MODULE L · OCR J277 2.1.3, 2.2.1 · CAMBRIDGE 0478 7.4, 8.1.4(D)

Standard algorithms

Linear search, **binary search**, **bubble sort**, **merge sort**, **insertion sort**, **comparing them**, and **totalling, counting, max, min and average**. **Cambridge 0478: you only need cards 01, 03 and 07**, without the GCSE-only rows. Spec versions as checked in November 2026.

01 Linear search (L1) · both boards

Steps Start at the first item. Compare with the target: equal, found, stop. Else next item. End of list: not found.

Pre-req None: works on unsorted data.

+/- Simple. Slow on large lists: may check every item.

Code A loop through the positions, an IF comparing each item with the target, a found flag.

Example 42, 17, 93, 8, 56, 71: find 56 in 5 comparisons (index 4 from 0; position 5 from 1). Find 30: all 6 checked, not found.

IGCSE ONLY Write it in pseudocode, e.g. `WHILE Found = FALSE AND Index <= 6 DO`.

02 Binary search (L2) GCSE ONLY

Pre-req The data must be sorted.

Steps Check the middle item. Equal: found. Target smaller: discard the middle and the upper half. Larger: discard the middle and the lower half. Repeat until found or no items left.

Middle `mid = (low + high) DIV 2` takes the left of two middles. Follow the question's rule.

Code low, high and mid; `low = mid + 1` or `high = mid - 1`.

Example 3, 9, 14, 20, 27, 31, 38, 45, 52, 60, 66: find 60 by checking 31, 52, 60 (linear: 10). Find 40: 31, 52, 38, 45, not found.

03 Bubble sort (L3) · both boards

Steps Compare each pair of adjacent items; swap if in the wrong order. One pass = once through the list. Repeat passes until a pass makes no swaps.

Pass 1 The largest value ends in its final place at the end.

Swap Use a temp variable: `temp = a`, `a = b`, `b = temp`.

+/- Simple; little extra memory. Slow on large lists.

Example 15, 4, 23, 8, 11. Pass 1: 4, 15, 8, 11, 23 (3 swaps). Pass 2: 4, 8, 11, 15, 23 (2 swaps). Pass 3: no swaps, stop.

IGCSE ONLY Write it: `REPEAT ... FOR ... UNTIL NOT Swapped`. Descending: change `>` to `<`.

04 Merge sort (L4) GCSE ONLY

Divide Split in half again and again until every list holds one item (already sorted).

Merge Compare the front items of two lists, move the smaller into the new list, repeat. Merge pairs until one list remains.

Code Split into left and right halves; merge loop compares `left[i]` with `right[j]`.

+/- Fastest of the three sorts on large lists. More memory (new lists); more complex to code.

Example 38 12 45 7 29 16 51 3 → [12 38] [7 45] [16 29] [3 51] → [7 12 38 45] [3 16 29 51] → 3 7 12 16 29 38 45 51.

05 Insertion sort (L5) GCSE ONLY

Parts A sorted part at the start; an unsorted part after it.

Steps First item is sorted. Take the next item; shift larger sorted items one place right; insert it into the gap. Repeat for every item.

Code Outer for loop; `current = items[i]`; inner while loop shifts right; then insert.

+/- Simple; little extra memory; quick on small or nearly sorted lists. Slow on large unsorted lists.

Example 27 9 34 15 4 → 9 27 34 15 4 → 9 27 34 15 4 → 9 15 27 34 4 → 4 9 15 27 34.

06 Comparing them (L6) GCSE ONLY

Search Binary: large sorted lists (1 000 000 items: at most 20 checks). Linear: unsorted or small lists.

Sort Merge: large lists, memory to spare. Insertion: small or nearly sorted, little memory. Bubble: simple; stops early on a pass with no swaps.

Spot it low/high/mid = binary. `items[i] VS items[i + 1]` + temp = bubble. current + shift right = insertion. left/right halves = merge.

Recommend Name the algorithm and give reasons from the scenario (sorted? size? memory?).

07 Totalling, counting, max, min, average (L7) · both boards

Pattern Initialise before the loop, update inside it, output after it.

Total Start at 0; `total = total + value`.

Count Start at 0; `count = count + 1` when the condition is met.

Max / min Start with the first item; replace it when a larger (max) or smaller (min) value is found.

Average total ÷ number of values (not the conditional count), after the loop.

Example 12, 7, 25, 18, 9, 21: total 92; over 15: 3; max 25; min 7; average 92 ÷ 6 = 15.33.

IGCSE ONLY Named standard methods (7.4); `Total ← Total + Mark`, `Count ← Count + 1`.

GCSE ONLY Not named as standard algorithms; used in 2.2.1 programs and trace tables.

08 What each board asks · both boards

GCSE ONLY Main steps and code segments; apply to a data set; pre-requisites; identify from code, pseudocode or ERL. Code need not be remembered.

IGCSE ONLY Write and amend linear search, bubble sort, totalling, counting, max, min and average in pseudocode, including in the 15-mark question.

Notation OCR: Exam Reference Language, `=`, arrays from 0. Cambridge: pseudocode, `←`, arrays usually from 1.

Not on 0478 Binary search, merge sort, insertion sort, comparing them (L2, L4, L5, L6).

01
Linear search: say it starts at the **first item** and checks **each item in turn**.

02
Binary search needs **sorted** data. Include the step "middle = target: **found, stop**".

03
Bubble sort stops only after a **pass with no swaps**, even if the list already looks sorted.

04
Merging **compares the front items**: never join two lists end to end and then sort.

05
Set totals and counts to 0 **before** the loop, not inside it.

06
Start max or min with the **first item**, not 0: values can be negative (-18, -21, -16).