The efficiency of `element-of-set?` depends on the length of the list but not necessarily the length of the set. If there are duplicates, then `element-of-set?` could take longer to determine if an element is of a set compared to its counterpart without duplicates. For example, the number of steps required could grow as $\Theta(n^2)$, if each element of the set n has n many duplicates. However, as long as there are only a handful of duplicates for each element, then the constant is dropped from the Θ and the number of steps will grow as $\Theta(n)$. This is also negligible if duplicates are somewhat evenly spread and the element in question *is* a member of the set.

The efficiency of `adjoin-set` when allowing for duplicates is much improved as the number of steps grows $\Theta(1)$, since it will always only require one step to adjoin regardless of the number of elements in the set.

The efficency of `union-set` is also improved (as long as the size of the list is less than n^2 where n is the number of elements in the set) as no `element-of-set?` checks need to be performed. The number of steps grows as the size of the list, which may be $\Theta(n)$ or larger. This contrasts to the previous growth of $\Theta(n^2)$ due to the number of `element-of-set?` calls growing with the number of elements in the second set n.

The efficiency of `intersection-set` is identical if the sets do not have duplicates, but each duplicate is potentially doubly costly as duplicates result in a less efficient `element-of-set?` and more calls to `element-of-set?`, which could yield a growth in the number of steps larger than $\Theta(n^2)$.

This representation could be preferred to the one without duplicates when sets are mostly being adjoined or unions are being computed, or if the number of duplicates actually present is small. When intersections are being computed or the number of duplicates grows proportional to the number of elements, then the representation without duplicates will be more efficient.