

# Searching and Sorting Algorithms

Madiba Hudson-Quansah

# CONTENTS

## CHAPTER 1

### SEARCHING ALGORITHMS \_\_\_\_\_ PAGE 2 \_\_\_\_\_

- 1.1 Linear / Sequential Search \_\_\_\_\_ 2  
Input / Output (IO) – 2 • Algorithm – 2 • Time Complexity – 2
- 1.2 Binary Search \_\_\_\_\_ 2  
Input / Output (IO) – 2 • Algorithm – 3

## CHAPTER 2

### SORTING ALGORITHMS \_\_\_\_\_ PAGE 4 \_\_\_\_\_

# Chapter 1

## Searching Algorithms

### 1.1 Linear / Sequential Search

The linear search algorithm is a simple search algorithm that uses a brute force approach to locate a single target value  $t$  in a collection  $A$ .

#### 1.1.1 Input / Output (IO)

**Input** A non-empty collection  $A$  of  $n > 0$  elements and a target value  $t$ .

**Output** The index of the target value  $t$  in the collection  $A$  if it exists, otherwise  $-1$

#### 1.1.2 Algorithm

---

**Algorithm 1** LinearSearch( $A, n, t$ )

---

```
1: for  $i = 1$  to  $n$  do
2:   if  $A_i = t$  then
3:     return  $i$ 
4:   end if
5:   return  $-1$ 
6: end for
```

---

#### 1.1.3 Time Complexity

**Best Case** The best case for this algorithm is when the target value is at the first index of the collection  $A$ , for which the time complexity is  $O(1)$  or more specifically  $\Theta(1)$  as the algorithm will always use a constant amount of time to locate the target value.

**Worst Case** The worst case for this algorithm is when the target value is not found or the target value is at the last index of the collection  $A$ , for which the time complexity is  $O(n)$ .

### 1.2 Binary Search

The binary search algorithm is a more efficient search algorithm that uses a divide and conquer approach to locate a single target value  $t$  in a sorted collection  $A$ .

#### 1.2.1 Input / Output (IO)

**Input** A sorted non-empty collection  $A$  of  $n > 0$  elements and a target value  $t$ .

**Output** The index of the target value  $t$  in the collection  $A$  if it exists, otherwise  $-1$

### 1.2.2 Algorithm

---

**Algorithm 2** BinarySearch( $A, n, t$ )

---

```
1: low := 0
2: high := n
3: while low  $\leq$  high do
4:   mid :=  $\left\lfloor \frac{(\text{high} + \text{low})}{2} \right\rfloor$ 
5:   if  $t = A_{\text{mid}}$  then
6:     return mid
7:   end if
8:   if  $t < A_{\text{mid}}$  then
9:     high = mid - 1
10:  else if  $t > A_{\text{mid}}$  then
11:    low = mid + 1
12:  end if
13: end while
14: return -1
```

---

## **Chapter 2**

# **Sorting Algorithms**