GDB Cs301 spring 2012

June 25, 2012

Suppose you are given a task to develop an electronic phone book in which you have to store names and phone numbers of employees of a company. The data structures available to you are linked list and Binary Search Tree (BST) and you have to choose only one data structure. Moreover, while selecting any data structure, you also have to keep in consideration insertion, deletion and search operations, because, the company can hire some new employees, search for an employee's phone number and some old employees can also leave this company. Discuss which data structure do you think would be more suitable for the above discussed scenario. Also discuss the reason why would you prefer this data structure over the other?

Solution 1

A binary search tree will provide logarithmic searching which is pretty good. Insertions or deletions will be more costly because they may require some tree rebalancing.
For your linked list, are you still going to keep the data in order or just stuff the nodes one after another. If you are putting them in no order, retrieval will be slow (linear) and insertions and deletions still won't be that fast. The problem with a linked list, if I remember correctly, is that you can't jump in the middle of a linked list. You start at the root and go to the next node then the next, etc. So you really can't perform a binary search or any kind of hashing on the data

Solution 2

the most popular variation of the Binary Tree is the Binary Search Tree (BST). BST,s are used to quickly and efficiently search for an item in a collection. Say, for example, that you had a linked list of 1000 items, and you wanted to find if a single item exists within the list. You would be required to linearly look at every node starting from the beginning until you found it. If you're lucky, you might find it at the beginning of the list. However, it might also be at the end of the list, which means that you must search every item before it. This might not seem like such a big problem, especially nowadays with our super fast

computers, but imagine if the list was much larger, and the search was repeated many times. Such searches frequently happen on web servers and huge databases, which makes the need for a much faster searching technique much more apparent. Binary Search Trees aim to Divide and conquer the data, reducing the search time of the collection and making it several times faster than any linear sequential search.