3.1.3 Searching algorithms 1

Lesson plan and printable activities

Teacher notes

There are many searching and sorting algorithms. This lesson will focus on the linear search.

For small data sets, it makes little difference which algorithm is used for either searching or sorting.

Where record numbers are very large, eg the volume of records that might be held by a national organisation's database then there are definite performance implications.

It may be helpful to stress that the topics being studied here make most difference when the number of items being searched runs into hundreds of thousands or millions.

For background information:

Linear search: wikipedia.org/wiki/Linear_search

Efficiency of algorithms: wikipedia.org/wiki/Algorithmic_efficiency

Materials needed

1. 3.1.3 Lesson 1 PowerPoint.
2. Following a linear search algorithm worksheet.

Lesson aims

1. To get students to think about the basic principles of searching and sorting.
2. To see how one specific form of computerised searching works.

Lesson objectives

1. Understand and explain how the linear search algorithm works.

Starter activity (5 minutes)

1. **Slides 3–5:** Start by explaining what is meant by the concepts of searching and sorting.
2. **Slides 6–7:** Get the students to think about the large number of records that might be needed to be searched, eg how many customer records might a bank hold?

Main activities (30 minutes)

1. **Slide 8:** YouTube video. Watch from 1:14 minutes to 3:11 minutes.

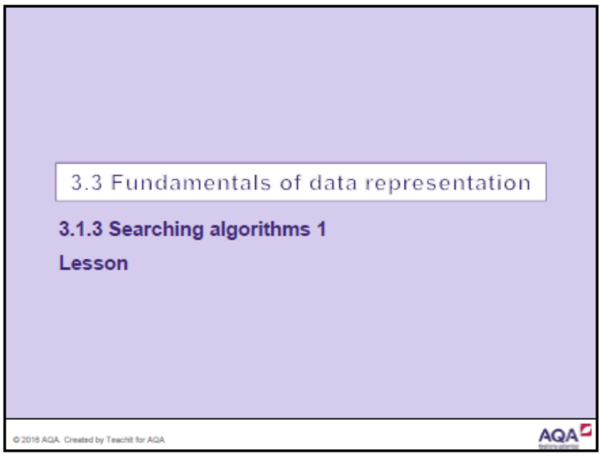
2. **Slides 9–14:** Introduce the linear search algorithm. Students should already have been introduced to pseudo-code in 3.1.1 Representing algorithms. They may not have covered 3.2 Programming yet so the algorithm may need some explanation. Make clear that the algorithm contains a loop that waits until a set condition is satisfied before exiting and that the loop may be entered many times before match is made.

3. **Slide 15:** Emphasise that the algorithm is simple and inefficient where large volumes of data are being searched and that there are other more efficient algorithms available – we will be studying a better one in the next lesson. In general, the simpler the search algorithm, the less efficient the search process is likely to be.

Plenary activity (10 minutes)

1. Use the opportunity to get the students to practise their skills in following an algorithm using the 'Following a linear search algorithm' worksheet. Do not spend more than 10 minutes on this as students will have other opportunities during the course to practise this skill.

Lesson

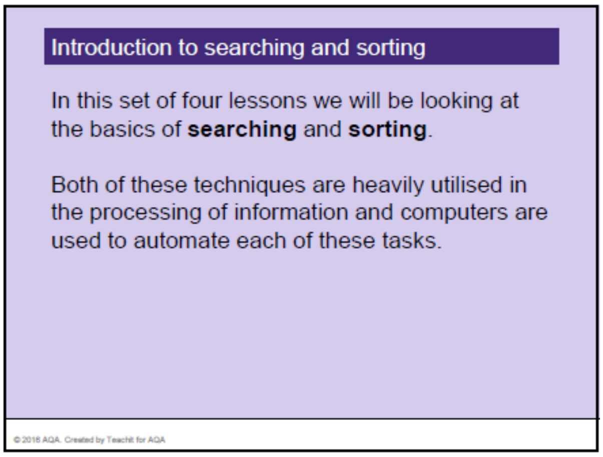


3.3 Fundamentals of data representation

3.1.3 Searching algorithms 1

Lesson

© 2016 AQA. Created by Teachit for AQA

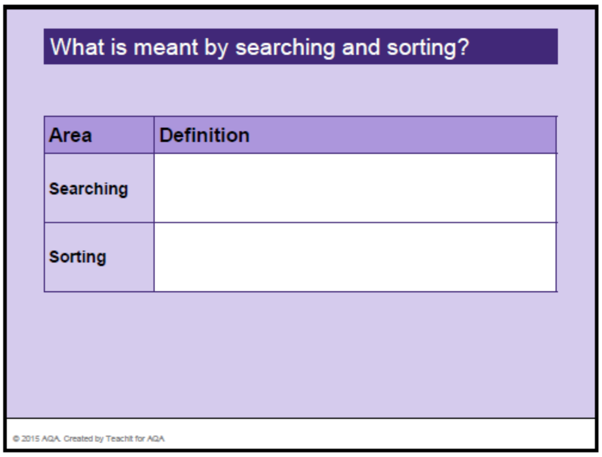


Introduction to searching and sorting

In this set of four lessons we will be looking at the basics of **searching** and **sorting**.

Both of these techniques are heavily utilised in the processing of information and computers are used to automate each of these tasks.

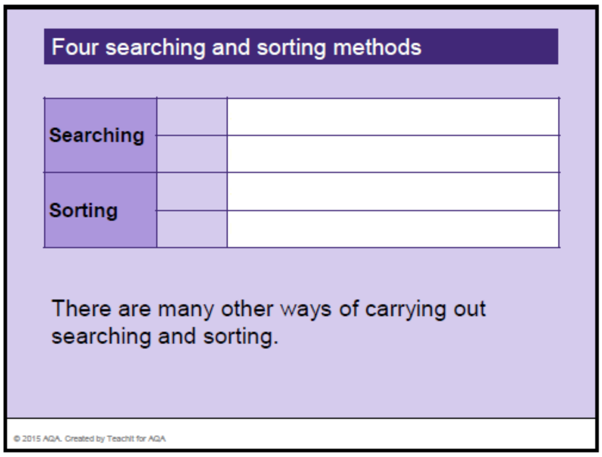
© 2016 AQA. Created by Teachit for AQA



What is meant by searching and sorting?

Area	Definition
Searching	
Sorting	

© 2015 AQA. Created by Teachit for AQA

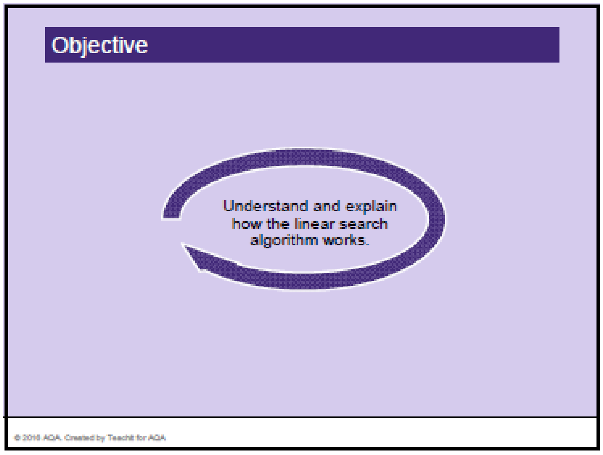


Four searching and sorting methods

Searching		
Sorting		

There are many other ways of carrying out searching and sorting.

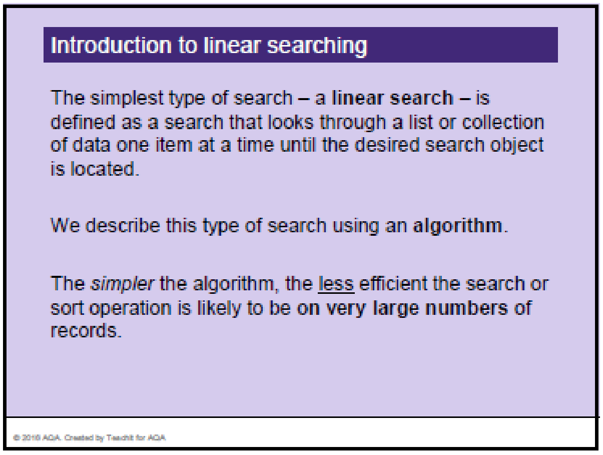
© 2015 AQA. Created by Teachit for AQA



Objective

Understand and explain how the linear search algorithm works.

© 2016 AQA. Created by Teachit for AQA



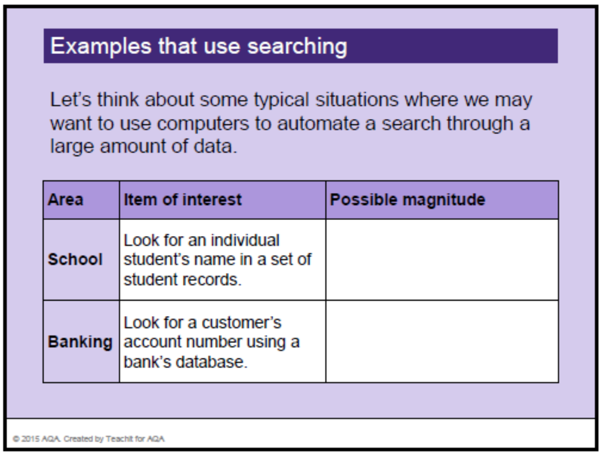
Introduction to linear searching

The simplest type of search – a **linear search** – is defined as a search that looks through a list or collection of data one item at a time until the desired search object is located.

We describe this type of search using an **algorithm**.

The *simpler* the algorithm, the less efficient the search or sort operation is likely to be **on very large numbers of** records.

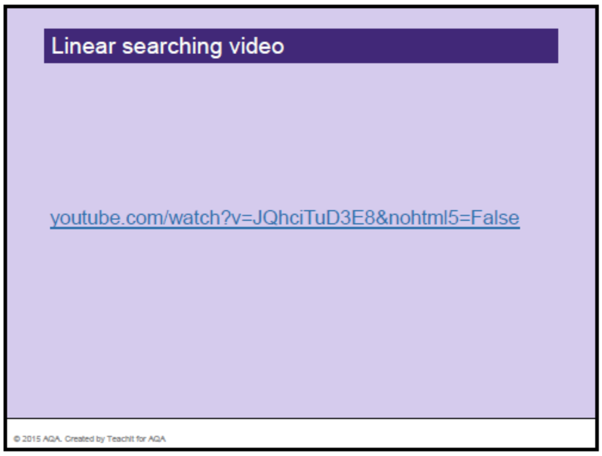
© 2016 AQA. Created by Teachit for AQA



Examples that use searching

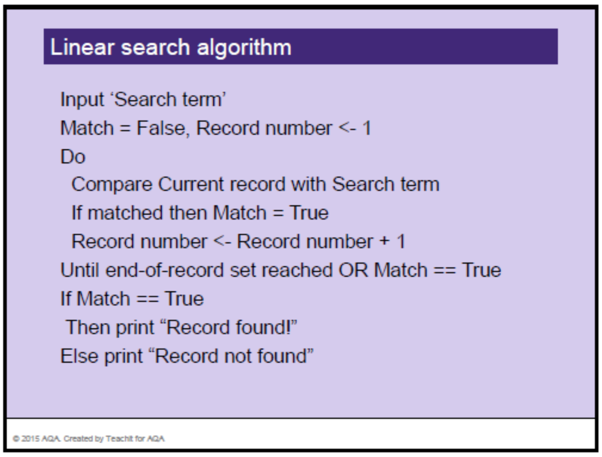
Let's think about some typical situations where we may want to use computers to automate a search through a large amount of data.

Area	Item of interest	Possible magnitude
School	Look for an individual student's name in a set of student records.	
Banking	Look for a customer's account number using a bank's database.	



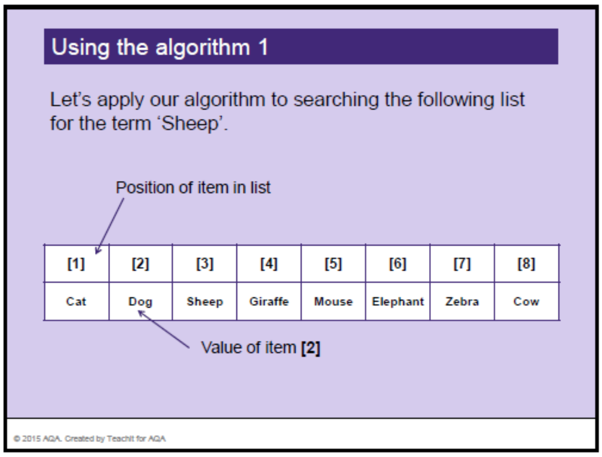
Linear searching video

youtube.com/watch?v=JQhciTuD3E8&nohtml5=False



Linear search algorithm

```
Input 'Search term'
Match = False, Record number <- 1
Do
  Compare Current record with Search term
  If matched then Match = True
  Record number <- Record number + 1
Until end-of-record set reached OR Match == True
If Match == True
 Then print "Record found!"
Else print "Record not found"
```



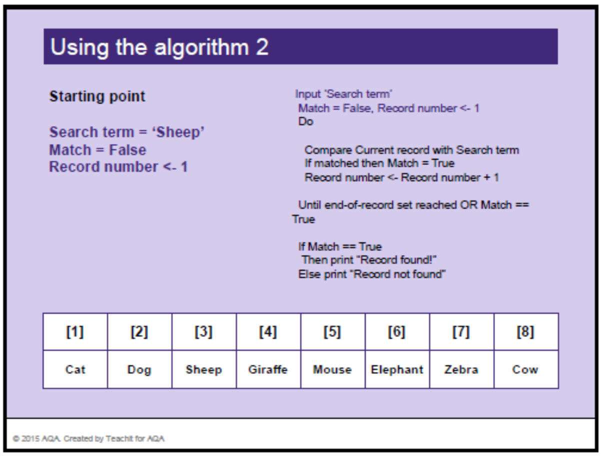
Using the algorithm 1

Let's apply our algorithm to searching the following list for the term 'Sheep'.

Position of item in list

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
Cat	Dog	Sheep	Giraffe	Mouse	Elephant	Zebra	Cow

Value of item **[2]**



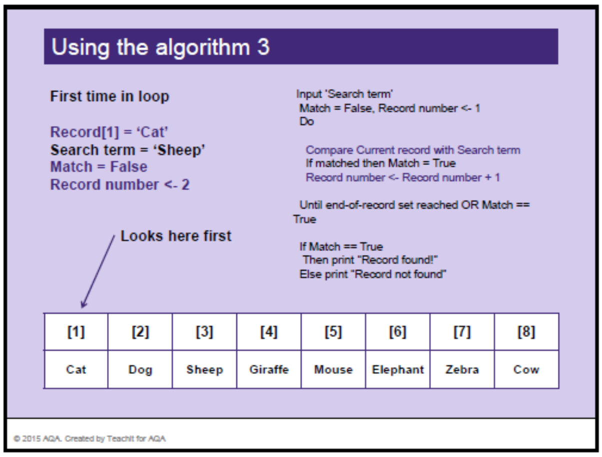
Using the algorithm 2

Starting point

Search term = 'Sheep'
Match = False
Record number <- 1

```
Input 'Search term'
 Match = False, Record number <- 1
 Do

  Compare Current record with Search term
  If matched then Match = True
  Record number <- Record number + 1

 Until end-of-record set reached OR Match ==
True

 If Match == True
  Then print "Record found!"
 Else print "Record not found"
```

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
Cat	Dog	Sheep	Giraffe	Mouse	Elephant	Zebra	Cow



Using the algorithm 3

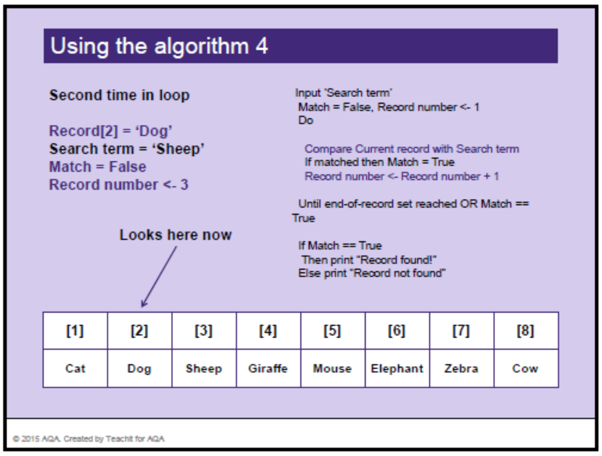
First time in loop

Record[1] = 'Cat'
Search term = 'Sheep'
Match = False
Record number <- 2

```
Input 'Search term'
 Match = False, Record number <- 1
 Do

  Compare Current record with Search term
  If matched then Match = True
  Record number <- Record number + 1

 Until end-of-record set reached OR Match ==
True

 If Match == True
  Then print "Record found!"
 Else print "Record not found"
```

Looks here first

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
Cat	Dog	Sheep	Giraffe	Mouse	Elephant	Zebra	Cow



Using the algorithm 4

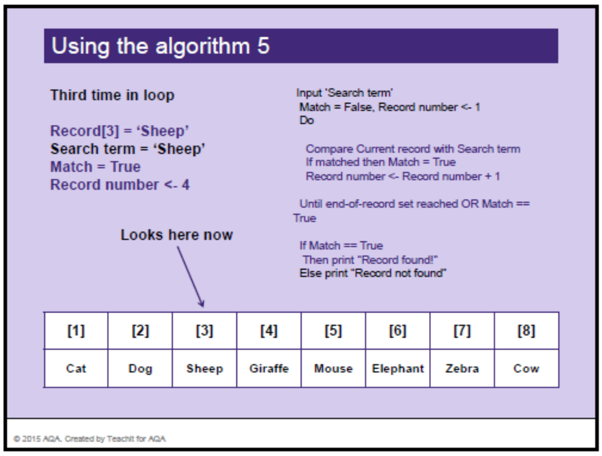
Second time in loop

Record[2] = 'Dog'
Search term = 'Sheep'
Match = False
Record number <- 3

```
Input 'Search term'
 Match = False, Record number <- 1
 Do

   Compare Current record with Search term
   If matched then Match = True
   Record number <- Record number + 1

 Until end-of-record set reached OR Match ==
True

 If Match == True
  Then print "Record found!"
 Else print "Record not found"
```

Looks here now

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
Cat	Dog	Sheep	Giraffe	Mouse	Elephant	Zebra	Cow



Using the algorithm 5

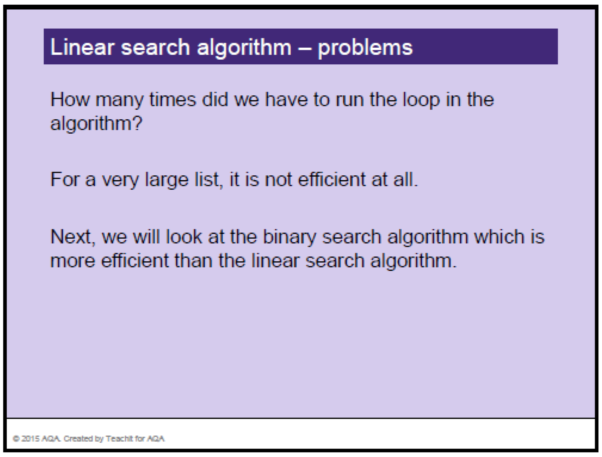
Third time in loop

Record[3] = 'Sheep'
Search term = 'Sheep'
Match = True
Record number <- 4

```
Input 'Search term'
 Match = False, Record number <- 1
 Do

   Compare Current record with Search term
   If matched then Match = True
   Record number <- Record number + 1

 Until end-of-record set reached OR Match ==
True

 If Match == True
  Then print "Record found!"
 Else print "Record not found"
```

Looks here now

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
Cat	Dog	Sheep	Giraffe	Mouse	Elephant	Zebra	Cow

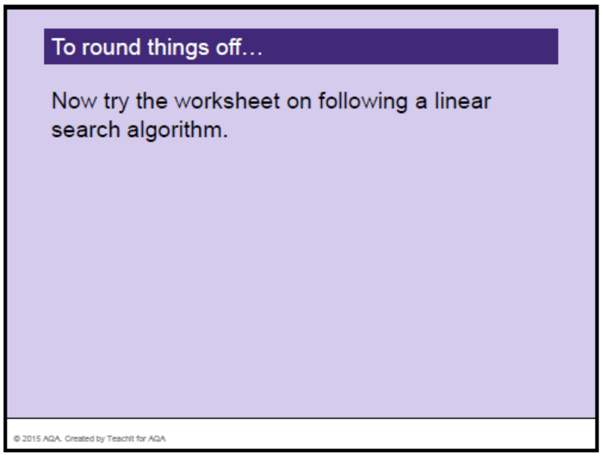


Linear search algorithm – problems

How many times did we have to run the loop in the algorithm?

For a very large list, it is not efficient at all.

Next, we will look at the binary search algorithm which is more efficient than the linear search algorithm.



To round things off…

Now try the worksheet on following a linear search algorithm.

Following a linear search algorithm

Follow this algorithm that allows you to find the position of the term ‘Fred’ in the following indexed array of strings.

```
Input ‘Search term’

  Match = False
  Record number <- 1
  Record position <- NULL

  Do
    Compare Current record with Search term
    If matched then Match = True, Record position = Record number
    Record number <- Record number + 1
  Until end-of-record set reached OR Match == True

  If Match == True AND Record position != NULL
   Then print “Record found at position “ ‘Record number’
  Else print “Record not found”
```

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
Joe	**Alan**	**Masie**	**Fred**	**John**	**Ahmed**	**Joelle**	**June**