S8.2: Purely Functional Data Structures, Amortization, Chapter 5

CSci 2041:

Advanced Programming Principles

University of Minnesota,
Prof. Van Wyk,
Spring 2018

Exercise #1:Queue example operations

Try an example. What is the OCaml representation of the queue

- initially,
- after inserting an element with `snoc` — do this 3 times, and
- after removing an element with `tail` — do this 2 times

Exercise #2:Amortized costs - Physicist's method

In pairs, work out the argument using the Physicist's method.

That is, how does each operation change Φ? Why is it always positive? Why does `tail` have an amortized cost of $O(1)$?

Recall,

- Potential function Φ over data.
- Initially 0, always non-negative.
- Sets a lower bound on accumulated savings.
- Let d_i be result of i^{th} operation, input for $(i+1)^{th}$
- $a_i = t_i + \Phi(d_i) - \Phi(d_{i-1})$

Exercise #3:Ephemeral vs. Persistent

- Consider adding n elements to an empty queue? Call it `q1`.
  What is in the front list? The rear list?

-
  ```
  let q2 = tail q1
  let q3 = tail q2
  let q4 = tail q3
  ...
  ```
  n calls to `tail`, each on result of the previous.
  What behavior do we see?

Exercise #4:But ...

- What about this?

  ```
  let q2 = tail q1
  let q3 = tail q1
  let q4 = tail q1
  ...
  ```

  n calls to `tail`, each on **the original** `q1`.

- What is the cost of constructing `q1` and these n calls to `tail`?

- What went wrong?