Algorithm Analysis

Josh Wilcox (jw14g24@soton.ac.uk)

February 25, 2025

Contents

1 Pseudocode

1.1 The need for pseudocode

- Pesudocode replaces precise languages such as Java which have repetitive and language specific details
- They are not **language dependent**
- Pseudocode is not typically concerned with software engineering - more logic and analysis

1.2 Rule of Thumb

- To be easily interpreted
- Needs to be consistent
- Shouldn't be *too* different from standard code
- Doesn't have a set standard however:
  - Arrays are often written in bold font

1.3 Assignment

$$x = x + 1$$

- Mathemeticians die after seeing this
  - $x = x + 1$ has no solution
- So to appease mathematicians who struggle to understand assignment, we use "$\leftarrow$"
- Multiplication we use either (whatever is most reasonable and consistent)
  - $x \cdot y$ - Can be confused with dot product
  - $x \times y$ - Can be confused with cross product
  - Or just xy - Might get messy

Algorithm 1 An algorithm with caption

Require: $n \geq 0$
Ensure: $y = x^n$

```
  y ← 1
  X ← x
  N ← n
  while N ≠ 0 do
      if N is even then
          X ← X × X
          N ← N/2                              ▷ This is a comment
      else if N is odd then
          y ← y × X
          N ← N − 1
      end if
  end while
```

2 Simple Searching

2.1 Sequential Search

Algorithm 2 Sequential Search Algorithm

```
for i ← 1 to a.length do
    if a[i] = x then
        return true
    end if
end for
return false
```

2.1.1 Best Case Complexity

- First element is the item we're searching for
- Need one comparion
- Therefore $\Theta(1)$

2.1.2 Worst Case Complexity

- The last element is the item we're searching for
- Takes n comparisons to find
- Therefore $\Theta(n)$

2.1.3 Average Case Complexity

- Number of comparisons

$$\frac{1}{n}\sum_{i=1}^{n} = \frac{n+1}{2}$$

- Therefore, $\Theta(n)$

2.2 Binary Search

- Requires the list to be **sorted!!**
- Pick Midpoint
  - If x is equal - STOP
  - If x is smaller - search in the left part
  - If x is greater - search in the right part
  - If the range of search is empty - STOP

Algorithm 3 Binary Search Algorithm

Require: **a** is a sorted array
Require: x is the element to search for
  $low \leftarrow 1$
  $high \leftarrow$ **a**.length
  while $low \leq high$ **do**
    $mid \leftarrow \lfloor \frac{low+high}{2} \rfloor$
    if **a**$[mid] = x$ **then**
      return true
    else if **a**$[mid] < x$ **then**
      $low \leftarrow mid + 1$
    else
      $high \leftarrow mid$ - 1
    end if
  end while
  return false

2.2.1 Best Case Complexity

- The element is at the midpoint on the first check
- Therefore $\Theta(1)$

2.2.2 Worst Case Complexity

- The element is not in the list or is at the end of the search range
- Takes $\log_2(n)$ comparisons to find
- Therefore $\Theta(\log n)$

2.2.3 Average Case Complexity

- On average, the element will be found in the middle of the search range
- Takes $\log_2(n)$ comparisons on average
- Therefore $\Theta(\log n)$