**华南农业大学**

**《算法分析与设计》课程实验**

**实验6：通讯覆盖**

年级专业： 21信息与计算科学

学生姓名： 吴嘉豪

学生学号： 202125810321

指导老师： 赵峰

实验时间： 2024 年05 月 14日

一、实验内容

假设海岸线是一条无限延伸的直线，以海岸线为轴建立坐标系，海洋侧为轴上方区域。

海洋里有个小岛，每个小岛相当于轴上方的一个点。

为实现小岛与陆地的通讯，需要在海岸线（轴）上设置若干个基站，各基站的覆盖半径均为。

要求设计贪心算法，根据小岛的坐标，给出一组基站的坐标，要求每个小岛都要被其中一个基站覆盖（离该基站的距离小于等于），并且所用基站数目最少。

假设小岛离海岸线的距离都不超过。

二、实验步骤

1、选择合适的贪心策略，编写贪心算法程序；

2、证明所选贪心策略的贪心选择性质（即证明由该策略得到的贪心解就是最优解）；

3、运行测试数据，检查贪心算法的正确性；分析贪心算法的时间复杂度（要有分析过程）；

4、随机产生小岛坐标，运行算法求解，并检验结果的合理性：计算小岛与覆盖该小岛的基站的距离，检查该距离是否不超过；或者以基站为圆心，为半径画圆，检查是否所有小岛都在这些圆内；

5、填写实验报告。

三、实验过程与结果

2. 我是先将海岛映射到x轴，

图示

描述已自动生成

转换为区间问题。对于这个区间问题， 我们可以根据尾区间进行排序，然后记录当前基站的最远的位置为cursor，

那么如果下一个元素的首区间 小于等于cursor， 那么 就不需要额外建立基站， 不然就需要给这个岛屿建立一个基站， 如果不建立， 那么这个岛屿 就不能被覆盖到。 这是必要的。

那么这个岛屿的基站建立再哪里会更好呢？ 我们知道前面的岛屿肯定都被基站覆盖。（如果没有就给他加上基站）那么这个基站一定是放在尾区间是最优的，因为这有益于后续的岛屿。 这个证明了它是最优的

3.

*for*(int i = 0; i < n; ++i){

*if*((x*[*i*]*.first - cursor) <= inf) *continue*;

        cursor = x*[*i*]*.second;

// *建立一个基站*

        cnt++;

    }

再处理建立基站的过程中是一个O(n)， 算法的瓶颈在于对岛屿再x轴的映射的排序过程，这是O(nlogn)的。

四、源程序代码

#*include*<iostream>

#*include*<vector>

#*include*<string>

#*include*<algorithm>

#*include*<ctime>

#*include* <iomanip>

#*include*<cmath>

using namespace std;

*const* float inf = 1e-6;

void *solve*() {

// *岛屿的数量*

    int n;

    n = *rand*() % 10 + 5;

    int d;

    d = *rand*() % 30 + 10;

// *第一个int 代表 横坐标， 第二个int 代表 纵坐标方向*

    vector<pair<int,int>> *site*(n);

*for*(int i = 0; i < n; ++i){

        site*[*i*]*.first = *rand*() %  100;

        site*[*i*]*.second = *rand*() % d;

    }

    cout*<<*"现在有"*<<*n*<<*"个岛屿"*<<endl*;

    cout*<<*"分别在"*<<endl*;

*for*(int i = 0; i < n; ++i){

        cout*<<*"横坐标为："*<<*site*[*i*]*.first*<<*" 纵坐标为："*<<*site*[*i*]*.second*<<endl*;

    }

    cout*<<*"-----------------------"*<<endl*;

    cout*<<*"d的大小为: "*<<*d*<<endl*;

// *排序岛屿*

// *首先根据 岛屿 画一个圆 然后会和 x轴产生 两个交点（当然一个交点可以理解为两个交点结合)*

// *这样就会在x轴构成多个区间 ， 就可以转换为 区间问题 来进行 求解*

// *在x轴的映射*

    vector<pair<double,double>> *x*(n);

    cout*<<*"-----------------------"*<<endl*;

    cout*<<*"在x轴的映射为"*<<endl*;

*for*(int i = 0; i < n; ++i){

// *dx =  sqrt(d^2 - site[i].second)*

        double dx = *sqrt*(d \* d - site*[*i*]*.second \* site*[*i*]*.second);

        x*[*i*]*.first = site*[*i*]*.first - dx;

        x*[*i*]*.second = site*[*i*]*.first + dx;

        cout*<<*"开始位置："*<<*x*[*i*]*.first*<<*"结束位置："*<<*x*[*i*]*.second*<<endl*;

    }

// *根据岛屿在x轴的后边的那一段进行映射*

*sort*(x.*begin*(), x.*end*(), [&](auto p1, auto p2){

*return* p1.second < p2.second;

    });

    double cursor = -1.0;

    int cnt = 0;

    vector<double>pos;

*for*(int i = 0; i < n; ++i){

*if*((x*[*i*]*.first - cursor) <= inf) *continue*;

        cursor = x*[*i*]*.second;

// *建立一个基站*

        pos.*push\_back*(x*[*i*]*.second);

        cnt++;

    }

    cout*<<*"需要建立基站"*<<*cnt*<<*" 个"*<<endl*;

    cout*<<*"基站的x轴位置(横坐标)分别为"*<<endl*;

*for*(int i = 0; i < cnt; ++i){

        cout*<<*pos*[*i*]<<*" ";

    }

}

int *main*() {

    int t;

    t = 1;

*srand*(*time*(nullptr));

*while*(t--){

*solve*();

    }

*return* 0;

}

五、评语及评分

|  |
| --- |
| 评语及评分  评阅人签名： |