目前CCF一共搞了4届，这不是一个比赛，就是类似于4/6级那种性质，一共5题，每题满分100分，据了解，基本上对了1题就能拿到证，证上会有你的分数和排名，能考高分的尽量考高分，就像英语6级，430分和600多分，虽然都是发张纸给你，但还是有区别的， 第一题都比较简单，大家尽量把第一题拿下， 提交代码不会返回对错给你，以你最后一次提交为你的答案，结束后再打分，也就是说，你的代码可能有点小错误，但也许能得个60分，80分这样，大概就是这样=.=

第一届CCF第一题

|  |
| --- |
| 201403-1 |
| 试题名称： | 相反数 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　有 N 个非零且各不相同的整数。请你编一个程序求出它们中有多少对相反数(a 和 -a 为一对相反数)。  **输入格式**  　　第一行包含一个正整数 N。(1 ≤ N ≤ 500)。 　　第二行为 N 个用单个空格隔开的非零整数,每个数的绝对值不超过1000,保证这些整数各不相同。  **输出格式**  　　只输出一个整数,即这 N 个数中包含多少对相反数。  **样例输入**  5 1 2 3 -1 -2  **样例输出**  2 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# define LL long long

using namespace std ;

int main ()

{

//freopen("in.txt","r",stdin) ;

int n ;

int a[520] ;

scanf("%d" , &n) ;

int i , j;

int sum = 0 ;

for (i = 0 ; i < n ; i++)

scanf("%d" , &a[i]) ;

for (i = 0 ; i < n ; i++)

for (j = 0 ; j < n ; j++)

{

if (i == j )

continue ;

if (a[i] == -a[j])

sum++ ;

}

printf("%d\n" , sum/2) ;

return 0 ;

}

第一届CCF第二题

|  |  |
| --- | --- |
| 试题名称： | 窗口 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　在某图形操作系统中,有 N 个窗口,每个窗口都是一个两边与坐标轴分别平行的矩形区域。窗口的边界上的点也属于该窗口。窗口之间有层次的区别,在多于一个窗口重叠的区域里,只会显示位于顶层的窗口里的内容。 　　当你点击屏幕上一个点的时候,你就选择了处于被点击位置的最顶层窗口,并且这个窗口就会被移到所有窗口的最顶层,而剩余的窗口的层次顺序不变。如果你点击的位置不属于任何窗口,则系统会忽略你这次点击。 　　现在我们希望你写一个程序模拟点击窗口的过程。  **输入格式**  　　输入的第一行有两个正整数,即 N 和 M。(1 ≤ N ≤ 10,1 ≤ M ≤ 10) 　　接下来 N 行按照从最下层到最顶层的顺序给出 N 个窗口的位置。 每行包含四个非负整数 x1, y1, x2, y2,表示该窗口的一对顶点坐标分别为 (x1, y1) 和 (x2, y2)。保证 x1 < x2,y1 2。 　　接下来 M 行每行包含两个非负整数 x, y,表示一次鼠标点击的坐标。 　　题目中涉及到的所有点和矩形的顶点的 x, y 坐标分别不超过 2559 和　　1439。  **输出格式**  　　输出包括 M 行,每一行表示一次鼠标点击的结果。如果该次鼠标点击选择了一个窗口,则输出这个窗口的编号(窗口按照输入中的顺序从 1 编号到 N);如果没有,则输出"IGNORED"(不含双引号)。  **样例输入**  3 4 0 0 4 4 1 1 5 5 2 2 6 6 1 1 0 0 4 4 0 5  **样例输出**  2 1 1 IGNORED  **样例说明**  　　第一次点击的位置同时属于第 1 和第 2 个窗口,但是由于第 2 个窗口在上面,它被选择并且被置于顶层。 　　第二次点击的位置只属于第 1 个窗口,因此该次点击选择了此窗口并将其置于顶层。现在的三个窗口的层次关系与初始状态恰好相反了。 　　第三次点击的位置同时属于三个窗口的范围,但是由于现在第 1 个窗口处于顶层,它被选择。 　　最后点击的 (0, 5) 不属于任何窗口。 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# define LL long long

using namespace std ;

struct window

{

int x1 ;

int y1 ;

int x2 ;

int y2 ;

int id ;

}w[12];

int main ()

{

//freopen("in.txt","r",stdin) ;

int n , m ;

scanf("%d %d" , &n , &m) ;

int i , j;

for (i = 1 ; i <= n ; i++)

{

scanf("%d%d%d%d" , &w[i].x1 ,&w[i].y1 ,&w[i].x2 ,&w[i].y2 ) ;

w[i].id = i ;

}

int x , y , num;

while(m--)

{

scanf("%d%d" , &x , &y) ;

for (i = n ; i >= 1 ; i--)

{

if (x >= w[i].x1 && x <= w[i].x2 && y >= w[i].y1 && y <= w[i].y2)

{

num = w[i].id ;

window temp = w[i] ;

for (int k = i ; k <= n ; k++)

w[k] = w[k+1] ;

w[n] = temp ;

break ;

}

}

if (i != 0 )

printf("%d\n" , num) ;

else

printf("IGNORED\n") ;

}

return 0 ;

}

第一届CCF第三题

|  |
| --- |
| 201403-3 |
| 试题名称： | 命令行选项 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　请你写一个命令行分析程序,用以分析给定的命令行里包含哪些选项。每个命令行由若干个字符串组成,它们之间恰好由一个空格分隔。这些字符串中的第一个为该命令行工具的名字,由小写字母组成,你的程序不用对它进行处理。在工具名字之后可能会包含若干选项,然后可能会包含一 些不是选项的参数。 　　选项有两类:带参数的选项和不带参数的选项。一个合法的无参数选项的形式是一个减号后面跟单个小写字母,如"-a" 或"-b"。而带参数选项则由两个由空格分隔的字符串构成,前者的格式要求与无参数选项相同,后者则是该选项的参数,是由小写字母,数字和减号组成的非空字符串。 　　该命令行工具的作者提供给你一个格式字符串以指定他的命令行工具需要接受哪些选项。这个字符串由若干小写字母和冒号组成,其中的每个小写字母表示一个该程序接受的选项。如果该小写字母后面紧跟了一个冒号,它就表示一个带参数的选项,否则则为不带参数的选项。例如, "ab:m:" 表示该程序接受三种选项,即"-a"(不带参数),"-b"(带参数), 以及"-m"(带参数)。 　　命令行工具的作者准备了若干条命令行用以测试你的程序。对于每个命令行,你的工具应当一直向后分析。当你的工具遇到某个字符串既不是合法的选项,又不是某个合法选项的参数时,分析就停止。命令行剩余的未分析部分不构成该命令的选项,因此你的程序应当忽略它们。  **输入格式**  　　输入的第一行是一个格式字符串,它至少包含一个字符,且长度不超过 52。格式字符串只包含小写字母和冒号,保证每个小写字母至多出现一次,不会有两个相邻的冒号,也不会以冒号开头。 　　输入的第二行是一个正整数 N(1 ≤ N ≤ 20),表示你需要处理的命令行的个数。 　　接下来有 N 行,每行是一个待处理的命令行,它包括不超过 256 个字符。该命令行一定是若干个由单个空格分隔的字符串构成,每个字符串里只包含小写字母,数字和减号。  **输出格式**  　　输出有 N 行。其中第 i 行以"Case i:" 开始,然后应当有恰好一个空格,然后应当按照字母升序输出该命令行中用到的所有选项的名称,对于带参数的选项,在输出它的名称之后还要输出它的参数。如果一个选项在命令行中出现了多次,只输出一次。如果一个带参数的选项在命令行中出 现了多次,只输出最后一次出现时所带的参数。  **样例输入**  albw:x 4 ls -a -l -a documents -b ls ls -w 10 -x -w 15 ls -a -b -c -d -e -l  **样例输出**  Case 1: -a -l Case 2: Case 3: -w 15 -x Case 4: -a -b |

第一届CCF第四题

|  |
| --- |
| 201403-4 |
| 试题名称： | 无线网络 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　目前在一个很大的平面房间里有 n 个无线路由器,每个无线路由器都固定在某个点上。任何两个无线路由器只要距离不超过 r 就能互相建立网络连接。 　　除此以外,另有 m 个可以摆放无线路由器的位置。你可以在这些位置中选择至多 k 个增设新的路由器。 　　你的目标是使得第 1 个路由器和第 2 个路由器之间的网络连接经过尽量少的中转路由器。请问在最优方案下中转路由器的最少个数是多少?  **输入格式**  　　第一行包含四个正整数 n,m,k,r。(2 ≤ n ≤ 100,1 ≤ k ≤ m ≤ 100, 1 ≤ r ≤ 108)。 　　接下来 n 行,每行包含两个整数 xi 和 yi,表示一个已经放置好的无线 路由器在 (xi, yi) 点处。输入数据保证第 1 和第 2 个路由器在仅有这 n 个路由器的情况下已经可以互相连接(经过一系列的中转路由器)。 　　接下来 m 行,每行包含两个整数 xi 和 yi,表示 (xi, yi) 点处可以增设 一个路由器。 　　输入中所有的坐标的绝对值不超过 108,保证输入中的坐标各不相同。  **输出格式**  　　输出只有一个数,即在指定的位置中增设 k 个路由器后,从第 1 个路 由器到第 2 个路由器最少经过的中转路由器的个数。  **样例输入**  5 3 1 3 0 0 5 5 0 3 0 5 3 5 3 3 4 4 3 0  **样例输出**  2 |

第一届CCF第五题

|  |
| --- |
| 201403-5 |
| 试题名称： | 任务调度 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　有若干个任务需要在一台机器上运行。它们之间没有依赖关系,因此 可以被按照任意顺序执行。 　　该机器有两个 CPU 和一个 GPU。对于每个任务,你可以为它分配不 同的硬件资源: 　　1. 在单个 CPU 上运行。 　　2. 在两个 CPU 上同时运行。 　　3. 在单个 CPU 和 GPU 上同时运行。 　　4. 在两个 CPU 和 GPU 上同时运行。 　　一个任务开始执行以后,将会独占它所用到的所有硬件资源,不得中 断,直到执行结束为止。第 i 个任务用单个 CPU,两个 CPU,单个 CPU 加 GPU,两个 CPU 加 GPU 运行所消耗的时间分别为 ai,bi,ci 和 di。 　　现在需要你计算出至少需要花多少时间可以把所有给定的任务完成。  **输入格式**  　　输入的第一行只有一个正整数 n(1 ≤ n ≤ 40), 是总共需要执行的任 务个数。 　　接下来的 n 行每行有四个正整数 ai, bi, ci, di(ai, bi, ci, di 均不超过 10), 以空格隔开。  **输出格式**  　　输出只有一个整数,即完成给定的所有任务所需的最少时间。  **样例输入**  3 4 4 2 2 7 4 7 4 3 3 3 3  **样例输出**  7  **样例说明**  　　有很多种调度方案可以在 7 个时间单位里完成给定的三个任务,以下是其中的一种方案: 　　同时运行第一个任务(单 CPU 加上 GPU)和第三个任务(单 CPU), 它们分别在时刻 2 和时刻 3 完成。在时刻 3 开始双 CPU 运行任务 2,在 时刻 7 完成。 |

第二届CCF第一题

|  |
| --- |
| 201409-1 |
| 试题名称： | 相邻数对 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　给定n个不同的整数，问这些数中有多少对整数，它们的值正好相差1。  **输入格式**  　　输入的第一行包含一个整数n，表示给定整数的个数。 　　第二行包含所给定的n个整数。  **输出格式**  　　输出一个整数，表示值正好相差1的数对的个数。  **样例输入**  6 10 2 6 3 7 8  **样例输出**  3  **样例说明**  　　值正好相差1的数对包括(2, 3), (6, 7), (7, 8)。  **评测用例规模与约定**  　　1<=n<=1000，给定的整数为不超过10000的非负整数。 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# define LL long long

using namespace std ;

int a[1010] ;

int main ()

{

//freopen("in.txt","r",stdin) ;

int n ;

scanf("%d" , &n) ;

int i , j ;

int sum = 0 ;

for (i = 0 ; i < n ; i++)

{

scanf("%d" , &a[i]) ;

}

sort(a , a+n) ;

for (i = 0 ; i < n ; i++)

{

if (a[i+1] - a[i] != 1)

continue;

else

sum++ ;

}

printf("%d\n" , sum) ;

return 0 ;

}

第二届CCF第二题

|  |
| --- |
| 201409-2 |
| 试题名称： | 画图 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　在一个定义了直角坐标系的纸上，画一个(x1,y1)到(x2,y2)的矩形指将横坐标范围从x1到x2，纵坐标范围从y1到y2之间的区域涂上颜色。 　　下图给出了一个画了两个矩形的例子。第一个矩形是(1,1) 到(4, 4)，用绿色和紫色表示。第二个矩形是(2, 3)到(6, 5)，用蓝色和紫色表示。图中，一共有15个单位的面积被涂上颜色，其中紫色部分被涂了两次，但在计算面积时只计算一次。在实际的涂色过程中，所有的矩形都涂成统一的颜色，图中显示不同颜色仅为说明方便。  　　给出所有要画的矩形，请问总共有多少个单位的面积被涂上颜色。  **输入格式**  　　输入的第一行包含一个整数n，表示要画的矩形的个数。 　　接下来n行，每行4个非负整数，分别表示要画的矩形的左下角的横坐标与纵坐标，以及右上角的横坐标与纵坐标。  **输出格式**  　　输出一个整数，表示有多少个单位的面积被涂上颜色。  **样例输入**  2 1 1 4 4 2 3 6 5  **样例输出**  15  **评测用例规模与约定**  　　1<=n<=100，0<=横坐标、纵坐标<=100。 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# define LL long long

using namespace std ;

int a[110][110] ;

int main ()

{

//freopen("in.txt","r",stdin) ;

int n ;

scanf("%d" , &n) ;

int i , j ;

int x1 , y1 , x2 , y2 ;

int sum = 0 ;

while(n--)

{

scanf("%d%d%d%d" , &x1,&y1,&x2,&y2) ;

for (i = x1+1 ; i <= x2 ; i++)

for (j = y1+1 ; j <= y2 ; j++)

a[i][j] = 1 ;

}

for (i = 1 ; i <= 105 ; i++)

for (j = 1 ; j <= 105 ; j++)

sum += a[i][j] ;

printf("%d\n" , sum) ;

return 0 ;

}

第二届CCF第三题（KMP）

|  |
| --- |
| 201409-3 |
| 试题名称： | 字符串匹配 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　给出一个字符串和多行文字，在这些文字中找到字符串出现的那些行。你的程序还需支持大小写敏感选项：当选项打开时，表示同一个字母的大写和小写看作不同的字符；当选项关闭时，表示同一个字母的大写和小写看作相同的字符。  **输入格式**  　　输入的第一行包含一个字符串S，由大小写英文字母组成。 　　第二行包含一个数字，表示大小写敏感的选项，当数字为0时表示大小写不敏感，当数字为1时表示大小写敏感。 　　第三行包含一个整数n，表示给出的文字的行数。 　　接下来n行，每行包含一个字符串，字符串由大小写英文字母组成，不含空格和其他字符。  **输出格式**  　　输出多行，每行包含一个字符串，按出现的顺序依次给出那些包含了字符串S的行。  **样例输入**  Hello 1 5 HelloWorld HiHiHelloHiHi GrepIsAGreatTool HELLO HELLOisNOTHello  **样例输出**  HelloWorld HiHiHelloHiHi HELLOisNOTHello  **样例说明**  　　在上面的样例中，第四个字符串虽然也是Hello，但是大小写不正确。如果将输入的第二行改为0，则第四个字符串应该输出。  **评测用例规模与约定**  　　1<=n<=100，每个字符串的长度不超过100。 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# define LL long long

using namespace std ;

const int N = 110;

int next[N];

char S[N], T[N] , S1[N];

int slen, tlen;

void getNext()

{

int j, k;

j = 0; k = -1; next[0] = -1;

while(j < tlen)

if(k == -1 || T[j] == T[k])

next[++j] = ++k;

else

k = next[k];

}

int KMP\_Count()

{

int ans = 0;

int i, j = 0;

if(slen == 1 && tlen == 1)

{

if(S[0] == T[0])

return 1;

else

return 0;

}

getNext();

for(i = 0; i < slen; i++)

{

while(j > 0 && S[i] != T[j])

j = next[j];

if(S[i] == T[j])

j++;

if(j == tlen)

{

ans++;

j = next[j];

}

}

return ans;

}

int main ()

{

//freopen("in.txt","r",stdin) ;

int n , sum , tag ;

int i ;

cin>>T ;

tlen = strlen(T);

cin>>tag>>n ;

if(tag == 1)

{

while(n--)

{

cin>>S ;

slen = strlen(S);

sum = KMP\_Count() ;

if (sum >= 1)

cout<<S<<endl ;

}

}

else

{

for (i = 0 ; i < tlen ; i++)

T[i] = tolower(T[i]) ;

while(n--)

{

cin>>S1 ;

slen = strlen(S1);

for (i = 0 ; i < slen ; i++)

S[i] = tolower(S1[i]) ;

sum = KMP\_Count() ;

if (sum >= 1)

cout<<S1<<endl ;

}

}

return 0 ;

}

第二届CCF第四题(bfs)

|  |
| --- |
| 201409-4 |
| 试题名称： | 最优配餐 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　栋栋最近开了一家餐饮连锁店，提供外卖服务。随着连锁店越来越多，怎么合理的给客户送餐成为了一个急需解决的问题。 　　栋栋的连锁店所在的区域可以看成是一个n×n的方格图（如下图所示），方格的格点上的位置上可能包含栋栋的分店（绿色标注）或者客户（蓝色标注），有一些格点是不能经过的（红色标注）。 　　方格图中的线表示可以行走的道路，相邻两个格点的距离为1。栋栋要送餐必须走可以行走的道路，而且不能经过红色标注的点。   　　送餐的主要成本体现在路上所花的时间，每一份餐每走一个单位的距离需要花费1块钱。每个客户的需求都可以由栋栋的任意分店配送，每个分店没有配送总量的限制。 　　现在你得到了栋栋的客户的需求，请问在最优的送餐方式下，送这些餐需要花费多大的成本。  **输入格式**  　　输入的第一行包含四个整数n, m, k, d，分别表示方格图的大小、栋栋的分店数量、客户的数量，以及不能经过的点的数量。 　　接下来m行，每行两个整数xi, yi，表示栋栋的一个分店在方格图中的横坐标和纵坐标。 　　接下来k行，每行三个整数xi, yi, ci，分别表示每个客户在方格图中的横坐标、纵坐标和订餐的量。（注意，可能有多个客户在方格图中的同一个位置） 　　接下来d行，每行两个整数，分别表示每个不能经过的点的横坐标和纵坐标。  **输出格式**  　　输出一个整数，表示最优送餐方式下所需要花费的成本。  **样例输入**  10 2 3 3 1 1 8 8 1 5 1 2 3 3 6 7 2 1 2 2 2 6 8  **样例输出**  29  **评测用例规模与约定**  　　前30%的评测用例满足：1<=n <=20。 　　前60%的评测用例满足：1<=n<=100。 　　所有评测用例都满足：1<=n<=1000，1<=m, k, d<=n^2。可能有多个客户在同一个格点上。每个客户的订餐量不超过1000，每个客户所需要的餐都能被送到。 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# include <queue>

# define LL long long

using namespace std ;

int map[110][110] ;

bool v[110][110] ;

int n , m , k , d ;

struct kehu

{

int x ;

int y ;

int num ;

}s[10010];

struct node

{

int x , y , step ;

};

int dx[] = {1,-1,0,0} ;

int dy[] = {0,0,1,-1} ;

int bfs(int sx , int sy )

{

queue<node> q ;

int i , fx ,fy ;

node now , t ;

now.x = sx ;

now.y = sy ;

now.step = 0 ;

q.push(now) ;

memset(v , 0 , sizeof(v)) ;

v[sx][sy] = 1 ;

while(!q.empty())

{

now = q.front() ;

q.pop() ;

for (i = 0 ; i < 4 ; i++)

{

fx = now.x + dx[i] ;

fy = now.y + dy[i] ;

if (fx<1 || fy<1 || fx> n || fy> n || map[fx][fy] == 1 || v[fx][fy] == 1)

continue ;

if (map[fx][fy] == 2)

{

return now.step+1 ;

}

t.x = fx ;

t.y = fy ;

t.step = now.step+1 ;

q.push(t) ;

v[fx][fy] = 1 ;

}

}

}

int main()

{

//freopen("in.txt","r",stdin) ;

int i , j ;

int t1 , t2 ;

int st = 0 ;

int sum = 0 ;

scanf("%d%d%d%d" , &n , &m , &k , &d) ;

for (i = 0 ; i < m ; i++)

{

scanf("%d%d" , &t1 , &t2) ;

map[t1][t2] = 2 ;

}

for (i = 0 ; i < k ; i++)

scanf("%d%d%d" , &s[i].x , &s[i].y ,&s[i].num) ;

for (i = 0 ; i < d ; i++)

{

scanf("%d%d" , &t1 , &t2) ;

map[t1][t2] = 1 ;

}

for (i = 0 ; i < k ; i++)

{

st = bfs(s[i].x , s[i].y ) ;

sum += (st \* s[i].num) ;

}

printf("%d\n" , sum) ;

return 0 ;

}

第二届CCF第五题

|  |
| --- |
| 201409-5 |
| 试题名称： | 拼图 |
| 时间限制： | 3.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　给出一个n×m的方格图，现在要用如下L型的积木拼到这个图中，使得方格图正好被拼满，请问总共有多少种拼法。其中，方格图的每一个方格正好能放积木中的一块。积木可以任意旋转。  **输入格式**  　　输入的第一行包含两个整数n, m，表示方格图的大小。  **输出格式**  　　输出一行，表示可以放的方案数，由于方案数可能很多，所以请输出方案数除以1,000,000,007的余数。  **样例输入**  6 2  **样例输出**  4  **样例说明**  　　四种拼法如下图所示：  **评测用例规模与约定**  　　在评测时将使用10个评测用例对你的程序进行评测。 　　评测用例1和2满足：1<=n<=30，m=2。 　　评测用例3和4满足：1<=n, m<=6。 　　评测用例5满足：1<=n<=100，1<=m<=6。 　　评测用例6和7满足：1<=n<=1000，1<=m<=6。 　　评测用例8、9和10满足：1<=n<=10^15，1<=m<=7。 |

第三届CCF第一题

|  |
| --- |
| 201412-1 |
| 试题名称： | 门禁系统 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　涛涛最近要负责图书馆的管理工作，需要记录下每天读者的到访情况。每位读者有一个编号，每条记录用读者的编号来表示。给出读者的来访记录，请问每一条记录中的读者是第几次出现。  **输入格式**  　　输入的第一行包含一个整数n，表示涛涛的记录条数。 　　第二行包含n个整数，依次表示涛涛的记录中每位读者的编号。  **输出格式**  　　输出一行，包含n个整数，由空格分隔，依次表示每条记录中的读者编号是第几次出现。  **样例输入**  5 1 2 1 1 3  **样例输出**  1 1 2 3 1  **评测用例规模与约定**  　　1≤n≤1,000，读者的编号为不超过n的正整数。 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# define LL long long

using namespace std ;

int a[1010] ;

int main ()

{

//freopen("in.txt","r",stdin) ;

int n ;

scanf("%d" , &n) ;

int i , x ;

for (i = 1 ; i <= n-1 ; i++)

{

scanf("%d" , &x) ;

a[x]++ ;

printf("%d " , a[x]) ;

}

scanf("%d" , &x) ;

a[x]++ ;

printf("%d\n" , a[x]) ;

return 0 ;

}

第三届CCF第二题

|  |  |
| --- | --- |
| 问题描述： | **问题描述**  　　在图像编码的算法中，需要将一个给定的方形矩阵进行Z字形扫描(Zigzag Scan)。给定一个n×n的矩阵，Z字形扫描的过程如下图所示：  　　对于下面的4×4的矩阵， 　　1 5 3 9 　　3 7 5 6 　　9 4 6 4 　　7 3 1 3 　　对其进行Z字形扫描后得到长度为16的序列： 　　1 5 3 9 7 3 9 5 4 7 3 6 6 4 1 3 　　请实现一个Z字形扫描的程序，给定一个n×n的矩阵，输出对这个矩阵进行Z字形扫描的结果。  **输入格式**  　　输入的第一行包含一个整数n，表示矩阵的大小。 　　输入的第二行到第n+1行每行包含n个正整数，由空格分隔，表示给定的矩阵。  **输出格式**  　　输出一行，包含n×n个整数，由空格分隔，表示输入的矩阵经过Z字形扫描后的结果。  **样例输入**  4 1 5 3 9 3 7 5 6 9 4 6 4 7 3 1 3  **样例输出**  1 5 3 9 7 3 9 5 4 7 3 6 6 4 1 3  **评测用例规模与约定**  　　1≤n≤500，矩阵元素为不超过1000的正整数。 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# define LL long long

using namespace std ;

int map[500][500] ;

int main ()

{

//freopen("in.txt","r",stdin) ;

int n ;

scanf("%d" , &n) ;

int i , j ;

for (i = 1 ; i <= n ; i++)

for (j = 1 ; j <= n ; j++)

scanf("%d" , &map[i][j]) ;

i = j = 1 ;

int tag1 = 1 ;

int tag2 = 0 ;

while(i != n || j != n)

{

while (tag1)

{

if (i != 1 && j != n)

{

printf("%d " , map[i][j]) ;

i-- ;

j++ ;

}

else if (j == n)

{

printf("%d " , map[i][j]) ;

i++ ;

tag1 = 0 ;

tag2 = 1 ;

}

else if (i == 1)

{

printf("%d " , map[i][j]) ;

j++ ;

tag1 = 0 ;

tag2 = 1 ;

}

}

while(tag2)

{

if (i != n && j != 1)

{

printf("%d " , map[i][j]) ;

i++ ;

j-- ;

}

else if (i == n)

{

printf("%d " , map[i][j]) ;

j++ ;

tag2 = 0 ;

tag1 = 1 ;

}

else if (j == 1)

{

printf("%d " , map[i][j]) ;

i++ ;

tag2 = 0 ;

tag1 = 1 ;

}

}

}

printf("%d\n" , map[i][j]) ;

return 0 ;

}

第三届CCF第三题

|  |
| --- |
| 201412-3 |
| 试题名称： | 集合竞价 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　某股票交易所请你编写一个程序，根据开盘前客户提交的订单来确定某特定股票的开盘价和开盘成交量。 　　该程序的输入由很多行构成，每一行为一条记录，记录可能有以下几种： 　　1. buy p s 表示一个购买股票的买单，每手出价为p，购买股数为s。 　　2. sell p s 表示一个出售股票的卖单，每手出价为p，出售股数为s。 　　3. cancel i表示撤销第i行的记录。 　　如果开盘价为p0，则系统可以将所有出价至少为p0的买单和所有出价至多为p0的卖单进行匹配。因此，此时的开盘成交量为出价至少为p0的买单的总股数和所有出价至多为p0的卖单的总股数之间的较小值。 　　你的程序需要确定一个开盘价，使得开盘成交量尽可能地大。如果有多个符合条件的开盘价，你的程序应当输出最高的那一个。  **输入格式**  　　输入数据有任意多行，每一行是一条记录。保证输入合法。股数为不超过108的正整数，出价为精确到恰好小数点后两位的正实数，且不超过10000.00。  **输出格式**  　　你需要输出一行，包含两个数，以一个空格分隔。第一个数是开盘价，第二个是此开盘价下的成交量。开盘价需要精确到小数点后恰好两位。  **样例输入**  buy 9.25 100 buy 8.88 175 sell 9.00 1000 buy 9.00 400 sell 8.92 400 cancel 1 buy 100.00 50  **样例输出**  9.00 450  **评测用例规模与约定**  　　对于100%的数据，输入的行数不超过5000。 |

第三届CCF第四题(最小生成树)

|  |
| --- |
| 201412-4 |
| 试题名称： | 最优灌溉 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　雷雷承包了很多片麦田，为了灌溉这些麦田，雷雷在第一个麦田挖了一口很深的水井，所有的麦田都从这口井来引水灌溉。 　　为了灌溉，雷雷需要建立一些水渠，以连接水井和麦田，雷雷也可以利用部分麦田作为“中转站”，利用水渠连接不同的麦田，这样只要一片麦田能被灌溉，则与其连接的麦田也能被灌溉。 　　现在雷雷知道哪些麦田之间可以建设水渠和建设每个水渠所需要的费用（注意不是所有麦田之间都可以建立水渠）。请问灌溉所有麦田最少需要多少费用来修建水渠。  **输入格式**  　　输入的第一行包含两个正整数n, m，分别表示麦田的片数和雷雷可以建立的水渠的数量。麦田使用1, 2, 3, ……依次标号。 　　接下来m行，每行包含三个整数ai, bi, ci，表示第ai片麦田与第bi片麦田之间可以建立一条水渠，所需要的费用为ci。  **输出格式**  　　输出一行，包含一个整数，表示灌溉所有麦田所需要的最小费用。  **样例输入**  4 4 1 2 1 2 3 4 2 4 2 3 4 3  **样例输出**  6  **样例说明**  　　建立以下三条水渠：麦田1与麦田2、麦田2与麦田4、麦田4与麦田3。  **评测用例规模与约定**  　　前20%的评测用例满足：n≤5。 　　前40%的评测用例满足：n≤20。 　　前60%的评测用例满足：n≤100。 　　所有评测用例都满足：1≤n≤1000，1≤m≤100,000，1≤ci≤10,000。 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# define LL long long

using namespace std ;

int n ;

const int MAXN=1010;

const int MAXM=100010;

int F[MAXN];

struct Edge

{

int u,v,w;

}edge[MAXM];

int tol;

void addedge(int u,int v,int w)

{

edge[tol].u=u;

edge[tol].v=v;

edge[tol++].w=w;

}

bool cmp(Edge a,Edge b)

{

return a.w<b.w;

}

int find(int x)

{

if(F[x]==-1)return x;

else return F[x]=find(F[x]);

}

int Kruskal()

{

memset(F,-1,sizeof(F));

sort(edge,edge+tol,cmp);

int cnt=0;

int ans=0;

for(int i=0;i<tol;i++)

{

int u=edge[i].u;

int v=edge[i].v;

int w=edge[i].w;

int t1=find(u);

int t2=find(v);

if(t1!=t2)

{

ans+=w;

F[t1]=t2;

cnt++;

}

if(cnt==n-1)break;

}

if(cnt<n-1)return -1;

else return ans;

}

int main()

{

//freopen("in.txt","r",stdin) ;

int m ;

scanf("%d %d" , &n , &m) ;

int i ;

int u , v , w ;

tol = 0 ;

while(m--)

{

scanf("%d %d %d" , &u , &v , &w) ;

addedge(u , v , w) ;

addedge(v , u , w) ;

}

int k = Kruskal() ;

printf("%d\n" , k) ;

return 0 ;

}

第三届CCF第五题

|  |
| --- |
| 201412-5 |
| 试题名称： | 货物调度 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　某公司要处理一个周期性的物流问题。 　　有n个城市，第i个城市在每周的第j(1≤j≤7) 天会生产aij吨某种货物，同时需要消耗bij吨该种货物。已知每周的产量等于消耗量（即aij之和等于bij之和）。 　　城市之间有m条道路，第k条道路连接了城市sk和tk。一条道路上运输1吨货物有一个固定的成本ck。道路都可以双向使用。每天运输的货物量没有限制。城市之间的距离并不远，货物可以从任意一个城市运输到任意另一个城市并且在当天到达。 　　货物如果在当天没有被消耗掉，就需要存放在仓库里过夜。第i个城市的仓库容量为vi，存放1 吨货物过一夜所需的成本是wi。 　　请你计算该公司如果每周循环性地按照一个固定的流程调度货物的话，该公司在最优方案下每周需要为货物的运输和存储消耗多少成本。  **输入格式**  　　输入的第一行有两个正整数n和m，即城市的个数和道路的条数。 　　接下来有n行，每行包含16个整数，用以描述第i个城市的相关数据。其中第i行包含的数为ai1, ai2, ai3, ai4, ai5, ai6, ai7, bi1, bi2, bi3, bi4, bi5, bi6, bi7, vi, wi。 　　接下来有m行，每行包含3个整数，用以描述一条道路的相关数据。其中第k行包含的数为sk, tk和ck。 　　输入数据中城市的编号均为1到n之间。输入数据的每行的行首行尾均保证没有空格，两个数之间恰好被一个空格隔开。  **输出格式**  　　你只需要输出一个数，即最优方案下每周的支出。  **样例输入**  3 3 0 0 0 0 5 0 0 0 0 0 0 0 0 0 2 4 0 0 0 0 0 0 0 2 0 0 0 0 0 0 2 1 0 0 0 0 0 0 0 0 0 3 0 0 0 0 2 5 1 2 1 1 3 5 2 3 1  **样例输出**  67  **样例说明**  　　城市1 每周五生产5 吨货物，把其中2 吨运到存储费用低廉的城市2 存储，把1 吨运到城市3 存储，剩下的2 吨留在城市1。 　　在次周一的时候城市2 会消耗掉存放在那里的2 吨货物。为了节约存储成本，将囤放在城市1 的货物运到城市2 存放。周三再将所有货物运到城市3 以满足该城市的需求。 　　在此方案下，每周的运输成本为8，每周的存储成本为59，因此每周的总支出为67。  **评测用例规模与约定**  　　对于100%的数据，1≤n≤100，1≤m≤500，0≤aij,bij,vi≤100，1≤wi,ck≤100。 |

第四届CCF第一题

|  |
| --- |
| 201503-1 |
| 试题名称： | 图像旋转 |
| 时间限制： | 5.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　旋转是图像处理的基本操作，在这个问题中，你需要将一个图像逆时针旋转90度。 　　计算机中的图像表示可以用一个矩阵来表示，为了旋转一个图像，只需要将对应的矩阵旋转即可。  **输入格式**  　　输入的第一行包含两个整数n, m，分别表示图像矩阵的行数和列数。 　　接下来n行每行包含m个整数，表示输入的图像。  **输出格式**  　　输出m行，每行包含n个整数，表示原始矩阵逆时针旋转90度后的矩阵。  **样例输入**  2 3 1 5 3 3 2 4  **样例输出**  3 4 5 2 1 3  **评测用例规模与约定**  　　1 ≤ n, m ≤ 1,000，矩阵中的数都是不超过1000的非负整数。 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# define LL long long

using namespace std ;

int a[1010][1010] ;

int main ()

{

//freopen("in.txt","r",stdin) ;

int n , m ;

scanf("%d%d" , &n , &m) ;

int i , j ;

for (i = 0 ; i < n ; i++)

for (j = 0 ; j < m ; j++)

scanf("%d" , &a[i][j]) ;

for (i = m-1 ; i >= 0 ; i--)

{

for (j = 0 ; j < n ; j++)

{

printf("%d " , a[j][i]) ;

}

printf("\n") ;

}

return 0 ;

}

第四届CCF第二题

|  |
| --- |
| 201503-2 |
| 试题名称： | 数字排序 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　给定n个整数，请统计出每个整数出现的次数，按出现次数从多到少的顺序输出。  **输入格式**  　　输入的第一行包含一个整数n，表示给定数字的个数。 　　第二行包含n个整数，相邻的整数之间用一个空格分隔，表示所给定的整数。  **输出格式**  　　输出多行，每行包含两个整数，分别表示一个给定的整数和它出现的次数。按出现次数递减的顺序输出。如果两个整数出现的次数一样多，则先输出值较小的，然后输出值较大的。  **样例输入**  12 5 2 3 3 1 3 4 2 5 2 3 5  **样例输出**  3 4 2 3 5 3 1 1 4 1  **评测用例规模与约定**  　　1 ≤ n ≤ 1000，给出的数都是不超过1000的非负整数。 |

# include <iostream>

# include <cstdio>

# include <cstring>

# include <algorithm>

# include <cmath>

# define LL long long

using namespace std ;

struct shu

{

int num ;

int id ;

}a[1010];

bool cmp(shu x , shu y)

{

if (x.num == y.num)

return x.id < y.id ;

else

return x.num > y.num ;

}

int main ()

{

//freopen("in.txt","r",stdin) ;

int n ;

scanf("%d" , &n) ;

int i , x ;

for (i = 1 ; i <= 1005 ; i++)

{

a[i].id = i ;

a[i].num = 0 ;

}

while(n--)

{

scanf("%d" , &x) ;

a[x].num++ ;

}

sort(a+1 , a+1005, cmp) ;

for (i = 1 ; i <= 1005 ; i++)

{

if (a[i].num == 0)

break ;

printf("%d %d\n" , a[i].id , a[i].num) ;

}

return 0 ;

}

第四届CCF第三题

|  |
| --- |
| 201503-3 |
| 试题名称： | 节日 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　有一类节日的日期并不是固定的，而是以“a月的第b个星期c”的形式定下来的，比如说母亲节就定为每年的五月的第二个星期日。 　　现在，给你a，b，c和y1, y2(1850 ≤ y1, y2 ≤ 2050)，希望你输出从公元y1年到公元y2年间的每年的a月的第b个星期c的日期。 　　提示：关于闰年的规则：年份是400的整数倍时是闰年，否则年份是4的倍数并且不是100的倍数时是闰年，其他年份都不是闰年。例如1900年就不是闰年，而2000年是闰年。 　　为了方便你推算，已知1850年1月1日是星期二。  **输入格式**  　　输入包含恰好一行，有五个整数a, b, c, y1, y2。其中c=1, 2, ……, 6, 7分别表示星期一、二、……、六、日。  **输出格式**  　　对于y1和y2之间的每一个年份，包括y1和y2，按照年份从小到大的顺序输出一行。 　　如果该年的a月第b个星期c确实存在，则以"yyyy/mm/dd"的格式输出，即输出四位数的年份，两位数的月份，两位数的日期，中间用斜杠“/”分隔，位数不足时前补零。 　　如果该年的a月第b个星期c并不存在，则输出"none"（不包含双引号)。  **样例输入**  5 2 7 2014 2015  **样例输出**  2014/05/11 2015/05/10  **评测用例规模与约定**  　　所有评测用例都满足：1 ≤ a ≤ 12，1 ≤ b ≤ 5，1 ≤ c ≤ 7，1850 ≤ y1, y2 ≤ 2050。 |

第四届CCF第四题

|  |
| --- |
| 201503-4 |
| 试题名称： | 网络延时 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　给定一个公司的网络，由n台交换机和m台终端电脑组成，交换机与交换机、交换机与电脑之间使用网络连接。交换机按层级设置，编号为1的交换机为根交换机，层级为1。其他的交换机都连接到一台比自己上一层的交换机上，其层级为对应交换机的层级加1。所有的终端电脑都直接连接到交换机上。 　　当信息在电脑、交换机之间传递时，每一步只能通过自己传递到自己所连接的另一台电脑或交换机。请问，电脑与电脑之间传递消息、或者电脑与交换机之间传递消息、或者交换机与交换机之间传递消息最多需要多少步。  **输入格式**  　　输入的第一行包含两个整数n, m，分别表示交换机的台数和终端电脑的台数。 　　第二行包含n - 1个整数，分别表示第2、3、……、n台交换机所连接的比自己上一层的交换机的编号。第i台交换机所连接的上一层的交换机编号一定比自己的编号小。 　　第三行包含m个整数，分别表示第1、2、……、m台终端电脑所连接的交换机的编号。  **输出格式**  　　输出一个整数，表示消息传递最多需要的步数。  **样例输入**  4 2 1 1 3 2 1  **样例输出**  4  **样例说明**  　　样例的网络连接模式如下，其中圆圈表示交换机，方框表示电脑：  　　其中电脑1与交换机4之间的消息传递花费的时间最长，为4个单位时间。  **样例输入**  4 4 1 2 2 3 4 4 4  **样例输出**  4  **样例说明**  　　样例的网络连接模式如下：  　　其中电脑1与电脑4之间的消息传递花费的时间最长，为4个单位时间。  **评测用例规模与约定**  　　前30%的评测用例满足：n ≤ 5, m ≤ 5。 　　前50%的评测用例满足：n ≤ 20, m ≤ 20。 　　前70%的评测用例满足：n ≤ 100, m ≤ 100。 　　所有评测用例都满足：1 ≤ n ≤ 10000，1 ≤ m ≤ 10000。 |

第四届CCF第五题

|  |
| --- |
| 201503-5 |
| 试题名称： | 最小花费 |
| 时间限制： | 4.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　C国共有n个城市。有n-1条双向道路，每条道路连接两个城市，任意两个城市之间能互相到达。小R来到C国旅行，他共规划了m条旅行的路线，第i条旅行路线的起点是si，终点是ti。在旅行过程中，小R每行走一单位长度的路需要吃一单位的食物。C国的食物只能在各个城市中买到，而且不同城市的食物价格可能不同。 　　然而，小R不希望在旅行中为了购买较低价的粮食而绕远路，因此他总会选择最近的路走。现在，请你计算小R规划的每条旅行路线的最小花费是多少。  **输入格式**  　　第一行包含2个整数n和m。 　　第二行包含n个整数。第i个整数wi表示城市i的食物价格。 　　接下来n-1行，每行包括3个整数u, v, e，表示城市u和城市v之间有一条长为e的双向道路。 　　接下来m行，每行包含2个整数si和ti，分别表示一条旅行路线的起点和终点。  **输出格式**  　　输出m行，分别代表每一条旅行方案的最小花费。  **样例输入**  6 4 1 7 3 2 5 6 1 2 4 1 3 5 2 4 1 3 5 2 3 6 1 2 5 4 6 6 4 5 6  **样例输出**  35 16 26 13  **样例说明**  　　对于第一条路线，小R会经过2->1->3->5。其中在城市2处以7的价格购买4单位粮食，到城市1时全部吃完，并用1的价格购买7单位粮食，然后到达终点。  **评测用例规模与约定**  　　前10%的评测用例满足：n, m ≤ 20, wi ≤ 20； 　　前30%的评测用例满足：n, m ≤ 200； 　　另有40%的评测用例满足：一个城市至多与其它两个城市相连。 　　所有评测用例都满足：1 ≤ n, m ≤ 105，1 ≤ wi ≤ 106，1 ≤ e ≤ 10000。 |