# Efficient Graph Algorithm Implementations: A Study of Prim's, Kruskal's, BFS, DFS, and Tree Traversals with Linked List, Queue, and Stack Applications


| SHANMUKHA | SATHWIK | SOMA SHEKAR | ARAVIND |
|---|---|---|---|
| 22BDS019 | 22BDS020 | 22BDS022 | 22BDS015 |



**Abstract:**
The report evaluates ten fundamental graph algorithms, including Prim's, Kruskal's, Breadth-First Search, Depth-First Search, and tree traversal methods, to determine their efficiency, complexity, and applicability in solving graph-related problems. It highlights BFS for shortest paths in unweighted graphs, DFS for cycle detection and topological sorting, and examines tree traversal methods' applications


## Linked List:

The code written for linked list is to perform the operation of reversing first N elements using Linked list.

Input given for linked list is

Input: 1 2 3 4 5 6

We are reversing only first three elements. So N =3.
Output:

Linked List before reversing first 3 elements: 1 2 3 4 5 6
Linked List after reversing first 3 elements: 3 2 1 4 5 6

Here only first three (1 2 3) are reversed.

## Queue:

The code written for Queue is to reverse the elements of queue, and add another elements to the said queue and, then revers hem again.

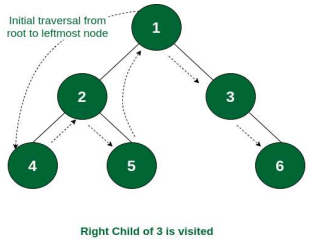
Input: 1 2 3 4 5
Input: 100 200
Output:
Queue elements are:
1 2 3 4 5
Reverse Queue, elements are:
5 4 3 2 1
Add two elements to the said queue:
Queue elements are:
5 4 3 2 1 100 200
Reverse Queue, elements are:
200 100 1 2 3 4 5
Here we first gave an input as 1 2 3 4 5 then we reversed them.
Then we added 100 and 200.Again reversed the entire queue.

## Stacks:

The code written for stacks is to perform the insertion and deletion of elements in Stacks.
Input: 10 20 30 40
We pushed the above values into the stacks.
Output:
Pushed: 10
Pushed: 50
Pushed: 30
Pushed: 40
Stack items:
40 30 50 10
Popped: 40
Popped: 30
Stack items:
50 10
Pushed: 50
Pushed: 60
Stack items:
60 50 50 10

## Tree Traversal:

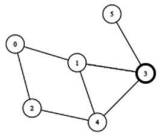
The code written for tree traversal is to perform In-order traversal.
Input:




Output:
In order traversal of binary tree is:
4 2 5 1 3 6
Here we traversed the left node (4) first, then the current node, and finally, the right node.

## Breadth first search (BFS):

The code written for BFS to perform BFS on a binary tree.
Input:

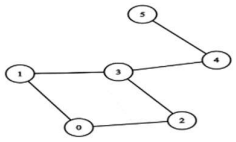


We gave input as shown in the above-mentioned graph.
Output:
BFS Traversal starting from vertex 0:
Visited vertex: 0
Visited vertex: 1

Visited vertex: 2
Visited vertex: 3
Visited vertex: 4
Visited vertex: 5

## Depth First search (DFS):

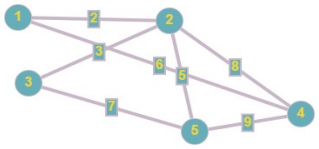
The code written is to perform DFS on a binary tree.
Input:



We gave input as shown in the above-mentioned graph.
Output:
Depth-First Traversal (starting from vertex 0):
Visited vertex: 0
Visited vertex: 2
Visited vertex: 3
Visited vertex: 4
Visited vertex: 5
Visited vertex: 1

## Prim's Algorithm:

The code written for Prim's is to find MST using Prim's algorithm.
Input:



We created a matrix for the above- mentioned graph as follows:
{0, 2, 0, 6, 0},

{2, 0, 3, 8, 5},

{0, 3, 0, 0, 7},

{6, 8, 0, 0, 9},

{0, 5, 7, 9, 0},
Output:

Edge Weight

0 - 1 2

1 - 2 3

0 - 3 6 1 - 4 5

## Kruskal's Algorithm:

The code written is to find MST using Kruskal' Algorithm.
Input:
createEdge(0,1,4)
createEdge(0,2,3)
createEdge(1,2,1)
createEdge(1,3,2)
createEdge(2,3,4)
createEdge(3,4,2)
createEdge(4,5,6)

Output:
Minimum spanning Tree:
1 – 2 : 1
1 – 3 : 2
3 – 4 : 2
0 – 2 : 3
4 – 5 : 6

## Dijkstra's Algorithm:

The code written is to find the shortest path using Dijkstra's algorithm.

Input:
{ 0, 4, 0, 0, 0, 0, 0, 8, 0 },
{ 4, 0, 8, 0, 0, 0, 0, 11, 0 },
{ 0, 8, 0, 7, 0, 4, 0, 0, 2 },
{ 0, 0, 7, 0, 9, 14, 0, 0, 0 },
{ 0, 0, 0, 9, 0, 10, 0, 0, 0 },
{ 0, 0, 4, 14, 10, 0, 2, 0, 0 },
{ 0, 0, 0, 0, 0, 2, 0, 1, 6 },
{ 8, 11, 0, 0, 0, 0, 1, 0, 7 },
{ 0, 0, 2, 0, 0, 0, 6, 7, 0 };

Output:

| Vertex | Distance |
|---|---|
| 0 | 0 |
| 1 | 4 |
| 2 | 12 |
| 3 | 19 |
| 4 | 21 |
| 5 | 11 |
| 6 | 9 |
| 7 | 8 |
| 8 | 14 |

Here distance refers to Distance from source.

## Bellmanford Algorithm:

The code written is to find the shortest path from single source vertex to all other vertices using Bellmanford algorithm.

Input:
(graph, 0, 1, -1);
(graph, 0, 2, 4);
(graph, 1, 2, 3);
(graph, 1, 3, 2);
(graph, 1, 4, 2);
(graph, 3, 2, 5);
(graph, 3, 1, 1);
(graph, 4, 3, -3);

Output:
Shortest Paths from Source Vertex 0:

| Vertex | Distance from Source |
|---|---|
| 0 | 0 |
| 1 | -1 |
| 2 | 2 |
| 3 | -2 |
| 4 | 1 |

# Assignment (Lab 11)

| Question no. | INPUT | OUTPUT |
|---|---|---|
| 1 | 1)shanmukha marks – 85.2<br>2)sathwik marks – 80.5<br>3)somasekhar marks –75.5 | List after deletion:<br>1)shanmukha marks – 85.2<br>2)sathwik marks – 80.5 |
| 2 (a) | List 1: 1 2 3<br>List 2: 4 5 6 | Concatenated List:<br>1 2 3 4 5 6 |
| 2(b) | Elements:<br>20 40 10 50 30 | Sorted list:<br>10 20 30 40 50 |
| 2(c) | Elements:<br><br>{10 20 30} | Deque after insertion at the front: 10 20 30<br>Deleted element from the rear: 30<br>Deque after deletion from the rear: 10 20 |