

岁阳是一种不具固定形体的能量生物。一些不怀好意的岁阳会扰乱正常的社会秩序。现在，霍霍被委派捉拿岁阳。霍霍会选择一个点 $(x1, y1)$ ，以 $(0, 0)$ 为左下角， $(x1, y1)$ 为右上角，平行于坐标轴，构成一个矩形陷阱，如果岁阳位于这个矩形内部或边界上，就可以成功捉拿岁阳。岁阳的位置被记作 $(x2, y2)$ 。

下面，给出坐标 $(xy)$ 的构成方法：

给定一个长度为 $n$ 的整数序列 $a$ ，和一个长度为 $m$ 的整数序列 $b$ ，从 $a$ 中选择其中一个数字作为 $x$ ，从 $b$ 中选择一个数字作为 $y$ ，这样可以得到一组坐标 $(x, y)$ 。

霍霍所选择的 $(x1, y1)$ 和岁阳的位置 $(x2, y2)$ 均通过上述办法产生。

你的任务是计算在所有可能的情形中，霍霍能够成功捉拿岁阳的情形数量。

输入描述：

第一行两个整数 $n$ 和 $m$ ，分别表示数组 $a$ 和 $b$ 的长度。

第二行 $n$ 个整数 $a_1, a_2, \dots, a_n$ ，表示序列 $a$ 。

第三行 $m$ 个整数 $b_1, b_2, \dots, b_m$ ，表示序列 $b$ 。

对于全部数据， $1 \leq n, m \leq 5 \times 10^4$ ， $1 \leq a_i, b_i \leq 10^9$

输出描述：

输出一行，一个整数，表示答案

样例输入：

3 3

1 2 3

1 1 3

样例输出：

42

Code:

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
const long long N=1e5;
long long n,m;
long long x[N],y[N]; //记录坐标
long long cnt_x[N],cnt_y[N]; //记录坐标中 x[i] y[i] 的个数
int main() {
    freopen("data4.in","r",stdin);
    int t;
    cin>>t;
    while(t--){
        cin>>n>>m;
        for(int i=0;i<n;i++){
            cin>>x[i];
            cnt_x[x[i]]++;
        }
        for(int i=0;i<m;i++){
            cin>>y[i];
            cnt_y[y[i]]++;
        }
        // 重小到排序坐标
        sort(x,x+n); // 1 2 3
        sort(y,y+m);
        //遍历 x 获取满足 x1<=x2 的数量
        long long cnt1=0,res=0;
        for(int i=0;i<n;i++){
            cnt1+=i+cnt_x[x[i]]; //统计 x<=x[i] 所有的数量
```

```

        cnt_x[x[i]]--;
    }
    //cout<<cnt1<<endl;
    //遍历 y 获取满足 x1<=x2 且 y1<=y2 的所有数量
    for(int i=0;i<m;i++){
        long long cnt2=i+cnt_y[y[i]]; //统计 y<=y[i] 的个数
        cnt_y[y[i]]--;
        res+=cnt1*cnt2;
    }
    cout<<res<<endl;
}
return 0;
}

```

Input:

```

5
3 3
1 2 3
1 1 3
12 11
123 -234 4535 76 6456 4324 98 54 3 324 55 98
65 7576 68 6456 45 78 546 -77 556 -6456 -987
7 7
-3123 123 5453 8696 7987 98 67
454 645 765 879 -978 78 78
7 4
4353 -6456 -7657 -6765 0 87 7
435 646 76 -98
3 4
-99889 -989 98
5345 6474 756 476

```

Output:

```

42
5214
812
280
60

```

运行结果:

```
D:\Users\dell\Desktop\米哈游笔试\岁阳.exe
42
5214
696·7987·98·67·
812
280
60
-6765·0·87·7·
-----
Process exited after 0.5158 seconds with return value
请按任意键继续. . .
```

在潜入下层工厂后，蓬莱寺九霄在其中一个房间中发现了 $N$  ( $1 \leq N \leq 5000$ ) 台机器。这些机器排成一排，从左到右依次编号为1、2、...、 $N$ 。且每台机器都有一个能量值，编号为 $i$ 的机器的能量值为 $a_i$  ( $-10^{12} \leq a_i \leq 10^{12}$ )。现在蓬莱寺九霄想要启动其中一些机器。在简单尝试后，她发现若两台启动的机器之间没有其他任何其他启动的机器，则这两台机器会尝试连接；但是若一台机器左右两边都有机器尝试与其连接，且左右两台机器能量值的平均值大于等于中间这台机器的能量值，则会使得中间的机器系统崩溃。现在所有的机器都已恢复关机状态，九霄想知道的是，在不引发机器系统崩溃的情况下，最多可以同时启动多少台机器？

输入描述：

输入的第一行包括一个整数 $N$ ，表示机器的数量。

第二行包括 $N$ 个整数  $a_1$ 、 $a_2$ 、...、 $a_N$ ，表示这 $N$ 台机器的能量值。

样例输入：

5

1 2 3 2 1

输出描述：

输出只包括一行，即最多可以同时启动的机器数量。

样例输出：

4

Code:

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<cstring>
using namespace std;
#define DEBUGx
typedef long long LL;
const int N=5e3+10;
LL a[N];
LL dp[N][N]={0};
int n;
//定义第 u 台机器左边尝试连接 i 右边尝试连接 j 时，前 u 台机器系统可启动最大机器
数为 dfs(u,i,j)
LL dfs(int u,int i,int j){
    if(u>n){
        return 0;
```

```

    }
    if(dpx[i][j]!=0){
        # ifdef DEBUG
            cout<<"("<<u<<","<<i<<","<<j<<")="<<dpx[i][j]<<endl;
        #endif
        return dpx[i][j];
    }
    //不连接 u
    LL res=dfs(u+1,i,j);
    //连接 u
    if(i==0){
        res=max(res,dfs(u+1,u,j)+1);
    }else if(j==0){
        res=max(res,dfs(u+1,i,u)+1);
    }else if(a[u]+a[i]<2*a[j]){ // 尝试连接 u 如果连接 u 后
        res=max(res,dfs(u+1,j,u)+1);
    }
    #ifdef DEBUG
        cout<<"("<<u<<","<<i<<","<<j<<")="<<dpx[i][j]<<endl;
    #endif
    return dpx[i][j]=res;
}

LL solve(LL a[],int n){
    //定义 dp
    /*
        dp[i][j] 前 j 台机器中末尾两台机器为 i, j 时, 可启动机器的数量的最大值
    */
    LL dp[n+1][n+1];
    memset(dp,0,sizeof(dp));
    //初始化 dp
    /*
        dp[i][i] 前 i 台机器中末尾两台机器为 i 时, 可启动机器的数量就是该机器本身
        dp[1][i] i 大于 1 前 i 台机器中末尾两台机器为 i 和 1 时, 由于在 1 的左侧不存在机器, 故第 1 台和第 i 台都可以启动
        换一种说法 dp[1][i] 表示只启动第一台和第 i 台
    */
    for(int i=1;i<=n;i++){
        dp[i][i]=1;
        if(i>1)
            dp[1][i]=2;
    }
    //定义递推公式
    /*
        [k,i,j]

```

```

    dp[k][i] 前 i 台机器中末尾两台机器为 k, i 可启动机器的数量的最大值
    如果  $2*a[i] > a[j] + a[k]$  说明 i 的加入不会让系统崩溃
    dp[i][j] = max{dp[k][i] + 1,  $2*a[i] > a[j] + a[k]$ ,  $k < i$ }
*/
LL ans = 0;
for(int j = 1; j <= n; j++) {
    for(int i = 1; i < j; i++) {
        LL temp = dp[i][j];
        for(int k = 1; k <= i; k++) {
            if( $2*a[i] > a[j] + a[k]$ ) //判断新增的 j 是否满足条件
                temp = max(temp, dp[k][i] + 1); //开启该机器
        }
        dp[i][j] = temp; //更新前 i 台机器 末尾为 i, j 最多开启机器数之和
        ans = max(ans, dp[i][j]);
    }
}

#ifdef DEBUG
for(int i = 1; i <= n; i++) {
    for(int j = i; j <= n; j++) {
        cout << "dp[" << i << "][" << j << "]: " << dp[i][j] << " ";
    }
    cout << endl;
}
#endif
return ans;
}

int main() {
    //1. 启动一些机器 -> 机器最多可以启动 n 台
    //2. 两台机器之间没有其它启动的机器 这两台机器（尝试）连接
    //3. 若一台机器 a[i] 左右两边都有机器尝试连接 且  $2*a[i] \leq (a[i-1] + a[i+1])$  -> 系统崩溃
    //4. 最多可以启动多少台?
    freopen("data5.in", "r", stdin);
    int t;
    cin >> t;
    while(t--) {
        cin >> n;
        memset(dpx, 0, sizeof(dpx));
        for(int i = 1; i <= n; i++) {
            cin >> a[i];
        }
        cout << dfs(1, 0, 0) << " " << solve(a, n) << endl;
    }
    return 0;
}

```

```
}
```

Input:

7

5

1 2 3 4 5

2

1 2

3

1 2 1

5

1 2 3 2 1

12

343 5435 65 535 1434 656 8768 656 -6546 887 5 12

4

1 2 3 1

5

1 2 2 3 3

Output:

3 3

2 2

3 3

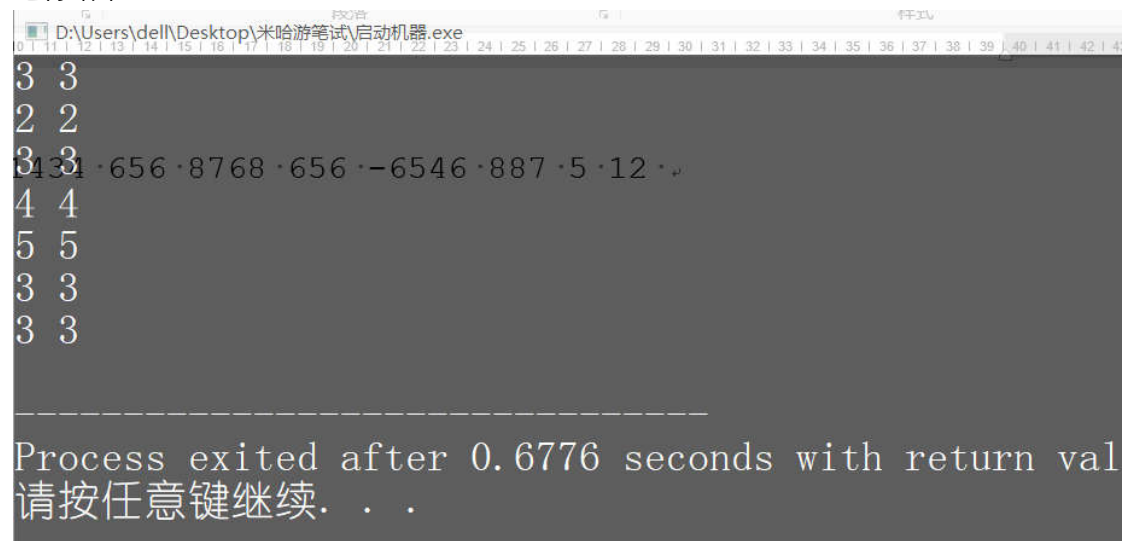
4 4

5 5

3 3

3 3

运行结果:



```
D:\Users\dell\Desktop\米哈游笔试\启动机器.exe
3 3
2 2
3 3
343 5435 65 535 1434 656 8768 656 -6546 887 5 12
4 4
5 5
3 3
3 3
-----
Process exited after 0.6776 seconds with return val
请按任意键继续. . .
```