演習問題 7.1

`rotate` のパターンマッチを **fun lazy** として考える。

`snoc` を繰り返すと、`rotate` のサンクは $1, 3, 7, 15, \ldots, 2^k - 1$ 回目にできる。

$2^k - 1$ 回 `snoc` したキューに `tail` すると、`$Cons` のパターンマッチ時に k 回の `rotate` の評価が実行される。

そのため、`tail` は最悪時に $O(\log n)$ の計算量となる。

`head` も `$Cons` のパターンマッチを行うので同様に $O(\log n)$ となる。

`snoc` は、停止計算を作るが、進行させないので $O(1)$ となる…？

観察: n=7 の場合

7 回目の `snoc` では、`f` の部分に `rotate [1, 2, 3][7, 6, 5, 4] []` が作られる。

- （簡単のためにHaskell **風** に書いています）

ところで、`snoc` は停止計算を作るだけなので `[1, 2, 3]` の部分は実際には未評価であり、`rotate [1] [2,3] []` のようになっているはずである。

さらに、`[1]` も実際には `rotate [] [1] []` となっているはずである。

以上から、7 回目の `snoc` では、`rotate (rotate (rotate [] [1] []) [2,3] []) [7, 6, 5, 4] []` のようなサンクが作られる。

このような `f` を持つキューに対し、`tail` 等で先頭要素をパターンマッチしようとすると、3 回 `rotate` を評価する必要がある。

この 3 つの `rotate` サンクは、それぞれ、1, 3, 7 回目の `snoc` で作られたものである。