5.3 비선형 자료 구조

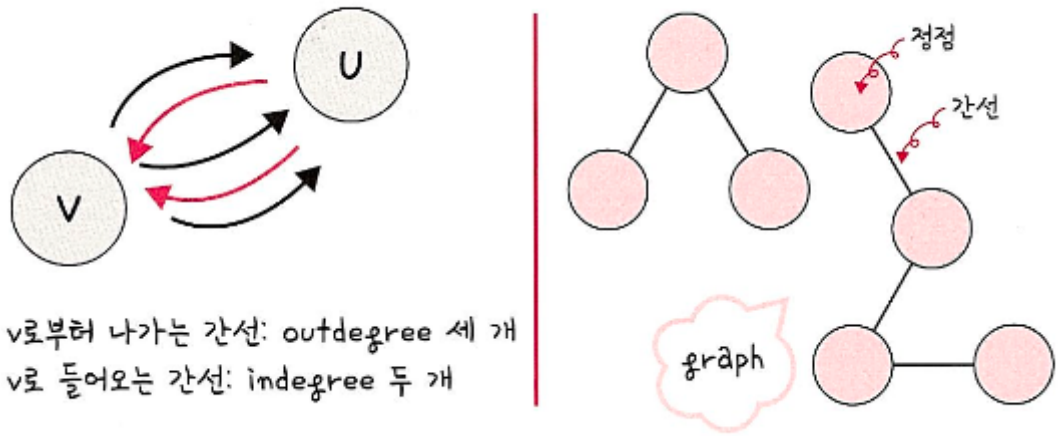
5.3.1 그래프

- 그래프는 정점과 간선으로 이루어진 자료 구조를 말한다.

정점과 간선

- 어떠한 곳에서 어떠한 곳으로 무언가를 통해 간다고 했을 때 어떠한 곳은 정점이고 무언가는 간선이 된다.

▼ 그림 5-12 그래프



- 정점으로 나가는 간선을 해당 점점의 outdegree라고 하며, 들어오는 간선을 해당 정점의 indegree라고 한다.
- 앞 그림의 정점은 약자로 V또는 U라고 하며, 보통 어떤 정점으로부터 시작해서 어떤 정점까지 간다를 U에서 V로 간다. 라고 표현 한다.

가중치

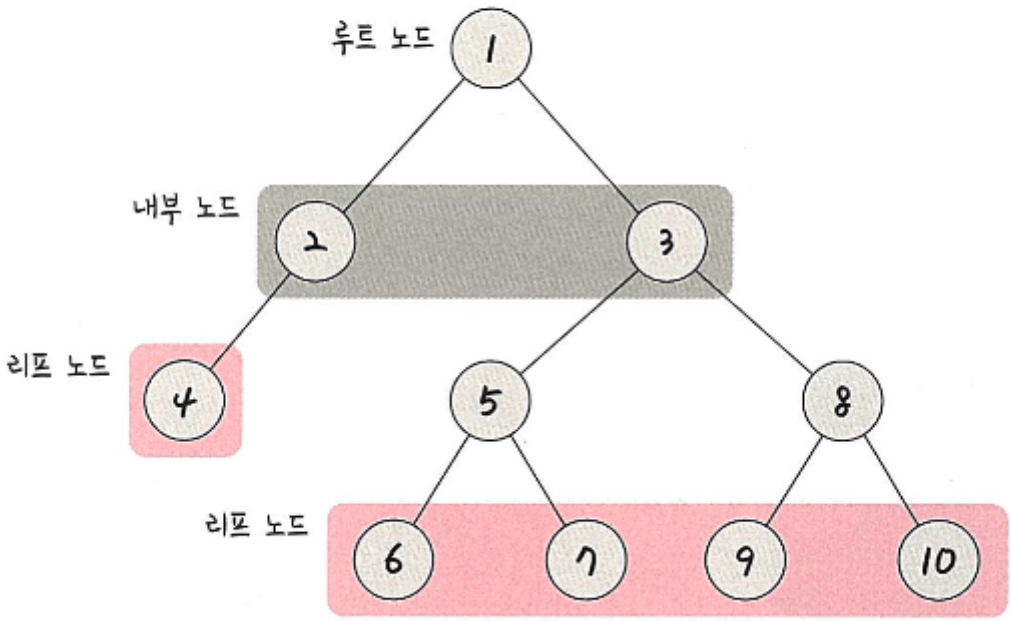
- 가중치는 간선과 정점 사이에 드는 비용을 뜻한다.
- 1번 노드와 2번 노드까지 가는 비용이 한 칸이라면 1번 노드에서 2번 노드까지의 가중치는 한 칸이다.

5.3.2 트리

- 트리는 그래프 중 하나로 그래프의 특징 처럼 정점과 간선으로 이루어져 있고, 트리 구조로 배열된 일종의 계층적 데이터 집합이다. 루트노드, 내부 노드, 리프 노드 등으로 구성된다.

트리의 특징

▼ 그림 5-13 트리의 특징



1. 부모, 자식 계층 구조를 가집니다. 지금 보면 5번 노드는 6번 노드와 7번 노드의 부모 노드이고, 6번 노드와 7번 노드는 5번 노드의 자식 노드입니다. 같은 경로상에서 어떤 노드보다 위에 있으면 부모, 아래에 있으면 자식 노드가 됩니다.
2. V - 1 = E라는 특징이 있습니다. 간선 수는 노드 수 - 1입니다.
3. 임의의 두 노드 사이의 경로는 '유일무이'하게 '존재'합니다. 즉, 트리 내의 어떤 노드와 어떤 노드까지의 경로는 반드시 있습니다.

트리의 구성

- 트리는 루트 노드, 내부 노드, 리프 노드로 이루어져 있다.

루트 노드

- 가장 위에 있는 노드를 뜻한다. 보통 트리 문제가 나오고 트리를 탐색할때 루트 노드를 중심으로 탐색하면 문제가 쉽게 풀린다.

내부 노드

- 루트 노드와 내부 노드 사이에 있는 노드를 뜻한다.

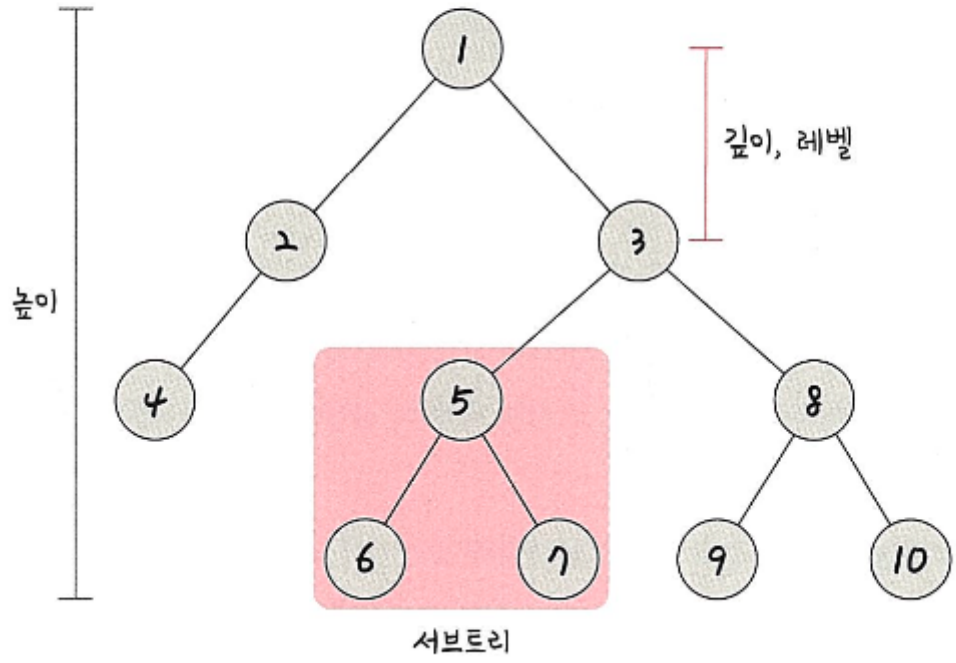
리프 노드

- 리프 노드는 자식 노드가 없는 노드를 뜻한다.

트리의 높이와 레벨

다음은 트리의 높이와 레벨을 설명한 그림입니다.

▼ **그림 5-15** 트리의 높이와 레벨

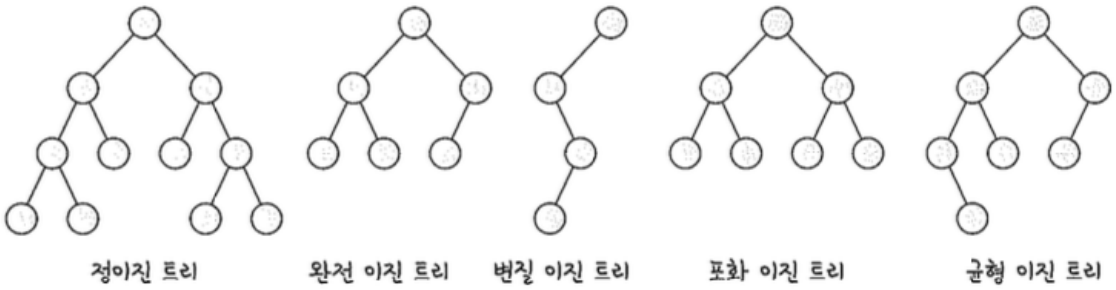


- **깊이**: 트리에서의 깊이는 각 노드마다 다르며, 루트 노드부터 특정 노드까지 최단 거리로 갔을 때의 거리를 말합니다. 예를 들어 4번 노드의 깊이는 2입니다.
- **높이**: 트리의 높이는 루트 노드부터 리프 노드까지 거리 중 가장 긴 거리를 의미하며, 앞 그림의 트리 높이는 3입니다.
- **레벨**: 트리의 레벨은 주어지는 문제마다 조금씩 다르지만 보통 깊이와 같은 의미를 지닙니다. 1번 노드를 0레벨이라고 하고 2번 노드, 3번 노드까지의 레벨을 1레벨이라고 할 수도 있고, 1번 노드를 1레벨이라고 한다면 2번 노드와 3번 노드는 2레벨이 됩니다.
- **서브트리**: 트리 내의 하위 집합을 서브트리라고 합니다. 트리 내에 있는 부분집합이라고도 보면 됩니다. 지금 보면 5번, 6번, 7번 노드가 이 트리의 하위 집합으로 "저 노드들은 서브트리이다."라고 합니다.

이진 트리

- 이진 트리는 자식의 노드 수가 두 개 이하인 트리를 의미하며, 다음과 같이 분류한다.

▼ 그림 5-16 트리의 종류

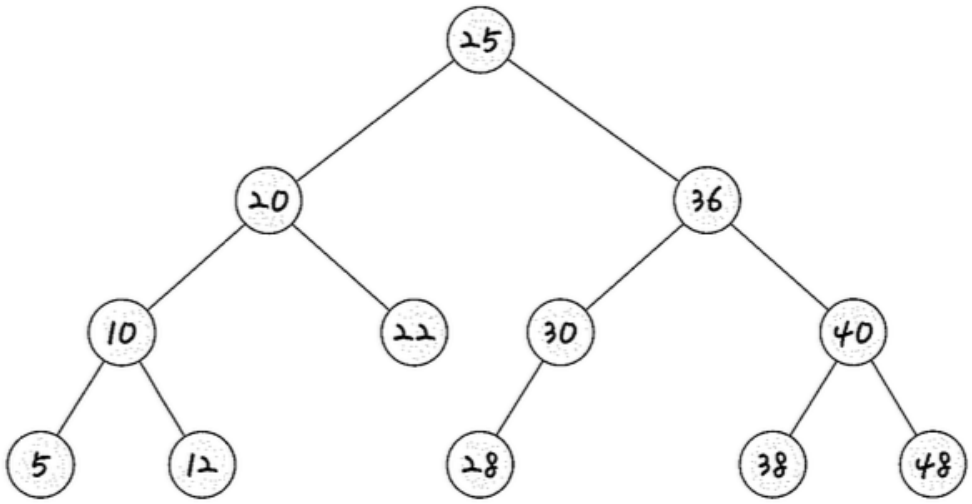


- **정이진 트리(full binary tree)**: 자식 노드가 0 또는 두 개인 이진 트리를 의미합니다.
- **완전 이진 트리(complete binary tree)**: 왼쪽에서부터 채워져 있는 이진 트리를 의미합니다. 마지막 레벨을 제외하고는 모든 레벨이 완전히 채워져 있으며, 마지막 레벨의 경우 왼쪽부터 채워져 있습니다.
- **변질 이진 트리(degenerate binary tree)**: 자식 노드가 하나밖에 없는 이진 트리를 의미합니다.
- **포화 이진 트리(perfect binary tree)**: 모든 노드가 꽉 차 있는 이진 트리를 의미합니다.
- **균형 이진 트리(balanced binary tree)**: 왼쪽과 오른쪽 노드의 높이 차이가 1 이하인 이진 트리를 의미합니다. map, set을 구성하는 레드 블랙 트리는 균형 이진 트리 중 하나입니다.

이진 탐색 트리

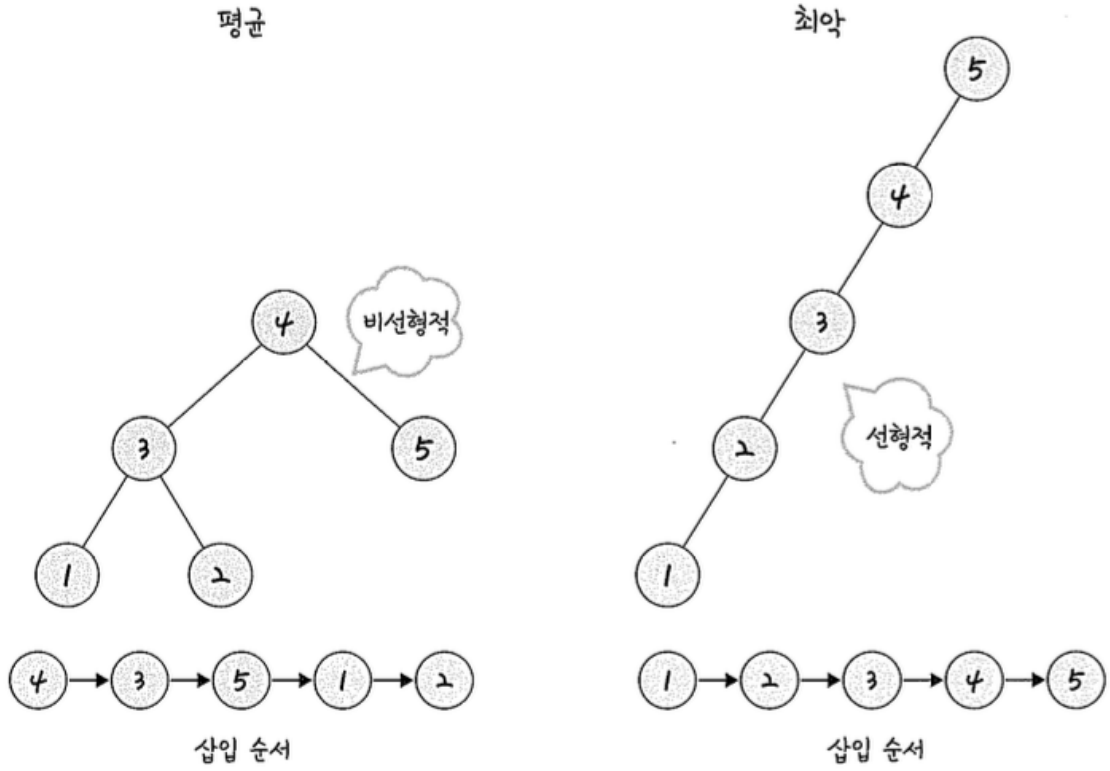
이진 탐색 트리(BST)는 노드의 오른쪽 하위 트리에는 노드 값보다 큰 값이 있는 노드만 포함되고, 왼쪽 하위 트리에는 "노드 값보다 작은 값"이 들어 있는 트리를 말한다.

▼ 그림 5-17 이진 탐색 트리



- 이때 왼쪽 및 오른쪽 하위 트리도 해당 특성을 가진다.
- 검색을 하기에 용이하고 왼쪽에는 작은값, 오른쪽에는 큰 값이 이미 정해져 있기 때문에 O(logn)이 걸린다.
- 하지만 최악의 경우 O(n)이 걸린다.
- 그 이유는 이진 탐색 트리는 삽입 순서에 따라 선형적일 수 있기 때문이다.

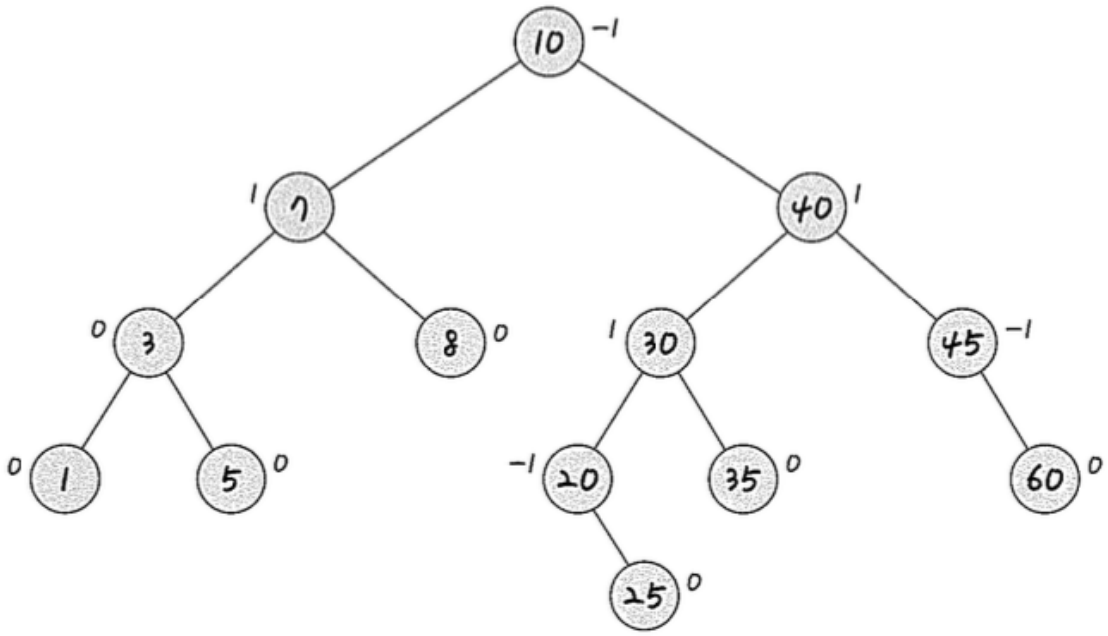
▼ 그림 5-18 이진 탐색 트리의 선형적인 모습



AVL 트리

- AVL 트리는 앞서 설명한 최악의 경우 선형적인 트리가 되는 것을 방지하고 스스로 균형을 잡는 이진 탐색 트리이다. 두 자식 서브트리의 높이는 항상 최대 1만큼 차이 난다는 특징이 있다.

▼ 그림 5-19 AVL 트리

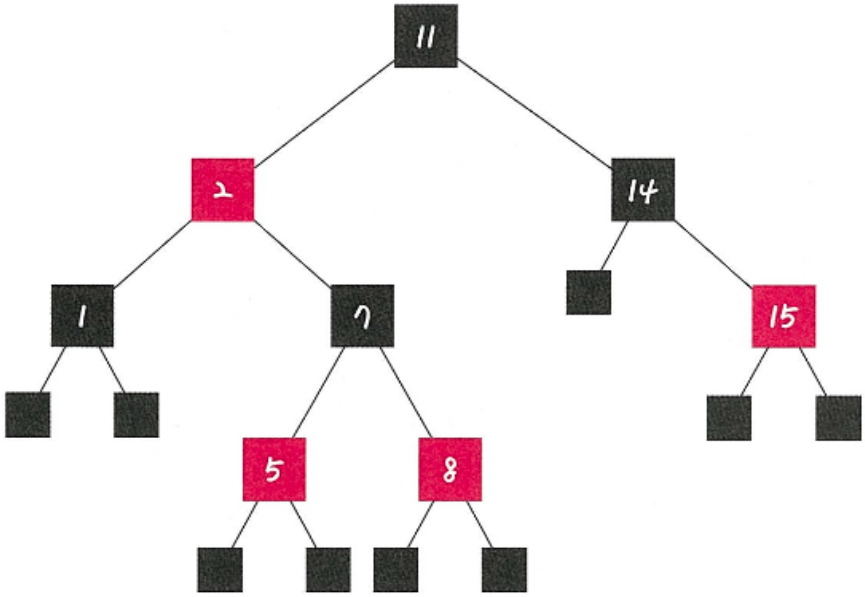


- 이진 탐색 트리는 선형적인 트리 형태를 가질 때 최악의 경우 O(n)의 시간복잡도를 가진다. 이러한 최악의 경우를 배제하고 항상 균형 잡힌 트리로 만들자 라는 개념을 가진 트리이다.
- 탐색, 삽입, 삭제 모두 시간 복잡도가 O(logn)이며 삽입, 삭제를 할 때마다 균형이 안맞는것을 맞추기 위해 트리 일부를 왼쪽 혹은 오른쪽으로 회전시키며 균형을 잡는다.

레드블랙트리

- 레드블랙트리는 균형 이진 트리로 탐색, 삽입, 삭제가 모두 시간복잡도가 O(logn)이다.
- 각 노드는 빨간색 또는 검은색 색상을 나타내는 추가 비트를 저장하며, 삽입 및 삭제 중에 트리가 균형을 유지하도록 하는데 사용 된다.

▼ 그림 5-20 레드 블랙 트리



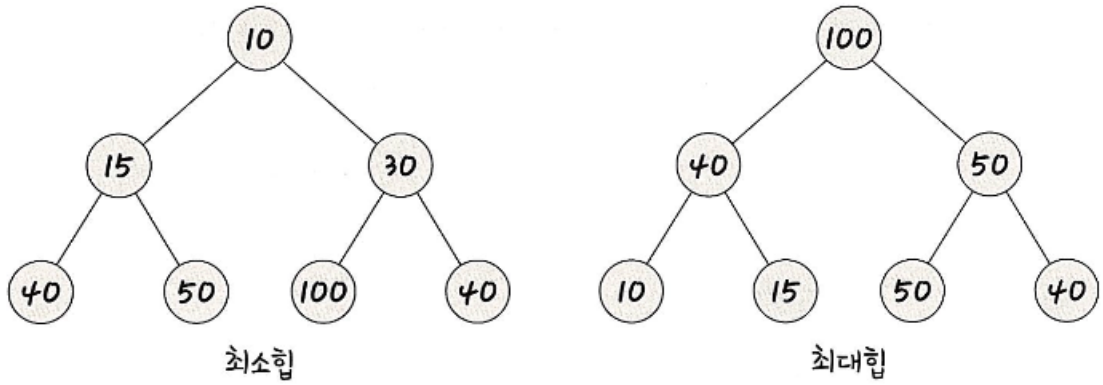
참고로 "모든 리프 노드와 루트 노드는 블랙이고 어떤 노드가 레드이면 그 노드의 자식은 반드시 블랙이다." 등의 규칙을 기반으로 균형을 잡는 트리입니다.

5.3.3 힙

- 힙은 완전 이진 트리 기반의 자료구조이며, 최소힙과 최대힙 두 가지가 있고 해당 힙에 따라 특정한 특징을 지킨 트리를 말한다.

- **최대힙**: 루트 노드에 있는 키는 모든 자식에 있는 키 중에서 가장 커야 합니다. 또한, 각 노드의 자식 노드와의 관계도 이와 같은 특징이 재귀적으로 이루어져야 합니다.
- **최소힙**: 최소힙에서 루트 노드에 있는 키는 모든 자식에 있는 키 중에서 최솟값이어야 합니다. 또한, 각 노드의 자식 노드와의 관계도 이와 같은 특징이 재귀적으로 이루어져야 합니다.

▼ 그림 5-21 최소힙과 최대힙

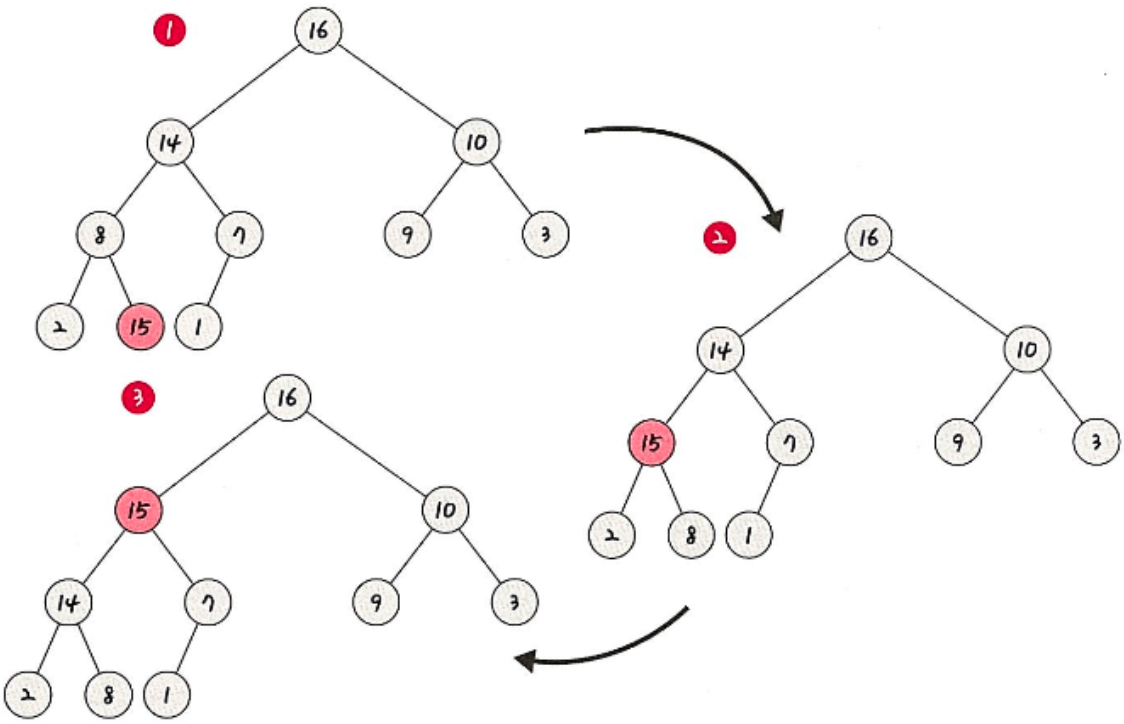


- 힙에는 어떠한 값이 들어와도 특정 힙의 규칙을 지키게 만들어져 있다.

최대힙의 삽입

- 힙에 새로운 요소가 들어오면, 일단 새로운 노드를 힙의 마지막 노드에 이어서 삽입한다.
- 이 새로운 노드를 부모 노드를과의 크기를 비교하며 교환해서 힙의 성징을 만족시킨다.

▼ 그림 5-22 최대힙의 삽입



- 예를 들어 8이라는 값을 가진 노드 밑에 15라는 값을 가진 노드를 삽입한다고 하면, 이 노드가 점차 올라가면서 해당 노드 위에 있는 노드와 스왑하는 과정을 거쳐 최대힙 조건을 만족하게 된다.

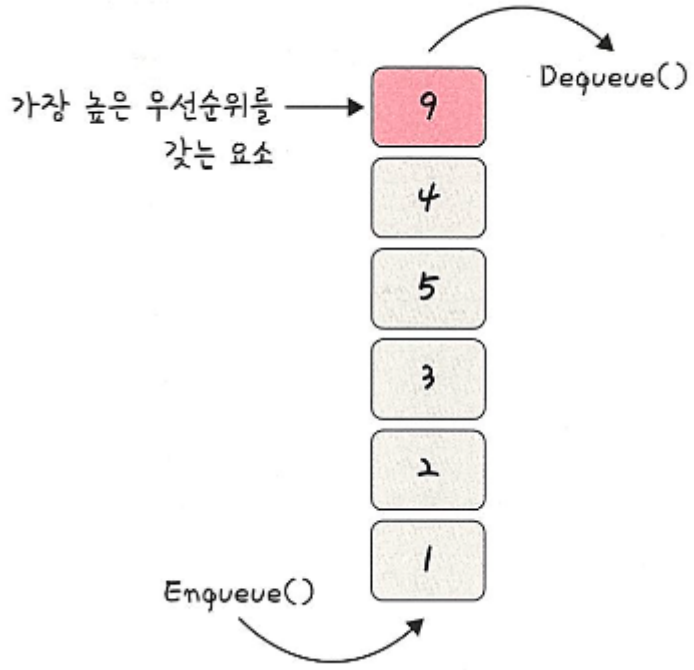
최대힙의 삭제

- 최대힙에서 최대값은 루트 노드이므로 루트 노드가 삭제되고, 그 이후 마지막 노드와 루트 노드를 스왑하여 또 다시 스왑 과정을 거쳐 재구성 된다.

5.3.4 우선순위 큐

- 우선순위 큐는 우선순위 대기열이라고도 하며, 대기열에서 우선순위가 높은 요소가 우선 순위가 낮은 요소보다 먼저 제공되는 자료구조이다.

▼ 그림 5-23 우선순위 큐



- 우선순위 큐는 힙을 기반으로 구현된다.

5.3.5 맵

- 맵은 특정 순서에 따라 키와 매핑된 값의 조합으로 형성된 자료구조이다.
- 예를 들어 “이승철” : 1 같은 방식으로 string : int 형태로 값을 할당해야 할 때 map을 사용한다.
- 레드 블랙 트리 자료구조를 기반으로 형성되고 삽입하면 자동으로 정렬된다.

- 맵은 해시 테이블을 구현할 때 쓰며, 정렬을 보장하지 않는 unordered_map과 정렬을 보잡하는 map 두가지가 있다.

5.3.6 셋

- 셋은 특정 순서에 따라 고유한 요소를 저장하는 컨테이너이며, 중복되는 요소는 없고 오로지 희소한 값만 저장하는 자료구조이다.

5.3.7 해시 테이블

- 해시테이블은 무한에 가까운 데이터들을 유한한 개수의 해시 값으로 매핑한 테이블이다.
- 삽입, 삭제, 탐색 시 평균적으로 O(1) 의 시간 복잡도를 가지며 unordered_map으로 구현한다.