Assignment 3

1. capture the result of `pacman.py` with layout `test71.lay`

```
$ python ./pacman.py --agent MyAgent --layout test71 -c
Pacman emerges victorious! Score: 764
Average Score: 764.7257232666013
Scores:        764.7257232666013
Win Rate:      1/1 (1.00)
Record:        Win
(2023AI)
```

2. Description of your agents.
   - 반복적인 계산을 줄인 BFS
   - 반복적인 계산을 줄였다는 것은 매 스탭마다 경로를 재계산하지 않고, 현재 목표 위치가 없거나, 목표 위치에 있던 점이 사라진 경우에만 다시 경로를 계산한다는 것을 의미한다.
3. Three discussions when playing Pacman
   - Discuss cases where the agent implemented by yourself is better than the baseline.
     - `test71`을 비롯한 많은 케이스에서, `MyAgent`는 `ClosestDotAgent`보다 높은 성능을 보였다. `ClosestDotAgent`는 우선순위 큐 기반으로 동작하지만, `MyAgent`는 FIFO 큐 기반으로 동작하여 오버헤드가 더 적기 때문이다. 우선순위 큐는 원소의 삽입과 삭제에 $O(\lg n)$의 복잡도를 지니지만, FIFO 큐는 $O(1)$의 복잡도를 지닌다.
     - 비용이 모두 동일한 영역에서 우선순위 큐를 이용한 UCS의 경로는 BFS의 그것과 동등하다.
     - 따라서, `ClosestDotAgent`와 `MyAgent`는 같은 해(최적의 해)를 찾아내지만, `MyAgent`가 더 빠르다.
   - Discuss cases where the agent implemented by yourself is worse than the baseline.
     - 전술한 것처럼, 비용이 모두 동일한 영역에서 UCS와 BFS의 경로는 모두 최적이다.

    - 계산 시간에 있어서 BFS는 UCS에 대해 무조건적인 우위(Fringe의 삽입, 삭제 비용)를 지닌다.
    - 따라서 이동 비용이 모두 동일하다면, BFS는 UCS보다 항상 더 낫다. 만약 이동 비용이 동일하지 않다면 BFS는 최적의 해를 찾지 못하므로 UCS가 더 나을 수 있을 것이다.
- 게임에 대한 정보가 주어지는 상황에서, A* 알고리즘과 같은 Informed Search 대신 Uninformed Search(BFS)를 사용한 이유가 무엇인가?
    - 비록 게임에 대한 정보가 주어진다고 해도, 해당 내용을 기반으로 휴리스틱을 계산하는 데에는 비용이 들어간다. 이를테면 가장 가까운 점으로의 맨해튼 거리를 구한다고 하면 점들을 순회하면서 가장 가까운 점을 선택하고 그 점까지의 거리를 구하는 오버헤드가 있다. 동시에, A* 알고리즘을 사용하기 위해서는 우선순위 큐를 사용해야 하고, 전술했듯 우선순위 큐에도 오버헤드가 존재한다.