Mergesort

MERGE-SORT(A)
if len(A) < 2
    return a copy of A
else
    L ← 1st half of A
    R ← rest of A
    L ← MERGE-SORT(L)
    R ← MERGE-SORT(R)
    return MERGE(L, R)

MERGE(L, R)
A ← empty list
i ← 0
j ← 0
while i < len(L) and j < len(R)
    if L[i] ≤ R[j]
        append L[i] to A
        i ← i + 1
    else
        append R[j] to A
        j ← j + 1
append L[i..len(L)-1] to A
append R[j..len(R)-1] to A
return A
Mergesort Example

The diagram illustrates the merge sort algorithm with a sorted list of numbers: 1, 2, 3, 4, 5, 6, 8, 9.

The top portion shows the initial list:

1, 8, 4, 2, 5, 9, 3, 6

The divide process is shown recursively, breaking the list into smaller sublists until single element lists are reached.

The merge process then takes place, combining these sublists back together in a sorted manner.

The diagram highlights the number of comparisons and copies made during the merge sort process, indicating that the total number of comparisons is \( O(n \log n) \).

The formula for the number of comparisons is given by:

\[
\text{total comparisons} = n \log_2 n
\]

Where \( n \) is the number of elements in the list.

The number of copies made at each level of the divide process is also shown.

Total copies made in divide and sort steps:

\[
\text{total copies} = n \log_2 n
\]
List of Generic Elements

Array-based list

num    33

list-add(1, &num)

l    3

elt    6

list-id = list-create(1)
for (i=1; i<=5; i++)
   list-add(l, &i)