

# COMP 363 - Algorithms: Homework #1

Due on 22 January, 2026

*Leo Irakliotis*

**Nathan Hogg**

## Problem 1

### Background

The goal of this assignment is to prove the time complexity of the mergesort algorithm. We are given the base complexity formula as:

$$T(n) = 2T(n/2) + f(n) \quad (1)$$

In this case,  $T(n)$  is the time it takes to mergesort an array of size  $n$ , and  $f(n)$  is the time it takes to assemble two partial solutions of size  $n/2$  to a sorted array with  $n$  elements. Generally,  $f(n) \approx n$ .

For an array with  $n = 8$ , we can write:

$$T(8) = 2T(4) + f(8) \quad (2)$$

Then,  $T(4) = 2T(2) + f(4)$ ; and  $T(2) = 2T(1) + f(2)$ ; and finally  $T(1) = f(1)$ . Using the last finding, we can solve backwards:

$$T(2) = 2T(1) + f(2) \quad (3)$$

$$= 2f(1) + f(2) \quad (4)$$

$$T(4) = 2T(2) + f(4) \quad (5)$$

$$= 2(2f(1) + f(2)) + f(4) \quad (6)$$

$$= 4f(1) + 2f(2) + f(4) \quad (7)$$

$$T(8) = 2T(4) + f(8) \quad (8)$$

$$= 2(4f(1) + 2f(2) + f(4)) + f(8) \quad (9)$$

$$= 8f(1) + 4f(2) + 2f(4) + f(8) \quad (10)$$

Given that  $f(n) \equiv n$ , the last equation can be rewritten as:

$$T(8) \equiv 8 \times 1 + 4 \times 2 + 2 \times 4 + 8 \quad (11)$$

$$= 32 = 8 \times 4 \quad (12)$$

$$= 8(1 + \log_2 8) \quad (13)$$

We have shown that, for mergesort,  $T(n) \equiv n \log_2 n$  (or,  $\mathcal{O}(n \log_2 n)$ ),

### Proof

We begin by writing:

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + f(n) \\ &= 2 \left[ 2T\left(\frac{n}{4}\right) + f\left(\frac{n}{2}\right) \right] + f(n) \\ &= 2 \left[ 2 \left[ 2T\left(\frac{n}{8}\right) + f\left(\frac{n}{4}\right) \right] + f\left(\frac{n}{2}\right) \right] + f(n) \\ &= 2 \left[ 2 \left[ 2 \left[ 2T\left(\frac{n}{16}\right) + f\left(\frac{n}{8}\right) \right] + f\left(\frac{n}{4}\right) \right] + f\left(\frac{n}{2}\right) \right] + f(n) \end{aligned}$$