**两条直线**

问题描述

给定平面上n个点。

求两条直线，这两条直线互相垂直，而且它们与x轴的夹角为45度，并且n个点中离这两条直线的曼哈顿距离的最大值最小。

两点之间的曼哈顿距离定义为横坐标的差的绝对值与纵坐标的差的绝对值之和，一个点到两条直线的曼哈顿距离是指该点到两条直线上的所有点的曼哈顿距离中的最小值。

输入格式

第一行包含一个数n。

接下来n行，每行包含两个整数，表示n个点的坐标（横纵坐标的绝对值小于109）。

输出格式

输出一个值，表示最小的最大曼哈顿距离的值，保留一位小数。

样例输入

4  
1 0  
0 1  
2 1  
1 2

样例输出

1.0

数据规模与约定

对于30%的数据，n<=100。

对于另外30%的数据，坐标范的绝对值小于100。

对于100%的数据，n<=105。

锦囊1

使用排序算法。

锦囊2

对所有的点按(x+y, x-y)的顺序排序，使用二分答案的方法二分答案的最小值，利用x+y的单调性来快速判断所二分的答案的可行性。

C++代码：

#include<iostream>

#include<algorithm>

#include<cstdio>

#include<cmath>

using namespace std;

const int N=100000;

struct P{int x,y;};

bool cmp(P a,P b){

if(a.x==b.x)return a.y<b.y;

return a.x<b.x;

}

P d[N+5];

struct F{int max,min;};

F fl[N+5],fr[N+5];

inline double Max(double a,double b){return a>b?a:b;}

inline double Min(double a,double b){return a>b?b:a;}

bool check(double m,int n){

m\*=2;

int i,j=0;

for(i=0;i<n;i++){

while(j<n&&d[j].x-d[i].x<=m)j++;

double MAX=-1e10;

double MIN=1e10;

if(j!=n){

MAX=Max(MAX,fr[j].max);

MIN=Min(MIN,fr[j].min);

}

if(i-1>=0){

MAX=Max(MAX,fl[i-1].max);

MIN=Min(MIN,fl[i-1].min);

}

// cout<<i<<" "<<j<<" "<<MAX<<" "<<MIN<<endl;

if(MAX-MIN<=m)return true;

}

return false;

}

void init(int n){

int i;

fl[0].min=fl[0].max=d[0].y;

for(i=1;i<n;i++){

fl[i].max=Max(fl[i-1].max,d[i].y);

fl[i].min=Min(fl[i-1].min,d[i].y);

}

fr[n-1].min=fr[n-1].max=d[n-1].y;

for(i=n-2;i>=0;i--){

fr[i].max=Max(fr[i+1].max,d[i].y);

fr[i].min=Min(fr[i+1].min,d[i].y);

}

}

int main(){

int i,n;

cin>>n;

for(i=0;i<n;i++){

int x,y;

scanf("%d%d",&x,&y);

d[i].x=x+y;

d[i].y=x-y;

}

sort(d,d+n,cmp);

init(n);

double l=0.0;

double r=1000000000;

while(r-l>=0.01){

double m=(l+r)/2;

// cout<<m<<endl;

if(check(m,n))r=m;

else l=m;

}

printf("%.1f\n",r);

return 0;

}

C代码：

#include<stdio.h>

#include<stdlib.h>

struct point

{

int s, d;

} p[100005];

int bl[100005], sl[100005], br[100005], sr[100005];

int cmp(const void \*a, const void \*b)

{

struct point \*pa = (struct point\*)a;

struct point \*pb = (struct point\*)b;

return pa->s - pb->s;

}

int main(void)

{

int n, x, y;

int min, max;

double mind, maxd, d;

int i, j;

scanf("%d", &n);

for (i=0; i<n; ++i)

{

scanf("%d%d", &x, &y);

p[i].s = x+y;

p[i].d = x-y;

}

qsort(p, n, sizeof(struct point), cmp);

bl[0] = sl[0] = p[0].d;

for (i=1; i<n; ++i)

{

bl[i] = bl[i-1]>p[i].d ? bl[i-1] : p[i].d;

sl[i] = sl[i-1]<p[i].d ? sl[i-1] : p[i].d;

}

br[n-1] = sr[n-1] = p[n-1].d;

for (i=n-2; i>=0; --i)

{

br[i] = br[i+1]>p[i].d ? br[i+1] : p[i].d;

sr[i] = sr[i+1]<p[i].d ? sr[i+1] : p[i].d;

}

mind = 0;

maxd = br[0]-sr[0] > p[n-1].s-p[0].s ? br[0]-sr[0] : p[n-1].s-p[0].s;

while (maxd-mind > 0.1)

{// printf("%f %f\n", mind, maxd);

d = (maxd+mind)/2;

j = 0;

for (i=0; i<n; ++i)

{

while (j<n && p[j].s-p[i].s<=d)

{

++j;

}

if (i==0)

{

max = br[j];

min = sr[j];

}

else if (j==n)

{

max = bl[i-1];

min = sl[i-1];

}

else

{

max = bl[i-1]>br[j] ? bl[i-1] : br[j];

min = sl[i-1]<sr[j] ? sl[i-1] : sr[j];

}

if (max-min<=d)

{

break;

}

}

if (max-min<=d)

{

maxd = d;

}

else

{

mind = d;

}

}

printf("%.1f\n", maxd/2);

return 0;

}