UNIVERSITY OF SOUTHAMPTON AICE1005W1

SEMESTER 2 EXAMINATION 2024/25

ALGORITHMS AND ANALYSIS

Duration 120 mins (2 hours)

This paper is a WRITE-ON examination paper.

You ***must*** write your Student ID on this Page and must not write your name anywhere on the paper.

All answers should be written within the designated boxes in this examination paper and sufficient space is provided for each question.

If, for some reason, space is required to complete or correct an answer to a question, use the "Additional Space" provided on the facing or adjacent page to the question. Clearly indicate which question the answer corresponds to.

No credit will be given for answers presented elsewhere and without clear indication of to what question they correspond. Blue answer books may be used for scratch; they will be discarded without being looked at.

Answer ALL parts of question 1 in part A (worth 40 marks each) and TWO questions in part B worth 30 marks each.

Student ID:

Question	Mark	*Arithmetic checked*	*Double Marked*
A1	/40		
B2	/30		
B3	/30		
B4	/30		
B5	/30		
Total:	/100		

University approved calculators MAY be used.

A foreign language translation dictionary (paper version) is permitted provided it contains no notes, additions or annotations.

10 page examination paper

Section A

A 1

(a) If a program takes 1s on an input of size $n = 100$, how long will take on an input of size $n = 1000$ if the time complexity is

(i) logarithmic, i.e. $\Theta(\log(n))$

(ii) linear, i.e. $\Theta(n)$

(iii) log-linear, i.e. $\Theta(n \log(n))$

(iv) quadratic, i.e. $\Theta(n^2)$

(v) cubic, i.e. $\Theta(n^3)$

[10 marks]

(i) Logarithmic implies $T(n) = c \log_{10}(n)$ (I'm free to choose the base of the logarithm). Thus $T(100) = 1 = 3c$ or $c = 1/3$.
$T(1000) = \log(1000)/3 = 4/3 \approx 1.33$**.**

(ii) Linear implies $T(n) = c\, n$ so $c = T(100)/100 = 1.0 \times 10^{-3}$ and
$T(1000) = c\, 1000 = 10s$

(iii) Log-linear scaling implies $T(n) = cn \log_{10}(n)$ so $c = 1/3000$ and
$T(1000) = 40/3 \approx 13.3$

(iv) Quadratic scaling implies $T(n) = cn^2$ so $c = 10^{-6}$ and
$T(1000) = 10^{-6} \times 10^8 = 100$

(v) Cubic scaling implies $T(n) = cn^3$ so $c = 10^{-9}$ and
$T(1000) = 10^{-9} \times 10^{12} = 1000$

(b) Give the average time complexity for checking the time it takes to search for an item in a set implemented as a (1) a binary tree, (2) a hash table and (3) a trie. Discuss any assumptions being made and give the worst case complexity when the assumptions fail. [10 marks]

(3 marks for part and 1 discretionary mark for a good answer).

(i) For a binary tree the average time complexity is $\Theta(\log(n))$ assuming the tree is balanced (which is guaranteed for AVL or red-black trees). If the tree is not balanced the the worst case could be $\Theta(n)$ if all the items formed a singled branch (which happens if the entries are add in order).

(ii) For hash sets the average time complexity is $\Theta(1)$ although this take many operations to compute a hash code. This assumes that the hashing sends the entry to random positions in the hash table. The worst case would be when all the entries were hashed to the same position in which case the time complexity is $\Theta(n)$.

(iii) For a trie the average time complexity depends on the length on the record rather than the number of elements. This is lower bounded by $\log_{|\mathcal{A}|}(n)$, where $|\mathcal{A}|$ is the cardinality of the alphabet used in the multiway table. The depth of the tree will typically be $\log_{|\mathcal{A}|}(n)$ although even when you reach a unique branch then you need to check that a string matches. If the strings mostly shared a common long prefix then the number of comparisons before reaching a unique branch may the length of the string.

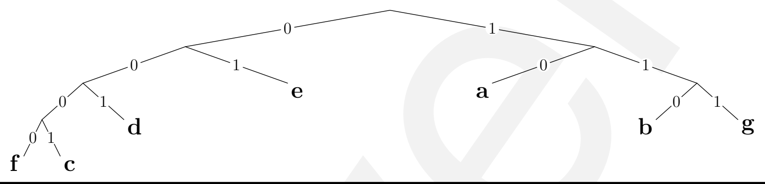
(c) Give the average (amortorised) time complexity for a resizable array (e.g. `vector<T>`) and a doubly linked list (e.g. `list<T>`) for the following operations. [10 marks]

	`vector<T>`	**`list<T>` list**
Adding an element to end of list	$\Theta(1)$	$\Theta(1)$
Adding an element to beginning of list	$\Theta(n)$	$\Theta(1)$
Accessing the i^{th} element	$\Theta(1)$	$\Theta(n)$
Searching for an element	$\Theta(n)$	$\Theta(n)$
Adding two lists of size n	$\Theta(n)$	$\Theta(1)$

(d) Compute a Huffman tree for the following alphabet.

Letter	a	b	c	d	e	f	g
Frequency	120	65	33	54	110	10	83

[7 marks]



(e) How would the word “bagged” be coded in your Huffman tree (show the letter break with a hyphen). [3 marks]

110-10-111-111-01-001

End of question A1

Section B

B 2 The Towers of Hanoi problem is solved by the program

```
hanoi(n, A, B, C)
{
   if (n>0) {
     hanoi(n-1, A, C, B);
     move(A, C);
     hanoi(n-1, B, A, C);
   }
}
```

(a) Let $T(n)$ be the number of times `move` is called to solve the Hanoi problem of size n. Write down a recurrence relation for $T(n)$. [5 marks]

$T(n) = 2\,T(n-1) + 1$

(b) Write down the boundary condition $T(1)$ and use the recurrence relation to compute $T(2)$, $T(3)$, and $T(4)$

[5 marks]

$T(1) = 1$

$T(2) = 2 \times 1 + 1 = 3$

$T(3) = 2 \times 3 + 1 = 7$

$T(4) = 2 \times 7 + 1 = 15$

(c) Prove by induction that, $f(n) = 2^n - 1$ satisfies the recurrence relation in part (a) [20 marks]

Base case: $f(1) = 2^1 - 1 = 1 = T(n)$
Recursive case: Assume $T(n-1) = f(n-1) = 2^{n-1} - 1$

$$\begin{aligned} T(n) &= 2\,T(n-1) + 1 \\ &= 2\left(2^{n-1} - 1\right) + 1 \\ &= 2^n - 1 \end{aligned}$$

End of question B2

B 3

(a) Briefly describe Prim's algorithm. [10 marks]

In Prim's algorithm you start from an arbitrary node and then take the shortest edge from that node to a node you have not visited. You repeat this always adding the shortest edge from the nodes that you have visited to a node you have not visited. This finishes when you have visited every node (when you add $n-1$ edges).

(b) What is the time complexity of Prim's algorithm for constructing a minimum spanning tree.

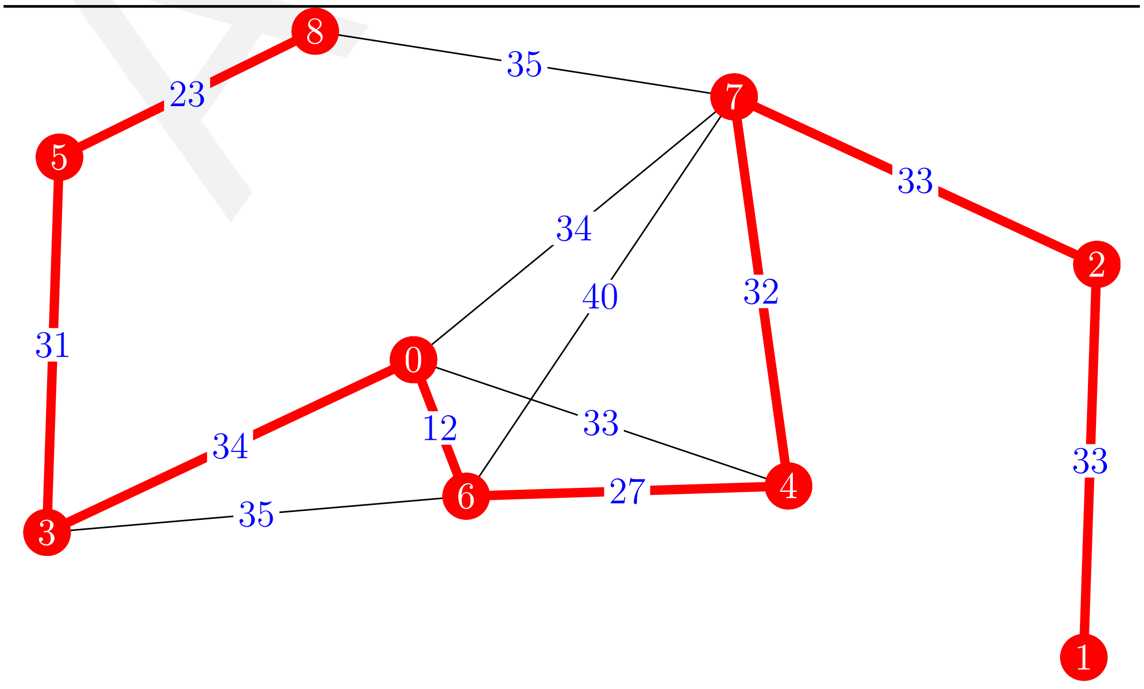
[5 marks]

$$O(|\mathcal{E}|\ \log(|\mathcal{V}|)$$

where $\mathcal{E}$ is the set of edges and $\mathcal{V}$ is the set of vertices (and $|\mathcal{S}|$ denotes the cardinality of set $|\mathcal{S}|$).

(c) In the graph below draw the minimum spanning tree (by highlighting the edges) and write down the order in which the edges would be found by Prim's algorithm.

[15 marks]



1. 0–6	**2.** 6–4
3. 4–7	**4.** 7–2
5. 2–1	**6.** 0–3
7. 3–5	**8.** 5–8

End of question B3

B 4

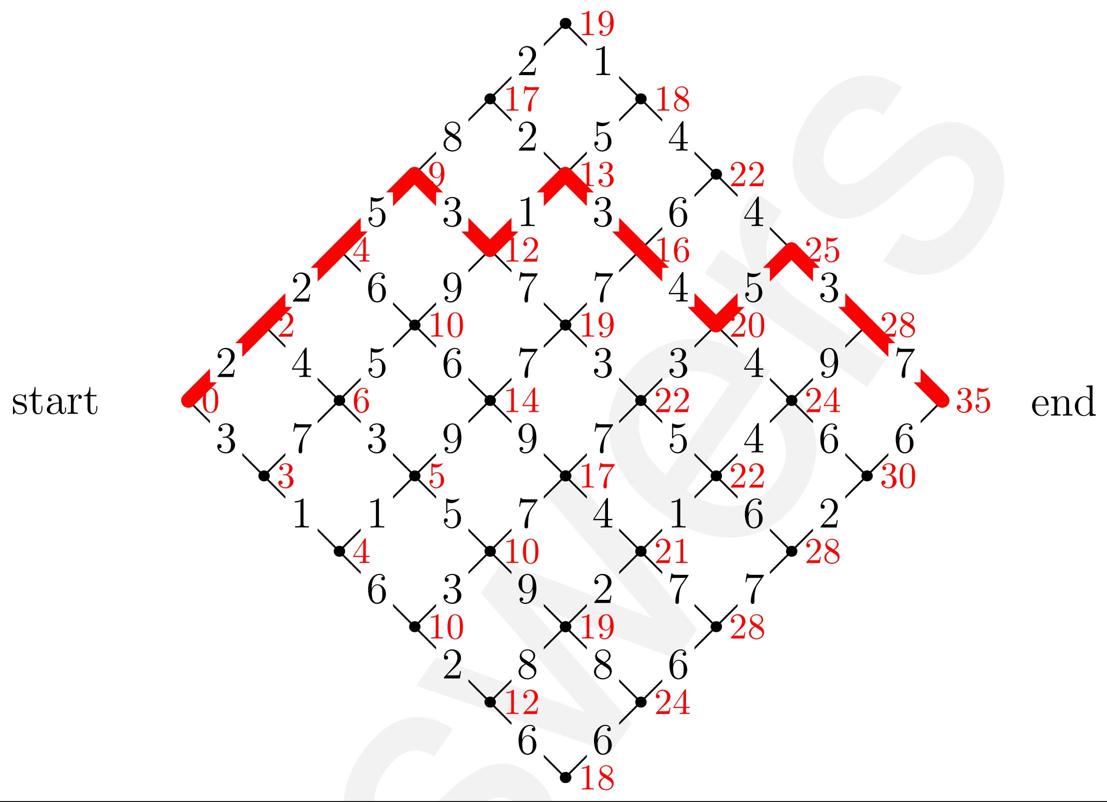
(a) Briefly describe dynamic programming (both forward and backward algorithms). [10 marks]

In dynamic programming a set of optimal costs is built up, starting from a source, until the destination is reached. At each step of the algorithm the optimal costs are computed from those that have already been computed. This provides the cost of an optimal solution from source to destination. To find the solution we run a backward algorithm starting from the destiation and choosing the partial solution at the previous step whose cost would give the optimal cost observed in the current step.

(b) Give an example of where dynamic programming is used. [5 marks]

String matching based on minimising an edit distance.

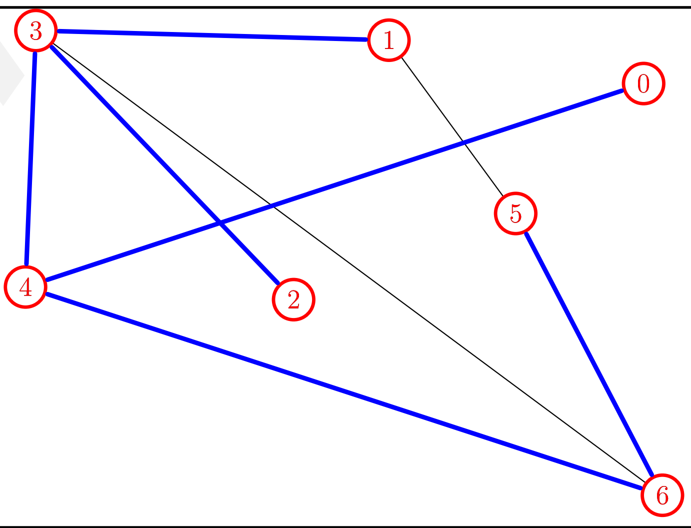
(c) Use the dynamic programming forward algorithm to compute the minimum cost of each path from the left most node to each other node, where the cost of moving along an edge is equal to the number shown. An edge can only be traversed from left to right. Use the backwards algorithm to find the minimum cost path across the graph. [15 marks]



End of question B4

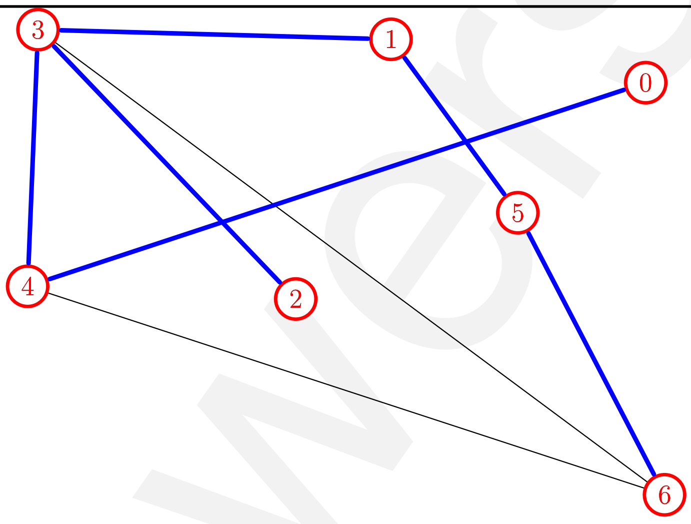
B 5

(a) Draw the edges on the graph used to find the vertices using **breadth first search** starting from vertex 0 where the lower numbered vertices are searched first. Write the order in which the vertices are discovered. [10 marks]



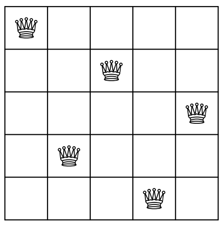
bfs order = 0, 4, 3, 6, 1, 2, 5

(b) Draw the edges on the graph used to find the vertices using **depth first search** starting from vertex 0 where the lower numbered vertices are searched first. Write the order in which the vertices are discovered. [10 marks]



dfs order = 0, 4, 3, 1, 5, 6, 2

(c) The n-queens problem is to put n queens on a chess board such that no queen is in the same row, column or diagonal as any other queen. Either write pseudo code or describe in outline an algorithm to solve the n-queens problem.



[10 marks]

```
List nextRow(row, positionList) {
  if (row==n) {
     return positionList
  }
  for col = 1 to n {
    if legalQueen(col, row, positionList) {
       positionList.add(col)
       solution = nextRow(row+1, positionList)
       if (solution ≠ null) {
          return solution
       }
    }
  }
}
```

where `legalQueen(col, row, positionList)` is true if placing a queen in column `col` of row `row` does not interfere with any of the queens in the earlier rows given by `positionList`.

End of question B5

END OF PAPER