Lab 1

Andrés Calderón

February 5, 2025

1 Introduction

The primary objective of this lab is to conduct an empirical analysis of two algorithms that solve the *unimodal search* problem. You are expected to implement both algorithms and evaluate their performance using a programming language of your choice. The goal is to understand the asymptotic complexity of each algorithm and empirically verify their theoretical behavior.

2 Problem Definition

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

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

and

$$A[m] > A[m+1] > \dots > A[n]$$

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 it is 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 $A[1 \dots n]$ in $O(n)$ time. This algorithm should perform a single linear pass over the array until it finds the maximum.

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

3. Empirically compare both implementations to evaluate the overall 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 should submit a digital copy of your report in **PDF** format, along with the code for your implementations and any additional materials used to solve the problem, in a **ZIP** file. The submission must be completed by Thursday, **February 13th, 2025**, before the next lab, 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 in the MIT OpenCourseWare, taught by Prof. Leiserson and Prof. Demaine [1, 2]. Please try to solve the problem on your own first and then verify your solution with the provided one. However, if you find yourself stuck after some 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 searching for external sources. This webpage [3] may be a useful resource if you find yourself lost 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.