B2 PROGRAMMING · B2.4

Programming algorithms

Measuring efficiency with **Big O**, the two searches, the two sorts, and **recursion** (HL only).

01 Big O efficiency

O(1)	Constant: array lookup.
O(log n)	Halving: binary search.
O(n)	Linear: linear search.
O(n log n)	Log-linear: quicksort.
O(n²)	Quadratic: bubble/selection.

02 Searching

Linear	Check each item in turn.
Linear cost	O(n); any order works.
Binary	Halve a sorted list each step.
Binary cost	O(log n); needs sorted data.
Method	Check middle, discard a half.

03 Four things to master

1

Big O

State and justify how work grows with input size.

2

Searching

Trace a linear search and a binary search.

3

Sorting

Trace bubble sort and selection sort.

4

Recursion · HL

Write and trace a function that calls itself.

04 Sorting

Bubble	Swap out-of-order neighbours.
Bubble effect	Largest bubbles to the end.
Selection	Find largest, place at end.
Both cost	$O(n^2)$.
Why sort?	Enables a binary search.

05 Recursion · HL

Definition	A function that calls itself.
Base case	The stopping point.
Recursive case	Calls itself, smaller input.
fact(4)	$4 \times 3 \times 2 \times 1 = 24$.
No base case	Infinite recursion, overflow.

06 Know the difference

Linear vs binary	O(n) on any order versus O(log n) by halving, but binary needs sorted data.	SEARCH
Bubble vs selection	Swap adjacent pairs versus find the largest and place it; both are $O(n^2)$.	SORT
Time vs space	How the number of steps grows versus how much extra memory the algorithm needs.	BIG O
Base vs recursive case	The case that stops the recursion versus the case that calls itself with a smaller input. (HL)	HL

FINAL PASS BEFORE THE EXAM

Rapid exam tips

Eight things that lose marks if you slip on them. Recursion tips are HL only. Skim before you walk in.

01

Big O is about growth, not seconds. Flatter growth scales better.

02

Linear search = O(n); **binary search** = O(log n) and needs **sorted** data.

03

Never run a **binary search on unsorted data**; sort it first or use linear.

04

Bubble and **selection** sort are both **O(n²)**. Learn the pairing.

05

When tracing, watch the **middle index** rounding and your **low/high** values.

06

Binary search finds 7 in a 6-item list in **2 checks**; linear takes up to 6.

07

HL: every recursion needs a **base case** and a step that shrinks the input.

08

HL: no base case means **infinite recursion** and a stack overflow.