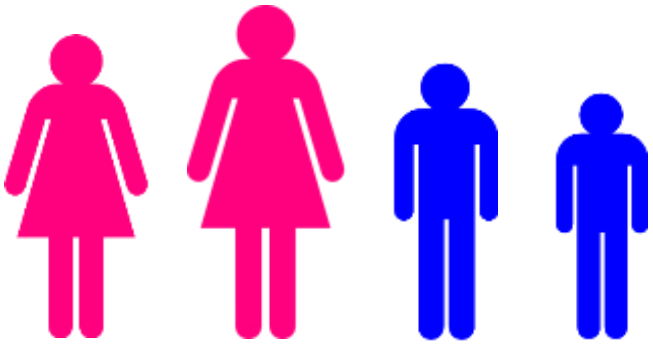
Year 8 Computing
Sorting and Searching Algorithms

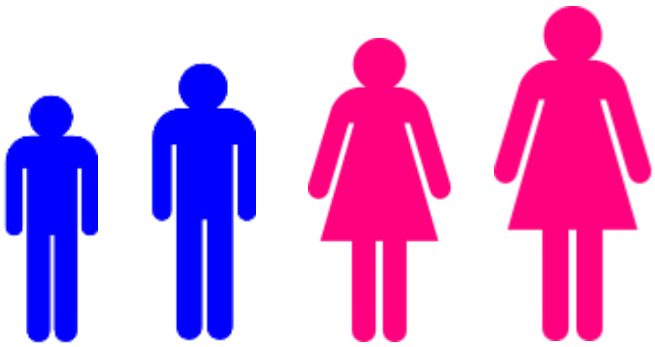
Name:

Lesson 1

In this lesson, we are learning how to use a **selection sort** algorithm to get from this…



to this…

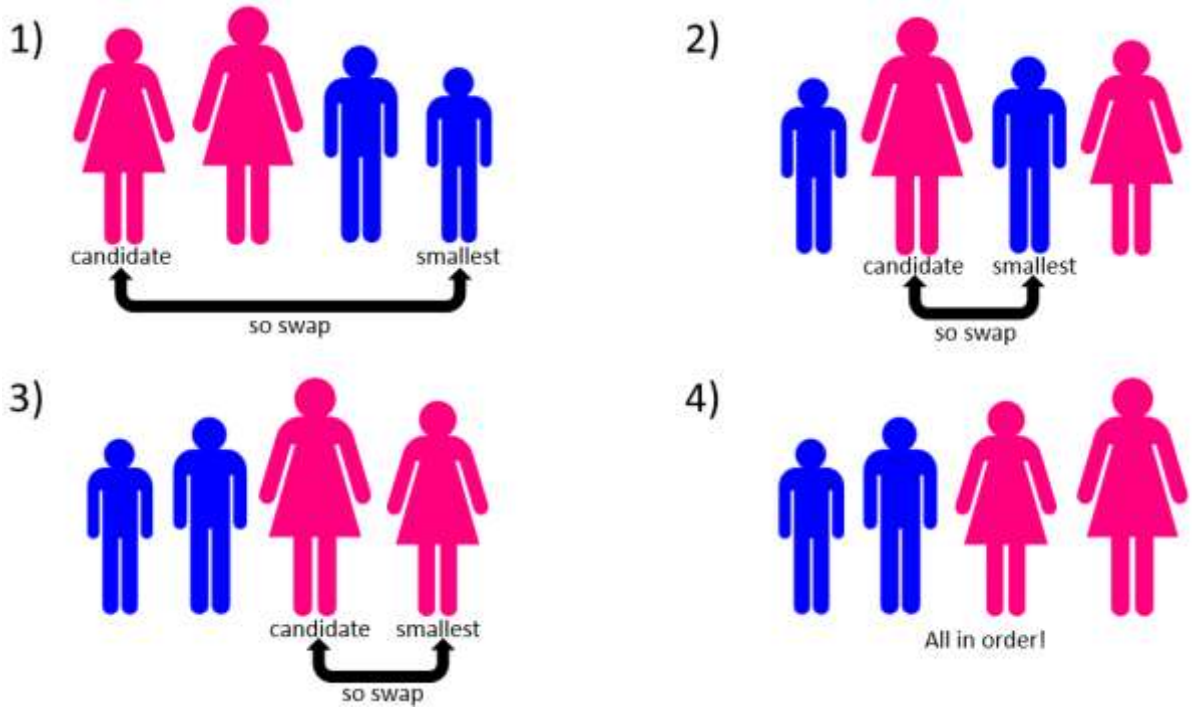


You will see from the pictures above that the figures have been organised into order of height. I have used a specific set of step-by-step instructions – an algorithm – to organise the figures into order moving from the left to the right.

The algorithm I used is called a **selection sort** and it goes like this:

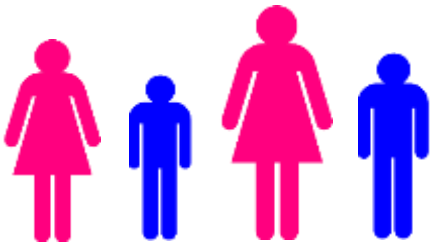
1. Start with the first figure in the unsorted list – this is known as the candidate.
2. Moving to the right, compare the candidate with the other figures. Swap the candidate with the smallest figure found. If no smaller figure is found, no swap is made.
3. The second figure now becomes the candidate. Repeat Step 2.
4. The third figure now becomes the candidate. Repeat Step 2.
5. The figure should now be in order!

You will see an illustration of sorting the figures on the next page.



Challenge

Have a go at sorting the following figures (draw the figures at the different stages of the selection sort, like in the example above):



Lesson 2

Searching for data can be very difficult, especially if there is a lot of data to search through! **Searching algorithms** make the process of searching for data much easier.

In this lesson, we will be looking at the **serial search** algorithm.

A serial search is possibly the most basic kind of search algorithm. At the start of the algorithm, we need to know what the search **criteria** is (i.e. the value that we are looking for).

The search then starts with the first item of data and then moves to next item in turn, until:

1) a match is found

OR

2) the search reaches the end of the data with no match found.

Example

Imagine that 'Mutts' is a national competition for dogs, similar to Crufts. Only one dog from each breed is entered into the competition. The organisers keep a list of the dog breeds that have been entered for the competition. An owner of a 'Boxer' wants to enter so the organisers want to check if there is already a 'Boxer' in the list. If there is a match the dog cannot be entered. If there is no match, the dog breed is added to the list. The current list is as follows:

The following steps are taken in the serial search of the dog breeds list:

Step 1: The search criteria is defined as 'Boxer'.

Step 2: The first dog breed in the list ('Bull Terrier') is compared with 'Boxer' to see if a match is found. There is no match so the search moves to the next dog breed in the list.

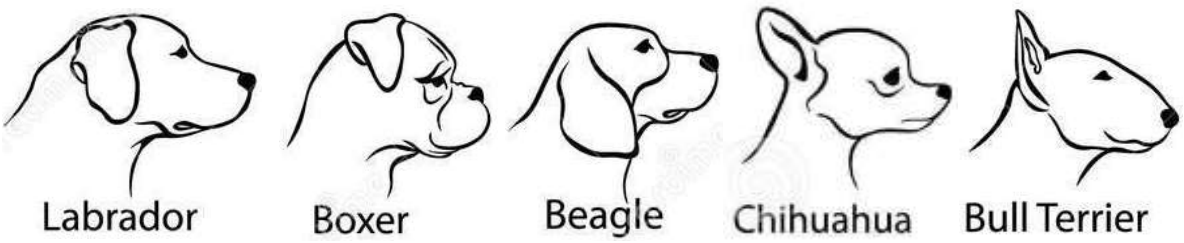
Step 3: The second dog breed in the list ('Chihuahua') is compared with 'Boxer' to see if a match is found. There is no match so the search moves to the next dog breed in the list.

Step 4: The third dog breed in the list ('Chihuahua') is compared with 'Boxer' to see if a match is found. There is no match so the search moves to the next dog breed in the list.

Step 5: The fourth dog breed in the list ('Boxer') is compared with 'Boxer' to see if a match is found. A MATCH IS FOUND! The owner is told that they cannot enter the dog because there is already a Boxer in the competition.

Challenge

Using the example on the previous page, write down the steps needed to carry out a **serial search** for a 'Dalmatian'. Here is a reminder of the current list of dog breeds for the 'Mutts' competition.



The first step of the serial search has been done for you…

Step 1: The search criteria is defined as 'Dalmatian'.

Lesson 3

Last lesson, we looked at the **serial search** algorithm as a way of searching through an **unordered list** of values. In this lesson, we will look at a different **searching algorithm**.

A **binary search** is a fast method for searching for an item that is in an **ordered list**.

An ordered list is one where the sequence of items in the list is important. An ordered list does not necessarily contain a sequence of numbers (e.g. 1, 2, 3, 4) or characters (e.g. A, B, C, D). It might also contain, e.g. a list of names in alphabetical order, a list of files from smallest to largest or a list of records from earliest to most recent.

Example

Imagine that you have a list of teachers and want to search for the teacher called "Scott". We first need the list to be sorted into ascending alphabetical order (i.e. A-Z). We then carry out a binary search for 'Scott' using the following steps:

1) Split the list in half so that it can go directly to the name in the middle of the list.

2) It checks to see if the name in the middle matches our search criteria ('Scott').

3) If there is a match is found, the search is complete and the algorithm ends.

4) If a match is NOT found, it works out if 'Scott' comes before or after the name in the middle. It is able to do this because the list is in alphabetical order.

5) If 'Scott' comes alphabetically AFTER the name in the middle, the first half of the list is discarded. The algorithm then jumps back to step 2 with only half of the names left to search through.

6) If 'Scott' comes alphabetically BEFORE the name in the middle, the last half of the list is discarded. The algorithm then jumps back to step 2 with only half of the names left to search through, until the search criteria is found.

7) If a match is NOT found anywhere in the list, a suitable message would be output, such as "No match was found."

Binary search divides the data in half and discards, or 'bins' the half that does not contain the search term. **This makes the binary search a faster and more effective searching algorithm than the serial search.**

Challenge

A teacher has a small pile of homework from her pupils. She has organised the 7 pieces of work into alphabetical order of first name. The work she has is:

Andie **Brandy** **Candy** **Gandhi** **Mandy** **Sandy** **Suzie**

A couple of days later, the teacher is ready to mark the homework. She can't remember who has handed their homework in and she wants to check that Suzie's is in the pile.

On the next page, try to describe the binary search that the teacher would carry out step by step in order to find Suzie's work.

Note: A solution is given on the following page but be brave and have a go before looking at it!

Solution for a Binary Search of Suzie's Work:

1) Split the list in half. Find '**Gandhi**' in the middle.
2) Check to see if '**Gandhi**' matches our search criteria ('**Suzie**').
3) A match is NOT found. '**Suzie**' appears after '**Gandhi**'. The first half of the list (and '**Gandhi**') is discarded.
4) Split the list in half. Find '**Sandy**' in the middle.
5) Check to see if '**Sandy**' matches our search criteria ('**Suzie**').
6) A match is NOT found. '**Suzie**' appears after '**Sandy**'. The first half of the list (and '**Sandy**') is discarded.
7) Split the list in half. Find '**Suzie**' in the middle.
8) Check to see if '**Suzie**' matches our search criteria ('**Suzie**').
9) A match is found, the search is complete and the algorithm ends.

Challenge

You will now see several lists of data. You must firstly work out if the data is sorted or unsorted. You must then choose which type of search algorithm you might use on the data and give your reason why.

Note: You must not answer **binary search** for each question, even though it is the most effective searching algorithm!

1) 3 13 23 33 43

I would use a [ ] search because the data is [ ].
serial *OR* **binary** **sorted** *OR* **unsorted**

2) 43 33 23 13 -3

I would use a [ ] search because the data is [ ].
serial *OR* **binary** **sorted** *OR* **unsorted**

3) 3 33 23 43 13

I would use a [ ] search because the data is [ ].
serial *OR* **binary** **sorted** *OR* **unsorted**

4) -3 -33 -23 -43 -13

I would use a [ ] search because the data is [ ].
serial *OR* **binary** **sorted** *OR* **unsorted**

5) A F J M P

I would use a [ ] search because the data is [ ].
serial *OR* **binary** **sorted** *OR* **unsorted**

6) Y T K E G

I would use a [ ] search because the data is [ ].
serial *OR* **binary** **sorted** *OR* **unsorted**