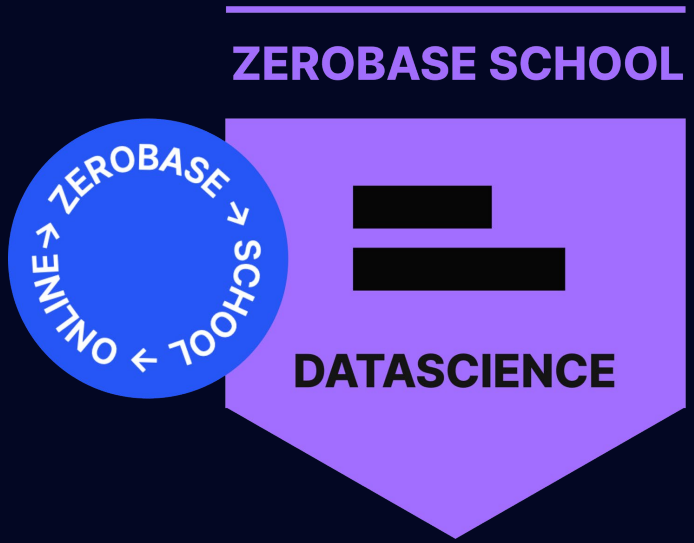
데이터 사이언스

20개 프로젝트를 통해 누구나 배워서 취업하는
데이터 사이언스 온라인 교육 과정



신 제 용

안녕하세요.
전임강사 신제용입니다.

현) zerobase 전임강사
전) LG이노텍 선임

신 제 용

ZEROBASE SCHOOL

코딩테스트 기본

01 알고리즘과 복잡도

02 파이썬과 코딩테스트

신 제 용

ZEROBASE SCHOOL

알고리즘과 복잡도

01 자료구조와 알고리즘
02 알고리즘의 복잡도
03 복잡도의 점진적 표기법

신 제 용

01 자료구조와 알고리즘

자료구조와 알고리즘에 대해 기초적인 내용을
학습합니다.

학습 키워드 - 추상 자료형, 자료구조, 알고리즘

Chapter 01
자료구조와 알고리즘

ZEROBASE SCHOOL

이 수업에서 배울 내용

- 코딩테스트를 대비하기 위한 자료구조와 알고리즘의 기본을 학습합니다.
- 이론과 함께 응용 문제를 해결하는 능력을 기릅니다.
- 실제 코딩테스트에서 사용할 수 있는 파이썬 코드를 다룹니다.

자료구조 (Data structure)

- 자료 값의 모임, 자료 간의 관계, 그리고 자료에 적용할 수 있는 함수나 명령
- 만능인 자료구조는 없으며, 상황에 맞는 자료 구조를 사용해야 한다!
- 자료(Data) - 현실 세계로부터 수집한 사실이나 개념의 값 또는 이들의 집합
- cf) 정보(Information) – 자료를 특정 용도로 사용하기 위해 처리/가공한 것

추상자료형 (Abstract data type)

- 추상자료형은 자료구조와 유사하지만, 구체적인 구현 방법은 정의되어 있지 않다.
- 자료구조를 구현할 때에는 자료구조 자체를 자세히 알아야 하지만, 자료구조를 활용하기 위해서는 추상자료형만 알아도 된다.
- OOP 관점에서 보면 추상자료형은 추상 클래스 (Abstract class) 역할을 한다.

자료구조의 분류 (1)

- 선형 자료구조 (Linear data structure)
  - 배열 (Array)
  - 리스트 (List)
  - 스택 (Stack)
  - 큐 (Queue)
  - 해시 셋 (Hash set)
  - 해시 테이블 (Hash table)

자료구조의 분류 (2)

- 비선형 자료구조 (Nonlinear data structure)
  - 트리 (Tree)
  - 그래프 (Graph)
  - 힙 (Heap, Priority queue)
  - 트라이 (Trie; Re**trie**val tree)

자료구조의 필요성

- 필요한 자료에 효율적으로 빠르게 접근할 수 있게 한다.
- 저장장치를 효율적으로 사용할 수 있게 한다.
- 자료구조 별로 적절한 알고리즘을 기계적으로 적용할 수 있다.
- 동료들과 협업하는 데에 큰 도움이 된다.

알고리즘 (Algorithm)

어떤 문제 해결을 위한 절차나 방법

- 알고리즘의 조건
  - 입력 (Input)
  - 출력 (Output)
  - 명확성 (Definiteness)
  - 유한성 (Finiteness)
  - 효과성 (Effectiveness)

알고리즘의 분류

- 정렬 (Sorting)
- 이진 탐색 (Binary search)
- 투 포인터 (Two pointers)
- 탐욕법 (Greedy)
- 분할 정복 (Divide & conquer)
- 동적계획법 (Dynamic programming)
- 백트래킹 (Backtracking)
- 최단 경로 (Minimum distance path)
- 최소 신장 트리 (Minimum spanning tree)

02 알고리즘의 복잡도

다음 챕터에서는 알고리즘이 무엇인지,
알고리즘의 복잡도가 무엇인지 배웁니다.

Chapter 01
자료구조와 알고리즘

ZEROBASE SCHOOL

02 알고리즘의 복잡도

알고리즘을 평가하는 지표인 시간복잡도와
공간복잡도에 대해 학습합니다.

학습 키워드 – 알고리즘, 시간복잡도, 공간복잡도

Chapter 02
알고리즘의 복잡도

ZEROBASE SCHOOL

복잡도 (Complexity)

- 알고리즘의 성능을 나타내는 척도
- 시간 복잡도 (Time complexity)
  → 알고리즘을 동작하는 데에 필요한 연산의 횟수
- 공간 복잡도 (Space complexity)
  → 알고리즘의 동작에 필요한 메모리의 크기
- 일반적으로 시간 복잡도와 공간 복잡도는 trade-off 관계가 있다.

시간 복잡도 (Time complexity)

- 계산 복잡도 (Computational complexity) 라고도 부르며, 알고리즘의 동작에 필요한 연산의 횟수를 말한다.
  → 여기서 연산은 기초 연산(elementary operation)을 의미한다.

- 시간 복잡도가 낮을 수록 더 알고리즘의 성능(performance)가 좋다고 한다.

공간 복잡도 (Space complexity)

- 알고리즘의 동작에 필요한 메모리(RAM)의 크기
- 일반적으로 알고리즘이 동작하기 위해 필요한 메모리의 최대치 (Peak memory)를 공간 복잡도라고 한다.

03 복잡도의 점진적 표기법

다음 챕터에는 일반적으로 알고리즘 비교를 위해 많이 사용되는
복잡도의 점진적 표기법에 대해 알아봅니다.

Chapter 02
알고리즘의 복잡도

ZEROBASE SCHOOL

03 복잡도의 점진적 표기법

알고리즘을 간편하게 정량적으로 비교하는 대표적인
방법인 점진적 표기법을 배웁니다.

학습 키워드 – 점진적 표기법, Big-O notation

Chapter 03
복잡도의 점진적
표기법

ZEROBASE SCHOOL

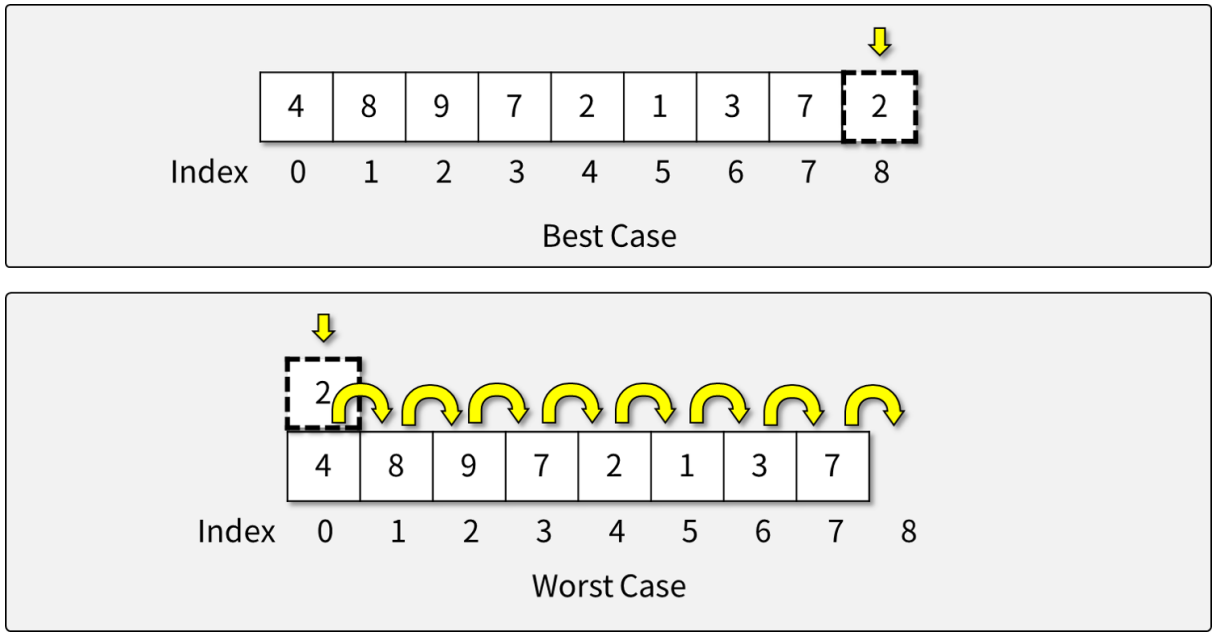
복잡도의 비교

- 절대적으로 성능이 더 좋은 알고리즘이 있을까?
- 복잡도를 비교할 때 단위는 어떻게 정할 것인가?
- 알고리즘 복잡도 비교의 어려운 점
    - 시간 복잡도 vs. 공간 복잡도 trade-off
    - 자료의 크기에 따른 복잡도의 차이
    - 자료의 내용에 따른 복잡도의 차이

복잡도의 분류

- 알고리즘 동작 상황에 따라 구분
  - 최악의 경우
  - 최선의 경우
  - 평균적인 경우
- 일반적으로 **최악의 경우**에 대해 알고리즘 복잡도 정의

최악의 경우 vs. 최선의 경우



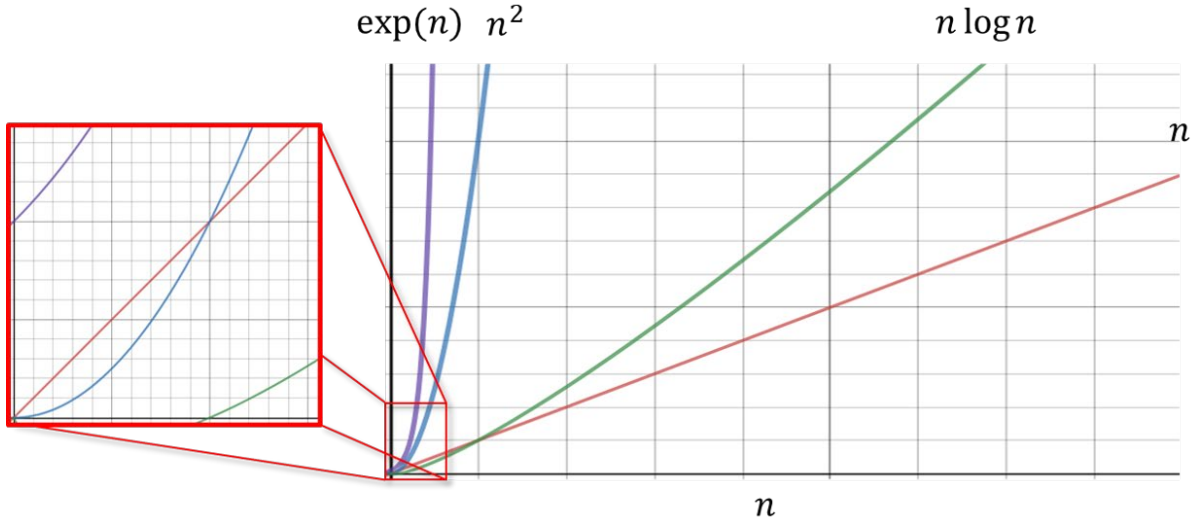
Chapter 03

복잡도의 점진적 표기법

ZEROBASE SCHOOL

점진적 표기법

- 알고리즘에 입력되는 자료의 개수가 충분히 많다고 가정
- 성능 평가에 공평한 비교를 하기 위해 사용



빅오 표기법 (Big-O Notation)

- 충분히 큰 N에서 가장 큰 영향력을 주는 항을 기준으로 표기법 결정

빅오 표기법	설명
$O(1)$	상수 시간
$O(\log N)$	로그 시간
$O(N)$	선형 시간
$O(N \log N)$	로그 선형 시간
$O(N^2)$	이차 시간
$O(2^N)$	지수 시간