2024학년도 1학기 중간과제물(온라인 제출용)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **교과목명** | : | 인공지능 |
|  | **학번** | : | 202034-153746 |
|  | **성명** | : | 이동열 |
|  | **연락처** | : | 010-5264-5565 |

EMB000023580f65 ※ A4용지 편집 사용

(가).

상태 공간 탐색 방법은 문제에서 답을 찾아갈 때 발생할 수 있는 모든 상황을 상태로 정의하고 각 상태에서 수행 가능한 모든 동작을 행동으로 정의해 목표 상태 도달을 위한 최적의 상태 공간 집합을 찾는 과정을 의미한다.

먼저 상태공간에서 탐색을 통해 문제를 풀기 위해서는 목표 상태에 도달할 수 있는 일련의 연산자를 찾는 것이다. 이 문제는 각 도시를 거쳐 최종 목적지 h까지의 경로를 탐색하고 거리를 구하는 것이 목표이므로 도시 간 거리를 더하는 연산을 통해 후계노드를 생성할 수 있다.

초기 상태는 (a, 0)으로 시작점인 a와 총 거리 0을 포함하는 상태이다. a에서 접근 가능한 도시는 b, f, c가 있으므로 후계노드 및 상태는 아래와 같이 생성할 수 있다.

* (a, 0) > (b, 10)
* (a, 0) > (f, 11)
* (a, 0) > (c, 7)

같은 방식으로 이 과정을 반복하면 아래와 같이 상태공간들이 생성된다.

* (a, 0) > (b, 10) > (e, 18) > (h, 25)
* (a, 0) > (b, 10) > (e, 18) > (f, 27) > (g, 35) > (h, 40)
* (a, 0) > (b, 10) > (e, 18) > (f, 27) > (g, 35) > (d, 40) > (c, 46)
* (a, 0) > (b, 10) > (e, 18) > (f, 27) > (c, 32) > (d, 38) > (g, 43) > (h, 48)
* (a, 0) > (f, 11) > (e, 20) > (h, 27)
* (a, 0) > (f, 11) > (e, 20) > (b, 28)
* (a, 0) > (f, 11) > (g, 19) > (h, 24)
* (a, 0) > (f, 11) > (g, 19) > (d, 24) > (c, 30)
* (a, 0) > (f, 11) > (c, 16) > (d, 22) > (g, 27) > (h, 32)
* (a, 0) > (c, 7) > (d, 13) > (g, 18) > (h, 23)
* (a, 0) > (c, 7) > (d, 13) > (g, 18) > (f, 26) > (e, 35) > (b, 43)
* (a, 0) > (c, 7) > (d, 13) > (g, 18) > (f, 26) > (e, 35) > (h, 42)
* (a, 0) > (c, 7) > (f, 12) > (g, 20) > (h, 25)
* (a, 0) > (c, 7) > (f, 12) > (g, 20) > (d, 25)
* (a, 0) > (c, 7) > (f, 12) > (e, 21) > (h, 28)
* (a, 0) > (c, 7) > (f, 12) > (e, 21) > (b, 29)

위 상태공간 중 목표상태에 도달한 공간들만 고르면 다음과 같다.

* (a, 0) > (b, 10) > (e, 18) > (h, 25)
* (a, 0) > (b, 10) > (e, 18) > (f, 27) > (g, 35) > (h, 40)
* (a, 0) > (b, 10) > (e, 18) > (f, 27) > (c, 32) > (d, 38) > (g, 43) > (h, 48)
* (a, 0) > (f, 11) > (e, 20) > (h, 27)
* (a, 0) > (f, 11) > (g, 19) > (h, 24)
* (a, 0) > (f, 11) > (c, 16) > (d, 22) > (g, 27) > (h, 32)
* (a, 0) > (c, 7) > (d, 13) > (g, 18) > (h, 23)
* (a, 0) > (c, 7) > (d, 13) > (g, 18) > (f, 26) > (e, 35) > (h, 42)
* (a, 0) > (c, 7) > (f, 12) > (g, 20) > (h, 25)
* (a, 0) > (c, 7) > (f, 12) > (e, 21) > (h, 28)

(나).

평가함수란 상태가 주어졌을 때 목표상태를 향해갈 때 필요한 비용, 목표상태로 향하는 경로상에 존재할 가능성 등을 평가하는 함수를 의미한다. A\*의 평가함수는 현재 상태 노드까지의 소비 비용 g(n)과 현재 상태 노드에서 목표 노드까지 도달하는데 소비되는 비용 h(n)을 사용하며 전체 비용을 f(n) = g(n) + h(n)으로 계산한다. 하지만 h(n)은 확인이 불가능하기 때문에 실제 비용인 h(n) 대신 예상 비용인 h^(n)을 사용해 실제 평가함수는 f^(n) = g(n) + h^(n)이 된다. 이때 g(n)은 현재 도시까지의 총 거리 합이되고 h^(n)은 [그림2]의 각 도시별 거리 예측치가 된다.

스크린샷, 텍스트, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

(다).

(나)와 동일하게 평가함수는 f^(n) = g(n) + h^(n)를 사용한다. 이때 g(n)은 현재 상태 노드까지의 각 도로의 거리를 평균시속으로 이동했을 때의 시간을 합한 값이고 h^(n)은 [그림 2]의 각 거리를 16km/h로 이동했을 때의 시간이다.

라인, 텍스트, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

(라)

A\* 알고리즘은 경로 탐색 중 선택된 상태 노드로부터의 예상 비용이 실제 항상 실제 비용보다 같거나 작다면 최소비용 경로인 것을 보장한다. 따라서 항상 h(n) h^(n)이 성립되어야 한다.

(나)에서 탐색한 값들을 살펴보면 다음과 같다.

(단, n = 선택된 상태 노드, h^(n) = 예측 값, h(n) = 실제 값)

* n = a일 때, h^(n) = 22, h(n) = 23
* n = c일 때, h^(n) = 14, h(n) = 16
* n = d일 때, h^(n) = 9, h(n) = 10
* n = g일 때, h^(n) = 3.5, h(n) = 5

모두 h(n) h^(n) 조건을 만족한다. 추가적으로 위 조건 외에도 다른 경로와 비교했을 때 아래 조건을 만족해야 한다.

* g(n) + h(n) g(n2) + h^(n2)

다른 경로의 탐색값은 다음과 같다.

* n = b일 때, g(n) = 10, h^(n) = 14
* n = f일 때, g(n) = 11, h^(n) = 11.5

n이 f일 경우 g(n) + h(n) g(n2) + h^(n2)를 만족하지 못하기 때문에

g(n) + h(n) g(n2) + h(n2)도 만족하지 못해 (나)의 방식이 최단경로를 탐색할 수 있다고 보장할 수 없다.

(마)

(라)와 마찬가지로 아래 두 가지 조건을 만족해야 한다.

- 선택한 상태 노드에 대해 h(n) h^(n) 만족

- 서로 다른 경로를 가지는 상태 노드 n, n2에 대해 g(n) + h(n) g(n2) + h^(n2) 만족

(다)에서 탐색한 값들을 살펴보면 다음과 같다.

(단, n = 선택된 상태 노드, h^(n) = 예측 값, h(n) = 실제 값)

* n = a일 때, h^(n) = 1.375, h(n) = 1.356209150326797444…
* n = b일 때, h^(n) = 0.875, h(n) = 0.856209150326797444…
* n = e일 때, h^(n) = 0.375, h(n) = 0.411764705882353

n = e일 때 h(n) h^(n)를 만족하지 못하므로 (다)의 방식이 최소시간 경로를 탐색할 수 있다고 보장할 수 없다.