Character-Based Binary Tree Sorting for Integer Numbers

(Shin's Sorting Algorithm for Integer Numbers)

by

Dong-Keun Shin, July 10th, 1998

< Abstract >

In this paper, the author tests his algorithm (if it has not been discovered by other person) with integer numbers as input. For integer numbers, the data structure for tree node is different from that of the tree for alphabet strings input. Thus, another information which is "digit position" is added, and the difference is notified compared with the data structure in July 4, 1998's paper.

98.7.10 작성 완료

98.7.10 신태근 [illegible]

98.7.10 [illegible]

[illegible]-2-568-6007

July 10, 1998

< Description of the Sorting Algorithm >

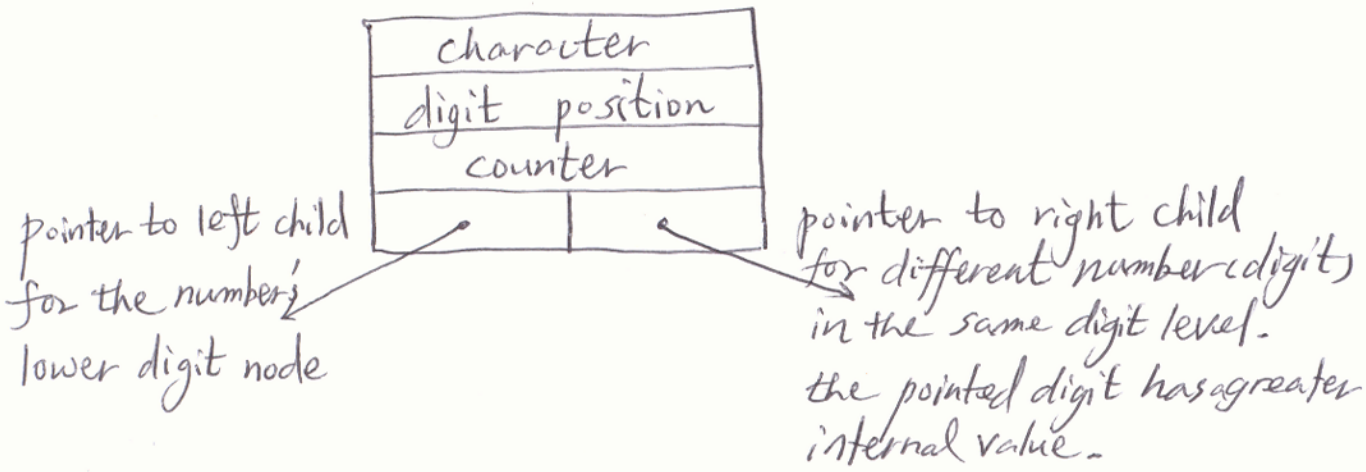
The sorting algorithm reads and stores character strings or integer numbers into a binary tree. Each node in the tree primarily represents a character in an input key string or a digit in an input integer number. For example, if five characters (or five digits) are in an input, five nodes are linked together by their left child pointers. Thus, each node in a tree has data structure for a character, digit position (for only integer numbers input), counter, and pointer to left child node and pointer to right child node. Left child pointer of a node points to the next character in a key string or lower digit in an integer number. If head portion of an input key string has been stored in a tree already, the algorithm uses its nodes for the portion and lets right child of the lowest node in the portion points to the first character (or digit) node in the other portion. This case happens to only character strings not to integers.

When the algorithm uses right child of a node, it inserts the new node in a right place by comparing internal expression value (e.g., ASCII, EBCDIC, or integer (0-9)) and traversing down through nodes' right childs. If the algorithm finds right child points to a nil, it cannot go any further. Sometimes the algorithm has to switch the current node with a new one and link the ~~new~~ current one using the new ones right child pointer if the new node's character's or integer's internal value is smaller than the current node's value. (or integer number)

If an identical character string has been already stored in a tree, the algorithm ~~increment~~ increments the lowest nodes counter by one.

For integer numbers input, digit position of new one should be compared to that of nodes in the tree from the begining.

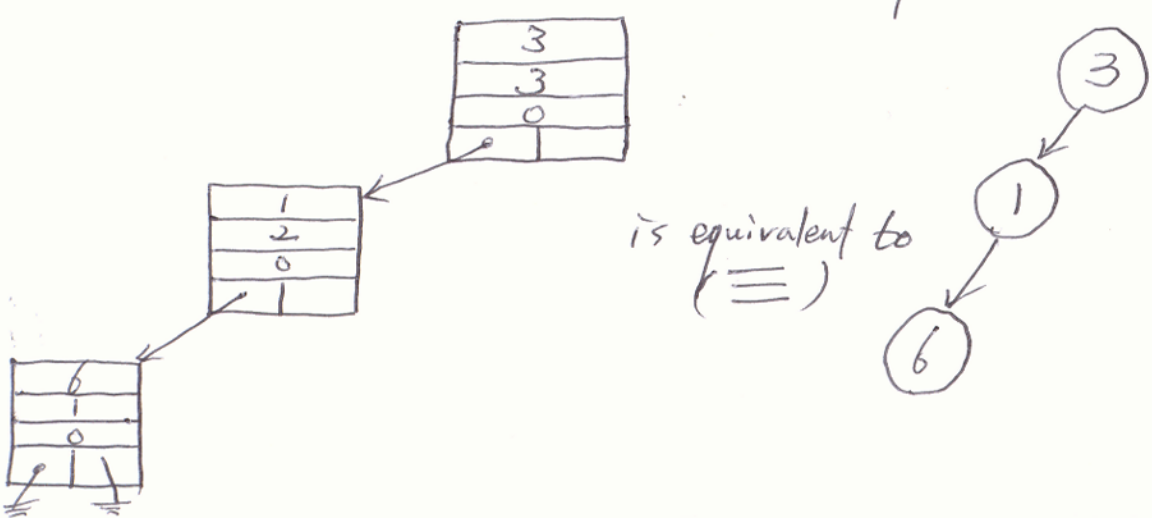
< Data Structure for Integer Numbers Input >



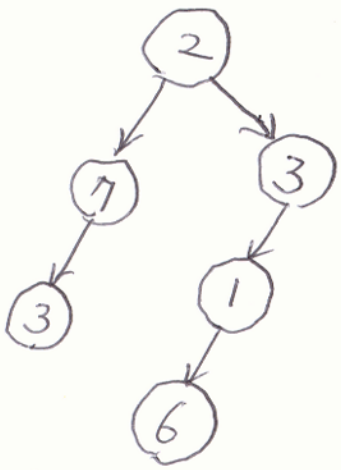
< Character-Based Binary Tree Sorting for Integer Numbers >

Input List : 316, 273, 4329, 9837, 16, 274, 107, 273, 5, 272.

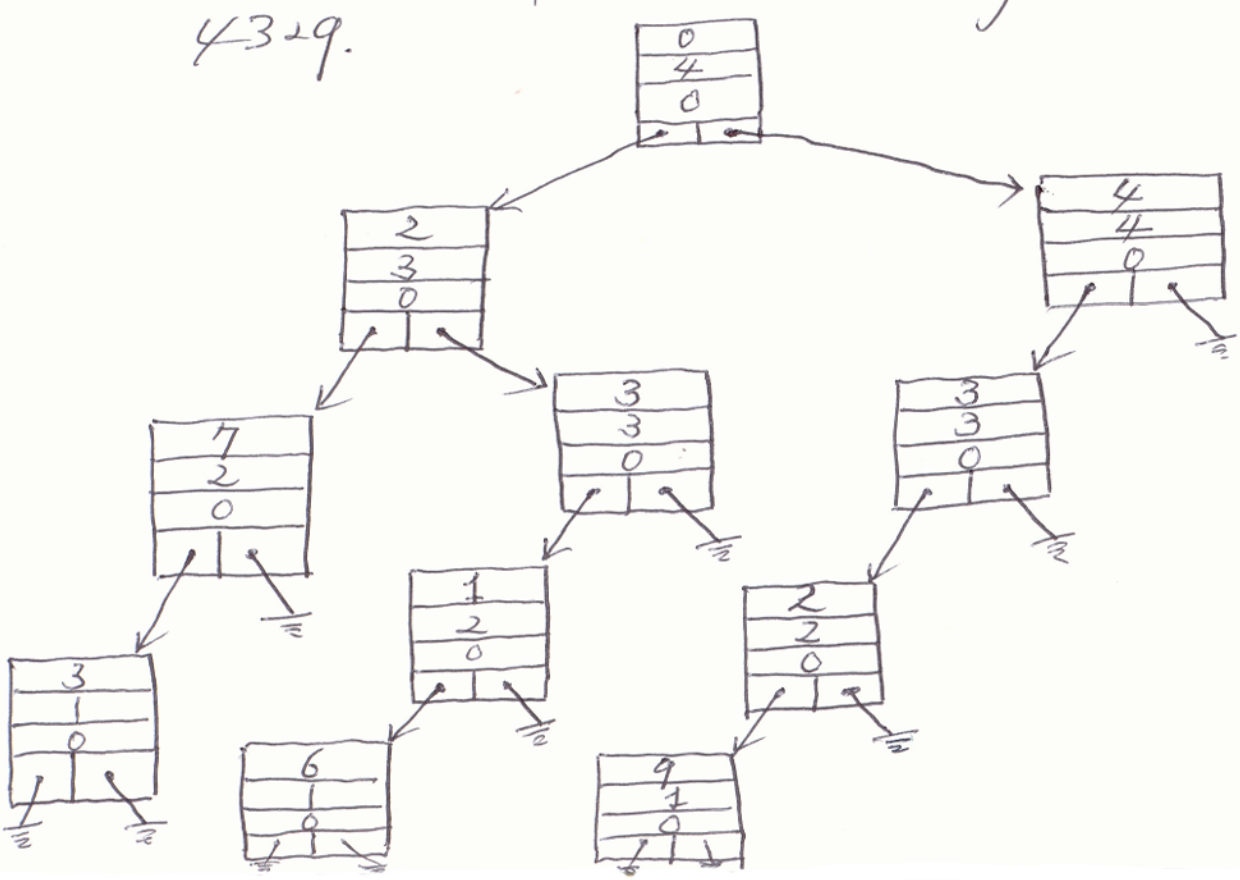
(1) The first input 316 is read and three nodes (e.g., 3, 1, 6) are made and linked together with the nodes' left child pointers.



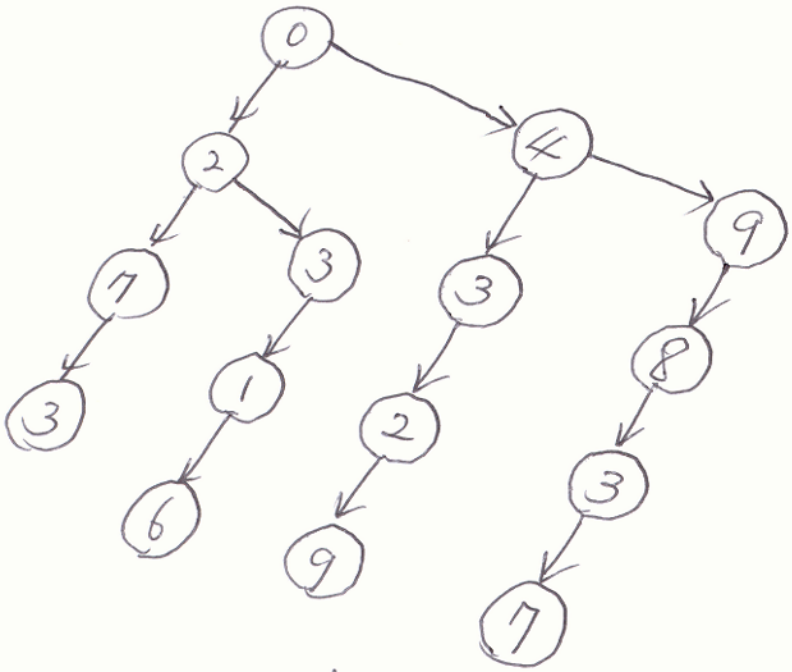
(2) The second input 273 is read and inserted into the tree. "2" in 273 is smaller than "3" in 316, so the node (2) will have (3) node as (2)'s right child.



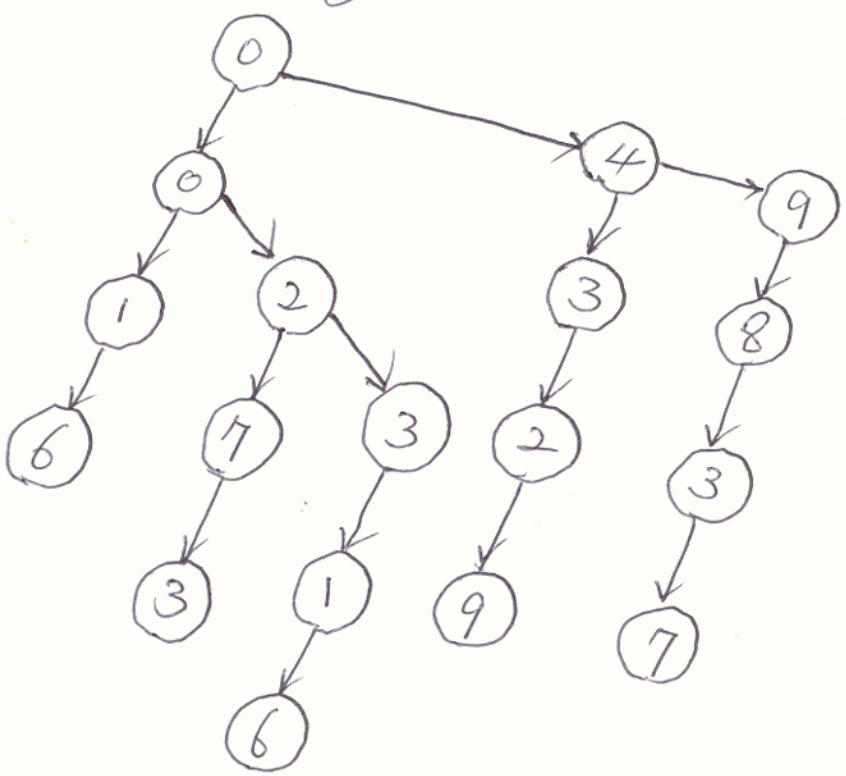
(3) The third input 4329 is read and found to have different digit position. Thus, (0) (zero) becomes the root and tree is changed to include 4329.



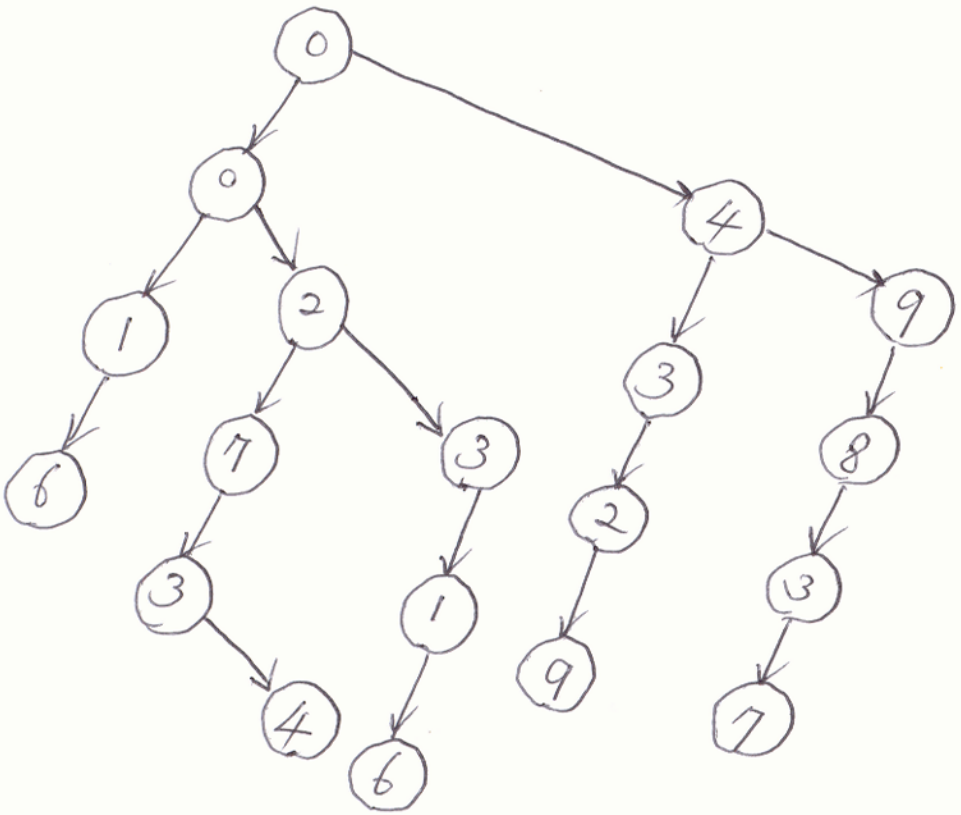
(4) Input 9837 is read and inserted into the tree.



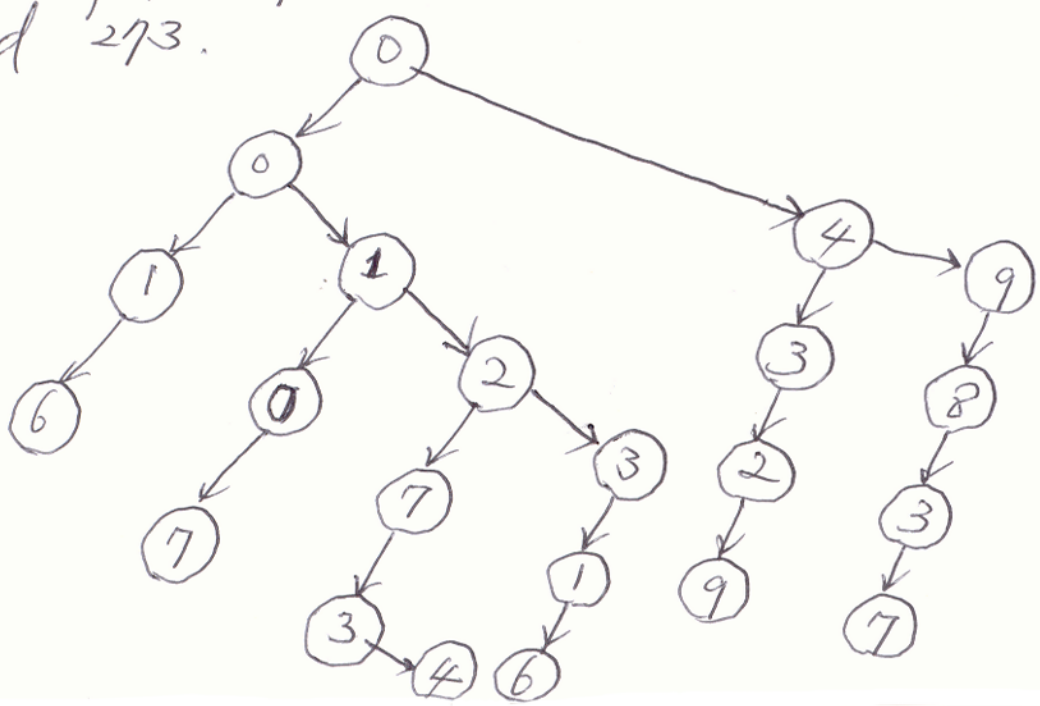
(5) Input 16 is read, and due to its lower digit position than 273, zero node is added.



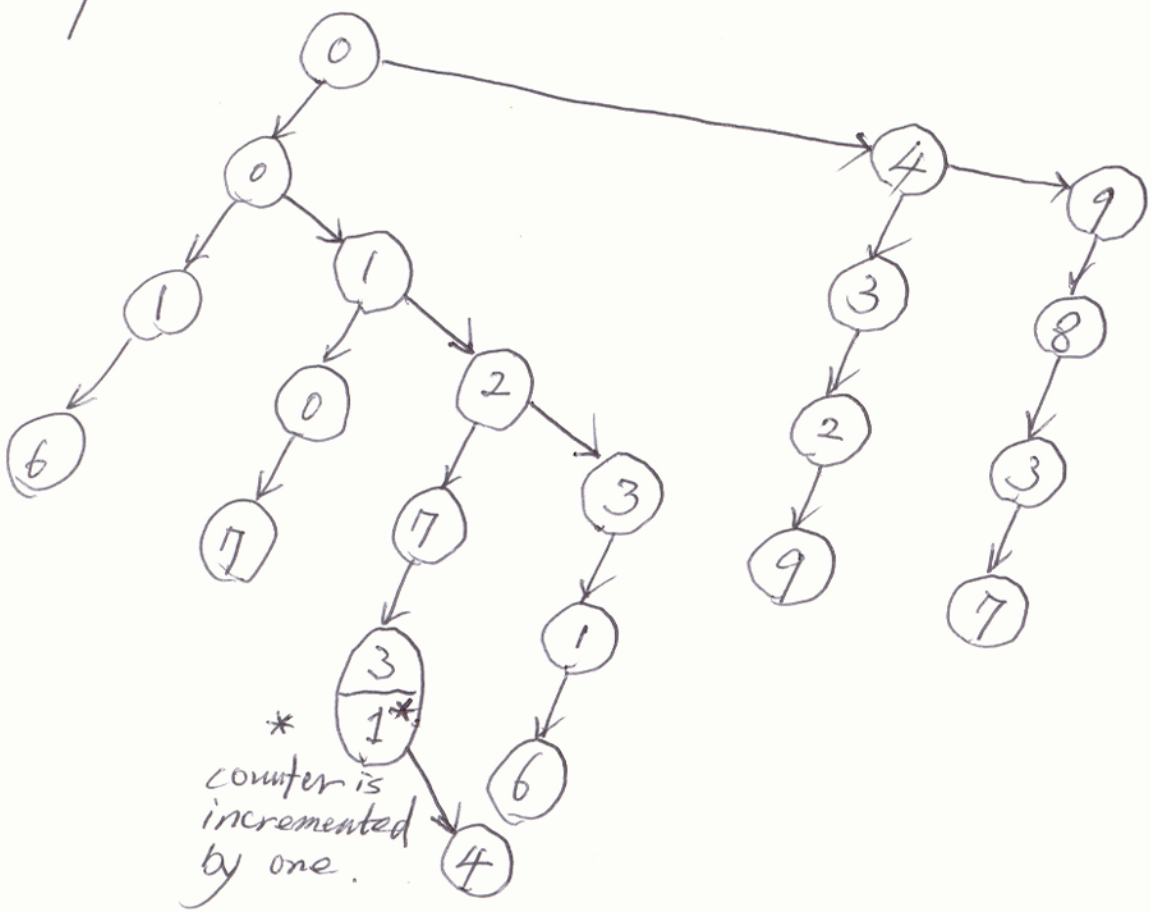
(6) Input 274 is read. The algorithm recognizes '27' portion in previous input 273, so it puts 274's (4) in the right child of (3) in 273.



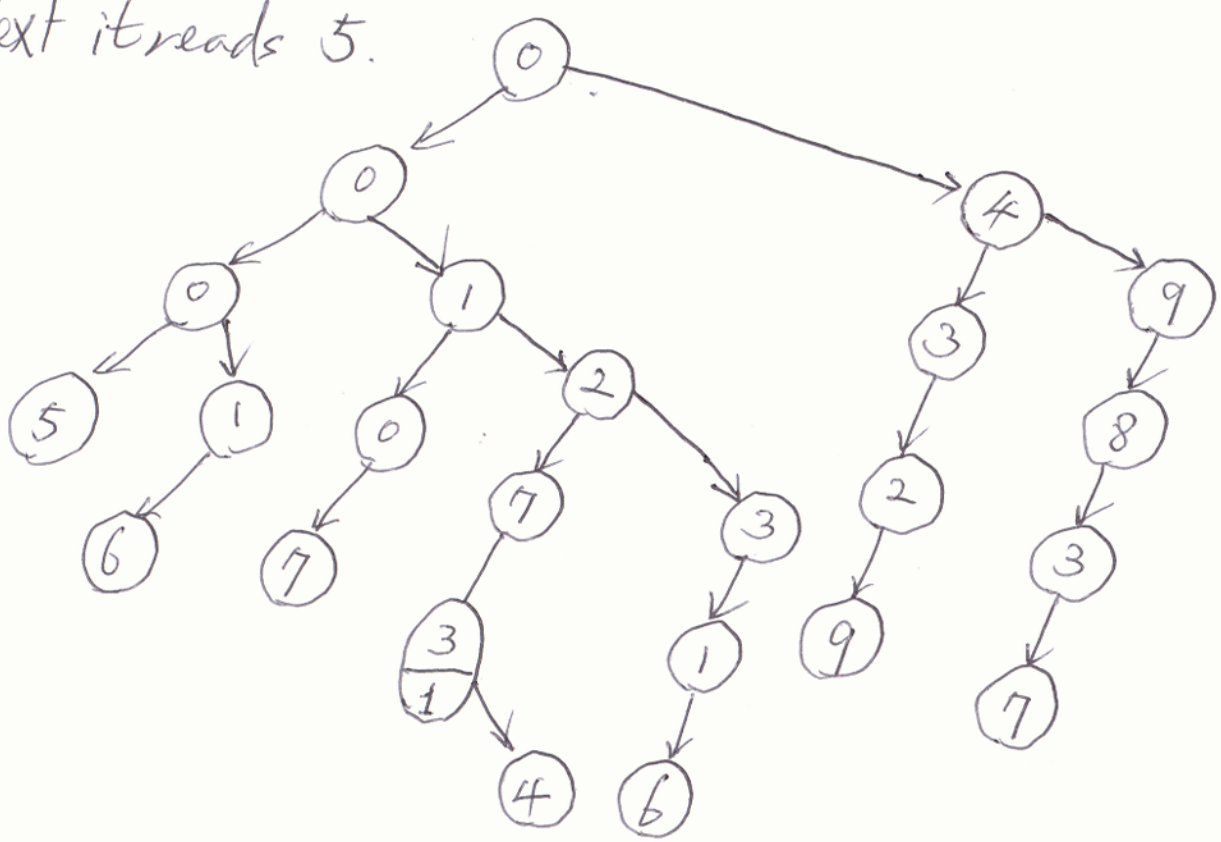
(7) Then input 107 is read and inserted between 16 and 273.



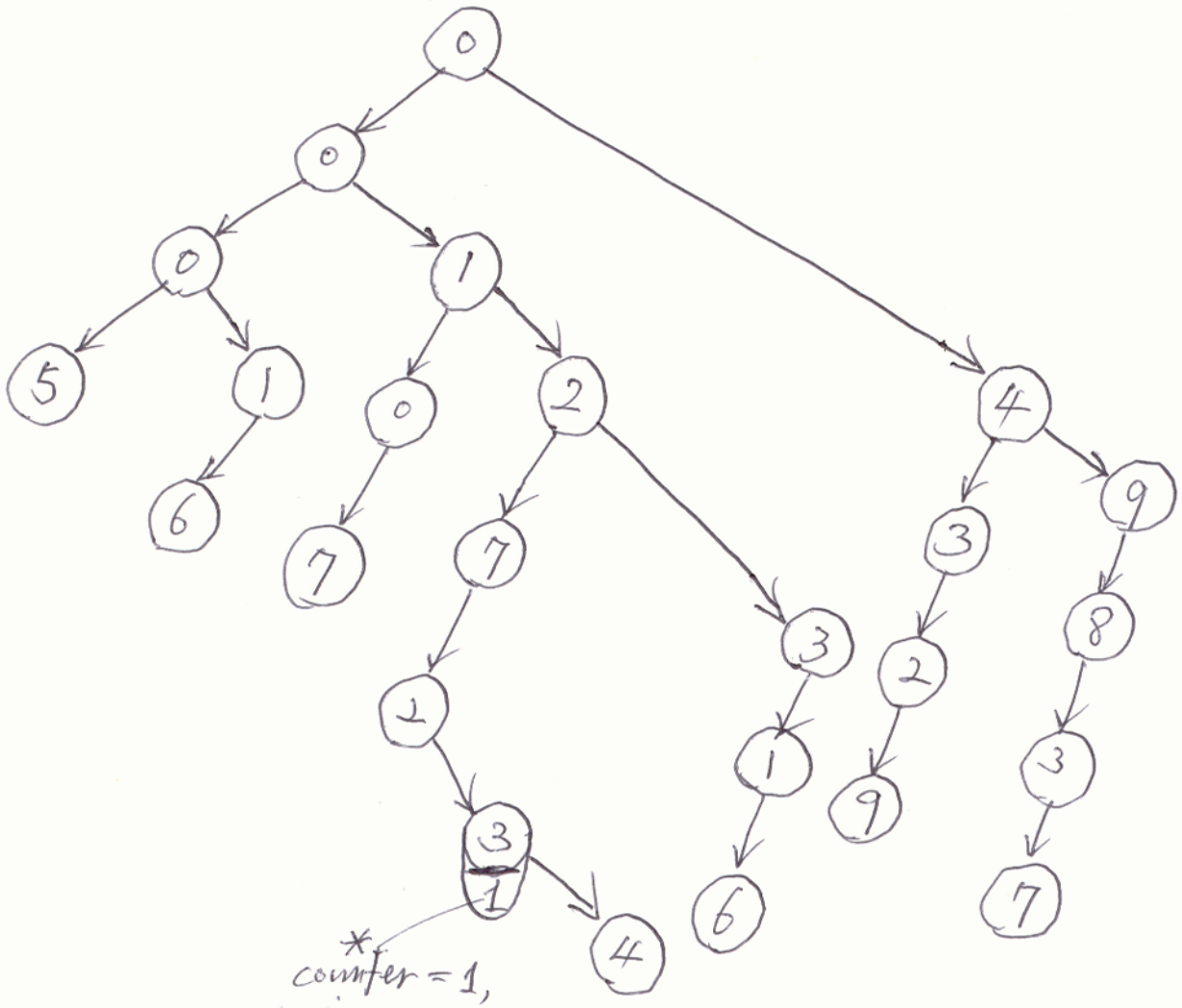
(8) Then the algorithm reads another 273 from the input list.



(9) Next it reads 5.



(10) Next it reads 272 as a final input.
It has to insert it right before 273.



(11) Prints out numbers stored in the tree.

0005, 0016, 0107, 0272, 0273, 0273*, 0274,
0316, 4329, 9837. — They are sorted.

* since counter in (3) of 273 has been incremented, and the value is set to one. Thus 273 is printed one more time.

Doo-Keun Sun
July 10, 1998.

< Combining branches of a tree together >

For a good parallel processing, the trees should be created in many subregions. Each parallel processing element (can be both hardware element or software element) shall create its own tree using its own allocated memory space (e.g. main memory or auxiliary storage such as disk space).

Because the sorting algorithm sorts input data based on their characters or digits, the subtrees which are created by processing elements can be efficiently combined (or merged) by alphanumeric characters. In other words, after a parallel sorting, processing elements send branches of created tree to destined processing elements. For example, there is a processing element for strings that starts with "A" (e.g. Apple, Ape, Ace, ...) and there is another processing element for string with the first character "B" (e.g., Boy, Bat, ...). Then it is easier to merge the subtrees, and, as a result, the algorithm reduces data movements. Relevant Architectures for the parallel processings

will be provided in future papers.

< Conclusion >

In this paper, an example with integer numbers is illustrated to show how the character-Based Binary Tree Sorting sorts the numbers. Parallel processing for this sorting method is considered. It is found that architecture-wise there is a great similarity between this sorting and Shin's massive cross referencing because processing elements performs work in their own regions and sends data to other processing elements in both cases. However, Shin does not know he has discovered this sorting algorithm for the first time.

Doyle Kenneth Shin, July 10, 1998

98.7.10 98.7.10 98.7.10 11