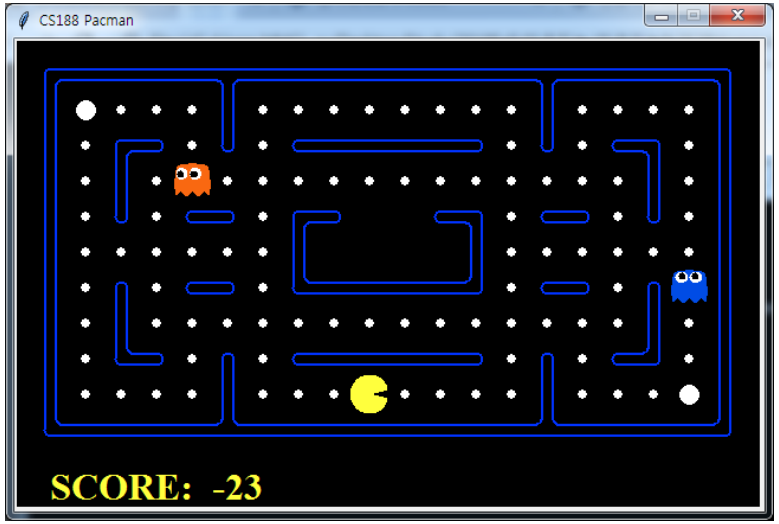
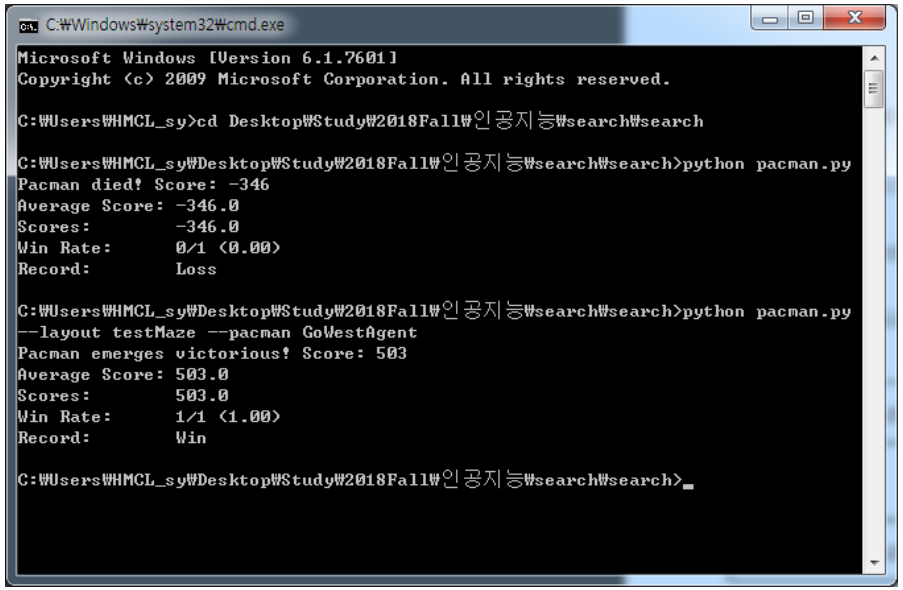
인공지능
Homework 관련

2018-10-15

HW0

- 기존 PACMAN은 2.7버전이나, 이를 3.x버전으로 포팅
- 만약 PACMAN실행이 안된다면, python 버전 확인 (python 3.x)





HW1

```
function GRAPH-SEARCH(problem, fringe) return a solution, or fa
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        if STATE[node] is not in closed then
            add STATE[node] to closed
            for child-node in EXPAND(STATE[node], problem) do
                fringe ← INSERT(child-node, fringe)
            end
    end
```

```
def depthFirstSearch(problem):
    search_space = util.Stack()
    visited = []
    path = []
    search_space.push( (problem.getStartState(), path) )

    while ( search_space.isEmpty() == 0 ) :
        state, path = search_space.pop() # REMOVE_FRONT

        if problem.isGoalState(state): # GOAL_TEST
            return path

        if state not in visited:
            visited.append( state )

            for child in problem.getSuccessors(state):
                next_state = child[0]
                next_direction = child[1]
                search_space.push ( (next_state, path + [next_direction] ) )
```

HW2

- HW2 숙제는 search.py 파일과 searchAgent 수정

<search.py>
- def uniformCostSearch(problem):
- def aStarSearch(problem, heuristic=nullHeuristic):

<searchAgent.py>
class CornersProblem(search.SearchProblem):
def cornersHeuristic(state, problem):

HW2 : UCS & A*

[문제1]에서는 Heuristic이 주어짐. 따라서, Search함수만 구현

<search.py>

- def uniformCostSearch(problem):
- def aStarSearch(problem, heuristic=nullHeuristic):

```
python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic
```

SearchAgent

- problem = default (PositionSearchProblem)
- SearchAgent.getCostOfActions()

```
for action in actions:
        dx, dy = Actions.directionToVector(action)
        x, y = int(x + dx), int(y + dy)
        if self.walls[x][y]: return 999999
            cost += self.costFn((x,y))
```

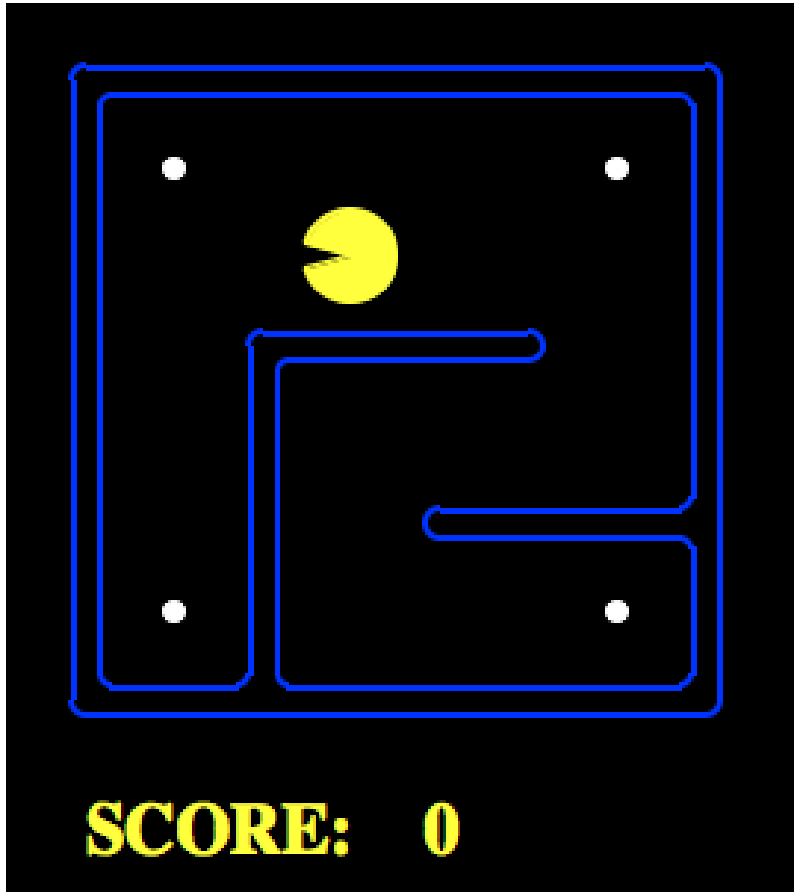
※ costFn = lambda x: 1

Priority Queue

```
class PriorityQueue:

    def push(self, item, priority):  # push를 할 때 item과 priority를 삽입
        entry = (priority, self.count, item)
        heapq.heappush(self.heap, entry)

    def pop(self):  # pop 할 때는 item만 나옴
        (_, _, item) = heapq.heappop(self.heap)
        return item

    def update(self, item, priority): # priority update 가능 (없을때는 push)
```

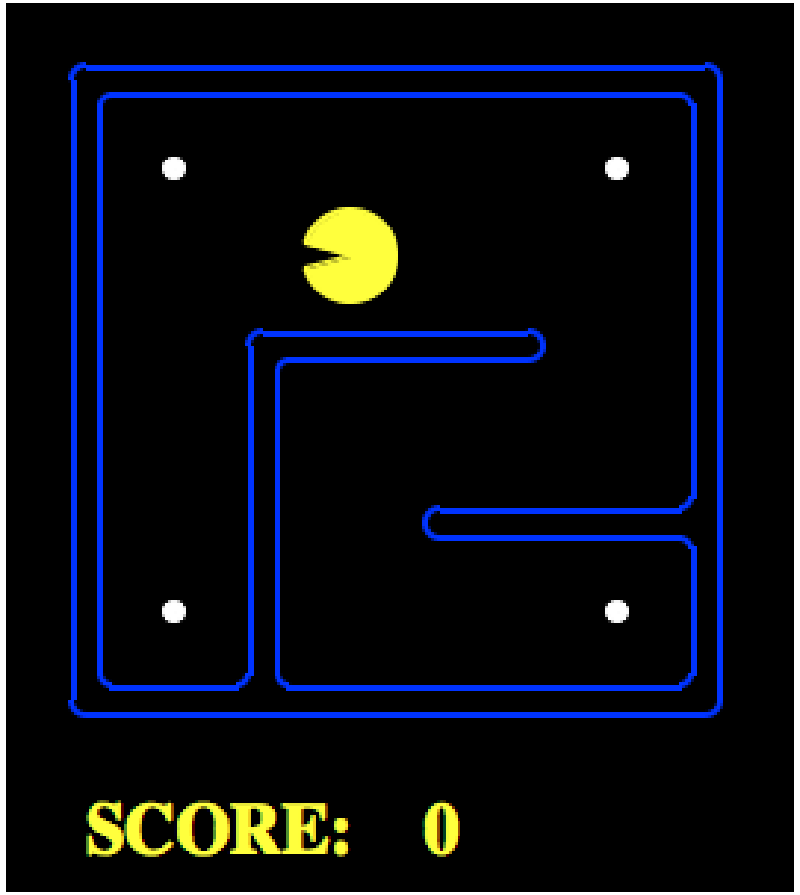
[문제2] CornerProblem 구현

- 지금까지는 주어진 환경만 사용
  (예: PositionSearchProblem)
- 새로운 문제에 대해서 환경 구성을 하는 문제
- 즉, SearchAgent.py 파일 안의
  CornerProblem 클래스 구현
- def __init__(self, startingGameState):
- def getStartState(self):
- def isGoalState(self, state):
- def getSuccessors(self, state):
- def getCostOfActions(self, actions):



[문제2] CornerProblem 구현

- 힌트 :
  기본적으로 state에는 pacman의 현재 위치
  + 현재 목표에 도달했는지 확인 가능해야함



* 4개의 코너 중 방문한 코너에 대한 정보 필요

- def isGoalState(self, state):

- def getSuccessors(self, state):

[문제3] cornersHeuristic 구현

- 힌트 :
  현재 state부터 Goal까지의 (예측)거리를 리턴

- def cornersHeuristic(state, problem):

- 기존에는 하나의 목표에 대한 거리만 계산

- 현재는 4개의 코너를 다 돌아야 목표 달성