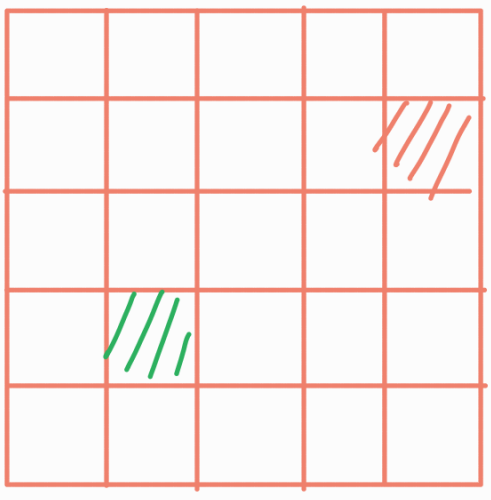
A* 알고리즘



f = total cost

g = cost from start

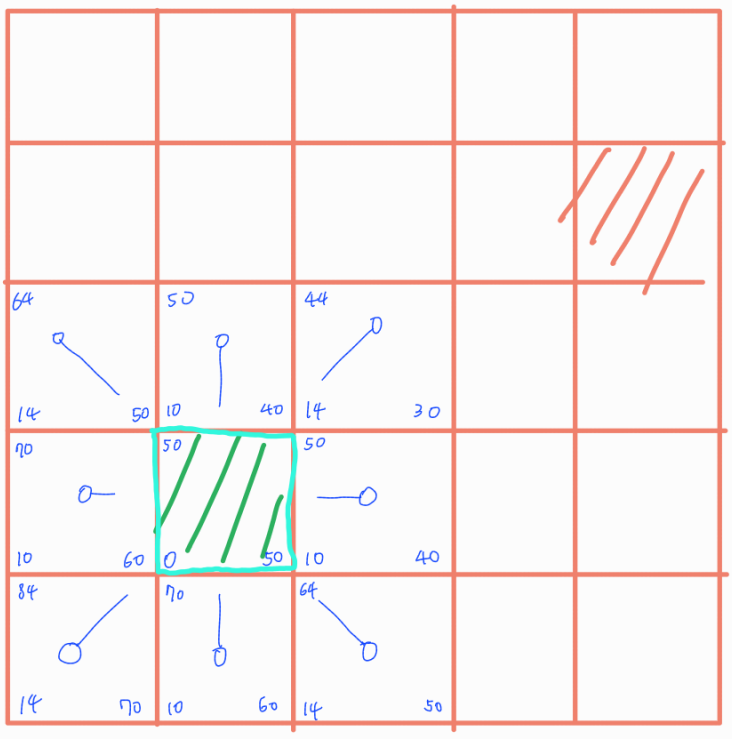
h = cost to end

= Start = End

여기서 g(n)은 출발 노드에서 현재 노드까지의 거리인데, 각 사각형의 가로/세로 이동당 10, 대각선 이동을 14로 설정한다. (대각선이 14인 이유는 피타고라스 정리에따라 대각선의 길이는 약 1.414배이기 때문에 정확히는 14.14이지만, 계산의 편의를 위해 14로 가정한다.

h(n)은 다양한 측정 방식을 사용할 수 있는데 여기서는 맨하탄 거리를 사용한다. 이는 대각선 이동이 불가능 하고 오직 가로/세로로만 이동 가능하다고 가정한다. h(n)은 현재 노드에서 도착 노드까지의 맨하탄 거리를 값으로 가진다.

f(n)은 g(n)과 h(n)을 더한 총 비용이다.



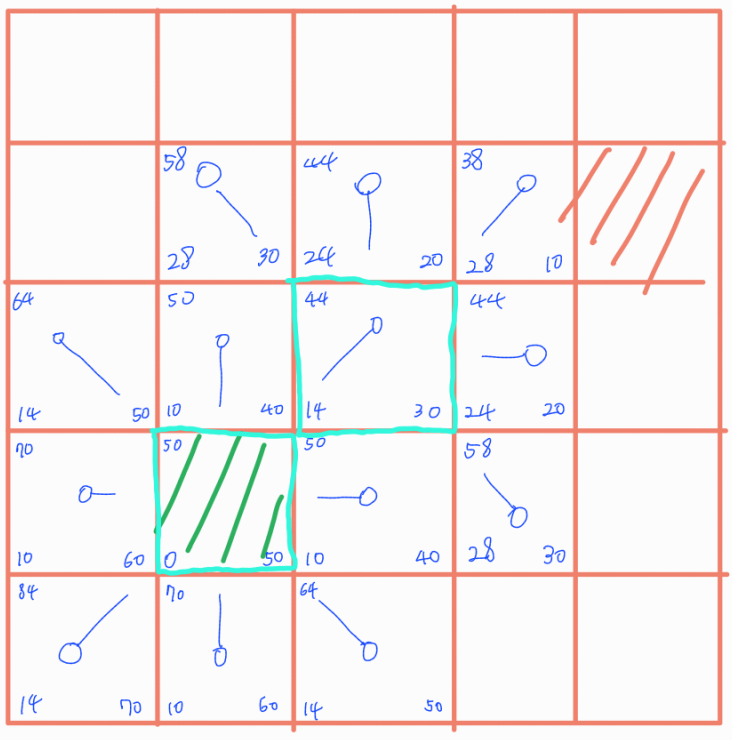
경로 탐색 1

1. 출발 사각형(초록색 노드)를 '열린 목록'에 넣는다.
2. 출발 사각형에 인접한 장애물은 무시하고, 지나갈 수 있는 사각형을 '열린 목록'에 추가한다. 이 사각형들은 출발 사각형을 부모로 지정한다. 이 사각형들은 출발 사각형을 부모로 지정한다. (부모 노드는 경로를 다 찾고 거슬러 올라갈 때 사용된다.) 출발노드를 비롯한 열린 목록에 있는 사각형들의 f(n), g(n), h(n)을 계산해 각 노드에 기입해준다.
3. '열린 목록'에서 출발 사각형을 없애고 '닫힌 목록'에 추가해 준다. 위 그림에서 닫힌 목록에 들어간 노드들은 하늘색 윤곽선으로 표현한다.

위의 과정을 거치면 위 그림과 같은 형태가 된다.
하늘색 윤곽선 테두리는 출발 노드가 '닫힌 목록'에 추가되어 더이상 볼 필요 없다는 것을 의미한다.

경로 탐색2

4. 탐색을 계속하기 위해 가장 작은 f(n)을 가지고 있는 사각형을 선택한다.
5. 선택한 사각형(노드)을 '열린목록'에서 배고 '닫힌 목록'에 넣어준다.
6. 인접한 사각형을 확인한다.



경로 탐색2

4. 탐색을 계속하기 위해 가장 작은 f(n)을 가지고 있는 사각형을 선택한다.
5. 선택한 사각형(노드)을 '열린 목록'에서 빼고 '닫힌 목록'에 넣어준다.
6. 인접한 사각형을 확인한다. 인접 사각형 중에 '닫힌 목록'에 있거나 장애물인 것들을 제외하고, 나머지 '열린 목록'에 사각형이 없다면 '열린 목록'에 추가한다. 그리고 현재 사각형을 '열린 목록'에 새롭게 추가된 사각형들의 '부모'로 만든다. 이때 열린 목록에 있는 사각형들의 f(n), g(n), h(n)을 계산해 노드에 기입한다.
7. 인접한 사각형이 이미 기존의 '열린 목록'에 있다면 현재 사각형을 기준으로 해당 인접한 사각형까지 이동할 때 g(n)비용이 낮아지는 지 확인한다. 비용이 낮아지지 않는다면 아무것도 하지 않는다. 만약 현재 사각형을 통해 해당 인접 사각형까지 이동하는 g(n)이 더 낮다면, 해당 인접 사각형의 부모노드를 현재 사각형으로 바꾼다.(다익스트라와 같은 듯?) 해당 인접 사각형의 f(n), g(n)을 다시 계산한다.
8. 위 과정이 끝나면 '열린 목록'에 있는 사각형들 중 f(n)이 가장 낮은 사각형을 선택한다. 그리고 다시 경로 탐색 중 빨간색 목표노드가 '열린 목록'에 추가되거나 '열린 목록이 비어있게 될 때 까지 (4)~(7) 과정을 반복한다.
9. 빨간색 목표노드가 '열린 목록'에 추가되면 목표 사각형으로부터 각각의 사각형의 부모 사각형을 향하여 시작 사각형에 도착할때 까지 거슬러 올라간다.