AQA Computer Science A-Level

4.3.5 Sorting algorithms

Concise Notes

Specification:

4.3.5.1 Bubble sort

Know and be able to trace and analyse the time complexity of the bubble sort algorithm. This is included as an example of a particularly inefficient sorting algorithm, time-wise. Time complexity is $O(n^2)$.

4.3.5.2 Merge sort

Be able to trace and analyse the time complexity of the merge sort algorithm. The 'merge' sort is an example of 'Divide and Conquer' approach to problem solving. Time complexity is O(nlogn).

Sorting Algorithms

- Used to put the elements of an array into a specific order
- The binary search algorithm can only be carried out on sorted arrays, so a sorting algorithm must be used before the search if the array is not ordered

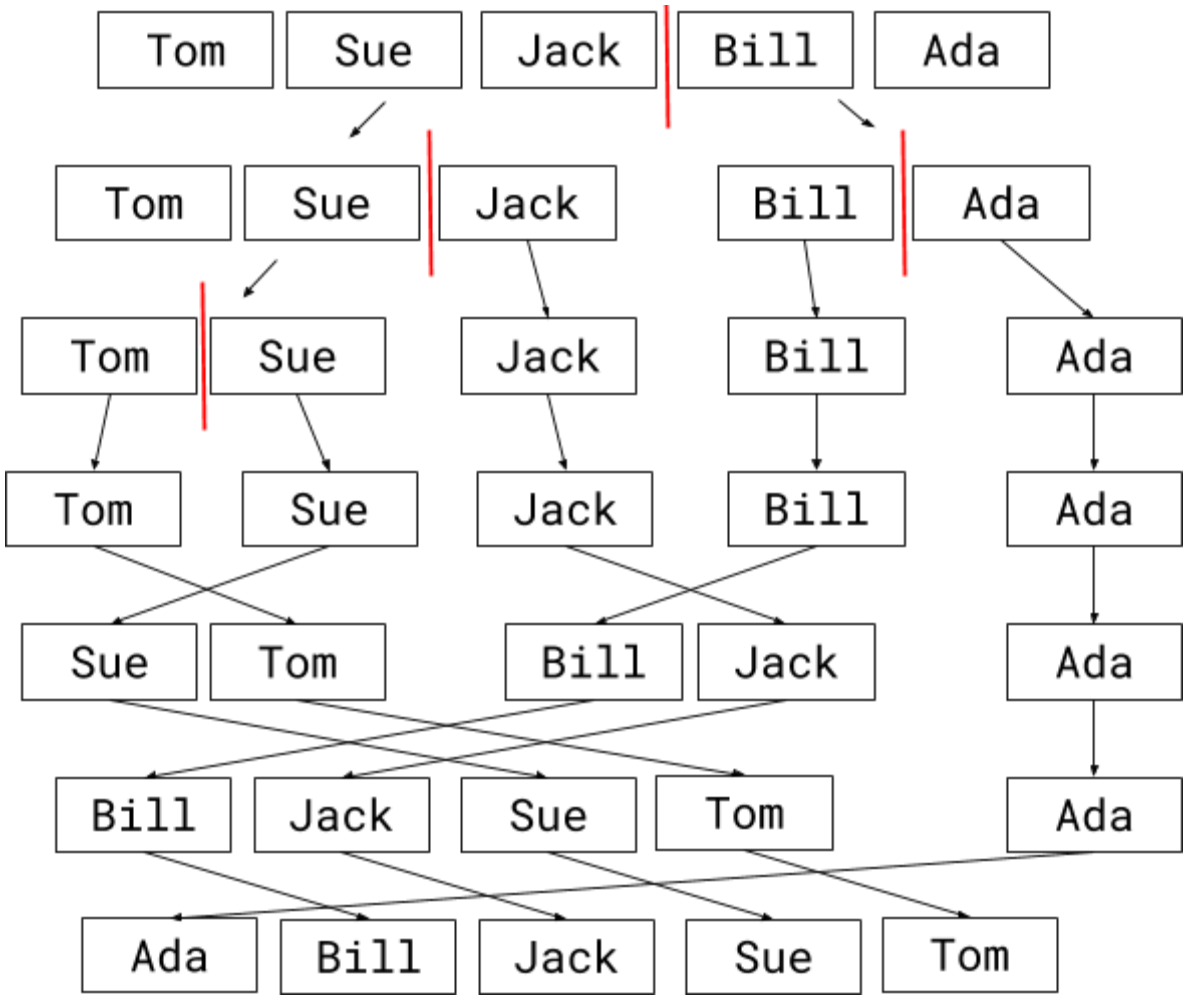
Bubble Sort

- Swaps adjacent items in an array until the array is in order
- Has a time complexity of $O(n^2)$
- A particularly inefficient sorting algorithm

In the example above, a green loop represents two items that are in the correct order and do not require swapping. A red loop represents two items that are in the wrong order and must be swapped. The algorithm completes after a pass in which no swaps occur.

Merge Sort

- An example of a "divide and conquer" sorting algorithm
- Divides an array into smaller arrays until each array contains just one element
- When an array has just one element, it can be considered an ordered array
- Arrays are then merged back together in order to form an ordered array
- Has a time complexity of O(nlogn)
- An efficient sorting algorithm



In the example above, the array is split down the middle and then continues to be split until each resulting array is just one element in size. The arrays are then merged back together in the correct order.