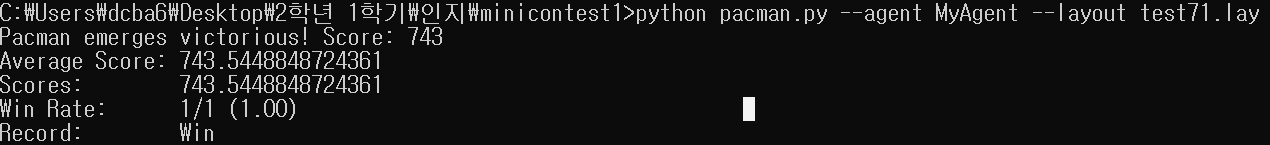
COSE361(03) - Final project 1/2

2021320122 김정우

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



1. Description of your agents. (2 points)

구동 방식 : bfs를 통해 가장 가까운 food을 찾습니다. 단, 이미 다른 agent가 향하고 있는 food는 가깝더라도 제외합니다. 이는 단순하지만 강력한 두 가지의 장점을 가진 코드입니다.

첫 번째 장점은 여러 pacman이 중복된 food를 향하지 않기 때문에 무리 지어 움직이는 듯한 비효율적인 상황을 예방할 수 있다는 점입니다. 이 때문에 여러 agent는 map을 잘 나누어 활동하게 됩니다.

두 번째 장점은 각 agent들이 한 food를 정하기만 하면, 그 food로 향하는 path를 그대로 따라가기만 하면 되기에 bfs의 쓸데없는 중복 사용을 막을 수 있다는 점입니다.

* 1. Initialize

텍스트이(가) 표시된 사진

자동 생성된 설명

Initialize 부분입니다. Pacman의 수가 8개를 넘지 않는 것으로 보여 크기를 list 크기들을 8로 설정했습니다.

Check는 각 agent들이 향하고 있는 목적지 (x,y)들을 담고 있습니다. 예를 들어 1번 agent가 (1,3)에 있는 food를 향하고 있다면 check[1]에는 (1,3)이 저장되어 있는 방식입니다.

Path\_는 말 그대로 각 check까지 향하는 path입니다.

Numpath는 각 agent들이 path\_배열의 몇 번째까지 이동했는지 나타내는 값들입니다. 예를 들어 1번 agent의 path\_가(즉 path\_[1]이) [NORTH, NORTH, EAST]이고 두 칸째 이동했다면, numpath[1]은 2가 됩니다.

* 1. getAction

기본적으로 bfs를 사용하기 때문에, search.py에 있는 breadth-first-search 코드를 대부분 그대로 차용했습니다.

그러나 앞서 설명드린 두 번째 장점을 활용하기 위해 bfs 이전에 다음과 같은 코드를 추가합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Numpath < len(path\_)의 의미는, 아직 목적지까지의 path가 끝나지 않았음을 의미합니다.

만약 저 if문의 값이 false가 되면 목적지에 도달한 것이기에 새로운 목적지를 찾아줘야 합니다. Check를 (-1,-1)로 초기화해주고, bfs를 실행합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

원본과 다른 점은 첫 번째 조건문, 그리고 return 직전에 있는데요, 첫 번째 조건문에서는 다른 agent의 목적지가 아님을 확인하기 위해 node not in check가 추가되었습니다.

만약 목적지가 겹친다면 저 if문을 통과하지 못하고 새로운 목적지를 찾게 될 것입니다.

마지막에는 numpath를 재설정하고 path\_에 path를 복사하는 과정이 있습니다.

1. Three discussions when playing Pacman
   1. Discuss cases where the agent implemented by yourself is better than the baseline.

앞서 설명드렸듯, baseline에 없는 두 가지의 큰 장점이 있습니다. Baseline에서는 매 순간마다 search를 해 줘야 하는 엄청난 비효율성이 있는데요, 제 코드에서는 한번 목적지가 정해지면 그 경로 내에서는 추가적인 search가 필요 없습니다.

또한 baseline에서는 여러 agent가 붙어있는 경우 가까운 dot을 향해 무리지어 행동하는 듯한 현상이 있는데, 이것 역시 해결했습니다.

* 1. Discuss cases where the agent implemented by yourself is worse than the baseline.

엄청나게 멀리 있는 agent가, 그 주변에서 dot이 전부 사라지면 멀리 있는 dot을 목적지로 설정하게 됩니다. 그 agent가 도달하기 전까지, ‘점 찍어둔’ dot은 다른 agent들이 코앞에 있더라도 무시하게 됩니다. 이러한 현상이 발생하는 경우 baseline보다 비효율적이라고 할 수 있습니다.

* 1. Ask & Answer your own question about the above discussion.

그렇다면 아주 작은 임의의 Manhattan distance를 설정하고, 매 순간 agent들이 그 범위 내를 우선적으로 bfs 탐색하게 하면 효율성이 늘어날까 하는 의문이 듭니다. 그러나 그렇게 구현할 경우 너무 특정한 case에 대한 효율성만 늘어나는 꼴이 될 것 같고, check list에 있는 food들이 아직도 남아있는지 확인하는 과정도 필요해집니다. 따라서 좋은 해결 방안이 아니라고 결론을 내렸습니다.