**CS430 Lecture 12 Activities**

Opening Questions

Previously we discussed the order-statistic selection problem: given a collection of data, find the kth largest element. We saw that this is possible in O(n) time for each kth element you are trying to find. The naïve method would just be to sort the data in O(n lg n) and then access each kth element in O(1) time.

1. Which of these two methods would you use if you knew you would be asked to find multiple kth largest elements from a set of static data?

2. What if our collection of data is changing (dynamic), would either these approaches work efficiently for a collection of data that has inserts and deletes happening?

Augmenting Data Structures  
For particular applications, it is useful to modify a standard data structure to support additional functionality. Sometimes the modification is as simple as by storing additional information in it, and by modifying the ordinary operations of the data structure to maintain the additional information.

Dynamic Order-Statistic Trees (Augmenting Balanced Trees)

1. What can we do with a binary search tree (and more efficiently with a balanced binary search tree)?

2. Consider the naïve method of finding the kth largest item is to sort the array and then access the kth item in O(1) time. Can we do this with a (balanced) BST? What if we augment it (HINT: recall how counting sort worked)?

3. What is the recursive formula to find the size of a subtree at node x

4. Discuss in detail how would you keep the size at a node correct when you insert a new node, and possibly rotate to keep the tree balanced?

|  |  |
| --- | --- |
|  |  |

5. Discuss in general how would you keep the size at a node correct when you delete anode, and possibly rotate to keep the tree balanced?

6. How can we use the augmented data at each node (size) in a balanced binary search tree to solve the kth largest item problem?

­­

7. How can we use the augmented data at each node (size) in a balanced binary search tree so when given a pointer to a node in the tree we can determine its rank (the index position of the node of the tree data in sorted order)?

8. Why not just use this approach always (not just with dynamically changing data) instead of the O(n) one?

9. Can we use this approach on a regular BST?