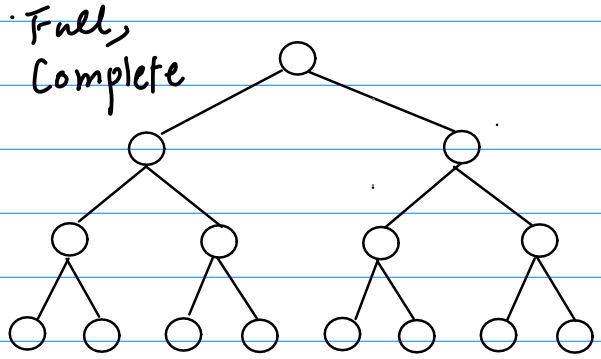
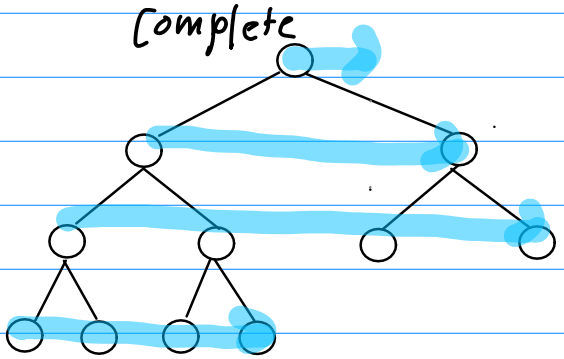
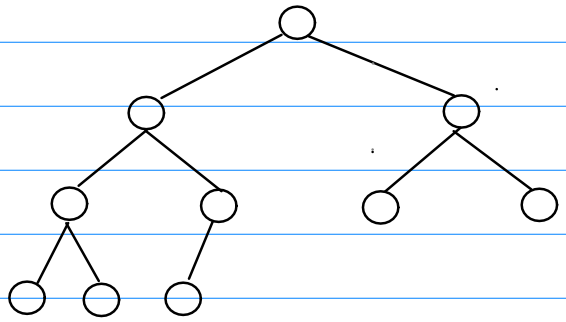
Heap (H1)

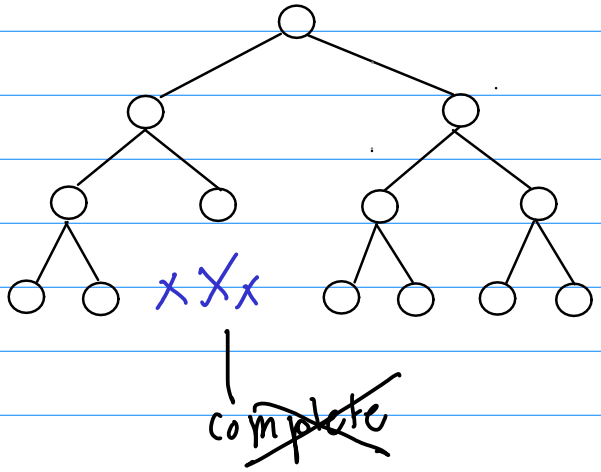
Heap

* Full Binary Tree



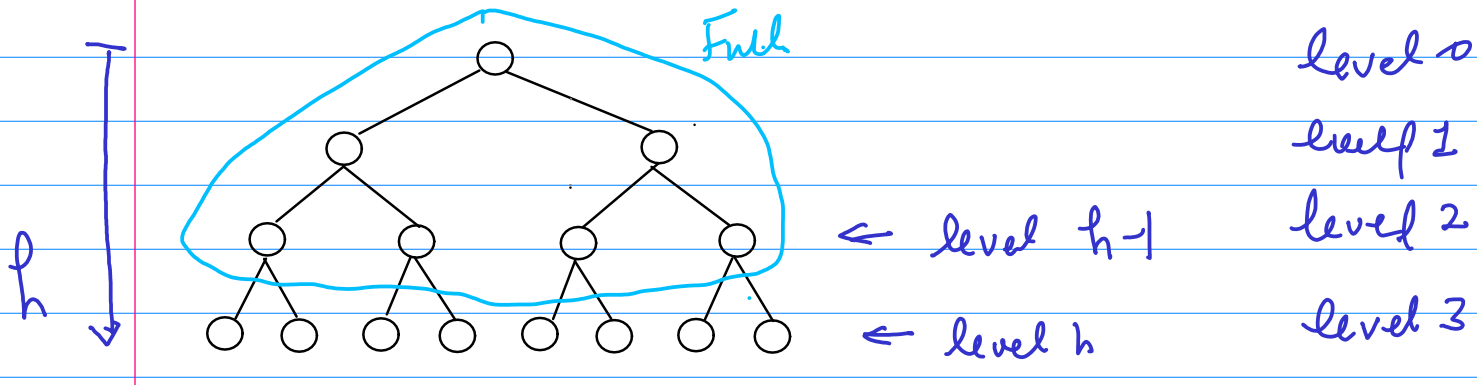




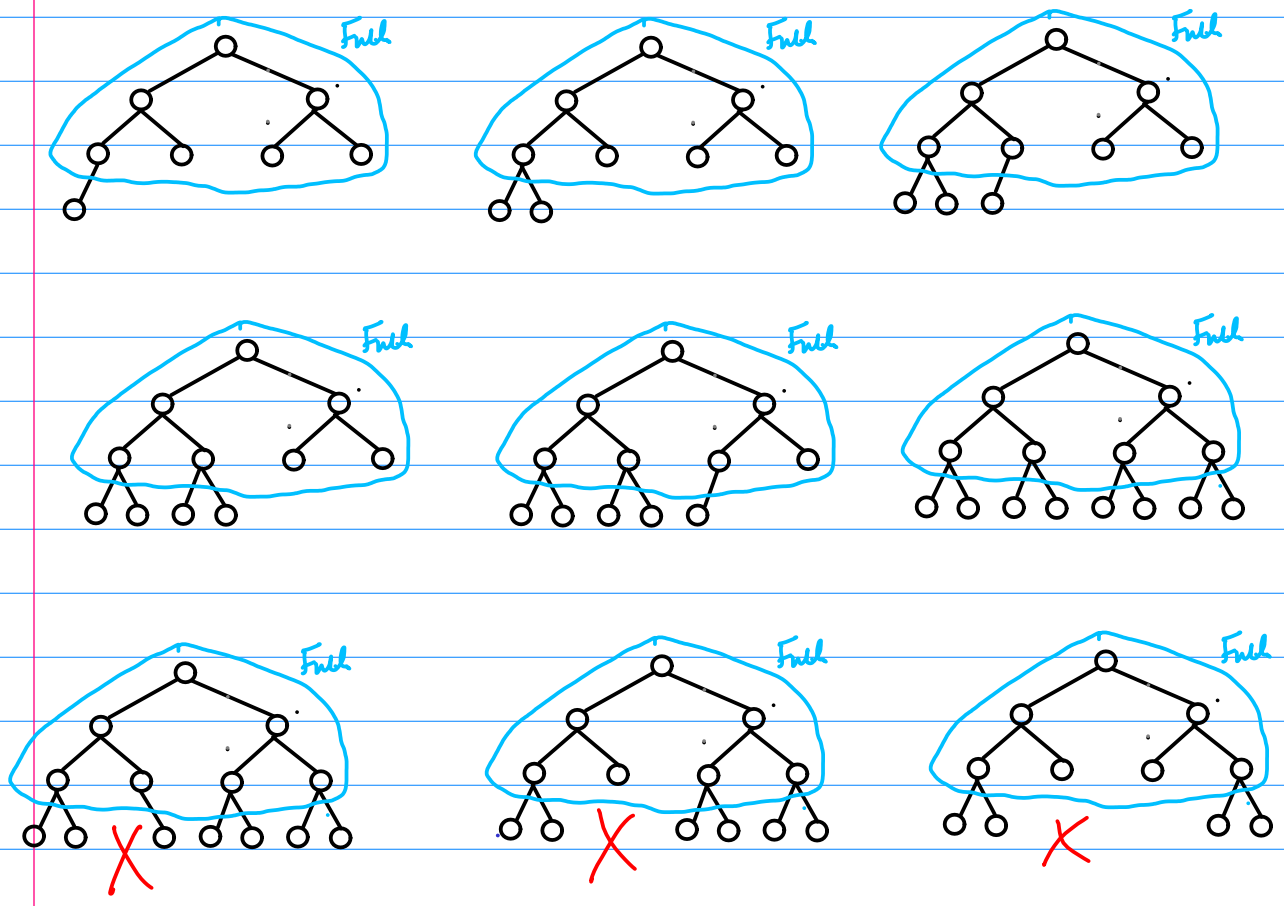


Complete Binary Tree

level (h-1) 까지는 Full
level h 에서는 왼쪽 → 오른쪽 으로 가면서 leaf 를 채운것
하나라도 빠지면 complete X



complete binary tree

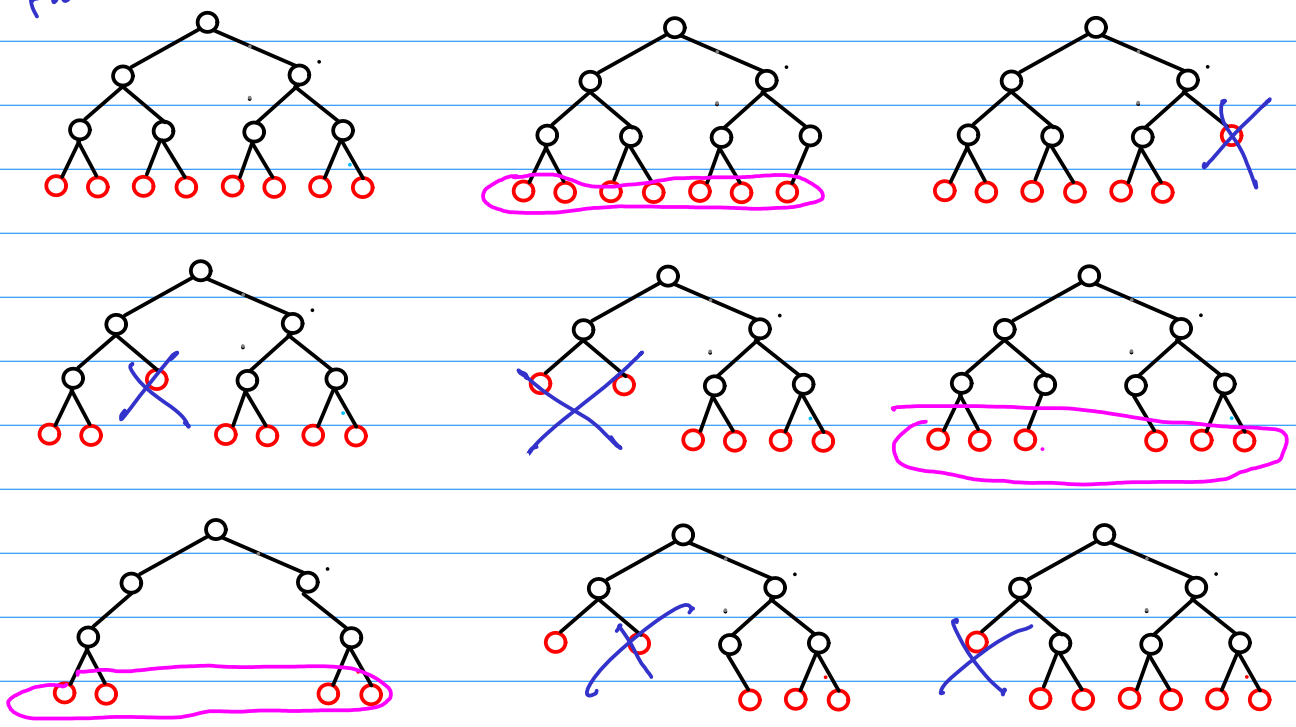


Full Binary Tree

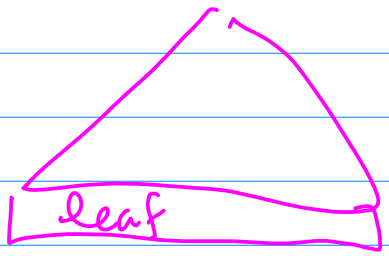
? 높이가 h인 binary tree에서 모든 leaf node가 level h에 있는 tree이다.

이것과 차이점 leaf node 위쪽의 모든 node는 반드시 자식 node를 두개 가지고 있어야함

Full

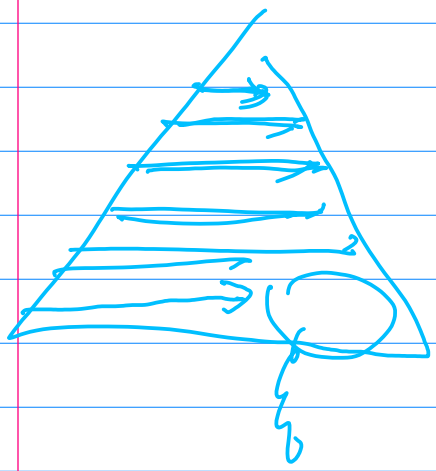


Two restricted forms of binary tree are sufficiently important to warrant special names. Each node in a **full** binary tree is either (1) an internal node with exactly two non-empty children or (2) a leaf. A **complete** binary tree has a restricted shape obtained by starting at the root and filling the tree by levels from left to right. In the complete binary tree of height d, all levels except possibly level $d-1$ are completely full. The bottom level has its nodes filled in from the left side.

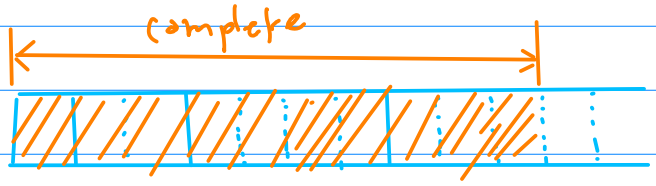


full binary tree

모든 internal node 들이 2개의 children 을 가짐.



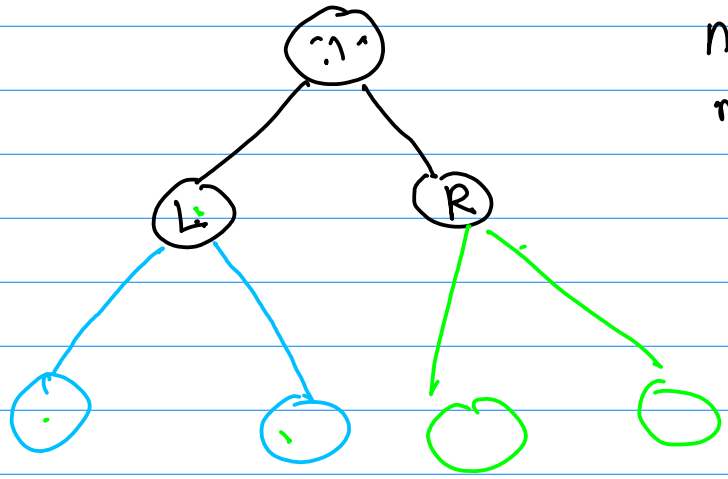
complete



Heap.

① Complete Binary Tree.

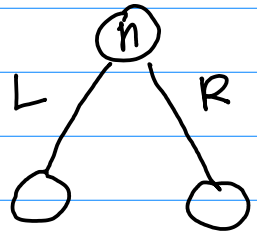
② root의 key ≥ 왼쪽 과식
root의 key ≥ 오른쪽 과식
→ 크기 순서는 상관없다

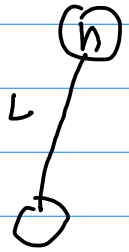


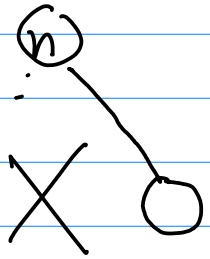
n.key ≥ L.key
n.key ≥ R.key

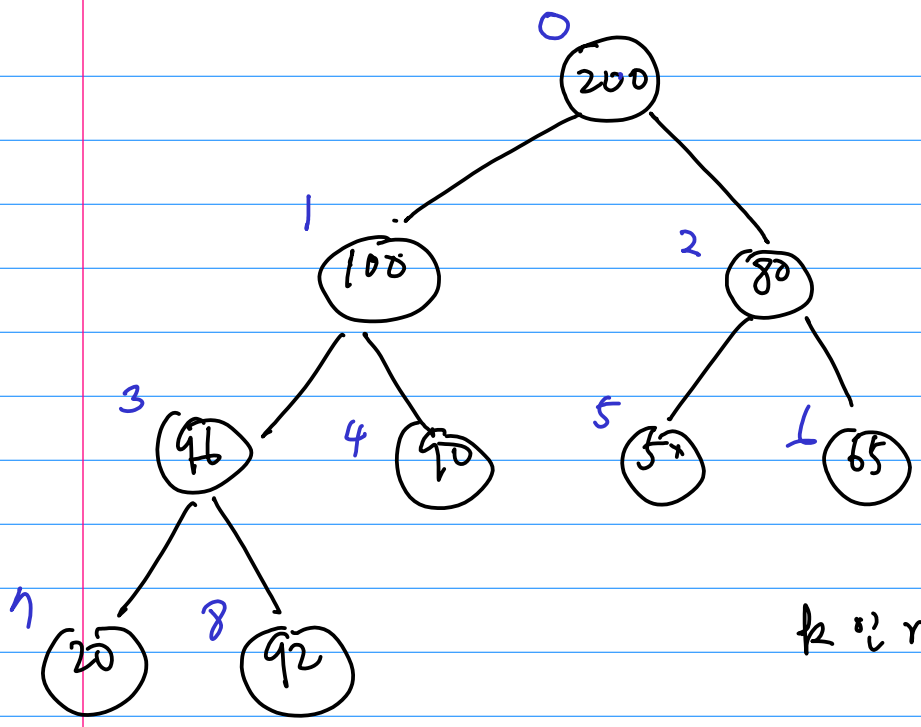
L.key ≷ R.key

Complete BTree





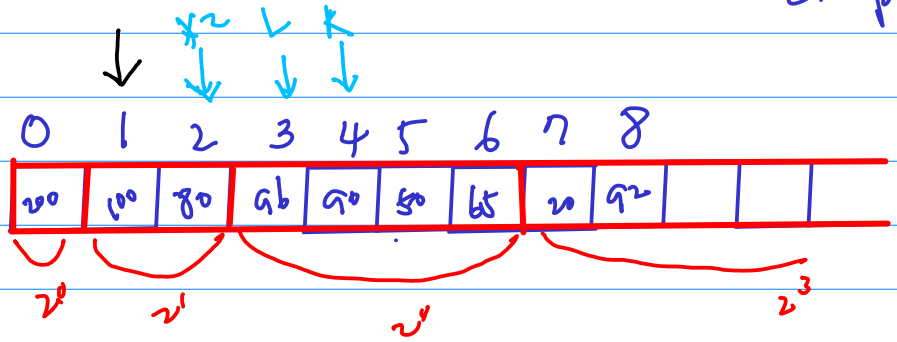


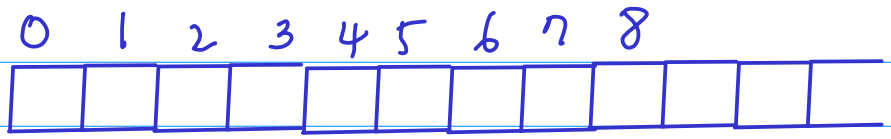


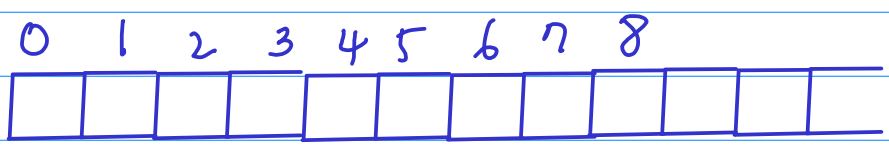
complete binary
→ 마지막 1
개념

k 번째 node의 L → (2k+1)
R → (2k+2)

k 번째 node의 parent (k-1)/2

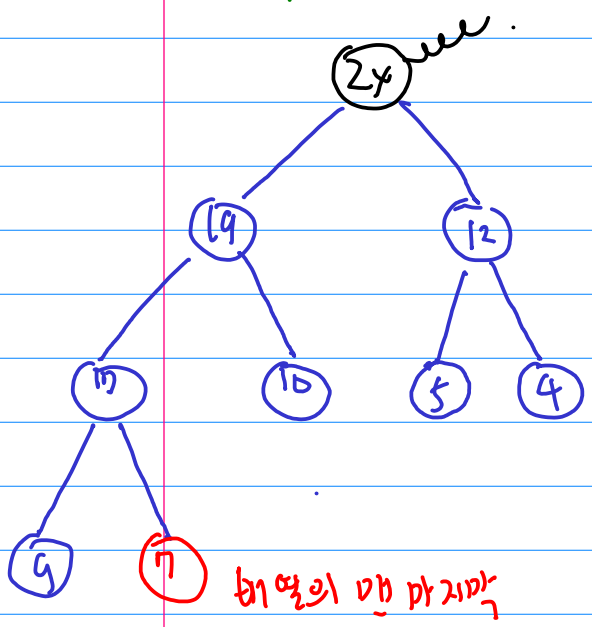


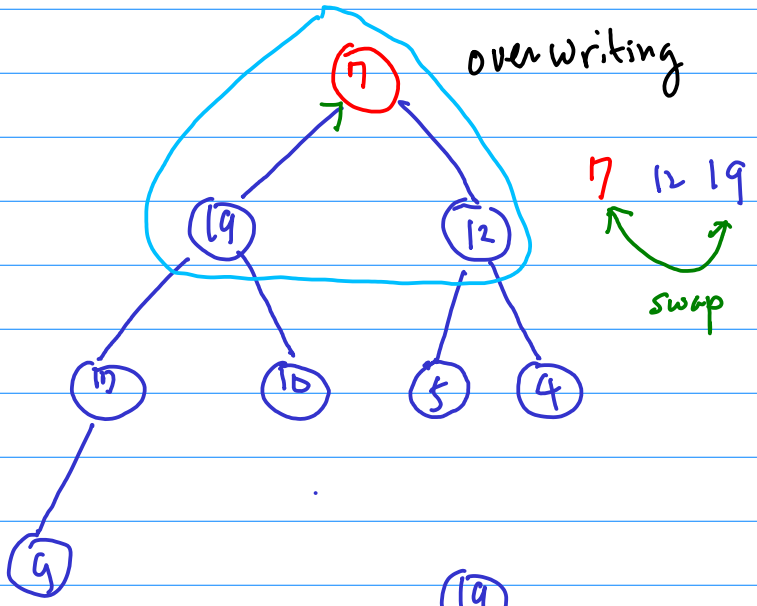




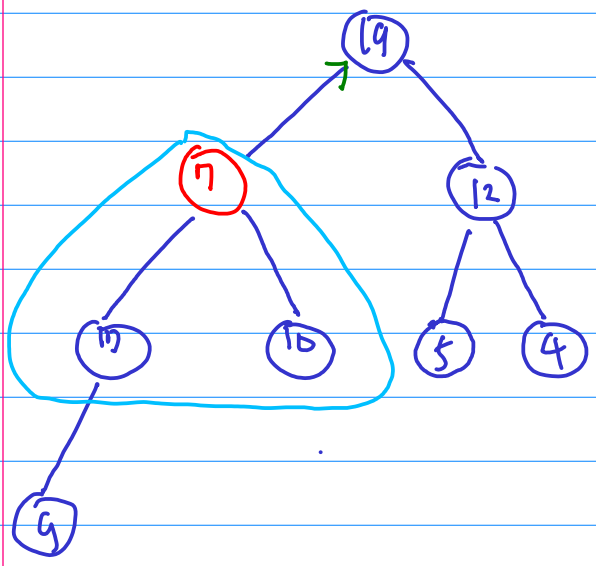
Heap ··· 우선순위가 큰 node가 위로.

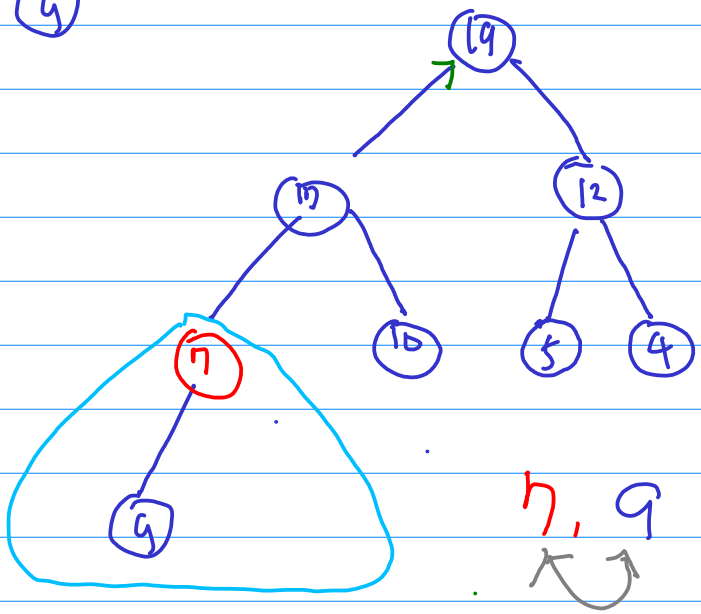
root node가 삭제되어야 함

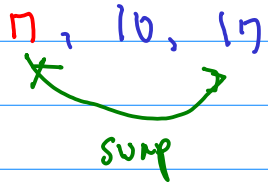




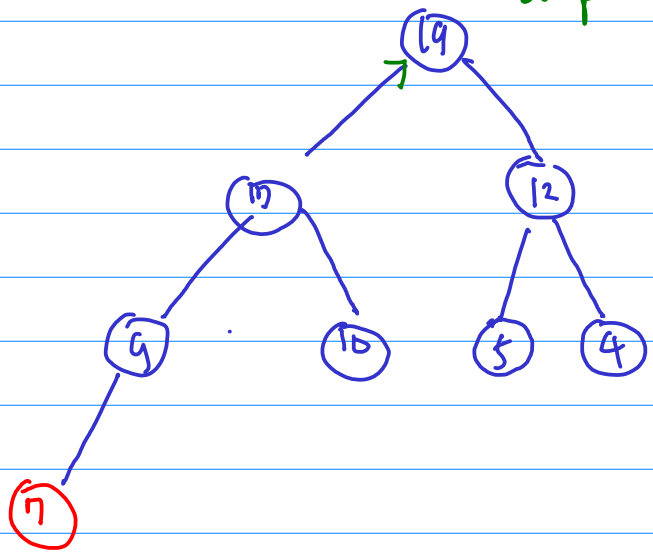
제일 마지막 node





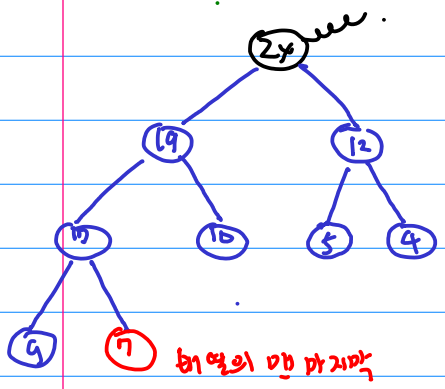


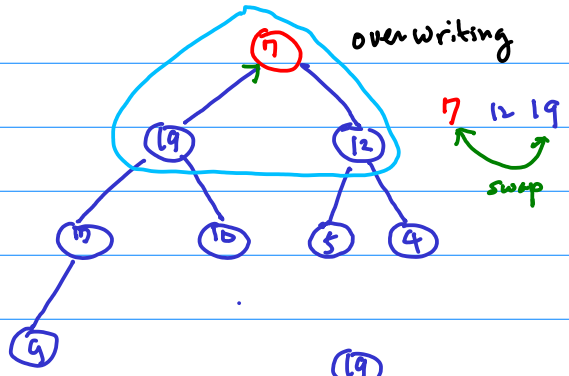
Heap Rebuilding

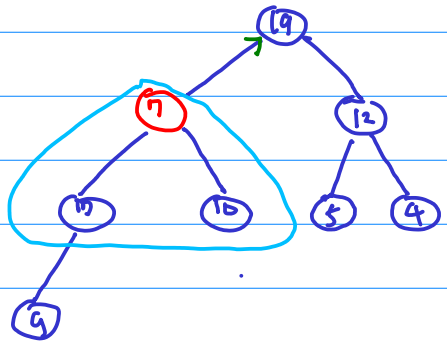


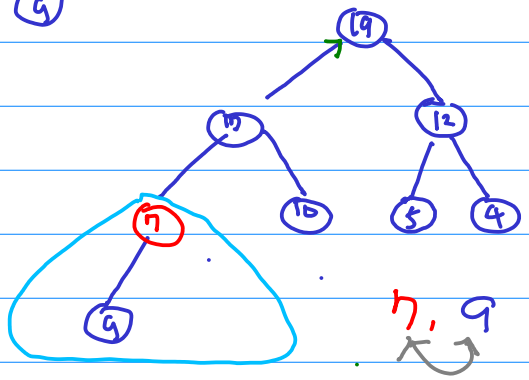
Heap … 부모노드보다 큰 node가 위로.

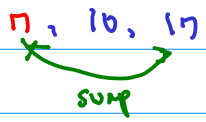
root node가 삭제되어야 함.



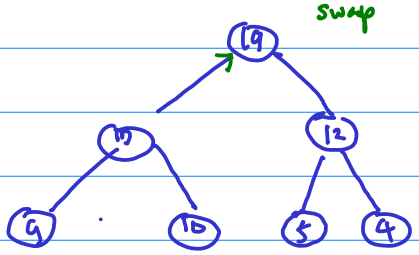


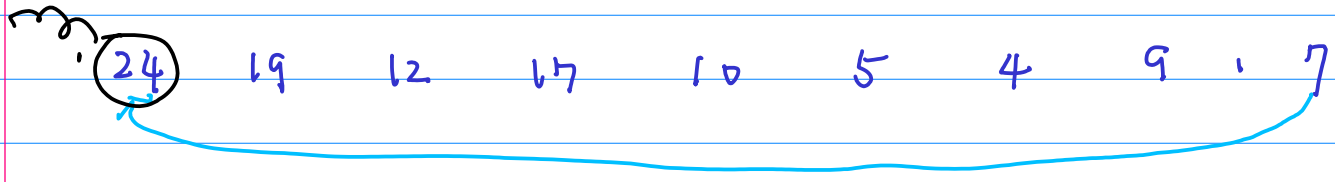


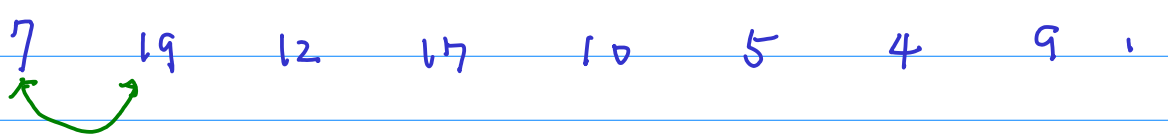


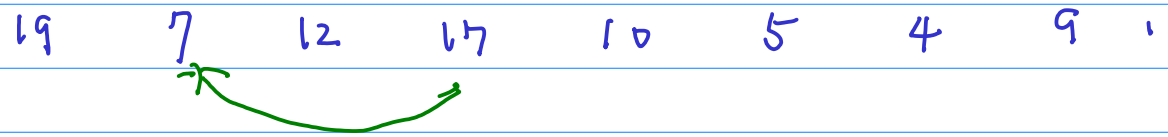


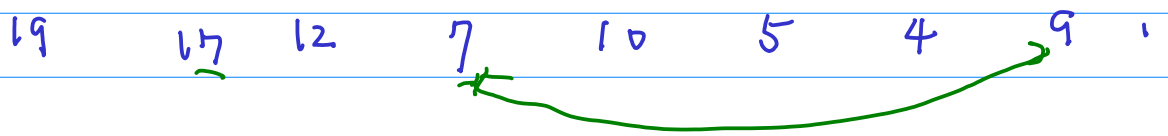
Heap Rebuilding

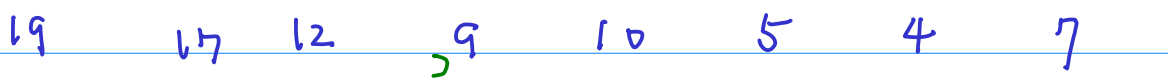


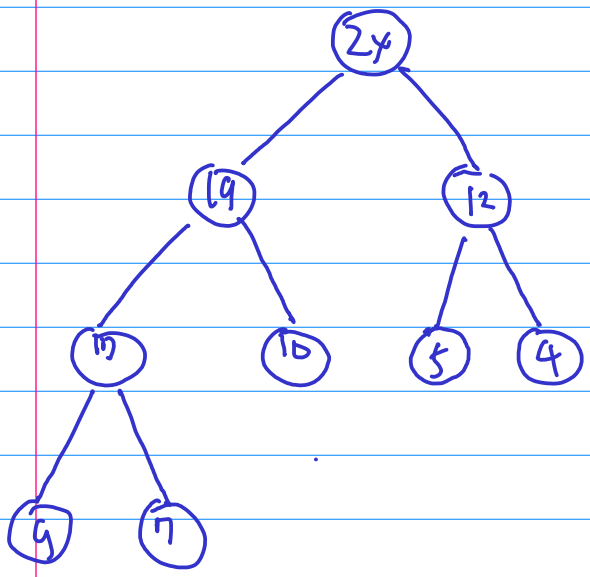


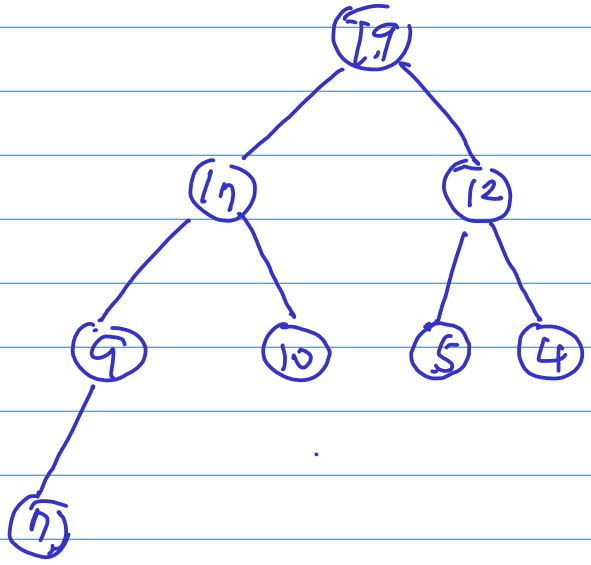












19 17 12 9 10 5 4 7