1.[编程题] 最大乘积

时间限制：1秒

空间限制：32768K

给定一个无序数组，包含正数、负数和0，要求从中找出3个数的乘积，使得乘积最大，要求时间复杂度：O(n)，空间复杂度：O(1)

**输入描述:**

无序整数数组A[n]

**输出描述:**

满足条件的最大乘积

**输入例子1:**

3 4 1 2

**输出例子1:**

24

2. [编程题] 大整数相乘

时间限制：1秒

空间限制：32768K

有两个用字符串表示的非常大的大整数,算出他们的乘积，也是用字符串表示。不能用系统自带的大整数类型。

**输入描述:**

空格分隔的两个字符串，代表输入的两个大整数

**输出描述:**

输入的乘积，用字符串表示

**输入例子1:**

72106547548473106236 982161082972751393

**输出例子1:**

70820244829634538040848656466105986748

3. [编程题] 六一儿童节

时间限制：1秒

空间限制：32768K

六一儿童节，老师带了很多好吃的巧克力到幼儿园。每块巧克力j的重量为w[j]，对于每个小朋友i，当他分到的巧克力大小达到h[i] (即w[j]>=h[i])，他才会上去表演节目。老师的目标是将巧克力分发给孩子们，使得最多的小孩上台表演。可以保证每个w[i]> 0且不能将多块巧克力分给一个孩子或将一块分给多个孩子。

**输入描述:**

第一行：n，表示h数组元素个数  
 第二行：n个h数组元素  
 第三行：m，表示w数组元素个数  
 第四行：m个w数组元素

**输出描述:**

上台表演学生人数

**输入例子1:**

3   
 2 2 3  
 2  
 3 1

**输出例子1:**

1

4. [编程题] 迷宫寻路

时间限制：1秒

空间限制：131072K

假设一个探险家被困在了地底的迷宫之中，要从当前位置开始找到一条通往迷宫出口的路径。迷宫可以用一个二维矩阵组成，有的部分是墙，有的部分是路。迷宫之中有的路上还有门，每扇门都在迷宫的某个地方有与之匹配的钥匙，只有先拿到钥匙才能打开门。请设计一个算法，帮助探险家找到脱困的最短路径。如前所述，迷宫是通过一个二维矩阵表示的，每个元素的值的含义如下 0-墙，1-路，2-探险家的起始位置，3-迷宫的出口，大写字母-门，小写字母-对应大写字母所代表的门的钥匙

**输入描述:**

迷宫的地图，用二维矩阵表示。第一行是表示矩阵的行数和列数M和N

后面的M行是矩阵的数据，每一行对应与矩阵的一行（中间没有空格）。M和N都不超过100, 门不超过10扇。

**输出描述:**

路径的长度，是一个整数

**输入例子1:**

5 5

02111

01a0A

01003

01001

01111

**输出例子1:**

7

参考答案：

1. #include<iostream>

#include<algorithm>

#include<limits.h>

using namespace std;

int main(){

    int n;

    cin>>n;

    vector<long long>nums(n);

    for(int i=0;i<n;i++){

        cin>>nums[i];

    }

    long long max1=1,max2=1,max3=1,min1=1,min2=1;

    for(int i=0;i<n;i++){

        if(nums[i]>max1){

           max3=max2;

           max2=max1;

           max1=nums[i];

        }

        else if(nums[i]>max2){

           max3=max2;

           max2=nums[i];

        }

        else if(nums[i]>max3){

           max3=nums[i];

        }

        else if(nums[i]<min1){

           min2=min1;

           min1=nums[i];

        }

        else if(nums[i]<min2){

           min2=nums[i];

        }

    }

    long long result1,result2,result;

    result1=max1\*max2\*max3;

    result2=max1\*min1\*min2;

    result=max(result1,result2);

    cout<<result<<endl;

    return 0;

}

2. //AC代码:

#include<stdio.h>

#include<string>

#include<iostream>

using namespace std;

const int L=11000;

string mul(string,string);

int main(){

    string x,y;

    while(cin>>x>>y)

        cout<<mul(x,y)<<endl;

}

string mul(string a,string b) {

    string s;

    int na[L],nb[L],nc[L],La=a.size(),Lb=b.size(),i,j;

    fill(na,na+L,0);fill(nb,nb+L,0);fill(nc,nc+L,0);

    for(i=La-1;i>=0;i--) na[La-i]=a[i]-'0';

    for(i=Lb-1;i>=0;i--) nb[Lb-i]=b[i]-'0';

    for(i=1;i<=La;i++)

        for(j=1;j<=Lb;j++)

        nc[i+j-1]+=na[i]\*nb[j];

    for(i=1;i<=La+Lb;i++)

        nc[i+1]+=nc[i]/10,nc[i]%=10;

    if(nc[La+Lb]) s+=nc[La+Lb]+'0';

    for(i=La+Lb-1;i>=1;i--)

        s+=nc[i]+'0';

    return s;

}

3. //AC代码:

#include<stdio.h>

#include<algorithm>

#include<vector>

using namespace std;

int main(){

    int N,M,i,j;

    //freopen("input.txt","r",stdin);

    scanf("%d",&N);

    vector<int> child(N);

    for(i=0;i<N;i++) scanf("%d",&child[i]);

    scanf("%d",&M);

    vector<int> cho(M);

    for(i=0;i<M;i++) scanf("%d",&cho[i]);

    sort(child.begin(),child.end());

    sort(cho.begin(),cho.end());

    int res=0;

    for(i=0,j=0;i<M&&j<N;i++)

        if(cho[i]>=child[j])

            res++,j++;

    printf("%d\n",res);

}

4. //AC代码:

#include<stdio.h>

#include<queue>

#include<string.h>

#include<vector>

using namespace std;

char G[105][105];

int book[105][105][1200],N,M;

int Next[4][2]={0,1,0,-1,1,0,-1,0};

int bfs(int,int);

struct node{

    int x,y,k,step;

    node(int x,int y,int k,int step):x(x),y(y),k(k),step(step){}

};

int main(){

    int i,j;

    //freopen("input.txt","r",stdin);

    while(scanf("%d%d",&N,&M)!=EOF){

        for(i=0;i<N;i++) scanf("%s",G[i]);

        memset(book,0,sizeof(book));

        int flag=0;

        for(i=0;i<N;i++){

            if(flag==1) break;

            for(j=0;j<M;j++)

                if(G[i][j]=='2'){

                    flag=1;

                    book[i][j][0]=1;

                    printf("%d\n",bfs(i,j));

                    break;

                }

        }

    }

}

int bfs(int startX,int startY){

    queue<node> Q;

    Q.push(node(startX,startY,0,0));

    while(!Q.empty()){

        node head=Q.front();Q.pop();

        if(G[head.x][head.y]=='3') return head.step;

        for(int i=0;i<4;i++){

            int nx=head.x+Next[i][0],ny=head.y+Next[i][1];

            if(nx>=N||nx<0||ny>=M||ny<0||G[nx][ny]=='0') continue;

            int key=head.k;

            if('a'<=G[nx][ny]&&G[nx][ny]<='z') key=key|(1<<(G[nx][ny]-'a'));

            if('A'<=G[nx][ny]&&G[nx][ny]<='Z'&&(key&(1<<(G[nx][ny]-'A')))==0) continue;

            if(!book[nx][ny][key]){

                book[nx][ny][key]=1;

                Q.push(node(nx,ny,key,head.step+1));

            }

        }

    }

    return 0;

}//这题就是普通的bfs多了‘钥匙’这个状态

 //所以book[x][y][key]的意义就是 横坐标为x,纵坐标为y,钥匙状态为key的点是否访问过

 //钥匙的状态 就用二进制数表示 最多10 把钥匙 那就是1024

 //比如我现在有第二把钥匙和第四把钥匙  那么我的钥匙状态就是 0101000000 也就是 320