CCF 全国信息学奥林匹克联赛（NOIP2012）复赛

普及组

（请选手务必仔细阅读本页内容）

一．题目概况

中文题目名称	质因数分解	寻宝	摆花	文化之旅
英文题目与子目录	prime	treasure	flower	culture
可执行文件名	prime	treasure	flower	culture
输入文件名	prime.in	treasure.in	flower.in	culture.in
输出文件名	prime.out	treasure.out	flower.out	culture.out
每个测试点时限	1 秒	1 秒	1 秒	1 秒
测试点数目	10	10	10	10
每个测试点分值	10	10	10	10
附加样例文件	有	有	有	有
结果比较方式	全文比较（过滤行末空格及文末回车）			
题目类型	传统	传统	传统	传统

二．提交源程序文件名

对于 C++语言	prime.cpp	treasure.cpp	flower.cpp	culture.cpp
对于 C 语言	prime.c	treasure.c	flower.c	culture.c
对于 pascal 语	prime.pas	treasure.pas	flower.pas	culture.pas

三．编译命令（不包含任何优化开关）

对于 C++语言	g++ -o prime prime.cpp -lm	g++ -o treasure treasure.cpp -lm	g++ -o flower flower.cpp -lm	g++ -o culture culture.cpp -lm
对于 C 语言	gcc -o prime prime.c -lm	gcc -o treasure treasure.c -lm	gcc -o flower flower.c -lm	gcc -o culture culture.c -lm
对于 pascal 语	fpc prime.pas	fpc	fpc flower.pas	fpc culture.pas

四．运行内存限制

内存上限	128M	128M	128M	128M

注意事项：

1、文件名（程序名和输入输出文件名）必须使用英文小写。

2、C/C++中函数 main()的返回值类型必须是 int，程序正常结束时的返回值必须是 0。

3、全国统一评测时采用的机器配置为：CPU Intel Core2 Quad Q8200 2.33GHz，内存 2G，上 述时限以此配置为准。

4、特别提醒：评测在 **NOI Linux** 下进行。

Created by Tinghua Yuan

1.质因数分解

(prime.cpp/c/pas)

【问题描述】

已知正整数 n 是两个不同的质数的乘积，试求出较大的那个质数。

【输入】

输入文件名为 prime.in。 输入只有一行，包含一个正整数 n。

【输出】

输出文件名为 prime.out。 输出只有一行，包含一个正整数 p，即较大的那个质数。

【输入输出样例】

prime.in	prime.out
21	7

【数据范围】

对于 60%的数据，$6 \le n \le 1000$。 对于 100%的数据，$6 \le n \le 2*10^9$。

[解题思路]

1.根据数据范围，可以判断数据范围为长整型（long=4 bytes）

补充数据类型：

整型		
	char	1byte
	Int	4bytes
	Short	2bytes
	Long	4bytes
	Long long	8bytes

程序：

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

int a[44722]={0};                    //用于筛选法求素数,有题目数据范围可知,涉及到的最大素数一定小于44722
int b[10000]={0};                    //用于存储素数,b[i]用于存储第i+1个素数
int count=0;

void calculatePrime(void)            //筛选法求素数，主要考虑到时间效率，所以用此法求素数
{
    int i,j;

    memset(a,-1,sizeof(a));
    memset(b,0,sizeof(b));
    a[0]=a[1]=0;

    for (i=2;i<50000;i++)
    {
            if (a[i]==-1)
            {
                    for (j=i+i;j<50000;j+=i)
                    {
                            a[j]=0;
                    }
            }
    }
    for (i=2;i<50000;i++)
```

Created by Jinghua Yuan

```
    {
            if (a[i]==-1)
            {
                    b[count++]=i;
            }
    }
}
int main(void)
{
    int i,n,squre,temp;
    FILE *inputfp;
    FILE *outputfp;

    calculatePrime();

    inputfp=fopen("prime.in.txt","r");
    outputfp=fopen("prime.out.txt","w");

    while (fscanf(inputfp,"%d",&n)!=EOF)
    {
            printf("%d",n);
            squre=(int)sqrt(n);
            for (i=0;i<=squre&&i<count;i++)
            {
                    if (n%b[i]==0) //核心:两个素数的积一定为合适,且只有3个因子
                    {
                            temp=n/b[i];
                            break;
                    }
            }
            fprintf(outputfp,"%d\n",temp);
    }
    fclose(inputfp);
    fclose(outputfp);

    return 0;
}
```

Created by Jinghua Yuan

2. 寻宝

(treasure.cpp/c/pas)

传说很遥远的藏宝楼顶层藏着诱人的宝藏。小明历尽千辛万苦终于找到传说中的这个藏 宝楼，藏宝楼的门口竖着一个木板，上面写有几个大字：寻宝说明书。说明书的内容如下：

藏宝楼共有 N+1 层，最上面一层是顶层，顶层有一个房间里面藏着宝藏。除了顶层外，藏宝楼另有 N 层，每层 M 个房间，这 M 个房间围成一圈并按逆时针方向依次编号为 0，…， M-1。其中一些房间有通往上一层的楼梯，每层楼的楼梯设计可能不同。每个房间里有一个 指示牌，指示牌上有一个数字 x，表示从这个房间开始按逆时针方向选择第 x 个有楼梯的房 间（假定该房间的编号为 k），从该房间上楼，上楼后到达上一层的 k 号房间。比如当前房 间的指示牌上写着 2，则按逆时针方向开始尝试，找到第 2 个有楼梯的房间，从该房间上楼。如果当前房间本身就有楼梯通向上层，该房间作为第一个有楼梯的房间。

寻宝说明书的最后用红色大号字体写着：“寻宝须知：帮助你找到每层上楼房间的指示 牌上的数字（即每层第一个进入的房间内指示牌上的数字）总和为打开宝箱的密钥”。

请帮助小明算出这个打开宝箱的密钥。

【输入】

输入文件为 treasure.in

第一行 2 个整数 N 和 M，之间用一个空格隔开。N 表示除了顶层外藏宝楼共 N 层楼，

M 表示除顶层外每层楼有 M 个房间。

接下来 N*M 行，每行两个整数，之间用一个空格隔开，每行描述一个房间内的情况， 其中第(i-1)*M+j 行表示第 i 层 j-1 号房间的情况（i=1, 2, …, N；j=1, 2, … ,M）。第一个整数 表示该房间是否有楼梯通往上一层（0 表示没有，1 表示有），第二个整数表示指示牌上的数 字。注意，从 **j** 号房间的楼梯爬到上一层到达的房间一定也是 **j** 号房间。

最后一行，一个整数，表示小明从藏宝楼底层的几号房间进入开始寻宝（注：房间编号 从 0 开始）。

【输出】

输出文件名为 treasure.out。 输出只有一行，一个整数，表示打开宝箱的密钥，这个数可能会很大，请输出对 20123

取模的结果即可。

【输入输出样例】

treasure.in	treasure.out
2 3 1 2 0 3 1 4 0 1 1 5 1 2 1	5

【输入输出样例说明】

第一层：

0 号房间，有楼梯通往上层，指示牌上的数字是 2；

1 号房间，无楼梯通往上层，指示牌上的数字是 3；

2 号房间，有楼梯通往上层，指示牌上的数字是 4； 第二层：

0 号房间，无楼梯通往上层，指示牌上的数字是 1；

1 号房间，有楼梯通往上层，指示牌上的数字是 5；

Created by Jinghua Yuan

2 号房间，有楼梯通往上层，指示牌上的数字是 2；

小明首先进入第一层（底层）的 1 号房间，记下指示牌上的数字为 3，然后从这个房间 开始，沿逆时针方向选择第 3 个有楼梯的房间 2 号房间进入，上楼后到达第二层的 2 号房间， 记下指示牌上的数字为 2，由于当前房间本身有楼梯通向上层，该房间作为第一个有楼梯的 房间。因此，此时沿逆时针方向选择第 2 个有楼梯的房间即为 1 号房间，进入后上楼梯到达 顶层。这时把上述记下的指示牌上的数字加起来，即 3+2=5，所以打开宝箱的密钥就是 5。

【数据范围】

对于 50%数据，有 0<N≤1000，0<x≤10000；

对于 100%数据，有 0<N≤10000，0<M≤100，0<x≤1,000,000。

[解题思路]

1. 建立数组，存储数据，然后模拟寻宝过程，统计总和

2. 将每一层有楼梯的房间数统计为一个数组

3. 标牌上的数字x可能远远大于每一层有楼梯的房间数，所以可以对房间数求余。（考虑余数为0的情况）

```
#include <stdio.h>

const int mod=20123;
int a[10000][100][2];
int b[10000];

int main(void)
{
    int n,m,start,i,j,k,temp;
    FILE *inputfp;
    FILE *outputfp;

    inputfp=fopen("treasure.in.txt","r");
    outputfp=fopen("treasure.out.txt","w");

    while(fscanf(inputfp,"%d%d",&n,&m)!=EOF)
    {
            for (i=0;i<n;i++)
            {
                    b[i]=0;
                    for (j=0;j<m;j++)
                    {
                            fscanf(inputfp,"%d%d",&a[i][j][0],&a[i][j][1]);
                            if (a[i][j][0]==1)              //统计各层有楼梯的房间的个数
                            {
                                    b[i]++;
                            }
                    }
            }
            fscanf(inputfp,"%d",&start);
            for (temp=0,i=0;i<n;i++)
            {
                    temp=(temp+a[i][start][1])%mod;         //累加求密钥
                    k=(a[i][start][1]%b[i]==0)?b[i]:a[i][start][1]%b[i];
                                                    //对牌号做取模处理,减少无效循环次数
```

Created by Jinghua Yuan

```
                start--;
                do{
                        start++;
                        start%=m;               //start是刚进入第i+1层时的房间号
                        if (a[i][start][0]==1) //此房间有楼梯
                        {
                                k--;
                        }
                } while (k!=0);                 //k!=0表示尚未找到牌号所指示的房间
        }
        fprintf(outputfp,"%d\n",temp);
    }
    fclose(inputfp);
    fclose(outputfp);
    return 0;
}
```

3. 摆花

(flower.cpp/c/pas)

小明的花店新开张，为了吸引顾客，他想在花店的门口摆上一排花，共 m 盆。通过调 查顾客的喜好，小明列出了顾客最喜欢的 n 种花，从 1 到 n 标号。为了在门口展出更多种花， 规定第 i 种花不能超过 a_i 盆，摆花时同一种花放在一起，且不同种类的花需按标号的从小到 大的顺序依次摆列。

试编程计算，一共有多少种不同的摆花方案。

【输入】

输入文件 flower.in，共 2 行。

第一行包含两个正整数 n 和 m，中间用一个空格隔开。

第二行有 n 个整数，每两个整数之间用一个空格隔开，依次表示 a_1、a_2、……a_n。

【输出】

输出文件名为 flower.out。 输出只有一行，一个整数，表示有多少种方案。注意：因为方案数可能很多，请输出

方案数对 **1000007** 取模的结果。

【输入输出样例 1】

flower.in	flower.out
2 4 3 2	2

【输入输出样例说明】

有 2 种摆花的方案，分别是(1，1，1，2)， (1，1，2，2)。括号里的 1 和 2 表示两种花， 比如第一个方案是前三个位置摆第一种花，第四个位置摆第二种花。

【数据范围】

对于 20%数据，有 $0<n\leq 8$，$0<m\leq 8$，$0\leq a_i\leq 8$；

对于 50%数据，有 $0<n\leq 20$，$0<m\leq 20$，$0\leq a_i\leq 20$；

对于 100%数据，有 $0<n\leq 100$，$0<m\leq 100$，$0\leq a_i\leq 100$。

[解题思路]

1. 建立数组，存储数据

2. 动态规划问题，同背包问题类似。只不过是要求所有的可能组合方案数。

Created by Jinghua Yuan

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

const int mod=1000007;

int main(void)
{
    int n,m,i,j,k;
    int a[101],f[101][101];

    FILE *inputfp;
    FILE *outputfp;

    inputfp=fopen("treasure.in.txt","r");
    outputfp=fopen("treasure.out.txt","w");

    while(fscanf(inputfp,"%d%d",&n,&m)!=EOF)
    {
            memset(f,0,sizeof(f));
            f[0][0]=1;
            for (i=1;i<=n;i++)
            {
                    fscanf(inputfp,"%d",&a[i]);
            }
            for (i=1;i<=n;i++)                      //动态规划,和背包问题相似。求组合和方案数目
            {
                    for (j=0;j<=m;j++)
                    {
                            for (k=0;k<=a[i]&&k<=j;k++)
                            {
                                    f[i][j]=(f[i][j]+f[i-1][j-k])%mod;
                            }
                    }
            }
            fprintf(outputfp,"%d\n",f[n][m]);
    }
    fclose(inputfp);
    fclose(outputfp);
    return 0;
}
```

Created by Jinghua Yuan

4. 文化之旅

(culture.cpp/c/pas)

有一位使者要游历各国，他每到一个国家，都能学到一种文化，但他不愿意学习任何一 种文化超过一次（即如果他学习了某种文化，则他就不能到达其他有这种文化的国家）。不 同的国家可能有相同的文化。不同文化的国家对其他文化的看法不同，有些文化会排斥外来 文化（即如果他学习了某种文化，则他不能到达排斥这种文化的其他国家）。

现给定各个国家间的地理关系，各个国家的文化，每种文化对其他文化的看法，以及这 位使者游历的起点和终点（在起点和终点也会学习当地的文化），国家间的道路距离，试求 从起点到终点最少需走多少路。

【输入】

输入文件 culture.in。

第一行为五个整数 N，K，M，S，T，每两个整数之间用一个空格隔开，依次代表国家 个数（国家编号为 1 到 N），文化种数（文化编号为 1 到 K），道路的条数，以及起点和终点 的编号（保证 S 不等于 T）；

第二行为 N 个整数，每两个整数之间用一个空格隔开，其中第 i 个数 C_i，表示国家 i 的文化为 C_i。

接下来的 K 行，每行 K 个整数，每两个整数之间用一个空格隔开，记第 i 行的第 j 个数 为 a_{ij}，a_{ij}= 1 表示文化 i 排斥外来文化 j（i 等于 j 时表示排斥相同文化的外来人），a_{ij}= 0 表示不排斥（注意 i 排斥 j 并不保证 j 一定也排斥 i）。

接下来的 M 行，每行三个整数 u，v，d，每两个整数之间用一个空格隔开，表示国家 u 与国家 v 有一条距离为 d 的可双向通行的道路（保证 u 不等于 v，两个国家之间可能有多条 道路）。

【输出】

输出文件名为 culture.out。 输出只有一行，一个整数，表示使者从起点国家到达终点国家最少需要走的距离数（如果无解则输出-1）。

【输入输出样例 1】

culture.in	culture.out
2 2 1 1 2 1 2 0 1 1 0 1 2 10	-1

【输入输出样例说明】

由于到国家 2 必须要经过国家 1，而国家 2 的文明却排斥国家 1 的文明，所以不可能到 达国家 2。

Created by Jinghua Yuan

【输入输出样例 2】

culture.in	culture.out
2 2 1 1 2 1 2 0 1 0 0 1 2 10	10

【输入输出样例说明】

路线为 1 -> 2。

【数据范围】

对于 20%的数据，有 2≤N≤8，K≤5；

对于 30%的数据，有 2≤N≤10，K≤5； 对于 50%的数据，有 2≤N≤20，K≤8； 对于 70%的数据，有 2≤N≤100，K≤10；

对于 100%的数据，有 2≤N≤100，1≤K≤100，1≤M≤N^2，1≤k_i≤K，1≤u, v≤N，1≤d≤1000，

S≠T，1 ≤S, T≤N。

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

const int MAX=100001;

int main(void)
{
    int N, K, M, S, T,i,j,k,sum;
    int C[101],D[101],a[101][101],u[10001],v[10001],d[10001];

    FILE *inputfp;
    FILE *outputfp;

    inputfp=fopen("culture.in","r");
    outputfp=fopen("culture.out.txt","w");

    while(fscanf(inputfp,"%d%d%d%d%d",&N,&K,&M,&S,&T)!=EOF)
    {
        for (i=1;i<=N;i++) fscanf(inputfp,"%d",&C[i]);
        for (i=1;i<=K;i++) {
            for (j=1;j<=K;j++) fscanf(inputfp,"%d",&a[i][j]);
        }
        for (i=0;i<M;i++) fscanf(inputfp,"%d%d%d",&u[i],&v[i],&d[i]);
        for (i=1;i<=N;i++) D[i] = MAX;
        D[S] = 0;
        for (i=1;i<=N;i++) {                        //动态规划
            for (j=1; j<=N; j++) {
                for (k=0;k<M;k++) {
                    sum = D[j] + d[k];
                    if (u[k]==j && sum<D[v[k]] && a[C[v[k]]][C[j]]==0) {
                        D[v[k]] = sum;
                    }
                    if (v[k]==j && sum<D[u[k]]&& a[C[u[k]]][C[j]]==0) {
                        D[u[k]] = sum;
                    }
```

Created by Jinghua Yuan

```
                        }
                }
        }
        if (D[T]<100001)
                fprintf(outputfp,"%d\n",D[T]);
        else
                fprintf(outputfp,"%d\n",-1);
    }
    fclose(inputfp);
    fclose(outputfp);
    return 0;
}
```

Created by Jinghua Yuan