数据结构与算法课程设计

王晓琳
xlwang@sdu.edu.cn

课程目标

1、程序设计能力

- 熟练地使用程序设计语言及其开发环境；熟练地调试程序。

2、数据结构的设计能力

- 巩固和加深对数据结构基本知识的理解，熟练地掌握常见数据结构的存储结构设计和操作（算法）的实现。

3、问题求解能力

- 能够针对实际问题，选择适当的数据结构，设计有效算法；
- 综合运用相关的理论知识、技术和方法，对实际问题进行独立分析，并给出解决方案；
- 了解和掌握问题求解和软件开发的一般过程、方法和途径；
- 学会用系统的观点和软件开发一般规范进行软件开发和问题求解，培养问题分析、问题求解和软件开发能力。

课程要求

- 1. 完成要求题目的软件设计开发，运行得到正确的结果，所编写的程序应是一个独立的产品。
  - 界面设计：可输入、输出，操作简单清晰
  - 功能设计：尽可能满足实际运行要求。有健壮性
  - 编程实现：
- 2. 撰写课程设计报告
- 3. 课堂报告(*)

课程设计题目

- Ⅰ．基本数据结构及其应用
- Ⅱ．数据结构与算法的综合应用

Ⅰ.基本数据结构及其应用

- 1、 跳表实现与分析
- **问题描述：**

  实现并分析跳表结构。
- **基本要求：**
- ①构造并实现跳表ADT，跳表ADT 中应包括初始化、查找、插入、删除指定关键字的元素、删除关键字最小的元素、删除关键字最大的元素等基本操作。
- ② 分析各基本操作的时间复杂性。
- ③针对一个实例实现跳表的动态演示(鼓励使用图形演示)。
- *④能对跳表维护动态数据集合的效率进行实验验证，获得一定量的实验数据，如给定随机产生1000 个数据并将其初始化为严格跳表，在此基础上进行一系列插入、删除、查找操作（操作序列也可以随机生成），获得各种操作的平均时间（或统计其基本操作个数）；获得各操作执行时间的变化情况，应该是越来越大，当大到一定程度后应该进行适当的整理，需设计相应的整理算法，并从数量上确定何时较为合适；能和其他简单线性数据结构，如排序数组上的折半查找进行各类操作效率上的数量对比。

Ⅰ.基本数据结构及其应用

- 2、 外排序
- **问题描述：**

  应用竞赛树结构模拟实现外排序。
- **基本要求：**
- ①设计实现最小竞赛树结构。
- ②设计实现外排序，外部排序中的生成最初归并串以及K路归并都应用竞赛树结构实现；
- ③随机创建一个较长的文件；设置归并路数以及缓冲区的大小；获得外排序的访问磁盘的次数并进行分析。可采用小文件来模拟磁盘块。
- *④ 用某种内部排序法生成最初归并串，然后进行K路归并，给出实验结果，比较访问磁盘次数。

Ⅰ.基本数据结构及其应用

- 3、 关键活动
- **问题描述：**
- 一个工程项目一般有一些子任务（活动）构成，子任务之间有的可以并行执行，有的则必须在完成了其它一些任务后才能执行，如果给出了完成每个任务需要的时间，则可以算出完成整个工程需要的最短时间。在这些子任务中，有些任务即使推迟几天，也不会影响全局的工期，但是有些任务必须准时完成，否则整个工程项目就要因此延误，这种任务就叫“关键活动”。
- **基本要求：**
- 判断一个工程项目的任务调度是否可行，如果可行，则计算出完成整个工程需要的最短时间，输出所有的关键活动，各项活动的最早开始时间，不影响全局工期的最晚开始时间。

- **4. 残缺棋盘的问题:**
- **问题描述**

残缺棋盘(defective chessboard)：是一个有$2^k \times 2^k$个方格的棋盘，其中恰有一个方格残缺。对于任意k，恰好存在2^{2k}种不同的残缺棋盘。

在残缺棋盘中，要求用三格板(triominoes)覆盖残缺棋盘。在覆盖中，任意两个三格板不能重叠，任意一个三格板不能覆盖残缺方格，但三格板必须覆盖其他所有方格。

- **基本要求**

输入棋盘大小和残缺方格的位置，输出覆盖后的棋盘，输出棋盘时要着色，共享同一边界的覆盖应着不同的颜色。棋盘是平面图，因此最多只需4种颜色，为覆盖着色，要求设计贪婪着色启发式方法，以尽量使用较少的颜色。

课程设计题目

- II．数据结构与算法的综合应用
- 综合运用数据结构与算法课程的知识，设计并实现一个软件，解决较复杂的问题。

成绩评估

- 1. 系统（平时检查、验收及提交系统）
- 2. 课堂练习/报告(*)
- 3. 中期报告（只对第II部分题目）
- 4. 课程设计报告
- 5. 考勤
  - 实验考勤、课堂考勤
- 总计：折算成
  - 优、良、中、合格、不合格

开发语言和环境

- C++　　Eclipse 或　VC++等

上机时间、地点

- 计机17，1-2班：
- 时间：
  - 第2-5周，周三下午第7-8节，
  - 第6-9周，周三上午第3-4节，周三下午第7-8节
  - 第10-13周，周三下午第7-8节，
- 地点：
- 1班：　N3楼-109(111)房间
- 2班：　N3楼-104(106)房间

上机时间、地点

- 计机17，3-4班：
- 时间：
  - 第2-5周，周四上午第3-4节，
  - 第6-9周，周三下午第5-6节，周四上午第3-4节
  - 第10-13周，周四上午第3-4节
- 地点：
- 3班：109(111)房间
- 4班：104(106)房间

助教

- 一班：俞旭铮
- 二班：李云开
- 三班：崔晨
- 四班：于海洋

课程设计提交

- 1、课程设计报告
  - ✓ **设计报告文件名**命名格式： 17.班号-学号-姓名-数据结构课程设计报告-题目序号.doc
- 2、中期报告（只对第II部分题目）
  - **中期报告文件名**命名格式： 17.班号-学号-姓名-数据结构课程设计中期报告.doc
- 3、课堂报告PPT（*）
- 4、程序
  - ✓ 上机时间，现场运行验收
  - ✓ ？以文件夹形式提交，文件夹名称以“17.班号-学号-姓名-程序源代码”格式命名
- 课代表收齐统一提交给助教

程序验收及报告提交时间

- 程序**最晚**验收时间：
  - Ⅰ.1 ： 第4周
  - Ⅰ.2 ： 第6周
  - Ⅰ.3 ： 第8周
  - Ⅰ.4 ： 第10周
  - Ⅱ ： 第13周
- 报告**最晚提交**时间：
  - Ⅰ.1 ： 第5周
  - Ⅰ.2 ： 第7周
  - Ⅰ.3 ： 第9周
  - Ⅰ.4 ： 第11周
  - Ⅱ ： 第14周
- **中期报告提交时间：** 第9周

课程设计报告格式要求

学号：	姓名：	班级：
上机学时：	日期：	
课程设计题目：		
软件开发环境：		

报告内容：

1.需求描述

1.1 问题描述

1.2 基本要求

1.3 输入说明

1.4 输出说明

2.设计

2.1 系统结构设计

2.2 设计思路

2.3 数据及数据类(型)定义

2.4.算法设计及分析

（各模块算法及类内函数的算法伪码表示）

3. 测试结果

4. 分析与探讨

5. 附录：实现源代码

报告内容要求

- 报告内容：
- **1. 需求描述**

  1.1 问题描述

  1.2 基本要求

  1.3 输入说明

  - 针对问题，设计输入界面，给出输入界面的形式及格式要求

  1.4 输出说明

  - 针对问题，设计输出界面，给出输出界面的形式及格式要求

报告内容要求

- 2. 设计

  2.1 系统结构设计

  - 用图或文字说明系统由哪几部分（模块）组成；
  - 系统各模块功能说明(功能设计)

  2.2 设计思路

  - 分析问题，确定适合的数据结构
  - 各模块的设计思路

报告内容要求

2.3 数据及数据类(型)定义

- 给出选择的数据结构对应**类的定义**，对每一个类说明：
  - 各属性成员表示的含义
  - 成员函数的定义
- 如果使用**全局变量**，说明各全局变量表示的含义及用途

2.4 算法设计及分析

- 各模块算法伪码描述
- 类内关键操作的算法伪码描述
- 各算法复杂性分析

报告内容要求

3. 测试结果

- 测试输入
- 测试输出
- 测试中的问题及解决
- 注：
  - 测试输入及对应的测试输出可放入一张表格中
  - 有充分的测试数据(简单、一般、无解、错误、边界、小数据、大数据、最好、最坏等各种情况),必须包括容错测试数据
  - 可使用系统截图给出程序的测试结果。

报告内容要求

4. 分析与探讨

- 测试结果分析
- 探讨更多解决问题的途径，或者提出自己的见解，改进算法以得到更好结果的建议。

➢ **5. 附录：实现源代码**

- 程序风格清晰易理解
- 有充分的注释

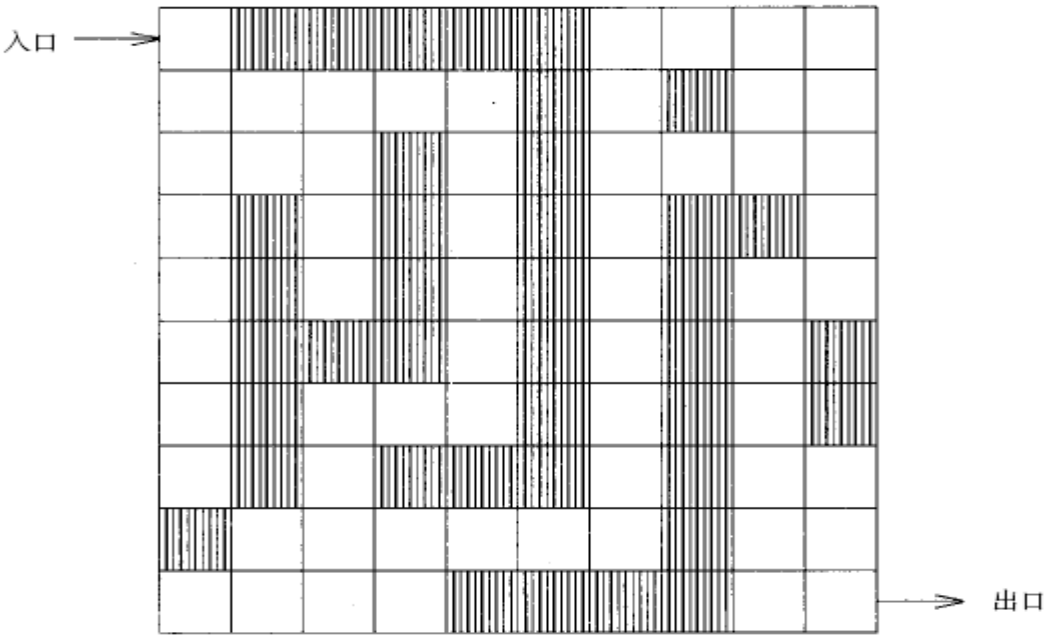
课程设计报告要求

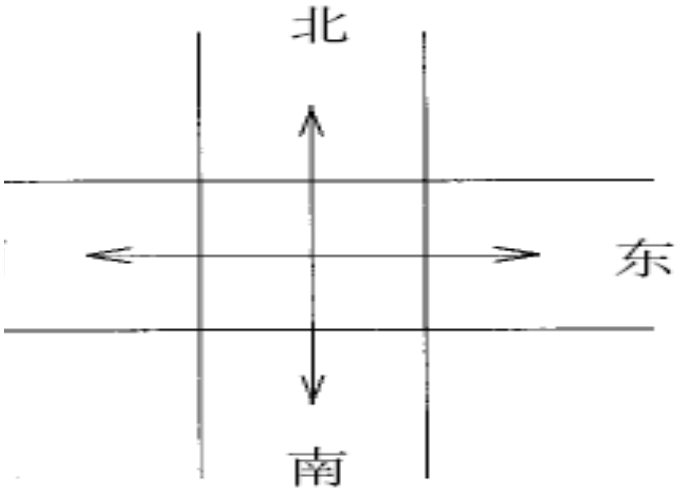
- 课程设计必须独立完成，课程设计报告有雷同的将不得分。
- 必须使用面向对对象方法设计与实现。
- 按时提交。
- 报告中的文字使用5号宋体，单倍行距。

示例：迷宫老鼠

- **1. 问题描述**
  - 迷宫是一个矩形区域，它有一个入口和一个出口。在迷宫的内部包含不能穿越的墙或障碍。在图8-9所示的迷宫中，障碍物沿着行和列放置，它们与迷宫的矩形边界平行。迷宫的入口在左上角，出口在右下角。
  - 迷宫老鼠问题要求寻找一条从入口到出口的路径。每个位置上都没有障碍，且每个位置（第一个除外）都是前一个位置的东、南、西或北的邻居。

需求描述





- **基本要求：**
  自动或手动建立迷宫，然后输出迷宫从入口到出口的路径
  。

需求描述

- **输入说明：**

键盘输入迷宫数据，输入中的每一步要在显示器上给出相应的提示信息和输入的格式要求，迷宫输入完成后，对数据进行有效性检查，若数据有错，要给出错误信息提示。

可给出输入数据的例子进一步说明。

- **输出说明：**

在显示器上以图形(字符)方式输出迷宫矩阵，从入口到出口的路径上用特别符号表示。

可通过输出数据的例子进一步说明。

2.设计

- 2.1　系统结构设计
  - 用图或文字说明系统由哪几部分（模块）组成；
  - 系统各模块功能说明

- 程序主要功能模块
  - 欢迎模块—开始显示欢迎界面
  - 输入模块—按迷宫的设计要求输入迷宫
  - 寻找路径模块—求输入迷宫从入口到出口的一条路径
  - 输出模块—输出寻找路径模块求出的路径

2.1 系统结构设计

- 欢迎模块：
  - 在显示器上显示用文字表示的欢迎界面，欢迎界面格式为：
  - ……
- 寻找路径模块：
  - 求迷宫从入口到出口的一条路径，若找到，则返回true,否则，返回false;
- 输出模块：
  - 在显示器上以迷宫矩阵形式输出从入口到出口的路径，路径上的位置用符号“#”表示。

2.1 系统结构设计

- **输入模块：**
  - 设从键盘输入迷宫矩阵
  - 在输入矩阵数据之前，提示用户输入行数、列数，提示界面为：

    ……
  - 提示用户逐行输入数据，显示提示信息和输入格式要求

    ……
  - 矩阵输入完成后，对用户的输入错误进行验证，
    - 如果在迷宫的入口或出口处有障碍物，则提示输入错误。
    - 如果没有足够的内存来创建数组maze，应输出一个错误信息……。

      ……

人机交互接口的设计

- 欢迎模块、输入模块、输出模块：人机交互接口的设计
- 输入模块主要考虑：
  - 输入方式及输入界面：
  - 图形界面+鼠标点击？键盘输入？文件读入？菜单选择？…
  - 输入数据的有效性验证？
  - 输入提示？
- 输出模块主要考虑：
  - 图形界面？文字输出？文件输出？

- 也可以考虑使用：图像？声音？动画？等

2.1 系统结构设计

- 主程序算法框架：

```
Void main()
{
  1. 显示欢迎界面；//欢迎模块
  2. 按迷宫的设计要求输入迷宫；//输入模块
  3. 求输入迷宫从入口到出口的一条路径；//寻找路径模块
  4. 若找到，则 输出寻找路径模块求出的路径;//输出模块
                否则，cout<<"Nopath"<<endl;
}
```

2.2 设计思路

1、分析问题，确定适合的数据结构

- 假设迷宫有n行和m列，则迷宫有n×m个方格(位置)组成，每个方格上有两种状态：有障碍物/无障碍物；某个位置上是否有障碍物可以用0/1表示，这样一个迷宫就可以用n×m的0/1矩阵表示；
- 迷宫中的每个位置都可用其行号和列号来指定。
- 从迷宫入口到出口的路径，是一组位置的序列，每个位置上都没有障碍，且每个位置(第一个除外)都是前一个位置的东、南、西或北的邻居。

2.2 设计思路

2、各模块的设计思路

- **寻找路径模块的设计思路：**

  首先把迷宫的入口作为当前位置。

  如果当前位置是迷宫出口，那么已经找到了一条路径，搜索工作结束。

  如果当前位置不是迷宫出口，则在当前位置上放置障碍物，以便阻止搜索过程又绕回到这个位置。

  检查相邻的位置中是否有空闲的(即没有障碍物)，如果有，就移动到这个新的相邻位置上，然后从这个位置开始搜索通往出口的路径。如果不成功，选择另一个相邻的空闲位置，并从它开始搜索通往出口的路径。在进入新的相邻位置之前，把**当前位置保存在一个栈**中，…。

  如果相邻的位置中没有空闲的，则回退到上一位置。

  如果所有相邻的空闲位置都已经被探索过，并且未能找到路径，则表明在迷宫中不存在从入口到出口的路径。

- ……其它模块的设计思路：

2.3 数据及数据结构定义

- **1、位置表示:**
- class position{

  private:

  int row; //位置的行坐标

  int col;//位置的列坐标}
- **2、迷宫表示：**n×m的0/1矩阵

0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	0	1	0	0
0	0	0	1	0	1	0	0	0	0
0	1	0	1	0	1	0	1	1	0
0	1	0	1	0	1	0	1	0	0
0	1	1	1	0	1	0	1	0	1
0	1	0	0	0	1	0	1	0	1
0	1	0	1	1	1	0	1	0	0
1	0	0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	1	0	0

在找一个位置的相邻位置时，内部位置和边界位置的处理存在差别

2.3 数据及数据结构定义

- 2、迷宫表示：（n+2)×(n+2)的0/1矩阵,

  int **maze;

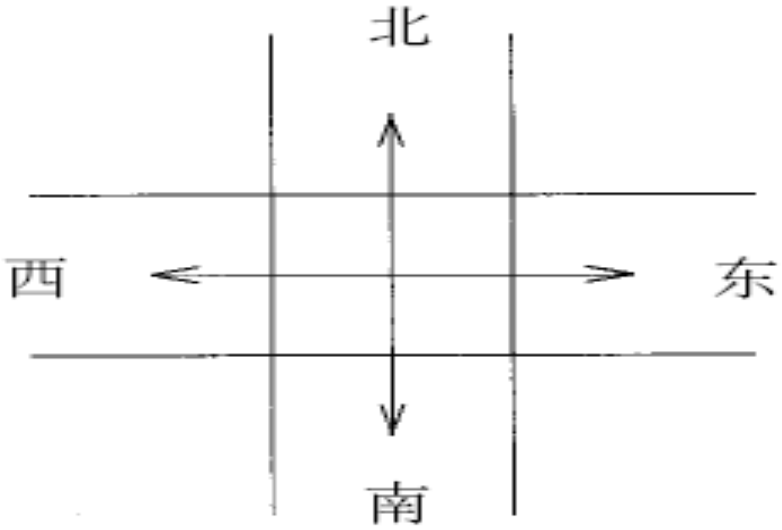
1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	0	0	0	0	1
1	0	0	0	0	0	1	0	1	0	0	1
1	0	0	0	1	0	1	0	0	0	0	1
1	0	1	0	1	0	1	0	1	1	0	1
1	0	1	0	1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0	1	0	1	1
1	0	1	0	0	0	1	0	1	0	1	1
1	0	1	0	1	1	1	0	1	0	0	1
1	1	0	0	0	0	0	0	1	0	0	1
1	0	0	0	0	1	1	1	1	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1

2.3 数据及数据结构定义

- 3. 路径表示：
- 保存的位置序列，保存结构：栈
- 路径path：数组描述的栈arrayStack类型，栈中的元素类型是位置类型Position ：
  - path=new arrayStack<Position> ;

2.3 数据及数据结构定义

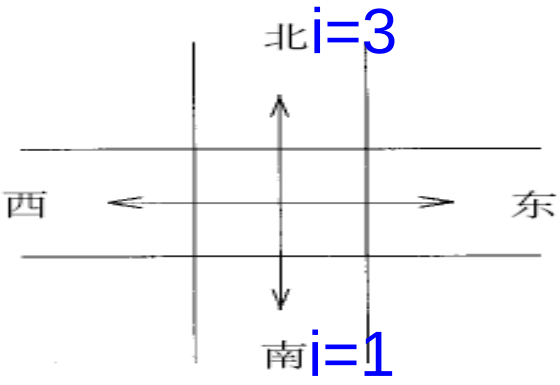
- 4. 相邻位置表示：



	(2,4)	
(3,3)	(3,4)	(3,5)
	(4,4)	

- 用相邻位置与当前位置行坐标和列坐标上的增量表示;
- Position offset[4]; //offset[i].row, offset[i].col的值取决于相邻位置的方向

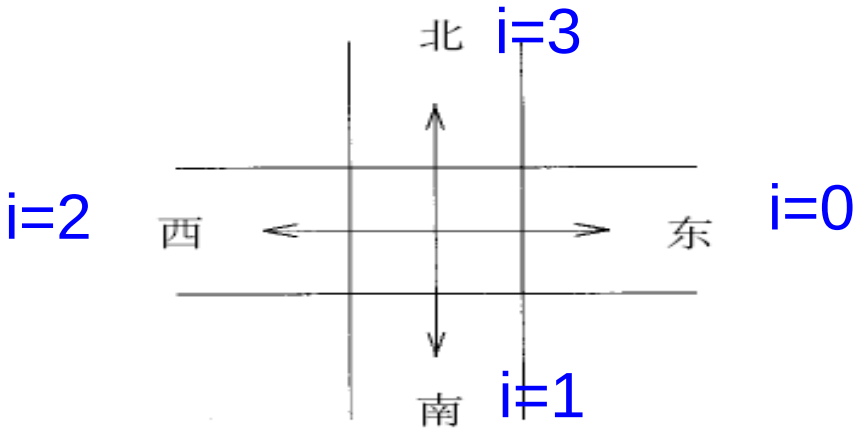
4. 相邻位置表示



	(2,4)	
(3,3)	(3,4)	**(3,5)**
	(4,4)	

i	移动方向	offset[i].row	offset[i].col
0	向右	0	1
1	向下	1	0
2	向左	0	-1
3	向上	-1	0

- 当前位置here的在方向i上的相邻位置(r,c)
- r=here[i].row+offset[i].row
- c=here[i].col+offset[i].col



- 思考：如何选择下一个可行的相邻位置？

2.4 算法设计及分析

- 欢迎模块算法

  Void Welcome()

  ……

- 输入模块算法

  Void InputMaze()

  ……

- 输出模块算法

  Void OutputPath()

  ……

- 寻找路径模块算法

  bool FindPath()

  ……

```
bool FindPath()
{ //寻找从位置(1,1)到出口(n,m)的路径
在迷宫的四周增加一圈障碍物;
//对跟踪当前位置的变量进行初始化
Position here;//当前位置
here.row = 1; here.col = 1;
maze[1][1] = 1; // 放置障碍物,阻止返回入口
//寻找通往出口的路径
while (here不是出口) do {
  选择here的下一个可移动的相邻位置;
  if (存在这样一个相邻位置neighbor) {
    把当前位置here 放入栈path ;
    //移动到相邻位置，并在当前位置放上障碍物
    here = neighbor;
    maze[here.row][here.col] = 1;}
  else {
    //不能继续移动，需回溯
    if (堆栈path为空) return false;
    回溯到path栈顶中的位置; }
}
return true;
}
```

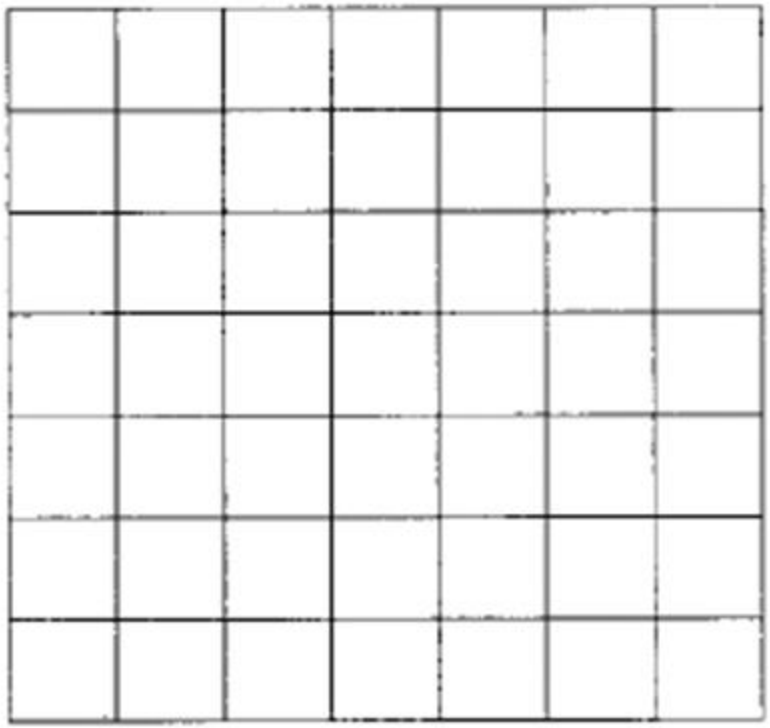
- 以上方法是深度优先搜索、回溯、栈的应用

思考

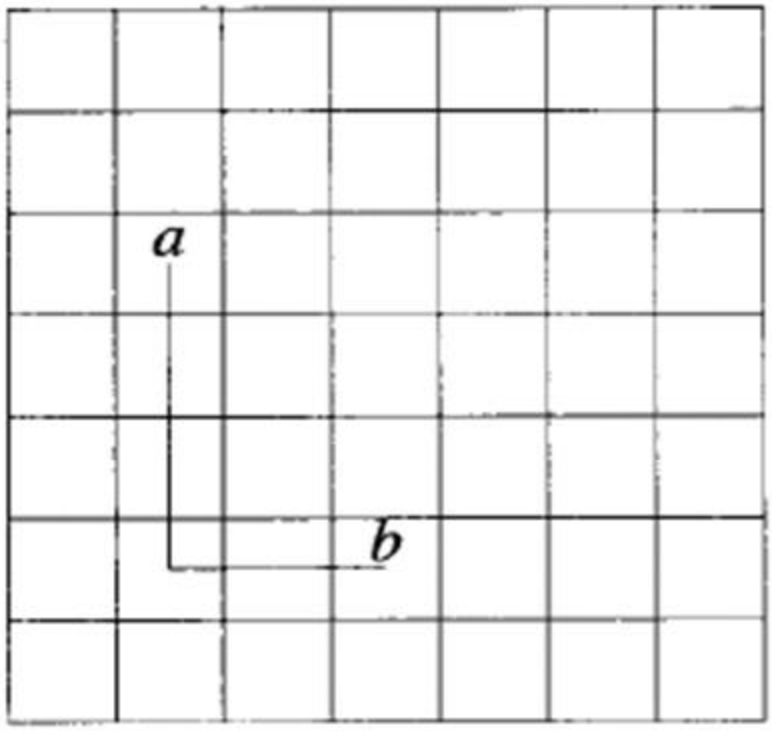
- 入口位置是s, 出口位置是t, 算法如何修改？能否最快地得到路径？
- 如何得到最短路径？

如何得到最短路径？

- 电路布线问题
- 问题描述：



a)



b)

电路布线问题

- 在解决电路布线问题时，一种很常用的方法就是在布线区域叠上一个网格，该网格把布线区域划分成n×m个方格，就像迷宫一样。
- 从一个方格a的中心点连接到另一个方格b的中心点时，转弯处必须采用直角。如果已经有某条线路经过一个方格，则封锁该方格。我们希望使用a和b之间的最短路径来作为布线的路径，以便减少信号的延迟。

电路布线问题

- 输入：网格，电路的起点a及终点b
- 输出：
  - 从方格a到方格b的布线路径：**a和b之间的最短路径**。**若无**布线路径，则给出提示。

电路布线问题

- 输入：网格，电路的起点a及终点b
- 输出：
  - 从方格a到方格b的布线路径：**a和b之间的最短路径。若无**布线路径，则给出提示。

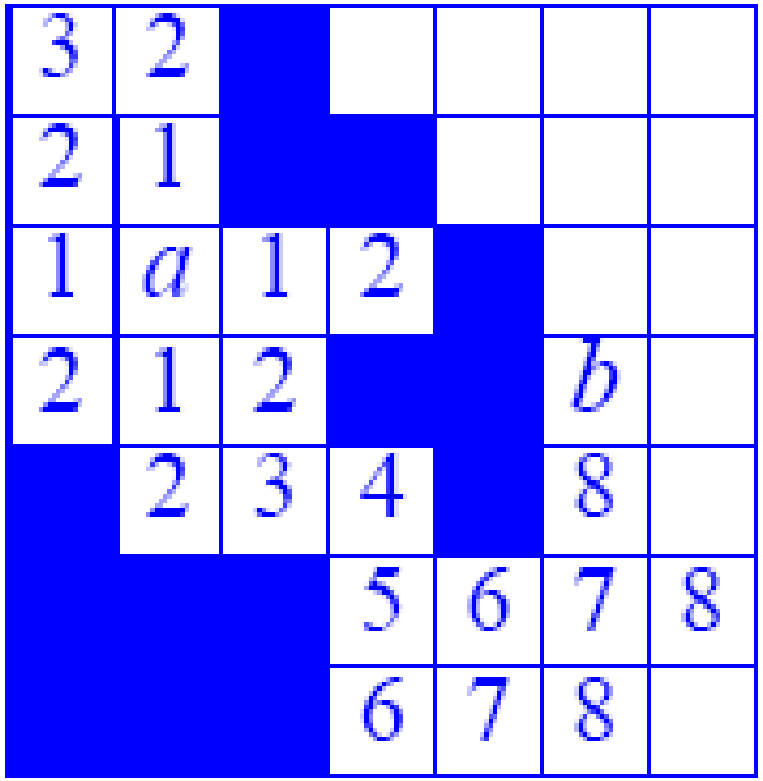
系统结构设计

- 系统结构设计同迷宫问题。
- 不同的是寻找路径: findPath()找的是从指定起点到指定终点的一条最短路径。

- 寻找路径设计思路：
  - 1. 标识网格
  - 2. 构造最短路径

寻找路径设计思路-1/3

1. 标识网格



- 先从位置a开始搜索，
- 把a可到达的相邻方格都标记为1(表示与a相距为1)，
- 然后把标号为1的方格可到达的相邻方格都标记为2(表示与a相距为2)，
- 继续进行下去…，
- 直到到达b或者找不到可到达的相邻方格为止。

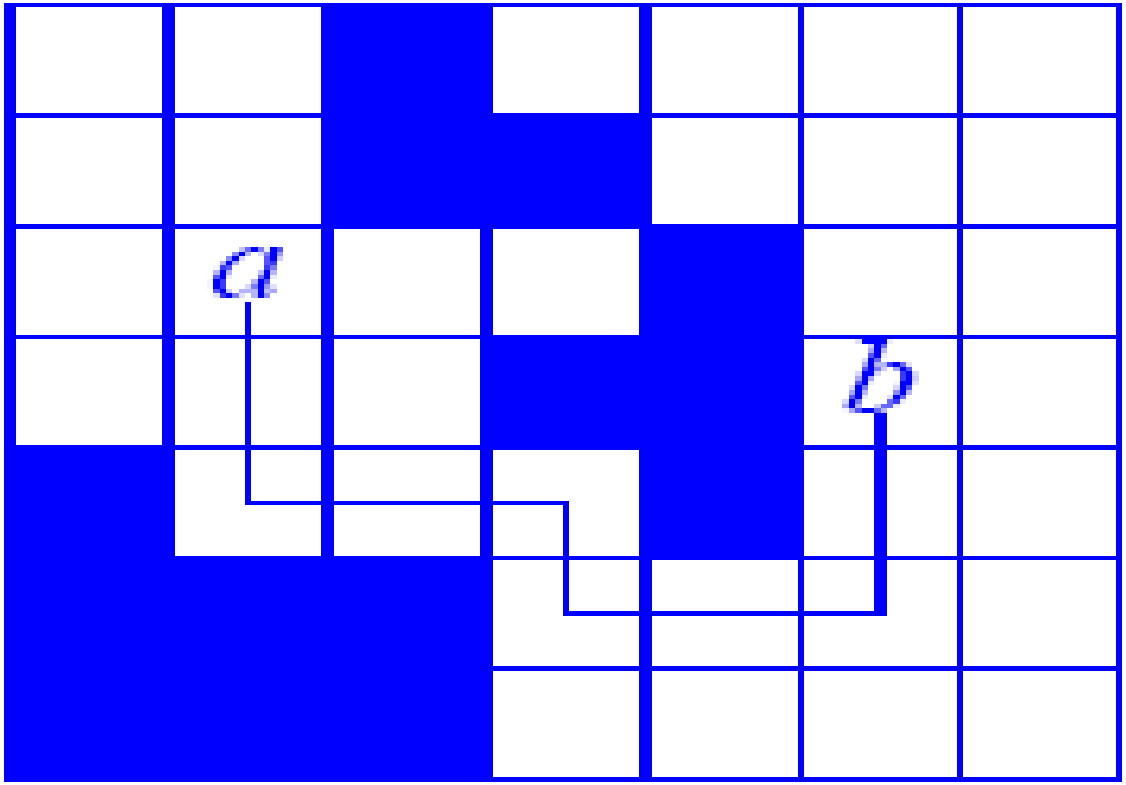
寻找路径设计思路-2/3

2. 构造最短路径

- 从b开始，首先移动到一个比b的编号小1的相邻位置上。一定存在这样的相邻位置，因为任一个方格上的标号与它相邻方格上的标号都至少相差1。
- 接下来，从当前位置开始，继续移动到比当前标号小1的相邻位置上。
- 重复这个过程，直至到达a为止。

寻找路径设计思路-3/3

2.构造最短路径



数据及数据结构定义

- 位置表示:
  - class position{

    private:

    int row; //位置的行坐标

    int col;//位置的列坐标}
- 网格表示：使用全局变量grid, m
  - int **grid,size;//grid表示网格矩阵,size:网格行数和列数
  - 二维数组grid中的元素
    - 0 ——空白的位置
    - 1 ——被阻塞的位置
  - 整个网格被包围在一堵由1构成的“墙”中。

数据及数据结构定义

- position offset[4];//当前位置here与移动方向i(=0,1,2,3)上的相邻位置之间的坐标增量
  - offset[i].row：row坐标的增量
  - offset[i].col：col坐标的增量
- arrayQueue<position> q;//队列q用来跟踪这样的方格：该方格本身已被编号，而它的相邻位置尚未被编号。
- position start,finish;//电路的起点和终点
- int pathLength;//路径长度
- position *path;//布线路径

寻找布线路径算法-1/3

```
bool findPath()
{ //寻找从start到finish的路径
  // 如果成功，则返回true，否则返回false
  if(start和finish相同)
        {路径长度为0; return true;}
  设置一堵“围墙”，把网格包围起来。
  数组 offsets初始化;
// 对跟踪当前位置的变量进行初始化
        position here, nbr;
        here.row=start.row;
        here.col=start.col;
  在起始位置上标记2;//因为数组中采用0和1来表示空白位置和封锁位置
```

寻找布线路径算法-2/3

```
// 标记网格
do {
    //标记当前位置here的相邻位置
    for (int i=0; i<相邻位置数; i++)
      {选择here的下一个的相邻位置nbr;
       if (nbr是空白位置)
                  {标记nbr的编号为here的编号+1;
                   if(nbr是finish)    break;
                   将位置nbr入队列q;}
      }
    if(nbr是finish)    break;
    if (队列q为空) return false;
    here←从队列中取出一个元素(相邻位置未被标记的位置)
}
```

寻找布线路径算法-3/3

```
//构造路径
路径长度pathLength=finish的编号-2;
path=new position[pathLength];
 //当前位置回溯至finish
here=finish;
for(int j=pathLength-1;j>=0;j--){
        path[j]=here;
        寻找比here编号小1的位置nbr
        //移动到前一个位置
        here=nbr;
       }

return true;
}
```