Beijing-Dublin International College

SEMESTER 2 EXAMINATION - 2018/2019

School of Computer Science

COMP2014J Data Structures and Algorithms II (Software Engineering)

Dr. Rosemary Monahan
Prof. Pádraig Cunningham
Dr. David Lillis *

Time Allowed: 120 minutes

Instructions for Candidates

Answer any 2 questions. All questions carry equal marks.

BJUT Student ID: _____ **UCD Student ID: _____**

I have read and clearly understand the Examination Rules of both Beijing University of Technology and University College Dublin. I am aware of the Punishment for Violating the Rules of Beijing University of Technology and/or University College Dublin. I hereby promise to abide by the relevant rules and regulations by not giving or receiving any help during the exam. If caught violating the rules, I accept the punishment thereof.

Honesty Pledge：_____ (Signature)

Instructions for Invigilators

No special instructions.

Question 1:

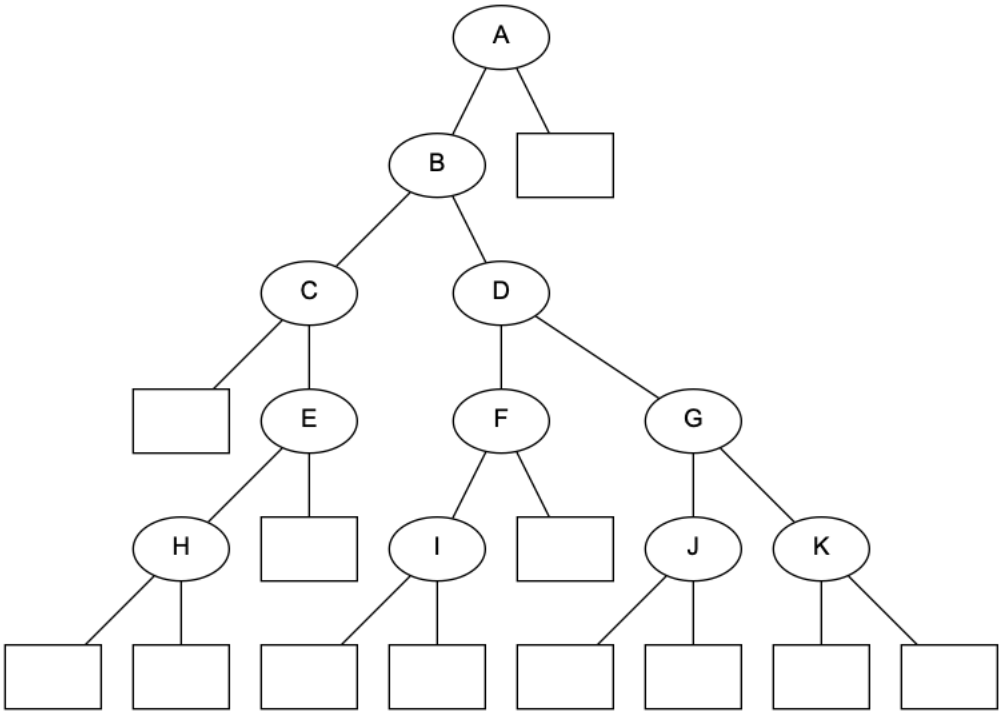

Figure 1

(a) Study the tree in Figure 1 and answer the questions that follow.

(i) List the siblings of node H.

(ii) What is the depth of node E?

(iii) What is the degree of node F?

(iv) What is the height of the tree?

(v) List the descendants of node D.

(vi) Is (H,E) an edge? Explain your answer.

(viii) List the nodes that are in the subtree that is rooted at C.

(ix) Which of the following phrases is the best description for this tree: *binary search tree*, *proper binary tree*, *complete binary tree*. Explain your answer.

[8 marks]

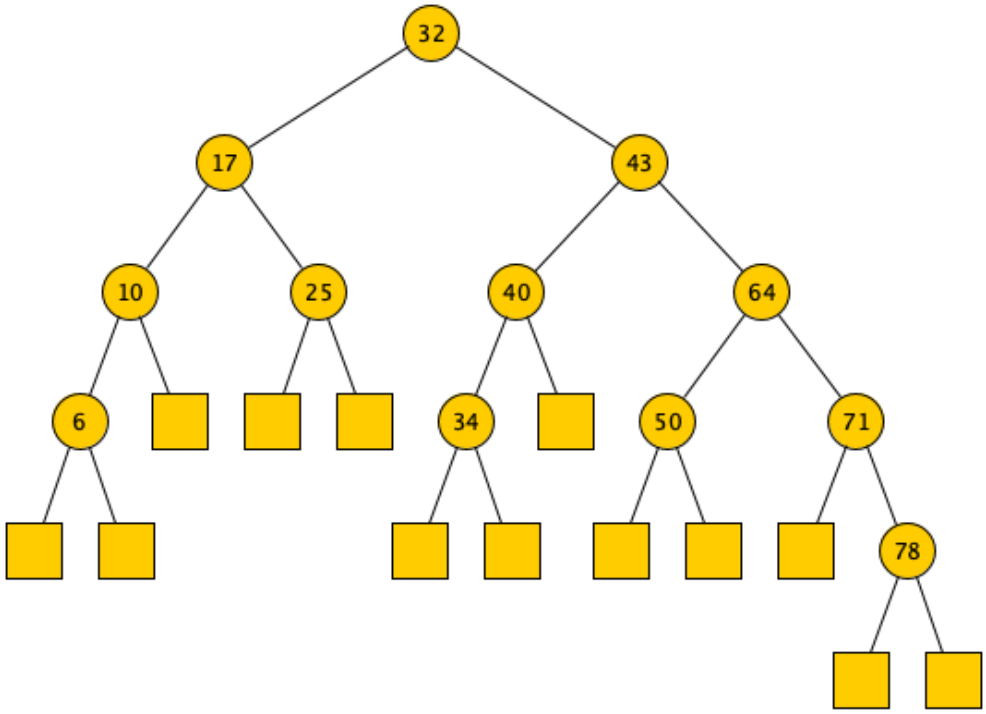

Figure 2

(b) Assume that the tree in Figure 2 is an **AVL Tree**. Draw the state of the tree after performing the following operations. In your answer, you should show the tree's state after each step and mention any restructuring that is required.

(i) Remove 25

(ii) Insert 35

(iii) Remove 50

(iv) Insert 3

(v) Remove 10

[12 marks]

(c) With regard to analysing the performance of a **Binary Search Tree**, discuss each of the following:

(i) Space requirements.

(ii) Time complexity of the insert() and remove() functions.

(iii) The effect that a balanced tree has on the time complexity of the insert() and remove() functions.

[10 marks]

(d) Assume that the tree in Figure 2 is a Splay Tree. If the next operation is to remove 78, draw trees to show all restructuring operations that will happen and state what restructuring operations these are.

[6 marks]

(e) A Splay Tree is said to be balanced in the *amortised sense*. Briefly describe what is meant by this.

[4 marks]

(f) Answer the following questions relating to the traversal of binary trees.

(i) List the values found during a *postorder* traversal of the tree in Figure 2.

(ii) List the values found during an *preorder* traversal of the tree in Figure 2.

(iii) The following are two traversals of a binary tree. Draw the tree[1].

Preorder: H, C, A, F, G, D, E, B

Inorder: C, H, A, G, F, B, E, D

[10 marks]

[Total 50 marks]

[1] In the original exam paper there was an error in this question that made it impossible to solve. This has been corrected in this version.

Question 2:

(a) Give definitions of the following graph concepts: *incident edges, adjacent vertices, degree of a vertex, paths,* and *cycles.*

[10 marks]

(b) One implementation of a *priority queue* ADT is to use a *heap*.

(i) Heaps make use of two key properties: the *order property* and the *structural completeness property*. Give definitions for each of these properties.

[4 marks]

(ii) Describe the steps by which items are removed from a heap. Illustrate your answer by removing an item from the heap in Figure 3.

[6 marks]

(iii) Describe the steps by which items are inserted into a heap. Illustrate your answer by inserting an item with priority 9 into the heap in Figure 3.

[6 marks]

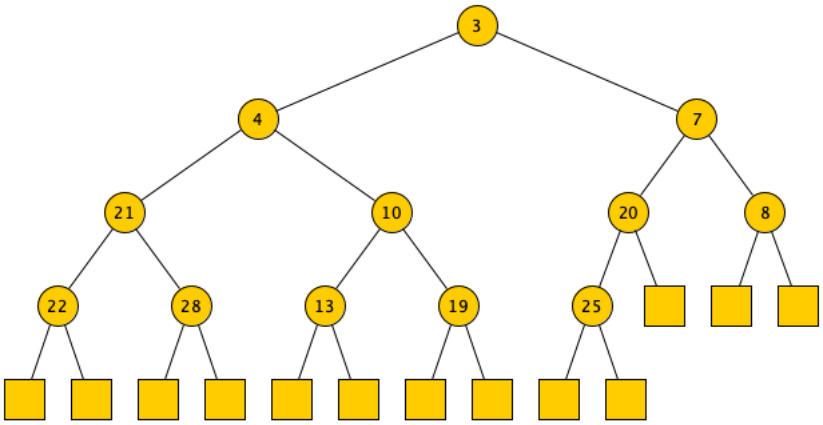

Figure 3

(c) An array-based list is an appropriate way to implement a *Complete Binary Tree*.

(i) Explain in detail the advantages of using this type of implementation type compared to using a linked structure.

[6 marks]

(ii) Why is an array-based list not suitable to represent an AVL tree?

[4 marks]

(d) *Huffman Encoding* is a technique for text compression that uses a *Huffman Tree*. Using the string “go go gophers” as an example, describe in detail how a Huffman Tree is created and how it can be used to generate a code to represent the string.

[14 marks]

[Total 50 marks]

Question 3:

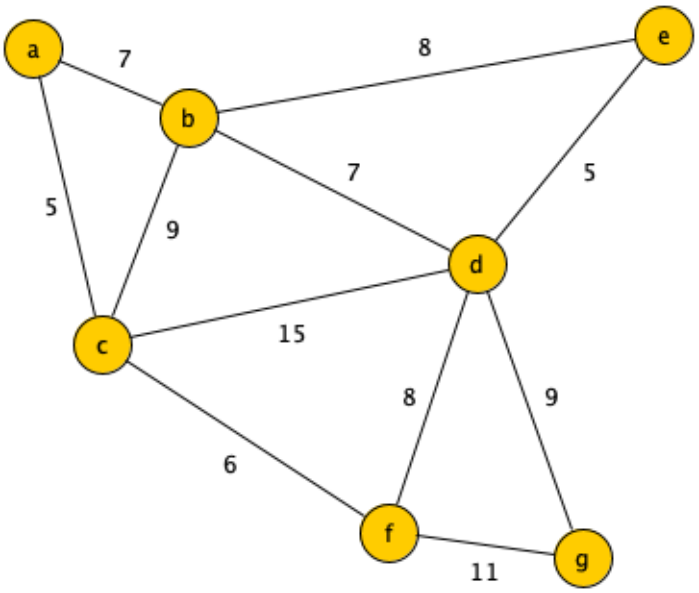

Figure 4

(a) Using the graph in Figure 4 as an example, show how Kruskal's algorithm can be used to compute a *Minimum Spanning Tree*. In your answer, you must explain each step that you take.

[12 marks]

(b) Using the graph in Figure 4 as an example, show how Dijkstra's algorithm can be used to compute a *shortest distance tree* beginning at the vertex containing "g". In your answer, you must explain each step that you take.

[12 marks]

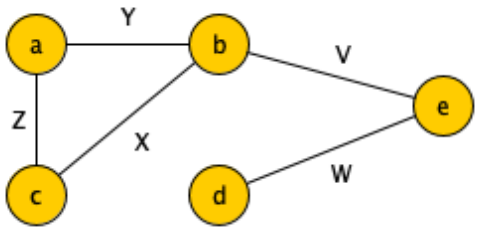

Figure 5

(c) Draw a diagram to show how the graph in Figure 5 can be represented using an *adjacency matrix* structure. In your answer, describe each object type and data structure used to explain its purpose and what data it stores.

[12 marks]

(d) There are three main implementation strategies for the Graph ADT: *edge list*, *adjacency list*, and *adjacency matrix*. Compare the performance of each implementation strategy for each of the following headings. In each case, briefly explain the reason for any difference in performance.

(i) Space (memory) usage.

(ii) incidentEdges(v) method.

(iii) areAdjacent(v,w) method.

(iv) removeVertex(v) method.

(v) removeEdge(e) method.

[14 marks]

[Total 50 marks]