IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2015

MSc in Computing Science
for Internal Students of the Imperial College of Science, Technology and Medicine

PAPER C580

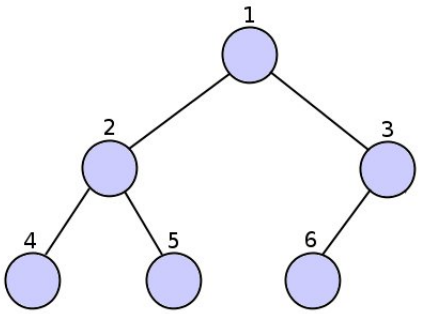
ALGORITHMS

Friday 8 May 2015, 14:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

1 a Objects in a (binary) max-heap are indexed from 1, beginning at the root and continuing down the tree, as shown in the diagram. The objects can be accessed by index, so $H[2]$ refers to the object in position 2 in a heap H. The number of objects in the heap is given by the attribute $H.size$.



i) How would you find the parent, the left child and the right child of object $H[k]$?

ii) Describe a $O(\log_2 H.size)$ algorithm to delete the object at position k in the heap. You do not need to write a coded procedure, but you should explain all steps that are required. Assume the objects in the heap have keys that can be compared, and that these keys determine the heap ordering.

b i) What is a *probe sequence* in an open address hash table?

ii) The linear probing method of collision resolution does not produce uniform hashing. Explain why not.

iii) It is simpler to support deletion in a hash table that uses chaining than in one that uses probing. Explain why, using an example.

iv) Professor Alfaman knows that in order to get optimal search performance from his hash table it must have a maximum load factor α_{max}. So, when table T, of size m, reaches α_{max} he copies the contents into a new table T' of size $2m$, setting $T'[i] = T[i]$, for all $0 \leq i < m$. Unfortunately, performance continues to degrade at the same rate as more entries are inserted. Explain how performance depends on load factor and what is behind this observed behaviour.

v) Can you suggest a way to resize Professor Alfaman's hash table that achieves the desired performance? Comment on the performance of insert when your method is used.

The two parts carry, respectively, 35% and 65% of the marks.

2a i) If the running time of an algorithm is "Big Omega (Ω) N^2", where N is the size of the input, what does this tell you?

ii) Using the formal definition of Big O, show whether

$$\log_2(N) + 2 = O\left(\frac{\log_2 N}{2}\right)$$

is true or not.

b Dr Beeterman wants to write an algorithm to determine whether the elements of an array A of size N are distinct.

i) Explain how to solve the problem for an array of integers in guaranteed $\Theta(N \log N)$ time. Code is not required.

ii) What is the lowest tight (Θ) bound that can be achieved if the array contains ASCII characters? Explain your answer.

c A k-bit binary number x is stored in an array B. $B[0]$ is the least significant bit and $B[k-1]$ the most significant. The following algorithm increments x by 1 $(\mathrm{mod}\ 2^k)$:

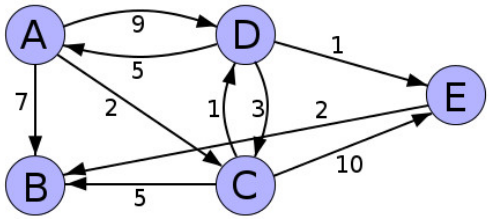
```
1: procedure INCREMENT(B, k)
2:     i = 0
3:     while i < k and B[i] == 1 do
4:         B[i] = 0
5:         i = i + 1
6:     if i < k then
7:         B[i] = 1
```

i) The cost of INCREMENT is determined by the number of bits in B that are flipped. Argue that a sequence of N increments starting from $x = 0$ will run in $O(kN)$ time.

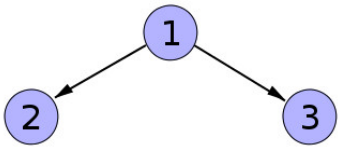
ii) Using the accountancy technique for amortised analysis (assigning costs to key operations) show that this bound can be tightened to $\Theta(N)$. *Hint* — differentiate between setting a bit to 1 and resetting it to 0.

The three parts carry, respectively, 30%, 30%, and 40% of the marks.

3 a Compute the shortest path from A to each vertex of the following graph using Dijkstra's algorithm. Draw a table showing, for each iteration of the algorithm, which edges are relaxed and which distances are updated.



b A *topological sort* of a directed, acyclic graph G computes an ordering of the vertices in which u must precede v if G contains an edge (u, v). So, in the graph below $\langle 1, 2, 3\rangle$ and $\langle 1, 3, 2\rangle$ are topologically sorted, but $\langle 2, 1, 3\rangle$ is not. Such a sort can be implemented by recording the *reverse* of the order that depth-first search colours vertices as finished. Explain why this method is correct while using the order that breadth-first search colours vertices finished is not. (Both searches colour vertices in the order *unvisited*, *visited*, *finished*.)



c The Reverse-Delete algorithm, outlined below, computes a minimum spanning tree (MST) for an undirected, connected, weighted graph G.

```
1: procedure REVERSE-DELETE(G)
2:     Sort the edges in G into decreasing order by weight
3:     T = G
4:     for all edges e (in decreasing order by weight) do
5:         if T is connected after removing e then
6:             remove e from T
7:     return T
```

i) Discuss what algorithms you would use to implement Reverse-Delete. What would be the time complexity of your implementation?

ii) Argue Reverse-Delete computes a correct MST for G. Assume edge weights in G are distinct, and the following property holds: if C is a cycle in G and e is the highest weight edge in C, then e is not in any MST of G. *Hints* — argue T is acyclic, that it spans G, and then that it is minimal.

The three parts carry, respectively, 20%, 20%, and 60% of the marks.

4a A craftsman makes several different products. The amount he earns for making each type of product, and the time he will spend making them is shown in the following table.

duration (hours)	1	2	3	5	7	8
earned	1	6	8	12	15	20

The craftsman has an 8-hour day to plan. Using a dynamic programming approach, compute his maximum possible earnings. Explain your method and show the intermediate amounts computed.

b The *Fibonacci numbers* are defined as follows:

$$F_0 = 0$$
$$F_1 = 1$$
$$F_i = F_{i-1} + F_{i-2}, \quad \text{when } i > 1$$

i) Write an algorithm to compute Fibonacci number F_n using a top-down (recursive) dynamic programming approach. You can use either Java or pseudocode.

ii) Write an algorithm to compute Fibonacci number F_n using a bottom-up (iterative) dynamic programming approach. You can use either Java or pseudocode.

c The *Mergesort* algorithm generates and solves subproblems according to the divide and conquer strategy. Could a dynamic programming approach be used within *Mergesort*? Would it improve the running time? Explain your answer.

d i) Using either Java or pseudocode, write a recursive procedure $Pow(x, N)$ to compute x^N, where N is a positive integer. Use a divide and conquer strategy. *Hint:*

$$x^N = x^{N/2} \times x^{N/2} \qquad \text{for even } N$$
$$x^N = x^{(N-1)/2} \times x^{(N-1)/2} \times x \qquad \text{for odd } N.$$

ii) Write recurrence expressions for the running time $T(N)$ of your Pow procedure in the following cases:

$$T(1) =$$
$$T(N) = \qquad \text{for even } N$$
$$T(N) = \qquad \text{for odd } N.$$

The four parts carry, respectively, 20%, 45%, 15%, and 20% of the marks.