香 港 中 文 大 學
The Chinese University of Hong Kong

Course Examinations 2011-2012

Course Code & Title : CSC 2100C Data Structures

Time allowed : 2 hours 0 minutes

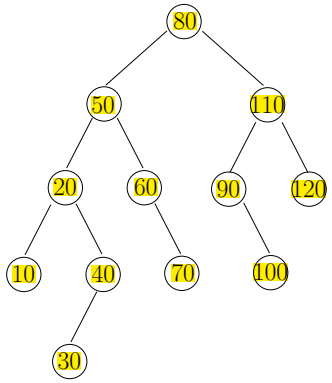
Student I.D. No. : ..... Seat No. : .....

Please write all your solutions in the answer book.

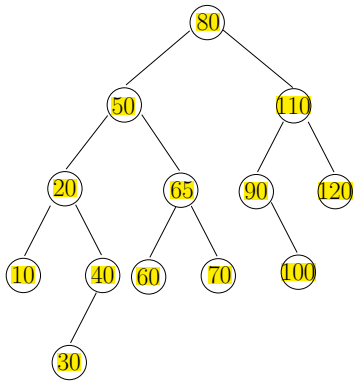
Problem 1 (10 marks) Let $f_1(n), f_2(n), g_1(n)$, and $g_2(n)$ be functions of n. Prove: if $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$, then $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$.

Solution: Since $f_1(n) = O(g_1(n))$, there exist constants n_1, c_1 such that $f_1(n) \leq c_1 \cdot g_1(n)$ for all $n \geq n_1$. Similarly, there are constants n_2, c_2 such that $f_2(n) \leq c_2 \cdot g_2(n)$ for all $n \geq n_2$. Set $n_0 = \max\{n_1, n_2\}$ and $c = \max\{c_1, c_2\}$. It follows that $f_1(n) + f_2(n) \leq c(g_1(n) + g_2(n))$ for all $n \geq n_0$.

Problem 2 (10 marks) Give the resulting AVL-tree after inserting 65 into the following AVL-tree.

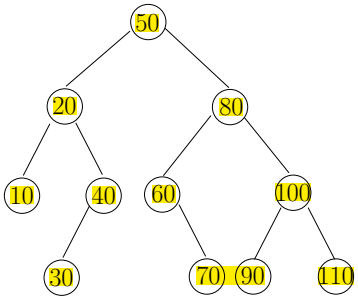


Solution:



Problem 3 (10 marks) Give the resulting AVL-tree after deleting 120 from the AVL-tree in Problem 2 (note: the original tree in the description of Problem 2, *not* the one in your solution).

Solution:



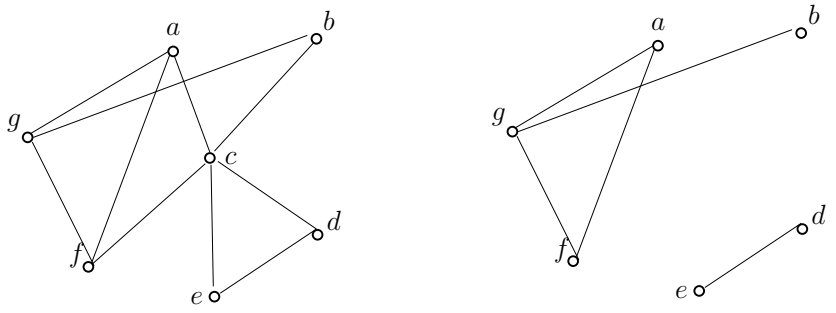
Problem 4 (10 marks) Let S and T be two sets of integers, neither of which is sorted. Given a number $x \in S$, define its *successor* $succ(x)$ as the smallest number in S that is greater than x (if x is already the maximum number in S, $succ(x) = \infty$). Describe an algorithm to output all numbers $x \in S$ such that $[x, succ(x)]$ does not contain any number in T. Your algorithm must terminate in $O((n+m)\log n)$ time, where $n = |S|$ and $m = |T|$.

For example, let $S = \{50, 30, 80, 20, 100\}$ and $T = \{15, 17, 9, 83, 55, 56, 42\}$. We have $succ(30) = 50$, and $succ(100) = \infty$. The numbers in S to be output are $20, 100$. Number 30, for instance, should not be output because $[30, succ(30)] = [30, 50]$ contains at least a number of T (i.e., 42).

Solution: Create an AVL-tree on S in $O(n \log n)$ time. For each number in T, find its predecessor x in S, and mark x if found. Doing so for all numbers in T takes $O(m \log n)$ time. Finally, scan S and output the unmarked numbers in $O(n)$ time.

Problem 5 (15 marks) Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. We know that G is *connected*, i.e., all pairs of vertices are reachable from each other. A vertex $v \in V$ is a *junction* if removing it and its incident edges makes G unconnected, i.e., two vertices in the remaining graph can no longer reach each other. Describe an algorithm to test whether a *given* vertex v is a junction. Your algorithm must terminate in $O(|E|)$ time.

For example, assume that G is the left graph below. Vertex c is a junction because, after removing it and its incident edges, the resulting graph (i.e., the right graph below) becomes unconnected – e.g., vertex e cannot reach vertex a. It is easy to verified that vertex a, for example, is not a junction. Your algorithm should therefore report "yes" when given c, and "no" when given a.



Solution: Simply start depth-first search from v. Consider the moment when v is popped out from the stack for the first time. If all the neighbors of v are already black, then return no (i.e., v is not a junction). Otherwise, return yes.

Problem 6 (20 marks) Let M be an $n \times n$ matrix of integers. All numbers in M are distinct. We know that every row is sorted in ascending order from left to right. Similarly, every column is sorted in ascending order from top to bottom. Give an algorithm to output the $k \leq n^2$ smallest numbers in M in $O(k \log k)$ time. Note that those k numbers must be output in ascending order. You can assume that every number in M can be accessed in constant time.

For example, assume that M is the following 5×5 matrix. For $k = 10$, the output should be (in this order): 10, 12, 13, 14, 18, 19, 20, 21, 25, 28.

$$\begin{pmatrix} 10 & 12 & 20 & 30 & 31 \\ 13 & 18 & 21 & 36 & 50 \\ 14 & 19 & 35 & 43 & 55 \\ 25 & 28 & 39 & 52 & 63 \\ 29 & 37 & 44 & 89 & 97 \end{pmatrix}$$

Solution: Initialize a heap H with the top-left number of M as the only element. In general, if a number of M is inserted in H, mark it in M. Repeat the following k times. Perform a delete-min from H. Let x be the number returned. Output x, and insert in H the numbers on its right and below it respectively, provided that (i) they exist, and (ii) they are unmarked.

Problem 7 (25 marks).

(a) Let S be a set of intervals, each of which has the form $[x, y]$ with x, y being integers. It is known that all those intervals contain a value C. Design a data structure such that, given any integer q, we can efficiently report all the intervals in S that contain q. Your structure must consume $O(|S|)$ space, and answer a query in $O(k)$ time, where k is the number of intervals reported.

For example, consider $S = \{[5, 80], [45, 75], [40, 60], [33, 55], [35, 82], [48, 52]\}$ and $C = 50$. Observe that all the intervals of S contain $C = 50$. Given $q = 80$, the query result should have $k = 2$ intervals [5, 80] and [35, 82].

(b) Let S be a set of intervals as in (a), but now each interval is associated with an integer *tag*. Design a data structure such that, given any integer q, we can efficiently report the the maximum tag of the intervals of S that contain q. Your structure must consume $O(|S|)$ space, and answer a query in $O(\log |S|)$ time.

For example, consider $S = \{([5, 80], 30), ([45, 75], 60), ([40, 60], 20), ([33, 55], 90), ([35, 82], 80), ([48, 52], 40)\}$ and $C = 50$. Pair $([5, 80], 30)$, for example, indicates an interval [5, 80] whose tag is 30. Given $q = 80$, we should report 80, i.e., the greater one of the tags of [5, 80] and [35, 82].

Solution:

(a) Let L (R) be the set of left (right) end points of the intervals in S. Sort L in ascending order, and R in descending order. Now consider a query with search value q. Assume $q > C$ (the opposite case $q < C$ can be handled similarly). Scan R in its sorted order, and report the interval of each right end point scanned, until coming across a right end point below q.

(b) Assume $q > C$ due to symmetry. Essentially we want to report the maximum tag of the intervals corresponding to those values in R that are greater than or equal to q. This can be easily achieved by indexing R with an AVL-tree and storing, at each intermediate node u, the maximum tag of the intervals corresponding to the nodes in the subtree of u. The tree consumes $O(|R|)$ space and allows us to solve a query in $O(\log |R|)$ time.

-End-