**The Quicksort:**

The Quick Sort Algorithm uses a divide & conquer algorithm to quickly reduce the size of a problem.

1. Choose one item to be the ‘pivot’ value. This can be either the first or last value in a data set.
2. Compare every item with the pivot.
3. Divide the list into two ‘partitions’. These are items that are either bigger or smaller than the pivot.
4. Continue recursively until each partition has one item.

**Visualised:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **9** | **5** | **4** | **15** | **3** | **8** | **11** |

In the unsorted data set above, we have chosen the first value to be our pivot. This is 9. We will now compare 9 with all other values in the data set and divide them into two partitions.

For example: 9 is bigger than 5. 5 is smaller then 9, so it will go into the partition that is smaller than 9. However, 15 is bigger than 9. 15 will go into the second partition that is bigger than our pivot value. In short:

* All elements less than the pivot value must be in the first partition.
* All elements greater than the pivot value must be in the second partition.

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 5 | 4 | 8 |

|  |
| --- |
| 9 |

|  |  |
| --- | --- |
| 15 | 11 |

3 and 15 are now the pivots in the left and right partitions. We now use recursion to repeat the process until the list is in sequence.

|  |
| --- |
| 3 |

|  |  |  |
| --- | --- | --- |
| 5 | 4 | 8 |

|  |
| --- |
| 9 |

|  |
| --- |
| 11 |

|  |
| --- |
| 15 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **3** | **4** | **5** | **8** | **9** | **11** | **15** |

On average, the Quicksort can be two to three times faster than the Merge Sort. However, it speeds varies a lot. Especially on the size of the dataset. It can either be extremely fast, or slow.

The average time complexity is n\*log-n while its worse can be n\*n.