软件工程第一次上机作业

姓名： 黄健民

班级： 软工1506班

学号： U201517147

问题描述：

In a box bounded by [-1, 1], given m balloons(they cannot overlap) with variable radio r and position mu, find the optimal value of r and mu which maximize

sum r^2

算法思想：

主要思想是计算所有情况寻找最优解.

圆位置有限是指其圆心x、y坐标的精度只能取到小数点后两位，

所以区域内圆心的位置有10000个，通过计算每个位

置满足条件的最大半径r，从而找出这些点的最大半径；

在寻找最大半径时，每个位置的最大半径的精度只能取到小数点后五位；

满足条件的最大圆将放入一个链表中，

这样一来之后的圆在进行位置判断时可以与已放入的圆进行比较，

从而判断其是否满足条件；

代码：

#include <stdlib.h>

#include <stdio.h>

#include <math.h>

//气球结构体

typedef struct balloon{

double x; //圆心x坐标

double y; //圆心y坐标

double r; //圆半径

}Balloon;

//用链表来储存已经放置的气球

typedef struct balloonList{

struct balloonList \* next;

Balloon balloon;

}BalloonList;

void insert(Balloon balloon);

double distance(Balloon b1, Balloon b2);

int judge(Balloon b);

void putBalloon();

BalloonList \*head = NULL;

double step = 0.01; //改变气球位置的最小单位

int num = 0; //放置气球的个数

double sumr = 0; //用来记录r^2之和

void main(void){

int n=0;

printf("请输入气球总数量: ");

scanf\_s("%d", &n);

printf("\n球编号\t x坐标\t y坐标\t 半径\t r^2之和\n");

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

putBalloon();

}

printf("\nr^2之和最大为:\t %lf\n", sumr);

system("pause");

}

//将气球插入气球队列

void insert(Balloon balloon){

BalloonList \* newBalloon = (BalloonList \*)malloc(sizeof(BalloonList));

newBalloon->balloon = balloon;

newBalloon->next = head;

head = newBalloon;

}

//改变气球的初始位置，求的满足条件的气球

void putBalloon(){

Balloon balloon = { -1 + step, -1 + step, 0 };

Balloon maxBalloon = balloon;

int i, j;

for (i = 0; balloon.x < 1; ++i){

balloon.x += step;

balloon.y = -1 + step;

for (j = 0; balloon.y < 1; ++j){

balloon.y += step;

balloon.r = 0;

double rstep = 0.1;

while (rstep > 0.00001){

if (balloon.r > maxBalloon.r){

maxBalloon = balloon;

}

balloon.r += rstep;

if (!judge(balloon)){//此气球不合适，减小半径与步长重新尝试

balloon.r -= rstep;

rstep /= 10;

}

}

}

}

if (judge(maxBalloon)){

insert(maxBalloon);

num++;

sumr += maxBalloon.r \* maxBalloon.r;

printf("%d\t %.3lf\t %.3lf\t %.3lf\t %lf \n", num, maxBalloon.x, maxBalloon.y, maxBalloon.r, sumr);

}

}

//判断新加入的气球是否符合规则

int judge(Balloon b){

//将气球限制在[-1，1]内

if ((fabs(b.x) + b.r) > 1 || (fabs(b.y) + b.r) > 1){

return 0;

}

//依次比较气球b与已有气球是否相交

BalloonList \*tmp = head;

while (tmp){

Balloon balloon = tmp->balloon;

//两个气球相交判断

if (distance(b, balloon) < b.r + balloon.r){

return 0;

}

tmp = tmp->next;

}

return 1;

}

//判断气球球心之间的距离

double distance(Balloon b1, Balloon b2){

double x1 = b1.x;

double y1 = b1.y; double x2 = b2.x;

double y2 = b2.y;

return pow((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2), 0.5);

}

运行结果：

