Vectors in Stak Scheme

@raviqqe

February 7, 2026

Contents

Stak Scheme

- A bytecode compiler and virtual machine (VM) for Scheme
  - The compiler is written in Scheme.
  - The VM is written in Rust.
- It supports the R7RS-small standard.

Progress

- `O(log(n))` vectors

Background

- The previous implementation of vectors in Stak Scheme was based on lists.
- Element access is `O(n)`.
    - 😃

Vectors in R7RS

- Two vector types are defined in R7RS; `vector` and `bytevector`.
- Their operations seem to expect the *raw* vectors.
  - `vector-set!`: destructive update of a vector element.
  - `vector-copy!`: destructive copy of elements from another vector.
  - `vector-append`: persistent appending of elements in multiple vectors.
    - Allocates a **new** vector!
- The situation is similar for `bytevector`.
- The philosophy appears to be providing very basic data structures but not high level abstractions. 🤔

Choices in Stak Scheme

1. Raw vectors
    - We implement the raw vectors as real contiguous vectors in heap.
    - This is a bit difficult due to the current design of the VM focused in simplicity.
2. Radix vector
    - A vector based on the radix tree.
    - Each node can be a raw vector.
3. RRB vector
    - The state of the art data structure of persistent vectors
    - Every operation is `O(log(n))`.
        - Including concatenation, splits, and slicing.
    - Relaxed nodes require index arrays.

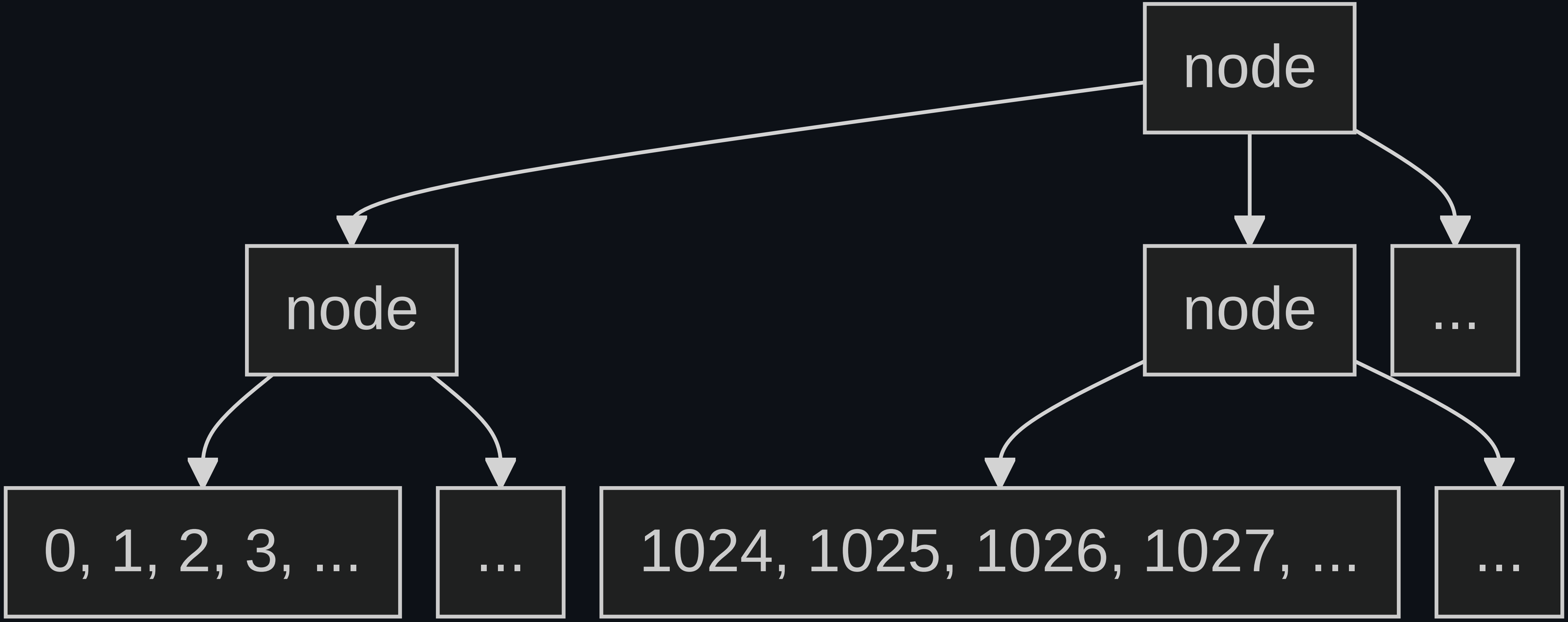
Choices in Stak Scheme

Stak Scheme implements the radix tree.

Why?

- RRB vector's optimality is very attractive.
- But with the costs of the algorithm and data structure complexity.
    - Especially, the additional index arrays do not seem to fit in the design of the VM.
    - In the worst case, it doubles the memory usage of vectors.

Radix tree



Implementation

- Each node is a list of elements.
    - The VM of Stak Scheme does not have any contiguous memory block but only cons cells.
- A slightly high branching factor of 64.
    - 32 is a popular choice for the cache line size?
    - But anyway, nodes are lists in Stak Scheme...
- The complexity of element access is `O(1)` practically.

Performance

- Baseline: `list`
  - `make-list`, `list-ref`, and `list-set!`
- Relative speed-up

Elements	**`vector`**
10	0.99
100	1.00
1000	0.96
5000	1.52
10000	3.50

Future work

- Soft float
- Rust integration

Summary

- Implementing `vector` is fun!