Computational Thinking and Algorithms

컴퓨터개론

(Introduction to Computer Systems)

GEN1030

컴퓨팅 사고력

컴퓨팅 사고력

- 컴퓨터가 문제를 해결하는 방식처럼 복잡한 문제를 단순화하고 이를 논리적, 효율적으로 해결하는 능력

- 컴퓨터 과학의 기본 개념과 원리 및 컴퓨팅 시스템을 활용하여 실생활 및 다양한 학문 분야의 문제를 이해하고 창의적 해법을 구현하여 적용할 수 있는 능력

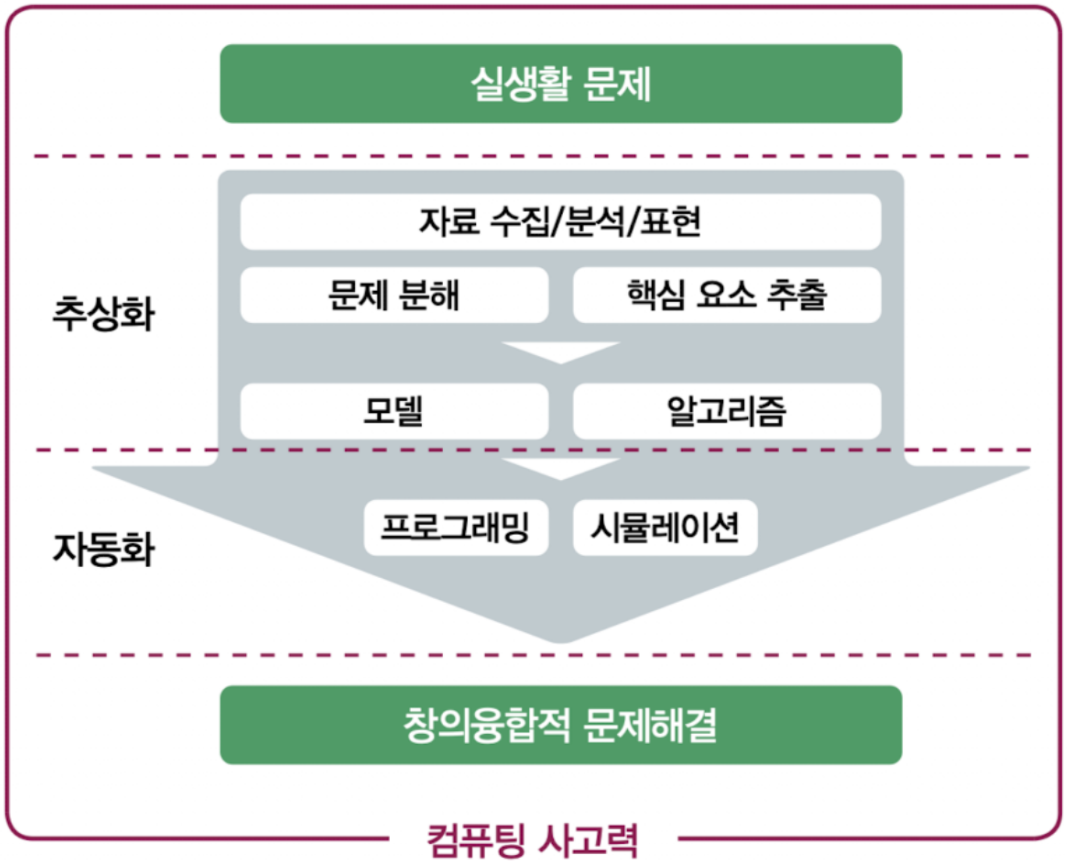
컴퓨팅 사고력 – 필요성

- 컴퓨터 분야의 문제 해결은 물론, 일상생활의 일반적인 문제 해결에 효율적으로 사용될 수 있는 방법을 제공

- 제4차 산업혁명 시대
  - 컴퓨터 기술을 근간으로 여러 산업이 융합하여 새로운 문제를 해결

- '프로그래밍 언어의 사용하여 프로그램을 작성하는 과정'인 코딩 능력을 말하는 것이 아님
  - 컴퓨팅 사고 능력을 키워 일반적인 문제에 적용·활용하는 것이 중요

컴퓨팅 사고력 – 구성 요소


실생활 문제
추상화
자료 수집/분석/표현
문제 분해
핵심 요소 추출
모델
알고리즘
자동화
프로그래밍
시뮬레이션
창의융합적 문제해결
컴퓨팅 사고력

컴퓨팅 사고력 – 코딩

- 컴퓨팅 사고력 키우는 방법 중 하나는 직접 프로그래밍을 하는 코딩(Coding)

- 코딩
  - 해결하고자 하는 문제를 주어진 절차와 방법으로 표현하는 것
  - 전문적 프로그래머(Programmer)가 되기 위한 것이 아니라, 논리적인 사고력 창의력, 문제 해결 능력 등을 키울 수 있음

컴퓨팅 사고력 준비

자료 수집, 분석, 표현

자료 수집

- Data collection
  - 문제를 해결하기 위한 해답을 찾기 위해 적절한 자료 수집하는 과정
  - 설문 조사, 인터뷰 같은 의견 청취
  - 공공데이터 또는 빅데이터 수집 등

자료	내용
자료의 정의	사람, 동물, 물건, 기계 등이 만들어 낸 사실이나, 뜻, 관측 결과 등을 숫자, 문자, 기호 등으로 표현한 것
자료의 종류	양적인 자료(값을 표현하는 자료)
	개수, 관측값 등으로 표현할 수 있는 양적인 자료로, 보통 실험이나 관찰, 설문 조사의 객관식 문제나 응답이 짧은 주관식 문제, 검색 등의 방법으로 수집할 수 있다. • 실험 자료의 관찰 및 기록 • 설문 조사를 통한 응답 결과 • 데이터베이스 검색
	질적인 자료(생각, 느낌 등을 표현하는 자료)
	글, 그림, 응답 등으로 표현되는 질적인 자료가 있다. 예를 들면, 인터뷰를 할 때 그 내용을 '녹음한 음성'이나, '그 내용을 기록한 글', '설문의 서술형 응답' 등과 같이 개인의 느낌이나 생각 등을 자유롭게 표현한 형태의 자료를 말한다. • 인터뷰 시 녹음한 음성이나 기록한 글 등 • 설문 조사의 서술형 응답

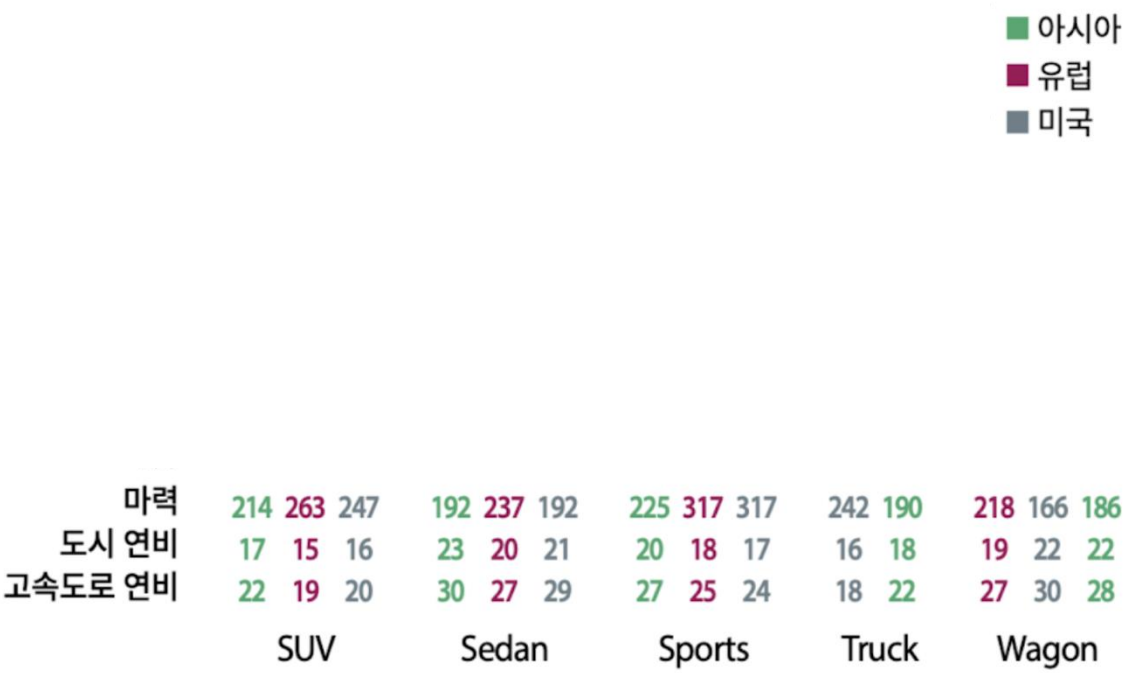
자료 분석과 표현

- Data analysis
  - 자료의 의미를 이해하고, 자료에서 특징을 찾아 일반화 하거나, 자료가 가진 패턴을 찾아 자료의 흐름이나 동향을 파악하는 과정
  - 적절한 자료 분석 방법을 선택하는 것이 중요
    - 자료의 다양한 통계처리 방법에 대한 지식을 필요

- Data representation
  - 자료가 가지고 있는 관계나 특성을 파악하고 적절한 표현 방법으로 시각화하는 과정
  - 다양한 방법 존재
    - 테이블, 차트, 계층 구조, 그래프, 트리 구조 등

다양한 자료 표현

- 테이블과 차트

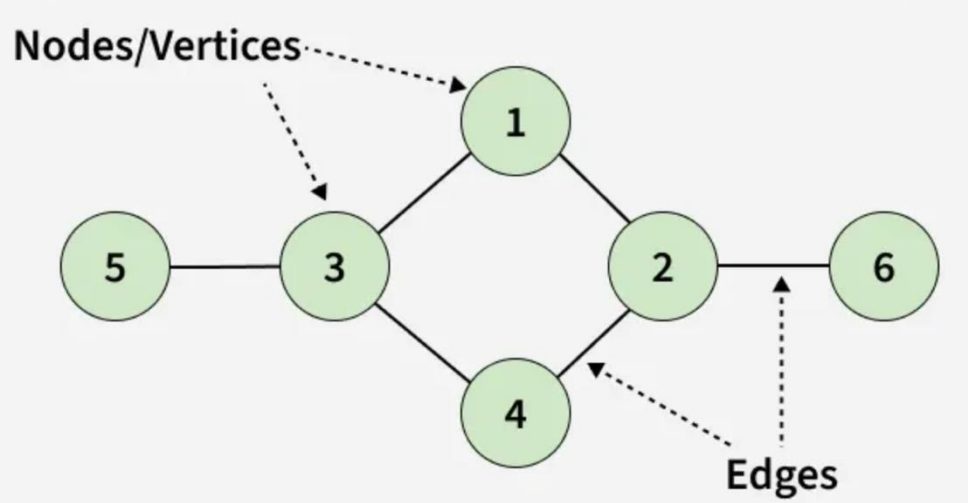


다양한 자료 표현

- 계층 구조

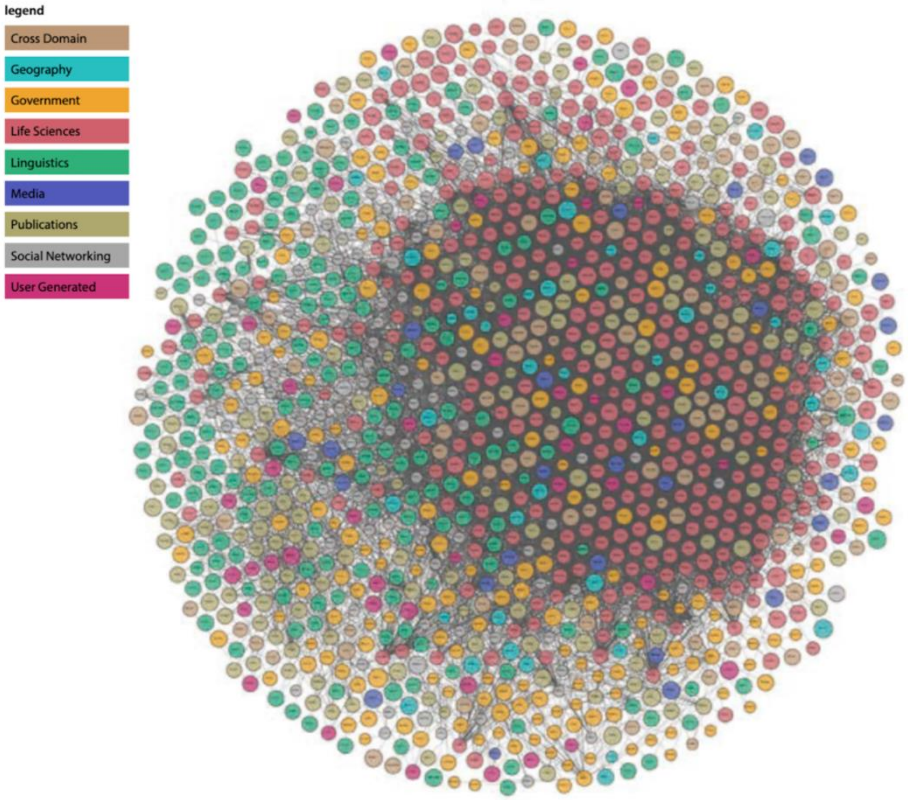
다양한 자료 표현

- 그래프(Graph)
  - 노드(Node): 데이터 저장하는 단위
  - 엣지(Edge): 노드와 노드를 연결하는 선
    - 방향이 있거나(Directed) 값을 가질 수 있음(Weighted)



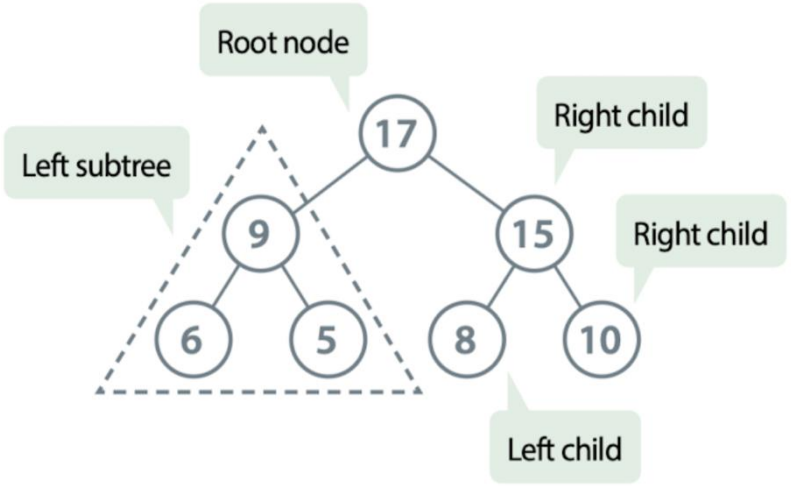
다양한 자료 표현

- 그래프(Graph)
  - 노드(Node): 데이터 저장하는 단위
  - 엣지(Edge): 노드와 노드를 연결하는 선
    - 방향이 있거나(Directed) 값을 가질 수 있음(Weighted)



다양한 자료 표현

- 트리(Tree) 구조
  - 그래프의 한 종류
  - 여러 개의 가지(branch)가 생겨나는 나무 모양 형상화
  - 계층 구조를 가짐
    - 부모-자식 관계가 명확함
  - 사이클이 없음
    - 특정 노드에서 출발해 다시 그 노드로 돌아오는 경로 없음
  - 노드 간의 관계나 수를 제한하여 이진 트리, 순서 트리, 균형 트리 등 다양한 구조로 표현 가능

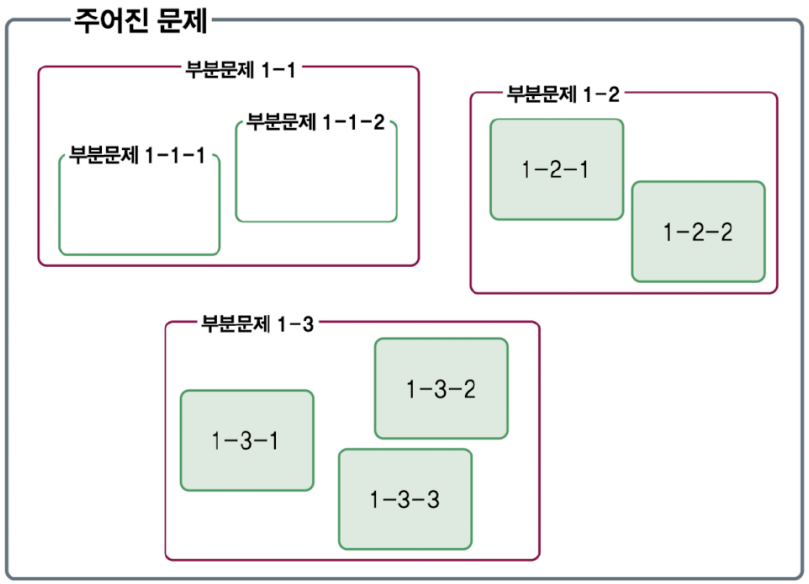


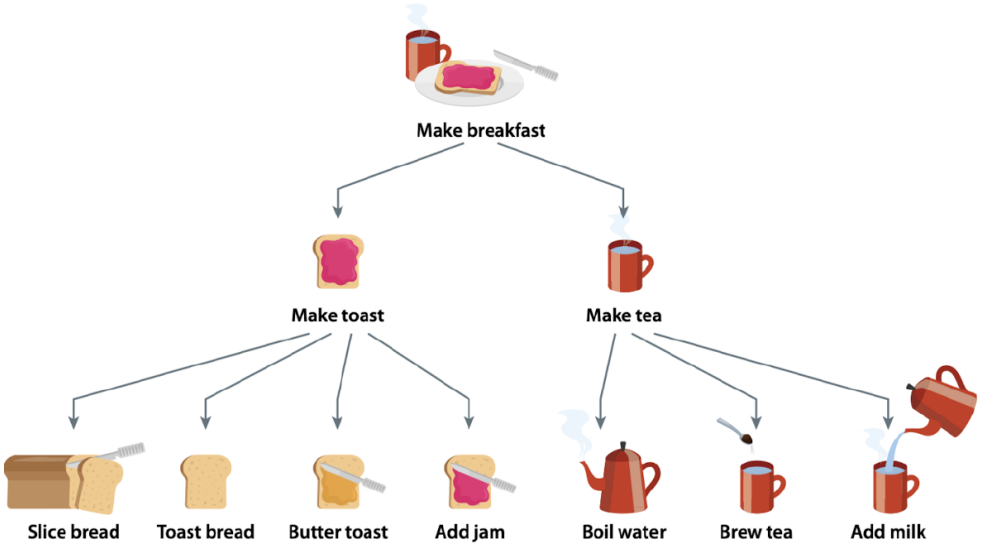
컴퓨팅 사고력 시작

문제 분해

문제 분해

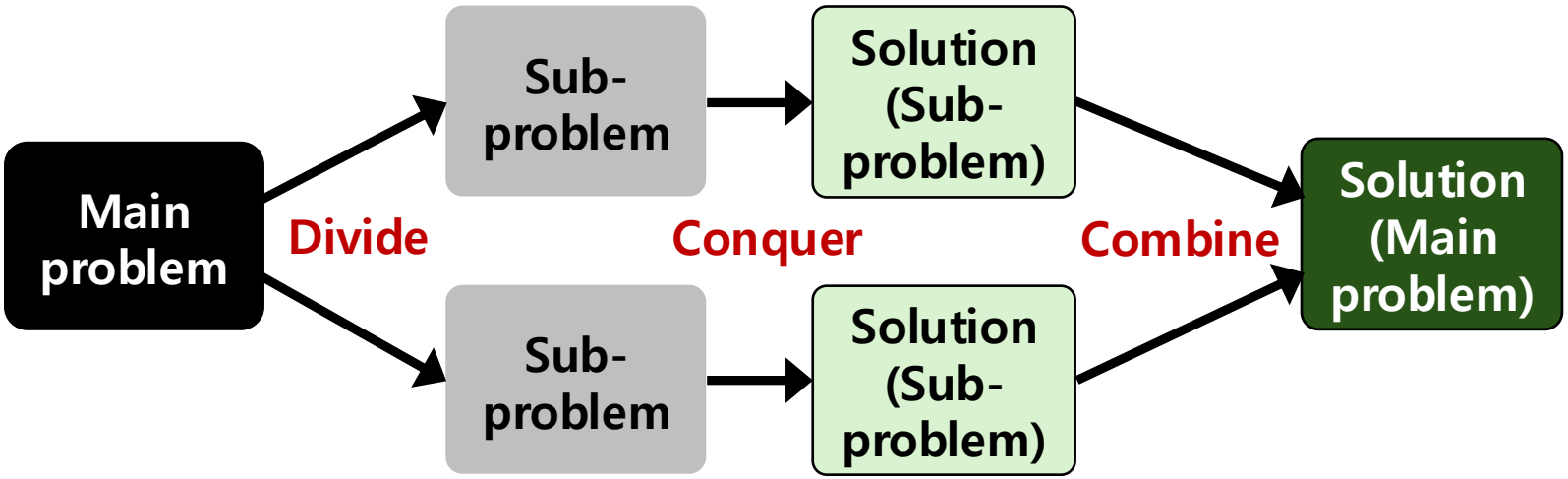
- Problem Decomposition
  - 문제를 작은 부분으로 나누고, 나눠진 부분은 다시 해결 가능한 부분으로 잘게 나누는 과정
  - 복잡한 문제를 작은 문제로 쪼개어 이해하고 해결
    - 분해된 작은 문제가 관리 가능하고 해결 가능하면 분해 성공
    - 해결 가능하다고 판단되지 않으면 더 작게 나눔





분할과 정복

- Divide and Conquer
  - 분할(Divide)
    - 분할 가능한 2개 이상의 작은 문제로 나눔
  - 정복(Conquer)
    - 나뉜 문제를 다시 작은 문제로 나눔
    - 혹은 문제를 해결
  - 통합(Combine)
    - 해결한 작은 문제를 통합



분할과 정복(Divide and Conquer)

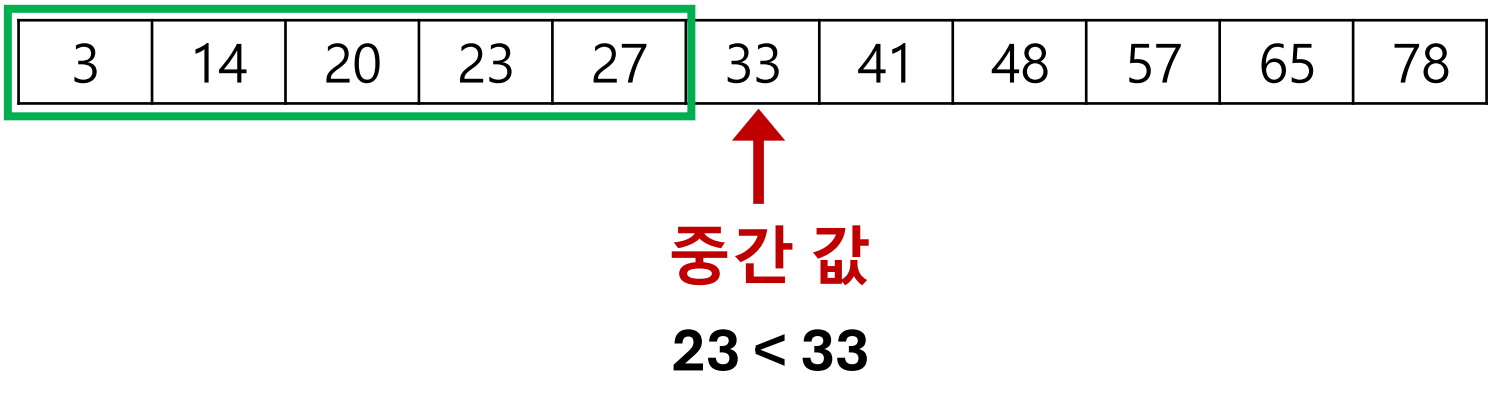
- Binary search (이진검색)
  - 정렬된 리스트에서 탐색 범위를 줄여 나가면서 검색 값을 찾는 알고리즘
  - 데이터가 정렬(오름차순 또는 내림차순)되어 있어야 함

- Binary search 과정
  - 찾는 값과 중앙에 위치한 값을 비교
  - if (찾는 값 == 중앙 값)
    - 종료
  - else if (찾는 값 < 중앙 값)
    - 중앙 값 왼쪽 그룹에 대해 다시 binary search 수행
  - else
    - 중앙 값 오른쪽 그룹에 대해 다시 binary search 수행

분할과 정복(Divide and Conquer)

- Binary search (이진검색)

찾는 값: 23



분할과 정복(Divide and Conquer)

- Binary search (이진검색)

찾는 값: 23

분할과 정복(Divide and Conquer)

- Binary search (이진검색)

찾는 값: 23

중앙 위치?
- 홀수: 중앙
- 짝수: 왼쪽 공간에서 가장 큰 수

분할과 정복(Divide and Conquer)

- Binary search (이진검색)

찾는 값: 23

중앙 위치?
- 홀수: 중앙
- 짝수: 왼쪽 공간에서 가장 큰 수

분할과 정복(Divide and Conquer)

- Binary search (이진검색)

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] > target:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```

arr: 탐색 배열, target: 찾는 값

탐색 범위 결정하는 인덱스(index) 지정
low: 왼쪽 끝, high: 오른쪽 끝

low가 high보다 커지면 탐색 범위 없는 것이므로 종료

중앙 인덱스 계산:
정수 나눗셈(//) 사용해서 소수점 버림
→ 짝수 개 남았을 경우 왼쪽 공간에서 가장 큰 수 선택

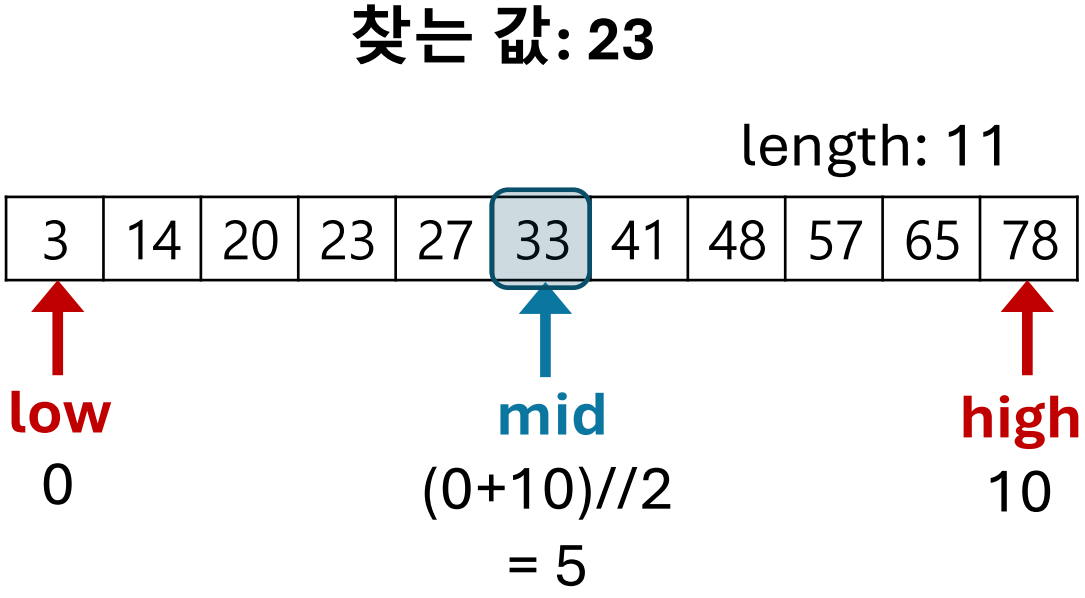
반복문 끝날 때까지 return 되지 않음
→ 찾는 데이터가 배열에 없음

분할과 정복(Divide and Conquer)

- Binary search (이진검색)

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] > target:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```

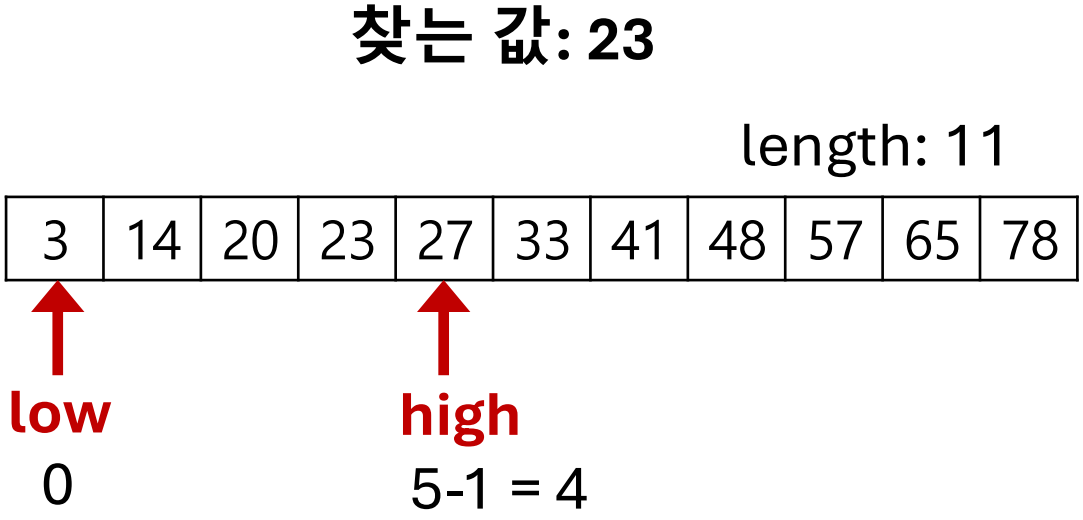


분할과 정복(Divide and Conquer)

- Binary search (이진검색)

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] > target:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```



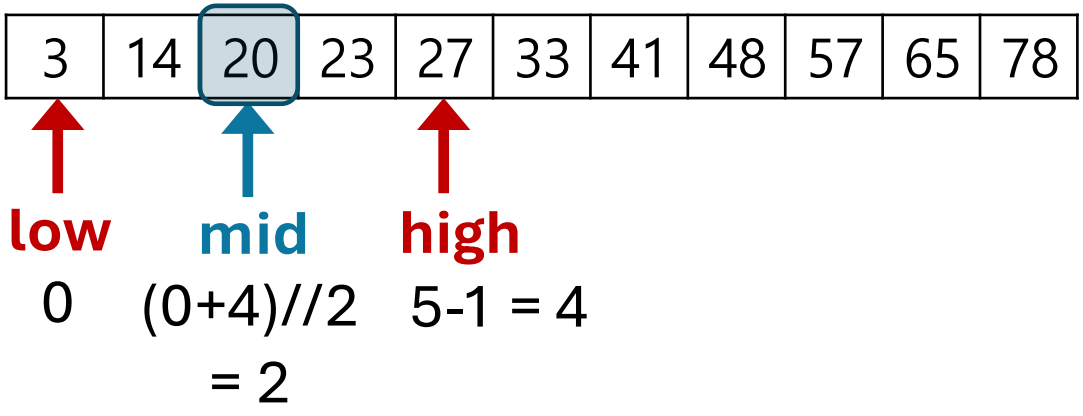
분할과 정복(Divide and Conquer)

- Binary search (이진검색)

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] > target:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```

찾는 값: 23

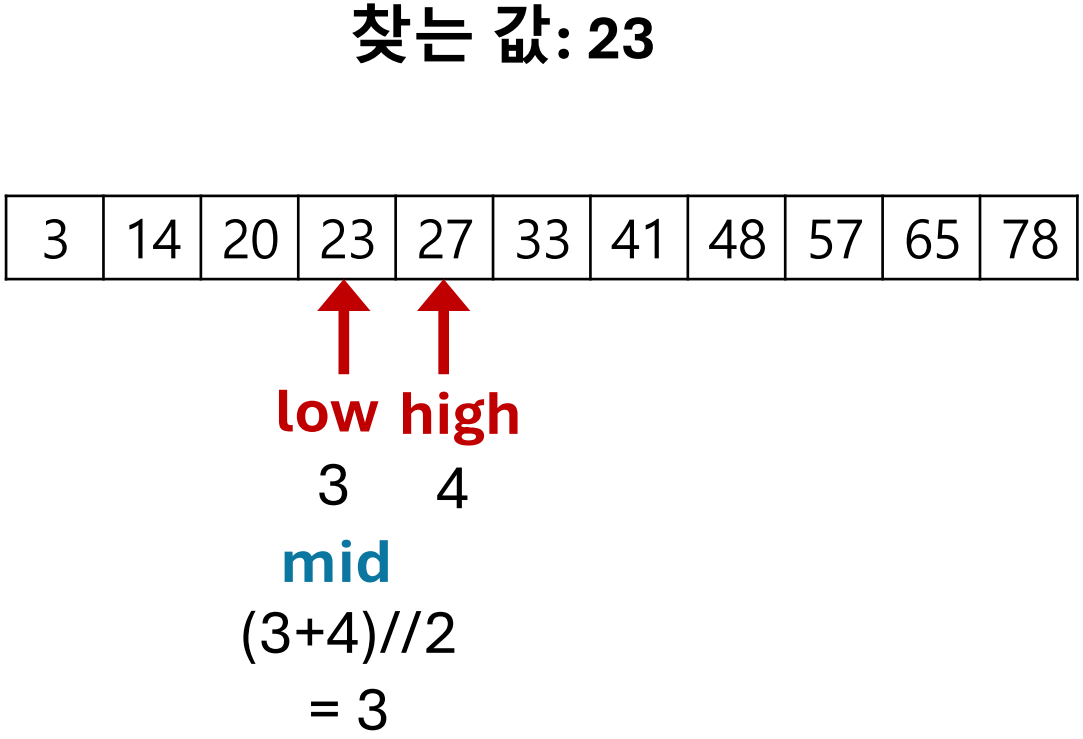


분할과 정복(Divide and Conquer)

- Binary search (이진검색)

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] > target:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```

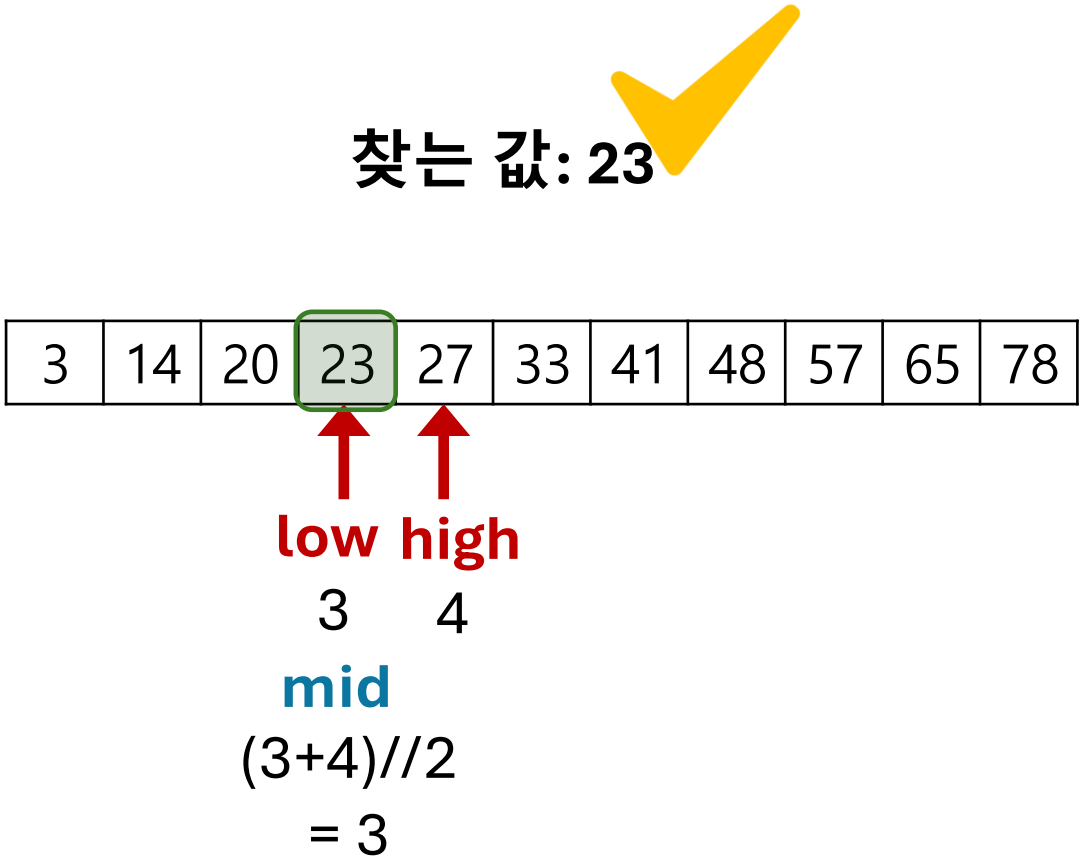


분할과 정복(Divide and Conquer)

- Binary search (이진검색)

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] > target:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```



컴퓨팅 사고력 핵심

패턴인식과 추상화

패턴 인식

- 복잡한 문제 분해하면 분해된 작은 문제들 사이에서 패턴(pattern)이 있을 수 있음
  - 패턴? 문제에서 나타나는 일정한 경향이나 반복적 규칙 또는 공통된 특성

- Pattern Recognition
  - 작게 분해된 문제들 사이에서 패턴을 탐색하는 과정
  - 패턴은 복잡한 문제를 단순하게 만들 수 있음
  - 단계
    - 문제 내의 개별 요소 식별
    - 공통적인 요소나 기능을 식별
    - 식별된 패턴을 설명/기술
    - 확인된 패턴을 기반으로 예측

추상화

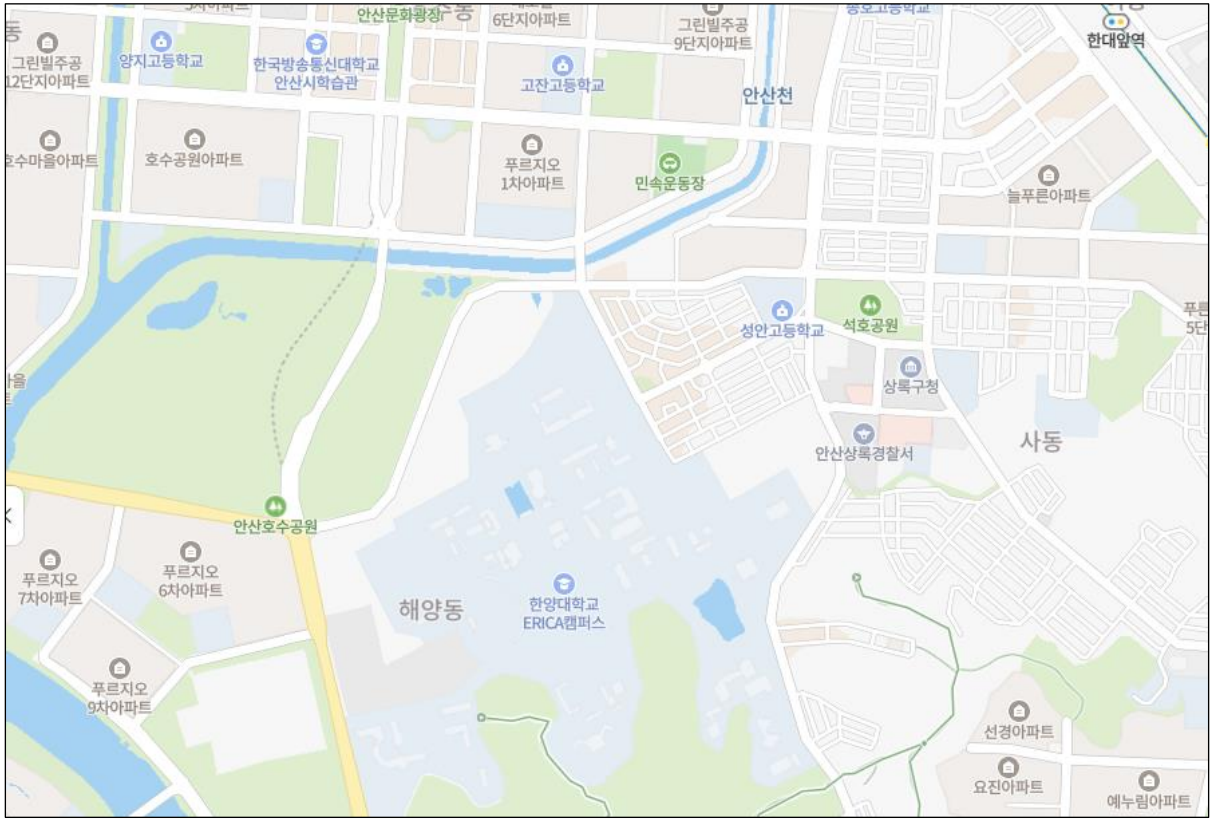
- Abstraction
  - 여러 가지 사물이나 개념에서 공통되는 특성, 속성 따위를 추출하여 파악하는 작용

- 컴퓨팅 사고력에서의 추상화
  - 핵심 요소와 개념 또는 기능을 간추려 일반화된 모델(generalized model)을 만드는 과정

  → 문제를 단순화시켜 불필요한 부분은 제거

추상화 예시

- 지도
  - 실제 지형 모습을 간결화



추상화 예시

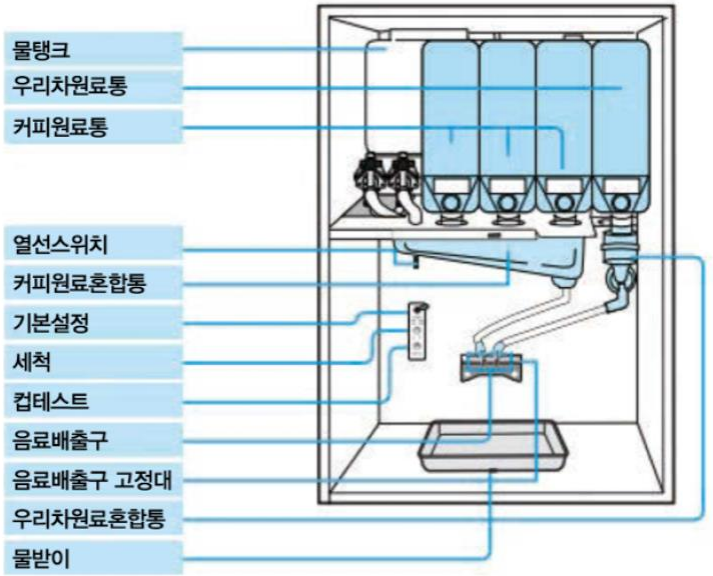
- 지하철 노선도
  - 구체적인 도로나 거리, 건물 등의 정보는 제거하고 역과 역을 연결한 선으로만 정보 표현

추상화 예시

- 동일한 문제라도 문제를 보는 관점에 따라 상이할 수 있음
- 커피 머신

일반 사용자

커피 머신 관리자



컴퓨팅 사고력 구현

알고리즘과 자동화

알고리즘(Algorithm)

- 문제 해결을 위해 추상화된 핵심 원리를 일련의 절차로 표현하는 과정
  - 문제 해결 방법을 정의한 일련의 단계적 절차
  - 일련의 절차? 순차적 표현, 반복적 표현, 조건 표현 등

- 일련의 절차 표현 방법
  - 순서도(flow chart)
  - 자연언어(natural language)
  - 의사코드(pseudo code)
  - 프로그래밍 언어(programming language) 등

- 알고리즘 예시
  - 라면 조리법 (자연언어)

알고리즘 사례

- 친구에게 집까지 오는 방법 설명
  - 방법 4개

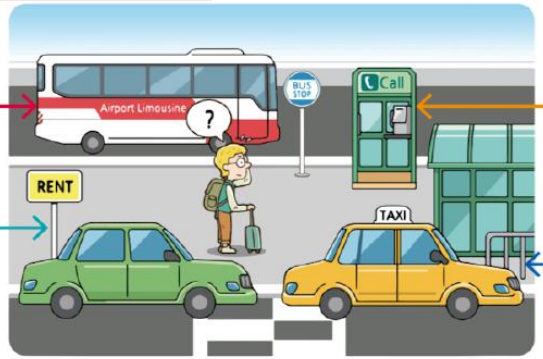
첫 번째는 버스를 타는 것이다.

1. 수화물을 찾은 다음 1120번 버스를 탄다.
2. 종로에서 250번 버스로 갈아탄다.
3. 홍대입구에서 내린다.
4. 9번 출구로 나와서 500 미터쯤 걸어서 도착한다.

두 번째는 내게 전화하는 것이다.

1. 공항에 도착해서 내게 전화한다.
2. 수화물을 찾은 다음에 공항에서 나를 만난다.

목적지 도착이라는 결과는 같지만 과정은 모두 다름 **"각각 다른 비용과 시간 소요"**



주어진 상황에 맞는 **가장 효율적인** 방법을 선택

세 번째는 렌트카를 사용하는 것이다.

1. 렌터카 회사까지 셔틀을 타고 간다.
2. 차를 빌린다.
3. 우리 집까지 경로를 따라 차를 몰고 온다.

네 번째는 택시를 이용하는 것이다.

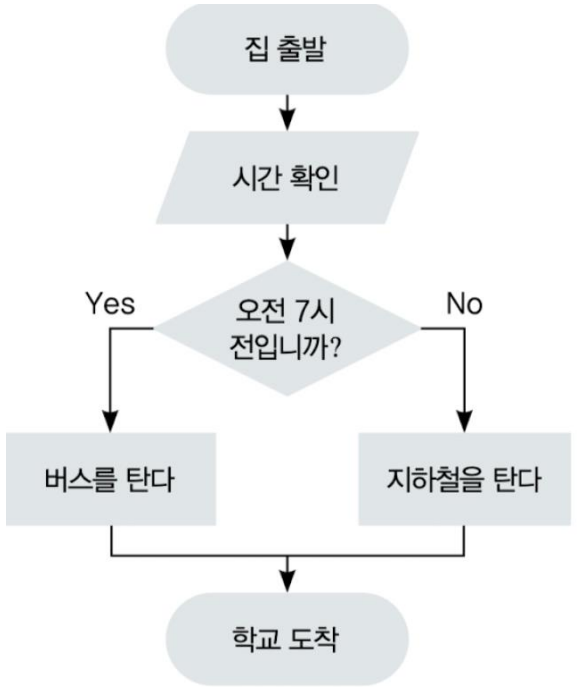
1. 택시 승강장에서 택시를 기다린다
2. 택시를 탄다.
3. 택시기사에게 우리집 주소를 알려준다.

순서도

- Flow chart
  - 표준화된 기호 및 도형을 사용해 데이터의 흐름과 수행되는 절차들의 순서를 표현하는 방법
  - 복잡한 알고리즘 표현시에는 그림도 복잡해질 수 있음

기호	기능
	터미널 순서도의 시작과 끝을 표시
	판단 여러 가지 경로 중 하나의 경로 선택을 표시
→	**흐름선** 처리간의 연결 기능을 표시
	서류 서류를 매체로 하는 입출력 표시
	수동입력 콘솔에 의한 입력

기호	기능
	처리 각종 연산, 데이터 이동 등을 처리
	입 · 출력 데이터의 입력 및 출력 표시
	연결자 흐름이 다른 곳과 연결되는 입출구를 나타냄
	준비 기억장소, 초기값 등 작업의 준비 과정을 나타냄
	천공카드 천공카드의 입출력



<예시 1>

의사코드(Pseudo code)

- 알고리즘을 프로그래밍 언어와 유사한 형식으로 자유롭게 기술한 코드
  - 특정 프로그래밍 언어의 문법에 얽매이지 않음
  - 우리가 사용하는 자연 언어를 사용할 수 있음

- Example) 두 수 중 큰 값 찾기

```
알고리즘: FindMax(a, b)
입력: 숫자 a, b
출력: 둘 중 더 큰 값

1. 만약 a가 b보다 크다면:
2.     a를 결과로 선택
3. 그렇지 않다면:
4.     b를 결과로 선택
5. 선택된 결과를 반환
```

의사코드(Pseudo code)

- 두 수 중 큰 값 찾기

알고리즘: FindMax(a, b)
입력: 숫자 a, b
출력: 둘 중 더 큰 값

1. 만약 a가 b보다 크다면:
2. a를 결과로 선택
3. 그렇지 않다면:
4. b를 결과로 선택
5. 선택된 결과를 반환

Algorithm FindMax(a, b)
Input: Number a, b
Output: Number result (the larger of the two numbers)

1. IF a > b THEN
2. SET result TO a
3. ELSE
4. SET result TO b
5. END IF
6. RETURN result

Python code

```
def find_max(a, b):
    if a > b:
        result = a
    else:
        result = b
    return result
```

의사코드(Pseudo code)

- 1부터 N까지 합계 구하기

```
알고리즘: SumUpToN(n)
입력: 양의 정수 n
출력: 1부터 n까지 더한 결과 sum

1. sum 변수를 0으로 초기화
2. i 변수를 1부터 n까지 1씩 증가시키며 반복:
3.     sum = sum + i
4. 반복이 끝나면 sum의 최종값을 출력
```

```
Algorithm SumUpToN(n)
Input: A positive integer n
Output: The sum of all integers from 1 to n

1. SET sum TO 0
2. FOR i FROM 1 TO n DO
3.     sum = sum + i
4. END FOR
5. RETURN sum
```

의사코드(Pseudo code)

- Binary search

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] > target:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```

Python code

```
Algorithm BinarySearch(A, target)
Input: A sorted array A, and a target value
Output: The index of the target if found, otherwise -1

1.  SET low TO 0
2.  SET high TO (length of A - 1)
3.
4.  WHILE low <= high DO
5.      SET mid TO (low + high) / 2  (truncate fraction)
6.
7.      IF A[mid] == target THEN
8.          RETURN mid
9.      ELSE IF A[mid] > target THEN
10.         high = mid - 1
11.     ELSE
12.         low = mid + 1
13.     END IF
14. END WHILE
15. RETURN -1
```

자동화(Automation)

- 프로그래밍 도구나 자동화 도구, 또는 모의실험(Simulation)을 이용해 알고리즘에서 찾은 일련의 과정을 수행하여 문제해결의 결과를 확인하는 과정
  - 알고리즘을 구현하는 프로그래밍 단계

- 적절한 프로그래밍 언어 선택 및 구현
  - Ex) Python, Java, Go, …

Summary

- 자료 수집, 분석, 표현
- 문제 분해
  - Divide and Conquer
- 패턴 인식, 추상화
- 알고리즘, 자동화