《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

[*]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

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Sep 9 19:25:08 2018

@author: duxiaoqin

Functions:
    (1) Array class;
"""

import random
import ctypes

class Array:
    def __init__(self, size):
        assert size > 0, 'Array size must be > 0'
        self.size = size
        PyArrayType = ctypes.py_object * size
        self.elements = PyArrayType()
        self.clear(None)

    def clone(self):
        newa = Array(len(self))
        for index in range(len(self)):
            newa[index] = self[index]
        return newa

    def print(self):
        for index in range(len(self)):
            print(self.elements[index], end=' ')

    def __len__(self):
        return self.size

    def __getitem__(self, index):
        assert index >= 0 and index < len(self), \
               'Array subscript out of range'
        return self.elements[index]

    def __setitem__(self, index, value):
        assert index >= 0 and index < len(self), \
               'Array subscript out of range'
        self.elements[index] = value

    def clear(self, value):
        for i in range(len(self)):
            self.elements[i] = value

    def __iter__(self):
        return ArrayIterator(self.elements)

class ArrayIterator:
    def __init__(self, theArray):
        self.arrayRef = theArray
        self.curNdx = 0

    def __iter__(self):
        return self

    def __next__(self):
        if self.curNdx < len(self.arrayRef):
            entry = self.arrayRef[self.curNdx]
            self.curNdx = self.curNdx + 1
            return entry
        else:
            raise StopIteration

def main():
    a = Array(10)
    for i in range(len(a)):
        a[i] = random.random()
    a.print()

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

2 Array2D 类

```
# -*- coding: utf-8 -*-
"""
Created on Mon Sep 9 20:25:08 2018

@author: duxiaoqin

Functions:
    (1) Array2D class;
"""

import random
from myarray import Array

```

```python
class Array2D:
    def __init__(self, numRows, numCols):
        self.theRows = Array(numRows)

        for i in range(numRows):
            self.theRows[i] = Array(numCols)

    def clone(self):
        newa2d = Array2D(self.numRows(), self.numCols())
        for row in range(self.numRows()):
            for col in range(self.numCols()):
                newa2d.theRows[row][col] = self.theRows[row][col]
        return newa2d

    def print(self):
        for i in range(self.numRows()):
            self.theRows[i].print()
            print()

    def numRows(self):
        return len(self.theRows)

    def numCols(self):
        return len(self.theRows[0])

    def clear(self, value):
        for row in range(self.numRows()):
            self.theRows[row].clear(value)

    def __getitem__(self, ndxTuple):
        assert len(ndxTuple) == 2, 'Invalid number of array subscripts.'
        row = ndxTuple[0]
        col = ndxTuple[1]
        assert row >= 0 and row < self.numRows() and \
               col >= 0 and col < self.numCols(), \
               "Array subscript out of range."
        the1dArray = self.theRows[row]
        return the1dArray[col]

    def __setitem__(self, ndxTuple, value):
         assert len(ndxTuple) == 2, 'Invalid number of array subscripts.'
         row = ndxTuple[0]
         col = ndxTuple[1]
         assert row >= 0 and row < self.numRows() and \
                col >= 0 and col < self.numCols(), \
                'Array subscript out of range.'
         the1dArray = self.theRows[row]
         the1dArray[col] = value

def main():
    a = Array2D(10, 5)
    for r in range(a.numRows()):
        for c in range(a.numCols()):
            a[r, c] = random.random()

    a.print()

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

3 Queue 类

```python
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.insert(0,item)

    def dequeue(self):
        return self.items.pop()

    def is_empty(self):
        return self.items == []

    def size(self):
        return len(self.items)

def main():
    q = Queue()
    q.enqueue(1)
    q.enqueue(2)
    q.enqueue(3)
    q.enqueue(4)
    q.enqueue(5)
    print(q.size())
    while not q.is_empty():
        print(q.dequeue())
    print(q.size())

if __name__ == '__main__':
    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：

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 25 15:46:53 2018

@author: duxiaoqin
Functions:
    (1)Maze class
"""

from random import *
from myarray2d import Array2D

```

```python
class Maze:
    EMPTY = 0
    OBSTACLE = -1
    OCCUPIED = 1
    def __init__(self, height, width):
        seed()
        self.maze = Array2D(height, width)
        self.start = (0, 0)
        self.goal = (height-1, width-1)
        self.maze.clear(Maze.EMPTY)
        for count in range(int(height*width*0.1)):
            row = int(random()*100 % height)
            col = int(random()*100 % width)
            if (row, col) == self.start or (row, col) == self.goal:
                continue
            self.maze[row, col] = Maze.OBSTACLE

    def __getitem__(self, ndxTuple):
        return self.maze.__getitem__(ndxTuple)

    def __setitem__(self, ndxTuple, value):
        self.maze.__setitem__(ndxTuple, value)

    def getAllMoves(self, row, col):
        width = self.numCols()
        height = self.numRows()
        moves = []
        offsets = [(0, -1), (-1, 0), (1, 0), (0, 1)]
        for x, y in offsets:
            x = col + x
            y = row + y
```

```python
                if x < 0 or x > width-1 or \
                   y < 0 or y > height-1:
                    continue
                if self.maze[y, x] != Maze.OBSTACLE:
                    moves.append((y, x))
        return moves

    def numRows(self):
        return self.maze.numRows()

    def numCols(self):
        return self.maze.numCols()

    def print(self):
        rows = self.numRows()
        cols = self.numCols()
        for row in range(rows):
            for col in range(cols):
                if self.maze[row, col] == Maze.EMPTY:
                    print('_', end=' ')
                elif self.maze[row, col] == Maze.OBSTACLE:
                    print('|', end=' ')
                else:
                    print('O', end=' ')
            print()

def main():
    maze = Maze(20, 20)
    maze.print()
    print()
    print(maze.getAllMoves(3, 3))
```

```python

if __name__ == '__main__':
    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：

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 25 17:02:19 2018

@author: duxiaoqin
Functions:
    (1)MazeDraw class
"""

from graphics import *
from maze import *

class MazeDraw:
    def __init__(self, win, maze):
        self.width = maze.numCols()
        self.height = maze.numRows()
        self.win = win
        self.win.setCoords(0.0, 0.0, self.width + 2, self.height + 2)

        self.rect_points = []
        for row in range(self.height):
            for col in range(self.width):
                point1 = Point(col+1, self.height-row)
                point2 = Point(col+1+1, self.height-row+1)
                if maze[row, col] == Maze.EMPTY:
                    color = 'green'
                else:
                    color = 'blue'
                self.rect_points.append((point1, point2, color))
        self.rectangles = []
        for p1, p2, color in self.rect_points:
            rect = Rectangle(p1, p2)
            rect.setFill(color)
            self.rectangles.append(rect)

    def draw(self):
        for rect in self.rectangles:
            rect.undraw()
        for rect in self.rectangles:
            rect.draw(self.win)
        update(30)

def main():
    win = GraphWin('MazeDraw', 600, 600, autoflush=False)
    maze = Maze(20, 20)
    maze.print()
    mazedraw = MazeDraw(win, maze)

```

```python
    while win.checkKey() != 'Escape':
        mazedraw.draw()
    win.close()

if __name__ == '__main__':
    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:

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Oct  2 20:28:21 2018

@author: duxiaoqin
Functions:
    (1)DFS for Maze;
"""

from graphics import *
from myarray2d import Array2D
from maze import Maze
from mazedraw import MazeDraw

def DFS(maze, v, goal, came_from):
    if came_from == {}:
        came_from[v] = None

    if v == goal:
        return v

    maze[v[0], v[1]] = Maze.OCCUPIED
    for w in maze.getAllMoves(v[0], v[1]):
        if maze[w[0], w[1]] == Maze.EMPTY:
            came_from[w] = v
            result = DFS(maze, w, goal, came_from)
            if result == goal:
                return result
    return None

def drawPath(win, maze, came_from):
    offsets = [(0, -1), (-1, 0), (1, 0), (0, 1)]
    add = lambda x,y:x+y
    current = maze.goal
    while current != maze.start:
        next = came_from[current]
        for offset in offsets:
            next_one = tuple(map(add, current, offset))
            if next_one == next:
                line = Line(Point(next[1]+1+0.5, \
                                  maze.numRows()-next[0]+1-0.5), \
                            Point(current[1]+1+0.5, \
                                  maze.numRows()-current[0]+1-0.5))
                line.setOutline('white')
                line.setArrow('last')
                line.draw(win)
        current = next

def main():
    win = GraphWin('DFS for Maze', 600, 600, autoflush=False)
    maze = Maze(20, 20)
    mazedraw = MazeDraw(win, maze)
    mazedraw.draw()
    #visited = Array2D(maze.numRows(), maze.numCols())
    #visited.clear(False)
    came_from = {}
    found = DFS(maze, maze.start, maze.goal, came_from)
    text = Text(Point(11, 0.5), '')
    if found == maze.goal:
        text.setText('Goal found!')
        drawPath(win, maze, came_from)
    else:
```

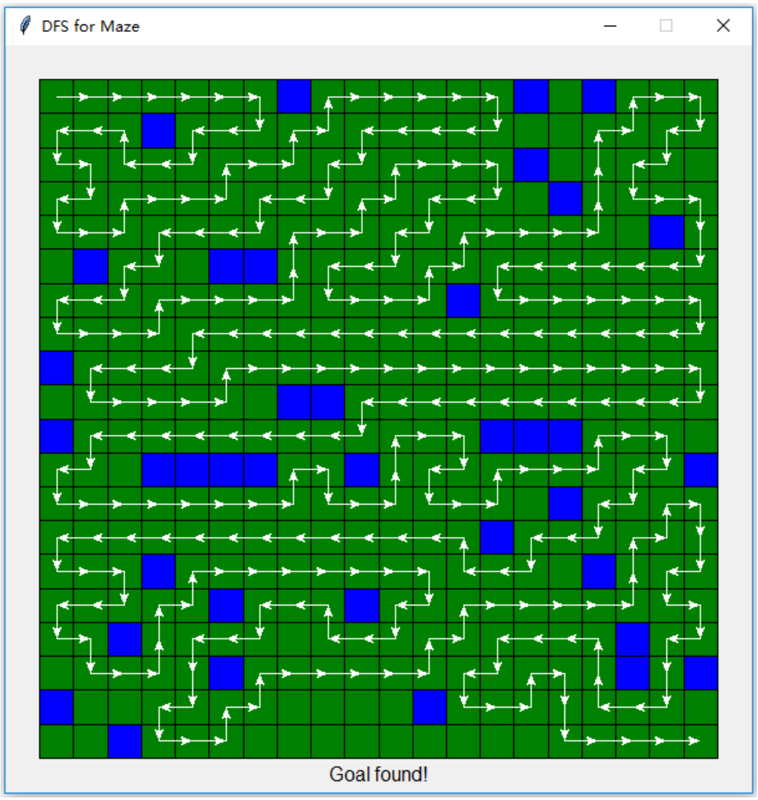

图 6-1: 深度优先搜索 (迷宫) 的一个结果

```
            text.setText('Goal not found!')
        text.draw(win)

        while win.checkKey() != 'Escape':
            pass
        win.close()

if __name__ == '__main__':
        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:

```python
# -*- coding: utf-8 -*-
"""
Created on Wed Oct  3 21:41:31 2018

@author: duxiaoqin
Functions:
    (1)BFS for Maze;
"""
```

```python

from myqueue import Queue
from graphics import *
from myarray2d import Array2D
from maze import Maze
from mazedraw import MazeDraw

def BFS(maze, v, goal, came_from):
    frontier = Queue()
    frontier.enqueue(v)
    came_from[v] = None
    while not frontier.is_empty():
        v = frontier.dequeue()
        if maze[v[0], v[1]] == Maze.EMPTY:
            if v == goal:
                return v
            else:
                maze[v[0], v[1]] = Maze.OCCUPIED
                for w in maze.getAllMoves(v[0], v[1]):
                    if maze[w[0], w[1]] == Maze.EMPTY:
                        frontier.enqueue(w)
                        came_from[w] = v
    return None

def drawPath(win, maze, came_from):
    offsets = [(0, -1), (-1, 0), (1, 0), (0, 1)]
    add = lambda x,y:x+y
    current = maze.goal
    while current != maze.start:
        next = came_from[current]
        for offset in offsets:
            next_one = tuple(map(add, current, offset))
            if next_one == next:
                line = Line(Point(next[1]+1+0.5, \
                                  maze.numRows()-next[0]+1-0.5), \
                            Point(current[1]+1+0.5, \
                                  maze.numRows()-current[0]+1-0.5))
                line.setOutline('white')
                line.setArrow('last')
                line.draw(win)
        current = next

def main():
    win = GraphWin('BFS for Maze', 600, 600, autoflush=False)
    maze = Maze(20, 20)
    mazedraw = MazeDraw(win, maze)
    mazedraw.draw()
    came_from = {}
    found = BFS(maze, maze.start, maze.goal, came_from)
    print(found)
    text = Text(Point(11, 0.5), '')
    if found == maze.goal:
        text.setText('Goal found!')
        drawPath(win, maze, came_from)
    else:
        text.setText('Goal not found!')
    text.draw(win)

    while win.checkKey() != 'Escape':
        pass
    win.close()

```

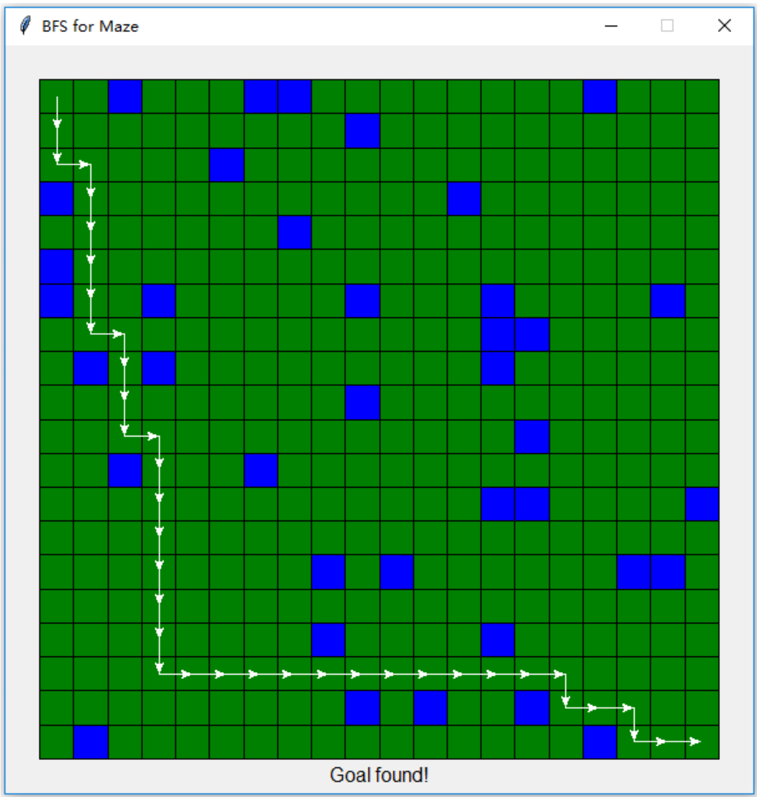

图 7-2: 宽度优先搜索 (迷宫) 的一个结果

```python
if __name__ == '__main__':
    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。