13주차 3차시 정렬 3

【학습목표】

1. 병합 정렬을 설명할 수 있다.
2. 선택 방식에 따라 히프 정렬과 트리 정렬을 구분할 수 있다.

학습내용1 : 병합 정렬

1. 개요

* 여러 개의 정렬된 자료의 집합을 병합하여 한 개의 정렬된 집합으로 만드는 방법

* 부분집합으로 분할(divide)하고, 각 부분집합에 대해서 정렬 작업을 완성(conquer)한 후에 정렬된 부분집합들을 다시 결합(combine)하는 분할 정복(divide and conquer) 기법 사용

* 병합 정렬 방법의 종류
- 2-way 병합 : 2개의 정렬된 자료의 집합을 결합하여 하나의 집합으로 만드는 병합 방법
- n-way 병합 : n개의 정렬된 자료의 집합을 결합하여 하나의 집합으로 만드는 병합 방법

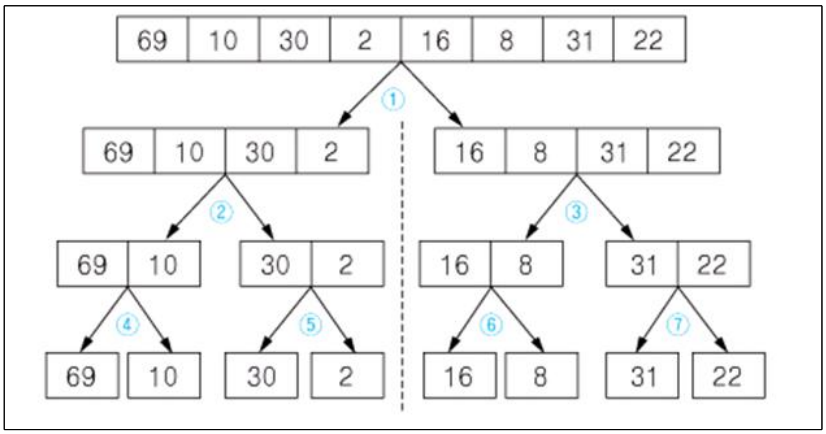
* 2-way 병합 정렬 : 세 가지 기본 작업을 반복 수행하면서 완성

(1) 분할(divide) : 입력 자료를 같은 크기의 부분집합 2개로 분할한다.
(2) 정복(conquer) : 부분집합의 원소들을 정렬한다.
부분집합의 크기가 충분히 작지 않으면 순환호출을 이용하여 다시 분할 정복 기법을 적용한다.
(3) 결합(combine) : 정렬된 부분집합들을 하나의 집합으로 결합한다.

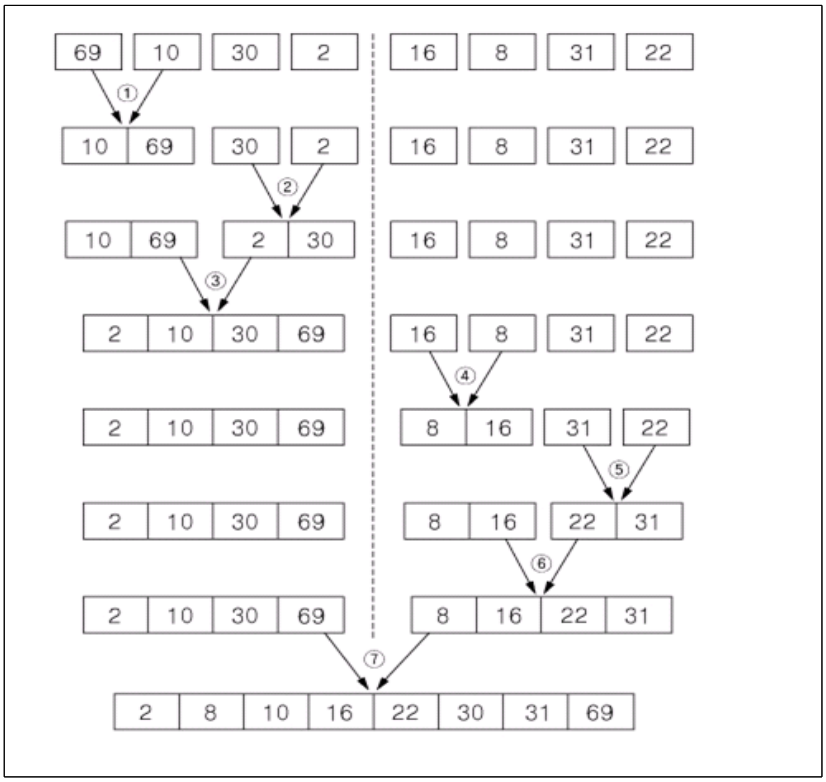
2. 병합 정렬 수행 과정

* 정렬되지 않은 {69, 10, 30, 2, 16, 8, 31, 22 }의 자료들을 병합 정렬 방법으로 정렬

① 분할 단계 : 정렬할 전체 자료의 집합에 대해서 최소 원소의 부분집합이 될 때까지 분할작업을 반복하여 1개의 원소를 가진 부분집합 8개를 만든다.



② 병합단계 : 2개의 부분집합을 정렬하면서 하나의 집합으로 병합한다. 8개의 부분집합이 1개로 병합될 때까지 반복한다.



3. 병합 정렬 알고리즘

알고리즘 10-7 병합 정렬 알고리즘

```
mergeSort(a[],m,n)
    if (a[m:n]의 원소수 > 1) then {
        전체 집합을 두 개의 부분집합으로 분할;
        mergeSort(a[], m, middle);
        mergeSort(a[], middle+1, n);
        merge(a[m:middle], a[middle+1:n]);
    }
end mergeSort()
```

* 메모리 사용공간
- 각 단계에서 새로 병합하여 만든 부분집합을 저장할 공간이 추가로 필요
- 원소 n개에 대해서 (2 x n)개의 메모리 공간 사용

* 연산 시간
- 분할 단계 : n개의 원소를 분할하기 위해서 log2n번의 단계 수행
- 병합 단계 : 부분집합의 원소를 비교하면서 병합하는 단계에서 최대 n번의 비교연산 수행
- 전체 병합 정렬의 시간 복잡도 : $O(n \log_2 n)$

학습내용2 : 히프 정렬

1. 개요

* 8장의 히프 자료구조를 이용한 정렬 방법

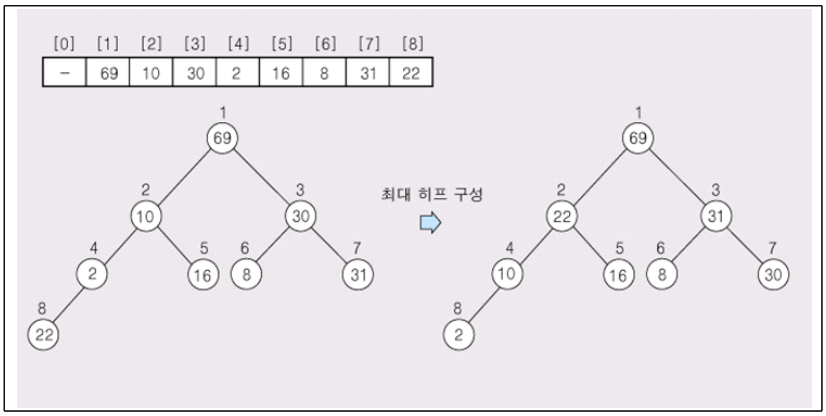
* 히프에서는 항상 가장 큰 원소가 루트 노드가 되고 삭제 연산을 수행하면 항상 루트 노드의 원소를 삭제하여 반환
- 최대 히프에 대해서 원소의 개수만큼 삭제 연산을 수행하여 내림차순으로 정렬 수행
- 최소 히프에 대해서 원소의 개수만큼 삭제 연산을 수행하여 오름차순으로 정렬 수행

* 히프 정렬 수행 방법

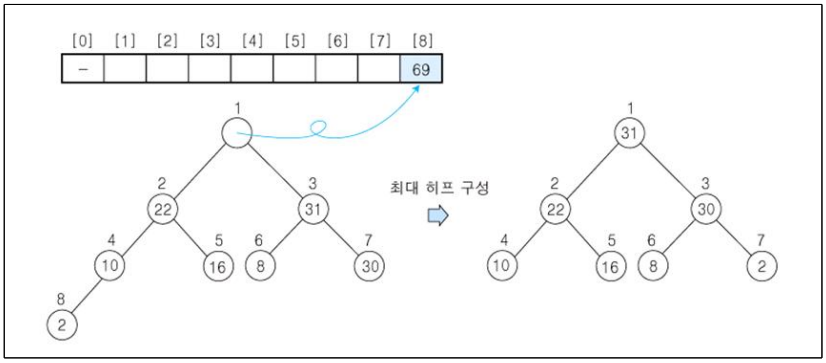
(1) 정렬할 원소들을 입력하여 최대 히프 구성
(2) 히프에 대해서 삭제 연산을 수행하여 얻은 원소를 마지막 자리에 배치
(3) 나머지 원소에 대해서 다시 최대 히프로 재구성
원소의 개수만큼 (2)~(3) 을 반복 수행

2. 히프 정렬 수행과정

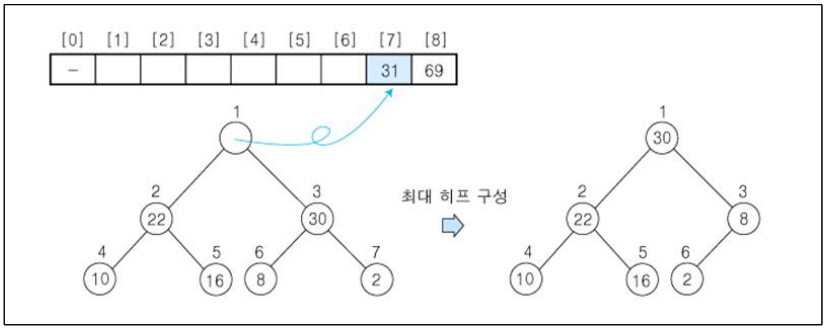
* 정렬되지 않은 {69, 10, 30, 2, 16, 8, 31, 22 }의 자료들을 히프 정렬 방법으로 정렬
- 초기 상태 : 정렬할 원소가 8개 이므로 노드가 8개인 완전 이진 트리를 만들고, 최대 히프로 구성한다.



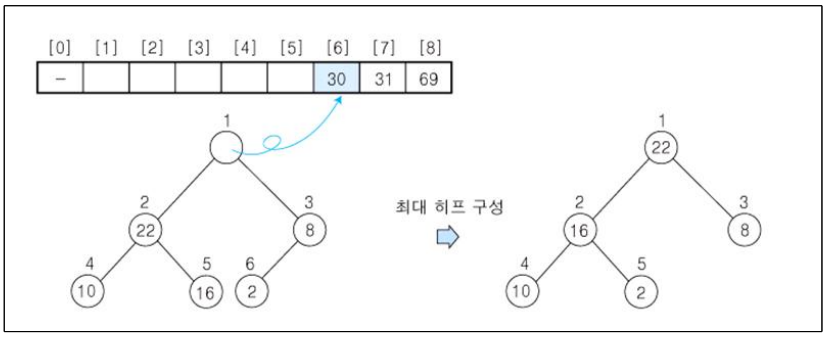
① 히프에 삭제 연산을 수행하여 루트 노드의 원소 69를 구해서 배열의 마지막 자리에 저장, 나머지 원소들에 대해서 최대 히프로 재구성



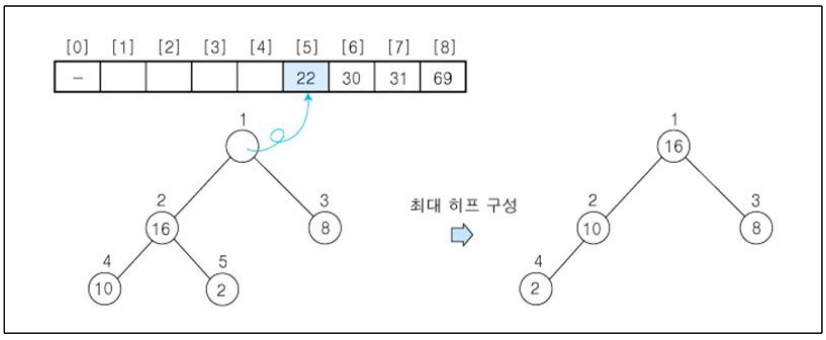
② 히프에 삭제 연산을 수행하여 루트 노드의 원소 31을 구해서 배열의 비어있는 마지막 자리에 저장, 나머지 히프를 최대 히프로 재구성



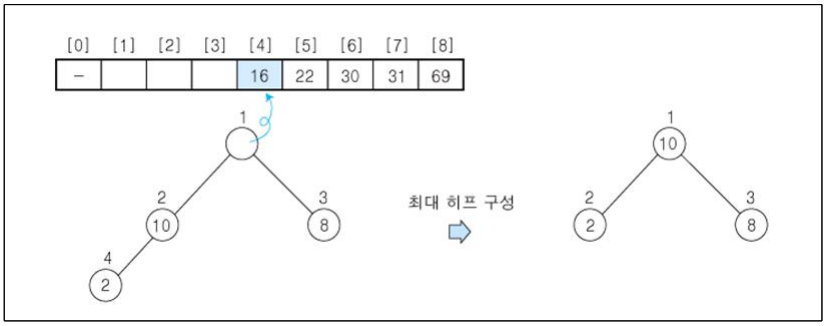
③ 히프에 삭제 연산을 수행하여 루트 노드의 원소 30을 구해서 배열의 비어있는 마지막 자리에 저장 나머지 원소들에 대해서 최대 히프로 재구성



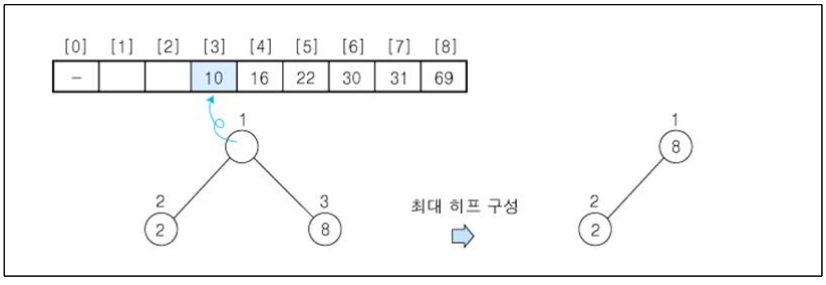
④ 히프에 삭제 연산을 수행하여 루트 노드의 원소 22를 구해서 배열의 비어있는 마지막 자리에 저장, 나머지 원소들에 대해서 최대 히프로 재구성



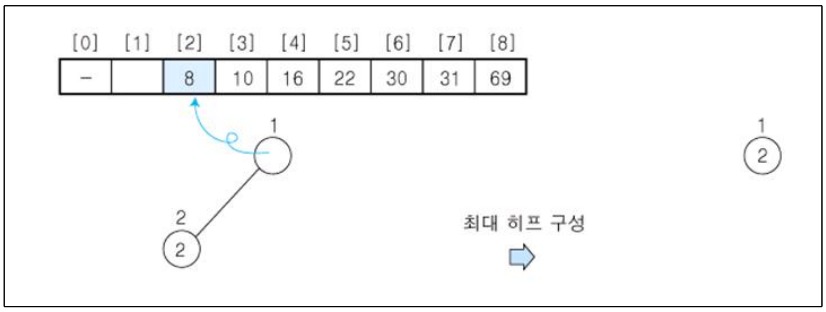
⑤ 히프에 삭제 연산을 수행하여 루트 노드의 원소 16을 구해서 배열의 비어있는 마지막 자리에 저장 나머지 원소들에 대해서 최대 히프로 재구성



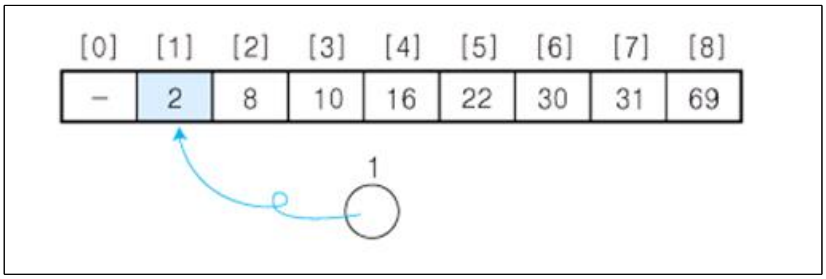
⑥ 히프에 삭제 연산을 수행하여 루트 노드의 원소 10을 구해서 배열의 비어있는 마지막 자리에 저장 나머지 원소들에 대해서 최대 히프로 재구성



⑦ 히프에 삭제 연산을 수행하여 루트 노드의 원소 8을 구해서 배열의 비어있는 마지막 자리에 저장 나머지 원소들에 대해서 최대 히프로 재구성



⑧ 히프에 삭제 연산을 수행하여 루트 노드의 원소 2를 구해서 배열의 비어있는 마지막 자리에 저장 나머지 히프를 최대 히프로 재구성하는데 공백 히프가 되었으므로 히프 정렬 종료



3. 히프 정렬 알고리즘

* 히프 정렬 알고리즘

알고리즘 10-9 히프 정렬 알고리즘

```
heapSort(a[])
    n ← a.length-1;
    for (i ← n/2; i ≥ 1; i ← i-1) do {    // 배열 a[]를 히프로 변환
        makeHeap(a, i, n);
    for (i ← n-1; i ≥ 1; i ← i-1) do {
        temp ← a[1];                      // 히프의 루트 노드 원소를
        a[1] ← a[i+1];                    // 배열의 비어있는
        a[i+1] ← temp;                    // 마지막 자리에 저장

        makeHeap(a, 1, i);
    }
end heapSort()
```

* 히프 정렬 알고리즘의 히프 재구성 알고리즘

알고리즘 10-10 히프 재구성 알고리즘

```
makeHeap(a[], h, m)
    for (j ← 2*h; j ≤ m; j ← 2*j) do {
        if (j < m) then
            if (a[j] < a[j+1]) then j ← j+1;
        if (a[h] ≥ a[j]) then exit;
        else a[j/2] ← a[j];
    }
    a[j/2] ← a[h];
end makeHeap()
```

* 메모리 사용공간
- 원소 n개에 대해서 n개의 메모리 공간 사용
- 크기 n의 히프 저장 공간

* 연산시간
- 히프 재구성 연산 시간
    - n개의 노드에 대해서 완전 이진 트리는 log2(n+1)의 레벨을 가지므로 완전 이진 트리를 히프로 구성하는 평균시간은 O(log2n)
    - n개의 노드에 대해서 n번의 히프 재구성 작업 수행
- 평균 시간 복잡도 : $O(n \log_2 n)$

학습내용3 : 트리 정렬

1. 개요

* 8장의 이진 탐색 트리를 이용하여 정렬하는 방법

* 히프 정렬 수행 방법

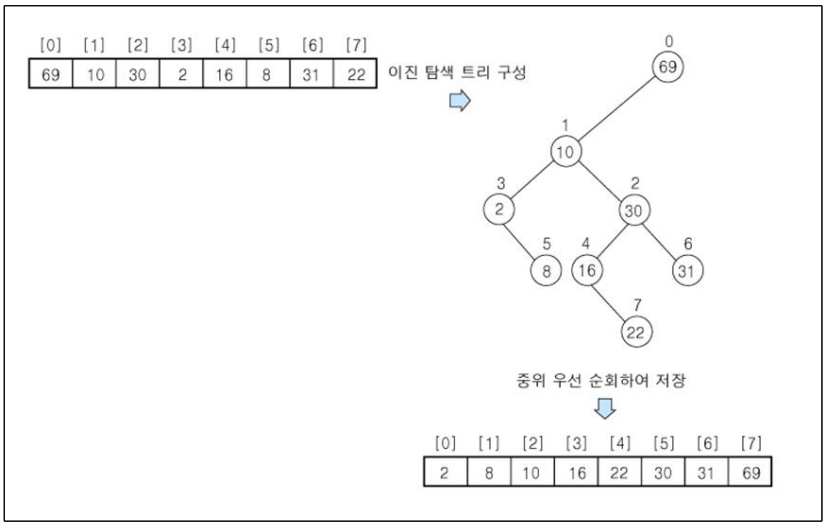
(1) 정렬할 원소들을 이진 탐색 트리로 구성한다.

(2) 이진 탐색 트리를 중위 우선 순회 한다.

-중위 순회 경로가 오름차순 정렬이 된다.

2. 트리 정렬 수행 과정

* 정렬되지 않은 {69, 10, 30, 2, 16, 8, 31, 22 }의 자료들을 트리 정렬
① 원소가 8개를 차례대로 트리에 삽입하여 이진 탐색 트리 구성
② 이진 탐색 트리를 중위 우선 순회 방법으로 순회하면서 저장



3. 트리 정렬 알고리즘

알고리즘 10-11 트리 정렬 알고리즘

```
treeSort(a[], n)
    for (i ← 0; i < n; i ← i+1) do
        insert(BST, a[i]);          // 이진 탐색 트리의 삽입 연산
    inorder(BST);                   // 중위 순회 연산
end treeSort()
```

* 메모리 사용공간
- 원소 n개에 대해서 n개의 메모리 공간 사용
- 크기 n의 이진 탐색 트리 저장 공간

* 연산 시간
- 노드 한 개에 대한 이진 탐색 트리 구성 시간 : O(log2n)
- n개의 노드에 대한 시간 복잡도 : $O(n \log_2 n)$

【학습정리】

1. 병합 정렬은 여러 개의 정렬된 자료의 집합을 부분집합으로 분할하고 각 부분집합에 대해서 정렬 작업을 완성한 후에 정렬된 부분집합들을 다시 결합하는 분할 정복 기법을 사용한다.
- n개의 정렬된 자료의 집합을 결합하여 하나의 집합으로 만드는 병합 방법을 n-way 병합이라고 한다.

2. 선택 방식에는 히프 정렬과 트리 정렬이 있다.
- 히프 정렬 : 히프 자료구조를 이용하여 정렬하는 방법
- 트리 정렬 : 이진 탐색 트리를 이용하여 정렬하는 방법이다. 정렬할 원소들을 이진 탐색 트리로 구성하고 중위 우선 순회 방법을 사용하여 이진 탐색 트리의 원소들을 순회하여 꺼내면 오름차순 정렬이 된다.