POJ3714：

算法分析：

类似于最近点对，只不过要求两个点要在不同的集合里面。

采用分治法，分为两个集合，求两个集合的最近点对，以及求两个点分别在两个集合的点对。所求的点的位置，一定在于mid-d,mid+d 之间。然后，就在这个区间开始找点，并不断更新d值，最后就可以得到d了。根据题意，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

#include <stdio.h>

#include <math.h>

#include <algorithm>

using namespace std;

// 不光有点，还要有集合类别标志

struct Point

{

double x,y;

int flag;

}p[1000001];

int arr[1000001];

double Min(double a,double b)

{

return a<b?a:b;

}

// 求两点之间的距离

double dis(Point a,Point b)

{

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( Point a,Point 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;

}

// 剩下三个点，判断集合类别

else if( r==l+2 )

{

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[l+2].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;

scanf("%d",&t);

while( t-- )

{

scanf("%d",&n);

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

{

scanf("%lf%lf",&p[i].x,&p[i].y);

p[i].flag=1;

}

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

{

scanf("%lf%lf",&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;

}