1(a). A salesman travels around the country, stopping at specific places, and then returning to the starting place.

Fig 6.1 shows an example map of places that the salesman visits.

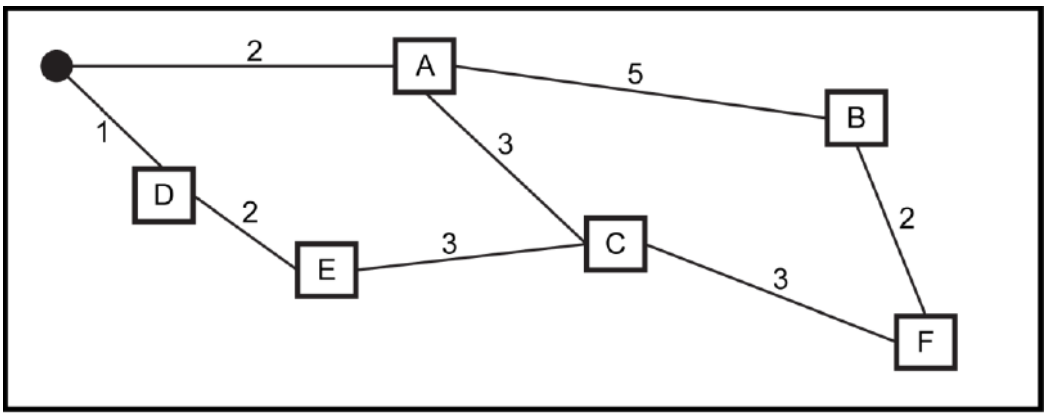


Fig 6.1

The filled in circle represents the start and end point. The letters represent the places to visit. The lines are the routes available and the numbers are the length of time each route takes to travel.

Explain how abstraction has been applied in the production of Fig 6.1

--- [2]

(b). The travelling salesman aims to find the shortest route between these places to visit.

A programmer is writing an algorithm to solve the travelling salesman problem.

The programmer is using a tree to find the most efficient route. Fig 6.2 shows part of the tree with three levels completed.

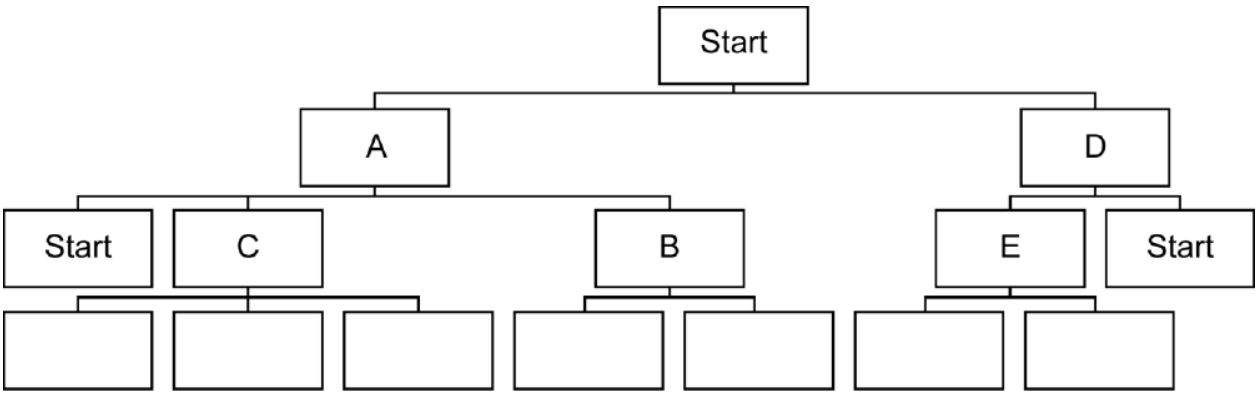


Fig 6.2

(i) The 'Start' nodes on level three are not expanded again as this is a repeat, 'Start' has already been expanded.

Write the place names in the boxes in Fig 6.2, to complete the fourth level of the tree structure for the map shown in Fig 6.1.

[3]

(ii) Explain why the tree in Fig 6.2 is **not** a binary tree.

---

[1]

(c). The programmer has decided to use a graph instead of a tree structure.

(i) Describe what is meant by a graph structure.

---

[2]

(ii) The pseudocode below shows part of an algorithm which uses a queue to traverse the graph breadth-first. Complete the missing elements of the algorithm.

```
markAllVertices (notVisited)
createQueue()
start = ..............
markAsVisited(..............)
pushIntoQueue(start)
while QueueIsEmpty() == ..............
    currentNode = removeFromQueue()
    while allNodesVisited() == false
        markAsVisited(..............)
        //following sub-routine pushes all nodes connected to
        //currentNode AND that are unvisited
        pushUnvisitedAdjacents()
    endwhile
endwhile
```

[4]

(d). Fig 6.3 is a graph representation of the places that the travelling salesman visits. Using this graph, show how Dijkstra's algorithm would find the shortest path from place A to place F.

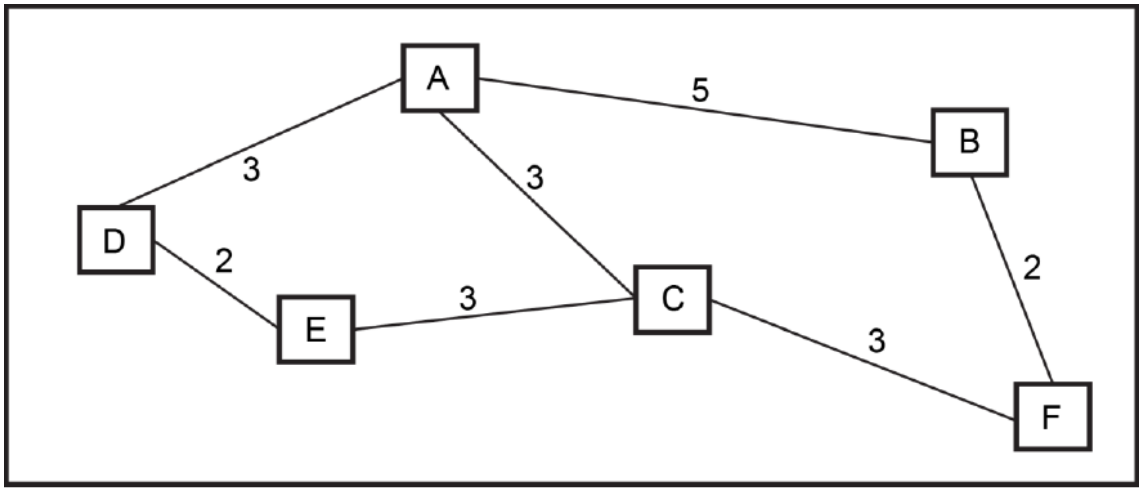


Fig 6.3

[6]

2(a). Fig. 2.1 shows the flight paths between a country's airports. The value in bold beneath each node is the heuristic value from E.

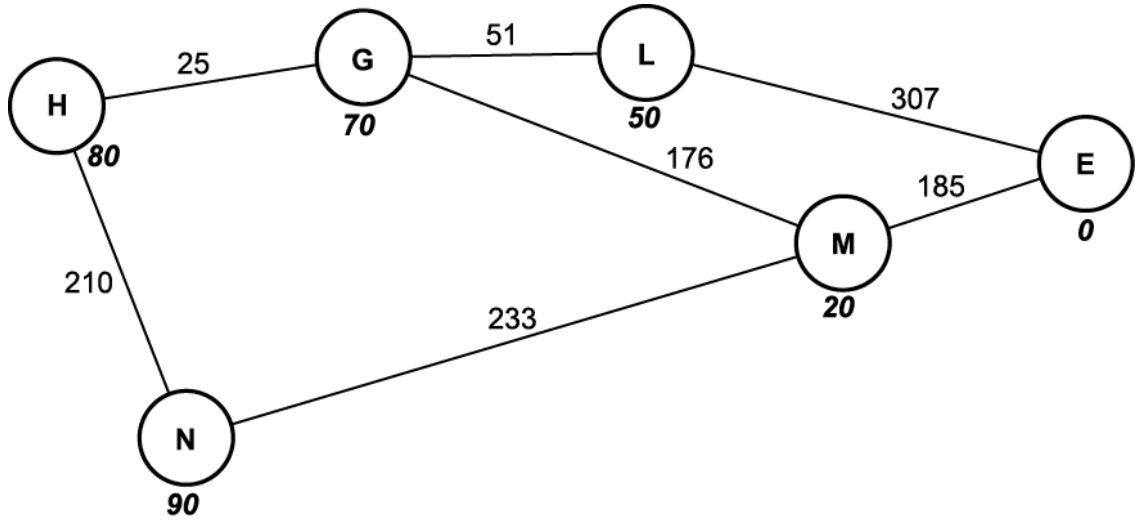


Fig. 2.1

State the full name of the data structure shown in Fig. 2.1.

---

[2]

(b). The structure in Fig. 2.1 is searched using the A* algorithm making use of the heuristic values.

(i) State what the heuristic values could represent in Fig. 2.1.

---

[1]

(ii) State the purpose of heuristic values in the A* algorithm.

---

[1]

(iii) Perform an A* algorithm on the data structure in Fig. 2.1 to find the shortest distance between H and E. Show each step of the process, and the calculations performed for each node visited.

---

[8]

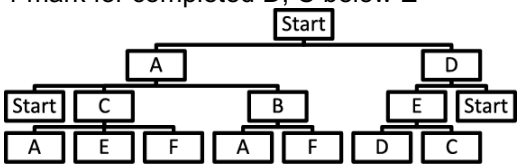
(iv) Give **one** decision that is made in the A* algorithm, and describe the effect of this decision on the next step(s) of the algorithm.

Decision _____

Effect _____

[3]

END OF QUESTION PAPER

Question			Answer/Indicative content	Marks	Guidance
1	a		1 mark per bullet to max 2 e.g. • Places have been replaced with variables … (1) • … e.g. a place has been replaced with A (1) • Irrelevant information has been removed … (1) • … e.g. only the routes and places are shown (1) • Time is given as a numeric value (1) … • … e.g. 1 rather than 1 hour, or 1 minute (1) • Relative geographic location may not be accurate (1) • … e.g. positions of the towns may not be proportional to actual distance (1)	2	
	b	i	1 mark for completing A, E, F below C 1 mark for completing A, F below B 1 mark for completed D, C below E 	3	
		ii	In a binary tree a node can only have two children	1	
	c	i	1 mark per bullet to max 2 • Collection of data nodes / vertices (1) • Connections / edges are set between nodes / vertices (1) • Graph (edges) can be directional or bi-directional (1) • Graphs (edges) can be directed or undirected (1)	2	

Question			Answer/Indicative content	Marks	Guidance

		ii	1 mark each
			
			markAllVertices (notVisited)
			createQueue()
			start = **currentNode** (1)
			markAsVisited(**start**) (1)
			pushIntoQueue(start)
			while QueueIsEmpty() == **false** **(1)**
			popFromQueue(currentNode)
			while allNodesVisited() == false
			markAsVisited(**currentNode**) (1)
			//following sub-routine pushes all nodes connected to
			//currentNode AND that are unvisited
			pushUnvisitedAdjacents()
			endwhile
			endwhile
				4	

	d		Max 6. 1 mark for final solution, max 5 for showing the stages • Mark A as the current node (1) • Record B is 5, C is 3, D is 3 (1) • Mark A as visited (1) • C is shortest distance from A (1) • (C as current) Record E as 6, F as 6 (1) • Mark C as visited (1) • (D as current) Record E as 5 (1) • Mark D as visited (1) • (B as current) Record F as 7, do not update table as longer (1) • Mark B as visited (1) • (E as current) Record D as 8, do not update table as longer and E as visited (1) • A-C-F found as shortest (1)	6	
			Total	**18**	

<table>
<tr><th colspan="3">Question</th><th>Answer/Indicative content</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>2</td><td>a</td><td></td><td>1 mark per bullet

• Weighted/Undirected
• Graph</td><td>2</td><td></td></tr>
<tr><td></td><td>b</td><td>i</td><td>E.g. Weighting/cost based on estimated distance from final node</td><td>1</td><td></td></tr>
<tr><td></td><td></td><td>ii</td><td>Used to speed up process of finding solution</td><td>1</td><td></td></tr>
<tr><td></td><td></td><td>iii</td><td>1 mark per bullet, max 7 for calculations/explanation, max 1 for correct final path

• Visiting H with correct heuristic
• Visiting G and N from H
◦ Calculating correct distance+heuristic for G and N
• Identifying G as the smallest value
• Visiting L and M from G
◦ Calculating distance+heuristic for L and M
• Identifying L as the smallest value
• Visiting E
◦ Calculating distance+heuristic for E
• Final path: H-G-L-E

e.g.
<table><tr><th>Node</th><th>Distance travelled</th><th>Heuristic</th><th>distance+heuristic</th><th>previous node</th></tr><tr><td>H</td><td>0</td><td>80</td><td>80</td><td>-</td></tr><tr><td>G</td><td>25</td><td>70</td><td>95</td><td>H</td></tr><tr><td>N</td><td>210</td><td>90</td><td>300</td><td>H</td></tr><tr><td>L</td><td>51+25=76</td><td>50</td><td>126</td><td>G</td></tr><tr><td>M</td><td>176+25=201
~~233+210=443~~</td><td>20</td><td>221
~~463~~</td><td>G
~~N~~</td></tr><tr><td>E</td><td>307+76=383</td><td>0</td><td>383</td><td>L</td></tr></table></td><td>8</td><td></td></tr>
</table>

Question			Answer/Indicative content	Marks	Guidance
		iv	1 mark for decision, 2 marks for effect e.g. Decision: • Choosing which node to take next • The shortest distance+heuristic is taken Effect: • All adjoining nodes from this new node are taken • Other nodes are compared again in future checks • Assumed that this node is a shorter distance • Adjoining nodes may not be shortest path … • ….may need to backtrack to previous nodes	3	
			Total	15	