1/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)

//思路：先统计一下输入的所有数中，零的个数、>0 的个数、<0 的个数

//然后枚举各种情况，确定一种情况，然后扫描整个数组两遍就行了

```
#include<iostream>
usingnamespacestd;
///拼多多
#if1
intmain()
{
intn;
cin>>n;
int*arr=newint[n];
for(inti=0;i<n;i++)
{
cin>>arr[i];
}
longlongposnum=0;
longlongminnum=0;
longlongzeronum=0;
longlongres;
```

```
for(inti=0;i<n;i++)
{
if(arr[i]>0)
{
posnum++;
}
elseif(arr[i]<0)
{
minnum++;
}
else
{
zeronum++;
}
}
if(posnum==0&&minnum==0)
{
res=0;
}
elseif(minnum==0)
{
if(posnum-zeronum<3)
```

```
{
res=0;
}
else
{
longlongmax=0;
longlongmid=0;
longlongmin=0;
longlongmax_index=-1;
longlongmid_index=-1;
longlongmin_index=-1;
for(inti=0;i<n;i++)
{
if(arr[i]>=max)
{
max=arr[i];
max_index=i;
}
}
for(inti=0;i<n;i++)
{
if(i==max_index)
```

```
{
continue;
}
if(arr[i]>=mid&&arr[i]<=max)
{
mid=arr[i];
mid_index=i;
}
}
for(inti=0;i<n;i++)
{
if(i==max_index||i==mid_index)
{
continue;
}
if(arr[i]>=min&&arr[i]<=mid&&arr[i]<=max)
{
min=arr[i];
min_index=i;
}
}
res=max*mid*min;
```

```
}
}
elseif(posnum==0)
{
if(minnum-zeronum<3)
{
res=0;
}
else
{
longlongmax=0x80000000;
longlongmid=0x80000000;
longlongmin=0x80000000;
longlongmax_index=-1;
longlongmid_index=-1;
longlongmin_index=-1;
for(inti=0;i<n;i++)
{
if(arr[i]>=max)
{
max=arr[i];
max_index=i;
```

```
}
}
for(inti=0;i<n;i++)
{
if(i==max_index)
{
continue;
}
if(arr[i]>=mid&&arr[i]<=max)
{
mid=arr[i];
mid_index=i;
}
}
for(inti=0;i<n;i++)
{
if(i==max_index||i==mid_index)
{
continue;
}
if(arr[i]>=min&&arr[i]<=mid&&arr[i]<=max)
{
```

```
min=arr[i];
min_index=i;
}
}
res=max*mid*min;
}
}
else
{
longlongmin_num=0;
longlongmin_max=-1;
longlongmin_max_index=-1;
longlongmin_sec=-1;
longlongmin_sec_index=-1;
longlongtmp=0;
for(inti=0;i<n;i++)
{
if(arr[i]<0)
{
min_num++;
if(min_max>=arr[i])
{
```

```
min_max=arr[i];
min_max_index=i;
}
}
/*else
{
posnum++;
}*/
}
if(min_num>=2)
{
for(inti=0;i<n;i++)
{
if(arr[i]<0)
{
if(min_sec>=arr[i]&&min_sec>=min_max&&min_max_index!=i)
{
min_sec=arr[i];
min_sec_index=i;
}
}
}
```

```
intthird=0;
for(inti=0;i<n;i++)
{
if(arr[i]>0&&arr[i]>third)
{
third=arr[i];
}
}
tmp=min_sec*min_max*third;
}
longlongpos_num=0;
longlongmax_max=-1;
longlongmax_max_index=-1;
longlongmax_sec=-1;
longlongmax_sec_index=-1;
longlongtmp1=0;
for(inti=0;i<n;i++)
{
if(arr[i]>0)
{
pos_num++;
if(max_max<=arr[i])
```

```
{
max_max=arr[i];
max_max_index=i;
}
}
}
```

```
if(pos_num>=2)
{
for(inti=0;i<n;i++)
{
if(arr[i]>0)
{
if(max_sec<=arr[i]&&arr[i]<=max_max&&max_max_index!=i)
{
max_sec=arr[i];
max_sec_index=i;
}
}
}
```

```
intthird1=0;
for(inti=0;i<n;i++)
{
if(arr[i]>0&&arr[i]>third1&&i!=max_max_index&&i!=max_sec_index)
{
third1=arr[i];
}
}
tmp1=max_sec*max_max*third1;
}
if(tmp>=tmp1)
{
res=tmp;
}
else
{
res=tmp1;
}
}
cout<<res<<endl;
return0;
}
```

```
#endif
```

2/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。

```
#include<iostream>

#include<sstream>

#include<algorithm>

#include<cstring>

#include<string>

usingnamespacestd;

///拼多多

#if1

stringBigNumMultiply(conststring&strNum1,conststring&strNum2)

{

stringstrMultiply;

intbit=0;

intlen1=strNum1.size()-1;

intlen2=strNum2.size()-1;



for(inti=0;i<len1+len2+2;++i)

{

inttmp=0;

for(intj=i;j>=0;--j)
```

```
{
if(j>len1||(i-j)>len2)
continue;

tmp+=(strNum1[len1-j]-'0')*(strNum2[len2-(i-j)]-'0');
}
tmp+=bit;

if(tmp==0&&i==len1+len2+1)
break;
strMultiply+=tmp%10+'0';
bit=tmp/10;
}
reverse(strMultiply.begin(),strMultiply.end());

returnstrMultiply;
}

intmain()
{
stringstr1;
stringstr2;
```

```
cin>>str1;

cin>>str2;

/*getline(cin,str1);

getline(cin,str2);*/

stringres;

res=BigNumMultiply(str1,str2);

cout<<res<<endl;

return0;

}

#endif
```

3/六一儿童节，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。

```
//AC 代码:
#include<stdio.h>
#include<algorithm>
#include<vector>
usingnamespacestd;
intmain(){
intN,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());
intres=0;
for(i=0,j=0;i<M&&j<N;i++)
```

```
if(cho[i]>=child[j])
res++,j++;
printf("%d\n",res);
}
```

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

```
//AC 代码:
#include<stdio.h>
#include<queue>
#include<string.h>
#include<vector>
usingnamespacestd;
charG[105][105];
intbook[105][105][1200],N,M;
intNext[4][2]={0,1,0,-1,1,0,-1,0};
intbfs(int,int);
structnode{
intx,y,k,step;
node(intx,inty,intk,intstep):x(x),y(y),k(k),step(step){}
};
intmain(){
inti,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));
intflag=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;
}
}
```

```
}
}
intbfs(intstartX,intstartY){
queue<node>Q;
Q.push(node(startX,startY,0,0));
while(!Q.empty()){
nodehead=Q.front();Q.pop();
if(G[head.x][head.y]=='3')returnhead.step;
for(inti=0;i<4;i++){
intnx=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;
intkey=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)cont
inue;
if(!book[nx][ny][key]){
book[nx][ny][key]=1;
Q.push(node(nx,ny,key,head.step+1));
}
}
}
return0;
}//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
```