땅울림

08

힙 기반 우선순위 큐 Heap

자료구조 스터디 8회차

- 힙
  - 최소 힙
  - 최대 힙
- 지난 문제 유형들

1

힙 Heap

부모 노드가 자식 노드보다 우선순위가 높다

-> 가장 우선순위 높은 노드가 루트 노드

-> 최대 값이나 최소 값을 항상 빠르게 찾아낼 수 있다.

* 힙은 정렬되지 않아도 된다.

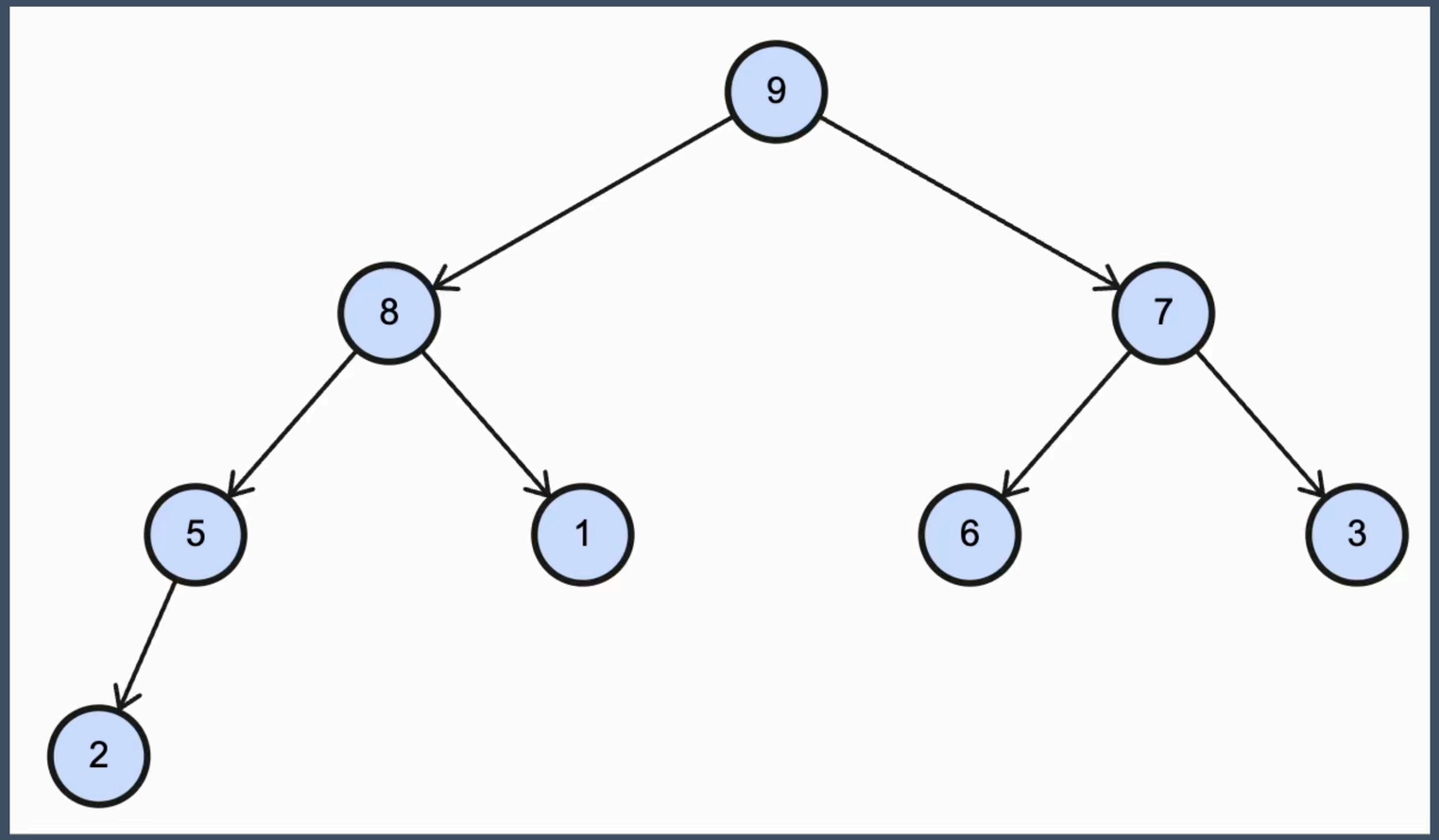
일반적으로 완전 이진 트리의 형태로 구성, 힙은 이진트리의 일종

주로 우선순위 큐를 구현하는데 사용됨

힙의 종류 - 최대 힙, 최소 힙

노드 삽입

: 노드 삽입 후 우선순위에 맞는 위치 찾아서 힙 재구성



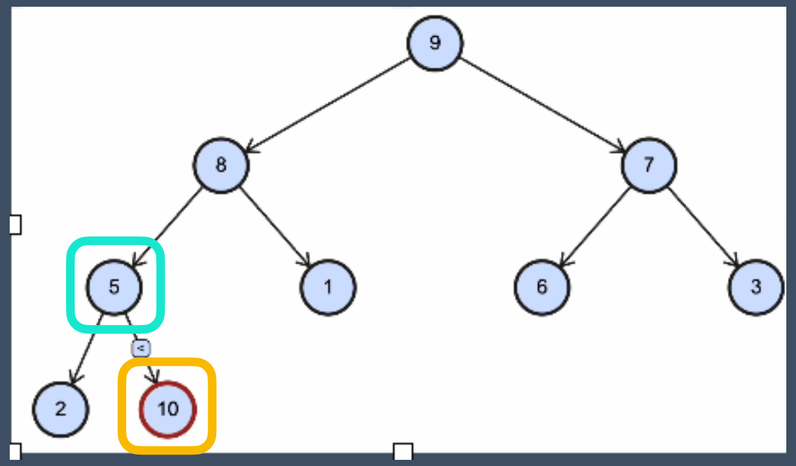
〈영상〉

새로운 노드 '10' 추가
사이즈 증가
올라가면서 힙 재구성

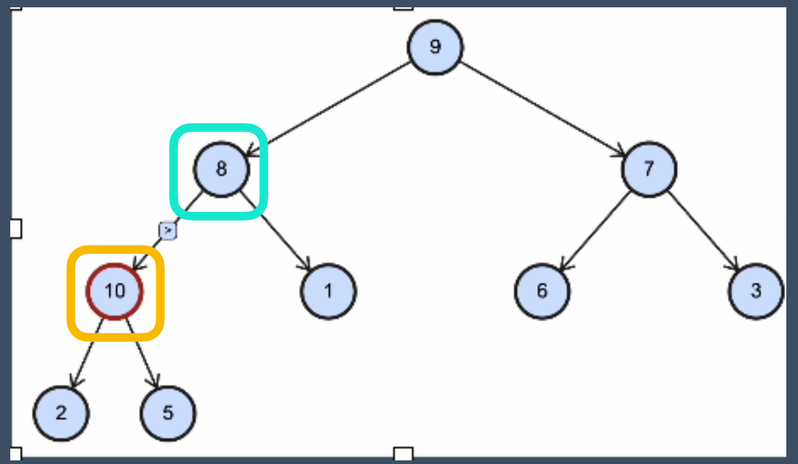
v.push_back(10)
heapSize++
upHeap(10의 인덱스)

upHeap

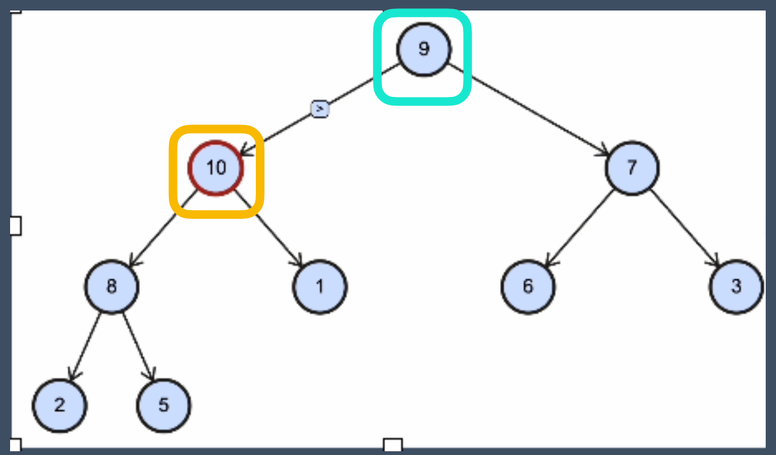
현재 노드에서
올라가면서 힙 재구성



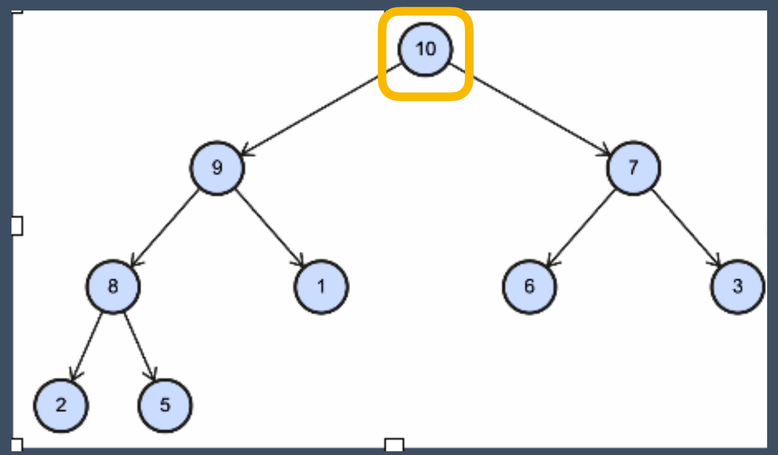
1. 부모 노드(5)보다 우선순위 크므로 부모랑 swap



2. 부모 노드(8)보다 우선순위 크므로 부모랑 swap



3. 부모 노드(9)보다 우선순위 크므로 부모랑 swap



4. 루트까지 올라가거나,
부모 노드보다 우선순위 더 작으면(정상적이라면) 종료

▢ : 현재 노드

▢ : 부모 노드

```
upHeap(int index) {

  If 현재노드가 루트노드면 return
  if (부모 노드 우선순위 > 현재 노드 우선순위)
      swap하기
      upHeap(부모 노드)

}
```

upHeap

현재 노드에서
올라가면서 힙 재구성

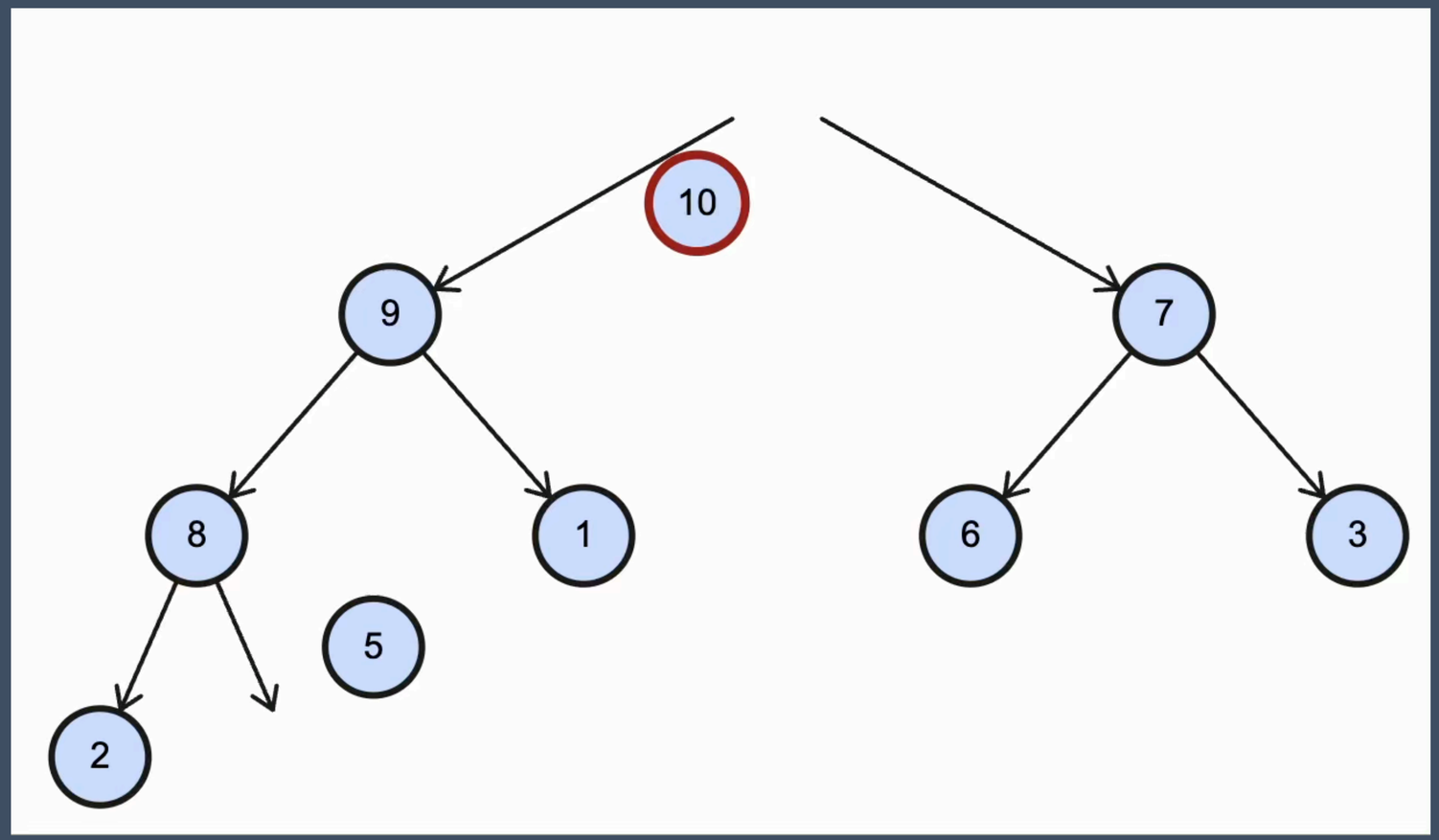
재귀 함수

〈재귀 탈출 조건 1〉
: 현재 노드가
루트 노드면 리턴

〈재귀 탈출 조건 2〉
: 부모 노드보다
우선순위 낮으면
재귀 호출 더이상 안함

노드 삭제

: 힙에서 가장 높은 우선순위를 가지는 요소를 삭제
-> **루트**노드를 삭제



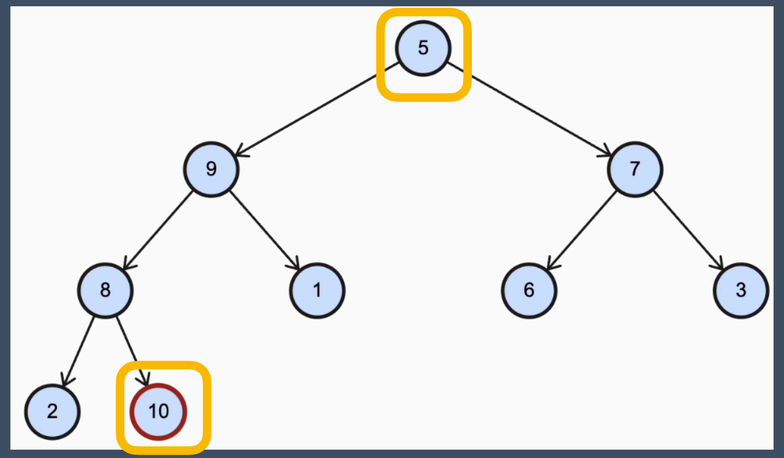
〈영상〉

말단 노드와 루트 노드를 바꾼다.
사이즈 감소
마지막 노드 삭제
위에서부터 내려가면서 힙 재구성

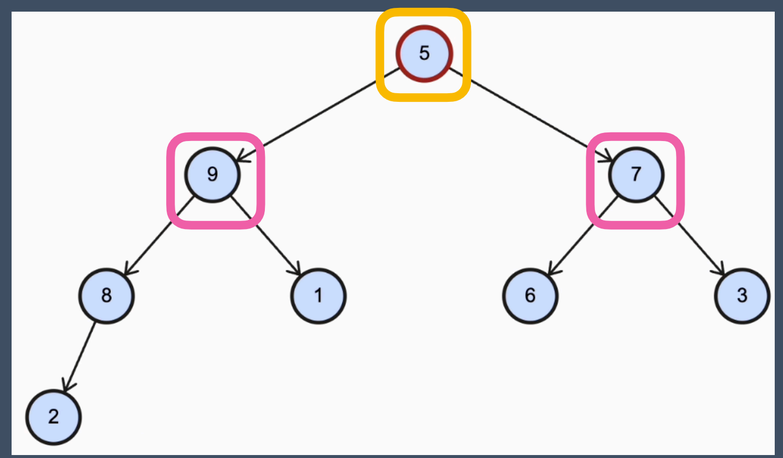
v[루트인덱스] = v[힙사이즈]
heapSize--
v.pop_back()
downHeap(루트인덱스)

downHeap

현재 노드에서
내려가면서 힙 재구성



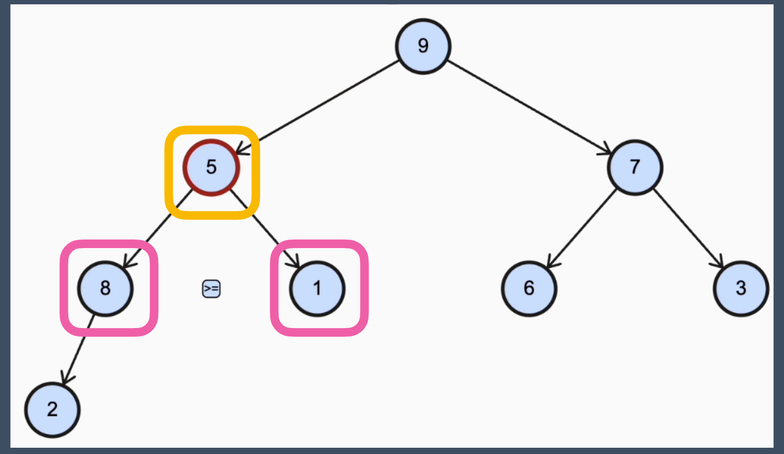
0. 루트노드(10)과 말단노드(5)를
서로 바꾸고 루트 노드(10 삭제



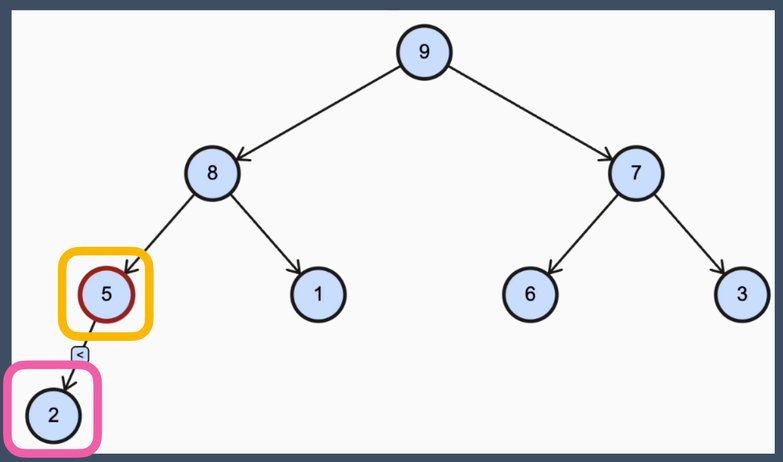
1. 왼쪽 자식(9)과 오른쪽 자식(7)
중에 우선순위 더 높은 9와 swap

: 현재 노드

: 자식 노드



2. 왼쪽 자식(8)과 오른쪽 자식(1)
중에 우선순위 더 높은 8과 swap



4. 더이상 자식 노드가 없거나
자식 노드의 우선순위가 더 낮다면(정상적이라면) 재귀 종료

```
downHeap(int index)
    int left = index*2, right = index*2+1
    int idxOfBestPriority = index    // 초기화

    if (left 우선순위 > 현재 idxOfBestPriority 우선순위
    &&  left <= heapSize)
        idxOfBestPriority = left    // 변수값 업데이트

    if (right 우선순위 > 현재 idxOfBestPriority 우선순위
    &&  right <= heapSize)
        idxOfBestPriority = right    // 변수값 업데이트

    If (idxOfBestPriority != index)
        swap하기
        downHeap(idxOfBestPriority)
```

downHeap

현재 노드에서
내려가면서 힙 재구성

재귀 함수

〈재귀 탈출 조건 1〉
: 자식 노드가 없으면
재귀 호출 안함

〈재귀 탈출 조건 2〉
: 자식 노드보다 우선순위
내가 더 높으면 재귀 호출
더이상 안함

가장 우선순위 높은 값을 가진 노드의
인덱스를 나타내는 변수 선언 후 초기화

if를 통해 왼쪽 자식 노드와 오른쪽 자식 노드 중
더 우선순위 높은 값을 가진 노드를 선택

선택된 노드와 현재 노드의 값을 비교하여
더 우선순위 높은 값을 가진 노드가 새롭게 선택된 경우,

1) 두 노드의 값을 교환

2) downheap호출
(선택된 노드(자식노드)에서부터 다시 downHeap 과정을 시작)

downHeap

현재 노드에서
내려가면서 힙 재구성

재귀 함수

〈재귀 탈출 조건 1〉
: 자식 노드가 없으면
재귀 호출 안함

〈재귀 탈출 조건 2〉
: 자식 노드보다 우선순위
내가 더 높으면 재귀 호출
더이상 안함

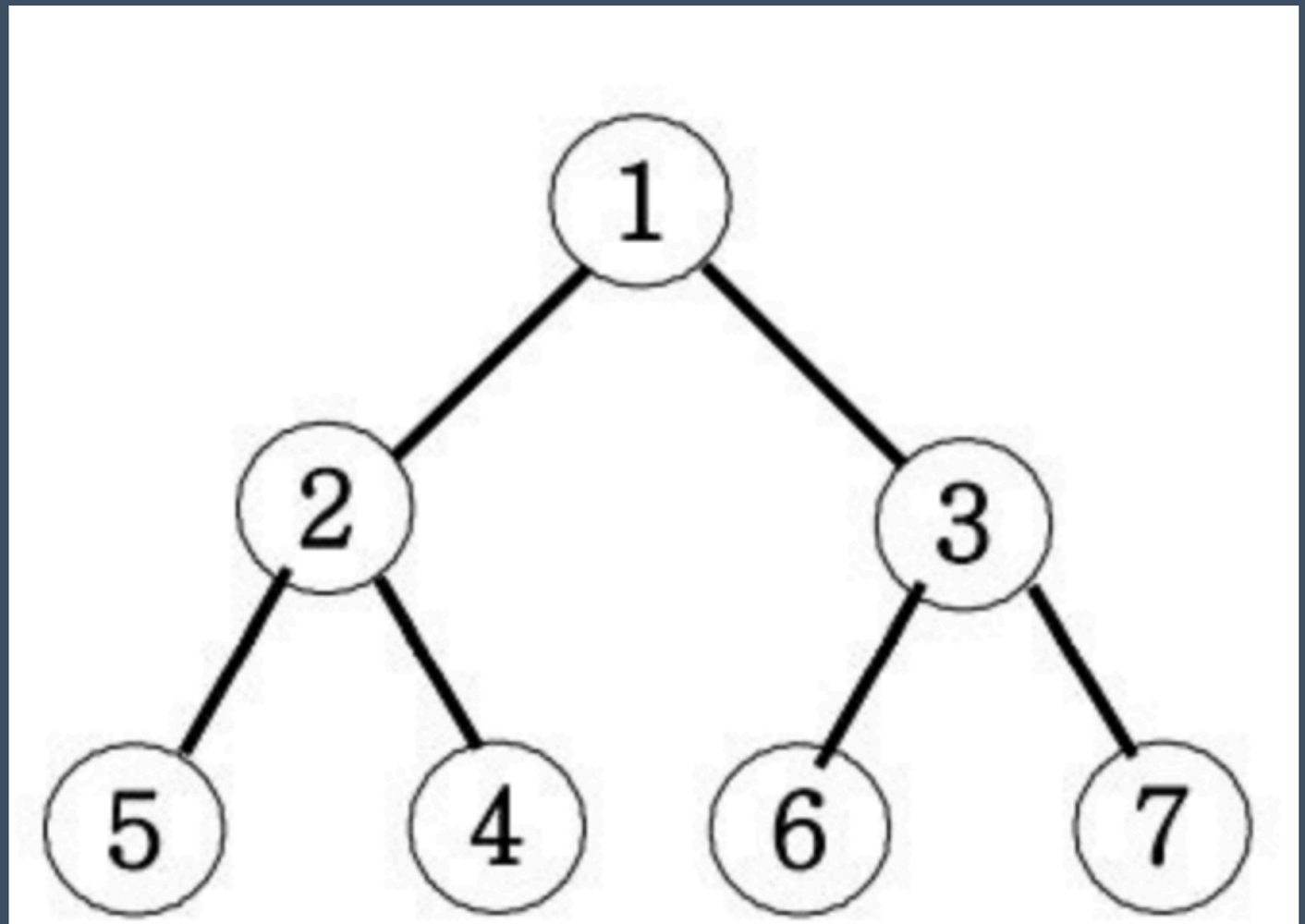
1-1

최소힙

힙

루트 노드가 가장 작은 값
(우선순위 = 작은 수)

부모 노드의 값이 자식 노드의 값보다 작거나 같은 이진트리



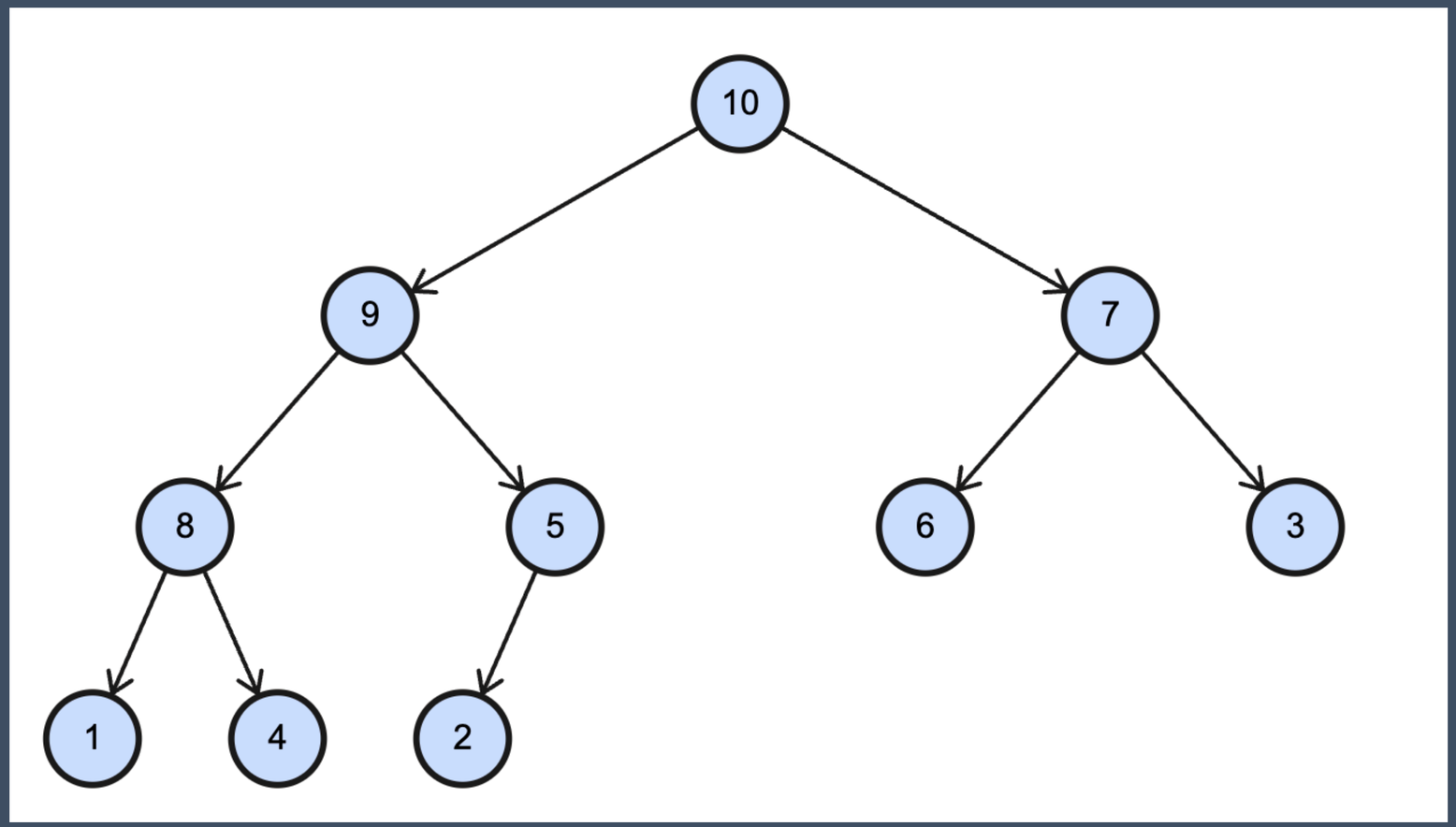
1-2

최대힘

힘

루트 노드가 가장 큰 값
(우선순위 = 큰 수)

부모 노드의 값이 자식 노드의 값보다 크거나 같은 이진트리



2

힙 기반 우선순위 큐 문제 유형

https://github.com/Landvibe-DataStructure-2023Study/Prob/tree/main/21%20%EC%8B%A4%EC%8A%B5%20%EB%AC%B8%EC%A0%9C/Prob_W10

기본 구현 함수	최소힙 기반 우선순위 큐 구현 후 함수 구현 21년도 문제 3번 22년도 문제 1번 22년도 김영호 문제 1번	최대힙 기반 우선순위 큐 구현 후 함수 구현 21년도 문제 1번 22년도 문제 3번 22년도 김영호 문제 3번
약간 응용 문제	힙 기반 우선순위 큐 구현 후 모든 물질을 특정 온도 이하로 만들기 최대힙 기반 22년도 문제 2번 22년도 김영호 문제 2번	힙 기반 우선순위 큐 구현 후 모든 물질을 특정 온도 이상으로 만들기 최소힙 기반 21년도 문제 2, 4번 22년도 문제 4번 22년도 김영호 문제 4번