1. 通过深度优先搜索得到每个节点的度数，之后再遍历每一条边选择度数较多的顶点即可

Algorithm Cover (T)

Input: Tree T = (V, E)

Output: the cover

begin

Use DFS to count v.component for all vertex in V; for all vertex v.cover=false

Use DFS with the following postWork

if w was unmarked and v.cover=false and w.cover=false then

v.component > w.component ? v.cover=true : w.cover=true.

end

2. 建立坐标系简单判断即可

Algorithm if line (V, n)

Input V, n

Output if line

begin

$$\text{If: } \frac{y - V[1].y_0}{y - V[2].y_0} = \frac{x - V[1].x_0}{x - V[2].x_0}$$

verify if all vertex in line; return;

end

1. 利用凸包的思想去求解这个问题，如果点在一个凸包内，那么相交边最少为1，如果在多个凸包内则先删除最外的凸包再求解

Algorithm least edge $(p_1, p_2, \dots p_n, p)$

Input: $p_1, p_2, \dots p_n$ (a set of points in the plane),
p (that point)

Output: num, point.

begin.

$S = (p_1, p_2, \dots p_n)$.

P = Graham's Scan. $(p_1, p_2, \dots p_n, p)$.

for v in P:
for w link with v
if w both in P $S = S - v$
else add link $w - v$
if p in P
find vertex v out of P and $v-p$
only cross one point with S
return $(0, v)$

for every new polygon P_i and new added edge e_i

if least edge(P_i, p).point − p cross e_i

if min > least edge(P_i, p).num − 1

min = least edge(P_i, p).num − 1

point = least edge(P_i, p).point

else if min > least edge(P_i, p).num + 1

min = least edge(P_i, p).num + 1

point = a point in p − least edge(P_i, p).point − and out of P

return (min, point)