二分与贪心算法

郭炜 刘家瑛

POJ 1328
雷达安装问题

北京大学

问题描述

- 一共有n个小岛位于x轴之上，x轴为海岸，x轴上方为海洋
- 现需要在海岸上建立雷达，每个雷达的覆盖半径为d
- 也就是说每个雷达能覆盖距离它不超过d的所有点
- 要求求出在海岸建立最少的雷达的数目，使得可以覆盖所有的小岛. 可以认为每个小岛都是一个点

程序输入

- 输入包含若干组数据
- 每组数据的第一行是两个整数n和d (1 ≤ n ≤ 1000),

  分别表示小岛的数量和雷达的半径
- 接下来有n行，每行的两个整数，分别表示各个小岛的坐标
- 输入以0 0结束

程序输出

- 对于每组数据，输出一行
  - 最小的雷达数，如果无解则输出-1

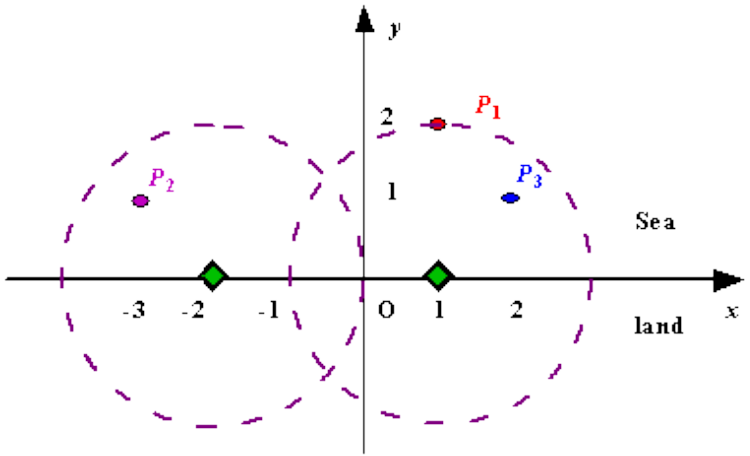
样例输入

```
3 2
1 2
-3 1
2 1

1 2
0 2

0 0
```

样例输出

Case 1: 2

Case 2: 1



解题思路

- 找最少的雷达覆盖所有的小岛

直接求解并不容易

- 反过来, 从小岛入手,

  考虑每个小岛被覆盖所需雷达的位置!

- 根据每个小岛的位置

→ 求出这个 小岛i 对应覆盖它的雷达的位置

→ 必须在x轴的 [s[i].left, s[i].right] 之间

解题思路

- 问题转化为:
  - 每个小岛对应x轴上的一段线段[s[i].left, s[i].right]
  - 现在要求出最少的点, 使得所有的线段都被覆盖到
  - 贪心解决!

解题思路

- 由小到大将线段按左端点排序
  - 贪心: 找到一个点，能覆盖尽可能多的线段
- 因此, 依次加入一根 线段 i
  - 如果这个线段与当前的所有线段 (j，j+1，… i-1) 没有公共交点，
  - 则说明需新建雷达

    S[i].left > min(s[j].right，s[j+1].right，… ，s[i-1].right)
  - 否则加入当前的线段集合

解题思路

- 实现的时候, 只用维护 当前集合右端点的最小值
- 即判断新加入的线段的左端点是否大于当前集合右端点最小值
  - 若小于等于, 说明有公共交点
    - 加入当前集合, 更新右端点的最小值
  - 若大于, 说明无公共交点
    - 建立新雷达, 建立新集合

重点函数分析

- 求解每个case

```
int Solve() {
    int ans;  double  now;
    sort(s, s+n); //先按左端点排序
    ans = 1; now = s[0].right; //当前雷达数为1,
                               //当前集合右端点最小值为第一个点的右端点
    for (int i = 1;i < n;++i)
        if (s[i].left <= now) now = min(now, s[i].right);
        //若当前线段与目前集合中的线段有公共交点, 则更新右端点最小值
        else { //否则加入新雷达
            ++ans;
            now = s[i].right;
        }
    return ans;
}
```

参考程序

```
#include <stdio.h>
#include <iostream>
#include <algorithm>
#include <math.h>
using namespace std;

#define MAXN (1000+10)

//线段两端
struct Node {
    double left,  right;
};
```

参考程序

```
int T, n, d;
Node s[MAXN];

//Node按左端点大小排序
bool operator < (const Node &a,  const Node &b) {
    return a.left < b.left;
}

//求解每个case
int Solve() {
    int ans;
    double now;
```

参考程序

```
    //按左端点排序
    sort(s, s+n);
    //若当前线段与目前集合中的线段没有公共交点，则新加入一个雷达
    ans = 1; now = s[0].right;
    for (int i = 1;i < n;++i)
        if (s[i].left <= now) now = min(now,  s[i].right);
        else {
            ++ans;
            now = s[i].right;
        }

    return ans;
} //end Solve()
```

参考程序

```
int main() {
    int x,  y;
    bool flag;

    T = 0;
    while (1) {
        ++T;
        scanf("%d%d",  &n,  &d);
        if (n == 0 && d == 0) break;

        //读入
        flag = true;
```

参考程序

```
        for (int i = 0;i < n;++i){
            scanf("%d%d", &x, &y);
            if (y > d) flag = false;
            else {
                s[i].left  = x - sqrt( d*d - y*y );
                s[i].right = x + sqrt( d*d - y*y );
            }
        }
        //若有小岛距离x轴大于d,则输出-1
        if (flag) printf("Case %d:  %d\n", T, Solve());
        else printf("Case %d:  -1\n",  T);
    }
    return 0;
}//end main
```