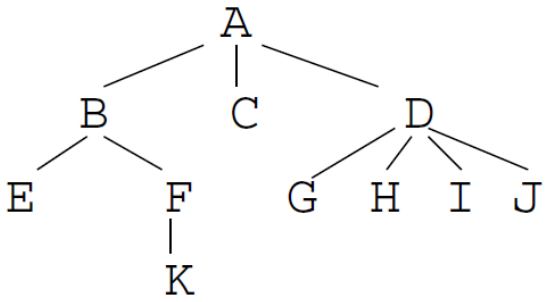
Serialize and Deserialize an N-ary Tree

Given an N-ary tree where every node has at-most N children. How to serialize and deserialze it? Serialization is to store tree in a file so that it can be later restored. The structure of tree must be maintained. Deserialization is reading tree back from file.

This post is mainly an extension of below post.
Serialize and Deserialize a Binary Tree

In an N-ary tree, there are no designated left and right children. An N-ary tree is represented by storing an array or list of child pointers with every node.
The idea is to store an 'end of children' marker with every node. The following diagram shows serialization where ')' is used as end of children marker. The diagram is taken from here.



ABE) FK))) C) DG) H) I) J)))

Following is C++ implementation of above idea.

```
// A C++ Program serialize and deserialize an N-ary tree
#include<cstdio>
#define MARKER ')'
#define N 5
using namespace std;

// A node of N-ary tree
struct Node {
   char key;
   Node *child[N];  // An array of pointers for N children
};
```

```
// A utility function to create a new N-ary tree node
Node *newNode(char key)
{
    Node *temp = new Node;
    temp->key = key;
    for (int i = 0; i < N; i++)
        temp->child[i] = NULL;
    return temp;
}

// This function stores the given N-ary tree in a file pointed by fp
void serialize(Node *root, FILE *fp)
{
    // Base case
    if (root == NULL) return;

    // Else, store current node and recur for its children
    fprintf(fp, "%c ", root->key);
    for (int i = 0; i < N && root->child[i]; i++)
         serialize(root->child[i],  fp);

    // Store marker at the end of children
    fprintf(fp, "%c ", MARKER);
}

// This function constructs N-ary tree from a file pointed by 'fp'.
// This functionr returns 0 to indicate that the next item is a valid
// tree key. Else returns 0
int deSerialize(Node *&root, FILE *fp)
{
    // Read next item from file. If theere are no more items or next
    // item is marker, then return 1 to indicate same
    char val;
    if ( !fscanf(fp, "%c ", &val) || val == MARKER )
       return 1;

    // Else create node with this item and recur for children
    root = newNode(val);
    for (int i = 0; i < N; i++)
      if (deSerialize(root->child[i], fp))
         break;

    // Finally return 0 for successful finish
    return 0;
}

// A utility function to create a dummy tree shown in above diagram
Node *createDummyTree()
{
    Node *root = newNode('A');
    root->child[0] = newNode('B');
    root->child[1] = newNode('C');
    root->child[2] = newNode('D');
    root->child[0]->child[0] = newNode('E');
    root->child[0]->child[1] = newNode('F');
    root->child[2]->child[0] = newNode('G');
    root->child[2]->child[1] = newNode('H');
    root->child[2]->child[2] = newNode('I');
    root->child[2]->child[3] = newNode('J');
    root->child[0]->child[1]->child[0] = newNode('K');
    return root;
}

// A utlity function to traverse the constructed N-ary tree
void traverse(Node *root)
```

```
{
    if (root)
    {
        printf("%c ", root->key);
        for (int i = 0; i < N; i++)
            traverse(root->child[i]);
    }
}

// Driver program to test above functions
int main()
{
    // Let us create an N-ary tree shown in above diagram
    Node *root = createDummyTree();

    // Let us open a file and serialize the tree into the file
    FILE *fp = fopen("tree.txt", "w");
    if (fp == NULL)
    {
        puts("Could not open file");
        return 0;
    }
    serialize(root, fp);
    fclose(fp);

    // Let us deserialize the storeed tree into root1
    Node *root1 = NULL;
    fp = fopen("tree.txt", "r");
    deSerialize(root1, fp);

    printf("Constructed N-Ary Tree from file is \n");
    traverse(root1);

    return 0;
}
```

Run on IDE

Output:

```
Constructed N-Ary Tree from file is
A B E F K C D G H I J
```

The above implementation can be optimized in many ways for example by using a vector in place of array of pointers. We have kept it this way to keep it simple to read and understand.

This article is contributed by **varun**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

11 Comments Category: Trees

Related Posts:

- Check sum of Covered and Uncovered nodes of Binary Tree
- Lowest Common Ancestor in a Binary Tree | Set 2 (Using Parent Pointer)
- Construct a Binary Search Tree from given postorder
- BFS vs DFS for Binary Tree
- Maximum difference between node and its ancestor in Binary Tree
- Inorder Non-threaded Binary Tree Traversal without Recursion or Stack
- Check if leaf traversal of two Binary Trees is same?
- Closest leaf to a given node in Binary Tree

(Login to Rate and Mark)

4.4 Average Difficulty : **4.4/5.0**
Based on **5** vote(s)

Add to TODO List

Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.