Formal Proof of Correctness: OverflowSort Algorithm

Scott Douglass

July 10, 2025
DOI: 10.5281/zenodo.15855899

Abstract

OverflowSort is a non-comparison-based sorting algorithm that simulates overflow detection as a selection mechanism. This document provides a proof sketch of its correctness and a discussion of its time complexity bounds under idealized and practical conditions.

1 Preliminaries

Given an array A of n positive integers, OverflowSort repeatedly multiplies each value in A by a constant factor (e.g., 2) until it "overflows" a fixed threshold T. The value that overflows first is assumed to be the largest.

Let:

- x_i be the i-th element of A
- k_i be the number of multiplications needed such that $x_i \cdot 2^{k_i} > T$
- The element with the smallest k_i is selected first

2 Correctness

We prove that if all x_i are distinct and positive, then for fixed T, the sorting order determined by smallest k_i is the same as descending numerical order.

Lemma

For any x_i, x_j with $x_i > x_j$, then $k_i < k_j$.

Proof:

Assume $x_i > x_j$. Then:

$$x_i \cdot 2^{k_i} > T \quad \text{and} \quad x_j \cdot 2^{k_j} > T$$

To overflow, we require:

$$2^{k_i} > \frac{T}{x_i}, \quad 2^{k_j} > \frac{T}{x_j}$$

Since $x_i > x_j$, it follows that:

$$\frac{T}{x_i} < \frac{T}{x_j} \Rightarrow 2^{k_i} < 2^{k_j} \Rightarrow k_i < k_j$$

■

3 Time Complexity

Each iteration simulates an overflow check for all n elements, bounded by $\log_2(T/x_{\min})$ rounds in the worst case. If a tick map is used to detect overflow with a single pass:

- Best-case time: $\mathcal{O}(n)$
- Worst-case time: $\mathcal{O}(n \log M)$ where M is the maximum value

Citation

If you use or reference this work, please cite:

Scott Douglass. *OverflowSort: Overflow-Based Sorting Algorithms.* Zenodo. DOI: 10.5281/zenodo.15855899. Version 1.0.0, July 2025.