

The University of Melbourne
Department of Computing and Information Systems

COMP20007
Design of Algorithms
June Assessment, 2014

Student Number:

ANSWER

Identical Examination papers: None.

Exam Duration: Three hours.

Reading Time: Fifteen minutes.

Open/Closed Book: Closed Book.

Length: This paper has 10 pages including this cover page.

Total Marks: 60

Authorized Materials: None.

Instructions to Invigilators: Students will write all of their answers on this examination paper. Students may not remove any part of the examination paper from the examination room.

Instructions to Students: This paper counts for 60% of your final grade. All questions should be answered in the spaces provided on the examination paper. You may make rough notes, and prepare draft answers, on the reverse of any page, and then copy them neatly into the boxes provided. You are not required to write comments in any of your code fragments or functions.

Throughout you should assume a RAM model of computation where input items fit in a word of memory, and basic operations such as $+$ $-$ \times $/$ and memory access are all constant time.

Calculators: Calculators are not permitted.

Library: This paper may not be held by the Baillieu Library.

Question 1 (10 marks).

- (a) (1 mark) What is the minimum number of bits required for codewords in a uniquely decipherable Binary code that is used to represent integers in the range 0 to 1023?

ANSWER

1 marks 10

- (b) (6 marks) As in Assignment 2, assuming a raster-scan order of grid square visits on a 3x3 grid, and a Unary code of the actual values, draw the actual grid mapped by the robot for the following bitfile?

1 001 00001 001 01 1 01 1 001

ANSWER

3/6 if begin decoding at 1, not 0

0	2	4
2	1	0
1	0	2

One mark for each unique digit decoded 0,1,2,4 and 2 marks for correct positions.

- (c) (2 marks) In what situation would you use a Unary code to encode symbols? Justify your answer.

1 mark "small numbers"

ANSWER

1 mark When the probability distribution of symbols is skewed,

1 mark in particular, $p_i = 2^{-i}$ for $i = 1, 2, \dots, n$.

- (d) (1 mark) How many bits would be required to code the grid in part (b) using a vbyte code on the actual values?

ANSWER

9*8 = 72 bits.

Question 2 (11 marks).

- (a) (2 marks) For a general, directed graph $G(V, E)$, define what is meant by a “source vertex”, and then define what is meant by a “sink vertex”.

ANSWER

1 mark A source vertex has no incoming edges.

1 mark A sink vertex has no outgoing edges.

- (b) (3 marks) What is the minimum number of components in an undirected graph $G(V, E)$ with $|V| \geq 2$ if G has one source vertex? Justify your answer.

ANSWER

1 mark 2

1 mark a source in an undirected graph must have no incident edges. Hence there is at least one isolated vertex. (Some reasoning about edges)

1 mark which would be in a component, and the other vertices could all be in the one component. (Some definition of component)

0 if use directed.

0 if say 2 with wrong justification

2 marks if they carry over definition from (a) where source must have outgoing edge

0 if say undirected must be connected

0 marks if they do not use the definition from (a)

- (c) (6 marks) Write a C code function that takes in a directed graph G , and returns a count of all of the cycles in G that contain two edges. Note that this will double count each cycle: for example, if edges (u, v) and (v, u) exist, there will be a cycle from u through v , and a cycle from v through u . You should assume that G is stored as an adjacency matrix $G[N][N]$, where N is a parameter to your function, and $G[u][v] \neq 0$ if there is an edge from u to v .

ANSWER

1 mark	<code>int</code>	or some suitable return type
1 mark	<code>count_cycles(int **G)</code>	some suitable 2D type
1 mark	<code>int count = 0;</code>	some counter initialisation
1 mark	<code>for (i = 0 ; i < N ; i++)</code> <code>for (j = 0 ; j < N ; j++)</code>	a loop over all edges
1 mark	<code>if (G[i][j] && G[j][i])</code> <code>count++;</code>	some check and increment
1 mark	<code>return count;</code>	

Subtract 0.5 for each syntax error???

Question 3 (11 marks).

- (a) (2 marks) Give a specific example of an input to the Knapsack Problem for which an algorithm that was greedy on weight would not give the optimal answer.

ANSWER

1 mark for defining capacity, and some items with weight and profit

1 mark for having light weight have light profit

- (b) (2 mark) Give a specific example of an input to the Knapsack Problem for which an algorithm that was greedy on profit would not give the optimal answer.

ANSWER

1 mark for defining capacity, and some items with weight and profit

1 mark for having heavy profit with heavy weight

- (c) (7 marks) Consider the following recursive algorithm for solving the Knapsack Problem. Find $L(n, n \times \max\{v_i\})$ where $L(i, p)$ is the lowest weight combination of items selected from items 1..i whose total profit is p . If no such combination of items exists, $L(i, p)$ is ∞ . Note item i has integer weight w_i and integer value v_i .

$$L(i, p) = \begin{cases} \infty & \text{if } i = 0 \\ L(i-1, p) & \text{if } v_i > p \\ \min\{L(i-1, p), w_i + L(i-1, p - v_i)\} & \text{otherwise} \end{cases}$$

1. Give a recurrence relation for the running time of this recursive algorithm.
2. Solve the recurrence to get an upper bound on the running time.
3. What is the running time of the Dynamic Programming solution based on this recursive algorithm? Why?
4. Assuming that the maximum v_i value is $O(\sqrt{n})$, what is the space occupied by an efficient implementation of the Dynamic Programming approach to this algorithm. Justify your answer.

ANSWER

1. $T(n) = 2T(n-1) + O(1)$ if $n > 0$ else 0
(1 mark) for base case and
(1 mark) for recursive case
2. (1 mark) $T(n) = O(2^n)$
3. (1 mark) $O(n^2 \max\{v_i\})$
(1 mark) because the table has n rows and nV columns.
4. (1 mark) $O(n\sqrt{n})$
(1 mark) as only need 2 rows at a time, not all n

1 mark $n^{5/2}$

Question 4 (9 marks).

Complete the following table of tight upper bounds on worst case performance for the operation listed in each row for the map data structure in each column. You should assume that the map contains n (key,value) pairs at the time of the operation. You may also assume that hashing and comparing keys requires $O(1)$ time.

	Hash Table with m slots and separate chaining	Binary Search Tree	AVL Tree
Insert			
Find			
Return all values			

ANSWER

	Hash Table with m slots and separate chaining	Binary Search Tree	AVL Tree
Insert	$O(1)$ or $O(n)$	$O(n)$	$O(\log n)$
Find	$O(1)$ or $O(n)$	$O(n)$	$O(\log n)$
Return all values	$O(m + n)$ $O(m)$ if state $n < m$	$O(n)$	$O(n)$ or $O(n \log n)$

Question 5 (12 marks).

We are interested in comparing implementations of Huffman's algorithm to generate optimal weighted-path trees. An underlying data structure for the algorithm must support two operations: `remove-min` and `insert`, where the sum of the leaf weights is the key of each subtree.

- (a) How many internal nodes are there in a Huffman tree with n leaves?

ANSWER

1 mark - $n - 1$

- (b) How many times is `remove-min` called in Huffman's algorithm for n items?

ANSWER

1 mark - 2 times for each internal node = $2(n - 1)$ **Accept $O(n)$**

- (c) How many times is `insert` called in Huffman's algorithm for n items, assuming the original n weights are already in the data structure?

ANSWER

Accept $O(n)$

1 mark - 1 time for each internal node = $n - 1$

- (d) Give a tight bound on the worst case running time of Huffman's algorithm when the subtrees constructed during the algorithm are kept in the following data structures. You should justify your answer with reference to the previous two answers. You should assume the n input weights are sorted.

1. A standard binary heap.

ANSWER

(1 mark) `remove-min` takes $O(\log n)$, trickle-down

(1 mark) `insert` takes $O(\log n)$, trickle-up

(1 mark) total is therefore $O(n \log n)$. accept carry over errors from above.

2. A sorted linked list with a finger pointing to the last inserted subtree.

ANSWER

(1 mark) `remove-min` takes $O(1)$, head ptr

(1 mark) `insert` takes $O(n)$ over the whole algorithm (finger moves right)

(1 mark) total is therefore $O(n + n)$. accept carry over errors from above.

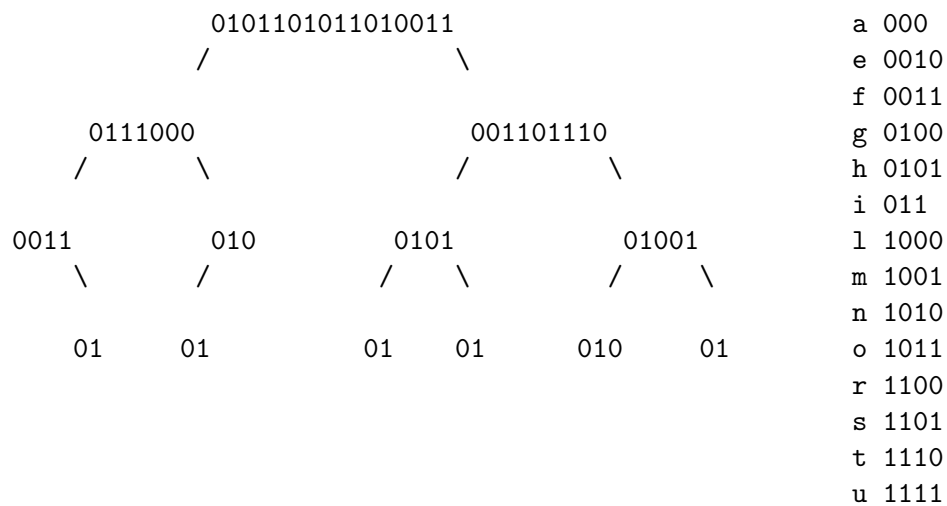
3. An AVL tree.

ANSWER

(1 mark) `remove-min` takes $O(\log n)$, left-left-left, and complicated

(1 mark) `insert` takes $O(\log n)$


(1 mark) total is therefore $O(n \log n)$. accept carry over errors from above.

Question 6 (7 marks).

- (a) What string is represented by this Wavelet Tree where each character in the string is represented by the code on the right? As usual, a 0-bit indicates left and a 1-bit indicates right.

ANSWER algorithmsarefun\$ **2 marks** algorithmsarefun\$

- (b) Assuming that rank on an individual bitvector of the tree is answered by simply scanning left-to-right and counting zeroes, how many bits at each level of the tree are inspected to answer the query rank(a, 11) in this tree? (Hint: in the first level of the tree there are 11 bits counted.)

ANSWER
1 mark per level
Level 1: 11 (obviously 0..10) 01011010110
Level 2: 5 01110
Level 3: 2 00
~~Level 4: none~~ 

- (c) Assuming rank on bitvectors can be done in $O(1)$ time, what is the cost of answering rank on a general alphabet of size σ for a string of length n using a Wavelet Tree?

ANSWER
(2 marks) $O(\log \sigma)$

Overflow Answers

The boxes here are for emergency use only. If you do need to use this page, indicate **CLEARLY** in your previous answer that you have continued onto this page. Without such an indication, it is possible that this part of your answer will be overlooked.