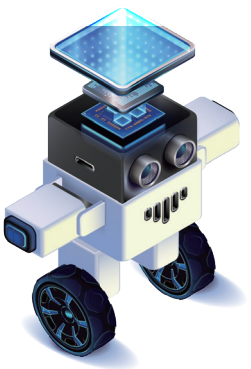
고등학교

인공지능 기초



(주) 삼양미디어

머리말

미래 사회를 정의하는 핵심 키워드를 뽑는다면, 대부분의 사람들이 주저 없이 '인공지능'을 선택할 것입니다. 이러한 인공지능 사회의 주인공이 되기 위해서는 어떤 능력이 필요할까요?

가장 먼저 데이터와 정보로 인한 디지털 세상의 변화를 인식하고, 정보의 사회적 가치를 탐구할 수 있어야 합니다. 왜냐하면 이를 바탕으로 디지털 세상의 주인공이 되기 위한 포용성을 갖출 수 있기 때문입니다.

또한, 정보를 처리하는 다양한 원리와 기술에 기반한 컴퓨팅 사고력을 함양하고, 실생활 및 다양한 학문 분야의 문제를 해결하는 능력과 태도를 길러야 합니다. 왜냐하면 이를 바탕으로 디지털 세상의 주인공이 되기 위한 주도성을 갖출 수 있기 때문입니다.

위와 같은 능력을 깊이 있게 갖추기 위해서는 '인공지능 기초' 과목을 학습해야 합니다.

'인공지능 기초' 과목을 학습함으로써 컴퓨터 과학, 데이터 과학, 정보 시스템의 내용을 기반으로, 개인의 삶과 다양한 분야에서 직접적인 영향을 미치고 있는 인공지능에 대해 깊이 있게 이해할 수 있습니다.

또한, 미래 사회의 변화와 불확실성 등으로 인한 진로와 직업의 변화에 능동적으로 대처하며, 인공지능의 주체적 사용자인 학습자가 인공지능을 인간 중심으로 안전하고 책임 있게 사용하는 자기 주도성을 갖춘 디지털 시민으로 성장할 수 있습니다. 특히, 프로젝트 기반의 프로그래밍을 통한 직접 구현, 모델에 대한 평가 등 인공지능에 대한 깊이 있는 학습을 기반으로 대학의 전공과 연계된 기초 경험을 쌓는 데에도 도움이 됩니다.

이를 위해 본 교과서는 각 단원별로 다음과 같은 목표를 달성할 수 있도록 집필하였습니다.

1단원에서는 인공지능에 대한 이해를 바탕으로 실생활의 문제를 인공지능의 관점에서 파악하고, 지능적 판단을 구현하기 위해 탐색과 추론 방식을 적용하는 능력과 태도를 기를 수 있는 내용과 활동으로 구성하였습니다.

2단원에서는 기계학습을 활용하여 해결할 수 있는 문제를 정의하고, 문제 해결 과정에서 필요한 데이터와 모델을 활용하여 문제를 효과적으로 해결하는 능력과 태도를 기를 수 있는 내용과 활동으로 구성하였습니다.

3단원에서는 인공지능의 발전에 따른 인간의 삶과 진로의 변화를 탐색하고, 인공지능의 다양한 측면에 대한 비판적인 자세를 바탕으로 인공지능과 관련된 윤리적 문제에 대해 올바른 가치관을 형성할 수 있는 태도를 기를 수 있는 내용과 활동으로 구성하였습니다.

4단원에서는 인공지능이 다양한 분야와 융합하여 새로운 가치를 창출할 수 있다는 점을 인식하고, 인류가 직면해 있는 문제를 인공지능을 활용하여 해결할 수 있는 능력과 태도를 기를 수 있는 내용과 활동으로 구성하였습니다.

본 교과서를 학습한 여러분들이 인공지능의 발전에 따른 사회의 변화를 파악하고, 인공지능의 원리에 대한 이해를 바탕으로 인공지능을 다양한 분야의 문제를 창의적으로 해결하기 위한 핵심 도구로서 프로그래밍할 수 있으며, 인공지능의 윤리적 쟁점에 관한 올바른 가치관과 태도를 함양할 수 있기를 바랍니다.

저자 일동

구성과 특징

시작하기

대단원 도입

대단원과 소단원 제목을 통해 앞으로 학습할 내용을 미리 살펴볼 수 있도록 구성했습니다. 또한 교육 과정의 핵심 아이디어를 소개하여 해당 단원에서 꼭 배워야 내용을 파악하고 이를 토대로 학습을 진행할 수 있도록 했습니다.

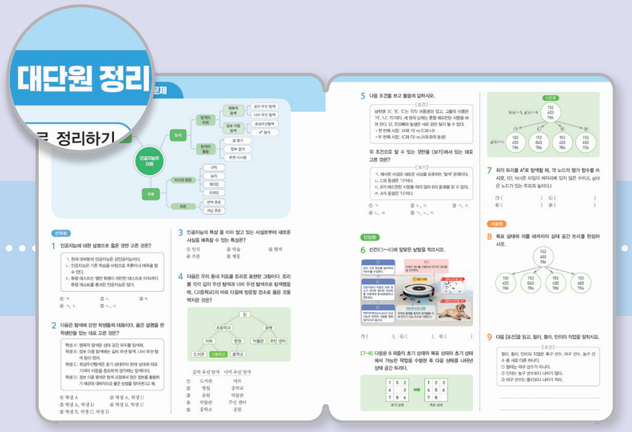
생각 열기

소단원 도입(생각 열기)

학습 목표와 관련된 일상적인 문제 상황이나 사례, 기사 등을 삽화나 사진과 함께 제시하여 학습자의 흥미와 관심을 돋우고, 질문을 통해 앞으로 배울 내용에 대해 고민해 보도록 했습니다.

정리하기

대단원 정리



대단원 정리 및 평가 문제

- **마인드맵으로 정리하기** 마인드맵을 통해 학습한 핵심 개념을 상기시키고 학습자가 스스로 정리할 수 있도록 했습니다.
- **평가 문제** 다양한 유형의 평가 문제(선택형, 단답형, 서술형 등)를 활용하여 학습한 내용을 한 번 더 정리하고 평가할 수 있도록 했습니다.

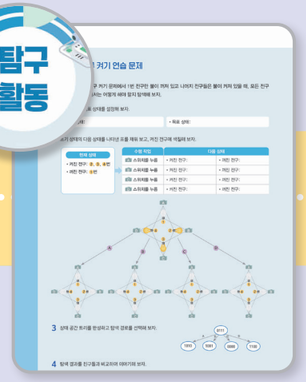
본문 전개하기

본문

단원에서 다루는 주요 개념, 기초 지식, 원리 등을 사진, 삽화, 통계 자료 등 다양한 시각 자료를 활용하여 구조화하여 제시함으로써 학습자의 이해도를 높이고, 자기 주도 학습을 촉진하도록 했습니다.

- **용어 설명** 본문에서 다루는 중요한 용어, 개념 등에 대해 보충 설명을 제공했습니다.
- **보조 설명 또는 팁** 학습 내용을 이해하는 데 도움이 되는 간단한 정보나 지식을 제공했습니다.

탐구 활동



탐구 활동

소단원 내용을 정리 · 포괄하는 활동을 제시하여 주요 개념을 이론적으로 이해하는 것뿐만 아니라 실제 상황에서 적용 가능한 문제 해결 능력을 기를 수 있도록 했습니다.

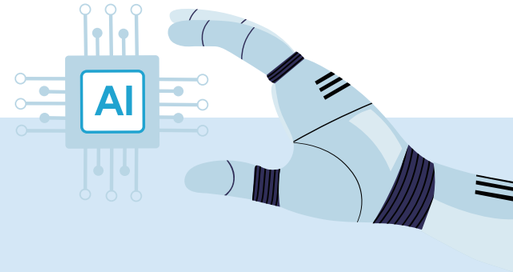
지식 충전소

지식 충전소

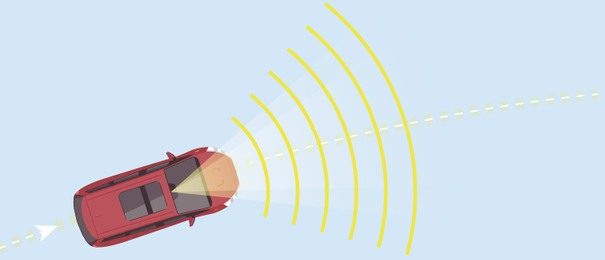
최신 정보와 다양한 분야의 읽거 자료를 제공하여 학습자의 흥미를 유발하고 인공지능 분야의 최근 동향을 파악할 수 있도록 안내했습니다.

- **해 보기** 조사, 검색, 비교, 분석, 토의, 토론, 표현, 적용 등의 비교적 쉽고 간단한 활동을 통해 학습 내용을 이해했는지 학습자 스스로 점검할 수 있도록 했습니다.
- **알고 가기** 본문에서 다룬 주요 개념과 관련된 보충 · 심화 자료를 제공하여 학습자가 한 걸음 더 다가갈 수 있도록 했습니다.
- **1분 요약** 본문에서 반드시 익혀야 하는 핵심 개념을 학습자가 스스로 확인할 수 있도록, 개념을 해당 개념을 3~4줄의 문장으로 요약하여 제시했습니다.

차례

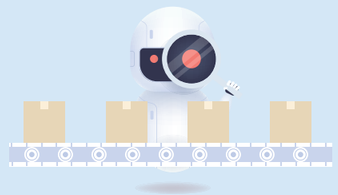


Ⅰ 인공지능의 이해



Ⅱ 인공지능과 학습

Ⅲ 인공지능의 사회적 영향



Ⅳ 인공지능 프로젝트

I 인공지능의 이해

이 단원을 시작하며

인공지능이 발전하여 우리 생활 속에 다양하게 적용되고 있다. 인공지능은 스마트폰 속 비서부터 자율주행 자동차, 제조 공정에서 불량품 감지 등 산업 분야까지 다양하게 활용되고 있다.

이 단원에서는 인공지능에서 지능적 탐색의 중요성과 탐색 원리를 이해하여 탐색 프로그램을 구현하고 인공지능이 학습하는 방식인 추론에 대한 이해를 통해 문제를 해결하는 역량을 기른다.

01 인공지능의 원리

학습 목표
- 인공지능의 지능적 판단 과정을 설명할 수 있다.
- 인공지능을 활용한 실생활 및 다양한 학문 분야의 문제 해결 사례를 비교 · 분석할 수 있다.

학습 요소 인공지능의 지능적 판단, 인공지능 문제 해결 사례

생각 열기 생활 속 인공지능

인공지능은 다양한 분야에서 활용되고 있으며, 그 활용 범위는 계속 넓어지고 있다. 매일 사용하는 스마트폰에 내장된 음성 비서도 인공지능의 한 사례다.

? 우리의 생활 속에서 찾을 수 있는 인공지능에는 어떤 것이 있을까?

1 인공지능의 개념과 특성

01 인공지능의 개념

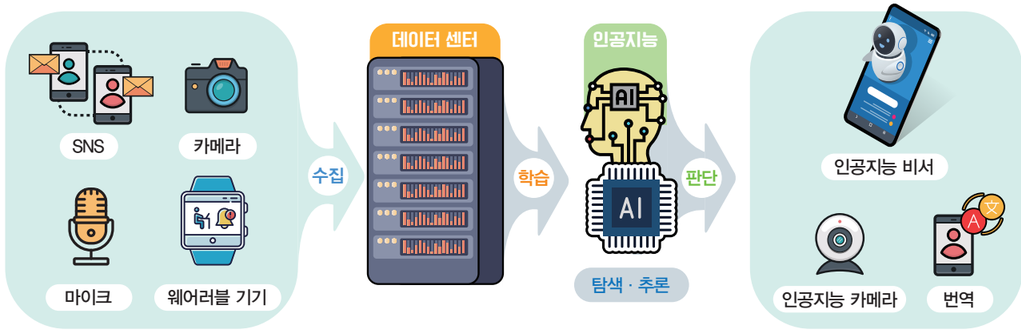
인공지능(AI: Artificial Intelligence)이란 인간의 지능이 갖는 학습, 탐색, 추론 등의 능력을 모방하여 컴퓨터로 구현한 시스템이나 구현하려는 컴퓨터 과학 기술을 말한다. 인공지능은 1950년경 앨런 튜링(Alan Turing)*에 의해 처음 제시되었으며, 현재는 새로운 부흥기를 맞이하여 빠르게 발전하고 있다.

현재의 인공지능은 목적에 맞는 대량의 데이터를 학습하여 주어진 문제에 적절한 판단을 내리는 시스템으로 볼 수 있으며, 이러한 동작을 통해 인간의 판단에 도움을 주고, 단순 작업을 대신하는 등의 편리함을 제공한다. 카메라, 마이크, 각종 감지 센서 등을 통해 입력받은 데이터는 빅데이터로 쌓이게 되고, 이러한 빅데이터를 학습한 인공지능은 탐색과 추론을 통해 판단을 한다.

인공지능의 다양한 정의
- 지성이 있는 기계를 만드는 과학과 공학이다. (존 매카시(John McCarthy))
- 인간의 지적 능력을 기계로 구현하는 과학기술이다. (우리나라 인공지능 국가전략)

***앨런 튜링**

영국의 수학자, 암호학자, 논리학자, 컴퓨터 과학자다. 기계가 인간과 같은 지능이 있는지 판단하는 튜링 테스트를 제안하였다.



[그림 Ⅰ-1] 인공지능의 이해: 인공지능은 SNS, 카메라, 마이크, 웨어러블 기기 등에 의해 생성된 빅데이터로 학습을 하고, 이를 바탕으로 목적에 맞는 판단을 한다.

해 보기 1 인공지능의 역할 탐색하기

● 인공지능이 적용된 제품이나 서비스를 생각해 보고, 인공지능이 어떤 것을 학습했을지 생각해 보자.

인공지능이 적용된 제품이나 서비스	인공지능이 학습한 것
예 로봇 청소기	예 실내 공간과 장애물

이렇게 동작하는 인공지능을 일상에서 이미 지능 에이전트의 형태로 다양하게 경험하고 있다. 스마트폰에 인공지능 대화형 비서가 기본적으로 탑재된 경우가 많으며, 필요에 따라 설치한 다양한 앱 중에 인공지능이 적용된 앱이 많아지고 있다. 번역 플랫폼에도 인공지능이 적용되어 번역의 정확도를 높여 주고, 입력에 오타가 있는 경우에도 정확하게 번역해 준다.

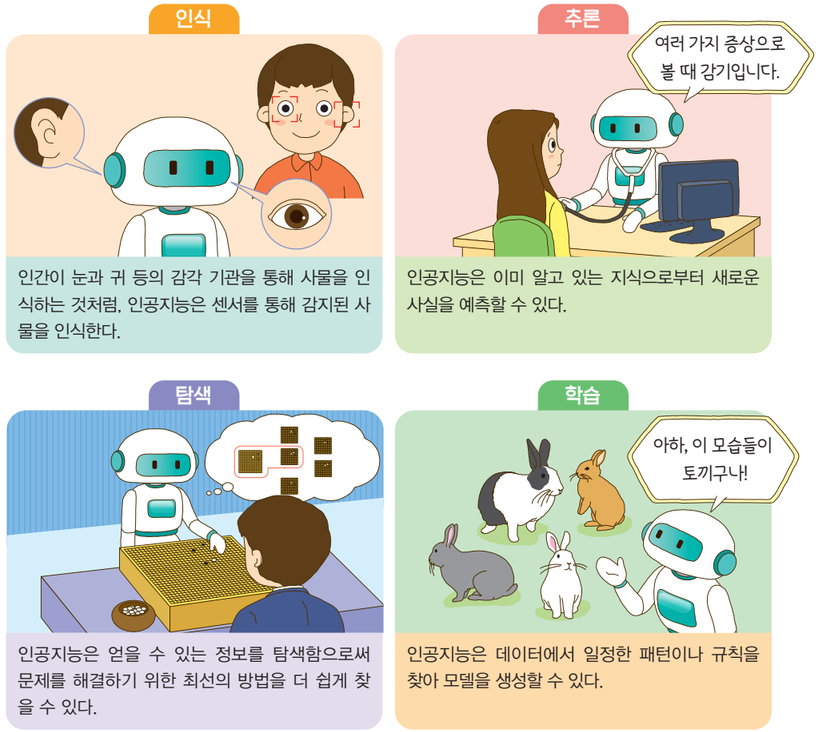
인공지능이 탑재된 로봇 청소기는 상단에 장착된 카메라를 이용하여 벽면과 천장의 특징을 구별하여 지도를 생성하고, 장애물을 탐지하여 이를 피하며 청소한다.

[그림 Ⅰ-2] **지능 에이전트인 로봇 청소기의 동작 방법 예:** 인공지능이 적용된 로봇 청소기는 센서와 카메라를 이용하여 주변 환경을 스캔하고 지도를 생성한 뒤, 지도 정보와 센서의 데이터를 바탕으로 청소를 한다.

***액추에이터**
지능 에이전트가 판단한 결과를 현실 세계에서 빛, 소리, 움직임 등의 물리적 동작이나 반응으로 만들어 내는 역할을 하며, 구동기라고도 한다.
예 LED, 버저, 전기 모터, 스피커 등

02 인공지능의 특성

인간의 지능을 모방한 인공지능은 인식, 추론, 탐색, 학습의 특성을 지니며, 데이터를 학습하고, 학습한 데이터를 근거로 추론하고 탐색한다.



인공지능의 특성

인공지능은 인간의 지능을 모방하려 하지만 인간의 지능과는 다른 특징을 보인다. 인간은 다양하고 복잡한 상황에서도 동시에 여러 가지 일을 처리할 수 있다. 하지만 지금까지의 인공지능은 다양하고 복잡한 상황에서는 한계를 보인다. 또한 특정한 한 가지 일에 특화된 경우가 대부분이다.

로봇 vs 인공지능

로봇은 기계적 장치인 하드웨어로 사전에 정해진 규칙에 따라 스스로 판단하고 행동하는 것을 말하며 기계의 하위 범주에 속한다. 반면, 인공지능은 기계 장치를 동작시키는 소프트웨어로 볼 수 있다.

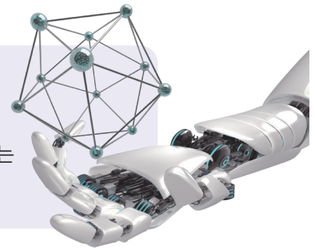
인공지능이 인간보다 강점을 보이는 부분도 있고, 인간이 인공지능보다 강점을 보이는 부분도 있다. 개와 고양이를 구분하거나 앉았다 일어나는 행동과 같이 인간이 간단히 할 수 있는 일이 인공지능에게는 어려울 수 있다. 하지만 기계 장치에서 동작하는 인공지능은 전기가 공급되는 동안 인간처럼 지치지 않고, 잠도 자지 않으며, 반복적인 작업을 오류 없이 수행할 수 있다. 또한 받아들인 정보를 빠르게 해석하고 분석할 수 있으며, 특정 영역에 대한 문제를 해결하는 방법이나 해답을 인간보다 빠르게 찾아낼 수도 있다.

인공지능은 일반적인 소프트웨어와도 다른 특징을 지닌다. 일반적인 소프트웨어는 프로그래머가 작성한 알고리즘대로 동작하고 결과를 출력하지만, 인공지능은 데이터를 기반으로 학습한 뒤, 새로운 입력이 있으면 기존 학습을 바탕으로 탐색과 추론을 할 수 있다. 인공지능이 적용되지 않은 소프트웨어와 인공지능이 적용된 소프트웨어의 특징을 비교하면 아래 표와 같다.

인공지능이 적용되지 않은 소프트웨어	VS	인공지능이 적용된 소프트웨어
• 고정된 단순 작업이나 반복되는 작업을 처리한다. • 입력 데이터가 같으면 이를 처리하는 시간과 결과가 같다.		• 추론이나 예측하는 작업이 가능하다. • 데이터 기반의 기계학습을 구현한 인공지능의 경우, 학습을 통해 성능이 개선되기도 한다.

알고 가기 약인공지능과 강인공지능

약인공지능은 주어진 역할만 정해진 대로 수행하는 인공지능을 말하며, 약인공지능이 적용된 사례로는 바둑을 두는 알파고, 퀴즈를 맞히는 왓슨 등이 있다. 현재까지 우리가 접한 모든 인공지능은 약인공지능이다. 강인공지능은 인간처럼 다양한 일을 할 수 있는 지능을 말하는 것으로, SF 영화나 소설에 나오는 인간과 비슷한 수준의 지능을 가진 인공지능이다.



해 보기 ② 자율주행 자동차에 적용할 수 있는 인공지능 탐색하기

● **자율주행 자동차에 적용될 수 있는 인공지능을 생각해 보고, 인공지능의 역할을 설정해 보자.**

인공지능이 적용될 수 있는 영역	인공지능의 역할
예 전방 사물 인식	전방에 있는 장애물을 인식하여 사물의 종류를 분류한다.

03 튜링 테스트

일반인으로 구성된 심사 위원이 5분 동안 컴퓨터와 채팅을 통한 텍스트로만 대화한다. 그 후 대화한 상대를 사람으로 판단하는 비율이 30% 이상이면, 해당 시스템에 인간처럼 지능이 있다고 판단한다.

[그림 Ⅰ-3] 튜링 테스트

1950년 앨런 튜링이 제안한 튜링 테스트(Turing test)는 기계가 인간과 비슷하게 대화할 수 있는지 판단하는 테스트로, 튜링 테스트를 통과한다면 기계에도 지능이 있다고 봐야 한다는 주장이다. 하지만, 튜링 테스트를 통과했다고 해서 기계에 지능이 있다고 판단하는 것은 한계가 있다는 지적도 있다. 인간이 다양한 감각을 통해 정보를 받아들이는 것처럼 인공지능은 다양한 센서로 주변 상황을 인식하는데, 튜링 테스트는 이런 것들이 빠진 채 문자(text) 대화에만 머물러 있었기 때문이다.

앨런 튜링은 튜링 테스트의 논리적인 방식만 제시했고, 이후 구체적이고 다양한 실험 방법이 추가로 제시되었다. 그러나 2014년 유진 구스트만(Eugene Goostman)*이 테스트를 통과했다고 주장하기 전까지 어떤 인공지능도 튜링 테스트를 통과하지 못했다. 게다가 유진 구스트만도 앞뒤가 맞지 않는 여러 문답을 했으나 13세의 어린 소년이라는 설정으로 기계라는 의심을 비켜 갔다고 볼 수 있다. 이런 텍스트 형태의 대화는 튜링 테스트 통과를 목적으로 일부러 인간처럼 맞춤법을 틀리게 대답하도록 설계된 사례도 있었다.

이처럼 튜링 테스트에 여러 비판 의견이 있지만, 튜링 테스트는 여전히 인공지능의 발전을 측정할 수 있는 하나의 척도이고 인공지능이 인간과 비슷해 보이는지 확인하는 데 충분한 도움을 줄 수 있다.

***유진 구스트만**

2014년 영국 레딩 대학(University of Reading)이 개발한 인공지능으로, 13세의 우크라이나 소년으로 설정된 채팅 프로그램이다. 당시 튜링 테스트 통과 기준은 30% 이상이었는데, 심사 위원의 33%(1/3)가 유진을 인간으로 착각하여 튜링 테스트를 통과한 최초의 인공지능이라고 주장했다.

해 보기 ③ 인공지능과 대화해 보기 분석하기

1 대화형 인공지능 챗봇(코파일럿(Copilot), 챗GPT(ChatGPT), 뤼튼(wrtn) 등)과 대화해 보자.

코파일럿 https://www.microsoft.com/ko-kr/microsoft-copilot | 챗GPT https://chat.openai.com/auth/login | 뤼튼 https://wrtn.ai

질문	답변
예 프로그래밍과 인공지능의 관계는?	프로그래밍과 인공지능은 깊게 연관된 분야입니다. 인공지능은 프로그래밍과 데이터 분석에 기반을 둔 기술이며, 적절한 프로그래밍이 없이는 인공지능 시스템을 구현할 수 없습니다.

2 인공지능의 답변이 지능적인 답변인지 아닌지 생각해 보고, 그렇게 생각한 이유를 적어 보자.

• 지능적인 답변인가? 예 지능적인 답변이다.

• 그렇게 생각한 이유는? 예 질문에 적합한 대답을 제시한다. 이것은 질문을 이해했다고 볼 수 있고, 적절한 답을 제시한 것이기 때문에 지능이 있다고 볼 수 있다.

2 인공지능을 활용한 문제 해결

01 인공지능을 활용한 문제 해결 사례

인공지능은 생활 속에서 발생하는 다양한 문제를 해결하는 데 활용되고 있다. 은행에서는 인공지능을 이용하여 스미싱(smishing)* 여부를 판단하고, 심리 상담 센터에서는 인공지능이 적용된 앱을 이용하여 어린아이의 그림을 분석하고 심리 상태를 파악한다. 또한, 인공지능 돌봄 로봇은 혼자 사는 노인의 대화 상대가 되어 주고, 건강 상태를 파악할 수 있다.

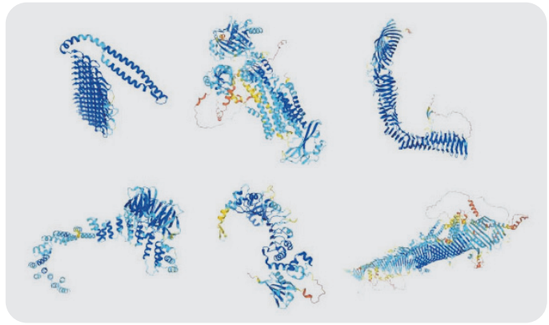
특히 빠른 속도로 고령화에 접어들고 있는 우리나라는 혼자 사는 노인 문제가 사회 문제로 주목받고 있다. 독거노인을 위한 인공지능 스피커는 노인들의 정신 건강뿐 아니라 신체 건강에도 도움을 준다. 웨어러블 기기와 사물에 부착된 센서로 행동을 감지하여 수집한 데이터를 분석한 뒤, 규칙적인 생활을 할 수 있도록 음성으로 안내하는 인공지능을 활용한 결과, 노인의 평균 걸음 수가 늘어났고 외출 시간도 증가하였다. 또한 우울증 평가 점수도 감소하고 삶의 질에 대한 만족도가 증가하는 등 정신 건강도 향상되는 결과가 나타났다.

항목	주요 지수 변화량
걸음 수	900~1,773보 증가
외출 시간	6~30분 증가
TV 시청 시간	평균 71% 감소
우울증 평가	4.8 → 2.2 감소
삶의 질 만족도	0.89 → 0.92 증가

▲ [표 Ⅰ-1] 인공지능 스피커 사용 후, 독거노인의 주요 지수 변화량

출처 용인특례시 터치케어 서비스

인공지능은 의학, 외국어 번역 및 문헌학, 예술 등 다양한 학문 분야의 문제를 해결하는 데도 활용되고 있다. 의료 영상을 보고 질병을 진단하고, 오타가 있는 문장도 제대로 번역한다. 변화된 한글을 인공지능에 학습시켜 고서적의 연대를 추정하기도 하고, 작곡도 하고 그림도 그린다. 또한 단백질 구조를 예측할 때도 인공지능이 활용되고 있다. 단백질 구조를 예측하는 학술 대회에서 인공지능 '알파폴드'가 우승하였으며, 코로나19의 경우 유전 정보가 공개되자 바이러스를 구성하는 단백질 구조 예측에 성공하였다. 알파폴드는 탄생한 지 몇 년 만에 인류가 오랜 시간 밝혀낸 약 2억 개의 단백질 구조를 예측하는 데 성공했다. 이제는 단백질 구조를 예측하는 데 그치지 않고 단백질 구조를 설계하기 시작하였으며, 이는 신약 개발 등의 생명 과학 분야에서 큰 성과를 거둘 수 있다.



▲ 알파폴드가 예측한 단백질의 3D 구조

***스미싱**
문자 메시지(SMS)를 통해 소액 결제를 유도하거나 개인 정보를 빼내 가는 피싱 사기 수법 중 하나다.

알고 가기 알파폴드의 원리

알파폴드(alphafold) 2는 2020년 12월에 열린 단백질 구조 예측 대회에서 월등한 차이로 우승했다. 딥마인드는 2022년 알파폴드를 오픈 소스로 공개했고, 2억 정도의 단백질 구조 예측 결과를 함께 발표했다. 이는 이 시기까지 밝혀진 모든 단백질 구조를 예측한 것으로 볼 수 있다. 알파폴드 2는 합성곱 신경망* 구조가 단백질 예측에는 적절하지 않은 것으로 판단하여 떨어져 있는 정보 간의 관계를 파악하는 데 뛰어난 효과를 보여 준 트랜스포머를 적용했다.

▲ 알파폴드와 실험으로 푼 단백질 구조 알파폴드(분홍색)가 예측한 구조와 실험(초록색)으로 풀어낸 구조가 거의 일치한다.

* 합성곱 신경망 이미지 처리에 탁월한 성능을 보이는 신경망으로 인접한 정보의 관계를 파악한다.

해 보기 4 인공지능을 활용한 문제 해결 사례 찾아보기

조사 · 분석하기

1 인공지능을 활용하여 실생활이나 학문 분야의 문제를 해결한 사례를 찾아보자.

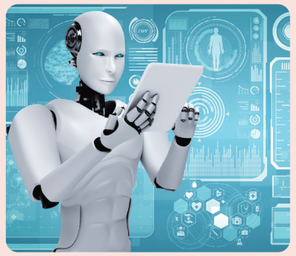
2 인공지능을 활용하였을 때와 활용하지 않았을 때의 차이점을 작성해 보고, 친구들과 이야기해 보자.

*AI+X
인공지능(AI)과 다양한 산업(X)의 융합을 의미한다. AI+교육, AI+의료 등 인공지능과 해당 산업과의 융합을 통칭하는 용어다.

02 인공지능의 미래 활용 분야(AI+X)*

인공지능 기술이 빠르게 발전하면서 의료, 교육, 운수, 번역, 산업 등 사회 전 분야에서 인공지능이 활용되고 있다. 앞으로 활용 분야는 계속 늘어날 것이다.

AI+의료



질환의 발증을 막는 '예방', 질환을 앓고 있는 사람을 찾아내는 '진단', 증세 악화를 개선하는 '치료'에 모두 적용되어 활용될 수 있다.

AI+비서

인공지능이 개인의 비서 역할을 하는 서비스로 인공지능 스피커가 대표적인 예다. 일정 관리, 외국어 번역, 검색, 음악 관리, 날씨 정보 제공, 사물 인터넷 제어 등을 제공한다.

AI+교육

학습자에게 자료 수집, 기초 분석, 시각화 등을 제공하고 교수자에게 수업 현황 보고, 문제에 대한 대안 제시 등 평가 업무와 학급 관리에 도움을 줄 수 있다. 학습자의 개별화 학습, 맞춤형 자기주도 학습이 가능하며, AI 디지털 교과서로 제공될 수 있다.

AI+운수

카메라, 라이다, 레이더 등의 센서로 사물을 인식하고 인공지능으로 최적의 판단을 하여 최종 목적지까지 안전하게 이동한다.

AI+번역

나는 오늘 저녁에 맛있는 저녁을 먹었습니다.

I had a delicious dinner this evening.

인공신경망을 이용한 기계 번역은 문맥을 파악할 수 있어 보다 자연스럽고 정확한 번역이 가능하다. 딥러닝으로 빅데이터를 학습해 시간이 지날수록 정확도는 높아진다.

AI+산업

공장 내 각종 장비에 센서가 설치되어 데이터를 수집하고, 수집한 데이터를 분석하여 최적의 공정을 설계하고 제어하는 시스템을 통해 생산 기간 단축, 맞춤형 제품 개발, 에너지 효율 증가, 재고 비용 감소 등의 효과가 나타날 수 있다.

소단원 1분 요약

1. 인공지능은 인간의 지능을 모방한 컴퓨터 시스템으로 탐색, 추론, 학습의 특성을 지닌다.
2. 튜링 테스트는 인공지능과의 대화를 통해 인공지능에게 지능이 있는지 판단하는 테스트지만, 인공지능의 여러 가지 특징은 배제하고 문자 대화만을 통해 지능을 판단한다는 제약이 있다.
3. 우리나라는 모든 산업과 인공지능과의 융합을 의미하는 AI+X를 추진하고 있으며, 인공지능은 실생활이나 다른 학문 분야에서의 문제 해결에 다양하게 활용되고 있다.

탐구 활동

교실(학교)에서 발생하는 문제를 해결하기 위한 인공지능 설계하기

1 교실(학교)에서 발생할 수 있는 문제를 찾아보자.

예 겨울철 난방기 가동으로 인해 수업 시간에 조는 학생들이 늘어나는 문제

(1)

(2)

(3)

2 발견한 문제 중 인공지능이 해결할 수 있는 문제와 해결하기 어려운 문제를 구분해 보고, 그 이유를 써 보자.

해결할 수 있는 문제	이유
예 따뜻한 온도에 조는 학생	예 온도에 따라 조는 학생이 늘어나는 것을 알 수 있다.
해결하기 어려운 문제	**이유**

3 발견한 문제를 해결하기 위해 인공지능에게 필요한 역할은 무엇인지 논의해 보자.

예 따뜻해서 졸리지 않도록 온도를 조절한다.

4 문제 해결을 위해 인공지능이 적용된 지능 에이전트의 형태를 그림으로 그리고, 기능을 설명해 보자.

12쪽의 로봇 청소기 그림을 참고하여 답안을 작성해 보세요.

02 인공지능과 탐색

학습 목표
- 인공지능에서 탐색의 중요성을 설명할 수 있다.
- 문제 해결을 위한 탐색 과정을 설계할 수 있다.

학습 요소 탐색의 중요성, 탐색 과정 설계

생각 열기 내비게이션의 길 찾기

내비게이션에 목적지를 입력하면 출발지에서 목적지로 가는 다양한 경로를 탐색하여 안내해 준다.

? 내비게이션의 AI 추천 경로는 무엇을 고려한 것일까?

1 탐색의 이해

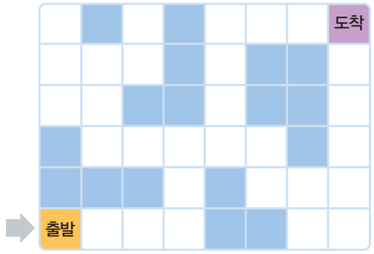
우리는 해결해야 할 문제가 생기면 여러 상황과 조건을 고려하여 실행 가능한 선택지 중에서 가장 좋은 방법을 선택하여 최적의 해결책을 찾아낸다.

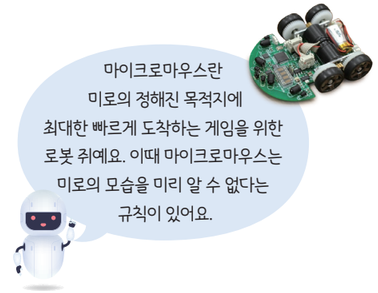
01 탐색의 개념

탐색이란 문제 상황에서 답을 찾아가는 과정으로, 컴퓨터과학자들은 인간의 탐색 방법을 모방하여 컴퓨터에 적용하고자 노력해 왔다. '미로 찾기' 문제를 예로 들면, 컴퓨터에 출발지와 목적지를 입력하고, 다양한 탐색 알고리즘을 적용함으로써 목적지까지의 경로를 자동으로 찾아내는 해결 전략을 구현하려고 노력하였다.

마이크로마우스가 아래 그림과 같은 미로를 다양한 알고리즘을 적용하여 탐색하는 과정을 살펴보며 탐색에 대해 알아보자.

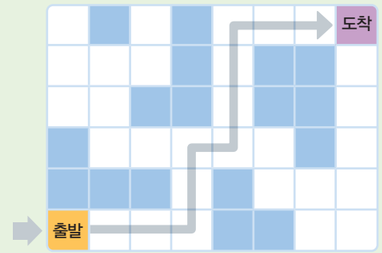
탐색의 예
- 도서관에서 책 찾기
- 체스에서 다음 수 찾기
- 인터넷에서 자료 검색하기





1 A 알고리즘

① 앞으로 세 칸 이동한다.
② 왼쪽으로 90도 회전한다.
③ 앞으로 두 칸 이동한다.
④ 오른쪽으로 90도 회전한다.
⑤ 앞으로 한 칸 이동한다.
⑥ 왼쪽으로 90도 회전한다.
⑦ 앞으로 세 칸 이동한다.
⑧ 오른쪽으로 90도 회전한다.
⑨ 앞으로 세 칸 이동한다.

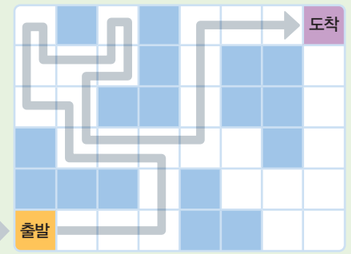


A 알고리즘은 출발부터 도착까지 모든 과정을 일일이 명령하면 그대로 수행하는 방식으로 그림의 미로를 빠져나갈 수 있다. 가장 간단한 방식이지만 출발지나 도착지가 변경되거나 미로의 모양이 바뀌면 매번 새롭게 명령해야 하는 번거로움이 있고, 미로의 모습을 알아야만 정확한 명령이 가능하기에 미로의 모습을 모르는 상황에서는 수행하기 어려운 알고리즘이다.

좌수법
미로 문제를 해결하기 위해 벽을 따라 이동하는 전략을 사용하는 방법이다. 왼손을 벽에 대고 왼쪽으로 회전할 수 있는 곳에서는 모두 왼쪽으로 회전하며 미로를 탈출한다.

2 B 알고리즘

① 왼쪽에 벽이 있다면, 그 벽을 따라 움직인다.
② 만약 왼쪽에 빈칸이 있다면, 왼쪽으로 회전한 후, 앞으로 직진한다.
③ 왼쪽으로 회전할 수 없다면, 앞으로 직진한다.
④ 만약 앞이 막혔다면 오른쪽으로 회전한다.
⑤ 목적지에 도달할 때까지 ①~④를 반복한다.



B 알고리즘은 왼쪽 벽을 따라가며 왼쪽으로 회전하여 미로 탐색 문제를 해결하는 방법을 사용한다. 이 방식은 미로의 모습을 알지 못해도, 간단한 단일 연결 미로는 반드시 탈출할 수 있는 탐색 방법이다. 하지만 탐색 경로가 항상 최단 경로를 보장하지는 못하며, 출발점이 미로의 가운데 있거나 이중으로 연결된 미로라면 탈출하지 못할 수도 있다는 한계점이 있다.

3 C 알고리즘

① 목적지에서 몇 칸이 떨어져 있는지 미로 칸에 숫자로 표시한다.
② 길을 따라 이동하고 분기점에서는 작은 숫자의 칸을 선택하여 이동한다.
③ 분기점의 숫자가 같다면 미로의 위쪽 칸을 선택한다.
④ 진행 방향에 더 이상 길이 없다면 직전의 분기점까지 되돌아오고 ③번과 다른 방향으로 탐색한다.
⑤ 목적지에 도착할 때까지 ①~④를 반복한다.

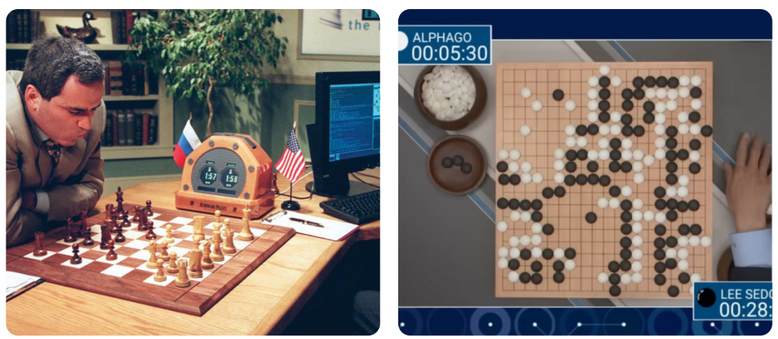
C 알고리즘은 각 미로 칸마다 숫자 정보를 부여하고 탐색하는 알고리즘이다. 사람이 길을 가는 방향을 정한다면 목적지의 반대로 이동하지 않고 목적지가 있는 방향으로 이동할 것이다. 미로 탐색도 목적지에 가까이 가는 방향으로 이동하게 하면, 불필요한 탐색 범위를 줄일 수 있다. 미로의 모습은 알 수 없어도 출발지와 목적지의 위치와 미로의 크기를 알고 있기에 그 정보를 활용하는 것이다. 미로의 칸이 벽인지 길인지 구분할 수 없는 상황에서 단순하게 목적지와 몇 칸 떨어져 있는지 미로를 구조화한 배열에 저장한다. 미로를 탐색하며 벽이 아닌 길을 따라 이동하게 하고, 분기점에서 배열에 저장된 숫자가 작은 칸을 선택하게 하면 목적지와 가까운 방향으로 이동하게 되어 탐색 범위를 좁힌다.

이처럼 같은 모습의 미로를 탐색해도 알고리즘에 따라 탐색 범위와 시간이 달라진다. 지능적인 탐색이란 탐색의 범위나 시간을 좁혀 탐색을 최적화한 것을 의미한다. C 알고리즘은 항상 최단 거리를 보장할 수는 없지만 대체로 좋은 경로를 선택하는 효율적인 탐색 방법으로 A, B 알고리즘보다 지능적인 탐색 방법이라 할 수 있다.

02 인공지능에서 탐색의 중요성

복잡한 문제를 탐색으로 해결하려는 노력은 인공지능 연구의 초창기부터 이어져 왔다. 세계 체스 챔피언에게 패배했지만, 계속 도전하여 결국 승리를 거둔 딥 블루(Deep Blue)는 인공지능의 놀라운 발전 가능성을 보여 주었다. 다른 대표적인 예는 알파고(AlphaGo)*다. 바둑은 대략 361!개의 방대한 경우의 수가 존재하기 때문에 성능이 아주 뛰어난 컴퓨터를 이용해도 모든 경우의 수를 계산하기 어렵다. 따라서 바둑은 컴퓨터가 인간을 이길 수 없는 영역으로 여겨졌다. 하지만 알파고는 수많은 경우의 수 중에서 승리에 유리한 수만을 탐색하는 방법으로 탐색 범위를 획기적으로 좁혔다. 또한 딥러닝(deep learning)*으로 방대한 분량의 바둑 전문가 기보* 데이터를 학습하고, 학습의 결과를 다시 탐색에 반영하며, 이길 확률이 높은 수를 선택함으로써 알파고는 인간에게 승리할 수 있었다.

실생활에서도 탐색을 사용하는 다양한 인공지능 사례들이 있는데 그 대표적인 예가 내비게이션이다. 내비게이션은 실시간 교통 정보를 연동하여 목적지까지 가는 최적의 길을 안내한다. 데이터를 기반으로 공사 중인 구간, 사고 현황 등의 데이터를 수집해 막히는 구간을 예측하고, 필요한 경우 우회 도로로 안내한다. 이때 다양한 상황을 반영하여 출발지부터 목적지까지 운행하는 수많은 경로를 모두 탐색하려면 많은 저장 공간과 시간이 필요하기 때문에 내비게이션 시스템을 제대로 제공하기 위해서는 탐색의 범위를 좁혀 가는 지능적 탐색으로 효율적이고 빠른 탐색을 하는 것이 필수다.



딥 블루와 알파고

***알파고**
대표적인 인공지능 바둑 프로그램으로 이세돌 9단과의 경기에서 5전 4승 1패를 기록했다.

***딥러닝**
128쪽을 참고한다.

***기보**
바둑에서 바둑돌이 놓이는 순서와 위치를 기록한 것을 말한다.

2 문제 해결을 위한 탐색 과정 설계

문제의 구조화
복잡한 문제를 쉽게 해결할 수 있도록 관리가 가능한 여러 하위 문제로 문제를 분해하거나 단순화하여 체계적으로 정리하는 작업을 의미한다.

탐색을 통해 문제를 효과적으로 해결하기 위해서는 문제를 구조화하는 설계 과정이 필요하다. 이후, 문제를 해결하기 위한 탐색 알고리즘을 선택하고 실행하여 해결 가능성을 확인하고 답을 구한다.

인공지능으로 문제를 해결하기 위해 탐색 과정을 설계하려면, 먼저 문제를 명확히 정의해야 한다.

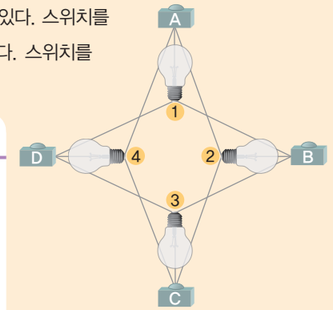
문제 해결을 위한 탐색 과정에서 만나게 되는 상황을 상태라고 하며, 모든 상태의 집합을 상태 공간이라고 한다. 그리고 문제가 주어진 상태, 즉 문제의 처음 상태를 초기 상태라고 하고, 문제가 모두 해결된 상태를 목표 상태라고 한다. 탐색은 '상태 공간 안에서 초기 상태로부터 목표 상태까지의 경로를 찾는 과정'이라고 정의할 수 있다.

문제 해결을 위한 탐색 과정 설계하기

단계 1	단계 2	단계 3	단계 4
문제의 상태 정의하기	초기 상태와 목표 상태 설정하기	수행 작업 파악하기	탐색 전략 세우기

예제 **전구 켜기 문제를 통해 탐색으로 문제를 해결하는 과정을 살펴보자.**

전구 4개(①~④)와 스위치 4개(A~D)가 연결되어 있다. 스위치를 누르면, 그 스위치에 연결된 전구 3개의 상태가 바뀐다. 스위치를 눌러서 모든 전구의 불을 켜 보자.



▲ [그림 Ⅰ-4] 전구 켜기 문제

풀이

단계 1 문제의 상태 정의하기

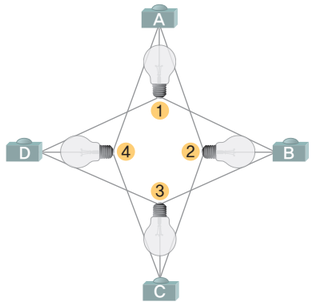
상태 정의는 문제 해결을 위한 첫 단계로서, 문제를 정확히 파악하는 것을 뜻한다. 만약 문제의 핵심을 잘못 파악한다면, 이후의 해결 방법도 신뢰하기 어려울 것이다. 전구 켜기 문제에서는 다음과 같이 상태를 정의할 수 있다.

- **상태**: 1번부터 4번까지 전구가 이웃하는 위치, 전구와 스위치가 연결된 상태, 전구가 켜진 상태, 전구와 스위치의 개수, 스위치의 눌림 상태

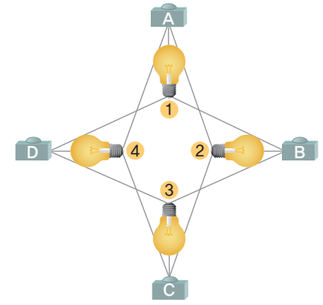
단계 2 초기 상태와 목표 상태 설정하기

그림에서 색이 없는 전구는 전구가 꺼져 있는 상태를, 노란색 전구는 전구가 켜져 있는 상태를 뜻한다. 초기 상태와 목표 상태를 설정해 보자.

- **초기 상태**: 1번부터 4번까지 전구가 모두 꺼져 있는 상태
- **목표 상태**: 1번부터 4번까지 전구가 모두 켜져 있는 상태



▲ [그림 Ⅰ-5] 초기 상태



▲ [그림 Ⅰ-6] 목표 상태

▲ 전구 켜기 문제의 초기 상태와 목표 상태

위 그림은 다음과 같이 이진수로 추상화하여 표로 나타낼 수 있다. 0은 전구의 불이 꺼진 상태를, 1은 전구의 불이 켜진 상태를 의미한다.

전구	1	2	3	4
상태	0	0	0	0

▲ [표 Ⅰ-2] 초기 상태

전구	1	2	3	4
상태	1	1	1	1

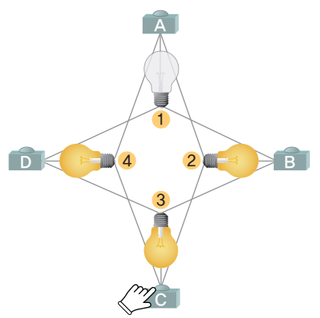
▲ [표 Ⅰ-3] 목표 상태

단계 3 수행 작업 파악하기

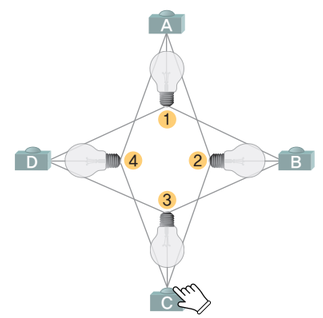
수행 작업이란 현재 상태로부터 다음 상태로 진행하기 위한 작업을 뜻한다. 수행 작업을 정의할 때는 문제의 핵심 요소*를 고려해야 한다. 전구 켜기 문제에서 고려해야 할 핵심 요소는 무엇일까?

전구를 켜기 위해서는 스위치를 누르는 작업을 해야 한다. 스위치를 누를 때 상태 변화를 파악해 보자. 먼저 스위치를 누르는 횟수를 생각해 보자. 예를 들어, 초기 상태에서 C 스위치를 한 번 누르면 2, 3, 4번 전구의 불이 켜지지만, 두 번 누르면 2, 3, 4번 전구의 불이 꺼져 초기 상태와 같아진다. 즉 스위치를 짝수 번 누르면 처음과 같은 상태가 된다.

***핵심 요소**
문제를 해결하기 위해 꼭 필요한 고려 사항으로, 상태 변화에 영향을 주는 데이터나 조건을 의미한다.

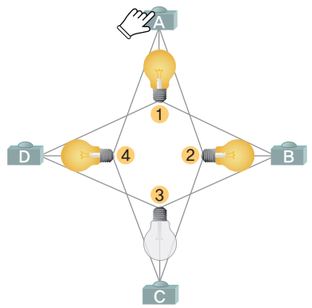


▲ [그림 Ⅰ-7] C 스위치를 처음 눌렀을 때(1회차)

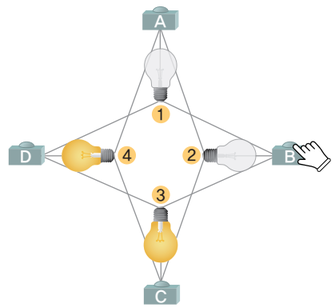


▲ [그림 Ⅰ-8] C 스위치를 다시 눌렀을 때(2회차)

이번에는 서로 다른 2개의 스위치를 연달아 누를 경우, 스위치를 누르는 순서가 결과에 영향을 미치는지 알아보자. 먼저 A 스위치를 누르면, 1, 2, 4번 전구가 켜질 것이다. 다음으로 B 스위치를 누르면, 1, 2번 전구는 꺼지고, 3번 전구가 켜져 3, 4번 전구가 켜져 있는 상태일 것이다.

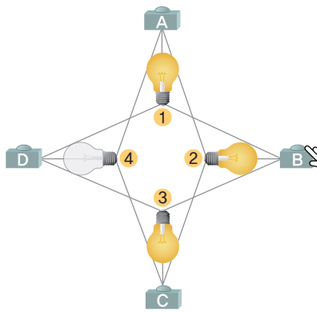


▲ [그림 Ⅰ-9] A 스위치를 먼저 눌렀을 때

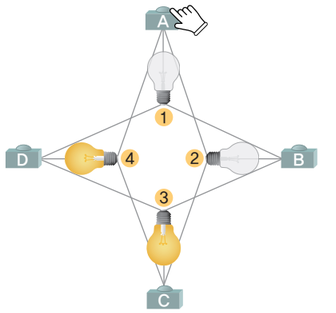


▲ [그림 Ⅰ-10] 그 이후에 B 스위치를 눌렀을 때

다시 초기 상태로 돌아가서 B 스위치를 먼저 눌러 보자. B 스위치를 누르면, 1, 2, 3번 전구에 불이 켜질 것이다. 다음으로 A 스위치를 누르면, 1, 2번 전구가 꺼지고, 4번 전구가 켜져 3, 4번 전구가 켜져 있는 상태일 것이다. 둘의 결과가 같음을 알 수 있다.



▲ [그림 Ⅰ-11] B 스위치를 먼저 눌렀을 때



▲ [그림 Ⅰ-12] 그 이후에 A 스위치를 눌렀을 때

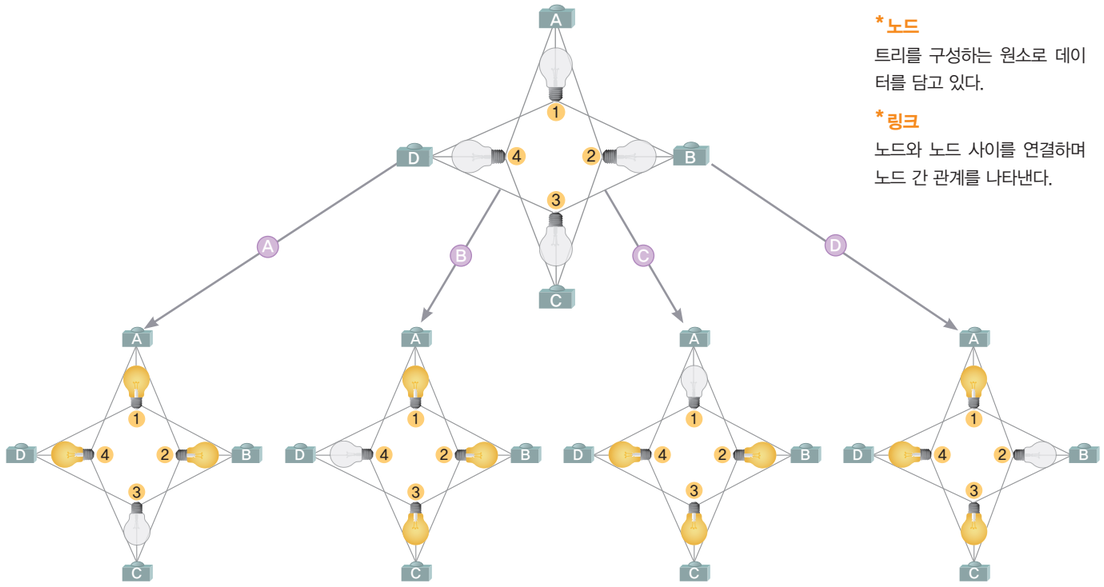
위의 과정을 통해 스위치를 누르는 순서는 결과에 영향을 미치지 않는다는 것을 알 수 있다. 이러한 조건들을 고려하여 다음과 같이 첫 번째 수행 작업과 결과를 파악할 수 있다.

현재 상태	수행 작업	다음 상태
켜진 전구: 없음 꺼진 전구: 1, 2, 3, 4	A 스위치를 누름	• 켜진 전구: 1, 2, 4 • 꺼진 전구: 3
	B 스위치를 누름	• 켜진 전구: 1, 2, 3 • 꺼진 전구: 4
	C 스위치를 누름	• 켜진 전구: 2, 3, 4 • 꺼진 전구: 1
	D 스위치를 누름	• 켜진 전구: 1, 3, 4 • 꺼진 전구: 2

단계 4 탐색 전략 세우기

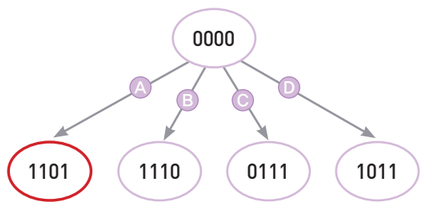
탐색 과정 설계의 마지막 단계는 탐색으로 문제를 효과적으로 해결하기 위한 전략을 세우는 것이다. 탐색 전략을 세우지 않는다면, 이미 탐색했던 것을 반복하거나 탐색 과정의 일부를 빠뜨리는 오류를 범할 수 있다. 이를 위해 먼저 문제의 상태 공간을 살펴볼 필요가 있다. 문제의 상태 공간을 구조화하여 표현하는 것은 탐색으로 문제를 해결하기 위해 사용되는 좋은 전략 중 하나다.

다음 그림은 초기 상태에서 첫 번째 스위치를 눌렀을 때의 상태를 트리구조*로 나타낸 것이다. 이와 같이 탐색 구조를 트리 형태로 나타낸 것을 상태 공간 탐색 트리라고 한다. 이때 노드(node)*는 상태를, 링크(link)*는 작업을 의미한다.

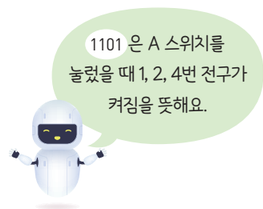


▲ [그림 Ⅰ-13] 초기 상태에서 첫 번째 스위치를 눌렀을 때의 상태 변화

위의 트리 그림은 다음과 같이 좀 더 간략하게 표현할 수 있다. 0은 전구의 불이 꺼진 상태를, 1은 전구의 불이 켜진 상태를 의미하며, 4비트 숫자는 순서대로 1, 2, 3, 4번 전구를 의미한다.



▲ [그림 Ⅰ-14] 더 구조화된 상태 공간 트리

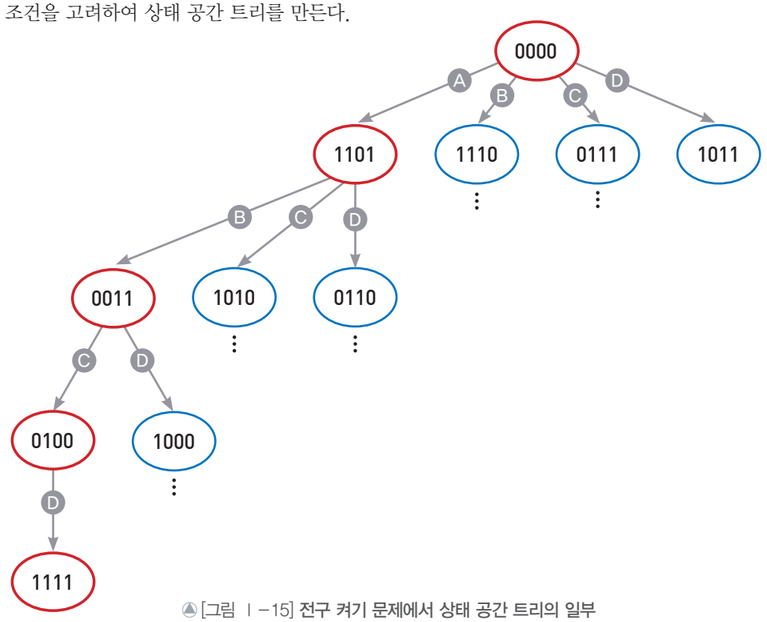


*트리구조
노드와 링크로 이루어져 있으며 계층적인 정보를 표현할 때 사용한다.

*노드
트리를 구성하는 원소로 데이터를 담고 있다.

*링크
노드와 노드 사이를 연결하며 노드 간 관계를 나타낸다.

다음은 문제 상황을 나타낸 상태 공간 트리의 일부이다. 스위치를 누르는 순서, 횟수에 대한 조건을 고려하여 상태 공간 트리를 만든다.



[그림 Ⅰ-15] 전구 켜기 문제에서 상태 공간 트리의 일부

상태 공간 구조화를 통해 단순한 탐색 전략을 세워 보았다.

탐색 전략 A → B → C → D 순서대로 스위치 누르기

이 문제에서는 스위치를 누르는 순서와 상관없이 모두 한 번씩 눌렀을 때, 문제를 해결할 수 있었다.

탐색 전략은 문제 상황에 따라 다르게 정할 수 있다. 탐색할 상태 공간을 구조화하여 탐색의 방향과 범위를 정하고, 어디서부터 탐색할지 탐색의 순서를 정해야 할 때도 있다. 또한 문제에 알맞은 탐색 알고리즘을 선택하는 것도 탐색 전략 중의 하나다. 탐색 전략에 따라 탐색 효율이 달라지기도 하므로 탐색 전략을 잘 세우는 것은 중요하다.

소단원 1분 요약

❶ 탐색은 문제 상황에서 답을 찾는 과정을 의미하고, 인공지능은 탐색을 통해 문제를 해결하며 발전했다.

❷ 탐색을 통해 문제를 효과적으로 해결하기 위해서는 문제를 구조화하는 '탐색 과정 설계'가 필요하다.

❸ 문제를 구조화하여 효율적으로 해결하기 위한 탐색 과정에는 ① 문제의 상태 정의하기, ② 초기 상태와 목표 상태 설정하기, ③ 수행 작업 파악하기, ④ 탐색 전략 세우기가 있다.

탐구 활동

전구 켜기 연습 문제

24쪽에 제시된 전구 켜기 문제에서 1번 전구만 불이 꺼져 있고 나머지 전구들은 불이 켜져 있을 때, 모든 전구의 불을 켜기 위해서는 어떻게 해야 할지 탐색해 보자.

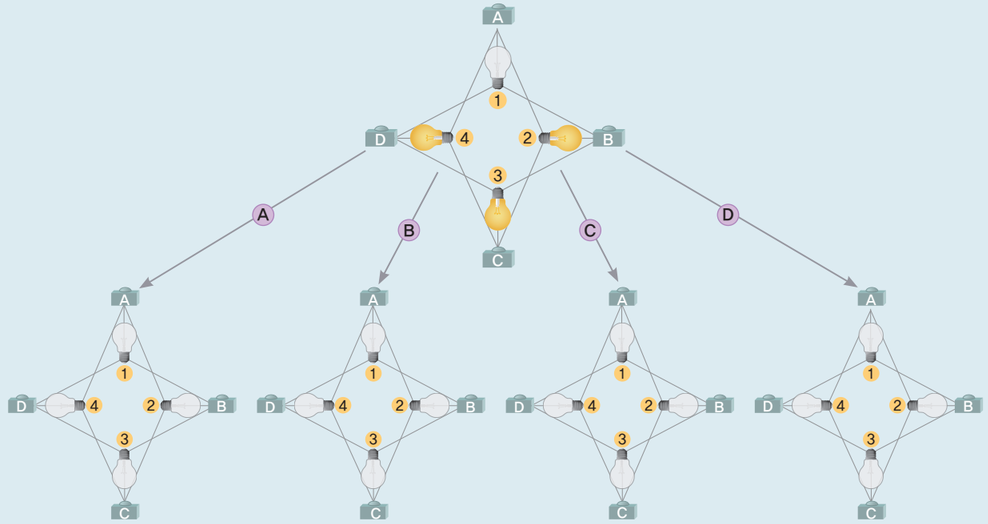
1 초기 상태와 목표 상태를 설정해 보자.

• 초기 상태:	• 목표 상태:

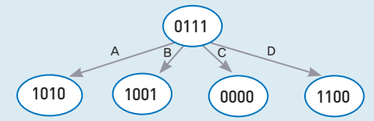
2 초기 상태의 다음 상태를 나타낸 표를 채워 보고, 켜진 전구에 색칠해 보자.

현재 상태
- 켜진 전구: 2, 3, 4번
- 켜진 전구: 1번

수행 작업	다음 상태	
A 스위치를 누름	• 켜진 전구:	• 꺼진 전구:
B 스위치를 누름	• 켜진 전구:	• 꺼진 전구:
C 스위치를 누름	• 켜진 전구:	• 꺼진 전구:
D 스위치를 누름	• 켜진 전구:	• 꺼진 전구:



3 상태 공간 트리를 완성하고 탐색 경로를 선택해 보자.



4 탐색 결과를 친구들과 비교하며 이야기해 보자.

03 맹목적 탐색과 정보 이용 탐색

학습 목표
- 맹목적 탐색과 정보 이용 탐색의 차이를 설명할 수 있다.
- 지능적 탐색의 원리를 설명할 수 있다.

학습 요소 맹목적 탐색, 정보 이용 탐색, 지능적 탐색의 원리

생각 열기 보물찾기 전략

체험학습에서 보물찾기 게임을 했다. 친구들은 각자 전략을 세워 보물을 찾기로 했다.

? 어떻게 하면 보물을 빨리 찾을 수 있을까?

1 맹목적 탐색

맹목적 탐색(blind search)이란 프로그래밍이나 기계학습으로 문제를 해결할 때 목표 상태 이외에 어떤 정보도 주어지지 않은 상태에서 탐색을 하는 것을 말한다. 예를 들어 서랍장에서 물건을 찾을 때 물건의 위치에 대한 사전 정보가 없다면 모든 서랍을 열어 봐야 할 것이다. 운이 좋으면 첫 번째 연 서랍에서 물건을 찾을 수 있겠지만, 이와 반대로 마지막에 열어 본 서랍에서 물건을 찾을 수도 있다. 이처럼 맹목적 탐색은 목표 상태에 도달할 때까지 모든 상태 공간을 탐색하는 알고리즘이다. 현재 상태 노드에서 다음 상태 노드를 방문할 때, 어떤 방향으로 탐색하는가에 따라 깊이 우선 탐색과 너비 우선 탐색 등이 있다.

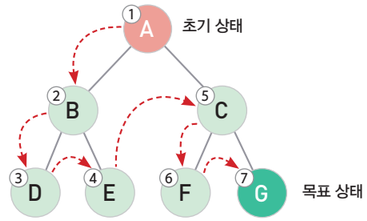
맹목적 탐색은 아무 사전 정보도 없기 때문에 무정보 탐색(uninformed search)이라고도 한다.

01 깊이 우선 탐색

깊이 우선 탐색(DFS: Depth First Search)은 상태 공간 트리에서 자식 노드(node)를 먼저 방문하는 수직 방향의 탐색 알고리즘이다. 자식 노드가 두 개 이상일 경우 주로 왼쪽 노드를 먼저 방문한다. 탐색 중에 단말 노드에 도착했는데도 목표 노드를 찾지 못했다면 다시 분기점으로 되돌아가 방문하지 않은 노드부터 다시 수직 방향으로 탐색한다. 만약 목표 노드가 수직 방향으로 깊이 있다면 문제를 빨리 해결할 수 있고, 필요한 저장 공간도 비교적 적게 드는 장점이 있다. 하지만 목표 노드가 없는 경로에 깊이 빠져 버리거나 목표 노드에 이르는 경로가 여러 개일 경우, 지금의 방문 경로가 최단 경로임을 보장할 수 없다는 단점이 있다.

노드의 종류
- 루트 노드: 최상위의 노드이다.
- 부모 노드: 직접 연결된 상위의 노드이다.
- 자식 노드: 직접 연결된 하위의 노드이다.
- 형제 노드: 나는 왼쪽에서부터 같은 부모 노드를 가진 노드이다.

자식 노드에서 답을 발견하지 못했을 때 부모 노드로 되돌아가는 것을 백트래킹(back tracking)이라고 한다.



탐색 순서

A → B → D → E → C → F → G

[그림 Ⅰ-16] 깊이 우선 탐색

해 보기 1 트리구조 파악하기 적용하기

● 본문에 제시된 '깊이 우선 탐색' 트리 그림을 보고 다음 물음에 답해 보자.

1 B 노드의 자식 노드를 찾아서 적어 보자.

2 C 노드의 부모 노드를 찾아서 적어 보자.

3 D 노드의 형제 노드를 찾아서 적어 보자.

예제 **8 퍼즐을 깊이 우선 탐색으로 해결해 보자.**

8 퍼즐은 1~8까지 쓰여 있는 8개의 숫자판과 한 개의 빈칸으로 구성된 슬라이딩 퍼즐이다. 움직이는 숫자판을 퍼즐의 빈칸으로 이동시키며 숫자를 정렬하여 목표 상태에 도달하는 게임이다. 8 퍼즐 문제 해결에 깊이 우선 탐색이 어떻게 적용되는지 살펴보자.

풀이

8 퍼즐 탐색 과정 설계

단계 1 문제의 상태 정의하기

상태는 퍼즐의 숫자판 위치다.

단계 2 초기 상태와 목표 상태 설정하기

초기 상태 숫자판과 목표 상태 숫자판을 설정한다.

2		3
1	8	4
7	6	5

초기 상태

1	2	3
8		4
7	6	5

목표 상태

단계 3 수행 작업 파악하기

빈칸 주변 타일의 움직임을 파악하고 정의한다.

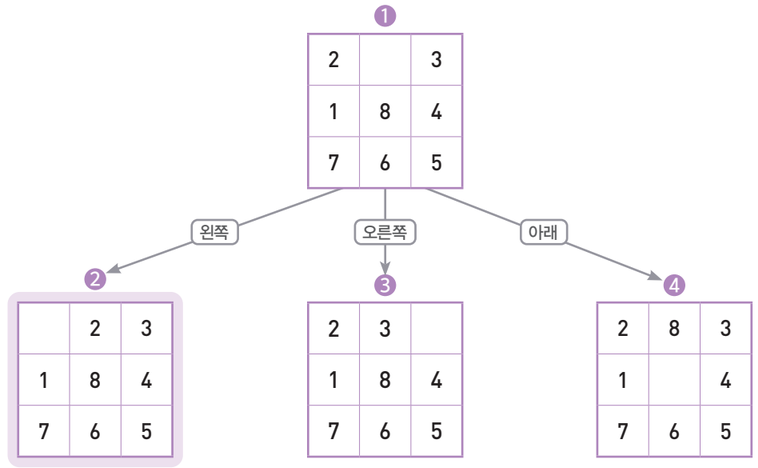
- 위: 빈칸의 위에 있는 숫자판을 빈칸으로 옮긴다.
- 왼쪽: 빈칸의 왼쪽에 있는 숫자판을 빈칸으로 옮긴다.
- 오른쪽: 빈칸의 오른쪽에 있는 숫자판을 빈칸으로 옮긴다.
- 아래: 빈칸의 아래에 있는 숫자판을 빈칸으로 옮긴다.

단계 4 탐색 전략 세우기

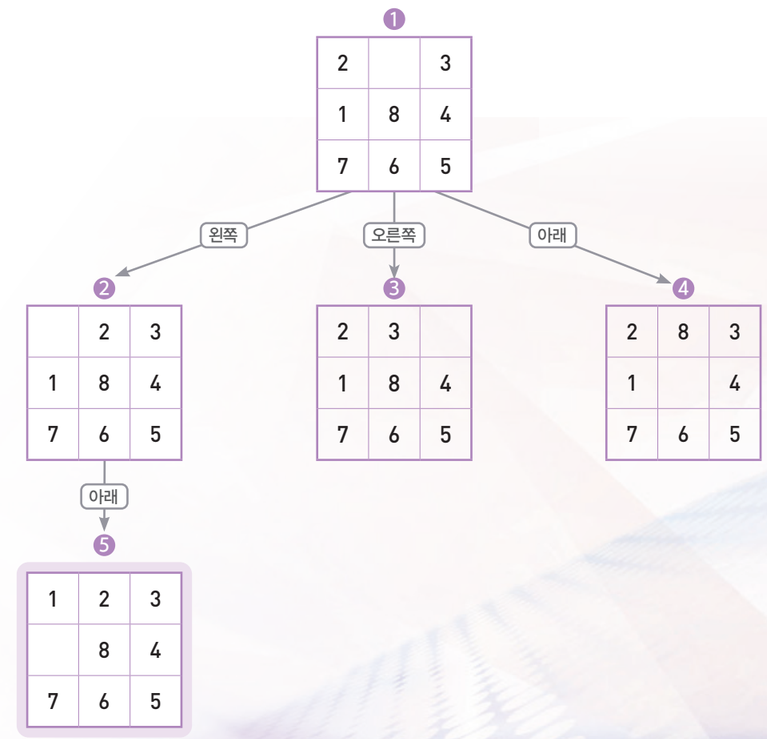
상태 트리를 다양한 방법으로 탐색한다.

8 퍼즐 깊이 우선 탐색은 현재 상태에서 가능한 작업을 수행하여 상태 공간에 자식 노드를 생성한다. 알고리즘에 따라 첫 번째 자식 노드, 즉 가장 왼쪽의 자식 노드를 방문하여 현재 상태를 자식 노드로 설정하고, 현재 상태와 목표 상태가 같은지 판단한다. 목표 상태가 아니라면 바뀐 현재 상태에서 가능한 작업을 수행하여 자식 노드를 생성하고, 첫 번째 자식 노드를 방문하여 목표 노드인지 판단하는 과정을 반복한다.

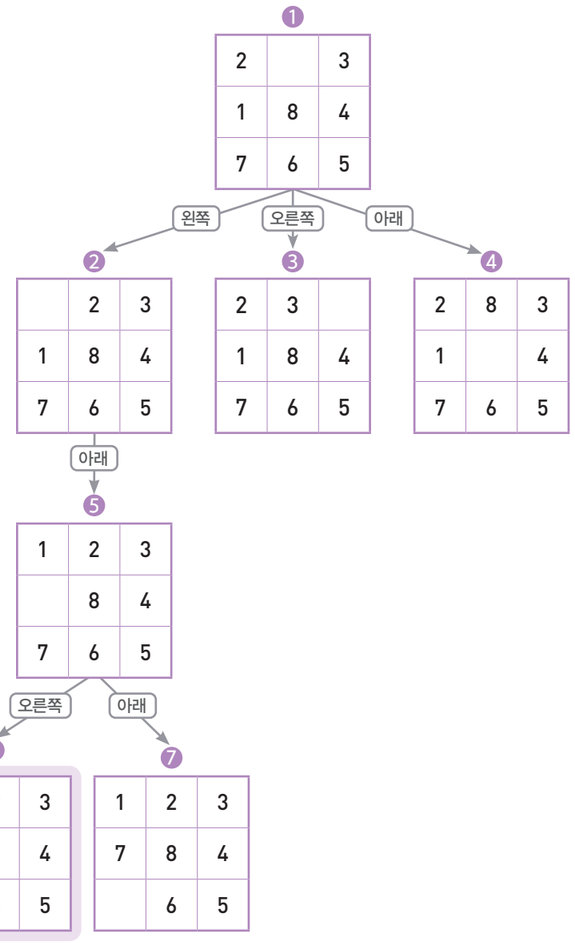
그림의 초기 상태를 살펴보면 빈칸의 위에는 숫자판이 없으므로 초기 상태에서 다음 상태를 만들기에는 왼쪽, 오른쪽, 아래의 작업만이 가능하다. 깊이 우선 탐색의 알고리즘에 따라 가장 왼쪽의 자식인 2번 노드를 선택하여 방문한다.



이제 현재 상태는 2번 노드이다. 목표 상태와 다르므로 2번 노드에서 가능한 작업을 수행하여 자식 노드를 생성한다. 빈칸의 위와 왼쪽에는 숫자판이 없고, 오른쪽 작업을 수행하면 결괏값이 초기 상태와 같아지므로 수행하지 않는다. 가능한 작업인 아래를 수행하여 자식 노드를 생성하고 방문한다.



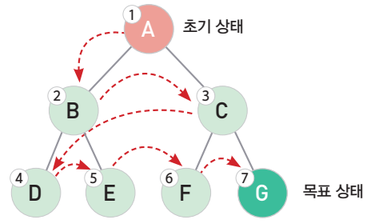
현재 상태 노드인 5번 노드도 목표 상태가 아니므로 가능한 작업을 수행하여 자식 노드를 생성한다. 이동이 가능한 위, 오른쪽, 아래 중에서 위 작업을 수행하면 결괏값이 부모 노드와 같아지므로 수행하지 않고 오른쪽, 아래 작업만 수행하여 자식 노드를 생성한 결과는 다음 그림과 같다.



현재 상태인 6번 노드와 목표 상태가 같으므로 탐색을 종료한다.

02 너비 우선 탐색

너비 우선 탐색(BFS: Breadth First Search)은 상태 공간 트리에서 형제 노드를 먼저 방문하는 수평 방향의 탐색 알고리즘이다. 상태 공간 트리의 깊이가 얕고 넓은 경우, 목표 노드를 빨리 찾을 수 있다. 또한 목표 노드를 찾을 때까지 모든 상태 공간을 탐색하므로 목표 노드가 존재한다면 반드시 찾을 수 있고, 최단 경로를 알 수 있다는 장점이 있다. 하지만 노드 수가 많아지거나 목표 노드가 수직 방향으로 깊은 곳에 있다면 탐색 시간이 오래 걸리고, 탐색을 위한 공간과 탐색 범위도 지나치게 넓어지는 단점이 있다.



탐색 순서

A → B → C → D → E → F → G

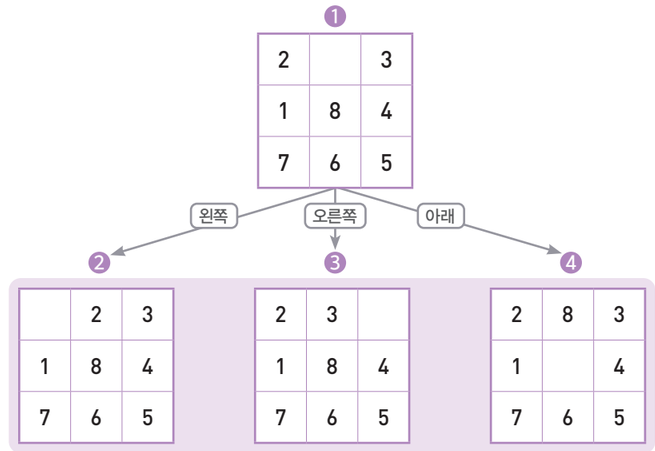
[그림 Ⅰ-17] 너비 우선 탐색

예제 **8 퍼즐을 너비 우선 탐색으로 해결해 보자.**

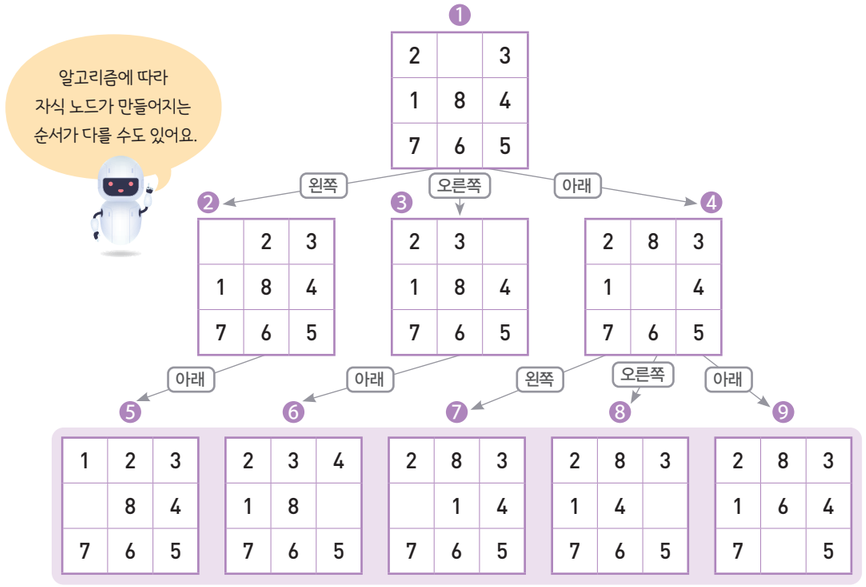
깊이 우선으로 탐색했던 8 퍼즐을 너비 우선으로 탐색하며 비교해 보자.

풀이

첫 번째 수행 결과는 깊이 우선 탐색의 결과와 같다. 초기 상태에서 수행 가능한 왼쪽, 오른쪽, 아래 작업을 수행하면 다음 그림과 같은 상태 공간 트리가 만들어진다.



깊이 우선 탐색이 수직 방향으로 진행되었던 것과 달리 너비 우선 탐색은 수평 방향으로 진행된다. 깊이 우선 탐색은 2번 노드를 방문하고 바로 5번 노드를 방문했지만, 너비 우선 탐색은 2번, 3번, 4번 노드를 모두 방문하며 목표 노드를 찾는다. 목표 노드가 없으면 2~4번 노드의 자식 노드를 생성하고 목표 노드를 찾는다.

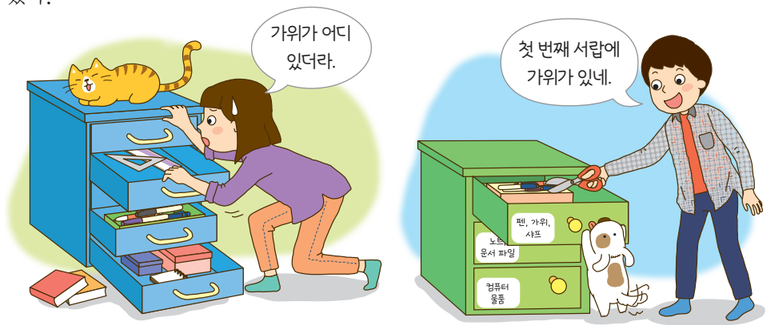


5~9번 노드에도 목표 노드가 없으므로 가능한 작업을 수행하여 5~9번 노드의 자식 노드를 생성하고 방문한다. 방문한 10번 노드와 목표 노드가 같으므로 탐색을 종료한다.

2 정보 이용 탐색

맹목적 탐색은 알고리즘이 간단하여 단순한 문제는 쉽게 해결할 수 있고, 답이 있다면 반드시 답을 찾을 수 있다. 하지만 실생활 문제나 인공지능으로 해결하려는 문제처럼 문제가 크고 복잡한 경우 모든 상태 공간을 방문해야 하기 때문에 시간이 오래 걸리고 문제를 해결하기 어려운 경우가 많다.

이에 비해 정보 이용 탐색은 경험이나 지식을 탐색에 적용하는 알고리즘으로, 필요 없는 경로를 줄여 문제 해결 시간과 공간을 절약할 수 있다. 예를 들어 서랍장에서 물건을 찾을 때 물건의 위치나 서랍 라벨을 알고 있으면 물건을 더 빨리 찾을 수 있다.



정보 이용 탐색은 모든 탐색 공간을 찾지 않고 일부 공간만 탐색하기 때문에 답을 찾지 못할 수도 있고, 항상 최단 경로로 목표 노드에 도달할 수는 없다. 하지만 대체로 좋은 결과를 내고, 효율적이며 빠르게 목표를 찾을 수 있어 내비게이션 등 인공지능 시스템의 탐색에 많이 사용된다.

정보 이용 탐색은 다음 상태를 정할 때 오직 휴리스틱(heuristic)*만 사용하는지, 시작 상태에서 현재 상태까지의 탐색 비용도 고려하는지에 따라 최상우선탐색과 A*(A-star) 탐색 등으로 구분된다.

*휴리스틱
시행착오와 같이 경험으로 얻은 지식이나 어림짐작 등을 뜻한다. 정보 이용 탐색에서 사용하는 정보의 특성이 휴리스틱하므로, 정보 이용 탐색을 휴리스틱 탐색이라고도 한다.

01 최상우선탐색

최상우선탐색은 초기 상태에서 현재 상태에 이르기까지의 비용을 고려하지 않고, 현재 가장 좋은 것을 선택하는 탐색 방법이다. 현재 상태 노드에서 다음 방문할 상태 노드를 결정할 때 목표에 가장 가까워 보이는 노드를 택한다.

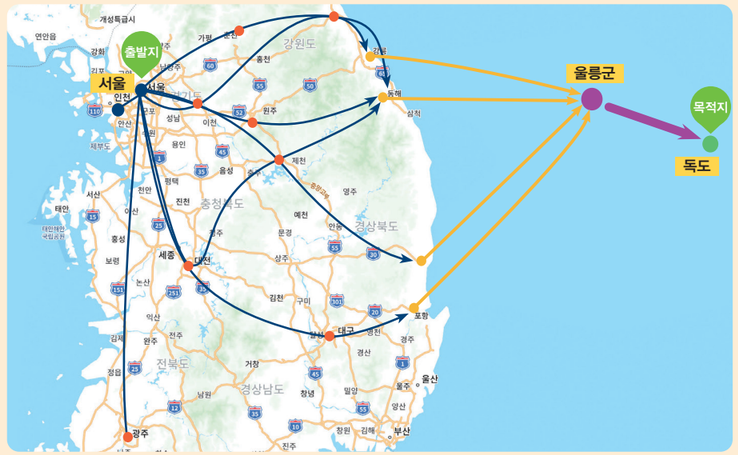
평가 함수는 현재 상태가 목표 상태에 얼마나 가까운지를 평가하는 함수로, 최상우선탐색은 이를 이용해 목표에서 가장 가까운 상태를 선택한다.

'독도로 가는 길 찾기' 문제를 통해 최상우선탐색이 문제 해결에 어떻게 적용되는지 살펴보자.

예제 '독도로 가는 길 찾기' 문제를 최상우선탐색으로 해결해 보자.

문제 상황 철수는 환경 프로젝트를 위해 독도를 직접 방문하려고 한다. 목적지까지 가장 빠르게 도착하기 위한 경로를 찾아보자.

조건
- 출발지: 서울
- 목적지: 독도
- 교통편: 육로는 차편, 바다는 배편을 이용한다.



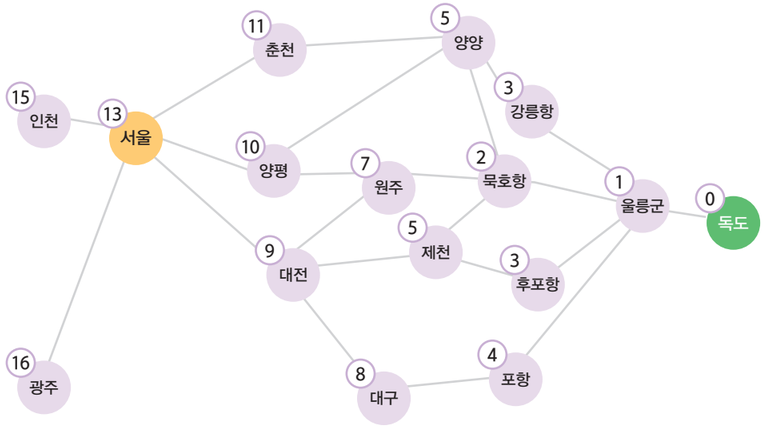
풀이

철수는 내비게이션 탐색에 사용되는 휴리스틱 값을 한 지역에서 목적지까지의 직선거리로 정한다는 것을 알고, 지도의 각 지역에서 독도까지의 거리를 직접 재어 측정하고, 그 결과를 표로 정리했다.

• h(n)은 각 노드의 휴리스틱 값을 의미한다.

지역명	h(n)	지역명	h(n)
광주	16	양양	5
인천	15	제천	5
서울	13	포항	4
춘천	11	강릉항	3
양평	10	후포항	3
대전	9	묵호항	2
대구	8	울릉군	1
원주	7		

최상우선탐색은 평가 함수로 휴리스틱 값만을 사용하기 때문에 다음 그림과 같이 추상화된 지도에 평가 함숫값을 표시하여 다음 상태를 선택하기 쉽도록 구조화했다.



이와 같이 정점*과 정점을 연결하는 간선*으로 구조화한 자료 구조를 그래프*라고 한다. 위의 그래프는 지역을 정점으로, 지역 간의 연결을 간선으로 표현하여 복잡한 지도를 단순하게 추상화했다.

'독도로 가는 길 찾기' 탐색 과정 설계

단계 1 문제의 상태 정의하기
상태는 철수가 있는 위치, 지역 간 위치로 정의할 수 있다.

단계 2 초기 상태와 목표 상태 설정하기
초기 상태는 철수가 서울에 있는 상태이고, 목표 상태는 철수가 독도에 도착한 상태다.

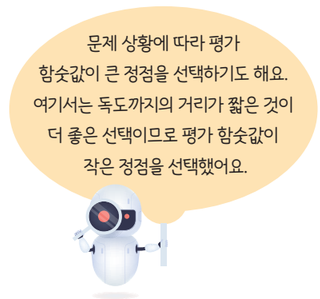
단계 3 수행 작업 파악하기
- 현재 상태 정점과 연결된 정점 중에서 현재 상태보다 평가 함숫값이 작은 정점은 모두 방문 예정 정점으로 정한다.
- 방문 예정 정점 중에서 평가 함숫값이 가장 작은 정점을 다음 상태로 정해 방문한다.

단계 4 탐색 전략 세우기
최상우선탐색 또는 A*로 탐색한다.

최상우선탐색에서 다음 상태를 선택할 때 현재 상태보다 평가 함숫값이 크다는 것은 목표 상태에서 멀어졌다는 뜻이다. 최상우선탐색은 목표와 가까운 상태를 선택하는 알고리즘이기 때문에 방문이 가능한 다음 상태 중에서 평가 함숫값이 가장 작은 노드를 다음 상태로 선택하여 방문한다.

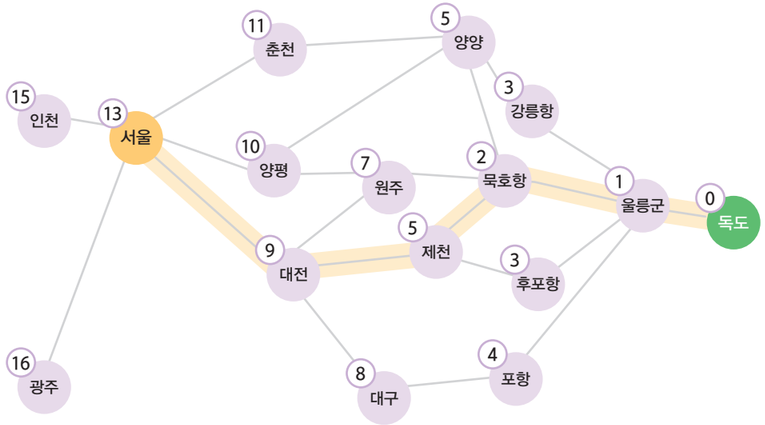
출발지인 서울과 연결된 지역 중 서울보다 평갓값이 큰 인천(15), 광주(16)는 탐색 범위에서 제외한다. 서울보다 평갓값이 작은 춘천(11), 양평(10), 대전(9) 중에서 평갓값이 가장 작은 대전을 선택하여 방문한다.

***그래프**
연결된 객체 즉 정점 사이의 관계를 보여 주는 자료 구조다.
- 정점: 트리의 노드와 같은 객체를 의미한다.
- 간선: 트리의 링크와 같이 객체 간의 연결을 의미한다.



이제 현재 상태인 대전과 연결된 원주(7), 제천(5), 대구(8)를 평가한다. 셋 중 평가 함숫값이 가장 작은 제천을 방문한다. 제천에서 연결된 묵호항(2), 후포항(3) 중 평갓값이 작은 묵호항을 선택한다. 묵호항에서 울릉군까지의 연결은 한 가지이고, 울릉군의 평갓값이 묵호항의 평갓값보다 작으므로 울릉군을 방문한다. 울릉군에서 독도를 방문하면 목적지에 도착하였기 때문에 탐색을 마친다.

탐색 경로를 지도에 표시하면 다음과 같다.



탐색 경로 서울 → 대전 → 제천 → 묵호항 → 울릉군 → 독도

02 A* 탐색 알고리즘

A* 탐색 알고리즘은 최상우선탐색을 개선한 알고리즘이다.

A* 탐색 알고리즘은 최상우선탐색을 개선한 알고리즘이다. 최상우선탐색은 현재 상태에서 유리한 선택을 하여 탐색 범위를 좁혔지만, 다른 길로 가는 것이 탐색 비용을 줄일 수 있는 경우에도 평가 함숫값이 더 크면 선택할 수 없었다. A* 탐색은 앞으로 남은 탐색 거리에, 초기 상태에서 현재 상태에 이르기까지의 비용도 고려하므로 보다 탐색 비용을 줄일 수 있는 알고리즘이다.

예제 **'독도로 가는 길 찾기' 문제를 A* 탐색으로 해결해 보자.**

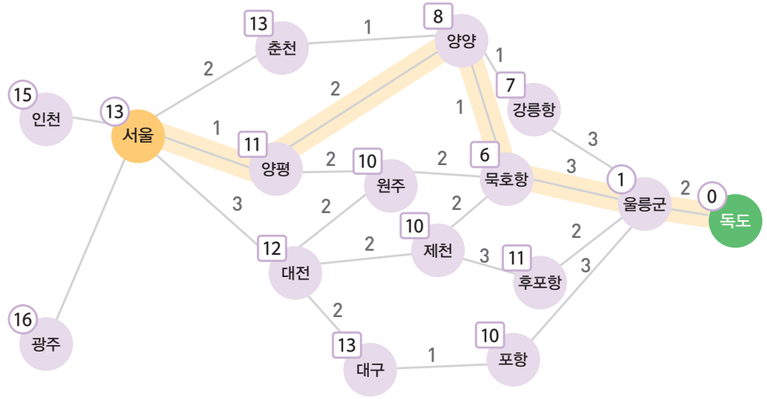
A* 탐색은 최상우선탐색과 같이, 현재 상태와 연결된 정점 중 평가 함숫값이 가장 작은 정점을 방문한다. 다만 최상우선탐색이 휴리스틱 함숫값만을 사용했던 것과 달리, 초기 상태에서 현재 상태까지의 탐색 비용을 반영하여 갱신된 평가 함숫값을 사용한다.

철수는 A* 알고리즘으로 독도에 가는 길을 탐색하기 위해 지역 간 이동 시간을 검색해서 얻은 정보를 반영시킨 새로운 평가 함숫값을 얻고 표로 정리했다.

• g(n)은 초기 상태에서 현재 상태까지의 탐색 비용을 의미한다.

현재 상태 경로	다음 상태	g(n)	h(n)	f(n) = g(n) + h(n)
서울	춘천	2	11	2+11=13
	양평	1	10	1+10=11
	대전	3	9	3+9=12
서울–양평	양양	1+2	5	1+2+5=8
	원주	1+2	7	1+2+7=10
서울–양평–양양	강릉항	1+2+1	3	1+2+1+3=7
	묵호항	1+2+1	2	1+2+1+2=6

다음 그래프의 간선 위 숫자는 각 정점 간의 이동 거리, 즉 탐색 비용을 의미한다. 또 네모 안의 숫자는 평가 함숫값이다. 38쪽 표에 제시된 휴리스틱 값에, 지금까지의 이동 시간을 더하여 얻은 새로운 평가 함숫값을 사용한다.



새로운 평가 함수를 반영하여 방문 순서를 정한다. 먼저 서울에서 평가 함숫값이 가장 낮은 양평(11)을 선택하고 방문한다. 그 다음, 양평에서 양양(8)과 원주(10) 중 양양을 선택하고, 양양에서 강릉항(7)과 묵호항(6) 중 묵호항을 선택한다. 울릉군을 거쳐 독도에 도착하면 목표에 도달하였으므로 탐색을 멈춘다.

탐색 경로 서울 → 양평 → 양양 → 묵호항 → 울릉군 → 독도

소단원 1분 요약

❶ 맹목적 탐색이란 목표 상태 외의 다른 정보 없이 일정한 순서대로 목표 상태에 이르기까지 모든 상태 공간을 탐색하는 것으로, 깊이 우선 탐색과 너비 우선 탐색 등이 있다.

❷ 정보 이용 탐색이란 휴리스틱 정보를 이용해 탐색 범위를 줄이고, 목표 상태까지 가기에 가장 좋은 경로라고 판단되는 노드를 먼저 방문하는 탐색으로 최상우선탐색, A* 탐색 등이 있다.

탐구 활동

A* 탐색으로 8 퍼즐 해결하기

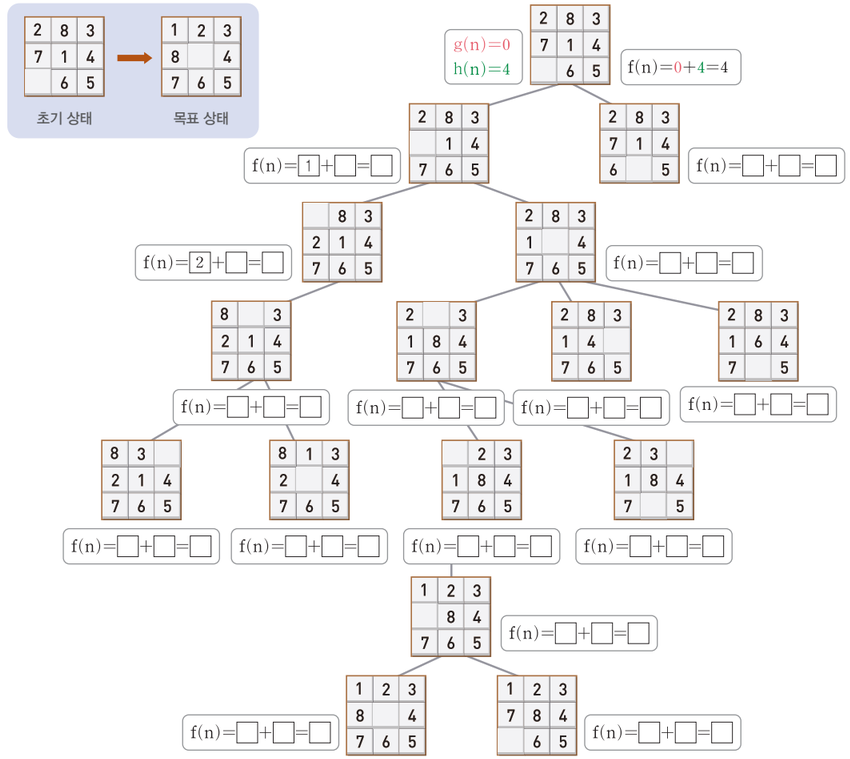
맹목적 탐색으로 해결하였던 8 퍼즐에 정보 이용 탐색을 적용하여 문제를 해결해 보자.

1 8 퍼즐의 평가 함수는 다음과 같이 정해 보자.

- 휴리스틱 h(n): 8 퍼즐의 각 타일이 목표 상태와 다른 위치에 있는 개수로 정한다.
- 탐색 비용 g(n): 현재까지의 이동 횟수로 정한다.
- 평가 함수 f(n): 현재까지의 이동 횟수와 휴리스틱 값을 더한 값으로 정한다. - 함수식: f(n) = g(n) + h(n)

2 다음은 8 퍼즐에 A* 탐색을 적용하여 문제를 해결하는 과정이다. 알맞은 g(n)과 h(n), f(n)을 적어 보자.

(단, 평갓값이 같은 경우에는 모두 방문하여 자식 노드를 생성한다.)



3 지능적 탐색이 필요한 다른 문제를 인터넷 검색 등을 통해 찾아보자.

4 평가 함수로 고려할 수 있는 또 다른 휴리스틱에는 무엇이 있을지 이야기해 보자.

지식 충전소

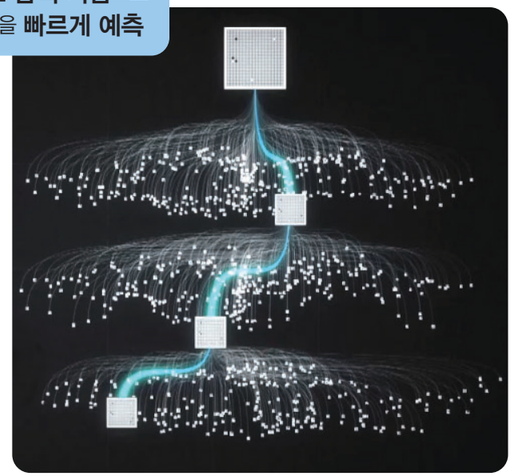
알파고와 몬테카를로 트리 탐색

몬테카를로 방법은 무작위로 선택한 표본을 이용한 확률적 결과를 사용한 값을 계산하는 알고리즘이다. 선거의 출구조사 결과가 실제 선거 결과와 크게 다르지 않은 것처럼, 만약 현재 상태에서 어느 한쪽이 유리한 국면에 있다면 유리한 국면에 있는 사람이 이길 확률이 높다는 것이다. 이러한 확률을 게임 트리에 접목한 것이 몬테카를로 트리 탐색이다. 몬테카를로 트리 탐색은 알파고의 결정 과정을 최적화하는 알고리즘으로 선택, 확장, 시뮬레이션, 역전파의 네 가지 단계로 동작한다. 알파고에서 이 네 가지 단계가 어떻게 동작하는지 살펴보자.

첫 번째 단계는 선택이다. 수많은 경우의 수를 계산하여 가장 승리 확률이 높은 수를 선택한다. 두 번째 단계는 확장이다. 선택의 결과로 이어지는 다음 수로 가능한 목록이 생겨난 것이다. 세 번째는 시뮬레이션이다. 다음 수에서 이어지는 게임의 결과를 예측해 보는 것이다. 모든 다음 수를 시뮬레이션하기에는 저장 장치의 한계와 시간적인 문제가 있으므로 다음 수 중에서 무작위로 임의의 수를 선택한다. 네 번째 단계는 역전파다. 역전파 단계에서는 시뮬레이션으로 얻은 정보를 현재 노드에서부터 시작 노드, 즉 역방향으로 되돌아가며 그 경로상에 있는 모든 노드의 정보를 갱신한다. 시뮬레이션의 결과, 승리했다면 승리한 상태부터 거슬러 올라가며 지금까지 둔 모든 수에 승점을 부여하는 것이다.

첫 번째부터 네 번째까지 과정을 반복해서 수행한 후 모든 단계가 끝나면 다음 수를 결정하는데 이때 가장 승점이 높은 수가 아닌, 가장 많이 진행한 수를 다음 수로 선택한다. 한두 번은 우연한 승리라고 생각할 수 있지만 승리가 쌓이면 더 이상 우연이라고 하기 어렵기에 더 높은 신뢰도를 부여하는 것이다. 몬테카를로 트리 탐색을 통해 알파고는 탐색의 범위를 획기적으로 좁히며 알파고의 성능 발전에 크게 이바지했다.

알파고는
몬테카를로 탐색 기법으로
유리한 지점을 **빠르게 예측**



참고 영상

EBS 이숍. 인공지능 첫 걸음, 알파고의 필승 전략. https://www.youtube.com/watch?v=QD-pNr-VLbD0(원출처: EBS)

참고 도서

박상길(2022). 『비전공자도 이해할 수 있는 AI 지식』. 반니.

프로젝트 실습

04 도전! 독도로 가는 길 찾기

학습 목표 정보 이용 탐색 알고리즘을 적용한 인공지능 프로그램을 개발할 수 있다.

학습 요소 탐색 방법을 적용한 인공지능 프로그램 개발

생각 열기 도전! 인공지능 프로그램

두 친구는 독도로 가는 길에 대해 이야기를 나누다가 독도로 가는 길을 찾는 인공지능 프로그램 개발에 도전하기로 했다.

내비게이션 프로그램은 어떻게 만들 수 있을까?

도전! 독도로 가는 길 찾기

길 찾기는 지능적 탐색 활동의 대표적인 예다. 최상우선탐색을 적용한 '독도로 가는 길 찾기 프로그램'을 개발하고, 인공지능 프로그램에 정보 이용 탐색이 어떻게 적용되는지 살펴보자.

프로그래밍 프로젝트의 절차

주제 선택 및 문제 정의 → 추상화 → 알고리즘 설계 → 프로그래밍 → 평가 및 공유

1 단계 주제 선택 및 문제 정의

- **주제 선택:** 독도로 가는 길 찾기
- **문제 정의:** 독도로 가는 길 편리하게 찾기

2 단계 추상화

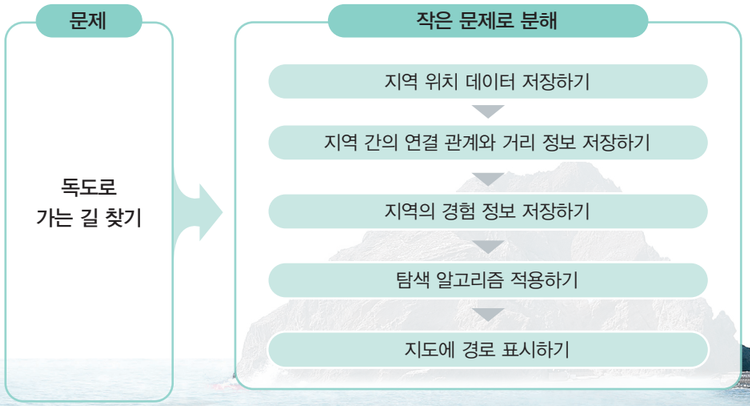
문제의 초기 상태, 목표 상태를 정의하고 더 작은 문제로 분해하여 해결하기 쉬운 문제로 추상화해 보자.

1 문제의 상태 정의

초기 상태	목표 상태
독도로 가는 길을 찾지 못한 상태	최상우선탐색을 적용하여 독도로 가는 길을 찾은 상태

2 문제 분해 및 모델링

복잡한 문제를 작은 문제로 분해하여 모델링하면 다음과 같이 표현할 수 있다.



프로그래밍 프로젝트의 절차

① 주제 선택 및 문제 정의
실생활이나 다양한 학문 분야의 주제를 탐색하여 선정한 후, 해결하고자 하는 문제를 구체적으로 정의하는 단계

② 추상화
문제의 복잡성을 제거하고 해결하기 쉬운 형태로 표현하는 단계
① 문제의 상태 정의: 문제가 있는 초기 상태와 문제가 해결된 목표 상태를 정의하는 단계
② 문제 분해 및 모델링: 크고 복잡한 문제를 해결 가능한 작은 문제로 나누고 글, 그림, 그래프, 표 등을 사용하여 문제를 쉽게 이해할 수 있는 형태로 다시 표현하는 단계

③ 알고리즘 설계
문제 분해 결과를 바탕으로 작은 문제별로 문제 해결을 위한 알고리즘을 설계하는 단계

④ 프로그래밍
설계한 알고리즘에 따라 프로그램 언어로 프로그램을 작성하는 단계

⑤ 평가 및 공유
개발 의도에 맞게 프로그램이 동작하는지, 분해한 작은 문제가 각각 해결되어 처음의 큰 문제를 모두 해결했는지 정확성과 효율성을 확인하고, 피드백을 반영하여 완성된 프로그램을 공유하는 단계

3 단계 알고리즘 설계

문제 분해를 바탕으로 알고리즘을 설계해 보자.

문제	알고리즘 설계
지역 위치 데이터 저장하기	지역의 x, y의 위치를 딕셔너리*에 저장하기
지역 간의 연결 관계와 이동 시간 저장하기	2차원 배열에 지역 간 이동 시간 정보 저장하기
지역의 휴리스틱 정보 저장하기	리스트에 지역의 휴리스틱 정보 저장하기
최상우선탐색 적용하기	① 현재 노드와 연결된 노드 중 경로에 없는 노드를 우선순위 큐*에 추가하기 ② 휴리스틱 정보가 가장 작은 노드에 가장 높은 우선순위 부여하기 ③ 우선순위가 가장 높은 노드를 큐*에서 꺼내 방문하기 ④ 방문한 노드를 방문집합과 경로 리스트 변수에 추가하고, 현재 노드까지의 이동 시간을 총 이동 시간 변수에 저장하여 갱신하기 ⑤ 현재 노드가 목적지가 아니고 큐에 남은 노드가 있다면 반복하기 ⑥ 실행이 끝나면 탐색 경로, 총 이동 시간 반환하기
지도에 경로 표시하기	지도에 탐색 경로와 총 이동 시간 표시하기

*딕셔너리
키와 값을 한 쌍으로 저장하기 위한 자료형을 말한다.
예 지역명을 키로 정하고, 평가함숫값을 한 쌍으로 저장하기
'인천': 15

*큐와 우선순위 큐
큐는 먼저 입력받은 것이 먼저 나오는, 선입 선출(First In First Out) 방식을 사용하는 자료 구조다. 입장할 때 줄을 선 순서대로 들어가는 것과 같은 원리다. 우선순위 큐는 큐 안의 데이터에 우선순위를 부여해 우선순위가 높은 것부터 먼저 나오도록 설계된 큐다.

4 단계 프로그래밍

코랩 사용법은 부록 222~223쪽을 참고한다.

파이썬(python)과 코랩(colab)을 이용하여 설계한 알고리즘을 바탕으로 독도로 가는 길 찾기 프로그램을 작성한다.

1 지역 위치 데이터 저장하기

*맷플롯립
데이터를 시각화하기 위해 사용하는 파이썬 라이브러리다.

길을 찾는 경로를 시각적으로 표현하기 위해 맷플롯립(matplotlib)*을 사용한다. 맷플롯립에서 한글을 출력하기 위해서는 별도의 절차가 필요하다. 탐색을 위해 필요한 큐, 그래프를 그리기 위한 맷플롯립, 한글 사용을 위한 모듈을 불러온다.

코드

```
# 라이브러리 불러오기
import koreanize_matplotlib      # 한글을 사용하기 위해 설치
```

코드

```
import queue                     # 정보를 저장하기 위해 큐를 사용
import matplotlib.pyplot as plt  # 시각적인 표현을 위해 사용
import koreanize_matplotlib      # 설치한 한글 모듈을 불러옴
```

지도에서 노드로 표현된 지역들의 위치 데이터를 사용자 함수 Pos( )로 정의한다. Pos( ) 함수는 딕셔너리 형태로 지역의 이름과 위치의 좌푯값이 한 쌍으로 이루어져 있다.

코드

```
# 지역의 위치 저장
def Pos():
  city = {
    '인천': [11, 292], '서울': [100, 257], '광주':[11, 90],
    '춘천': [205, 300], '양평': [195, 232],'대전': [209, 179],'양양': [295, 315],
    '원주': [255, 230], '제천': [265, 185],'대구': [250, 80],
    '강릉항': [332, 252],'묵호항': [319, 215], '후포항': [310, 165], '포항': [290, 100],
    '울릉군': [400, 223], '독도': [450, 200]
    }
  return city
```

코드 Tip 결과 창이 나타나지 않더라도, 왼쪽에 초록색으로 체크 표시가 나타나면 실행 성공이다.

2 지역 간의 연결 관계와 이동 시간 정보 저장하기

각 지역 간 연결 관계와 지역 간의 이동 시간을 사용자 함수 mTime( )으로 정의한다. 한 지역에서 다른 지역으로 연결된 길이 있다면 연결된 지역과 지역까지의 거리를 값으로 딕셔너리를 만든다. 예를 들어 '춘천': [['서울', 2], ['양양', 1]]의 의미는 춘천에서 서울과 양양으로 연결된 길이 있으며, 서울과의 이동 시간은 2, 양양과의 이동 시간은 1이라는 뜻이다. 이때 춘천–서울이 양방향으로 서로 연결되어 있으므로 키가 춘천인 곳에 서울의 값을 추가하고, 키가 서울인 곳에도 춘천의 값을 추가해야 한다.

코드

```
# 지역 간의 연결 관계와 이동 시간을 저장
def mTime():
  graph = {
    '인천': [['서울', 1]],
    '서울': [['인천', 1], ['춘천', 2], ['양평', 1], ['대전', 3], ['광주', 5]],
    '춘천': [['서울', 2], ['양양', 1]],
    '양평': [['서울', 1], ['양양', 2], ['원주', 2]],
    '대전': [['서울', 3], ['원주', 2], ['제천', 2], ['대구', 2]],
    '양양': [['춘천', 1], ['양평', 2], ['강릉항', 1], ['묵호항', 1]],
    '원주': [['양평', 2], ['대전', 2], ['묵호항', 2]],
    '제천': [['대전', 2], ['묵호항', 2], ['후포항', 3]],
    '대구': [['대전', 2], ['포항', 1]],
    '광주': [['서울', 5]],
    '강릉항': [['양양', 1], ['울릉군', 3]],
    '묵호항': [['양양', 1], ['원주', 2],['제천', 2], ['울릉군', 3]],
    '후포항': [['제천', 3], ['울릉군', 2]],
    '포항': [['대구', 1], ['울릉군', 3]],
    '울릉군': [['강릉항', 3], ['묵호항', 3], ['후포항', 2], ['포항', 3],['독도', 2]],
    '독도': [['울릉군', 2]]
    }
  return graph
```

각 지역의 위치는 지도상의 x, y 좌푯값으로 실습에 적절하도록 임의로 정하였다.

각 지역 간 이동 시간은 실습에 알맞도록 가공한 값이다.

코드 Tip 2차원 리스트는 복잡한 데이터를 구조화하고 효율적으로 저장하기 위해 사용한다. 한 지역에서 여러 지역으로 연결되어 있고, 연결된 지역의 이름과 이동 시간 등의 정보가 필요하므로 2차원 리스트를 사용하였다. [[ ]]로 대괄호가 2쌍인 것을 주의하자.

3 지역의 경험 정보 저장하기

이 프로그램에서 정보 이용 탐색을 적용하기 위해 지역의 휴리스틱 값을 담은 사용자 함수 H( )를 정의한다.

경험 정보는 각 지역에서 목적지인 독도와의 직선거리로 실습을 위해 가공한 값이다.

코드

```
def H():
  hn = {
    '인천': 15, '서울': 13, '광주': 16,
    '춘천': 11, '양평': 10, '대전': 9, '양양': 5,
    '원주': 7, '제천': 5, '대구': 8,
    '강릉항': 3, '묵호항': 2, '후포항': 3, '포항': 4,
    '울릉군': 1, '독도': 0
    }
  return hn
```

4 최상우선탐색 적용하기

bestFS( ) 함수는 최상우선탐색을 위해 구현된 사용자 함수다. start, hn, graph, goal 총 4개의 변수를 입력받는다.

- start: 출발지(서울)
- hn: 각 노드의 휴리스틱 값. H( ) 함수의 반환값을 사용
- graph: 지역 간의 연결 관계와 이동 시간. mTime( ) 함수의 반환값을 사용
- goal: 목적지(독도)

① 시작 노드를 큐에 저장하고 탐색을 위한 변수 초기화하기

bestFS( ) 함수는 최상우선탐색을 위해 구현된 함수다. 그래프의 시작 노드를 큐(queue)*에 추가하고, 탐색에 필요한 변수를 선언하여 초기화한다.

*큐
자료의 삽입과 삭제가 선입 선출(First In First Out)로 이루어지는 자료 구조를 말한다.

이 코드는 교육용으로 이해를 돕기 위해 단순화한 경로 탐색 예제이다.

*우선순위 큐
우선순위가 높은 데이터가 먼저 나오는 자료 구조를 말한다.

코드

```
def bestFS(start, hn, graph, goal = '독도'):
  pQueue = queue.PriorityQueue()  # 우선순위 큐* 생성
  pQueue.put((hn[start], start, 0))  #출발지의 이름, 출발지의 휴리스틱 값과 이동 시간을 큐에 추가
  path = []                       # 경로를 저장하기 위한 리스트 초기화
  best_time = 0                   # 총 이동 시간을 저장하기 위한 변수 초기화
  visited = set()                 # 이미 방문한 노드를 저장하기 위한 집합
```

코드 Tip

queue를 사용하기 위해서는 모듈을 import해야 한다.
set( ): 중복을 허용하지 않고 순서가 없는 자료형. 본문에서는 이미 방문한 도시를 기록해서 중복 방문을 방지한다.

② 탐색하기

현재 노드와 연결된 노드 중 아직 방문하지 않은 노드를 우선순위 큐에 추가하고, 그중에서 우선순위가 가장 높은 노드를 큐에서 꺼내 다음으로 방문한다. 방문한 노드를 visited 집합과 경로 리스트 path에 추가하고, 현재 노드까지의 이동 시간 c_time을 총 이동 시간 변수인 best_time에 저장하여 갱신한다. 방문한 노드가 목적지와 같다면 탐색을 종료하고, 탐색 경로 path와 총 이동 시간 best_time을 반환한다.

코드

```
while not pQueue.empty():              # 큐에 데이터가 있는 동안 반복
  current, c_time = pQueue.get()[1:]   # 우선순위가 가장 높은 노드를 꺼냄
  if current in visited:               # 현재 노드 visited에 있으면 건너뛰기
    continue
  visited.add(current)                 # 방문한 현재 노드를 visited 집합에 추가
  path.append(current)                 # 방문한 현재 노드를 경로 리스트에 추가
  best_time = c_time                 # 해당 노드까지의 이동 시간을 총 이동 시간에 더함
  if current == goal:                  # 현재 노드가 목적지일 경우
    break                              # 반복문 종료
  for next, cost in graph[current]:    # 현재 노드와 연결되어 있는 노드 방문
    if next not in visited:            # 방문하지 않은 노드만 탐색
      new_cost = c_time + cost         # 누적 이동 시간 계산
      pQueue.put((hn[next], next, new_cost))  # 연결된 노드의 정보를 큐에 추가
return path, best_time                 # 최종 경로와 총 이동 시간을 반환
```

코드 Tip

함수의 리턴값은 함수를 실행한 결과를 반환하는 것을 말한다. 함수가 리턴값을 반환하면 함수의 실행이 종료되고 함수를 호출한 곳에 리턴값을 돌려준다. 리턴값은 변수나 다른 함수의 인자로 쓰인다.
이 코드에서는 Pos( ) 함수의 city, mTime( ) 함수의 graph, H( ) 함수의 hn, bestFS( ) 함수의 path, best_time이 리턴값이다.

5 지도에 경로 표시하기

탐색 후의 경로를 시각화하기 위해 사용자 함수 drawMap( )을 정의한다. 각 지역의 위치는 빨간색 점으로 표시하고 지역의 연결 관계를 회색 선으로 보여 준다. 지역 간의 거리는 보라색 글자로 출력하고 최상우선탐색으로 찾은 경로를 노란색 선으로 보여 주도록 프로그램을 작성한다.

코드

```
def drawMap(city, bestfs, graph):                    # 최상우선탐색이 적용되는 그래프를 시각화
  for i, j in city.items():                          # 모든 지역에 대해 반복
    plt.plot(j[0], j[1], 'ro')                       # 지역의 위치에 빨간 점으로 표시
    plt.annotate(i, (j[0] + 5, j[1]), fontsize = 13)  # 지역의 이름을 표시
    for k in graph[i]:
      n = city[k[0]]                                 # 이웃 지역의 위치를 저장
      plt.plot([j[0], n[0]], [j[1], n[1]], 'gray')   # 지역 간의 연결선
      # 지역 간의 거리를 표시
      plt.annotate(str(k[1]), ((j[0] + n[0])/2, (j[1] + n[1])/2),color = 'purple', fontsize = 13)
  for i in range(len(bestfs)-1):
    first = city[bestfs[i]]                          # 경로의 현재 지역의 위치를 가져옴
    second = city[bestfs[i + 1]]                     # 경로의 다음 지역의 위치를 가져옴
    plt.plot([first[0], second[0]], [first[1], second[1]], 'yellow')  # 최상우선탐색 경로 표시
```

코드 Tip

- items( ) 함수는 딕셔너리의 모든 '키–값' 쌍을 반환하는 함수다.
- plt.plot( ) 함수는 맷플롯립 라이브러리의 파이플롯 모듈에 속한 함수로 데이터를 선 그래프 형태로 시각화한다.
- plt.annotate( ) 함수는 맷플롯립 라이브러리의 파이플롯 모듈에 속한 함수로 그래프에 주석을 추가하는 데 사용한다.

5 단계 평가 및 공유

정의한 사용자 함수를 호출하면 프로그램이 실행된다. 프로그램을 실행하여 성능에 대하여 친구들과 의견을 공유하고 프로그램을 평가해 보자.

코드

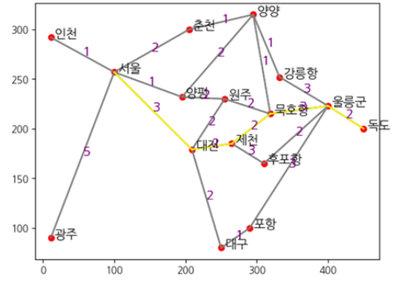
```
heuristic = H()          # 경험정보 함수 실행
graph = mTime()          # 지역의 연결 관계 및 이동 시간 정보 함수 실행
city = Pos()             # 도시의 좌표 설정
bestfs,best_time = bestFS('서울', heuristic, graph)  # 최상우선탐색 실행
drawMap(city, bestfs, graph)       # 지도에 시각화
print('AI Search => ', bestfs, '총 이동 시간: ', best_time)    # 탐색 결과 출력
```

코드 Tip

H( ), mTime( ) 등의 함수를 변수에 저장하면 함수의 실행 결괏값이 반환되어 heuristic, graph 등의 함수에 저장된다. heuristic, graph 변수는 bestFS( )의 인자로 쓰인다.

실행 결과

```
AI Search => ['서울', '대전', '제천', '묵호항', '울릉군', '독도'] 총 이동 시간: 12
```



프로그램 성능 평가 결과와 공유한 의견은 프로그램 개선에 사용된다.

성능 평가하기

평가 항목	결과	
문제 해결에 적합한 탐색 알고리즘인가?	○	X
프로그램이 문제를 효과적으로 해결하였는가?	○	X

소단원 1분 요약

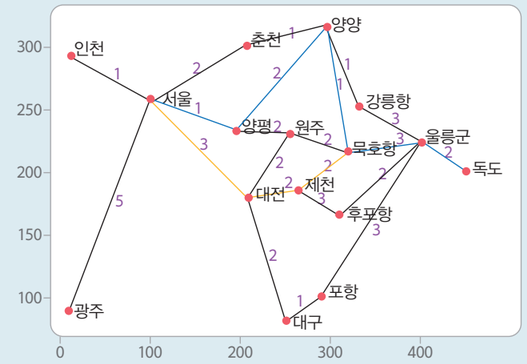
1. 정보 이용 탐색 프로그램을 개발하려면 문제 정의, 추상화, 알고리즘 설계 등의 구조화 과정이 필요하다.
2. 설계를 바탕으로 정보 이용 탐색 알고리즘을 적용한 프로그램을 구현하고, 성능을 평가한다.

탐구 활동

A* 탐색 알고리즘을 적용한 길 찾기 프로그램 구현

A* 탐색 알고리즘을 적용한 길 찾기 프로그램을 구현하여 프로그램을 개선하려고 한다. 빈칸(㉠~㉤)을 알맞은 내용으로 채워 프로그램을 완성해 보자.

최상우선 Search ⇒ ['서울', '대전', '제천', '묵호항', '울릉군', '독도'] 총 이동 시간: 12
A* Search ⇒ ['서울', '양평', '양양', '묵호항', '울릉군', '독도'] 총 이동 시간: 9



단계 ① A* 탐색과 최상우선탐색의 차이점을 파악하고, A* 탐색의 평가 함수 정의하기

- 차이점:
- 평가 함수 f(n) 정의하기
  - 최상우선탐색: f(n)=h(n)
  - A* 탐색: f(n)=(㉠)

단계 ② 알고리즘 설계하기

문제	알고리즘 설계
A* 탐색 적용하기	❶ 현재 노드와 연결된 노드 중 경로에 없는 노드를 우선순위 큐에 추가하기 ❷ 경험 정보와 현재까지의 이동 시간의 합이 가장 작은 노드에 가장 (㉡) 우선순위 부여하기 ❸ 우선순위가 가장 높은 노드를 큐에서 꺼내 방문하기 ❹ 방문한 노드를 방문집합과 경로 리스트 변수에 추가하고, 현재 노드까지의 이동 시간을 총 이동 시간 변수에 저장하여 갱신하기 ❺ 현재 노드가 목적지가 아니고 큐에 남은 노드가 있다면 반복하기 ❻ 실행이 끝나면 탐색 경로, 총 이동 시간 반환하기

탐구 활동

단계 ③ 프로그래밍

설계한 알고리즘을 바탕으로 A* 알고리즘을 적용한 독도로 가는 길 찾기 프로그램을 작성해 보자.

1 준비하기

① 라이브러리 불러오기
② 지역별 위치 데이터 설정하기
③ 지역 간의 연결과 이동 시간 정보 설정하기
④ 각 지역의 경험 정보 설정하기

※ 최상우선탐색 알고리즘으로 길 찾기 프로젝트(46~49쪽)에서 작성했던 소스 코드를 그대로 사용하도록 한다.

2 A* 탐색 알고리즘 구현하기

① 시작 노드를 큐에 저장하고 탐색을 위한 변수 초기화하기

Astar( ) 함수는 bestFS( )를 활용한 사용자 함수다. 기본 구조를 그대로 사용하되 시작 노드에서 현재 노드까지 이동한 시간을 저장하는 변수 gn을 새롭게 선언하고 0으로 초기화한다.

코드

```
def Astar(start, hn, graph, goal = '독도'):
    pQueue = queue.PriorityQueue()          # 우선순위 큐를 생성
    pQueue.put((hn[start], start, 0))       # 시작 노드를 큐에 삽입
    ㉢                                      # 시작 노드에서 현재 노드까지 이동한 시간을 저장하는 변수 초기화
    path = []                               # 경로를 저장할 리스트 생성
    astar_time = 0                          # 총 이동 시간을 저장할 변수 초기화
    visited = set()                         # 이미 방문한 노드를 저장하기 위한 집합
```

② 탐색하기

이번 단계도 bestFS( )의 일부만 수정하여 사용한다. 수정할 부분은 다음과 같다.
현재 노드까지의 이동 시간 c_time을 총 이동 시간 astar_time에 저장한다.
현재까지의 총 이동 시간에 연결된 노드까지의 이동 시간을 더하여 gn 변수에 저장한다.
인접 노드의 휴리스틱 값에 gn을 더한 값을 우선순위로 설정하여 큐에 추가한다.

코드

```
<생략>
 astar_time = c_time          # 현재 노드까지의 이동 시간을 총 이동 시간에 저장하기
 if current == goal:          # 현재 노드가 목적지라면
  break                       # 반복문 종료
                              # 현재 노드와 연결되어 있는 노드 방문
 for next, cost in graph[current]:
   if next not in visited:    # 방문하지 않은 노드만 탐색
    gn = astar_time + cost    # 현재까지의 총 이동 시간에 연결된 노드까지의 이동 시간을 더함
    pQueue.put((hn[next] + ㉣, next, gn)) # f = hn + gn로 우선순위 큐에 추가
return path, astar_time       # 최종 경로와 총 이동 시간을 반환
```

3 지도에 경로 표시하기

이번 단계도 기존의 drawMap( ) 함수를 수정하여 사용한다.

- astar 변수를 추가한다.
- bestfs 변수를 astar 변수로 바꿔 Astar( )로 탐색한 결과가 지도에 출력되도록 한다.

코드

```
def drawSearchMap(city, bestfs, ㉤, graph):
<생략>
  for i in range(len(astar)-1):
    first = city[astar[i]]
    second = city[astar[i + 1]]
    plt.plot([first[0], second[0]], [first[1], second[1]], 'blue')
```

4 프로그램 실행하기

- besFS( ) 함수 호출을 참고하여 Astar( ) 함수를 호출한다.
- drawMap( ) 함수에 astar 변수를 추가한다.
- 탐색 경로와 총 이동 시간을 화면에 출력하는 부분을 추가한다.

코드

```
<생략>
astar,astar_time = Astar('서울', heuristic, graph)
drawMap(city, bestfs, ㉥, graph)
print('A* Search => ', astar, '총 이동 시간: ', astar_time)
```

단계 ④ 평가 및 공유

프로젝트 과정을 평가해 보자.

평가 항목	결과	
문제 해결에 적합한 탐색 알고리즘인가?	○	X
프로그램이 문제를 효과적으로 해결하였는가?	○	X
bestFS에 비해 A*가 효율적인 탐색을 하였는가?	○	X

자기 평가, 동료 평가

평가 요소	자기 평가			동료 A			동료 B			동료 C		
	상	중	하	상	중	하	상	중	하	상	중	하
알고리즘 이해도												
문제 해결 방식의 창의성												
참여도 및 협업 능력												
프로그램의 완성도												

단계 ⑤ 확장하기

지능적 탐색을 적용하여 프로그래밍할 수 있는 또 다른 예를 찾아보자.

05 지식의 표현과 추론

학습 목표
- 규칙과 사실을 활용하여 지식을 표현할 수 있다.
- 새로운 지식을 추론하여 생성할 수 있다.

학습 요소 지식, 지식 표현, 추론

생각 열기 추론을 통한 지식 생성

인간은 기존에 알고 있는 사실이나 지식을 통해 새로운 사실이나 지식을 이끌어 낼 수 있다. 이러한 추론 능력은 인간의 지능을 모방하려는 인공지능도 갖고 있다.

인공지능은 어떻게 추론을 할까?

1 지식의 표현

01 인공지능과 추론

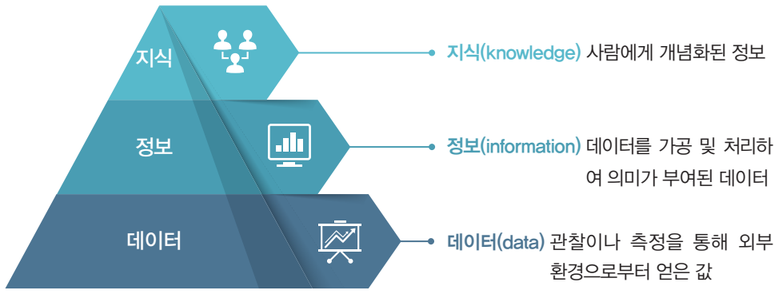
추론은 이미 알고 있는 지식을 토대로 새로운 결론을 이끌어 내는 사고 과정을 말한다. 즉, 알고 있는 사실로부터 모르고 있던 사실을 알게 되는 과정이라 할 수 있다. 예를 들어 A라는 사람의 자녀가 아들과 딸이 한 명씩 있고, 대학생과 고등학생이라는 것을 알고 있다. 그런데 아들이 고등학생이라는 사실을 알게 되었다. 그러면 A의 딸은 대학생이란 것을 추론할 수 있다.

추론은 인공지능에서도 중요한 영역이다. 인공지능은 데이터를 이용한 학습을 바탕으로 새로운 데이터로 추론을 하여 결과를 도출한다. 우리가 사용하는 스마트폰의 음성 인식, 이미지 검색, 이메일의 스팸 필터링 등에서 추론 기능이 사용되고 있다. 다양한 플랫폼의 추천 엔진 역시 추론이 이용된다.

지식 표현과 추론
추론을 하는 데 지식 표현 방식이 중요한 이유는 지식 표현 방식이 추론의 방식을 결정하기 때문이다.

02 데이터, 정보, 지식

데이터는 관찰이나 실험, 조사로 얻은 사실이나 자료를 의미하며, 가공하기 전의 상태이다. 반면, 정보는 다양한 데이터 중에서 사용자의 필요에 따라 처리한 데이터를 말한다. 지식은 개념화된 정보로 정보를 일반화하고 체계화한 것이라 할 수 있다. 예를 들어, 풍속, 풍향, 습도, 온도 등을 기록해 둔 것은 데이터이고, 이러한 데이터를 근거로 "내일 비가 올 확률은 80%"라고 예보하는 것은 정보다. 그리고 그동안의 정보에 대한 경험을 근거로 "비 올 확률이 80% 이상이면 우산을 소지하고 다니는 것이 좋다."라고 하는 것은 지식이라 할 수 있다.



데이터, 정보, 지식의 관계

인간은 지식을 활용하여 다른 사람의 질문을 이해하고 질문에 답변할 수 있다. 또한 다양한 상황과 목적에 맞게 지식을 적용할 수 있다. 인공지능도 문제를 해결하기 위해 지식이 필요하며, 인공지능에게 필요한 지식은 기계가 이해할 수 있는 형태여야 한다.

03 지식 표현

인공지능을 활용하기 위해서는 사람과 컴퓨터가 동시에 이해할 수 있는 형태로 지식을 나타내는 것이 필요하고, 지식을 이러한 형태로 나타낸 것을 지식 표현이라고 한다. 문제 해결을 위해 적절한 지식 표현 방법을 선택하는 것은 매우 중요하며, 표현된 지식은 추론 과정을 통해 새로운 사실을 생성할 수 있다. 지식을 표현하는 방법에는 규칙, 논리, 의미망, 프레임 등이 있다.

해 보기 ❶ 데이터, 정보, 지식 생성하기 분석하기

● 다음에 제시된 데이터로부터 정보와 지식을 생성해 보자.

데이터	정보	지식
예 컴퓨터 용품과 고기를 구매하기 위해 가까운 두 마트의 가격을 조사했다. 물품 / A마트 / B마트 마우스 / 15,000원 / 18,000원 키보드 / 18,000원 / 22,000원 소고기 / 40,000원 / 36,000원 돼지고기 / 12,000원 / 10,000원	예 마트마다 물건값이 서로 다르게 형성되어 있다. 전자 제품은 A마트가 저렴하고 고기류는 B마트가 저렴하다.	예 전자 제품을 사려면 A마트에 가고, 고기류를 사려면 B마트에 가면 더 저렴하게 물건을 구매할 수 있는 것을 알았다.
예 지난 몇 년간의 계절별 평균 강수량을 조사했다. 연도 / 봄 / 여름 / 가을 / 겨울 2016 / 277 / 479 / 124 / 87 2017 / 85 / 984 / 102 / 72 2018 / 401 / 559 / 268 / 40 2019 / 112 / 459 / 274 / 136 2020 / 146 / 1086 / 302 / 30 2021 / 418 / 484 / 250 / 18 2022 / 131 / 1211 / 410 / 63 (단위: mm)		
추가 심화활동 계절별 기온, 겨울철 적설량, 폭염일 등을 조사하여 표로 작성해 보자.		

1 규칙

다양한 지식 표현 방법 중 대표적인 한 가지는 규칙을 이용한 방법이다. 우리가 흔히 사용하는 '만약 〈A〉라면, 〈B〉이다.' 또는 '〈A〉하면, 〈B〉하다.'와 같은 형태로, IF 〈A〉 THEN 〈B〉로 나타낼 수 있다. 〈A〉는 주어진 정보나 사실에 대응될 조건이고, 〈B〉는 조건을 만족할 때 하는 판단이나 행동인 결론이다. A가 참이면, B를 수행하는 형태이고, 이를 생성 규칙이라고도 하며, 인과 관계, 추천, 전략, 경험적 지식 등을 표현할 수 있다.

지식 표현이 어려운 이유
- 지식으로 표현하기에 방대한 양의 정보를 가진 경우가 많다.
- 애매모호한 지식을 정확히 표현하기 어렵다.
- 지식은 상황에 따라 변하는 경우가 많다.

구분	지식	지식 표현(IF~THEN)
인과 관계	밤새고 공부하면, 다음날 피곤하다.	IF 원인 THEN 결과
		IF 밤새고 공부했다. (원인) THEN 다음날 피곤하다. (결과)
추천	기온이 영하로 내려가면, 방한이 잘 되는 옷을 입어라.	IF 상황 THEN 추천 내용
		IF 기온이 영하로 내려갔다. (상황) THEN 방한이 잘 되는 옷을 입어라. (추천 내용)
전략	차가 시동이 걸리지 않으면, 배터리를 확인한다. 배터리에 이상이 없으면, 연료를 확인한다.	일련의 규칙들로 표현 IF 차가 시동이 걸리지 않는다. THEN 배터리를 확인한다. AND 배터리 확인을 종료한다. IF 배터리 확인을 종료했다. AND 배터리는 충분하다. THEN 연료를 확인한다. AND 연료 확인을 종료한다.
경험적 지식	액체가 투명하고 100도에서 끓으며, 냄새가 없으면, 액체는 물이다.	경험적 지식을 표현 IF 액체는 투명하다. AND 액체는 100도에서 끓는다. AND 액체는 냄새가 없다. THEN 액체는 물이다.

▲ [표 Ⅰ-4] 규칙을 이용한 지식 표현

Tip 조건이나 결론이 여러 개인 경우의 지식 표현

❶ 조건이 여러 개인 경우

형식	예
IF 〈A1〉 AND 〈A2〉(〈A1〉 OR 〈A2〉) THEN 〈B〉	IF 송곳니가 있다. AND 포유류이다. THEN 육식동물이다.

❷ 결론이 여러 개인 경우

형식	예
IF 〈A〉 THEN 〈B1〉 AND 〈B2〉	IF 물건이 부족하다. THEN 물가가 오른다. AND 경제가 어려워진다.

AND와 OR는 섞어서 사용할 수 있다.

해 보기 ❷ 지식을 IF~THEN으로 표현해 보기 표현하기

● 내가 알고 있는 지식을 IF~THEN의 형태로 표현해 보자.

내가 알고 있는 지식	지식 표현(IF~THEN)
예 봄이 되면 제비가 돌아온다.	IF 봄이 된다. THEN 제비가 돌아온다.

2 논리

논리는 수학 논리학에서 사용하는 명제 논리나 술어 논리를 사용한다. 명제 논리는 명제를 가지고 논리를 판별하는 것이고, 술어 논리는 주어와 서술어 간의 관계를 다루는 것이다. 명제 논리에서 나타나는 단점은 술어 논리로 보완될 수 있다. '개는 동물이다'라는 평서문을 술어 논리로 표현하면, 'animal(개)'처럼 표현할 수 있다. 가장 명확한 방법이지만 복잡한 상황에서의 적용이 어렵다는 특징이 있다.

평서문	술어 논리 표현
개는 동물이다.	animal(개)
개는 사람을 좋아한다.	like(개, 사람)

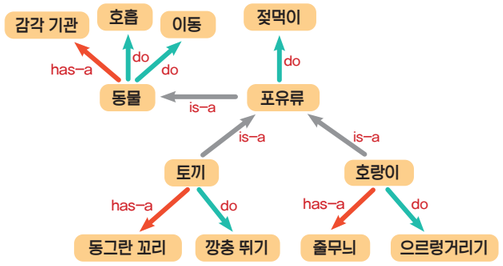
[표 Ⅰ-5] 논리 표현의 예

3 의미망

호는 에지(edge)나 링크(link)로 표현되기도 한다.

의미망(semantic net)은 지식이나 기억, 실세계를 노드(node)와 호(ark)를 이용하여 네트워크 구조로 표현한 것이다. 노드는 객체, 개념, 사건 등을 표현하고, 호는 노드 간의 관계를 화살표로 나타낸다.

아래 의미망을 보면 노드는 동물, 포유류, 토끼, 호랑이, 감각 기관, 호흡, 이동 등이 있고, 호는 has-a(~이 있다), is-a(~이다), do(~을 한다)가 있다. 동물과 감각 기관이 has-a로 연결된 것은 '동물은 감각 기관이 있다'를 의미하고, 포유류와 동물이 is-a로 연결된 것은 '포유류는 동물이다'를 의미한다.



[그림 Ⅰ-18] 의미망 표현의 예

출처 http://ai4school.org/?page_id=2522

알고 가기 지식 베이스와 전문가 시스템

지식 베이스는 전문가 시스템의 구성 요소 중 하나로 특정 분야 전문가의 지식이나 문제 해결에 필요한 사실, 규칙 등이 저장되어 있는 데이터베이스다. 전문 지식을 표현하기 위해서 IF~THEN 형식의 생성 규칙을 사용한다.

전문가 시스템은 인간의 지적 활동과 경험을 통해 축적된 전문가의 지식과 전문가에 의해 정의된 추론 규칙을 활용하여 결정을 내리거나 문제를 해결하는 시스템이다. 필수 구성 요소로 지식 베이스와 추론 엔진이 포함되어 있어야 한다.

출처 한국정보통신기술협회 정보통신 용어사전

앞의 의미망이 나타내는 지식은 다음 표와 같다.

호(arc)	지식	호(arc)	지식
is-a	포유류는 동물이다.	do	포유류는 젖을 먹인다.
	토끼는 포유류다.		동물은 호흡을 한다.
	호랑이는 포유류다.		동물은 이동을 한다.
has-a	동물은 감각 기관이 있다.		토끼는 깡충 뛰기를 한다.
	토끼는 동그란 꼬리가 있다.		호랑이는 으르렁거리기를 한다.
	호랑이는 줄무늬가 있다.		

[표 Ⅰ-6] 의미망이 나타내는 지식

4 프레임

프레임(frame)은 특정한 객체나 개념에 대한 지식을 슬롯(slot)으로 표현한 것이다. 하나의 프레임은 여러 개의 슬롯으로 구성되고, 슬롯은 슬롯 이름과 슬롯 값으로 구성된다. 제목이 '인공지능'인 책을 예로 들어 살펴보면, 책에는 작가가 있고, 출판 연도가 있다. 또한 책의 분량을 페이지로 나타낼 수 있다. 프레임은 관련된 지식을 한군데 모을 수 있다는 것과 표현력이 우수하다는 장점이 있지만, 슬롯에 대한 표준이 없고 지식을 생성하는 것이 매우 힘들다는 단점이 있다.

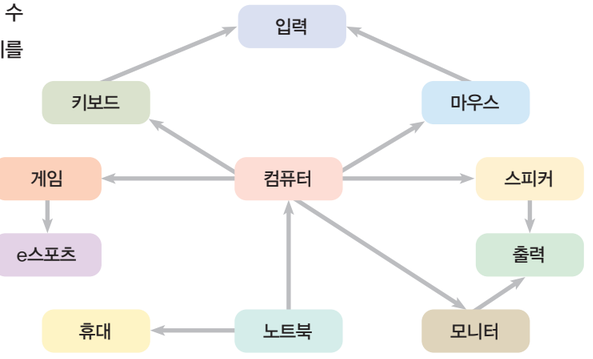
슬롯 이름	슬롯 값
제목	인공지능
작가	홍길동
출판 연도	2023
페이지	500

[표 Ⅰ-7] 책을 프레임으로 표현한 예

해 보기 ③ 의미망을 이용하여 지식 표현하기 표현하기

- '~은(는) ~이(가) 있다', '~은(는) ~을(를) 할 수 있다', '~은(는) ~이다'를 이용하여 호의 의미를 작성해 보자.

예 마우스는 입력을 할 수 있다.



2 추론

인공지능에서는 주로 연역 추론과 귀납 추론 중 하나의 방식을 사용하거나 이 둘을 조합하여 추론한다.

01 연역 추론

연역 추론이란 일반적인 전제를 바탕으로 새로운 결론을 논리적으로 도출하는 추론 방법이다. 대표적인 방법으로 삼단 논법이 있다. 삼단 논법은 두 개의 전제와 하나의 결론으로 된 형식을 가지고 있다. 최초의 대전제가 결론을 이끌어 내는 중요한 근거가 되며 논리의 일관성이 있다.

- 대전제: 모든 인간은 죽는다.
- 소전제: 소크라테스는 인간이다.
- 결론: 그러므로 소크라테스는 죽는다.

▲ 연역 추론의 예(삼단 논법)

02 귀납 추론

귀납 추론이란 구체적인 사례에서 일반적인 결론을 이끌어 내는 추론으로 이미 알고 있는 사실로부터 모르고 있던 사실을 유추해 내는 것이다. 그렇기 때문에 사례의 다양성이나 정확성이 결론에 큰 영향을 끼친다.

소크라테스는 죽었다.
공자는 죽었다.
맹자는 죽었다.
세종대왕도 죽었다.
소크라테스, 공자, 맹자, 세종대왕은 모두 인간이다. 그러므로 모든 인간은 죽는다.

▲ 귀납 추론의 예

03 추론의 오류

연역 추론이나 귀납 추론으로 언제나 옳은 결론에 도달하는 것은 아니다. 연역 추론에서 오류는 전제 조건을 생략하거나 잘못 설정했을 때 발생할 수 있고, 귀납 추론에서 오류는 관찰된 표본이 전체를 대표할 수 있는 표본이 아닌 경우나 전체에 대한 표본의 해석이 잘못되는 경우에 발생할 수 있다.

연역 추론의 오류	귀납 추론의 오류
• **대전제:** 네 발이 있으면 포유류다. • **소전제:** 거북이는 네 발이 있다. • **결론:** 거북이는 포유류다.	• **사례1:** 백조1은 희다. • **사례2:** 백조2는 희다. • **사례3:** 백조3은 희다. • **결론:** 그러므로 모든 백조는 희다.
•**오류:** '네 발이 있으면 포유류다'라는 대전제가 오류다. 네 발이 있어도 포유류가 아닌 경우는 많다.	•**오류:** 모든 백조를 본 것이 아니다. 실제 오스트레일리아에 검은 백조가 살고 있다.

04 추론을 통한 지식 생성

알고 있는 사실을 바탕으로 추론을 통해 새로운 지식을 생성할 수 있다. 주어진 상황과 조건을 보고 추론을 통해 새로운 지식을 생성하는 과정을 살펴보자.

예제 아인슈타인 문제

다음 문제 상황과 조건을 확인하고, 물을 마시는 사람과 얼룩말을 키우는 사람은 어느 나라 사람인지 맞혀 보자.

문제 상황 집의 색, 국적, 마시는 음료, 먹는 음식, 키우는 동물이 모두 다르다.

조건

① 모두 다섯 채의 집이 있다.
② 호주인은 빨간색 집에 산다.
③ 이탈리아인은 개를 기른다.
④ 초록색 집에 사는 사람은 커피를 마신다.
⑤ 우크라이나인은 차를 마신다.
⑥ 초록색 집은 아이보리색 집의 바로 왼쪽이다.
⑦ 버섯을 먹는 사람은 달팽이를 기른다.
⑧ 노란색 집에 사는 사람은 사과를 먹는다.
⑨ 한가운데 있는 집에 사는 사람은 우유를 마신다.
⑩ 노르웨이인은 왼쪽에서 첫 번째 집에 산다.
⑪ 양파를 먹는 사람은 여우를 기르는 사람 옆집에 산다.
⑫ 사과를 먹는 사람은 말을 기르는 사람 옆집에 산다.
⑬ 케이크를 먹는 사람은 오렌지 주스를 마신다.
⑭ 일본인은 바나나를 먹는다.
⑮ 노르웨이인은 파란색 집 옆집에 산다.

풀이

❶ 조건 10번과 15번을 보면 노르웨이인은 첫 번째 집에 살고, 오른쪽 옆집은 파란색이다. 또한 조건 9번을 통해 3번 집은 우유를 마신다는 것을 알 수 있다.

생성된 지식 16. 두 번째 집은 파란색이다.

구분	집1	집2	집3	집4	집5
색		파란색			
국적	노르웨이				
음료			우유		
음식					
동물					

❷ 조건 6번을 보면 초록색 집이 아이보리색 왼쪽 집이라고 하였다. 이 경우 초록색이 가능한 경우는 세 번째 집과 네 번째 집이 있지만, 조건 4번을 통해 초록색 집은 네 번째 집이고, 다섯 번째 집이 아이보리색 집인 것을 알 수 있다.

생성된 지식 17. 네 번째 집은 초록색이다. 18. 다섯 번째 집은 아이보리색 집이다.

구분	집1	집2	집3	집4	집5
색		파란색		초록색	아이보리색
국적	노르웨이				
음료			우유	커피	
음식					
동물					

❸ 조건 2번의 호주인은 빨간색 집에 산다고 되어 있는데 아직 색을 모르는 집은 1번과 3번 집이고, 1번 집은 노르웨이인이 살기 때문에 3번 집에 호주인이 사는 것을 알 수 있다. 따라서, 3번 집은 빨간색이고 1번 집은 마지막 남은 색인 노란색인 것을 알 수 있다.

생성된 지식 19. 세 번째 집은 빨간색이고, 호주인이 산다. 20. 첫 번째 집은 노란색이다.

구분	집1	집2	집3	집4	집5
색	노란색	파란색	빨간색	초록색	아이보리색
국적	노르웨이		호주		
음료			우유	커피	
음식					
동물					

❹ 조건 8번을 보면 1번 집이 사과를 먹는 것을 알 수 있고 조건 12번을 통해 1번 집의 옆집인 2번 집이 말을 기르는 것을 알 수 있다.

생성된 지식 21. 두 번째 집은 말을 기른다.

구분	집1	집2	집3	집4	집5
색	노란색	파란색	빨간색	초록색	아이보리색
국적	노르웨이		호주		
음료			우유	커피	
음식	사과				
동물		말			

❺ 조건 5번과 13번은 모두 2번 집과 5번 집만 입력이 가능하다. 따라서 조건 14번에서 제시한 조건이 가능한 집은 4번 집이다.

생성된 지식 22. 네 번째 집에 일본인이 산다.

구분	집1	집2	집3	집4	집5
색	노란색	파란색	빨간색	초록색	아이보리색
국적	노르웨이		호주	일본	
음료			우유	커피	
음식	사과			바나나	
동물		말			

❻ 3번 조건의 이탈리아인은 개를 기른다고 되어 있는데 국적이 비어 있는 것은 2번, 5번 집이지만, 2번 집은 말을 기르기 때문에 이탈리아인은 5번 집에 사는 것을 알 수 있다.

생성된 지식 23. 다섯 번째 집에 이탈리아인이 산다.

구분	집1	집2	집3	집4	집5
색	노란색	파란색	빨간색	초록색	아이보리색
국적	노르웨이		호주	일본	이탈리아
음료			우유	커피	
음식	사과			바나나	
동물		말			개

❼ 다시 5번 조건을 통해 2번 집에 우크라인이 사는 것을 알 수 있고, 이를 통해 13번 조건에 해당하는 집은 5번 집이다.

생성된 지식 24. 두 번째 집에 우크라이나인이 산다.

구분	집1	집2	집3	집4	집5
색	노란색	파란색	빨간색	초록색	아이보리색
국적	노르웨이	우크라이나	호주	일본	이탈리아
음료		차	우유	커피	주스
음식	사과			바나나	케이크
동물		말			개

❽ 1번 집의 음료가 비어 있는 것을 통해 1번 집의 음료는 물인 것을 알 수 있다.

생성된 지식 25. 첫 번째 집은 물을 마신다.

구분	집1	집2	집3	집4	집5
색	노란색	파란색	빨간색	초록색	아이보리색
국적	노르웨이	우크라이나	호주	일본	이탈리아
음료	물	차	우유	커피	주스
음식	사과			바나나	케이크
동물		말			개

⑨ 7번 조건에서 버섯을 먹는 사람은 달팽이를 기르는데, 2번 집은 말을 키우기 때문에 3번 집이 버섯을 먹는 것을 알 수 있다.

생성된 지식 **26. 세 번째 집은 버섯을 먹는다.**

구분	집1	집2	집3	집4	집5
색	노란색	파란색	빨간색	초록색	아이보리색
국적	노르웨이	우크라이나	호주	일본	이탈리아
음료	물	차	우유	커피	주스
음식	사과		버섯	바나나	케이크
동물		말	달팽이		개

⑩ 음식이 비어 있는 2번 집은 양파를 먹고, 11번 조건을 통해 양파를 먹는 사람의 옆집인 1번 집이 여우를 기르는 것을 알 수 있다.

생성된 지식 **27. 두 번째 집은 양파를 먹는다.** **28. 첫 번째 집은 여우를 기른다.**

구분	집1	집2	집3	집4	집5
색	노란색	파란색	빨간색	초록색	아이보리색
국적	노르웨이	우크라이나	호주	일본	이탈리아
음료	물	차	우유	커피	주스
음식	사과	양파	버섯	바나나	케이크
동물	여우	말	달팽이		개

⑪ 마지막으로 4번 집의 동물만 비어 있다. 이를 통해 4번 집은 얼룩말을 키우는 것을 알 수 있다.

생성된 지식 **29. 네 번째 집은 얼룩말을 키운다.**

구분	집1	집2	집3	집4	집5
색	노란색	파란색	빨간색	초록색	아이보리색
국적	노르웨이	우크라이나	호주	일본	이탈리아
음료	물	차	우유	커피	주스
음식	사과	양파	버섯	바나나	케이크
동물	여우	말	달팽이	얼룩말	개

소단원 1분 요약

❶ 데이터를 가공하면 정보, 정보를 가공하면 지식이 된다.

❷ IF~THEN의 형태로 규칙을 이용하여 지식을 표현할 수 있다.

❸ 알고 있는 사실을 바탕으로 추론하여 새로운 지식을 생성할 수 있다. 추론에는 연역 추론과 귀납 추론이 있다.

탐구 활동

기관사의 성은 무엇일까?

다음 문제 상황과 조건을 읽고, 직원의 직업과 승객이 사는 지역, 기관사의 성을 추론해 보자.

문제 상황 어떤 기차에 남궁, 독고, 제갈의 성(姓)을 가진 기관사, 승무원, 엔지니어가 타고 있다. 성의 순서대로 해당 직업을 가진 것인지는 알 수 없다. 어느 날 이들과 똑같은 성을 가진 승객 3명이 기차에 탑승했다. 성 앞에 승객을 붙여 같은 성을 구분했다.

조건

① 승객 제갈 씨는 부산에 산다.
② 엔지니어는 부산과 대전의 중간 지역에 산다.
③ 승객 독고 씨는 연봉 2,000만 원을 번다.
④ 남궁 씨는 당구로 승무원을 이긴다.
⑤ 승객 중 한 명은 엔지니어의 이웃이고, 엔지니어 연봉의 정확히 3배가 되는 연봉을 번다.
⑥ 대전에 사는 승객은 엔지니어와 이름이 같다.

예 [1단계] 조건 ①번과 ④번을 통해 승객 제갈 씨가 부산에 산다는 것과 직원 남궁 씨가 승무원이 아니라는 것을 알 수 있다.

직원	기관사	승무원	엔지니어
남궁 씨		×	
독고 씨			
제갈 씨			

승객	대전	중간 지역	부산
남궁 씨			×
독고 씨			×
제갈 씨			○

대단원 정리 및 평가 문제

마인드맵으로 정리하기

선택형

1 인공지능에 대한 설명으로 옳은 것만 고른 것은?

ㄱ. 현재 대부분의 인공지능은 강인공지능이다.
ㄴ. 인공지능은 기존 학습을 바탕으로 추론이나 예측을 할 수 있다.
ㄷ. 튜링 테스트는 엘런 튜링이 제안한 테스트로 아직까지 튜링 테스트를 통과한 인공지능은 없다.

① ㄱ ② ㄴ ③ ㄷ
④ ㄱ, ㄴ ⑤ ㄴ, ㄷ

2 다음은 탐색에 관한 학생들의 대화이다. 옳은 설명을 한 학생만을 있는 대로 고른 것은?

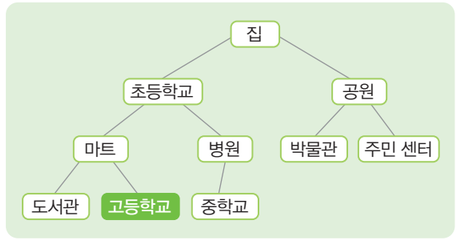
학생 A: 맹목적 탐색은 상태 공간 모두를 탐색해.
학생 B: 정보 이용 탐색에는 깊이 우선 탐색, 너비 우선 탐색 등이 있어.
학생 C: 최상우선탐색은 초기 상태부터 현재 상태에 이르기까지 비용을 중요하게 생각하는 탐색이야.
학생 D: 정보 이용 탐색은 탐색 과정에서 얻은 정보를 활용하기 때문에 대체적으로 좋은 방법을 찾아낸다고 해.

① 학생 A ② 학생 A, 학생 C
③ 학생 A, 학생 D ④ 학생 B, 학생 C
⑤ 학생 B, 학생 C, 학생 D

3 인공지능의 특성 중 이미 알고 있는 사실로부터 새로운 사실을 예측할 수 있는 특성은?

① 인식 ② 학습 ③ 탐색
④ 추론 ⑤ 행동

4 다음은 우리 동네 지도를 트리로 표현한 그림이다. 트리를 각각 깊이 우선 탐색과 너비 우선 탐색으로 탐색했을 때, 〈고등학교〉의 바로 다음에 방문할 장소로 옳은 것을 짝지은 것은?



	깊이 우선 탐색	너비 우선 탐색
①	도서관	마트
②	병원	중학교
③	공원	박물관
④	박물관	주민 센터
⑤	중학교	공원

5 다음 조건을 보고 물음에 답하시오.

[조건]
남학생 'A', 'B', 'C'는 각각 여동생이 있고, 그들의 이름은 '가', '나', '다'이다. 세 쌍의 남매는 혼합 배드민턴 시합을 하려 한다. 단, 친오빠와 동생은 서로 같은 팀이 될 수 없다.
• 첫 번째 시합: (A와 가) vs (C와 나)
• 두 번째 시합: (C와 다) vs (A와 B의 동생)

위 조건으로 알 수 있는 것만을 〈보기〉에서 있는 대로 고른 것은?

[보기]
ㄱ. 제시된 사실로 새로운 사실을 유추하는 '탐색' 문제이다.
ㄴ. C의 동생은 '가'이다.
ㄷ. B가 배드민턴 시합을 하지 않아 B의 동생을 알 수 없다.
ㄹ. A의 동생은 '다'이다.

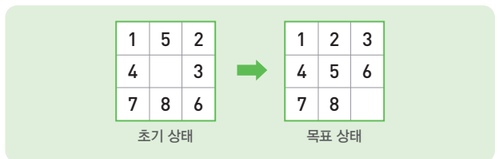
① ㄱ ② ㄴ, ㄷ ③ ㄱ, ㄷ
④ ㄴ, ㄹ ⑤ ㄱ, ㄴ, ㄷ

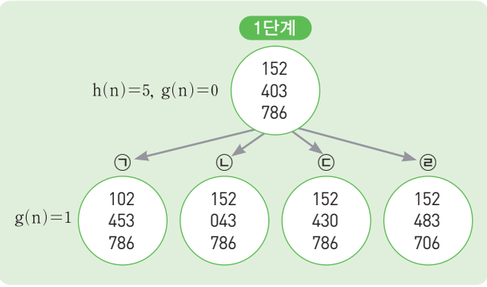
단답형

6 빈칸(㉠~㉢)에 알맞은 낱말을 적으시오.

㉠ ( ), ㉡ ( ), ㉢ ( )

[7~8] 다음은 8 퍼즐의 초기 상태와 목표 상태와 초기 상태에서 가능한 작업을 수행한 후 다음 상태를 나타낸 상태 공간 트리다.





7 위의 트리를 A*로 탐색할 때, 각 노드의 평가 함수를 쓰시오. (단, h(n)은 타일이 제자리에 있지 않은 수이고, g(n)은 노드가 있는 트리의 높이다.)

㉠ ( ) ㉡ ( )
㉢ ( ) ㉣ ( )

서술형

8 목표 상태에 이를 때까지의 상태 공간 트리를 완성하시오.

9 다음 [조건]을 읽고, 철이, 돌이, 민이의 직업을 맞히시오.

[조건]
철이, 돌이, 민이의 직업은 축구 선수, 야구 선수, 농구 선수 중 서로 다른 하나다.
① 철이는 야구 선수가 아니다.
② 민이는 농구 선수보다 나이가 많다.
③ 야구 선수는 돌이보다 나이가 적다.