2024 VCE Algorithmics (HESS) external assessment report

General comments

Students performed very well in Section A of the 2024 Algorithmics (HESS) examination, with all but two questions answered correctly by more than 60% of students. The two questions in Section A answered correctly by less than 50% of students were Question 18, which required students to understand the definition of NP-Hard, and Question 19, which required students to know the change-making problem. These are part of the required key knowledge described in the study design.

For Section B, there are several areas where students’ responses could be improved. Question 3, Question 12a. and Question 16 all required students to identify ADTs and describe how the ADTs could be used to store the required information. Although most students were able to identify appropriate ADTs, many did not describe clearly how the various features of the given scenario were represented within the ADTs.

Question 6b. and Question 13a. required students to analyse an algorithm and determine its time complexity. For these questions, students needed to clearly describe their calculation of the time complexity of an algorithm. Their response needed to make clear what the time complexity was for each of the components of the algorithm and how these time complexities combined.

Question 9 and Question 13 required students to write pseudocode. This skill has been assessed on every Algorithmics (HESS) examination. Students should practice writing pseudocode in preparation for the examination.

In recent years, Algorithmics (HESS) examinations have required students to write proofs about the correctness of algorithms. In preparation for the examination, students should practise writing proofs and ensure that these proofs utilise the relevant components of the algorithms.

Also, in recent years there have been questions near the end of the examination worth a substantial number of marks. Students should make sure they utilise their reading time to read the whole examination and identify questions such as Question 16, which is worth 10 marks.

Finally, many questions described various scenarios. Students should make sure their responses address the given features of the scenario, and not simply describe a generic approach.

Specific information

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

Section A – Multiple-choice questions

The correct answer is indicated by shading.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Question | Correct Answer  | %A | %B | %C | %D | Comment |
| 1 | B | 1 | **99** | 0 | 0 |  |
| 2 | C | 0 | 6 | **94** | 0 |  |
| 3 | D | 6 | 24 | 3 | **67** | This question required students to understand the depth-first search algorithm. Many students incorrectly chose the option in which U was visited after S, even though U was not a neighbour of S. |
| 4 | C | 10 | 1 | **85** | 3 |  |
| 5 | D | 9 | 1 | 14 | **75** |  |
| 6 | B | 0 | **95** | 3 | 1 |  |
| 7 | A | **92** | 1 | 2 | 4 |  |
| 8 | D | 1 | 11 | 2 | **86** |  |
| 9 | B | 4 | 79 | 10 | 6 |  |
| 10 | D | 1 | 5 | 6 | **86** |  |
| 11 | B | 9 | **80** | 6 | 3 |  |
| 12 | C | 3 | 1 | **90** | 5 |  |
| 13 | A | **71** | 21 | 5 | 2 | This question required students to understand the binary search algorithm. Many students chose the option corresponding to the third comparison value instead of the second comparison value. |
| 14 | C | 0 | 0 | **95** | 4 |  |
| 15 | A | **64** | 8 | 23 | 3 | This question required students to understand the importance of the Halting Problem as an example of an undecidable problem, and the role this problem has played in determining other problems to be undecidable. |
| 16 | B | 4 | **85** | 2 | 7 |  |
| 17 | B | 0 | **98** | 0 | 1 |  |
| 18 | D | 9 | 6 | 42 | **42** | This question required students to know the definition of NP-Hard. Many students incorrectly chose the option that NP-Hard problems must reduce to NP problems, rather than the other way around. |
| 19 | A | **38** | 8 | 32 | 21 | This question required students to know that the coin-change problem required that the number of coins be minimised, and to understand summation notation. |
| 20 | C | 3 | 14 | **81** | 1 |  |

Section B

Question 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | Average |
| % | 16 | 19 | 37 | 28 | 1.8 |

Many students did not define a directed edge or the term ‘acyclic’. High-scoring responses provided correct definitions for these properties of graphs.

Sample solution:

A directed graph is a graph with a set of nodes and a set of edges, where an edge is an ordered pair of nodes. In a directed acyclic graph, there are no paths consisting of one or more edges from any node that you can traverse which will result in you returning to the starting node.

Question 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | Average |
| % | 1 | 3 | 26 | 70 | 2.7 |

This problem was completed very well by most students. Some responses had convoluted examples of applications of stacks, or got the process for handling items in a stack incorrect. High-scoring responses tended to have standard simple examples and stated that items in a stack can be accessed on a Last-In, First-Out (LIFO) basis.

Sample solution:

A stack ADT is an ordered collection of items that can only be accessed on a Last-In, First-Out (LIFO) basis. When a new element is added to the stack, it is put on the top, and when an item is removed it can only be removed from the top. An example application is the function call stack in a computer program that records the memory address to resume computation after a function call has completed.

Question 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | 4 | 5 | Average |
| % | 1 | 2 | 16 | 44 | 23 | 14 | 3.3 |

This question required students to identify a combination of ADTs that were appropriate for representing the information required by the dry-cleaning business.

Common errors included not making it clear how the information was stored in the ADTs, not using dictionaries correctly, and storing dictionaries or arrays as jobs in a priority queue. High-scoring responses made it clear what information was stored in each ADT, justified the storage provided for the information, and made reference to the ease of access of the data using ADT-specific terminology.

Sample solution:

What matters to the business is being able to easily find information about each job and the order of jobs. To address the first requirement, information about jobs will be stored in a dictionary whose key is the job identification number, and the value is a list containing:

* the type: ‘same-day’, ‘standard’, or ‘deep’
* phone number
* the date and time of pick-up.

To address the second requirement of being able to determine the order in which jobs are to be processed, a priority queue would be used. Each job ID would be added to the queue and prioritised based on the job type, with same-day priority 1 and standard/deep priority 2. When getting the next job, the top item of the priority queue would be served and its job information retrieved from the list.

Question 4a.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 1 | 25 | 74 | 1.7 |

This question was completed very well by most students. A common error was to not include weights on the edges that represented the length of the cable connecting switches, power points and the central power source.

Sample solution:

Nodes can represent switches, power points and the central power source, and edges connecting nodes would be the cables. The edges could be weighted by weights giving the length of the cable connecting the corresponding electrical network components.

Question 4b.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | Average |
| % | 19 | 5 | 15 | 62 | 2.2 |

This question required students to consider using Prim’s algorithm, Dijkstra’s algorithm or the Bellman-Ford algorithm to solve a problem, and to justify what the most suitable option is.

This question was answered very well by most students. Some responses did not provide justifications for why Dijkstra and Bellman-Ford are not the most suitable approach for solving this problem. High-scoring responses compared all the different approaches and described how the problem was analogous to finding the MST of a graph.

Sample solution:

Finding the minimal length of cabling required to connect everything to the central power source is analogous to the problem of finding the minimal spanning tree (MST) of a graph. Prim’s algorithm is specifically designed to find this, so the electrician should use Prim’s. Dijkstra’s and the Bellman-Ford algorithm are both used to find the solution to the single source shortest path problem, which would not be useful in this instance.

Question 5a.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 26 | 13 | 61 | 1.4 |

This question required students to demonstrate that they understood how to perform feed forward propagation in a neural network. This question was completed very well by most students. High-scoring responses clearly described their method of calculating the result.

Sample solution:

Substituting values in:

Question 5b.

|  |  |  |  |
| --- | --- | --- | --- |
| Marks | 0 | 1 | Average |
| % | 49 | 51 | 0.5 |

This question required students to describe the output when a given transfer function was used. High-scoring responses provided the output value.

Sample solution:

Since the input to is 315, then the function would mean the output is set to 1 as this value is greater than 50.

Question 6a.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 4 | 44 | 52 | 1.5 |

This question required students to draw an undirected graph representing the given grid. High-scoring responses included all the edges with correct weights on the edges.

Sample solution:



Question 6b.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | 4 | Average |
| % | 25 | 42 | 7 | 13 | 13 | 1.5 |

This question required students to analyse an algorithm to determine its Big-O time complexity.

Some responses did not clearly identify the time complexity of the different components of the algorithm. Another common error was to incorrectly assume that the number of regions, n, was the same as the number of rows, r, times the number of columns, c.

Sample solution:

Lines 4–7 take time proportional to r∙c.

Lines 8–9 take time proportional to .

Lines 10–15 take time proportional to r∙c.

Lines 16–18 take time proportional to the number of edges, .

This gives .

Question 7a.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 10 | 52 | 38 | 1.3 |

This question required students to identify two potential risks that could arise from the use of AI. High-scoring responses identified risks that were related to the use of AI and not general risks related to online shopping.

Some acceptable responses included any two of the following:

* privacy of the data used to train the AI; potential for it to be used by other parties
* bias in the data used to train the AI; chance that it reflects certain groups liking certain things and hence reinforces historical biases
* transparency of the results provided by the AI; the difficulty in explaining why the program comes up with a particular solution.

Question 7b.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 47 | 25 | 28 | 0.8 |

This question required students to describe a problem that would result from training a large neural network with a relatively small amount of data. Some responses incorrectly identified this problem as ‘underfitting’. High-scoring responses correctly identified that this problem is known as ‘overfitting’.

Sample solution:

Training a neural network this large on this small amount of data would result in overfitting, as there are too few data points to fit a complex model. The model would be excellent at predicting all the training cases but would perform poorly with new data.

Question 8a.

|  |  |  |  |
| --- | --- | --- | --- |
| Marks | 0 | 1 | Average |
| % | 14 | 86 | 0.9 |

* knapsack problem

This question was completed very well, with most students identifying that the problem was analogous to the knapsack problem.

Question 8b.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | Average |
| % | 30 | 31 | 22 | 18 | 1.3 |

This question required students to describe a heuristic method to optimise the store’s shelves to be most profitable.

Some responses took a generic approach and did not address the given scenario, or did not provide a description of a heuristic. High-scoring responses clearly described a heuristic and provided a comprehensive description of how it could be used to optimise the store’s profitability.

Sample solution:

Determine the profit-per-size ratio for each display and sort the displays from highest profit/size to lowest. Select a random selection of displays up to the maximum total size. Randomly select displays to swap in and out, preferring swaps that improve the total profit and slowly reducing the likelihood of accepting swaps that decrease the profit. Return the best found solution.

Question 9a.

As a result of psychometric analysis and review, all students were awarded full marks for this question.

Question 9b.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 48 | 38 | 13 | 0.7 |

This question required students to discuss whether the brute-force approach could be feasibly applied to fences with 50 pickets and fences with 50 000 pickets.

A common error was to not consider fences with 50 pickets. High-scoring responses considered both possibilities of fence sizes and related their answers back to Question 9a.

Sample solution:

The brute-force approach would be able to quickly solve smaller fences and so would be feasible for fences with 50 pickets. For large fences, it would take an infeasibly large amount of time. With four nested loops, the algorithm grows as n4. At 50 000 pickets, this is ~1018 times through the body of the loop, and this could reasonably be expected to take many days to compute.

Question 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | Average |
| % | 18 | 19 | 31 | 32 | 1.8 |

This question required students to compare brute-force and backtracking search algorithm design strategies. High-scoring responses described how the two design strategies are similar, how they differ, and how backtracking improves brute-force search.

Sample solution:

Both design patterns systematically generate candidate solutions to search for the problem solution. They differ because brute-force search generates full solutions and checks to see whether they are valid, while backtracking generates partial solutions which are progressively extended. By discarding partial solutions that could never lead to a solution before extending them, backtracking is able to not search through the full solution space.

Question 11

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | Average |
| % | 39 | 45 | 7 | 9 | 0.9 |

This question required students to provide a proof by contradiction of the correctness of the algorithm equalDegreeNodes.

Most students were able to identify the initial assumption of the proof, demonstrating that they understood what is meant by a proof by contradiction. High-scoring responses were able to provide a proof analogous to the pigeonhole principle, and related the various steps of the proof to the components of the algorithm.

Sample solution:

Assume that equalDegreeNodes does not return a pair of nodes.

Therefore, the comparison on line 5 is false for all choices of d.

Therefore, there is only one node in G with degree |V|−1, |V|−2, |V|−3, …,1.

The loop on line 2 iterates through |V|−1 degrees, therefore the |V| nodes in G cannot all have a unique degree.

This contradicts the requirement that the comparison on line 5 is false for all choices of d and therefore the algorithm always returns a pair of nodes.

Question 12a.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 9 | 30 | 61 | 1.5 |

This question was completed well by most students. It required them to describe how an ADT could represent a single Oriento grid. A common error was to not describe how the grid features are represented within the ADT.

Sample solution:

Oriento grids could be stored using an undirected graph with nine nodes, with each node representing one of the dials. Each node would have a direction attribute. Nodes representing adjacent dials would have an edge connecting them.

Question 12b.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 42 | 28 | 31 | 0.9 |

This question required students to describe an algorithm to solve the game by turning the fewest number of dials.

Students needed to use a standard approach such as breadth-first search, Dijkstra’s algorithm, the Bellman-Ford algorithm or the Floyd-Warshall algorithm. High-scoring responses used a standard approach and described what edge weights were used.

Sample solution:

Perform a breadth-first search from node s in G and stop when any node representing an arrangement where all arrows point in the same direction is found. Return the first solution that is found using the breadth-first search.

Question 12c.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | Average |
| % | 50 | 18 | 17 | 16 | 1.0 |

This question required students to describe an algorithm for solving the game by performing the least amount of total rotation.

Some responses did not identify that the turn values could be used as weights, and that the 90 degrees and –90 degrees correspond to ¼ of a rotation, and 180 degrees corresponds to ½ of a rotation. High-scoring responses recognised how this problem was analogous to finding the shortest distance in a weighted graph, and described how using an appropriate algorithm, like Dijkstra’s algorithm, could be used to solve the problem.

Sample solution:

First, set weight of edges corresponding to turns of 90 degrees or −90 degrees to ¼, and set the weight of edges corresponding to turns of 180 to ½. Then apply Dijkstra’s algorithm on the graph G starting from node s, until all four solution nodes have been visited. Then return the solution node with the lowest weight solution.

Question 12d.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 11 | 25 | 64 | 1.5 |

This question was completed very well by most students. High-scoring responses identified a suitable AI concept and described that concept in relation to Zoni’s robot.

Sample solution:

When compared to the concept of strong AI, Zoni’s robot does not possess artificial intelligence, as it does not understand the problem like a human. It simply solves the problem by pattern matching and following rules, and hence imitates understanding. Therefore, it has weak AI but not strong AI.

Question 13a.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | 4 | Average |
| % | 21 | 19 | 5 | 11 | 44 | 2.4 |

This question required students to analyse an algorithm to determine its Big-O time complexity. High-scoring responses clearly identified the complexity of the different components of the algorithm.

Sample solution:

The time complexity of CalculateNumberOfCubes is O(column). As column depends on level, it follows that the time complexity of CalculateNumberOfCubes is O(s) and that the nested loops take time proportional to s2. Hence using the time complexity of CalculateNumberOfCubes, O(s), and the time complexity of DrawCubes, O(1), we obtain a time complexity of O(s3).

Question 13b.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | 4 | Average |
| % | 70 | 18 | 6 | 4 | 2 | 0.5 |

This question required students to describe a dynamic programming algorithm for drawing a diamond of cubes. Some responses did not demonstrate an understanding of how to use dynamic programming to draw a diamond. High-scoring responses demonstrated how each level depended upon the previous level.

Sample solution:

Algorithm BetterCubeDiamond(s):

 columns [0]

 For level in {0,...,(s – 1)} Do

 new\_columns [1]

 For i in {0,...,(level-1)} Do

 append (columns[i] + columns[i+1]) to new\_columns

Endfor

 append 0 to new\_columns

 columns new\_columns

 DrawLevel(level, columns)

 DrawLevel(2s-level-2, columns)

Endfor

Question 14a.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 51 | 43 | 7 | 0.6 |

This question required students to state whether they agreed with the given statement, and to provide reasons for their decision. High-scoring responses provided a clear description of classes P and NP.

Sample solution:

Yes, I agree with this statement.

P is the class of problems that can be solved in polynomial time by a deterministic Turing machine.

NP is the class of problems that can be verified in polynomial time. Some of these problems are considered intractable as they do not have any known algorithms to solve them in polynomial time.

So, if P = NP, it means existing problems in NP that are thought to be intractable can now be solved in (reasonable) polynomial time.

Question 14b.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | Average |
| % | 37 | 27 | 25 | 12 | 1.2 |

This question required students to state whether they agreed with the given statement, and to provide reasons for their decision. Many students had significant misunderstandings of the definitions of NP, NP‑Complete and NP-Hard. High-scoring responses provided a clear description of classes NP-Hard and NP-Complete.

Sample solution:

They are not the same. It is true all NP-Complete problems are also NP-Hard, as by definition NP‑Complete problems are in NP and in NP-Hard. But NP-Hard problems do not necessarily have to be in NP-Complete class. NP-Hard problems only need to be reducible in polynomial time from any other NP problem and form a superset of problems. For example, the optimisation version of the TSP is in NP-Hard, but it is not in NP, as the optimal solution cannot be in general verified in polynomial time.

Question 15

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | Average |
| % | 62 | 17 | 21 | 0.6 |

This question required students to explain the time complexity results that Lila observed when testing her algorithm. High-scoring responses used the Master Theorem and described an interpretation of the recurrence relation that could represent the time complexity of Lila’s algorithm.

Sample solution:

One possible explanation is that Lila has an algorithm with a recurrence relationship of the form:

Since the running time is quadratic, it follows from the Master Theorem that c = 2. Hence it follows that the process of reducing the search space and/or using the solution of the reduced search space has quadratic time complexity.

Question 16

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Marks | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Average |
| % | 6 | 6 | 8 | 9 | 19 | 15 | 15 | 13 | 6 | 3 | 0.3 | 4.6 |

This question required students to describe how to represent this problem, to describe an algorithmic approach for determining the fastest path for a character to travel, and to explain why their algorithmic choice was best.

Some responses did not clearly describe how all the features in the game could be represented. High-scoring responses described how all the features could be represented, how the A\* algorithm could be used, provided an appropriate heuristic that the A\* algorithm could use, and explained why their approach was best.

Sample solution:

Represent the map as a directed graph, with a node for each non-mountain square and edges connecting adjacent squares. Set the distance of each edge as the movement cost for moving into the square the edge points to (1 plains, 2 forest). Do not consider rain or other players, as this will change during the game.

Pre-compute the shortest path from all pairs of nodes using the Floyd-Warshall algorithm.

When a move command is given, use the A\* algorithm to determine the shortest path from the current location to the destination. When applying A\*, use the pre-computed shortest distances as the h(n) values for the algorithm.

This would be the best choice of algorithm because A\* explores away from the starting position in an efficient way towards the destination. With this choice of heuristic, h(n) is never greater than the true cost of travel and hence this would guarantee an optimal path is found. Using the pre-computed shortest distances, the algorithm would only search around the optimal path in locations where there is rain. Because the map does not change, pre-computing the shortest paths is efficient because it can be done once at the start of the game.