TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2016
TRIMESTER 1 *** WITH SOLUTIONS ***

# COMP 261
# ALGORITHMS and DATA STRUCTURES

**Time Allowed:** TWO HOURS

**CLOSED BOOK**

**Permitted materials:** Only silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted in this examination.

**Instructions:** Attempt ALL Questions.

Answer in the appropriate boxes if possible — if you write your answer elsewhere, make it clear where your answer can be found.

The exam will be marked out of 120 marks.

Non-electronic foreign to English language dictionaries are permitted.

Alphabetic order: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

| Questions | Marks |
|---|---|
| 1. MST | [5] |
| 2. Articulation Points | [20] |
| 3. Parsing | [20] |
| 4. Tries and String Search | [25] |
| 5. B+ Trees | [15] |
| 6. Compression | [15] |
| 7. Flood Fill | [20] |

## Question 1. MST [5 marks]

**(a)** [2 marks] Give an example of a smallest graph such that Kruskal's algorithm and Prim's algorithm will add edges to the MST in a different order (but the final trees may be the same). Show the order in which the trees are constructed.

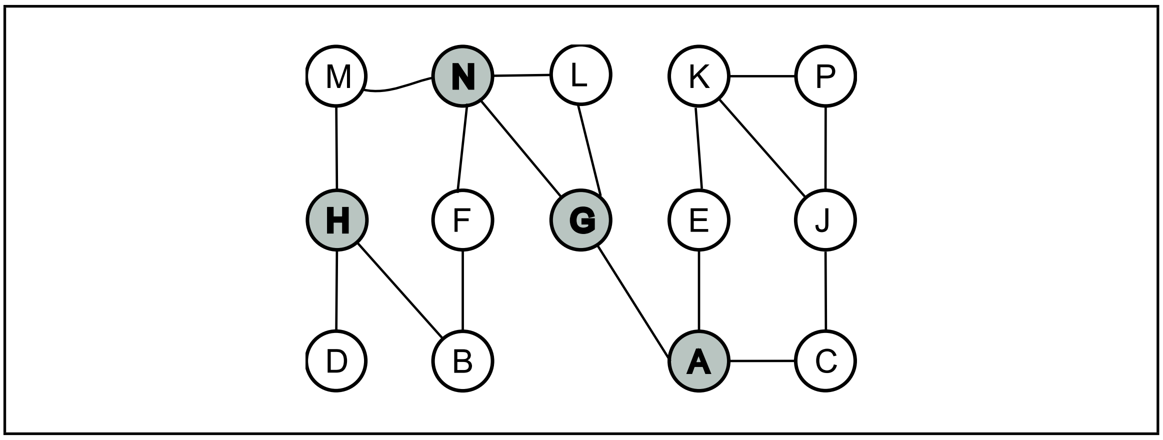
Yes, will lose points if not the smallest.

**(b)** [3 marks] Give an example of a smallest graph such that the two MSTs resulting from the two algorithms are different. Show the order in which the trees are constructed.

-

## Question 2. Articulation Points [20 marks]

**(a)** [4 marks] Circle the articulation points in the following graph.



**(b)** [6 marks] Give two examples of applications or problems where the Articulation Points algorithm would be useful that are not covered by assignment 2 (i.e. identifying critical intersections in a city).

0: Identifying intersections in a city that are critical for emergency services in that they would cut off access to some roads if they intersection were blocked. (NOT ACCEPTABLE ANSWER AS THIS IS A2)
1. Identifying nodes, exchanges, or connection points in a communications network that would disconnect the network if they became disabled.
2. Identifying key pipelines in a gas or similar supply network.

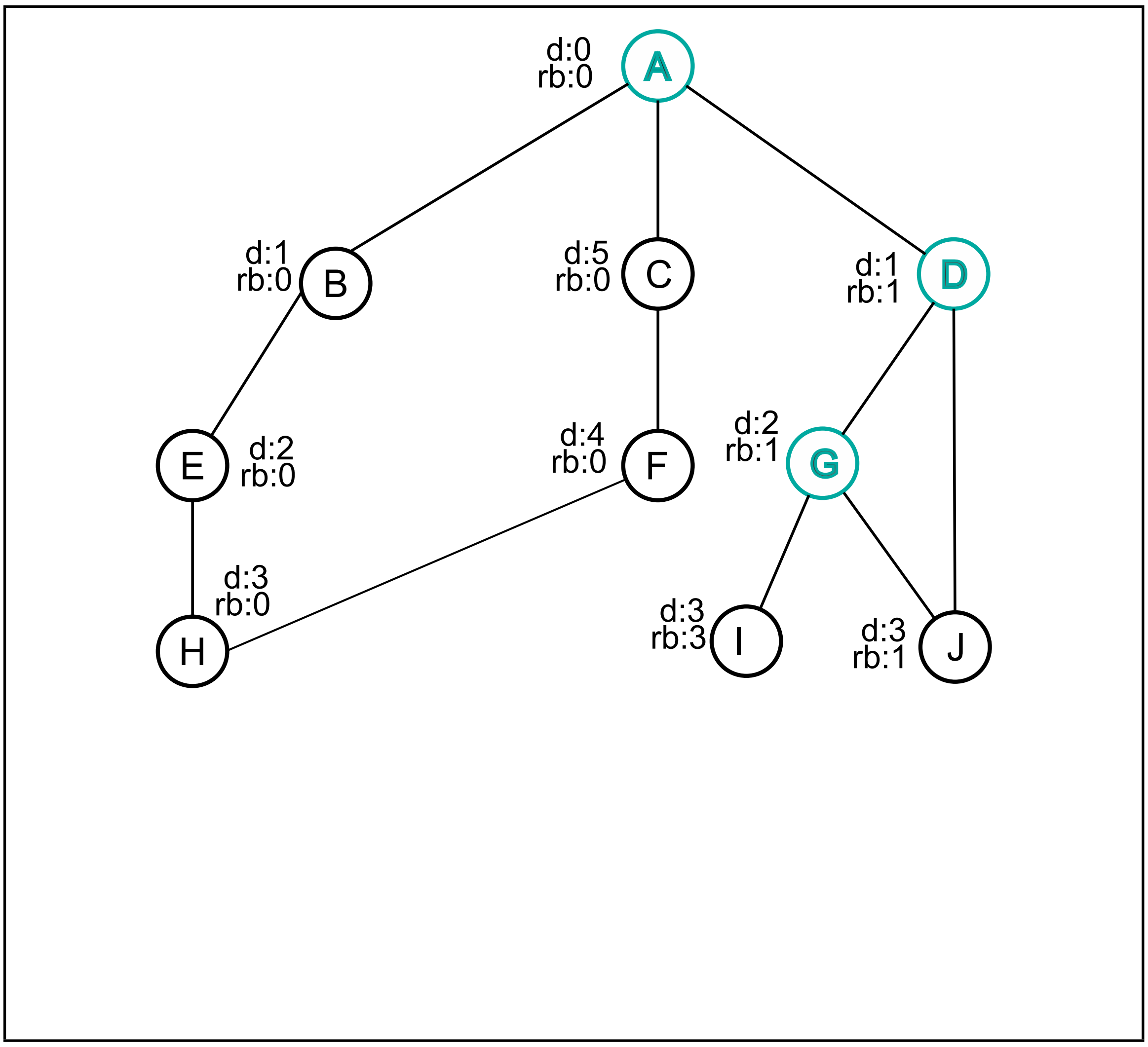
(Question 2 continued on next page)

**(Question 2 continued)**

**(c)** [10 marks] Show how the recursive depth-first-search articulation points algorithm would find the articulation points in the following graph, assuming that the search starts with node A and considers neighbours of nodes in alphabetical order. Mark on the graph

- the depth ("d:") and the reach-back ("rb:") of each node (node A is done for you),
- the return values from each recursive call,
- the nodes that are articulation points, indicating why they were marked.

The algorithm is given on the facing page for your reference.



(Question 2 continued on next page)

**(Question 2 continued)**

```
FindArticulationPoints (graph, start ):
    for each node: node.depth ← ∞, articulationPoints ← { }
    start .depth ← 0, numSubtrees ← 0
    for each neighbour of start
        if neighbour.depth = ∞ then
            RecursiveArtPts( neighbour, 1, start )
            numSubtrees ++
    if numSubtrees > 1 then add start to articulationPoints

RecursiveArtPts(node, depth, fromNode):
    node.depth ← depth, reachBack ← depth,
    for each neighbour of node other than fromNode
        if neighbour.depth < ∞ then
            reachBack ← min(neighbour.depth, reachBack)
        else
            childReach ← RecursiveArtPts(neighbour, depth +1, node)
            reachBack ← min(childReach, reachBack )
            if childReach ≥ depth then add node to articulationPoints
    return reachBack
```

## Question 3. Parsing [20 marks]

**(a)** [10 marks] Consider the following grammar, where terminals are always enclosed in quotation marks and nonterminals are always in capitals. Assume each nonterminal and terminal is separated with a space to form separate tokens for simplicity:

```
FOO  ::= BAR FEND
FEND ::= "start" BAZ | BAZ "end" | "mid" BAZ
BAR  ::= [d-z0-9]+ | BAZ
BAZ  ::= a*b*c+
```

In the following list put a tick next to sentences that belong to the language defined by this grammar and a cross next to sentences that don't:

i. abc start c (TICK)

ii. ccc ccc end (TICK)

iii. ac start aaabbb

iv. dddd09 mid c (TICK)

v. d mid ab

(Question 3 continued on next page)

**(Question 3 continued)**

**(b)** [10 marks] Write a parser that returns `true` or `false` and follows the grammar below. Assume each nonterminal and terminal is separated with a space to form separate tokens for simplicity:

```
QUERY ::= "SELECT" "*" ["FROM" NAME] ["WHERE" NAME "=" DATA] ";"
NAME ::= [A–Za–z]+
DATA ::= [A–za–z0–9 ]*
```

```
public boolean parseQuery(Scanner s) {

if (s.next().equals("SELECT")) {
if (s.next().equals("*")) {
String ss = s.next();
if (ss.equals("FROM") || ss.equals("WHERE") || ss.equals(";") {
if (ss.equals("FROM") && parseName(s)) return true;
if (ss.equals("WHERE") && parseName(s) && s.next().equals("=") && parseData(s))
return true;
if (ss.equals(";") return true;
}
}
}

return false;
}

public boolean parseName(Scanner s) {
if (s.hasNext("[A-Za-z]+")) { s.next(); return true; }
return false;
}

public boolean parseData(Scanner s) {
if (s.hasNext("[A-Za-z0-9 ]*")) { s.next(); return true; }
return false;
}
```

Student ID: . . . . . . . . . . . . . . . . . . . . . .

## Question 4. Tries and String Search [25 marks]

**(a)** [4 marks] Draw the Trie containing the following set of strings.
(Note: label the links of the Trie, not the nodes.)

```
baby      bottle    but       butt      button    body
bait      busy      bus       apple     aps
```

ANSWER CONSIDERED EASY

(Question 4 continued on next page)

**(Question 4 continued)**

**(b)** [6 marks] Give an example of a problem where using a Trie would be a more efficient implementation of a set of strings than a HashSet, and explain why the Trie would be more efficient.

(a) When you need to be able to list all the strings in order (or all the strings between two values, in order). With a trie, you can do a traversal of the trie, starting at the first string. With the HashSet, you would need to extract all the values and sort them.
(b) An autocomplete function that should list all the strings from a dictionary that start with the prefix the user has typed. With a trie, you search down the trie for the prefix, then traverse the subtree under that node listing all the strings. With a Hashset you would need to iterate through every string in the set checking whether hit has the prefix, which would be much slower.
(c) If you need to store a very large number of strings, where there are a lot of shared prefixes (for example, dictionaries or lists of names). The trie may use less memory since it only needs to store any shared prefix once.

(Question 4 continued on next page)

**(Question 4 continued)**

**(c)** [8 marks] Show the table that the following KMP table building algorithm would construct given the string "`fgrkffgfgren`":

```
computeKMPTable(string)
    initialise table to an array of integers, same length as string
    table[0] ← −1
    table[1] ← 0;
    pos ← 2;
    j ← 0;
    while pos < string.length
        if string[pos−1] = string[j]
            table[pos] ← j+1
            pos++
            j++
        else if j > 0
            j ← table[ j ]
        else
            table[pos] ← 0
            pos++
    return table
```

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| string: | f | g | r | k | f | f | g | f | g | r | e | n |
| table: | -1 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 1 | 2 | 3 | 0 |
| index: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

(Question 4 continued on next page)

**(Question 4 continued)**

**(d)** [4 marks] Explain, using an example, how the table built in part (c) would help you find the string in a large body of text? What string would you be searching for?

The string searched for will the one in previous question: fgrkffgfgren
Brute force algorithm will search for the string starting with its first character and trying to match it and when the match failes, it resets to the first character of the string.
With the help of the constructed table, we can skip the characters seen so far and use the table to know whether we found a smaller prefix of the string we are searching for.
The above gets 4 marks, to get 5, give any example including the one on wikipedia for KMP.

**(e)** [3 marks] What improvement over the brute force algorithm would KMP be able to make because of this table?

TO DO

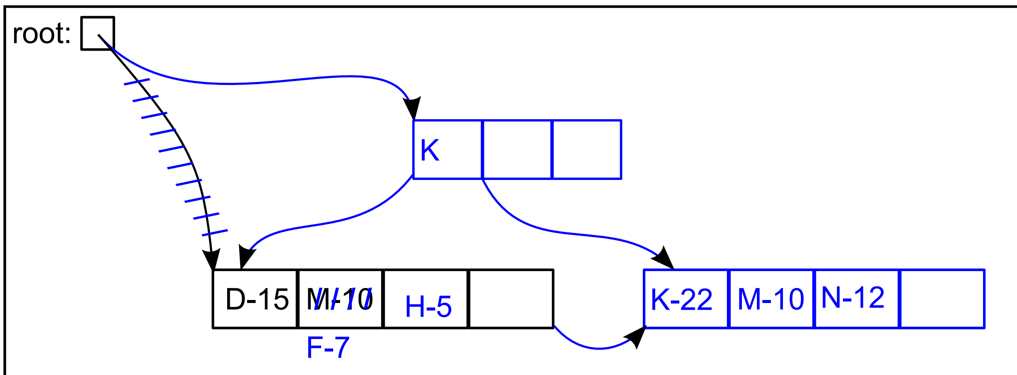
**Question 5. B+ Trees** [15 marks]

The following two subquestions concern a B+ tree that has internal nodes holding up to 3 keys, and leaf nodes holding up to 4 key-value pairs. The keys are letters; the values are numbers.

**(a)** [5 marks] Show how the B+ tree below would be changed if the following key-value pairs were added to it:

```
    H-5  F-7 K-22 N-12
```

State any assumptions you make.



Assumptions (if any): When the leaf is too full (5 k-v pairs), it could be split 2-3 or 3-2; Here, we have split it 3-2.

Alphabet: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

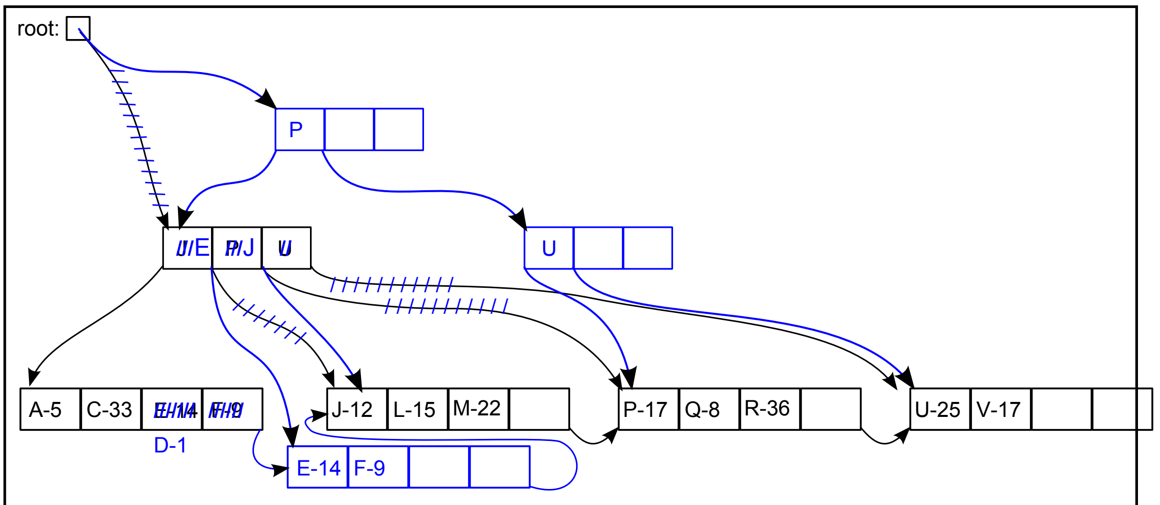
(Question 5 continued on next page)

**(Question 5 continued)**

**(b)** [5 marks] Show how the B+ tree below would be changed if the following key-value pair was added to it:

`D-1`

State any assumptions you make.



Assumptions (if any): When the leaf is too full, it could be split 2-3 or 3-2; Here, we have split it 3-2. When the internal node is too full (4 keys, 5 children), it could be split 1-1-2 (the 2nd key being promoted) or 2-1-1 (the third key being promoted). Here, we have split it 2-1-1.

(Question 5 continued on next page)

**(Question 5 continued)**

**(c)** [5 marks] Suppose the internal nodes of a B+ tree have at most 6 children (*i.e.*, 5 keys), and the leaves contain at most 4 key-value pairs. The maximum number of key-value pairs that can be in a tree is $4 \times 6^h$ where the height of the tree is $h$. What is the minimum number of key-value pairs that can be in tree of height $h$? Assume that $h \geq 1$. Show your working! Note: the height of a tree is the number of edges (= number of internal nodes) on a longest path from the root to a leaf.

Internal nodes have at least 3 children, except the root which may have only 2 children.

| level: | Minimum # leaf nodes |
|---|---|
| 1: | $2$ |
| 2: | $2 \times 3$ |
| 3: | $2 \times 3^2$ |
| $\vdots$ | $\vdots$ |
| h (leaves): | $2 \times 3^{h-1}$ |

Each leaf must have at least $4/2 = 2$ key-value pairs
Therefore:
Minimum number of K-V pairs: $2 \times 2 \times 3^{h-1} = 4 \times 3^{h-1}$

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Student ID: . . . . . . . . . . . . . . . . . . . . . .

## Question 6. Compression [15 marks]

**(a)** [5 marks] Construct a trie that can be used to encode the following sentence using Huffman Encoding. (Note: label the links of the Trie, not the nodes.)

```
WHERE WHERE DID WHY WHAT WHEN GO
```

NO ANSWER PROVIDED

(Question 6 continued on next page)

**(Question 6 continued)**

**(b)** [5 marks] Show the resulting encoding and state how many bits will be required to store the message from part (a):

```
WHERE WHERE DID WHY WHAT WHEN GO
```

NO ANSWER PROVIDED

(Question 6 continued on next page)

**(Question 6 continued)**

**(c)** [5 marks] What is the key idea of the Lempel-Ziv 77 compression algorithm? Please explain using an example.

Eliminate repeating patterns
“ababc” is [0,0, a][0,0,b][2,2,c]

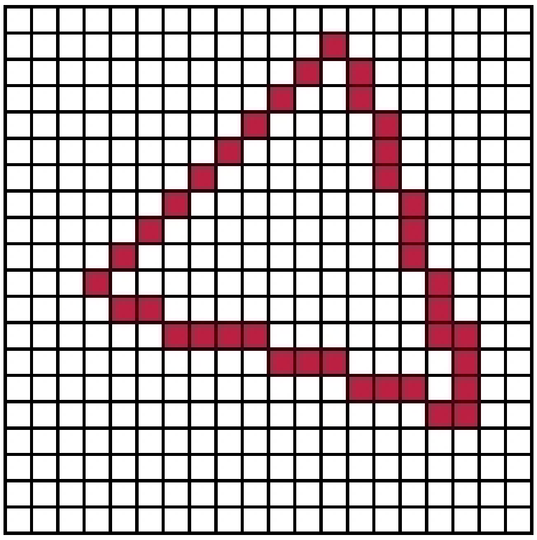
Student ID: ......................

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

## Question 7. Flood fill [20 marks]

**(a)** [10 marks] You are given a black and white image as a matrix. The matrix is filled with boolean values (0 - white, 1 - black) that correspond to a drawing of triangle edges. Outline an algorithm (pseudo code) to fill the triangle with ones.



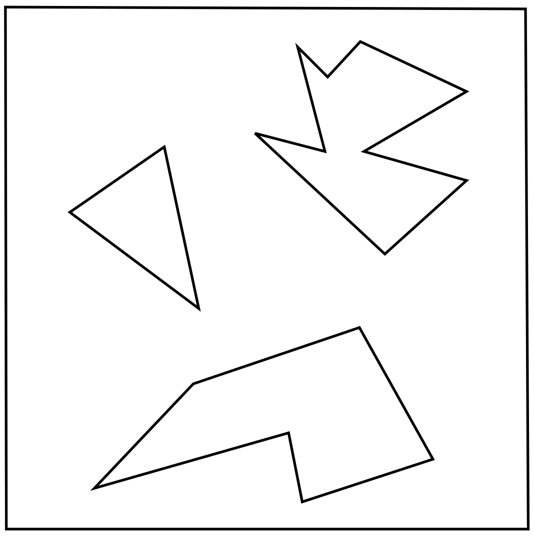
-

(Question 7 continued on next page)

**(Question 7 continued)**

**(b)** [10 marks] You are given an image as before, but now with more than one shape which can be arbitrary polygons. You are given the location (row and column) of a point inside a shape *S*. Outline an algorithm (pseudo code) to fill *S* (and only *S*).

**Hint**: A matrix can be viewed as a graph, where each cell is connected to 4 neighbours.



* * * * * * * * * * * * * * *

Student ID: . . . . . . . . . . . . . . . . . . . . . . .

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.