
**University of Limerick**
**Ollscoil Luimnigh**

College *of* Informatics *and* Electronics

Department of Computer Science
and Information Systems

## Final Assessment Paper

| | | | |
|---|---|---|---|
| Academic Year: | 2006/2007 | Semester: | Spring |
| Module Title: | Data Structures and Algorithms | Module Code: | CS4115 |
| Duration of Exam: | $2\frac{1}{2}$ hours | Percent of Semester Marks: | 65 |
| Lecturer: | P. Healy | Paper marked out of: | 100 |

**Instructions to Candidates:**

- There are two sections to the paper: Short Questions and Long Questions
- The mark distribution is 20 marks for Short Questions and 80 marks for the Long Questions
- Answer all questions in all sections

### Section 1. Short Questions ($5 \times 4$ marks).

- Please put your answers to these questions in the answer book provided to you, labelling your answers 1.1, 1.2, etc.

1. Two snowballs rolling down different part of the same hill grow at different rates. Snowball "Flakey" grows at a rate so that it triples in size every three hours, while snowball "Fluffy" quadruples in size every four hours. If they both started rolling at the same time and they were identical sizes at the outset and they reach the bottom together, which will be biggest. (Include the possibility that not enough information has been given to answer the question.) _____

2. Two galaxies in the same universe expand at different rates. Galaxy "Deuteronium" grows at a rate so that it doubles in size every two years, while galaxy "Séamus" quadruples in size every four years. If they both started expanding at the same time and they were identical sizes at the outset which will be bigger at time $T = 400,000$ years? (Include the possibility that not enough information has been given to answer the question.) _____.

3. A possible algorithm for solving an $n$-piece jigsaw puzzle is:

   (a) randomly pick an unplaced piece, $p$, "from the box" and place it "on the board"

   (b) let $C$, the set of pieces in the completed part of the puzzle, start off as $\{p\}$

i. for each of the remaining pieces in the box, $r$, in turn, see if it fits into the completed part $C$ by checking it against all exposed pieces of $C$
ii. if it does, place it in its position
iii. if it doesn't, return it and consider the next unplaced piece from the box

(c) repeat steps (i) to (iii) while there is something in the box

The running time of the algorithm in "Big-Oh" notation is _____.

4. Consider the following different algorithm for solving jigsaw puzzles with $n$ pieces:

(a) pick a random piece $p$ from the box
(b) let $C$, the set of pieces in the completed part of the puzzle start off as $\{p\}$
i. nominate an exposed piece of the completed part of the jigsaw; call it $c$
ii. for each of the remaining pieces in the box, $r$, in turn, see if it fits next to $c$
iii. if it does, place it in its position
iv. if it doesn't, return it and consider the next unplaced piece from the box

(c) repeat steps (i) to (iv) while there is something still left in the box

The running time of *this* algorithm in "Big-Oh" notation is _____.

5. Computing

$$S = \sum_{i=0}^{n} i^i$$

can be done in running time, in Big-Oh notation, _____. Note this question is not about the *value* of $S$; rather, it is about the time taken to compute $S$. You can assume that multiplication of two numbers is a basic operation and can be done in one step.

## Section 2. Long Questions (80 marks).

- Please put your answers to these questions in the answer book provided to you
- Label your answers 2.1, 2.2, 2.3, and 2.4 in your answer books

1. Answer parts (a) and (b) below. **(20 marks.)**

(a) Use induction to show that $n! > 2^n$, for $n > 3$. (10 marks.)

(b) What does the function `what()` below return? Give a clear explanation referring to a worked example. Hint: recall the tutorial problem of decomposing an integer into its binary representation. (10 marks.)

```
int what(int a, int b)
{
  int p = 0;
  while (a != 0) {
    if (a%2 == 1) p += b;
    a = a / 2;
    b = b * 2;
  }
  return p;
}
```

2. **(20 marks.)**

(a) Draw the height-4 AVL tree that has the minimum number of nodes. That is, draw the worst, most skewed tree that has depth, $d = 4$. (5 marks.)

(b) Give the recurrence relation for $n_d$ the smallest number of nodes possible in an AVL tree of depth $d$. What are the initial conditions? (5 marks.)

(c) In arguing that any sorting algorithm that relies on 2-way comparisons requires $\Omega(n \log n)$ comparisons we claimed that a binary tree with $L$ leaves had to have depth at least $\lceil \log L \rceil$.

Use induction to show that the number of leaves, $|L_d|$, of a binary tree of depth $d$ is at most $|L_d| \leq 2^d$. (10 marks.)

3. **(20 marks.)**

Given the input { 4371, 1323, 6173, 4199, 4344, 9679, 1989 } and a hash function $h(x) = x \bmod 10$

With $hsize = 11$, show the resulting tables under

(a) open hashing

(b) closed hashing using linear probing

(c) closed hashing using quadratic probing

(d) closed hashing with secondary hash function $h_2(x) = 7 - (x \bmod 7)$

(e) following rehashing what will be the table look like after closed hashing using linear probing

For each of the input values, $x$, the list of values $x \bmod 7$ is, respectively { 3, 0, 6, 6, 4, 5, 1 }.

4. **(20 marks.)**

When designing electronic chip circuitry it is often the case that key modules are laid out in a tree configuration, as shown in Fig. 1 over, with nodes representing modules and edges representing connections (wires) between those modules. A further restriction often requires that the tree is laid out along two parallel layers such that no two nodes connected by an edge can be on the same layer. Fig. 1(a) - (c) shows three different *embeddings* of the same tree. Note how no two nodes connected by an edge are on the same level. Note, also, how the leaves ($c$, $e$ and $g$) of the tree need not all be on the same level.

Edge crossings cause great complications for chip designers so it is imperative that when the tree is "compressed" in to two layers as few edges as possible cross. In this respect the three embeddings shown are not of equal value.

If we call the leftmost and rightmost nodes on the two levels of an embedding (that is the top-left, top-right, bottom-left and bottom-right nodes), the *extreme* nodes, show that in a *minimum-crossing* embedding

(a) not all of the four extreme nodes must be internal (non-leaf) nodes; (6 marks.)

(b) at least one of the four extreme nodes must be a leaf node; (10 marks.)

(c) at least two of the four extreme nodes must be leaf nodes; (4 marks.)

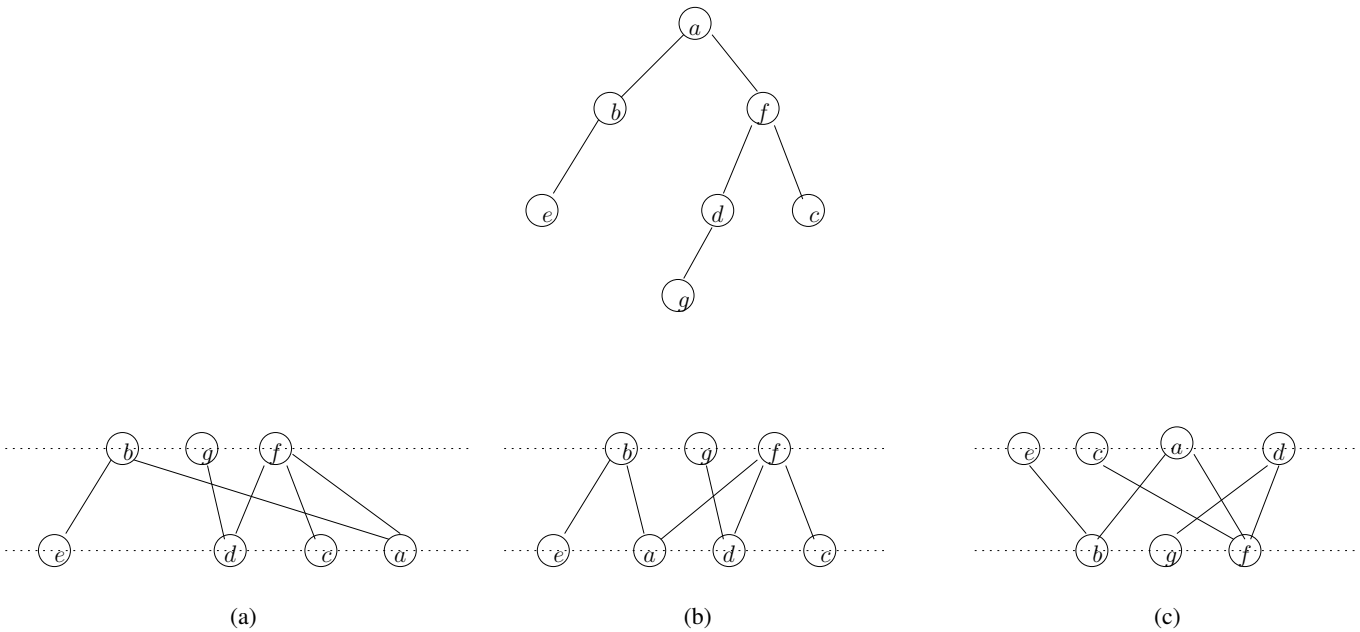


Figure 1: A tree and three different embeddings of the tree on 2 levels.