

Heap (H1)

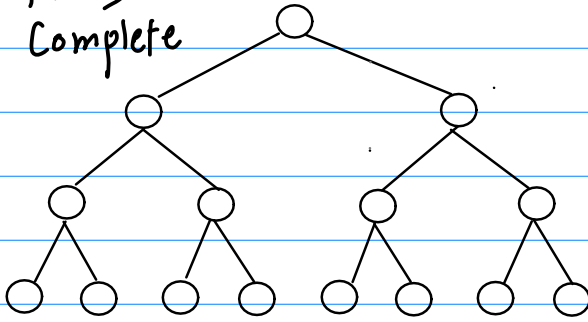
Copyright (c) 2015 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

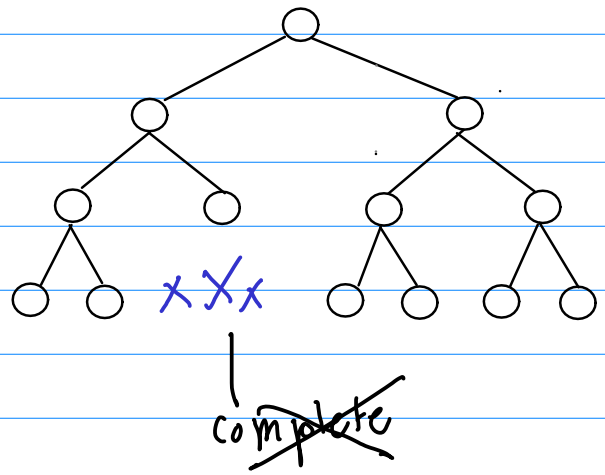
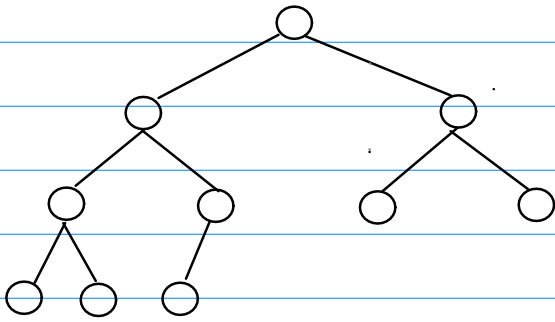
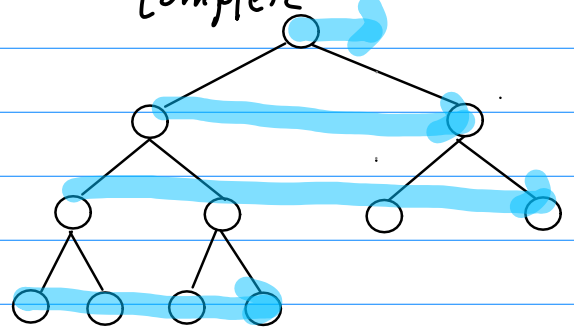
Heap

* Full Binary Tree

Full,
Complete



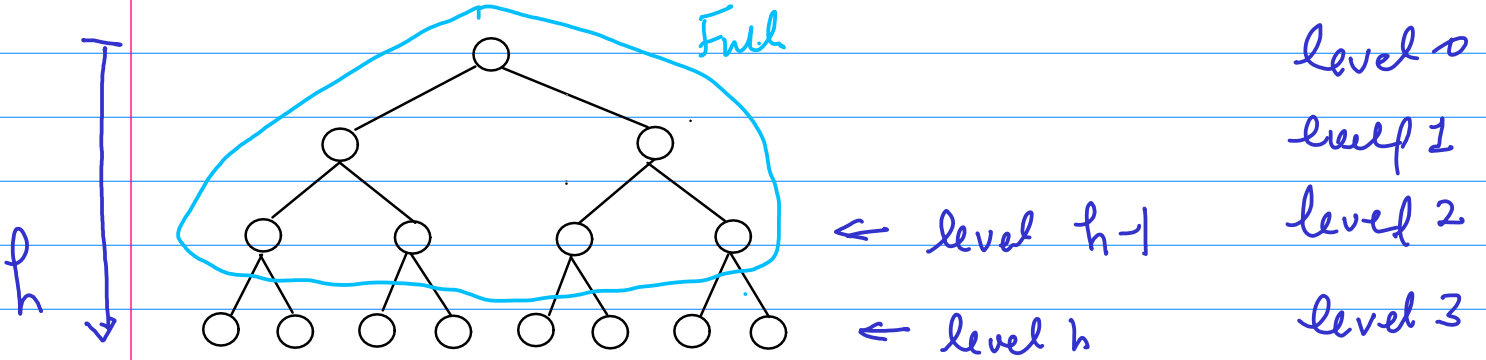
Complete



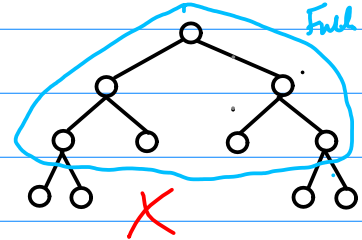
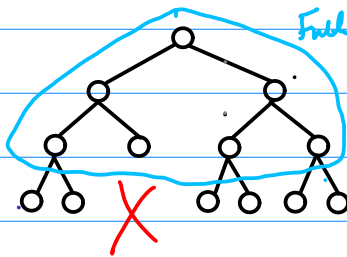
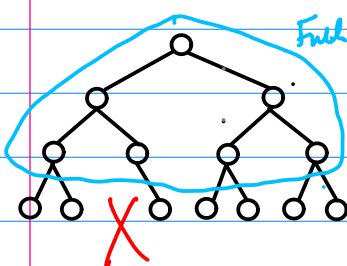
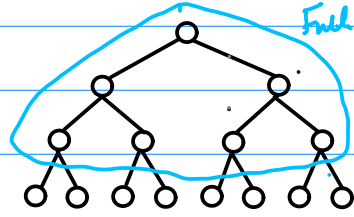
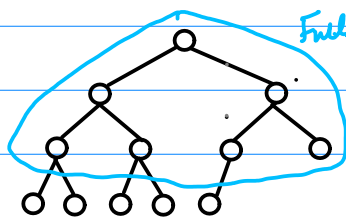
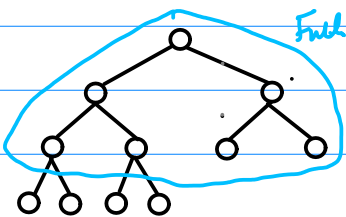
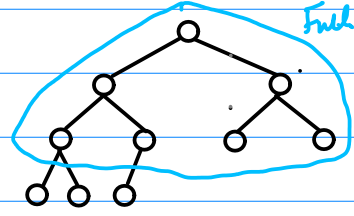
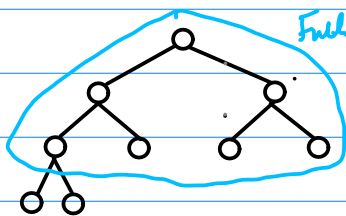
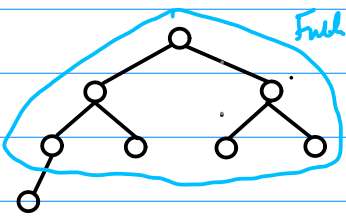
Complete Binary Tree

level (h-1) 까지는 Full

level h에서는 왼쪽 → 오른쪽 가면서 leaf를 채워나가는 구조라든 바르면 complete X



complete binary tree

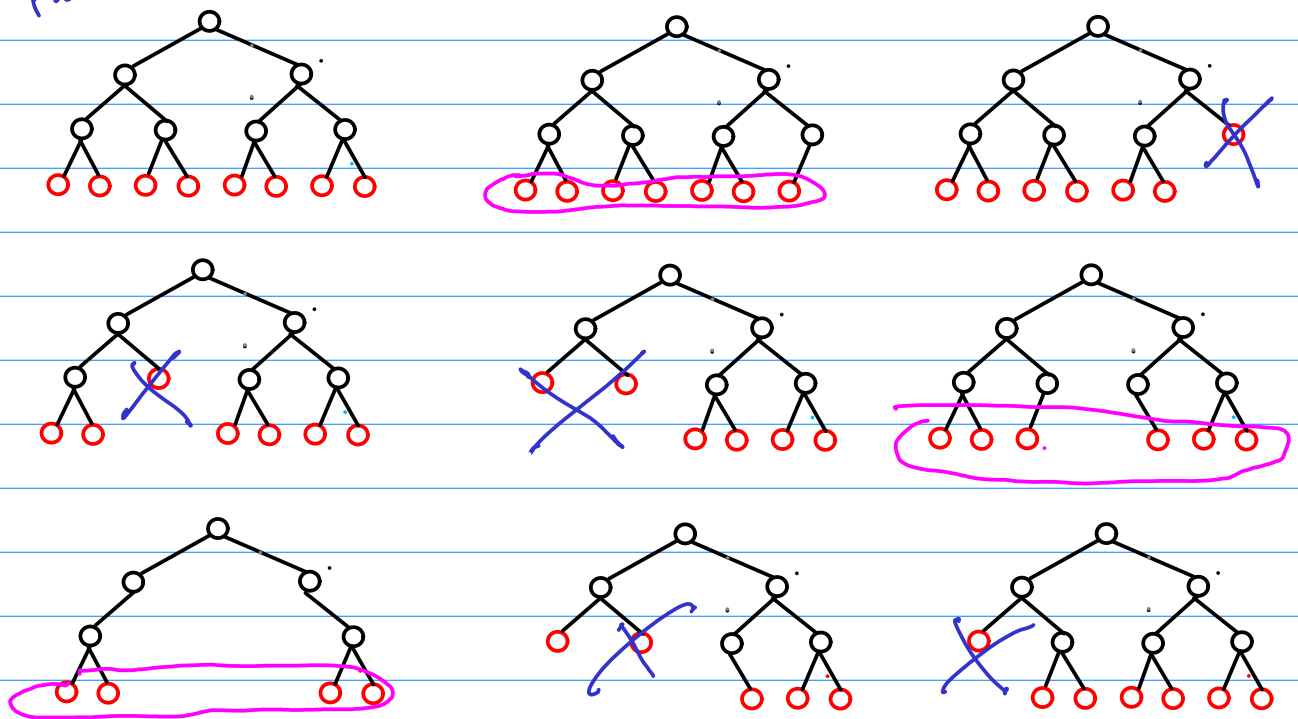


Full Binary Tree

? } 높이가 h 인 binary tree에서 모든 leaf node가 level h 에 있는 tree이다.

이렇게 리미트면 leaf node 위쪽의 모든 node는 반드시 가상 node를 두개 가지고 있어야 한다

Full



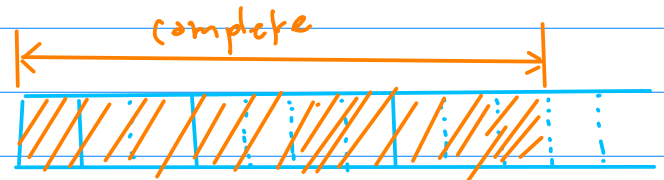
Two restricted forms of binary tree are sufficiently important to warrant special names. Each node in a **full** binary tree is either (1) an **internal node** with exactly two non-empty children or (2) a leaf. A **complete** binary tree has a restricted shape obtained by starting at the root and filling 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.



모든 internal node 가 2개의 children을 가짐.



complete

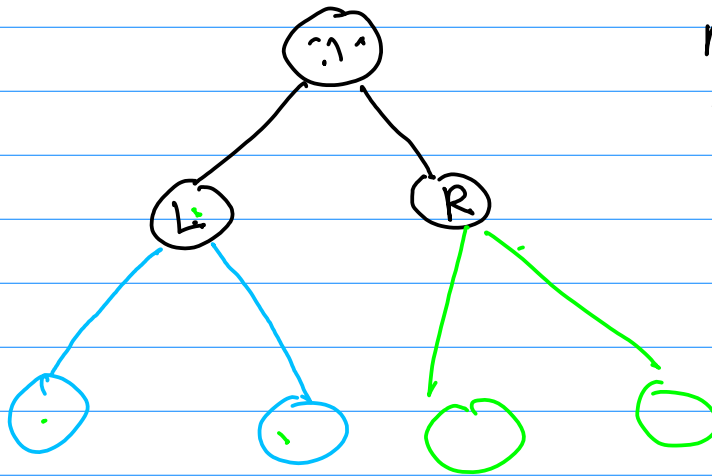


Heap.

①. Complete Binary tree.

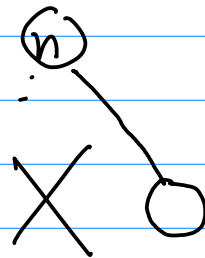
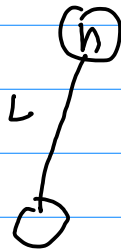
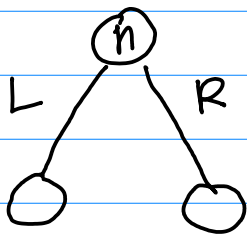


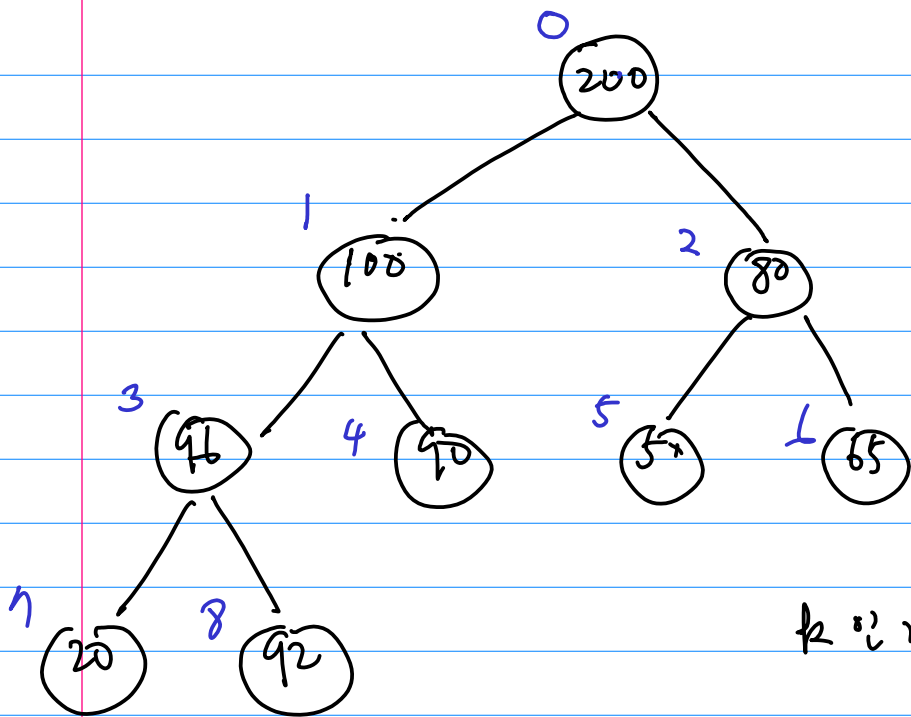
②. root의 key \geq 왼쪽 자식 \rightarrow 크기 순서로 생각할 수
root의 key \geq 오른쪽 자식



$n \text{ key} \geq L \text{ key}$
 $n \text{ key} \geq R \text{ key}$
 $L \text{ key} \geq R \text{ key}$

Complete BTree

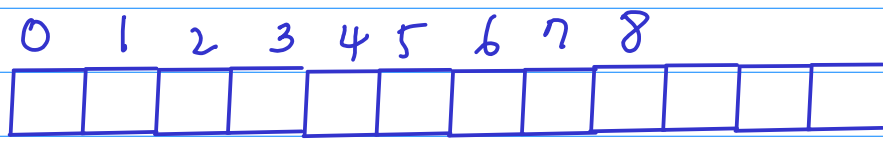
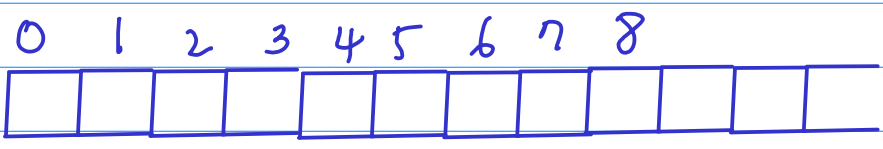
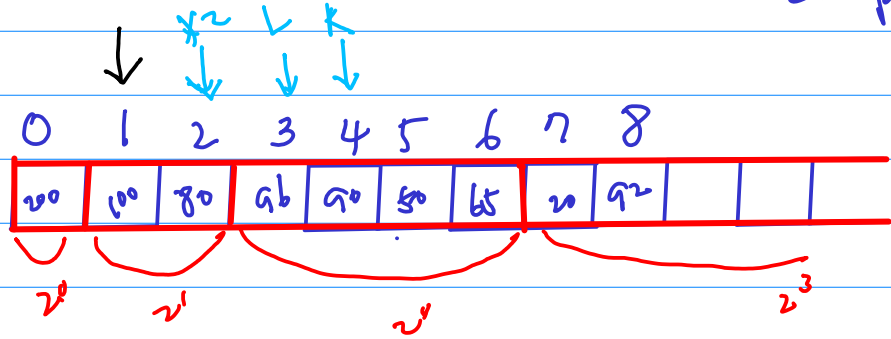




complete binary
 → 0110001
 7^진

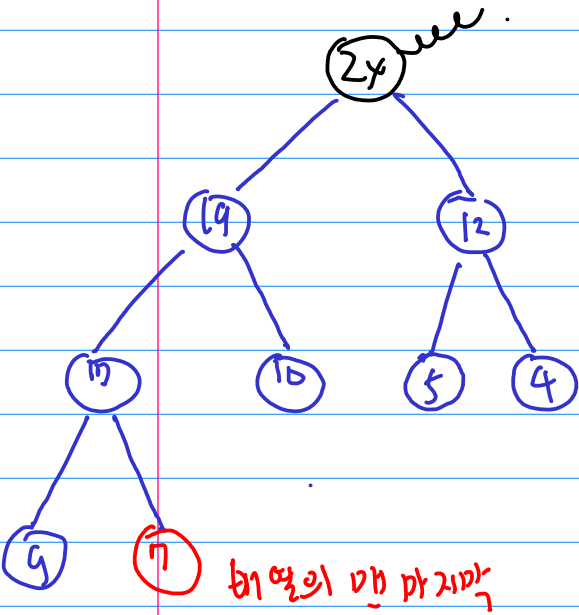
k 번째 node의 L → (2^{k+1})
 R → (2^{k+2})

k 번째 node의 parent $(k-1)/2$

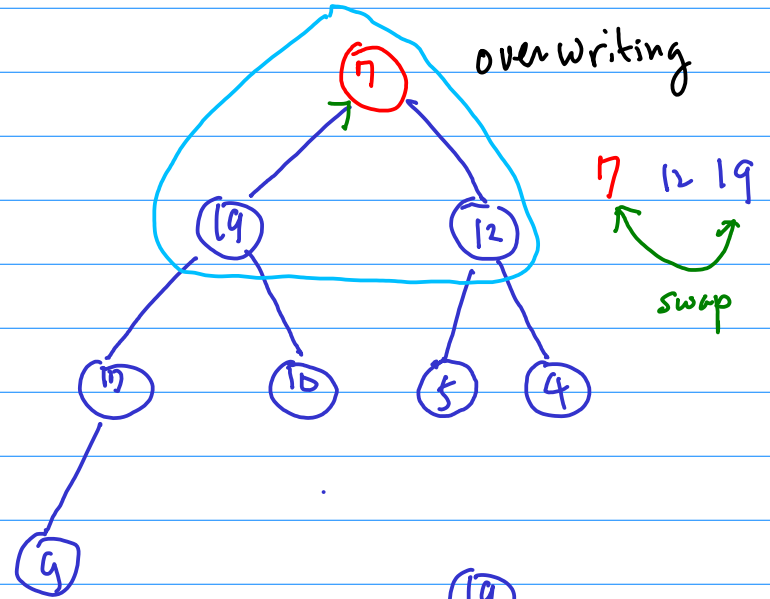


heap ... 우선순위가 큰 node가 위쪽.

root node가 삭제 되어야 함

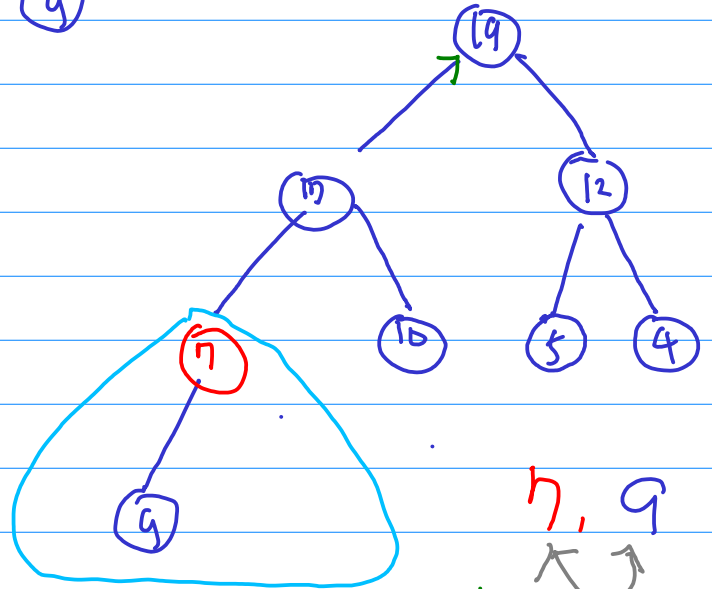
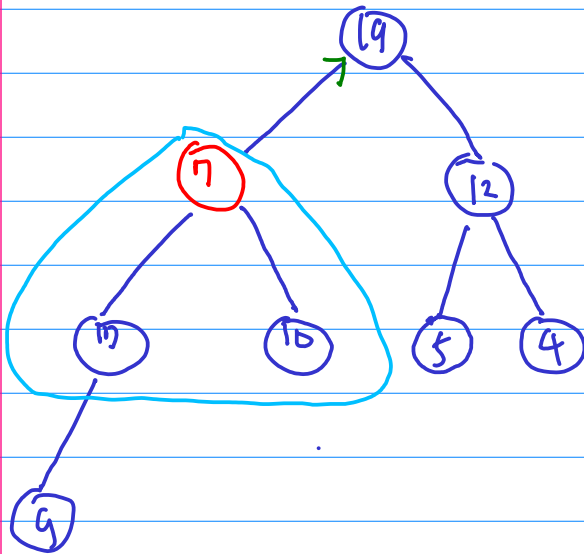


최대값의 맨 마지막



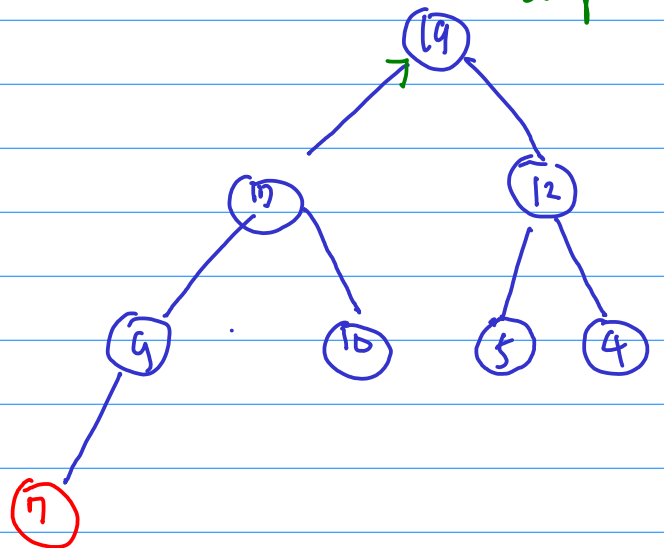
overwriting

7 12 19
swap



7, 9
swap

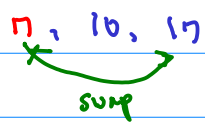
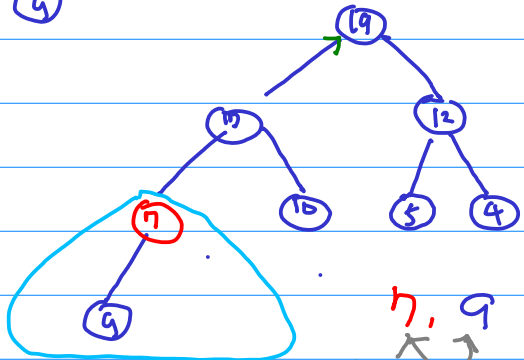
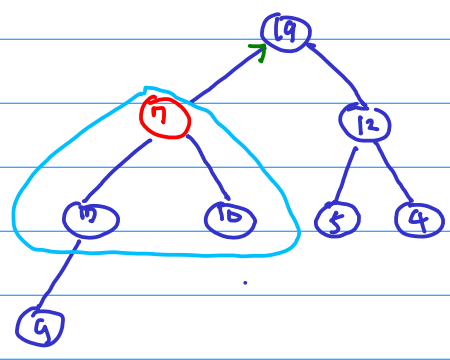
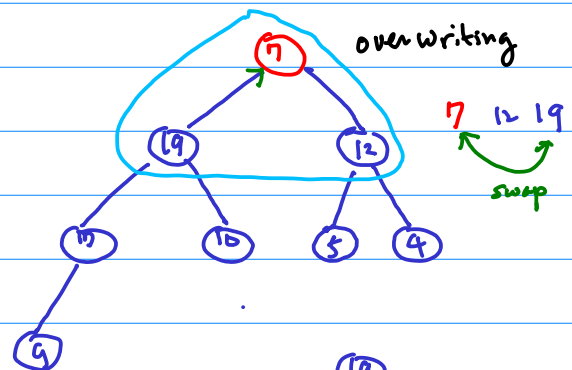
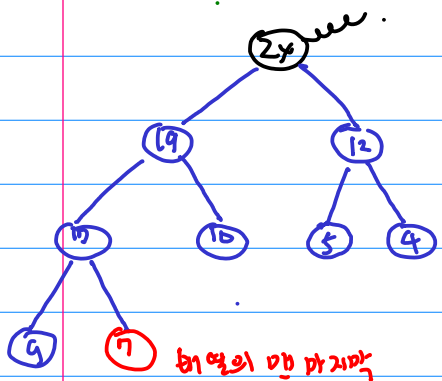
7, 10, 17
swap



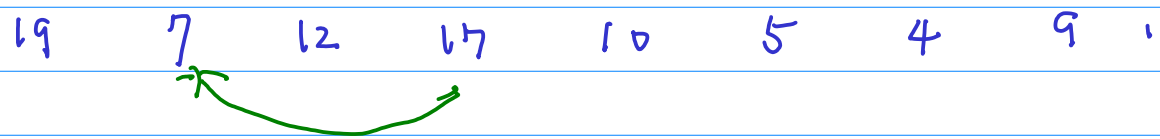
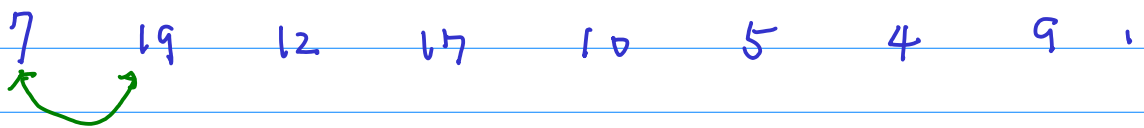
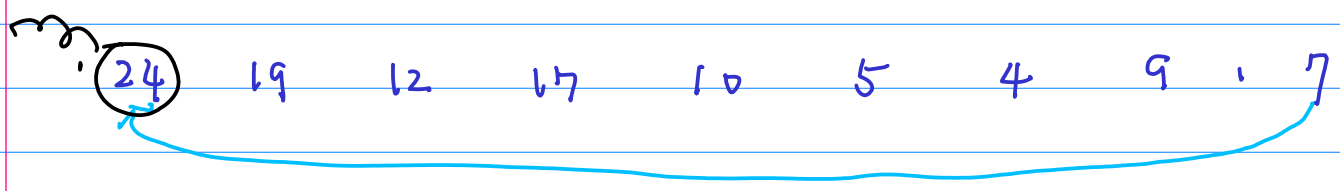
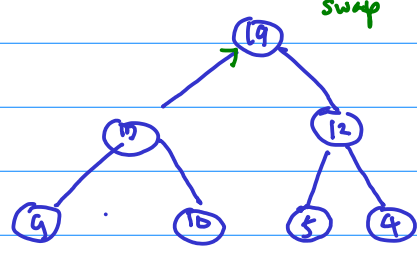
Heap Rebuilding

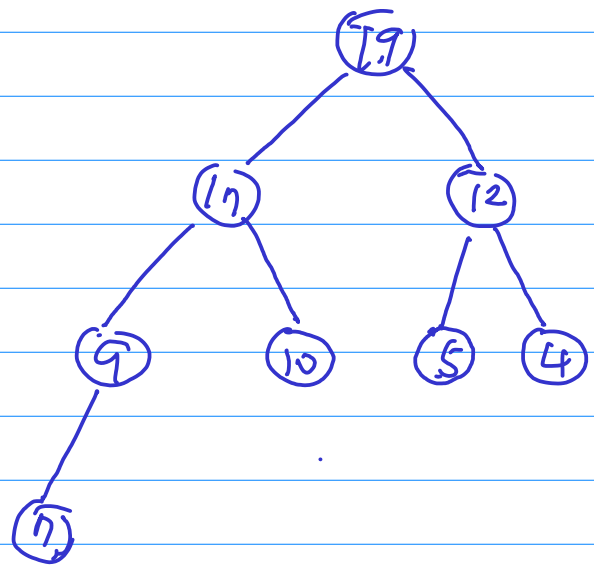
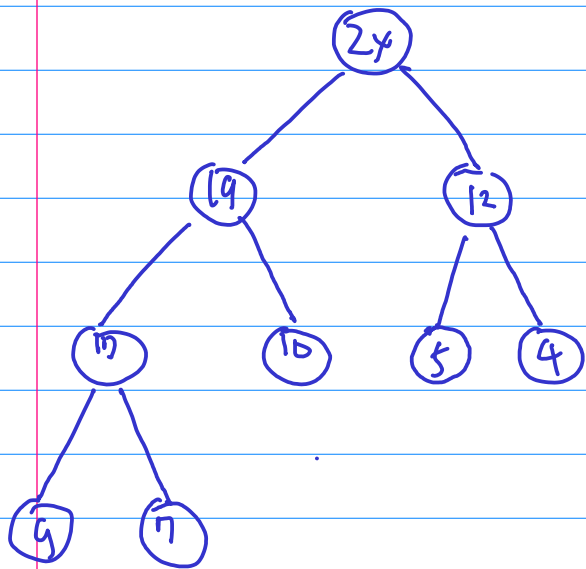
heap ... 우선순위가 큰 node가 위쪽.

root node가 삭제되어야 함



Heap Rebuilding





19 17 12 9 10 5 4 7