Heap

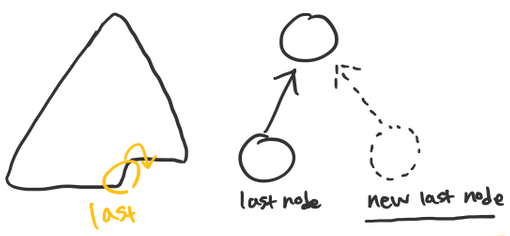
2018년 5월 24일 목요일 오후 12:03

- Complete Binary Tree
- Heap-Order: $key(v) \geq key(parent(v))$

• Height of a Heap

- n개의 키를 담는 힙의 높이는 $O(\log n)$이다.

• Heap order



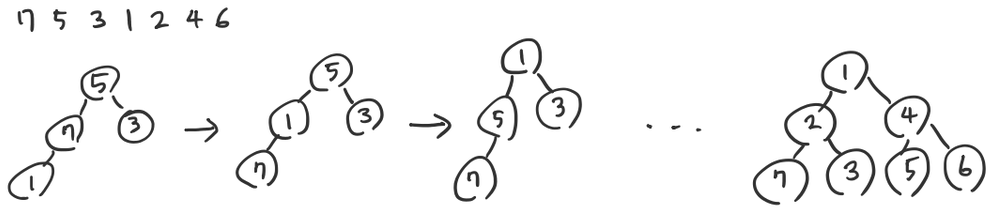
- Bubbling

추가하려면?

if self가 left child : right에 추가
else : 위로 쭉 올라갔다가 내려와서 추가

n개의 노드로 이뤄진 트리에서 1개의 노드를 insert : $O(\log n)$

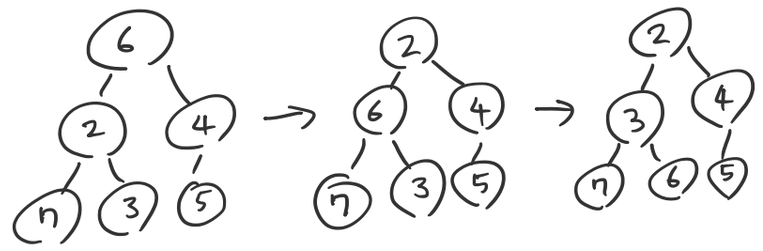
7 5 3 1 2 4 6



제거하려면?

① root를 last node로 overwrite.
② root의 child 중 작은 것과 root를 비교.
③ root가 작다면 해당 child와 교체.
④ 이를 반복.

$O(\log n)$



• Analysis of Heap sort

- 1개의 노드를 n개 노드로 이뤄진 트리에 insert: $O(\log n)$
- n개의 노드를 insert: $T(n) = \sum_{i=1}^{n} \log i$

※ $f(n) \leq c \cdot g(n) \Rightarrow f(n) \sim O(g(n))$
$f(n) \geq c \cdot g(n) \Rightarrow f(n) \sim \Omega(g(n))$

$f(n) = \log 1 + \log 2 + \cdots + \log n \leq \log n + \log n + \cdots + \log n \leq n \log n$

$\therefore f(n) \sim O(n \log n)$

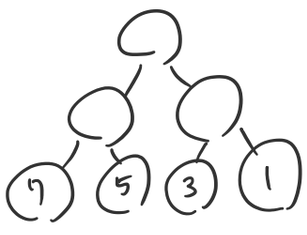
$f(n) = \log 1 + \log 2 + \cdots + \log n \geq \log 1 + \log 2 + \cdots + \log \frac{n}{2} \geq \log \frac{n}{2} + \log \frac{n}{2} + \cdots + \log \frac{n}{2}$

$= \frac{n}{2} \log \frac{n}{2}$

$= \frac{1}{2}[n \log n - \log 2] \Rightarrow f(n) \sim \Omega(n \log n)$

Heap sort는 $n \log n$보다 빠르지 않고, $n \log n$보다 빠르다. = Heap sort의 optimal은 $n \log n$

• Bottom of heap Construction



- $O(n)$ 시간이 걸림. 단, 몇개의 노드를 넣을지 개수가 정해져 있어야 함.
- Proxy method :

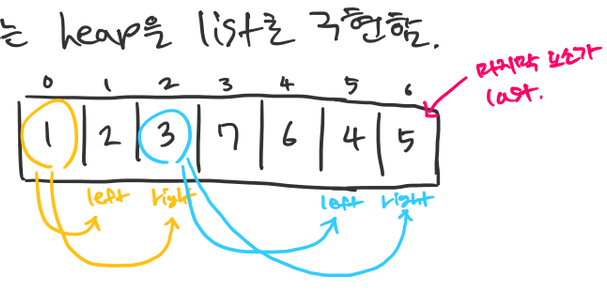
※ 과제

Heap Sort

- Binary tree를 inherit.
- Class variable last가 있음.
- insert와 remove min implementation

Practical하게는 heap을 list를 구현함.

index i → 0
node x → i
x.left → 2i+1
x.right → 2i+2



↳ Parent는 역순으로. / last 찾기는 매우 수월하지만 결국 Bubbling을 해야하기 때문에 time complexity는 $O(n \log n)$