```
struct TensorKeyIterator{G<:Sector,F₁<:FusionTree{G},F₂<:FusionTree{G}}
    rowr::SectorDict{G, FusionTreeDict{F₁, UnitRange{Int}}}
    colr::SectorDict{G, FusionTreeDict{F₂, UnitRange{Int}}}
end
struct TensorPairIterator{G<:Sector, F₁<:FusionTree{G}, F₂<:FusionTree{G},
A<:DenseMatrix}
    rowr::SectorDict{G, FusionTreeDict{F₁, UnitRange{Int}}}
    colr::SectorDict{G, FusionTreeDict{F₂, UnitRange{Int}}}
    data::SectorDict{G, A}
end

const TensorIterator{G<:Sector,F₁<:FusionTree{G},F₂<:FusionTree{G}} =
Union{TensorKeyIterator{G,F₁,F₂},TensorPairIterator{G,F₁,F₂}}

Base.IteratorSize(::Type{<:TensorIterator}) = Base.HasLength()
Base.IteratorEltype(::Type{<:TensorIterator}) = Base.HasEltype()
Base.eltype(T::Type{TensorKeyIterator{G,F₁,F₂}}) where {G,F₁,F₂} = Tuple{F₁,F₂}

function Base.length(t::TensorKeyIterator)
    l = 0
    for (rowdict, coldict) in zip(values(t.rowr), values(t.colr))
        l += length(rowdict) * length(coldict)
    end
    return l
end
function Base.iterate(it::TensorKeyIterator)
    i = 1
    i > length(it.rowr) && return nothing
    rowit, colit = it.rowr.values[i], it.colr.values[i]

    rownext = iterate(rowit)
    colnext = iterate(colit)
    # while rownext === nothing || colnext === nothing: Julia did not infer that
      after while loop, both were not nothing
    while true
        if rownext === nothing
            i += 1
        elseif colnext === nothing
            i += 1
        else
            break
        end
        i > length(it.rowr) && return nothing
        rowit, colit = it.rowr.values[i], it.colr.values[i]
        rownext = iterate(rowit)
        colnext = iterate(colit)
    end
    (f1, r1), rowstate = rownext
    (f2, r2), colstate = colnext

    return (f1, f2), (f2, i, rowstate, colstate)
end
function Base.iterate(it::TensorKeyIterator, state)
    (f2, i, rowstate, colstate) = state
```

```
    rowit, colit = it.rowr.values[i], it.colr.values[i]
    rownext = iterate(rowit, rowstate)
    if rownext !== nothing
        (f1, r1), rowstate = rownext
        return (f1, f2), (f2, i, rowstate, colstate)
    end
    colnext = iterate(colit, colstate)
    if colnext !== nothing
        rownext = iterate(rowit) # should not be nothing
        @assert rownext !== nothing
        (f1, r1), rowstate = rownext
        (f2, r2), colstate = colnext
        return (f1, f2), (f2, i, rowstate, colstate)
    end
    while true
        if rownext === nothing
            i += 1
        elseif colnext === nothing
            i += 1
        else
            break
        end
        i > length(it.rowr) && return nothing
        rowit, colit = it.rowr.values[i], it.colr.values[i]
        rownext = iterate(rowit)
        colnext = iterate(colit)
    end
    (f1, r1), rowstate = rownext
    (f2, r2), colstate = colnext

    return (f1, f2), (f2, i, rowstate, colstate)
end

# WARNING: This only works if both SectorDict and FusionTreeDict are VectorDict
# function Base.iterate(it::TensorKeyIterator, s = (1,1,1))
#     i,j,k = s
#     length(it.rowr) < i && return nothing
#     @inbounds begin
#         f1 = it.rowr.values[i].keys[j]
#         f2 = it.colr.values[i].keys[k]
#     end
#
#     if j < length(it.rowr.values[i])
#         j += 1
#     elseif k < length(it.colr.values[i])
#         j = 1
#         k += 1
#     else
#         j = 1
#         k = 1
#         i += 1
#     end
#
#     return (f1,f2), (i,j,k)
```

```
# end
```