演習問題 5.9

> 二項ヒープ、スプレーヒープ、ペアリングヒープにおいて、償却上限が示すより長い時間がかかってしまう例を示す。

償却上限の整理

実／償却	二項ヒープ	スプレーヒープ	ペアリングヒープ
`insert`	$O(\log n)$／$O(1)$	$O(n)$／$O(\log n)$	$O(1)$／$O(1)$
`merge`	$O(\log n)$／$O(\log n)$	$O(n)$／$O(\log n)$	$O(1)$／$O(1)$
`deleteMin`	$O(\log n)$／$O(\log n)$	$O(n)$／$O(\log n)$	$O(n)$／$O(\log n)$

方針

- 実／償却コストのギャップが存在する操作を使って例を作る
- 償却コストの埋め合わせ部分が発生する直前のデータを作る

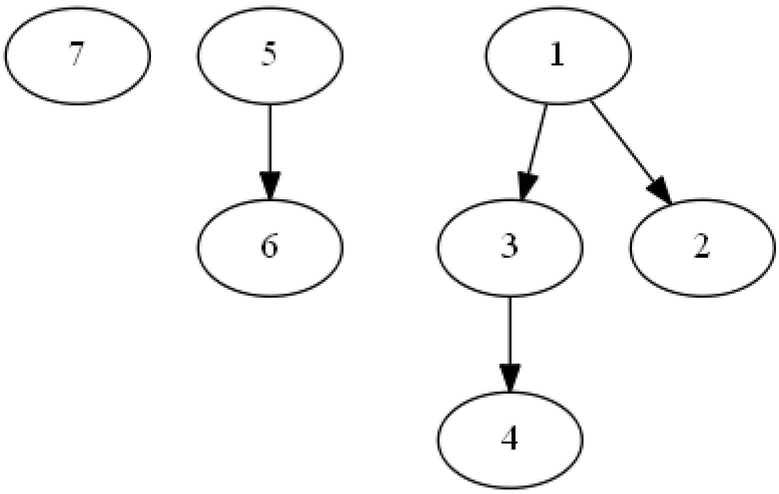
二項ヒープ

`insert` が $O(1)$ 償却上限で動作することは 5.3 節で示されている。

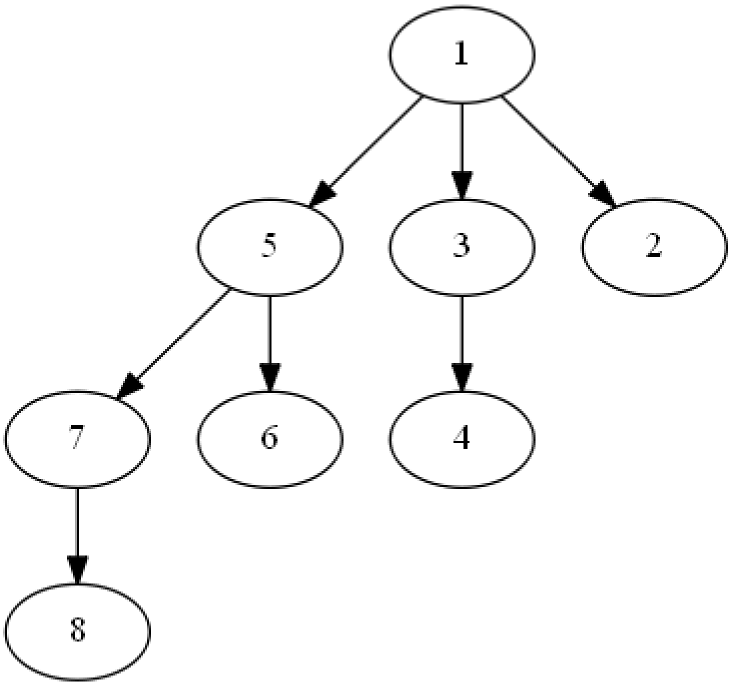
ここで、要素数 $n = 2^k - 1 = \underbrace{(1...1)_2}_{k}$ の二項ヒープへ `insert` すると、k 回 `link` されるため、`insert` の呼び出しは $k+1$ ステップすなわち $O(n)$ かかる。

参考

before



after



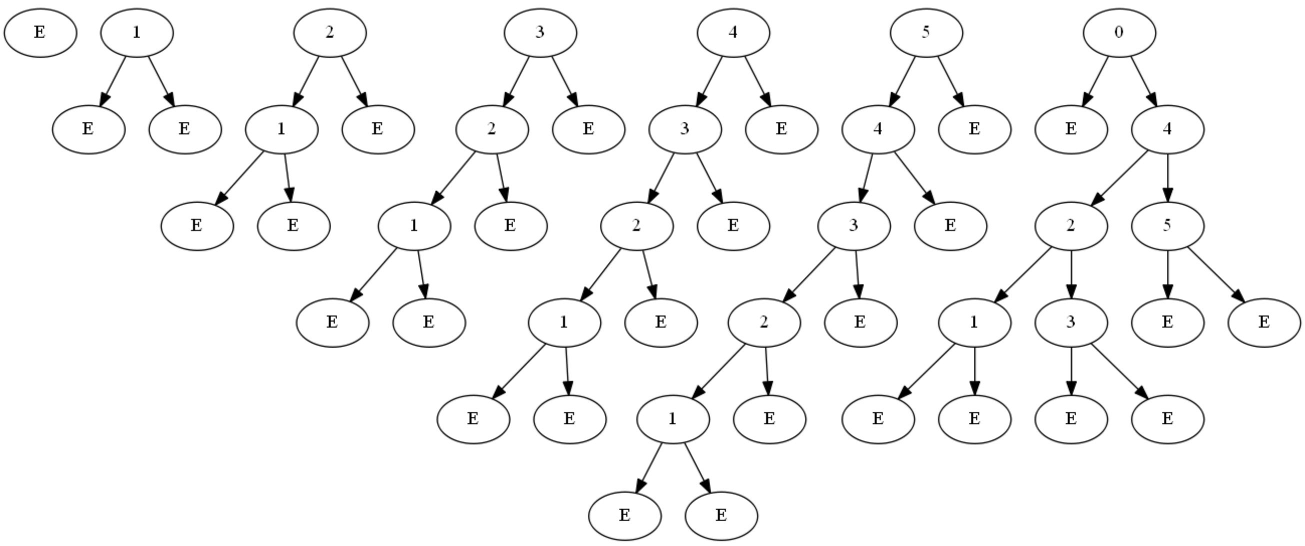
スプレーヒープ

`insert` が $O(\log n)$ 償却上限で動作することは 5.4 節で示されている。

ここで、空ヒープに対して n 個の値を昇順に `insert` したスプレーヒープを考えると、 `partition` で全く木の回転が発生せず、深さ n のヒープになる。

このヒープに対し、先述のどの n 個の値よりも小さい値を `insert` すると、最も深い位置まで `partition` が再帰するため $O(n)$ かかる。

参考



ペアリングヒープ

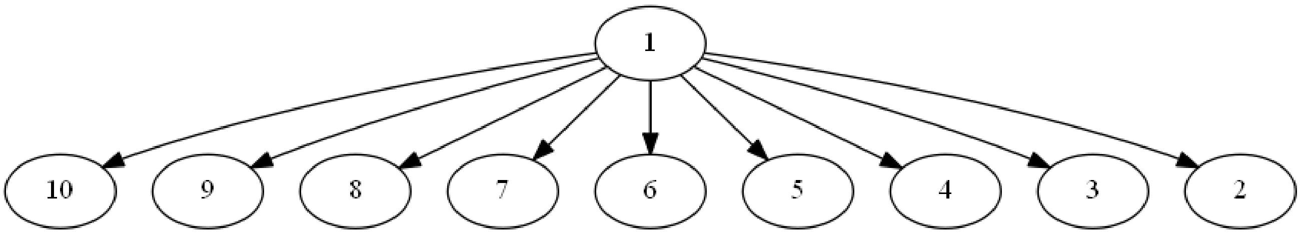
`deleteMin` が $O(\log n)$ 償却上限で動作することは 5.5 節で示されている。

ここで、空ヒープに対して n 個の値を昇順に `insert` したペアリングヒープを考えると、`merge` で挿入される側の木に要素が増え続け、$n > 1$ のとき根が $n-1$ 個の子のリストを持つヒープになる。

このヒープに対し `deleteMin` すると、根が取れて、 $n-1$ 個の子に対し `mergePairs` が走査するため $O(n)$ かかる。

参考

before



after