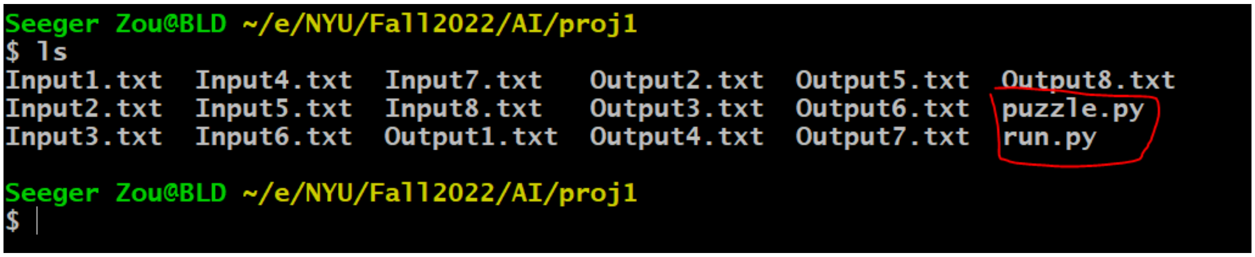
Seeger Zou, jz3714
Justin Cheok, jhc737

Fall 2022

CS4613 Artificial Intelligence Project 1: 15 Puzzle Solver

I. Instructions:

1. Make sure puzzle.py and run.py are in the directory



```
Seeger Zou@BLD ~/e/NYU/Fall2022/AI/proj1
$ ls
Input1.txt  Input4.txt  Input7.txt   Output2.txt  Output5.txt  Output8.txt
Input2.txt  Input5.txt  Input8.txt   Output3.txt  Output6.txt  puzzle.py
Input3.txt  Input6.txt  Output1.txt  Output4.txt  Output7.txt  run.py

Seeger Zou@BLD ~/e/NYU/Fall2022/AI/proj1
$ 
```

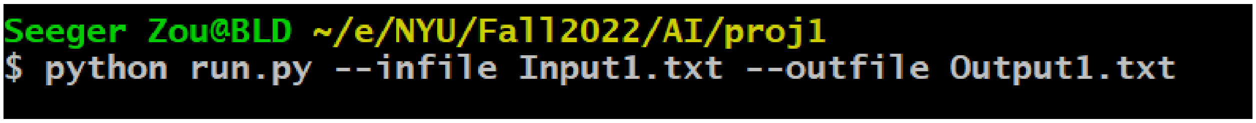
2. Run the following command

> python run.py –infile **arg1** –outfile **arg2**

***arg1** = input filename

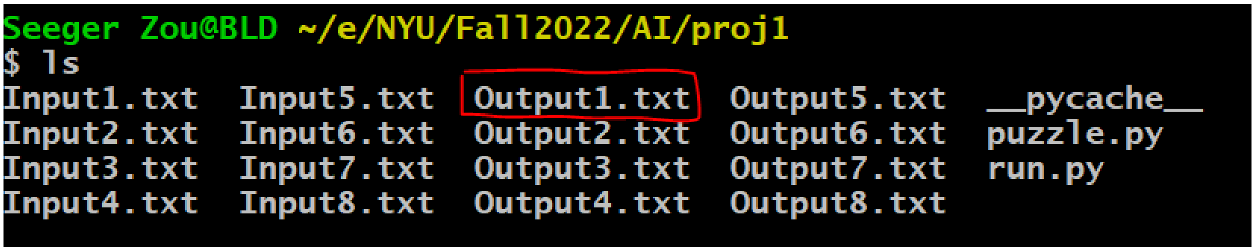
***arg2** = output filename

An example is shown below:

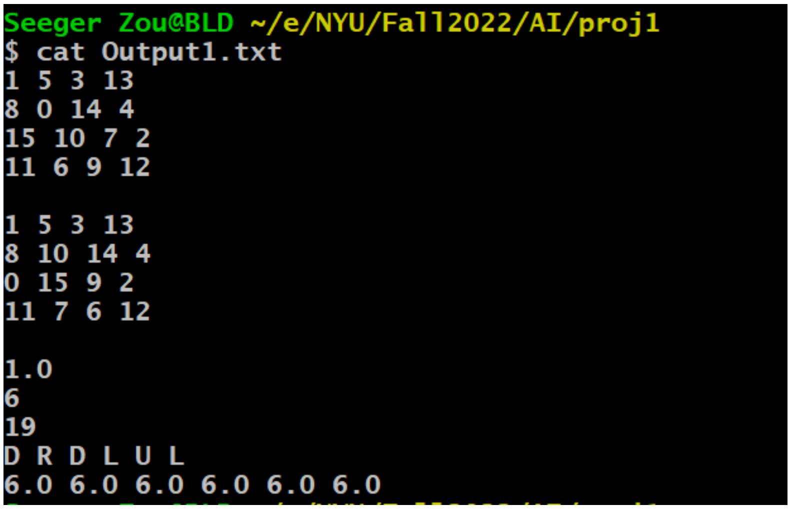


```
Seeger Zou@BLD ~/e/NYU/Fall2022/AI/proj1
$ python run.py --infile Input1.txt --outfile Output1.txt
```

3. Output file is generated



```
Seeger Zou@BLD ~/e/NYU/Fall2022/AI/proj1
$ ls
Input1.txt  Input5.txt  Output1.txt  Output5.txt  __pycache__
Input2.txt  Input6.txt  Output2.txt  Output6.txt  puzzle.py
Input3.txt  Input7.txt  Output3.txt  Output7.txt  run.py
Input4.txt  Input8.txt  Output4.txt  Output8.txt
```



```
Seeger Zou@BLD ~/e/NYU/Fall2022/AI/proj1
$ cat Output1.txt
1 5 3 13
8 0 14 4
15 10 7 2
11 6 9 12

1 5 3 13
8 10 14 4
0 15 9 2
11 7 6 12

1.0
6
19
D R D L U L
6.0 6.0 6.0 6.0 6.0 6.0
```

II. Source code:

Copy and pasted from puzzle.py and run.py

puzzle.py

```
class Puzzle:
    def __init__(self,board,weight,goal,path_cost=0,depth=0,actions=None,fvals = None):
        '''
        board -> lst of lst, matrix representation of the puzzle board
        or initial state
        ex. [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,0]]

        goal -> lst of lst, maxtrix representation of the goal state
        pathcost -> int, path cost of current board, if not passed in default to 0
        '''
        self.board = board
        self.ROWS = len(board)
        self.COLS = len(board[0])
        self.weight = weight
        self.goal = goal
        self.path_cost = path_cost
        self.depth = depth
        if actions is None:
            actions = []
        if fvals is None:
            fvals = []
        self.actions = actions
        self.fvals = fvals

        # Error checking
        if self.ROWS != self.COLS:
            raise ValueError("Invalid dimensions of puzzle board")

    def swap(self,x1,y1,x2,y2):
        '''
        given two sets of coordinates,
        returns a copy of the board with swapped positions
        x1,y1,x2,y2 -> int ~ (0-3)
        '''
        new_puz =  [list(row) for row in self.board]
        new_puz[x1][y1],new_puz[x2][y2] = new_puz[x2][y2],new_puz[x1][y1]

        return new_puz

    def getPosition(self,digit,board=None):
        '''
        given a digit and a board,
        returns the position coordinate of the given digit in the board
        digit -> int ~ (0,15)
        board -> lst of lst
        if board not given, use the current board
        if board is given, use the board given (will be goal state to calculate
        manhattan distance)
        '''
        # if a board is not given = current state
        if board==None:
            board=self.board

        for i in range(self.ROWS):
            for j in range(self.COLS):
                if board[i][j] == digit:
                    return i,j

        raise RuntimeError('Could not find digit in given board')

    def getMoves(self):
        '''
        searches for the position of the empty block (0)
        returns a list of all moves: left, right, up, down
        '''
        moves = []
        r,c = self.getPosition(0)

        # can we move left
        if c > 0:
            new = Puzzle(self.swap(r,c,r,c-1),self.weight, self.goal,self.path_cost+1,self.depth+1,
self.actions + ['L'], self.fvals + [self.heuristic()])
            moves.append(('L',new))

        # can we move right
        if c < self.COLS - 1:
            new = Puzzle(self.swap(r,c,r,c+1),self.weight, self.goal,self.path_cost+1,self.depth+1,
self.actions + ['R'], self.fvals + [self.heuristic()])
            moves.append(('R',new))

        # can we move up
        if r > 0:
            new = Puzzle(self.swap(r,c,r-1,c),self.weight, self.goal,self.path_cost+1,self.depth+1,
self.actions + ['U'], self.fvals + [self.heuristic()])
```

```
            moves.append(('U',new))

        # can we move down
        if r < self.ROWS - 1:
            new = Puzzle(self.swap(r,c,r+1,c),self.weight, self.goal,self.path_cost+1,self.depth+1,
self.actions + ['D'], self.fvals + [self.heuristic()])
            moves.append(('D',new))

        return moves

    def heuristic(self):
        '''
        returns the heuristic function value from current to goal
        weight -> float W > 1
        '''
        distance = 0

        for i in range(self.ROWS):
            for j in range(self.COLS):
                if self.board[i][j] == 0: # the blank spot does not count
                    continue
                r,c = self.getPosition(self.board[i][j],self.goal)
                distance += abs(i-r) + abs(j-c)
        return self.path_cost + self.weight*distance
```

run.py

```
from puzzle import Puzzle
from socket import inet_pton
import argparse

def listify(lst):
    '''
    turns the states into list of lists
    returnlst = list of lists (2D matrix for the 4x4 puzzle)
    '''
    returnlst = []
    for item in lst:
        nums = [int(x) for x in item.split()]
        returnlst.append(nums)
    return returnlst

def printlst(lst, f):
    '''
    formatting the list to print out each item in the list
    '''
    for item in lst:
        for enter in item:
            f.write(str(enter))
            f.write(" ")
        f.write('\n')
    f.write('\n')

def result(lst, w):
    acts = ""
    for item in lst[2]:
        acts += str(item) + " "
    hvals = ""
    for item in lst[3]:
        hvals += str(item) + " "
    return "{w}\n{d}\n{N}\n{actions}\n{hvals}".format(w =
w,d=lst[0],N=lst[1],actions=acts,hvals=hvals)

def solve(puzzle,weight):
    '''
    uses the a* algorithm to solve the puzzle
    weight -> float W > 1
    :returns:res -> lst
    res[0] = d, level of shallowest goal
    res[1] = N, total number of nodes generated not includig root
    res[2] = action list
    res[3] = heuristic values list
    '''
    res = [0]*4 # list of results
    d = 0
    N = 0


    frontier = [puzzle]
    visited = []
    while frontier:

        min_index = 0
        for i in range(len(frontier)):
            if frontier[min_index].heuristic() > frontier[i].heuristic():
                min_index = i

        curr=frontier[min_index]
        frontier.pop(min_index)
        # perform checks
        if curr.board == curr.goal:
            d = curr.depth
            break
        if curr.board in visited: # do not allow repeated states
            continue

        for move in curr.getMoves():
            if move[1].board in visited:
                continue
            N+=1
            frontier.append(move[1])
            visited.append(curr.board)
    res[0] = d
    res[1] = N
    res[2] = curr.actions
    res[3] = curr.fvals

    return res

def main():
    '''
    takes input file and outputs the result using argparse
    infile = list of lines from input file
    w = w-value
    start = initial state
    goal = goal state
    puz = Puzzle class
    res = solver for puzzle, given w-value
    '''
    parser = argparse.ArgumentParser(description='15 Puzzle solver')
    parser.add_argument('--infile',type=argparse.FileType('r'),help='input file')
    parser.add_argument('--outfile',type=argparse.FileType('w'),help='output file')
    args = parser.parse_args()
    #Reads the lines from the input file
    infile = args.infile.readlines()
    #Initialize output file
    outfile = args.outfile
    #Strips all whitespace
    infile = [k.rstrip() for k in infile]
    for item in infile:
        if not item:
            infile.remove(item)
    #Obtains input data from infile
    w = float(infile[0])
    start = listify(infile[1:5])
    goal = listify(infile[5:9])

    #Prints the initial & goal state

    #Create the puzzle with the initial & goal states
    puz = Puzzle(start,w,goal)

    #Input the puzzle and the w-value into the solver function
    res = solve(puz,w)

    #Prints the output
    printlst(start, outfile)
    printlst(goal, outfile)
    outfile.writelines(result(res, w))

main()

# Outputs are on the next page
```

III. Output Files

Output1.txt

```
1 5 3 13
8 0 14 4
15 10 7 2
11 6 9 12

1 5 3 13
8 10 14 4
0 15 9 2
11 7 6 12

1.0
6
19
D R D L U L
6.0 6.0 6.0 6.0 6.0 6.0
```

Output2.txt

```
2 13 7 4
```

```
12 3 0 1
9 15 5 14
6 10 11 8

13 3 7 4
2 1 0 14
12 9 5 8
6 15 10 11

1.0
12
26
R D D L L U L U U R D R
12.0 12.0 12.0 12.0 12.0 12.0 12.0 12.0 12.0 12.0 12.0 12.0
```

Output3.txt

```
13 12 9 11
10 1 8 2
0 3 15 6
14 4 7 5

10 13 12 11
8 1 9 2
3 4 15 5
14 0 6 7

1.0
16
178
R U R U L L D R D R R D L U L D
12.0 12.0 14.0 14.0 14.0 14.0 14.0 14.0 14.0 14.0 16.0 16.0 16.0 16.0 16.0 16.0
```

Output4.txt

```
13 12 9 11
10 1 8 2
0 3 15 6
14 4 7 5

10 13 12 11
8 1 9 2
3 4 15 5
14 0 6 7
```

```
1.2
16
70
R U R U L L D R D R R D L U L D
14.399999999999999 14.2 16.4 16.2 16.0 15.799999999999999 15.6 15.4 15.2 15.0 17.2 17.0
16.8 16.6 16.4 16.2
```

<u>Output5.txt</u>

```
13 12 9 11
10 1 8 2
0 3 15 6
14 4 7 5

10 13 12 11
8 1 9 2
3 4 15 5
14 0 6 7

1.4
16
67
R U R U L L D R D R R D L U L D
16.799999999999997 16.4 18.799999999999997 18.4 18.0 17.6 17.2 16.799999999999997
16.4 16.0 18.4 18.0 17.6 17.2 16.8 16.4
```

<u>Output6.txt</u>

```
7 1 4 12
5 3 9 10
15 14 8 6
13 11 0 2

4 9 10 12
1 7 0 6
15 5 3 2
13 11 14 8

1.0
20
82
U L U L U R R D R D D L U L U L U R R D
```

```
20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0
20.0
```

Output7.txt

```
7 1 4 12
5 3 9 10
15 14 8 6
13 11 0 2

4 9 10 12
1 7 0 6
15 5 3 2
13 11 14 8

1.2
20
45
U L U L U R R D R D D L U L U L U R R D
24.0 23.8 23.599999999999998 23.4 23.2 23.0 22.8 22.6 22.4 22.2 22.0 21.799999999999997
21.6 21.4 21.2 21.0 20.8 20.6 20.4 20.2
```

Output8.txt

```
7 1 4 12
5 3 9 10
15 14 8 6
13 11 0 2

4 9 10 12
1 7 0 6
15 5 3 2
13 11 14 8

1.4
20
45
U L U L U R R D R D D L U L U L U R R D
28.0 27.599999999999998 27.2 26.799999999999997 26.4 26.0 25.599999999999998 25.2
24.799999999999997 24.4 24.0 23.6 23.2 22.799999999999997 22.4 22.0 21.6 21.2 20.8 20.4
```