# 힙 HEAP

천수현

# HEAP

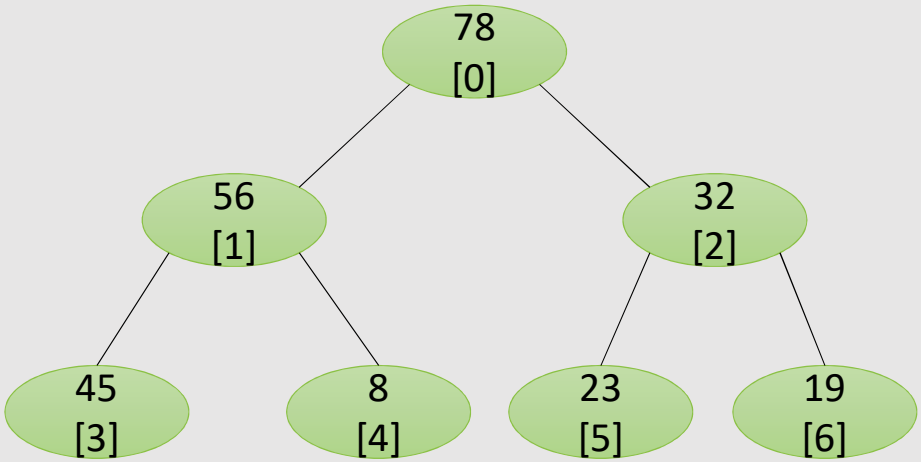
- 최대힙 : 완전트리이면서, Root가 모든 경우에 자식들보다 커야 함

  <서브트리의 Root도 그 자식들보다 커야 함>

- 최소힙 : 완전트리이면서, Root가 모든 경우에 자식들보다 작아야 함

  <서브트리의 Root도 그 자식들보다 작아야 함>

# 예 ) 최대힙




트리 구조

| 78 | 56 | 32 | 45 | 8 | 23 | 19 |
|---|---|---|---|---|---|---|
| [0] | [1] | [2] | [3] | [4] | [5] | [6] |

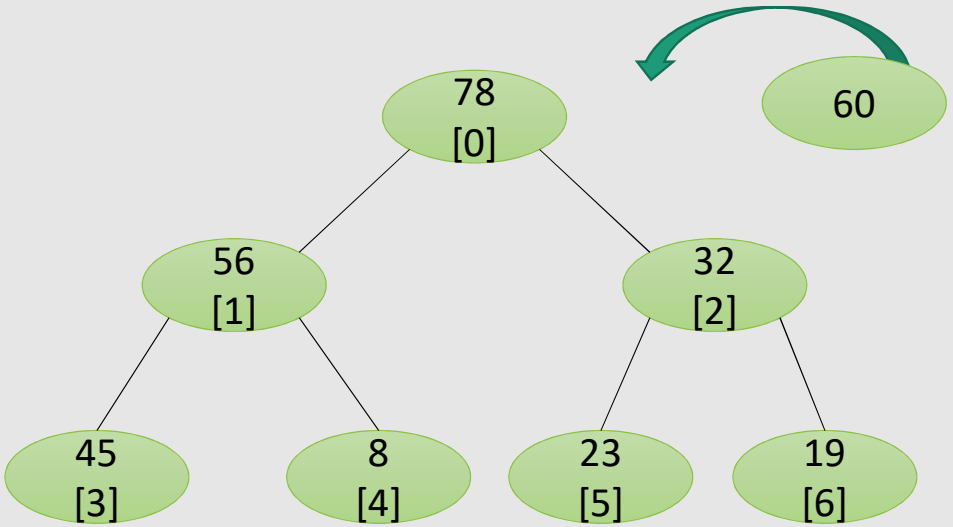
배열

# 힙 정렬

- 랜덤한 숫자의 순서를 힙으로 만들고 힙에서 하나씩 제거
- 이진 트리를 최대 힙으로 만들기 위하여 최대 힙으로 재구성 하는 과정 이 트리의 깊이 만큼 이루어 짐

  *N개의 데이터를 모두 빼야 정렬되므로 힙정렬의 시간 복잡도는 O(Nlog N)
- 우선 순위를 만들 때도 사용

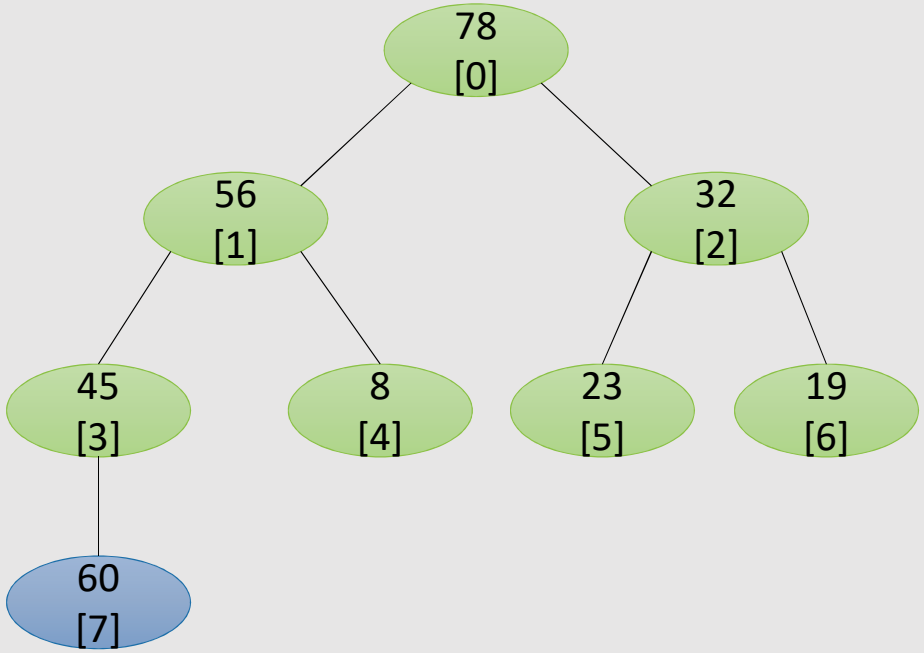
# 삽입




트리 구조

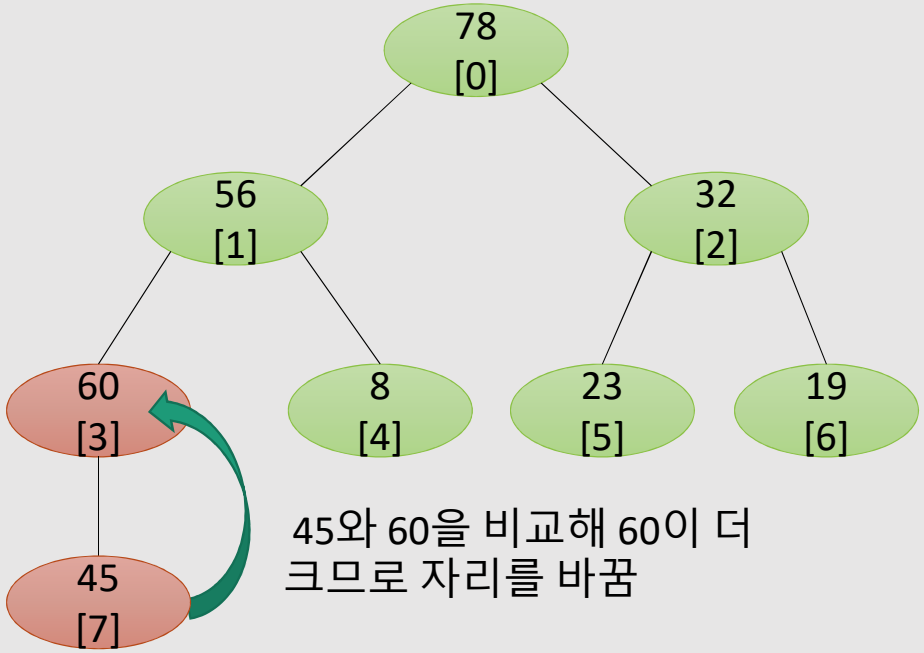
1. 마지막 노드로 추가
2. 부모노드와 크기를 비교해 자리를 바꿈

# 삽입




1. 마지막 노드로 추가
(완전 트리이므로 위치가 정해짐)

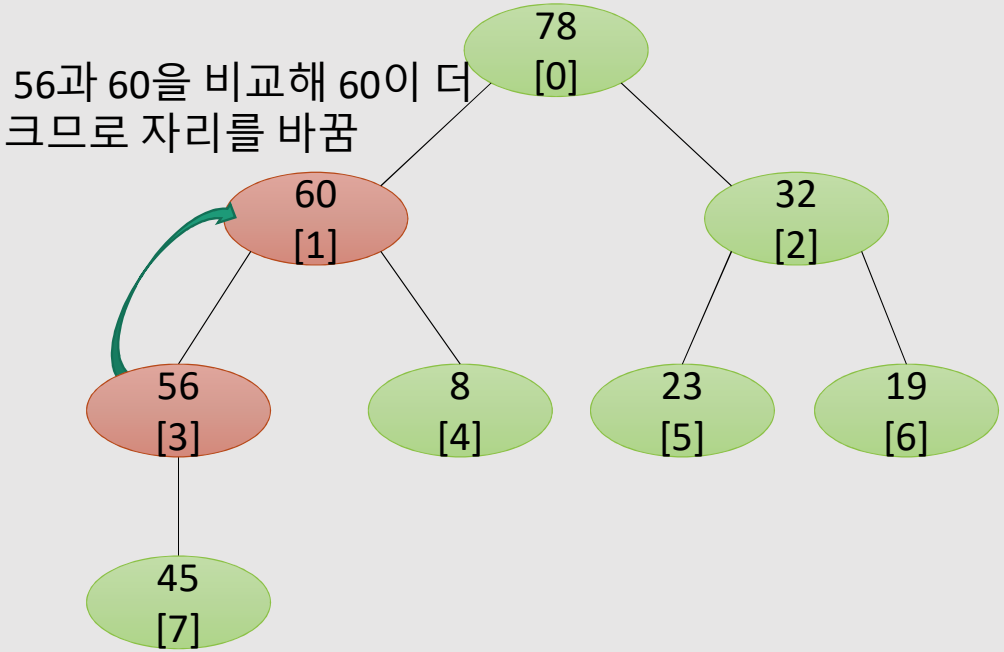
# 삽입




45와 60을 비교해 60이 더 크므로 자리를 바꿈

2. 부모노드와 크기를 비교해 자리를 바꿈

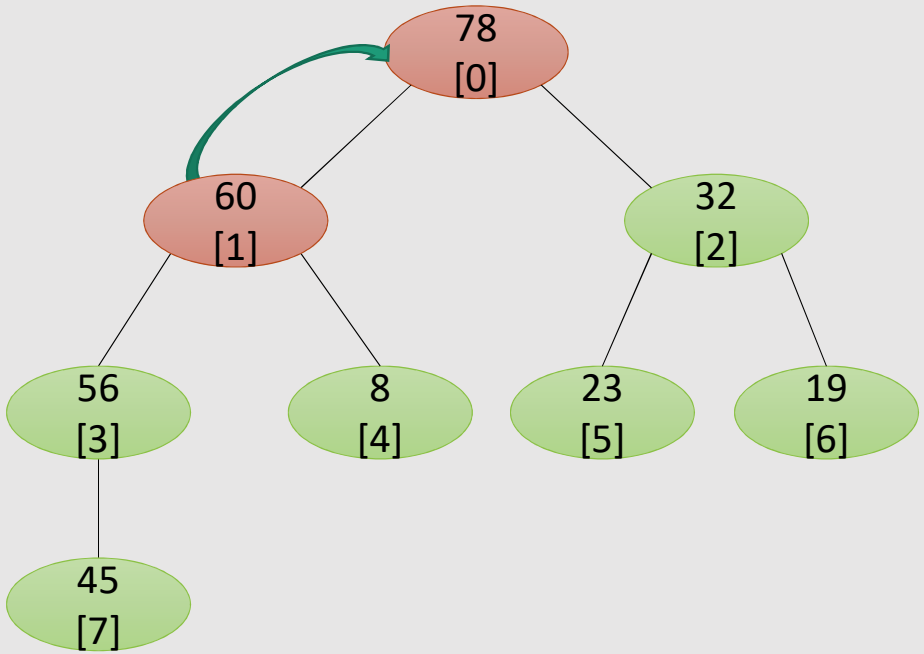
# 삽입




2. 부모노드와 크기를 비교해 자리를 바꿈

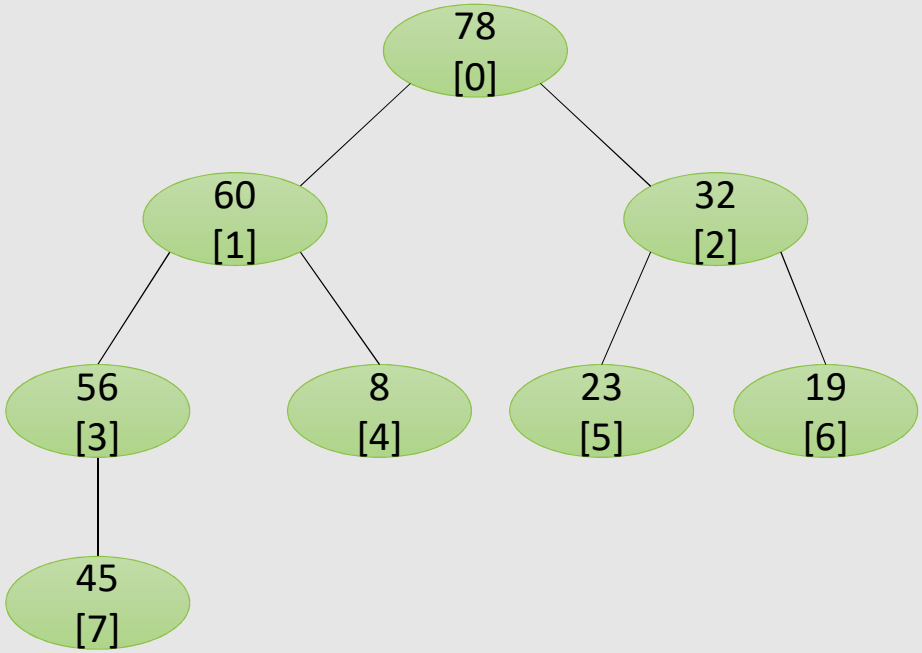
# 삽입

78과 60을 비교해 78이 더 크므로 삽입 끝

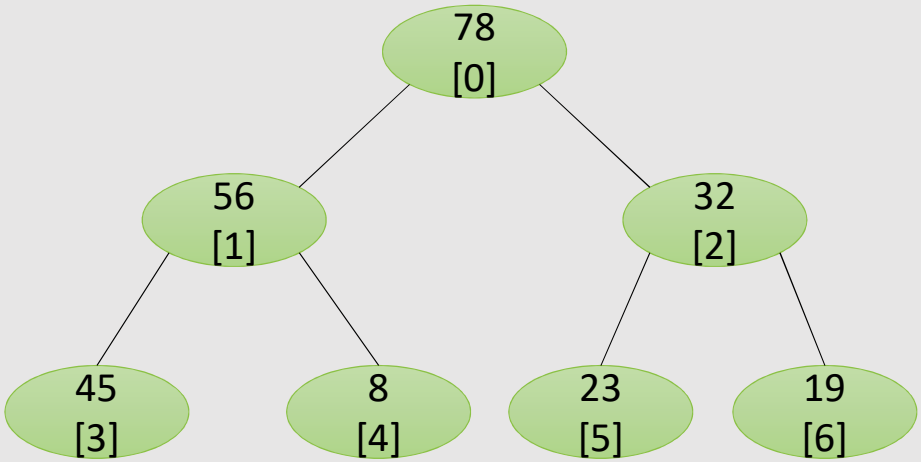



2. 부모노드와 크기를 비교해 자리를 바꿈

# 삽입



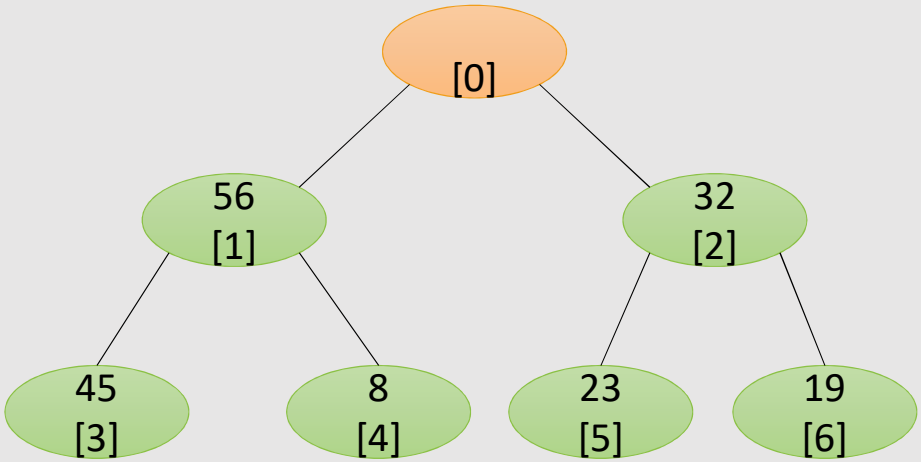
# 제거




트리 구조

1. Root 제거
2. 마지막 노드를 Root로 위치를 올려줌 (완전 트리 특성 이용)
3. Root가 된 노드와 자식 노드의 값을 비교해 자리를 바꿈
4. 재귀적으로 반복

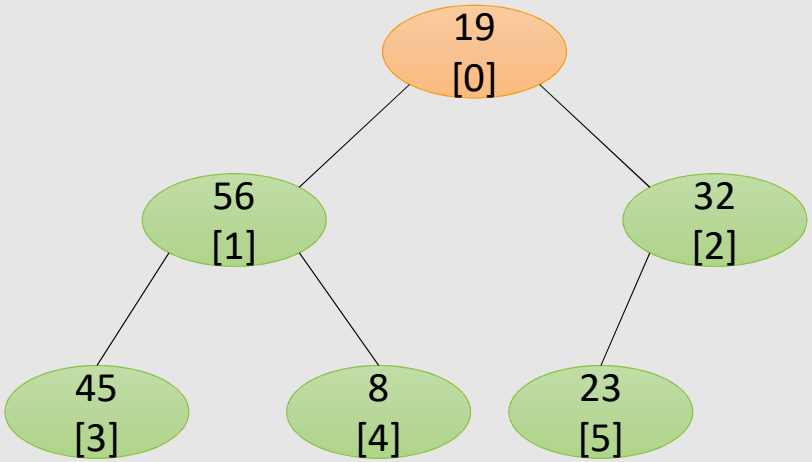
# 제거




트리 구조

1. Root 제거

# 제거

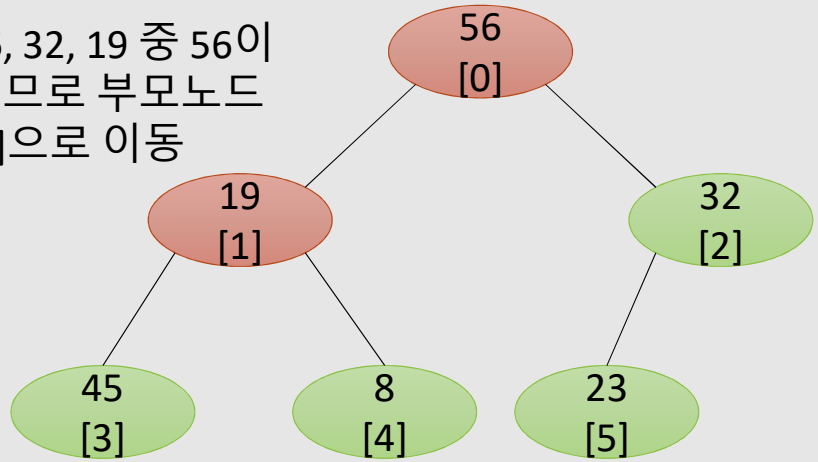



트리 구조

2. 마지막 노드를 Root로 위치를 올려줌
(완전 트리 특성 이용)

# 제거

56, 32, 19 중 56이 크므로 부모노드 [0]으로 이동

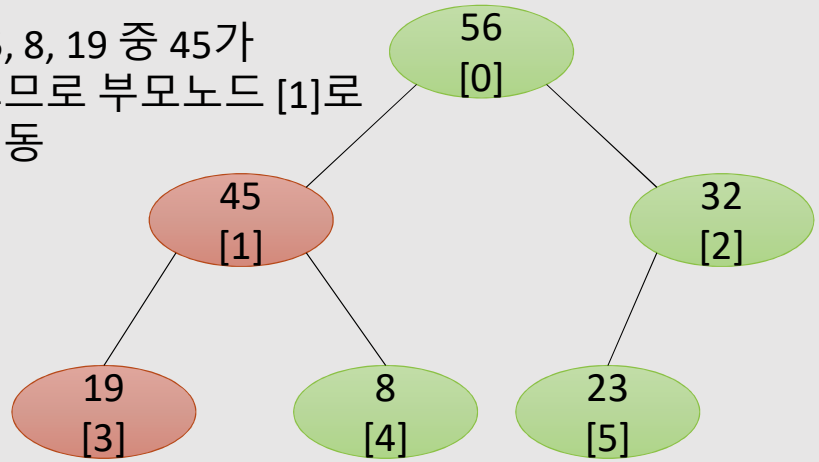



트리 구조

3. Root가 된 노드와 자식 노드의 값을 비교해 자리를 바꿈

# 제거

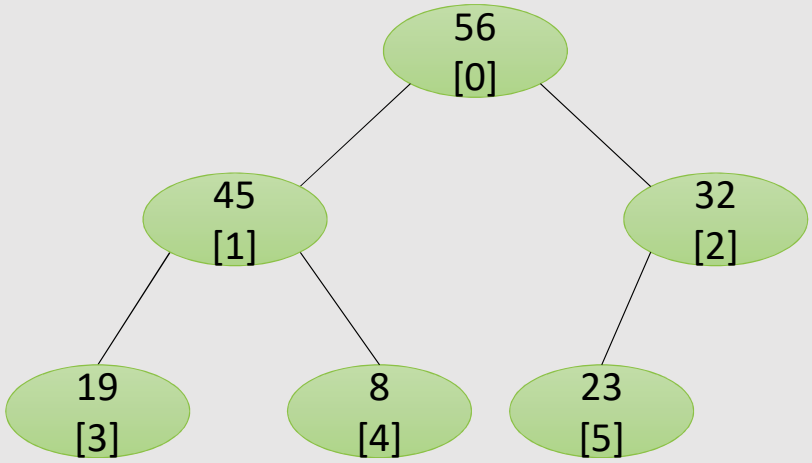
45, 8, 19 중 45가 크므로 부모노드 [1]로 이동




트리 구조

4. 재귀적으로 반복

# 제거



참고
http://emzei.tistory.com/125

https://www.zerocho.com/category/Algorithm/post/582de223d4416a001860e763