UNIVERSITY OF SOUTHAMPTON COMP1201W1

SEMESTER 2 EXAMINATION 2012/2013

ALGORITHMICS

Duration: 120 mins

You must enter your Student ID and your ISS login ID (as a cross-check) on this page. You must not write your name anywhere on the paper.

Student ID:

ISS ID:

Question	Marks
1	
2	
3	
4	
Total	

Answer the question in section A and TWO *questions out of* THREE *in section B.*

This examination is worth 75%. The tutorials were worth 25%.

University approved calculators MAY be used.

Each answer must be completely contained within the box under the corresponding question. No credit will be given for answers presented elsewhere.

You are advised to write using a soft pencil so that you may readily correct mistakes with an eraser.

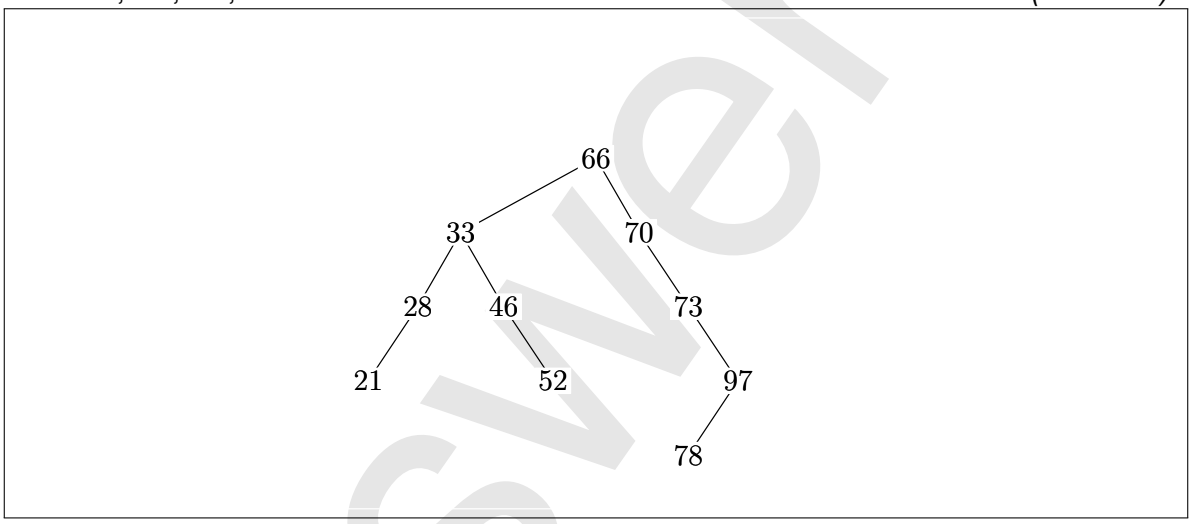
You may use a blue book for scratch—it will be discarded without being looked at.

 Number of pages 14

Section A

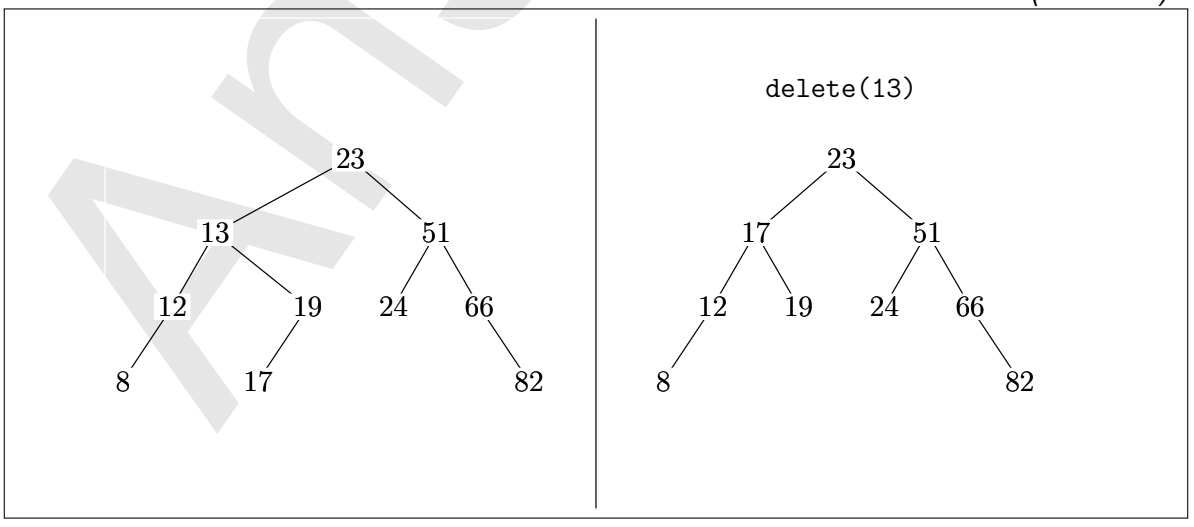
Question A 1

(a) Draw the binary search tree produced when inserting 66, 33, 70, 28, 46, 21, 73, 97, 52, 78. *(2 marks)*



2

(b) Draw the tree obtained by deleting 13 from the binary search tree shown. *(2 marks)*



2

(c) What is the expected time complexity of adding a new entry to a binary tree? *(1 marks)*

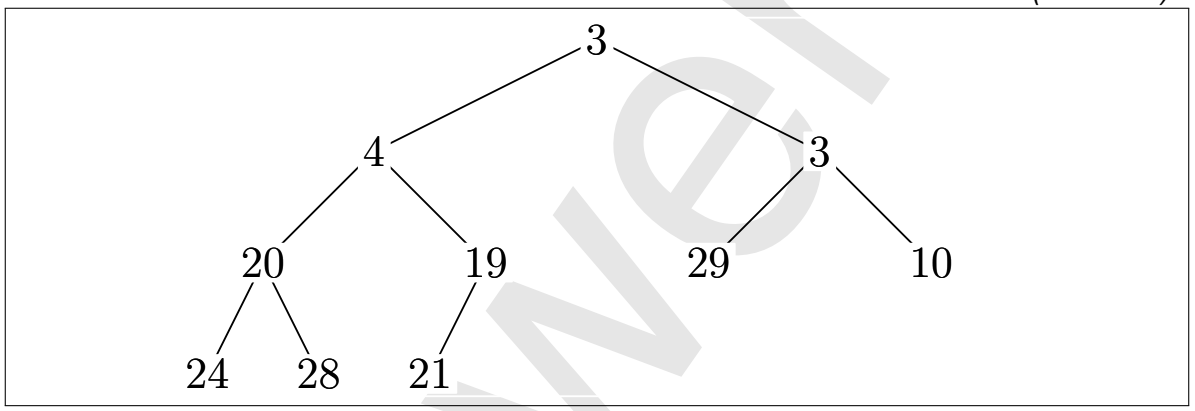
$\Theta(\log(n))$

1

(d) Heaps use a binary tree encoded into an array. Show the binary tree represented by the following array.

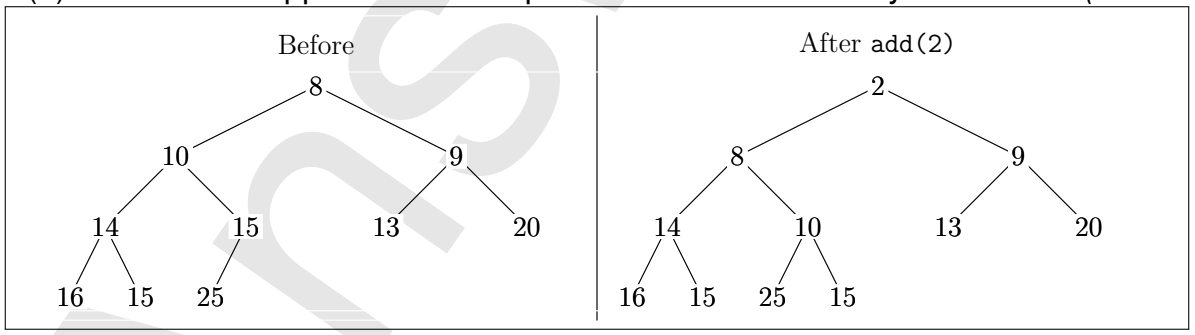
3	4	3	20	19	29	10	24	28	21

(1 marks)



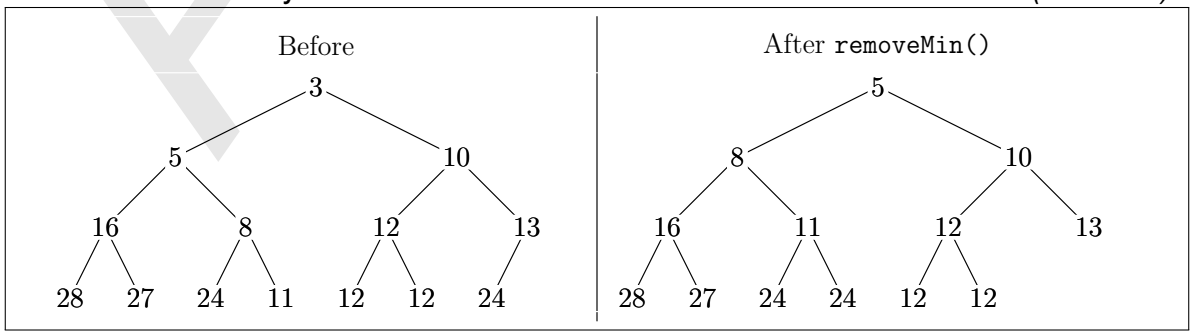
1

(e) Show what happens to the heap shown on the left when you add 2. *(2 marks)*



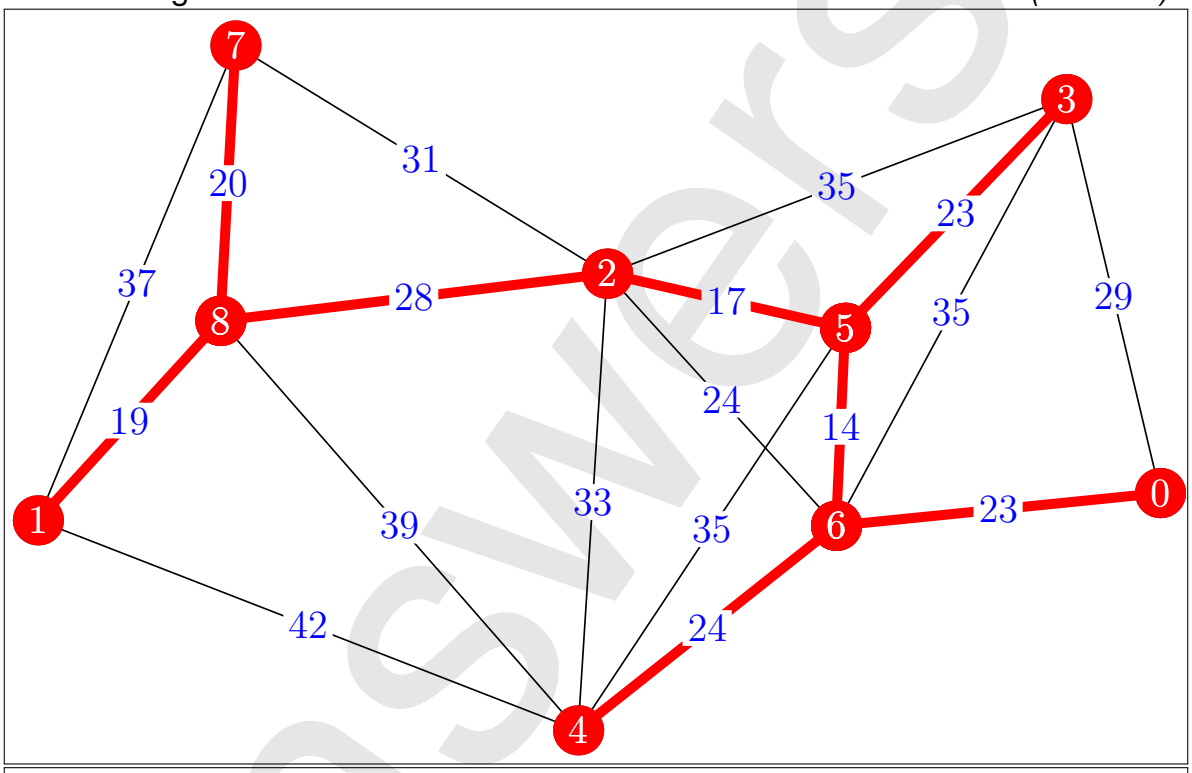
2

(f) Show what happens to the heap shown on the left when you remove the minimum entry. *(2 marks)*



2

(g) Highlight the edges of the minimum spanning tree found by Prim's algorithm for the graph below and write down the edges in the order they are found starting from node 0. *(5 marks)*

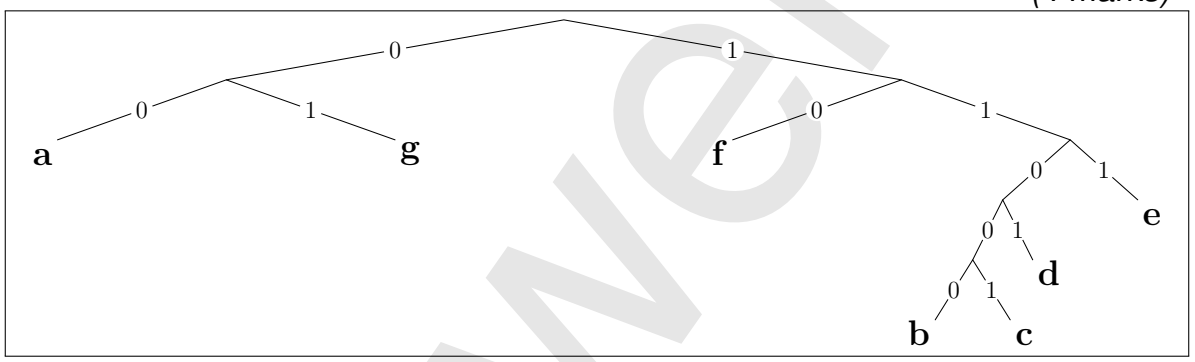


1. 0–6	**2.** 6–5
3. 5–2	**4.** 5–3
5. 6–4	**6.** 2–8
7. 8–1	**8.** 8–7

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

Letter	a	b	c	d	e	f	g
Frequency	13	1	3	5	11	17	15

(4 marks)



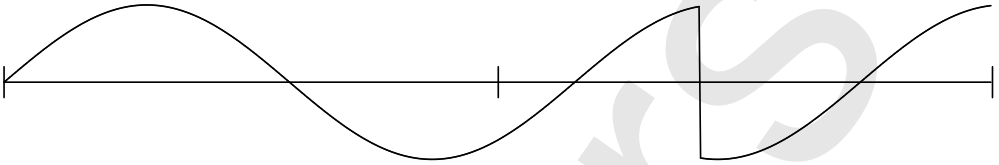
4

(i) How would the word “deface” be coded in your Huffman tree. *(1 marks)*

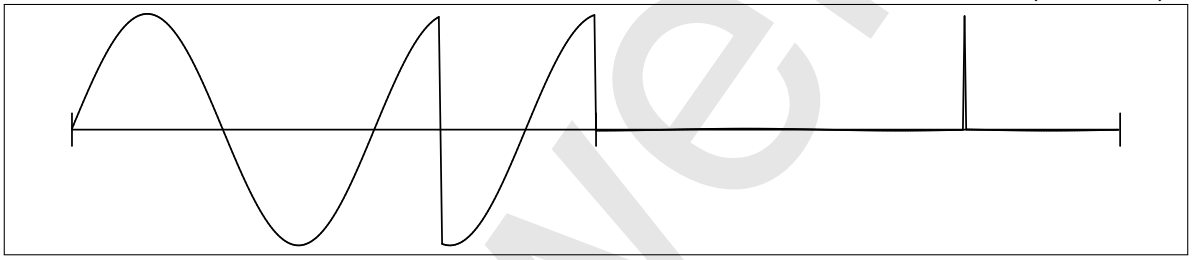
1101-111-10-00-11001-111

1

(j) Sketch the first Haar transform of the signal shown below.



(5 marks)



5

End of question 1

Q1: (a) $\frac{}{2}$ (b) $\frac{}{2}$ (c) $\frac{}{1}$ (d) $\frac{}{1}$ (e) $\frac{}{2}$ (f) $\frac{}{2}$ (g) $\frac{}{5}$ (h) $\frac{}{4}$ (i) $\frac{}{1}$ (j) $\frac{}{5}$ Total $\frac{}{25}$

Section B

Question B 2

(a) We can implement a fast set for a fixed number of integers using two arrays. Below we show the representation of the set $\{7, 3, 1, 4\}$.

	0	1	2	3	4	5	6	7	8	9
indexArray	-1	2	-1	1	3	-1	-1	0	-1	-1
memberArray	7	3	1	4						

Show the state of the arrays when we add 9 to the set. *(3 marks)*

	0	1	2	3	4	5	6	7	8	9
indexArray	-1	2	-1	1	3	-1	-1	0	-1	4
memberArray	7	3	1	4	9					

3

(b) Show the state of the arrays when you remove 3 from the set shown above. *(3 marks)*

	0	1	2	3	4	5	6	7	8	9
indexArray	-1	2	-1	-1	1	-1	-1	0	-1	-1
memberArray	7	4	1							

3

(c) Fill in the implementation of `add` and `remove`. *(8 marks)*

```
public class FastSet extends AbstractSet<Integer> {
    private int[] indexArray;
    private int[] memberArray;
    private int noMembers;

    public FastSet(int n) {
      indexArray = new int[n];
      memberArray = new int[n];
      for(int i=0; i<n; i++)
         indexArray [i] = -1;
      noMembers = 0;
    }

    public boolean add(int i) {
      if (indexArray[i]>-1)
         return false;
      indexArray[i] = noMembers;
      memberArray[noMembers] = i;
      ++noMembers;
      return true;
    }

    public boolean remove(int i) {
      if (indexArray[i]==-1)
         return false;
      --noMembers;
      memberArray[indexArray[i]] = memberArray[noMembers];
      indexArray[memberArray[noMembers]] = indexArray[i];
      indexArray[i] = -1;
      return true;
    }
```

8

(d) Implement an iterator class for the FastSet. *(11 marks)*

```
public Iterator<Integer> iterator() {
        return new FastSetIterator();
}

private class FastSetIterator implements Iterator<Integer> {
        int current = 0;

        public boolean hasNext() {
                return current < noMembers;
        }

        public Integer next() throws NoSuchElementException {
                if (current>=noMembers)
                        throw new NoSuchElementException();
                current++;
                return memberArray[current-1];
        }

        public void remove() throws IllegalStateException {
                if (current==0)
                        throw new IllegalStateException();
                indexArray[memberArray[current-1]] = -1;
                noMembers--;
                memberArray[current-1] = memberArray[noMembers];
                indexArray[memberArray[noMembers]] = current-1;
        }
}
```

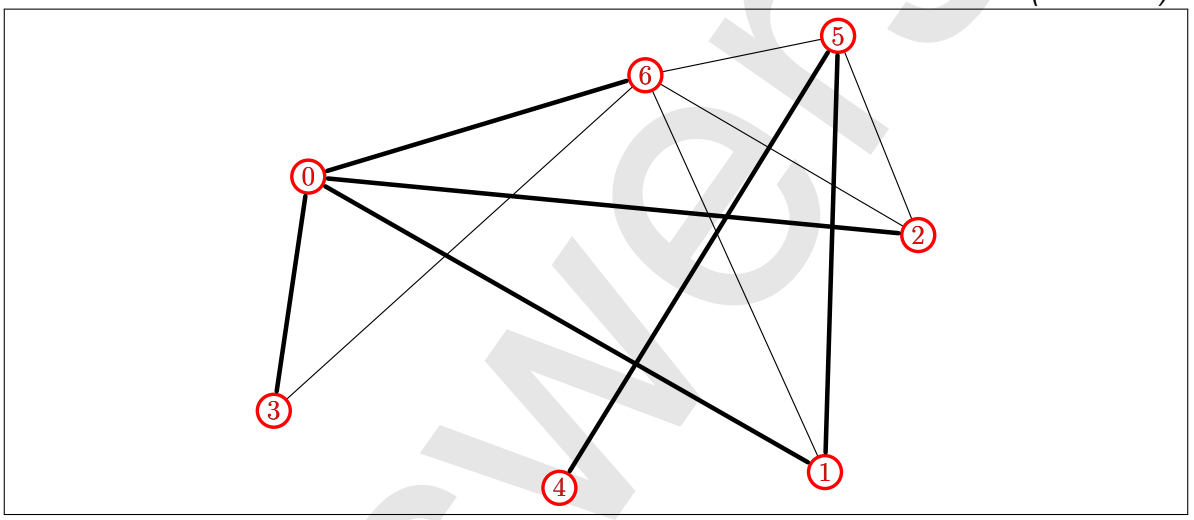
End of question 2 $\frac{}{11}$

Q2: (a) $\frac{}{3}$ (b) $\frac{}{3}$ (c) $\frac{}{8}$ (d) $\frac{}{11}$ Total $\frac{}{25}$

TURN OVER

Question B 3

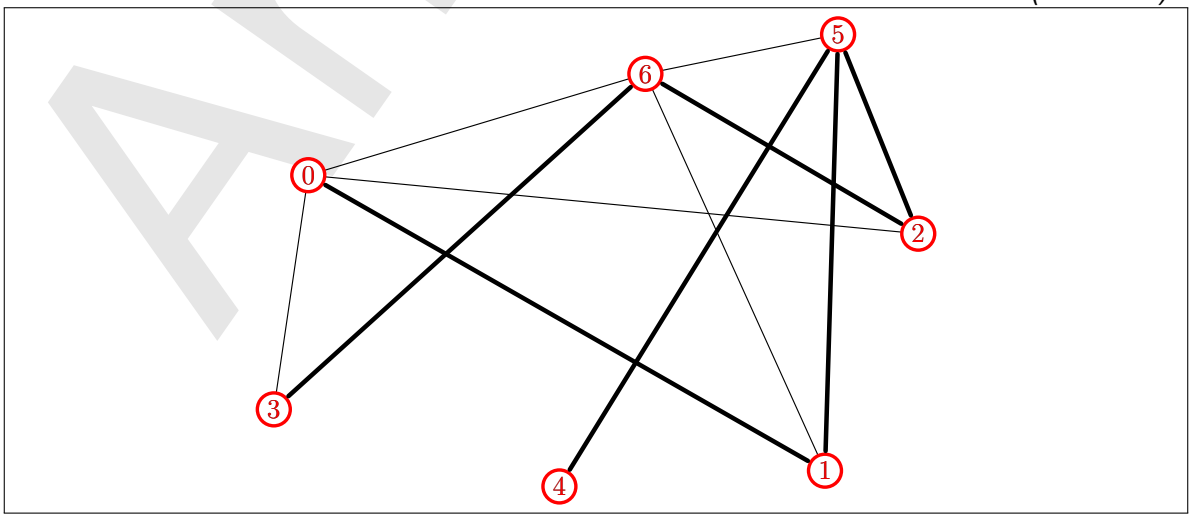
(a) Draw the edges on the graph used to find the vertices using **breadth first search** starting from vertex 0 where the lower numbered vertices are searched first. Write the order in which the vertices are discovered. *(5 marks)*



5

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

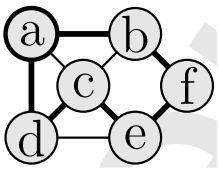
(b) Draw the edges on the graph used to find the vertices using **depth first search** starting from vertex 0 where the lower numbered vertices are searched first. Write the order in which the vertices are discovered. *(5 marks)*



5

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

(c) Write pseudo code for a recursive backtracking algorithm to solve a Hamiltonian circuit problem (see example below). Denote the graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the vertex set and $\mathcal{E}$ is the edge set. Let, $\mathcal{H}$ be the Hamiltonian cycle and `startVertex` be the starting vertex.



(8 marks)

Any approximation of the code given below is sufficient (the stopping condition is a bit crude, but I don't care about these subtleties)

```
nextMove(currentVertex, H) {
  if (|H| = |V|) {
    if ((currentVertex, startVertex) ∈ E)
      return H+startVertex;
  }
  for neighbour ∈ G.neighbourhood(currentVertex)
    if (neighbour ∉ H) {
      nextMove(neighbour, H+startVertex);
    }
  }
}
```

8

• Do not write in this space •

(d) Describe the branch and bound strategy. *(3 marks)*

Branch and bound is a backtracking strategy to find optimal solutions to combinatorial optimisation problems. It works by building partial solutions to explore the whole search space. However, it ceases to expand the partial solution when it exceeds the bound, which is the best cost found so far.

(e) Describe fours ways in which branch and bound can be modified to reduce the proportion of search space needed to find an optimal solution of a Euclidean TSP. *(4 marks)*

(i) Using the tour reversal symmetry we can insist on visiting city 1 before city 2.

(ii) Obtain a good initial bound by using a heuristic search algorithm

(iii) Use the fact that no tours cross in an optimal tour

(iv) Use the minimum spanning tree to get a lower bound on tour through the unseen cities (i.e. those not in the partial solution)

End of question 3

Q3: (a) $\frac{}{5}$ (b) $\frac{}{5}$ (c) $\frac{}{8}$ (d) $\frac{}{3}$ (e) $\frac{}{4}$ Total $\frac{}{25}$

Question B 4 Merge sort has the form

```
MERGESORT(a[1:n]) {
  if (n>1) {
    b ←a[1:n/2]
    c ←a[n/2+1:n]
    MERGESORT(b)
    MERGESORT(c)
    MERGE(b, c, a)
    }
  }
```

The number of comparison operations to merge two arrays of length $n/2$ is n.

(a) Let $T(n)$ be the number of comparison operations. Write down a recurrence relation for $T(n)$ valid if $n = 2^m$. *(4 marks)*

$$T(n) = 2\,T(n/2) + n$$

4

(b) Write down the boundary condition $T(1)$ and use the recurrence relation to compute $T(2)$, $T(4)$, and $T(8)$ *(4 marks)*

$$T(1) = 0$$

$$T(2) = 2 \times 0 + 2 = 2$$

$$T(4) = 2 \times 2 + 4 = 8$$

$$T(8) = 2 \times 8 + 8 = 24$$

4

(c) Demonstrate, for $n = 2^m$, that $f(n) = n \log_2(n)$ satisfies the recurrence relation in part (a). *(6 marks)*

$$\begin{aligned} T(n) &= 2\,T(n/2) + n \\ &= 2\,\frac{n}{2}\log_2\left(\frac{n}{2}\right) + n \\ &= n\left(\log_2(n) - \log_2(2)\right) + n \\ &= n\left(\log_2(n) - 1\right) + n = n\,\log_2(n) \end{aligned}$$

6

(d) Explain why the java collections class uses Merge short, while for arrays of primitive types java uses Quick sort. *(4 marks)*

Merge sort is stable while quick sort is not. The java collection class works on object that may be presorted on another field. There is no advantage to using a stable sort on primitive types as there is only a single field to sort on. As quick sort is faster java uses this for primitive types.

(e) Write pseudo code for quick sort, explaining briefly what each method does. *(7 marks)*

$\frac{}{4}$

```
QuickSort(a, left, right) {
  if (right-left < threshold)
    InsertionSort(a, left, right)
  else
    pivot = ChoosePivot(a, left, right)
    part = Partition(a, pivot, left, right)
    QuickSort(a, left, part)
    QuickSort(a, part+1, right)
  endif
}
```

- **ChoosePivot choose an element in the list to pivot around (e.g. median of first middle and last element)**
- **Partition sorts the elements so the left elements are less than the pivot and the right elements are greater than the pivot. It returns the position of the pivot**

$\frac{}{7}$

End of question 4

Q4: (a) $\frac{}{4}$ (b) $\frac{}{4}$ (c) $\frac{}{6}$ (d) $\frac{}{4}$ (e) $\frac{}{7}$ Total $\frac{}{25}$

END OF PAPER