演習問題 5.3

> 二項ヒープの `merge` と `deleteMin` が$O(\log n)$償却時間で実行されることを示す。

物理学者法のアプローチで書きます。

merge

マージする2つのヒープに含まれる木の数をそれぞれ t_1, t_2 とする。

また、2つのヒープの要素数の和を n とする。

`merge` の呼び出しには $t_1 + t_2 + k$ かかる。

ただし k は `link` の呼び出し回数で、$\max(t_1, t_2) \geq k$ を満たす。

`merge` 後の木の個数は $t_1 + t_2 - k$ となる。

このとき償却コストは $(t_1 + t_2 + k) + (t_1 + t_2 - k) - (t_1 + t_2) = t_1 + t_2$ となる。

`merge` 前の要素数の和と `merge` 後の要素数は等しく n であるため、

$t_1 + t_2 - k$ は n の2進表記における 1 の数に等しい。

よって、$t_1 + t_2 - k \leq \log_2 n$ であり、$\max(t_1, t_2) \geq k$ から k は $t_1 + t_2$ のオーダーに影響しないので、

$t_1 + t_2$ は $O(\log n)$ となる。

deleteMin

ヒープに含まれる木の数を t とする。

`deleteMin` の呼び出しでは、 `removeMinTree` の呼び出しに伴う t 回の走査と、 `reverse` に伴う t_1 回の走査、木の数が $t_1, t-1$ であるような 2 つのヒープのマージ、そして k 回の `link` の呼び出しが発生する。

すなわち呼び出しコストは $t + t_1 + (t_1 + t - 1) + k = 2t + 2t_1 + k - 1$

ただし、$t_1 < t$ を満たす。

呼び出し後の木の数は、$t - 1 - k$ となる。

このとき償却コストは $(2t + 2t_1 + k - 1) + (t - 1 - k) - t = 2t + 2t_1 - 2 < 4t - 2$

t は n の2進表記における 1 の数に等しく、償却コストは t の定数倍に収まるので、

償却コストは $O(\log n)$ となる。