Curtin University

School of Electrical Engineering, Computing and Mathematical Sciences

Discipline of Computing - Curtin University

COMP3001 Design and Analysis of Algorithms

Final Assessment - Semester 1/2020

Total Mark: 100

Time allowed: 4 hours (start to finish)

Test mode: Online assessment, open-book: you are allowed to access your hand-written notes, lecture slides, textbooks, and printed and electronic materials in your possession. However, you are **NOT allowed to simply copy and paste solutions** from your open sources. Doing so will be considered academic misconduct.

CONDITIONS

- The assessment must be completed by yourself only. No one else should do this test for you. Any attempts to compromise the system are strictly prohibited. Any breaches of this policy will be considered cheating and appropriate action will be taken as per University policy.
- You are prohibited from communicating with people other than the unit coordinator/lecturer and the tutors during the test.
- You are prohibited from providing information about your work and your assessment to others during and outside your test within two days. Some students may sit in the assessment after you.
- You must complete and submit the "Student Declaration Form".
- Some students sitting the assessment can be invited to an online interview as confirmation check to confirm the authorship of the submitted final assessment. In the interview, students will be asked to explain their answers and demonstrate their knowledge for randomly selected questions. Students will be shown the questions, as well as their written answers.

INSTRUCTION

- This assessment consists of four questions: QUESTION ONE to QUESTION FOUR of different types, with a total of 100 marks. **Attempt ALL questions**.
- You can submit your answers multiple times during the assessment, but only the last submission would be used for marking.

QUESTION ONE (Total: 25 marks).

Q1 a) (5 marks). Consider the following algorithm to print out all the keys in a binary search tree in an inorder tree walk.

```
INORDER-TREE-WALK (x)
  if x ≠ NIL then
        INORDER-TREE-WALK (left[x])
        Print key[x]
        INORDER-TREE-WALK (right[x])
```

The recurrence of the running time complexity of the algorithm is $T(n) = T(k) + T(n\text{-}k\text{-}1) + 1$, where k is the number of nodes in the left subtree and n-k is the number of nodes in the right subtree.

Use induction to prove that the recurrence is $O(n)$.

Q1 b) (Total: 10 marks). It is known that no comparison-based sorting algorithm has a running time complexity less than $\Omega(n \lg n)$.

(i) **(5 marks).** Suppose that your friend *A* claims to have a comparison-based sorting algorithm that satisfies a recurrence $T(n) = 3T(n/4) + cn$. Is your friend's claim reasonable? Justify your answer.

(ii) **(5 marks).** Suppose that your friend *B* claims to have a comparison-based sorting algorithm that satisfies a recurrence $T(n) = 5T(n/4) + cn$. Would you use your friend's algorithm over merge sort for large n? Justify your answer.

Hint: Solve the recurrence functions in (i) and (ii) to justify your answers. You don't need to check for the regularity condition in your solution.

Q1 c) (4 marks). Without using Stirling's formula, prove that $\log(n!)$ is $O(n \log n)$. You need to give the values of n_0 and c.

Hint: $\log (a * b) = \log a + \log b$.
$n! = n * (n\text{-}1) * (n\text{-}2) * \ldots * 2 * 1$.

Q1 d) (Total: 6 marks).

```
for (i ← 0 to n) do
   for (j ← 0 to n) do
      if (A[i] = B [j]) then
         return (TRUE)
```

(i) **(2 marks).** What does the algorithm do?

(ii) **(2 marks).** What is the *best case* upper bound running time complexity of the algorithm? Justify your answer.

(iii) **(2 marks).** What is the *worst case* upper bound running time complexity of the algorithm? Justify your answer.

END OF QUESTION ONE

QUESTION TWO (Total: 26 marks).

Q2 a) (Total: 10 marks). Suppose you were to run a marathon from a point **A** to another point **B** along a given route. In the marathon, you were allowed to stop and get water as many times as needed. However, since each stop takes time, you want to minimize the total number of stops as possible. Assume you know that you can run for **up to *k* km after every stop**. Further, you were given information about the locations and distances between water stations along the route. Let $S = (s_1, s_2, \ldots, s_n)$ be the set of n water stations along the route, and $D = (d_1, d_2, \ldots, d_n)$ be the set of their corresponding distances from point **A** to the station. In other words, s_i is the water station with distance d_i from point A. Note that $d_1 < d_2 < \ldots < d_n$, and assume the distance between neighbouring stations is at most k kms, and the distance between the last station and point **B** is k km. For example, consider for k = 15 kms and D = (5, 10, 13, 16, 20, 24, 27, 31, 36, 40, 45, 50). The optimal sequence of stops is (s_3, s_7, s_{10}, s_{12}), i.e., 4 stops.

(i) **(6 marks).** Write the pseudo code of your algorithm to determine a set of water stations s_i that minimizes the total number of stops.

(ii) **(2 marks).** Explain the idea of your algorithm, and use the given example for k=12 km to illustrate the working of your algorithm.

(iii) **(2 marks).** Analyse the time complexity of your algorithm as a function of n, where n is the total number of water stations along the route fro **A** to **B**. Also, argue why your algorithm is the most efficient possible.

Q2 b) (Total: 8 marks). Consider d sequences of elements. The elements in each sequence are already sorted in increasing order, and there are in total n elements. Design an $O(n \lg d)$ algorithm to merge all the sequences into one sorted sequence (in increasing order). As an example, for four sequences: (2, 5, 7), (4, 6), (1, 8), and (3, 8, 9), we have d = 4 and n = 10. Your algorithm should produce a sorted sequence (1, 2, 3, 4, 5, 6, 7, 8, 8, 9).

(i) **(6 marks).** Describe your algorithm. Show that the time complexity of your algorithm is $O(n \lg d)$. You can assume d is in a power of 2, e.g., 4, 8, 16. **You are NOT required to write the pseudocode of your algorithm.**

(ii) **(2 marks).** Show how your algorithm merges the d=4 sequences in the example.

Q2 c) (8 marks). Consider an array of records, A, whose keys to be sorted only consist of F's and M's, for Female and Male respectively. For example, array A may contain the a list of keys A=(F M M M F F F M F M F M M F F M).

- Write the pseudo code of a $O(n)$ algorithm to sort the array in place, i.e., using memory storage of no more than constant size in addition to that of the array.
- Illustrate the working of your algorithm using the example as its input.
- Analyse the time complexity of the algorithm and show that its complexity is $O(n)$.

END OF QUESTION TWO

QUESTION THREE (Total: 17 marks).

Q3 a) (Total: 6 marks).

(i) **(3 marks).** Working modulo 11, list all spurious hits that the Rabin-Karp matcher encounters in the text T = 3142531653586797 when looking for pattern P = 53.

(ii) **(3 marks).** Given t_3 = 3 for T = 3142531653586797, show in detail how Rabin-Karp matcher computes t_4.

Q3 b) (2 marks). Consider the following BF_String_Matcher (T, P) algorithm (from Lecture Slide).

```
BF_String_Matcher(T, P)
1     n = length [T]
2     m = length[P]
3     for s = 0 to n - m
4           do if P[1..m] = T[s+1..s+m]
5                 then shift s is valid
```

For T = <AAAAAAAAAA> and P = <AAC>, the algorithm requires

A. 3 comparisons between characters in T and P.
B. 14 comparisons between characters in T and P.
C. 24 comparisons between characters in T and P.
D. 21 comparisons between characters in T and P.
E. None of the answers are correct.

Q3 c) (Total: 9 marks). Data compression.

A file has alphabets {A, B, C, D, E}, where A has frequency 32, B has frequency 64, C has frequency 8, D has frequency 16, and E has frequency 8.

(i) **(3 marks).** Give the Huffman code for each of the alphabets.

(ii) **(3 marks).** How many bits are needed to represent the alphabets in the file? Explain your answer.

(iii) **(3 marks).** Use the Shannon's entropy to show that the computed Huffman code is optimal. Give the details of you computation.

END OF QUESTION THREE

QUESTION FOUR (Total: 32 marks).

Q4 a) (4 marks). Explain if the following statement is **True** or **False**. Your explanation must include detailed computation of the number of scalar multiplications of the two alternative full parenthesizations.

Consider the matrix chain multiplication problem with p_0=1000, p_1=100, p_2=20, p_3=10, and p_4=1000. A full parenthesization $(((A_1A_2)A_3)A_4)$ requires more scalar multiplications than another full parenthesization $((A_1(A_2A_3))A_4)$.

Q4 b) (Total: 10 marks). Consider the following **incomplete** tables generated by the dynamic programming algorithm (discussed in the lecture) for solving the matrix chain problem, where the four consecutive matrices are A, B, C, and D.

m

j \ *i*	1	2	3	4
1	0			
2	72	0		
3	108	36	0	
4	?	42	18	0

s

j \ *i*	1	2	3	4
1				
2	1			
3	1	2		
4	?	2	3	

(i) **(2 marks).** If matrices A and D have dimensions A = 6×4, and D = 3×2, what are the dimensions of matrices B and C? Explain your answer.

(ii) **(6 marks).** Calculate *m*[1, 4] and *s*[1, 4]. Show your detailed calculations.

(iii) **(2 marks).** Show the optimal bracketing of the matrices in part (i) that produces the number of multiplications in *m*[1, 4].

Q4 c) (Total: 8 marks). Consider 5 items with weights w = (2, 6, 2, 4, 5) and values v = (40, 40, 25, 30, 50). For example, the weight of item 1 is 2 and its value is 40, while item 5 has a weight of 5 and a value of 50.

(i) **(6 marks).** Use the 0/1 Knapsack dynamic programming algorithm (discussed in the lecture) to fill out the following Table P, assuming the total weight of the selected items is at most 9 units. In your answer, you can represent the entry at row i and column k in Table P as P[i, k]. For example, P[5, 9] = **x** and P[3, 7] = **y**.

i / k	0	1	2	3	4	5	6	7	8	9
5										**x**
4										
3								**y**		
2										
1										

(ii) **(2 marks).** Use Table P to determine what items should be selected to achieve maximum values. Explain your answer

Q4 d) (Total: 10 marks). Consider the following parallel search algorithm that requires CRCW model.

Algorithm Parallel_Search (*x*, *A*[1 .. *n*])

index ← -1
forall P_i **do in parallel**
  if $A[i] = x$ **then**
    index ← i
  endif
endfor

(i) **(4 marks).** Modify the algorithm to find each index in array A that stores x. Use array B to store the indices. **Hint.** $B[i] = -1$ means $A[i] \neq x$.

(ii) **(2 marks).** Is the modified algorithm cost optimal? Justify your answer.

(iii) **(2 marks).** Compute the speedup of the modified algorithm.

(iv) **(2 marks).** Does the modified algorithm also require CRCW model? Explain your answer.

END OF EXAMINATION