3. Standard algorithms

There are a number of algorithms the IB course will assume you know by memory. These are the "standard algorithms".

- Sequential search
- Binary search
- Bubble sort
- Selection sort

Any algorithm not defined as a "standard algorithm" can be considered a "novel algorithm" and will be either presented to you in the exam or is an algorithm you would be expected to devise.

3.1 Sequential & binary search

The premise of the sequential search is to iterate through a set of data sequentially until you find the item you are looking for.

```
// javascript
function sequential( haystack, needle ) {
    for (var i=0; i<haystack.length; i++) {
        if (needle == haystack[i]) {
            return(i);
        }
    }
    return(-1);
}
```

One interesting thing to note with the sequential search is it does not require your data to be sorted ahead of the search. This could save a lot of processing time. If, however, your data is already sorted, you can include within your sequential search a test to see if we have already gone past the point at which the record we are looking for would exist. In that case, we can abort the remainder of the search and return a "not found" result.

A binary search divides a range of values into halves, and continues to narrow down the field of search until the unknown value is found. It is the classic example of a "divide and conquer" algorithm. Your data must be pre-sorted!

Check this excellent illustration of binary vs sequential search!

- https://pbaumgarten.com/ib-compsci/unit-4/binary-and-linear-search-animations.gif

```javascript
// javascript
function binary( haystack, needle ) {
    var max = haystack.length-1;
    var min = 0;
    while (max >= min) {
        var mid = Math.floor((min+max)/2);
        if (haystack[mid] == needle) {
            return mid;
        }
        if (haystack[mid] > needle) {
            max = mid - 1;
        }
        if (haystack[mid] < needle) {
            min = mid + 1;
        }
    }
    return (-1);
}
```

This is known as an iterative binary search. There is also a recursive binary search which we will look at in a later unit. Just be aware if you have to google binary search that there are two types.

3.2 Search practice

Question 1: Tracing binary search

Create a trace table on the binary search algorithm above with the following values:

haystack = [1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37]
1st needle = 15
2nd needle = 5
Compare that on a trace table for a sequential search algorithm.

It should be fairly simple to code an implementation of this to test your algorithms - have a go.

Question 2: Simple spell checker

If a user inputs a sentence, separate the individual words, strip out any punctuation, check each word against your dictionary list. Print each word out with PASS or FAIL against it based on if it was found in your dictionary.

Devise an algorithm for this and have a go at implementing it. Will you use the sequential or binary search? Why?

You will need a "word list" to use as your dictionary. There are lots of good word lists available online. This is one I've used before but feel free to find your own.

- https://github.com/dolph/dictionary/blob/master/popular.txt

3.3 Bubble & selection sort

Check this animation of how bubble sort works:

- http://pbaumgarten.com/ib-compsci/unit-4/bubblesort.gif

The bubble sort algorithm looks like

```
// javascript
function bubbleSort( data ) {
    var done = false;
    while (!done) {
        done = true;
        for (var i=1; i<data.length; i++) {
            if (data[i] < data[i-1]) {
                done = false;
                var temp = data[i-1];
                data[i-1] = data[i];
                data[i] = temp;
            }
        }
    }
    return(data);
}
```

Check this animation of how bubble sort works:

- http://pbaumgarten.com/ib-compsci/unit-4/selectionsort.gif

The algorithm for selection sort looks like...

```
// javascript
function selectionSort( data ) {
    for (var i=0; i<data.length; i++) {
        var indexOfLowest = i;
        for (var j=i; j<data.length; j++) {
            if (data[j] < data[indexOfLowest]) {
                indexOfLowest = j;
            }
        }
        var temp = data[i];
        data[i] = data[indexOfLowest];
        data[indexOfLowest] = temp;
    }
    return(data);
}
```

Both of these sorting algorithms look very inefficient (and, generally speaking they are) but they do have unique edge-case advantages when they might be of real-world use.

Bubble sort is highly efficient when the dataset is already mostly sorted. It is ideal to use when you want to add a new value into it's sorted position to an already sorted dataset.

Selection sort is optimised for when writing data is very expensive (slow) when compared to reading. Eg:. writing to flash memory or EEPROM. No other sorting algorithm has less data movement! This is important to realise as you can be asked questions justifying the use of one algorithm over another.

3.4 Sort practice

Question 1

Here is a set of 50 integers from https://www.random.org/integer-sets/...

```
34, 43, 73, 88, 9, 91, 48, 10, 94, 3, 75, 87, 74, 63, 11, 36, 82, 100, 28, 68, 18,
60, 35, 81, 79, 23, 86, 41, 49, 2, 7, 83, 6, 58, 47, 39, 27, 54, 21, 12, 4, 5, 31,
46, 62, 55, 37, 57, 67, 93
```

Implement both sorting algorithms to process your data set. Compare and contrast how many comparison read operations each takes (ie: count the number of times the if statement operates), and how many write operations each takes (ie: how many times the array is written to).

Is it correct that the selection sort orders of magnitude more efficiently for the number of write operations?

```
100, 1, 2, 3, 5, 9, 10, 11, 12, 17, 20, 23, 25, 31, 33, 35, 39, 40, 42, 43, 44, 45,
46, 47, 51, 52, 55, 56, 59, 61, 62, 63, 64, 66, 69, 70, 75, 77, 78, 79, 80, 81, 83,
86, 87, 88, 89, 92, 94, 96, 98
```

Swapping to use the second set of numbers, is it true that the bubble sort is more efficient for this "almost sorted" set? By how much?

Question 2

Once you are successfully sorting numbers, how about sorting strings such as a list of names?

```
"Eustolia","Nathan","Milissa","Willie","Hoyt","Alexandria","Clelia","Alpha","Delbert
","Boyd","Milton","Vivan","Constance","Hilma","Irving","Carie","Nicky","Adele","Carl
ene","Hermina","Ayana","Frederica","Arianna","Zandra","Vina","Lory","Mao","Alona","L
ajuana","Coralie","Allyson","Corey","Geraldo","Sherryl","Monika","Charlesetta","Deon
","Coletta","Jed","Carlee","Lise","Teresita","Odelia","Adeline","Olive","Elisha","Ca
sey","Octavia","Alexandra","Franklyn"
```

From http://listofrandomnames.com/.

Question 3

Sort the given set of dates given in dd/mm/yyyy format into their correct calendar order so the date which occurs first, appears first in the list.

"29/06/2009","06/06/1984","16/06/1993","23/11/1996","23/09/1986","07/07/2002","29/01/1999","13/06/1998","14/02/2005","29/08/2013","24/12/2009","04/09/2019","02/02/2020","22/10/2015","08/11/1987","23/10/2018","14/10/2015","19/02/2013","05/06/1989","21/08/1991","06/06/2005","03/02/1993","01/12/1993","01/09/1995","24/01/2018"

From https://www.random.org/calendar-dates/