Lab 01: Analyzing Linear and Logarithmic-Time Algorithms for Unimodal Search

Andrés Oswaldo Calderón Romero

July 30, 2025

1 Introduction

The goal of this lab is to deepen your understanding of algorithmic design and analysis by studying the *unimodal search* problem, a classic example that illustrates the trade-offs between linear and logarithmic-time approaches. You will begin by developing a straightforward $O(n)$ algorithm that performs a single linear scan to locate the peak element in a unimodal array. This serves as a baseline for analyzing the performance of more sophisticated techniques.

Next, you will design and implement an optimized $O(\log n)$ algorithm based on a divide-and-conquer strategy. Beyond implementation, you will formally prove the correctness of your approach and evaluate its efficiency through empirical experiments. By comparing both solutions, you will gain insight into asymptotic complexity, practical performance differences, and the importance of algorithmic choice when solving real-world problems.

2 Problem Definition

An array $A[1 \ldots n]$ is **unimodal** (also known as **bitonic**) if it consists of an increasing sequence followed by a decreasing sequence. More precisely, there exists an index $m \in \{1, 2, \ldots, n\}$ such that:

$$A[1] < A[2] < \ldots < A[m]$$

and

$$A[m] > A[m+1] > \ldots > A[n],$$

where $A[m]$ is the **peak** of the array. This means that the values strictly increase up to m and then strictly decrease afterward. The task of **unimodal search** is to efficiently find the peak element $A[m]$ in such an array.

In particular, $A[m]$ is the maximum element and the unique "locally maximum" element, meaning it is surrounded by smaller elements, specifically $A[m-1]$ and $A[m+1]$.

2.1 Problem to Solve

1. Implement a naive algorithm to compute the maximum element of a unimodal input array $A[1 \ldots n]$ in $O(n)$ time. This algorithm should perform a single linear pass over the array to find the maximum.

2. Design and implement an algorithm to compute the maximum element of a unimodal input array $A[1 \ldots n]$ in $O(\log n)$ time. Prove the correctness of your algorithm and establish a bound on its running time.

3. Empirically compare both implementations to evaluate the performance of each algorithm.

4. Submit your findings in a well-structured report, including the code used and any additional relevant materials.

3 What We Expect

You must submit a well-structured report in **PDF** format, along with the code for your implementations and any additional materials used to solve the problem, packaged in a single **ZIP** file. The submission must be completed by **August 14, 2025**, using the link that will be available on the Brightspace™ platform.

4 Useful Resources

The problem was initially stated in Problem Set 1 (Problem 1-3), and a solution for item 2 in Section 2.1 is provided in the Problem Set 1 Solutions available through MIT OpenCourseWare, taught by Prof. Leiserson and Prof. Demaine [1, 2]. Please attempt to solve the problem on your own first and then verify your solution with the one provided. However, if you find yourself stuck after a reasonable amount of time, you may consult the solution as a last resort.

Similarly, plenty of code can be found on the Internet or generated by AI, but—as mentioned earlier—make an honest attempt on your own before seeking external sources. This webpage [3] may also be a useful resource if you find yourself struggling during the implementation.

References

[1] C. L. . E. Demaine, "Introduction to algorithms (sma 5503)." `https://ocw.mit.edu/courses/6-046j-introduction-to-algorithms-sma-5503-fall-2005/resources/ps1/`, 2005. Accessed on 05-02-2025.

[2] C. L. . E. Demaine, "Introduction to algorithms (sma 5503)." `https://ocw.mit.edu/courses/6-046j-introduction-to-algorithms-sma-5503-fall-2005/resources/ps1sol/`, 2005. Accessed on 05-02-2025.

[3] geeksforgeeks.org, "Find bitonic point in given bitonic sequence." `https://www.geeksforgeeks.org/find-bitonic-point-given-bitonic-sequence/`, 2023. Accessed on 05-02-2025.