# CPSC-354 Report

Gabriel Giancarlo
Chapman University

October 13, 2025

**Abstract**

This report documents my work on termination analysis through measure functions. I examined two classic algorithms: the Euclidean algorithm for computing greatest common divisors and merge sort for array sorting. Both examples demonstrate how to construct appropriate measure functions to prove algorithm termination.

## Contents



# 1 Introduction

Termination analysis is a fundamental problem in computer science. Given an algorithm, how can we prove that it will always halt? This question is crucial for ensuring the correctness and reliability of programs.

The key insight is to find a **measure function** that decreases with each step of the algorithm. If we can show that this measure is always non-negative and strictly decreases, then the algorithm must terminate because we cannot have an infinite sequence of decreasing non-negative integers.

# 2 Week by Week

## 2.1 Week 1: Termination Analysis

### 2.1.1 Problem 4.1: Euclidean Algorithm

Consider the following algorithm:

```
while b != 0:
    temp = b
```

```
    b = a mod b
    a = temp
return a
```

Under certain conditions (which?) this algorithm always terminates.

Find a measure function and prove termination.

### 2.1.2 Problem 4.2: Merge Sort

Consider the following fragment of an implementation of merge sort:

```
function merge_sort(arr, left, right):
    if left >= right:
        return
    mid = (left + right) / 2
    merge_sort(arr, left, mid)
    merge_sort(arr, mid+1, right)
    merge(arr, left, mid, right)
```

Prove that

$$\varphi(left, right) = right - left + 1$$

is a measure function for `merge_sort`.

# 3 Essay

Working through these termination proofs was an excellent exercise in mathematical rigor. The Euclidean algorithm example was particularly instructive because it demonstrates how a simple measure function can capture the essential property that guarantees termination.

The key insight for the Euclidean algorithm was recognizing that $b$ itself serves as a natural measure. Since $a \bmod b < b$ when $b \neq 0$, the value of $b$ strictly decreases with each iteration. This is a beautiful example of how the mathematical properties of the operations (in this case, the properties of modular arithmetic) directly provide the termination guarantee.

For merge sort, the challenge was different. Here, the measure function $\varphi(left, right) = right - left + 1$ represents the size of the subarray being sorted. The recursive calls operate on strictly smaller subarrays, so the measure decreases with each recursive call. This demonstrates how divide-and-conquer algorithms naturally provide their own termination guarantees through the shrinking of problem size.

These exercises highlighted the importance of choosing the right measure function. The measure must capture some essential property of the algorithm's state that changes in a predictable way. In both cases, the measure function was closely related to the algorithm's progress toward its goal.

# 4 Evidence of Participation

I completed both termination analysis problems:

- **Problem 4.1:** Analyzed the Euclidean algorithm, identified the measure function $\varphi(a, b) = b$, and proved termination using the properties of modular arithmetic.
- **Problem 4.2:** Analyzed merge sort recursion, verified that $\varphi(left, right) = right - left + 1$ is a valid measure function, and proved termination using the well-foundedness of natural numbers.

Each solution included:

- Formal definition of the measure function
- Proof of non-negativity
- Proof that the measure decreases with each step
- Application of well-foundedness to conclude termination

The solutions demonstrate understanding of measure functions, well-founded relations, and their application to proving algorithm termination.

# 5 Conclusion

These termination analysis exercises provided valuable insight into the mathematical foundations of algorithm correctness. The key lessons learned include:

- Measure functions provide a powerful tool for proving termination
- The choice of measure function is crucial and often relates to the algorithm's progress
- Mathematical properties of operations (like modular arithmetic) can directly provide termination guarantees
- Well-founded relations are the theoretical foundation underlying termination proofs

Understanding termination is essential for writing reliable software. These exercises have improved my ability to reason formally about algorithm behavior and to construct rigorous proofs of program properties.

# References

[Term] Nachum Dershowitz and Zohar Manna, Proving Termination with Multiset Orderings, Communications of the ACM, 1979.

[Euclid] Donald Knuth, The Art of Computer Programming, Addison-Wesley, 1997.