PES University
Department of Computer Science & Engineering
Data Structures and its Applications (UE19CS202)
UNIT 3
Trees and Heaps

Question and Answer

3) How many ancestors does a node at level n in a binary tree have? Prove your answer.

Solution:

In binary tree there are n ancestors at level n.

Proof :

Take P(0): At node level 0, it has no ancestors since this is a root node.

Take P(1): At node level 1, it has one ancestor. The ancestor is the root, its parents which is at level 0.

Take P(K): A node level K it has K ancestors. Its parent is at level K – 1.

Take P(K+1): At node K + 1 level have more than one ancestor than that of node at k level.

Thus there are n ancestors at level n in binary tree.

9) Two binary trees are similar if they are both empty or if they are both nonempty, their left subtrees are similar, and their right subtrees are similar. Write an algorithm to determine if two binary trees are similar.

Solution:

sameTree(tree1, tree2)

1. If both trees are empty then return 1.
2. Else If both trees are non -empty
   (a) Check data of the root nodes (tree1->data == tree2->data)
   (b) Check left subtrees recursively i.e., call sameTree(
   tree1->left_subtree, tree2->left_subtree)
   (c) Check right subtrees recursively i.e., call sameTree(
   tree1->right_subtree, tree2->right_subtree)
   (d) If a,b and c are true then return 1.
3. Else return 0 (one is empty and other is not)

21) Write C function to create a binary tree given the preorder and inorder traversals of that tree. The function should accept two character strings as parameters. The tree created should contain a single character in each node.

Solution:

/* A binary tree node has data, pointer to left child and a pointer to right child */

```
struct node {
	char data;
	struct node* left;
	struct node* right;
};

/* Prototypes for utility functions */
int search(char arr[], int strt, int end, char value);
struct node* newNode(char data);

/* Recursive function to construct binary of size len from Inorder traversal in[] and Preorder
traversal pre[]. Initial values of inStrt and inEnd should be 0 and len -1. The function doesn't
do any error checking for cases where inorder and preorder do not form a tree */
struct node* buildTree(char in[], char pre[], int inStrt, int inEnd)
{
	static int preIndex = 0;

	if (inStrt > inEnd)
		return NULL;

	/* Pick current node from Preorder traversal using preIndex  and increment preIndex
*/
	struct node* tNode = newNode(pre[preIndex++]);

	/* If this node has no children then return */
	if (inStrt == inEnd)
		return tNode;

	/* Else find the index of this node in Inorder traversal */
	int inIndex = search(in, inStrt, inEnd, tNode->data);

	/* Using index in Inorder traversal, construct left and right subtress */
	tNode->left = buildTree(in, pre, inStrt, inIndex - 1);
	tNode->right = buildTree(in, pre, inIndex + 1, inEnd);

	return tNode;
}

/* UTILITY FUNCTIONS */
/* Function to find index of value in arr[start...end]
The function assumes that value is present in in[] */
```

```
int search(char arr[], int strt, int end, char value)
{
	int i;
	for (i = strt; i <= end; i++) {
		if (arr[i] == value)
			return i;
	}
}

/* Helper function that allocates a new node with the given data and NULL left and right
pointers. */
struct node* newNode(char data)
{
	struct node* node = (struct node*)malloc(sizeof(struct node));
	node->data = data;
	node->left = NULL;
	node->right = NULL;

	return (node);
}

/* This funtcion is here just to test buildTree() */
void printInorder(struct node* node)
{
	if (node == NULL)
		return;

	/* first recur on left child */
	printInorder(node->left);

	/* then print the data of node */
	printf("%c ", node->data);

	/* now recur on right child */
	printInorder(node->right);
}
```