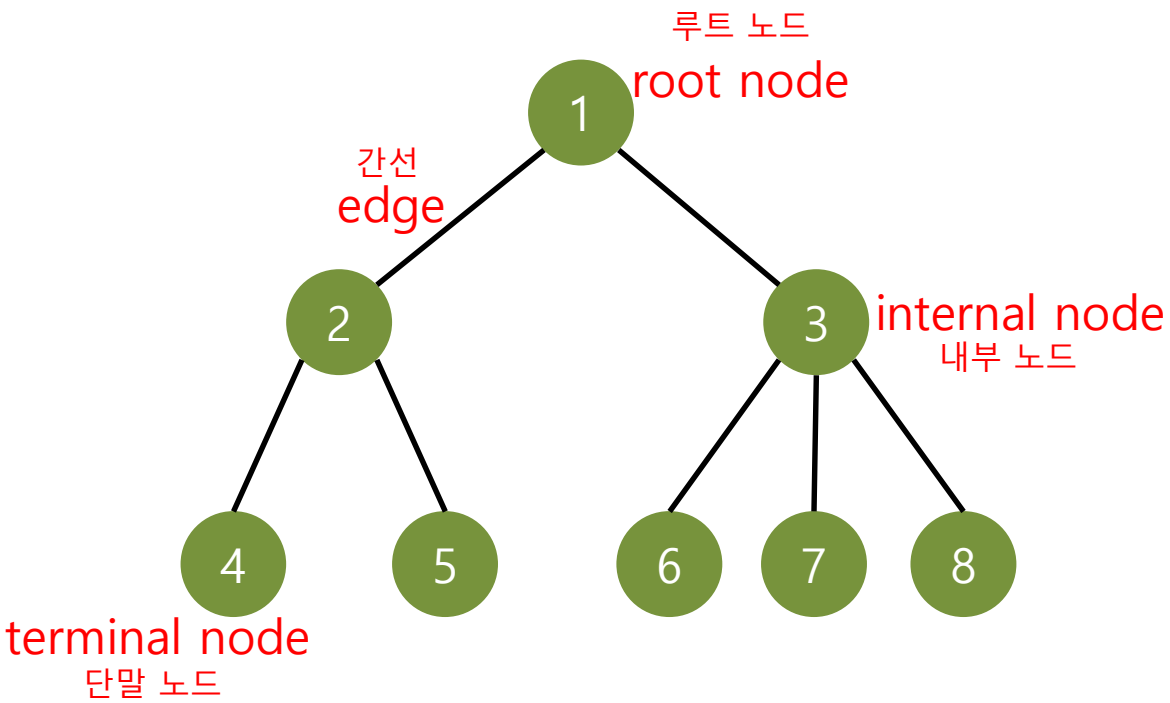
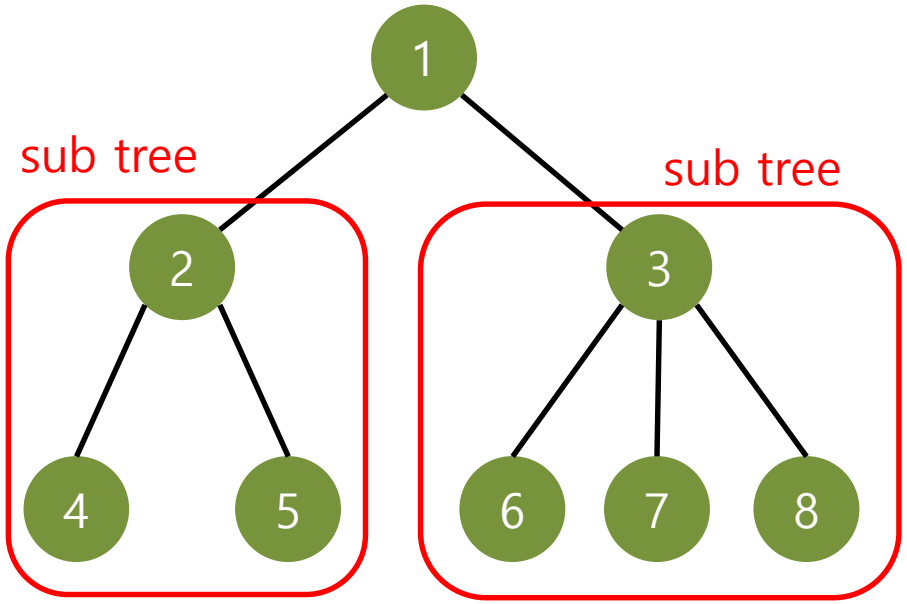
Data structure

Heap, priority queue

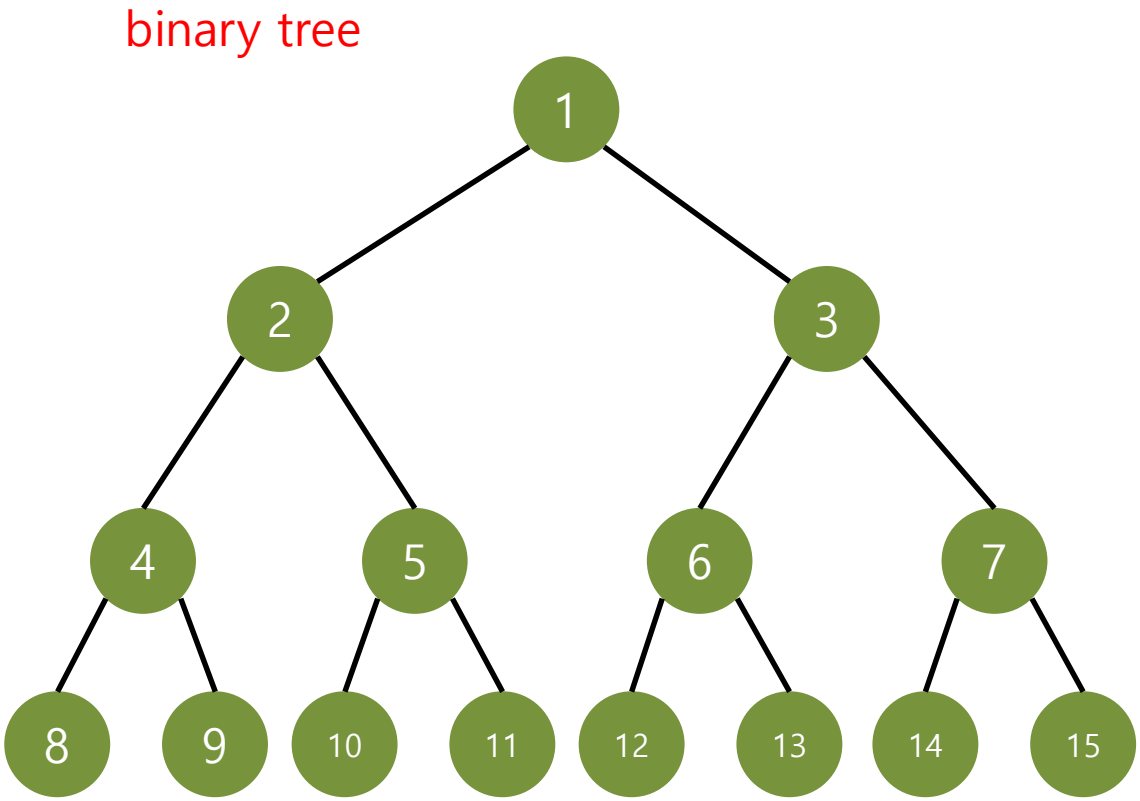
Tree란?


루트 노드
root node
1
간선
edge
2
3
internal node
내부 노드
4
5
6
7
8
terminal node
단말 노드

Tree란?



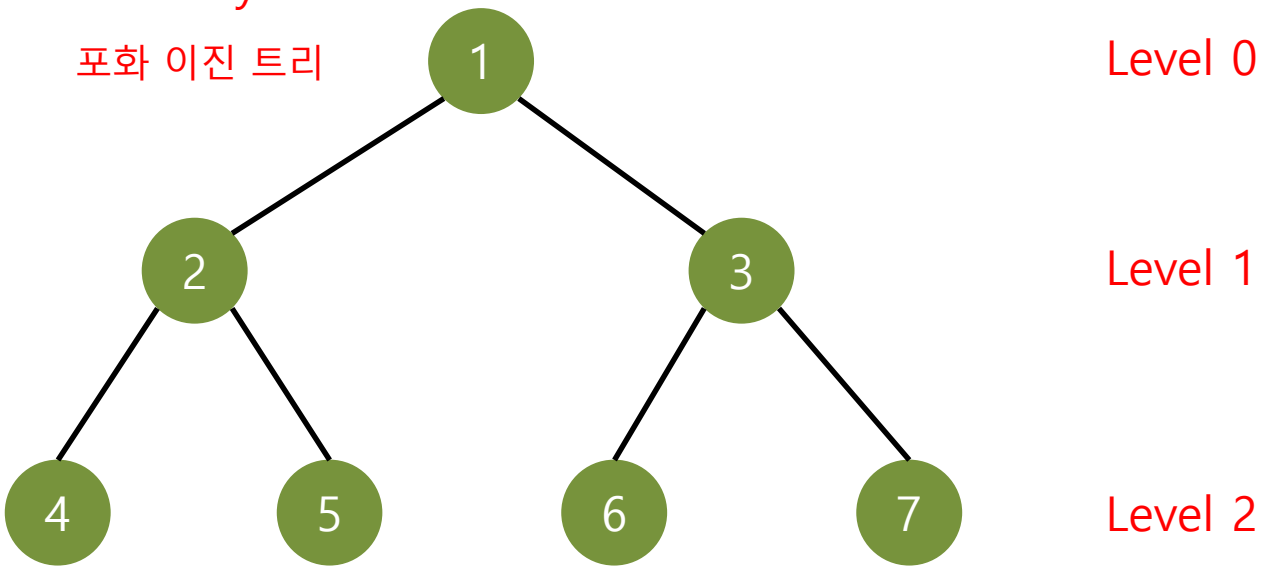
Tree란?



Tree란?

full binary tree

포화 이진 트리

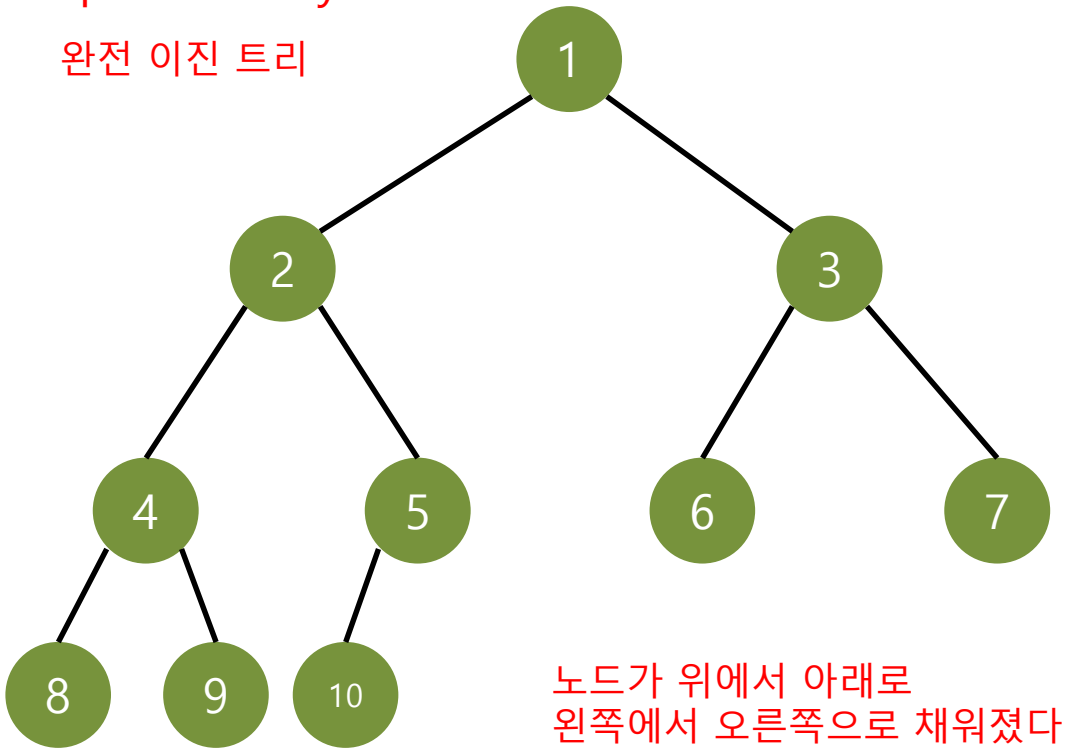


모든 레벨이 다 차 있다

Tree란?

complete binary tree

완전 이진 트리

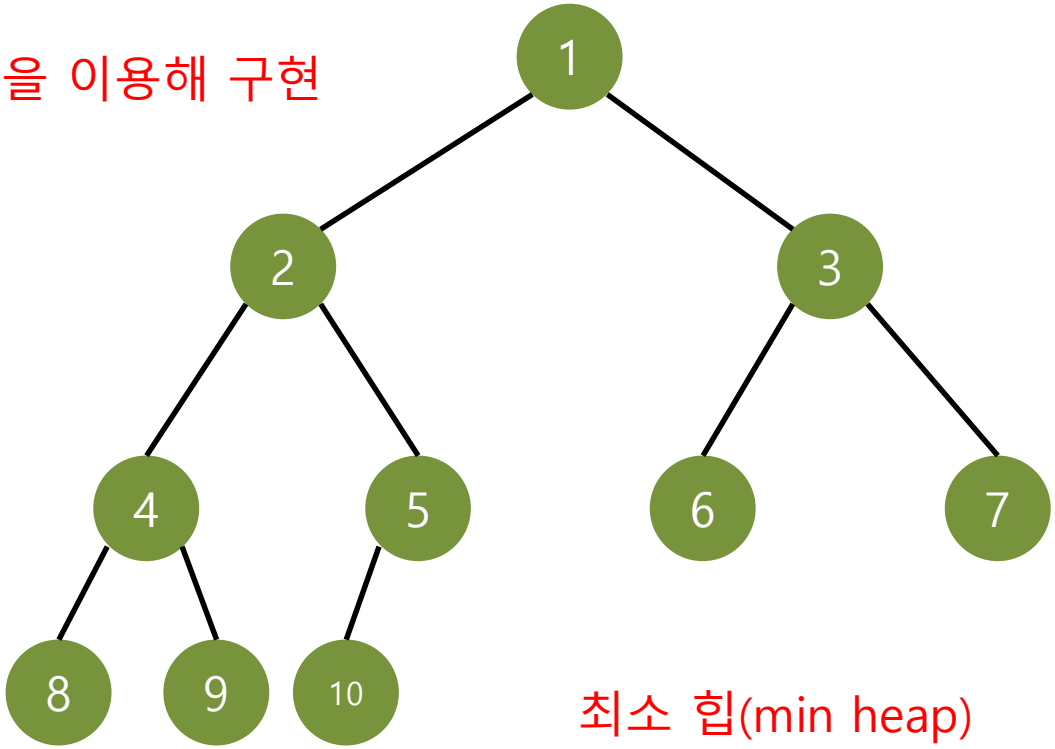


노드가 위에서 아래로
왼쪽에서 오른쪽으로 채워졌다

heap

Heap은 완전 이진 트리

전통적으로 배열을 이용해 구현

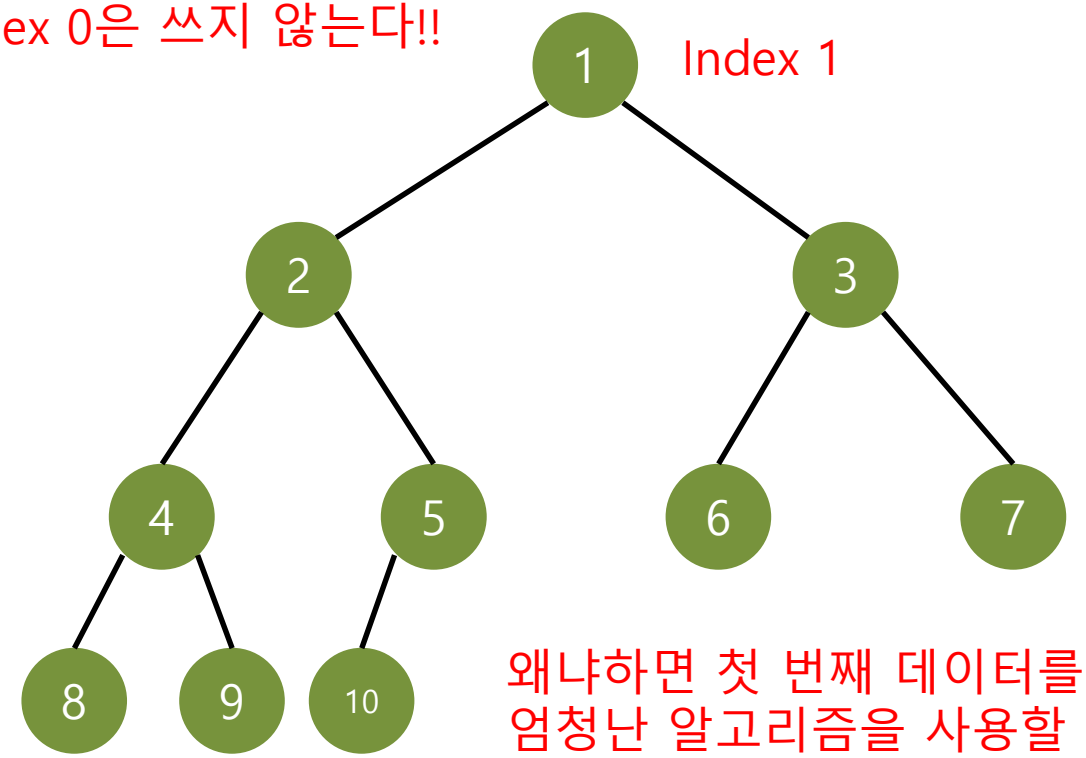


최소 힙(min heap)

heap

Heap을 배열을 이용해 구현하는 방법

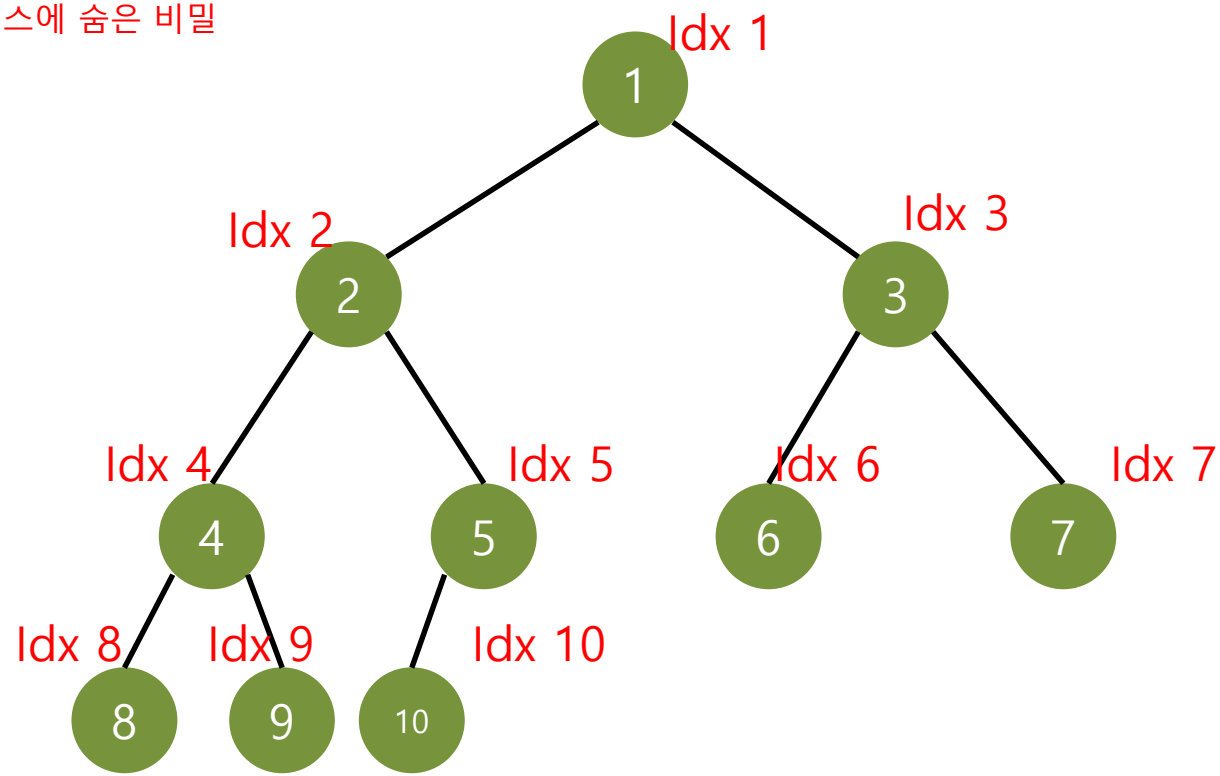
Index 0은 쓰지 않는다!!



왜냐하면 첫 번째 데이터를 index 1에 넣음으로써
엄청난 알고리즘을 사용할 수 있습니다!!

heap

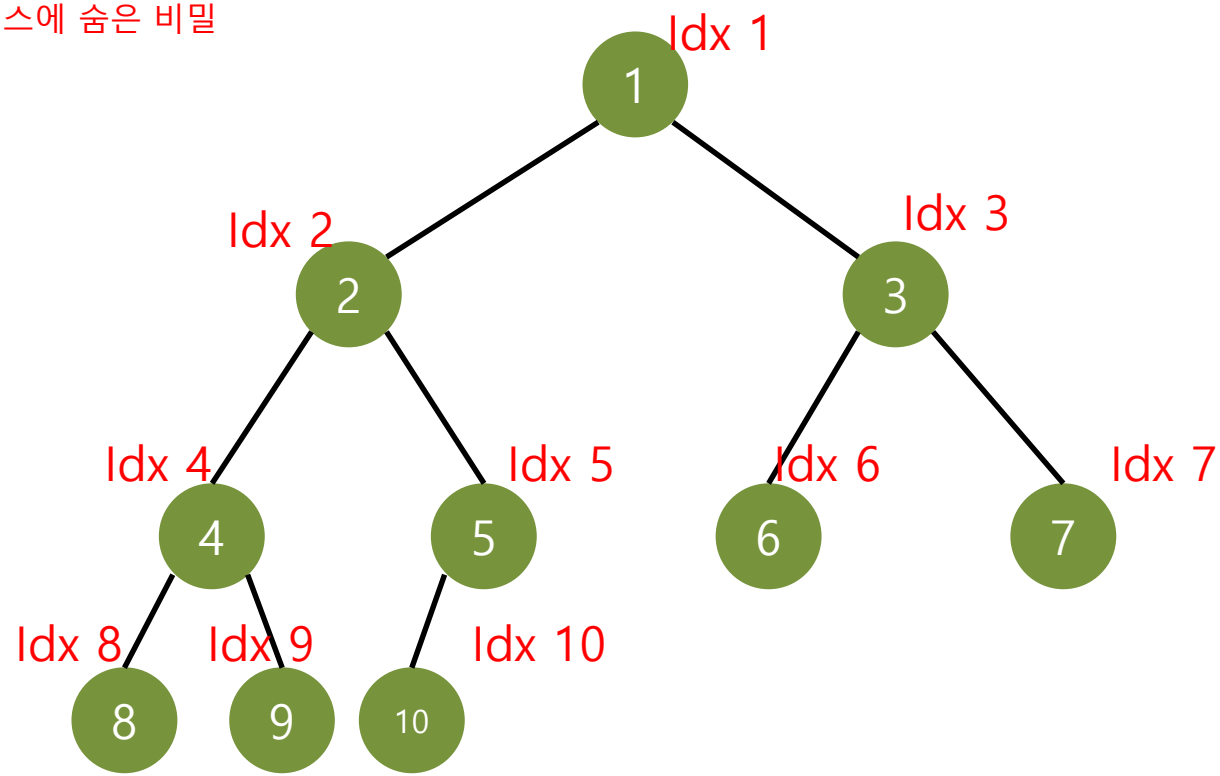
배열 인덱스에 숨은 비밀



Parent index = child index / 2

heap

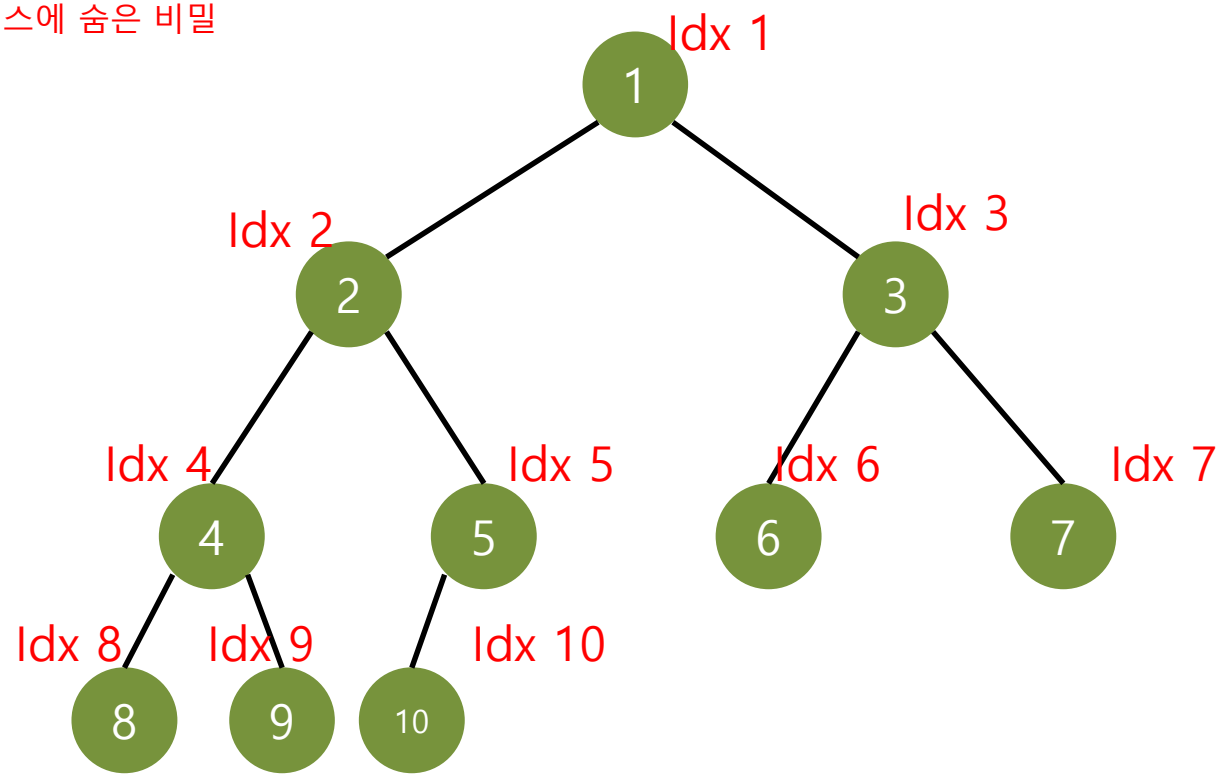
배열 인덱스에 숨은 비밀



Left child index = parent index * 2

heap

배열 인덱스에 숨은 비밀

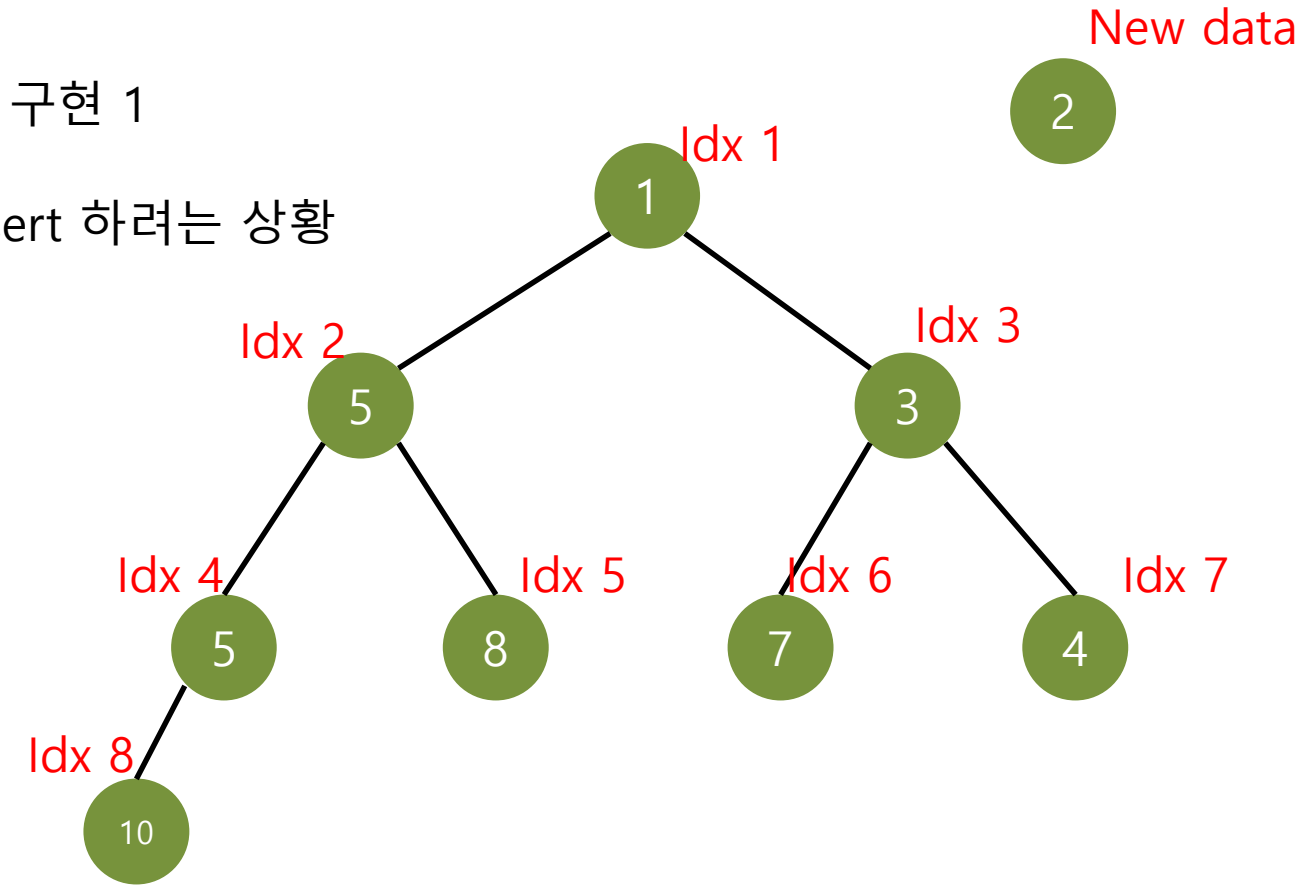


Right child index = parent index * 2 + 1

heap

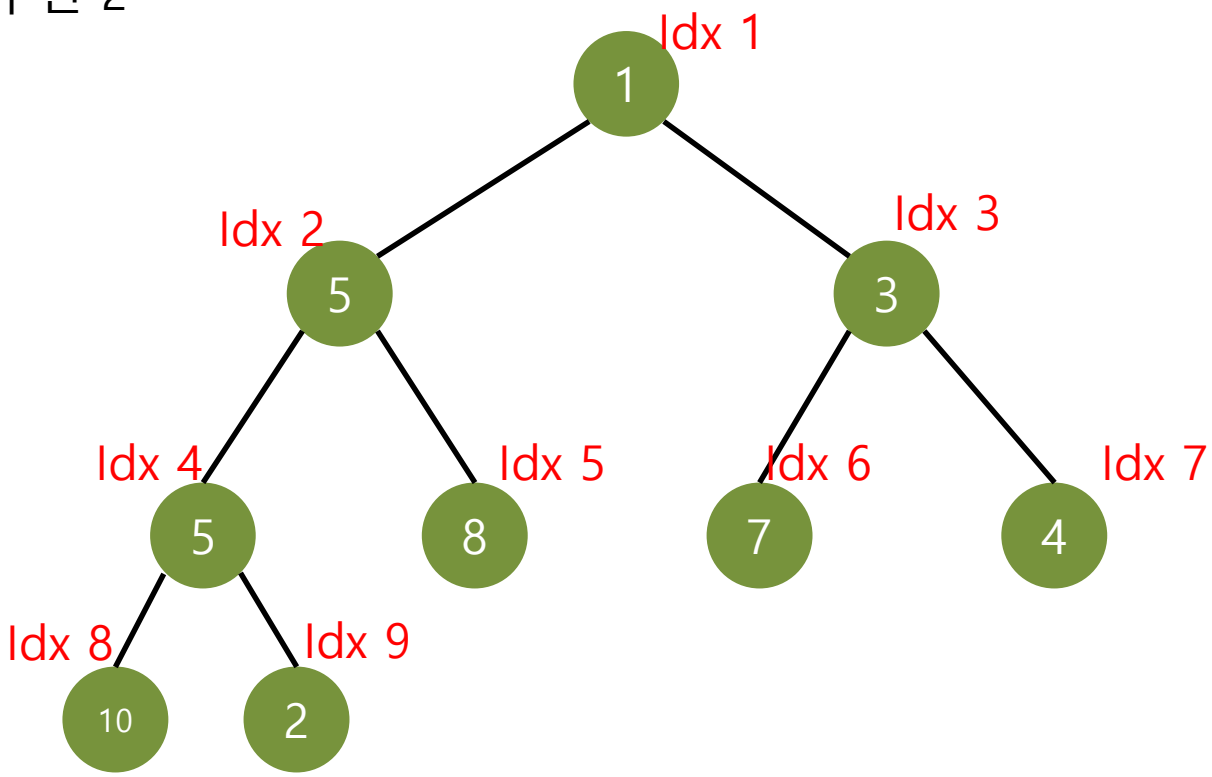
Insert() 구현 1

2를 insert 하려는 상황



heap

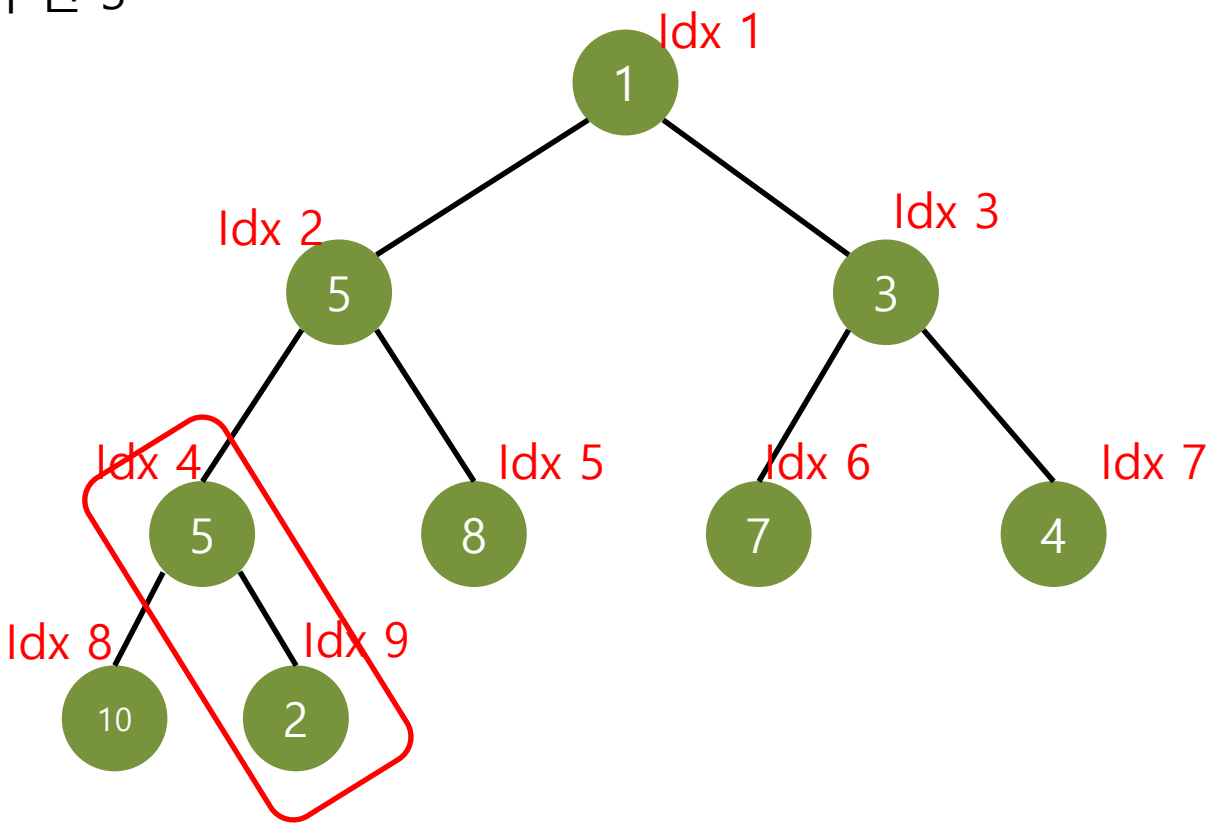
Insert() 구현 2



1. 맨 마지막 노드에 새로운 데이터를 추가

heap

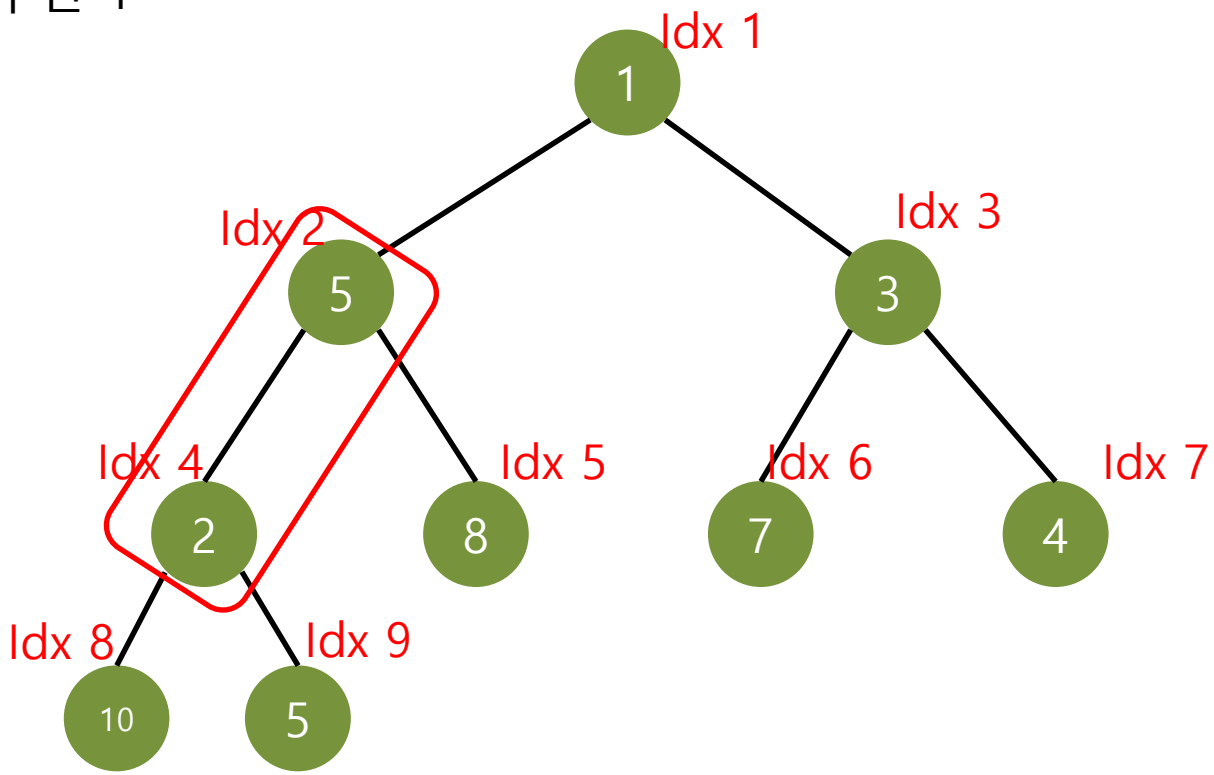
Insert() 구현 3



2. 부모 노드와 비교
최소 힙이므로 2가 5보다 작으면 교환

heap

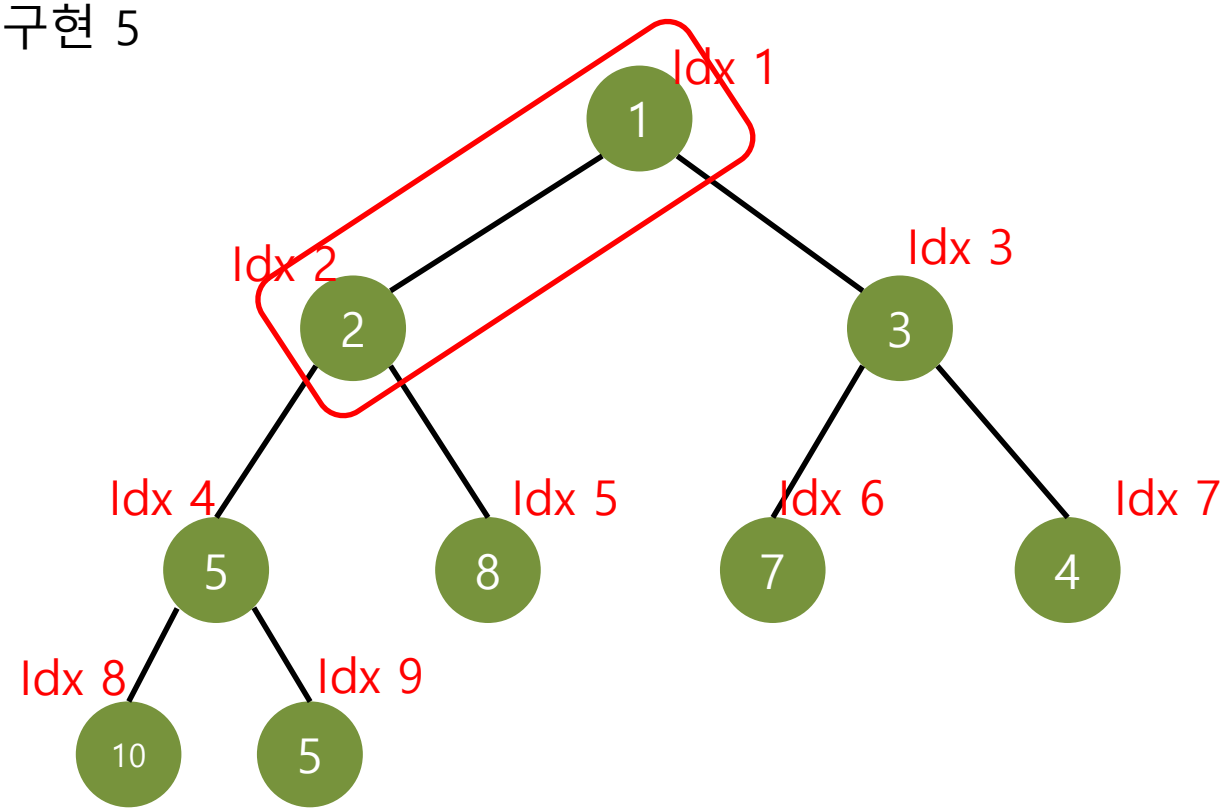
Insert() 구현 4



3. 부모 노드와 비교
최소 힙이므로 2가 5보다 작으면 교환

heap

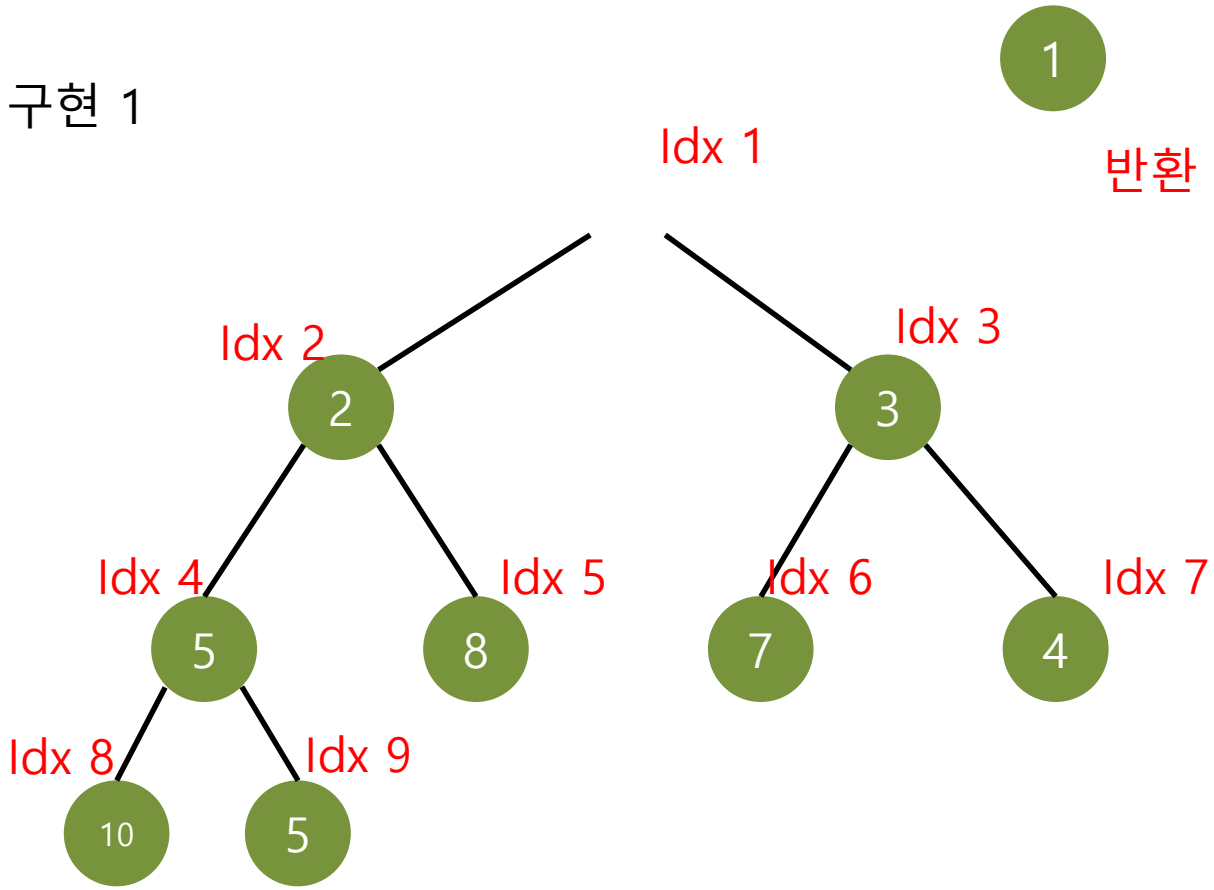
Insert() 구현 5



4. 루트 노드에 도달했거나 부모 노드가 더 작다면 멈춘다

heap

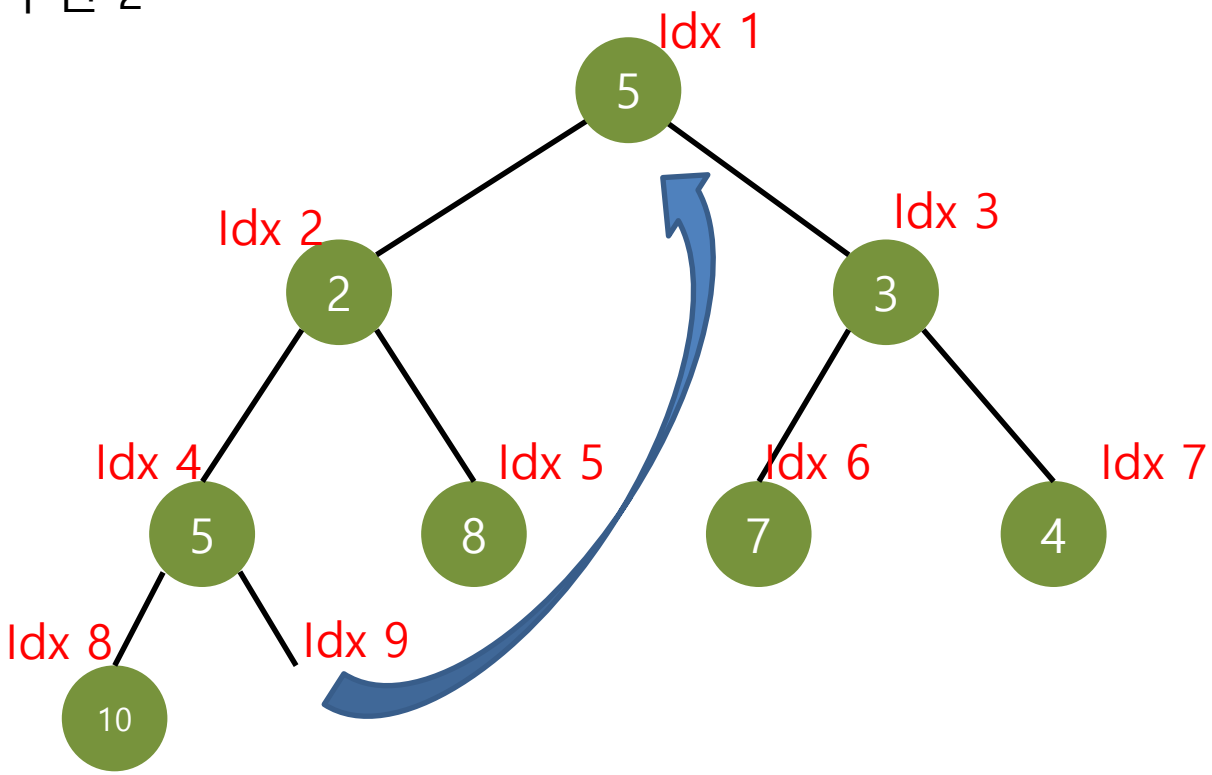
Delete() 구현 1



1. idx 1에 있는 값을 반환 하므로 1 반환

heap

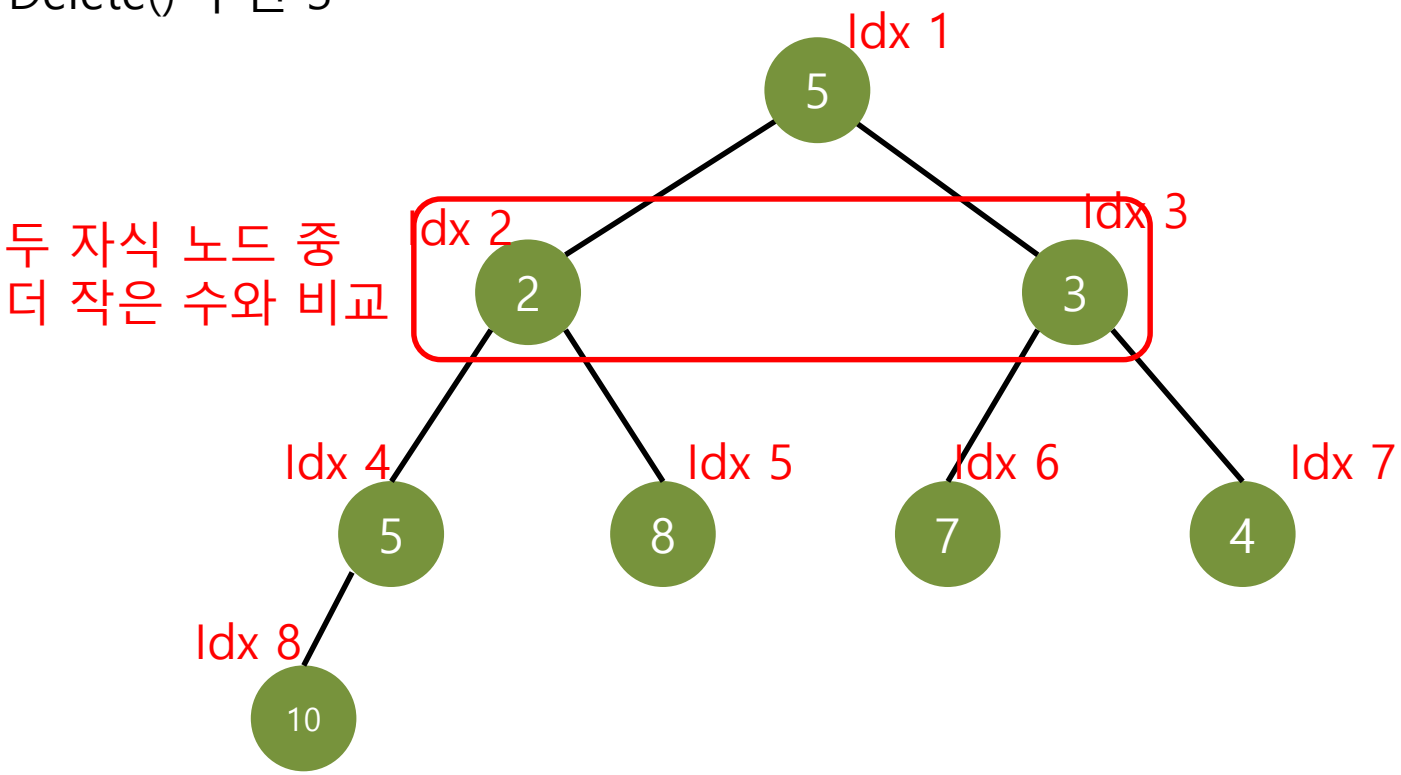
Delete() 구현 2



2. 맨 마지막 노드를 root 노드로

heap

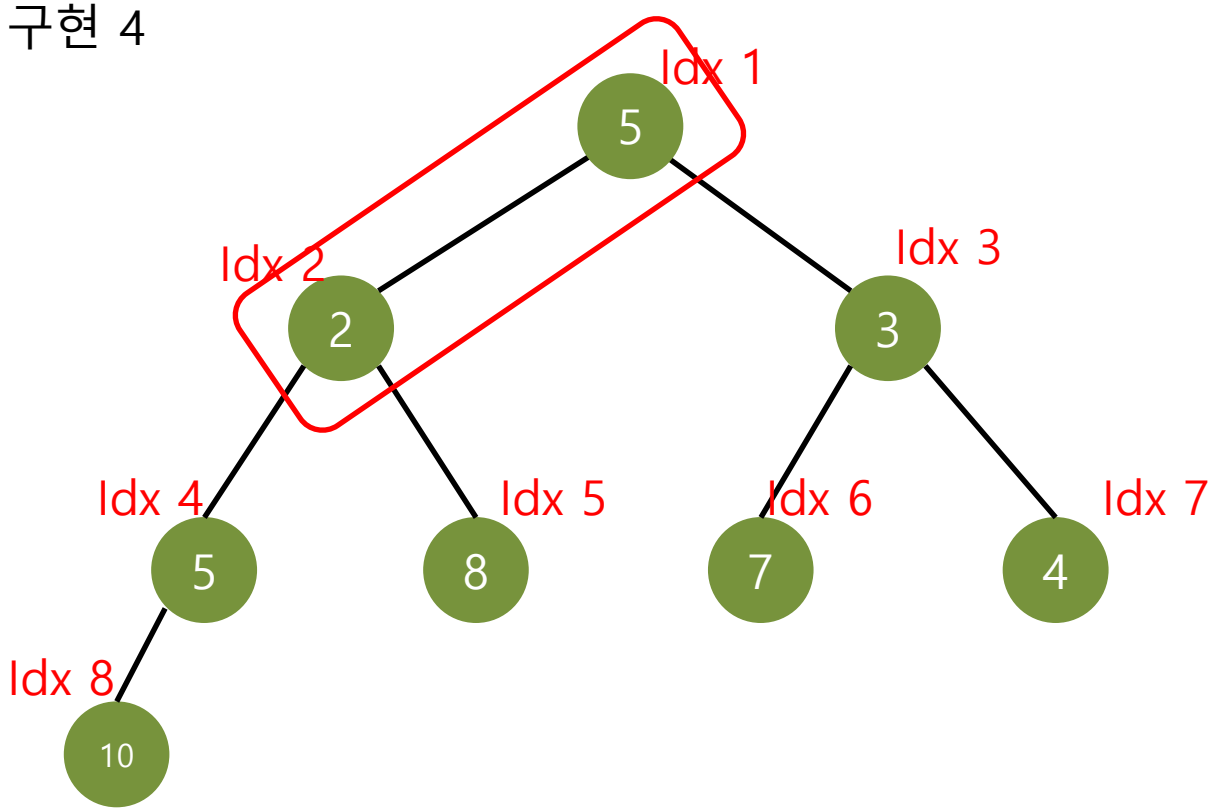
Delete() 구현 3



3. 두 자식 노드를 비교해 우선 순위가 높은 것
(여기서는 작은 수)

heap

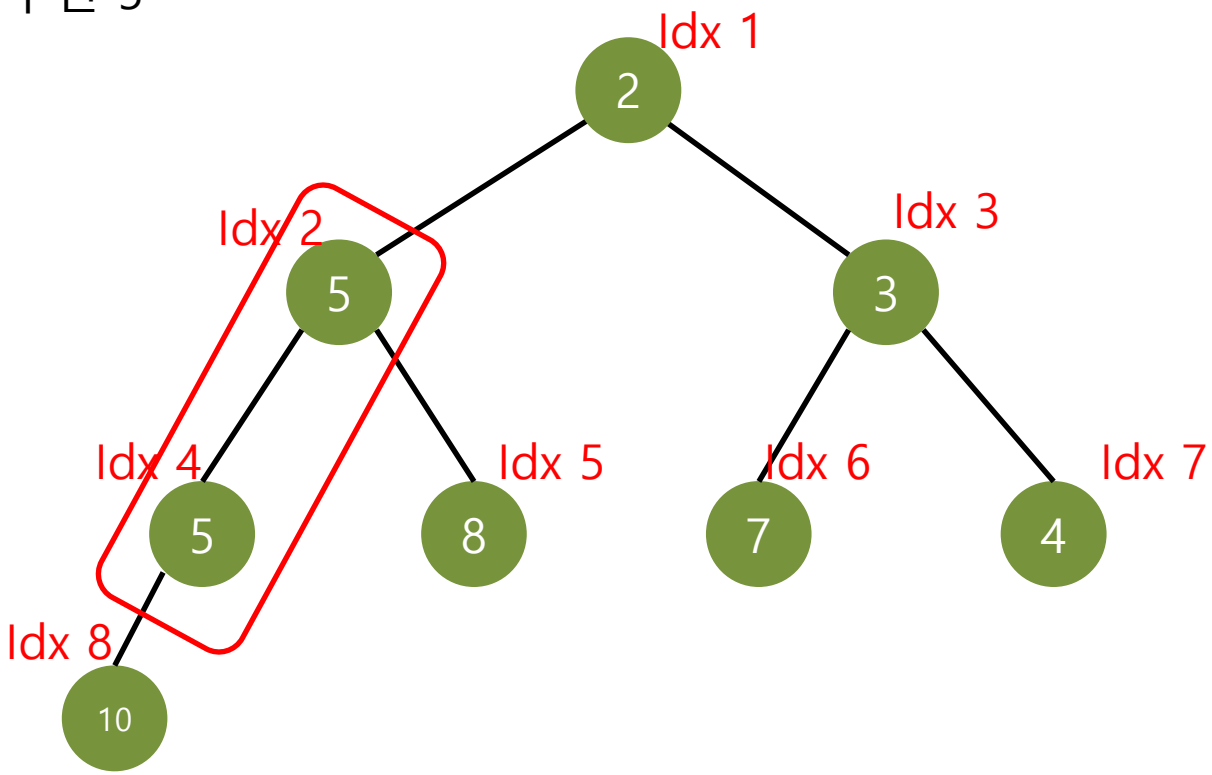
Delete() 구현 4



4. 구해진 자식 노드와 비교
부모 노드가 더 크므로 교환해야 한다

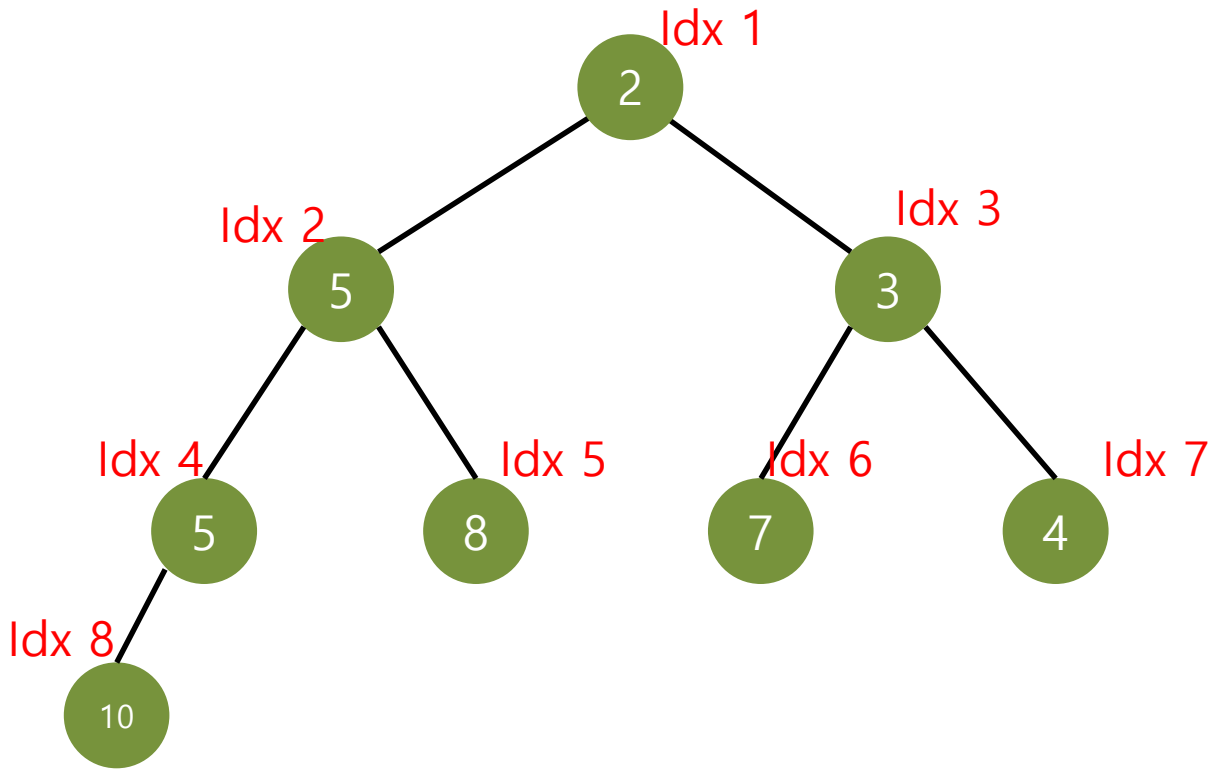
heap

Delete() 구현 5



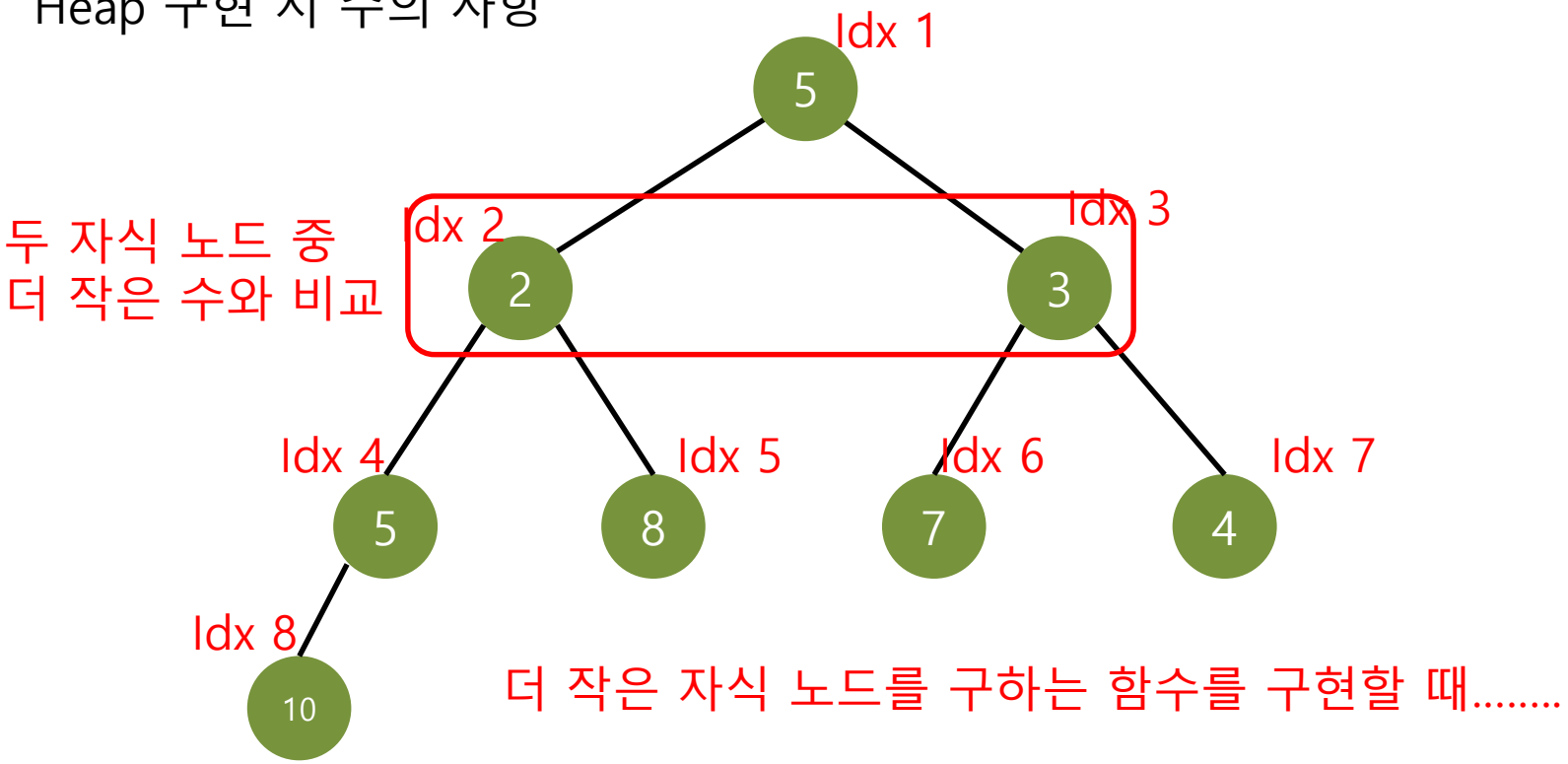
5. 자식 노드가 더 크거나 같을 때까지
같은 과정을 반복

heap



heap

Heap 구현 시 주의 사항



더 작은 자식 노드를 구하는 함수를 구현할 때.....

WhichIsPriorChild()
: 우선순위가 높은 (여기서는 더 작은 수)를 구하는 함수

heap

WhichIsPriorChild()
: 우선순위가 높은 (여기서는 더 작은 수)를 구하는 함수

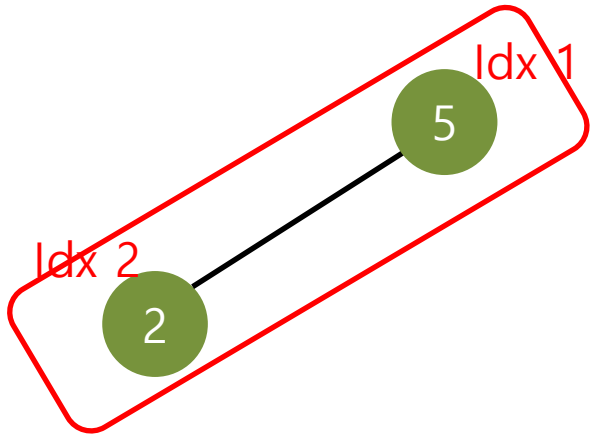
Case 1
: idx의 자식 노드가 없는 경우
즉 단말 노드인 경우

비교할 필요가 없다

heap

WhichIsPriorChild()
: 우선순위가 높은 (여기서는 더 작은 수)를 구하는 함수

Case 2
: idx의 자식 노드가 왼쪽 자식 노드만 있는 경우

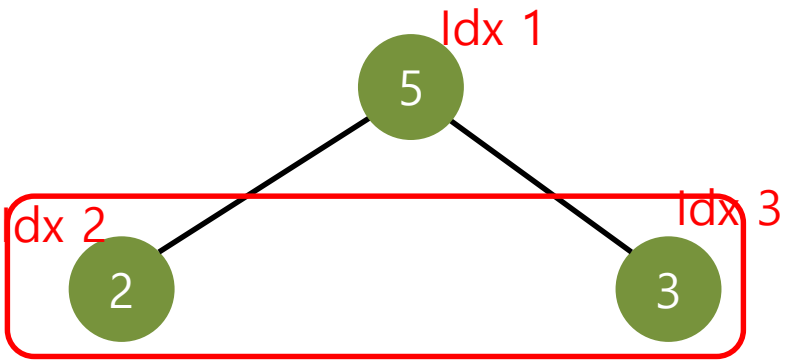


무조건 왼쪽 자식 노드랑 비교하면 된다

heap

WhichIsPriorChild()
: 우선순위가 높은 (여기서는 더 작은 수)를 구하는 함수

Case 3
: idx의 자식 노드가
왼쪽, 오른쪽 모두 있는 경우



두 자식 노드를 비교해 우선 순위가 높은
노드를 먼저 구한다

Priority queue

여기서 T는 저장하는 데이터 타입

Priority Queue ADT

bool IsEmpty();

void Enqueue(T data);

T Dequeue();

Priority Queue는 heap을 이용해 구현합니다

Priority queue

```
void Enqueue(T data)
{
    heap->Insert(data);
}

T Dequeue()
{
    return heap->Delete();
}
```