Searching and Sorting Algorithms

Madiba Hudson-Quansah

Contents

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)

```
for $i = 1$ to $n$ do
    if $A_i = t$ then
        return $i$
    end if
    return $-1$
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)

```
low := 0
high := n
while low $\leqslant$ high do
    mid := $\left\lfloor \frac{\text{(high+low)}}{2} \right\rfloor$
    if $t = A_{\text{mid}}$ then
        return mid
    end if
    if $t < A_{\text{mid}}$ then
        high = mid - 1
    else if $t > A_{\text{mid}}$ then
        low = mid + 1
    end if
end while
return -1
```

Chapter 2

Sorting Algorithms