인공지능
Homework 관련

2018-10-04

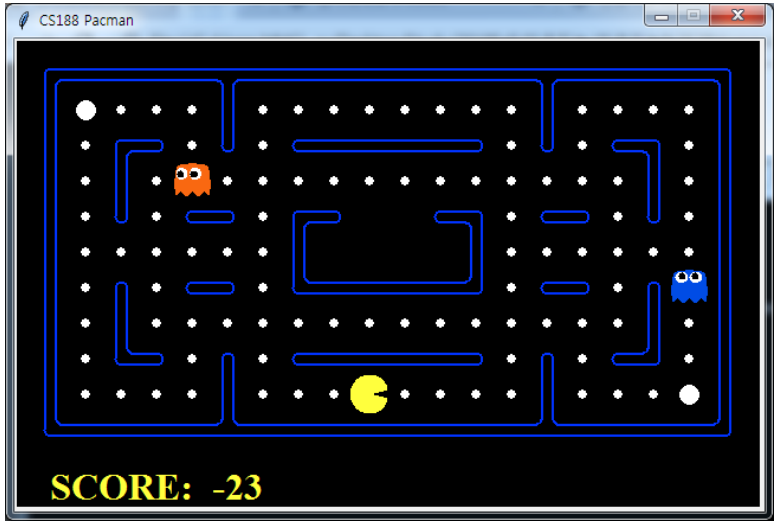
들어가기 전에

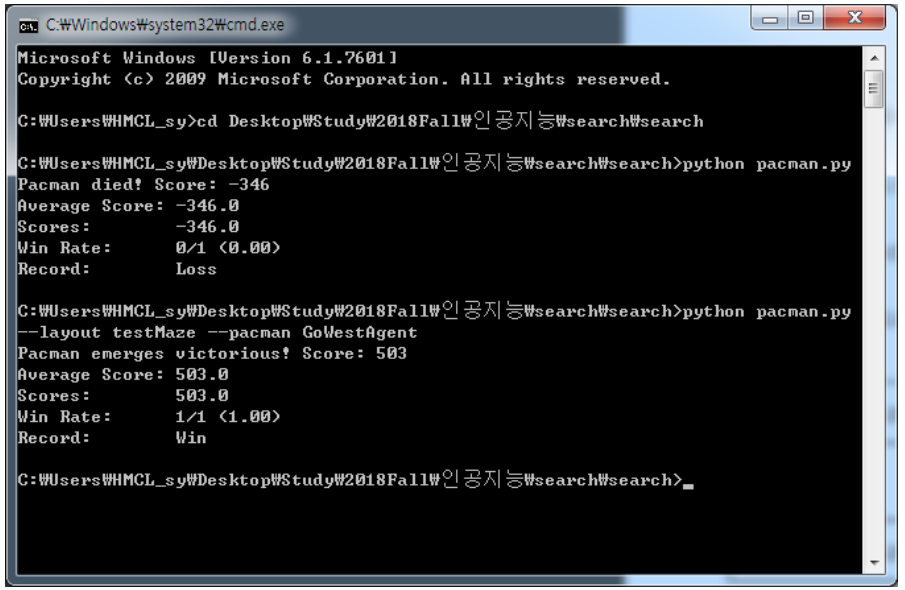
- PACMAN과제는 버클리 대학의 AI과목에서 사용하는 숙제프로그램입니다.

- 주의사항 : 해당 코드의 솔루션을 인터넷에 공개하는 것은 금지되어 있습니다.

HW0

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





HW1 (오늘 공지, 마감: 10/11 목)

- HW0 공지를 하면서, HW1 관련 내용도 같이 드렸습니다.
- 숙제는 DFS와 BFS를 구현하는 것입니다.
- 구현 자체는 쉬우나, PACMAN 프로그램이 어떻게 동작하는지 정보는 어디서 받아와야 하는지 등에 소스분석이 필요
- ** HW1 공지에 다음주에 나올 HW2 내용도 포함하였습니다.

PACMAN 소스 분석

- HW1 숙제는 search.py 파일만 수정

- class SearchProblem

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

PACMAN 소스 분석

class SearchProblm는 다음의 추상적인 함수를 제공

- def getStartState(self): # 시작 state를 반환함 (여기서 state는 (x,y) 좌표)
- def isGoalState(self, state): # 주어진 state가 GoalState인지 True/False 반환
- def getSuccessors(self, state): # 주어진 state에 다음 state관련 tuple의 리스트 반환
  # state관련 tuple : (state, action, cost)
- def getCostOfActions(self, actions): # actions들의 cost의 총합을 반환

실제 구현 예시는 searchAgent.py의 class PositionSearchProblem 참조

```
def getSuccessors(self, state):
    successors = []
    # 동, 서, 남, 북을 모두 체크
    for action in [Directions.NORTH, Directions.SOUTH, Directions.EAST, Directions.WEST]:
        x,y = state
        dx, dy = Actions.directionToVector(action)
        nextx, nexty = int(x + dx), int(y + dy)

        # 벽이 아니라면 (해당 방향으로 이동이 가능하다면)
        if not self.walls[nextx][nexty]:
            nextState = (nextx, nexty)
            cost = self.costFn(nextState)
            successors.append( ( nextState, action, cost) ) # (위치, 이동방향, 비용) 추가
        …
        return successors
```

<<getSuccessors의 활용>>

```
for next in problem.getSuccessors(state):    # next는 (위치, 이동방향, 비용)
    next_state     = next[0]
    next_direction = next[1]
    next_cost      = next[2]
```

PACMAN 소스 분석

Part 1 : DFS의 구현 << def depthFirstSearch(problem): >>

Q1 : search tree의 각 node에 어떤 정보가 포함되어야 하나?

(path 정보를 저장하기 위해서 어떻게 해야되나?)

Q2 : 기 방문한 node는 어떻게 점검할 것인가?

Q3 : search tree에서는 탐색할 다음 node를 선택이 알고리즘별로 차이가 있다.

DFS를 위해서는 어떤 자료구조를 이용해야하나?

(util.Stack(), util.Queue(), util.PriorityQueue())

* 자료구조 API는 util.py 체크 (push(), pop(), isEmpty(), …

PACMAN 소스 분석

```
def tinyMazeSearch(problem):
""" tinyMaze문제를 풀기위한 일련의 움직임(moves)를 리턴"""

    from game import Directions
    s = Directions.SOUTH
    w = Directions.WEST
    return  [s, s, w, s, w, w, s, w]
```

PACMAN 소스 분석

Part 1 : BFS의 구현 << def breadthFirstSearch(problem): >>

Q1 : search tree의 각 node에 어떤 정보가 포함되어야 하나?

(path 정보를 저장하기 위해서 어떻게 해야되나?)

Q2 : 기 방문한 node는 어떻게 점검할 것인가?

Q3 : search tree에서는 탐색할 다음 node를 선택이 알고리즘별로 차이가 있다.

BFS를 위해서는 어떤 자료구조를 이용해야하나?

(util.Stack(), util.Queue(), util.PriorityQueue())

* 자료구조 API는 util.py 체크 (push(), pop(), isEmpty(), …

PACMAN 소스 분석

- 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):