**算法设计与分析实验报告**

**指导老师: 曲衍鹏**

**姓 名: 汪广鑫**

**学 号:1120200298**

**实验日期:2020.12.21**

**正方形切割问题**

**一、实验目的**

在正方形内，随即分布不在同一行不在同一列的数个点（必包含原点），然后对每个点向右上方进行切割，要求每次切割出来的矩形面积尽量大，并且切割出来的矩形内部不能包含其他点。

**二、实验内容**

1、编程实现求解正方形切割问题的算法，并输出切割部分的面积。要求切割时，采用贪婪算法的思想，每次选择面积尽可能大的长方形进行切割。

2、问题描述：给定一个100\*100的正方形A，假设将A的左上角顶点视为原点，并定义其坐标为(0, 0)。在A中自动生成100个互不相同的点(为简单起见，可假设每个点的横坐标和纵坐标各不相同，这样可保证每条横线或竖线上至多只有一个点，虽然这个约定不是必须的)，且必有一个点恰好在原点上，要求依据这些给定的点切割正方形，切割方向只能向下和向右，每次都寻找最大的长方形进行切割，但所切割的长方形内部不能含有任何给定点(给定点可在切割线上)，已经切割的部分不可重复切割，且切割出的部分必须是长方形。对每个给定点都必须切割，并累计切割出的面积，使切割出的总面积尽可能大。

3、将给定的原点作为一个点，随机生成其余99个点，并使得这些点的横坐标和纵坐标各不相同。

4、要求至少随机构造10组数据(点)，输出每组数据的运行结果。若可能(选择做)，将切割过程可视化。

5、求每组数据的切割总面积与正方形面积的比，给出相应的结论。

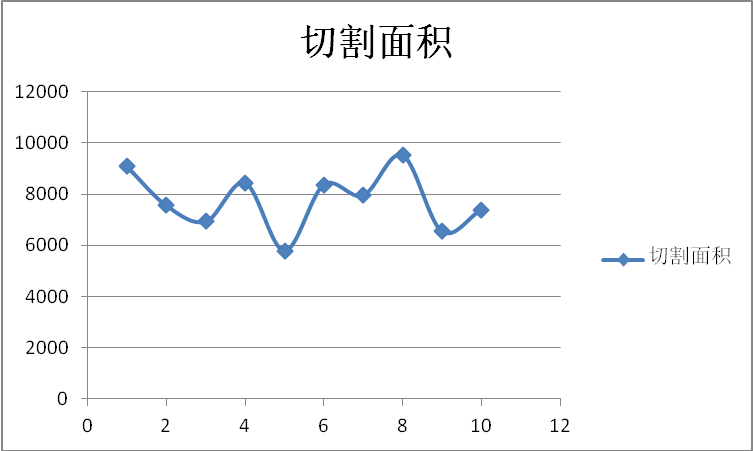
**三、实验数据**

|  |  |  |
| --- | --- | --- |
| 试验次数 | 切割面积 | 所占比例 |
| 1 | 9083 | 0.9083 |
| 2 | 7554 | 0.7554 |
| 3 | 6927 | 0.6927 |
| 4 | 8450 | 0.8450 |
| 5 | 5780 | 0.5780 |
| 6 | 8371 | 0.8371 |
| 7 | 7953 | 0.7953 |
| 8 | 9540 | 0.9540 |
| 9 | 6548 | 0.6548 |
| 10 | 7386 | 0.7386 |

**四、实验代码**

附后。

**五、实验结果**



**六、实验总结**

通过本次实验，明白了切割矩阵的求法，更加深入地理解了贪婪算法的思想。

**主要代码**：

#include "stdio.h"

#include<windows.h>

#include <time.h>

#include <stdlib.h>

#define MAX\_VERTEX\_NUM 20 /\*图的最大顶点数\*/

#define true 1

#define false 0

#define BILI 100

#define LAST 0

#define NOW 1

typedef struct

{

int point[MAX\_VERTEX\_NUM][2];

int nMaxNum;

//int flagVisited[MAX\_VERTEX\_NUM],boundRecord[MAX\_VERTEX\_NUM];

//int nRowCount;

}QieGePoint;

typedef struct

{

int flagVisited[MAX\_VERTEX\_NUM],boundRecord[MAX\_VERTEX\_NUM],arrayBound[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];

int boundArrayTemp[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];

int boudTempRecord[MAX\_VERTEX\_NUM];

int boudTempRecordLast;

int boudTempRecordNow;

int nRowCount;

}QieGePointTemp;

void createQieGePoint(QieGePoint \*QG)

{

int i=0,j=0;

int xTemp=0,yTemp=0;

int flagx=1,flagy=1;

int x[MAX\_VERTEX\_NUM]={0},y[MAX\_VERTEX\_NUM]={0};

srand((unsigned int)time(NULL));

QG->nMaxNum=rand()%(MAX\_VERTEX\_NUM-1)+1;

//printf("There are %d numbers",QG->nMaxNum);

// QG->point[0][0]=0;

// QG->point[0][1]=0;

// QG->point[QG->nMaxNum][0]=100;

// QG->point[QG->nMaxNum][1]=100;

//QG->nMaxNum+=2;

int n=QG->nMaxNum;

for(i=0;i<QG->nMaxNum;i++)

{

while((flagx))

{

xTemp=rand()%99+1;

flagx=0;

for(j=0;j<=i;j++)

{

if(xTemp==x[j])

{

flagx=1;

}

}

}

while(flagy)

{

yTemp=rand()%99+1;

flagy=0;

for(j=0;j<=i;j++)

{

if(yTemp==y[j])

{

flagy=1;

}

}

}

x[i]=xTemp;

y[i]=yTemp;

QG->point[i][0]=xTemp;

QG->point[i][1]=yTemp;

flagx=1;

flagy=1;

}

/////////////////////QG->nMaxNum+=1;

//printf("There are %d numbers",QG->nMaxNum);

}

//void printQieGePoint(QieGePoint \*QG)

//{

// int i;

// FILE \*f;

// f=fopen("b1.xls","w+");

// for(i=0;i<QG->nMaxNum;i++)

// {

// printf("point[%d]:(%d,%d)\n",i,QG->point[i][0],QG->point[i][1]);

// fprintf(f,"%d\t%d\t%d\t\n",i,QG->point[i][0],QG->point[i][1]);

// }

// fclose(f);

//}

void getOrderedPoint(QieGePoint \*QG,QieGePointTemp \*qgt)

{

int n=QG->nMaxNum,i=0,j=0,k=-1,nBound=0;

int nFoundTheNextElem=0;//control the round times, to find the line elements in the while(flagFound)

int abi=0,aby=0,nBoundCount=0,arrayRecord[MAX\_VERTEX\_NUM]={-1};//save the final bound of the elements

int flagVisited[MAX\_VERTEX\_NUM]={false},endF=0,findF=0,bound[MAX\_VERTEX\_NUM]={-1};

int nMaxY=0,nMaxX;//x,y临时值

int s[MAX\_VERTEX\_NUM]={0};

int flagFound=false;

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

{

flagVisited[i]=false;//initial the visited flag

bound[i]=-1;

qgt->flagVisited[i]=false;

}//for i initial flag\*/

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

{

for(j=0;j<MAX\_VERTEX\_NUM;j++)

{

qgt->arrayBound[i][j]=-1;

}

}

int l=n+1;

while((l)--){

k=-1;

nMaxY=0;

nBound=0;

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

{

if(flagVisited[i]==false)

{

if(QG->point[i][1]>nMaxY)

{

nMaxY=QG->point[i][1];

k=i;

}//if

}//if

bound[i]=-1;//initial the boundary

}//for i 每次得到一个划分的头元素标号k

if(k==-1)

{

continue;

}

else

{

//printf("\nThis time the big element's number is %d\n",k);

flagVisited[k]=true;//标记为已经访问过

nMaxX=QG->point[k][0];

nMaxY=QG->point[k][1];

bound[nBound++]=k;// cun ru tou

// printf("\nThis is the max point is %d, (%d,%d)\n",k,nMaxX,nMaxY);

for(j=0;j<n;j++)

{

int nyTemp=0;

int nk=-1;

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

{

if(flagVisited[i]==false )

{

if(QG->point[i][1]>nyTemp && QG->point[i][0]>nMaxX)

{

nyTemp=QG->point[i][1];

nk=i;

}//if

}//if;function: find the max y among unvisited elements

}//for i

if(nk==-1)

{

continue;

}

else

{

//printf("\nFind the next bigger one is %d\n",nk);

if(QG->point[nk][0]>nMaxX)

{

//printf("\n------------------------%d\n",nk);

bound[nBound++]=nk;

nMaxX=QG->point[nk][0];

nMaxY=QG->point[nk][1];

flagVisited[nk]=true;//标记为已经访问过

}//if

}

//}//if

}//for j 得到这一次要计算的边界的起点

}//if k==-1

//printf("This is the first bound\n");

/\*for(i=0;i<=nBound;i++)//print the bound

{

if(bound[i]==-1)

{

continue;

}

else

{

printf("%d->",bound[i]);

}

}\*/

// printf("\nThis is the end of the bound.\n");

for(i=0;i<=nBound;i++)//save the bound

{

if(bound[i]==-1)

{

continue;

}

else

{

qgt->arrayBound[abi][aby++]=bound[i];

}

}//for to save the bound to arrayBound

qgt->boundRecord[nBoundCount++]=aby;

abi++;

aby=0;

}//while

qgt->nRowCount=nBoundCount;

/\*printf("\nThis is the arrayBound\n");

for(i=0;i<qgt->nRowCount;i++)

{

printf("\nThe %d bound is :\n",(i+1));

for(j=0;j<qgt->boundRecord[i];j++)

{

printf("%d->",qgt->arrayBound[i][j]);

}

}\*/

}//getOrderedPoint

void initialArrayTemp(QieGePointTemp \*qgt)

{

//QieGePointTemp \*qgt;

int i=0,j=0;

for(i=0;i<MAX\_VERTEX\_NUM;i++)

{

for(j=0;j<MAX\_VERTEX\_NUM;j++)

{

qgt->boundArrayTemp[i][j]=-1;

}

}

}

long calculateSquare(QieGePoint \*QG,QieGePointTemp \*qgt)

{

int i=0,j=0,k=0;

int sTemp[MAX\_VERTEX\_NUM]={0};

int nx=0,ny=0;

long totalS=0;

int n=qgt->nRowCount;

/\*printf("\nThis is the arrayBound\n");

for( i=0;i<qgt->nRowCount;i++)

{

printf("\nThe %d bound is :\n",(i+1));

for( j=0;j<qgt->boundRecord[i];j++)

{

printf("%d->",qgt->arrayBound[i][j]);

}//for j print the bound

}//for i ;print the boundary arrays\*/

// calculate the square

initialArrayTemp(qgt);//initial the boundArrayTemp

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

{

int nc=0;//nCurrentPointNo initial is 0

//printf("\nThe %d bound is :\n",(i+1));

if(i==0)//the first bound

{

nx=ny=BILI;

for( j=0;j<qgt->boundRecord[i];j++)

{

nc=qgt->arrayBound[i][j];

sTemp[i]+=(nx-QG->point[nc][0])\*(ny-QG->point[nc][1]);

nx=BILI;

ny=QG->point[nc][1];

qgt->boundArrayTemp[i][j]=nc;// the first is the bound[0]

}//calculate every element in the first bound

qgt->boudTempRecord[i]=j;

//printf("\nThe %d round square is %d",i+1,sTemp[i]);

totalS+=sTemp[i];

}//if i==0,the first bound

else if(i<n)

{

ny=BILI;

nx=QG->point[qgt->arrayBound[i-1][0]][0];

/\*

\* 修改边界开始

\*/

nc=qgt->arrayBound[i][0];

if(QG->point[nc][0]<QG->point[qgt->arrayBound[i-1][0]][0])

{

for( j=0;j<qgt->boundRecord[i];j++)//if Element->x <last bound[0]

{

nc=qgt->arrayBound[i][j];

qgt->boundArrayTemp[i][j]=nc;// the second is the bound[1]

}//calculate every element in the first bound

qgt->boudTempRecord[i]=j;

}//if

else

{

qgt->boundArrayTemp[i][0]=qgt->boundArrayTemp[i-1][0];

for(j=1;j<=qgt->boundRecord[i];j++)

{

nc=qgt->arrayBound[i][j-1];

qgt->boundArrayTemp[i][j]=nc;// the second is the bound[1]

}//for

qgt->boudTempRecord[i]=j;

}//if find the right head and body

int ncElementY=QG->point[qgt->arrayBound[i][qgt->boundRecord[i]-1]][1];//get the last element of last arrayTemp

for(k=0;k<qgt->boundRecord[i-1];k++)

{

if(ncElementY>QG->point[qgt->arrayBound[i-1][k]][1])

{

qgt->boundArrayTemp[i][j++]=qgt->arrayBound[i-1][k];//modify the boundary ,attention should be paid to j

}

}//for modify boundaries

qgt->boudTempRecord[i]=j;

/\*

\* 修改边界结束

\*/

int ski[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM]={0};

int sm=0;

for(k=0;k<qgt->boudTempRecord[i];k++)

{

//ski[i][k]=0;

if(k==0)

{

//ski[i][k]=0;

for(int ki=0;ki<qgt->boudTempRecord[i-1];ki++)

{

if(ki==0)

{

ny=BILI;

nx=QG->point[qgt->boundArrayTemp[i-1][ki]][0];

}//if

else

{

ny=QG->point[qgt->boundArrayTemp[i-1][ki-1]][1];

nx=QG->point[qgt->boundArrayTemp[i-1][ki]][0];

}//else if

if(ski[i][k]<(ny-QG->point[qgt->boundArrayTemp[i][k]][1])\*(nx-QG->point[qgt->boundArrayTemp[i][k]][0]))

{

ski[i][k]=(ny-QG->point[qgt->boundArrayTemp[i][k]][1])\*(nx-QG->point[qgt->boundArrayTemp[i][k]][0]);

}//if

//ski[i][k]=(ny-QG->point[qgt->boundArrayTemp[i][k]][1])\*(nx-QG->point[qgt->boundArrayTemp[i][k]][0]);

}//for ki

}//if k

else

{

int kj=0;

//int start=0;

//ski[i][k]=0;

for(kj=0;kj<qgt->boudTempRecord[i-1];kj++)

{

if(QG->point[qgt->boundArrayTemp[i-1][kj]][0]<QG->point[qgt->boundArrayTemp[i][k]][0])

{

continue;

}//if

if(kj==0)

{

//printf("--------------------------------------------------kj==0");

ny=QG->point[qgt->boundArrayTemp[i][0]][1];

nx=QG->point[qgt->boundArrayTemp[i-1][1]][0];

}//if kj==0

else

{

ny=QG->point[qgt->boundArrayTemp[i-1][kj-1]][1];

nx=QG->point[qgt->boundArrayTemp[i-1][kj]][0];

//start=kj;

}

if(ski[i][k]<(ny-QG->point[qgt->boundArrayTemp[i][k]][1])\*(nx-QG->point[qgt->boundArrayTemp[i][k]][0]))

{

ski[i][k]=(ny-QG->point[qgt->boundArrayTemp[i][k]][1])\*(nx-QG->point[qgt->boundArrayTemp[i][k]][0]);

}//if

//start=kj;

}//for kj

}//else

//printf("\n[%d][%d]'s square is %ld\n",i+1,k,ski[i][k]);

sm+=ski[i][k];

}//for k

//printf("\nThe %d bound and %d boudn square is [%d]:\n",i,i+1,sm);

sTemp[i]=sm;

totalS+=sTemp[i];

sm=0;

}//else if(i<n)

if(i==n)//the last bound

{

sTemp[i]=0;

ny=BILI;

for( j=0;j<qgt->boudTempRecord[i-1];j++)

{

nc=qgt->boundArrayTemp[i-1][j];

nx=QG->point[nc][0];

if(sTemp[i]<(nx-0)\*(ny-0))

{

sTemp[i]=(nx-0)\*(ny-0);

}

if(j<qgt->boundRecord[i-1])

{

nx=QG->point[nc][0];

}

else

nx=BILI;

ny=QG->point[qgt->arrayBound[i-1][j-1]][1];

//ny=QG->point[nc][1];

// printf("%d->",qgt->boundArrayTemp[i-1][j]);

}//calculate every element in the last bound

//printf("\nThe %d round square is %d\n",i+1,sTemp[i]);

//sTemp[i]=sm;

totalS+=sTemp[i];

}//the last bound

}//calculate the square \*/

return totalS;

}

void printBoundTemp(QieGePointTemp \*qgt)

{

printf("\nNew Boundary Last is:\n");

for(int i=0;i<qgt->nRowCount;i++)

{

for(int j=0;j<qgt->boudTempRecord[i];j++)

{

printf("%d->",qgt->boundArrayTemp[i][j]);

}

printf("\n");

}

}

/\*主函数\*/

int main(int argc, char\* argv[])

{

QieGePoint QG;

QieGePointTemp qgt;

createQieGePoint(&QG);

//printQieGePoint(&QG);

//printf("Hello World!\n");

getOrderedPoint(&QG,&qgt);

if(calculateSquare(&QG,&qgt)>=10000 || calculateSquare(&QG,&qgt)<5000)

{

printf("\nResult is error ,please run it again.\n");

}

else

printf("The cut result is [%d]",calculateSquare(&QG,&qgt));

//printBoundTemp(&qgt);

return 0;

}