

OCR Computer Science AS Level

2.3.1 Searching Algorithms

Concise Notes



Specification:

- Standard algorithms
 - Binary search
 - Linear search



Searching Algorithms

- Used to **find a specified element** within a data structure
- Numerous different algorithms exist, each of which is **suited to a particular data structure** of **format of data**
- Different algorithms are used depending on each individual scenario

Binary Search

- Can only be applied on **sorted data**
- Works by finding the **middle element** in a list of data before deciding **which side** of the data the desired element is to be found in
- The unwanted half of the data is **discarded** and the process **repeated** until either
 - The desired element is **found**
 - Or it is known that the desired element **doesn't exist** in the data
- With each iteration, **half of the input data** is discarded
- The algorithm is very **efficient**
- The time complexity of binary search is $O(\log n)$

A = Array of data

x = Desired element

```
low = 0
high = A.length - 1
while low <= high:
    mid = (low + high) / 2
    if A[mid] == x:
        return mid
    else if A[mid] > x:
        high = mid - 1
    else:
        low = mid + 1
    endif
endwhile
return "Not found in data"
```



Linear Search

- The **most basic** searching algorithm
- Looks at elements one at a time until the desired element is found
- Doesn't require the data to be sorted
- A great deal of **pot luck**, but **easy** to implement
 - Sometimes **gets lucky** and finds the desired element almost **immediately**
 - In other situations, the algorithm is **incredibly inefficient**
- Time complexity of $O(n)$

A = Array of data

x = Desired element

```
i = 0
while i < A.length:
    if A[i] == x:
        return i
    else:
        i = i + 1
    endif
endwhile
return "Not found in data"
```

