**电子科技大学**

**计算机科学与工程学院**

**标 准 实 验 报 告**

**（实验）课程名称 软件开发综合实验**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

# 实 验 报 告

**学生姓名：覃翊 学 号： 2016060107008 指导教师：王丽杰**

**实验地点： 主楼B412 实验时间：11.09-11.16**

**一、实验室名称： 计算机学院实验中心A2-412**

**二、实验项目名称：迷宫**

**三、实验学时：16学时**

**四、实验原理：**

**设计并实现一个迷宫游戏，算法能够自动生成具有通路的迷宫地图，寻找迷宫路径并显示通路，能够利用文件形式保存或导入迷宫。**

**五、实验目的：**

**增加编程能力，考验调试能力，了解图形界面开发技术，理解离散集合的定义、表示方法；并查集的基本操作、路径压缩。掌握栈的定义与基本操作**

**六、实验内容：**

**采用算法生成具有通路的迷宫地图；寻找迷宫路径并显示通路。**

**七、实验步骤：**

首先，我先创造了选择界面，摁1从文件导入地图，摁2就新建一个地图。

如果用户输入1，switch语句将其匹配到1的代码块，让用户输入文件的名字，我给出的例子直接输入1就是文件名，读入用户输入的文件名后，用fopen函数，只读方式，打开文件，然后用while循环，读入信息。

当文件指针没有指向文件尾时，分别读入行数，列数，迷宫开头，迷宫结尾，以及迷宫数据。

当用户输入2时，就新建一个迷宫。

新建迷宫我采用的时dfs算法，用时间作种子，生成随机数，模4后得到四个方向，这四个方向不是固定方向，而是优先方向，分别对应上下左右。

什么叫优先方向呢？例如，随机数取模后得1，对应得优先方向是上，则先判断上能不能走。如果上能走，就走上。如果上不能走，再依次走左，下，右。如果随机数取模后得2，对应得优先方向是右，则判断右能不能走，如果右能走，就走右，如果右不能走，则走其他三个方向。同理，这就是优先方向，为此，我写了很长很长得if语句，还要判有没有走过。

**switch(tmp){**

**case 1:**

**if(road[i].lasty - 1 >= 0 && ditu[road[i].lastx][road[i].lasty - 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx - 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx][road[i].lasty - 2] == 1) {**

**road[i].x = road[i].lastx;**

**road[i].y = road[i].lasty - 1;**

**ditu[road[i].x][road[i].y] = 0;**

**}else if(road[i].lastx + 1 < w && ditu[road[i].lastx + 1][road[i].lasty] == 1 && ditu[road[i].lastx + 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx + 2][road[i].lasty] == 1) {**

**road[i].x = road[i].lastx + 1;**

**road[i].y = road[i].lasty;**

**ditu[road[i].x][road[i].y] = 0;**

**}else if(road[i].lasty + 1 < r && ditu[road[i].lastx][road[i].lasty + 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx - 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx][road[i].lasty + 2] == 1) {**

**road[i].x = road[i].lastx;**

**road[i].y = road[i].lasty + 1;**

**ditu[road[i].x][road[i].y] = 0;**

**}**

**break;**

**case 2:**

**if(road[i].lastx + 1 < w && ditu[road[i].lastx + 1][road[i].lasty] == 1 && ditu[road[i].lastx + 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx + 2][road[i].lasty] == 1) {**

**road[i].x = road[i].lastx + 1;**

**road[i].y = road[i].lasty;**

**ditu[road[i].x][road[i].y] = 0;**

**}else if(road[i].lasty - 1 >= 0 && ditu[road[i].lastx][road[i].lasty - 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx - 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx][road[i].lasty - 2] == 1) {**

**road[i].x = road[i].lastx;**

**road[i].y = road[i].lasty - 1;**

**ditu[road[i].x][road[i].y] = 0;**

**}else if(road[i].lasty + 1 < r && ditu[road[i].lastx][road[i].lasty + 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx - 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx][road[i].lasty + 2] == 1) {**

**road[i].x = road[i].lastx;**

**road[i].y = road[i].lasty + 1;**

**ditu[road[i].x][road[i].y] = 0;**

**}**

**break;**

**case 3:**

**if(road[i].lasty + 1 < r && ditu[road[i].lastx][road[i].lasty + 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx - 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx][road[i].lasty + 2] == 1) {**

**road[i].x = road[i].lastx;**

**road[i].y = road[i].lasty + 1;**

**ditu[road[i].x][road[i].y] = 0;**

**}else if(road[i].lasty - 1 >= 0 && ditu[road[i].lastx][road[i].lasty - 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx - 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx][road[i].lasty - 2] == 1) {**

**road[i].x = road[i].lastx;**

**road[i].y = road[i].lasty - 1;**

**ditu[road[i].x][road[i].y] = 0;**

**}else if(road[i].lastx + 1 < w && ditu[road[i].lastx + 1][road[i].lasty] == 1 && ditu[road[i].lastx + 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx + 2][road[i].lasty] == 1) {**

**road[i].x = road[i].lastx + 1;**

**road[i].y = road[i].lasty;**

**ditu[road[i].x][road[i].y] = 0;**

**}**

**break;**

**case 4:**

**if(road[i].lasty - 1 >= 0 && ditu[road[i].lastx][road[i].lasty - 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx - 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx][road[i].lasty - 2] == 1) {**

**road[i].x = road[i].lastx;**

**road[i].y = road[i].lasty - 1;**

**ditu[road[i].x][road[i].y] = 0;**

**}else if(road[i].lastx + 1 < w && ditu[road[i].lastx + 1][road[i].lasty] == 1 && ditu[road[i].lastx + 1][road[i].lasty - 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx + 2][road[i].lasty] == 1) {**

**road[i].x = road[i].lastx + 1;**

**road[i].y = road[i].lasty;**

**ditu[road[i].x][road[i].y] = 0;**

**}else if(road[i].lasty + 1 < r && ditu[road[i].lastx][road[i].lasty + 1] == 1 && ditu[road[i].lastx + 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx - 1][road[i].lasty + 1] == 1 && ditu[road[i].lastx][road[i].lasty + 2] == 1) {**

**road[i].x = road[i].lastx;**

**road[i].y = road[i].lasty + 1;**

**ditu[road[i].x][road[i].y] = 0;**

**}**

**break;**

**如代码所示，每个很长得if里得判断就是判断有没有走过以及是不是墙壁。**

**寻路算法与生成迷宫算法类似，用栈与并查集实现记录路径。**

**数据结构我采用的是结构，每个结构内有当前坐标的值以及上一个点的坐标的值**

**struct information {**

**int x;**

**int y;**

**int lastx;**

**int lasty;**

**};**

**用结构数组，数组的下标就是天然的栈，要回退，只需要在其他条件判断不成立的时候I - - ；**

**生成地图时，我先让用户输入长宽，然后默认1，1是入口，出口是右下角，默认全都是墙壁，然后开始挖通路，挖好一条通路后，再复用相同的代码，终点在地图中随机取几个，然后生成几条路径做干扰。迷宫就这样生成了。**

**寻找路径也类似，用相同的dfs算法寻路，如果进了死胡同就回退，寻找到终点后从终点不断得到自己上一个点的坐标，最后回到原点，打好标记。**

**然后就是输出，两层for循环打印，遇到路径标记就打印星号。程序结束。**

**所有的代码因为太长，又六百多行，所以粘在这的话显得不美观，所以我放在了Github上，源代码我也附送在了压缩包里，github上可以看到我每次修改了什么，每次增加了什么。改了哪里，删了哪里，都有高亮语法标注出来，非常方便。**

**八、总结及心得体会：**

通过本实验的练习，学会了并查集，以及设计了一个迷宫游戏，极大的提高了自己的变成能力，第一次编出了600行代码的程序，并且生成100 100的迷宫时电脑跑了十分钟还没有生成出来。并且很考验debug的耐心，很考验，代码量一大调试起来都很烦躁很无聊，但幸好最后没有bug了。

**九、对本实验过程及方法、手段的改进建议：**

本程序我还是有些不满意的，因为采用的时bfs算法，所以生成迷宫的时候挖的路是一块一块的，每次回退也只是一格一格的回退，甚至有些时候都挖过了，不剩什么障碍了，但又想不出办法，日后可以与同学讨论，学习其他同学的代码获得提升。

**报告评分：**

**指导教师签字：**