JOHN MONASH SCIENCE SCHOOL

VICTORIAN CERTIFICATE OF EDUCATION

2018

ALGORITHMICS (HESS)

Practice Exam 2

(STUDENT DESIGNED VERSION)

2018

SOLUTIONS

SECTION A

Question Number	Answer
1	B
2	D
3	C
4	A
5	C
6	A
7	C
8	D
9	B
10	C
11	C
12	B
13	B
14	C
15	B
16	A
17	NA
18	NA
19	NA
20	NA

SECTION B

Question 1 (2 marks)

The pet shop in Clayton that receives shipments of military-grade puppies has identified that some puppies do not like one another whilst others are best friends. Describe an approach that could be used to identify which puppies can be placed together and which cannot. As part of your answer, describe what ADT you would use to model the problem.

Recognition of the problem being modelled as a graph	**A1**
Relation to graph colouring or any other acceptable approach that will lead to a correct solution.	**A2**

Question 2 (13 marks)

Riley Purcell likes to wear crazy socks. So much so that he has decided to remove all John Monash Science School sanctioned socks from his drawers, and furthermore, sort his socks into colours of alphabetical order (eg: blue socks would be sorted before red socks and yellow socks would be sorted after blue and red socks. Thomas Reyment realised that Riley was too incompetent to sort his own sock drawer and being the helpful man he is, wrote the following algorithm to help a brother out:

```
Algorithm: sort_Rileys_socks(socks):
     Input:     A one-dimensional array, socks, where each
                element represents a socks colour.
     Output:    A one-dimensional array, socks, with socks sorted
                into colour order.

     for i = 0 to i = number of socks - 1:
          if socks[i] is JMSS Sanctioned:
               remove socks[i] and burn it
          min = i
          for j = 1 to j = number of socks:
               if socks[j] is alphabetically before socks[min]
                    min = j
          swap positions of socks[i] with socks[min]
     return True
```

a. Find the time complexity of the algorithm sort_Rileys_socks. 2 marks

A. T(n)=i=0n-1j=i+1n1 (1)
=12(n2+n)
Therefore O(n2) (1)
1 mark for correct summation notation. 1 mark for correct evaluation of O(n^2).

b. Niamh, curious to how Riley manages his extensive sock collection, asks to see Thomas' sock sorting algorithm. To Thomas' dismay, she takes offence to his use of nested for-loops.
Niamh claims *"These nested for-loops make this algorithm super slow! If you used multiple loops following each other, it would run significantly faster"*

Is Niamh correct? Explain your answer, comparing the complexities of the techniques. 3 marks

A. 1 mark for sequential for loops being faster. 1 mark for using sequential for loops means the time complexity of each for loop is added. 1 mark for using nested for loops means the time complexity of each for loop is multiplied.

c. Diana Lautenbach takes a look at Thomas' algorithm, because she is in Algorithmics and doesn't really have anything better to do. After reading the algorithm, Thomas and Diana proceed to argue over the classification of its time complexity:

Thomas claims *"This problem has P class time complexity."*

Diana states *"I don't really know what's going on but my dad told me the algorithm is NP - Complete as a solution can be verified in polynomial time."*

Who is correct? Explain your answer. 3 marks

1 mark for Thomas is correct. 1 mark for explaining that a P class problem can be solved in polynomial time complexity. 1 mark for the problem is not NP complete as it can be solved in polynomial time complexity.

d. Even though no one asks for Diana to give her input, as a result of her superiority complex she has developed from her algorithmics class, she proposes that the sock draw can be solved in $O(n\log(n))$ time. Write the name of the sorting algorithm Diana is referring to below. 1 mark

Mergesort.

e. Using your answer from part D, write out the pseudocode for the sorting algorithm below. 4 marks

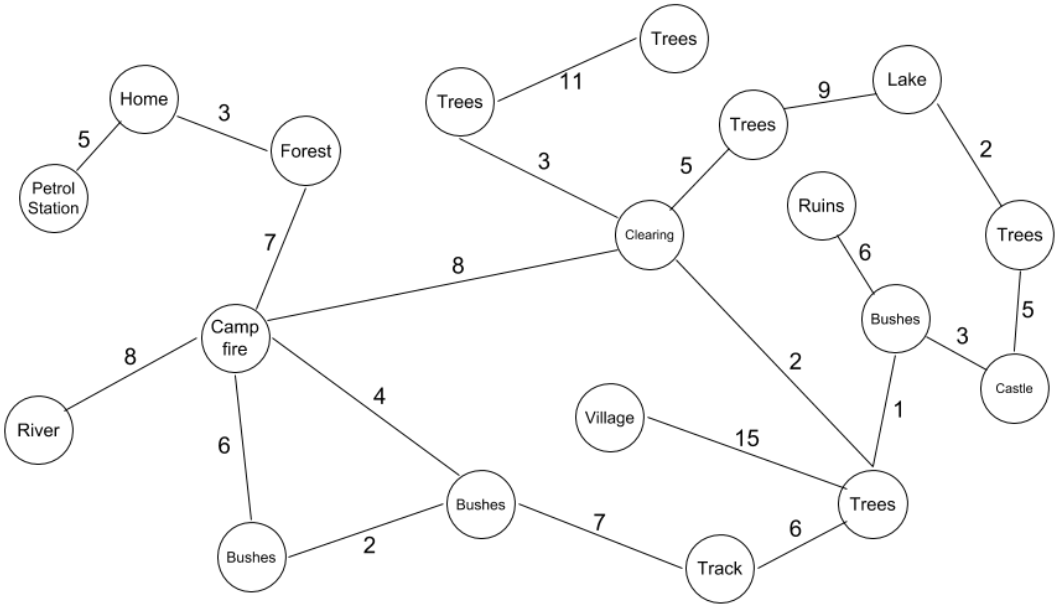
```
def Mergesort(socks[0..n-1]):
    ''' Intput: An array containing n of Riley's socks'''
    if n > 1:
        let midpoint = floor(n/2)
        copy socks[0..mid-1] to A[0..mid-1]
        copy socks[mid..n-1] to B[0..mid-1]
        Mergesort(A[0..mid-1])
        Mergesort(B[0..mid-1])
        Merge(A, B, socks)

def Merge(A[B..p-1], B[0..q-1], socks[0..p+q-1]):
    ''' Input: Sorted arrays of socks A and B of socks
        Output: Sorted array socks containing the elements of A and B'''
    i = 0, j = 0, k = 0
    while i < p and j < q do:
        if A[i] <= B[j]:
            socks[k] = A[i]; i = i + 1
        else:
            socks[k] = B[j]; j = j + 1
        k = k + 1
    if i = p:
        copy B[j..q-1] to socks[k..p+q-1]
    else:
        copy A[i..q-1] to socks[k..p+q-1]
```

Question 3 (7 marks)

Johnbob the Forest Legend is the most experienced Forest Ranger known to mankind. He's lived next to the forest all his life. No wife, no kids, just him and the forest. He knows the ins and outs of the forest like nobody, not even the forest creatures that call it home. Some people say "that man has the map of the forest in his head!" Some even go as far as saying "he is... THE FOREST..." No one actually knows how he came about; his family and origins all unknown to this day. A legend surfaced; man born from the forest, with only one purpose, to serve and protect. Man or not, he is THE FOREST.

After a busy day's work, Johnbob heads home. Upon entering the household, he realises that Bobjohn, his pet squirrel is not with him. After the anagnorisis, Johnbob figures that his squirrel may be in the deepest part of the forest. He writes down the following map from his memory. Before heading out in search for Bobjohn, Johnbob decides to get wasted to relieve the stress. Intoxicated, Johnbob needs your algorithmic skills to help find his lost pet Bobjohn. He thinks of the map below.



a. Johnbob has enough fuel to travel x km. He first needs to calculate the shortest distance to each node from his home. Which algorithm would he use and why? 2 marks

NO ANSWERS PROVIDED

b. What is the width of this graph? 1 mark

NO ANSWERS PROVIDED

c. Should Johnbob refuel at the petrol station before entering the forest and why/why not? 2 marks

NO ANSWERS PROVIDED

After finding Bobjohn at the deepest part of the forest, Johnbob realises that he forgot to account for enough fuel to get back home. Thus, he sleeps in the forest and recalls the memories of his childhood abandonment.

d. Describe the design pattern that the algorithm in **part a**. uses, stating it's advantages and disadvantages. 2 marks

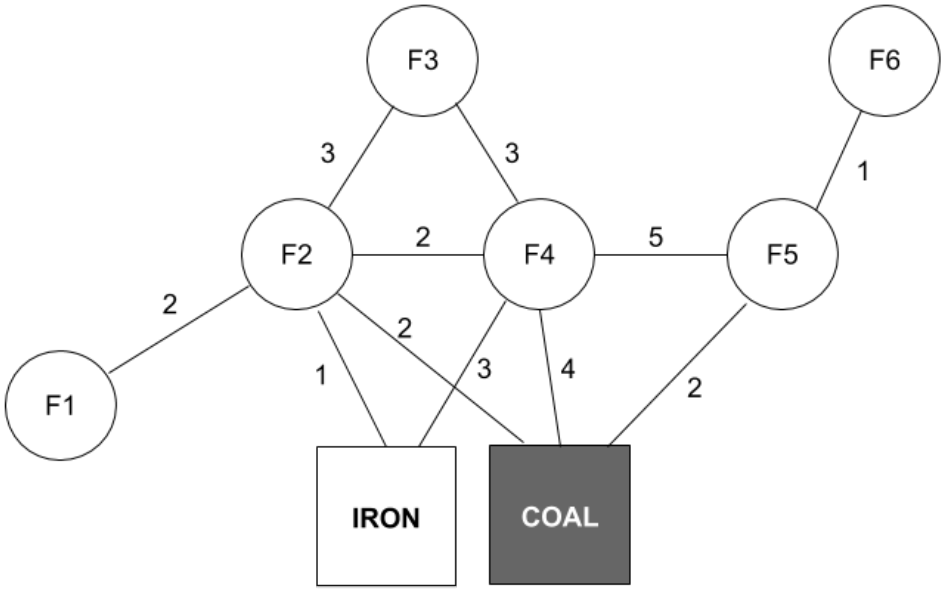
NO ANSWERS PROVIDED

SECTION B - continued

Question 4 (3 marks)

Ben Sutherland was playing a game of Factorio instead of doing his classwork. Ryan looked at Ben's game and decided that all his ratios were off and that he needed to rewire some of his conveyor belts in order to make a more efficient factory. Each furnace needs to be connected to one coal deposit and one iron deposit in order for Ben to make his favourite item; the iron gear wheel.

The diagram below shows an example of Ben's game.



Describe an algorithm that minimises the amount of conveyor belts required to connect each node to the network of belts and shows the answer on the graph above.

Use Prim's algorithm to construct the Minimum Spanning Spanning tree to minimise the number of conveyor belts needed.

Question 5 (3 marks)

Jeff is hosting a party. Unfortunately, many of his friends are brutal enemies. If he invites them both to the same party, they will try and fight to the death, which Jeff doesn't want to happen. With divine insight, Jeff figures out that if he hosts two separate parties, there would be no conflicts. He hires a secret investigator to obtain a list of guests who are incompatible with each other. Jeff decides that he wants to invite as many people as possible to the first party. Write an algorithm that figures out the maximum possible amount of people that can be invited to the first party, given a list of incompatible pairs.

Answer: Starting with a random person, use greedy graph colouring to generate a graph (redundant step). Now start with a random person on the graph, and use a greedy algorithm to colour the graph in 2 colours. Take the max number. Now do the same for every person who hasn't been visited. Take the sum of the max numbers.

Question 6 (3 marks)

Consider the following algorithm.

```
Algorithm: Algo1(n)
    if n = 1:
        return -1
    Else:
        If n is odd:
            return (-n + Algo1(n-1)
        Else:
            return (n + Algo1(n-1)
```

a. Convert **Algo1** into a tail recursive algorithm. 2 marks

Answer:

```
Define Algo1(n, running_total):
    If n = 1:
        Return (running_total - 1)
    Else:
        If n is odd:
            Algo1(n-1, running_total - n)
        Else:
            Algo1(n-1, running_total + n)
```

b. What advantages does the tail recursive algorithm have over the recursive version? 1 mark

Answer:

The tail recursive algorithm uses less memory as it only has to store the running total and not the results of the previous recursions of the algorithm, this is especially important for deep recursion depths where there are many previous recursions to store in memory.

Question 7 (5 marks)

Sai wanted to create a program that would allow him to hack into the Eduroam network, but he doesn't know where to start. He desires a program with various utilities that more often than not use the same base utilities.

a. Which approach would be more efficient; a top down approach or a bottom up approach? Justify your response by comparing the two techniques. 3 marks

Answer:
Because of the constant utilisation of the same base utilities, the code can be modularized to allow for reuse of code blocks. By breaking down the problem into sub utilities and various functions, multiple members of a team can work on different parts of the problems. In comparison to a bottom up approach where code is built from scratch and tested as time goes on, it can become difficult to keep track of the original intention and become tangled in code which may not be reusable most of the time.

b. Hugh was angry at the extent we went to create an interesting backstory to our questions and he seeks an algorithm to cut down on the words in the questions. What heuristics can he use in his algorithm? 2 marks

The problem is all about identifying ways to approach the limits of computability. Heuristics such as mentioning a certain word, like highlighting where the problem starts or where things get personal (i.e mention of Ben).

Question 8 (10 marks)

a. Compare the techniques used in the Bellman Ford algorithm and Floyd Warshall algorithm to calculate the distance from one node to every other and all nodes to all other nodes respectively. 3 marks

Bellman Ford iterates through each edge weight, iterating through each edge to calculate the shortest path from that one node to that other specific node. On the other hand, Floyd Warshall uses the nth node as an intermediate node, checking if the distance between the intermediate node and the immediate path is shorter, electing to choose the shorter path.

b. Jeff wants to create a path finding algorithm for Euro Truck Simulator 2, but he was told that Bellman Ford and Floyd Warshall are way to slow. He wants to find the guaranteed shortest route from his starting point to another location in Europe. He is also too lazy to learn A*. What ADT is most suited in modelling the cities and roads? What algorithm should he use? 2 marks

Answer: Graph ADT, Dijkstra's algorithm, but for the pathfinding part use the UCS variant (as seen on wikipedia page for Dijkstra, can use priority queue to simplify).

c. Jeff decides to compare the performance of Floyd Warshall and Dijkstra's algorithm for the problem above. How can he modify Floyd Warshall's algorithm to not only calculate the shortest path from each node to every other node, but also to store the shortest path? To avoid blowing up his 1GB of RAM, he can only use O(V) memory to store the paths 3 marks

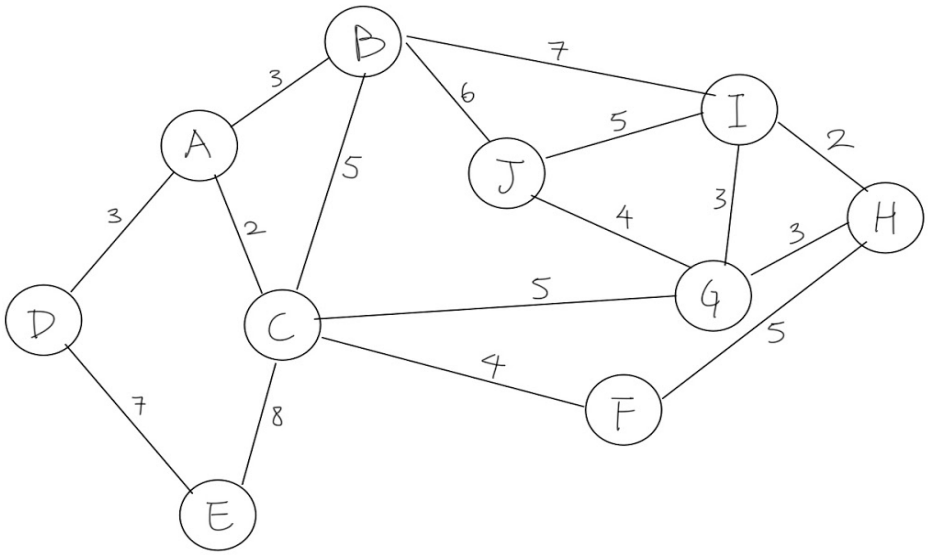
Answer: Instead of storing the previous edge/node for each node, just store each previous node as nextNode[i][k]. Eg: 1 → 2 → 3 → 4. nextNode[1][4] = 2, nextNode [2][4] = 3, nextNode[3][4] = 4. This just arises due to transitive closure(?) part of Floyd Warshall algorithm. (Pseudocode under "Path reconstruction" part in wiki for Floyd).

d. How would Jeff be able to detect whether a graph is a tree or not? 2 marks

Answer: Use DFS/BFS to traverse every node. If an adjacent node that isn't the parent has been visited before, then the graph contains a cycle and isn't a tree.

Question 9 (4 marks)

Consider the graph shown below:



a. What is the function of the Bellman-Ford Algorithm? 1 mark

To find the shortest path to all nodes from one node (1 mark)

b. Starting from node D, after the third iteration (assuming constant update after each loop) of the Bellman-Ford Algorithm identify which of the nodes' distance have been confirmed. Explain why this is the case. 4 marks

A	B	C	D	E	F	G	H	I	J

No circles - (1 mark)
Bellman Ford does not know the shortest distance to each node until the algorithm runs through the whole entire graph (1 mark) and no updates have occurred (1 mark)
This is because the algorithm does not know if there are negative cycles present in the graph (hence D is not confirmed too) (1 mark)

Question 10 (2 marks)

As part of a corny joke, Jeff is trying to make his way through a maze of maize. Describe an algorithm design pattern that Jeff could use to solve his problem.

Sample answer: Jeff could use backtracking. By representing the ma(i)ze as a graph with each node representing a choice, or fork in the maze, and the edges being paths between these sections. He can then travel as far as possible along any given path, then backtrack when he meets a dead end, or visited section, until he finds the end.

Question 11 (2 marks)

Describe why the Knapsack Problem is considered NP-Hard and provide an NP variant of the problem.

- Explains the knapsack problem is an optimisation problem
- Highlights a decision version of the problem

Question 12 (6 marks)

A number, c, is considered a cool number if $c + 1$ can be expressed as the product of two different cool numbers. 2 and 3 are the lowest cool numbers despite note displaying this property.
Using a dynamic programming approach, design an algorithm that determines if a positive integer, n, is a cool number.

Answer marks:

1. Algorithm displays elements of dynamic programming, storing previously generated Cool Numbers using appropriate data types.
2. Algorithm correctly checks if a number is Cool using stored Cool Numbers
3. Base cases of 2 and 3 are considered, and can be exclusively used to construct further Cool Numbers.
4. Correct use of recursion shown
5. Algorithm answers the prompt in most cases
6. Algorithm works in all edge cases (e.g. Returning False if n+1 is the square of another cool number, returning True on n=2 and n=3, False on n=1) Algorithm does not have to consider cases where n is not a positive integer.

Example Response:

```
coolList = empty list
Define coolTest(n):
        Return True if n = 2 or 3, or is in coolList
        Return False if n = 1
        For every integer (i) greater than one and less than the square root of n+1:
                If n+1 mod i = 0:
                        j = (n+1)/i
                        If i is in coolList, or coolTest(i) is True:
                                If j is in coolList, or coolTest(j) is True:
                                        Add n to coolList and then return True
        Return False once all integers stated above have been exhausted.
```

Question 13 (4 marks)

John Searle's Chinese Room Argument examines the nature of artificial intelligence.

a. Present an argument **against** Searle's position. 2 marks

Substantively refutes Searle's argument **- 1 mark**
Discusses significance in context of Searle's argument - **1 mark**

b. Present a **refutation** of the argument presented in **part a.** 2 marks

Substantively addresses argument given in part a) - **1 mark**
Discusses significance in context of Searle's argument - **1 mark**

Question 14 (4 marks)

List and discuss the significance of **two** limitations of Neural Networks.

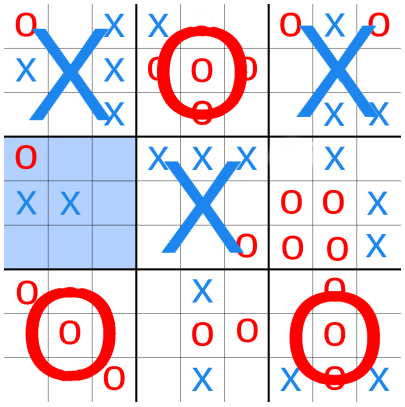
A: **1 mark per valid example, 1 mark for valid discussion of implications**
Examples:

- Unchanged time complexity
- Inexact solutions
- Suitable only for specific varieties of problems
- Extra time and computational investment for training
- Not necessarily computationally optimal
- Hyperparameter tuning (and other characteristics) poorly understood even in the state of the art

Other significant and reasonable items are acceptable.

Question 15 (6 marks)

Mr Bohni and Mr Huynh instead of deciding to teach today; sick of the study design, decide to organise an ultimate tic-tac-toe tournament. After some dodgy decisions, blatant bribes, and falsified forfeits Mr Bohni and Mr Huynh both make it to the final, Mr Bohni is playing as crosses.



Utilising an appropriate algorithm use a tree to determine what move Mr Bohni should make next and what is the best outcome for him, given the above state of play (Mr Bohni has to go in the blue highlighted square)?

- Draw appropriate tree diagram with relevant states of play (2 mark)
- Apply minimax on the tree diagram correctly (1 mark)
- Show that the best outcome Mr bohni can get is a draw (1 mark)
- Show that he should complete the line in box as his move (1 mark)

Question 16 (6 marks)

Callum wants to sort his collection of Star Trek bootlegs from A-Z. He is thinking of trying either Mergesort or Quicksort.

a. Describe the differences between Mergesort and Quicksort approaches. As part of your answer compare the algorithmic design patterns used by each and their relative time complexities. 3 marks

Key points in answer: Mergesort requires space to store the split up data. Quicksort does not. **(1)** Worst case time complexity for Quicksort is $O(n^2)$ while for Mergesort is O(n(n)) **(1)**

Callum instead decides to come up with his own, heuristic based algorithm to solve his problem.

b. Describe a possible limitation of using a heuristic based algorithm and a way that this limitation might be compensated for. 3 marks

Sample answer: Heuristic based algorithms will always only find the local maximum of an algorithm, rather than the global maximum. Adding some leeway when making decisions (like the 'temperature' variable in the case of simulated annealing), reduces this problem by accepting some worse solutions in order to approximate the global maximum.

END OF QUESTION AND ANSWER BOOK