

Assessment Report

New Zealand Scholarship Digital Technologies 2025

Performance standard 93604

General commentary

The Scholarship Digital Technologies assessment required candidates to demonstrate high-level algorithmic thinking and the ability to solve complex computational problems through rigorous analysis. Candidates were required evaluate various algorithmic paradigms – such as sorting, pathfinding, and optimisation – by accurately implementing solutions in a programming language and providing technical justifications for their chosen solutions.

A significant focus is placed on computational complexity, requiring candidates to determine time and space complexities (Big O) and explain how these metrics impact performance across different data scenarios.

To attain Scholarship and Outstanding Scholarship levels, candidates needed to provide insightful comparisons between alternative methods, such as identifying the limitations of greedy approaches compared to more robust strategies like dynamic programming.

Ultimately, the assessment rewarded convincing communication and the ability to critically evaluate the efficiency, scalability, and practical implications of diverse digital solutions.

Assessment

The 2025 Scholarship Digital Technologies assessment required candidates to evaluate and implement algorithmic solutions across three distinct technical domains: sorting, pathfinding, and optimisation. In the first section, candidates analysed the inefficiencies of a basic Bubble Sort implementation – specifically its $O(n^2)$ complexity and lack of early-termination optimisations – and provided $O(n \log n)$ alternatives such as QuickSort or MergeSort.

The second section focused on grid-based navigation, where candidates are expected to implement a Breadth-First Search (BFS) using a FIFO queue and a visited set to ensure $O(n^2)$ time and space efficiency. This section also tested the ability to transition to Dijkstra's algorithm when the grid introduces weighted traversal costs.

Finally, the Perfect Pastry Packing problem necessitated the use of Dynamic Programming (DP), either through a bottom-up approach or top-down memorisation (where previously computed subproblem results are stored and reused to avoid repeated computation), to achieve an $O(N \cdot B)$ solution. A critical component of this final task was explaining why greedy strategies fail to guarantee optimality, as they cannot account for future subproblem combinations.

Report on performance standard

Candidates who were awarded Scholarship with **Outstanding Performance** commonly:

- demonstrated precise implementation by writing accurate, optimal algorithms – such as QuickSort, Breadth-First Search (BFS), and bottom-up Dynamic Programming (DP) – while providing correct manual execution of logic and formulas

- justified data structure selection by providing technical justifications for every data structure used, such as the necessity of FIFO queues for level-by-level exploration and visited sets to prevent redundant processing
- applied rigorous complexity in analysis through accurate identification of both time and space complexities – for example, $O(n^2)$ for grids or $O(N \cdot B)$ for DP – including the identification of dominant terms and analysing performance across increasing input sizes
- recognised algorithmic transitions through their insights by identifying when problem constraints changed, such as recognising that adding traversal costs to a grid necessitates a shift from BFS to Dijkstra's algorithm
- evaluated alternatives with insight, including explaining why greedy strategies fail to guarantee optimality and comparing algorithms based on factors such as stability or cache performance
- delivered persuasive and clear technical explanations, breaking down complex recursive or iterative processes into logical steps that showed a comprehensive understanding of the problem space.

Candidates who were awarded **Scholarship** commonly:

- were successful in implementing standard algorithms with functional and accurate code for complex processes, including $O(n \log n)$ sorts (QuickSort or MergeSort), Breadth-First Search (BFS), and Dynamic Programming (DP) solutions, though implementations may have contained minor errors or lacked detailed documentation
- identified computational complexity by accurately determining the Big-O time complexity for various algorithms – such as $O(n^2)$ for nested loops and $O(N \cdot B)$ for optimisation tasks – and correctly applied these formulas to calculate costs for specific datasets
- performed manual execution with accuracy by demonstrating a clear understanding of algorithmic logic; they provided correct manual traces (dry runs) and final outputs for given inputs, such as pathfinding coordinates or sorted arrays
- recognised algorithmic inefficiencies where they identified the limitations of basic approaches, such as the high cost of non-optimised Bubble Sort on already sorted data, or they stated the inability of greedy algorithms to find the global optimum in the Perfect Pastry Packing problem
- utilised essential data structures in an accurate manner, such as queues for FIFO exploration in BFS and arrays / sets to track visited nodes to prevent infinite loops
- provided sound technical explanations of how algorithms function and their real-world advantages and disadvantages, demonstrating a solid grasp of the subject matter beyond simple coding.

Candidates who were **not awarded Scholarship** commonly:

- provided sub-optimal implementations where they often relied on less efficient approaches, such as Brute Force or Depth-First searches, rather than the required optimal solutions like Dynamic Programming or Breadth-First Search
- had written code or pseudocode which frequently included minor logic errors that hindered the algorithm's accuracy or efficiency
- demonstrated poor documentation where they often lacked clear commenting or failed to explain the logic and purpose of specific code blocks
- lacked essential data structure implementation in pathfinding tasks, where they did not use or correctly comment on the use of a queue, which is fundamental for an effective BFS
- offered superficial technical analysis or lacked the 'perception and insight' found in successful responses, although they may have identified a time complexity like $O(n^2)$
- provided incomplete evaluations; for example, they often identified that a greedy algorithm might fail, but provided limited examples or shallow comparisons when evaluating alternative solutions

- presented analysis that was frequently limited to time complexity, with little to no mention of how the algorithm utilised system memory or scaled in terms of space
- included explanations that were often descriptive rather than persuasive, which did not clearly justify why a specific approach was the most appropriate for the given constraints.