GeeksQuiz

Computer science mock tests for geeks

Binary Tree | Set 1 (Introduction)

Trees: Unlike Arrays, Linked Lists, Stack and queues, which are linear data structures, trees are hierarchical data structures.

Tree Vocabulary: The topmost node is called root of the tree. The elements that are directly under an element are called its children. The element directly above something is called its parent. For example, a is a child of f and f is the parent of a. Finally, elements with no children are called leaves.

```
      tree
      ----
       j    <-- root
     /   \
    f      k
  /   \      \
 a     h      z    <-- leaves
```

Why Trees?

1. One reason to use trees might be because you want to store information that naturally forms a hierarchy. For example, the file system on a computer:

```
file system
-----------
     /    <-- root
  /      \
...       home
      /          \
   ugrad        course
    /       /      |     \
  ...      cs101  cs112  cs113
```

2. Trees (with some ordering e.g., BST) provide moderate access/search (quicker than Linked List and slower than arrays).
3. Trees provide moderate insertion/deletion (quicker than Arrays and slower than Unordered Linked Lists).
4. Like Linked Lists and unlike Arrays, Trees don't have an upper limit on number of nodes as nodes are linked using pointers.

Main applications of trees include:
1. Manipulate hierarchical data.
2. Make information easy to search (see tree traversal).
3. Manipulate sorted lists of data.
4. As a workflow for compositing digital images for visual effects.
5. Router algorithms
6. Form of a multi-stage decision-making (see business chess).

Binary Tree: A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

Binary Tree Representation in C: A tree is represented by a pointer to the topmost node in tree. If the tree is empty, then value of root is NULL.
A Tree node contains following parts.
1. Data
2. Pointer to left child
3. Pointer to right child

In C, we can represent a tree node using structures. Below is an example of a tree node with an integer data.

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

First Simple Tree in C
Let us create a simple tree with 4 nodes in C. The created tree would be as following.

```
        tree
        ----
         1    <-- root
       /   \
      2     3
     /
```

```
    4
```

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

/* newNode() allocates a new node with the given data and NULL left and
   right pointers. */
struct node* newNode(int data)
{
  // Allocate memory for new node
  struct node* node = (struct node*)malloc(sizeof(struct node));

  // Assign data to this node
  node->data = data;

  // Initialize left and right children as NULL
  node->left = NULL;
  node->right = NULL;
  return(node);
}


int main()
{
  /*create root*/
  struct node *root = newNode(1);
  /* following is the tree after above statement

        1
      /   \
     NULL  NULL
  */


  root->left        = newNode(2);
  root->right       = newNode(3);
  /* 2 and 3 become left and right children of 1
           1
         /   \
        2      3
      /    \    /  \
    NULL NULL NULL NULL
  */


  root->left->left  = newNode(4);
  /* 4 becomes left child of 2
           1
       /       \
      2          3
    /   \       /  \
   4    NULL  NULL  NULL
  /  \
NULL NULL
*/

  getchar();
  return 0;
```

```
}
```

Summary: Tree is a hierarchical data structure. Main uses of trees include maintaining hierarchical data, providing moderate access and insert/delete operations. Binary trees are special cases of tree where every node has at most two children.

Below are set 2 and set 3 of this post.
Properties of Binary Tree
Types of Binary Tree

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Category: Data Structures

Nitin verma • a year ago
Can you provide a link to the subsequent articles in every section, rather than going back to the page and clicking the next topic. Just a small suggestion. And, the articles are great, easy to understand & grab.
48 • Reply • Share ›

ghost Nitin verma • 9 months ago
Yes Nitin, I was looking for the same.
@GeeksforGeeks team, please provide link for next article, so users need not to go back on main page just to click for next article.
3 • Reply • Share ›

skkar.2k2 Nitin verma • 3 months ago
I was looking for the link for next but after seeing this I realized it's missing.

1 • Reply • Share ›

Holden Nitin verma • a month ago
I need second part ... but I couldn't find it :(
• Reply • Share ›

code_dweller Nitin verma • 10 months ago
I second!
• Reply • Share ›

CodeMe • a year ago
Could you please add few illustrations to the five applications of trees that you had mentioned ?

1. Manipulate hierarchical data.
2. Make information easy to search (see tree traversal).
3. Manipulate sorted lists of data.
4. As a workflow for compositing digital images for visual effects.
5. Router algorithms
6. Form of a multi-stage decision-making (see business chess).

16 • Reply • Share ›

Tamil mani • a year ago
i too want what nitin verma wants...
6 • Reply • Share ›

Anonim • 3 months ago
I supposed that "(...)elements with no children are called leaves"
should be changed to leafs. Leafs (elements with no children in tree)
instead of leaves.
1 • Reply • Share ›

Sid • 6 months ago
Why we assigned data type- "struct node" to left and right pointers.. What it does
2. Can we use only structure name ie node instead of "struct node" above in tree node representation.
1 • Reply • Share ›

neeraj kumar Sid • 6 months ago
1. This is called self-referential structure. A let or right pointer will be pointing to the a similar struct block as the root is, and hence "struct node" type has been used.
2. no, you can't. That would result in an error, the struct tag is not complete until it is closed. Use typedef further to minimize the code, but modifying struct node would not be recommended.

1 • Reply • Share ›

Piyush Sinha ➔ Sid • 4 months ago
u can use typedef to avoid repeated use of struct.
• Reply • Share ›

Ankit Singh • a year ago
Why is the getchar() function at the end of main()?
Please reply.
1 • Reply • Share ›

GeeksforGeeks Mod ➔ Ankit Singh • a year ago
getchar() is not mandatory here. In some IDEs like Turbo C, Dev CPP, etc, the output screen doesn't stay after running the program. getchar() is used to hold the output screen.
4 • Reply • Share ›

Ankit Singh ➔ GeeksforGeeks • a year ago
Thank you Team.
• Reply • Share ›

BeautifulCode • 3 days ago
In trees deletion /insertion :

It is about finding the node to be deleted, where the complexity lies NOT in the operations for pointer swapping etc.

To find the node to be deleted in Binay tree would be O(logN) whereas, in linked list to find the node to be deleted it is O(N).

Hence BinaryTRee deletions would work FASTER than unordered linked list deletions in general.

GeeksforGeeks team : Please verify the facts before you post them.
• Reply • Share ›

sumit kapoor • 16 days ago
How is searching in tree is slower than array as in array it takes O(n) to search the elements in worst case of linear search and tree with some ordering liske BST take O(nlogn) to search the elements so If you compare the running time for both trees and array the running time of trees is faster than array in terms of searching so how it is given above that running time of arrays is faster than trees in terms of searching
• Reply • Share ›

BeautifulCode sumit kapoor • 4 days ago

In worst case array takes O(n) time to search an element .but AN ORDERED TREE(eg.BST) will take log(N) time

and log(N)

• Reply • Share ›

jerky sumit kapoor • 15 days ago

it is O(log N) in case of BST so it is better than unsorted array where it is O(n).

• Reply • Share ›

Komal Singh • a month ago

Can anyone provide the code which takes node value and position of the node w.r.t root from user. I tried this code but it's giving wrong output:

```
void insertNode(struct node **root_ref,int val){

if(*root_ref==NULL){

struct node *new_node=(struct node *)malloc(sizeof(struct node));

new_node->value=val;

new_node->left=NULL;

new_node->right=NULL;

*root_ref=new_node;

return;

}
```

see more

• Reply • Share ›

ChandraSheker Shivvolla • a month ago

Thank you good information.....

• Reply • Share ›

Shubham Aggarwal • 4 months ago

why tree access is slower than an array?

• Reply • Share ›

BeautifulCode Shubham Aggarwal • 4 days ago

In worst case array takes O(n) time to search an element .but AN ORDERED TREE(eg.BST) will take log(N) time

TREE(eg.BST) will take log(N) time

and log(N) <o(n) in="" value.="">

• Reply • Share ›

Avizz • 9 months ago

How will you deallocate the memory allocated through malloc ??

• Reply • Share ›

Shamil Choudhury Avizz • 7 months ago

To deallocate the memory dynamically, you need to use the free() function and provide the pointer (in which you have kept the reference of the allocated memory) as the argument to the function.

for example, here here the reference of the allocated memory is kept in the pointer node, so to deallocate the memory, you write,

free(node); //before the getchar()

1 • Reply • Share ›

MAHAshwetha rao • 10 months ago

Articles are all good in this page and gives topic wise qs and answers :) Keep it up..It would be great like others mentioned to keep the link of teh next few topics to navigate from current topic.

• Reply • Share ›

Karthik • a year ago

very helpful..thanks a lot..

• Reply • Share ›

Himanshu Dagar • a year ago

Really too good
(thanks)

• Reply • Share ›

Ashraful Islam Rafi • a year ago

Very well written helpful article :)

• Reply • Share ›