**Merge Sort: (GCSE & A-Level)**

The Merge Sort can sort a data set extremely quickly using divide and conquer. The principle of divide and conquer is to create two or more identical sub-problems from the large problem, solving them individually and combining their solutions to solve the bigger problem. With the merge set, the data set is repeatedly split in half until each item is in its own list. Adjacent lists are merged back together, with each item in the sub-list being entered into the correct place in the new, combined list.

*Simplified Explanation:*

1. Repeatedly divide the list in half into two smaller lists until each item is in its own list
2. Take two adjacent lists and start with the first item in each one.
3. Compare the two items
4. Insert the lowest item into a new list. Move to the next item in the list it was taken from.
5. Repeat step 4 until all the items from one of the lists have been put into the new list.
6. Append all the items from the list still containing items to the new list.
7. Remove both old lists.
8. Repeat from step 2 until only one list remains.

**Merge Sort in Python:**

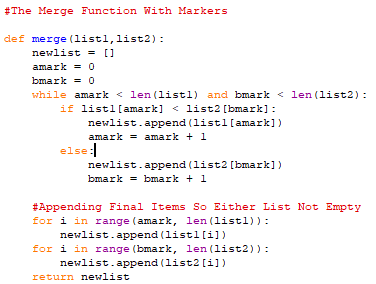
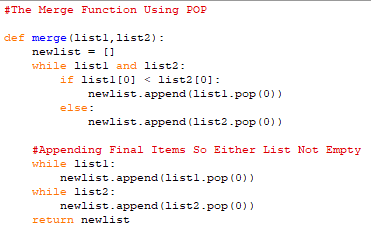
**Note:** For my variant version of the Merge Sort. We require two functions. One function is the ‘Merge’ part of the Merge Sort. Basically, this takes two sorted lists and sorts them into one sorted list. This only works if we have two smaller already-sorted lists.

The other function is the actually sorting part of the function.

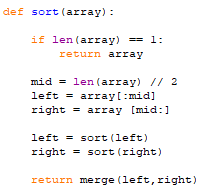
The Merge Sort works best with Recurision.

The command ‘.pop’ is unique to python only. The Merge Function with markers is more universal to other programming languages. You can use ‘.pop’ in your exam. It all depends if your examiner understands the meaning of ‘.pop’. Bit risky, but it should be fine.

**The ‘Merge Function’:**



**The ‘Sorting Part’:**



|  |  |
| --- | --- |
| **Advantages:** | **Disadvantages:** |
| * Extremely Quick | * Uses a lot of memory space |
| * Fantastic for sorting large data sets | * Complicated algorithm |
| * Best are sorting data that is normally accessed sequentially | * Recursive calls result in additional overhead making it unsuitable for small data sets |

**EXAM NOTE:**

* **GCSE |**  You will never be expected to write out the complete Merge Sort due to its complexity. The best they can ask of you is to fill parts in or draw a visual respresentation.
* **A-Level |** You may not be expected to write out the complete Merge Sort. But you may be asked to write parts of the alogirthm depending on the question asked.

**Big O Notation (A-Level)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Best Case:** | **Average Case:** | **Worst Case:** | **Space Complexity** |
| * Linearithmic | * Linearithmic | * Linearithmic | * Linear |
| * O(n log n) | * O(n log n) | * O(n log n) | * O(n) |