

# 《人工智能》课程系列

Maze 实验平台的设计与实现\*

武汉纺织大学数学与计算机学院

杜小勤

2018/09/25

## Contents

1	Array 类	2
2	Array2D 类	4
3	Queue 类	7
4	Maze 类	8
5	MazeDraw 类	11
6	深度优先搜索	13
7	宽度优先搜索	16
8	参考文献	20

---

\*本系列文档属于讲义性质，仅用于学习目的。Last updated on: October 4, 2018。

# 1 Array 类

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Mon Sep 9 19:25:08 2018
4
5  @author: duxiaoqin
6
7  Functions:
8      (1) Array class;
9  """
10
11 import random
12 import ctypes
13
14 class Array:
15     def __init__(self, size):
16         assert size > 0, 'Array size must be > 0'
17         self.size = size
18         PyArrayType = ctypes.py_object * size
19         self.elements = PyArrayType()
20         self.clear(None)
21
22     def clone(self):
23         newa = Array(len(self))
24         for index in range(len(self)):
25             newa[index] = self[index]
26         return newa
27
28     def print(self):
29         for index in range(len(self)):
```

```
30         print(self.elements[index], end=' ')
31
32     def __len__(self):
33         return self.size
34
35     def __getitem__(self, index):
36         assert index >= 0 and index < len(self), \
37             'Array subscript out of range'
38         return self.elements[index]
39
40     def __setitem__(self, index, value):
41         assert index >= 0 and index < len(self), \
42             'Array subscript out of range'
43         self.elements[index] = value
44
45     def clear(self, value):
46         for i in range(len(self)):
47             self.elements[i] = value
48
49     def __iter__(self):
50         return ArrayIterator(self.elements)
51
52     class ArrayIterator:
53         def __init__(self, theArray):
54             self.arrayRef = theArray
55             self.curNdx = 0
56
57         def __iter__(self):
58             return self
59
60         def __next__(self):
```

```
61         if self.curNdx < len(self.arrayRef):
62             entry = self.arrayRef[self.curNdx]
63             self.curNdx = self.curNdx + 1
64             return entry
65         else:
66             raise StopIteration
67
68 def main():
69     a = Array(10)
70     for i in range(len(a)):
71         a[i] = random.random()
72     a.print()
73
74 if __name__ == '__main__':
75     main()
```

## 2 Array2D 类

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Mon Sep 9 20:25:08 2018
4
5  @author: duxiaoqin
6
7  Functions:
8      (1) Array2D class;
9  """
10
11 import random
12 from myarray import Array
13
```

```
14 class Array2D:
15     def __init__(self, numRows, numCols):
16         self.theRows = Array(numRows)
17
18         for i in range(numRows):
19             self.theRows[i] = Array(numCols)
20
21     def clone(self):
22         newa2d = Array2D(self.numRows(), self.numCols())
23         for row in range(self.numRows()):
24             for col in range(self.numCols()):
25                 newa2d.theRows[row][col] = self.theRows[row][col]
26         return newa2d
27
28     def print(self):
29         for i in range(self.numRows()):
30             self.theRows[i].print()
31             print()
32
33     def numRows(self):
34         return len(self.theRows)
35
36     def numCols(self):
37         return len(self.theRows[0])
38
39     def clear(self, value):
40         for row in range(self.numRows()):
41             self.theRows[row].clear(value)
42
43     def __getitem__(self, ndxTuple):
44         assert len(ndxTuple) == 2, 'Invalid number of array subscripts.'
```

```
45         row = ndxTuple[0]
46         col = ndxTuple[1]
47         assert row >= 0 and row < self.numRows() and \
48             col >= 0 and col < self.numCols(), \
49             "Array subscript out of range."
50         the1dArray = self.theRows[row]
51         return the1dArray[col]
52
53     def __setitem__(self, ndxTuple, value):
54         assert len(ndxTuple) == 2, 'Invalid number of array subscripts.'
55         row = ndxTuple[0]
56         col = ndxTuple[1]
57         assert row >= 0 and row < self.numRows() and \
58             col >= 0 and col < self.numCols(), \
59             'Array subscript out of range.'
60         the1dArray = self.theRows[row]
61         the1dArray[col] = value
62
63     def main():
64         a = Array2D(10, 5)
65         for r in range(a.numRows()):
66             for c in range(a.numCols()):
67                 a[r, c] = random.random()
68
69         a.print()
70
71     if __name__ == '__main__':
72         main()
```

### 3 Queue 类

```
1 class Queue:
2     def __init__(self):
3         self.items = []
4
5     def enqueue(self, item):
6         self.items.insert(0,item)
7
8     def dequeue(self):
9         return self.items.pop()
10
11    def is_empty(self):
12        return self.items == []
13
14    def size(self):
15        return len(self.items)
16
17 def main():
18     q = Queue()
19     q.enqueue(1)
20     q.enqueue(2)
21     q.enqueue(3)
22     q.enqueue(4)
23     q.enqueue(5)
24     print(q.size())
25     while not q.is_empty():
26         print(q.dequeue())
27     print(q.size())
28
29 if __name__ == '__main__':
```

30        `main()`

## 4 Maze 类

Maze 类实现迷宫的单元管理。迷宫单元分为以下几类：空白单元、障碍物、已访问。下面给出它的 ADT 定义：

- `Maze(width, height)`

以指定宽度与高度，创建一个迷宫对象，按一定的比例随机初始化障碍物单元；

- `getValue(row, col)`

获取指定位置 (row, col) 单元的值；

- `setValue(row, col, value)`

设置指定位置 (row, col) 单元的值为 value；

- `getAllMoves(row, col)`

获取指定单元处所有可通行的直接邻居；

下面给出 Maze 类的实现：

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Sep 25 15:46:53 2018
4
5  @author: duxiaoqin
6  Functions:
7      (1)Maze class
8  """
9
10 from random import *
11 from myarray2d import Array2D
12
```



```
13 class Maze:
14     EMPTY = 0
15     OBSTACLE = -1
16     OCCUPIED = 1
17     def __init__(self, height, width):
18         seed()
19         self.maze = Array2D(height, width)
20         self.start = (0, 0)
21         self.goal = (height-1, width-1)
22         self.maze.clear(Maze.EMPTY)
23         for count in range(int(height*width*0.1)):
24             row = int(random()*100 % height)
25             col = int(random()*100 % width)
26             if (row, col) == self.start or (row, col) == self.goal:
27                 continue
28             self.maze[row, col] = Maze.OBSTACLE
29
30     def __getitem__(self, ndxTuple):
31         return self.maze.__getitem__(ndxTuple)
32
33     def __setitem__(self, ndxTuple, value):
34         self.maze.__setitem__(ndxTuple, value)
35
36     def getAllMoves(self, row, col):
37         width = self.numCols()
38         height = self.numRows()
39         moves = []
40         offsets = [(0, -1), (-1, 0), (1, 0), (0, 1)]
41         for x, y in offsets:
42             x = col + x
43             y = row + y
```

```
44         if x < 0 or x > width-1 or \
45             y < 0 or y > height-1:
46             continue
47         if self.maze[y, x] != Maze.OBSTACLE:
48             moves.append((y, x))
49     return moves
50
51     def numRows(self):
52         return self.maze.numRows()
53
54     def numCols(self):
55         return self.maze.numCols()
56
57     def print(self):
58         rows = self.numRows()
59         cols = self.numCols()
60         for row in range(rows):
61             for col in range(cols):
62                 if self.maze[row, col] == Maze.EMPTY:
63                     print('_', end=' ')
64                 elif self.maze[row, col] == Maze.OBSTACLE:
65                     print('|', end=' ')
66                 else:
67                     print('0', end=' ')
68             print()
69
70     def main():
71         maze = Maze(20, 20)
72         maze.print()
73         print()
74         print(maze.getAllMoves(3, 3))
```

```
75
76 if __name__ == '__main__':
77     main()
```

## 5 MazeDraw 类

MazeDraw 类实现迷宫的绘制功能。下面是 MazeDraw 类的 ADT 定义：

- MazeDraw(gui, maze)

创建一个 MazeDraw 对象，参数 gui 为图形接口，参数 maze 为迷宫实例；

- draw()

绘制迷宫，迷宫单元有 2 种状态：空白、障碍物；

下面给出 MazeDraw 类的 ADT 实现：

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Sep 25 17:02:19 2018
4
5  @author: duxiaoqin
6  Functions:
7      (1)MazeDraw class
8  """
9
10 from graphics import *
11 from maze import *
12
13 class MazeDraw:
14     def __init__(self, win, maze):
15         self.width = maze.numCols()
16         self.height = maze.numRows()
17         self.win = win
```

```
18         self.win.setCoords(0.0, 0.0, self.width + 2, self.height + 2)
19
20         self.rect_points = []
21         for row in range(self.height):
22             for col in range(self.width):
23                 point1 = Point(col+1, self.height-row)
24                 point2 = Point(col+1+1, self.height-row+1)
25                 if maze[row, col] == Maze.EMPTY:
26                     color = 'green'
27                 else:
28                     color = 'blue'
29                 self.rect_points.append((point1, point2, color))
30         self.rectangles = []
31         for p1, p2, color in self.rect_points:
32             rect = Rectangle(p1, p2)
33             rect.setFill(color)
34             self.rectangles.append(rect)
35
36         def draw(self):
37             for rect in self.rectangles:
38                 rect.undraw()
39             for rect in self.rectangles:
40                 rect.draw(self.win)
41             update(30)
42
43     def main():
44         win = GraphWin('MazeDraw', 600, 600, autoflush=False)
45         maze = Maze(20, 20)
46         maze.print()
47         mazedraw = MazeDraw(win, maze)
48
```

```
49     while win.checkKey() != 'Escape':
50         mazedraw.draw()
51     win.close()
52
53 if __name__ == '__main__':
54     main()
```

## 6 深度优先搜索

下面给出深度优先搜索 (递归) 的算法:

Input:

A maze and a start position  $v$

came\_from is a DICT, initialized with {}

Output:

return: GOAL or None

came\_from is changed

```
def DFS(maze, v, came_from):
```

```
    if came_from == {}:
```

```
        came_from[v] = None
```

```
    if v is a goal:
```

```
        return v
```

```
    label v as discovered
```

```
    for all possible moves from v to w in maze.getAllMoves(v):
```

```
        if vertex w is not labeled as discovered:
```

```
            came_from[w] = v
```

```
            result = DFS(maze, w, came_from)
```

```
            if result is a goal:
```

```
                return result
```

```
    return None
```

相应的迷宫搜索程序如下:

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Oct  2 20:28:21 2018
4
5  @author: duxiaoqin
6  Functions:
7      (1)DFS for Maze;
8  """
9
10 from graphics import *
11 from myarray2d import Array2D
12 from maze import Maze
13 from mazedraw import MazeDraw
14
15 def DFS(maze, v, goal, came_from):
16     if came_from == {}:
17         came_from[v] = None
18
19     if v == goal:
20         return v
21
22     maze[v[0], v[1]] = Maze.OCCUPIED
23     for w in maze.getAllMoves(v[0], v[1]):
24         if maze[w[0], w[1]] == Maze.EMPTY:
25             came_from[w] = v
26             result = DFS(maze, w, goal, came_from)
27             if result == goal:
28                 return result
29     return None
30
31 def drawPath(win, maze, came_from):
```

```
32     offsets = [(0, -1), (-1, 0), (1, 0), (0, 1)]
33     add = lambda x,y:x+y
34     current = maze.goal
35     while current != maze.start:
36         next = came_from[current]
37         for offset in offsets:
38             next_one = tuple(map(add, current, offset))
39             if next_one == next:
40                 line = Line(Point(next[1]+1+0.5, \
41                                     maze.numRows()-next[0]+1-0.5), \
42                               Point(current[1]+1+0.5, \
43                                     maze.numRows()-current[0]+1-0.5))
44                 line.setOutline('white')
45                 line.setArrow('last')
46                 line.draw(win)
47             current = next
48
49 def main():
50     win = GraphWin('DFS for Maze', 600, 600, autoflush=False)
51     maze = Maze(20, 20)
52     mazedraw = MazeDraw(win, maze)
53     mazedraw.draw()
54     #visited = Array2D(maze.numRows(), maze.numCols())
55     #visited.clear(False)
56     came_from = {}
57     found = DFS(maze, maze.start, maze.goal, came_from)
58     text = Text(Point(11, 0.5), '')
59     if found == maze.goal:
60         text.setText('Goal found!')
61         drawPath(win, maze, came_from)
62     else:
```

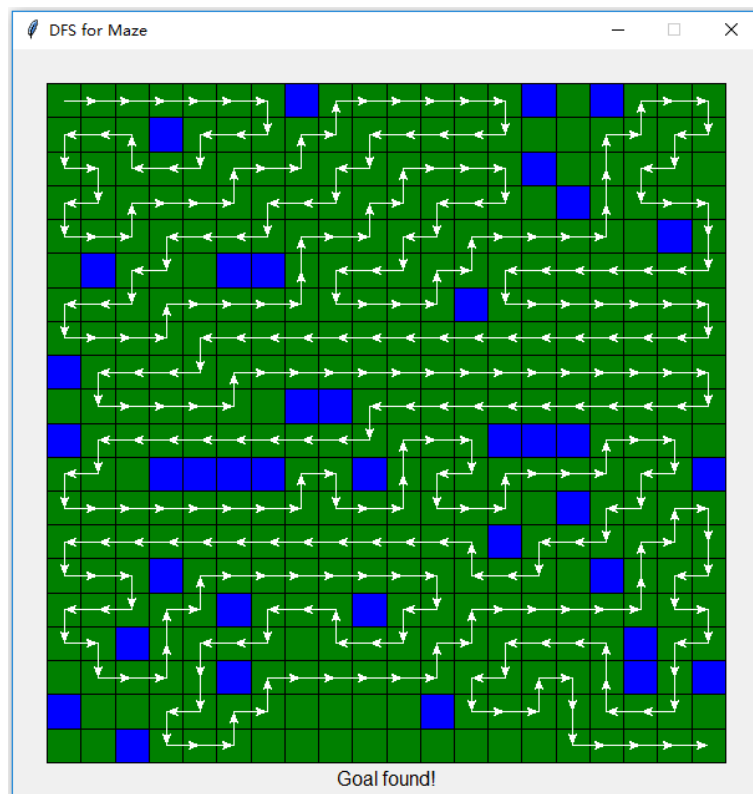


图 6-1: 深度优先搜索 (迷宫) 的一个结果

```
63         text.setText('Goal not found!')
64     text.draw(win)
65
66     while win.checkKey() != 'Escape':
67         pass
68     win.close()
69
70 if __name__ == '__main__':
71     main()
```

运行的一个结果如图6-1所示。

## 7 宽度优先搜索

下面先给出宽度优先搜索 (队列) 的算法:



Input:

A maze and a start position

came\_from is a DICT, initialized with {}

Output:

return: GOAL or None

came\_from is changed

def BFS(maze, v, came\_from):

    frontier = Queue()

    frontier.enqueue(v)

    came\_from[v] = None

    while not frontier.is\_empty():

        v = frontier.dequeue()

        if v is not labeled as discovered:

            if v is a goal:

                return v

            else:

                label v as discovered

                for all possible moves from v to w in maze.getAllMoves(v):

                    if w is not labeled as discovered:

                        frontier.enqueue(w)

                        came\_from[w] = v

    return None

相应的迷宫搜索程序如下:

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Wed Oct 3 21:41:31 2018
4
5  @author: duxiaoqin
6  Functions:
7      (1)BFS for Maze;
8  """

```

```
9
10 from myqueue import Queue
11 from graphics import *
12 from myarray2d import Array2D
13 from maze import Maze
14 from mazedraw import MazeDraw
15
16 def BFS(maze, v, goal, came_from):
17     frontier = Queue()
18     frontier.enqueue(v)
19     came_from[v] = None
20     while not frontier.is_empty():
21         v = frontier.dequeue()
22         if maze[v[0], v[1]] == Maze.EMPTY:
23             if v == goal:
24                 return v
25             else:
26                 maze[v[0], v[1]] = Maze.OCCUPIED
27                 for w in maze.getAllMoves(v[0], v[1]):
28                     if maze[w[0], w[1]] == Maze.EMPTY:
29                         frontier.enqueue(w)
30                         came_from[w] = v
31     return None
32
33 def drawPath(win, maze, came_from):
34     offsets = [(0, -1), (-1, 0), (1, 0), (0, 1)]
35     add = lambda x,y:x+y
36     current = maze.goal
37     while current != maze.start:
38         next = came_from[current]
39         for offset in offsets:
```

```
40         next_one = tuple(map(add, current, offset))
41         if next_one == next:
42             line = Line(Point(next[1]+1+0.5, \
43                             maze.numRows()-next[0]+1-0.5), \
44                             Point(current[1]+1+0.5, \
45                             maze.numRows()-current[0]+1-0.5))
46             line.setOutline('white')
47             line.setArrow('last')
48             line.draw(win)
49         current = next
50
51 def main():
52     win = GraphWin('BFS for Maze', 600, 600, autoflush=False)
53     maze = Maze(20, 20)
54     mazedraw = MazeDraw(win, maze)
55     mazedraw.draw()
56     came_from = {}
57     found = BFS(maze, maze.start, maze.goal, came_from)
58     print(found)
59     text = Text(Point(11, 0.5), '')
60     if found == maze.goal:
61         text.setText('Goal found!')
62         drawPath(win, maze, came_from)
63     else:
64         text.setText('Goal not found!')
65     text.draw(win)
66
67     while win.checkKey() != 'Escape':
68         pass
69     win.close()
70
```

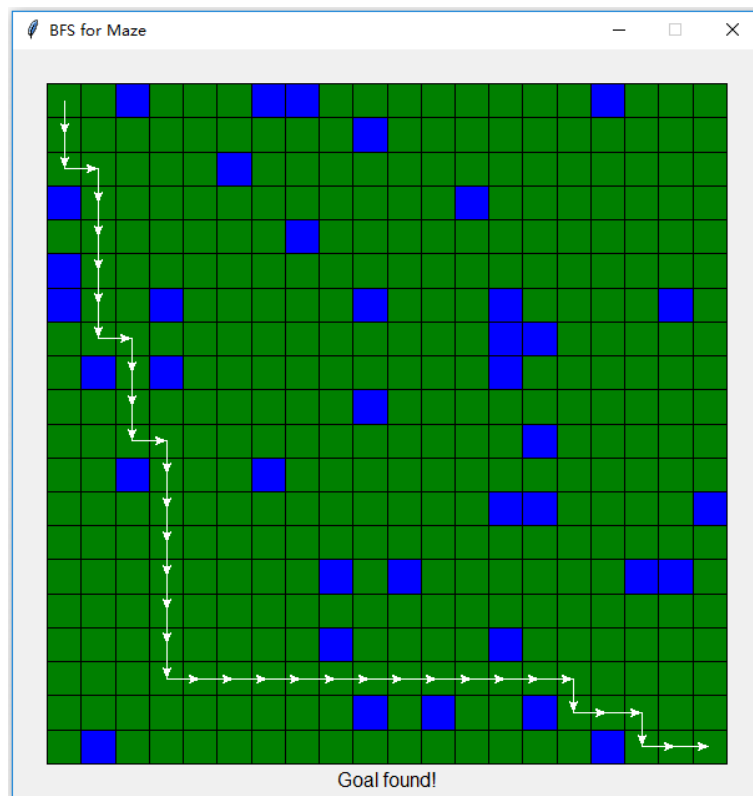


图 7-2: 宽度优先搜索 (迷宫) 的一个结果

```
71 if __name__ == '__main__':  
72     main()
```

运行的一个结果如图7-2所示。

## 8 参考文献

1. 杜小勤。《人工智能》课程系列, Part I: Python 程序设计基础, 2018/06/13。
2. 杜小勤。《人工智能》课程系列, Part II: Python 算法基础, 2018/07/31。