Practical Assignment

Data Structures and Algorithms Lab (CS 253)

October 19, 2022

Lab Assignment VIII

Write a C++ program to:

- Build a tree from a given in-order and pre-order traversal. Also Display the tree with graphviz tools. Please note that your C program should generate .gv file which will be used to create jpeg file using Graphviz tools.

Source: https://graphviz.org/

Example code: Example2.gv

```
graph {
A -- B
A -- C
B -- D
B -- E
C -- F
}
```

Command to use Graphviz tool:

For Windows:

- >> <path to dot.exe> -Tjpg <path to .gv file> -o <path to jpeg file>
- C:\Users\Anuj>"C:\Program Files (x86)\Graphviz\bin\dot.exe" -Tjpg "C:\Users\Anuj\Desktop\203\203.gv" -o "C:\Users\Anuj\Desktop\203\205.jpg"

```
/* createTree.c */
/* program to construct tree using
inorder and preorder
traversals */
#include<iostream>
#define MAX_NO_NODES_TREE 100
/* 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);
struct node* buildTree(char in[], char pre[], int
inStrt, int inEnd);
void printInorder(struct node* node);
void writeNodeInGVFile(struct node* node);
void createGVFile(struct node* node);


/* 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;
}
}
```

```
/* 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 */
std::cout << node->data;
/* now recur on right child */
printInorder(node->right);
}
/* 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 node in GV file */
void writeNodeInGVFile(struct node* node) {
if (node == NULL)
return;
/* then print the data of node */
if(node->left != NULL)
std::cout   << node->data << " -- " << node->left-
>data << "\n";
if(node->right != NULL)
std::cout  <<  node->data << " -- " <<  node->right-
>data << "\n";
/* first recur on left child */
writeNodeInGVFile(node->left);
/* now recur on right child */
writeNodeInGVFile(node->right);
}
/* This funtcion is here just to create GV file*/
void createGVFile(struct node* node) {
std::cout << "\ngraph {\n" ;
writeNodeInGVFile(node);
std::cout << "}\n" ;
}
```

```
/* 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;
}
```

```
/* Driver program to test above functions */
int main(){
//char inOrder[MAX_NO_NODES_TREE];
//char preOrder[MAX_NO_NODES_TREE];
char in[] = { 'D', 'B', 'E', 'A', 'F', 'C','G' };
char pre[] = { 'A', 'B', 'D', 'E', 'C', 'F','G' };
int len = sizeof(in) / sizeof(in[0]);
struct node* root = buildTree(in, pre, 0, len - 1);
/* Let us test the built tree by printing Insorder traversal */
std::cout << "Inorder traversal of the constructed tree is \n";
printInorder(root);
/* Let us Create GV file for the tree */
createGVFile(root);
}
```

Lab Assignment VIII

A graph representation is given by Adjacency Matrix. Write a C++ program to:

- Convert it into Adjacency List Representation and display the adjacency list and graph using Graphviz.
- Implement Depth First Search (DFS) and Breadth First Search (BFS) algorithms for a graph that is represented by an Adjacency Matrix. (*use an appropriate data structure that has been implemented earlier and consider the number of vertex greater than eight*)

Lab Assignment VIII

Write a C++ program to:

- Find a minimum cost-spanning tree using prim's and kruskal's algorithms for a graph represented with an adjacency matrix.