2018 VCE Algorithmics (HESS) examination report

General comments

The 2018 Algorithmics (HESS) examination contained two sections. Section A was composed of 20 multiple-choice questions and Section B was composed of 18 short-answer and extended-answer questions. Students achieved scores across the range of available marks, with high-scoring students demonstrating an impressive grasp of the examination content.

The majority of students attempted most questions, demonstrating both good time management and thorough engagement with the study design.

Students demonstrated a strong capacity for writing pseudocode in response to unseen contexts (Questions 7, 9c., 18a. and 18b.), with few students adhering too closely to the syntax of a programming language. Where programming language syntax was evident, it was mostly Python syntax, which is generally acceptable due to its easy readability. Students are cautioned, however, about using convenient Python functions. For instance, in Question 9b., some students used the mean() function, which is not a standard operation on arrays.

While most students showed a good breadth of knowledge, it was evident that some students relied on memorisation rather than understanding. This was the case in Questions 5, 10c., 15, 16 and 17, where a significant number of students were able to name a relevant problem or algorithm, but were unable to demonstrate an understanding of how it could be solved or applied. Students are encouraged to work on developing deeper understanding of key problems in the study by practising describing appropriate algorithms for these problems in words and implementing these algorithms in a programming language to solidify their understanding.

A number of questions on the examination asked students to justify their response, with Questions 3a., 6b., 10c., 14a., 16, 18c. and 18d. either directly or indirectly calling for a justification as part of an answer. Often this was the element that students found difficult. Another common source of error was the inability to articulate how a chosen abstract data type (ADT) would model the given problem; for example, where the chosen ADT was a graph, some students did not clearly state what the nodes and edges would represent.

Specific information

Note: Student responses reproduced in this report have not been corrected for grammar, spelling or factual information.

This report provides sample answers or an indication of what answers may have included. Unless otherwise stated, these are not intended to be exemplary or complete responses.

The statistics in this report may be subject to rounding resulting in a total more or less than 100 per cent.

Section A – Multiple-choice questions

The table below indicates the percentage of students who chose each option. The correct answer is indicated by shading.

Question	% A	% B	% C	% D	% No answer	Comments
1	0	1	99	0	0	
2	41	16	31	10	1	Cobham theorised that problems that are feasibly computable (also known as easy problems) are those that are decidable in polynomial time.
3	3	0	6	91	0	
4	0	0	0	100	0	
5	13	67	12	7	0	
6	51	7	33	9	0	Both options A and option C were accepted.
7	6	72	5	16	0	
8	5	8	7	79	0	
9	5	20	10	65	0	
10	3	57	9	31	0	The recursive call is made as soon as an undiscovered neighbour is found, as in depth-first search.
11	92	2	5	1	0	
12	3	1	94	2	0	
13	26	67	2	5	0	Worst-case running time of an algorithm gives no insight about the best-case running time of the same algorithm, therefore Algorithm A could terminate in $O(1)$ time.
14	5	13	16	65	0	
15	2	41	53	4	0	Sergei will be caught when pedalling at 5 km/h, 10 km/h, 20 km/h and 40 km/h.
16	66	4	28	2	0	
17	5	9	82	3	0	
18	10	14	62	13	0	
19	12	65	16	6	0	
20	11	76	6	6	0	

Section B

Question 1

Marks	0	1	2	Average
%	6	36	58	1.5

A wide variety of responses were accepted. When describing similarities, some students made reference to both Turing machines and modern computers having a method for storage, while other students referred to the idea that modern computers are Turing complete. When describing differences, a number of responses alluded to Turing machines being a theoretical construct while modern computers are objects in the world, and other responses distinguished between the finite storage on a modern computer and the infinite tape of a Turing machine.

Students are advised to take care not to be too broad when comparing. For instance, the claim that Turing machines are simpler than modern computers could be referring to a number of different aspects, and so was not awarded a mark where it was unaccompanied by a brief elaboration.

Question 2

Marks	0	1	2	3	4	Average
%	22	15	25	23	15	2

The following is a possible solution with a minimal set of operations. Other operations were included by some students and accepted where valid. Additionally, a variety of nomenclature for the operations was accepted as long as it was unambiguous. For instance, the final operation listed was variously described in responses as *front*, *top*, *head*, *peek* and so on.

```
name:          priorityQueue

operations:    newpqueue:  → pqueue
               isEmpty: pqueue → boolean
               enqueue: pqueue x element x integer → pqueue
               dequeue: pqueue → pqueue
               front: pqueue → element
```

Most students demonstrated a familiarity with the general structure of ADT specifications and many students were able to include a number of relevant operations in their answer. Few responses were awarded full marks, with common mistakes including the absence of a constructor – that is, an operation that creates a new priority queue – and incorrect input and output spaces for a given operation.

Students are also reminded that in both priority queues and standard queues, the dequeue operation should return a queue. That is, the ADT specification for queues must include some way of amending the queue by removing the front element of the queue. This is distinct from the front or head operation, which simply returns the value of the front of the queue without amending the queue itself.

Question 3a.

Marks	0	1	2	3	Average
%	18	24	23	35	**1.8**

$$T(n) = \begin{cases} O(1) & if\ n = 1 \\ T(n-1) + O(1) & if\ n > 1 \end{cases}$$

$\therefore T(n) \in O(n)$

Most students were able to identify that $T(n) = T(n-1) + constant$; however, a significant number of students neglected to include a base case, while others wrote $T(1) = 0$, which indicates a lack of understanding of recurrence relations.

Of the students who had the correct recurrence relation, some seemed to recognise this as a standard $O(n)$ recurrence relation by inspection, while others attempted to use backward substitution, with varying degrees of success.

Question 3b.

Marks	0	1	Average
%	60	40	**0.4**

No, the algorithm always checks every element in the list and therefore best-case running time must be equal to worst-case running time.

Many students incorrectly suggested that the best-case running time would occur for a list of length 1. It is crucial to understand that when considering best-case, worst-case or average-case running time, the input size must stay consistent. Where best-case is better than worst-case for a given algorithm, it is because for a given input size, some structures of the input are conducive to faster execution than others.

Question 4a.

Marks	0	1	Average
%	29	71	**0.7**

$S(n) \in O(n^{log_3 2})$

Question 4b.

Marks	0	1	Average
%	28	72	**0.7**

$T(n) \in O(nlogn)$

Question 5

Marks	0	1	2	3	Average
%	58	19	17	6	**0.7**

This question was not answered well. Many students did not recognise that this problem was analogous to the knapsack problem, and of those who did recognise it, some failed to explicitly connect the initial cost and profit in each company to the weights and values of items in the knapsack problem.

Additionally, it was evident that a significant number of students had memorised key terms related to dynamic programming but had limited capacity to describe how a dynamic programming algorithm proceeds. While pseudocode is not expected for this sort of question, understanding the key recurrence relation in the dynamic programming solutions to standard problems (such as the knapsack or coin change problems) is crucial to being able to describe these solutions succinctly.

The following is an example of a high-scoring response.

> *Solve using the approach for the 1|0 knapsack problem. Through a bottom up approach with memorization, she could progressively find the most profitable combinations from T(0) to T(budget), at each stage setting T(n) as max$\{c_n, T(n-i)+T(i)$ and incrementing i by 1 at each iteration. Once T(budget) is obtained, she can backtrack to find the combination of most profitable companies to invest in for her budget, as each company can only be invested once.*

Question 6a.

Marks	0	1	2	3	Average
%	9	20	33	38	2

A wide variety of suitable graph ADTs were used by students in response to this question. Both undirected and directed graph ADTs were accepted, as long as there was a coherent justification included. The same applied to both unweighted and weighted graph ADTs.

Some students did not engage with the requirement in the question to justify a graph, while others did not include any description of what the nodes and edges in their graph would represent. Students are reminded that they must answer the question in its entirety and ensure they are specific about any ADT representation.

The following is an example of a high-scoring response.

> *A graph ADT where the shops and info kiosks are the nodes and the physical distance between them are weights on the undirected edges between nodes. This would allow graph traversal algorithms to be used to find things like shortest paths.*

Question 6b.

Marks	0	1	2	Average
%	28	32	40	1.1

For full marks, students needed to state that another node would be required to represent the additional information kiosk and explain how this new node would be connected to the existing graph.

Once again, a number of students were able to identify that a new node would be required, but were not sufficiently explicit about how the new node would be integrated into their answer from Question 6a.

Question 7

Marks	0	1	2	3	4	Average
%	2	0	3	8	87	3.8

The majority of responses were awarded full marks, and students had a strong understanding of branching.

Some students attempted to define functions within their pseudocode, with varied levels of success. While using functions may make the pseudocode slightly more concise, students are strongly encouraged to aim for clarity over conciseness when writing pseudocode.

The following is an example of a high-scoring response.

```
If weekday
      If time > 9am and time < 3pm
            School
      Else
            If homework
                  Study
            Else
                  Sleep
Else
      If homework
            Study
      Else
            Sleep
```

Question 8

Marks	0	1	2	3	4	Average
%	33	22	10	19	16	**1.7**

The n^{th} odd number is given by $2n - 1$.

Let $S_n = 1 + 3 + 5 + \cdots + (2n - 1)$

The question asked for a proof of $S_n = n^2$ for all $n \geq 1$

Base case: $S_1 = 1 = 1^2$

Inductive hypothesis: Assume $S_k = k^2$

$$
\begin{aligned}
S_{k+1} &= 1 + 3 + 5 + \cdots + (2k - 1) + (2(k + 1) - 1) \\
S_{k+1} &= S_k + 2(k + 1) - 1 \\
S_{k+1} &= S_k + 2k + 1 \\
S_{k+1} &= k^2 + 2k + 1 \\
\therefore S_{k+1} &= (k + 1)^2
\end{aligned}
$$

Since $S_1 = 1^2$ and $S_k = k^2 \Rightarrow S_{k+1} = (k + 1)^2$ it follows that $S_n = n^2$ for all $n \geq 1$

Many responses engaged well with this question, although few were able to navigate all of the required steps correctly. A significant number of responses did not explicitly check that the base case is true or glossed over the algebra in the inductive step. Other responses did not explicitly make the assumption that $S_k = k^2$. Students are reminded that when the question requires a proof, extra diligence and care is required as their primary goal is to communicate the truth of the claim to the reader.

A small number of responses used the property that $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$ in a direct proof approach, with varied levels of success. This approach was not accepted as the question required a proof via induction.

Question 9a.

Marks	0	1	Average
%	9	91	**0.9**

The majority of responses correctly identified device O as the only one that could not successfully connect with any other device.

Question 9b.

Marks	0	1	2	Average
%	13	38	49	1.4

Floyd-Warshall finds the shortest distance between all pairs of vertices. If, after running Floyd-Warshall there exists a pair that has a shortest distance equal to $-\infty$, then not all devices can communicate with each other.

There were a significant number of good responses to this question, with some students taking an approach similar to the above, while others discussed transitive closure. Lower-scoring responses merely described Floyd-Warshall without explaining how it would check if all devices can communicate with any other device.

Question 9c.

Marks	0	1	2	3	4	5	6	Average
%	8	3	3	2	13	12	60	4.9

This question was answered well, with most students being able to incorporate iteration, branching and an attempt to keep track of the number of devices. The most common error was omitting the initialisation of the variables.

The following is an example of a high-scoring response.

```
Avg_Temp(temperature_list):
      sum = 0
      total = 0
      for temperature in temperature_list:
            if temperature not equal to -255:
                  total = total + 1
                  sum = sum + temperature
      if total = 0:
            return "Error"
      else
            return sum / total
```

Question 10a.

Marks	0	1	Average
%	42	58	0.6

5! (120) routes

Question 10b.

Marks	0	1	Average
%	16	84	0.9

Joe's algorithm is an example of a brute force algorithm design.

Question 10c.

Marks	0	1	2	Average
%	12	21	67	1.6

Many students scored full marks for this question, with two common approaches. The first approach involved identifying that the described problem was analogous to the Travelling Salesman Problem (TSP), and since TSP is an NP-Hard problem, the problem in the question was also NP-Hard. Therefore, brute force is not useful for large instances of the problem.

The second approach involved identifying that the number of possible routes grows as $n!$, which quickly becomes intractable for large instances of the problem.

The following is an example of the second approach.

> *No, as the order of growth will be n!, where n=number of houses, so the number of possible routes will rapidly become intractable and his approach will not be useful.*

A small number of responses had a reasonable approach but did not answer the question, namely 'will Joe's algorithmic approach still be useful?'. Another common mistake involved correctly identifying that the given problem was analogous to the Travelling Salesman Problem, but not articulating why this was problematic as the business grows.

Finally, several students gave responses that touched on pragmatic concerns such as the size of Joe's truck or Joe's fatigue levels. No marks were awarded for these responses.

Question 11

Both parts of Question 11 were answered well, with most errors due to small mistakes such as not including 10 integers in the list or having the lists for parts a. and b. swapped.

Question 11a.

Marks	0	1	2	Average
%	7	21	72	1.7

The best-case running time is $O(n)$ and any strictly ascending list of 10 integers would generate this best case.

Question 11b.

Marks	0	1	2	Average
%	12	20	68	1.6

The worst-case running time is $O(n^2)$ and any strictly descending list of 10 integers would generate this worst case.

Question 12

Marks	0	1	2	Average
%	23	29	49	1.3

The graph has a negative cycle at C-E-D, and therefore a naive implementation of Bellman-Ford is not suitable.

Most students were able to correctly state that Bellman-Ford will not work correctly on a graph with a negative cycle; however, a significant number of responses omitted any mention of where this cycle was in the graph. Students are reminded to ensure their responses are specific where

possible. A small number of students suggested that the graph is not suitable due to negative edges.

Question 13

Marks	0	1	2	3	Average
%	12	21	47	20	1.8

DNA computing is a type of parallel computing that uses DNA molecules to perform many simultaneous operations. It can be used as an alternative method of computation to solve complex problems that are traditionally intractable using binary systems (this would include all NP-Hard problems, for instance). As an example, a DNA computer was used to solve an instance of the Hamiltonian Path Problem.

Many responses were able to engage with this question, but few answered the question in its entirety. The question directed students to 'describe DNA computing', 'explain how it can be used as an alternative method of computation' and 'provide an example'. Many students were able to achieve the mark for only two of the elements.

Question 14a.

Marks	0	1	2	3	Average
%	5	11	40	45	2.3

A wide variety of responses was given. Many students answered the question well, but did not engage with the last part of the question, which required justification for the choices the students had made.

The following is an example of a high-scoring response.

> *A graph ADT could be used with footholds as nodes and physical distance being the edge weight. This is best because (using graph traversal) it is clear which paths are shortest and which paths have closely spaced handholds which are the two requirements. The information for each climber is endurance, maximum arm/foot length/span.*

Question 14b.

Marks	0	1	2	3	4	5	6	Average
%	8	5	14	16	25	22	10	3.6

Students built upon their answer from part a. to develop an algorithm. High-scoring responses generally contained the following four elements:

- naming a suitable single-source shortest path (SSSP) algorithm (usually Dijkstra's algorithm)
- a considered elaboration of how the aforementioned SSSP algorithm would be adapted to suit the representation described in part a. of the question
- a clear description of how the nodes traversed in the graph would be stored and communicated to the climber
- an estimation of the time complexity of their proposed algorithm with a comparison to the time complexity of the brute force approach.

Most students were able to name a suitable SSSP algorithm and describe how the shortest path would be communicated to the climber. Of the mid-scoring responses, a significant number lacked elaboration of how the SSSP algorithm would be adapted to their particular representation and constraints. Additionally, students are reminded that when asked to make a comparison, they should use comparative language. For instance, simply stating that their proposed algorithm would

run in $O(n^3)$ while brute force might run in $O(n!)$ is not making a comparison, as there is no indication which option the student believes is preferred.

Question 15

Marks	0	1	2	3	Average
%	5	34	35	25	1.8

A significant number of students named PageRank, but did not fully describe how it would be implemented for this particular problem. It should be noted that a simple implementation of PageRank with profiles as websites and friendships as links was not a good option for this problem, since outgoing friendships and incoming friendships for any given profile are always identical. High-scoring responses tended to articulate a more nuanced or complex approach using PageRank. Other high-scoring responses eschewed using PageRank altogether and instead calculated a weighted average.

The following is an example of a high-scoring response.

> *An algorithm similar to Google pagerank should be implemented. Each user is assigned a relative popularity value, which increases if they have more friends and are referenced by others more, and if they are referenced by other popular users. All users can then be added to a max priority queue, with a priority of their popularity value. Users can then be targeted in the order they are dequeued.*

Question 16

Marks	0	1	2	3	4	Average
%	22	10	20	22	26	2.2

The problem Bernie is trying to solve is analogous to the optimisation version of the Travelling Salesman Problem. The brute force approach of enumerating every possible order of feeding station and then returning the route with minimum distance will guarantee an optimal solution; however, where the number of feeding stations is very large, this approach is not feasible. Alternatively, a heuristic approach such as 2-opt or simulated annealing will solve the problem quickly, but is not guaranteed to return an optimal solution.

Most students were able to identify the problem as analogous to TSP; however, it was evident that a significant number of students did not have a good grasp of how TSP could be solved using heuristic methods. In particular, there were a significant number of responses that suggested using Prim's algorithm. While finding a minimum spanning tree is a good way to derive an upper bound for the solution of a TSP problem, and acts as the first step in some approximation algorithms, Prim's in and of itself is in no way an algorithm for solving the Travelling Salesman Problem.

A number of entirely inappropriate algorithms, including shortest path algorithms, were also named by students.

Question 17

Marks	0	1	2	3	4	Average
%	40	25	19	12	4	1.2

The following general elements were evident in most high-scoring responses:

- the need to represent all possible decisions of both farmers with a decision tree
- an acknowledgement of the constraint of two years, meaning that the decision tree had a depth of two (if the question was interpreted as concurrent moves) or a depth of four (if the question was interpreted as alternating moves)

- a clear description of how the decision tree would be traversed (for instance, backtracking, tree search, Minimax, etc.)
- a clear scoring criteria with 'losing the farm' designated as $-\infty$ or similar.

This question was not answered well. Students are encouraged to ensure that they practise applying Minimax to specific contexts, and drawing the game trees when they do so.

In this particular context, Minimax would have required significant amendments to provide a good solution (the game is not zero-sum, and moves in the game are concurrent), and several high-scoring responses did not explicitly use Minimax at all. Other high-scoring responses did use Minimax, but clearly articulated necessary amendments.

Question 18a.

Marks	0	1	2	Average
%	21	7	72	1.5

```
square_area(side_length)
      return side_length × side_length
```

This question was answered well; however, many students gave long, unclear responses. These types of responses were much more likely to have been marred by an inadvertent error.

Question 18b.

Marks	0	1	2	3	4	Average
%	11	8	31	20	30	2.5

```
choose_hypotenuse(a, b, c, percentage)
      hyp = (1 - percentage) × square_area(c)
      not_hyp = square_area(a) + square_area(b)
      if hyp > not_hyp
            return True
      else
            return False
```

Most students were able to make a comparison in their pseudocode and subsequently return a Boolean value; however, the percentage that Blackbeard has to give to Monty if he chooses the hypotenuse was often missing. Additionally, poor naming of variables, trying to fit too much into one line, reusing variable names (e.g. $c = c^2$) and not explicitly stating the inputs to the algorithm resulted in errors. Students are reminded that pseudocode is used to communicate an algorithm to a human reader.

Question 18c.

Marks	0	1	2	Average
%	15	19	66	1.5

Blackbeard should always choose to take sides a and b. By Pythagoras's theorem, $c^2 = a^2 + b^2 \Rightarrow (1 - percentage) \times c^2 < a^2 + b^2$.

Question 18d.

Marks	0	1	2	Average
%	35	58	7	**0.8**

Blackbeard's decision is contingent on the value of the percentage that he must give to Monty if he takes the hypotenuse. If Blackbeard should take the hypotenuse cubed, then:

$$\begin{aligned}(1-p)c^3 &> a^3 + b^3\\ c^3 - pc^3 &> a^3 + b^3\\ pc^3 &< c^3 - a^3 - b^3\\ p &< \frac{c^3 - a^3 - b^3}{c^3}\end{aligned}$$

Therefore, Blackbeard should take the hypotenuse cubed if $p < \frac{c^3-a^3-b^3}{c^3}$ and take the other two sides cubed otherwise (noting that at $p = \frac{c^3-a^3-b^3}{c^3}$ Blackbeard should be indifferent).

Many responses correctly stated $c^3 > a^3 + b^3$ for any right-angled triangle, but ignored the percentage component, leading them to conclude that taking the hypotenuse cubed is always preferable. Other responses treated $\frac{c^3}{a^3+b^3}$ as a constant, which was incorrect.