CS161 Notes - Merge Sort

Yuren Hao

June 27, 2025

1 Merge Sort

Divide-and-conquer sorting algorithm: recursively divide array into halves, sort each half, then merge.

Algorithm:

1. **Divide:** Split $A[1..n]$ at midpoint $\lfloor n/2 \rfloor$
2. **Conquer:** Recursively sort both halves
3. **Combine:** Merge sorted halves

2 Pseudocode

MERGE-SORT(A, p, r):

1. **if** $p < r$:
   - $q = \lfloor (p + r)/2 \rfloor$
   - **MERGE-SORT**(A, p, q)
   - **MERGE-SORT**$(A, q + 1, r)$
   - **MERGE**(A, p, q, r)

MERGE(A, p, q, r):

1. $n_1 = q - p + 1$, $n_2 = r - q$
2. Create arrays $L[1..n_1 + 1]$, $R[1..n_2 + 1]$
3. Copy $A[p..q] \to L[1..n_1]$, $A[q + 1..r] \to R[1..n_2]$
4. $L[n_1 + 1] = R[n_2 + 1] = \infty$
5. $i = j = 1$
6. **for** $k = p$ **to** r:
   - **if** $L[i] \leq R[j]$: $A[k] = L[i++]$
   - **else**: $A[k] = R[j++]$

3 Complexity Analysis

Time: $T(n) = 2T(n/2) + \Theta(n) = \Theta(n \log n)$ (Master Theorem)

Space: $\Theta(n)$ auxiliary arrays + $\Theta(\log n)$ recursion stack = $\Theta(n)$

4 Properties

Advantages: Stable, predictable $O(n \log n)$, parallelizable

Disadvantages: $O(n)$ extra space, not in-place