After successive failures in the battles against the Union, the Empire retreated to its last stronghold. Depending on its powerful defense system, the Empire repelled the six waves of Union's attack. After several sleepless nights of thinking, Arthur, General of the Union, noticed that the only weakness of the defense system was its energy supply. The system was charged by *N* nuclear power stations and breaking down any of them would disable the system.

The general soon started a raid to the stations by *N* special agents who were paradroped into the stronghold. Unfortunately they failed to land at the expected positions due to the attack by the Empire Air Force. As an experienced general, Arthur soon realized that he needed to rearrange the plan. The first thing he wants to know now is that which agent is the nearest to any power station. Could you, the chief officer, help the general to calculate the minimum distance between an agent and a station?

Input

The first line is a integer *T* representing the number of test cases.  
Each test case begins with an integer *N* (1 ≤ *N* ≤ 100000).  
The next *N* lines describe the positions of the stations. Each line consists of two integers *X* (0 ≤ *X* ≤ 1000000000) and *Y* (0 ≤ *Y* ≤ 1000000000) indicating the positions of the station.  
The next following *N* lines describe the positions of the agents. Each line consists of two integers *X* (0 ≤ *X* ≤ 1000000000) and *Y* (0 ≤ *Y* ≤ 1000000000) indicating the positions of the agent.

Output

For each test case output the minimum distance with precision of three decimal placed in a separate line.

Sample Input

2

4

0 0

0 1

1 0

1 1

2 2

2 3

3 2

3 3

4

0 0

0 0

0 0

0 0

0 0

0 0

0 0

0 0

Sample Output

1.414

0.000

计算几何求最近点对的变种。

如何求最近点对，在 <http://blog.csdn.net/lttree/article/details/25156173> 中已经介绍过。

这个题属于变种吧，要求两个点分别属于不同集合。

其实也很简单，按照上面的代码，稍微改点地方就OK了。

就是在返回长度的时候判断是否属于同一集合，若属于同一个集合，就返回一个很大的数，我用的999999。

#include<iostream>

#include <stdio.h>

#include <math.h>

#include <algorithm>

using namespace std;

struct node

{

double x,y;

int flag;

}p[1000001];

int arr[1000001];

double dis(node a,node b) // 求两点之间的距离

{

return sqrt((a.x-b.x)\*(a.x-b.x)+(a.y-b.y)\*(a.y-b.y));

}

// 根据点横坐标or纵坐标排序

bool cmp\_y( int a,int b)

{

return p[a].y<p[b].y;

}

bool cmp\_x(node a,node b)

{

return a.x<b.x;

}

// 求最近点对

double close\_pair( int l,int r )

{

// 如果只剩下两个点，判断是否属于同一集合

if( r==l+1 )

if( p[l].flag!=p[r].flag )

return dis(p[l],p[r]);

else

return 999999;

else

if( r==l+2 ) // 剩下三个点，判断集合类别

{

if( p[l].flag==p[l+1].flag )

{

if( p[l].flag==p[l+2].flag )

return 999999;

else

return min( dis(p[l],p[l+2]),dis(p[l+1],p[l+2]) );

}

else

{

if(p[l].flag==p[r].flag)

return min( dis(p[l],p[l+1]),dis(p[l+2],p[l+1]) );

else

return min( dis(p[l],p[l+1]),dis(p[l],p[l+2]) );

}

return min( dis(p[l],p[r]),min( dis(p[l],p[l+1]),dis(p[l+1],p[r]) ) );

}

int mid=(l+r)>>1;

double ans=min(close\_pair(l,mid),close\_pair(mid+1,r));

int i,j,cnt=0;

for(i=l; i<=r; ++i)

if( p[i].x>=p[mid].x-ans && p[i].x<=p[mid].x+ans )

arr[cnt++]=i;

sort(arr,arr+cnt,cmp\_y);

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

for(j=i+1; j<cnt; j++)

{

if( p[arr[j]].flag != p[arr[i]].flag )

{

if(p[arr[j]].y-p[arr[i]].y>=ans)

break;

ans=min(ans,dis(p[arr[i]],p[arr[j]]));

}

}

return ans;

}

int main()

{

int i,n,t;

cin>>t;

while(t--)

{

cin>>n;

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

{

cin>>p[i].x>>p[i].y;

p[i].flag=1;

}

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

{

cin>>p[i].x>>p[i].y;

p[i].flag=2;

}

sort(p,p+n+n,cmp\_x);

printf("%.3lf\n",close\_pair(0,n+n-1));

}

return 0;

}