5주차 1차시 분할과 정복

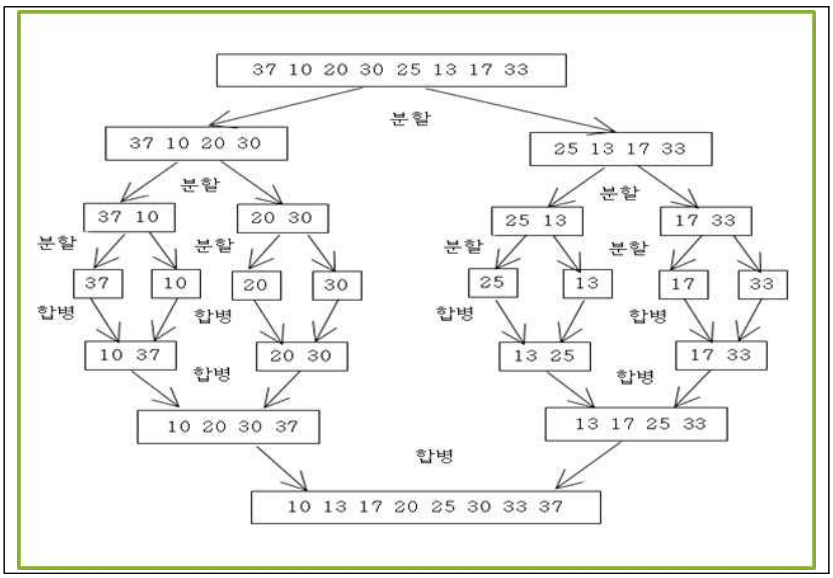
【학습목표】

1. 분할과 정복 개념을 이해할 수 있다.
2. 퀵 정렬, 힙 정렬을 이해할 수 있다.

학습내용1 : 퀵 정렬

1. 분할 정복

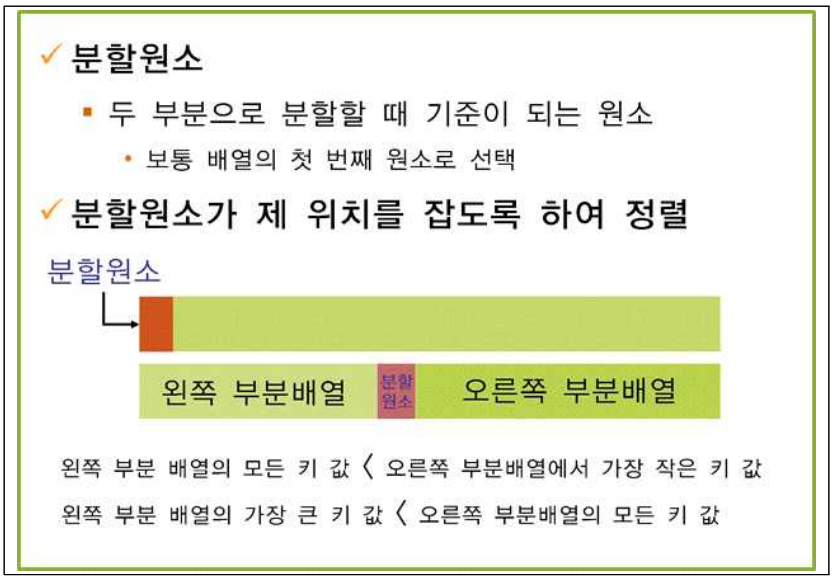
분할정복 방법의 세 단계는 원래의 문제를 소문제로 분할하는 분할 단계, 소문제의 해를 순환적으로 구하는 정복단계, 소문제의 해를 합하여 원래 문제에 대한 해를 만들어 내는 합병단계로 구성된다.

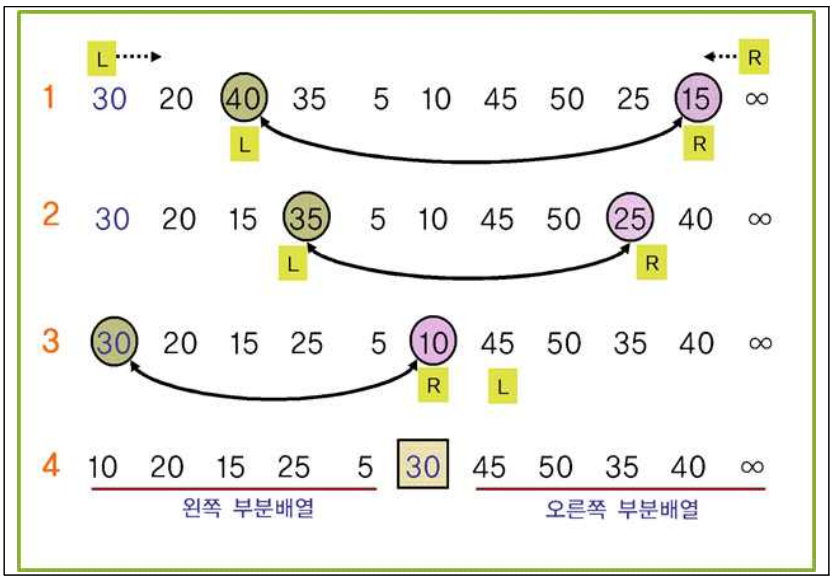


2. 퀵 정렬(quick sort)

퀵 정렬의 기본 개념은 기준이 되는 숫자(Pivot)를 정한 후에 그 숫자를 기준으로 작은 데이터와 큰 데이터를 나누고 다시 분리된 양쪽의 데이터에서 다시 기준이 되는 숫자를 정하고 그 숫자를 기준으로 작은 데이터와 큰 데이터를 나누는 작업을 반복하는 것이다.
이러한 작업을 위해서 기준이 되는 값보다 작은 값을 찾기 위한 인덱스와 큰 값을 찾기 위한 인덱스가 필요하다. 작은 값을 찾기 위한 인덱스는 왼쪽으로부터 진행하고 큰 값을 찾기 위한 인덱스는 오른쪽부터 진행한다.
왼쪽에서 진행하는 인덱스는 기준값보다 크거나 같은 경우에 멈추고 오른쪽에서 진행하는 인덱스는 기준값보다 작거나 같은 경우 멈춘다. 그 상태에서 해당 인덱스에 위치한 데이터의 위치를 서로 바꾼다. 이 작업을 반복하면 기준값을 기준으로 좌측에는 작거나 같은 값, 우측에는 크거나 같은 값이 위치하게 된다.
이렇게 정렬된 각각의 데이터 집합에서 다시 퀵정렬을 적용하여 정렬하고자 하는 데이터가 하나가 될 때까지 반복하면 전체 데이터가 정렬이 된다.

* 기본적인 동작의 세 단계
① 피벗(pivot)값을 정한다.
② 피벗값을 중심으로 파티션을 나눈다.
③ 나눈 파티션을 재귀적으로 1,2번을 반복한다.





1) 퀵 정렬의 예

<원본 데이터>

순서	10	6	14	2	11	9	4	3	7	5	13	12	1	8
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
j	14	13	12	11	10	9	8	7	6	5	4	3	2	1
data	46	39	97	16	51	44	32	29	40	37	76	69	12	42

먼저 이해를 돕기 위해 위의 그림을 설명하겠다. 가장 위의 행은 실제 데이터를 오름차순으로 정렬했을 때의 순서를 나타낸다. 이후 정렬 작업이 완료되면 순서는 1부터 14까지 오름차순으로 정렬될 것이다. 아래 행은 Pivot을 기준으로 작거나 같은 값을 찾기 위한 index 이며 그 아래 행은 Pivot을 기준으로 크거나 같은 값을 찾기 위한 index 이다. 그리고 마지막 행은 실제 데이터를 나타낸다.

<1단계>
먼저 기준값이 되는 Pivot을 정한다. 편의상 가운데 위치한 데이터 32를 Pivot으로 정한다. 이 값을 기준으로 왼쪽에서부터 값을 비교하여 크거나 같은 데이터를 검색한다. 맨 처음에 있는 46이 선택되었다. 다음으로 오른쪽에서부터 값을 비교하여 작거나 같은 값을 검색한다. j의 값이 2인 데이터 12가 선택되었다. 이제 두 데이터의 위치를 바꾼다.

순서	1	6	14	2	11	9	4	3	7	5	13	12	10	8
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
j	14	13	12	11	10	9	8	7	6	5	4	3	2	1
data	12	39	97	16	51	44	32	29	40	37	76	69	46	42

<2단계>
다시 왼쪽에서부터 크거나 같은 값을 검색한다. 이번에는 i=2 인 데이터가 검색되었다. 오른쪽에서부터 작거나 같은 값을 검색한다. j=7 인 데이터가 검색되었다. 다시 두 데이터의 위치를 바꾼다.

순서	1	3	14	2	11	9	4	6	7	5	13	12	10	8
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
j	14	13	12	11	10	9	8	7	6	5	4	3	2	1
data	12	29	97	16	51	44	32	39	40	37	76	69	46	42

<3단계>
왼쪽에서부터 크거나 같은 값을 검색한다. i=3 인 데이터가 검색되었다. 오른쪽에서부터 작거나 같은 값을 검색한다. j=8 인 데이터 자기 자신이 검색되었다. i=3 인 데이터와 자기 자신의 위치를 바꾼다.

순서	1	3	4	2	11	9	14	6	7	5	13	12	10	8
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
j	14	13	12	11	10	9	8	7	6	5	4	3	2	1
data	12	29	32	16	51	44	97	39	40	37	76	69	46	42

<4단계>
왼쪽에서 크거나 같은 값을 검색한다. i=5 인 데이터가 검색되었다. 오른쪽에서부터 작거나 같은 값을 검색한다. j=4 인 데이터가 검색되었다. 이와 같이 i 보다 j 가 작아지는 경우에는 j 가 있는 위치의 데이터와 Pivot의 위치를 바꾼다.

순서	1	3	2	4	11	9	14	6	7	5	13	12	10	8
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
j	14	13	12	11	10	9	8	7	6	5	4	3	2	1
data	12	29	16	32	51	44	97	39	40	37	76	69	46	42

<5단계>
여기까지의 과정을 거치면 최초에 Pivot으로 설정했던 데이터의 좌측에는 작은 값들이 위치하고 우측에는 큰 값들이 위치한다. 여기서 좌측의 데이터들을 하나의 subfile로 보고 다시 퀵정렬을 진행한다. 마찬가지로 우측의 데이터들을 하나의 subfile로 보고 다시 퀵정렬을 진행한다. 이렇게 재귀적인 호출로 계속 퀵정렬을 진행하면 다음과 같은 결과를 얻을 수 있다.

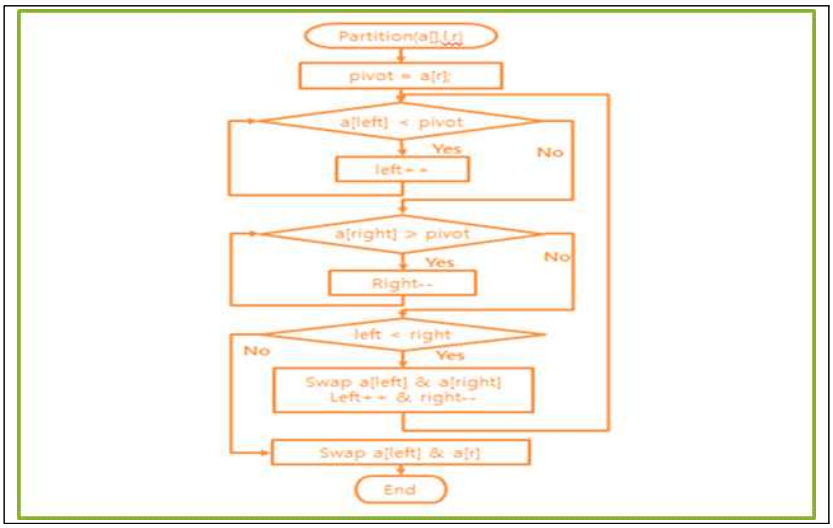
순서	1	2	3	4	5	6	7	8	9	10	11	12	13	14
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
j	14	13	12	11	10	9	8	7	6	5	4	3	2	1
data	12	16	29	32	37	39	40	42	44	46	51	69	76	97

3. 퀵 정렬(quick sort)의 요약

평균 실행시간이 O(NlogN)인 알고리즘으로서 실제로 평균 시간복잡도를 보일 가능성이 높은 방법이다.
퀵 정렬은 프로그래밍하기가 쉽고 자원을 적게 사용하고 범용이기 때문에 많이 사용되는 정렬 방법이다.

① 퀵 정렬은 정렬할 부분배열의 선두에 있는 원소가 제 위치를 잡도록 하여 정렬하는 방법이다.
② 분할정복방식의 전형적인 한 예의며 원래의 배열을 두 부분 배열로 분할하고 분할된 부분 문제를 순환적으로 정복한다.
③ 최선의 경우는 분할원소를 중심으로 두 부분 배열이 정확히 둘로 나뉘는 경우이다.
④ 키들의 상대적 위치가 유지되지 않는 불안정한 정렬 알고리즘이다.
⑤ 스택을 위한 여분의 공간이 필요하므로 제자리 정렬이 아니다.
⑥ 퀵 정렬 알고리즘의 시간복잡도는 최악 $O(n^2)$/평균 O(nlogn) 이다.
⑦ 퀵 정렬에서 분할원소의 역할은 분할원소는 주어진 배열을 두 개의 부분배열로 분할하는 데 사용된다.

* 퀵 정렬 순서도



학습내용2 : 힙 정렬

1. 힙 정렬이란?

1) 힙(heep)의 개념과 특징
힙 정렬을 알아보기 전에 먼저 heap 의 개념과 특징을 알아보겠다. 힙이란 여러 개의 노드들 가운데 가장 큰 키 값을 가지는 노드나 가장 작은 값을 가지는 노드를 빠른 시간 내에 찾아내도록 만들어진 자료구조를 말한다. heap은 완전 이진 트리의 하나로서 각각의 노드는 유일한 키 값을 가지며 우선 순위를 가지는 큐라는 의미에서 Priority Queue 라고도 한다. 이러한 힙은 크게 두가지로 나눌 수 있는데 하나는 max-heap 이며 또 하나는 min-heap 이다.
max-heap의 한 노드는 그 노드의 모든 자식들보다 큰 키 값을 가진다. 루트(root)에는 항상 큰 키 값을 가지는 노드가 위치하므로 우선순위 큐(priority queue)를 구성하는데 적합한 자료구조이다. min-heap은 max-heap과 반대로 한 노드는 그 노드의 모든 후손 노드들보다 작은 키 값을 가진다.

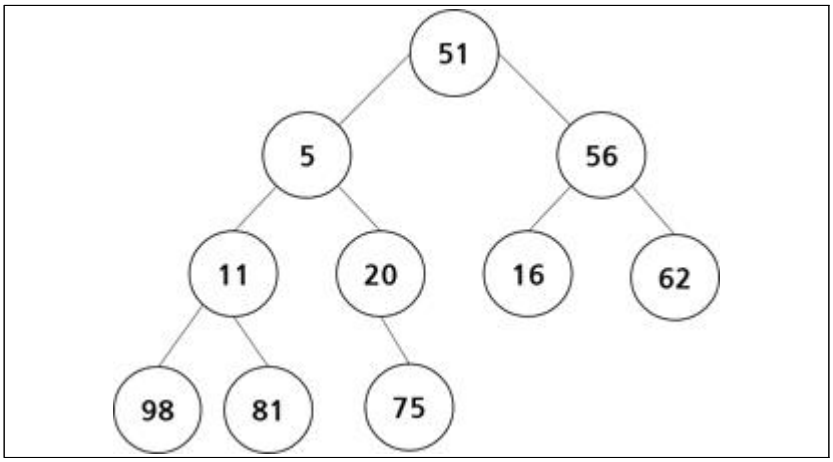
2) 힙 정렬의 과정
힙 정렬은 위에서 언급한 max-heap을 이용한다. 원본 데이터를 max-heap을 이용하여 정렬하면 최상단 노드(root)에 최대값이 위치하게 된다. 배열의 개념으로 보면 맨 앞에 위치하게 된다. 이렇게 추출된 최대값을 배열의 맨 마지막으로 이동시키고 나머지 자료를 가지고 다시 max-heap을 이용하여 정렬하면 또 다시 남은 데이터 중에서 최대값이 최상단 노드에 위치하고 배열의 개념으로 맨 앞에 위치하게 되면 앞서 최대값으로 지정된 값의 바로 앞에 위치시킨다. 이러한 방법을 반복하면 결과적으로 오름차순 정렬된 배열 데이터를 얻을 수 있다.
그림으로 이 과정을 알아보기로 한다.

3) 힙 정렬의 예

<원본 데이터 - 배열>

5	1	6	2	4	3	7	10	9	8
51	5	56	11	20	16	62	98	81	75

<원본 데이터 - 트리구조>



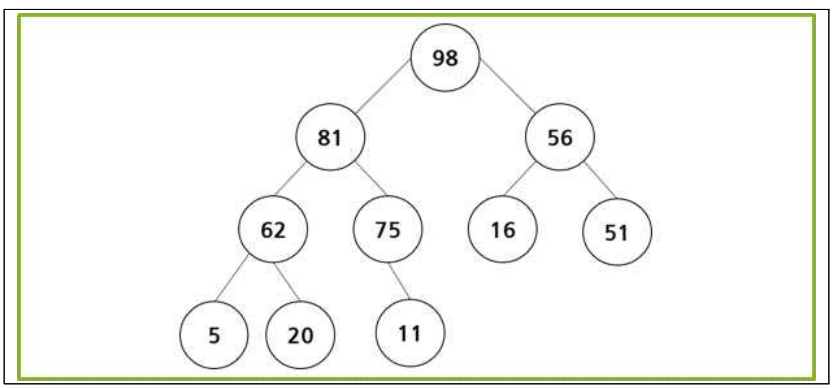
<1단계 - 배열>

max-heap 의 구조로 데이터를 정렬한다. 이 과정을 거치면 배열의 첫 번째에 가장 큰 값이 위치하게 된다.

10	9	6	7	8	3	5	1	4	2
98	81	56	62	75	16	51	5	20	11

<1단계 - 트리>

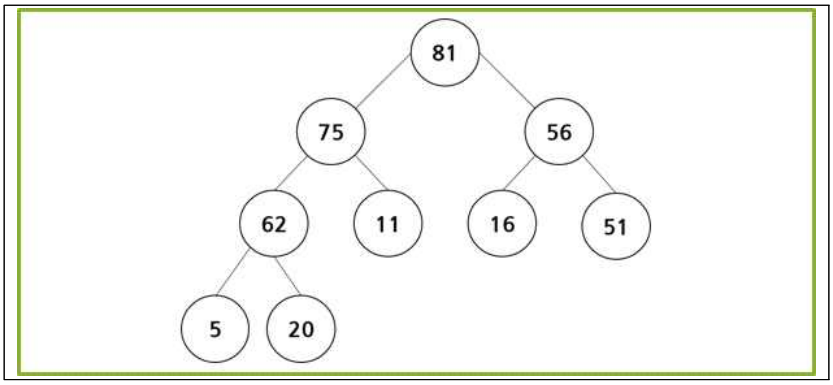
max-heap 의 구조로 데이터를 정렬한다. 트리구조의 경우 root에 가장 큰 값이 위치하게 된다.



〈2단계 - 배열〉
찾아낸 최대값을 배열의 맨 마지막으로 이동시킨다. 그리고 나머지 배열을 다시 max-heap 구조의 트리로 정렬한다.

9	8	6	7	2	3	5	1	4	10
81	75	56	62	11	16	51	5	20	98

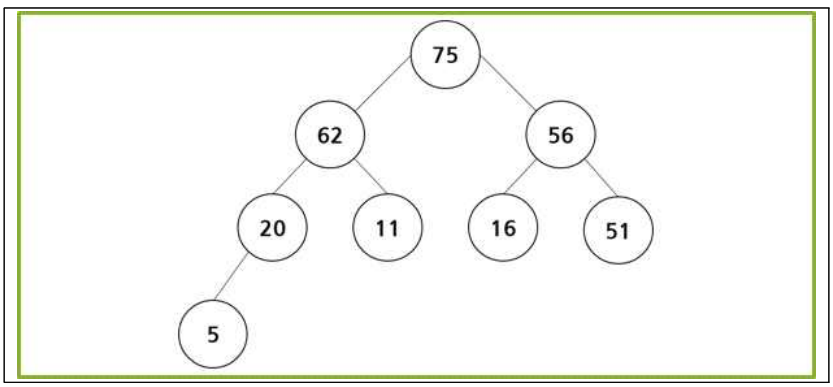
〈2단계 - 트리〉
최대값을 제외하고 나머지 배열로 다시 max-heap 트리를 만들면 다음과 같다. 이번에는 81이 최대값이 되어 트리의 root로 이동하게 된다.



〈3단계 - 배열〉
마찬가지로 찾아낸 최대값을 배열의 마지막으로 이동시킨다. 그리고 나머지 배열을 다시 max-heap 구조의 트리로 정렬한다.

8	7	6	4	2	3	5	1	9	10
75	62	56	20	11	16	51	5	81	98

〈3단계 - 트리〉
최대값을 제외하고 나머지 배열로 다시 max-heap 트리를 만들면 다음과 같다. 이번에는 75가 최대값이 되어 트리의 root로 이동하게 된다.



위의 단계를 계속 반복하면 최종적으로 다음과 같은 결과를 얻을 수 있다.

1	2	3	4	5	6	7	8	9	10
5	11	16	20	51	56	62	75	81	98

2. 힙 정렬(heap sort)의 요약

힙 정렬은 힙 자료구조의 장점인 최대값을 삭제하거나 임의의 값을 삽입하는 연산을 수월하게 하는 점을 이용한 정렬이다.

① 힙은 완전이진나무(complete binary tree)로서 나무의 각 노드의 값은 그 노드가 자식노드를 갖는 경우에 자식의 값보다 크거나 같아야 한다.
② 우선순위 큐를 구현(배열을 써서 구현할 수 있다.)
③ 나무의 뿌리에 최대값이 오도록 한 힙을 최대값 힙이라 한다.
④ 제자리 정렬이나 안정적이지가 않다.
⑤ 우선순위 큐란 새 원소의 삽입과 최대 원소의 삭제가 가능한 자료구조
⑥ 힙정렬을 두 단계
　　단계1 : 힙의 구축
　　단계2 : 힙에서 최대치를 제거해 나감에 의해 정렬
⑦ 초기 힙의 구축 방법은 상향식 힙 구축/하향식 힙 구축
⑧ 힙이 구축된 후 정렬 방법은 뿌리와 제일 끝 쪽의 노드를 서로 교환하고 힙 조건이 유지되도록 힙을 재조정하는 과정을 반복한다.
⑨ 힙정렬 알고리즘의 시간복잡도는 평균/최악 모두 O(nlogn)이다.

【학습정리】

1. 분할 정복

분할정복 방법의 세 단계는 원래의 문제를 소문제로 분할하는 분할 단계, 소문제의 해를 순환적으로 구하는 정복단계, 소문제의 해를 합하여 원래 문제에 대한 해를 만들어 내는 합병단계로 구성된다.

2, 퀵 정렬(quick sort)

평균 실행시간이 O(NlogN)인 알고리즘으로서 실제로 평균 시간복잡도를 보일 가능성이 높은 방법이다. 퀵 정렬은 프로그래밍하기가 쉽고 자원을 적게 사용하고 범용이기 때문에 많이 사용되는 정렬 방법이다.

3. 히프정렬(heap sort)

- 히프정렬은 히프 자료구조의 장점인 최대값을 삭제하거나 임의의 값을 삽입하는 연산을 수월하게 하는 점을 이용한 정렬이다.

① 히프는 완전이진나무(complete binary tree)로서 나무의 각 노드의 값은 그 노드가 자식노드를 갖는 경우에 자식의 값보다 크거나 같아야 한다.
② 우선순위 큐를 구현(배열을 써서 구현할 수 있다.)
③ 나무의 뿌리에 최대값이 오도록 한 히프를 최대값 히프라 한다.
④ 제자리 정렬이나 안정적이지가 않다.
⑤ 우선순위 큐란 새 원소의 삽입과 최대 원소의 삭제가 가능한 자료구조
⑥ 히프정렬을 두 단계
- 단계1 : 히프의 구축
- 단계2 : 히프에서 최대치를 제거해 나감에 의해 정렬
⑦ 초기 히프의 구축 방법은 상향식 히프 구축/하향식 히프 구축
⑧ 히프가 구축된 후 정렬 방법은 뿌리와 제일 끝 쪽의 노드를 서로 교환하고 히프 조건이 유지되도록 히프를 재조정하는 과정을 반복한다.
⑨ 히프정렬 알고리즘의 시간복잡도는 평균/최악 모두 O(nlogn)이다.