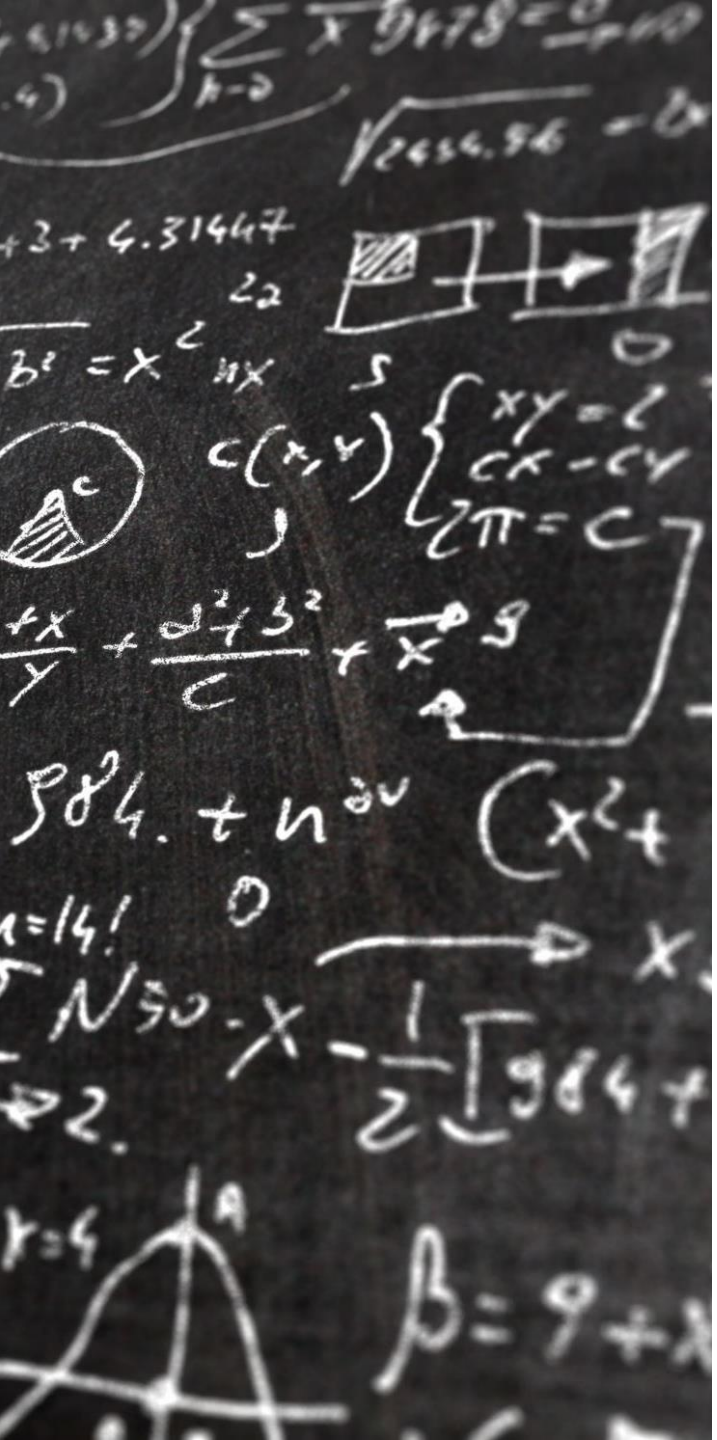
시간 복잡도

매시업 백엔드 9기
김다롬



목차

- 시간복잡도
  - 알고리즘
  - Big-O 표기
- 자료구조별 시간복잡도
- 정리

1. 시간복잡도

1-1 알고리즘

알고리즘은 어떤 목적을 달성하거나 결과물을 만들어내기 위해 거쳐야 하는 일련의 과정들을 의미한다.

알고리즘은 각기 다른 모양과 형태를 지니고 있기에, 시간 복잡도를 설명하는데 자주 사용된다.

시간복잡도를 분석하는 것은 input n 에 대하여 알고리즘이 문제를 해결하는 데에 **얼마나 오랜 시간이 걸리는 지**를 분석하는 것과 같다.

그리고 이는 'Big-O 표기' 를 이용하여 정의할 수 있다.

1-2 BIG-O 표기

- 시간복잡도에서 중요한 것은 정해진 표현식에 가장 큰 영향을 미치는 n 의 단위이다.

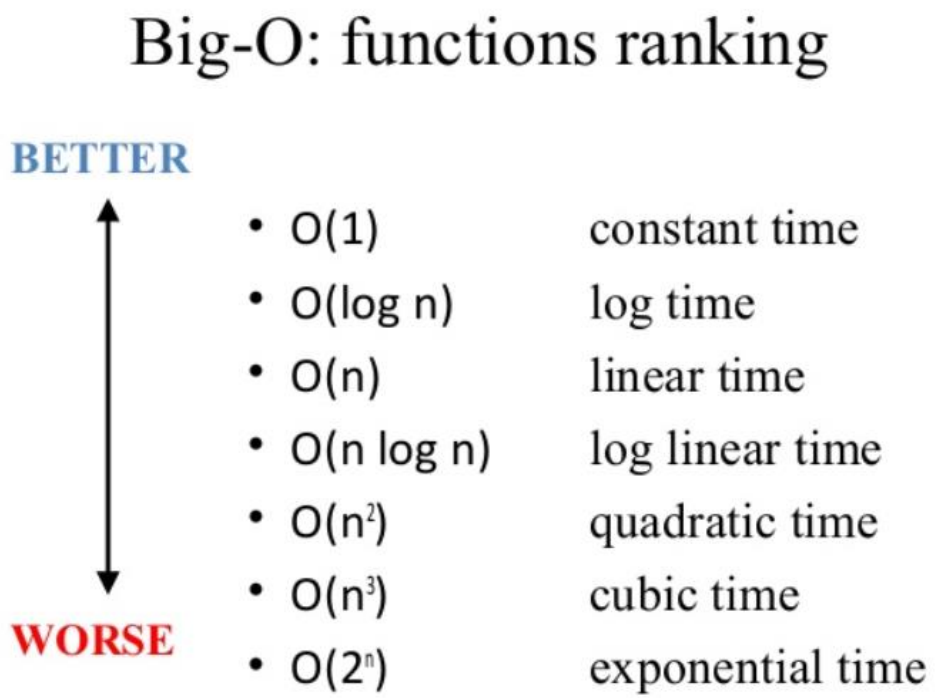
1.O(1) – 상수 시간 : 입력값 n 이 주어졌을 때, 알고리즘이 문제를 해결하는데 오직 한 단계만 거칩니다.

2.O(log n) – 로그 시간 : 입력값 n 이 주어졌을 때, 알고리즘이 문제를 해결하는데 필요한 단계들이 연산마다 특정 요인에 의해 줄어듭니다.

3.O(n) – 직선적 시간 : 문제를 해결하기 위한 단계의 수와 입력값 n이 1:1 관계를 가집니다.

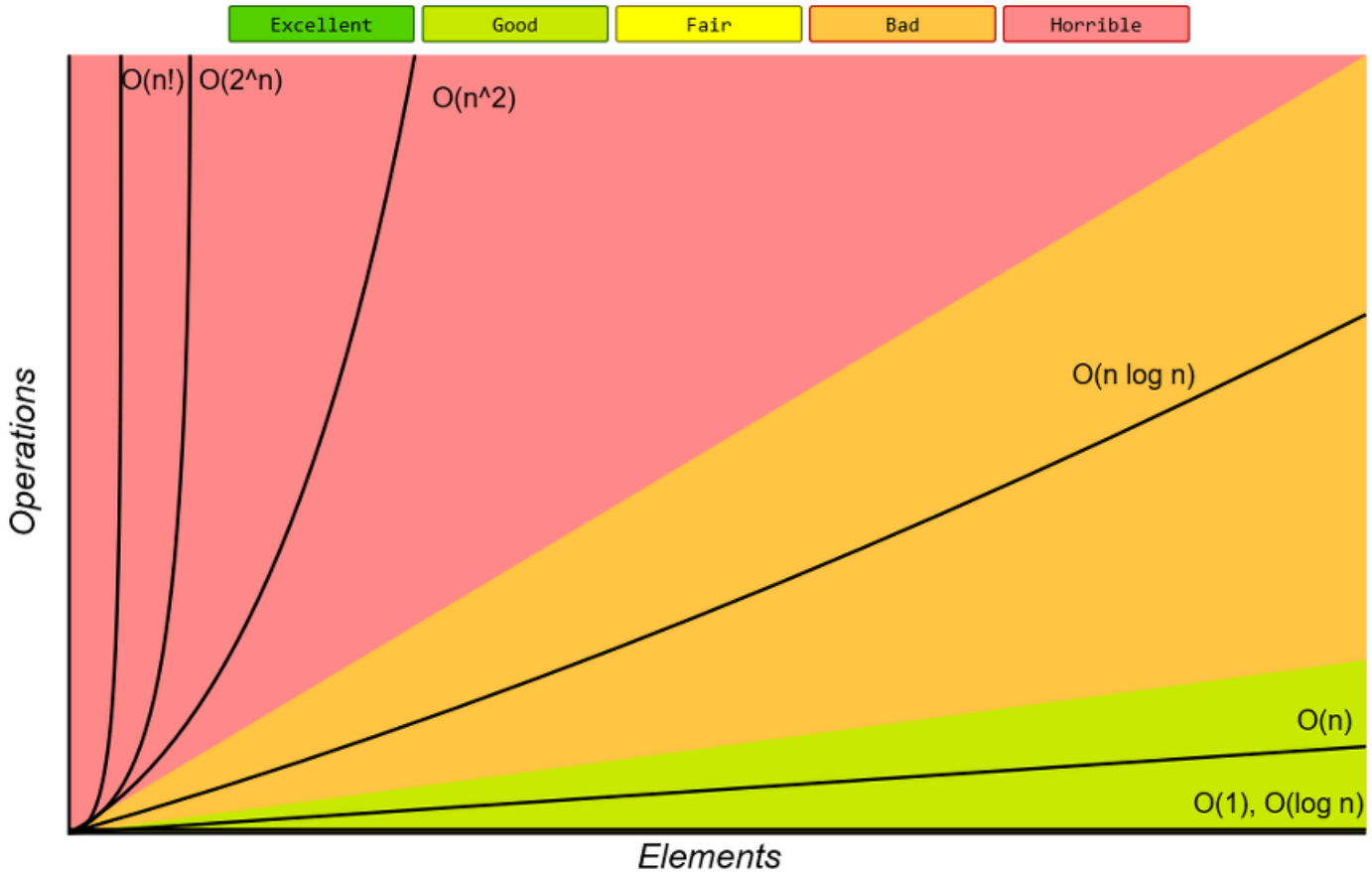
4.O(n^2) – 2차 시간 : 문제를 해결하기 위한 단계의 수는 입력값 n의 제곱입니다.

5.O(C^n) – 지수 시간 : 문제를 해결하기 위한 단계의 수는 주어진 상수값 C 의 n 제곱입니다.



2. 자료구조별 시간복잡도

Big-O Complexity Chart



Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	O(n log(n))	O(n log(n))	O(n^2)	O(log(n))
Mergesort	O(n log(n))	O(n log(n))	O(n log(n))	O(n)
Timsort	O(n)	O(n log(n))	O(n log(n))	O(n)
Heapsort	O(n log(n))	O(n log(n))	O(n log(n))	O(1)
Bubble Sort	O(n)	O(n^2)	O(n^2)	O(1)
Insertion Sort	O(n)	O(n^2)	O(n^2)	O(1)
Selection Sort	O(n^2)	O(n^2)	O(n^2)	O(1)
Tree Sort	O(n log(n))	O(n log(n))	O(n^2)	O(n)
Shell Sort	O(n log(n))	O(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	O(n+k)	O(n+k)	O(n^2)	O(n)
Radix Sort	O(nk)	O(nk)	O(nk)	O(n+k)
Counting Sort	O(n+k)	O(n+k)	O(n+k)	O(k)
Cubesort	O(n)	O(n log(n))	O(n log(n))	O(n)

Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	O(1)	O(1)	O(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Heap Data Structure Operations

Data Structure	Time Complexity					
	Find Max	Extract Max	Increase Key	Insert	Delete	Merge
Binary Heap	O(1)	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(m+n)
Pairing Heap	O(1)	O(log(n))	O(log(n))	O(1)	O(log(n))	O(1)
Binomial Heap	O(1)	O(log(n))	O(log(n))	O(1)	O(log(n))	O(log(n))
Fibonacci Heap	O(1)	O(log(n))	O(1)	O(1)	O(log(n))	O(1)

Graph Data Structure Operations

Data Structure	Time Complexity					
	Storage	Add Vertex	Add Edge	Remove Vertex	Remove Edge	Query
Adjacency list	O(|V|+|E|)	O(1)	O(1)	O(|V| + |E|)	O(|E|)	O(|V|)
Incidence list	O(|V|+|E|)	O(1)	O(1)	O(|E|)	O(|E|)	O(|E|)
Adjacency matrix	O(|V|^2)	O(|V|^2)	O(1)	O(|V|^2)	O(1)	O(1)
Incidence matrix	O(|V| · |E|)	O(|V| · |E|)	O(|V| · |E|)	O(|V| · |E|)	O(|V| · |E|)	O(|E|)

Graph Algorithms

Algorithm	Time Complexity		Space Complexity
	Average	Worst	Worst
Dijkstra's algorithm	O(|E| log |V|)	O(|V|^2)	O(|V| + |E|)
A* search algorithm	O(|E|)	O(b^d)	O(b^d)
Prim's algorithm	O(|E| log |V|)	O(|V|^2)	O(|V| + |E|)
Bellman뺄ord algorithm	O(|E| · |V|)	O(|E| · |V|)	O(|V|)
Floyd-Warshall algorithm	O(|V|^3)	O(|V|^3)	O(|V|^2)
Topological sort	O(|V| + |E|)	O(|V| + |E|)	O(|V| + |E|)

정리

- 시간 복잡도란 문제를 해결할 때 해당 방법(알고리즘)이 문제를 해결하는 데에 얼마나 오랜 시간이 걸리는지 분석하는 것이다.
- 표기는 Big-O 를 사용한다.
- 문제를 해결하려고 할 때마다 시간복잡도를 분석하는 습관을 들이면 좋은 개발자가 될 수 있다고 합니다.
- 문제라는 것은 정답이나 최선의 답의 관점에서 접근하는 것보다, 상황에 더 맞는 답인지 아닌지의 관점에서 접근해야 합니다.

참고문서

- 블로그
  - https://joshuajangblog.wordpress.com/2016/09/21/time_complexity_big_o_in_easy_explanation/
  - https://velog.io/@bathingape/Time-Complexity%EC%8B%9C%EA%B0%84%EB%B3%B5%EC%9E%A1%EB%8F%84
- 시간복잡도 표
  - https://www.bigocheatsheet.com/

Q & A