

Title: Implementation of a dictionary using binary search tree for sorting of searching.

Problem statement: A dictionary Stone keywords & its meanings provide facility for adding new keywords updating values of any entry ascending/ Descending order. Also find how many maximum comparisons may require for finding any keyward. Use binary search tree for implementation,

objectives: To implement a dictionary using binary search tree for sorting & searching.

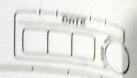
software Requirements:

Theory: A binarry Search tree (BST) is a binarry tree data structure where each node how at most two children & the values of left child are less than the parent of the value of the right child are greter than present & values of no The BST property allows for efficient searching.

inserting & deleting of hodes. In the case of dictionary implementation the keywords would be stored as the key; values and their concesponding meaning as the data values To implement the required

functionalities we can use the following operations.

I Moling a new keywords; To odd a new keyword we stant at the Free and compare the keyword with value of the current . If the keyword is less than the value of the node we more to the left child. If the keyword is greater more for right child. If we reach a null child nade for educational use



we insert the new keyword of its meaning as trans expeleting keywords.

skupdating values of any entry

41 Displaying whole data stored in arending decerding st finding the maximum comparisons required for

finding any keyword.

The maximum competitions required for finding any in ha BST is equal to the height of the tree A bounted has a height of log(n) where n is the numbers of note

- There are different types of BST's that can be used dependent on the specific requirements of the application some of the Common types include AVI Trees. Red Black tres ? 200 thee to implement the required functionalities we can us the following operations.

is Adding a new keywords: To add a new keyword? war start at the root of thee and compare the buys with the value of the current node. If the keyword? less than the value of the node we have to the less child. If the keyword is greater more to night child. If we reach a hull child node, we insert the real

2) Define an empty tree

3) Define function insert (1 to add a new keyword 4) Define function delete () to deleting a keyword.

5) Define a function update() to updating the value of

of Define functions as inorder transversal () to displaying the data in ascending order.

Face in			-	
FIA	FF	-1	1	
DA	16	1		
Commercial		1	- 1	

petine the function as max comparisions () for finding the maximum Companisions required for finding any so stop.

In General BSTs are efficient for search of inserting element in a sorted Collection. However the worst case time complexity, of a(n) can occur when the BST is unbalance leading to degraded performance to avoid this, it is important to balance the BST after every insertion of deletion aperation.

We can use a BST to implement a dictionary with efficient add deleted update operations of the ability to display the data in sorted order. The maximum companisions required for finding any keyword is equal to the height of the free which is log(n) for the balanced BST.

SU.

June