힙(heap)

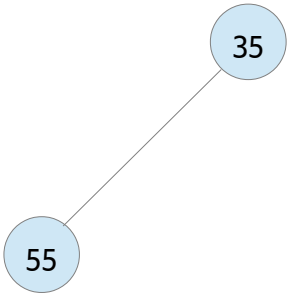
sianke1991

우선순위 큐(priority queue): 입력한 값을 제거할 때, 우선순위가 높은 순서대로 제거되는 자료형

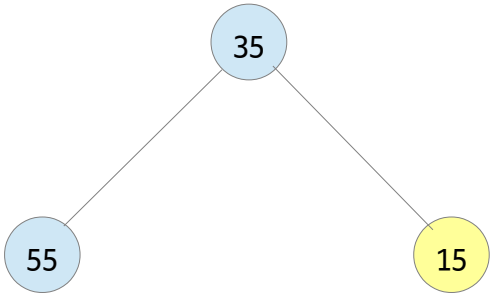
힙(heap): 이진 트리의 일종으로 부모 노드의 값이 두 자식 노드의 값 이하를 만족하는 것(최소 힙) 또는 부모 노드의 값이 두 자식 노드의 값 이상을 만족하는 것(최대 힙). 부모 노드의 값이 두 자식 노드의 값 사이에 들어가는 이진 탐색 트리와 달리 힙에서는 부모 노드의 값이 두 자식 노드의 값의 최소치 이하이거나 최대치 이상이 된다. 힙에서 값을 채워나갈 때는 깊이가 낮은 노드부터, 같은 깊이라면 왼쪽 노드부터 채워나간다. (따라서 이진 탐색 트리와 달리 불균형이 발생하지 않는다.)
힙에서 자료 추가, 자료 제거는 log N의 시간복잡도, 자료 조회는 1의 시간복잡도를 갖는다.

힙의 자료 추가 (최소 힙을 기준으로 함)

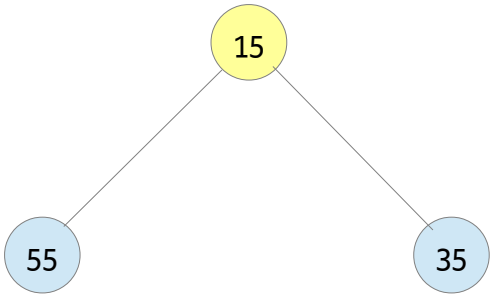
처음 입력된 자료는 루트 노드가 된다.



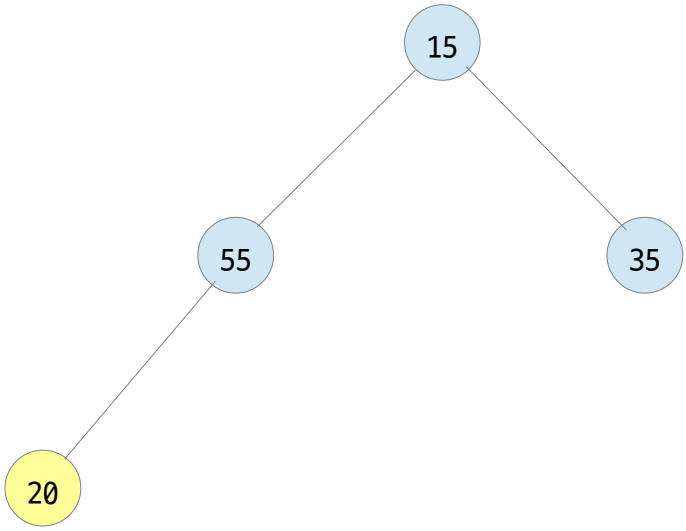
이후에 입력된 자료는 일단 이진 트리의 리프로 추가한다. 리프로 추가할 때는 깊이가 낮은 노드부터, 같은 깊이라면 왼쪽 노드부터 채워나간다. 그 이후에 [추가된 노드]와 부모 노드를 비교하여 [추가된 노드]가 부모 노드보다 작으면 [추가된 노드]와 부모 노드를 서로 뒤바꾼다. (이 작업은 [추가된 노드]가 부모 노드의 이상이 되거나, [추가된 노드]가 루트가 될 때까지 반복한다.) 이 경우 [추가된 노드] 55는 부모 노드 35보다 크므로 두 노드는 서로 뒤바꾸지 않는다.



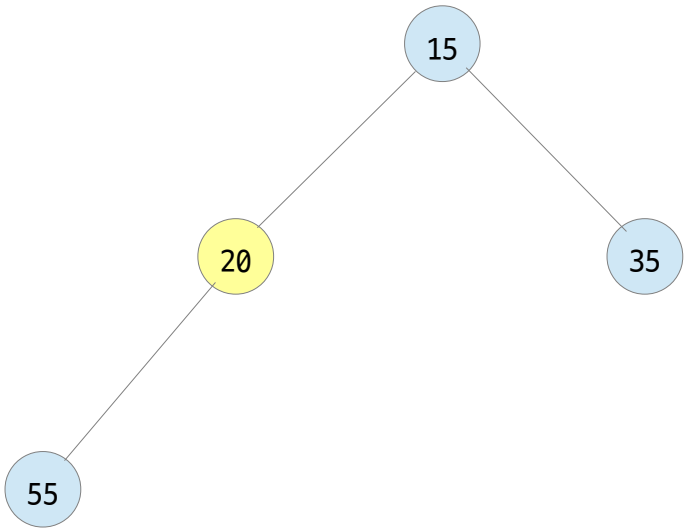
이후에 추가된 자료는 우선은 이진 트리의 리프로 추가한다. 이번에 [추가된 노드] 15는 부모 노드 35보다 작으므로 [추가된 노드]와 부모 노드는 서로 뒤바뀌어야 한다.



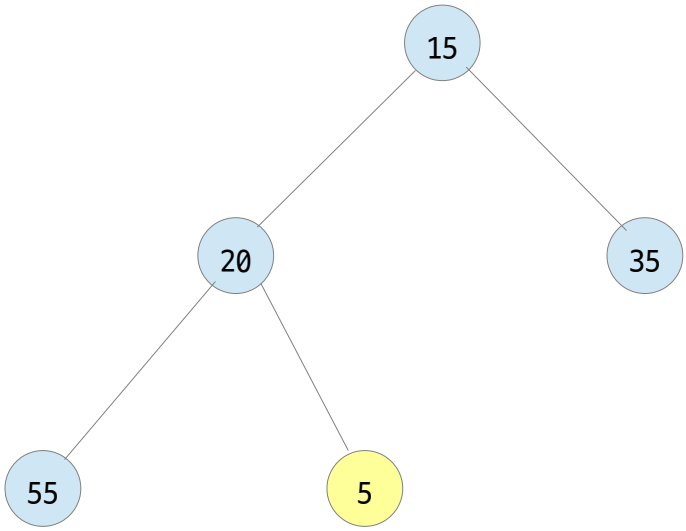
뒤바꿈 작업 이후 [추가된 노드] 15는 루트가 되었으므로 더 이상의 뒤바꿈 작업은 이루어지지 않는다.



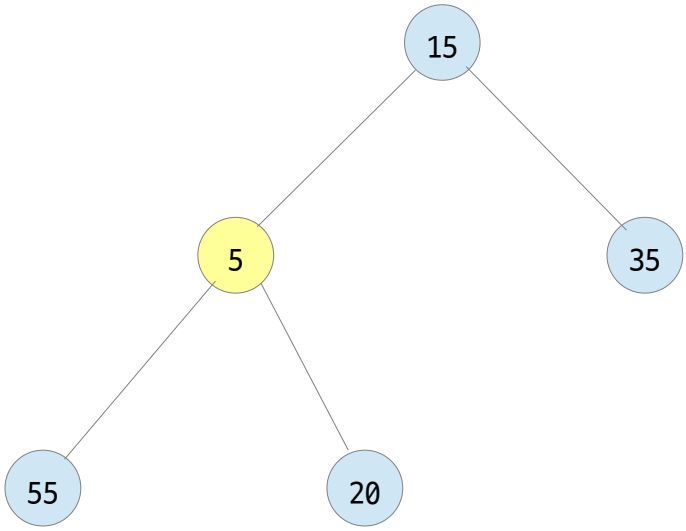
이번에 [추가된 노드] 20은 55 노드의 왼쪽 자식이 되는데, 이 [추가된 노드] 역시 부모 노드보다 작다. 따라서 뒤바꿈 작업을 수행해야 한다.



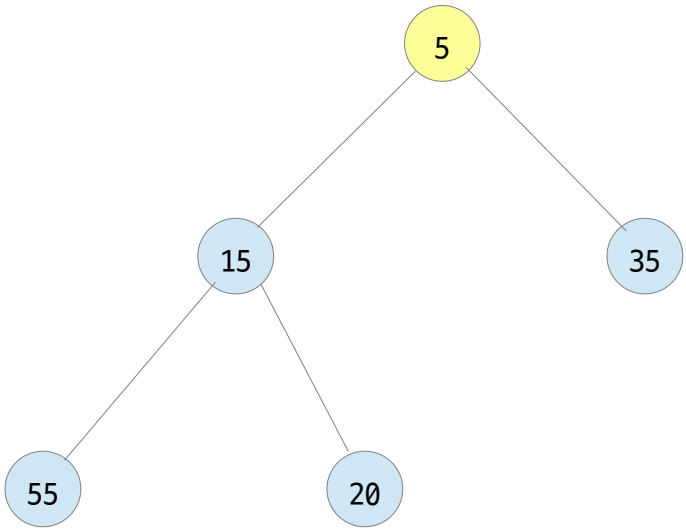
뒤바꿈 작업 이후 [추가된 노드] 20은 부모 노드 15 보다 크므로 더 이상의 뒤바꿈 작업은 이루어지지 않는다. (설령 [추가된 노드]가 루트가 되지 않았더라도 이 [추가된 노드]는 제 자리를 찾은 것이다.)



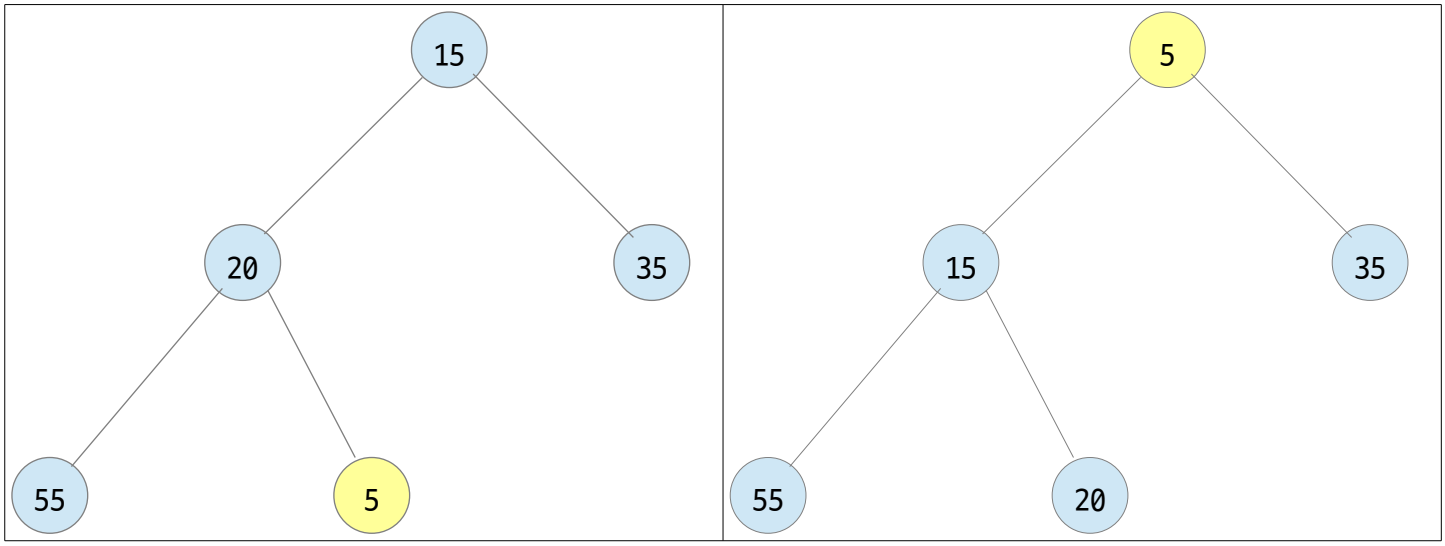
이번에는 5를 힙에 추가해 본다. 처음 위치는 노드 20의 오른쪽 자식이다. 이 [추가된 노드]는 부모 노드 20 보다 작으므로 뒤바꿈 작업을 수행해야 한다.



뒤바꿈 작업 이후 [추가된 노드] 5는 부모 노드 15 보다 작으므로 추가적인 뒤바꿈 작업이 필요하다.

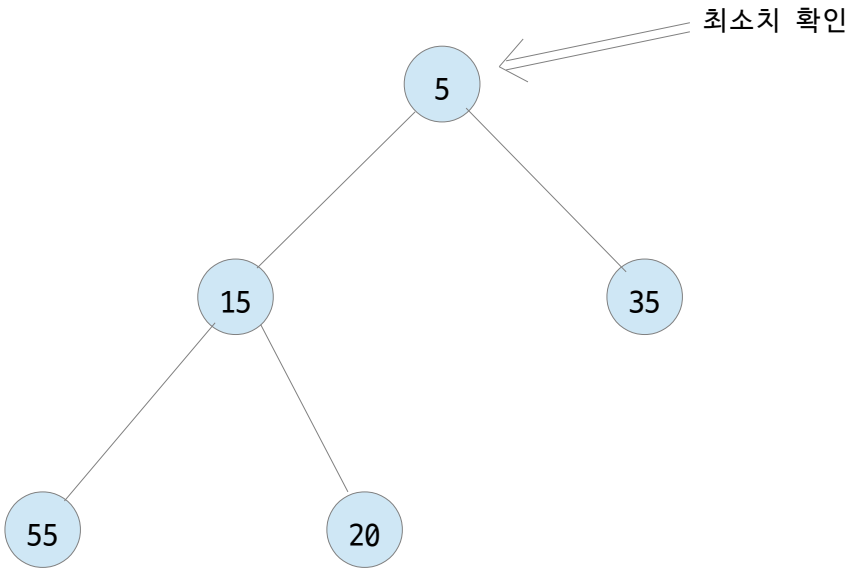


뒤바꿈 작업 이후 [추가된 노드] 5는 루트가 되었으므로 더 이상의 뒤바꿈 작업은 이루어지지 않는다.



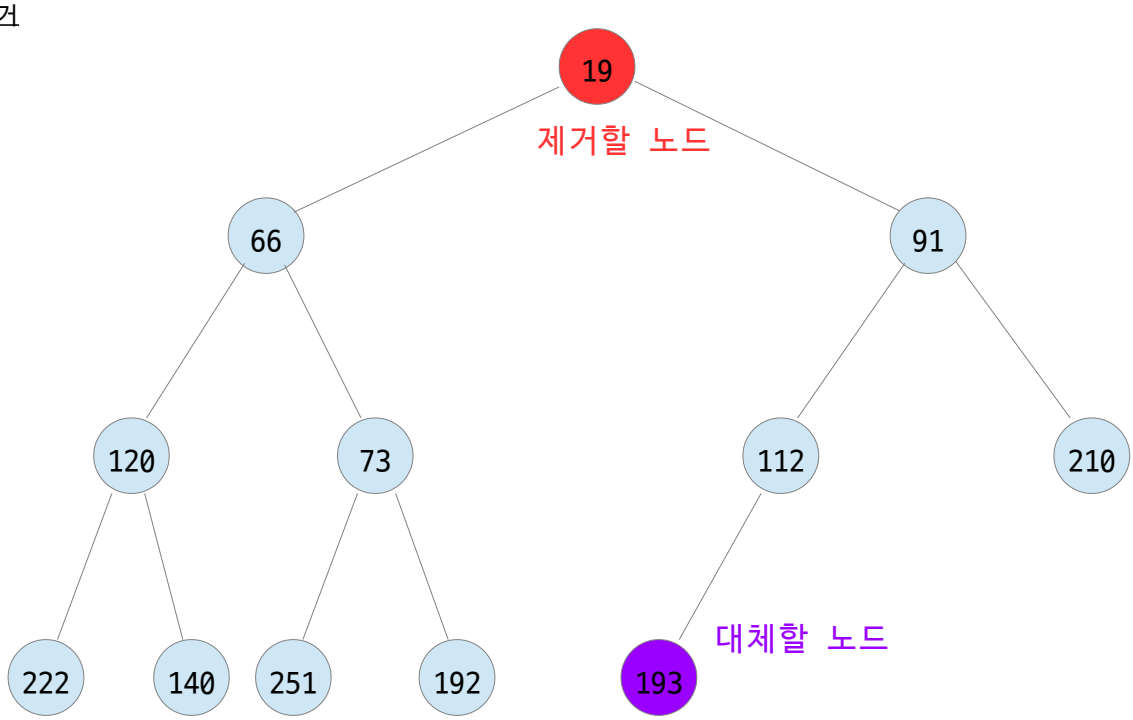
최소 힙에 최소치가 추가되는 경우, [추가된 노드]는 리프에서 출발해서 루트까지 올라가게 된다. 즉, 트리의 높이만큼 뒤바꿈 작업이 이루어진다. 트리의 높이는 log N에 비례하므로 힙의 자료 추가는 log N의 시간복잡도를 갖는다.

힙의 자료 조회

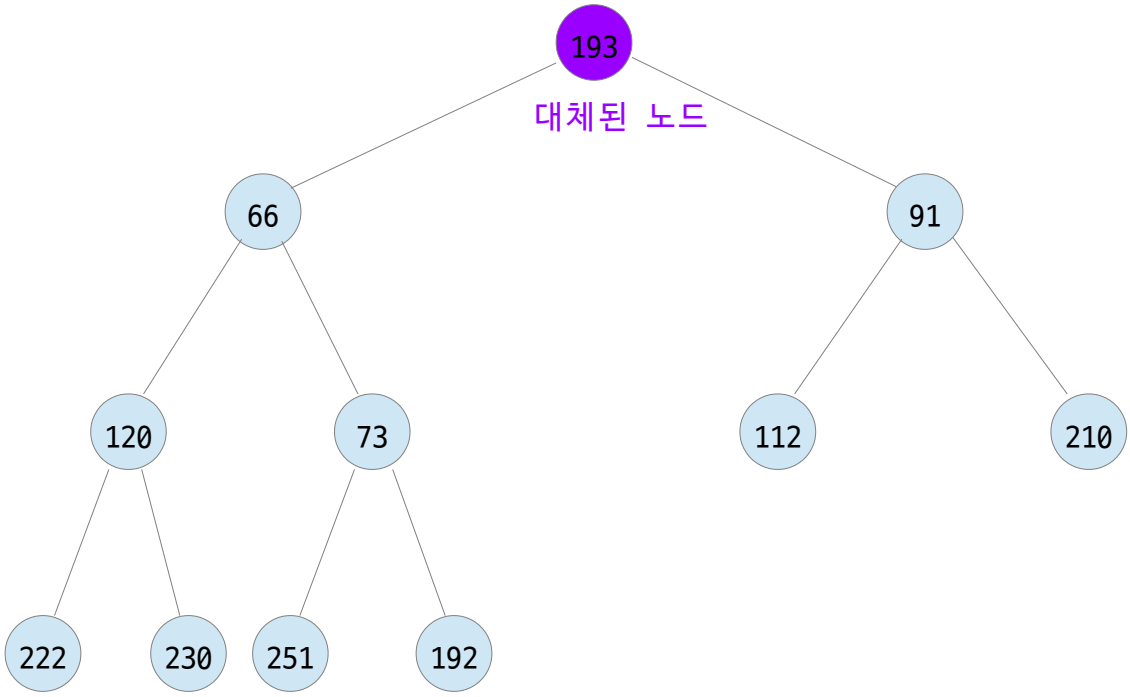


최소 힙의 경우 최소치는 루트 노드에 있다. 따라서 최소치 조회는 상수 시간에 수행 가능하다. 하지만 몇 번째로 작은 원소 등의 조회는 힙에 있는 전체 노드를 조회해야 수행 가능하다.

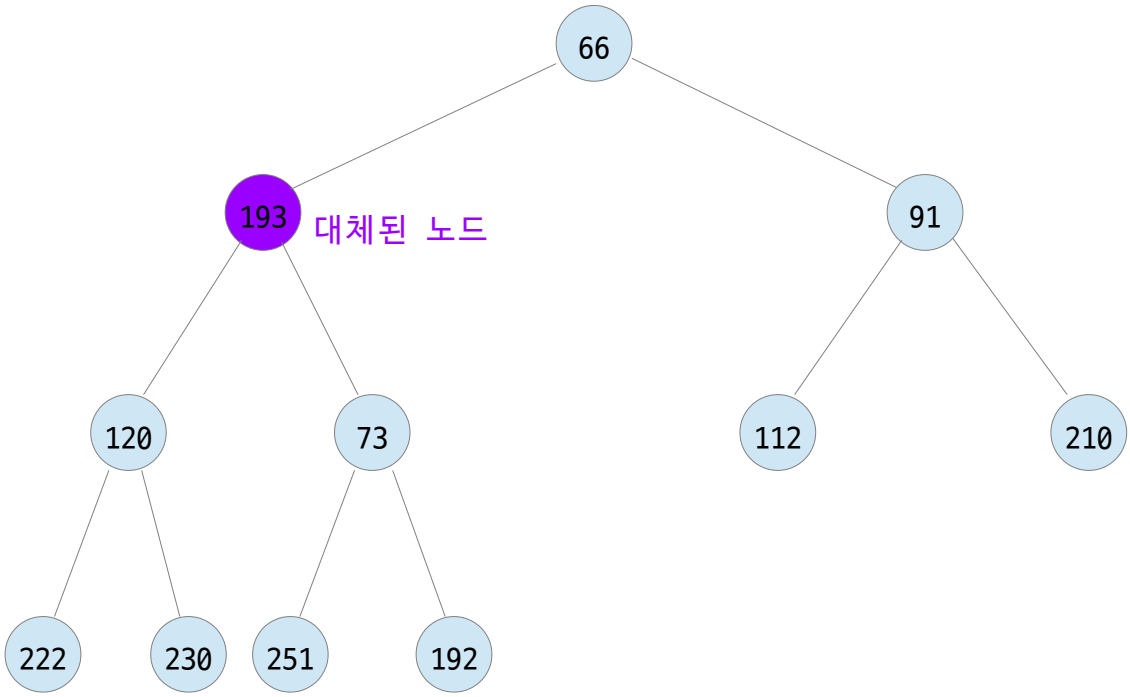
힙의 자료 제거



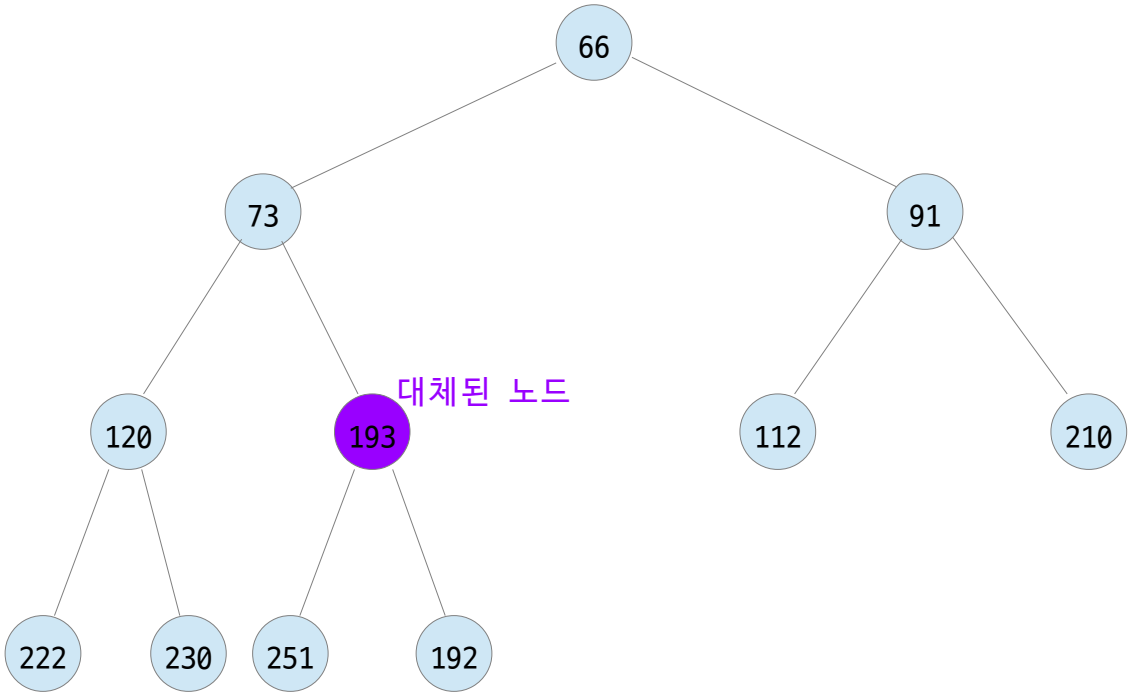
(최소)힙에서 최소치를 제거하면 우선 [대체할 노드]가 제거할 노드(루트 노드)를 대체한다. [대체할 노드]는 리프 노드 중 가장 오른쪽에 있는 노드가 된다. (리프 노드 중 가장 오른쪽에 있는 노드를 [대체한 노드]로 선택해야 힙의 특징인, '같은 깊이라면 왼쪽 노드부터 채워나간다'가 깨지지 않는다.)



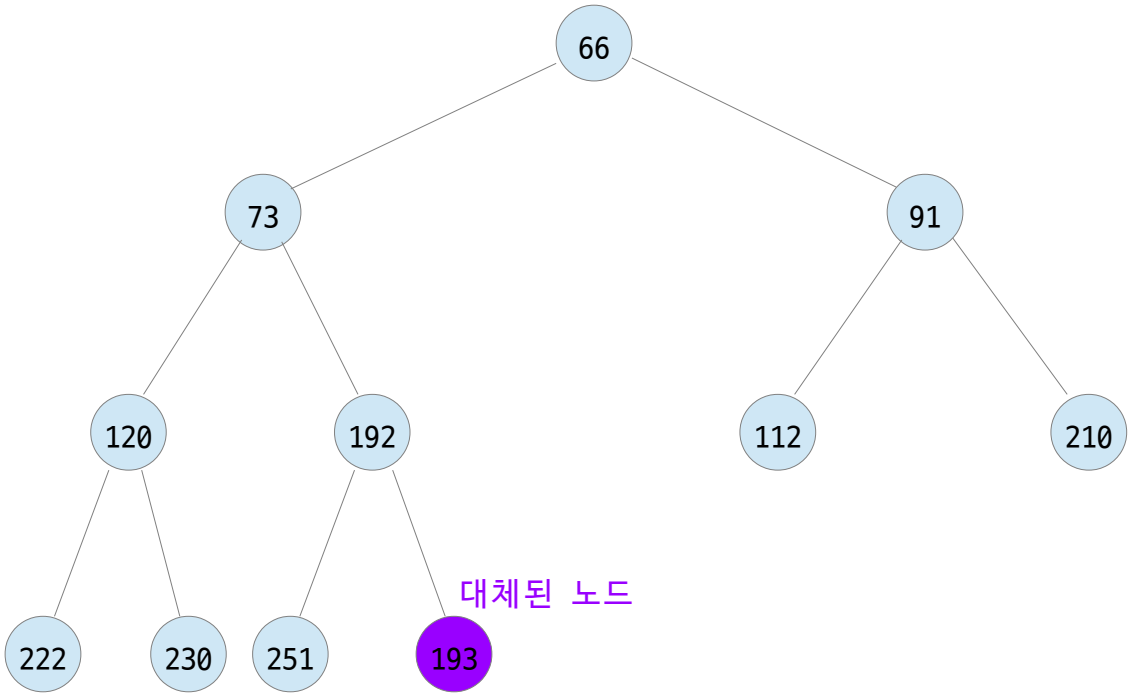
대체 작업 이후 [대체된 노드]가 자식 노드 보다 크면 [대체된 노드]와 [자식 노드 중 최소치 노드]를 서로 뒤바꾼다. (이 작업은 [대체된 노드]가 자식 노드 이하가 되거나, [대체된 노드]가 리프가 될 때까지 반복한다.) 이 경우 [대체된 노드] 193은 자식 노드 66, 91 보다 크므로 [대체된 노드]를 자식 노드 66과 뒤바꾼다.



뒤바뀜 작업 이후 [대체된 노드]는 자식 노드 120, 73보다 크다. 따라서 [대체된 노드] 193을 자식 노드 73과 뒤바꾼다.

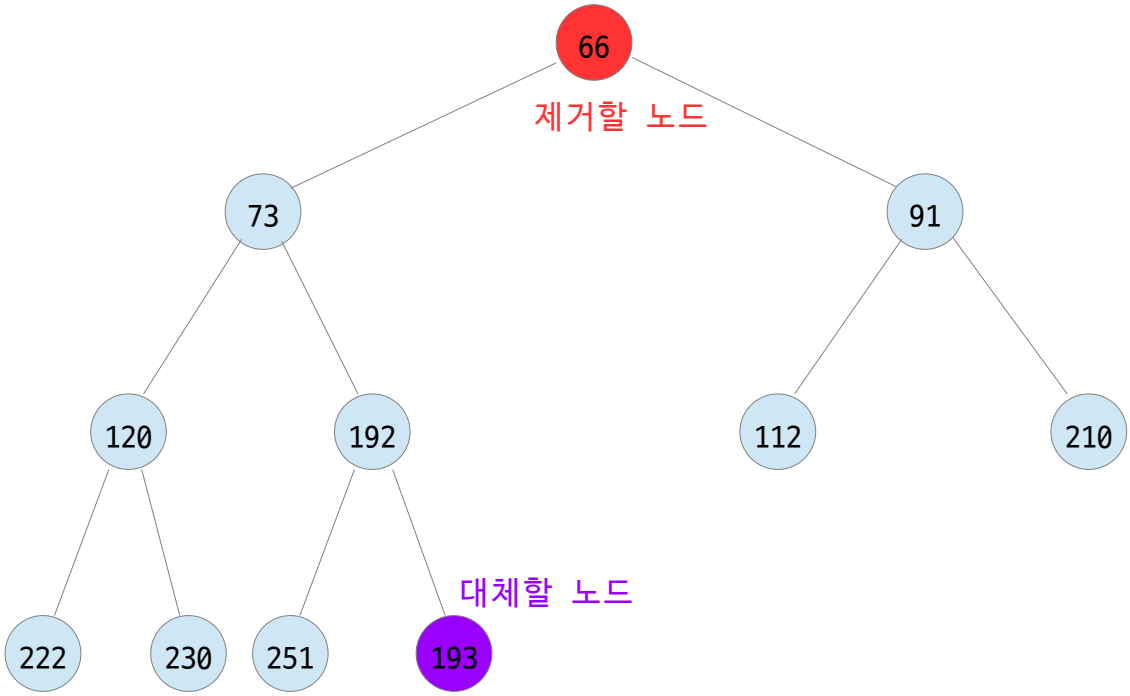


뒤바꿈 작업 이후 [대체된 노드]는 자식 노드 251, 192 사이의 값이 되었다. 최소힙의 경우 부모 노드는 자식 노드들 이하의 값을 가져야 하므로 이 경우에도 뒤바꿈 작업이 이루어져야 한다. 이 경우 [대체된 노드] 193을 자식 노드 192와 뒤바꾼다.

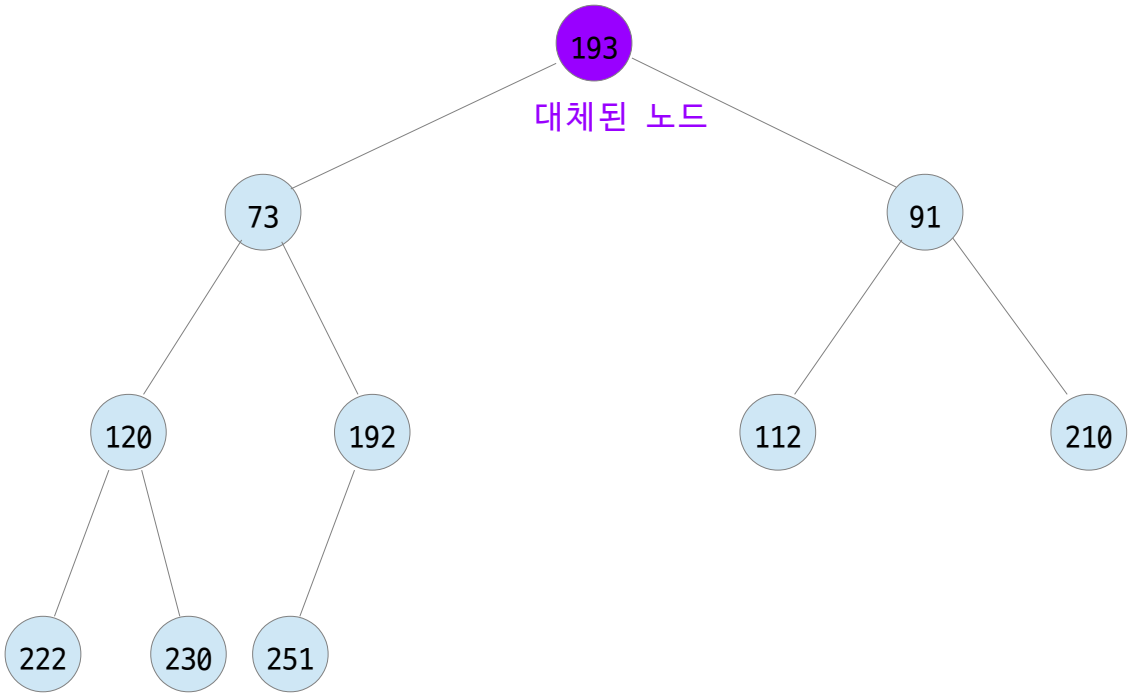


뒤바꿈 작업 이후 [대체된 노드]는 리프가 되었으므로 더 이상의 뒤바꿈 작업은 이루어지지 않는다.

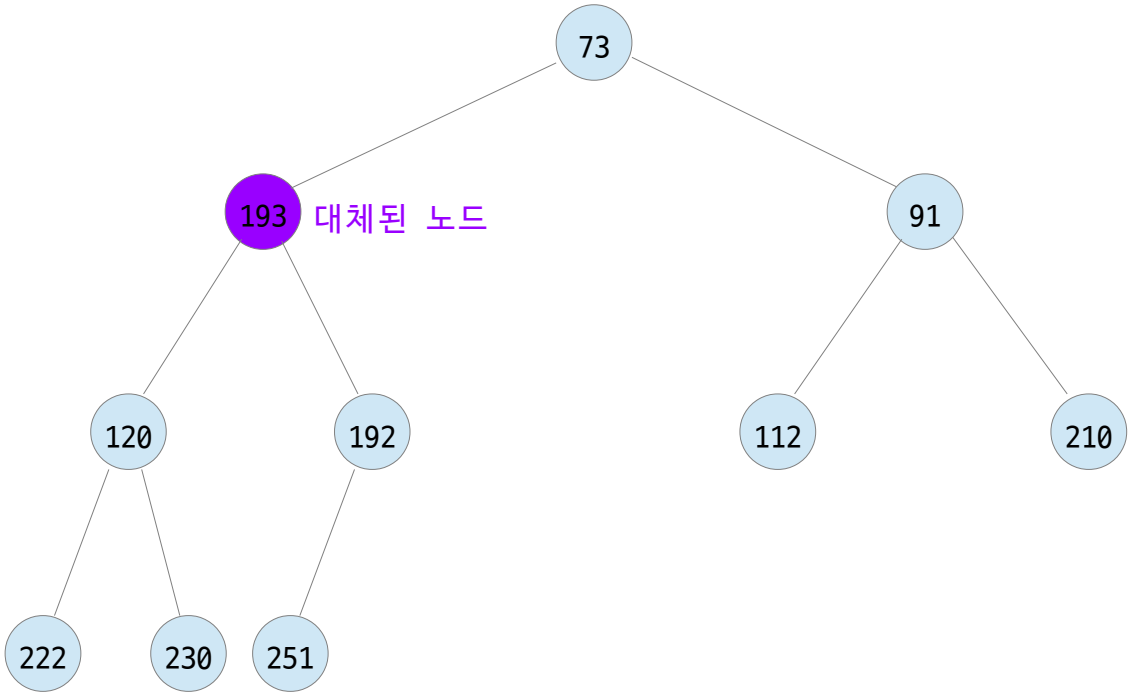
이 예시의 경우 [대체된 노드] 193은 루트까지 올라간 다음 다시 리프까지 내려왔다. 즉, 트리의 높이만큼 뒤바꿈 작업이 이루어진다. 트리의 높이는 log N에 비례하므로 힙의 자료 제거는 log N의 시간복잡도를 갖는다.



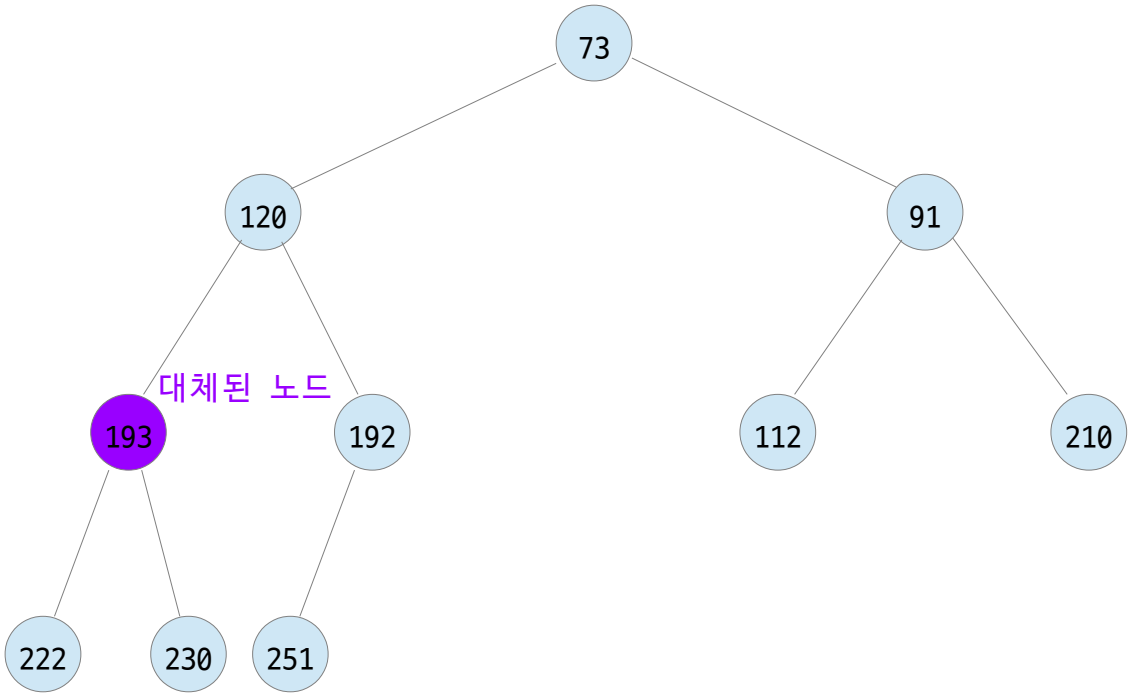
다시 한 번 힙에서 값을 제거해 본다. (최소)힙에서 최소치를 제거하면 우선 [대체할 노드]가 제거할 노드(루트 노드)를 대체한다. 이 예시에서도 [대체할 노드]는 193이 된다.



대체 작업 이후 [대체된 노드]가 자식 노드 보다 크면 [대체된 노드]와 [자식 노드 중 최소치 노드]를 서로 뒤바꾼다. (이 작업은 [대체된 노드]가 자식 노드 이하가 되거나, [대체된 노드]가 리프가 될 때까지 반복한다.) 이 경우 [대체된 노드] 193은 자식 노드 73, 91 보다 크므로 [대체된 노드]를 자식 노드 73과 뒤바꾼다.



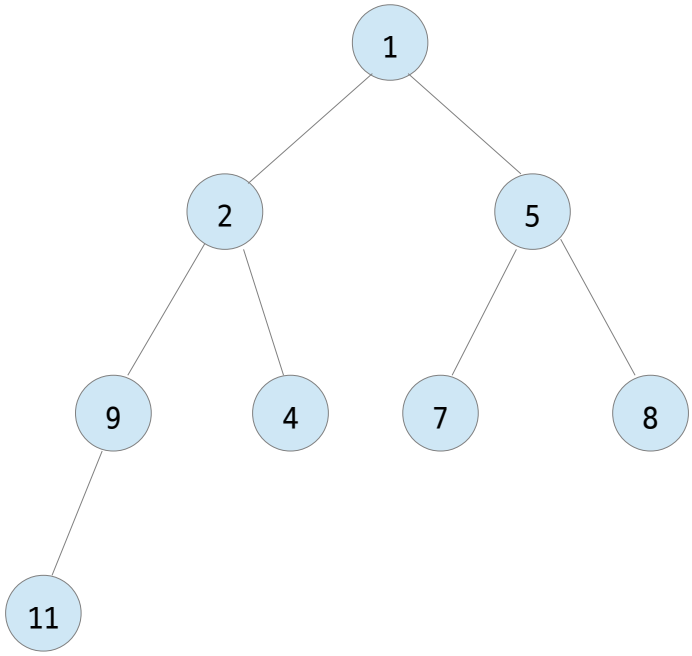
뒤바뀜 작업 이후 [대체된 노드]는 자식 노드 120, 192보다 크다. 따라서 [대체된 노드] 193을 자식 노드 120과 뒤바꾼다.



뒤바꿈 작업 이후 [대체된 노드]는 자식 노드 222, 230보다 작다. 따라서 더 이상의 뒤바꿈 작업은 이루어지지 않는다. (설령 [대체된 노드]가 리프가 되지 않았더라도 이 [대체된 노드]는 제 자리를 찾은 것이다.)

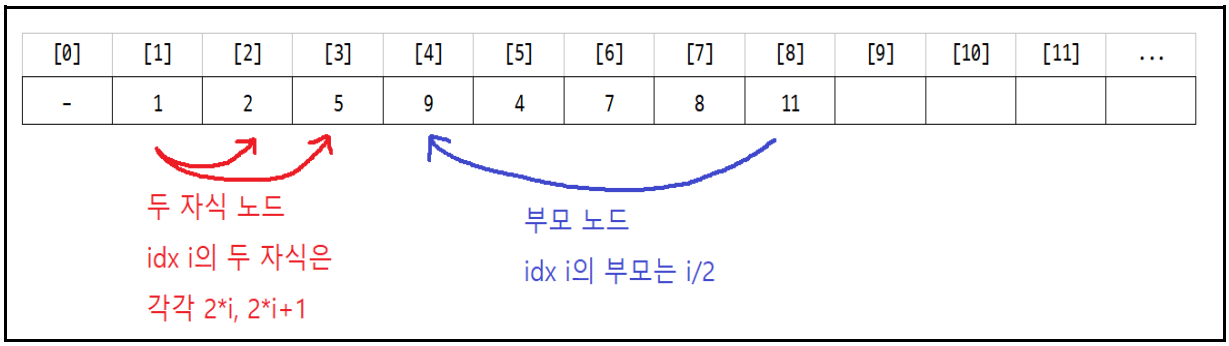
배열을 사용한 힙의 구현

배열을 사용하여 힙을 구현할 때 다음과 같은 방법으로 힙의 각 노드를 배열의 각 원소에 대응시킨다.



[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	...
-	1	2	5	9	4	7	8	11				

루트 노드를 인덱스 1에, 그 자식 노드를 인덱스 2, 3에, 그 자식 노드를 인덱스 4 – 7에 대응시킨다.



특정 노드의 부모, 자식은 해당 노드의 인덱스에 2를 곱하거나 나눠서 구할 수 있다.

```
void push(int x);
int top();
void pop();
```

등의 함수를 구현해야 한다.