Sorting

Most searching algorithms rely on the fact that the data is sorted into numerical order

> Remeber that letters are in numerical order according to ASCII or Unicode

Selection Sort

My Attempt

```
Loop through all the numbers
    Loop through all the rest of the numbers
        Find the smallest number
    Swap the number at the start of the loop with the smallest number
```

```c
#include <stdio.h>

void print_array(int array_length, int array[], char* message);

int main(){
    // Initalise data
    int arr[] = { 7, 2, 5, 4, 1, 6, 0, 3};
    int arr_len = 8;

    // Print starting state of array
    print_array(arr_len, arr, "Starting Array: ");

    // Sort
    for(int i = 0; i < arr_len, i++){
        int s_val = arr[i];
        int s_index = i;
        for(int j = i; j < arr_len; j++){
            if(arr[j] < s_val){
                s_val = arr[j];
                s_index = j;
            }
        }

        // * Could do a XOR swap however, this is easier to learn from.
        int temp = arr[i];
        arr[i] = s_val;
        arr[s_index] = temp;
    }

    // Print ending state of array
    print_array(arr_len, arr, "Final Array   : ");
}

void print_array(int array_length, int array[], char* message){
    printf("%s", message);
    for(int i = 0; i < array_length, i++){
        printf("%i ", array[i]);
    }
    printf("\n");
}
```

```
Output:
> ./selection_sort
Starting Array: 7 2 5 4 1 6 0 3
Final Array   : 0 1 2 3 4 5 6 7
```

CS50X Solution

```
For i from 0 to n-1
    Find smallest number between numbers[i] and numbers[n-1]
    Swap smallest number with numbers[i]
```

Time Complexity

Because of the double `for` loop, selection sort is $O(n^2)$, $\Omega(n^2)$ and, therefore, $\Theta(n^2)$.

The $O(n^2)$ can be figured out by the following:

$$(n-1)+(n-2)+(n-3)+...+1$$

$$n(n-1)/2$$

$$(n^2-n)/2$$

$$(n^2)/2-n/2$$

In big O notation, we only care about the thing in the equation, which is the n^2 portion of this, therefore the algorithm is considered $O(n^2)$.

The reason why I say it is $\Omega(n^2)$, and thus $\Theta(n^2)$, is because the algorithm has no concept to check if the list is already sorted, this could be improved with a flag and breaking out early, potentially making the algorithm $\Omega(n)$ as it will only need to check through the list once.

Bubble Sort

My Attempt

```
Loop through all bumbers
```

```
    Loop from 0 to end
        If value is bigger than value next to it
            Swap value with value next to it
```

```c
#include <stdio.h>

void print_array(int array_length, int array[], char* message);

int main(){
    // Initalise data
    int arr[] = { 7, 2, 5, 4, 1, 6, 0, 3};
    int arr_len = 8;

    // Print starting state of array
    print_array(arr_len, arr, "Starting Array: ");

    // Sort
    for(int i = 0; i < arr_len; i++){
        for(int j = 0; j < arr_len - i - 1; j++){
            if(arr[j] > arr[j + 1]){
                int temp = arr[j + 1];
                arr[j + 1] = arr[j];
                arr[j] = temp;
            }
        }
    }

    // Print ending state of array
    print_array(arr_len, arr, "Final Array   : ");
}

void print_array(int array_length, int array[], char* message){
    printf("%s", message);
    for(int i = 0; i < array_length, i++){
        printf("%i ", array[i]);
    }
    printf("\n");
}
```

CS50X Solution

v1

```
Repeat n-1 times
    For i from 0 to n-2
        If numbers[i] and numbers[i+1] out of order
            Swap them
```

v2

```
Repeat n-1 times
    For i from 0 to n-2
        If numbers[i] and numbers[i+1] out of order
            Swap them
    If no swaps
        Quit
```

Time Complexity

This again uses a double `for` loop, therefore it is $O(n^2)$. As well as this, it also doesn't implement a check for whether the array is already sorted, therefore it is $\Omega(n^2)$ and therefore $\Theta(n^2)$.

To improve upon this algorithm, we could have a flag that checks if any swaps were completed, if not the loop breaks. This means that the time complexity would now be $\Omega(n)$.

Merge Sort

CS50X Solution

Psuedo Code

```
If only one number
    Quit
Else
    Sort left half of numbers
    Sort right half of numbers
    Merge numbers together
```

```
pie title Pets adopted by volunteers
    "Dogs" : 386
    "Cats" : 85
    "Rats" : 15
```