[Part 0]

해당 위치를 식별하기 위해 x, y 값을 가지는 셀 클래스 유형의 2차원 배열을 사용하여 그리드를 나타낸다.
블록은 무작위로 그리드에 배치되어 약 30%를 차지한다.

[Part 1]

a) 에이전트 월드는 블록을 알지 못하기 때문에 완전히 차단되지 않는다. 따라서 A*를 실행하면 오른쪽으로 직선인 가장 짧은 추정 비차단 경로를 반환한다. A*는 맨해튼 거리를 휴리스틱으로 사용하여 이 경로를 반환한다.

b) 대상에 도달하거나 불가능하다는 것을 발견할 때까지 에이전트의 이동 횟수는 위에서부터 차단되지 않은 셀 제곱으로 제한된다. 차단되지 않은 셀 제곱은 에이전트가 수행할 수 있는 총 이동 횟수를 제공한다. 각 A* 실행 후 에이전트가 수행할 수 있는 이동 수는 최대 총 셀 수이므로 이동 수(m)는 총 셀 수(n)보다 작거나 같아야 한다. A*를 반복적으로 호출하기 때문에 최대 총 셀 수와 같을 수 있으므로 반복(i)이 총 셀 수(n)보다 작거나 같아야 한다. 두 부등식을 결합하면 반복 A*가 에이전트가 수행하는 총 이동 수에 의해 위에서 제한된다는 것을 알 수 있다.

[Part 2]

낮은 g-값을 선호하는 관계 단절 시간 분석
총 실행시간: 약 7초
평균: 약 0.15초
평균 경로 길이: 155

높은 g-값을 선호하는 관계 단절 시간 분석
총 실행시간: 약 5초
평균: 약 0.09초
평균 경로 길이: 155

낮은 g-값 대신 높은 g-값을 선호하여 연결을 끊음으로써 평균적으로 더 나은 실행 시간을 갖게 된다. 이는 높은 g-값을 선호하여 셀를 탐색할 때 에이전트가 더 많은 셀를 바깥쪽으로 탐색할 수 있기 때문에 최종 상태에 더 가깝게 이동할 수 있기 때문이다. 낮은 g-값을 선호하여 셀을 탐색할 때 에이전트는 시작 상태에 더 가까운 셀을 탐색한다. 더 작은 g-값을 선호하여, 시작 상태 근처의 많은 불필요한 셀을 확장하는 것에 대해 종료 상태를 향해 확장하는 셀을 확장한다.

[Part 3]

반복 전위 A* 반복 시간 분석
총 실행시간: 4.1 seconds
평균: 0.09 seconds
평균 경로 길이: 155

반복 후위 A* 반복 시간 분석
총 실행시간: 35.44 seconds
평균: 0.8 seconds
평균 경로 길이: 160

반복 전위 A*가 반복 후위 A*보다 상당히 빠르다. 이는 반복 후위 A*가 구현되는 방식 때문이다. 반복 후위 A*는 끝에서 시작까지의 경로를 찾지만, 우리는 그 경로를 처음부터 끝까지 통과한다. 이로 인해 반복 후위 A*는 시작에 가까운 곳에서 더 많은 셀을 이동하고 확장할 수 있으며, 이는 반복 전위 A*에 비해 필요한 것보다 비효율적이다.

[Part 4]

맨하탄 거리는 시작 좌표와 종료 좌표의 차이의 절대값으로 정해진다. 이는 에이전트를 위 또는 아래, 왼쪽 또는 오른쪽으로 이동하여 시작에서 끝까지의 거리를 계산하는 것과 같다. 이 거리가 일정하지 않을 수 있는 유일한 방법은 에이전트가 두 사각형 사이에서 대각선으로 이동하는 방법이며, 이는 허용되지 않는 방법이다.

맨해튼 거리가 일정하지 않다고 가정하고, 4방향 중 1방향으로 이동하는 비용을 cost(n, a, n′), n에서 목표로 이동하는 비용을 h(n')는 이며 h(n)는 n에서 목표로 이동하는 비용입니다.

$h(n) \geq cost(n, a, n') + n(m)$

여기서 비용(n, a, n')은 4방향 중 1방향으로 이동하는 비용이고 h(n)는 n에서 목표로 이동하는 비용이며 h(n')는 n'에서 목표로 이동하는 비용이라고 한다.

$h(n) - h(n')$는 $| x(n) - x(n') | + | y(n) - y(n') |$ 이므로
$|x(n) - x(n') | + | y(n) - y(n') | \geq cost(n, a, n')$ 이다.

대각선 방향으로 이동하면 맨해튼 거리로는 불가능하다. 따라서 맨해튼 거리는 일정하다.

적응형 A*는 작업 비용이 증가할 수 있더라도 초기에 일관된 h-값을 유지한다. 이는 A*의 연속적인 반복에서 h-값을 계산하는 방법 때문이다.

적응형 A*는 다음과 같은 휴리스틱 기능을 사용한다.
$h(s) = g(target) - g(s) \Rightarrow h(n) \leq cost(n, a, n') + h(n')$

휴리스틱 함수를 대체하여 단순화하면
$g(target) - g(n) \leq cost(n, a, n') + g(target) - g(n') \Rightarrow g(n') - g(n) \leq cost(n, a, n')$

맨하탄 거리를 사용하여 g(n') - g(n)를 계산하면 n'과 n 사이의 거리를 얻을 수 있다. 이 값은 비용(n, a, n')과 동일하다. 따라서 새로운 휴리스틱인 h(s) = g(target) - g(s)도 일관되게 적용가능하다.

[Part 5]

반복 전위 A* 반복 시간 분석
총 실행시간: 4.1 seconds
평균: 0.09 seconds
평균 경로 길이: 155

적응형 A* 반복 시간 분석
총 실행시간: 4 seconds
평균: 0.085 seconds
평균 경로 길이: 158

적응형 A*는 반복 전진 A*보다 상당히 빠르다. 적응형 A*는 향상된 휴리스틱 계산을 사용하기 때문이다. 적응형 A*는 이미 방문한 셀을 고려하기 때문에 End 상태를 더 빨리 찾는 업데이트된 h-cost를 계산한다. 이것은 또한 적응형 A*가 반복 전위 A*보다 더 적은 셀을 확장하는 이유이기도 하다.

[Part 6]

가설)
귀무 가설은 두 검색 알고리즘의 성능에 차이가 없다는 것.
대안 가설은 두 검색 알고리즘의 성능에 차이가 있다는 것.

이 가설을 검정하는 데 사용할 수 있는 통계 검정은 쌍체 t-검정인데, 쌍체 t-검정은 쌍체 표본 두 개의 평균을 비교하는 데 사용되며, 이 경우 쌍을 이룬 두 표본은 동일한 테스트 사례 집합에 대한 두 검색 알고리즘의 결과가 된다. 유의 수준은 귀무 가설이 참일 때 귀무 가설을 기각할 확률이고, 공통 유의 수준은 0.05이며, 이는 귀무 가설이 참일 때 귀무 가설을 기각할 확률이 5%임을 의미한다. 쌍체 t-검정은 귀무 가설이 참일 경우 두 검색 알고리즘 간에 관측된 성능 차이를 얻을 확률을 계산하는 데 사용할 수 있다. 이 확률이 유의 수준보다 작으면 귀무 가설이 기각될 수 있으며 두 검색 알고리즘의 성능에 차이가 있다는 결론을 내릴 수 있다. 귀무 가설이 기각되면 두 검색 알고리즘의 성능에 차이가 있다는 결론을 내릴 수 있다. 그러나 통계 검정은 성능에 차이가 있다는 증거만 제공한다는 점에 유의해야 합니다. 성능의 차이가 두 검색 알고리즘 간의 체계적인 차이 때문이라는 것을 증명하지는 않는다.

예시)
우리가 100개의 테스트 사례 세트에서 두 개의 검색 알고리즘인 A와 B의 성능을 비교하는 데 관심이 있다고 가정해 보자. 동일한 테스트 사례 집합에서 각 검색 알고리즘을 실행하고 각 테스트 사례에 대한 솔루션을 찾는 데 걸리는 시간을 기록한다.

검색 알고리즘 A가 솔루션을 찾는 데 걸리는 평균 시간은 10초.
검색 알고리즘 B가 솔루션을 찾는 데 걸리는 평균 시간은 15초.

우리는 null 가설이 참일 경우 쌍체 t-test를 사용하여 두 검색 알고리즘 간에 5초의 성능 차이를 얻을 확률을 계산할 수 있다. 이 성능 차이를 얻을 확률이 유의 수준 0.05보다 작으므로 귀무 가설을 기각할 수 있다. 검색 알고리즘 A가 검색 알고리즘 B보다 빠르다는 점에서 두 검색 알고리즘의 성능에 차이가 있다는 결론을 내릴 수 있다.