인공지능

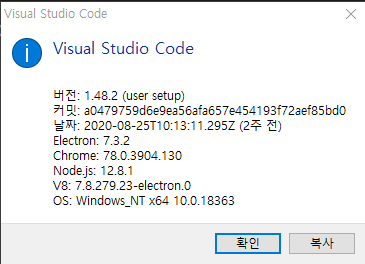
(2주차 과제)

20164091

송희령

0 > 파이썬 버전은 3.8버전이며 에디터로 Visual Studio Code를 이용하였습니다.





1 > 8\_puzzle 코드

class State:

    def \_\_init\_\_(self, board, goal, moves = 0):

        self.board = board

        self.moves = moves

        self.goal = goal

현재 상태와 목표상태를 정의하는 생성자 입니다. 게임판 board와 이동 수, 목표를 나타냅니다.

def get\_new\_board(self, i1, i2 ,moves):

        new\_board = self.board[:]

        new\_board[i1], new\_board[i2] = new\_board[i2], new\_board[i1]

        return State(new\_board, self.goal, moves)

숫자 i1과 i2를 서로 바꾸어 새로운 상태로 반환시키는 메소드 입니다.

def expand(self, moves):

        result = []

        i = self.board.index(0)

        print(str(i) + "\n")

        if i not in [0, 1, 2]:

            result.append(self.get\_new\_board(i, i - 3, moves))

        if i not in [0, 3, 6]:

            result.append(self.get\_new\_board(i, i - 1, moves))

        if i not in [2, 5, 8]:

            result.append(self.get\_new\_board(i, i + 1, moves))

        if i not in [6, 7, 8]:

            result.append(self.get\_new\_board(i, i + 3, moves))

        return result

연산자를 통하여 결과값에 도달하도록 하는 expand 입니다.

각 사각형 변 끝부분에 맞게 연산자를 구성하여 사각형을 벗어나지 못하도록 합니다.

def \_\_str\_\_(self):

        return str(self.board[:3]) + "\n" + str(self.board[3:6]) + "\n" + str(self.board[6:9]) + "\n" + "------------------"

게임판 출력문 입니다. 3x3 칸에 맞도록 배열을 3칸씩 출력하게 합니다.

#초기 상태

puzzle = [1,2,3,0,4,6,7,5,8]

#목표 상태

goal = [1,2,3,4,5,6,7,8,0]

초기 상태와 목표 상태를 배열 형식으로 제공합니다.

open\_queue = []

open\_queue.append(State(puzzle,goal))

closed\_queue = []

moves = 0

초기상태를 객체로 오픈큐에 저장하고 닫힌큐를 만들어준 후 이동을 나타내는 moves를 0으로 표현합니다.

while len(open\_queue) != 0:

    current = open\_queue.pop(0)

    print(current)

    if(current.board == goal): #목표 상태 확인

        print("탐색 성공")

        break

    moves = current.moves+1

    closed\_queue.append(moves)

    for state in current.expand(moves):

        if(state in closed\_queue) or (state in open\_queue):

            continue

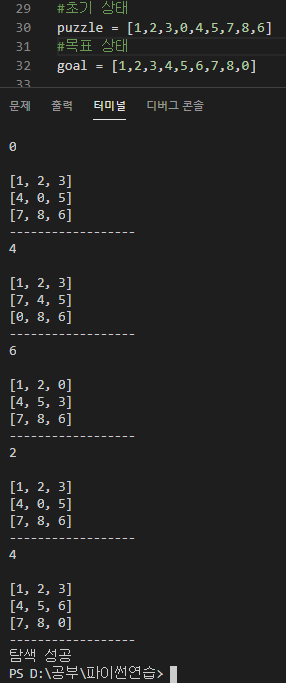
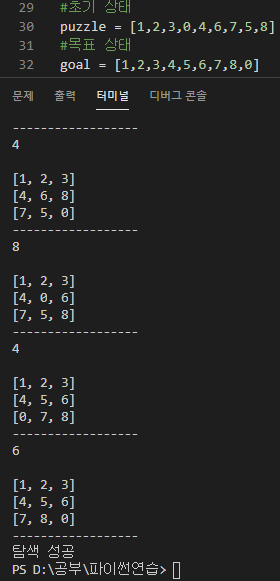
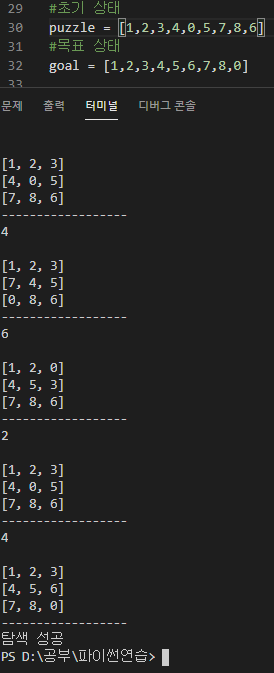
        else:

            open\_queue.append(state)

실제 초기상태에서 목표상태로 값을 내어줄 while문 입니다.

해당 루프를 통과하면 탐색에 성공하게 됩니다.

2> 실행결과



각기 다른 초기상태를 3번 제공하여 실행하였습니다.

3> 15\_puzzle 로 변경

기존 8\_puzzle에서 퍼즐의 크기와 이동값에 변화를 주어 완성하였습니다.

def expand(self, moves):

        result = []

        i = self.board.index(0)

        if i not in [0, 1, 2, 3]:

            result.append(self.get\_new\_board(i, i - 4, moves))

        if i not in [0, 4, 8, 12]:

            result.append(self.get\_new\_board(i, i - 1, moves))

        if i not in [3, 7, 11, 15]:

            result.append(self.get\_new\_board(i, i + 1, moves))

        if i not in [12, 13, 14, 15]:

            result.append(self.get\_new\_board(i, i + 4, moves))

        return result

기존 3x3 크기에서 4x4크기로 늘어난 만큼 사각형의 끝변에서 움직임을 수정해야 합니다. 각 끝변에 맞게 숫자들을 수정해 줍니다.

def \_\_str\_\_(self):

        return str(self.board[:4]) + "\n" + str(self.board[4:8]) + "\n" + str(self.board[8:12]) + "\n" + str(self.board[12:16]) + "\n" + "------------------"

역시 출력문 또한 4x4 크기에 맞게 변화시켜 줍니다.

#초기 상태

puzzle = [1,2,3,4,5,6,7,8,9,10,11,0,13,14,15,12]

#목표 상태

goal = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0]

초기상태와 목표상태 배열 모두 16개 숫자로 입력해 줍니다.

실행 결과로 같은 방식의 3가지 초기상태로 실행하였습니다.

