**COSE361 인공지능 중간 프로젝트**

기한: 2021년 5월 12일 수요일 오후 11시 59분

중간프로젝트는 A\* 알고리즘을 maze planner에 구현해보는 프로젝트입니다. 개별 프로젝트이며, 수업을 듣는 다른 학생과 상의하여 수행할 수 없습니다.

**Maze Planner 프로그램 실행하기**

Graphical user interface, diagram, application

Description automatically generated

* 코드를 다운받아 Pycharm에서 열고, MazePlanner.py 파일을 실행시킵니다.
* 중간은 알고리즘이 탐색하고자 하는 미로이며, 칸의 진하기는 cost 입니다.
* 왼쪽 패널에서 미로 설정을 바꿀 수 있습니다.
  + New Flat 버튼은 cost가 없는 미로를 생성합니다.
  + New Hilly 버튼은 cost가 랜덤으로 부여된 미로를 생성합니다.
  + Edit Maze 패널에서 시작 위치, 목표 위치, cost을 조정할 수 있습니다.
* Load/Save Maze에서 미로를 불러오거나 현재 미로를 저장할 수 있습니다.

**미로에서 탐색 알고리즘 실행하기**

Application

Description automatically generated

* 오른쪽 패널에서 탐색 알고리즘을 실행할 수 있습니다.
  + 사용하고자 하는 탐색 알고리즘을 선택합니다.
  + ‘Set Up Search’ 버튼을 클릭합니다.
  + ‘Step Search’는 탐색 과정의 한 단계를 보여줍니다.
  + ‘Run Search’는 전체 탐색 과정을 보여줍니다.
  + ‘Quit Search’는 미로의 마킹들을 제거하고 미로 설정 모드로 돌아갑니다.

**Q1. A\* 탐색은 구현되어 있지 않습니다. 중간 프로젝트는 이 알고리즘을 구현하여 Maze Planner 프로그램에 적용시키는 것입니다.**

* **구현해야 할 파일**
  + MazeStateAdvisors.py
    - class AStarMazeAdvisor
      * def \_setupInitialState
        + GreedyMazeAdvisor의 GreedyMazeAdvisor와 유사하지만 g와 h를 계산하여 AStarMazeState을 반환
      * def \_buildNeighbor
        + GreedyMazeAdvisor의 \_buildNeighbor와 유사하지만 A\*에 맞는 heuristic function 사용, MazeState대신 AStarMazeState을 반환
      * def \_calDistToGoal
        + GreedyMazeAdvisor의 \_ calDistToGoal와 동일
  + SearchSolver.py
    - class BestFirstSearchSolver
      * def searchStep
        + NoCostSearchSolver의 searchStep와 유사
* **위 함수들을 구현하기 위해 이해해야 할 클래스 및 함수**
  + MazeStateAdvisors > MazeState, MazeTaskAdvisor, UCSMazeAdvisor, AStarMazeState
  + SearchSolver.py > NoCostSearchSolver의 searchStep
  + MazePlanner.py > resetSearch
* **구현 시 주의사항**
  + 구현 전에 여러 미로에 프로그램을 실행해보고 (Depth-First알고리즘만 실행 가능) .py 파일을 읽어 전체 코드 구성을 숙지합니다.
  + 각 함수의 docstring을 읽고 숙지한 코드를 바탕으로 구현합니다.
  + UCSMazeAdvisor, GreedyMazeAdvisor을 읽고 이를 바탕으로 구현하는 것을 권장합니다.

**Q2. ‘Load Maze’ 버튼을 클릭하고 ‘bigmaze1.txt’ 미로를 로드하세요. 이 미로에 uniform-cost search, greedy best-first, depth-first, A\* 탐색 알고리즘을 적용해보세요.**

어느 탐색 알고리즘이

1) 가장 짧은 탐색 시간

2) 가장 긴 탐색 시간

3) 가장 좋은 path을 찾았는지 탐색 결과를 제시하고 이를 바탕으로 서술하세요.

(탐색 결과: total path cost, path length, nodes created, nodes visitied)

결과물 제출 사항

* Q1 코드는 MazeStateAdvisors.py와 SearchSolver.py 파일만 제출
* Q2 작성은 새 문서나 이 문서에 추가하여 작성하던 자유롭게 작성 가능 (한글/워드/PDF 모두 가능)

채점 기준

* smallmaze2.txt 미로에 각 알고리즘이 탐색하고 찾은 cell의 정답과 비교
* Uniform Cost Search
  + Chart, treemap chart

    Description automatically generated
* Greedy Best-First
  + A screenshot of a computer game

    Description automatically generated with low confidence
* Depth-First
  + A screenshot of a game

    Description automatically generated with medium confidence
* A-Star Search
  + Chart, treemap chart

    Description automatically generated