**COSE361 인공지능 실습 1**

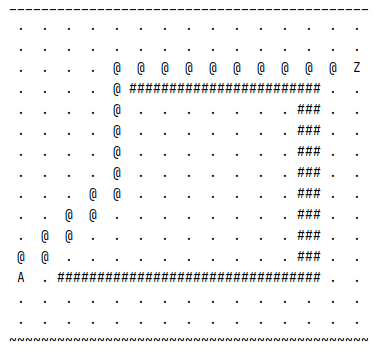
제출기한:2021년 4월 1일 목요일 오후 1:59PM 까지

(2:00PM 제출시 지각 제출로 간주)

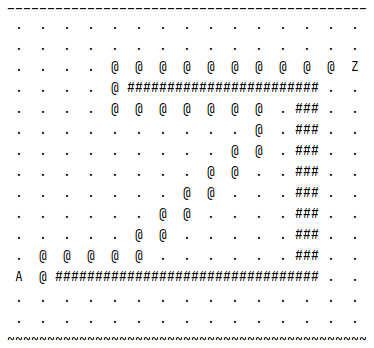
학번: 2016160311

이름: 이재윤

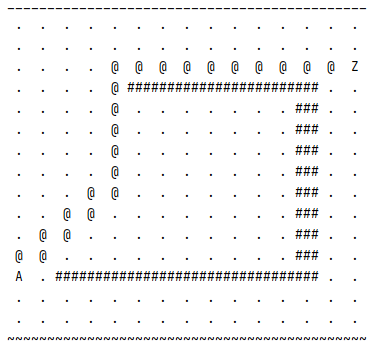
1. Euclidean distance을 계산하는 함수 heuristic\_euclidean를 생성하시오.
2. greedy\_best\_first\_search, a\_star\_search의 heuristic 함수를 1번에서 만든 함수로 교체하시오.
3. grid3 그리드에 dijkstra\_search, greedy\_best\_first\_search, a\_star\_search를 적용하고 draw\_grid 함수를 이용하여 그리드를 출력하시오. (start3, goal3 사용)
4. 위 코드에서 생성된 각 path의 스크린샷을 첨부하시오.
5. dijkstra\_search



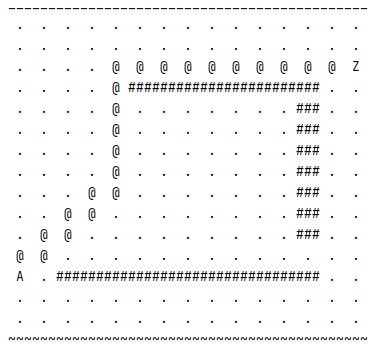
1. greedy\_best\_first\_search



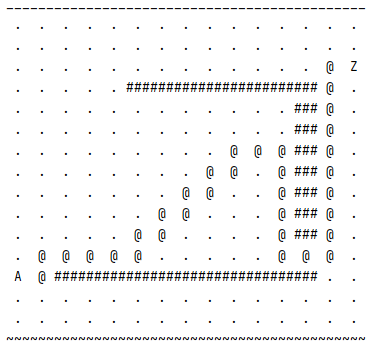
1. a\_star\_search



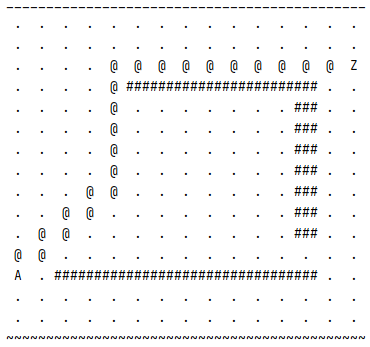
1. implementation.py 파일의 grid3 그리드 코드를 복사하여 grid4를 생성하시오. grid4는 디귿자인 벽 중 한군데에 구멍이 뚫려 있어야 합니다. (구멍 위치 – 177 인덱스)
2. grid4 그리드에 dijkstra\_search, greedy\_best\_first\_search, a\_star\_search를 적용하고 draw\_grid 함수를 이용하여 그리드를 출력하시오. (heuristic 함수 Euclidean, start3, goal3 그대로 사용)
3. 위 코드에서 생성된 각 path의 스크린샷을 첨부하시오.
4. dijkstra\_search



1. greedy\_best\_first\_search



1. a\_star\_search



1. 위에서 밝혀진 Dijkstra’s Algorithm, Greedy Best-First Search, A\* Search을 통해 찾아진 path들을 optimality 관점으로 비교 및 분석하시오.

* Dijkstra’s Algorithm과 A\* Search는 optimality가 보장되므로 주어진 조건 내에서 최단거리의 path를 찾아내어 나타내고 있지만 Greedy Best-First Search의 경우 optimality가 보장되지 않기 때문에 시작지점에서 목표지점까지 가는 path가 최소 거리가 되지 않는 것을 확인할 수 있다. 주어진 조건에 따르면 path의 cost가 전부 동일하므로 Dijkstra’s Algorithm으로 구한 경로가 A\* Search로 얻은 경로와 동일하게 나오는 것을 확인할 수 있다. 벽에 구멍을 뚫었을 때, Greedy Best-First Search의 경로는 구멍을 뚫기 전보다는 더 빠른 경로를 찾아내었다. 하지만 Dijkstra’s Algorithm과 A\* Search의 경로는 구멍을 뚫기 전과 후의 가장 optimal한 경로가 동일하기 때문에 경로가 변하지 않았음을 확인할 수 있다.