```
function linearizepermutation(p1::NTuple{N₁,Int}, p2::NTuple{N₂}, n₁::Int,
n₂::Int) where {N₁,N₂}
    p1′ = ntuple(StaticLength(N₁)) do n
        p1[n] > n₁ ? n₂+2n₁+1-p1[n] : p1[n]
    end
    p2′ = ntuple(StaticLength(N₂)) do n
        p2[N₂+1-n] > n₁ ? n₂+2n₁+1-p2[N₂+1-n] : p2[N₂+1-n]
    end
    return (p1′..., p2′...)
end

function permutation2swaps(perm)
    p = collect(perm)
    @assert isperm(p)
    swaps = Vector{Int}()
    N = length(p)
    for k = 1:N-1
        append!(swaps, p[k]-1:-1:k)
        for l = k+1:N
            if p[l] < p[k]
                p[l] += 1
            end
        end
        p[k] = k
    end
    return swaps
end

function _kron(A, B)
    sA = size(A)
    sB = size(B)
    s = map(*, sA, sB)
    C = similar(A, promote_type(eltype(A),eltype(B)), s)
    for IA in eachindex(IndexCartesian(), A)
        for IB in eachindex(IndexCartesian(), B)
            I = CartesianIndex(IB.I .+ (IA.I .- 1) .* sB)
            C[I] = A[IA]*B[IB]
        end
    end
    return C
end
```