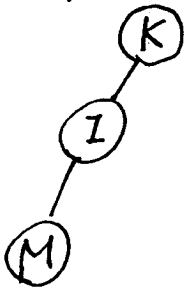
Sorting Method by Dong-Keun Shin 7/4/98

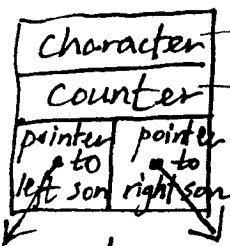
<An Example>

Input: KIM, KING, LION, KIND, JADE,
KIN, KENT, KILE, QUEEN

(1) The first input "KIM" is read and a binary tree is created. 'I' becomes left child (son) of 'K' node. 'M' becomes again a left child (son) of the 'I' node.



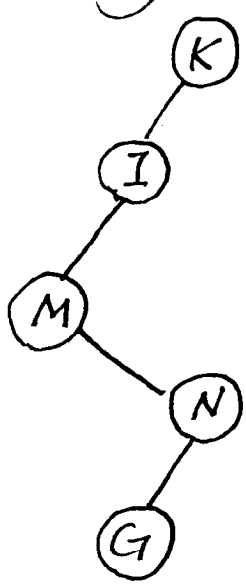
The data structure of each node is:



e.g. 'K', 'I', or 'M'

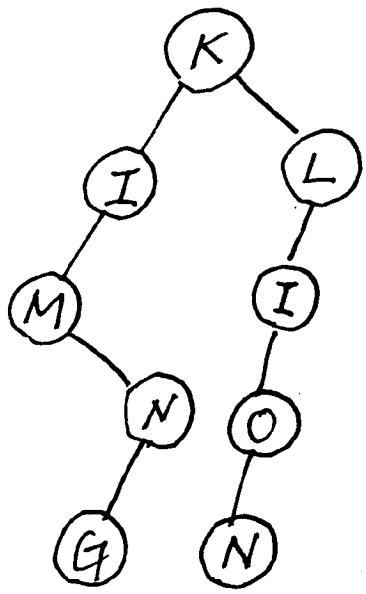
counts number of appearances of an input string in input.

(2) Then the second input "KING" is read and inserted to the binary tree. "KI" in the string "KING" is identical to "KI" in "KIM". Thus, 'N' becomes the right child of the node 'M' because the ASCII (or EBCDIC) or internal character value of 'N' is greater than that of the character 'M'.

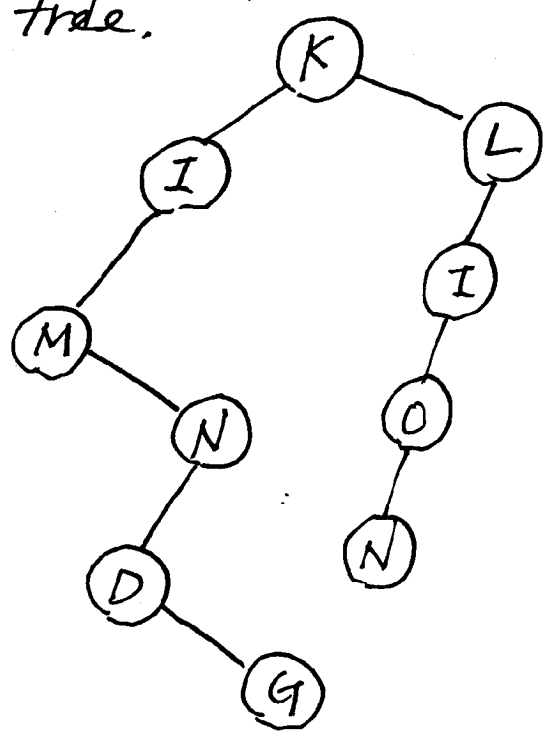


심동근 PB.1.4 에서
신태근 98.7.4

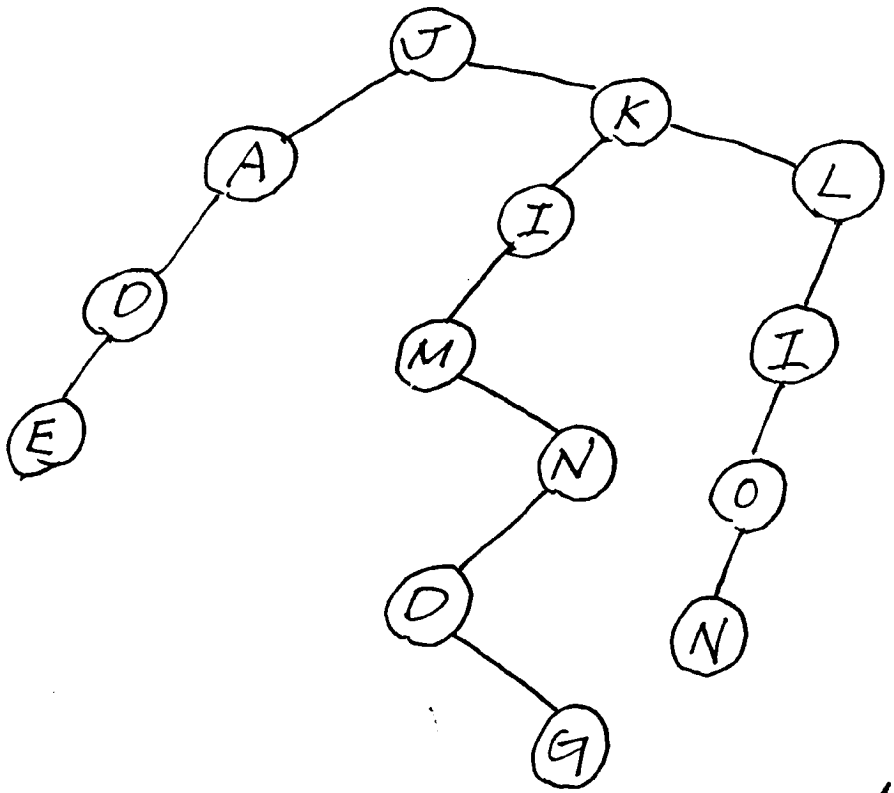
(3) The third input "LION" is read and inserted into the binary tree. The first character 'L' in "LION" is greater than 'K' in root node, so 'L' node will be the right child of 'K' root node.



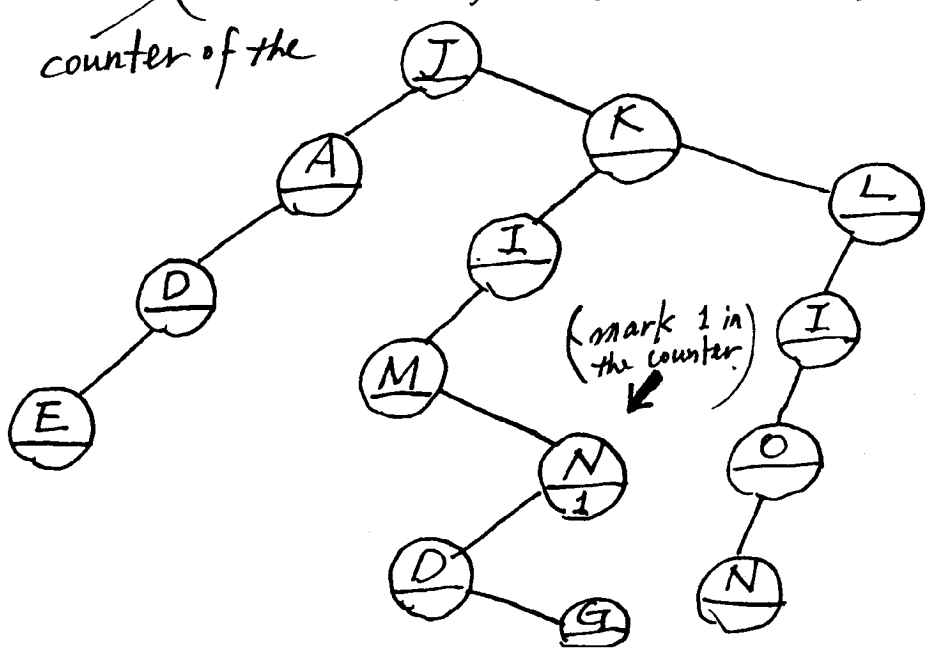
(4) Then next input "KIND" is read and inserted into the tree.



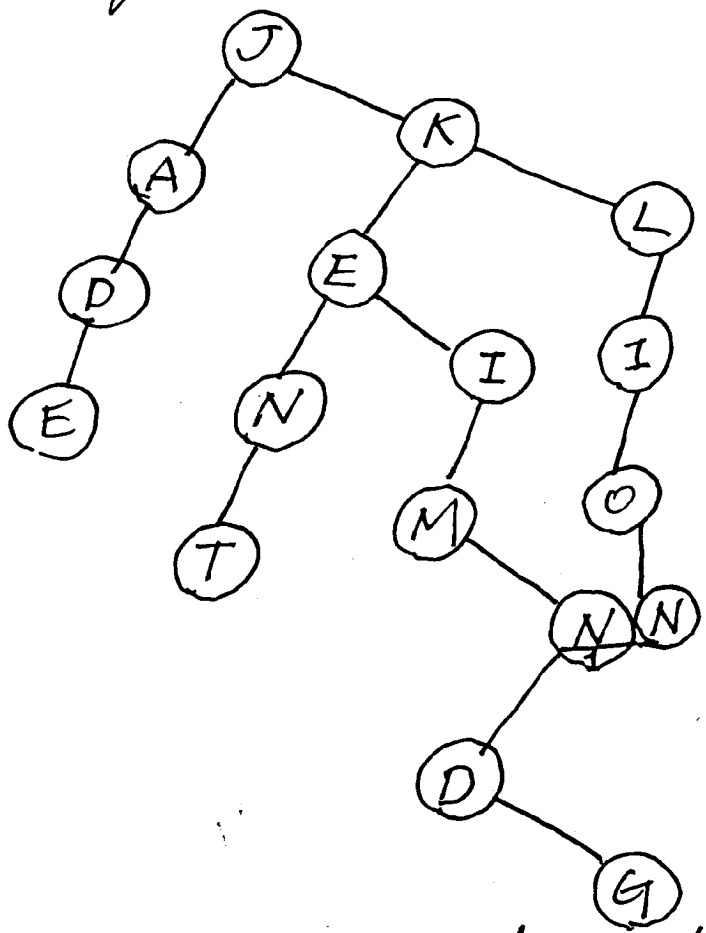
5) Fifth input "JADE" is read and inserted as shown below. 'J' becomes root node because it's internal value is smaller than 'K' in the tree's current root node.



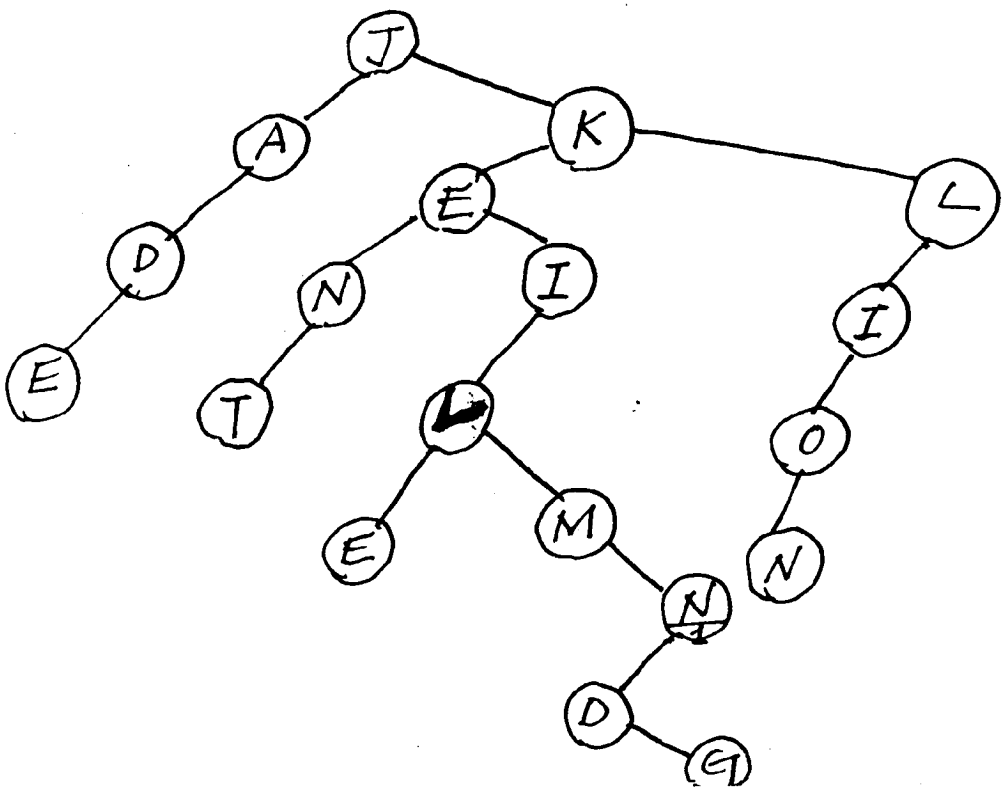
(6) Six input "KIN" is read and inserted. In this case, "KIN" has been already there, so have to mark 1 in the node 'N' for "KIN"s first appearance counter of the



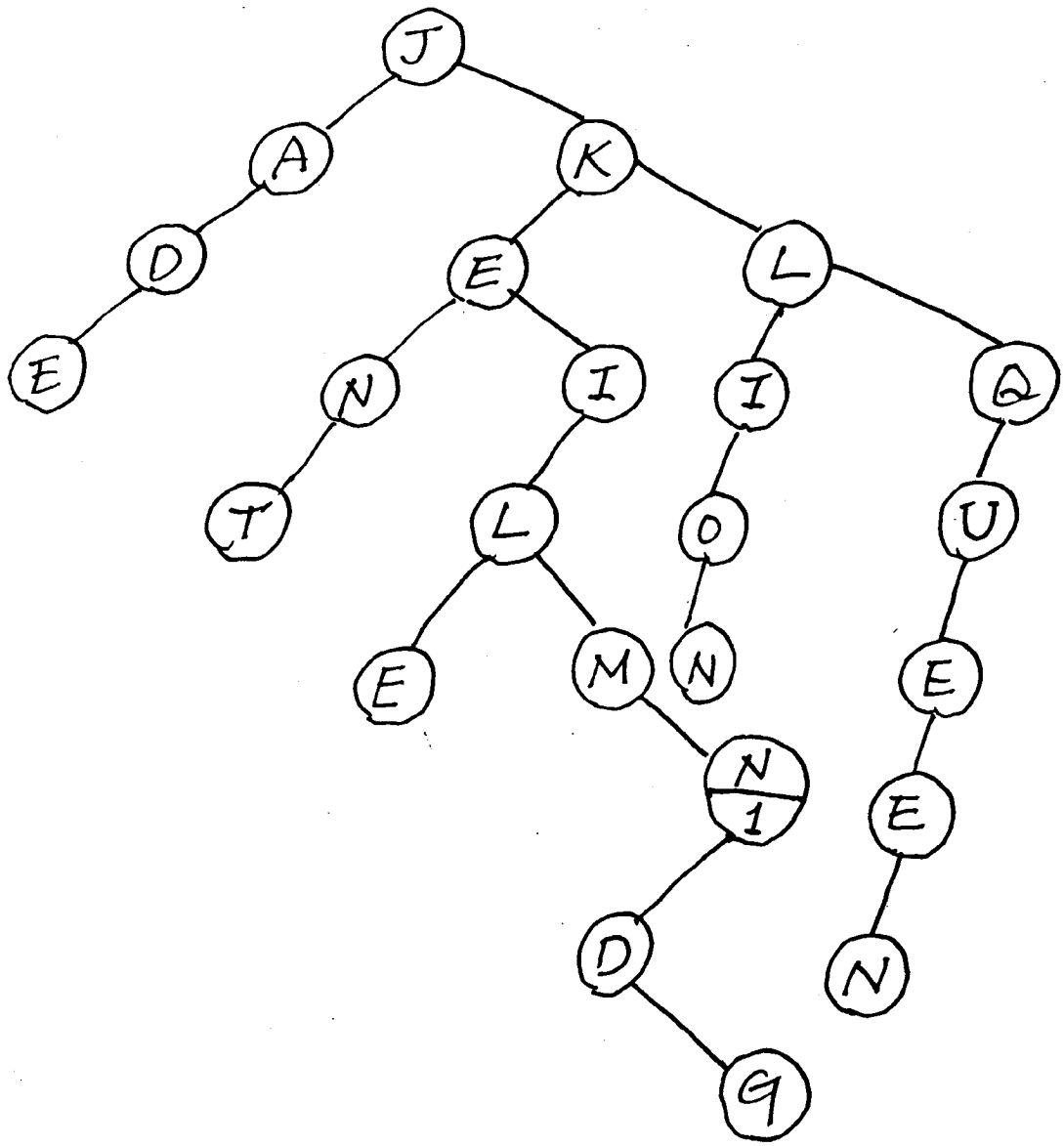
7) Seventh input "KENT" is read and inserted.
The input string "KENT" is inserted right before "KIM



8) Then "KILE" is read and inserted



(9) The last input "QUEEN" is read and inserted into the binary tree. "QUEEN" has the greatest internal value in the tree.



Oey Kem Shin 7/4/98.

Sorted list: JADE, KENT, KILE, KIM, KIN
KIND, KING, LION, QUEEN.

< A Sorting Algorithm developed by Dr. Dong-Keun Shin on July 3rd & 4th, 1998. >

This sorting algorithm creates ~~a tree~~ s a binary tree data structure while reading input data strings. It stores each character in an input string in a node and links them together using each node's pointer for its left child (or left son). However, if a part of the current input string has been already stored in the created tree, it uses the relevant portion of the tree and it uses the lowest node of the portion for different part of the current input string. The lowest node's right child will point to the different part of the string which will create a branch in the tree. This algorithm may read an input string which has been read already, or ~~a portion of an input~~ it may read an input string that is identical to a portion of prestored input string in the tree. In both cases, the algorithm increments the counter in the lowest node of the input string or the portion of prestored input string.

To print out strings in the tree, start from the root printing out character by character going to left son only. When a node's left son is empty or nil, the string is complete. Pointer to root or any header node for each string should be saved for subsequent uses. From a header node's character and its following left nodes' characters will be printed like "JADE". The printing algorithm for this tree recognizes no more left-out under 'J'. It moves to 'K', then prints out "KENT". The algorithm checks right child of each node from the bottom. It finds 'E' node has its right son. Therefore, 'E' node is skipped, and it goes to 'I' node. After "KI", 'L' and 'E' are left sons followed. Thus "KILE" is printed. The algorithm checks from the leaf node (which is 'E' node) to see the node has right child. If none, it pops up a stack to move up one level in the tree. It finds 'L' node has right son 'M', and it recognizes "KIM" with no left son in 'M' node. Therefore KIM is printed out.

Because identical keys may be stored or an input key may be the same with a first portion of other key, the algorithm has to check each node's counter if it

has been incremented. If incremented, the node contains an input's last character; therefore, the string should be printed out. An example "KIN" show next. Since 'N' node's counter has been incremente to be '1', It has to print out 'KIN' although the node 'N' has left child 'D' node. Then the algorithm visits 'D' node and finds that ithe node has no left child, so it prints out "KIND" for next one. The algorithm will check the right child of 'O' node and will find that there is 'G' node. The 'G' node does not have its left child; thus, the algorithm figures oat 'G' is at the end of a stored string. The algorithm prints oat relevant characters for the stored string in the current path, and the string is "KING". The 'G' node does not have right child; thus, there is no more string left in this brach. The algorithm pops up the stack up to 'K' node and goes to the right child of 'K' node, which is 'L' node. The algori saves the current pointer to 'L' node and goes to left childs deep enough to recognize string "LION".

Since "ION" in the string "LION" do not have right child, no other string uses ~~this any part~~ this path. Therefore stack has to be popped again to go back to 'L' node and the algorithm moves to 'Q' node. ~~and~~ It will do the same process it has done for "LION" for the string "QUEEN". It will go down to hit 'N' node in "QUEEN". No more left child in N node will make "QUEEN" to be printed out. Then the algorithm will check if nodes for "QUEEN" have any right child. When it finally figures out the highest node 'Q' does not have right child, the printing strings for output is finished. This printing process is (almost) the same with preorder binary tree traversal.

< Discussion and Analysis >

This sorting method is an O(N) Sorting algorithm. Comparing with Radix Sorting method, which is also O(N) sort meth. it is better in several respects:

① Requires much less memory space

② Better in parallel processing.

<Conclusion>

Although author does not know whether this algorithm has been discovered by someone already, he believes that he has not yet seen this sorting method. Dong-Keun Shin, thus, claims this method is a new sorting algorithm. If no one has found this algorithm before, ~~he Shin wants~~ author (Dong-Keun Shin) wants to name this algorithm "Shin Sort" or "Shin's sort" While considering most sorting algorithms require O(N log N) or $O(N^2)$, this algorithm require O(K*N) when K is number of characters in keys; thus, it requires only O(N). Which is good. Although Radix Sort (currently one O(N) method) requires O(N) time complexity, the aforementioned method is better in memory space saving and parallel processing. Hence it is recommendable.

Written by Dong-Keun Shin on July 4, 1998.