - 1][j - 1] + a[i][j], dp[i - 1][j] + a[i][j]);
            res = MAX(dp[i][j], res);
        }
    }
    cout << res << endl;
    return 0;
}
```

30. 回文质数 [素数] [普及-]

P1217 [USACO1.5]回文质数 Prime Palindromes

题目描述：

因为 151 既是一个质数又是一个回文数（从左到右和从右到左是看一样的），所以 151 是回文质数。

写一个程序来找出范围 [a,b] (5≤*a*<*b*≤100,000,000)(一亿)间的所有回文质数。

关键点：

首先是回文数的判断方法：

```
bool pd_h(int x)
{

    int y=x,num=0;//int y=x,防止x被改变
    while (y!=0)
    {
        num=num*10+y%10;//上一次数字的记录进位再加上下一位数
        y/=10;
    }
    if (num==x) return 1;
    else return 0;
}
```

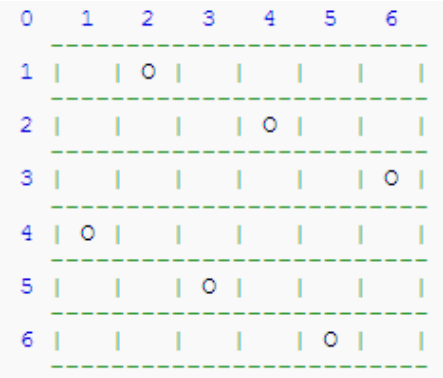
- 偶数位数回文数（除11）必定不是质数（自行百度），所以只要运行到10000000。
- 偶数肯定不是质数。这样至少排除一半多的数据量。
- 你回文数已经判断出来了，在此中判断即可。

31.八皇后 [dfs] [普及/提高-]

P1219 [USACO1.5]八皇后 Checker Challenge

题目描述：

一个如下的 6×6 的跳棋棋盘，有六个棋子被放置在棋盘上，使得每行、每列有且只有一个，每条对角线（包括两条主对角线的所有平行线）上至多有一个棋子。



上面的布局可以用序列 2 4 6 1 3 5 来描述，第 *i* 个数字表示在第 *i* 行的相应位置有一个棋子，如下：

行号 1 2 3 4 5 6

列号 2 4 6 1 3 5

这只是棋子放置的一个解。请编一个程序找出所有棋子放置的解。
并把它们以上面的序列方法输出，解按字典顺序排列。
请输出前 3 个解。最后一行是解的总个数。

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
int n, a[14][14], cnt = 0;
bool check(int r, int c)
{
    for (int i = 1; i <= r; ++i)
    {
        if (a[i][c])
            return false;
    }
    for (int j = 1; j <= n; ++j)
    {
        if (a[r][j])
            return false;
    }
    int i = r, j = c;
    while (i >= 1 && j >= 1)
    {
        if (a[i][j])
            return false;
        --i;
        --j;
    }
    i = r, j = c;
    while (i >= 1 && j <= n)
    {
        if (a[i][j])
            return false;
        --i;
        ++j;
    }
    return true;
}
void dfs(int r)
{
    if (r == n + 1)
    {
        ++cnt;
        if (cnt >= 4)
            return;
        for (int i = 1; i <= n; ++i)
        {
            for (int j = 1; j <= n; ++j)
            {
                if (a[i][j])
                    printf("%d%c", j, i == n ? '\n' : ' ');
            }
        }
    }
    for (int j = 1; j <= n; ++j)
    {
        if (check(r, j))
```

```
        {
            a[r][j] = 1;
            dfs(r + 1);
            a[r][j] = 0;
        }
    }
}
int main()
{
    scanf("%d", &n);
    dfs(1);
    printf("%d\n", cnt);
    return 0;
}
```

32. 排队接水 [贪心] [普及-]

P1223 排队接水

题目描述：

有 n 个人在一个水龙头前排队接水，假如每个人接水的时间为 T_i*，请编程找出这 n 个人排队的一种顺序，使得 n 个人的平均等待时间最小。

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int n;
struct A
{
    int v, in;
};
A t[1001];
double ans;
bool cmp(const A &a1, const A &a2)
{
    if (a1.v == a2.v)
        return a1.in < a2.in;
    return a1.v < a2.v;
}
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; ++i)
    {
        scanf("%d", &t[i].v);
        t[i].in = i + 1;
    }
    sort(t, t + n, cmp);
    for (int i = 0; i < n; ++i)
    {
        printf("%d%c", t[i].in, i == n - 1 ? '\n' : ' ');
        ans += (double)(n - i-1) * t[i].v;
    }
    printf("%.2lf\n", ans / n);
```

```
    return 0;
}
```

33. 快速幂||取余运算 [递推/递归] [普及-]

P1226 【模板】快速幂||取余运算

题目描述：

给你三个整数 b,p,k*，求 $b^p \mod k$。

```
#include <iostream>
using namespace std;
int main()
{
    long long b, p, k, ans = 1;
    cin >> b >> p >> k;
    long long b1 = b, p1 = p;
    if (p == 0)
    {
        ans = 0;
    }
    else
    {
        while (p)
        {
            if (p & 1)
                ans = (ans * b) % k;
            b = (b * b) % k;
            p >>= 1;
        }
    }
    cout << b1 << "^" << p1 << " mod " << k << "=" << ans << endl;
    return 0;
}
```

34. 数楼梯 [斐波那契/递推/高精度] [普及-]

P1255 数楼梯

题目描述：

楼梯有 N*N* 阶，上楼可以一步上一阶，也可以一步上二阶。

编一个程序，计算共有多少种不同的走法。

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int n, f[5001][5000], len[5001];
```

```
void add(int x, int y, int z)
{
    len[z] = max(len[x], len[y]);
    int jw = 0;
    for (int i = 0; i < len[z]; ++i)
    {
        if (f[z][i])
            cout << "f[" << z << "][" << i << "] " << f[z][i] << endl;
        f[z][i] = f[x][i] + f[y][i] + jw;
        jw = f[z][i] / 10;
        f[z][i] %= 10;
    }
    if (jw)
    {
        // if (jw > 1)
        //     cout << x << "  jw>1" << endl;
        f[z][len[z]++] = jw;
    }
}
int main()
{
    scanf("%d", &n);
    f[1][0] = 1;
    f[2][0] = 2;
    len[1] = len[2] = 1;
    for (int i = 3; i <= n; ++i)
    {
        add(i - 2, i - 1, i);
    }
    for (int i = len[n] - 1; i >= 0; --i)
    {
        printf("%d", f[n][i]);
    }
    if (n == 0)
        printf("0");
    printf("\n");
    return 0;
}
```

35. A*B Problem [高精度] [普及-]

P1303 A*B Problem

题目描述:

求两数的积。

```
#include <iostream>
#include <cstring>
using namespace std;
char s1[2001], s2[2001];
int a[2001], b[2001], c[4001];
int main()
{
    scanf("%s%s", s1, s2);
    int fa = 1, fb = 1, fc = 1;
```

```
    if (s1[0] == '-')
    {
        fa = -1;
        strcpy(s1, &s1[1]);
    }
    if (s2[0] == '-')
    {
        fb = -1;
        strcpy(s2, &s2[1]);
    }
    if ((fa == -1 && fb == 1) || (fa == 1 && fb == -1))
        fc = -1;
    int lena = strlen(s1), lenb = strlen(s2);
    for (int i = 0; i < lena; ++i)
    {
        a[i] = s1[lena - i - 1] - 48;
    }
    for (int i = 0; i < lenb; ++i)
    {
        b[i] = s2[lenb - i - 1] - 48;
    }
    int jw = 0;
    for (int i = 0; i < lena; ++i)
    {
        jw = 0;
        for (int j = 0; j < lenb; ++j)
        {
            c[i + j] = c[i + j] + jw + a[i] * b[j];
            jw = c[i + j] / 10;
            c[i + j] %= 10;
        }
        c[i + lenb] += jw;
    }
    int lenc = lena + lenb;
    while (lenc > 1 && !c[lenc - 1])
        --lenc;
    if (fc == -1)
        printf("-");
    for (int i = lenc - 1; i >= 0; --i)
    {
        printf("%d", c[i]);
    }
    printf("\n");
    return 0;
}
```

36. 寻找段落 [二分/最大化平均值] [普及+/提高]

P1419 寻找段落

题目描述：

给定一个长度为 n 的序列 a，定义 a_i 为第 i 个元素的价值。现在需要找出序列中最有价值的“段落”。段落的定义是长度在 $[S,T]$ 之间的连续序列。最有价值段落是指平均值最大的段落。

段落的平均值 等于 **段落总价值** 除以 **段落长度。**

```cpp
#include <iostream>
#include <queue>
using namespace std;
int S, T, n, a[100001];
double b[100001], sum[100001] = {0};
bool judge(double x)
{
    for (int i = 1; i <= n; ++i)
    {
        b[i] = a[i - 1] - x;
        sum[i] = sum[i - 1] + b[i];
    }
    deque<int> dq;
    double maxs = -2e9;
    dq.push_back(0);
    for (int i = S; i <= n; ++i)
    {
        while (!dq.empty() && i - T > dq.front())
        {
            dq.pop_front();
        }
        maxs = maxs > (sum[i] - sum[dq.front()]) ? maxs : (sum[i] -
sum[dq.front()]);
        if (maxs >= 0)
            return true;
        while (!dq.empty() && sum[i - S + 1] <= sum[dq.back()])
        {
            dq.pop_back();
        }
        dq.push_back(i - S + 1);
    }
    return false;
}
int main()
{
    n = read();
    S = read();
    T = read();
    for (int i = 0; i < n; ++i)
    {
        a[i] = read();
    }
    double lb = -2e9, ub = 2e9, mid = 0;
    while (ub - lb >= 0.000001)
    {
        mid = (lb + ub) / 2;
        if (judge(mid))
            lb = mid + 0.000001;
        else
            ub = mid - 0.000001;
    }
    printf("%.3f\n", mid);
    return 0;
}
```

37. 马的遍历 [bfs] [普及/提高-]

P1443 马的遍历

题目描述：

有一个n*m的棋盘(1<n,m<=400)，在某个点上有一个马,要求你计算出马到达棋盘上任意一个点最少要走几步。

```
#include <iostream>
#include <cstdio>
#include <queue>
using namespace std;
struct A
{
    int x, y;
    A(int _x, int _y)
    {
        x = _x;
        y = _y;
    }
};
int n, m, a[402][402], h[402][402], x0, y0, cnt;
queue<A> que;
void bfs(int x, int y)
{
    if (!h[x][y])
        h[x][y] = cnt;
    else
        return;
}
int main()
{
    scanf("%d%d%d%d", &n, &m, &x0, &y0);
    que.push(A(x0, y0));
    while (!que.empty())
    {
        A u = que.front();
        que.pop();
        int x = u.x, y = u.y;
        if (x + 2 <= n && y - 1 >= 1 && !a[x + 2][y - 1])
        {
            a[x + 2][y - 1] = a[x][y] + 1;
            que.push(A(x + 2, y - 1));
        }
        if (x + 2 <= n && y + 1 <= m && !a[x + 2][y + 1])
        {
            a[x + 2][y + 1] = a[x][y] + 1;
            que.push(A(x + 2, y + 1));
        }
        if (x - 2 >= 1 && y + 1 <= m && !a[x - 2][y + 1])
        {
            a[x - 2][y + 1] = a[x][y] + 1;
            que.push(A(x - 2, y + 1));
        }
        if (x - 2 >= 1 && y - 1 >= 1 && !a[x - 2][y - 1])
        {
            a[x - 2][y - 1] = a[x][y] + 1;
```

```
            que.push(A(x - 2, y - 1));
        }
        if (x + 1 <= n && y - 2 >= 1 && !a[x + 1][y - 2])
        {
            a[x + 1][y - 2] = a[x][y] + 1;
            que.push(A(x + 1, y - 2));
        }
        if (x + 1 <= n && y + 2 <= m && !a[x + 1][y + 2])
        {
            a[x + 1][y + 2] = a[x][y] + 1;
            que.push(A(x + 1, y + 2));
        }
        if (x - 1 >= 1 && y + 2 <= m && !a[x - 1][y + 2])
        {
            a[x - 1][y + 2] = a[x][y] + 1;
            que.push(A(x - 1, y + 2));
        }
        if (x - 1 >= 1 && y - 2 >= 1 && !a[x - 1][y - 2])
        {
            a[x - 1][y - 2] = a[x][y] + 1;
            que.push(A(x - 1, y - 2));
        }
    }
    a[x0][y0] = 0;
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= m; ++j)
        {
            if (!(i == x0 && j == y0) && !a[i][j])
                a[i][j] = -1;
            printf("%-5d", a[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

38. 包裹快递 [二分/最小化最大值] [普及/提高+]

P1542 包裹快递

题目描述：

小K成功地破解了密文。但是乘车到X国的时候，发现钱包被偷了，于是无奈之下只好作快递员来攒足路费去Orz教主……

一个快递公司要将n个包裹分别送到n个地方，并分配给邮递员小K一个事先设定好的路线，小K需要开车按照路线给的地点顺序相继送达，且不能遗漏一个地点。小K得到每个地方可以签收的时间段，并且也知道路线中一个地方到下一个地方的距离。若到达某一个地方的时间早于可以签收的时间段，则必须在这个地方停留至可以签收，但不能晚于签收的时间段，可以认为签收的过程是瞬间完成的。

为了节省燃料，小K希望在全部送达的情况下，车的最大速度越小越好，就找到了你给他设计一种方案，并求出车的最大速度最小是多少。

输入格式

第1行为一个正整数n，表示需要运送包裹的地点数。

下面n行，第i+1行有3个正整数xi，yi，si，表示按路线顺序给出第i个地点签收包裹的时间段为[xi, yi]，即最早为距出发时刻xi，最晚为距出发时刻yi，从前一个地点到达第i个地点距离为si，且保证路线中xi递增。

可以认为s1为出发的地方到第1个地点的距离，且出发时刻为0。

```
#include <iostream>
#include <cstdio>
#include <iomanip>
using namespace std;
int n, xi, yi, si;
int a[200001][3];
bool judge(long double v)
{
    long double t = 0;
    for (int i = 0; i < n; ++i)
    {
        t = t + a[i][2] / v;
        if (t > a[i][1])
            return false;
        if (t < a[i][0])
            t = a[i][0];
    }
    return true;
}
int main()
{
    n = read();
    for (int i = 0; i < n; ++i)
    {
        a[i][0] = read();
        a[i][1] = read();
        a[i][2] = read();
    }
    long double lb = 0, ub = 3.0*100000001.00, mid = 0.0;
    while (ub - lb > 0.00001)
    {
        mid = (lb + ub) / 2;
        if (judge(mid))
            ub = mid;
        else
            lb = mid;
    }
    cout<<fixed<<setprecision(2)<<mid<<endl;
    return 0;
}
```

39. 哥德巴赫猜想（升级版）[枚举/暴力/素数] [普及-]

P1579 哥德巴赫猜想（升级版）

题目描述：

现在请你编一个程序验证哥德巴赫猜想。

先给出一个奇数n，要求输出3个质数，这3个质数之和等于输入的奇数。

```cpp
#include <iostream>
#include <cmath>
using namespace std;
bool prime(int a);
int main()
{
    int n;
    int h[6000];
    cin >> n;
    int k=0;
    for (int i = 2; i < n; ++i)
    {
        if (prime(i))
        {
            h[k++]=i;
        }
    }
    int f=0;
    int p;
    for(int i=0;i<k;++i){
        for(int j=i;j<k;++j){
            for(int p=k-1;p>=j;--p){
                if(h[i]+h[j]+h[p]==n){
                    cout<<h[i]<<" "<<h[j]<<" "<<h[p]<<endl;
                    f=1;
                    break;
                }
            }
            if(f==1){
                f=2;
                break;
            }
        }
        if(f==2){
            f=3;
            break;
        }
    }
    return 0;
}

bool prime(int a)
{
    if (a % 2 == 0&&a!=2)
    {
        return false;
    }
    int r = int(sqrt(a));
    for (int i = 2; i <= r; ++i)
    {
        if (a % i == 0)
        {
            return false;
        }
    }
    return true;
}
```

40. 阶乘数码 [高精度] [普及-]

P1591 阶乘数码

题目描述：

求 $n!$ 中某个数码出现的次数。

第一行为 t($t \le 10$)，表示数据组数。接下来 t 行，每行一个正整数 ($n \le 1000$) 和数码 a。

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
int n, t, a, v[1000000], vlen, w[1000000], wlen;
int main()
{
    scanf("%d", &t);
    for (int x = 0; x < t; ++x)
    {
        scanf("%d%d", &n, &a);
        v[0] = 1;
        vlen = 1;
        for (int i = 1; i <= n; ++i)
        {
            int jw = 0;
            for (int j = 0; j < vlen; ++j)
            {
                v[j] = v[j] * i + jw;
                jw = v[j] / 10;
                v[j] %= 10;
            }
            while (jw > 0)
            {
                v[vlen++] = jw % 10;
                jw /= 10;
            }
        }
        int ans = 0;
        for (int i = 0; i < vlen; ++i)
        {
            if (v[i] == a)
            {
                ++ans;
                v[i] = 0;
            }
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

41. Lake Counting S [dfs] [普及-]

P1596 [USACO10OCT]Lake Counting S

题目描述：

由于近期的降雨，雨水汇集在农民约翰的田地不同的地方。我们用一个NxM(1<=N<=100;1<=M<=100)网格图表示。每个网格中有水('W') 或是旱地('.')。一个网格与其周围的八个网格相连，而一组相连的网格视为一个水坑。约翰想弄清楚他的田地已经形成了多少水坑。给出约翰田地的示意图，确定当中有多少水坑。

输入1：

```
10 12
W........WW.
.WWW.....WWW
....WW...WW.
.........WW.
.........W..
..W......W..
.W.W.....WW.
W.W.W.....W.
.W.W......W.
..W.......W.
```

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
char pond[100][100];
int m, n, res;
void solve();
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < m; ++j)
        {
            cin >> pond[i][j];
        }
    }
    solve();
    return 0;
}

void dfs(int x, int y)
{
    pond[x][y] = '.';
    for (int i = -1; i <= 1; ++i)
    {
        for (int j = -1; j <= 1; ++j)
        {
            int nx = x + i, ny = y + j;
            if (nx >= 0 && nx < n && ny >= 0 && ny < m && pond[nx][ny] == 'W')
            {
                dfs(nx, ny);
            }
        }
```

```
    }
}
void solve()
{
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < m; ++j)
        {
            if (pond[i][j] == 'W')
            {
                dfs(i, j);
                ++res;
            }
        }
    }
    printf("%d\n", res);
}
```

42. 垂直柱状图 [模拟] [普及-]

P1598 垂直柱状图

题目描述：

写一个程序从输入文件中去读取四行大写字母（全都是大写的，每行不超过100个字符），然后用柱状图输出每个字符在输入文件中出现的次数。严格地按照输出样例来安排你的输出格式。

输入格式

四行字符，由大写字母组成，每行不超过100个字符

输出格式

由若干行组成，前几行由空格和星号组成，最后一行则是由空格和字母组成的。在任何一行末尾不要打印不需要的多余空格。不要打印任何空行。

输入1：

```
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
THIS IS AN EXAMPLE TO TEST FOR YOUR
HISTOGRAM PROGRAM.
HELLO!
```

输出1：

```
                            *
                            *
        *                   *
        *                   *     *   *
        *                   *     *   *
*       *     *             *     *   *
*       *     * *     * *   *     * * *
*       *   * * *     * *   * *   * * * *
*     * * * * * *     * * * * *   * * * *     * *
* * * * * * * * * * * * * * * * * * * * * * * * * *
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

```cpp
#include <iostream>
#include <string>

using namespace std;

int max(int m[26]);

int main() {
    string string1[4];
    int m[26] = {0};
    for (int i = 0; i < 4; ++i) {
        getline(cin, string1[i]);
    }
    for (int j = 0; j < 4; ++j) {
        for (int i = 0; i < string1[j].length(); ++i) {
            for (int k = 0; k < 26; ++k) {
                if (string1[j][i] == (k + 65)) {
                    m[k]++;
                }
            }
        }
    }
    int max1 = max(m);
    for (int l = 0; l < max1; ++l) {
        for (int i = 0; i < 26; ++i) {
            if (max1 - l - m[i] <= 0) {
                cout << "* ";
            } else {
                cout << "  ";
            }
        }
        cout << endl;
    }
    for (int n = 0; n < 26; ++n) {
        if (n != 25) {
            cout << (char) (n + 65) << " ";
        } else {
            cout << (char) (n + 65);
        }
    }
    return 0;
}

int max(int m[26]) {
    int t = m[0];
    for (int i = 0; i < 26; ++i) {
        if (m[i] > t) {
            t = m[i];
        }
    }
    return t;
}
```

43. A+B Problem(高精) [高精度] [普及-]

P1601 A+B Problem（高精）

题目描述：

高精度加法，相当于a+b problem，**不用考虑负数**.

```cpp
#include <iostream>
#include <cstring>
using namespace std;
char s1[5001], s2[5001];
int a[5001], b[5001], c[10001];
int main()
{
    scanf("%s%s", s1, s2);
    int lena = strlen(s1);
    int lenb = strlen(s2);
    for (int i = 0; i < lena; ++i)
    {
        a[i] = s1[lena - i - 1] - 48;
    }
    for (int i = 0; i < lenb; ++i)
    {
        b[i] = s2[lenb - i - 1] - 48;
    }
    int lenc = lena > lenb ? lena : lenb;
    int jw = 0;
    for (int i = 0; i < lenc; ++i)
    {
        c[i] = a[i] + b[i] + jw;
        jw = c[i] / 10;
        c[i] %= 10;
    }
    if (jw)
    {
        lenc++;
        c[lenc - 1] = jw;
    }
    for (int i = lenc - 1; i >= 0; --i)
    {
        printf("%d",c[i]);
    }
    printf("\n");
    return 0;
}
```

44. 迷宫 [dfs/枚举/暴力] [普及-]

P1605 迷宫

题目描述：

给定一个N*M方格的迷宫，迷宫里有T处障碍，障碍处不可通过。给定起点坐标和终点坐标，问: 每个方格最多经过1次，有多少种从起点坐标到终点坐标的方案。在迷宫中移动有上下左右四种方式，每次只能移动一个方格。数据保证起点上没有障碍。

输入格式

第一行N、M和T，N为行，M为列，T为障碍总数。第二行起点坐标SX,SY，终点坐标FX,FY。接下来T行，每行为障碍点的坐标。

输出格式

给定起点坐标和终点坐标，问每个方格最多经过1次，从起点坐标到终点坐标的方案总数。

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
int n, m, t, g[30][30], a[30][30], x, y, x0, y0, x1, y1;
int dfs(int x, int y)
{
    if (x == x1 && y == y1)
        return 1;
    int cnt = 0;
    if (x + 1 <= n && !a[x + 1][y] && !g[x + 1][y])
    {
        a[x + 1][y] = 1;
        cnt += dfs(x + 1, y);
        a[x + 1][y] = 0;
    }
    if (x - 1 >= 1 && !a[x - 1][y] && !g[x - 1][y])
    {
        a[x - 1][y] = 1;
        cnt += dfs(x - 1, y);
        a[x - 1][y] = 0;
    }
    if (y + 1 <= m && !a[x][y + 1] && !g[x][y + 1])
    {
        a[x][y + 1] = 1;
        cnt += dfs(x, y + 1);
        a[x][y + 1] = 0;
    }
    if (y - 1 >= 1 && !a[x][y - 1] && !g[x][y - 1])
    {
        a[x][y - 1] = 1;
        cnt += dfs(x, y - 1);
        a[x][y - 1] = 0;
    }
    return cnt;
}
int main()
{
    scanf("%d%d%d", &n, &m, &t);
    scanf("%d%d%d%d", &x0, &y0, &x1, &y1);
    for (int i = 0; i < t; ++i)
    {
        scanf("%d%d", &x, &y);
        g[x][y] = 1;
    }
    a[x0][y0] = 1;//不能忘记写这句了。
    printf("%d\n", dfs(x0, y0));
    return 0;
}
```

45.烦恼的高考志愿 [二分/贪心] [普及-]

P1678 烦恼的高考志愿

题目描述：

现有 m(m*≤100000) 所学校，每所学校预计分数线是 a_i $(a_i \leq 106)$。有 n(n≤100000) 位学生，估分分别为 $b_i(b_i \leq 106)$。

根据n位学生的估分情况，分别给每位学生推荐一所学校，要求学校的预计分数线和学生的估分相差最小（可高可低，毕竟是估分嘛），这个最小值为不满意度。求所有学生不满意度和的最小值。

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int n, m, f[100001], x[100001];
long long ans = 0;
int solve(int x1)
{
    int l = 0, r = m, mid;
    while (l < r - 1)
    {
        mid = (l + r) / 2;
        if (f[mid] == x1)
            return 0;
        else if (f[mid] > x1)
            r = mid;
        else
            l = mid;
    }
    return abs(x1 - f[l]) > abs(x1 - f[r]) ? abs(x1 - f[r]) : abs(x1 - f[l]);
}
int main()
{
    scanf("%d%d", &m, &n);
    for (int i = 0; i < m; ++i)
    {
        scanf("%d", &f[i]);
    }
    for (int i = 0; i < n; ++i)
    {
        scanf("%d", &x[i]);
    }
    sort(f, f + m);
    for (int i = 0; i < n; ++i)
    {
        ans += solve(x[i]);
    }
    printf("%lld\n",ans);
    return 0;
}
```

46.全排列问题 [dfs] [普及-]

P1706 全排列问题

题目描述：

输出自然数 1 到 n 所有不重复的排列，即 n 的全排列，要求所产生的任一数字序列中不允许出现重复的数字。

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
int n, a[10], b[10], used[10];
void dfs(int k)
{
    if (k == n)
    {
        for (int i = 0; i < n; ++i)
        {
            printf("%5d", b[i]);
        }
        printf("\n");
    }
    else
    {
        for (int i = 0; i < n; ++i)
        {
            if (!used[i])
            {
                used[i] = 1;
                b[k] = a[i];
                dfs(k + 1);
                used[i] = 0;
            }
        }
    }
}
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; ++i)
    {
        a[i] = i + 1;
    }
    dfs(0);
    return 0;
}
```

47. 宇宙总统 [高精度/字符串] [普及-]

P1781 宇宙总统

题目描述

地球历公元 6036 年，全宇宙准备竞选一个最贤能的人当总统，共有 n 个非凡拔尖的人竞选总统，现在票数已经统计完毕，请你算出谁能够当上总统。

输入格式

第一行为一个整数 n，代表竞选总统的人数。

接下来有 n 行，分别为第一个候选人到第 n 个候选人的票数。

输出格式

共两行，第一行是一个整数 m，为当上总统的人的号数。

第二行是当上总统的人的选票。

```cpp
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
pair<string, int> p[21];
int n;
bool cmp(const pair<string, int> &p1, const pair<string, int> &p2)
{
    if (p1.first.size() == p2.first.size())
        return p1.first > p2.first;
    return p1.first.length() > p2.first.length();
}
int main()
{
    cin >> n;
    for (int i = 0; i < n; ++i)
    {
        cin >> p[i].first;
        p[i].second = i + 1;
    }
    sort(p, p + n, cmp);
    cout << p[0].second << endl;
    cout << p[0].first << endl;
    return 0;
}
```

48. 5倍经验日 [dp] [普及-]

P1802 5倍经验日

题目描述:

现在absi2011拿出了x个迷你装药物(嗑药打人可耻....)，准备开始与那些人打了

由于迷你装一个只能管一次，所以absi2011要谨慎的使用这些药，悲剧的是，没到达最少打败该人所用的属性药了他打人必输>.<所以他用2个药去打别人，别人却表明3个药才能打过，那么相当于你输了并且这两个属性药浪费了。

现在有n个好友，有输掉拿的经验、赢了拿的经验、要嗑几个药才能打过。求出最大经验（注意，最后要乘以5）。

输入格式

第一行两个数，n和x

后面n行每行三个数，分别表示输了拿到的经验(lose[i])、赢了拿到的经验(win[i])、打过要至少使用的药数量(use[i])。

输出格式

一个整数，最多获得的经验

```
#include <iostream>
using namespace std;
long long dp[1001][1001];
int lose[1001], win[1001], use[1001], n, x;
#define MAX(x, y) (x) > (y) ? (x) : (y)
int main()
{
    cin >> n >> x;
    for (int i = 1; i <= n; ++i)
    {
        cin >> lose[i] >> win[i] >> use[i];
    }
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 0; j <= x; ++j)
        {
            if (j >= use[i])
            {
                dp[i][j] = MAX(dp[i - 1][j] + lose[i], win[i] + dp[i - 1][j - use[i]]);
            }
            else
            {
                dp[i][j] = dp[i - 1][j] + lose[i];
            }
        }
    }
    cout<<dp[n][x]*5<<endl;
    return 0;
}
```

49. 线段覆盖 [贪心] [普及-]

P1803 凌乱的yyy / 线段覆盖

题目描述：

现在各大 oj 上有 *n* 个比赛，每个比赛的开始、结束的时间点是知道的。

yyy 认为，参加越多的比赛，noip 就能考的越好（假的）。

所以，他想知道他最多能参加几个比赛。

由于 yyy 是蒟蒻，如果要参加一个比赛必须善始善终，而且不能同时参加 2 个及以上的比赛。

输入格式

第一行是一个整数 *n* ，接下来 *n* 行每行是 2 个整数 $a_i, b_i (a_i < b_i)$，表示比赛开始、结束的时间。

输出格式

一个整数最多参加的比赛数目。

```cpp
#include <iostream>
#include <algorithm>
#include <cstdio>
using namespace std;
int n, ans = 0;
struct A
{
    int s = 0, e = 0;
};
A a[1000001];
bool cmp(const A &a1, const A &a2)
{
    if (a1.e == a2.e)
        return a1.s > a2.s;
    return a1.e < a2.e;
}
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; ++i)
    {
        scanf("%d%d", &a[i].s, &a[i].e);
    }
    sort(a, a + n, cmp);
    int e = 0;
    for (int i = 0; i < n; ++i)
    {
        if (a[i].s >= e)
        {
            e = a[i].e;
            ++ans;
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

50. 砍树 [二分] [普及/提高-]

P1873 砍树

题目描述：

伐木工人米尔科需要砍倒M米长的木材。这是一个对米尔科来说很容易的工作，因为他有一个漂亮的新伐木机，可以像野火一样砍倒森林。不过，米尔科只被允许砍倒单行树木。

米尔科的伐木机工作过程如下：米尔科设置一个高度参数H（米），伐木机升起一个巨大的锯片到高度H，并锯掉所有的树比H高的部分（当然，树木不高于H米的部分保持不变）。米尔科就行到树木被锯下的部分。

例如，如果一行树的高度分别为20，15，10和17，米尔科把锯片升到15米的高度，切割后树木剩下的高度将是15，15，10和15，而米尔科将从第1棵树得到5米，从第4棵树得到2米，共得到7米木材。

米尔科非常关注生态保护，所以他不会砍掉过多的木材。这正是他为什么尽可能高地设定伐木机锯片的原因。帮助米尔科找到伐木机锯片的最大的整数高度H，使得他能得到木材至少为M米。换句话说，如果再升高1米，则他将得不到M米木材。

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int m, n, A[1000000];
void solve();
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 0; i < n; ++i)
    {
        scanf("%d", &A[i]);
    }
    solve();
    return 0;
}
bool query(int h)
{
    long long res = 0;
    int s = upper_bound(A, A + n, h) - A;
    for (int i = s; i < n; ++i)
    {
        res += A[i] - h;
    }
    return res >= m;
}

void solve()
{
    sort(A, A + n);
    int ub = 1000000000, lb = 0;
    while (ub - lb > 1)
    {
        int md = (ub + lb) / 2;
        if (!query(md))
            ub = md;
        else
            lb = md;
    }
    printf("%d\n",lb);
}
```

51. 约瑟夫问题 [模拟/数状数组/队列] [普及-]

P1996 约瑟夫问题

题目描述

n 个人围成一圈，从第一个人开始报数,数到 m 的人出列，再由下一个人重新从 1 开始报数，数到 m 的人再出圈，依次类推，直到所有的人都出圈，请输出依次出圈人的编号。

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
int a[201], n, m;
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; ++i)
    {
        a[i] = i;
    }
    int j = 1, c = 0;
    while (true)
    {
        int i = j, cnt = 0;
        for (; i <= n + 1; ++i)
        {
            if (i > n)
                i = 1;
            if (a[i] != 0)
                ++cnt;
            if (cnt == m)
                break;
        }
        ++c;
        printf("%d%c", i, c == n ? '\n' : ' ');
        a[i] = 0;
        int k = i + 1;
        for (; k <= n + 1; ++k)
        {
            if (k > n)
                k = 1;
            if (a[k] != 0)
                break;
            if (k == i)
                break;
        }
        if (k == i)
            break;
        else
            j = k;
    }
    return 0;
}
```

52. PERKET [dfs] [普及-]

P2036 [COCI2008-2009#2] PERKET

题目描述：

Perket 是一种流行的美食。为了做好 Perket，厨师必须谨慎选择食材，以在保持传统风味的同时尽可能获得最全面的味道。你有 n 种可支配的配料。对于每一种配料，我们知道它们各自的酸度 s 和苦度 b。当我们添加配料时，总的酸度为每一种配料的酸度总乘积；总的苦度为每一种配料的苦度的总和。

众所周知，美食应该做到口感适中，所以我们希望选取配料，以使得酸度和苦度的绝对差最小。

另外，我们必须添加至少一种配料，因为没有任何食物以水为配料的。

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
inline int read()
{
    int x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9')
    {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9')
    {
        x = (x << 3) + (x << 1) + (ch ^ 48);
        ch = getchar();
    }
    return x * f;
}
int n, a[2][11], acid=1, ans = 2147483647, bitter=0;
void dfs(int k)
{
    if (k == n)
    {
        if (acid != 1 && bitter != 0)
        {
            ans = min(ans, abs(acid - bitter));
        }
    }
    else
    {
        dfs(k + 1);
        acid *= a[0][k];
        bitter += a[1][k];
        dfs(k + 1);
        acid /= a[0][k];
        bitter -= a[1][k];
    }
}
int main()
{
    n = read();
    for (int i = 0; i < n; ++i)
    {
        a[0][i] = read();
        a[1][i] = read();
    }
    dfs(0);
    printf("%d\n", ans);
    return 0;
}
```

53. 烤鸡 [dfs] [普及-]

P2089 烤鸡

题目描述

猪猪 Hanke 特别喜欢吃烤鸡（本是同畜牲，相煎何太急！）Hanke 吃鸡很特别，为什么特别呢？因为他有 10 种配料（芥末、孜然等），每种配料可以放 1 到 3 克，任意烤鸡的美味程度为所有配料质量之和。

现在， Hanke 想要知道，如果给你一个美味程度 n ，请输出这 10 种配料的所有搭配方案。

输入格式

一个正整数 n，表示美味程度。

输出格式

第一行，方案总数。

第二行至结束，10 个数，表示每种配料所放的质量，按字典序排列。

如果没有符合要求的方法，就只要在第一行输出一个 0。

输入1：

```
11
```

输出1：

```
10
1 1 1 1 1 1 1 1 1 2
1 1 1 1 1 1 1 1 2 1
1 1 1 1 1 1 1 2 1 1
1 1 1 1 1 1 2 1 1 1
1 1 1 1 1 2 1 1 1 1
1 1 1 1 2 1 1 1 1 1
1 1 1 2 1 1 1 1 1 1
1 1 2 1 1 1 1 1 1 1
1 2 1 1 1 1 1 1 1 1
2 1 1 1 1 1 1 1 1 1
```

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
int n;
int res[10000][10], a[10], ans;
void dfs(int total, int cnt)
{
    if (cnt >= 10)
    {
        if (total == n)
        {
            for (int i = 0; i < 10; ++i)
            {
                res[ans][i] = a[i];
            }
            ans++;
        }
    }
```

```
    else
    {
        for (int i = 1; i <= 3; ++i)
        {
            a[cnt] = i;
            dfs(total + i, cnt + 1);
            a[cnt] = 0;
        }
    }
}
int main()
{
    scanf("%d", &n);
    dfs(0, 0);
    printf("%d\n", ans);
    for (int i = 0; i < ans; ++i)
    {
        for (int j = 0; j < 10; ++j)
        {
            printf("%d%c", res[i][j], j == 9 ? '\n' : ' ');
        }
    }
    return 0;
}
```

54. 统计方形（数据加强版）[枚举/暴力] [普及-]

P2241 统计方形（数据加强版）

题目描述：

有一个 n×m* 方格的棋盘，求其方格包含多少正方形、长方形（不包含正方形）。

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
long long n, m;
long long z, c;
int main()
{
    scanf("%d%d", &n, &m);
    long long q = min(n, m);
    for (int i = 1; i < q; ++i)
    {
        z += i * i;
    }
    z = z + q * m * n - ((q - 1) * q / 2) * n - ((q - 1) * q / 2) * m;
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= m; ++j)
        {
            c += (n - i + 1) * (m - j + 1);
        }
    }
    printf("%lld %lld\n", z, c - z);
    return 0;
}
```

55. 木材加工 [二分/贪心/递归] [普及/提高-]

P2440 木材加工

题目描述：

木材厂有一些原木，现在想把这些木头切割成一些长度相同的小段木头（木头有可能有剩余），需要得到的小段的数目是给定的。当然，我们希望得到的小段木头越长越好，你的任务是计算能够得到的小段木头的最大长度。木头长度的单位是cm。原木的长度都是正整数，我们要求切割得到的小段木头的长度也是正整数。

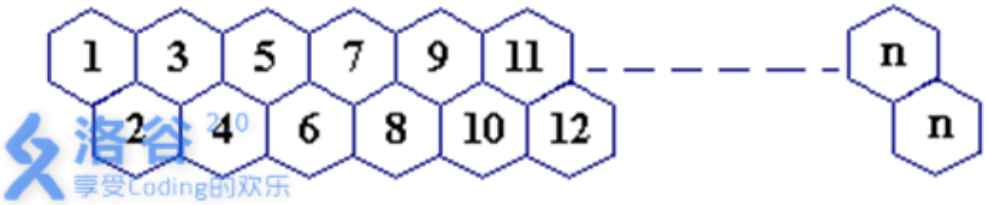
例如有两根原木长度分别为11和21，要求切割成到等长的6段，很明显能切割出来的小段木头长度最长为5.

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int n, k, a[100001];
bool solve(int x)
{
    int c = 0;
    for (int i = 0; i < n; ++i)
    {
        c += a[i] / x;
    }
    return c >= k;
}
int main()
{
    scanf("%d%d", &n, &k);
    for (int i = 0; i < n; ++i)
    {
        scanf("%d", &a[i]);
    }
    int mid, l = 0, r = 200000000;
    while (l < r - 1)
    {
        mid = (l + r) / 2;
        if (solve(mid))
            l = mid;
        else
            r = mid;
    }
    printf("%d\n", l);
    return 0;
}
```

56. 蜜蜂路线 [高精度/递推/斐波那契] [普及-]

题目描述：

一只蜜蜂在下图所示的数字蜂房上爬动,已知它只能从标号小的蜂房爬到标号大的相邻蜂房,现在问你：蜜蜂从蜂房 m 开始爬到蜂房 n，$m<n$，有多少种爬行路线？（备注：题面有误，右上角应为 $n-1$）




```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int n, m, dp[1001][10000], len[1001];
void add(int x, int y, int z)
{
    len[z] = max(len[x], len[y]);
    for (int i = 0; i < len[z]; ++i)
    {
        dp[z][i] += dp[x][i] + dp[y][i];
        dp[z][i + 1] += dp[z][i] / 10;
        dp[z][i] %= 10;
    }
    if (dp[z][len[z]])
        ++len[z];
}
int main()
{
    scanf("%d%d", &m, &n);
    n = n - (m - 1);
    m = 1;
    dp[1][0] = 1;
    dp[2][0] = 1;
    len[1] = len[2] = 1;
    for (int i = 3; i <= n; ++i)
    {
        add(i - 2, i - 1, i);
    }
    for (int i = len[n] - 1; i >= 0; --i)
    {
        printf("%d", dp[n][i]);
    }
    printf("\n");
    return 0;
}
```

57.【模板】矩阵快速幂 [矩阵乘法] [普及/提高-]

P3390 【模板】矩阵快速幂

题目描述：

给定 $n \times n$ 的矩阵 A，求 A_k。

```cpp
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
long long A[101][101], B[101][101], C[101][101];
```

```cpp
long long k;
int n, large = sizeof(A);
const int mod = 1000000007;
void pow(long long k);
int main()
{
    scanf("%d%lld", &n, &k);
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            scanf("%lld", &B[i][j]);
            A[i][j] = 0;
        }
        A[i][i] = 1;
    }
    pow(k);
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            printf("%lld%c", A[i][j], j == n - 1 ? '\n' : ' ');
        }
    }
    return 0;
}
void mul(long long D[101][101], long long S[101][101])
{
    memset(D, 0, large);
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            for (int k = 0; k < n; ++k)
            {
                D[i][j] = (D[i][j] + C[i][k] * S[k][j] % mod) % mod;
            }
        }
    }
}
void pow(long long k)
{
    while (k > 0)
    {
        if (k & 1)
        {
            memcpy(C, A, large);
            mul(A, B);
        }
        memcpy(C, B, large);
        mul(B, C);
        k >>= 1;
    }
}
```

58. 跳石头 [二分/最大化最小值] [普及/提高-]

P2678 [NOIP2015 提高组] 跳石头

题目描述：

这项比赛将在一条笔直的河道中进行，河道中分布着一些巨大岩石。组委会已经选择好了两块岩石作为比赛起点和终点。在起点和终点之间，有 *N* 块岩石（不含起点和终点的岩石）。在比赛过程中，选手们将从起点出发，每一步跳向相邻的岩石，直至到达终点。

为了提高比赛难度，组委会计划移走一些岩石，使得选手们在比赛过程中的最短跳跃距离尽可能长。由于预算限制，组委会至多从起点和终点之间移走 *M* 块岩石（不能移走起点和终点的岩石）。

输入格式

第一行包含三个整数 L,N,M，分别表示起点到终点的距离，起点和终点之间的岩石数，以及组委会至多移走的岩石数。保证 L L≥1 且N≥*M*≥0。

接下来 N行，每行一个整数，第 *i* 行的整数 $D_i(0 < D_i < L)$， 表示第 i*i* 块岩石与起点的距离。这些岩石按与起点距离从小到大的顺序给出，且不会有两个岩石出现在同一个位置。

输出格式

一个整数，即最短跳跃距离的最大值。

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int p, n, m, d[50002];
inline bool solve(int x)
{
    int j = 1, i = 0, cnt = 0;
    while (i < n)
    {
        while (j <= n && d[j] - d[i] < x)
        {
            ++cnt;
            ++j;
        }
        if (j > n)
        {
            if (d[j] - d[i] < x && i != 0)
                ++cnt;
            else if (d[j] - d[i] < x && i == 0)
                return false;
            break;
        }
        else if (j <= n && d[j] - d[i] >= x)
        {
            i = j;
            ++j;
        }
    }
    return cnt <= m;
}
int main()
{
    scanf("%d%d%d", &p, &n, &m);
    for (int i = 1; i <= n; ++i)
    {
        scanf("%d", &d[i]);
    }
```

```
    d[0] = 0;
    d[n + 1] = p;
    int l = 0, r = p * 2;
    while (l < r - 1)
    {
        int mid = (l + r) / 2;
        if (solve(mid))
            l = mid;
        else
            r = mid;
    }
    if(l>p)
        l=p;
    printf("%d\n", l);
    return 0;
}
```

59. Monthly Expense S [二分/最小化最大值] [普及/提高-]

P2884 [USACO07MAR]Monthly Expense S

题目描述：

给出农夫在n天中每天的花费，要求把这n天分作m组，每组的天数必然是连续的，要求分得各组的花费之和应该尽可能地小，最后输出各组花费之和中的最大值

```
#include <iostream>
#include <cstdio>
using namespace std;
int N, M, MAX = 0, A[100000];
inline int read()
{
    int x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9')
    {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9')
    {
        x = (x << 3) + (x << 1) + (ch ^ 48);
        ch = getchar();
    }
    return x * f;
}
bool judge(int mid)
{
    int sum = 0, cnt = 0;
    for (int i = 0; i < N; ++i)
    {
        sum += A[i];
        if (sum > mid)
        {
            sum = A[i];
            ++cnt;
        }
    }
    ++cnt;
    if (cnt <= M)
```

```
        return true;
    return false;
}

int main()
{
    N = read();
    M = read();
    for (int i = 0; i < N; ++i)
    {
        A[i] = read();
        MAX = A[i] > MAX ? A[i] : MAX;
    }
    int lb = MAX - 1, ub = 21000000, mid = 0;
    while (ub - lb > 1)
    {
        mid = (lb + ub) >> 1;
        if (judge(mid))
        {
            ub = mid;
        }
        else
        {
            lb = mid;
        }
    }
    printf("%d\n", ub);
    return 0;
}
```

60. Meteor Shower S [bfs] [普及/提高-]

P2895 [USACO08FEB]Meteor Shower S

题目描述：

贝茜听说了一个骇人听闻的消息：一场流星雨即将袭击整个农场，由于流星体积过大，它们无法在撞击到地面前燃烧殆尽，届时将会对它撞到的一切东西造成毁灭性的打击。很自然地，贝茜开始担心自己的安全问题。以 Farmer John 牧场中最聪明的奶牛的名誉起誓，她一定要在被流星砸到前，到达一个安全的地方（也就是说，一块不会被任何流星砸到的土地）。如果将牧场放入一个直角坐标系中，贝茜现在的位置是原点，并且，贝茜不能踏上一块被流星砸过的土地。 根据预报，一共有 M 颗流星 ($1 \le M \le 50,000$) 会坠落在农场上，其中第i颗流星会在时刻 $T_i (0 \le T_i \le 1,000)$ 砸在坐标为 $(X_i, Y_i) (0 \le X_i \le 300, 0 \le Y_i \le 300)$ 的格子里。流星的力量会将它所在的格子，以及周围 4 个相邻的格子都化为焦土，当然贝茜也无法再在这些格子上行走。

贝茜在时刻 0 开始行动，它只能在第一象限中，平行于坐标轴行动，每 1 个时刻中，她能移动到相邻的（一般是 4 个）格子中的任意一个，当然目标格子要没有被烧焦才行。如果一个格子在时刻 t 被流星撞击或烧焦，那么贝茜只能在 t 之前的时刻在这个格子里出现。 贝西一开始在(0,0)。

请你计算一下，贝茜最少需要多少时间才能到达一个安全的格子。如果不可能到达输出 -1。

```
#include <iostream>
#include <cstdio>
#include <queue>
using namespace std;
struct A
{
    int x, y;
    A(int _x, int _y)
    {
        x = _x;
        y = _y;
```

```
    }
};
int m, x, y, temp, t[305][305], a[305][305], dx[4] = {-1, 0, 0, 1}, dy[4] = {0, 1, -1, 0}, res = 2147483647;
queue<A> que;
bool check(int x, int y) //注意是302而不是300
{
    bool j1 = (t[x][y] == -1) || (t[x][y] != -1 && a[x][y] < t[x][y]);
    bool j2 = x - 1 < 0 || ((x - 1 >= 0) && (t[x - 1][y] == -1 || (t[x - 1][y] != -1 && a[x][y] < t[x - 1][y])));
    bool j3 = x + 1 > 302 || ((x + 1 <= 302) && (t[x + 1][y] == -1 || (t[x + 1][y] != -1 && a[x][y] < t[x + 1][y])));
    bool j4 = y + 1 > 302 || ((y + 1 <= 302) && (t[x][y + 1] == -1 || (t[x][y + 1] != -1 && a[x][y] < t[x][y + 1])));
    bool j5 = y - 1 < 0 || ((y - 1 >= 0) && (t[x][y - 1] == -1 || (t[x][y - 1] != -1 && a[x][y] < t[x][y - 1])));
    return j1 && j2 && j3 && j4 && j5;
}
int main()
{
    scanf("%d", &m);
    for (int i = 0; i <= 302; ++i)
    {
        for (int j = 0; j <= 302; ++j)
        {
            a[i][j] = 2147483646;
            t[i][j] = -1;
        }
    }
    for (int i = 0; i < m; ++i)
    {
        scanf("%d%d%d", &x, &y, &temp);
        t[x][y] = temp;
    }
    a[0][0] = 0;
    que.push(A(0, 0));
    while (!que.empty())
    {
        A u = que.front();
        que.pop();
        for (int i = 0; i < 4; ++i)
        {
            x = u.x + dx[i];
            y = u.y + dy[i];
            if (x >= 0 && x <= 302 && y >= 0 && y <= 302 && a[x][y] > a[u.x][u.y] + 1)
            {
                a[x][y] = a[u.x][u.y] + 1;
                if (check(x, y))
                    que.push(A(x, y));
                else
                    a[x][y] = -1;
            }
        }
    }
    for (int i = 0; i <= 302; ++i)
    {
        for (int j = 0; j <= 302; ++j)
        {
            if (t[i][j] == -1 && a[i][j] != -1)
            {
```

```
                bool j1 = i - 1 < 0 || (i - 1 >= 0 && t[i - 1][j] == -1);
                bool j2 = i + 1 > 302 || (i + 1 <= 302 && t[i + 1][j] == -1);
                bool j3 = j - 1 < 0 || (j - 1 >= 0 && t[i][j - 1] == -1);
                bool j4 = j + 1 > 302 || (j + 1 <= 302 && t[i][j + 1] == -1);
                if (j1 && j2 && j3 && j4)
                {
                    if (a[i][j] < res)
                        res = a[i][j];
                }
            }
        }
    }
    if (res == 2147483646)
        res = -1;
    printf("%d\n", res);
    return 0;
}
```

61. 涂国旗 [模拟] [普及-]

P3392 涂国旗

题目描述：

某国法律规定，只要一个由 N×M* 个小方块组成的旗帜符合如下规则，就是合法的国旗。（毛熊：阿嚏——）

- 从最上方若干行（至少一行）的格子全部是白色的；
- 接下来若干行（至少一行）的格子全部是蓝色的；
- 剩下的行（至少一行）全部是红色的；

现有一个棋盘状的布，分成了 *N* 行 *M* 列的格子，每个格子是白色蓝色红色之一，小 a 希望把这个布改成该国国旗，方法是在一些格子上涂颜料，盖住之前的颜色。

小a很懒，希望涂最少的格子，使这块布成为一个合法的国旗。

```
#include <iostream>
#include <cstdio>
using namespace std;
int n, m, res = 2147483647;
struct A
{
    int w = 0, b = 0, r = 0;
};
A a[51];
char ch;
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= m; ++j)
        {
            scanf(" %c", &ch);
            if (ch == 'W')
                a[i].w++;
            else if (ch == 'B')
                a[i].b++;
            else
                a[i].r++;
```

```
        }
    }
    for (int i = 2; i < n; ++i)
    {
        for (int j = i; j < n; ++j)
        {
            int temp = 0;
            for (int k = 1; k < i; ++k)
            {
                temp += a[k].b + a[k].r;
            }
            for (int k = i; k <= j; ++k)
            {
                temp += a[k].r + a[k].w;
            }
            for (int k = j + 1; k <= n; ++k)
            {
                temp += a[k].b + a[k].w;
            }
            if (res > temp)
                res = temp;
        }
    }
    // res += a[1].b + a[1].r + a[n].b + a[n].w;
    printf("%d\n",res);
    return 0;
}
```

62. Secret Cow Code S [模拟] [普及-]

P3612 [USACO17JAN]Secret Cow Code S

题目描述：

奶牛正在试验秘密代码，并设计了一种方法来创建一个无限长的字符串作为其代码的一部分使用。

给定一个字符串，让后面的字符旋转一次（每一次正确的旋转，最后一个字符都会成为新的第一个字符）。也就是说，给定一个初始字符串，之后的每一步都会增加当前字符串的长度。

给定初始字符串和索引，请帮助奶牛计算无限字符串中位置N的字符。

输入格式

第一行输入一个字符串。该字符串包含最多30个大写字母，并 $N \leq 10^{18}$ 。

第二行输入N。请注意，数据可能很大，放进一个标准的32位整数可能不够，所以你可能要使用一个64位的整数类型（例如，在C / C++ 中是 long long）。

输出格式

请输出从初始字符串生成的无限字符串中的位置的字符。第一个字符是 $N=1$.。

输入输出样例

输入 #1

```
COW 8
```

输出 #1

```
C
```

说明/提示

In this example, the initial string COW expands as follows:

COW -> COWWCO -> COWWCOOCOWWC

12345678

```cpp
#include <iostream>
#include <string>
using namespace std;
long long n;
string s;
int main()
{
    cin >> s;
    cin >> n;
    int m = s.length();
    long long N = m;
    while (N < n)
        N = N * 2;
    while (N > m)
    {
        N /= 2;
        if (n == N + 1)
            --n;
        else
        {
            n = n - (N + 1);
            N = m;
            while (N < n)
                N = N * 2;
        }
    }
    cout << s[n-1] << endl;
    return 0;
}
```

63. 路标设置 [二分/最小化最大值] [普及+/提高]

P3853 [TJOI2007]路标设置

题目背景

B市和T市之间有一条长长的高速公路，这条公路的某些地方设有路标，但是大家都感觉路标设得太少了，相邻两个路标之间往往隔着相当长的一段距离。为了便于研究这个问题，我们把公路上相邻路标的最大距离定义为该公路的“空旷指数”。

题目描述

现在政府决定在公路上增设一些路标，使得公路的“空旷指数”最小。他们请求你设计一个程序计算能达到的最小值是多少。请注意，公路的起点和终点保证已设有路标，公路的长度为整数，并且原有路标和新设路标都必须距起点整数个单位距离。

输入格式

第1行包括三个数L、N、K，分别表示公路的长度，原有路标的数量，以及最多可增设的路标数量。

第2行包括递增排列的N个整数，分别表示原有的N个路标的位置。路标的位置用距起点的距离表示，且一定位于区间[0,L]内。

输出格式

输出1行，包含一个整数，表示增设路标后能达到的最小“空旷指数”值。

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int p, n, k, d[50002];
inline bool solve(int x)
{
    int i = 0, cnt = 0;
    while (i < n - 1)
    {
        if (d[i + 1] - d[i] > x)
        {
            cnt += (d[i + 1] - d[i]) / x;
            if ((d[i + 1] - d[i]) % x == 0)
                --cnt;
        }
        ++i;
    }
    return cnt <= k;
}
int main()
{
    scanf("%d%d%d", &p, &n, &k);
    for (int i = 0; i < n; ++i)
    {
        scanf("%d", &d[i]);
    }
    int l = 0, r = p * 2;
    while (l < r - 1)
    {
        int mid = (l + r) / 2;
        if (solve(mid))
            r = mid;
        else
            l = mid;
    }
    printf("%d\n", r);
    return 0;
}
```

64. 魔法少女小Scarlet [模拟/递归/枚举/暴力] [普及/提高-]

P4924 [1007]魔法少女小Scarlet

题目描述

Scarlet最近学会了一个数组魔法，她会在$n*n$二维数组上将一个奇数阶方阵按照顺时针或者逆时针旋转90°，

首先，Scarlet会把1到n^2的正整数按照从左往右，从上至下的顺序填入初始的二维数组中，然后她会施放一些简易的魔法。

Scarlet既不会什么分块特技，也不会什么Splay套Splay，她现在提供给你她的魔法执行顺序，想让你来告诉她魔法按次执行完毕后的二维数组。

输入格式

第一行两个整数n,m，表示方阵大小和魔法施放次数。

接下来*m*行，每行44个整数x,y,r,z，*表示在这次魔法中，Scarlet会把以第x行第y列为中心的2r+1阶矩阵按照某种时针方向旋转，其中z=0表示顺时针*，z*=1表示逆时针。

输出格式

输出*n*行，每行*n*个用空格隔开的数，表示最终所得的矩阵

```
#include <iostream>
#include <cstdio>
using namespace std;
inline int read()
{
    int x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9')
    {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9')
    {
        x = (x << 3) + (x << 1) + (ch ^ 48);
        ch = getchar();
    }
    return x * f;
}
int a[501][501], n, m, x, y, r, z, temp[501][501];
void transfer_0(int r)
{
    for (int i = -r; i <= r; ++i)
    {
        temp[x + i][y + r] = a[x + i][y + r];
        temp[x - r][y + i] = a[x - r][y + i];
        temp[x + i][y - r] = a[x + i][y - r];
        temp[x + r][y + i] = a[x + r][y + i];
    }
    for (int i = -r; i <= r; ++i)
    {
        a[x + i][y + r] = temp[x - r][y + i];
        a[x - r][y + i] = temp[x - i][y - r];
        a[x - i][y - r] = temp[x + r][y - i];
        a[x + r][y - i] = temp[x + i][y + r];
    }
}
void transfer_1(int r)
{
    for (int i = -r; i <= r; ++i)
    {
        temp[x + i][y + r] = a[x + i][y + r];
        temp[x - r][y + i] = a[x - r][y + i];
        temp[x + i][y - r] = a[x + i][y - r];
        temp[x + r][y + i] = a[x + r][y + i];
    }
    for (int i = -r; i <= r; ++i)
    {
        a[x - r][y + i] = temp[x + i][y + r];
        a[x - i][y - r] = temp[x - r][y + i];
        a[x + r][y - i] = temp[x - i][y - r];
        a[x + i][y + r] = temp[x + r][y - i];
    }
```

```
}
int main()
{
    n = read();
    m = read();
    int t = 1;
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= n; ++j)
        {
            a[i][j] = t++;
        }
    }
    for (int i = 0; i < m; ++i)
    {
        x = read();
        y = read();
        r = read();
        z = read();
        if (z == 0)
        {
            for (int j = 1; j <= r; ++j)
            {
                transfer_0(j);
            }
        }
        else
        {
            for (int j = 1; j <= r; ++j)
            {
                transfer_1(j);
            }
        }
    }
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= n; ++j)
        {
            printf("%d%c", a[i][j], j == n ? '\n' : ' ');
        }
    }
    return 0;
}
```

65. 求第 k 小的数 [普及/提高-]

P1923 【深基9.例4】求第 k 小的数

题目描述

输入 n(n<5000000 且 n 为奇数) 个数字 $a_i(0 < a_i < 10^9)$，输出这些数字的第 k 小的数。最小的数是第 0 小。

请尽量不要使用 `nth_element` 来写本题，因为本题的重点在于练习分治算法。

```
#include <iostream>
#include <cstdio>
using namespace std;
inline int read()
{
    int x = 0, f = 1;
```

```
    char ch = getchar();
    while (ch < '0' || ch > '9')
    {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9')
    {
        x = (x << 3) + (x << 1) + (ch ^ 48);
        ch = getchar();
    }
    return x * f;
}
int n, a[5000000], k;
int quicksort(int left, int right)
{
    int mid = a[left];
    while (left < right)
    {
        while (left < right && mid <= a[right])
            --right;
        a[left] = a[right];
        while (left < right && mid >= a[left])
            ++left;
        a[right] = a[left];
    }
    return left;
}
void find_k(int left, int right)
{
    int temp = quicksort(left, right);
    if (temp == k)
        printf("%d\n", a[temp]);
    else if (k - 1 < temp)
        find_k(left, temp - 1);
    else
        find_k(temp + 1, right);
}
int main()
{
    n = read();
    k = read();
    for (int i = 0; i < n; ++i)
    {
        a[i] = read();
    }
    find_k(0, n - 1);
    return 0;
}
```

66. 查找 [二分] [普及-]

P2249 【深基13.例1】查找

题目描述

输入 n(n≤106) 个不超过 10^9 的单调不减的（就是后面的数字不小于前面的数字）非负整数 $a_1, a_2, \ldots, a_n$，然后进行 $m(m \leq 10^5)$ 次询问。对于每次询问，给出一个整数 $q(q \leq 10^9)$，要求输出这个数字在序列中第一次出现的编号，如果没有找到的话输出 -1 。

输入格式

第一行 2 个整数 n 和 m，表示数字个数和询问次数。

第二行 n 个整数，表示这些待查询的数字。

第三行 m 个整数，表示询问这些数字的编号，从 1 开始编号。

输出格式

m 个整数表示答案。

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
int n, m, A[1000001];
int query(int q);
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 0; i < n; ++i)
    {
        scanf("%d", &A[i]);
    }
    int q;
    for (int i = 0; i < m; ++i)
    {
        scanf("%d", &q);
        printf("%d%c", query(q), i == m - 1 ? '\n' : ' ');
    }
    return 0;
}
int query(int q)
{
    int lb = -1, ub = n - 1;
    while (ub - lb > 1)
    {
        int md = (lb + ub) / 2;
        int x = A[md];
        if (x >= q)
            ub = md;
        else
            lb = md;
    }
    return A[ub] == q ? ub+1 : -1;
}
```

67. 滑动窗口 /【模板】单调队列 [单调队列/线段树] [普及/提高-]

P1886 滑动窗口 /【模板】单调队列

题目描述

有一个长为 n*n* 的序列 a*a*，以及一个大小为 k*k* 的窗口。现在这个从左边开始向右滑动，每次滑动一个单位，求出每次滑动后窗口中的最大值和最小值。

例如：

The array is [1,3,-1,-3,5,3,6,7][1,3,−1,−3,5,3,6,7], and k = 3*k*=3。

Window position	Minimum value	Maximum value
[1 3 -1] -3 5 3 6 7	-1	3
1 [3 -1 -3] 5 3 6 7	-3	3
1 3 [-1 -3 5] 3 6 7	-3	5
1 3 -1 [-3 5 3] 6 7	-3	5
1 3 -1 -3 [5 3 6] 7	3	6
1 3 -1 -3 5 [3 6 7]	3	7

输入格式

输入一共有两行，第一行有两个正整数 n,k*n*,*k*。 第二行 n*n* 个整数，表示序列 a*a*

输出格式

输出共两行，第一行为每次窗口滑动的最小值
第二行为每次窗口滑动的最大值

分析：

此题为单调队列模板题，其实楼下把单调队列与优先队列混为一谈本人并不赞同。

单调队列有两个性质

1. 队列中的元素其对应在原来的列表中的顺序必须是单调递增的。
2. 队列中元素的大小必须是单调递*(增/减/甚至是自定义也可以)

单调队列与普通队列不一样的地方就在于单调队列既可以从队首出队，也可以从队尾出队。

那么我们应该怎样实现单调队列呢?

就拿样例来谈谈，设以最小的为标准。

```
8 3
1 3 -1 -3 5 3 6 7
```

下文中我们用q来表示单调队列，p来表示其所对应的在原列表里的序号。

1. 由于此时队中没有一个元素，我们直接令1进队。此时，q={1},p={1}。
2. 现在3面临着抉择。下面基于这样一个思想:假如把3放进去，如果后面2个数都比它大，那么3在其有生之年就有可能成为最小的。此时，q={1,3},p={1,2}
3. 下面出现了-1。队尾元素3比-1大，那么意味着只要-1进队，那么3在其有生之年必定成为不了最小值，原因很明显:因为当下面3被框起来，那么-1也一定被框起来，所以3永远不能当最小值。所以，3从**队尾**出队。同理，1从队尾出队。最后-1进队，此时q={-1},p={3}
4. 出现-3，同上面分析，-1>-3，-1从队尾出队,-3从队尾进队。q={-3}，p={4}。
5. 出现5，因为5>-3，同第二条分析，5在有生之年还是有希望的，所以5进队。此时，q={-3,5},p={4,5}
6. 出现3。3先与队尾的5比较，3<5，按照第3条的分析，5从队尾出队。3再与-3比较，同第二条分析，3进队。此时，q={-3,3},p={4,6}
7. 出现6。6与3比较，因为3<6，所以3不必出队。由于3以前元素都＜3，所以不必再比较，6进队。因为-3此时已经在滑动窗口之外，所以-3从**队首**出队。此时，q={3,6},p={6,7}
8. 出现7。队尾元素6小于7，7进队。此时，q={3,6,7},p={6,7,8}。

那么，我们对单调队列的基本操作已经分析完毕。因为单调队列中元素大小单调递*(增/减/自定义比较)，因此，队首元素必定是最值。按题意输出即可。

代码：

```
#include <iostream>
```

```cpp
#include <queue>
using namespace std;
int n, k, a[1000002], maxs[1000002], mins[1000002];
deque<int> maxdq, mindq;
int main()
{
    n = read();
    k = read();
    for (int i = 0; i < n; ++i)
    {
        a[i] = read();
    }
    int maxi = 0, mini = 0;
    for (int i = 0; i < n; ++i)
    {
        while (!maxdq.empty() && i - k >= maxdq.front())
        {
            maxdq.pop_front();
        }
        while (!mindq.empty() && i - k >= mindq.front())
        {
            mindq.pop_front();
        }
        while (!maxdq.empty() && a[maxdq.back()] <= a[i])
        {
            maxdq.pop_back();
        }
        while (!mindq.empty() && a[mindq.back()] >= a[i])
        {
            mindq.pop_back();
        }
        maxdq.push_back(i);
        mindq.push_back(i);
        if (i >= k - 1)
        {
            maxs[maxi++] = a[maxdq.front()];
            mins[mini++] = a[mindq.front()];
        }
    }

    for (int i = 0; i < mini - 1; ++i)
    {
        printf("%d ", mins[i]);
    }
    printf("%d\n", mins[mini - 1]);

    for (int i = 0; i < maxi - 1; ++i)
    {
        printf("%d ", maxs[i]);
    }
    printf("%d\n", maxs[maxi - 1]);

    system("pause");
    return 0;
}

```

68. 【模板】单调栈 [单调栈] [普及/提高-]

P5788 【模板】单调栈

题目描述

给出项数为 *n* 的整数数列 $a_{1...n}$。

定义函数 *f*(*i*) 代表数列中第 *i* 个元素之后第一个大于 a_i 的元素的**下标**，即 $f(i)=\min_{i<j\leq n,a_j>a_i}\{j\}$。若不存在，则 f(i*)=0。

试求出 $f(1...n)$。

输入格式

第一行一个正整数 *n*。

第二行 n*n* 个正整数 $a_{1...n}$。

输出格式

一行 *n* 个整数 $f(1...n)$ 的值。

分析：

单调栈

首先了解一下什么是单调栈吧！

就跟单调队列差不多。单调队列主要处理的是一个区间内的最大/小值，而单调栈处理的是寻找以某个值为最小/大值的最大区间。相比较，实际上单调栈用的虽然少一些，但是比单调队列更加灵活多变。

单调栈就是使栈内元素单调递增或者单调递减的栈，单调栈也只能在栈顶操作。

我们先来模拟一遍吧！

做一个比喻，比方说：有个集训队招人，一个数代表了一个选手的能力值，先进来的选手年龄会比较大，后面的选手年龄比较小，但是这个集训队没有人数限制，那么如果遇到一个比你小还比你强的人那么准备退役吧。

比如有 5 个能力值分别是：

1 7 5 6 3

要加进来这个单调栈。

首先是 1 也就是 选手1 ，那么集训队没有人和他比较所以进入集训队。

单调栈的情况：1 。

然后是 7 也就是 选手2 ，那么我们可以发现，选手2 比 选手1 要小，并且 选手2 的能力比 选手1 要强，那么 选手1 留着也没啥用，淘汰！好残酷呀！

单调栈的情况：7 。

然后是 5 选手3 ，选手3 比 选手2 年龄要小，但是 选手3 的能力没有 选手2 强，那么 选手3 招进集训队。

单调栈的情况：7 5 。

那么接下来是 6 选手4 ，选手4 比 选手3 年龄要小，比他还要强， 选手3 淘汰！往后比较，发现 选手2 虽然比 选手4 要大，但是他能力很强！所以不会被淘汰，将 选手4 招进集训队。

单调栈的情况：7 6 。

最后是 3 选手5 ，选手5 比 选手4 要小，但是 选手4 的能力比 选手5 要强，所以 选手4 必定也比不过 选手2 ，但由于 选手5 小所以不淘汰他。

单调栈的情况：7 6 3 。

通过这个模拟我们发现，如果你很强，就算你年龄大也不会被淘汰，其实不是这样，在单调队列(比单调栈高级的东西)里面你就算再强也终究有时候会退役的，所以好好珍惜吧！

```cpp
#include <iostream>
#include <stack>
using namespace std;
int n, a[3000001], res[3000001];
inline int read()
{
    int x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9')
    {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9')
    {
        x = (x << 3) + (x << 1) + (ch ^ 48);
        ch = getchar();
    }
    return x * f;
}
stack<int> s;
int main()
{
    n = read();
    for (int i = 1; i <= n; ++i)
    {
        a[i] = read();
    }
    for (int i = 1; i <= n; ++i)
    {
        while (!s.empty() && a[s.top()] < a[i])
        {
            res[s.top()] = i;
            s.pop();
        }
        s.push(i);
    }
    while (!s.empty())
    {
        res[s.top()] = 0;
        s.pop();
    }
    for (int i = 1; i <= n; ++i)
    {
        printf("%d%c", res[i], i == n ? '\n' : ' ');
    }
    system("pause");
    return 0;
}
```

69. 【模板】并查集 [并查集] [普及-]

P3367 【模板】并查集

题目描述

如题，现在有一个并查集，你需要完成合并和查询操作。

输入格式

第一行包含两个整数 *N*,*M* ,表示共有 *N* 个元素和 *M* 个操作。

接下来 *M* 行，每行包含三个整数 Z_i, X_i, Y_i。

当 $Z_i = 1$ 时，将 X_i与 Y_i 所在的集合合并。

当 $Z_i = 2$ 时，输出 X_i 与 Y_i是否在同一集合内，是的输出 `Y` ；否则输出 `N` 。

输出格式

对于每一个 $Z_i = 2$ 的操作，都有一行输出，每行包含一个大写字母，为 `Y` 或者 `N` 。

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

using namespace std;
int parent[10002], m, n;

void unite(int x, int y);

int find_root(int x);

bool isSame(int x, int y);
void init();

int main()
{
    cin >> n >> m;
    init();
    int t1, t2, t3;
    for (int i = 0; i < m; ++i)
    {
        cin >> t1 >> t2 >> t3;
        if (t1 == 1)
        {
            unite(t2, t3);
        }
        else if (t1 == 2)
        {
            if (isSame(t2, t3))
                cout << "Y" << endl;
            else
                cout << "N" << endl;
```

```
        }
    }
    return 0;
}

void init()
{
    for (int i = 1; i <= n; ++i)
    {
        parent[i] = i;
    }
}

void unite(int x, int y)
{
    int x_root=find_root(x);
    int y_root=find_root(y);
    if (x_root != y_root)
        parent[x_root] = y_root;
}

int find_root(int x)
{
    if (parent[x] == x)
        return x;
    return parent[x] = find_root(parent[x]);
}

bool isSame(int x, int y)
{
    return find_root(x) == find_root(y);
}
```