# PG4200 Algorithms and Data Structures

# Re-sit exam

## 6/8/2025

Candidate: 31

### Question 4:

### LO3: Sorting Algorithms

Merge Sort is a divide-and-conquer algorithm. It works by repeatedly dividing the array into smaller subarrays until each contains only one element, then merging them back in sorted order (Sedgewick & Wayne, 2011, Ch.2.2; Goodrich & Tamassia, 2006, Ch.11.3).

Algorithm steps *(adapted from Sedgewick & Wayne, 2011):*

1. If the array has only one element, return (it is already sorted)
2. Otherwise:
3. Find the middle of the array
4. Split the array into two halves

* Left half(U)
* Right half(V)

1. Recursively apply Merge Sort to both halves
2. Merge the two sorted halves:
3. Create an empty array result
4. Compare the first elements of both halves
5. Append the smaller element to result and remove it from its half
6. Repeat until one half is empty
7. Append the remaining elements from the non-empty half to result
8. Return result

Merge Process Table, merging two arrays U and V into one array S

Initial values:

70, 50, 30, 10, 20, 40, 60

|  |  |  |  |
| --- | --- | --- | --- |
| Step | U (Left halv) | V (Right halv) | S (Result) |
| 1 | 70 50 30 | 10 20 40 60 | 10 |
| 2 | 70 50 30 | 10 20 40 60 | 10 20 |
| 3 | 70 50 30 | 10 20 40 60 | 10 20 30 |
| 4 | 70 50 30 | 10 20 40 60 | 10 20 30 40 |
| 5 | 70 50 30 | 10 20 40 60 | 10 20 30 40 50 |
| 6 | 70 50 30 | 10 20 40 60 | 10 20 30 40 50 60 |
| 7 | 70 50 30 | 10 20 40 60 | **10 20 30 40 50 60 70 (Final values)** |

Input: Positive integer n, array S indexed from 1 to n

For this case: S = {70, 50, 30, 10, 20, 40, 60}

Output: Array S containing the keys in non-decreasing order:

{10, 20, 30, 40, 50, 60, 70}

Merge Sort Algorithm:

* Merges the two arrays U and V created by the recursive calls to mergesort
* Input size
  + h the number of items in U
  + m the number of items in V
* Basic operation:
  + Comparison of U[i] to V[i]

Void mergeSort{int n, keytype S[]) {

If (n > 1) {

const int h = [n/2];

const int m = n-h;

keytype U[1..h], V[1..m];

copy S[1] through S[h] to U[1] through U[h];

copy S[h+1] through S[n] to V[1] through V[m];

mergeSort(h, U);

mergeSort(m, V);

merge(h, U, V, S);

}

}

Graphical representation of solving the given problem with Merge Sort:

Et bilde som inneholder tekst, håndskrift, Font, nummer

KI-generert innhold kan være feil.

## Referances

Goodrich, M. T., & Tamassia, R. (2006). Data Structures and Algorithms in Java (4th ed., Ch. 11.3). Wiley.

Sedgewick, R., & Wayne, K. (2011). Algorithms (4th ed., Ch. 2.2). Addison‑Wesley.

Gupta, R. (n.d.). LO 3: Sorting Algorithms [Lecture slides]. Kristiania University College.