TopCoder SRM 727 Div2 500-point TwoDiagonals

#题目

Bearland can be represented as a horizontal plane with N distinct points, denoting positions of N cities.The i-th city has coordinates (x[i], y[i]).

Limak is planning to build two infinitely long roads in Bearland.One should go northwest, and the other should go northeast (both these directions are tilted exactly 45 degrees from the vertical direction).It implies that the two roads are perpendicular to each other, and that they have an intersection point (this point can be a city but not necessarily).

Roads increase the trade significantly.A city will be happy if it lies on at least one of the two roads.Help Limak and find the maximum possible number of happy cities.

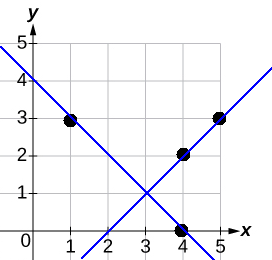
###Example 0

x { 1, 4, 4, 5 }

y { 3, 0, 2, 3 }

Returns 4

There are four cities: (1,3), (4,0), (4,2), (5,3).It's possible to draw two lines in such a way that all cities will be happy:



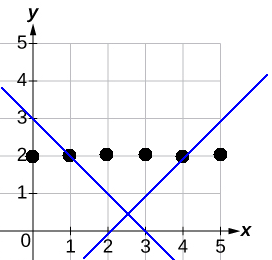
###Example 1

x { 0, 1, 2, 3, 4, 5 }

y { 2, 2, 2, 2, 2, 2 }

Returns 2

The new roads can go through at most two cities in this case.One of many optimal placements is:



#分析

（1）题意

给定坐标中一些点，在坐标系中画两条相互垂直的先，斜率为+1和-1，试问如何画这两条线，使得最终在线上的点的个数最多，求出最大点数。

（2）

k = 1时，y = x + b1 ==> b1 = y -x

k = -1时，y = -x + b2 ==> b2 = y + x

这里b1和b2分别是两条线的截距，分别对应着两条线。

（3）枚举b1和b2这两条线的所有组合，分别求出这条线上共有多少点（需要减去交点），取最大值，即为本题所求。

以Example 0这例



b1 = 4的线上有2个点，b1 = 6的线上有1个点，b1 = 8的线上有1个点

b2 = -4的线上有1个点，b2 = -2的线上有2个点，b2 = 2的线上有1个点

对于b1 = 4和b2 = -4这两条线，共有2 + 1 - 1（两条线都通过）= 2个点

对于b1 = 4和b2 = -2这两条线，共有2 + 2 = 4个点

对于b1 = 4和b2 = 2这两条线，共有2 + 1 - 1（两条线都通过）= 2个点

对于b1 = 6和b2 = -4这两条线，共有1 + 1 = 2个点

对于b1 = 6和b2 = -2这两条线，共有2 + 1 - 1（两条线都通过）= 2个点

对于b1 = 6和b2 = 2这两条线，共有1 + 1 = 2个点

对于b1 = 8和b2 = -4这两条线，共有1 + 1 = 2个点

对于b1 = 8和b2 = -2这两条线，共有2 + 1 - 1（两条线都通过）= 2个点

对于b1 = 8和b2 = 2这两条线，共有1 + 1 = 2个点

所以，最大值取b1 = 4和b2 = -2两条线，共有4个点

#代码

```

#include <iostream>

#include <vector>

#include <map>

#include <set>

using namespace std;

class TwoDiagonals

{

public:

int maxPoints(vector<int> x, vector<int> y)

{

map<int, int> b1; // y = -x + b1 ==> b1 = y + x

map<int, int> b2; // y = x + b2 ==> b2 = y - x

set<pair<int, int> > points;

int maxCities = 0;

int cityCnt;

double crossX;

double crossY;

for(unsigned int i = 0; i < x.size(); i++)

{

b1[y[i] + x[i]]++;

b2[y[i] - x[i]]++;

points.insert(make\_pair(x[i], y[i]));

}

for(map<int, int>::iterator iter1 = b1.begin(); iter1 != b1.end(); iter1++)

{

for(map<int, int>::iterator iter2 = b2.begin(); iter2 != b2.end(); iter2++)

{

cityCnt = (\*iter1).second + (\*iter2).second;

crossX = (double)((\*iter1).first - (\*iter2).first)/2;

crossY = (double)((\*iter1).first + (\*iter2).first)/2;

pair<double, double> crossPoint = make\_pair(crossX, crossY);

if(points.count(crossPoint))

{

cityCnt--;

}

maxCities = max(maxCities, cityCnt);

}

}

return maxCities;

}

};

int main()

{

TwoDiagonals t;

vector<int> x;

x.push\_back(1);

x.push\_back(4);

x.push\_back(4);

x.push\_back(5);

// x.push\_back(0);

// x.push\_back(1);

// x.push\_back(2);

// x.push\_back(3);

// x.push\_back(4);

// x.push\_back(5);

vector<int> y;

y.push\_back(3);

y.push\_back(0);

y.push\_back(2);

y.push\_back(3);

// y.push\_back(2);

// y.push\_back(2);

// y.push\_back(2);

// y.push\_back(2);

// y.push\_back(2);

// y.push\_back(2);

cout << t.maxPoints(x, y) << endl;

return 0;

}

```

#参考

<http://blog.csdn.net/zarlove/article/details/79038268>