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：

```
struct Point{
    double x, y;
}p[N], t[N];
bool cmpx(Point A, Point B){
    if(A.x == B.x)  return A.y < B.y;
    return A.x < B.x;
}
bool cmpy(Point A, Point B){
    if(A.y == B.y)  return A.x < B.x;
    return A.y < B.y;
}

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)); }

double solve(int l, int r){
    if(l == r)  return INF;
    if(r - l == 1)  return dis(p[l], p[r]);
    int mid = (l + r) >> 1;
    double d = min(solve(l, mid), solve(mid+1, r));
    pt = 0;
    for(int i = l; i <= r; i++)
        if(fabs(p[i].x - p[mid].x) <= d)
            t[++pt] = p[i];
    sort(t+1, t+pt+1, cmpy);
    for(int i = 1; i <= pt; i++)
        for(int j = i+1; j <= pt && t[j].y - t[i].y <= d; j++)
            d = min(d, dis(t[i], t[j]));
    return d;
}

int main(){
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
        scanf("%lf%lf", &p[i].x, &p[i].y);
    sort(p+1, p+n+1, cmpx);
    printf("%.4f\n", solve(1, n));
    return 0;
}
```