题目描述

妮妮有个朋n友，第i个朋友的家住在(x_i, y_i)。

妮妮想举行q次聚会，每次聚会都会选择一个聚会中心(X_i, Y_i)，所有距离这个中心不超过Ri的朋友都会来参加妮妮的聚会（此处距离为两点间线段距离）。

妮妮想快速知道每次聚会会有多少个朋友来参加，以便妮妮提前准备聚会物资。

P.S. 可惜妮妮的朋友都很懒，每次聚会的Ri都很小。

输入格式

第一行两个正整数n,q（$1 \leq n, q \leq 5 \times 10^4$），表示妮妮的朋友数量和举行聚会的次数。

接下来n行，每行两个整数x_i, y_i（$-10^9 \leq x_i, y_i \leq 10^9$），表示朋友家的坐标。可能有多个朋友的家坐标相同。

接下来q行，每行三个整数X_i, Y_i, R_i（$-10^9 \leq X_i, Y_i \leq 10^9$，$1 \leq R_i \leq 10$），表示第$i$ 次聚会的聚会中心以及半径。

输出格式

输出q行，第i行一个非负整数，表示第i次聚会的答案。

输入样例

```
5 3
0 0
1 0
1 1
1 2
2 2
1 1 1
3 3 2
3 2 2
```

输出样例

```
3
1
2
```

题解

对于范围在$1 \leq n \leq 5 \times 10^4$的朋友数量来说，遍历每一个朋友的坐标并判断与聚会中心的距离是否$<= R_i$的时间复杂度在O(n)级别，而n本身的值较大。而由于R_i的值较小，R_i取最大值10时也仅有400个点有可能符合距离要求。

所以问题转化为，先将朋友的坐标保存，再根据聚会中心和R_i遍历可能满足距离要求的点，反向搜索是否在朋友的坐标集合里，以及坐标集合中有多少个该点。

运用的思路如下：

1.保存朋友的坐标

使用c++中的map结构保存朋友的坐标；其中，键是朋友的坐标，类型为pair<LL,LL>，值是该坐标的数量，类型为LL；由于map中的键不能重复，题干中又说明可能有多个朋友家的坐标相同，于是遇到相同的坐标时，不向map中插入新的元素，而是累加具有相同键的元素的值的大小。

具体操作为，利用下标获取对应键的值：

```
pair<LL,LL> p1(x,y);
point[p1]++;
```

两种情况下该式均可满足需求。在map中已经存在该键时对它的值进行“+1”操作；在map中没有该键时point[p1]自动向map中插入键为p1的新元素，值会被初始化为0，“+1”操作将值变为1，恰好代表位于该坐标处的有一位朋友。

2.遍历可能位于R_i范围内的点

按照聚会中心的坐标以及R_i的值，利用双层循环遍历可能符合要求的坐标，并查看在map中的数量。

具体操作为，利用下标查看键对应的值：

```
pair<LL,LL> p2(i,j);
cnt+=point[p2];
```

使用下标可以直接获取对应键的值，若是map中没有该键，则插入一个新的元素，此时被初始化的值为0，并不影响cnt的累加。

标程代码

```
#include<iostream>
#include<string>
#include<vector>
#include<queue>
#include<cstring>
#include<iomanip>
#include<algorithm>
#include<cmath>
#include<map>
using namespace std;
typedef long long LL;
LL n,q,cnt=0;
LL x,y,r;
map<pair<LL,LL>,LL> point;
LL GetDistance(LL x1,LL y1) {
    return ((x1-x) * (x1-x) + (y1-y) * (y1-y));
}
int main(){
```

```
    ios::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);
    cin>>n>>q;
    for(LL i=0;i<n;i++){
        cin>>x>>y;
        pair<LL,LL> p1(x,y);
        point[p1]++;
    }

    for(LL i=0;i<q;i++){
        cnt=0;
        cin>>x>>y>>r;
        for(LL i=x-r;i<=x+r;i++){
            for(LL j=y-r;j<=y+r;j++){
                if(GetDistance(i,j)<=r*r){
                    pair<LL,LL> p2(i,j);
                    cnt+=point[p2];
                }
            }
        }
        cout<<cnt<<"\n";
    }
}
```