The new array module

Richard Carlsson
Dan Gudmundsson
Features

- Functional data structure
- Uses integer indices only (starting at 0)
- Default-value for uninitialized entries
- Can grow automatically, or have fixed size
- Can be resized
- Convert to/from lists
- Map/fold over array contents
Comparison to dict/gb_trees

Lookup: 10-30% faster
Update: 200-400% faster
Insert: 400-600% faster
Technology

- Trees of N-tuples (where N is approx. 10)
- Uses only integer arithmetic to find right slot
- No rebalancing of trees
- Easy to extend size
- Sparse representation for uninitialized slots
- Optimized for good read/write tradeoff
Creating new arrays

- array:new()
  - empty, extendible array

- array:new(Options)
  - Options is a term or list of terms:
    - N | {size, N} (N::integer(), fixed array by default)
    - {default, Value}
    - fixed | {fixed, true | false}

- array:new(Size, Options)
  - A convenience function
New arrays: examples

- array:new()
- array:new(10)
- array:new([10, {fixed,false}, {default,""}])
- array:new(100, {default,0})
Fixed vs. extendible

- **Extendible arrays:**
  - grow automatically on out-of-range stores
  - return the default value for out-of-range reads

- **Fixed arrays:**
  - 'badarg' error on all out-of-range accesses

- **Change mode at any time:**
  - `FixedArray = array:fix(SomeArray)`
  - `ExtendibleArray = array:relax(SomeArray)`
Storing and reading data

- array:set(Index, Value, Array)
- array:get(Index, Array)
  - default value is returned for uninitialized entries
    (unless index out of range and array is fixed)
- array:reset(Index, Array)
  - sets entry back to default value
  - no difference between uninitialized and initialized-to-the-default-value
Whole-array operations

- to_list, from_list
- to_orddict, from_orddict
- map
- foldl, foldr
“sparse” functions

• sparse_... versions of whole-array operations ignore default-valued entries (often useful)

• Example:
  - A = array:from_orddict([\{4, four\}, \{7, seven\}])
  - \([\{4, four\}, \{7, seven\}] = array:sparse_to_orddict(A)
  - array:to_orddict(A) = \([\{0, undefined\}, ...]\)

• array:sparse_size(A) scans from the end for the last non-default valued entry
Resizing arrays

- NewArray = array:resize(NewSize, Array)
  - preserves “fixedness”
- NewArray = array:resize(Array)
  - resizes to sparse_size(Array)
- Currently, resizing down does not shrink the representation; only the reported size
...that's basically it

Questions?