The University of Nottingham Ningbo China

Centre for English Language Education

Semester One, 2020-2021

Introduction to Algorithms

Time allowed 2 Hours

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced.

This paper contains SIX questions which carry equal marks.

An indication is given of the weighting of each subsection of a question by means of a figure enclosed by square brackets, eg. [3], *immediately following that subsection.*

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

Do NOT turn examination paper over until instructed to do so.

ADDITIONAL MATERIAL: *NONE*

INFORMATION FOR INVIGILATORS:

1. *Please give a 15 minute warning.*
2. *Please collect Answer Booklets, Question Papers, and Formula Sheet at the end of the exam.*

This page is intentionally left blank

1. (a) Consider a calendar system with 12 months in a year. The first 6 months of the year are each 31 days long. The next 5 months are each 30 days long and the last month of the year is 29 days long.

   Write an algorithm called `myCal()`, which takes an integer $(1 \leq n \leq 365)$ and returns a list containing `[day,month]`. For instance calling `myCal(120)` should return `[27,4]`, i.e. the 27th day of the 4th month.

   [6]

   (b) A leap year is a year that has one extra day (366 days). Leap years occur every 4 years however, century years (like 1700, 1800,$\cdots$) are not leap years in general unless they are divisible by 400 (like the year 2000).

   Write an algorithm `isLeap()` that takes a number (year) and returns a Boolean value True if the year is leap and False otherwise. Your algorithm must have one line of compounded conditional statement. What is the result of `isLeap(1964)`?

   [4]

2. (a) (i) Write a recursive algorithm called `collatz(n)` which takes a positive integer `n`. If `n` is even the algorithm returns half of `n` and if `n` is odd the algorithm returns three times `n` plus one. The algorithm stops when `n=1` and returns `1`.

   (ii) Trace your algorithm for `n=10` showing all the intermediate steps.

   [5]

   (b) (i) Suppose you are given a list whose elements are positive integers. Write a recursive algorithm called `listMax()` that takes such a list as its input and returns the maximum element of that list. You should use a helper function called `ListMaxHelper()`.

   (ii) Trace your algorithm for the following list: `L=[10,8,12]`.

   [5]

3. (a) Write a recursive algorithm called `mySqrt(n)` that takes a positive integer n $(n > 1)$ and returns another integer m such that $m \times m \leq n$. You must use a helper function called `mySqrtHelper`. For instance: `mySqrt(20)=4; mySqrt(105)=10`.

[4]

(b) (i) Write a recursive algorithm `isPrime(p)` that takes a positive integer p $(p > 1)$ and returns True if p is prime and False otherwise. You should call your `mySqrt()` function from above in `isPrime(p)` algorithm.

(ii) Trace your algorithm for `isPrime(19)`.

[6]

4. (a) Write a recursive algorithm `product()` that takes two *non-empty* lists of numbers, which are also of the *same length* and returns a list whose elements are the product of the elements that are in similar positions in the input lists. For example:

`product([1,2,3],[9,3,7])=[1×9,2×3,3×7]=[9,6,21]`

`product([2],[6])=[2×6]=[12]`

[4]

(b) Consider the unsorted list `L=[9,0,6,8,12]`. Apply the *divide & conquer* scheme of merge-sort to the given list until you get a sorted (ascending) list. You must draw a clear diagram showing each step of divide & conquer and label the relevant action in each step (i.e. split, sort, merge).

[4]

(c) Draw a balanced binary search tree based on the sorted list that you have obtained in 4(b).

[2]

5. (a) Draw the tree given in the following nested command:

`node(node(node(leaf,1,node(leaf,2,leaf)),3,leaf),5,node(leaf,7,node(leaf,8,leaf)))`

[2]

(b) Is the tree drawn in 5(a), a binary search tree? Briefly explain why or why not.

[2]

(c) What is the depth of this tree? Justify your answer briefly.

[2]

(d) Add two more nodes to this tree with values 0 and 10. Redraw your tree with the added nodes in a way that the depth of the tree remains unchanged.

[2]

(e) Which one of the tree traversal schemes applied on binary search trees will return a sorted (ascending) list? Apply this scheme to the tree below (Fig 5.1) by carefully showing each step.

[2]

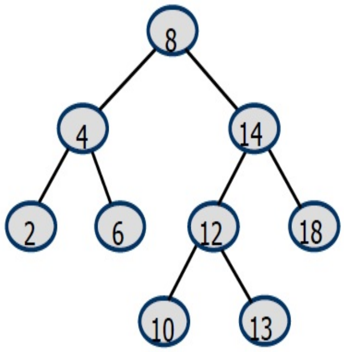

Fig 5.1

6. (a) The following graph (Fig 6.1) is a bipartite graph.

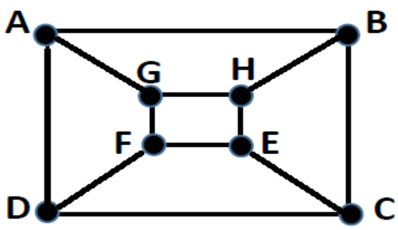

Fig 6.1

(i) Give a reason why it is bipartite?

(ii) Redraw the graph by arranging the vertices in two separate sets.

[2]

(b) (i) State the formula for the number of edges of a complete graph with n vertices.

(ii) Draw a complete graph with 5 vertices and check that your formula for the number of its edges is correct.

[2]

(c) Decide which one of the following graphs (Fig 6.2) has an Eulerian path or Eulerian tour or neither. In each case give a brief reason.

[3]

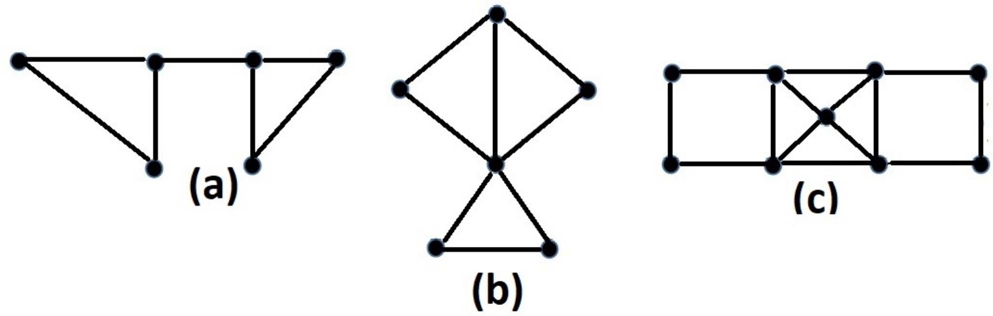

Fig 6.2

(d) (i) Use Kruskal's algorithm to find the minimum spanning tree for the weighted graph in Fig 6.3. Carefully state which edge is being selected by writing the necessary steps.

(ii) Compute the minimum cost of traversal in the spanning tree.

[3]

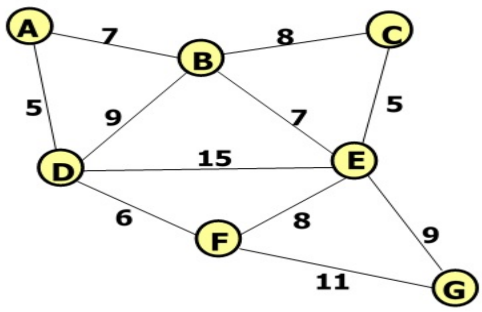

Fig 6.3