Software Notes and General Questions

Big O notation: Algorithm efficiency

Big O notation looks at analysing the efficiency of an algorithm, this can be measured with n which is determined by the input size of the parameter (think as if it is an array of n elements. This will return a quantitative figure that we can use to compare the algorithm time and space use. To analyses the algorithm we look at the steps within the code, ignoring constants and most general code that is placed outside a loop.

$$O(1) < O(logn) < O(n) < O(nlogn) < O(n^2) < O(2^n) < O(n!)$$

Where….
O(1) = same time no matter the input
O(n) = for (int i = 0; i < n; i++) lines of code
$O(n^2)$ = two counter loops

Immutables data structures and thread-safe programs

Immutable data structures are structures that cannot be changed (added, removed, etc). This can be a pre-defined array, where we cannot add an extra element.

When we discuss multi-threading we are referring to two functions occurring simultaneously, this can become a problem with writing to immutable data structs, as multiple overwrites at the same time can result in lost data.

There are several solutions to this problem including.

- Modifying the data struct one at a time
- Implementing data structures that can be modified at the same time
- Explicitly keeping data immutable, meaning that only one thread can access their data.