4.14 Choosing a Project

This guide is designed to help students choose a strong idea for the AQA A-level Computer Science non-exam assessment (NEA). A good project should be interesting, realistic, technically challenging, and suitable for demonstrating programming skill through analysis, design, development, testing and evaluation.

What makes a good programming project?

- **It solves a real problem or investigates a genuine question.** The best projects have a clear purpose and a specific user or client.
- **It is complex enough for A-level.** Avoid projects that are mainly forms, menus and simple CRUD operations unless they include significant algorithms, data processing or decision-making.
- **It gives you chances to show technical skill.** You should be able to include data structures, algorithms, validation, testing, modular design and iterative improvement.
- **It is achievable.** The scope should be challenging but possible within the time available.
- **You can explain the problem domain.** Choose a context you understand or can research properly.

Use Group A algorithms for the specification

AQA's algorithm content includes graph traversal, tree traversal, Reverse Polish notation, searching algorithms and sorting algorithms. These can be used as starting points for NEA ideas, but the project should go beyond simply implementing the algorithm. Aim to apply the algorithm to a realistic situation.

Algorithm area	What it could support	Possible NEA idea
Breadth-first search	Finding shortest paths in an unweighted graph	Campus route planner, escape-room solver, public transport stop finder, maze shortest-route tool
Depth-first search	Exploring paths, solving mazes, checking connectivity	Procedural maze generator and solver, puzzle validator, network exploration tool
Dijkstra's shortest-path algorithm	Finding lowest-cost routes in a weighted graph	Journey planner using distance, time or cost; delivery-route optimiser; school-site navigation tool
A* search	Efficient pathfinding using a heuristic to guide the search	Game enemy pathfinding, robot navigation simulation, map route planner with estimated distance to goal

Tree traversal	Processing hierarchical data such as trees, menus, expressions or decision structures	Quiz/question decision tree, expression tree visualiser, file-system analyser, tournament bracket manager
Minimax	Choosing moves in a two-player game by looking ahead at possible outcomes	Noughts and crosses AI, connect-four style game opponent, strategy-game move recommender
Alpha-beta pruning	Improving minimax by cutting off branches that do not need to be searched	More efficient board-game AI that compares search depth, time taken and quality of moves
Reverse Polish notation	Parsing and evaluating expressions using a stack	Scientific calculator, formula checker, spreadsheet-style expression evaluator, logic expression parser
Linear and binary search	Finding records efficiently, comparing search performance	Library catalogue, revision card finder, product database, student progress tracker
Binary tree search	Organising searchable data in a tree structure	Dictionary app, autocomplete prototype, searchable glossary, ranking system
Hashing and hash tables	Fast lookup, duplicate detection and efficient storage of key-value data	Login system prototype, spell checker, plagiarism-style similarity checker, fast inventory lookup
Sorting algorithms	Ordering data and comparing algorithm efficiency	Leaderboard system, stock sorter, task prioritiser, algorithm visualisation tool
Dynamic programming	Solving optimisation problems by storing and reusing previous results	Timetable optimiser, budget planner, revision-plan optimiser, knapsack-style resource allocation tool

Questions to ask before choosing an idea

1. Who is the user or client?
2. What problem are they trying to solve?
3. What data will the system need to store, process or generate?
4. Which algorithms could be used for a meaningful purpose?
5. How could the system be split into modules or classes?
6. What features would make the project more than a basic database application?
7. How will you test whether the solution works?
8. What could be improved after an initial prototype?

Example Project Ideas

- **Route planner:** Represent locations as a graph and use traversal or shortest-path logic to suggest routes.
- **Revision recommendation system:** Store topics, quiz results and confidence scores, then prioritise revision tasks using sorting and searching.
- **Maze or puzzle game:** Generate puzzles, validate solutions and use search algorithms to solve or score them.
- **Expression calculator:** Convert between infix and Reverse Polish notation, then evaluate expressions using a stack.
- **Algorithm visualiser:** Let users compare how searching or sorting algorithms behave on different data sets.
- **Decision-support tool:** Use trees or graph structures to recommend options, classify cases or guide users through choices.
- **Weighted route optimiser:** Use Dijkstra's algorithm to find the lowest-cost route through a network where edges have weights such as distance, time, difficulty or cost.
- **Heuristic pathfinding game:** Use A* search to control an enemy, robot or character that must navigate around obstacles towards a target efficiently.
- **Two-player strategy game AI:** Use minimax to build a computer opponent that looks ahead at possible moves and chooses the best option.
- **Efficient board-game opponent:** Extend a minimax-based game with alpha-beta pruning, then compare how pruning affects search depth, response time and move quality.
- **Fast lookup or matching system:** Use hashing or hash tables to store and retrieve records quickly, detect duplicates, or compare items such as words, usernames or product codes.
- **Optimisation planner:** Use dynamic programming to make decisions where earlier results can be reused, such as revision scheduling, budget allocation or selecting the best combination of tasks.

Project Suitability Checklist

Check	Yes / No	Notes
I can describe a real problem or investigation.		
I know who the intended user is.		
The project uses at least one substantial algorithm or data structure.		
The project is more complex than a simple menu-based database.		
I can collect or create suitable test data.		
I can build and test the project in stages.		
I can explain how the final solution will be evaluated.		