

Assignmant01 – Maze Game 결과보고서

전공: 컴퓨터공학

학년: 3학년

학번: 20171602

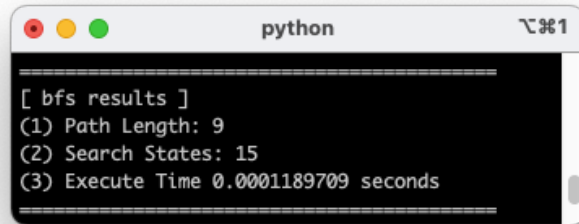
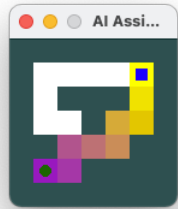
이름: 강지혁

1. Stage 1

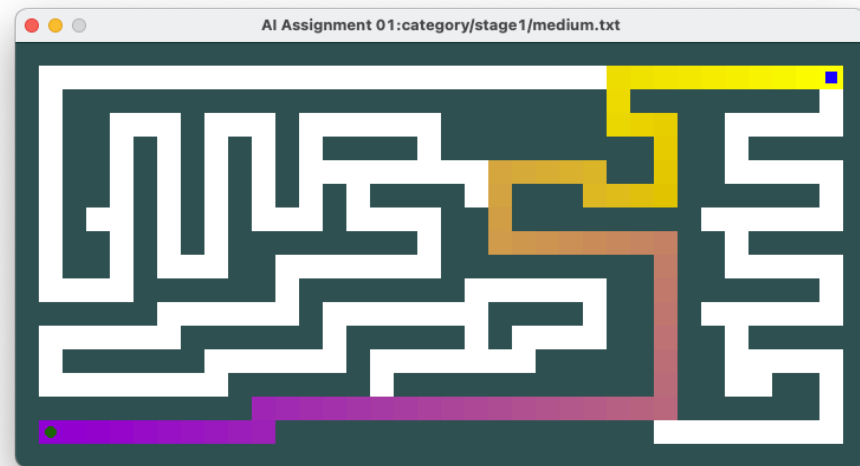
1.1 BFS algorithm

1차원 리스트인 bfsQueue를 선언하여 방문할 노드를 insert, 방문한 노드를 delete 해 가며 isObjective 값이 True가 될 때까지 너비 우선 탐색을 수행하는 알고리즘이다. isObjective가 True가 되면 경로를 path 리스트에 저장하고 return한다.

A. small.txt --method bfs

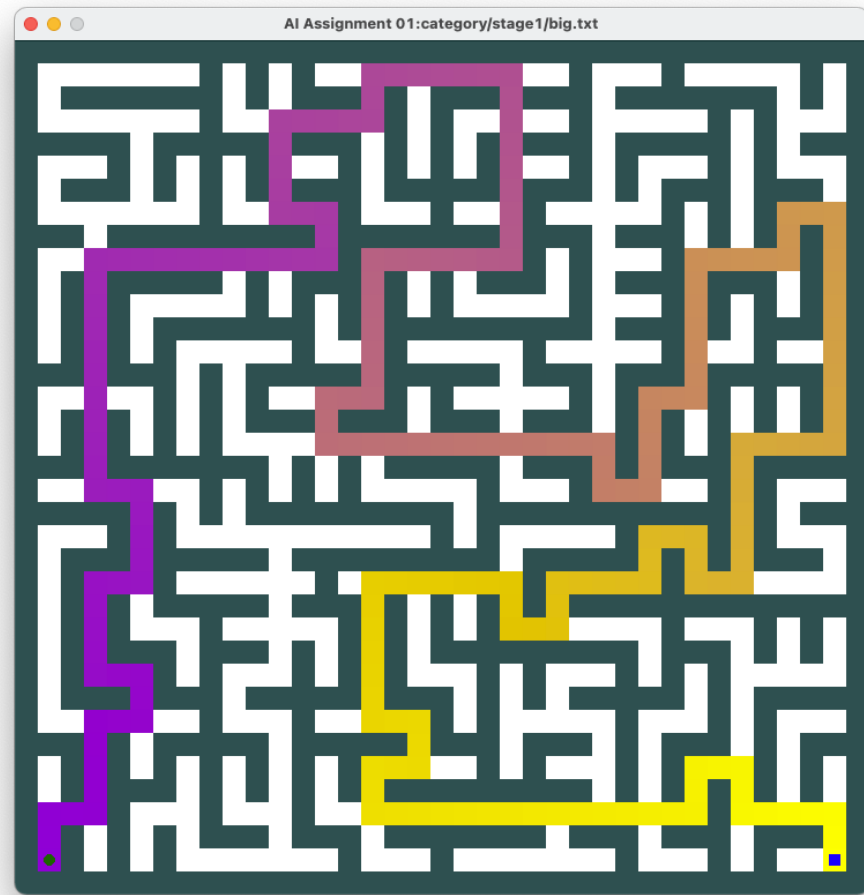


B. medium.txt --method bfs



```
python 1
=====
[ bfs results ]
(1) Path Length: 69
(2) Search States: 269
(3) Execute Time 0.0018899441 seconds
=====
```

C. big.txt --method bfs

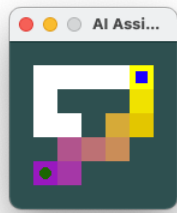


```
python
=====
[ bfs results ]
(1) Path Length: 211
(2) Search States: 619
(3) Execute Time 0.0028967857 seconds
=====
```

1.2 A* algorithm

지금까지 진행한 거리와 남은 거리의 Manhattan Distance 값의 합을 F로 하는 Heuristic Function을 사용한 A* 알고리즘이다. loop를 순회할 때마다 방문하지 않은 neighbor를 탐색하며 neighbor 중에서 Heuristic Function 값이 가장 작은 node를 경로로 선택한다.

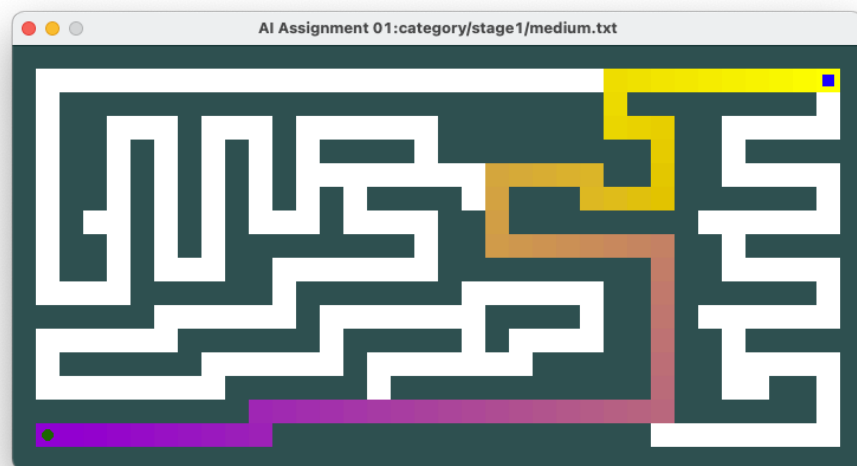
A. small.txt --method astar



```
python 1

[ astar results ]
(1) Path Length: 9
(2) Search States: 14
(3) Execute Time 0.0005011559 seconds
```

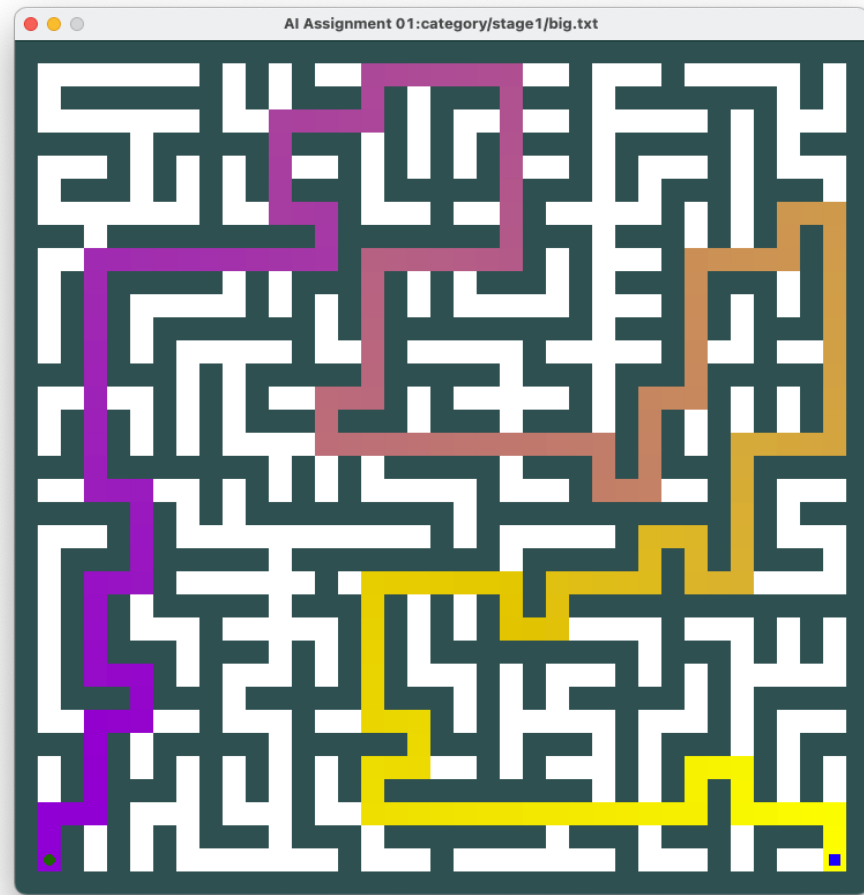
B. medium.txt --method astar



```
python 1

[ astar results ]
(1) Path Length: 69
(2) Search States: 244
(3) Execute Time 0.0053970814 seconds
```

C. big.txt --method astar



```
python
=====
[ astar results ]
(1) Path Length: 211
(2) Search States: 570
(3) Execute Time 0.0183558464 seconds
=====
```

2. Stage 2

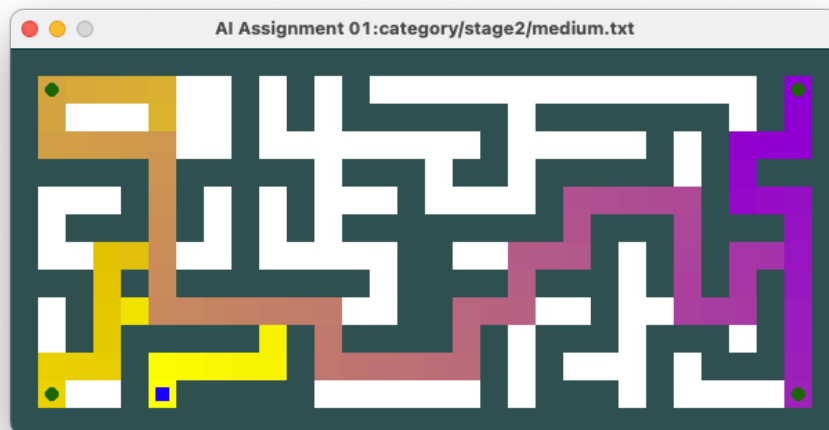
Agent의 현재 위치로부터 네 모서리의 end_points까지 Manhattan Distance 값 중 최솟값과 지금까지 진행한 거리의 합을 F로 하는 Heuristic Function을 사용한 A* 알고리즘이다. loop를 순회할 때마다 방문하지 않은 neighbor를 탐색하며 neighbor 중에서 Heuristic Function 값이 가장 작은 node를 경로로 선택한다. 특정 end_point에 도착하면 모든 end_points를 찾았는지 확인하고 아직 남은 end_point가 있으면 지금까지의 경로를 저장하고 다시 진행한다.

A. small.txt --method astar_four_circles



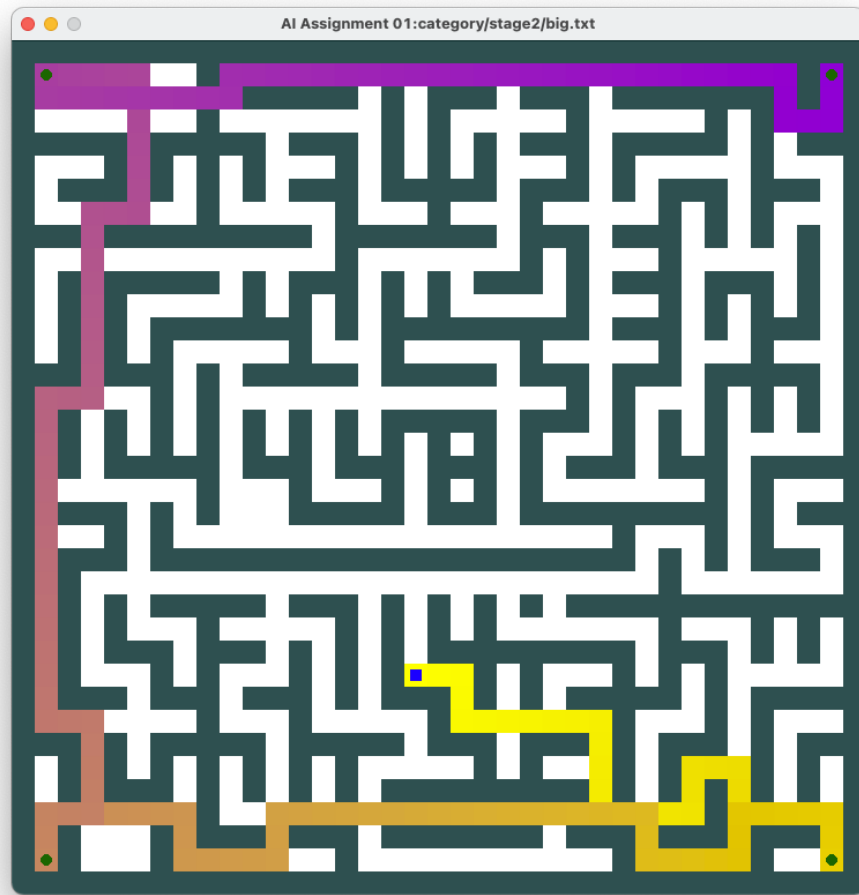
```
python
=====
[ astar_four_circles results ]
(1) Path Length: 36
(2) Search States: 99
(3) Execute Time 0.0014111996 seconds
=====
```

B. medium.txt --method astar_four_circles



```
python
=====
[ astar_four_circles results ]
(1) Path Length: 110
(2) Search States: 3719
(3) Execute Time 0.0566320419 seconds
=====
```

C. big.txt --method astar_four_circles

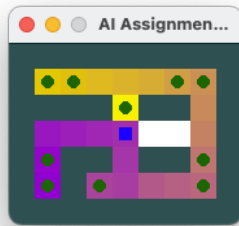


```
python
=====
[ astar_four_circles results ]
(1) Path Length: 166
(2) Search States: 685
(3) Execute Time 0.0130188465 seconds
=====
```

3. Stage 3

Minimum Spanning Tree을 이용한 Heuristic Function을 구현하지 못해 Stage 2에서 사용한 Heuristic Function을 이용해 A* 알고리즘을 구현하였다. Agent의 현재 위치로부터 모든 end_points까지 Manhattan Distance 값 중 최솟값과 지금까지 진행한 거리의 합을 F로 사용한다. loop를 순회할 때마다 방문하지 않은 neighbor를 탐색하며 neighbor 중에서 Heuristic Function 값이 가장 작은 node를 경로로 선택한다. 특정 end_point에 도착하면 남은 end_points가 존재하는지 확인하고 아직 남은 end_point가 있으면 지금까지의 경로를 저장하고 다시 진행한다.

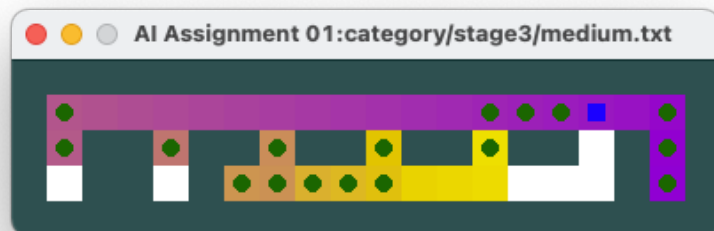
A. small.txt --method astar_many_circles



```
python

=====
[ astar_many_circles results ]
(1) Path Length: 37
(2) Search States: 47
(3) Execute Time 0.0020461082 seconds
=====
```

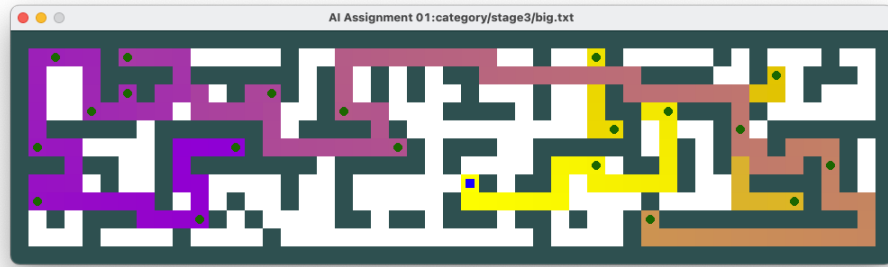
B. medium.txt --method astar_many_circles



```
python

=====
[ astar_many_circles results ]
(1) Path Length: 65
(2) Search States: 82
(3) Execute Time 0.0023839474 seconds
=====
```

C. big.txt --method astar_many_circles



```
python
```

```
[ astar_many_circles results ]  
(1) Path Length: 247  
(2) Search States: 859  
(3) Execute Time 0.0211901665 seconds
```