PhD Qualifier Examination

Department of Computer Science and Engineering

Date: 03-Nov-2016 **Maximum Marks:** 100

[*Answer any <u>five</u> questions from Group A, and any <u>five</u> questions from Groups B and C.*]

Group A

A.1 Write only the answers (no need of any explanation). **($2\frac{1}{2}\times 4$)**

(a) In each case, state whether $f(n) = O(g(n))$, or $f(n) = \Omega(g(n))$, or both (that is, $f(n) = \Theta(g(n))$).

i) $f(n) = 5\log(n)$, and $g(n) = \log(n^3)$.

ii) $f(n) = n2^n$, and $g(n) = 3^n$.

(b) Given a sorted array A and an interval $[a,b]$, we have to find how many elements of A lie in this interval. What is the worst-case time complexity of the best known algorithm for this? $\log n + |b-a|$

(c) Let S and T be two binary search trees with m and n nodes, respectively. All the elements in S are distinct. Similarly, all the elements in T are distinct. We have to find the common elements of S and T. What is the worst-case time complexity of the best known algorithm for this? $O(m+n)$

(d) Which of the following operations is/are asymptotically more efficient in a height-balanced binary search tree compared to a binary max heap?

i) searching a key element ✓

ii) insertion

iii) deletion of maximum element

iv) deletion of minimum element

A.2 You are given an array $A[0\ldots n-1]$ storing exactly n of the $n+1$ integers $0,1,\ldots,n$. This means that exactly one integer x in the range $0,1,\ldots,n$ is missing in A. Your task is to determine x.

(a) If A is unsorted, propose an $O(n)$-time $O(1)$-space algorithm to find x. **(5)**

(b) If A is sorted (in ascending order), propose an $O(\log n)$-time $O(1)$-space algorithm to find x. **(5)**

A.3 Consider a rooted tree T with each node storing an integer key. Each node may have an arbitrary number of children. Let us impose a left-to-right ordering on the children of each node. We then index the nodes as $0,1,2,3,\ldots$ level by level starting from the root, and in each level from left to right (see Figure 1 for an example). We use an array A of integer pairs to store the tree as follows. Consider the node at index i in the tree. Let k be the key stored at this node, and let the index of the parent of this node be p. Then, the i-th entry in the array A is the pair (k,p). The index of the parent of the root is -1. Let n be the number of nodes in T. Assume that A is read-only and cannot be modified.

(a) Propose an algorithm that, given the array A and its size n, prints the keys stored in all the leaf nodes of the tree. Your algorithm should run in $O(n)$ time and use $O(1)$ extra space. **(5)**

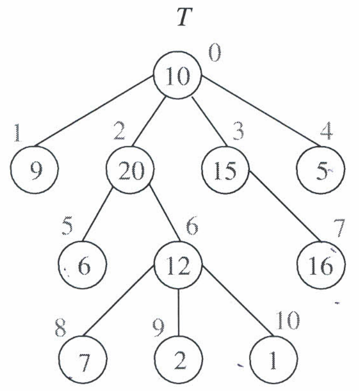
(b) Propose an algorithm that, given A, its size n, and an index $i \in \{0,1,2,\ldots,n-1\}$, prints the keys stored in all the child nodes of the node stored at index i. Your algorithm should run in $O(t+\log n)$ time and use $O(1)$ extra space, where t is the number of children of the node at index i. **(5)**

A.4 You are doing some stress-testing on a particular model of glass jars to determine the height from which they can be dropped without being broken. The setup for this experiment is as follows. You have k jars of the particular model, and a ladder with n rungs. You want to find the highest rung from which you can drop a jar and not have it broken. Call this the *highest safe rung*. The problem is to determine the highest safe rung in as few jar-drops as possible. You can reuse a jar as long as it is not broken.

(a) Suppose $k = 1$. Devise a strategy to determine the highest safe rung with at most n jar-drops. **(3)**

(b) Suppose $k = 2$. Devise a strategy to determine the highest safe rung with at most $f(n)$ jar-drops, where $f(n)$ is $o(n)$ (that is, $\lim_{n\to\infty} \frac{f(n)}{n} = 0$). **(7)**

Figure 1: Example for Question A.3



	0	1	2	3	4	5	6	7	8	9	10
A	10	9	20	15	5	6	12	16	7	2	1
	−1	0	0	0	0	2	2	3	6	6	6

A.5 (a) Define a weighted graph, and explain its adjacency-list representation with a suitable example. **(1+2)**

(b) Let $G = (V, E)$ be a weighted, undirected, connected graph. Each of its edges has weight either 1 or 2. What can be the minimum possible weight of a/the Minimum Spanning Tree (MST) of such a graph? Suggest an efficient algorithm to decide whether the given graph G has an MST with this minimum possible weight. Explain its worst-case time complexity. Assume that the input graph G is specified by an adjacency-list representation. **(1+4+2)**

A.6 You are given an unsorted array $A = (a_0, a_1, \ldots, a_{n-1})$ of n integers (positive, negative, or zero). A *run* in A is an interval $[i, j]$ of indices (not necessarily maximal) such that $a_i \leqslant a_{i+1} \leqslant \cdots \leqslant a_{j-1} \leqslant a_j$. Write an efficient C function that, given A and n, computes a run in A with the largest possible sum. Notice that the sum of an empty run is zero. **(10)**

A.7 Let $A = (a_0, a_1, \ldots, a_{n-1})$ be an array of n positive integers. We call A *super-increasing* if $a_i > a_0 + a_1 + \cdots + a_{i-1}$ for all $i = 1, 2, 3, \ldots, n-1$. For example, the array $(2, 3, 6, 14, 28, 60, 125)$ is super-increasing, whereas the array $(2, 3, 6, 11, 20, 50, 100)$ is not super-increasing.

(a) Write an efficient C function that, given A and n, determines whether A is super-increasing. **(4)**

(b) Write an efficient C function that, given a super-increasing array A, its size n, and a positive integer x, determines whether x can be written as

$$x = a_{i_1} + a_{i_2} + \cdots + a_{i_k}$$

for some $k \geqslant 1$, and for array indices satisfying $0 \leqslant i_1 < i_2 < \cdots < i_k \leqslant n-1$. **(6)**

A.8 Let $A = (a_0, a_1, \ldots, a_{n-1})$ be a sorted array of n distinct positive integers, B a positive integer, and $C = (c_0, c_1, \ldots, c_{n-1})$ a second array of n positive integers (not sorted, and may contain duplicate entries). A set $(i_1, i_2, \ldots, i_k)$ of indices satisfying $0 \leqslant i_1 < i_2 < \cdots < i_k \leqslant n-1$ for some $k \geqslant 1$ is called *feasible* if $a_{i_j} - a_{i_{j-1}} \geqslant B$ for all $j = 2, 3, \ldots, k$. The *cost* of a feasible set $(i_1, i_2, \ldots, i_k)$ of indices is $c_{i_1} + c_{i_2} + \cdots + c_{i_k}$. Your task is to maximize the cost over all feasible sets of indices. Call this maximum cost $M(n)$.

(a) Derive a recurrence relation for $M(n)$, that is, express $M(n)$ in terms of $M(i)$ for $i = 1, 2, \ldots, n-1$, where $M(i)$ is the maximum cost for the first i items $(A[0 \ldots i-1], B, C[0 \ldots i-1])$. Also supply the required initial condition(s). (**Hint:** For deriving the recurrence, consider the two cases: $i_k < n-1$ and $i_k = n-1$, where i_k is the last index in a feasible set. If $i_k < n-1$, then all the indices in the feasible set are chosen from $0, 1, 2, \ldots, n-2$. If $i_k = n-1$, consider the constraint $a_{i_k} - a_{i_{k-1}} \geqslant B$.) **(4)**

(b) Write an efficient C function to compute $M(n)$ using the recurrence of Part (a). **(6)**

Group B

B.1 Let $f : S_1 \rightarrow S_2$, $g : S_2 \rightarrow S_3$, and $h : S_1 \rightarrow S_3$ be functions. Suppose that $h = g \circ f$, that is, $h(x) = g(f(x))$ for all $x \in S_1$.

(a) Argue that if h is surjective (onto) and f is injective (one-one), then g must be surjective. **(5)**

(b) If h is injective, show that f must be injective. **(5)**

B.2 You are given a bag containing n unbiased coins. You are told that $n-1$ of these coins are normal, with heads on one side and tails on the other, whereas one coin is a fake, with head on both sides. You reach into the bag, pick out a coin uniformly at random, and toss the coin that you pick out for a total of k times.

(a) What is the probability that you see heads in all the k tosses? **(4)**

(b) If you see heads in all the k tosses, what is the conditional probability that you picked the fake coin? **(6)**

B.3 Solve the recurrence relation: $a(n) = v \times a(n/v) + e$, where $v, e \in \mathbb{N}$, $v > 1$, and $e > 0$. Assume that $n = v^k$ for some $k \in \mathbb{N}$. **(10)**

B.4 Prove or disprove the following.

(a) If M is an NFA that recognizes language L_1, swapping the accept and non-accept states in M yields a new NFA that recognizes $\overline{L_1}$ (the complement of L_1). **(4)**

(b) Regular languages are closed under doubling, that is, if the language L is regular, then so also is the language $L_2 = \{\text{two}(x) \mid x \in L\}$, where string doubling (two) is defined inductively as $\text{two}(\varepsilon) = \varepsilon$, and $\text{two}(ax) = aa \cdot \text{two}(x)$. **(6)**

B.5 Provide the following constructions.

(a) A DFA over the alphabet $\{a, b\}$, that accepts only those strings which have $aababb$ as a substring. **(4)**

(b) A CFG for the language $L_3 = \{a^i b^j c^k \mid j > 2i + k\}$ **(6)**

B.6 (a) Use Pumping Lemma to prove that $L_4 = \{wtw^R \mid w, t \in \{0,1\}^* \text{ and } |w| = |t|\}$ is not a regular language. **(7)**

(b) Consider the following grammar over the alphabet $\{a, b\}$ and with start symbol S.

$$S \rightarrow SS \mid ab \mid a \mid \varepsilon$$

Is the language generated by this grammar regular? Justify. **(3)**

Group C

C.1 (a) A combinational logic circuit has four inputs A_0, A_1, A_2, A_3 which represent four bits of an operand A, and one output Z. The output is 1 iff the input has at least three consecutive 0's or at least three consecutive 1's. For example, if $A_0A_1A_2A_3 = 1000$, then $Z = 1$, but if $A_0A_1A_2A_3 = 0100$, then $Z = 0$. Find a minimum product-of-sum expression for Z. **(5)**

(b) A switching circuit has two control inputs C_1 and C_2, two data inputs X_1 and X_2, and one output Z. The circuit performs one of the logic operations AND, OR, EQU (equivalence) and XOR on the two data inputs. The output depends on the control inputs as follows.

C_1	C_2	Z
0	0	X_1 OR X_2
0	1	X_1 XOR X_2
1	0	X_1 AND X_2
1	1	X_1 EQU X_2

Find an expression for Z. **(5)**

C.2 (a) Convert a D flip-flop to a JK flip-flop (do not touch the clock input). Draw the circuit diagram. (5)

(b) A binary ripple counter counts up from 0 to 16,383 (and then back to 0). How many flip-flops are required to design the counter? If the clock frequency is 8,192 MHz, what is the frequency at the output of the last stage? ($2\frac{1}{2}$+$2\frac{1}{2}$)

C.3 We say that from bit position i, a transfer of carry occurs if a carry is generated or propagated. For binary adders, the auxiliary transfer signal $t_i = g_i \vee p_i$ is derived by an OR gate, that is, $t_i = g_i \vee p_i = x_i y_i + (x_i \oplus y_i) = x_i \vee y_i$. The logic behind the change is that an OR gate is often faster than an XOR gate, so t_i can be produced faster than p_i.

Answer the following questions with respect to this modified adder.

(a) Comment on the correctness of the modified adder by showing the validity of the carry recurrence, when p_i is replaced by t_i. (5)

(b) Draw the complete architecture of a 4-bit adder using this modification. Comment on the impact on area and time because of the above change. (5)

C.4 Matrix transposition is an important operation in many signal-processing and scientific computations. Consider transposing a 4×4 matrix A, stored in row-major order starting at address 0, and placing the result in B, stored in row-major order beginning at address 16. We use two nested loops (indices i and j), copying $A_{i,j}$ into $B_{j,i}$ within the inner loop j. Assume word-addressed cache and main memory units. Note that each matrix element is accessed exactly once. Draw two 4×4 tables representing the matrix elements, and place **H** for hit and **M** for miss in each entry for the following cache organizations of an 8-word cache.

(a) Direct-mapped, 2-word lines (5)

(b) Two-way set-associative, 1-word lines, LRU replacement policy (5)

C.5 (a) Define a semaphore. (3)

(b) Consider two processes P and Q. P prints all odd integers from 1 to 100, and Q prints all even integers from 1 and 100. Write pseudocodes of P and Q using semaphores such that when the two processes are run, all integers from 1 and 100 are printed consecutively. (5)

(c) Name four information about a process that are typically kept in the PCB of a process. Your answer should contain all of the following: basic information, scheduling-related, and memory-related information of a process. (2)

C.6 (a) Consider a demand-paged memory-management system with 8KB page size, 32-bit address space, and 16GB RAM. The page table stores 4 extra bits per entry, in addition to the page-frame information. What is the size of the page table if the memory is aligned at byte boundaries? Show all calculations. (5)

(b) Given that the page table is stored in the main memory, and page faults cause costly disk accesses, why is paging still used? Identify the additional techniques over pure paging that make paging effective, briefly explaining how. (5)

Qualifier