2. Sequence Algorithm Concepts

Section authors: David R. Musser and Brian Osman.

2.1. Sequence Algorithm

A sequence algorithm is an algorithm (§1.2) that takes one or more linear sequences as inputs.
Refinement of: Algorithm Specialized by Input (§1.3).
A comparison based sequence algorithm is a sequence algorithm (§2.1) whose computation depends on comparisons between pair of values in the sequence. Such an algorithm depends upon a comparison operator, one that is either previously defined as < or is passed to the algorithm. In either case the comparison operator must compute a Strict Weak Ordering (§5.1) on the value type of the sequence.

Refinement of: Sequence Algorithm (§2.1).
A *index based sequence algorithm* is a sequence algorithm (§2.1) that operates only on the positions within the sequence, independently of the values stored.

**Refinement of:** Sequence Algorithm (§2.1).
A *predicate based sequence algorithm* is a sequence algorithm (§2.1) whose computation depends on the results of applying a given predicate to values in the sequence.

**Refinement of:** Sequence Algorithm (§2.1).
A sequence permuting algorithm is a sequence algorithm (§2.1) whose output is a permutation of its input.

Refinement of: Sequence Algorithm (§2.1).
2.6. Sequence Sorting Algorithm

Refinement of: Comparison Based (§2.2), Permuting (§2.5), Sequence Algorithm (§2.1).

Input: Iterators \texttt{first} and \texttt{last} delimiting a range of elements [\texttt{first}, \texttt{last}) and optionally a comparison operator (§2.2) \texttt{comp}.

Output: A modified sequence of elements in the same range.

Effects:
After execution, the elements in \([\text{first}, \text{last})\) are a permutation (§2.5) of the input.

After execution, the elements in \([\text{first}, \text{last})\) are in nondecreasing order according to the comparison operator defined on the value type of the sequence or passed to the algorithm as parameter \text{comp}.
2.7. Sequence Selection Algorithm

Refinement of: Comparison Based (§2.2), Permuting (§2.5), Sequence Algorithm (§2.1).

Input: Iterators first, nth and last such that nth is in the range [first, last), and optionally a comparison operator (§2.2) comp.

Output: A modified sequence of elements in the same range.

Effects:
• After execution, the elements in \([\text{first}, \text{last})\) are a permutation \((\S 2.5)\) of the input.

• After execution, the element pointed to by the iterator \(\text{nth}\) is the same as the element that would be in that position if the entire range \([\text{first}, \text{last})\) had been sorted, and none of the elements in \([\text{nth}, \text{last})\) are less than any of the elements in the range \([\text{first}, \text{nth})\).

• The reordering is done according to the comparison operator defined on the value type of the sequence or passed to the algorithm as parameter \(\text{comp}\).
2.8. Sequence Rotation Algorithm

Refinement of: Index Based (§2.3), Permuting (§2.5), Sequence Algorithm (§2.1).

Input: Iterators first, middle, and last such that first and last delimit a range of elements [first, last) and the range [first, middle) is a prefix of [first, last).

Output: A modified sequence of elements in the range [first, last).
Effects: After execution, the elements in \([\text{first}, \text{last})\) are those that were in \([\text{middle}, \text{last})\) in the input, followed by those that were in \([\text{first}, \text{middle})\) in the input.
2.9. **Sequence Reversal Algorithm**

![Diagram showing the relationship between Index Based, Permuting, and Sequence Algorithm]

**Refinement of:** Index Based (§2.3), Permuting (§2.5), Sequence Algorithm (§2.1).

**Input:** A sequence of elements in a range [first, last).

**Output:** A modified sequence of elements in the same range.

**Effects:** After execution, the elements in [first, last) are the same as those in the input, but in the reverse order.