L19: Complexity and Algorithm Analysis

Merge Sort

\( a[ ] = 15, 6, 18, 3, 7, 17, 20, 1, 4, 13, 9 \)
MergeSort(a[], lower, upper) {
if lower < upper
    q = ⌊(low + upper)/2⌋
    MergeSort(a[], lower, q) T(n/2)
    MergeSort(a[], q+1, upper) T(n/2)
    Merge(a[], lower, q, upper) n
    return(a) 1
}

Recursion Expression:
T(1) = 1
T(n) = 2*T(n/2) + n
L19: Complexity and Algorithm Analysis

Recurrences

- **Substitution method**: guess, substitute, induction
Recurrences

Useful formulas

When you have simple recurrence relations of the form:

- \( T(n) = T(n - 1) + f(n) \), then the complexity of the algorithm is \( O(f(n) \cdot n) \).

- \( T(n) = T(n/k) + f(n) \), then the complexity of the algorithm is \( O(f(n) \cdot \log_k(n)) \).
Assignment

• Read Sections 14.6 – 14.8

• Additional readings posted on Blackboard (QuickSort-1 and QuickSort-2)