FIT2004 SAMPLE EXAM PAPER

INSTRUCTIONS

- You must answer ALL the questions.
- Answers to each question should be in the space DIRECTLY BELOW the questions and (if required) on the blank page overleaf of each question.
- Script book may be used if ADDITIONAL SPACE is required for answering these questions

General exam technique

Some candidates throw marks away by not attempting all questions. Suppose you get 7/10 on a question for a 20 minutes effort. Spending another half hour on the question gets *at most* 3 more marks. On the other hand, if you spend that time on a new question, you might get another 10 marks, or more.

Answer the question that is asked. If the question asks for Insertion sort, do not give Quicksort. Where necessary, especially where it says "No explanation, no marks", justify your answer with a clear explanation.

Some of the questions ask you to write Pseudocode. A Pseudocode is essentially a high-level description of your program, that should allow a human to understand it. If you feel more comfortable using Python/Java syntax, you are welcome to use it but don't get bogged down in syntax. What will essentially be assessed in such questions is your basic understanding of the algorithm (and not the syntactical correctness).

Good Luck!

Section 1 (Short Questions) (Section weight = 25 marks)

This section is composed of 6 questions: (A) to (F)

(A) Let the array `arr[1...N]` contains N characters from the alphabet {'A','B','C'}. Write pseudocode to group these characters (so that same characters are adjacent to each other) in the order defined by the alphabet. **(4 marks)**

4

(Page intentionally left blank for working space)

(B) The amount of time, T_N taken by an algorithm is typically a function of the size, N, of the input data. Assume that you wrote a program that shows the following time recurrence:

$$T_N = \begin{cases} T_{\frac{N}{2}} + a, & \text{if } N > 1 \\ b & \text{if } N = 1, \end{cases}$$

where a and b are constants.

(i) Work out the solution of T_N from this recurrence.

(ii) What is the time complexity of your program in Big-O notation?

(4 marks)

4

(Page intentionally left blank for working space)

(C) Assume you are given an array of N integers, `a[1...N]`.

(i) Write a pseudocode to sort these numbers using insertion sort.

(ii) Identify the loop invariant of your algorithm.

(3 marks)

3

(Page intentionally left blank for working space)

(D) Explain (in no more than a few sentences) the answers to following questions.

(i) How many undirected graphs are possible containing n vertices? (Assume vertices with NO self-edges; vertices can also be disconnected.)

(ii) What is the worst-case time complexity of Radix sort? (Specify any assumptions you are making when describing its complexity.)

(iii) What is the upper bound on the length of any simple path in an undirected and unweighted graph?

(5 marks)

5

(Page intentionally left blank for working space)

(E) Draw the Directed Graph that corresponds to the following information:

```
Gough is older than Mel, Paul and Kylie.
Claudia is older than Gough and Kylie.
Kylie is older than Natalie.
Natalie is older than Paul.
Mel is older than Natalie and Paul.
```

In addition, suggest if the graph you have drawn is acyclic or not.

(4 marks)

4

(Page intentionally left blank for working space)

(F) What is the space complexity to store a suffix tree (*i.e.*, a compact suffix trie) of a string of n characters? (Assume the alphabet from which the string is derived to be a constant.) You should justify your answer in clear terms. **(5 marks)**

5

(Page intentionally left blank for working space)

Section 2 (Program verification and correctness) (Section weight = 8 marks)

This section is composed of 3 question: (A) to (C)

(A) Write pseudocode describing a routine that searches for a given number in an array of sorted integers.

(B) What is the loop invariant of this program?

(C) Reason why your program always terminates correctly?

(8 marks)

8

(Page intentionally left blank for working space)

Section 3 (Retrieval Data structures) (Section weight = 8 marks)

This section is composed of 2 questions: (A) and (B)

(A) Starting with an initially empty AVL-tree, draw the resulting AVL-tree after inserting the following elements one after another. 50, 70, 30, 10, 20, 15

(4 marks)

4

(Page intentionally left blank for working space)

(B) Describe (in plain english) the general steps involved in inserting an element into a hash table using Cuckoo Hashing. **(4 marks)**

4

(Page intentionally left blank for working space)

Section 4 (Graphs) **(Section weight = 10 marks)**

This section is composed of 2 questions, (A) and (B)

(A) Write pseudocode describing Bellman-Ford algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted graph.

(5 marks)

5

(Page intentionally left blank for working space)

(B) List the steps involved in using the Ford-Fulkerson's algorithm to find the maximum flow in a given directed network with positive integer capacities associated with its edges.

(5 marks)

5

(Page intentionally left blank for working space)

Section 5 (Miscellaneous) (Section weight = 9 marks)

This section is composed of 2 questions: (A) and (B)

(A) Write the pseudocode describing a function that generates all permutations of a given set of N elements using N-ary recursion. (**Your program should strictly use N-ary recursion to generate the permutations. Other methods will not be marked.**)

(4 marks)

4

(Page intentionally left blank for working space)

(B) The *longest common subsequence* (LCS) problem requires one to find the *longest* possible *subsequence* shared between two different strings $s1$ and $s2$. Note that a **subsequence** is *different* from a **substring**, in that the letters in a subsequence need not appear consecutively in the original string, whereas in a substring they do appear consecutively.

For example, if a string s = `abcdefghijklm`, a **subsequence** is some (or all) of its elements, in the same order, for instance, `cfijm` is the subsequence of s. If $s1$ = `arun` and $s2$ = `arvind`, the LCS between these two string is `arn` – there is no other subsequence common between $s1$ and $s2$ that is longer than the common subsequence `arn`.

The LCS problem between any two given strings $s1$ and $s2$ can be solved using a dynamic programming approach, which is similar to the approach used to solving the edit distance problem.

Write the dynamic programming **recurrence relationship** for this problem, including the boundary conditions.

(5 marks)

5

(Page intentionally left blank for working space)

End of Exam