

VCE Algorithmics (HESS) (2023–2026)

Implementation webinar



VICTORIAN CURRICULUM
AND ASSESSMENT AUTHORITY



VCE Algorithmics (HESS) (2023–2026)

Implementation webinar

Georgia Gouros
Virtual School Victoria

&

Phil Feain
Digital Technologies Curriculum Manager
VCAA



VICTORIAN CURRICULUM
AND ASSESSMENT AUTHORITY



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.



Outline of this presentation

- Changes to Algorithmics (HESS)
- Changes to School-based Assessment
- Unit 3 Outcomes 1–3
- Unit 4 Outcomes 1–3
- Support material (formally Advice for teachers)

Changes to Algorithmics (HESS)

Minor changes:

- Unit 3 Area of Studies 1–3
- Unit 4 Area of Studies 1–2

Major changes:

- Unit 4 Area of Study 3
- School-based Assessment (from 2 SATs to 3 SACs and 1 SAT(3 outcomes))

Changes to School-based Assessment

- The 2023 – 2026 study design now has different assessment tasks.
- It now has **three** School-assessed Coursework tasks (SACs):
 - Unit 3 Outcome 1
 - Unit 3 Outcome 2
 - Unit 4 Outcome 3
- And **one** School-assessed Task (SAT):
 - Unit 3 Outcome 3
 - Unit 4 Outcome 1
 - Unit 4 Outcome 2

Changes to School-based Assessment

Percentage contributions to the study score in VCE Algorithmics (HESS) are as follows:

- Units 3 and 4 School-assessed Coursework: 20 per cent
- Units 3 and 4 School-assessed Task: 20 per cent
- End-of-year examination: 60 per cent.

Question:

**How many of you will be teaching
Algorithmics for the first time next
year?**

Unit 3: Algorithmic problem solving

Unit 3 Area of Study 1

Data modelling with abstract data types

In this area of study, students develop and apply knowledge and skills in data abstraction. Students consider the structure of information through a study of the definition and properties of abstract data types (ADTs). They select appropriate ADTs and use them to model salient aspects of real-world problems. Students study a variety of collection-based data types, with a particular focus on the graph ADT, which encapsulates a set of nodes along with their interconnections. Students explore how graph ADTs can be applied to network problems, such as social or transport network problems, and planning problems.

Unit 3 Outcome 1

On completion of this unit the student should be able to 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.

Unit 3 Outcome 1 SAC task

Outcome 1	Marks allocated	Assessment task
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.

Sample U3 O1 SAC content

- Create a data model using abstract data types to represent the characters and the relationship of the characters in novel XYZ.
- Explain how your data model captures the details of the characters and the relationship between the characters.

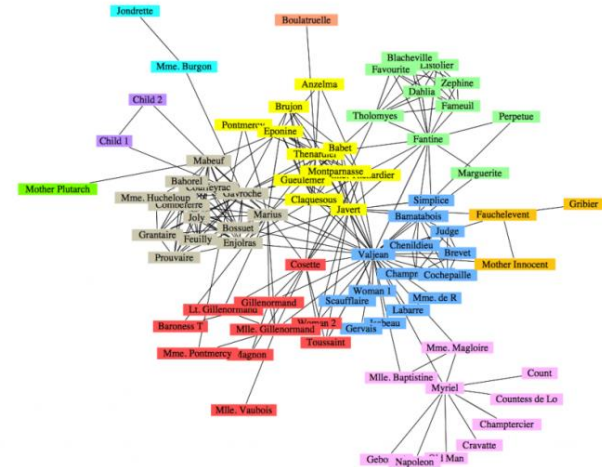


FIG. 12: The network of interactions between major characters in the novel *Les Misérables* by Victor Hugo. The greatest modularity achieved in the shortest-path version of our algorithm is $Q = 0.54$ and corresponds to the 11 communities represented by the colors.

<https://studentwork.prattsi.org/infovis/visualization/les-miserables-character-network-visualization/>

U3 O1 Draft Performance descriptors

Very low	Low	Medium	High	Very high
Identifies some motivations for the abstraction of data.	Discusses how an ADT property could be used to model an aspect of a particular problem. Little discrimination is demonstrated when identifying features of the problem.	Explains the role of ADTs for data modelling.	Describes in detail the suitability of appropriate ADTs for creating a model in a given problem context.	Compares and justifies the selection of appropriate ADTs for creating a model and outlines limitations of different representations.
Uses limited metalanguage when describing ADTs.	Executes a sequence of ADT operations to a given ADT instance.	Reads, writes and uses ADTs.	Writes complete signature specifications for several ADTs, fully in appropriate metalanguage.	Specifies a non-trivial new operation for one of the standard ADTs to meet requirements that cannot be satisfied by the standard definition.
Limited use of terminology in describing graph properties.	Confirms or rejects the properties of a graph given as a diagram.	Identifies and describes the properties of graphs.	Analyses the interconnections between the properties of graphs using correct terminology.	Analyses the properties satisfied by a given graph and derives another graph property using as evidence the existing specified properties of graphs.
Identifies an example problem attribute that could be modelled by a graph node or edge.	Discusses some aspects of a problem, including planning problems from a given data model instance.	Applies ADTs to real-world problems. The full range of problem instances can be represented.	Models and fully represents a specific problem instance as a data model using a combination of ADT representations.	Models fully a specific problem instance as a data model with a combination of ADTs, and appropriately justifies the assigned priority of several aspects of the problem to the specific context of the problem.
Scaffolding is required to create a basic model.	Some aspects of the problem are modelled.	Models basic network and planning problems with graphs.	Models and fully represents planning problems using the graph ADT in combination with other ADTs where appropriate.	Models fully and justifies the priorities used in the representation of the planning problem using graph and other ADTs.

This is a **draft** and subject to change.

This will be included in the Support material.

Question:

Any questions regarding U3 O1?

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

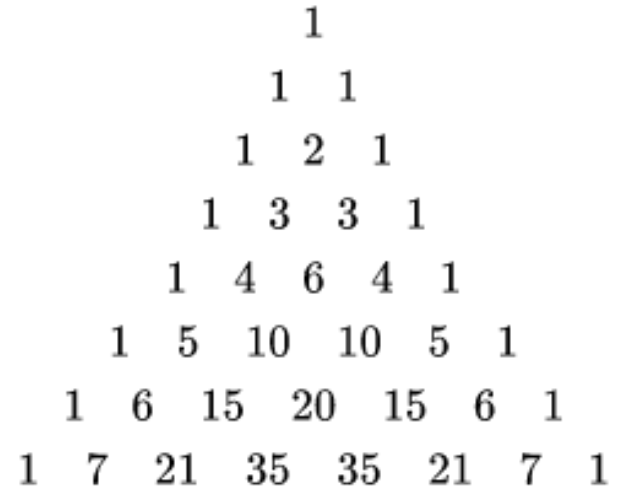
On completion of this unit the student should be able to define and explain algorithmic design principles, design algorithms to solve information problems using basic algorithm design patterns, and implement the algorithms.

Unit 3 Outcome 2 SAC task

Outcome 2	Marks allocated	Assessment task
Define and explain algorithmic design principles, design algorithms to solve information problems using basic algorithm design patterns, and implement the algorithms.	50	<p>In response to given stimulus material:</p> <ul style="list-style-type: none">• create one or more designs of algorithms that apply algorithm design patterns or select appropriate graph algorithms to solve information problems• implement an algorithm.

Sample U3 O2 SAC content

- Write and implement an algorithm to generate a Pascal's triangle to level k as a hierarchical connected tree.
- Implement your algorithm in a coding language.
 - Implement algorithms in a coding language with a graph library SnapApps/Edgy, Python/NetworkX



https://en.wikipedia.org/wiki/Pascal%27s_triangle

U3 O2 Draft Performance descriptors

This is a **draft** and subject to change.

This will be included in the Support material.

Very low	Low	Medium	High	Very high
<p>Identifies limited elements of sequence, selection and repetition in a given algorithm.</p> <p>Manually executes a simple sequence of simple steps within pseudocode.</p>	<p>Interprets simple structured pseudocode with sequence, selection and simple iteration with minimal errors.</p> <p>Identifies some recursive, iterative, brute-force search design pattern and greedy design pattern features within pseudocode for an algorithm.</p> <p>Describes some elements of the concept of modularisation.</p>	<p>Interprets and manually executes pseudocode containing nested iteration and the use of ACTs.</p> <p>Describes the concepts of the recursive, iterative, brute-force search and greedy design patterns.</p> <p>Describes how the concept modularisation has been applied within a particular piece of pseudocode for an algorithm.</p>	<p>Interprets and manually executes pseudocode containing complex use of ACTs or simple recursion.</p> <p>Describes the recursive, iterative, brute-force search or greedy design pattern have been applied within a particular piece of pseudocode for an algorithm.</p>	<p>Interprets and manually executes pseudocode containing complex use of ACTs and recursion.</p> <p>Completely and precisely describes how the recursive, iterative, brute-force search or greedy design pattern have been applied within a particular piece of pseudocode for an algorithm.</p>
<p>Designs simple algorithms and writes these in pseudocode with significant task scaffolding required, and the final algorithm is only an effective method for a non-trivial subset of problem instances.</p> <p>Identifies some algorithm design approaches.</p>	<p>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.</p> <p>Explains the principles of the brute-force search or greedy algorithm design patterns, utilising appropriate examples.</p>	<p>Designs and applies algorithms including use of iteration and writes the pseudocode with minimal errors.</p> <p>Designs modular algorithms.</p> <p>Applies a given algorithm design pattern to design an algorithm to solve a context.</p>	<p>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.</p> <p>Explains the attributes of algorithm design patterns that are applied.</p>	<p>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.</p> <p>Selects suitable algorithm design patterns for solving information problems and applies the design patterns to design algorithms and find solutions.</p>
<p>Names and states correctly the computational applications of most of the specified graph algorithms.</p> <p>States informally the input types of the specified graph algorithms.</p>	<p>Explains informally how some of the specified graph algorithms perform their computation and utilises approximate pseudocode.</p>	<p>Selects and explains the input type of the specified graph algorithms.</p> <p>Selects a suitable algorithm to apply to solve a simple problem.</p> <p>Explains the input type of the specified graph algorithms.</p>	<p>Calculates, without error, any of the specified graph algorithms using manual techniques for complex graphs.</p>	<p>Justifies a selection of a suitable graph algorithm for solving a complex problem based on the properties and limitations of the algorithm.</p> <p>Explains in precise terms why any of the specified graph algorithms are not valid for some classes of graph or graphs with certain properties.</p>
<p>Identifies the first few nodes visited by either the breadth-first or depth-first search algorithm when applied to a decision tree.</p>	<p>Executes the breadth-first or depth-first search algorithm on a decision tree.</p>	<p>Applies a given graph search method to a decision tree to solve a planning problem.</p>	<p>Selects a suitable graph search method and applies it to a decision tree to solve a planning problem.</p>	<p>Evaluates the relative advantages of different graph search methods for solving a planning problem.</p>
<p>Implements simple algorithms with sequential, conditional and iterative elements.</p>	<p>Implements simple iterative algorithms that utilise recursive ACTs.</p>	<p>Implements graph traversal algorithms and simple recursive algorithms.</p> <p>Applies an implementation of a simple iterative algorithm that utilises ACTs to solve a particular problem instance.</p>	<p>Implements shortest-path graph algorithms.</p> <p>Applies an implementation of a graph traversal algorithm or simple recursive algorithm to solve particular problem instances.</p>	<p>Efficiently implements shortest-path graph algorithms and applies them to solve particular problem instances.</p>
<p>Limited and unstructured arguments given for correctness of graph algorithms.</p>	<p>Describes an argument for the correctness of a graph algorithm that considers only the correctness of a specific example.</p>	<p>Demonstrates the correctness of specified graph algorithms using induction and contradiction for some input cases.</p>	<p>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.</p>	<p>Describes a valid argument for the correctness of at least one of the specified graph algorithms using either the induction or contradiction methods.</p>

Question:

Any questions regarding U3 O2?

Unit 3 Area of Study 3

Applied algorithms

In this area of study, students combine their knowledge of data modelling and algorithm design to solve real-world problems. Students consider a variety of algorithms and ADTs before selecting a suitable combination. They justify their chosen combination of algorithms and data types relative to other possible choices. Typically the fitness of a chosen combination could be measured in terms of the selection of salient features to achieve an appropriate level of abstraction and the quality of result produced by the algorithm.

Unit 3 Outcome 3

On completion of this unit the student should be able to design suitable solutions for real-world problems that require the integration of algorithms and data types, including the communication of solutions and their justification.

Unit 3 Outcome 3 SAT task

Outcome 3

Design suitable solutions for real-world problems that require the integration of algorithms and data types, including the communication of solutions and their justification.

Assessment task

The design of a data model and algorithm combination to solve a real-world/applied problem, including:

- a specification of the problem
- a consideration of multiple solution options
- the selection of a suitable, coherent, clear and fit-for-purpose solution

Sample U3 O3 SAT content

- Design an algorithmic solution for the Travelling Archaeologist problem as defined by these constraints
- Data model {graph, nodes=islands, edges=ferry routes, dictionary holds ferry time-table}
- Algorithms {shortest path adaptations, best first search adaptations}
- Justify your selection of abstract data types and algorithms in your solution.



https://en.wikipedia.org/wiki/List_of_islands_of_Greece

U3 O3 Draft SAT Criteria (SAT – Part 1)

1. Skills in understanding and modelling a real-world problem.
2. Skills in the selection and design of algorithms to solve a real-world problem.
3. Skills in the implementation of algorithmic solutions to a real-world problems.
4. Skills in the evaluation of algorithmic solutions to a real-word problem.

This is a **draft**
and subject to
change.

Question:

Any questions regarding U3 O3?

Unit 4: Principles of algorithmics

Unit 4 Area of Study 1

Formal algorithm analysis

In this area of study, students investigate the efficiency of algorithms using mathematical techniques. Students learn how some computable problems require such a large amount of resources that in practice it is not possible to solve these exactly for realistic problem sizes. Students examine specific, widely occurring instances of such problems and the reasons why these problems cannot be solved. Students analyse time complexity formally and informally, while they study space complexity as a general concept. Students are not expected to derive the space complexity of algorithms.

Unit 4 Outcome 1

On completion of this unit the student should be able to establish the efficiency of simple algorithms and explain soft limits of computability.

Unit 4 Outcome 1 SAT task

Outcome 1

Establish the efficiency of simple algorithms and explain soft limits of computability.

Assessment task

A formal time complexity analysis of the designed algorithm for the applied problem and an explanation of the consequences of these results on the algorithm's real-world application.

Sample U4 O1 SAT content

- Establish the time complexity of your solution for the Travelling Archaeologist problem as defined by these constraints
- Data model {graph, nodes=islands, edges=ferry routes, dictionary holds ferry time-table}
- Algorithms {shortest path adaptations, best first search adaptations}



https://en.wikipedia.org/wiki/List_of_islands_of_Greece

U4 O1 Draft SAT Criteria (SAT – Part 2)

1. Skills in the classification of problems.
2. Skills in determining the time complexity of algorithms.
3. Skills in the comparative formal analysis of algorithmic solutions to a real-world problem.

This is a **draft**
and subject to
change.

Question:

Any questions regarding U4 O1?

Unit 4 Area of Study 2

Advanced algorithm design

In this area of study, students examine more advanced algorithm design patterns. Students learn how to select algorithmic approaches from a wider range of options, depending on the structure of the problem that is being addressed. They investigate how some problems are solvable in principle while being intractable in practice. They explore examples of such problems with real-world relevance and learn how such problems can be tackled by computing near-optimal solutions.

Unit 4 Outcome 2

On completion of this unit the student should be able to solve a variety of information problems using algorithm design patterns and explain how heuristics can address the intractability of problems.

Unit 4 Outcome 2 SAT task

Outcome 2

Solve a variety of information problems using algorithm design patterns and explain how heuristics can address the intractability of problems.

Assessment task

A design of an improved data model and algorithm combination to solve the applied problem, including:

- the selection of an efficient, coherent and fit-for-purpose solution
- a time complexity analysis
- a comparison to the original solution.

Sample U4 O2 SAT content

- Improve your solution for the Travelling Archaeologist problem as defined by these constraints {and these new constraints}.....
 - Data model {graph, nodes=islands, edges=ferry routes, dictionary holds ferry time-table}
 - Algorithms {shortest path adaptations, best first search adaptations}
- Compare your solution to the original solution.



https://en.wikipedia.org/wiki/List_of_islands_of_Greece

U4 O2 Draft SAT Criteria (SAT – Part 3)

1. Skills in the selection and design of advanced algorithms to solve a real-world problem.
2. Skills in the implementation of advanced algorithmic solutions to a real-world problems.
3. Skills in the evaluation of advanced algorithmic solutions to a real-word problem.

This is a **draft**
and subject to
change.

Question:

Any questions regarding U4 O2?

Unit 4 Area of Study 3

Computer science: past and present

In this area of study, students examine the emergence of computer science as a field and the philosophical and technical ideas that support the emergence of modern artificial intelligence (AI). They explore how the quest to develop methods for mathematical proof led to the proof that there exist problems that may not be computed automatically. Students investigate how machine learning algorithms learn from data and engage with several conceptions of artificial intelligence and whether it is possible. They examine and discuss some of the ethical issues posed by the application of data-driven algorithms. Students are not required to produce proofs or formal explanations concerning undecidability.

Unit 4 Outcome 3

On completion of this unit the student should be able to explain the historical context for the emergence of computer science as a field and discuss modern machine learning techniques and the philosophical issues they raise.

Unit 4 Outcome 3 SAC task

Outcome 3	Marks allocated	Assessment task
Explain the historical context for the emergence of computer science as a field and discuss modern machine learning techniques and the philosophical issues they raise.	50	Select at least one task from the following: <ul style="list-style-type: none">• a response to a case study or stimulus material• a written report• an annotated visual report• an oral report• structured questions.

Sample U4 O3 SAC content

Support Vector Machines - What are they? How do they learn to make decisions?

What is a hyperplane? Give an example to explain how it is used.

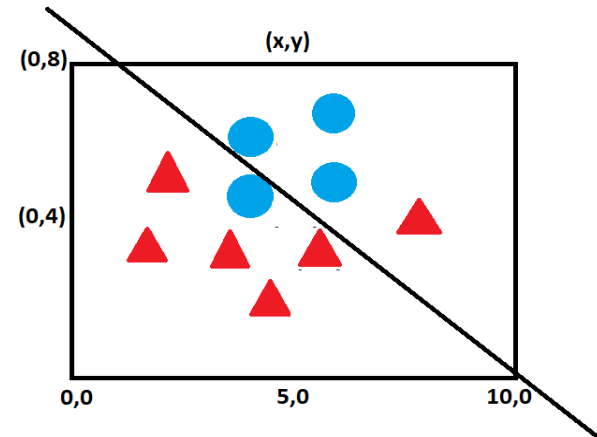
What are Support Vectors?

Show the hyperplane and support vectors for the following input data representation(s).

Sample U4 O3 SAC content

The XYZ company came up with the following SVM hyperplane as shown in the diagram below to make the decision. What are the issues with the result?

- **Technical concerns:**
 - Underfitting
 - Not enough training data
- **Ethical concerns:**
 - Opacity concerns
 - Human interaction concerns
 - Privacy concerns
 - Effects of AI concerns



U4 O3 Draft Performance descriptors

Very low	Low	Medium	High	Very high
Very limited description of the historical connections between the foundational crisis in mathematics and the origin of computer science.	Some explanation of the historical connections between the foundational crisis in mathematics and the origin of computer science.	Satisfactory explanation of the historical connections between the foundational crisis in mathematics and the origin of computer science.	Detailed explanation of the historical connections between the foundational crisis in mathematics and the origin of computer science.	Thorough explanation of the historical connections between the foundational crisis in mathematics and the origin of computer science.
Very limited description of the concept of undecidability and its implications for the limits of computation.	Some explanation of the concept of undecidability and its implications for the limits of computation.	Satisfactory explanation of the concept of undecidability and its implications for the limits of computation.	Detailed explanation of the concept of undecidability and its implications for the limits of computation.	Thorough explanation of the concept of undecidability and its implications for the limits of computation.
Very limited description of philosophical conceptions of artificial intelligence and the Chinese Room Argument.	Some explanation of philosophical conceptions of artificial intelligence and the Chinese Room Argument.	Satisfactory explanation of philosophical conceptions of artificial intelligence and the Chinese Room Argument with some analysis of these ideas.	Detailed explanation of philosophical conceptions of artificial intelligence and the Chinese Room Argument with a clear analysis of these ideas.	Thorough explanation of philosophical conceptions of artificial intelligence and the Chinese Room Argument with a sophisticated analysis of these ideas.
Very limited description of data-driven algorithms, including support vector machines and neural networks.	Some explanation of data-driven algorithms, including support vector machines and neural networks.	Satisfactory explanation of data-driven algorithms, including support vector machines and neural networks.	Detailed explanation of data-driven algorithms, including support vector machines and neural networks.	Thorough explanation of data-driven algorithms, including support vector machines and neural networks.
Very limited description of ethical issues related to artificial intelligence and data-driven algorithms.	Some explanation of ethical issues related to artificial intelligence and data-driven algorithms.	Some analysis of ethical issues related to artificial intelligence and data-driven algorithms with a limited synthesis of how these can result from the training of algorithms using data.	Clear analysis of ethical issues related to artificial intelligence and data-driven algorithms with some synthesis of how these can result from the training of algorithms using data.	Sophisticated analysis of ethical issues related to artificial intelligence and data-driven algorithms with a detailed synthesis of how these can result from the training of algorithms using data.

This is a **draft** and subject to change.

This will be included in the Support material.

Question:

Any questions regarding U4 O3?

Support material
(formally Advice for teachers)

Support material

- Overview of Unit 3: Algorithmic problem solving
 - Teaching and learning activities
 - Detailed examples
 - Sample approaches to developing an assessment task
 - Unit 3 Outcome 1
 - Unit 3 Outcome 2
 - School-assessed Task
 - Unit 3 Outcome 3
 - Performance descriptors
 - Unit 3 Outcome 1
 - Unit 3 Outcome 2
 - Unit 3 Sample weekly planner
- Overview of Unit 4: Principles of algorithmics
 - Teaching and learning activities
 - Detailed examples
 - Sample approaches to developing an assessment task
 - Unit 4 Outcome 3
 - School-assessed Task
 - Unit 4 Outcome 1
 - Unit 4 Outcome 2
 - Performance descriptors
 - Unit 4 Outcome 3
 - Unit 4 Sample weekly planner

Question:

**Any questions regarding the
Support material?**

Purpose of this presentation

- Changes to Algorithmics (HESS)
- Changes to School-based Assessment
- Unit 3 Outcomes 1–3
- Unit 4 Outcomes 1–3
- Support material


Resources already available

Study design for implementation in 2023

Teachers are advised that the study design listed below is for use from 2023. This is available to teachers in preparation for the implementation of the new study in 2023. Additional resources will be added progressively as they become available.

▼ 2023 Implementation

For accreditation period 2023-2026

-  [VCE Algorithmics \(HESS\) Study Design](#) for implementation in 2023.
VCE Algorithmics (HESS) contains Units 3 and 4 only.
- [Implementation of VCE Algorithmics \(HESS\) Study Design Units 3 and 4: 2023-2026.](#)
Online video presentations which provide teachers with an overview of the VCE Algorithmics (HESS) Study Design and other relevant VCAA documents that can be used to plan their teaching and learning programs

Contact

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

Questions