```
function _verify_indexorder(network, indexorder)
    allindices = Vector{Any}()
    for ind in network
        append!(allindices, ind)
    end
    while length(allindices) > 0
        i = pop!(allindices)
        if i ∈ indexorder # contraction index
            k = findfirst(isequal(i), allindices)
            k === nothing && return false
            l = findnext(isequal(i), allindices, k+1)
            l !== nothing && return false
            deleteat!(allindices, k)
        else
            findfirst(isequal(i), allindices) === nothing || return false
        end
    end
    return true
end

function indexordertree(network, indexorder)
    _verify_indexorder(network, indexorder) ||
        throw(ArgumentError("not a valid contraction tree with contraction indices
$indexorder"))
    contractionindices = Vector{Vector{Any}}(undef, length(network))
    for k = 1:length(network)
        indices = network[k]
        # trace indices have already been removed, remove open indices by filtering
          on positive values
        contractionindices[k] = filter(in(indexorder), indices)
    end
    partialtrees = collect(Any, 1:length(network))
    _indexordertree!(partialtrees, contractionindices, indexorder)
end

function _indexordertree!(partialtrees, contractionindices, indexorder)
    while length(indexorder) > 0
        c = indexorder[1]
        i1 = findfirst(x->in(c,x), contractionindices)
        if i1 === nothing
            indexorder = indexorder[2:end]
        else
            i2 = findnext(x->in(c,x), contractionindices, i1+1)
            @assert i2 !== nothing
            cs = intersect(contractionindices[i1], contractionindices[i2])
            indexorder = setdiff(indexorder, cs)
            newindices = unique2(vcat(contractionindices[i1],
contractionindices[i2]))
            newtree = Any[partialtrees[i1], partialtrees[i2]]
            partialtrees[i1] = newtree
            deleteat!(partialtrees, i2)
            contractionindices[i1] = newindices
            deleteat!(contractionindices, i2)
        end
```

```
    end
    if length(partialtrees) > 1
        @assert all(isempty, contractionindices) # disconnected network
        while length(partialtrees) > 1
            partialtrees[end-1] = Any[partialtrees[end-1], partialtrees[end]]
            pop!(partialtrees)
            pop!(contractionindices)
        end
    end
    return partialtrees[1]
end
```