**参考文献**

[1]刘玉珍,刘润涛.简单多边形的最小外接矩形算法[J].哈尔滨理工大学学报,2008(02):5-7.

[2]金文华,何涛,刘晓平,唐卫清,唐荣锡.基于有序简单多边形的平面点集凸包快速求取算法[J].计算机学报,1998(06):533-539.

**附录1**

**BFS时间测算**

clc;

clear;

load out2.txt;

a=out2;

dx=[1;1;1;0;0;-1;-1;-1;2;0;0;-2];

dy=[1;-1;0;1;-1;1;-1;0;0;2;-2;0];

for i=1:420

for j=1:560

v(i,j)=0;

end

end

tic;

point=[];% 1为序号 2为个数

for i=1:20

point(i,1)=i;

end

id=0;

for i=1:420

for j=1:560

if a(i,j)==1&&v(i,j)==0

id=id+1;

tail=1;

head=0;

queue(tail,1)=i;

queue(tail,2)=j;

while head~=tail

head=head+1;

for k=1:16

cur\_x=queue(head,1)+dx(k,1);%移动x

cur\_y=queue(head,2)+dy(k,1);%移动y

if a(cur\_x,cur\_y)~=1||v(cur\_x,cur\_y)~=0

continue ;%可以访问 已经访问

end

if cur\_x<0||cur\_x>420||cur\_y<0||cur\_y>560

continue ;%当前坐标是否超出界线

end

v(cur\_x,cur\_y)=id;

tail=tail+1;

queue(tail,1)=cur\_x;

queue(tail,2)=cur\_y;

end

end

end

end

end

cnt=[zeros(20,1)];

for i=1:420

for j=1:560

if v(i,j)~=0

cur=v(i,j);

cnt(cur)=cnt(cur)+1;

end

end

end

toc;

disp(['运行时间: ',num2str(toc)]);

**附录2**

**BFS求连通块**

#include<iostream>

#include<cmath>

#include<ctime>

#include<iomanip>

#include<algorithm>

using namespace std;

int v[500][600];

int a[500][600];

int dx[30]={0,1,1,1,0,0,-1,-1,-1,2,0,0,-2};

int dy[30]={0,1,-1,0,1,-1,1,-1,0,0,2,-2,0};

int x,y;

struct node

{

int cnt;

int num;

}ans[20];

double cur;

bool cmp(node,node);

void DFS(int,int,int);

void local\_base\_point(int);

int main ()

{

ios::sync\_with\_stdio(false);

freopen("spare.txt","r",stdin);

// freopen("out.txt","w",stdout);

for (int i=1 ; i<=420 ; i++ )

for (int j=1 ; j<=560 ; j++ )

{

cin >> cur;

// cout << cur << '\n' ;

if (cur==0)

a[i][j]=0 ;

else

{

a[i][j]=1 ;

}

}

int start=clock();

int id=0;

for (int i=1 ; i<=20 ; i++ )

ans[i].num = i ;

for (int i=1 ; i<=420 ; i++ )

for (int j=1 ; j<=560 ; j++ )

if (a[i][j]==1&&v[i][j]==0)

DFS(i,j,++id);

for (int i=1 ; i<=420 ; i++ )

for (int j=1 ; j<=560 ; j++ )

if (v[i][j]!=0)

ans[v[i][j]].cnt++ ;//连通块个数计算

sort(ans+1,ans+20+1,cmp);//降序排序

for (int i=1 ; i<=10 ; i++ )

cout << ans[i].num << ':' << ans[i].cnt << '\n' ;//输出结果展示

for (int i=1 ; i<=10 ; i++ )

{

local\_base\_point(i);//求圆心

cout << "id:" << i << ' ' << "x:" << x << ' ' << "y:" << y << '\n' ;

}

return 0 ;

}

bool cmp(node x,node y)

{

return x.cnt>y.cnt;

}

void DFS(int x,int y,int id)

{

if (x<0||x>420||y<0||y>560)

return ;//出界格子

if (v[x][y]!=0||a[x][y]!=1)

return ;//是否可行、是否访问

v[x][y]=id ;//标记编号

for (int i=1 ; i<=16 ; i++ )

DFS(x+dx[i],y+dy[i],id);

}

void local\_base\_point(int id)

{

int head = 0 ;

int tail = 0 ;

int X[700][3] ;//420行

int distance\_X[700];//head tail 距离

int distance\_Y[700];//head tail 距离

int MAX = 0 ;

for (int i=1 ; i<=420 ; i++ )

{

head = 0 ;

tail = 0 ;

for (int j=1 ; j<=560 ; j++ )

{

if (v[i][j]==id)

if (a[i][j]==1)

{

head = j ;

break ;

}

}

for (int j=560 ; j>=1 ; j-- )

{

if (v[i][j]==id)

if (a[i][j]==1)

{

tail = j ;

break ;

}

}

X[i][1] = head ;

X[i][2] = tail ;

}

for (int i=1 ; i<=420 ; i++ )

distance\_X[i] = X[i][2]-X[i][1] ;//求head tail 距离

MAX = 0 ;

//定位坐标轴x

for (int i=1 ; i<=420 ; i++ )

{

if (MAX<distance\_X[i])

{

MAX = distance\_X[i] ;

head = i ;

tail = i ;

}

if (MAX==distance\_X[i])

tail = i ;

}

x = floor((tail+head)/2);

//-------------------------------------------------------------

int Y[700][3] ; //560列

for (int i=1 ; i<=560 ; i++ )

{

head = 0 ;

tail = 0 ;

for (int j=1 ; j<=420 ; j++ )

{

if (v[j][i]==id)

if (a[j][i]==1)

{

head = j ;

break ;

}

}

for (int j=420 ; j>=1 ; j-- )

{

if (v[j][i]==id)

if (a[j][i]==1)

{

tail = j ;

break ;

}

}

Y[i][1] = head ;

Y[i][2] = tail ;

}

for (int i=1 ; i<=560 ; i++ )

distance\_Y[i] = Y[i][2]-Y[i][1] ;//求head tail 距离

MAX = 0 ;

//定位坐标轴y

for (int i=1 ; i<=560 ; i++ )

{

if (MAX<distance\_Y[i])

{

MAX = distance\_Y[i] ;

head = i ;

tail = i ;

}

if (MAX==distance\_Y[i])

tail = i ;

}

y = floor((tail+head)/2);

}

**附录3**

**最小面积矩形**

clc;

clear;

out2=imread('C:\Users\leafpigbirds\Desktop\不规则图形.bmp');

tic;

% t1=clock;

[r c]=find(out2==1);

[rectx,recty,area,perimeter]=minboundrect(c,r,'a');

figure(1);

imshow(out2);

line(rectx(:),recty(:),'color','r');

hold on;

plot(rectx(1,1),recty(1,1),'r+',rectx(2,1),recty(2,1),'g+');

[wid hei]=minboxing(rectx(1:end-1),recty(1:end-1));

%Slope斜率判断方位

point=[];%1为x,2为y

for i=1:4

point(i,1)=rectx(i,1);

point(i,2)=recty(i,1);

end

%1-2

dis(1,1)=sqrt((point(1,1)-point(2,1))^2+(point(1,2)-point(2,2))^2);

%2-3

dis(2,1)=sqrt((point(2,1)-point(3,1))^2+(point(2,2)-point(3,2))^2);

%3-4

dis(3,1)=sqrt((point(3,1)-point(4,1))^2+(point(3,2)-point(4,2))^2);

%4-1

dis(4,1)=sqrt((point(4,1)-point(1,1))^2+(point(4,2)-point(1,2))^2);

%斜率计算

cnt1=0;

cnt2=0;

cur=0;%当前位置

k12=(point(1,2)-point(2,2))/(point(1,1)-point(2,1));

k23=(point(2,2)-point(3,2))/(point(2,1)-point(3,1));

b12=-k12\*rectx(1,1)+recty(1,1);

b34=-k12\*rectx(3,1)+recty(3,1);

for i=floor(rectx(1,1)):floor(rectx(2,1))

cur=k12\*i+b12;

% hold on;

% plot(i,floor(cur),'b.',i,floor(cur+2),'b.');

if out2(floor(cur),i)==1

cnt1=cnt1+1;

end

if out2(floor(cur+2),i)==1

cnt1=cnt1+1;

end

if out2(floor(cur-2),i)==1

cnt1=cnt1+1;

end

end

for i=floor(rectx(3,1)):-1:floor(rectx(4,1))

cur=k12\*i+b34;

% hold on;

% plot(i,floor(cur),'b.',i,floor(cur+2),'b.');

if out2(floor(cur),i)==1

cnt2=cnt2+1;

end

if out2(floor(cur+2),i)==1

cnt2=cnt2+1;

end

if out2(floor(cur-2),i)==1

cnt2=cnt2+1;

end

end

mid\_x=floor((rectx(3,1)+rectx(4,1))/2);

mid\_y=floor((recty(3,1)+recty(4,1))/2);

% hold on;

% plot(mid\_x,mid\_y,'b\*');

Long=-point(4,2)+mid\_y;

Short=point(4,1)-mid\_x;

Ans\_x=mid\_x+Long;

Ans\_y=mid\_y+Short;

toc;

disp(['运行时间: ',num2str(toc)]);