Algorithms -1
Tutorial 1
August 2, 2018

Solution Sketch

1. Prove mathematically that $3n^2 + 4n + 1 = O(n^2)$

   For this to be true, $3n^2 + 4n + 1 \le cn^2$ must be true for all $n > n0$ for some $n0 > 0$. Rearrange the inequality and find such a c and n and show it holds for some c and n0 (for example, easy to show for c = 10, n0 = 2).

2. Arrange the following in increasing order of growth (show your calculations):
   $(\lg n)^{\lg n}$, $n^{(1/(\lg n))}$, $\lg(\lg n)$, $(\log_{10} n)^2$, n, $(\mathrm{sqrt}(2))^{\lg n}$

   $n^{(1/(\lg n))}$, $\lg(\lg n)$, $(\log_{10} n)^2$, $(\lg n)^{\lg n}$, n, $(\mathrm{sqrt}(2))^{\lg n}$

3. Given a sorted sequence (ascending order) of n distinct integers a[1], a[2],...,a[n], design an O(log n) time algorithm to find if there exists an index i such that a[i] = i.

   Use concept of binary search.

   ```
   Let low = 1, high = n initially
   If (high < low) return "not found"
   Else mid = (high + low)/2
       If  (A[mid] = mid) then return "found"
       Else if (A[mid] > mid) search the left subarray with high = mid-1
           Else search the right subarray with low = mid+1
   ```

4. A cyclically sorted array is the array obtained by sorting an array and then doing a cyclic rotation. For example, [1,2,3],[3,1,2],[2,3,1] are cyclically sorted arrays. Design a O(log n) time algorithm to search for an element in a cyclically sorted array of n integers.

   A cyclically sorted array can be represented as a concatenation of two sorted arrays. We just need to identify the position where the first array ends (let's call this position pivot). e.g. in [3, 1, 2], the arrays are [3] and [1, 2], and the pivot is the index of 3. Once we find pivot, we can binary search in the two arrays individually. Note that the pivot is the maximum index i such that array[0] < array[i]. So, we can find pivot by doing a binary search in the original array.

5. Is the delete operation in a BST commutative (i.e., given a BST and two nodes x and y in it, does deleting first x then y and deleting first y then x result in the same final BST)?

   No. Counterexample shown in class.

6. Professor X designs a BST with one additional field per node which stores the number of nodes in the subtree rooted at that node (including the node itself). He claims that if a BST satisfies the property that the number of nodes in the left subtree is the same as the number of nodes in the right subtree for every node then the BST is a balanced BST. Is he right? Justify your answer.

Yes. Read about weight-balanced trees on the net, this is a special case of such trees.

7. What is the largest possible number of internal nodes in a red-black tree with black-height k? What is the smallest possible number? Justify your answer. Note that black height of a red-black tree is the black height of its root.

   Largest: $2^{2k} - 1$, Smallest: $2^{k} - 1$

8. Can you design an algorithm to find the median element in a Balanced BST with n nodes in $O(\log n)$ time? If n is even, choose the larger of the two median elements.

   At each node in the BST, we also store size of the sub-tree rooted at that node. Now, we create a balanced BST (e.g. Red-Black tree) and update the size accordingly while insertion and deletion. Now, we need to find the (n/2)-th element. Let's take k = n/2. We start at root node. If size of sub-tree of current node is equal to k, we return the key of current node as median. If left child has more than or equal to k elements, we continue searching in the left child. Otherwise, we update k = k − size(left child sub-tree) − 1 and continue searching in the right child.

9. Design an ADT that stores 2-tuples of the form (x, y), where x and y are integers. It is given that for any two tuples (x1, y1) and (x2, y2) in the ADT, x1, x2, y1, y2 are all distinct. The ADT must support the following operations:
   a. Search(a, b) – returns all tuples with x = a OR y = b OR both
   b. Delete(a, b) – deletes all tuples with x = a OR y = b OR both
   c. Insert(a, b) – inserts the tuple (a,b) in the ADT

   All the operations must run in O(log n) time. Write psuedocodes for the operations.

   Create two Balanced BSTs. Both store all tuples, but the first one (say A) uses x as key and the second one (say B) uses y as key.
   Search(a,b) – Search with a in A and with b in B. Return all tuples that are found
   Delete(a,b) – Search similarly, just delete what is found in both trees
   Insert(a, b) – Insert in both trees

10. Design an ADT that stores (element, value) pairs (x, v) where x and v are integers. Assume that all x's (elements) are distinct, though two elements may have the same value. The ADT must support the following operations:
    a. FindValue(x, v) – finds if a pair (x,v) is in the ADT or not
    b. Insert(x, v) – inserts the pair (x,v) in the ADT
    c. AddValueToAll(v) – adds the value v to the existing values of **<u>all</u>** elements in the tree

    All of the operations must run in O(log n) time.

    Use a Balanced BST, plus a variable offset, initialized to 0. For FindValue(x, v), search for the value (v – offset) and return the result. For Insert(x, v), insert the value (v – offset). For AddValueToAll(v), just add v to offset.