VCE Algorithmics (HESS) 2023-2026 Implementation on-demand video

Video 3
Background to the Unit 3 Outcome 2 SAC

VICTORIAN CURRICULUM AND ASSESSMENT AUTHORITY
VICTORIA State Government

Acknowledgement of Country

The VCAA respectfully acknowledges the Traditional Owners of Country throughout Victoria and pays respect to the ongoing living cultures of First Peoples.

VCE Algorithmics (HESS) 2023-2026 Implementation on-demand video

Video 3
Background to the Unit 3 Outcome 2 SAC

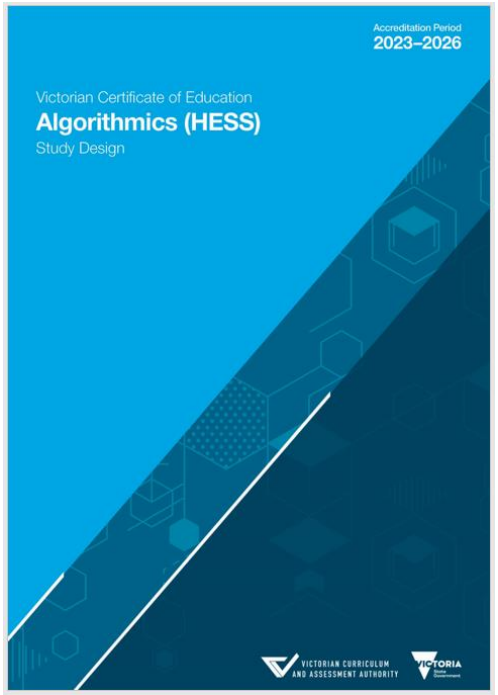
VICTORIAN CURRICULUM AND ASSESSMENT AUTHORITY
VICTORIA State Government

Purpose of this presentation

This presentation will cover:

- The study design
- Area of Study statement
- Outcome statement
- The assessment task
- Key knowledge
- Key skills
- Performance descriptors
- Advice for teachers

Resources



- This is the new study design.
- It will be accredited for 2023–2026.
- This is available on the Algorithmics (HESS) study page right now.
- Unit 3 Outcome 2 is now a School-assessed Coursework (SAC) task.

Unit 3 Area of Study 2

Algorithm design

In this area of study, students learn how to formalise processes as algorithms and to execute them automatically. They use the language of algorithms to describe general approaches to problem-solving and to give precise descriptions of how specific problems can be solved. Students learn how to decompose problems into smaller parts that can be solved independently. This forms the basis of modularisation. Students explore a variety of problem-solving strategies and algorithm design patterns. Students explore example applications of these design patterns and learn about their implications for efficiently solving problems. They learn about recursion as a method for constructing solutions to problems by drawing on solutions to smaller instances of the same problem.

Students are required to implement algorithms as computer programs. The programming language used must explicitly support the ADTs listed in the key knowledge in Area of Study 1 either directly or by using a library.

Unit 3 Outcome 2 – The outcome

On completion of this unit the student should be able to define and explain algorithmic design principles, design algorithms to solve information problems using basic algorithm design patterns, and implement the algorithms.

Unit 3 Outcome 2 – The assessment task

Contribution to final assessment

School-assessed Coursework for Unit 3 will contribute 12 per cent to the study score.

Outcomes	Marks allocated	Assessment tasks
Outcome 1 Define and explain the representation of information using abstract data types, and devise formal representations for modelling various kinds of real-world information problems using appropriate abstract data types.	50	In response to given stimulus material, create one or more designs of a data model using abstract data types to capture the salient aspects of a real-world information problem.
Outcome 2 Define and explain algorithmic design principles, design algorithms to solve information problems using basic algorithm design patterns, and implement the algorithms.	50	In response to given stimulus material: • create one or more designs of algorithms that apply algorithm design patterns or select appropriate graph algorithms to solve information problems • implement an algorithm.
Total marks	100	

Key knowledge

- basic structure of algorithms
- pseudocode concepts, including variables and assignment, sequence, iteration, conditionals and functions
- programming language constructs that directly correspond to pseudocode concepts
- conditional expressions using the logical operations of AND, OR, NOT
- recursion and iteration and their uses in algorithm design
- modular design of algorithms and ADTs
- characteristics and suitability of the brute-force search and greedy algorithm design patterns
- graph traversal techniques, including breadth-first search and depth-first search
- specification, correctness and limitations of the following graph algorithms:
  - Prim's algorithm for computing the minimal spanning tree of a graph
  - Dijkstra's algorithm and the Bellman-Ford algorithm for the single-source shortest path problem
  - the Floyd-Warshall algorithm for the all-pairs shortest path problem and its application to the transitive closure problem
  - the PageRank algorithm for estimating the importance of a node based on its links
- induction and contradiction as methods for demonstrating the correctness of simple iterative and recursive algorithms

Key skills

- interpret pseudocode and execute it manually on given input
- write pseudocode
- identify and describe recursive, iterative, brute-force search and greedy design patterns within algorithms
- design recursive and iterative algorithms
- design algorithms by applying the brute-force search or greedy algorithm design pattern
- write modular algorithms using ADTs and functional abstractions
- select appropriate graph algorithms and justify the choice based on their properties and limitations
- explain the correctness of the specified graph algorithms
- use search methods on decision trees and graphs to solve planning problems
- implement algorithms, including graph algorithms, as computer programs in a very high-level programming language that directly supports a graph ADT
- demonstrate the correctness of simple iterative or recursive algorithms using structured arguments that apply the methods of induction or contradiction

VCAA Performance descriptors

ALGORITHMICS (HESS) UNIT 3 OUTCOME 2
SCHOOL-ASSESSED COURSEWORK

Performance descriptors

	DESCRIPTOR: typical performance in each range				
	Very low	**Low**	**Medium**	**High**	**Very high**
Unit 3 *Outcome 2* **On completion of this unit the student should be able to define and explain algorithmic design principles, design algorithms to solve information problems using basic algorithm design patterns, and implement the algorithms.**	Identifies limited elements of sequence, selection and repetition in a given algorithm.	Interprets simple structured pseudocode with sequence, selection and simple iteration with minimal errors.	Interpret, write and execute pseudocode, including pseudocode with nested iteration.	Interprets correctly and identifies cases where the given pseudocode does not perform the desired operation correctly.	Interprets correctly and improves a piece of pseudocode by restructuring and modularisation using non-trivial user-defined functions.
	Designs simple algorithms and writes these in pseudocode with significant task scaffolding required, and the final algorithm is only an effective method for a trivial subset of problem instances. Identifies some algorithm design approaches.	Designs simple algorithms and writes these in pseudocode, with some task scaffolding. The algorithm is an effective method for a non-trivial subset of problem instances. Explains the principles of the brute-force search or greedy algorithm design patterns, utilising appropriate examples.	Designs and applies algorithms including use of iteration and writes these in pseudocode with minimal errors.	Designs algorithms using iteration and recursion for problems that have a structure that does not allow for the direct application of one of the studied algorithms. Explains the attributes required of problems for one of the algorithm design patterns to be applied.	Designs algorithms using iteration, recursion and non-trivial functions for problems that have a structure that does not allow for the direct application of one of the studied algorithms.
		Improves a piece of pseudocode by restructuring and modularisation.	Writes modular algorithms. Improves a piece of pseudocode by restructuring and modularisation using nontrivial user-defined functions. Applies a given algorithm design pattern to design an algorithm to solve a problem.		Selects a suitable algorithm design patterns for solving an information problem and applies the design patterns to design an algorithm to solve the problem.
	Names and states correctly the computational applications of most of the specified graph algorithms. States informally the input types of the specified graph algorithms.	Explains informally how some of the specified graph algorithms perform their computation and writes the approximate pseudocode for these.	Selects and explains graph algorithms. Selects a suitable graph algorithm to apply to solve a complex problem.	Executes, without error, any of the specified graph algorithms using manual techniques for complex graphs.	Selects with detailed justification a suitable graph algorithm to apply to solve a complex problem Explains in precise terms why any of the specified graph algorithms are not valid for some classes of graph or graphs with certain properties.
			States precisely the input types of the specified graph algorithms.		
	Identifies an appropriate search algorithm to apply to a given problem.	Compares the execution of depth-first search (DFS), breadth-first search (BFS) on a specific problem instance.	Uses search methods. Compares the relative advantages of the different graph traversal techniques.		
	Implements simple algorithms with sequential, conditional and iterative elements.	Implements simple iterative algorithms that utilise collection ADTs.	Implements graph traversal algorithms and simple recursive algorithms and applies these to solve particular problem instances.	Implements shortest-path graph algorithms and applies them to solve particular problem instances.	
	Limited and unstructured arguments given for correctness of graph algorithms.	Describes an argument for the correctness of a graph algorithm that considers only the correctness of a specific example.	Demonstrates the correctness of specified graph algorithms using induction and contradiction for some input cases.	Describes an argument for the correctness of one of the specified graph algorithms that considers the general case of the problem, but not all steps in the chain of argument are explained.	Describes a valid argument for the correctness of at least one of the specified graph algorithms using either the induction or contradiction methods.

Advice for teachers

- Overview of Unit 3: Algorithmic problem solving
- Unit 3 Outcome 2
  - Teaching and learning activities
  - Detailed examples
  - Sample approaches to developing an assessment task
  - Performance descriptors
- Unit 3 Sample weekly planner

Contact

- **Phil Feain – Digital Technologies Curriculum Manager (VCAA)**
- **Ph: (03) 9059 5146**
- **Philip.Feain@education.vic.gov.au**

Authorised and published by the
Victorian Curriculum and Assessment Authority