

# 平面最近点对

## Nearest Pair of Points

**Idea:** 分治。选取一条垂直线，分左右递归求解。合并步骤中，只需考虑宽度为  $2d$  的长条中的点，而对于每一个长条中的点，又只需考虑在其上垂直距离不超过  $d$  的点，可以证明，这样考虑的点不超过 7 个。若事先存储一个按  $y$  值排好序的数组，则理论上时间复杂度为  $T(n) = 2T(n/2) + O(n) = O(n \lg n)$ ，但代码似乎不太好写。退而求其次，可以在合并步骤中排序，则时间复杂度为  $T(n) = 2T(n/2) + O(n \lg n) = O(n \lg^2 n)$ 。

**Complexity:**  $O(n \lg n)$  (理论上可以达到) ;  $O(n \lg^2 n)$  (更好写的代码)

**Code:**

```
1  struct Point{
2      double x, y;
3  }p[N], t[N];
4  bool cmpx(Point A, Point B){
5      if(A.x == B.x) return A.y < B.y;
6      return A.x < B.x;
7  }
8  bool cmpy(Point A, Point B){
9      if(A.y == B.y) return A.x < B.x;
10     return A.y < B.y;
11 }
12
13 inline double dis(Point A, Point B){ return sqrt((A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y)); }
14
15 double solve(int l, int r){
16     if(l == r) return INF;
17     if(r - l == 1) return dis(p[l], p[r]);
18     int mid = (l + r) >> 1;
19     double d = min(solve(l, mid), solve(mid+1, r));
20     pt = 0;
21     for(int i = l; i <= r; i++)
22         if(fabs(p[i].x - p[mid].x) <= d)
23             t[++pt] = p[i];
24     sort(t+1, t+pt+1, cmpy);
25     for(int i = 1; i <= pt; i++)
26         for(int j = i+1; j <= pt && t[j].y - t[i].y <= d; j++)
27             d = min(d, dis(t[i], t[j]));
28     return d;
29 }
30
31 int main(){
32     scanf("%d", &n);
33     for(int i = 1; i <= n; i++)
34         scanf("%lf%lf", &p[i].x, &p[i].y);
35     sort(p+1, p+n+1, cmpx);
36     printf("%.4f\n", solve(1, n));
37     return 0;
38 }
```

