**数据结构课程设计**

**班级：1615403**

**学号：161540205**

**姓名：张佳钰**

**指导老师：孙涵**

9.迷宫小游戏

采用的数据结构：

静态数组 栈

算法思想：

并查集生成迷宫：

先随机选取两个点为起点和终点

初始化矩阵 每个点都初始化一个父节点 father[(i-1)\*width+j]=(i-1)\*width+j

随机选取两个相邻点

若两个相邻点的father相同 说明两个点在同一个集合内

否则 说明两个点之间没有路相连通 两个点没有相连通 则 father[第一个 点]=father[第二个点] 将两个点连通 然后 将两个点之间的墙打通（置0）

当起点的father和终点的father相同时 说明两个点之间有路相连

路径搜索：

深搜

若当前点有方向可走入栈 若无 则出栈

同时 将每一步操作入栈 (pop 或 push) 方便显示路径

源程序：

函数主要写在 maze2.js中

//maze2.js

/// <reference path="jquery.min.js" />

var on\_going=0;

var width=0;

var height=0;

var hard=0;

var flag=parseInt(0);

var a;

var count\_visit=0;

var correct\_bomb=0;

var visit=new Array();

var father;

var cur;

var start\_point;

var end\_point;

var rank;

var count;

var record;

/\*

var app = new Vue({

el: '#record',

data: {

bomb\_num: bomb\_num,

bomb\_cur: bomb\_cur

}

});

\*/

function init\_visit(visit,flag)

{

visit = new Array();

for(var i=0;i<=height+1;i++)

{

visit[i]=new Array();

for(var j=0;j<=width+1;j++)

{

visit[i][j]=flag;

}

}

return visit;

}

function init\_rank(rank)

{

rank = new Array();

for(var i=0;i<(height+1)\*(width+1);i++)

{

rank[i]=1;

}

return rank;

}

function Create\_init\_maze(width\_w,height\_w,hard\_w)

{

a=new Array();

var count=0;

height=height\_w;

width=width\_w;

hard=hard\_w;

console.log(width,height,hard);

visit=new Array();

visit = init\_visit(visit,-1);

console.log(visit);

for(var i=1;i<=height\_w;i++)

{

a[i]=new Array();

if(i%2==0)

{

a[i][0]=-1;

a[i][1]=-1;

for(var j=2;j<=width\_w;j+=2)

{

a[i][j]=0;

visit[i][j]=0;

count++;

a[i][j+1]=-1;

}

}

else

{

for(var j=1;j<=width\_w;j++)

{

a[i][j]=-1;

}

}

}

console.log(count);

a[0]=new Array();

a[height+1]=new Array();

for(var i=0;i<=height\_w;i++)

{

a[i][0]=-1;

a[i][(width\_w+1)]=-1;

}

for(var j=0;j<=width\_w+1;j++)

{

a[0][j]=-1;

a[height+1][j]=-1;

/// a[6][j]=-1;

}

console.log(a);

count\_visit=count;

return a;

}

function pick\_in\_out()

{

var flag=1;

var i=0;

flag=Math.floor(Math.random()\*2);

if(flag)

{

var j;

var down,up;

j=Math.floor(Math.random()\*width)+1;

down = j%2==0?j:j+1;

j=Math.floor(Math.random()\*width)+1;

up= j%2==0?j:j+1;

start\_point={x:height-1,y:down};

end\_point={x:2,y:up};

}

else

{

var j;

var right,left;

j=Math.floor(Math.random()\*height)+1;

left = j%2==0?j:j+1;

j=Math.floor(Math.random()\*height)+1;

right= j%2==0?j:j+1;

start\_point={x:left,y:2};

end\_point={x:right,y:width-1};

}

cur=start\_point;

return {in:start\_point,out:end\_point};

}

function init\_father(father\_f)

{

father\_f = new Array();

for(var i=0;i<=(height+1)\*(width+1);i++)

{

father\_f[i]=i;

}

return father\_f;

}

function find\_father(x)

{

var root\_r;

root\_r=x;

while(father[root\_r]!=root\_r)

{

root\_r=father[root\_r];

}

while(root\_r!=x)//查找的同时简化集合

{

var temp=father[x];

father[x]=root\_r;

x=temp;

}

/// console.log(root\_r);

return root\_r;

}

function Union\_root(point\_1,point\_2)

{

var x=find\_father(point\_1);

var y=find\_father(point\_2);

console.log(x,y);

if(x!=y)

{

if(rank[x]<=rank[y])

{

father[x]=y;

rank[y]=rank[x]+rank[y];

}

else

{

father[y]=x;

rank[x]=rank[x]+rank[y];

}

}

console.log("union ："+x+" "+y,father[y],father[x]);

}

function point\_to\_count(i,j)

{

//console.log((i-1)\*(width)+j);

return (i-1)\*(width)+j;

}

function has\_way(cur\_x,cur\_y)

{

var x=[-2,0,2,0];

var y=[0,2,0,-2];

for(var i=0;i<x.length;i++)

{

new\_x=cur\_x+x[i];

new\_y=cur\_y+y[i];

if(new\_x>0&&new\_x<height&&new\_y>0&&new\_y<width)

{

if(visit[new\_x][new\_y]==0)

{

return 1;

}

}

}

return 0;

}

function Union\_make\_maze(a,point)

{

var cc=0;

console.log(point);

start\_point = point.in;

end\_point = point.out;

var flag=0;

var k;

while(find\_father(point\_to\_count(point.in.x,point.in.y))!=find\_father(point\_to\_count(point.out.x,point.out.y)))

{

console.log("first: "+find\_father(point\_to\_count(point.in.x,point.in.y)),"second: "+find\_father(point\_to\_count(point.out.x,point.out.y)));

var cur\_x = Math.floor(Math.random()\*(height-2))+1;

cur\_x = cur\_x%2==0?cur\_x:(cur\_x+1);

var cur\_y = Math.floor(Math.random()\*(width-2))+1;

cur\_y = cur\_y%2==0?cur\_y:(cur\_y+1);

cc++;

if(cc>2000)

{

return 0;

}

console.log("cur\_x: "+cur\_x,"cur\_y: "+cur\_y);

// var j=prompt("pp");

var x=[-2,0,2,0];

var y=[0,2,0,-2];

var x\_1=[-1,0,1,0];

var y\_1=[0,1,0,-1];

var tmp=Math.floor(Math.random()\*4);

var new\_x,new\_y;

console.log("cur\_x: "+cur\_x,"cur\_y: "+cur\_y,"tmp: "+tmp);

new\_x=cur\_x+x[tmp];

new\_y=cur\_y+y[tmp];

console.log(cc);

if(new\_x>0&&new\_x<height&&new\_y>0&&new\_y<width)

{

if(a[new\_x][new\_y]==0&&a[cur\_x][cur\_y]==0)

{

console.log("1 find\_fa: cur: "+find\_father(point\_to\_count(cur\_x,cur\_y))+"find\_fa cur : "+find\_father(point\_to\_count(new\_x,new\_y)));

if(find\_father(point\_to\_count(cur\_x,cur\_y))!=find\_father(point\_to\_count(new\_x,new\_y)))

{

console.log("2 find\_fa: cur: "+find\_father(point\_to\_count(cur\_x,cur\_y))+"find\_fa cur : "+find\_father(point\_to\_count(new\_x,new\_y)));

Union\_root(point\_to\_count(new\_x,new\_y),point\_to\_count(cur\_x,cur\_y));

a[cur\_x+x\_1[tmp]][cur\_y+y\_1[tmp]]=0;

console.log(cur\_x,cur\_y,new\_x,new\_y,cur\_x+x\_1[tmp],cur\_y+y\_1[tmp]);

console.log(a);

}

}

}

}

//console.log("final: ",cur\_x,cur\_y,new\_x,new\_y,cur\_x+x\_1[tmp],cur\_y+y\_1[tmp]);

console.log(a);

return 1;

}

function back\_place(num,type="normal")

{

//setTimeout($(`input[id=${num}]`).removeClass("walk"), 3000);

//setTimeout($(`input[id=${num}]`).addClass("default"), 3000);

if(type=="help")

{

$(`input[id=${num}]`).removeClass("help");

}

else

{

$(`input[id=${num}]`).removeClass("walk");

}

$(`input[id=${num}]`).addClass("default");

}

function show\_place(num,index,type="normal")

{

// $(`input[id=${num}]`).val(index);

//console.log("index: "+index);

//setTimeout($(`input[id=${num}]`).removeClass("default"), 3000);

//setTimeout($(`input[id=${num}]`).addClass("walk"), 3000);

if(type=="help")

{

$(`input[id=${num}]`).addClass("help");

}

else

{

$(`input[id=${num}]`).addClass("walk");

}

$(`input[id=${num}]`).removeClass("default");

//$(`input[id=${num}]`).addClass("walk");

}

function hello()

{

console.log("hello");

}

function DFS\_search(cur)

{

var x=[-1,0,1,0];

var y=[0,1,0,-1];

var p;

var new\_x,new\_y;

var flag=1;

var index=0;

var st=new Array();

record=new Array();

console.log(cur);

st.push(cur);

console.log("st: "+st);

p=cur;

var i;

visit[p.x][p.y]=1;

while(((p.x!=end\_point.x)||(p.y!=end\_point.y))&&st.length)

{

// setTimeout("console.log('pp')",5000);

if(flag==0)

{

//if((p.x!=end\_point.x)||(p.y!=end\_point.y))

//setTimeout(back\_place(point\_to\_count(p.x,p.y)), 10000);

//back\_place(point\_to\_count(p.x,p.y));

record.push({point:point\_to\_count(p.x,p.y),todo:"pop"});

console.log(st);

p=st.pop();

//record.push({point:point\_to\_count(p.x,p.y),todo:"pop"});

console.log("pop(): "+point\_to\_count(p.x,p.y));

}

flag=0;

console.log("index: "+index+" point\_to\_count(p.x,p.y) : "+point\_to\_count(p.x,p.y));

//console.log(point\_to\_count(p.x,p.y));

console.log("p.x: "+p.x+" p.y: "+p.y+" i : "+i);

for(i=0;i<x.length&&!flag;i++)

{

new\_x=p.x+x[i];

new\_y=p.y+y[i];

console.log("new\_x: "+new\_x+" new\_y: "+new\_y+" i : "+i);

if(new\_x>0&&new\_x<height&&new\_y>0&&new\_y<width)

{

//console.log("visit: "+visit[new\_x][new\_y]);

// console.log("st.length: "+st.length);

if(a[new\_x][new\_y]==0&&(visit[new\_x][new\_y]==0))

{

flag=1;

//st.push(p);

st.push({x:p.x,y:p.y});

//setTimeout(show\_place(point\_to\_count(p.x,p.y),index++), 10000);

//show\_place(point\_to\_count(p.x,p.y),index++);

//record.push({point:point\_to\_count(p.x,p.y),todo:"push",index:index++});

record.push({point:point\_to\_count(new\_x,new\_y),todo:"push",index:index++});

visit[new\_x][new\_y]=1;

p={x:new\_x,y:new\_y};

// console.log(visit);

//break;

}

}

}

}

console.log(a);

//console.log(st);

}

function make\_maze(w,h)

{

var c=0;

c++;

var flag=0;

var point;

var d;

// var w=$('#weight').val();

// var h=$('#width').val();

if(w==''||h=='')

{

alert('Please fill the blank.');

return ;

}

while(!flag)

{

d=Create\_init\_maze(w,h,5);

father=init\_father(father);

rank=init\_rank(rank);

point=pick\_in\_out();

flag=Union\_make\_maze(d,point);

console.log(a);

console.log("flag: "+flag);

}

a=d;

console.log(c);

console.log(d);

var text='';

var class\_type;

var count=0;

for(var i=1;i<=height;i++)

{

for(var j=1;j<=width;j++)

{

class\_type= d[i][j]==0?"default":"block";

count++;

text+=`<input type="button" class="${class\_type}" value="" name="${count}" id="${count}" onmousedown="check\_bomb(this.id)">`;

}

text+=`</br>`;

}

console.log(point);

console.log(point\_to\_count(point.in.x,point.in.y));

console.log(point\_to\_count(point.out.x,point.out.y));

$('#bg').append(text);

$(`input[id=${point\_to\_count(point.in.x,point.in.y)}]`).removeClass("default");

$(`input[id=${point\_to\_count(point.in.x,point.in.y)}]`).addClass("in");

$(`input[id=${point\_to\_count(point.out.x,point.out.y)}]`).removeClass("default");

$(`input[id=${point\_to\_count(point.out.x,point.out.y)}]`).addClass("out");

$('#re\_start').show();

$('#help').show();

$('#ask\_height\_width').hide();

}

function key\_event()

{

var cur\_x=cur.x;

var cur\_y=cur.y;

var x=[-1,0,1,0];

var y=[0,1,0,-1];

visit=init\_visit(visit,0);

visit[cur\_x][cur\_y]=1;

var key=[38,39,40,37];

document.onkeydown=function(event){

var new\_x,new\_y;

var e = event || window.event || arguments.callee.caller.arguments[0];

if(e)

{

for(var i=0;i<4;i++)

{

if(e.keyCode==key[i])

{

//要做的事情

new\_x=cur\_x+x[i];

new\_y=cur\_y+y[i];

if(new\_x>1&&new\_x<height&&new\_y>1&&new\_y<width)

{

if(a[new\_x][new\_y]==0)

{

if(visit[new\_x][new\_y]==0)

{

show\_place(point\_to\_count(new\_x,new\_y),0);

cur\_x=new\_x;

cur\_y=new\_y;

visit[new\_x][new\_y]=1;

}

else if(visit[new\_x][new\_y]==1)

{

back\_place(point\_to\_count(cur\_x,cur\_y));

visit[cur\_x][cur\_y]=0;

cur\_x=new\_x;

cur\_y=new\_y;

}

}

}

if(cur\_x==end\_point.x&&cur\_y==end\_point.y)

{

alert('Congradulations!!');

re\_start();

}

return false;

break;

}

}

}

};

}

function make()

{

if(on\_going==0)

{

if($('#height').val()==''||$('#width').val()=='')

{

console.log('pp');

alert('please complete the height and width.');

}

else

{

var h=$('#height').val();

var w=$('#width').val();

console.log(h,w);

w=parseInt(w);

h=parseInt(h);

if(w%2==0||h%2==0)

{

alert("请输入奇数!");

}

else

{

on\_going=1;

console.log(w+1,h+2);

make\_maze(w,h);

key\_event();

}

}

}

}

function show\_path()

{

var p;

var show = setInterval(function(){

p=record.shift();

console.log(p);

console.log('a',p.index);

if(p.todo=="push")

{

setTimeout(show\_place, 500, p.point, p.index,"help");

console.log('b',p.index);

}

else

{

setTimeout(back\_place, 500, p.point,"help");

}

console.log('c');

if(record.length==0)

{

clearInterval(show);

}

}, 500);

}

function help()

{

visit=init\_visit(visit,0);

console.log(visit);

DFS\_search(cur);

show\_path();

}

function re\_start()

{

flag=0;

on\_going=0;

$('#bg').empty();

$('#help').hide();

$('#re\_start').hide();

$('#ask\_height\_width').show();

}

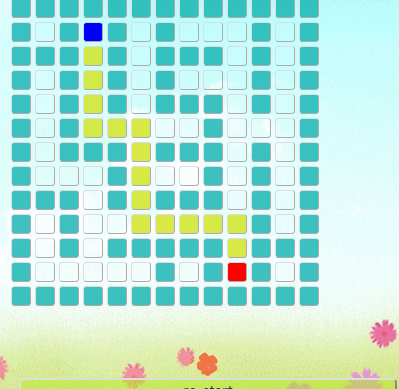
js代码行数：

541行

//注：

点击re\_start 可以重新开始

点击 help\_me 可以 得到帮助



如图

不足：

只是写了前端页面 还没写数据库 存储分数