알고리즘과 자료구조 워크북

교과목명 : 알고리즘과 자료구조
차시명: 1차시 알고리즘 개요

◈ 담당교수: 신일훈 **(서울과학기술대학교)**

◉ 세부목차
- 알고리즘의 개념
- 알고리즘의 조건
- 알고리즘의 표현
- 알고리즘의 효율성분석
- 알고리즘의 설계 기법

학습에 앞서

■ 학습개요

컴퓨터를 활용한 문제 해결의 핵심은 문제 해결을 위한 알고리즘을 설계하는 것이다. 본 장에서는 먼저 알고리즘이란 무엇이며 알고리즘이 갖추어야 하는 특성에는 어떤 것들이 있는지를 구체적인 예시와 함께 설명한다. 또한 설계된 알고리즘을 표현하는 방법에는 어떤 것들이 있는지, 그리고 알고리즘의 효율성과 정확성은 어떻게 분석하고 검증하는 지를 설명한다. 마지막으로 알고리즘 설계를 위해 활용할 수 있는 문제 해결 전략들에 대해서 소개한다.

■ 학습목표

1	알고리즘의 개념과 알고리즘이 갖추어야 하는 조건을 설명할 수 있다.
2	알고리즘의 표현 방법을 설명할 수 있다.
3	알고리즘의 정확성을 검증하는 방법을 설명할 수 있다.
4	알고리즘의 효율성을 분석하기 위한 시간복잡도의 개념을 이해하고, 주어진 알고리즘에 대한 시간복잡도를 계산할 수 있다.

■ 주요용어

용어	해설
알고리즘	어떤 문제의 해답을 구하기 위한 단계적인 절차를 순서대로 명확하게 나타낸 것이다.
의사코드	알고리즘의 표현 방법 중 하나로 프로그램 코드와 유사한 형태이다. 상세한 디테일은 생략 가능하며, 프로그램 문법을 지킬 필요가 없다.

시간복잡도	데이터 크기 N이 증가함에 따라 필요한 연산의 수가 얼마만큼 증가하는가를 나타낸다. 일반적으로 빅오 표기 방식을 사용한다.

학습하기

1. 알고리즘의 개념

- 알고리즘이란 어떤 문제의 해답을 구하기 위한 단계적인 절차를 순서대로 명확하게 나타낸 것이다.
- 알고리즘에 기술된 절차를 순서대로 수행하면 문제의 해답이 구해져야한다.

- 두 변수의 값을 맞교환하는 올바른 알고리즘 예시

```
1. 변수 C를 준비한다.
2. 변수 A의 값을 변수 C에 저장한다.
3. 변수 B의 값을 변수 A에 저장한다.
4. 변수 C의 값을 변수 B에 저장한다.
```

=> 위 알고리즘의 경우, 각 단계를 순서대로 수행하면 변수 A와 변수 B의 값이 올바르게 맞교환된다. 또한 각 단계에서 수행해야 하는 일이 명확하게 기술되어 있다.

- 두 변수의 값을 맞교환하는 잘못된 알고리즘 예시1

```
1. 변수 C를 준비한다.
2. 변수 B의 값을 변수 A에 저장한다.
3. 변수 C의 값을 변수 B에 저장한다.
4. 변수 A의 값을 변수 C에 저장한다.
```

=> 위 알고리즘의 경우, 각 단계에서 수행해야 하는 일은 명확하게 기술되어 있지만, 각 단계를 순서대로 수행하더라도 변수 A와 변수 B의 값이 올바르게 맞교환되지 않는다. 따라서 잘못된 알고리즘이다. 이 알고리즘과 올바른 알고리즘을 비교하면 각 단계의 순서가 매우 중요함을 알 수 있다.

- 두 변수의 값을 맞교환하는 잘못된 알고리즘 예시2

```
1. 변수 A와 변수 B의 값을 맞교환한다.
```

=> 위 알고리즘의 경우, 각 단계에서 수행해야 하는 일이 추상적으로 기술되어 있다. 즉, 어떻게 값을 맞교환해야 하는지 분명하지 않다. 따라서 적절한 알고리즘으로 보기 어렵다.

2. 알고리즘의 조건(특성)

가. 문제

- 알고리즘을 구성하는 각 명령의 의미는 모호하지 않고 명확해야 한다.

나. 유한성

- 알고리즘은 일정한 시간 내에 종료되어야 한다. 즉, 무한루프를 포함하면 안 된다.
- 무한루프를 포함한 잘못된 알고리즘 예시

```
sum = 0
num = 1
while (True) :
        sum += num
        num += 1
```

=> 위 알고리즘의 경우, 포함된 while 문이 종료되지 않고 무한 루프에 빠지게 된다. 따라서 잘못 작성된 알고리즘이다.

다. 유효성

- 컴퓨터에서 실행 가능한 형태로 작성해야 한다.

라. 효율성

- 효율적인 (빠르고 메모리 사용량이 적은) 알고리즘일수록 가치가 높다.
- 1 ~ N까지의 합계를 구하는 알고리즘1

```
1. sum = 0
2. num = 1
3. sum = sum + num
4. num = num + 1
5. if (num <= N) then go to step 3
6. otherwise print(sum)
```

=> 위 알고리즘의 경우, 각 단계에서 수행해야 하는 일이 명확하게 기술되어 있으며 컴퓨터에서 실행 가능하고 일정한 시간 내에 종료되는 올바르게 작성된 알고리즘이다. 다만, 3-5번 단계의 연산이 약 N번 반복되므로, N이 커지면 반복 횟수도 함께 비례하여 증가한다.

- 1 ~ N까지의 합계를 구하는 알고리즘1

```
1. sum = (N * (N+1)) / 2
2. print(sum)
```

=> 위 알고리즘의 경우, 각 단계에서 수행해야 하는 일이 명확하게 기술되어 있으며 컴퓨터에서 실행 가능하고 일정한 시간 내에 종료되는 올바르게 작성된 알고리즘이다. 또한 N의 크기와 관계없이 1-2번 단계를 한 번만 수행하고 종료된다. 따라서 알고리즘1에 비해 더 효율적인 알고리즘이다.

3. 알고리즘의 표현 방법

가. 자연어(한글, 영어 등)로 표현

- 자연어로 표현된 알고리즘 예시

```
1. 변수 C를 준비한다.
2. 변수 A의 값을 변수 C에 저장한다.
3. 변수 B의 값을 변수 A에 저장한다.
4. 변수 C의 값을 변수 B에 저장한다.
```

=> 위 알고리즘의 경우, 각 단계가 자연어인 한글로 표현되어 있다. 각 단계의 의미가 명확하고 해당 단계를 프로그램으로 작성할 수 있으므로 올바르게 표현된 알고리즘이다.

나. 흐름도(flow chart)로 표현

- 흐름도는 의미가 정해진 기호를 사용하여 알고리즘을 도식화하여 표현한다.
- 프로그램에 익숙치 않은 사람이라도 흐름도를 이해할 수 있는 장점이 있다.
- 알고리즘이 복잡해지면 흐름도가 지나치게 비대하고 복잡해질 수 있으므로 복잡한 알고리즘을 표현하

기에는 적합하지 않다.

- 흐름도로 표현된 알고리즘 예시

=> 위 알고리즘은 변수 A와 변수 B의 값을 맞바꾸는 알고리즘을 흐름도로 표현하였다.

다. 의사코드 (pseudo-code)로 표현

- 프로그램 코드와 유사한 형태로 알고리즘을 작성.
- 의미가 이해되는 범주 내에서 프로그램 언어의 문법을 엄격하게 지킬 필요가 없음.
- 논문 등에서 알고리즘을 표현할 때 널리 활용됨.
- 의사코드로 표현된 알고리즘 예시

```
C <= A
A <= B
B <= C
```

=> 위 알고리즘은 변수 A와 변수 B의 값을 맞바꾸는 알고리즘을 의사코드로 표현한다. <= 기호는 값을 대입하는 연산자를 의미한다. 따라서 =로 대체하여 표현해도 무방하다.

라. 프로그램 언어로 표현

- 파이썬과 같은 프로그램 언어로 알고리즘을 작성할 수 있다.
- 컴퓨터에서 알고리즘을 바로 실행할 수 있는 장점이 있지만, 표현된 알고리즘이 지나치게 상세하고 비대할 수 있는 문제가 있다.
- 다만, 스크립트 언어인 파이썬의 경우, C나 자바에 비해 단순하고 명료하게 알고리즘을 표현하는 것이 가능하다.
- 파이썬으로 표현된 알고리즘 예시

```
C = A
A = B
B = C
```

=> 위 알고리즘은 파이썬으로 표현된 변수 A와 변수 B의 값을 맞바꾸는 알고리즘이다. 단순한 알고리즘이긴 하지만 파이썬 표현과 의사코드 표현이 별반 차이가 없음을 알 수 있다.

4. 알고리즘의 정확성 검증

가. 실험적 분석 (테스트)

- 다양한 입력값을 이용해 예상된 결과가 도출되는지를 검증한다.
- 동치(동등) 분할 입력과 경계값 입력 기법 등을 활용한다.
- 점수를 학점으로 변환하는 알고리즘의 정확성 검증 예시
  - 80 ~100점까지가 A학점이라고 한다면, 동치 분할 테스트를 적용하여 중간값이 90을 입력하여 A가 도출되는지를 검증한다.
  - 또한 경계값 입력 기법을 적용하여 하단 경계값인 80을 입력하여 A가 도출되는지를 검증하고 상단 경계값인 100을 입력하여 A가 도출되는지를 검증한다. 또한 79를 입력하여 A가 도출되지 않는지를 검증하고 101을 입력하여 A가 도출되지 않는지를 검증한다. 많은 경우, 중간값보다는 경계값

에서 오류가 발견된 확률이 높다.
- 테스트를 통해 오류가 발견되지 않는다고 하더라도 정확성을 100% 보장할 수 있는 것은 아니다. 즉 발견되지 않은 오류가 있을 수 있다.

나. 증명적 분석
- 수학적 귀납법과 같은 수학적인 방법을 활용하여 알고리즘의 정확성을 검증할 수 있다.
- 증명이 쉽지 않는 단점이 있으나, 정확성을 100% 보장할 수 있다.

5. 알고리즘의 효율성 분석

가. 효율성의 종류
- 시간 효율성은 알고리즘이 얼마나 빠른 시간 안에 결과를 도출하는가를 의미한다.
- 공간 효율성은 알고리즘이 얼마나 많은 메모리를 사용하는가를 의미한다.

나. 시간복잡도
- 시간 효율성은 일반적으로 데이터의 입력 크기에 따른 시간복잡도(time complexity)로 표현한다.
- 시간복잡도는 데이터의 입력 크기에 따른 연산의 수로 표현된다.
- 즉, 데이터 크기 N이 증가함에 따라 필요한 연산의 수가 얼만큼 증가하는가를 나타내는 개념이다.

다. 시간복잡도 예시 (빅오 표기)
- O(1) => 연산의 수가 데이터 크기 N과 관계없이 일정함을 의미한다.
- O(logN) => 연산의 수가 데이터 크기 N이 증가함에 따라 logN에 비례하여 증가함을 의미한다.
- O(N) => 연산의 수가 데이터 크기 N이 증가함에 따라 N에 비례하여 증가함을 의미한다.

라. 시간복잡도 비교
- O(1) 알고리즘의 효율성이 가장 우수하다.

마. 시간복잡도 종류
- 최선(best case) 시간복잡도: 입력값이나 조건에 따라 알고리즘의 시간복잡도가 다른 경우 가장 효율성이 높은 경우의 시간복잡도를 의미한다.
- 평균(average) 시간복잡도: 평균적인 경우의 알고리즘의 시간복잡도를 의미한다.
- 최악(worst case) 시간복잡도: 가장 최악의 경우에 알고리즘이 보이는 시간복잡도를 의미한다. 즉 알고리즘은 최악 시간복잡도 이상의 효율성을 갖는다고 볼 수 있다.

바. 알고리즘들의 시간복잡도 예시
- O(1) 알고리즘 예시

```
sum = (N * (N+1)) / 2
print(sum)
```

=> 위 알고리즘은 1 ~ N까지의 합계를 구하는 알고리즘이다. N의 크기와 상관없이 수행해야 하는 연산의 수가 항상 일정하다. 따라서 O(1)의 시간복잡도를 갖는다.

- O(N) 알고리즘 예시

```
sum = 0
num = 1
while (num <= N) :
        sum += num
print(sum)
```

=> 위 알고리즘은 1 ~ N까지의 합계를 구하는 알고리즘이다. while 문의 반복 횟수가 N에 비례한다. 따라서 O(N)의 시간복잡도를 갖는다.

사. 전체 알고리즘이 여러 단계의 알고리즘으로 구성된 경우의 시간복잡도

- 전체 알고리즘이 여러 단계의 알고리즘으로 구성된 경우의 전체 시간복잡도는 각 단계의 시간복잡도 중 가장 큰 값에 의해 결정된다.
- 가령 알고리즘이 3단계로 구성되어 있고, 각 단계의 시간복잡도가 각각 O(1), O(N), O(logN)이라면 전체 알고리즘의 시간복잡도는 O(N)이다.

6. 알고리즘의 설계 기법

- brute-force(억지) 기법
- 축소정복(decrease and conquer)
- 분할정복(divide and conquer)
- 동적 프로그래밍
- 탐욕 기법
- 몬테카를로 시뮬레이션
- ...
- 각 알고리즘 설계 기법은 이후 강의에서 보다 상세하게 학습하도록 한다.

연습문제

1. 알고리즘을 정의하시오.

정답 : 어떤 문제의 해답을 구하기 위한 단계적인 절차를 순서대로 명확하게 나타낸 것.
해설 : 알고리즘의 목적은 문제의 해답을 찾는 것이며, 목적을 달성하기 위한 절차를 순서대로 명확하게 나타내야 한다.

2. 알고리즘이 갖추어야 하는 조건을 나열하시오.

정답 : 명확성, 유한성, 유효성, 효율성
해설 : 알고리즘은 의미가 모호하지 않고 명확해야 하며, 일정한 시간 안에 종료되어야 한다. 또한 현재의 기술을 사용하여 구현 가능해야 하고, 가능한 빠르고 자원을 적게 소모하는 알고리즘이 우수하다.

3. 서로 다른 N개의 숫자가 무작위로 저장된 파이썬 리스트에서 최솟값을 찾는 알고리즘의 최악의 시간복잡도를 구하시오. (빅오 표기)

정답 : O(N)
해설 : 최솟값을 첫 번째 값으로 초기화한 후, 나머지 모든 원소와 현재의 최솟값을 비교하고 현재의 최솟값이 더 크다면 이를 비교한 원소값으로 수정한다. 즉 N개의 연산이 필요하다. 따라서 최악의 시간복잡도는 O(N)이다.

정리하기

1. 알고리즘은 어떤 문제의 해답을 구하기 위한 단계적인 절차를 순서대로 명확하게 나타낸 것이다.
2. 알고리즘의 조건은 명확성, 유한성, 유효성, 효율성 등이다.
3. 알고리즘의 효율성은 시간복잡도 등으로 표현하며 시간복잡도는 데이터 입력크기 N이 증가함에 따라 필요한 연산의 수가 무엇에 비례하는지를 나타낸다.
4. 알고리즘의 설계 기법에는 brute-force, 축소정복, 분할정복, 동적프로그래밍, 탐욕 기법 등이 있다.

참고자료

- 파이썬 알고리즘, 최영규, 2021, 생능출판

다음 차시 예고

- 알고리즘 설계 기법 중 억지기법(brute-force) 전략에 대해 학습한다.