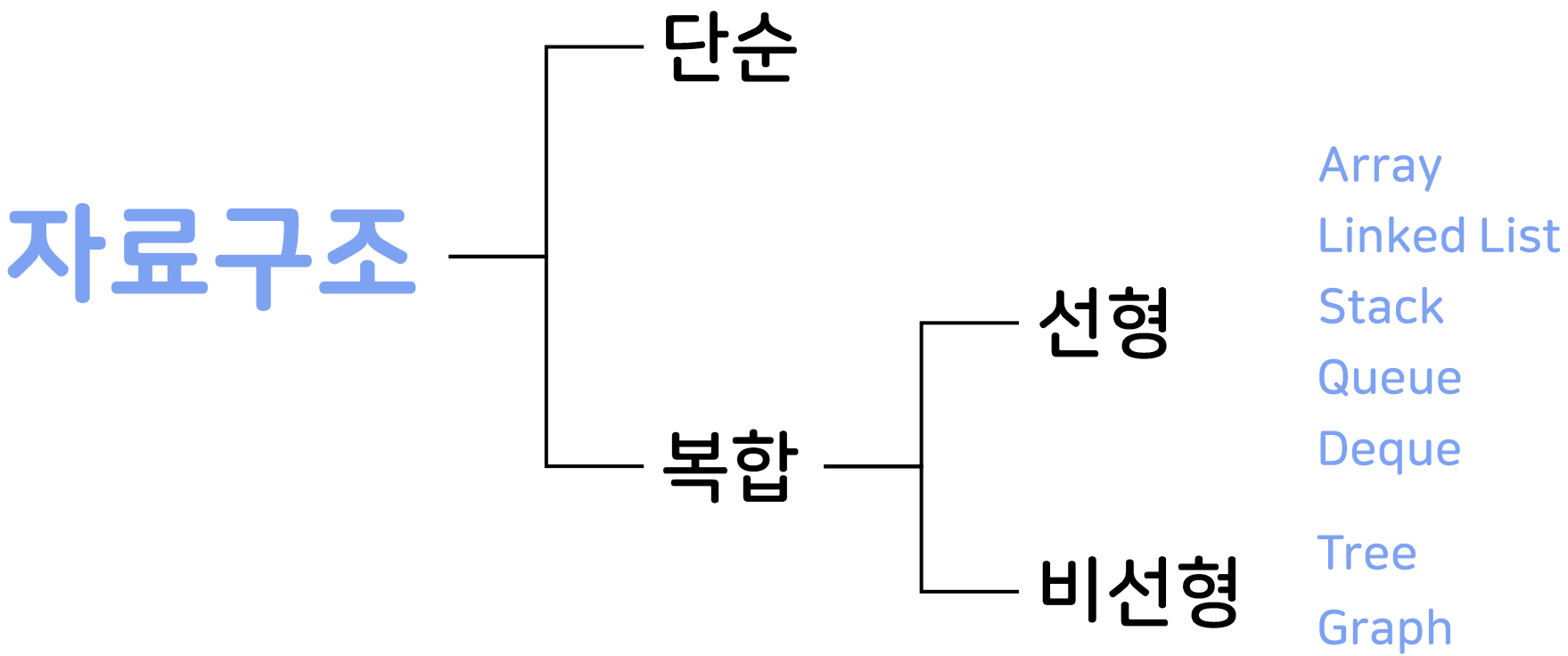
Data Structure (자료구조)

사전 발표 준비 및 전체적인 이론 학습

자료구조 란 무엇이고, 왜 필요할까?


자료구조
단순
복합
선형
비선형
Array
Linked List
Stack
Queue
Deque
Tree
Graph

Array (배열 리스트)

김혜란

- 연관된 data를 **메모리상에 연속적이며 순차적**으로 **미리 할당된 크기**만큼 저장하는 자료구조

- 특징

1. 고정된 저장 공간(fixed-size)
2. 순차적인 데이터 저장(order)

- 장점: lookup과 append가 빨라서 조회를 많이 하는 작업에서 유용
- 단점: fixed-size 특성 상 선언할 때 크기 미리 정함 -> 메모리 낭비나 overhead 발생 가능 -> Dynamic Array

Dynamic Array

저장공간이 가득 차게 되면 **resize**를 하여 유동적으로 size를 조절하여 데이터를 저장

(resize: data를 계속 추가하다가 기존에 할당된 memory를 초과하게 되면, size를 늘린 배열을 선언하고 그곳으로 모든 데이터를 옮김으로써 늘어난 크기의 size를 가진 배열이 되게 하는 것 ex) 두 배 큰 배열로 만드는 Doubling)

- 맨 뒤에 데이터를 추가하거나 삭제하는 것이 상대적으로 빠름 – 시간 복잡도가 $O(1)$

- 맨 끝이 아닌 곳에 data를 추가하거나 삭제할 때 연속적으로 데이터들이 저장되어 있기 때문에 뒤에 있는 data 들을 모두 한 칸씩 shift 해야 되기 때문에 느림 – 시간 복잡도가 $O(n)$

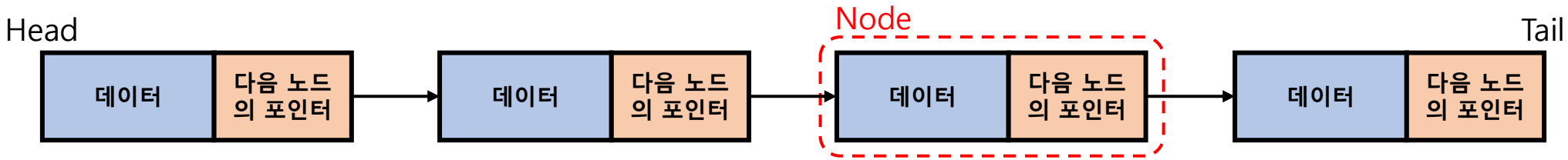
Linked List (연결 리스트)

김혜란

- **노드**라는 구조체 안에 **데이터**와 **다음 노드의 address**를 가지고 있는 자료구조
- 물리적인 메모리상에는 비연속적이나 다음 노드를 가리키는 address가 있기때문에 논리적인 연속성을 가지고 있다고 할 수 있음

- 장점
- 데이터가 추가되는 시점에 메모리가 할당되기 때문에 메모리를 효율적으로 사용 가능
- 어느 원소를 추가, 삭제 하더라도 node에서 다음주소를 가리키는 부분만 다른 주소 값으로 변경하면 되기 때문에 shift할 필요 없어 시간 복잡도가 $O(1)$

- 단점
- 순차 접근(Sequential Access)만 가능 -> 특정 index의 데이터를 조회하기 위해 $O(n)$의 시간 걸림
- 다음 노드의 address를 가지고 있어야 하기 때문에 데이터 하나당 차지하는 메모리가 커짐



Array vs Linked List

Stack (스택)

김소명

- a.k.a. **LIFO**

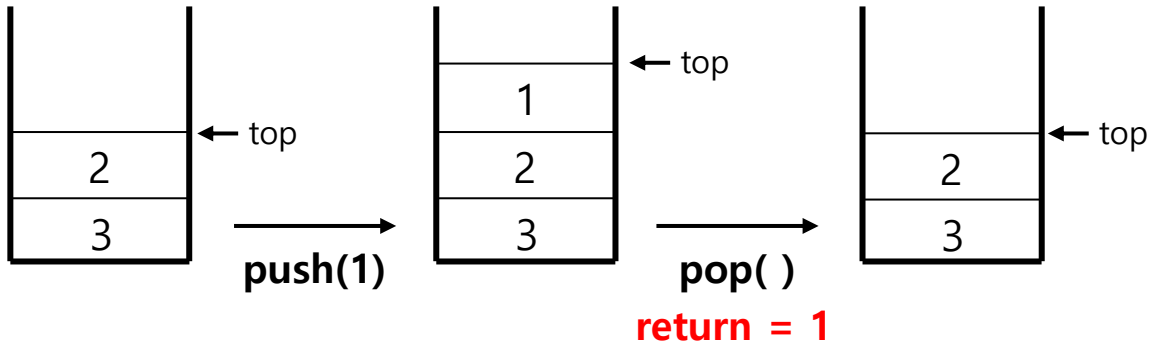
➔ 나중에 추가한 원소를 먼저 반환한다.

- USES

- Undo(실행취소)
- Back (뒤로가기)

- Methods

- Stack에서 데이터의 출입은 항상 top에서 이루어진다.
- Push : top 위에 값을 추가한다.
- Pop : top 값을 반환한다.



Queue (큐)

김소명

- a.k.a. **FIFO**

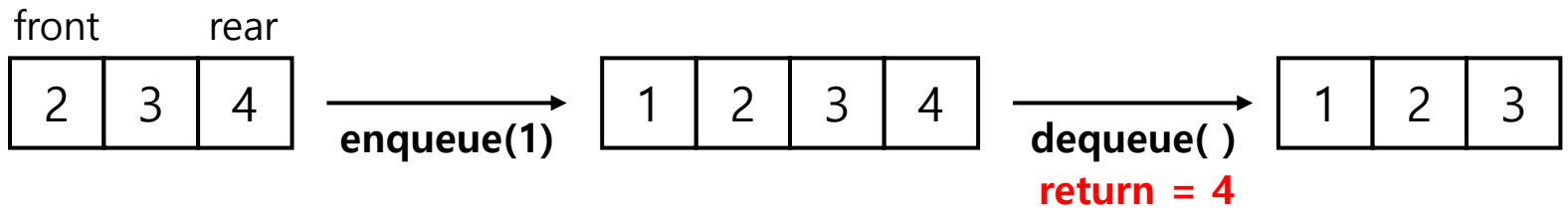
➔ 먼저 추가한 원소를 먼저 반환한다.

- USES

- Buffer
- 선입력된 동작을 먼저 처리할 때 사용하는 자료구조

- Methods

- Queue에서 데이터는 Rear로 들어가고 Front로 나간다.
- Enqueue : Rear 뒤에 값을 추가한다.
- Dequeue : Front 값을 반환한다.

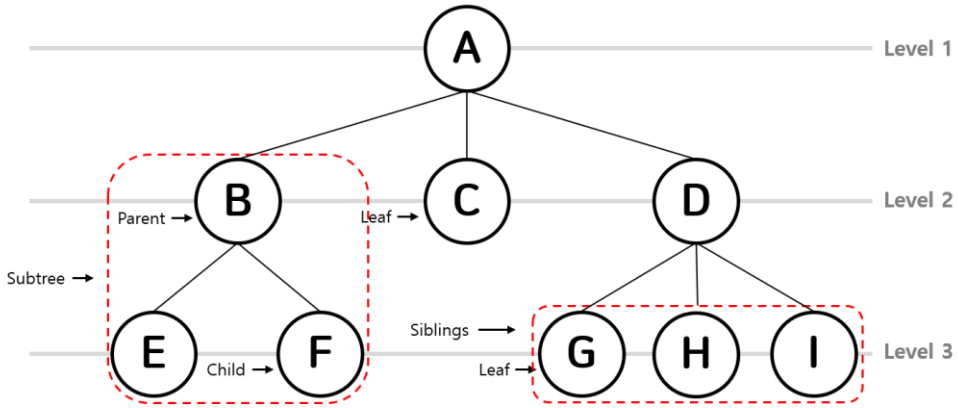


Vertex (정점)
Edge (간선)

단방향 / 양방향

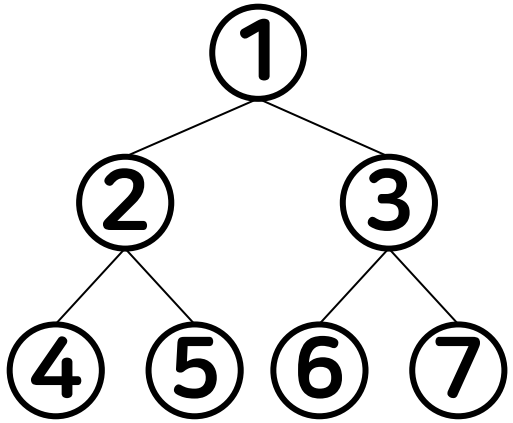
Tree (트리)

문강민



이진 트리

이진 트리는 2개 이하의 자식 노드를 가지는 트리를 일컫는 용어



트리의 순회

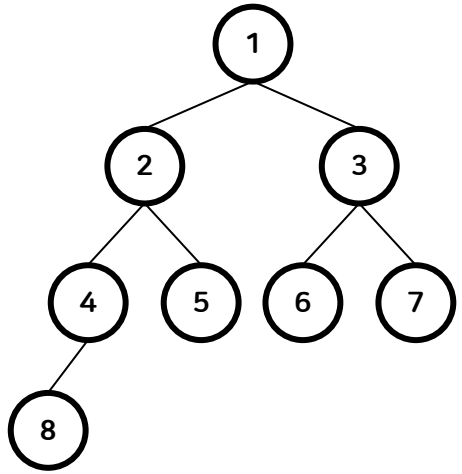
전위 순회: 레벨이 높은 계층에서 부터 왼쪽 하단으로 내려가는 방식

중위 순회: 부모 노드를 기준으로 왼쪽-중앙-오른쪽 순서로 탐색하는 방식

후위 순회: 레벨이 낮은 왼쪽에서 부터 위쪽으로 올라가는 방식

완전 이진 트리

모든 노드가 왼쪽에서 부터 채워지는 이진 트리를 의미한다.

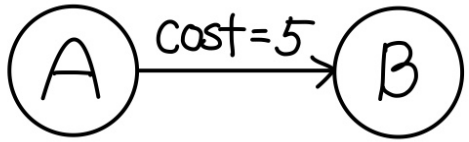


Graph (그래프)

양경식

: **정점끼리의 관계**(=간선)을 나타낸 구조

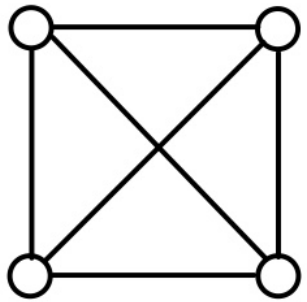
- 무방향 : 방향이 없음
- 단방향 : 방향이 일방향
- 양방향 : 방향이 양쪽

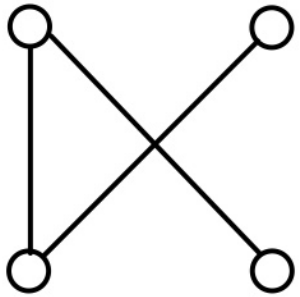


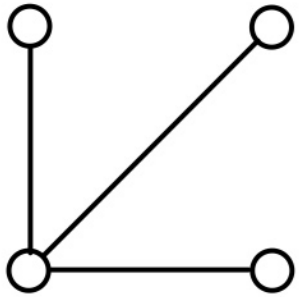
단방향 가중치 그래프

- 가중치 : 거리나 비용 등을 포함

- 완전 : 모든 정점끼리 연결 (E =)
- 신장트리 (Spanning Tree) : 모든 정점이 연결되어있으나, 사이클(순환)이 없음 (E =)







탐색하는 방법

BFS DFS

Heap (힙)

문강민

힙이란 완전 이진 트리의 일종으로 최댓값과 최솟값을 빠르게 찾기 위해 만들어진 자료구조이다.

최대힙: 부모 노드의 값이 자식 노드의 값보다 크거나 같은 완전 이진 트리

최소힙: 부모 노드의 값이 자식 노드의 값보다 작거나 같은 완전 이진 트리

힙의 삽입

힙의 삭제

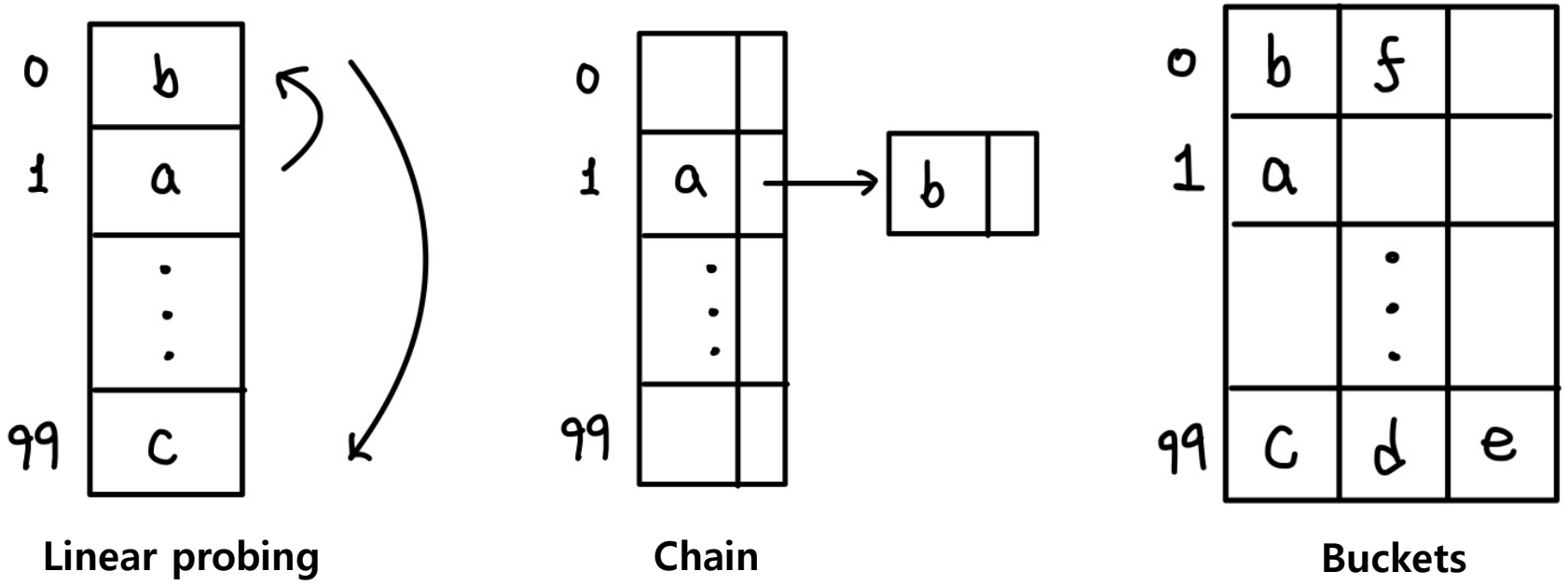
Hash (해쉬)

양경식

임의의 데이터를 정해진 숫자가 나오도록 매핑(mapping)

만약 값이 겹친다면 → **충돌 (Collision)**

Opened (메모리가 무한정)
Closed (메모리가 한정)



Stack
Queue

어떻게 구현할까

시간 복잡도

공간 복잡도

최고 or 평균 or 최악

	접근	탐색	삽입	삭제
Array				
Linked List (Double)				
Stack				
Queue				
Binary Search Tree				

T/F : 지식을 판단하기 위함

T/F : 지식을 판단하기 위함

서술형 : 논의 혹은 생각을 묻기 위함

알고리즘

Divide and Conquer (분할과 정복)

Dynamic Programming (동적 프로그래밍)

Greedy Algorithm (탐욕 알고리즘)

Backtracking (되추적법)

Branch (분기한정법)

Sort (분류)

Search (탐색)