자료 구조

(우선순위 큐, 히프)

https://youtu.be/qSUYW2-gx0E

송경주

목차

우선순위 큐

- 우선순위 큐란?

기존 큐에 우선순위 개념을 도입한 것.
데이터들이 우선순위를 가지고 있음.
스택이나 큐도 우선순위 큐로 규현이 가능함.

자료구조	삭제되는 요소
스택	가장 최근에 들어온 데이터
큐	가장 먼저 들어온 데이터
우선순위 큐	가장 우선순위가 높은 데이터

우선순위 큐의 구현 - 배열사용

<배열을 사용하여 우선순위 큐 구현>

1. 정렬이 안 된 배열

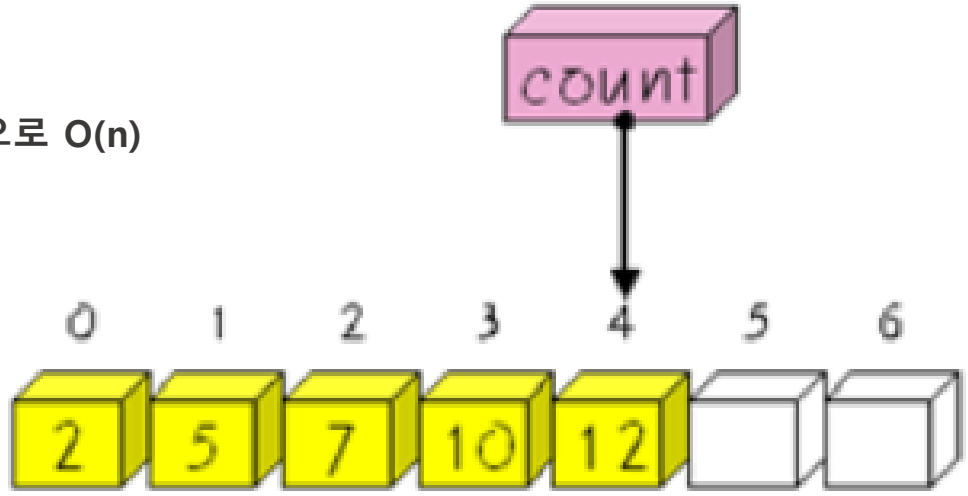
삽입 : 기존 요소들의 맨 끝에 붙임 → O(1)

삭제 : 처음부터 끝까지 모든 요소 스캔 → O(n)

2. 정렬이 되어 있는 배열

삽입 : 다른 요소와 비교하여 우선순위에 따라 삽입 위치를 결정 → 일반적으로 O(n)

삭제 : 이미 정렬 되어 있기 때문에 위치에 있는 요소 삭제 → O(1)



정렬된 배열로 구현된 우선순위 큐

우선순위 큐의 구현 - 연결리스트 사용

<연결리스트를 사용하여 우선순위 큐 구현>

1. 정렬이 안 된 리스트

삽입 : 첫번째 노드에 삽입 → O(1)

삭제 : 처음부터 끝까지 모든 노드 스캔 → O(n)

2. 정렬이 되어 있는 리스트

삽입 : 우선순위 값을 기준으로 삽입위치를 찾음 → O(n)

삭제 : 첫번째 노드 삭제 → O(1)

정렬된 연결리스트로 구현된 우선순위 큐

우선순위 큐의 구현 - 히프 사용

<히프를 사용하여 우선순위 큐 구현>

히프 : 완전 이진트리의 일종. 우선순위 큐를 위하여 만들어짐.
반정렬 상태를 유지하고 반정렬 상태를 이용하여 우선순위 큐를 구현.

표현 방법	삽 입	삭 제
순서없는 배열	O(1)	O(n)
순서없는 연결 리스트	O(1)	O(n)
정렬된 배열	O(n)	O(1)
정렬된 연결 리스트	O(n)	O(1)
히프	O(logn)	O(logn)

히프(heap)

히프 (heap) : 부모 노드의 키 값이 자식 노드의 키 값보다 항상 큰(작은) 이진 트리

Ex) A가 B의 부모 노드일 때, 부모 노드의 키 값이 자식 노드의 키 값보다 항상 크다고 하면
key(A)≥key(B) 히프 조건이 항상 성립하는 트리.

히프의 종류

1. 최대 히프(max heap)

부모 노드의 키 값이 자식 노드의 키 값보다 크거나 같은 완전 이진 트리

key(부모노드) ≥ key(자식노드)

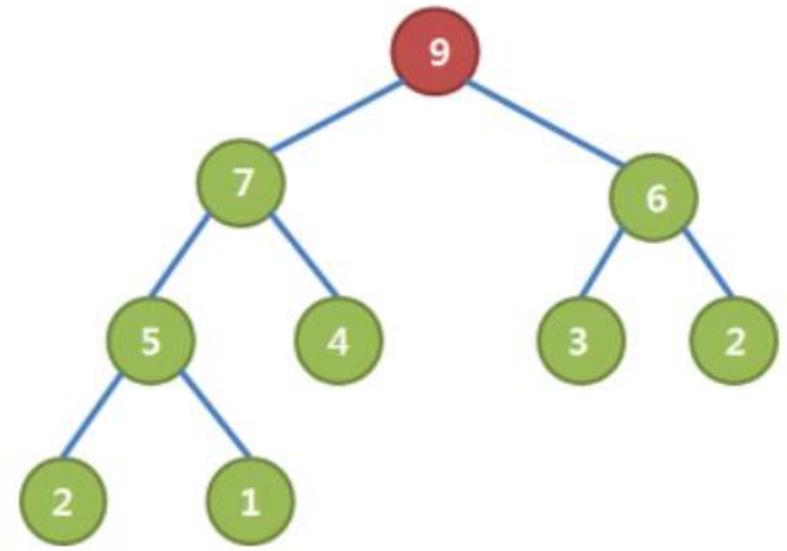
2. 최소 히프(min heap)

부모 노드의 키 값이 자식 노드의 키 값보다 작거나 같은 완전 이진 트리

key(부모노드) ≤ key(자식노드)

히프(heap) - 최대 히프

최대 히프 (heap) : 부모 노드의 키 값이 자식 노드의 키 값보다 크거나 같은 완전 이진 트리

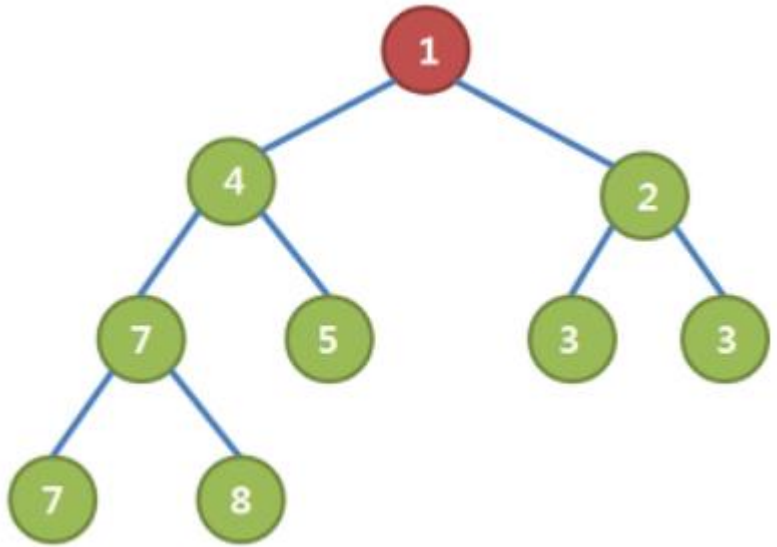


완전 이진 트리를 만족.
부모 노드의 값이 자식 노드 값보다 크거나 같음.

최대 히프 조건을 만족함.

힙(heap) - 최소 힙

최소 힙 (heap) : 부모 노드의 키 값이 자식 노드의 키 값보다 작거나 같은 완전 이진 트리

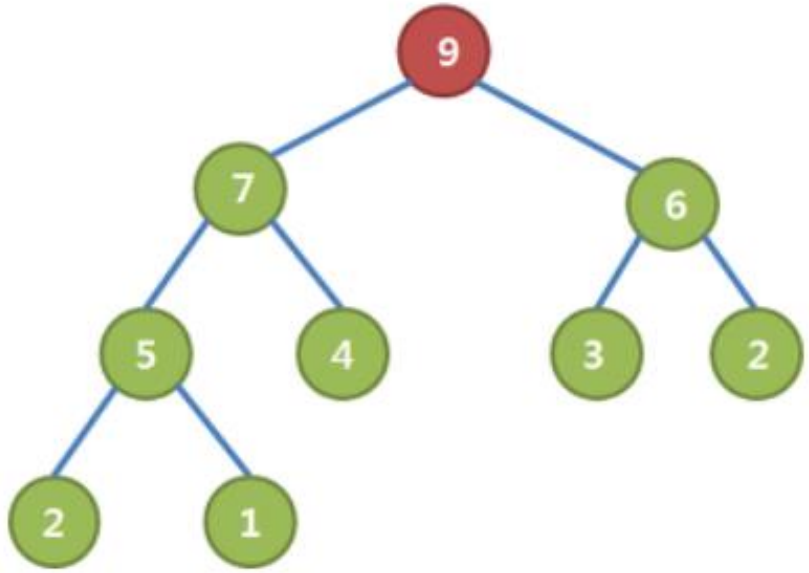


완전 이진 트리를 만족.

부모 노드의 값이 자식 노드 값보다 작거나 같음.

최소 힙 조건을 만족함.

히프의 구현

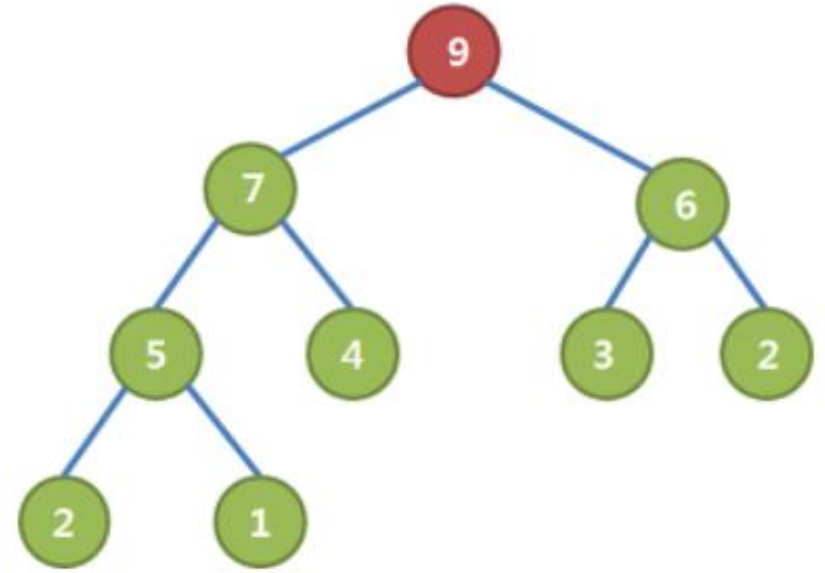


	9	7	6	5	4	3	2	2	1	

히프는 완전 이진 트리 이므로 차례대로 번호를 붙일 수 있으며 이 번호를 인덱스로 생각하면 배열에 히프의 노드들을 저장할 수 있다.

→ 히프를 저장하는 표준 자료구조 : 배열

히프의 구현



0	1	2	3	4	5	6	7	8	9
	9	7	6	5	4	3	2	2	1

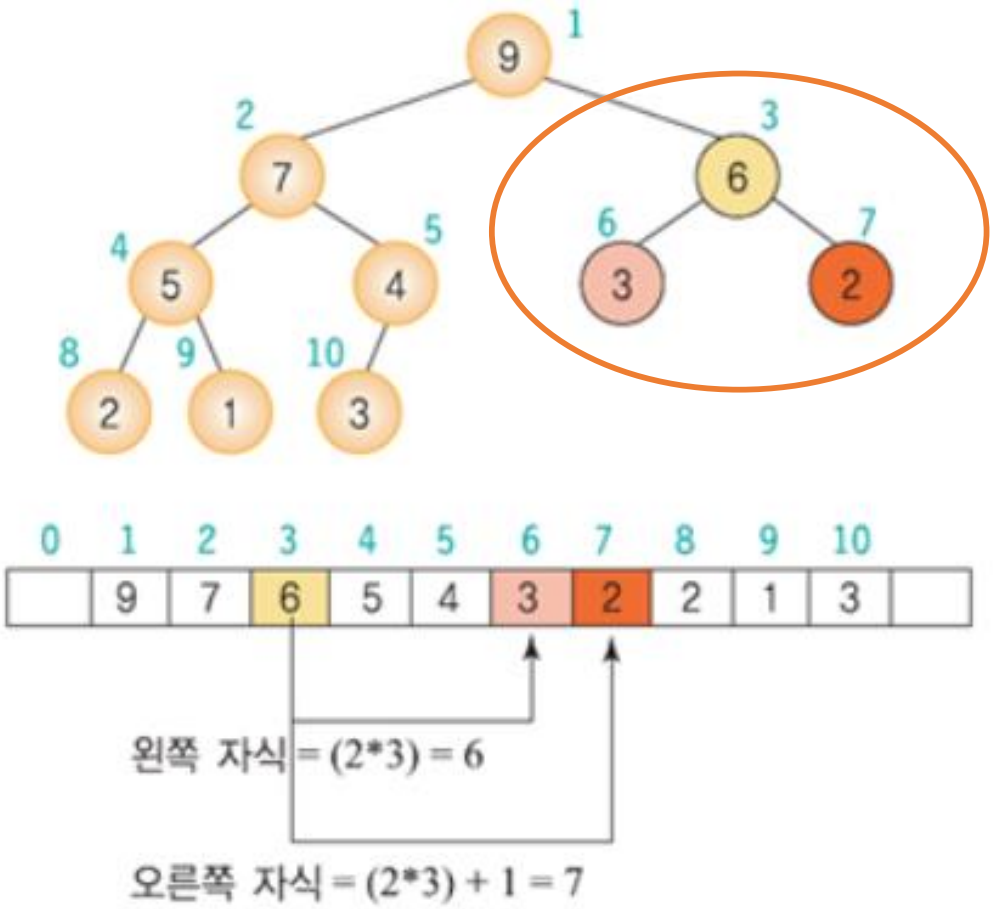
자식 노드와 부모 노드의 관계

왼쪽 자식의 인덱스 = (부모의 인덱스)*2

오른쪽 자식의 인덱스 = (부모의 인덱스)*2 + 1

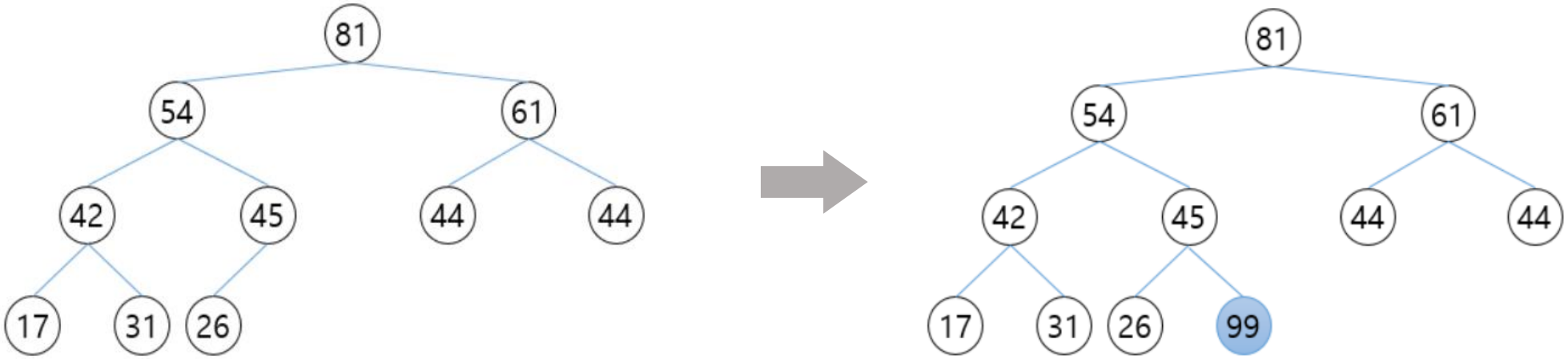
부모의 인덱스 = 자식의 인덱스/2

히프의 구현


1
9
2
7
3
6
4
5
4
5
6
3
7
2
8
2
9
1
10
3
0 1 2 3 4 5 6 7 8 9 10
9 7 6 5 4 3 2 2 1 3
왼쪽 자식 = (2*3) = 6
오른쪽 자식 = (2*3) + 1 = 7

히프의 삽입

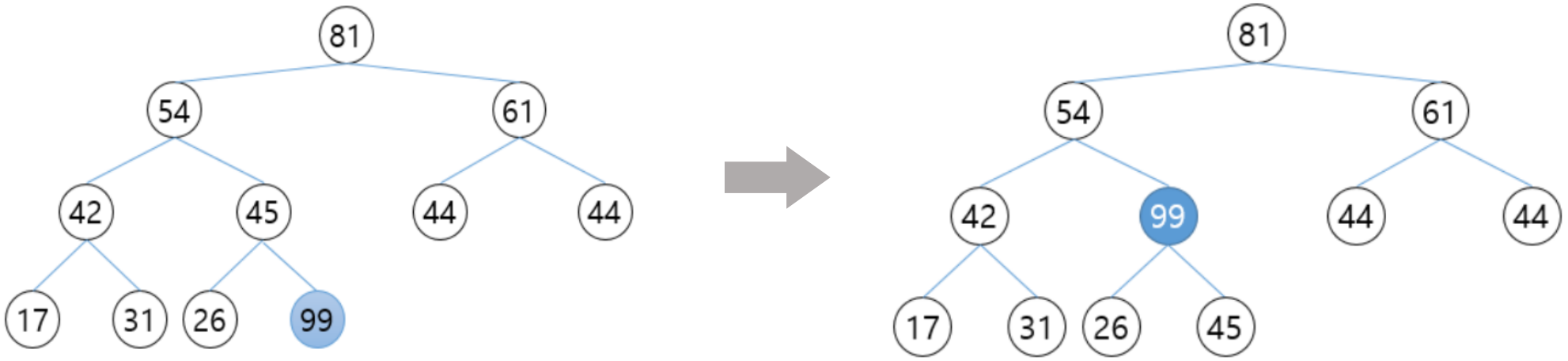
새로운 요소 99 삽입



히프의 마지막 자리에 99 노드를 추가한다.

히프의 삽입

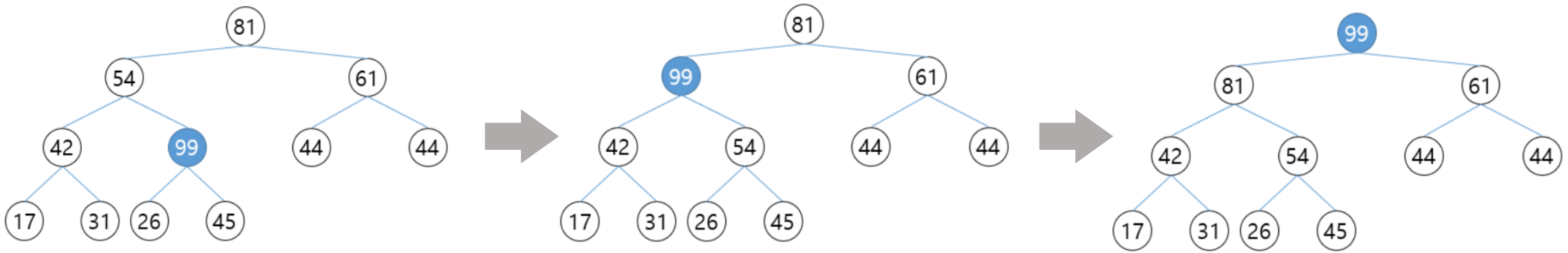
새로운 요소 99 삽입



부모 노드와 대소관계를 비교하여 자식 노드인 99가 부모 노드인 49보다 크므로 자리를 바꿔준다.

히프의 삽입

새로운 요소 99 삽입



계속 부모와 대소를 비교해가며 자리를 바꿔준다.

최종 대소관계가 올바르므로 최대값 히프 완성.

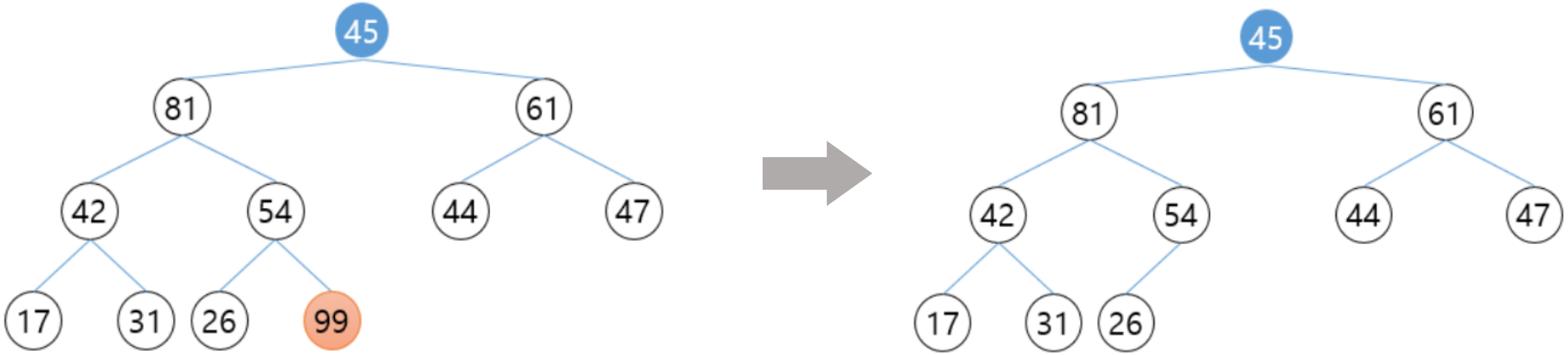
히프의 삭제

루트 노드 요소 99 삭제

루트 노드와 히프의 맨 마지막 자리의 노드 45의 자리를 바꾼다.

히프의 삭제

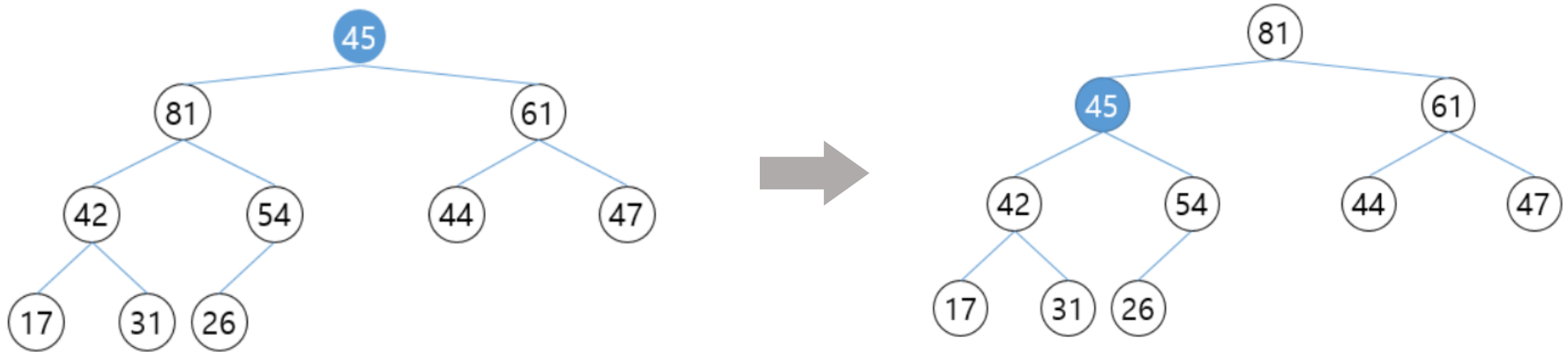
루트 노드 요소 99 삭제



노드 99를 삭제한다.

히프의 삭제

루트 노드 요소 99 삭제

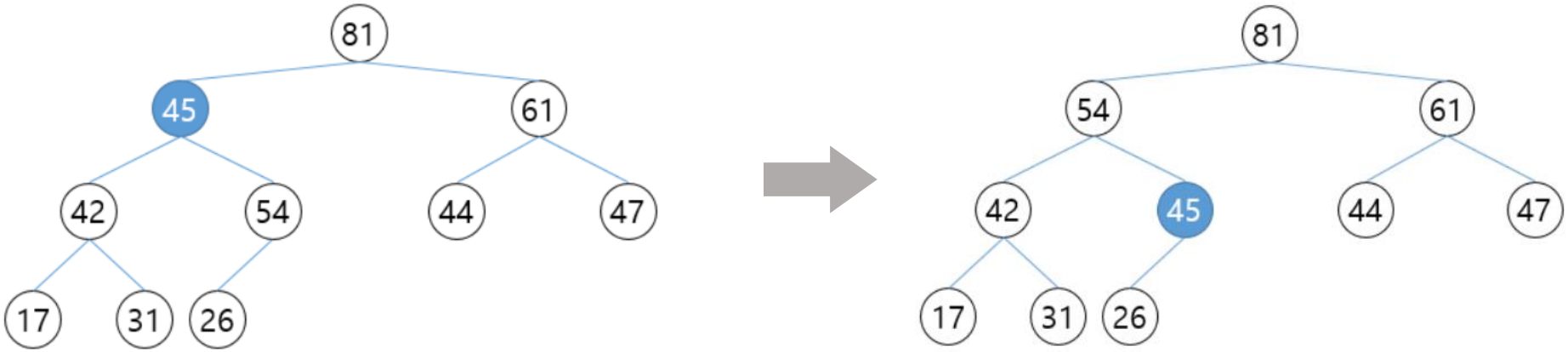


노드 99가 삭제 되었으므로 히프를 재구축 한다.

부모 노드 45가 자식 노드 81, 61 보다 더 작으므로 자식 노드 중 더 큰 자식 노드와 자리를 바꾼다.

히프의 삭제

루트 노드 요소 99 삭제



45와 자식 노드들의 대소 관계를 다시 비교하여 위치를 54와 바꿔준다.
마지막으로 대소관계를 확인하여 히프 규칙에 맞게 배치된 것을 확인한다.

히프의 응용

히프 정렬: 히프를 사용하는 정렬 알고리즘.

이산 이벤트 시뮬레이션 : 모든 시간의 진행은 이벤트의 발생에 의해서만 이루어지는 시뮬레이션

EX) 아이스크림 가게에서 발생되는 이벤트 3가지

손님이 도착하는 이벤트 (ARRIVAL)

손님이 주문하는 이벤트 (ORDER)

손님이 가게를 떠나는 이벤트 (LEAVE)

감사합니다 :-)