数据结构与算法（Python版）

递归的应用：探索迷宫

陈斌 北京大学 gischen@pku.edu.cn

探索迷宫：古希腊的迷宫

❖ 古希腊克里特岛米诺斯王

牛头人身怪物米诺陶洛斯

童男童女献祭，雅典王子忒修斯

公主，利剑，线团

老国王投海……爱琴海

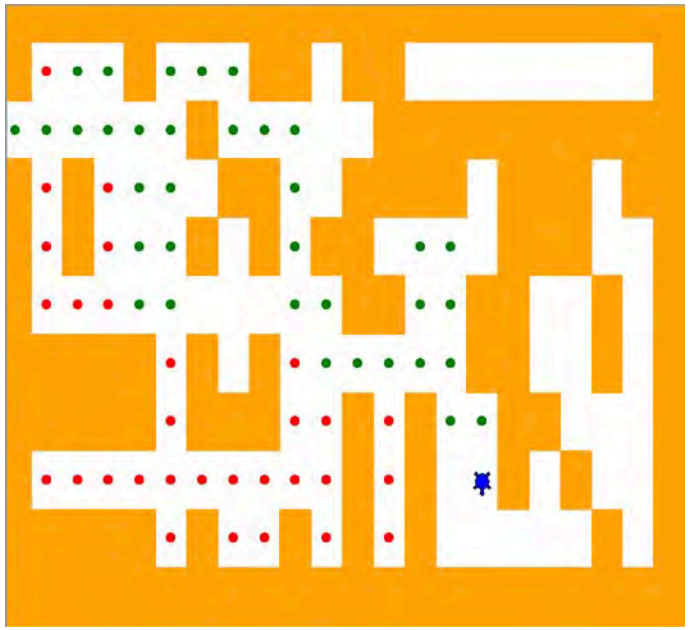
探索迷宫：圆明园的黄花阵

- 位于圆明园西洋楼景区

探索迷宫

- 将海龟放在迷宫中间，如何能找到出口
- 首先，我们将整个迷宫的空间（矩形）分为行列整齐的方格，区分出墙壁和通道。

  给每个方格具有行列位置，并赋予“墙壁”、“通道”的属性



迷宫的数据结构

❖ 考虑用矩阵方式来实现迷宫数据结构

采用“数据项为字符列表的列表”这种两级列表的方式来保存方格内容

采用不同字符来分别代表“墙壁+”、“通道 ”、“海龟投放点S”

从一个文本文件

逐行读入迷宫数据

maze2.tx

```
1  ++++++++++++++++++++++
2  +    +    ++ ++          +
3  +       +       ++++++++++
4  + +      ++   ++++ +++  ++
5  + +    + + ++     +++    +
6  +              ++   ++   + +
7  +++++  + +         ++   + +
8  +++++  +++   + +   ++     +
9  +              + + S+  +   +
10 +++++  +   + + +        + +
11 ++++++++++++++++++++++
```

迷宫的数据结构：Maze Class


```
class Maze:
    def __init__(self,mazeFileName):
        rowsInMaze = 0
        columnsInMaze = 0
        self.mazelist = []
        mazeFile = open(mazeFileName,'r')
        rowsInMaze = 0
        for line in mazeFile:
            rowList = []
            col = 0
            for ch in line[:-1]:
                rowList.append(ch)
                if ch == 'S':
                    self.startRow = rowsInMaze
                    self.startCol = col
                col = col + 1
            rowsInMaze = rowsInMaze + 1
            self.mazelist.append(rowList)
            columnsInMaze = len(rowList)
```

保存矩阵

迷宫的数据结构：Maze Class

❖读入数据文件成功后

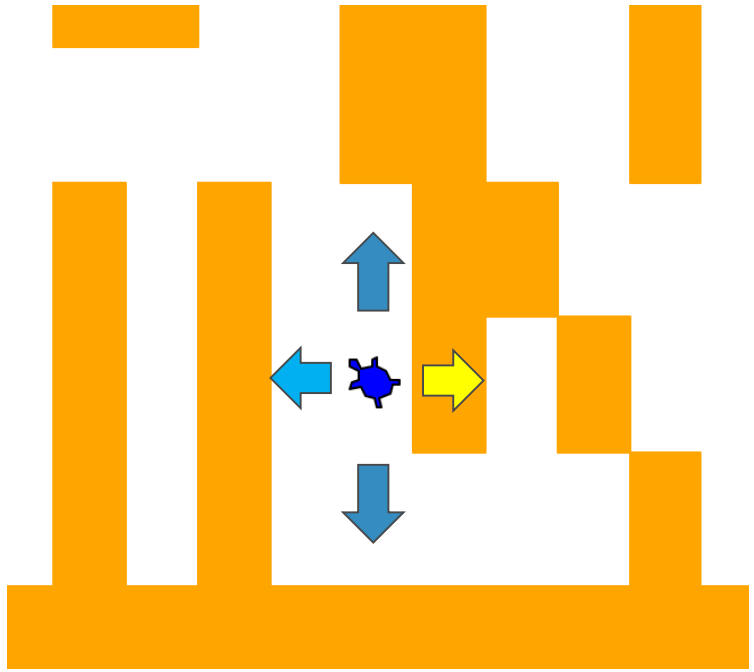
mazelist如下图示意

mazelist[row][col]=='+'

```
[ ['+','+','+','+',...,'+','+','+','+','+','+','+'],
  ['+',' ',' ',' ',...,' ',' ',' ','+',' ',' ',' '],
  ['+',' ','+',' ',...,'+','+',' ','+',' ','+','+'],
  ['+',' ','+',' ',...,' ',' ',' ','+',' ','+','+'],
  ['+','+','+',' ',...,'+','+',' ','+',' ',' ','+'],
  ['+',' ',' ',' ',...,'+','+',' ',' ',' ',' ','+'],
  ['+','+','+','+',...,'+','+','+','+','+',' ','+'],
  ['+',' ',' ',' ',...,'+','+',' ',' ','+',' ','+'],
  ['+',' ','+','+',...,' ',' ','+',' ',' ',' ','+'],
  ['+',' ',' ',' ',...,' ',' ','+',' ','+','+','+'],
  ['+','+','+','+',...,'+','+','+',' ','+','+','+']]
```

探索迷宫：算法思路

- 确定了迷宫数据结构之后，我们知道，对于海龟来说，其身处某个方格之中
  - 它所能移动的方向，必须是向着通道的方向
  - 如果某个方向是墙壁方格，就要换一个方向移动



探索迷宫：算法思路

- **这样，探索迷宫的递归算法思路如下：**

  将海龟从原位置向北移动一步，以新位置递归调用探索迷宫寻找出口；

  如果上面的步骤找不到出口，那么将海龟从原位置向南移动一步，以新位置递归调用探索迷宫；

  如果向南还找不到出口，那么将海龟从原位置向西移动一步，以新位置递归调用探索迷宫；

  如果向西还找不到出口，那么将海龟从原位置向东移动一步，以新位置递归调用探索迷宫；

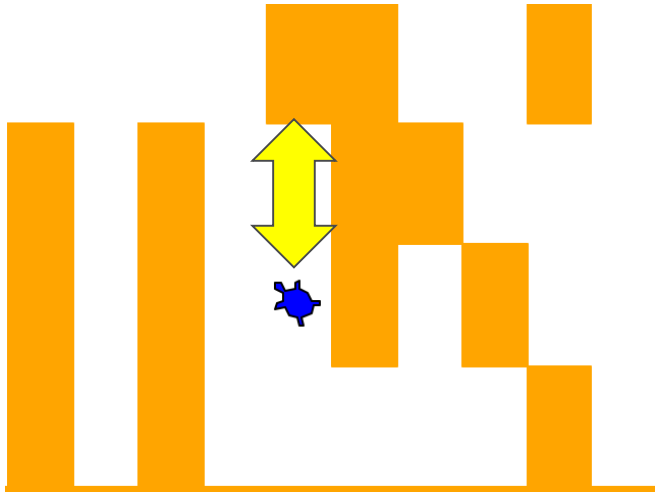
  如果上面四个方向都找不到出口，那么这个迷宫没有出口！

探索迷宫：算法思路

- **思路看起来很完美，但有些细节至关重要**

  如果我们向某个方向（如北）移动了海龟，那么如果新位置的北正好是一堵墙壁，那么在新位置上的递归调用就会让海龟向南尝试

  可是新位置的南边一格，正好就是递归调用之前的原位置，这样就陷入了无限递归的死循环之中

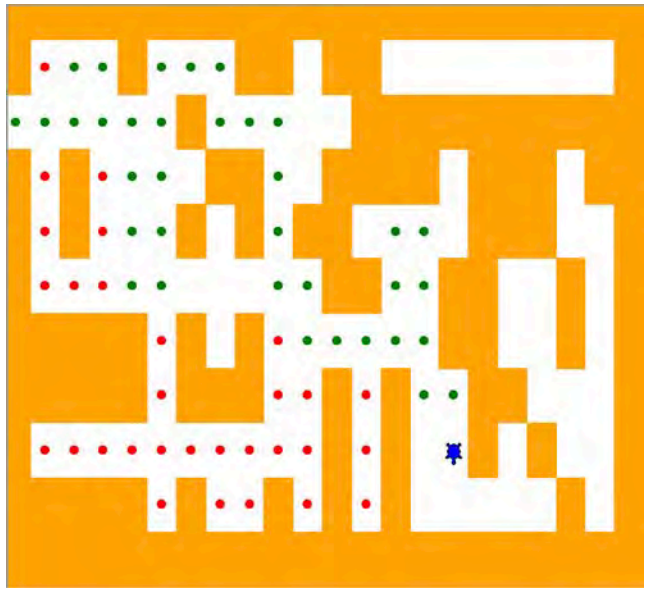


探索迷宫：算法思路

- 所以需要有个机制记录海龟所走过的路径

  沿途洒“面包屑”，一旦前进方向发现“面包屑”，就不能再踩上去，而必须换下一个方向尝试

  对于递归调用来说，就是某方向的方格上发现“面包屑”，就立即从递归调用返回上一级。



探索迷宫：算法思路

- 递归调用的“**基本结束条件**”归纳如下：

  海龟碰到“墙壁”方格，递归调用结束，返回失败；

  海龟碰到“面包屑”方格，表示此方格已访问过，递归调用结束，返回失败；

  海龟碰到“出口”方格，即“位于边缘的通道”方格，递归调用结束，返回成功！

  海龟在四个方向上探索都失败，递归调用结束，返回失败

探索迷宫：辅助的动画过程

- 为了让海龟在迷宫图里跑起来，我们给迷宫数据结构Maze Class添加一些成员和方法

  t：一个作图的海龟，设置其shape为海龟的样子（缺省是一个箭头）

  drawMaze()：绘制出迷宫的图形，墙壁用实心方格绘制

  updatePosition(row, col, val)：更新海龟的位置，并做标注

  isExit(row, col)：判断是否“出口”

```
96  def searchFrom(maze, startRow, startColumn):
97      #  1. 碰到墙壁，返回失败
98      maze.updatePosition(startRow, startColumn)
99      if maze[startRow][startColumn] == OBSTACLE :
100         return False
101
102     #  2. 碰到面包屑，或者死胡同，返回失败
103     if maze[startRow][startColumn] == TRIED or \
104        maze[startRow][startColumn] == DEAD_END:
105         return False
106
107     #  3. 碰到了出口，返回成功!
108     if maze.isExit(startRow,startColumn):
109         maze.updatePosition(startRow, startColumn, PART_OF_PATH)
110         return True
111
112     #  4. 洒一下面包屑，继续探索
113     maze.updatePosition(startRow, startColumn, TRIED)
114
115     # 向北南西东4个方向依次探索，or操作符具有短路效应
116     found = searchFrom(maze, startRow-1, startColumn) or \
117             searchFrom(maze, startRow+1, startColumn) or \
118             searchFrom(maze, startRow, startColumn-1) or \
119             searchFrom(maze, startRow, startColumn+1)
120
121     # 如果探索成功，标记当前点，失败则标记为“死胡同”
122     if found:
123         maze.updatePosition(startRow, startColumn, PART_OF_PATH)
124     else:
125         maze.updatePosition(startRow, startColumn, DEAD_END)
126     return found
```