1 Elias-Fano Implementation Notes in Julia

1.1 Source

The contents of this document were compiled from

- Julia's official docs (https://docs.julialang.org/en/v1/), and
- Julia's source code on GitHub (https://github.com/JuliaLang/julia).

1.2 `AbstractArray` Interface in Julia

`EFArray` should be of subtype `AbstractArray`.

- Required methods:
  - `size(A)`
  - `getindex(A, i::Int)`
  - `setindex!(A, v, i::Int)`
- Optional methods:
  - `IndexStyle(::Type)`
  - `getindex(A, I...)`
  - `setindex!(A, I...)`
  - `iterate`
  - `length(A)`
  - `similar(A)`
  - `similar(A, ::Type{S})`
  - `similar(A, dims::Dims)`
  - `similar(A, ::Type{S}, dims::Dims)`
- – TODO – what is the expected behavior of these functions?

1.3 `IndexStyle`

Array data structures are usually defined in one of two ways:

1. Use one index to efficiently access an array's elements – known as linear indexing
2. Use indices specified for every dimension to intrinsically access an array's elements

In Julia, `IndexLinear()` is used to declare arrays of the first type, and `IndexCartesian()` is used for arrays of the second type.

1.3.1 `IndexLinear()`

`IndexLinear()` requires only `getindex(A::ArrayType, i::Int)`.

- If the array is indexed with multidimensional indices, the fallback `getindex(A::ArrayType, I...)()` converts the indices into one linear index and calls the previous method.

1.3.2 `IndexCartesian()`

`IndexCartesian()` requires methods to be defined for each supported dimensionality with `ndims(A)` `Int` indices.

1.4 Parameters of `AbstractArray`

There are two important parameters when defining a subtype of `AbstractArray`: `eltype` and `ndims`. Defining these two parameters and the three required methods (see § 1.2) allows our subtype to act as a fully functioning `Array`. (The most important qualities being indexable and iterable.)

1. `eltype(type)`
   - Type of elements generated by iterating over a collection of the given `type`
   - New types must define `eltype(::Type)`.
   - The declaration

     ```
     eltype(x) = eltype(typeof(x))
     ```

     is provided to allow an instance to be passed instead of a type.
2. `ndims(A::AbstractArray) -> Integer`
   - Returns the number of dimensions of `A`

1.5 Other Specific Implementation Notes

– TODO –