Sorting

Introduction

If we were to manage big databases, we would need a way to quickly navigate these databases. For example, a database that had millions of entries would need a **sorting algorithm** to function correctly. Searching for a name in an unorganized database could potentially take forever. Because of this, computer scientists seek simple solutions that considers **rate of growth** and **complexity**.

Example

Consider the following code:

```
for(int i=0; i<n; i++)
  for(int j=0; j<n; j++)
    System.out.println("HelloWorld");
```

If we were to define `n` as 5, we would have 25 prints. For 10, 100 prints. The number of operation increases equal to `n` squared. The **complexity** of this function is O(n^2). The **rate of growth**, as we may guess, is `n` squared. Keep in mind that when we define complexity, we **only consider the variable. Any constant is ignored when writing complexity**. For example, if we were to replace the ending condition `j < n` with `j < n/10`, the complexity would still be O(n^2).