UNIVERSITY OF OTAGO EXAMINATIONS 2020

COMPUTER SCIENCE

Paper COSC242

ALGORITHMS & DATA STRUCTURES

Semester 2

(TIME ALLOWED: THREE HOURS)

This examination comprises 7 pages.

Candidates should answer questions as follows:

Candidates must answer **all** questions.
Questions are worth various marks are shown thus:

(5)

The total number of marks available for this examination is 100.
You should keep your answers short.
In general, if there are two marks for an answer, you should have two things to say.
If you leave an answer blank, you will receive zero marks for that question. It is in your best interest to attempt every question, as you may earn partial marks.

The following material is provided:

Appendix A: Pseudocode for RBT-Insert-Fixup function
Appendix B: Pseudocode for RBT-Delete-Fixup function

Use of calculators and other material:

No calculators are permitted. No other materials are provided or permitted.

Other instructions:

Cross out your rough working so that your answers are clear. If it is unclear what parts are your answers, you may lose significant marks.

TURN OVER

1. **Complexity classes**

   (a) What are two input cases where linear search on an input array of length n will achieve worst case time complexity?

   (2)

   (b) What is the worst case time complexity of Binary Search on a sorted input array of length n?

   (2)

   (c) What is the worst case time complexity of Quicksort on an input array of length n?

   (2)

   (d) Suppose you need to sort an array of 1 million integers. Memory is very tight. Should you use Insertion Sort, Merge Sort, or Quicksort? Justify your answer.

   (4)

2. **Recurrences, Big-O, Proof Techniques**

   (a) Given the function $f(n) = 3n^2 + 15n + 2$ for all n, show that $f = \Theta(n^2)$.

   (5)

   (b) Use proof by induction to show that $2^n = O(n!)$

   (5)

   (c) Use the iteration method to solve the recurrence equations:

   $$f(1) = 1$$
   $$f(n) = n \cdot f(n-1)$$

   (You need not prove that your solution is correct.)

   (5)

   (d) Using the recursion tree method, show that the average case running time by Quicksort using a 9-to-1 split (90%/10%), is $O(n \log n)$. Draw the complete tree down to depth 3, and include work done at each tree depth, and indicate important termination base cases.

   (10)

3. **Hash Tables**

(a) Given a table of size 10 and input keys k = $\{23, 9, 3, 26, 19, 43, 18\}$ (in that order), and the hash function $h_1(k) = ((a \cdot k + b)\%p)\%m$, where $a = 10, b = 9, m = 10, p = 101$, draw the hash tables that results from:

(i) Chaining.

(5)

(ii) Open addressing with double hashing, where $h(k, i) = (h_1(k)+i \cdot h_2(k))\%m$, with the secondary hash function $h_2(k) = 1 + (k\%9)$.

(5)

(b) Suppose you were using a perfect hashing scheme to create a hash table from the keys in Q3a (above). Would $h_1(k)$ be acceptable as the primary hash function? Give your reasoning. Note: no marks will be awarded for a yes/no answer without reasoning.

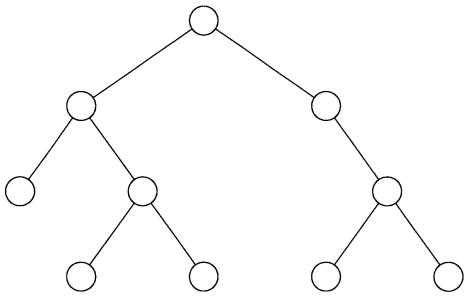
(5)

4. **Trees**

(a) Write a recursive algorithm for calculating the height of a binary tree. *Hint: the height of a tree is the number of edges from root to the deepest leaf.*

(4)

(b) Using the tree structure below, enter node keys such that an in-order walk will produce the output {A, B, C, D, E, F, G, H, I, J}.

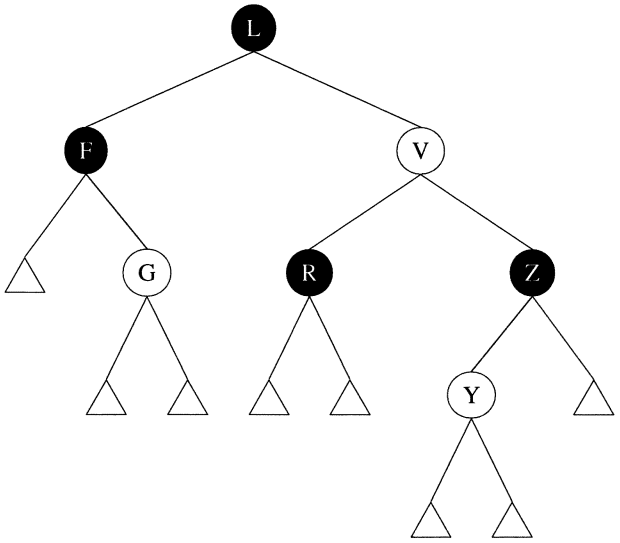


(3)

(c) Show all the red-black trees that result after successively inserting the keys $20, 15, 10, 18, 16$ into an initially empty red-black tree. State which cases apply. Make sure you label each node either red or black.

(8)

(d) Consider the following RBT. In this figure, Black nodes are heavily shaded (e.g., L), while Red nodes are unshaded (e.g., V):



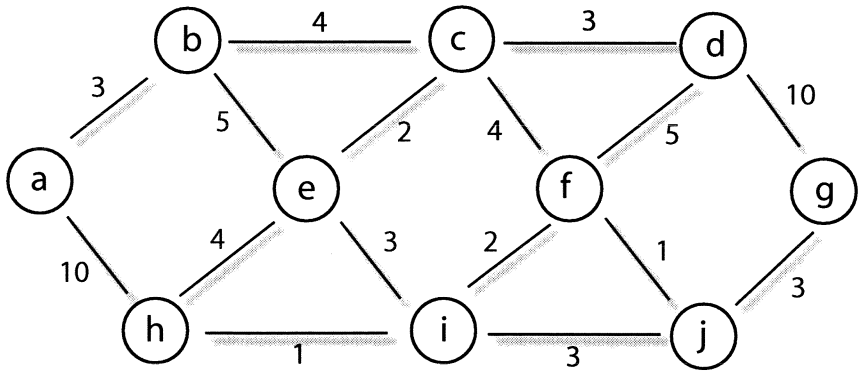
Show all the intermediate trees and the cases that apply when deleting G, then F, then R, then L from the tree in that order.

(10)

5. **Graphs**

Copy the following weighted undirected graph into your answer book. Show how Dijkstra's algorithm would find the shortest paths from source *a*. Show clearly the state of the priority queue and the current state of the shortest path graph after each operation. You will not get any marks if you only draw the final shortest path graph.

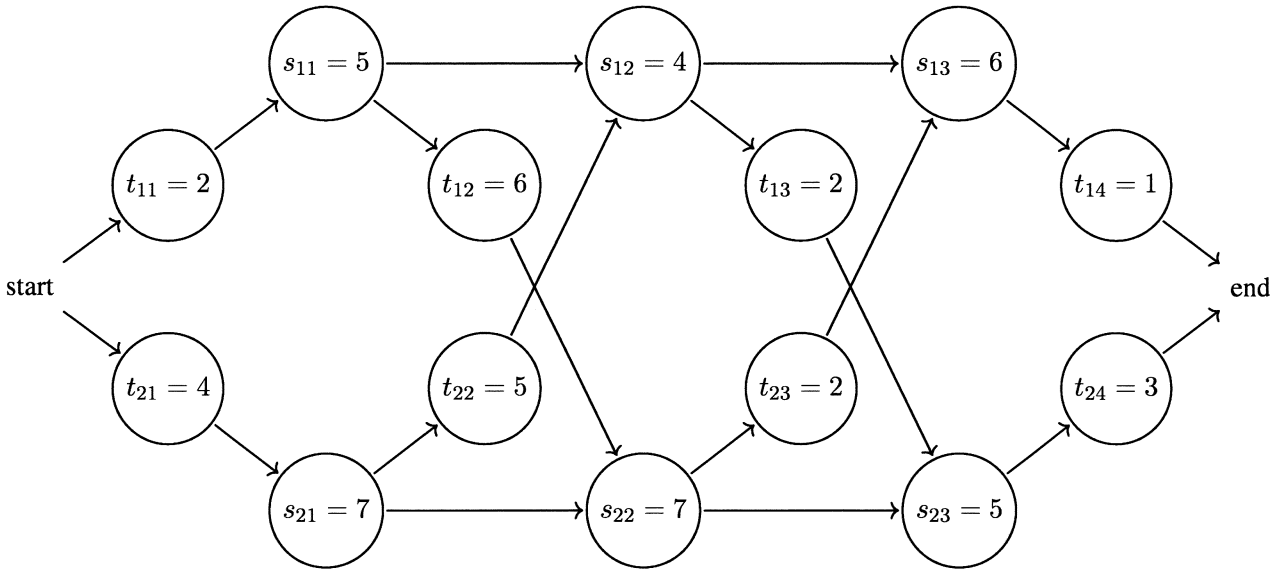
(10)



6. **Dynamic programming**

Consider the assembly line scheduling problem below. Stations are indicated as nodes, and the assembly time required at each station is notated. For example, the time taken by Station 2 on Line 1, s_{12}, is 4. Give a dynamic programming solution. Show any bottom-up tables used in your solution and any calculations you perform. Explain what the entries in your tables mean.

(10)



7. **P and NP**

In a few well-chosen sentences, explain what the classes P and NP are, and what it means to say that a problem is NP-complete. Give one example of an NP-complete problem.

(5)

Appendix A: Pseudocode for RBT_Insert_Fixup

```
procedure RBT_Insert_Fixup(T, z)
1:      while z→parent.colour == RED
2:        if z→parent == z→parent→parent→left
3:          y = z→parent→parent→right
4:          if y→colour == RED
5:            z→parent.colour = BLACK                  // case 1
6:            y.colour = BLACK                         // case 1
7:            z→parent→parent = RED                    // case 1
8:            z = z→parent→parent                      // case 1
9:          else
10:           if z = z→parent→right
11:             z = z→parent                           // case 2
12:             RBT_Rotate_Left(T, z)                  // case 2
13:           end if
14:           z→parent.colour = BLACK                  // case 3
15:           z→parent→parent = RED                    // case 3
16:         RBT_Rotate_Right(T, z→parent→parent)       // case 3
17:       else ... (same as then clause, with "right" and "left" exchanged)
18      T→root.colour = BLACK
```

Appendix B: Pseudocode for RBT_Delete_Fixup

```
procedure RBT_delete_fixup(T, x)
1:    while x != root and x.colour == BLACK
2:       if x == x→parent→left
3:          w = x→parent→right
4:          if w.colour == RED
5:             w.colour = BLACK                  // Case 1
6:             x→parent.colour = RED             // Case 1
7:             Left_rotate(T, x→parent)          // Case 1
8:             w = x→parent→right                // Case 1
9:          if w→left.colour == BLACK and w→right.colour == BLACK
10:            w.colour = RED                    // Case 2
11:            x = x→parent                      // Case 2
12:         else if w→right.colour == BLACK
13:               w→left.colour = BLACK          // Case 3
14:               w.colour = RED                 // Case 3
15:               Right_rotate(T, w)             // Case 3
16:               w = x→parent→right             // Case 3
17:            w.colour = x→parent.colour
18:            x→parent.colour = BLACK           // Case 4
19:            w→right.colour = BLACK            // Case 4
20:            Left_rotate(T, x→parent)          // Case 4
21:            x = T→root                        // Case 4
22:      else ... (same as L2 if clause, with "right" and "left" exchanged)
23:   x.colour = BLACK
end function
```


END

COSC 242

PLEASE DO NOT

TURN OVER YOUR

EXAMINATION

PAPER UNTIL

INSTRUCTED TO

DO SO