The University of Nottingham

SCHOOL OF COMPUTER SCIENCE
AND INFORMATION TECHNOLOGY

A LEVEL B MODULE, AUTUMN SEMESTER 2006-2007

ALGORITHMS AND DATA STRUCTURES

Time allowed TWO hours

Candidates must NOT start writing their answers until told to do so

Answer QUESTION ONE and THREE other questions

Marks available for sections of questions are shown in brackets in the right-hand margin.

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a dictionary to translate between that language and English provided that neither language is the subject of this examination.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn examination paper over until instructed to do so

Question 1

This multiple choice question is compulsory. Please choose one answer in every part.

(a) You need to store a large and more or less unchanging collection of (key, value) pairs. It is important that you can retrieve the value given the key as fast as possible. The most suitable data structure for this application is:

i unordered list

ii ordered list

iii hash table

iv unbalanced binary search tree

v heap

(4)

(b) The running time of the binary search method on an ordered array is

i constant in the size of the array

ii logarithmic in the size of the array

iii linear in the size of the array

iv quadratic in the size of the array

v exponential in the size of the array

(3)

(c) You need to find the shortest path between two vertices in an unweighted graph (in which there are no weights or distances associated with the edges of the graph). Which algorithm is best suited for solving this problem:

i breadth-first search

ii depth-first search

iii Dijkstra's shortest path algorithm

iv Prim's algorithm for computing a minimal spanning tree

v topological sort

(3)

(d) Which of the following is an invariant of the loop in the method below (assume `n` and `p` are both non-negative integers)

```
int power(int n, int p) {
   int res = 1;
   int i = 0;
   while (i < p) {
      res = res * n;
      i++;
   }
   return res;
}
```

i $\texttt{res} = \texttt{n}^{\texttt{p}}$

ii $\texttt{res} = 1$

iii $\texttt{i} < \texttt{p}$

iv $\texttt{res} = \texttt{n}^{\texttt{i}}$

v $\texttt{res} \leq 1$

(3)

(e) What is the worst case running time complexity of the method below:

```
int power(int n, int p) {
   int res = 1;
   int i = 0;
   while (i < p) {
      res = res * n;
      i++;
   }
   return res;
}
```

i $O(1)$

ii $O(n)$

iii $O(p)$

iv $O(n^p)$

v $O(log_p n)$

(3)

(f) Suppose a hash table uses open addressing with double hashing. The primary hash function is $h(k) = k \ mod \ 31$ and the secondary hash function is $d(k) = k \ mod \ 7$. What are the first two numbers in the probing sequence generated for the key $k = 51$:

i $20, 21$

ii $20, 22$

iii $20, 2$

iv $22, 24$

v none of the above

(3)

(g) Suppose you have an algorithm which, on an input of size 1000, runs in 3 milliseconds. You know that the running time of the algorithm is quadratic in the size of the input. What would be your best guess for the algorithm's running time on an input of size 10,000:

i 9 ms

ii 30 ms

iii 90 ms

iv 300 ms

v 900 ms

(3)

(h) An algorithm's running time is described by the following function: $t(n) = 2n^2 + log_2 n$. What is the time complexity of this algorithm:

i $O(1)$

ii $O(log_2 n)$

iii $O(n)$

iv $O(n \ log_2 n)$

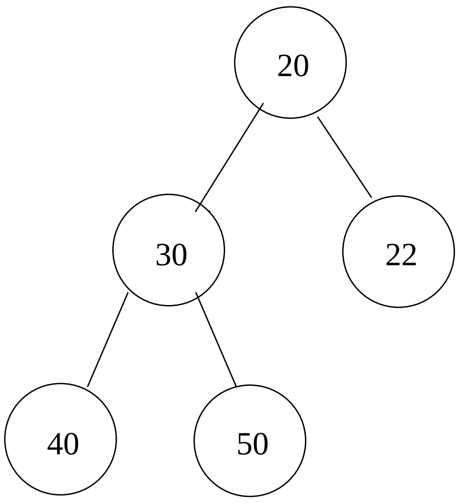
v $O(n^2)$

(3)

Question 2

(a) What is a heap data structure? What kind of application it it useful for and why? (6)

(b) Describe how insertion in heaps works. Show the result of inserting an item with key 21 in the following heap: (5)



(c) Describe how deletion in heaps works. Show the result of removing the root from the heap above. (5)

(d) Describe how to implement a heap using an ArrayList (give Java code or pseudocode for `insert` and `remove` methods). (9)

Question 3

Write the Java code for implementing a doubly linked list class `DList` for storing Objects using the following auxiliary `DNode` class:

```
class DNode {
   Object element;
   DNode next;
   DNode prev;

   DNode (Object o, DNode n, DNode p) {
      element = o;
      next = n;
      pre = p;
   }
}
```

The `DList` class will have fields `DNode head` and `DNode tail`. You can add any other fields if you need them for implementing the methods below.

(a) Write a constructor `DList()` for the `DList` class which creates an empty list. (5)

(b) Write `void insertAtHead(Object o)` method which inserts a node with `o` as its element at the head of the list. (5)

(c) Write `Object removeHead()` method which removes the head of the list and return the object which was stored there. If the list is empty, returns null. (5)

(d) Write `void insertAtTail(Object o)` method which inserts a node with `o` as its element at the end of the list. (5)

(e) Write `Object removeTail()` method which removes the last node in the list and returns the Object stored there. If the list is empty, returns null. (5)

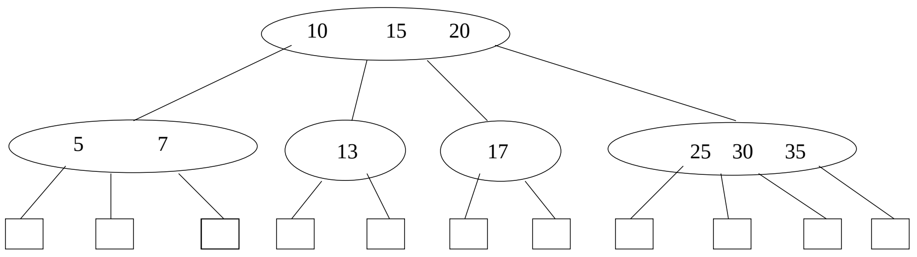
Question 4

(a) What is a 2-4 tree? Describe how many (key, value) pairs can a node hold, how are they ordered, how many children can a node have, and what is the order on the children. What is the shape of the tree (is it balanced and how?). Assume that the leaves store no items. (5)

(b) Describe (in English or in pseudocode) the search method for 2-4 trees. Trace search for an item with the key 27 in the 2-4 tree below. (5)

(c) Describe (in English or in pseudocode) insertion in 2-4 trees. Draw the result of inserting an item with key 33 in the 2-4 tree below. (5)

(d) Describe (in English or in pseudocode) deletion in 2-4 trees. Draw the result of deleting an item with key 17 in the 2-4 tree below. (10)



Question 5

Consider a data structure which stores a list of modules, where each module has a list of pre-requisites, for example:

- Mathematics for Computer Scientists, pre-requisites: none
- Introduction to Image Processing, pre-requisites: Mathematics for Computer Scientists
- Advanced Vision, pre-requisites: Introduction to Image Processing, Advanced Graphics

If the set of modules is badly designed, there may be cycles in pre-requisites, when for example module M_1 has module M_2 as one of its pre-requisites, M_2 has M_3, ..., and M_n has M_1 as a pre-requisite.

Describe an algorithm (in pseudocode) which, given a list of modules, returns true if there is a cycle in pre-requisites, and false otherwise. The algorithm does not have to be efficient to get full marks, but you need to state what is the complexity of your algorithm as a function of the number of modules.

You cannot assume that the list of modules is ordered in any particular way. You can assume that each module has a method `prerequisites` which returns a list of its pre-requisites. The list of pre-requisites for a module is also not ordered in any way. (25)

Question 6

(a) Explain what is meant by the term comparison sorting. (5)

(b) Name 4 comparison sorting algorithms. Explain in English or pseudocode how one of them works. (5)

(c) Prove that comparison sorting cannot be done faster than in $log_2(n!)$ (which is $O(n\ log_2 n)$) steps. (15)