1.1 다음 중 입력이 반드시 필요하다고는 볼 수 없는 알고리즘은?

① 배열에서 최댓값을 찾는 알고리즘
② 두 수의 최대 공약수를 구하는 알고리즘
③ 평면상에서 두 점 사이의 거리를 구하는 알고리즘
④ 5개의 로또 번호 예측하는 알고리즘 ?

1.2 다음 중 추상 자료형의 설명으로 옳지 않은 것은?

① 사용자들은 추상 자료형이 제공하는 연산만을 사용할 수 있다.
② 사용자들은 추상 자료형을 어떻게 사용하는지를 알아야 한다.
③ 사용자들은 추상 자료형 내부의 데이터를 직접 접근할 수 없다.
④ 사용자들은 추상 자료형 어떻게 구현되었는지 정확히 알아야 이용할 수 있다.

? 인터페이스제공

1.3 다음 중 파이썬에서 실행시간 측정을 위해 사용할 수 있는 모듈은?

① time ② date ③ numpy ④ copy

1.4 알고리즘 시간 복잡도 $O(1)$이 의미하는 것은?

① 컴퓨터 처리가 불가
② 입력 데이터 수가 한 개
③ 수행시간이 입력 데이터 수와 관계없이 일정
④ 알고리즘 길이가 입력 데이터보다 작음

1.5 다음 설명이 옳으면 ○, 틀리면 ×를 표시하라.

① 어떤 알고리즘의 효율성은 사용하는 자료구조와 밀접한 관련이 있다. ○
② 추상 자료형은 어떤 자료들과 자료에 가해지는 연산들을 구체적으로 표시하는데, 어떤(what?) 자료나 연산이 제공되는가 뿐만 아니라 이들이 어떻게(how?) 구현되는가도 정의한다. ×
③ 크기가 n인 배열의 모든 항목의 합을 구하는 알고리즘은 최선과 최악의 경우에 대한 시간 복잡도가 다르다. ○

④ 배열에 같은 항목이 있는지는 판단하는 알고리즘은 최선과 최악의 경우에 대한 시간 복잡도가 다르다. O

⑤ 배열에 같은 항목이 있는지는 최악의 경우에도 $O(n)$에 판단할 수 있다. X

1.6 다음의 시간 복잡도 함수를 빅오 표기법으로 나타내라.

(1) $T(n)=n^2+10n+8$

(2) $T(n)=n^3+10000n^2+50n$

(3) $T(n)=n^2\log_2 n+n^3+3$

(4) $T(n)=7(2^n)+3^n$

(5) $T(n)=3^n+n!$

1.7 다음의 빅오 표기법들을 실행시간이 적게 걸리는 것부터 나열하라.

$O(1)$ $O(n)$ $O(n^2)$ $O(n^3)$ $O(\log n)$ $O(n\log n)$ $O(n!)$ $O(2^n)$

$O(1) < O(\log n) < O(n$ [redacted] $) < O(n\log n) < O(n^2) < O(n^3) <$

1.8 다음 알고리즘의 시간 복잡도를 빅오 표기법으로 나타내라. $O(2n^2) < O($ [redacted] $n!)$

```
def algorithm1(n) :
```

```
def algorithm2(n) :
```

1.10 다음과 같은 순환적인 프로그램에서 sub(3)과 같이 호출할 때 함수 sub()가 호출되는 횟수는?

```
def sub(n) :
    if n <= 1 :
        return n
    return sub(n-1) + sub(n-2)
```

1.11 다음 함수에서 asterisk(5)와 같이 호출할 때 출력되는 *의 개수는?

```
def asterisk(i) :
    if i > 1 :
        asterisk(i/2)
        asterisk(i/2)
    print("*", end='')
```

자료구조와 알고리즘

○ 일상생활에서 자료를 정리하고 조직화 하는 이유

- 찾거나 사용할때 편리함을 추구
- 방대하고 복잡한 자료들을 효율적으로 처리하기위해
- 결론 편리 와 효율을 위해

자료구조란?

- 데이터에 편리하게 접근하고, 변경하기 위해서 데이터를 저장하거나 조직하는방법
- 자료를 효율적으로사용하기 위해 자료의 특성(응용)에 따라 분류하여 구성하고 저장 및 처리하는 모든 작업(메모리)
- 자료들을 정리하고 조직
- 단일 자료구조 : 숫자, 문자(등)
- 복잡 자료구조 : 여러 자료들을 한꺼번에 보관하는 컨테이너와같은구조

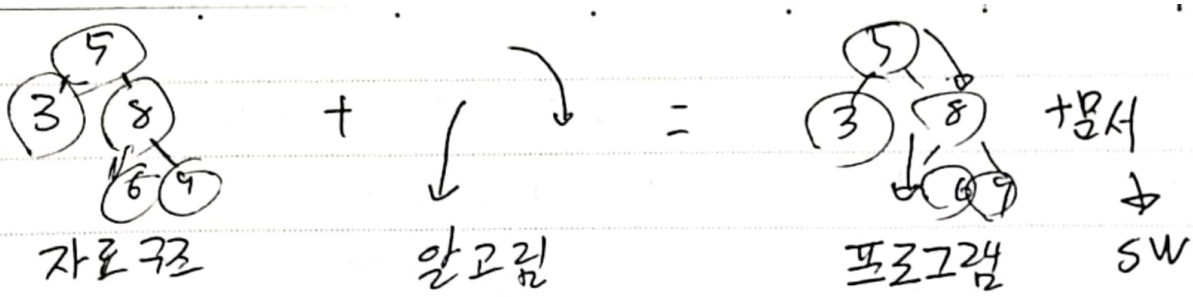
○ 컴퓨터에서의 자료구조 (Data structure)

- 컴퓨터에서 자료를 정리하고 조직화 하는 다양한 구조 (복합자료구조)

선형 자료구조	비선형 자료구조
-항목들을 순차적으로 나열하여 저장하는 창고 -항목 접근 방법에 따라 세분화 ○ 리스트 : 가장 자유로운 선형자료구조 스택, 큐, 덱 : 항목의 접근이 맨앞 아 맨뒤로제한	항목들이 보다 복잡한 연결 관계 트리 : 회사의 조직도나 컴퓨터의 폴더와 같은계층구조 -그래프 : 가장 복잡한 연결관계를 표현

○ 알고리즘

- 데이터는 자료구조로 표현, 이를 이용하여 문제를 해결하기 위한 효과적인 절차론
- 어떤 문제를 해결하는 절차(조직) / 좁은 의미 → 알고리즘
- 어떤 값을 입력을 받아서 원하는 값으로 출력하는 잘 정의된 계산 절차



〈자료구조 & 알고리즘〉

↳데이터구조 (Data structure)

• 자료를 효율적으로 사용 → 자료의 특성(종류)에 따라서 보관 → 구성하고 저장및 처리하는 모든 작업(데이터표현) /Array, List, Queue, Tree

↳컴퓨터알고리즘 (computer Algorithm)

◦ 컴퓨터를 이용하여 주어진 문제를 해결하기위한 방법이나 절차

• Sorting, Hashing

↳컴퓨터언어 (computer Language)

◦ 컴퓨터와 대화하기 위해서 사용하는 언어

• C, C++, C#, Java

↳컴퓨터 프로그램 (computer Program)

◦ 컴퓨터가 특정 작업을 수행하기 위해 작성된 명령수행

-

프로그램? => 자료구조+ 알고리즘

예)최대값 탐색 프로그램 = 배열 + 순차탐색

자료구조 알고리즘

〈자료구조역할〉

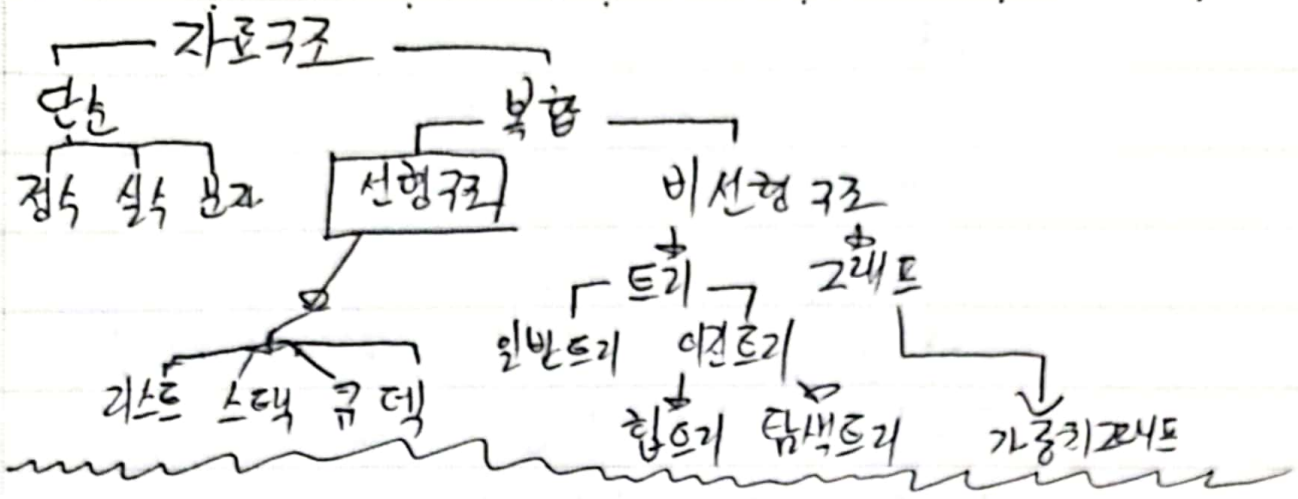
-컴퓨터 프로그래밍에 있어서 가장기초적인 학문분야

-컴퓨터프로그램의 기본골격

◦프로그램이 효율적이고 안전하게 동작하게 하기 위해서 반드시 필요

대형 프로젝트의 최종단계

→구조적인 결함 발생



알고리즘조건

1. 입력 → 0개이상 입력
2. 출력 → 1개 이상 출력
3. 명백성 → 모호하지않고 명확 (각 명령어 의미)
4. 유한성 → 한정된 수의 단계 후에는 반드시종료
5. 유효성 → 각 명령어들은 실행 가능한 연산이어야 함.

기술방법

1. 자연어로 표현
2. 흐름도로 표현 (flowchart)
3. 유사코드로 표현 (Pseudo-code)
4. 프로그래밍으로 표현

〈 추상 자료형 (ADT) 〉 ← 객체지향

→ 프로그래머가 추상적으로 정의한 자료형 ←

DATA → 무언 → 어떻게 / 자료구조를 표현하는 대표적 방법

자료구조가 어떤 자료를 다루고 이들에 대해 어떤 연산이 제공되는지 기술

뇌 → 추상화기능

기억할 대상의 구현되는 특징 만 추출 단순화 하여 저장하는 기능

→ 구현의 세부사항 공개하지않는다.

인터페이스 통해 어떤 연산이 가능한지를 알려줌

목적

- 데이터 구조의 핵심적인 내용을 간략하고, 복잡한 내부구현부분을 감추기 위해서
- 크고 복잡한 문제를 단순화 시켜 쉽게 해결하기 위함
- 정보은닉 정보은닉
  - 중요한 정보만 나타내고 중요하지않은건 숨김
- 데이터 구조를 사용 → 인터페이스정의
  - 이 데이터구조의 연산 들의 정의

〈Bag 추상자료형〉 (예시)

(데이터)

→ 중복된 항목을 허용하는 자료들의 저장소

→ 항목 간의 비교 가능해야함

(연산)

- Bag() → 비어있는 가방을 만듦
- insert(e) → 가방에 e를 넣는다
- remove(e) → 가방에 e가 있는지검사 후 있으면 항목을 꺼낸다
- contains(e) → 있으면 True 없으면 False 반환
- count() → 가방에 들어있는 항목들 수를 반환

알고리즘의 성능분석

기법

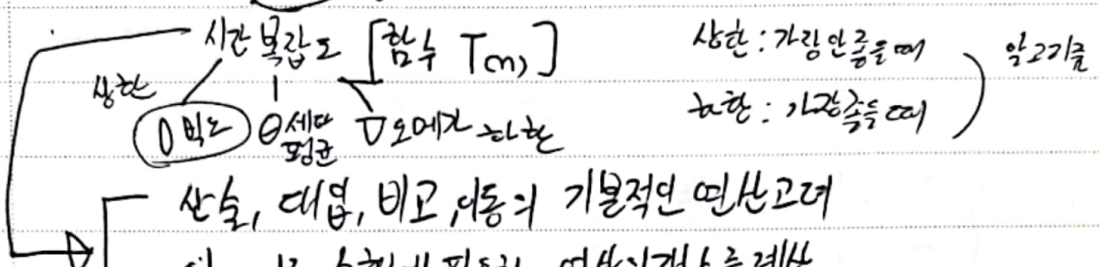
- 실행시간측정
  - 두개의 알고리즘의 실제 실행시간측정
  - 실제로 구현하는것이 필요
  - 동일한 소프트웨어 환경
- 알고리즘의 복잡도분석
  - 직접구현하지않고서 분석
  - 알고리즘이 수행하는 연산횟수측정비교
  - 시간복잡도분석 : 수행 시간 분석
  - 공간복잡도분석 : 필요한 메모리공간 분석

성능분석 기법

기준 정확성, 명확성, 수행량, 메모리사용량, 최적성 등 있음

- 정확성 올바른 자료입력시 유한한 시간 내에 올바른 결과출력 여부
- 명확성 알고리즘이 얼마나 이해하기 쉽고 명확하게 작성되었는가!
- 수행량 일반적인 연산제외, 알고리즘 특성 나타내는 중요 연산 분석
- 메모리사용량
- 최적성: 가장중요

알고리즘 성능 : 시간 and 공간



빅오 표기법 (상한값)

정의 → 두개의 함수 f(n)과 g(n)이 주어졌을때 모든 $n \geq n_0$에 대해 $|f(n)| \leq c|g(n)|$을 만족하는 상수 c와 n_0가 존재하면 $f(n) = O(g(n))$이다.

특징
- 차수가 가장 큰 항이 절대적인 영향 ← 상한값
- 다른항들은 상대적으로 무시
- 연산의 횟수를 대략적으로 표기한것

예시)
- $f(n)=5 \Rightarrow O(1)$
- $f(n)=2n+1 \rightarrow O(n)$

종류

$O(1) \rightarrow O(\log n) \rightarrow O(n) \rightarrow O(n \log n) \leftarrow O(n^2) \rightarrow O(n^3) \leftarrow O(2^n)$

$\rightarrow O(3^n) \rightarrow O(n!)$ => → 으로 갈수록 값이 높아진다 (상한)

처음부터(상수, 로그, 선형, n log n 선형로그, 2차형, 3차형, 지수형, 팩토리얼)

그외 빅오메가 표기법 (하한값)

→ 두개의 함수 f(n)과 g(n)이 주어졌을때 모든 $n \geq n_0$에 대해 $|f(n)| \geq c|g(n)|$ 만족하는 상수 c와 n_0가 존재하면 $f(n) = \Omega(g(n))$이다.

빅세타 표기법 (평균)

조건이 서로 동일 ⓐ $c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|$를 만족하는 c_1, c_2와 n_0가 존재하면 $f(n) = \Theta(g(n))$이다.

성능차이 (복도)

실행시간
- Worst — 가장 널리 사용됨
- 복도메가 — 계산하기 쉽고 응용에 따라 중요한 의미를 가짐
- average — 계산하기가 상당히 어려움
- 빅세타 best — 의미가 없는 경우도 많음.

재귀 → 자기자신을 다시 호출 ← (순환) 알고리즘 → 반복으로 구현 가능
- 알고리즘이나 함수가 수행 도중에 자기자신을 재호출하여 문제를 해결
- 정의 자체가 순환적으로 되어있는 경우에 적합

(팩토리얼 (n!)
피보나치 (fib(n))
그외 이항계수, 하노이의 탑, 이진탐색

순환알고리즘 특징
- 자기자신을 호출
- 프로그램의 크기를 줄이고 간단하게 작성가능
- 내가 나를 호출하는것
  - 줄어든 단위 작업
- 복잡한 작업을 컴퓨터 처럼하는 방법

베이스케이스 → 가장작은
→ 재귀호출하는 과정에서 반복하다 보면, 한번에 해결할 수 있을 정도로 문제의 단위가 충분히 작아지는 단계

순환과 반복

순환 : O(n)

순환적인 문제에서는 자연스러운 방법

함수호출의 오버헤드

반복 : O(n)

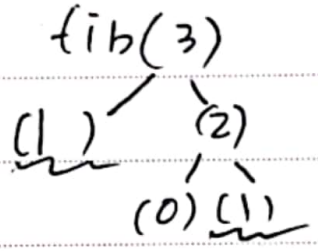
for, while 문 이용, 수행속도가 빠름

순환적인 문제에서는 프로그램 작성이 어려울 수도 있음

→ 대부분 순환은 반복으로 바꾸어 작성할 수 있음

예 순환이 > 반복 = 지능계공계산

느릴때 => 피보나치수열 왜? 같은 항이 중복되어계산!



강의

자료구조 → 역할 개념적인 기본골격, 효율적이고 안전하게

└자료를 정리하고 조직화 => 프레임워크들끼리

└선형, 비선형

알고리즘 = 잘 정의된 계산절차 / 성능기법

조건! 입력, 출력, 명백성, 유한성 유효성 / 명확성, 정확성, 수행량(메모리), 최적성

추상형자료형

└추상적으로 표현한 자료형 / 특징만을 단순화하여...,

정보은닉 = 공개한정보O, 공개하지않는정보X

시간복잡도 (빅오 제일많이씀, 상한값)

O(1)	상수	O(nlogn)	선형log	O(2^n)	지수형
O(logn)	log	O(n^2)	2차형	O(3^n)	?
O(n)	선형	O(n^3)	3차형	O(n!)	팩토리얼