



# UNIVERSITY of LIMERICK

OLLSCOIL LUIMNIGH

## COLLEGE of SCIENCE and ENGINEERING

Department of Computer Science  
and Information Systems

### Annual Repeats Assessment Paper

Academic Year:	2007/2008	Semester:	Repeat Exam
Module Title:	Data Structures and Algorithms	Module Code:	CS4115
Duration of Exam:	2½ hours	Percent of Semester Marks:	65
Lecturer:	P. Healy	Paper marked out of:	100

#### Instructions to Candidates:

- There are two sections to the paper: Fill In The Blank-type Questions and Long Questions
- The mark distribution is 60 marks for Fill In The Blank Questions and 40 marks for the Long Questions
- Answer all questions in all sections

#### Section 1. Fill In The Blank Questions (15 × 4 marks).

- Please label your answers 1.1, 1.2, etc.

1. In a multiple choice exam, if negative marking is being used, what should be the penalty for wrong answers so that somebody who guesses throughout ends up with 0; each question has  $k$  possible answers and the questions are worth  $n$  marks each?
2. Evaluate the sum  $\sum_{i=0}^{\infty} 4^{-i}$ .
3. Evaluate the sum  $\sum_{i=2}^{\infty} 4^{-i}$ .
4. In “Big-Oh” notation, what best approximates  $\sum_{i=0}^n i^5$ ?
5. Order the following functions of  $n$  in ascending order:  $n \log n$ ,  $n!$ ,  $n \log \log n$ ,  $n \log^2 n$ ?
6. Give the time complexity of *radix sort* in “Big-Oh” notation.
7. An  $O(n)$ -time algorithm for sorting integers in the range  $[0, 2^{32} - 1]$  is possible by making 4 passes of *radix sort*, with buckets of what size?
8. Given  $A$ , a square matrix of size  $n \times n$ , what is the time-complexity of computing  $A^2$ ?
9. Given  $A$ , a square matrix of size  $n \times n$ , what is the time-complexity of computing  $A^3$ ?
10. Given  $A$ , a square matrix of size  $n \times n$ , what is the time-complexity of computing  $A^k$ , for some integer  $k$ ?
11. What is the time-complexity of the following piece of code in “Big-Oh” notation?  

```
sum = 0;
for (int i = 0; i < n; i++)
    for (int j = 1; j < n; j = j*2)
        sum = sum + n;
```
12. Suppose we exchange elements  $A[i]$  and  $A[i+k]$  of an array, which were originally out of order. How many inversions does this remove from the array, *at most*?
13. What is time complexity of testing a graph,  $G$ , for *acyclicity*?
14. Comment on the truth or falsity of the following two statements below, explaining your answer.  
**S1:** In a closed hash table, a key insertion followed by a deletion of that key returns the hash table exactly to its state prior to the insertion;  
**S2:** In a closed hash table, an insertion of a non-existing key can be performed faster than a deletion of an existing key.

15. Comment on the truth or falsity of the following two statements below, explaining your answer.  
**S1:** The step of Heapsort that accounts for its asymptotic running time is the `buildheap()`

step;

**S2:** On pre-sorted input, Heapsort's running time is  $O(n)$ .

## Section 2. Long Questions (40 marks).

- Label your answers 2.1 and 2.2 in your answer books

1. (20 marks.)

(i) (5 marks.)

Which of the following sorting algorithms is the odd one out, Insertion, Heapsort, Mergesort and Quicksort? Explain your answer carefully.

(ii) (15 marks.)

Give a detailed explanation of the operation of Quicksort. Explaining according to an example will be helpful.

2. (20 marks.)

(i) (8 marks.)

Graph colouring is an important problem in such problems as generating timetables or allocating frequencies to mobile phone masts. The *edge colouring* problem is to consider each vertex of the graph and to assign a different colour to every edge incident on it in such a way that the overall number of different colours used is as small as possible. This is sometimes referred to as  $\mathcal{X}'(G)$ .

Give as good a lower bound as you can for  $\mathcal{X}'(G)$ . That is, show some property  $y$ , so that  $y \leq \mathcal{X}'(G)$ .

How good is your property – how close to “equality” is your property? Demonstrate a graph where it does well or badly.

(ii) (12 marks.)

Give a detailed explanation of the operation of the algorithm to find the biconnected components of a graph. Explaining according to an example will be helpful.