《人工智能》课程系列

8 数码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

2018/10/18

Contents

1 Array 类

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Sep 9 19:25:08 2018
4
```

*本系列文档属于讲义性质，仅用于学习目的。Last updated on: November 12, 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):
```

```
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 Puzzle8 类

Puzzle8 类实现 8 数码的管理。下面给出它的 ADT 定义：

- Puzzle8()

  创建一个 8 数码对象，随机初始化单元；

- clone()

  克隆 8 数码对象并返回；

- isGoal()

  判断是否到达目标状态；

- getCost()

  返回实际移动代价 (g 函数)；

- getHeuristics()

  返回启发值 (h 函数)；

- 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()

  获取所有可移动棋子的位置；

- move(row, col)

  将指定棋子移动到空格位置 (即与空格交换)，实际代价将递增 (+1)，启发值也会发生变化，需要重新计算；

Puzzle8 类的实现如下：

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 18 17:37:54 2018

@author: duxiaoqin
Functions:
    (1)Puzzle8 class;
"""

from random import *
```

```
11 from myarray2d import Array2D
12
13 class Puzzle8:
14
15     HEIGHT = 3
16     WIDTH = 3
17     ITEMS = [' ', '1', '2', '3', '4', '5', '6', '7', '8']
18     GOAL = Array2D(HEIGHT, WIDTH)
19     for row in range(HEIGHT):
20         for col in range(WIDTH):
21             GOAL[row, col] = ITEMS[row*WIDTH+col]
22
23     def __init__(self, clone=False):
24         self.__puzzle8 = Array2D(Puzzle8.HEIGHT, Puzzle8.WIDTH)
25         self.__cost_so_far = 0
26         self.__inversions = 0
27
28         if (clone == False):
29             self._genRandomItems()
30             self._calcInversions()
31             while self.__inversions % 2 != 0:
32                 self._genRandomItems()
33                 self._calcInversions()
34             self.__heuristics = self._calcHeuristics()
35
36     def _genRandomItems(self):
37         items = Puzzle8.ITEMS[:]
38         for row in range(Puzzle8.HEIGHT):
39             for col in range(Puzzle8.WIDTH):
40                 index = randint(0, len(items)-1)
41                 item = items[index]
```

```
42                 self[row, col] = item
43                 if item == ' ':
44                     self.__space = (row, col)
45                 items.remove(item)
46 
47     def _calcInversions(self):
48         def merge_sort(items):
49             if len(items) <= 1:
50                 return items
51             pos = len(items) // 2
52             half1 = items[:pos]
53             half2 = items[pos:]
54             left = merge_sort(half1)
55             right = merge_sort(half2)
56             return merge(left, right)
57 
58         def merge(left, right):
59             list = []
60             while len(left) > 0 and len(right) > 0:
61                 item1 = left[0]
62                 item2 = right[0]
63                 if item1 <= item2:
64                     list.append(left.pop(0))
65                 else:
66                     self.__inversions += len(left)
67                     list.append(right.pop(0))
68             list.extend(left)
69             list.extend(right)
70             return list
71 
72         self.__inversions = 0
```

```
73          items = [int(self[row, col]) \
74                       for row in range(Puzzle8.HEIGHT) \
75                           for col in range(Puzzle8.WIDTH) \
76                               if self[row, col] != ' ']
77          merge_sort(items)
78
79      def clone(self):
80          new = Puzzle8(clone=True)
81          for row in range(Puzzle8.HEIGHT):
82              for col in range(Puzzle8.WIDTH):
83                  new[row, col] = self[row, col]
84          new.__cost_so_far = self.__cost_so_far
85          new.__heuristics = self.__heuristics
86          new.__space = self.__space
87          return new
88
89      def isGoal(self):
90          for row in range(Puzzle8.HEIGHT):
91              for col in range(Puzzle8.WIDTH):
92                  if self[row, col] != Puzzle8.GOAL[row, col]:
93                      return False
94          return True
95
96      def __lt__(self, other):#for PriorityQueue
97          return self.heuristics < other.heuristics
98
99      def _calcHeuristics(self):
100         heuristics = 0
101         for row in range(Puzzle8.HEIGHT):
102             for col in range(Puzzle8.WIDTH):
103                 item = self[row, col]
```

```
104                 if item != ' ':
105                     index = Puzzle8.ITEMS.index(item)
106                     row1 = index // Puzzle8.WIDTH
107                     col1 = index % Puzzle8.WIDTH
108                     heuristics += abs(row1-row) + abs(col1-col)
109         return heuristics
110 
111     def __getitem__(self, ndxTuple):
112         return self.__puzzle8.__getitem__(ndxTuple)
113 
114     def __setitem__(self, ndxTuple, value):
115         self.__puzzle8.__setitem__(ndxTuple, value)
116 
117     def numRows(self):
118         return self.__puzzle8.numRows()
119 
120     def numCols(self):
121         return self.__puzzle8.numCols()
122 
123     def getAllMoves(self):
124         row = self.__space[0]
125         col = self.__space[1]
126         moves = []
127         offsets = [(0, -1), (-1, 0), (1, 0), (0, 1)]
128         for x, y in offsets:
129             x = col + x
130             y = row + y
131             if x < 0 or x > Puzzle8.WIDTH-1 or \
132                y < 0 or y > Puzzle8.HEIGHT-1:
133                 continue
134             moves.append((y, x))
```

```
135         return moves
136
137     def move(self, row, col):
138         self[self.__space[0], self.__space[1]] = self[row, col]
139         self[row, col] = ' '
140         self.__space = (row, col)
141         self.__cost_so_far += 1
142         self.__heuristics = self._calcHeuristics()
143
144     @property
145     def cost(self):
146         return self.__cost_so_far
147
148     @property
149     def heuristics(self):
150         return self.__heuristics
151
152     def ToString(self):
153         items = [self[row, col] for row in range(Puzzle8.HEIGHT) \
154                                     for col in range(Puzzle8.WIDTH)]
155         return ''.join(items)
156
157     def print(self):
158         for row in range(Puzzle8.HEIGHT):
159             for col in range(Puzzle8.WIDTH):
160                 print(self[row, col], end=' ')
161             print()
162
163 def main():
164     seed()
165     puzzle8 = Puzzle8()
```

```
166     puzzle8.print()
167     Puzzle8.GOAL.print()
168     print('cost=', puzzle8.cost)
169     print('heuristics=', puzzle8.heuristics)
170     for i in range(3):
171         moves = puzzle8.getAllMoves()
172         print(moves)
173         puzzle8.move(*moves[randint(0, len(moves)-1)])
174         print('cost=', puzzle8.cost)
175         print('heuristics=', puzzle8.heuristics)
176         puzzle8.print()
177 
178     print()
179 
180     new = puzzle8.clone()
181     new.print()
182     print('cost=', new.cost)
183     print('heuristics=', new.heuristics)
184     for i in range(3):
185         moves = new.getAllMoves()
186         print(moves)
187         new.move(*moves[randint(0, len(moves)-1)])
188         print('cost=', new.cost)
189         print('heuristics=', new.heuristics)
190         new.print()
191 
192 if __name__ == '__main__':
193     main()
```

4 Puzzle8Draw 类

Puzzle8Draw 类实现 8 数码棋盘的绘制功能。下面是 Puzzle8Draw 类的 ADT 定义：

- Puzzle8Draw(gui)

  创建一个 Puzzle8Draw 对象，参数 gui 为图形接口；

- draw(puzzle8)

  依据参数 puzzle8 绘制 8 数码方格。参数 puzzle8 是 Puzzle8 类的实例；

Puzzle8Draw 类的实现如下：

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Thu Oct 18 18:31:40 2018
4
5  @author: duxiaoqin
6  Functions:
7      (1)Puzzle8Draw class;
8  """
9
10 from graphics import *
11 from puzzle8 import *
12
13 class Puzzle8Draw:
14     WIDTH = 5.0
15     HEIGHT = 5.0
16     START = 1.0
17     END = 4.0
18
19     def __init__(self, win):
20         self.win = win
```

```
21          self.win.setCoords(0.0, 0.0, Puzzle8Draw.WIDTH, Puzzle8Draw.HEIGHT)
22
23          self.lines = []
24          for offset in range(4):
25              l = Line(Point(Puzzle8Draw.START, Puzzle8Draw.START+offset), \
26                       Point(Puzzle8Draw.END, Puzzle8Draw.START+offset))
27              l.setWidth(3)
28              self.lines.append(l)
29              l = Line(Point(Puzzle8Draw.START+offset, Puzzle8Draw.START), \
30                       Point(Puzzle8Draw.START+offset, Puzzle8Draw.END))
31              l.setWidth(3)
32              self.lines.append(l)
33
34          self.items = Puzzle8.ITEMS[:]
35          self.stones = []
36          for item in self.items:
37              text = Text(Point(0, 0), item)
38              text.setSize(36)
39              text.setStyle('bold')
40              text.setOutline('red')
41              self.stones.append(text)
42
43          self.text = Text(Point(2.5, 0.5), '8 Puzzle')
44          self.text.setTextColor('red')
45
46      def draw_lines(self):
47          for l in self.lines:
48              l.undraw()
49          for l in self.lines:
50              l.draw(self.win)
51
```

```
52      def draw_puzzle8(self, puzzle8):
53          self.text.undraw()
54          self.text.draw(self.win)
55
56          for i in range(len(self.stones)):
57              self.stones[i].undraw()
58
59          for row in range(puzzle8.numRows()):
60              for col in range(puzzle8.numCols()):
61                  item = puzzle8[row, col]
62                  index = self.items.index(item)
63                  self.stones[index].anchor = \
64                       Point(Puzzle8Draw.START+1/2+col, \
65                             Puzzle8Draw.END-1/2-row)
66
67          for i in range(len(self.stones)):
68              self.stones[i].draw(self.win)
69
70      def draw(self, puzzle8):
71          self.draw_lines()
72          self.draw_puzzle8(puzzle8)
73          self.win.update()
74
75  def main():
76      win = GraphWin('8 Puzzle Draw', 600, 600, autoflush=False)
77      puzzle8 = Puzzle8()
78      puzzle8.print()
79      puzzle8draw = Puzzle8Draw(win)
80
81      while win.checkKey() != 'Escape':
82          puzzle8draw.draw(puzzle8)
```

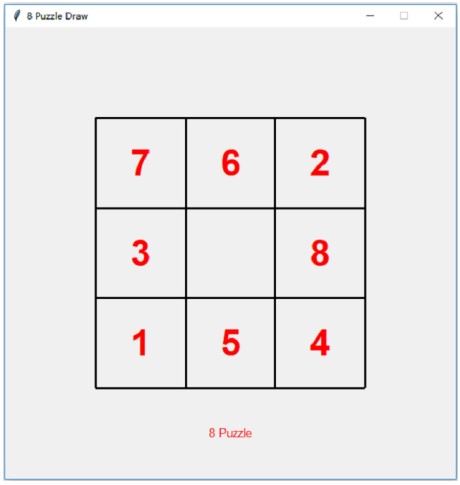

图 4-1: 8 数码示例

```
83      win.close()
84
85  if __name__ == '__main__':
86      main()
```

程序运行的一个结果如图4-1所示。

5 A* 搜索

A* 搜索是最佳优先搜索的特例，形式上完全一样。下面先给出它的一般算法形式：

```
Input:
    An 8 puzzle as a start state
    came_from is a DICT, initialized with {}
Output:
    return: GOAL or None
    came_from is changed
def Astar(puzzle8, came_from):
    frontier = PriorityQueue()
    cost_so_far = {}
```

```
    frontier.enqueue(puzzle8, 0)
    cost_so_far[puzzle8] = puzzle8.cost
    came_from[puzzle8] = None
    while not frontier.is_empty():
        puzzle8 = frontier.dequeue()
        if puzzle8 is a goal:
            return puzzle8
        else:
            for newpuzzle8 in all neighbors of puzzle8:
                new_cost = newpuzzle8.cost
                if newpuzzle8 not in cost_so_far or
                new_cost < cost_so_far[newpuzzle8]:
                    cost_so_far[newpuzzle8] = new_cost
                    priority = new_cost + newpuzzle8.heuristics
                    frontier.put(newpuzzle8, priority)
                    came_from[newpuzzle8] = puzzle8
    return None
```

注：可以把 frontier.put 理解为“如果该项已经存在，则用新的优先级替换；否则，按优先级插入该项”[1]。

8 数码的 A* 程序如下：

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Oct 21 15:25:43 2018
4
5 @author: duxiaoqin
6 Functions:
7     (1)A* class for 8 puzzle
```

[1]在实际实现时，如果明确知道问题有解，可以简单地使用 frontier.enqueue 方法替换 put 方法，即无论存在与否，均按优先级插入该项。这种变形的处理方法，不会影响算法的功能实现。原因在于，即使该项已经存在，新插入项的优先级高，必将优先扩展。另外，算法在找到一个目标后会停止运行，即使 frontier 优先级队列里还包含节点。

```
8  """
9
10 from time import *
11 from random import *
12 from priorityqueue import PriorityQueue
13 from stack import Stack
14 from puzzle8 import *
15 from puzzle8draw import *
16 from graphics import *
17
18 def Astar(puzzle8, came_from):
19     frontier = PriorityQueue()
20     cost_so_far = {}
21     frontier.enqueue(puzzle8, 0)
22     cost_so_far[puzzle8.ToString()] = puzzle8.cost
23     came_from[puzzle8.ToString()] = None
24     while not frontier.is_empty():
25         puzzle8 = frontier.dequeue()
26
27         if puzzle8.isGoal():
28             return puzzle8
29         else:
30             moves = puzzle8.getAllMoves()
31             for move in moves:
32                 newpuzzle8 = puzzle8.clone()
33                 newpuzzle8.move(*move)
34                 new_cost = newpuzzle8.cost
35                 if cost_so_far.get(newpuzzle8.ToString()) == None or \
36                    new_cost < cost_so_far[newpuzzle8.ToString()]:
37                     cost_so_far[newpuzzle8.ToString()] = new_cost
38                     priority = new_cost + newpuzzle8.heuristics
```

```
                    frontier.enqueue(newpuzzle8, priority)
                    came_from[newpuzzle8.ToString()] = puzzle8
    return None

def main():
    seed()
    came_from = {}
    puzzle8 = Puzzle8()
    found = Astar(puzzle8, came_from)
    if found != None:
        s = Stack()
        s.push(found)
        found = came_from.get(found.ToString())
        while found != None:
            s.push(found)
            found = came_from.get(found.ToString())
        win = GraphWin('A* for 8 Puzzle', 600, 600, autoflush=False)
        puzzle8draw = Puzzle8Draw(win)

        while win.checkKey() != 'Escape':
            while not s.is_empty():
                puzzle8draw.draw(s.pop())
                time.sleep(1.25)
        win.close()
    else:
        print('Path not found!')

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

6 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。

3. 杜小勤。《人工智能》课程系列，Chapter 4：启发式搜索技术，2018/10/08。