AQA GCSE

COMPUTER SCIENCE

Fundamentals of ALGORITHMS

Section 3.1

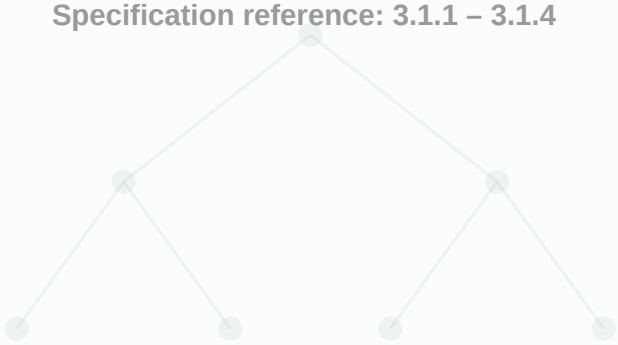
Revision & Exam Practice Booklet

For High-Achieving Students · Grades 8–9

Definitions | Exam Tips | Exam Questions | Stretch & Challenge

Complete coverage of specification 3.1.1 – 3.1.4: representing algorithms, efficiency, searching (linear & binary) and sorting (bubble & merge). Worked examples, AQA-style questions, mark schemes and trace tables.

Specification reference: 3.1.1 – 3.1.4



How to Use This Booklet

This booklet covers every element of the AQA GCSE Computer Science specification, Section 3.1 – Fundamentals of Algorithms. It is designed to consolidate your knowledge, stretch your thinking, and prepare you for the highest grades in the written examination.

Each topic in this booklet contains:

- **Key knowledge** — the exact content required by the AQA specification.
- **Worked examples** — model answers showing how to apply the knowledge.
- **Quick-check questions** — short questions to confirm understanding.
- **Exam-style questions** — written in the AQA house style, with mark allocations.
- **Stretch & Challenge** — higher-order questions for top-grade candidates (Grade 8/9).

Key to symbols

Definition

Precise definition you should be able to reproduce in an exam.

Exam Tip

Examiner insight — common errors, what gains marks, how to phrase answers.

Exam-Style Question

AQA-style practice question with mark allocations, just like the real paper.

Stretch & Challenge

Higher-order questions that go beyond the standard syllabus to deepen thinking.

Contents

Exam Tip

A frequent mistake is to write that decomposition and abstraction are “the same thing.” They are not.

- Decomposition splits a problem into smaller parts.
- Abstraction removes detail that isn’t needed.

If asked to define either term, give the specification’s definition and follow it with a short example.

Quick-check questions

1. State, in one sentence, what is meant by the term algorithm. [1]
2. Explain the difference between an algorithm and a computer program. [2]
3. Define the term decomposition. [1]
4. Define the term abstraction. [1]
5. Give one example of abstraction used in everyday life and justify your choice. [2]

Exam-Style Question 1.1A [6 marks]

A games developer wishes to design a new mobile game in which a character must navigate a maze, avoiding enemies and collecting tokens.

(a) Explain how decomposition could be used by the developer when designing the game. **[3]**

(b) Explain how abstraction could be used by the developer when designing the game. **[3]**

Stretch & Challenge 1.1

Two students each design an algorithm for the same problem. One uses heavy decomposition into many small sub-problems; the other uses few but larger sub-problems.

Discuss the advantages and disadvantages of each approach. In your answer, consider readability, debugging, reuse and overall efficiency. [6]

1.2 Pseudo-code, program code and flowcharts

The AQA specification requires you to be able to represent the same algorithm in three forms: pseudo-code, program code (typically Python in this specification), and flowcharts.

Exam Tip

Any exam question that gives you pseudo-code will use the AQA standard version (Appendix B of the specification). You will not be asked to write it in this exact form, but you must be able to read it confidently.

Read the rubric carefully: the question will tell you which form to use (pseudo-code, program code or flowchart). Using the wrong form may lose marks.

AQA pseudo-code essentials

Below is a summary of the most commonly used AQA pseudo-code constructs.

- **Advantages:** Far fewer comparisons on large lists; typically much faster than linear search.
- **Disadvantages:** Cannot be used on unsorted data; more complex to implement.
- **Worst case:** Approximately $\log_2(N)$ comparisons.
- **Example:** A list of 1,000,000 items requires at most about 20 comparisons.

Exam Tip

In long-answer questions on binary search, examiners reward candidates who explicitly state that the list must be sorted before binary search can be used. This is a 1-mark point that many students miss.

3.3 Comparing linear and binary search

Comparison table

Criterion	Linear Search	Binary Search
List must be sorted?	No	Yes
Speed on small lists	Fast enough	Fast
Speed on large lists	Slow (up to N comparisons)	Very fast (~$\log_2$ N comparisons)
Complexity to implement	Very simple	More complex
Works on any data?	Yes — any list	Only on ordered data
Best case	1 comparison	1 comparison
Worst case	N comparisons	~$\log_2(N)$ comparisons
Example (N = 1,000,000)	Up to 1,000,000	At most ~20

When to use each

- **Use linear search when:** the list is small, unsorted, or the cost of sorting the list first would outweigh the benefit.
- **Use binary search when:** the list is already sorted and reasonably large.

Exam Tip

When asked to compare the two algorithms, structure your answer:

- State a clear point of comparison (e.g. “Speed on large lists…”).
- State what linear search does.
- State what binary search does.
- Repeat for the next criterion. This ensures you actually compare rather than just describe.

Quick-check questions

1. State one advantage of linear search over binary search. [1]
2. State one advantage of binary search over linear search. [1]
3. Explain why binary search cannot be performed on an unsorted list. [2]
4. A list contains 64 sorted items. State the maximum number of comparisons required by binary search. [2]

Exam-Style Question 3A [9 marks]