Algorithms (continued)

Linear search for \( x \) in a set \( \{ a_i, i = 1..n \} \)

For \( i = 1 \) to \( n \)
    If \( x = a_i \), Return \( i \)
Return False

Complexity: \( O(n) \)

Binary search for \( x \) in an ordered sequence
\( \{ a_i, i = 1..n \mid i < j \rightarrow a_i \leq a_j \} \)

\( n_1 := 1 \),
\( n_2 := n \),
while \( n_1 < n_2 \)
    \( m := \text{floor}((n_1+n_2)/2) \)
    If \( x = a_m \), Return \( m \)
    Else if \( x < a_m \), \( n_2 := m \)
    Else \( n_1 := m + 1 \)
Return False

Complexity: \( O(\log n) \)

Bubble sort of a sequence \( \{ a_i, i = 1..n \} \)

For \( i = n-1 \) down to 1,
    For \( j = 1 \) to \( i \)
        If \( a_j > a_{j+1} \), Swap( \( a_j, a_{j+1} \) )

Complexity: \( O(n^2) \)
c = \text{Merge}(a, b): \text{ Merging two ordered sequences}
\{a_i, i = 1..n \mid i < j \rightarrow a_i \leq a_j\}, \{b_j, i = 1..m \mid i < j \rightarrow b_i \leq b_j\}
\text{Into one sequence } \{c_j, i = 1..m+n \mid i < j \rightarrow c_i \leq c_j\}

i := 1
j := 1
for k = 1 to m + n
    if i \leq m \text{ and } (j > m \text{ or } a_i < b_j) \quad // \text{take the next element from } a
        c_k := a_i
        i := i + 1
    else \quad // \text{take the next element from } b
        c_k := b_j
        j := j + 1

\text{Merge-sort algorithm for sorting a sequence } \{a_i, i = 1..n\}

\text{Traveling salesman problem}

\text{Lexicographic ordering of permutations}

\text{Algorithm for enumerating permutations of } n \text{ numbers, } 1 \text{ to } n

1. Start with the permutation \{1, 2, \ldots, n\}
2. Repeat until the last permutation \{n, n-1, \ldots, 1\} is reached:
   Given permutation \{p_1, p_2, \ldots p_n\}, determine the next permutation \{q_1, q_2, \ldots q_n\} as follows:
   1) Find the largest \(j\) such that \(p_j < p_{j+1}\)
   2) \(m := \min \{ p_i \mid i > j, p_i > p_j \}\)
   3) \(S := \{1, 2, \ldots, n\} \setminus \{p_1, p_2, \ldots p_{j-1}, m\}\)
   4) \(S := \text{Sort}(S)\)
   5) \{q_1 \ldots q_n\} = \text{Concatenate}(\{p_1, p_2, \ldots p_{j-1}, m\}, S)\)

Complexity: \(O(n!n)\)