Tecnológico de Monterrey

Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

# Homework 14:
# 8 Tree

Group: 607
Team #3
Luis Salomón Flores Ugalde

Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida a01722925

File Edit Selection View Go Run Terminal Help

2.Advanced Algorithms

A01571222_A01722925_8puzzle.py U

Homework14_Puzzle > A01571222_A01722925_8puzzle.py > Node > __init__

```python
# Analysis and Design of Advanced Algorithms
# Group #607
# Team 3
# Luis Salomón Flores Ugalde

# Santiago Quintana Moreno A01571222
# Miguel Ángel Álvarez Hermida A01722925

# ------ HOMEWORK 14 PUZZLE SOLVER ------

import random
import itertools
import collections
import time

class Node:
    """
    A class representing an Solver node
    - 'puzzle' is a Puzzle instance see that move() creates a new puzzle depending on the move
    - 'parent' is the preceding node generated by the solver, if any
    - 'action' is the action taken to produce puzzle, if any
    """

    def __init__(self, puzzle, parent=None, action=None):
        self.puzzle = puzzle
        self.parent = parent
        self.action = action
        if parent:
            self.g = parent.g + 1
        else:
            self.g = 0


    def state(self):
        """
        Return a hashable representation of self
        """
        return str(self)

    def path(self):
        """
        Reconstruct path by walking back from current node to start
        """
        node, p = self, []
        while node:
            p.append(node)
            node = node.parent
        return p[::-1]
```

powershell

```
(.jupy) PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & "D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms\.jupy\Scripts\python.exe" "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Homework14_Puzzle/A01571222_A01722925_8puzzle.py"
None
[1, 2, 3]
[4, 5, 0]
[6, 7, 8]

U
[1, 2, 0]
[4, 5, 3]
[6, 7, 8]

L
[1, 0, 2]
[4, 5, 3]
[6, 7, 8]

D
[1, 5, 2]
[4, 0, 3]
[6, 7, 8]

L
[1, 5, 2]
[0, 4, 3]
[6, 7, 8]

U
[0, 5, 2]
[1, 4, 3]
[6, 7, 8]

R
[5, 0, 2]
[1, 4, 3]
[6, 7, 8]

R
[5, 2, 0]
[1, 4, 3]
[6, 7, 8]

D
[5, 2, 3]
[1, 4, 0]
[6, 7, 8]

D
[5, 2, 3]
[1, 4, 8]
[6, 7, 0]

L
[5, 2, 3]
```

main* AplicativoBodega/AplicativoApp#12 needs reviewers 0 0 Go Live

File Edit Selection View Go Run Terminal Help

2.Advanced Algorithms

A01571222_A01722925_8puzzle.py U

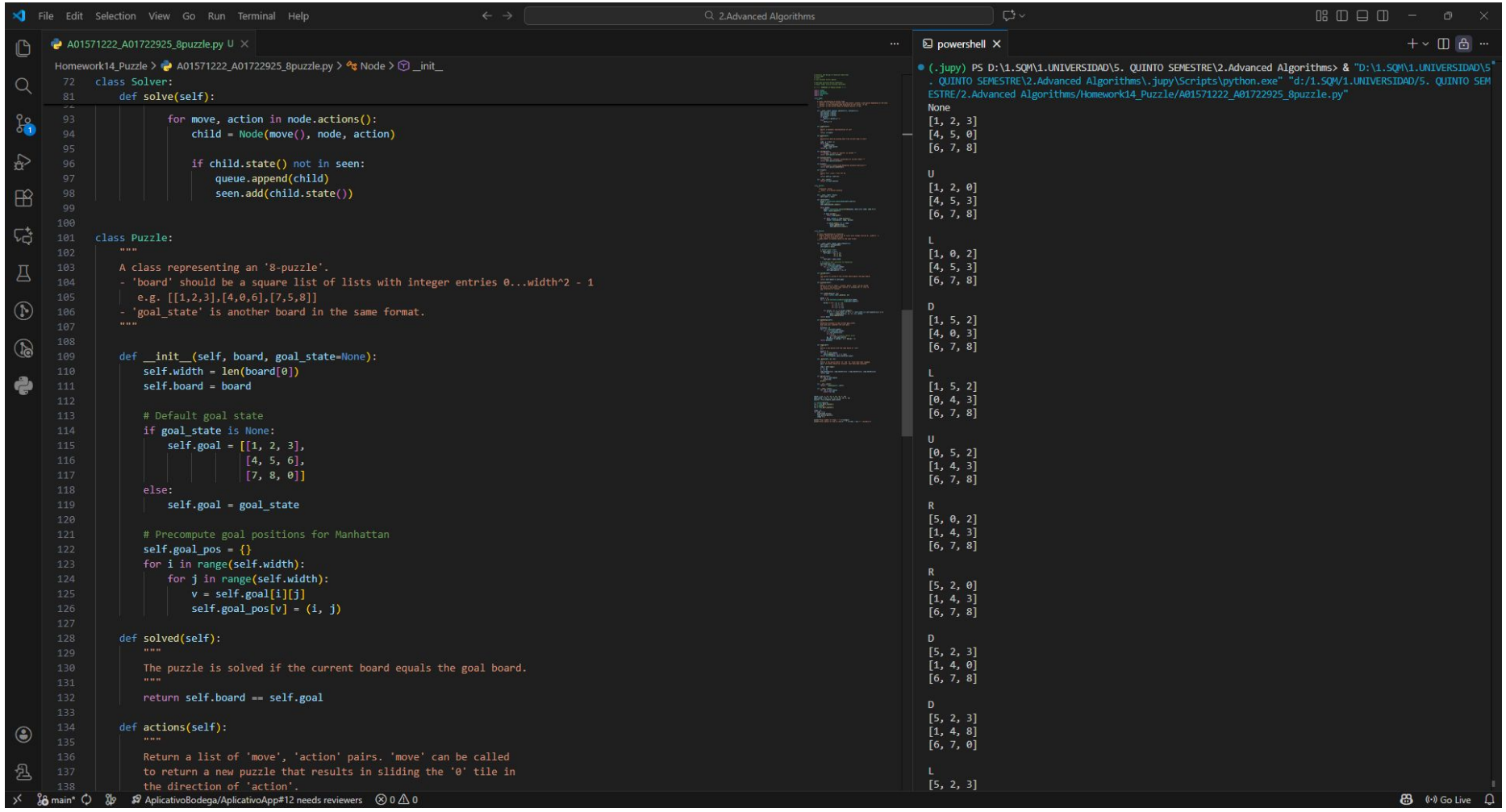
Homework14_Puzzle > A01571222_A01722925_8puzzle.py > Node > __init__

```
16  class Node:
40      def path(self):
48          return p[::-1]
49
50      def solved(self):
51          """ Wrapper to check if 'puzzle' is solved """
52          return self.puzzle.solved()
53
54      def actions(self):
55          """ Wrapper for 'actions' accessible at current state """
56          return self.puzzle.actions()
57
58      def h(self):
59          """Calculate h value using Manhattan distance heuristic"""
60          return self.puzzle.manhattan()
61
62      def f(self):
63          """
64          Return f(n) = g(n) + h(n) for A*.
65          """
66          return self.g + self.h()
67
68      def __str__(self):
69          return str(self.puzzle)
70
71
72  class Solver:
73      """
74        '8-puzzle' solver
75        - 'start' is a Puzzle instance
76       """
77
78      def __init__(self, start):
79          self.start = start
80
81      def solve(self):
82          queue = collections.deque([Node(self.start)])
83          seen = set()
84          seen.add(queue[0].state())
85
86          while queue:
87              queue = collections.deque(sorted(queue, key=lambda node: node.f()))
88              node = queue.popleft()
89
90              if node.solved():
91                  return node.path()
92
93              for move, action in node.actions():
```

powershell

```
(.jupy) PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & "D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms\.jupy\Scripts\python.exe" "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Homework14_Puzzle/A01571222_A01722925_8puzzle.py"
None
[1, 2, 3]
[4, 5, 0]
[6, 7, 8]

U
[1, 2, 0]
[4, 5, 3]
[6, 7, 8]

L
[1, 0, 2]
[4, 5, 3]
[6, 7, 8]

D
[1, 5, 2]
[4, 0, 3]
[6, 7, 8]

L
[1, 5, 2]
[0, 4, 3]
[6, 7, 8]

U
[0, 5, 2]
[1, 4, 3]
[6, 7, 8]

R
[5, 0, 2]
[1, 4, 3]
[6, 7, 8]

R
[5, 2, 0]
[1, 4, 3]
[6, 7, 8]

D
[5, 2, 3]
[1, 4, 0]
[6, 7, 8]

D
[5, 2, 3]
[1, 4, 8]
[6, 7, 0]

L
[5, 2, 3]
```

main* AplicativoBodega/AplicativoApp#12 needs reviewers 0 0 Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
A01571222_A01722925_8puzzle.py U
Homework14_Puzzle > A01571222_A01722925_8puzzle.py > Node > __init__
class Solver:
def solve(self):
for move, action in node.actions():
child = Node(move(), node, action)
if child.state() not in seen:
queue.append(child)
seen.add(child.state())
class Puzzle:
"""
A class representing an '8-puzzle'.
- 'board' should be a square list of lists with integer entries 0...width^2 - 1
e.g. [[1,2,3],[4,0,6],[7,5,8]]
- 'goal_state' is another board in the same format.
"""
def __init__(self, board, goal_state=None):
self.width = len(board[0])
self.board = board
# Default goal state
if goal_state is None:
self.goal = [[1, 2, 3],
[4, 5, 6],
[7, 8, 0]]
else:
self.goal = goal_state
# Precompute goal positions for Manhattan
self.goal_pos = {}
for i in range(self.width):
for j in range(self.width):
v = self.goal[i][j]
self.goal_pos[v] = (i, j)
def solved(self):
"""
The puzzle is solved if the current board equals the goal board.
"""
return self.board == self.goal
def actions(self):
"""
Return a list of 'move', 'action' pairs. 'move' can be called
to return a new puzzle that results in sliding the '0' tile in
the direction of 'action'.
powershell
(.jupy) PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & "D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms\.jupy\Scripts\python.exe" "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Homework14_Puzzle/A01571222_A01722925_8puzzle.py"
None
[1, 2, 3]
[4, 5, 0]
[6, 7, 8]
U
[1, 2, 0]
[4, 5, 3]
[6, 7, 8]
L
[1, 0, 2]
[4, 5, 3]
[6, 7, 8]
D
[1, 5, 2]
[4, 0, 3]
[6, 7, 8]
L
[1, 5, 2]
[0, 4, 3]
[6, 7, 8]
U
[0, 5, 2]
[1, 4, 3]
[6, 7, 8]
R
[5, 0, 2]
[1, 4, 3]
[6, 7, 8]
R
[5, 2, 0]
[1, 4, 3]
[6, 7, 8]
D
[5, 2, 3]
[1, 4, 0]
[6, 7, 8]
D
[5, 2, 3]
[1, 4, 8]
[6, 7, 0]
L
[5, 2, 3]
main*
AplicativoBodega/AplicativoApp#12 needs reviewers
0 0
Go Live

File Edit Selection View Go Run Terminal Help

2.Advanced Algorithms

A01571222_A01722925_8puzzle.py U

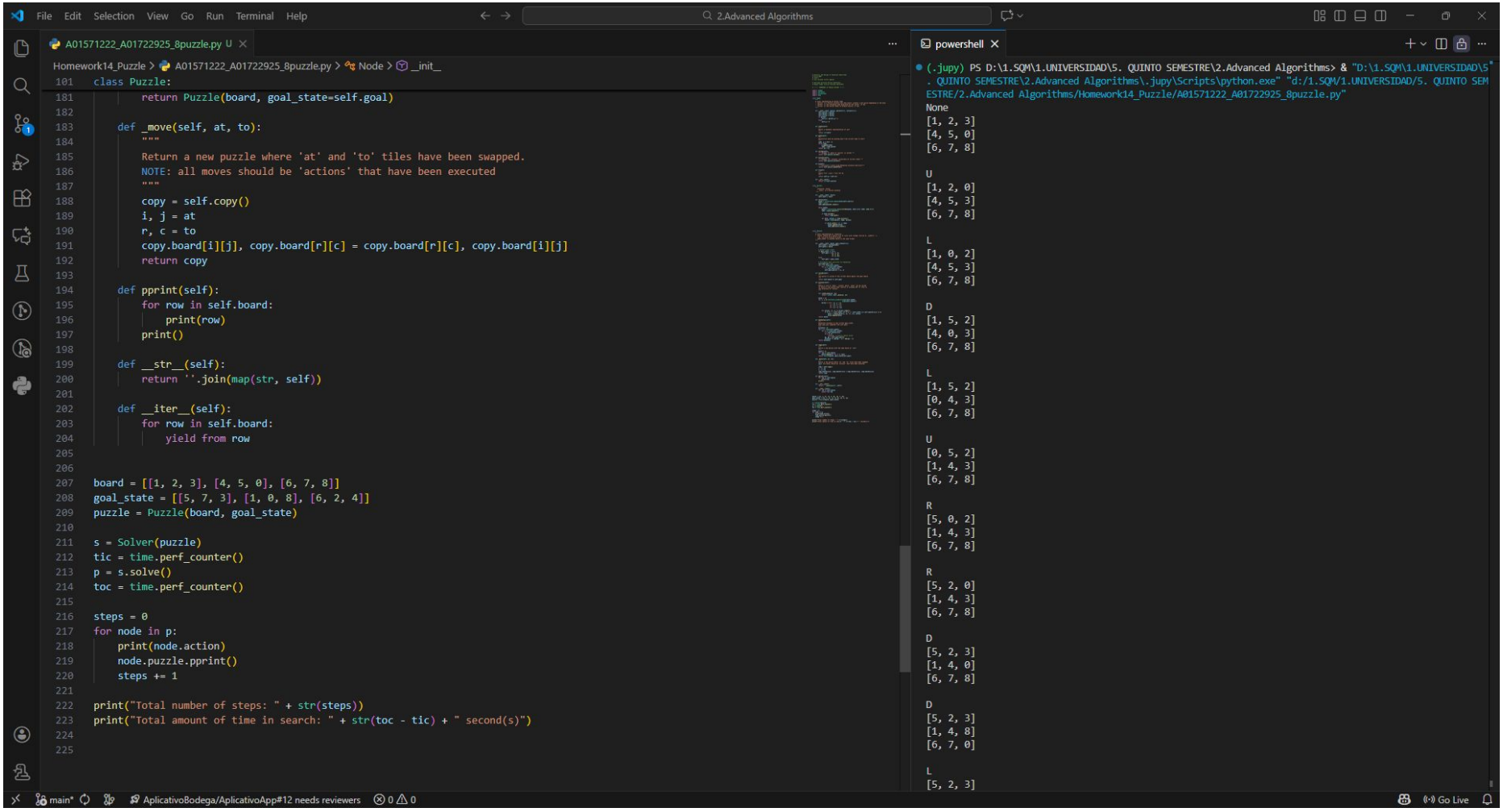
Homework14_Puzzle > A01571222_A01722925_8puzzle.py > Node > __init_

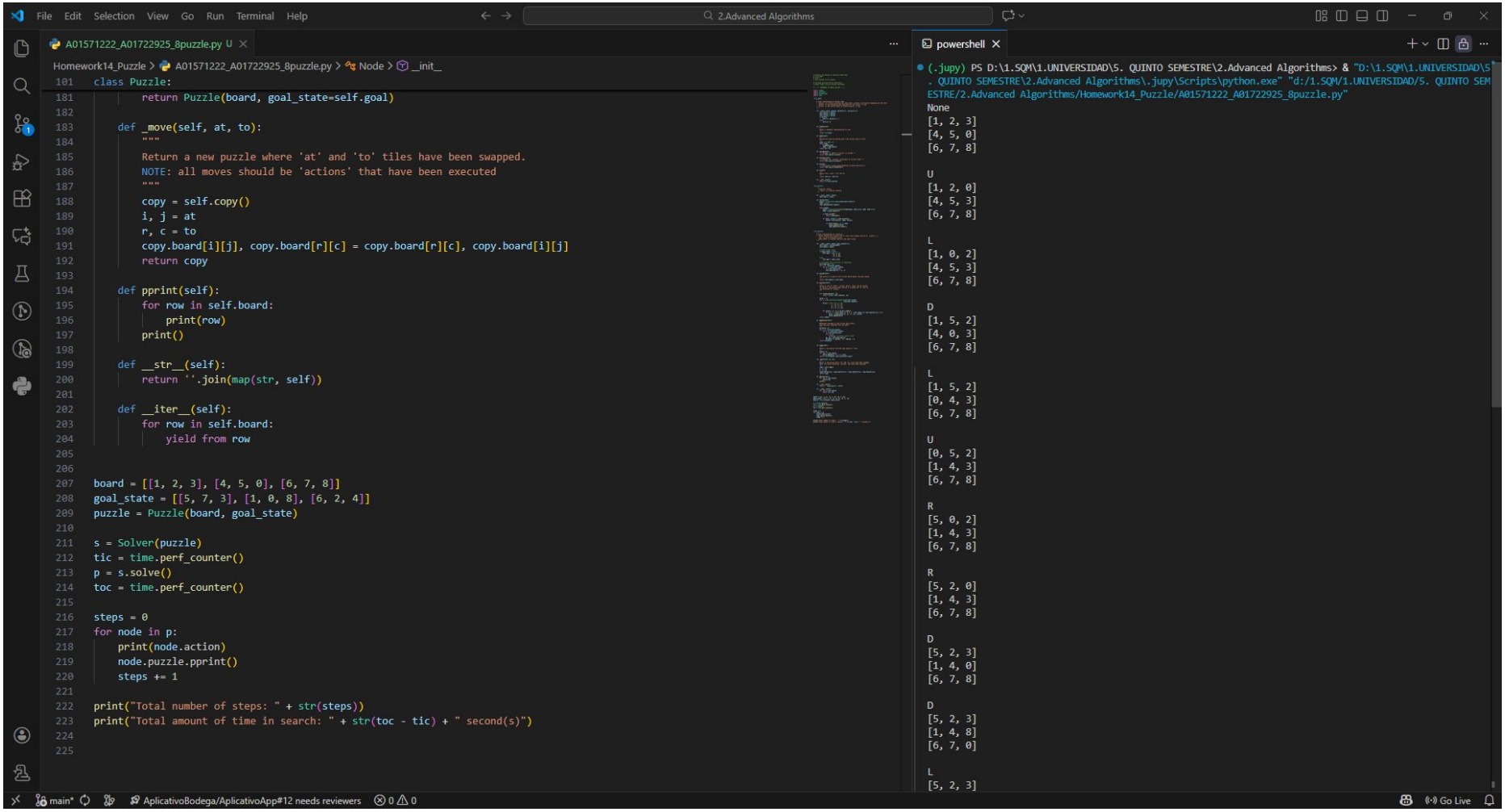
```python
class Puzzle:
    def actions(self):
        to return a new puzzle that results in sliding the '0' tile in
        the direction of 'action'.
        """

        def create_move(at, to):
            return lambda: self._move(at, to)

        moves = []
        for i, j in itertools.product(range(self.width),
                                      range(self.width)):
            direcs = {'R': (i, j - 1),
                      'L': (i, j + 1),
                      'D': (i - 1, j),
                      'U': (i + 1, j)}

            for action, (r, c) in direcs.items():
                if 0 <= r < self.width and 0 <= c < self.width and self.board[r][c] == 0:
                    move = create_move((i, j), (r, c)), action
                    moves.append(move)
        return moves

    def manhattan(self):
        """
        Manhattan distance to the current goal_state.
        Uses goal_pos computed from self.goal.
        """
        distance = 0
        for i in range(self.width):
            for j in range(self.width):
                v = self.board[i][j]
                if v == 0:
                    continue  # usually ignore blank
                gi, gj = self.goal_pos[v]
                distance += abs(gi - i) + abs(gj - j)
        return distance


    def copy(self):
        """
        Return a new puzzle with the same board as 'self'
        """
        board = []
        for row in self.board:
            board.append([x for x in row])
        return Puzzle(board, goal_state=self.goal)

    def _move(self, at, to):
```

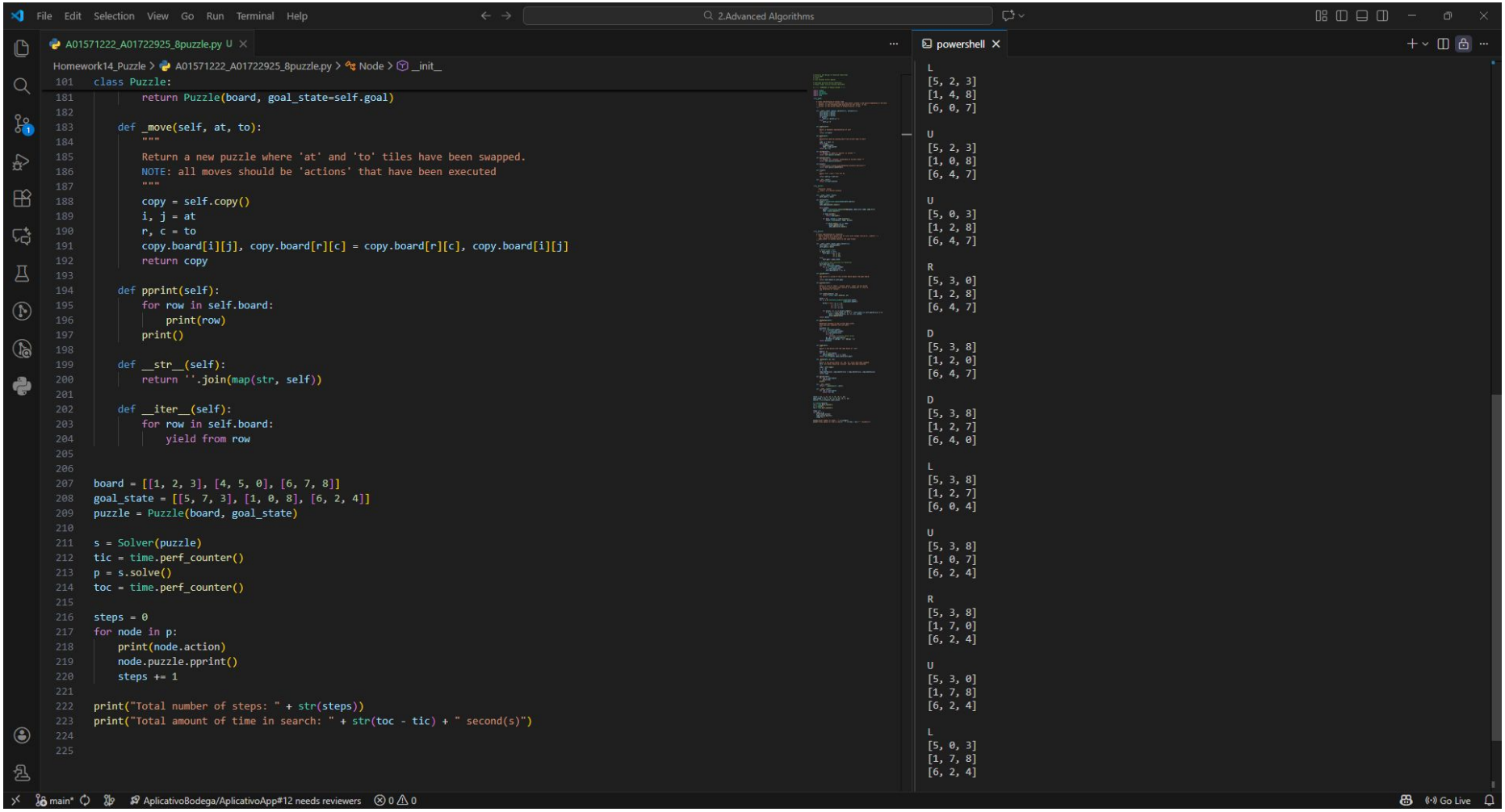
powershell

```
(.jupy) PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & "D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms\.jupy\Scripts\python.exe" "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Homework14_Puzzle/A01571222_A01722925_8puzzle.py"
None
[1, 2, 3]
[4, 5, 0]
[6, 7, 8]

U
[1, 2, 0]
[4, 5, 3]
[6, 7, 8]

L
[1, 0, 2]
[4, 5, 3]
[6, 7, 8]

D
[1, 5, 2]
[4, 0, 3]
[6, 7, 8]

L
[1, 5, 2]
[0, 4, 3]
[6, 7, 8]

U
[0, 5, 2]
[1, 4, 3]
[6, 7, 8]

R
[5, 0, 2]
[1, 4, 3]
[6, 7, 8]

R
[5, 2, 0]
[1, 4, 3]
[6, 7, 8]

D
[5, 2, 3]
[1, 4, 0]
[6, 7, 8]

D
[5, 2, 3]
[1, 4, 8]
[6, 7, 0]

L
[5, 2, 3]
```

main* AplicativoBodega/AplicativoApp#12 needs reviewers 0 0 Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
A01571222_A01722925_8puzzle.py U
Homework14_Puzzle > A01571222_A01722925_8puzzle.py > Node > __init__
class Puzzle:
return Puzzle(board, goal_state=self.goal)
def _move(self, at, to):
"""
Return a new puzzle where 'at' and 'to' tiles have been swapped.
NOTE: all moves should be 'actions' that have been executed
"""
copy = self.copy()
i, j = at
r, c = to
copy.board[i][j], copy.board[r][c] = copy.board[r][c], copy.board[i][j]
return copy
def pprint(self):
for row in self.board:
print(row)
print()
def __str__(self):
return ''.join(map(str, self))
def __iter__(self):
for row in self.board:
yield from row
board = [[1, 2, 3], [4, 5, 0], [6, 7, 8]]
goal_state = [[5, 7, 3], [1, 0, 8], [6, 2, 4]]
puzzle = Puzzle(board, goal_state)
s = Solver(puzzle)
tic = time.perf_counter()
p = s.solve()
toc = time.perf_counter()
steps = 0
for node in p:
print(node.action)
node.puzzle.pprint()
steps += 1
print("Total number of steps: " + str(steps))
print("Total amount of time in search: " + str(toc - tic) + " second(s)")
powershell
(.jupy) PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & "D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms\.jupy\Scripts\python.exe" "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Homework14_Puzzle/A01571222_A01722925_8puzzle.py"
None
[1, 2, 3]
[4, 5, 0]
[6, 7, 8]
U
[1, 2, 0]
[4, 5, 3]
[6, 7, 8]
L
[1, 0, 2]
[4, 5, 3]
[6, 7, 8]
D
[1, 5, 2]
[4, 0, 3]
[6, 7, 8]
L
[1, 5, 2]
[0, 4, 3]
[6, 7, 8]
U
[0, 5, 2]
[1, 4, 3]
[6, 7, 8]
R
[5, 0, 2]
[1, 4, 3]
[6, 7, 8]
R
[5, 2, 0]
[1, 4, 3]
[6, 7, 8]
D
[5, 2, 3]
[1, 4, 0]
[6, 7, 8]
D
[5, 2, 3]
[1, 4, 8]
[6, 7, 0]
L
[5, 2, 3]
main*
AplicativoBodega/AplicativoApp#12 needs reviewers
0 0
Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
A01571222_A01722925_8puzzle.py U
Homework14_Puzzle > A01571222_A01722925_8puzzle.py > Node > __init__
class Puzzle:
return Puzzle(board, goal_state=self.goal)
def _move(self, at, to):
"""
Return a new puzzle where 'at' and 'to' tiles have been swapped.
NOTE: all moves should be 'actions' that have been executed
"""
copy = self.copy()
i, j = at
r, c = to
copy.board[i][j], copy.board[r][c] = copy.board[r][c], copy.board[i][j]
return copy
def pprint(self):
for row in self.board:
print(row)
print()
def __str__(self):
return ''.join(map(str, self))
def __iter__(self):
for row in self.board:
yield from row
board = [[1, 2, 3], [4, 5, 0], [6, 7, 8]]
goal_state = [[5, 7, 3], [1, 0, 8], [6, 2, 4]]
puzzle = Puzzle(board, goal_state)
s = Solver(puzzle)
tic = time.perf_counter()
p = s.solve()
toc = time.perf_counter()
steps = 0
for node in p:
print(node.action)
node.puzzle.pprint()
steps += 1
print("Total number of steps: " + str(steps))
print("Total amount of time in search: " + str(toc - tic) + " second(s)")
powershell
(.jupy) PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & "D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms\.jupy\Scripts\python.exe" "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/Homework14_Puzzle/A01571222_A01722925_8puzzle.py"
None
[1, 2, 3]
[4, 5, 0]
[6, 7, 8]
U
[1, 2, 0]
[4, 5, 3]
[6, 7, 8]
L
[1, 0, 2]
[4, 5, 3]
[6, 7, 8]
D
[1, 5, 2]
[4, 0, 3]
[6, 7, 8]
L
[1, 5, 2]
[0, 4, 3]
[6, 7, 8]
U
[0, 5, 2]
[1, 4, 3]
[6, 7, 8]
R
[5, 0, 2]
[1, 4, 3]
[6, 7, 8]
R
[5, 2, 0]
[1, 4, 3]
[6, 7, 8]
D
[5, 2, 3]
[1, 4, 0]
[6, 7, 8]
D
[5, 2, 3]
[1, 4, 8]
[6, 7, 0]
L
[5, 2, 3]
main*
AplicativoBodega/AplicativoApp#12 needs reviewers
0 0
Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
A01571222_A01722925_8puzzle.py U
Homework14_Puzzle > A01571222_A01722925_8puzzle.py > Node > __init__
class Puzzle:
return Puzzle(board, goal_state=self.goal)
def _move(self, at, to):
"""
Return a new puzzle where 'at' and 'to' tiles have been swapped.
NOTE: all moves should be 'actions' that have been executed
"""
copy = self.copy()
i, j = at
r, c = to
copy.board[i][j], copy.board[r][c] = copy.board[r][c], copy.board[i][j]
return copy
def pprint(self):
for row in self.board:
print(row)
print()
def __str__(self):
return ''.join(map(str, self))
def __iter__(self):
for row in self.board:
yield from row
board = [[1, 2, 3], [4, 5, 0], [6, 7, 8]]
goal_state = [[5, 7, 3], [1, 0, 8], [6, 2, 4]]
puzzle = Puzzle(board, goal_state)
s = Solver(puzzle)
tic = time.perf_counter()
p = s.solve()
toc = time.perf_counter()
steps = 0
for node in p:
print(node.action)
node.puzzle.pprint()
steps += 1
print("Total number of steps: " + str(steps))
print("Total amount of time in search: " + str(toc - tic) + " second(s)")
powershell
L
[5, 2, 3]
[1, 4, 8]
[6, 0, 7]
U
[5, 2, 3]
[1, 0, 8]
[6, 4, 7]
U
[5, 0, 3]
[1, 2, 8]
[6, 4, 7]
R
[5, 3, 0]
[1, 2, 8]
[6, 4, 7]
D
[5, 3, 8]
[1, 2, 0]
[6, 4, 7]
D
[5, 3, 8]
[1, 2, 7]
[6, 4, 0]
L
[5, 3, 8]
[1, 2, 7]
[6, 0, 4]
U
[5, 3, 8]
[1, 0, 7]
[6, 2, 4]
R
[5, 3, 8]
[1, 7, 0]
[6, 2, 4]
U
[5, 3, 0]
[1, 7, 8]
[6, 2, 4]
L
[5, 0, 3]
[1, 7, 8]
[6, 2, 4]
main*
AplicativoBodega/AplicativoApp#12 needs reviewers
0 0
Go Live

File Edit Selection View Go Run Terminal Help

2.Advanced Algorithms

A01571222_A01722925_8puzzle.py U

Homework14_Puzzle > A01571222_A01722925_8puzzle.py > Node > __init__

```
class Puzzle:
        return Puzzle(board, goal_state=self.goal)

    def _move(self, at, to):
        """
        Return a new puzzle where 'at' and 'to' tiles have been swapped.
        NOTE: all moves should be 'actions' that have been executed
        """
        copy = self.copy()
        i, j = at
        r, c = to
        copy.board[i][j], copy.board[r][c] = copy.board[r][c], copy.board[i][j]
        return copy

    def pprint(self):
        for row in self.board:
            print(row)
        print()

    def __str__(self):
        return ''.join(map(str, self))

    def __iter__(self):
        for row in self.board:
            yield from row


board = [[1, 2, 3], [4, 5, 0], [6, 7, 8]]
goal_state = [[5, 7, 3], [1, 0, 8], [6, 2, 4]]
puzzle = Puzzle(board, goal_state)

s = Solver(puzzle)
tic = time.perf_counter()
p = s.solve()
toc = time.perf_counter()

steps = 0
for node in p:
    print(node.action)
    node.puzzle.pprint()
    steps += 1

print("Total number of steps: " + str(steps))
print("Total amount of time in search: " + str(toc - tic) + " second(s)")
```

powershell

```
[6, 4, 7]

U
[5, 0, 3]
[1, 2, 8]
[6, 4, 7]

R
[5, 3, 0]
[1, 2, 8]
[6, 4, 7]

D
[5, 3, 8]
[1, 2, 0]
[6, 4, 7]

D
[5, 3, 8]
[1, 2, 7]
[6, 4, 0]

L
[5, 3, 8]
[1, 2, 7]
[6, 0, 4]

U
[5, 3, 8]
[1, 0, 7]
[6, 2, 4]

R
[5, 3, 8]
[1, 7, 0]
[6, 2, 4]

U
[5, 3, 0]
[1, 7, 8]
[6, 2, 4]

L
[5, 0, 3]
[1, 7, 8]
[6, 2, 4]

D
[5, 7, 3]
[1, 0, 8]
[6, 2, 4]

Total number of steps: 22
Total amount of time in search: 0.2641739000027883 second(s)
(.jupy) PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```

main* AplicativoBodega/AplicativoApp#12 needs reviewers 0 0 Go Live

https://colab.research.google.com/drive/14YoHYCqee8-XPWc_uEUSPSjNXZ6M_Oa5?usp=sharing