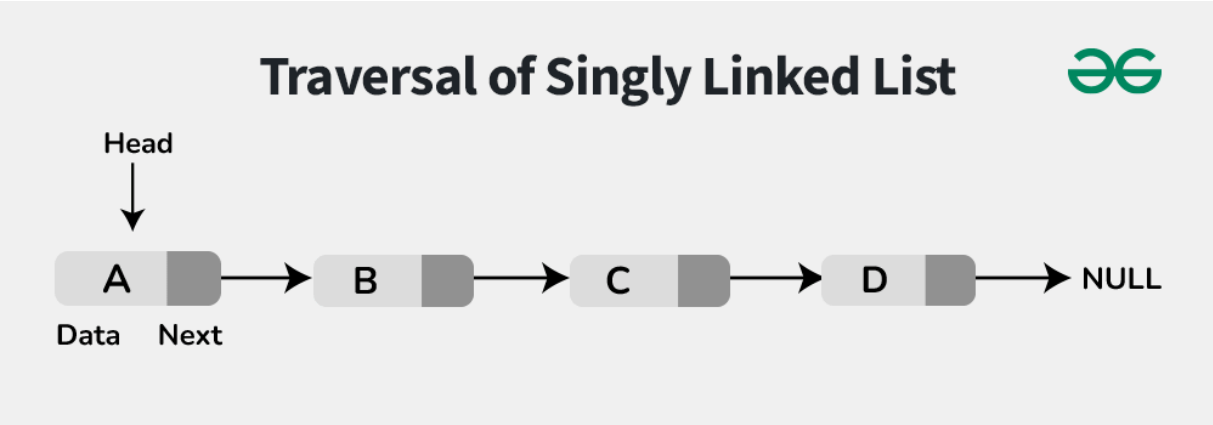
Traversal of Singly Linked List

Last Updated : 18 Feb, 2025

Traversal of Singly Linked List is one of the fundamental operations, where we traverse or visit each node of the linked list. In this article, we will cover how to traverse all the nodes of a singly linked list along with its implementation.



Examples:

Input: *1->2->3->4->5->null*
Output: *1 2 3 4 5*
Explanation: *Every element of each node from head node to last node is printed which means we have traversed each node successfully.*

Input: *10->20->30->40->50->null*
Output: *10 20 30 40 50*
Explanation: *Every element of each node from head node to last node is printed which means we have traversed each node successfully.*

Input: `5->10->15->20->25->null`
Output: `5 10 15 20 25`
Explanation: *Each node's value is printed sequentially from the head to the last node, confirming successful traversal.*

Traversal of Singly Linked List (Iterative Approach)

The process of traversing a singly linked list involves printing the value of each node and then going on to the next node and print that node's value also and so on, till we reach the last node in the singly linked list, whose next node points towards the null.

Step-by-Step Algorithm:

- We will initialize a temporary pointer to the head node of the singly linked list.
- After that, we will check if that pointer is null or not null, if it is null, then return.
- While the pointer is not null, we will access and print the data of the current node, then we move the pointer to next node.

C++ C Java Python C# JavaScript

```cpp
#include <iostream>

using namespace std;

// A linked list node
class Node {
public:
    int data;
    Node* next;

    // Constructor to initialize a new node with data
    Node(int new_data) {
        this->data = new_data;
        this->next = nullptr;
    }
};

// Function to traverse and print the singly linked list
void traverseList(Node* head) {

    // A loop that runs till head is nullptr
    while (head != nullptr) {

        // Printing data of current node
        cout << head->data << " ";

        // Moving to the next node
        head = head->next;
    }
    cout << endl;
}

// Driver Code
int main() {
```

```
    // Create a hard-coded linked list:
    // 10 -> 20 -> 30 -> 40
    Node* head = new Node(10);
    head->next = new Node(20);
    head->next->next = new Node(30);
    head->next->next->next = new Node(40);

    // Example of traversing the node and printing
    traverseList(head);

    return 0;
}
```

```
10 20 30 40
```

Output

```
10 20 30 40
```

Time Complexity: O(n), where n is the number of nodes in the linked list.
Auxiliary Space: O(1)

Traversal of Singly Linked List (Recursive Approach)

We can also traverse the singly linked list using recursion. We start at the head node of the singly linked list, check if it is null or not and print its value. We then call the traversal function again with the next node passed as pointer.

Step-by-Step Algorithm:

- Firstly, we define a recursive method to traverse the singly linked list, which takes a node as a parameter.
- In this function, the base case is that if the node is null then we will return from the recursive method.
- We then pass the head node as the parameter to this function.
- After that, we access and print the data of the current node.
- At last, we will make a recursive call to this function with the next node as the parameter.

C++ C Java Python C# JavaScript

```cpp
#include <iostream>

using namespace std;

// A linked list node
class Node {
public:
    int data;
    Node* next;

    // Constructor to initialize a new node with data
    Node(int new_data) {
        this->data = new_data;
        this->next = nullptr;
    }
};

// Function to traverse and print the singly linked list
void traverseList(Node* head) {

    // Base condition is when the head is nullptr
    if (head == nullptr) {
        cout << endl;
        return;
    }

    // Printing the current node data
    cout << head->data << " ";

    // Moving to the next node
    traverseList(head->next);
}

// Driver code
int main() {

    // Create a hard-coded linked list:
    // 10 -> 20 -> 30 -> 40
    Node* head = new Node(10);
    head->next = new Node(20);
    head->next->next = new Node(30);
    head->next->next->next = new Node(40);

    // Example of traversing the node and printing
```

```
    traverseList(head);

    return 0;
}
```

Output

```
10 20 30 40
```

Time Complexity: O(n), where n is number of nodes in the linked list.
Auxiliary Space: O(n) because of recursive stack space.

Next Article
Singly Linked List Problems

Similar Reads

Singly Linked List Tutorial

A singly linked list is a fundamental data structure, it consists of nodes where each node contains a data field and a reference to the next node in the linked list. The next of the last node is null, indicating the en...

8 min read

Traversal of Circular Linked List

Given a circular linked list, the task is to print all the elements of this circular linked list.Example:Input: Output: 1 2 3 4 5 6Input: Output: 2 4 6 8 10 12Table of Content[Expected Approach - 1] Using Recursion...

8 min read

Types of Linked List

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. In simple words, a linked list consists of...

15+ min read

Singly Linked List Problems

Singly linked list is a linear data structure in which the elements are not stored in contiguous memory locations and each element is connected only to its next element using a pointer.Learn Basics of Singly...

3 min read

Singly Linked List in Python

AÂ Singly Linked ListÂ is a type of data structure that is made up of nodes that are created using self-referential structures. Each node contains a data element and a reference (link) to the next node in the...

10 min read

Traversal in Doubly Linked List

Traversal of Doubly Linked List is one of the fundamental operations, where we traverse or visit each node of the linked list. In this article, we will cover how to traverse all the nodes of a doubly linked list and its...

15+ min read

Sum of the nodes of a Singly Linked List

Given a singly linked list. The task is to find the sum of nodes of the given linked list. Task is to do A + B + C + D. Examples: Input: 7->6->8->4->1 Output: 26 Sum of nodes: 7 + 6 + 8 + 4 + 1 = 26 Input: 1->7->3-...

12 min read

XOR Linked List - Reversal of a List

Given a XOR linked list, the task is to reverse the XOR linked list. Examples: Input: 4 <â€“> 7 <â€“> 9 <â€“> 7Output: 7 <â€“> 9 <â€“> 7 <â€“> 4Explanation:Reversing the linked list modifies the XOR linked...

12 min read

Reverse a sublist of linked list

Given a linked list and positions m and n. We need to reverse the linked list from position m to n.Examples: Input : linkedlist : 10->20->30->40->50->60->70->NULL , m = 3 and n = 6Output : 10->20->60->50-...

15+ min read

Reverse a Linked List

Given a linked list, the task is to reverse the linked list by changing the links between nodes.Examples: Input: head: 1 -> 2 -> 3 -> 4 -> NULLOutput: head: 4 -> 3 -> 2 -> 1 -> NULLExplanation: Reversed Linke...

15+ min read

Company
About Us
Legal
Privacy Policy
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program

Languages
Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

DSA
Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Science & ML
Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies
HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial
Python Programming Examples
Python Projects
Python Tkinter
Python Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps
Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design
High Level Design
Low Level Design
UML Diagrams

Inteview Preparation
Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process