Breadth First Search

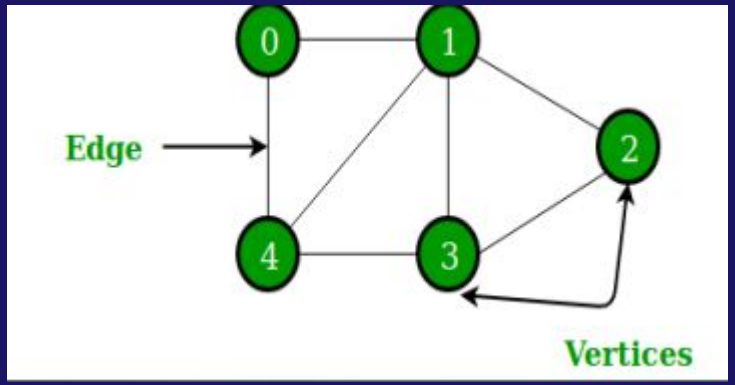
Prepared by Dungeon Team

What is Breadth First Search?

Breadth-first search (BFS) is an algorithm for traversing or searching Tree or Graph data structures. It starts at the tree root or some arbitrary node of a graph, sometimes referred to as a 'search key' and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.

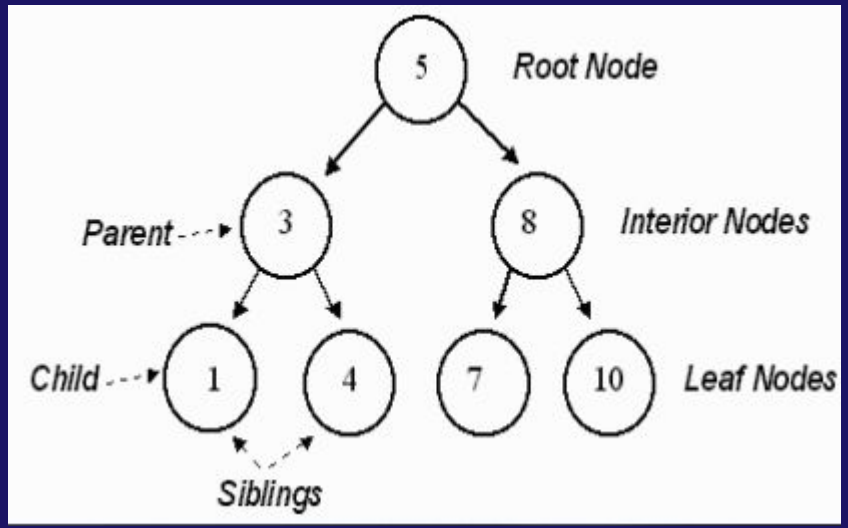
We Should Know

A Graph G=(V, E) consists a set of vertices, V, and a set of edges, E.



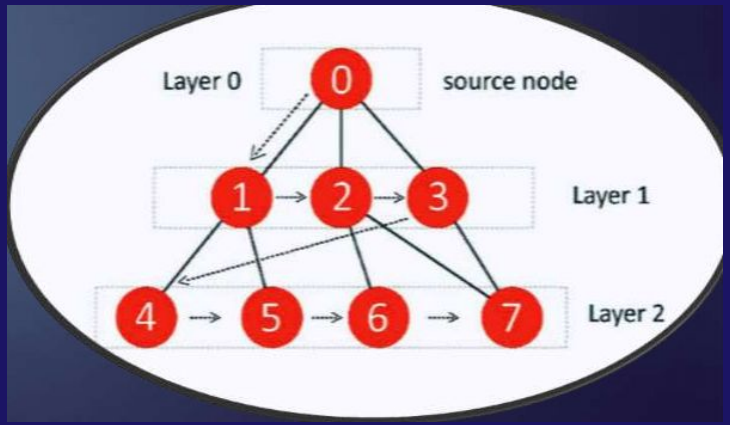
A tree is a collection of entities called nodes. Nodes are connected by edges. Each node contains a value or data, and it may or may not have a child node

Tree is also a Graph.



We Should Know

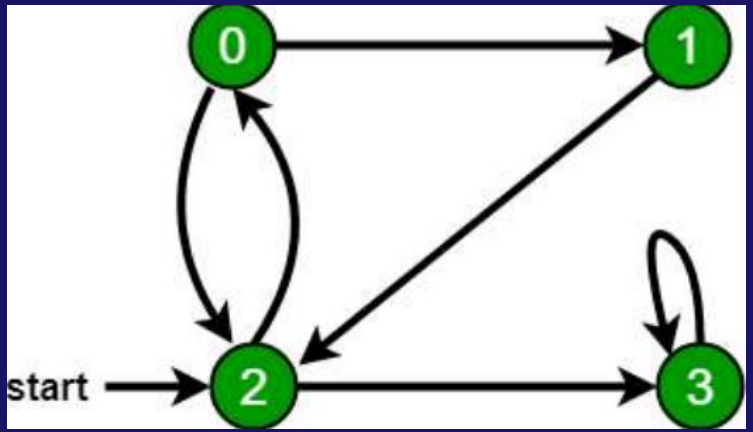
- Visiting a Vertex: It means going on a particular vertex or checking a vertex's value.
- Exploration of Vertex: It means visiting all the adjacent vertices of a selected node.
- Traversing: Traversing means passing through nodes in a specific order
- Level-Order: It is a traversing method, where we have to visit every node on a level before going to a lower level.



- Breadth means the distance or measurement from side to side of something; width; wideness; diameter; range; extent.

- Breadth First Search for a graph is similar to Breadth First Traversal of a tree. The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a Boolean visited array. For simplicity, it is assumed that all vertices are reachable from the starting vertex.

- For example, in the following graph, we start traversal from vertex 2. When we come to vertex 0, we look for all adjacent vertices of it. 2 is also an adjacent vertex of 0. If we don't mark visited vertices, then 2 will be processed again and it will become a non-terminating process. A Breadth First Traversal of the following graph is 2, 0, 3, 1.



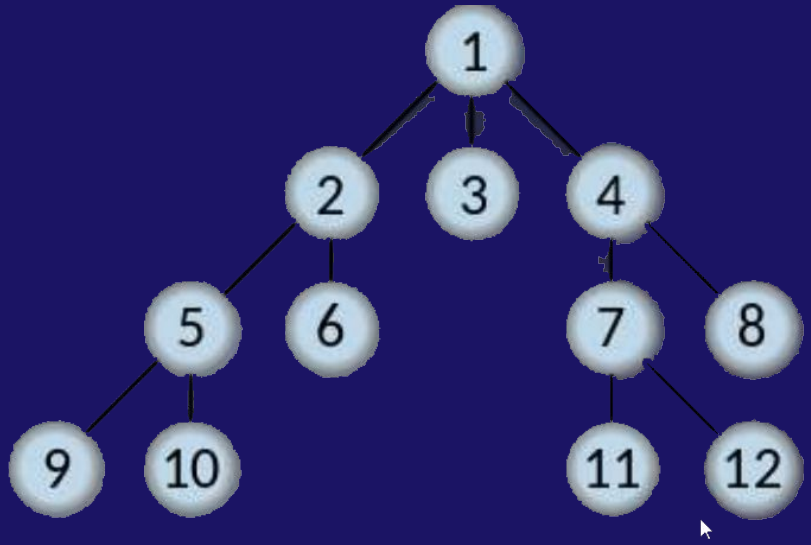
What is Breadth First Search?

Breadth-first search (BFS) is an algorithm for traversing or searching Tree or Graph data structures. It starts at the tree root or some arbitrary node of a graph, sometimes referred to as a 'search key' and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.

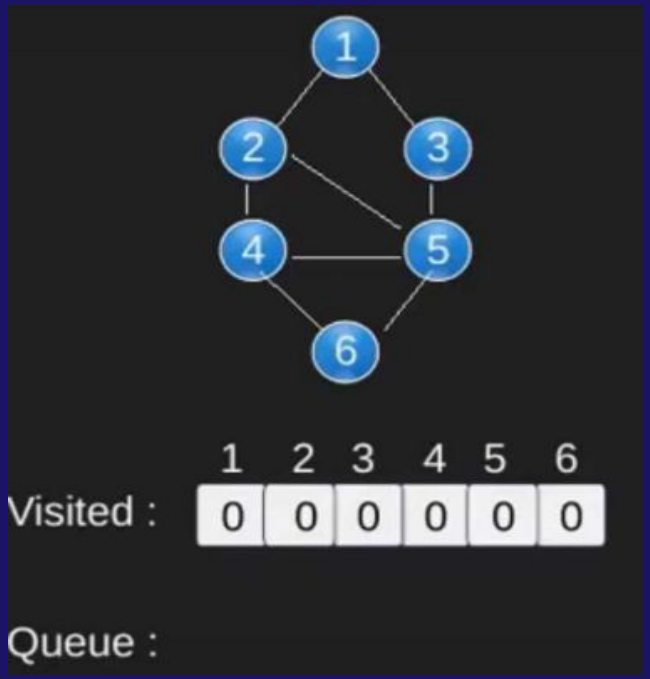
BFS & Level Order Example

Level-Order: 1 234 5678 9101112

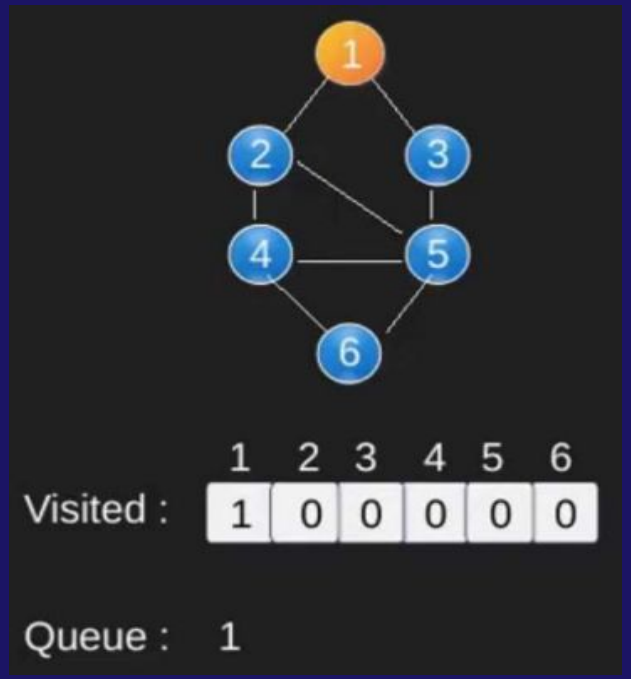
BFS: 1 234 5678 9101112



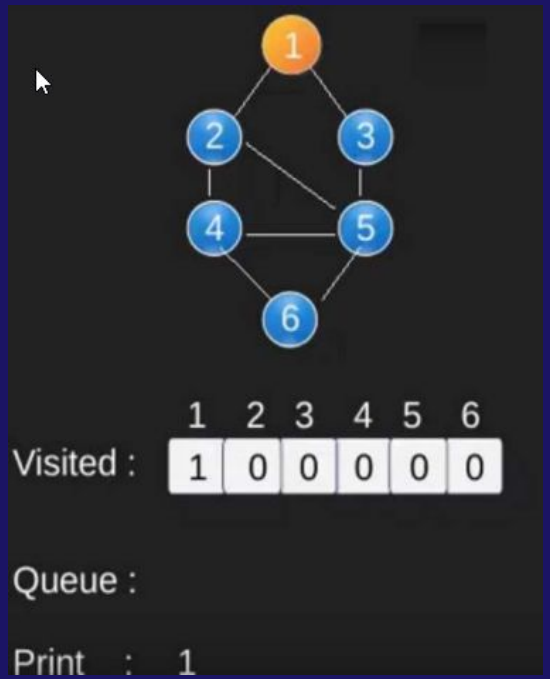
BFS Illustration



Step 1

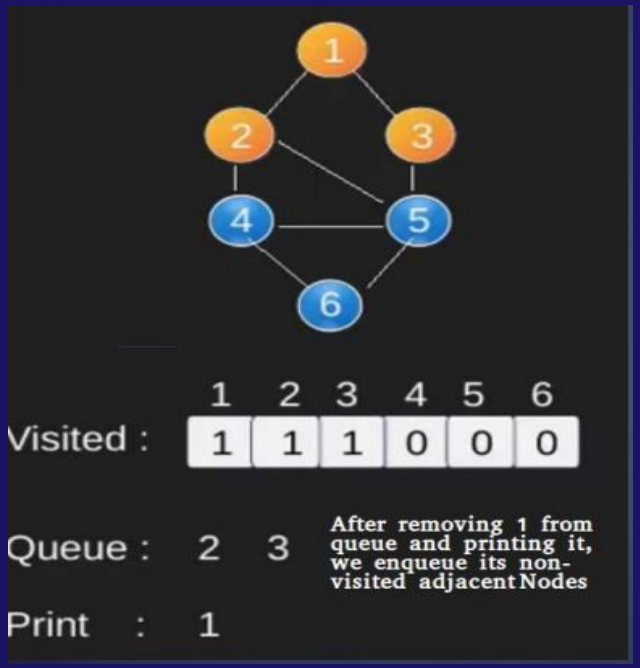


Step 2

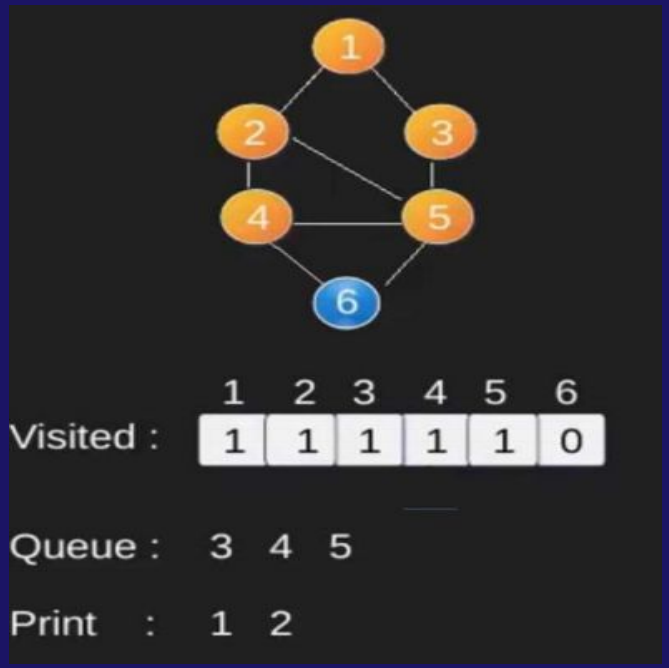


Step 3

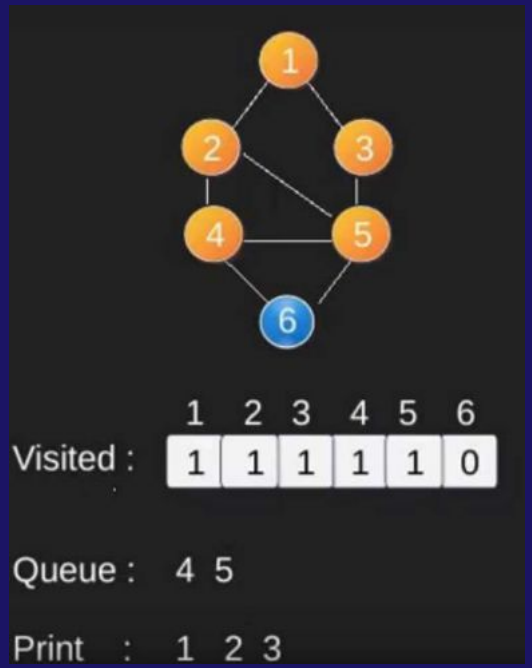
BFS Illustration



Step 4

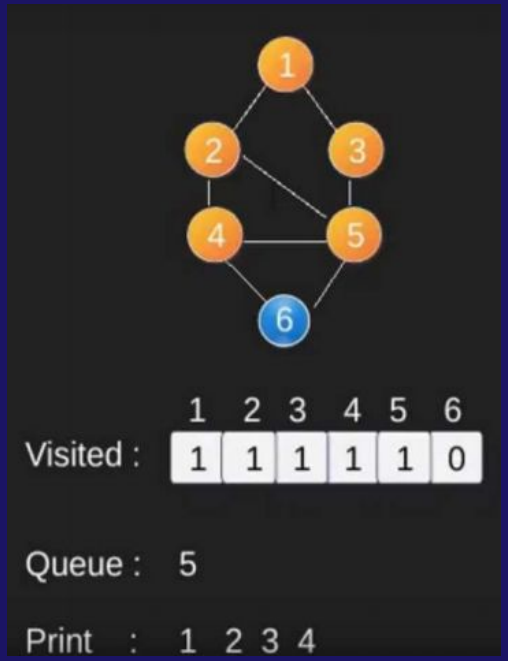


Step 5

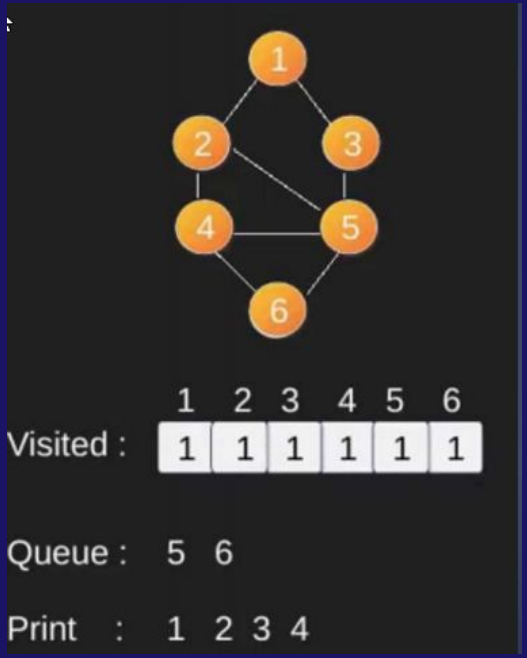


Step 6

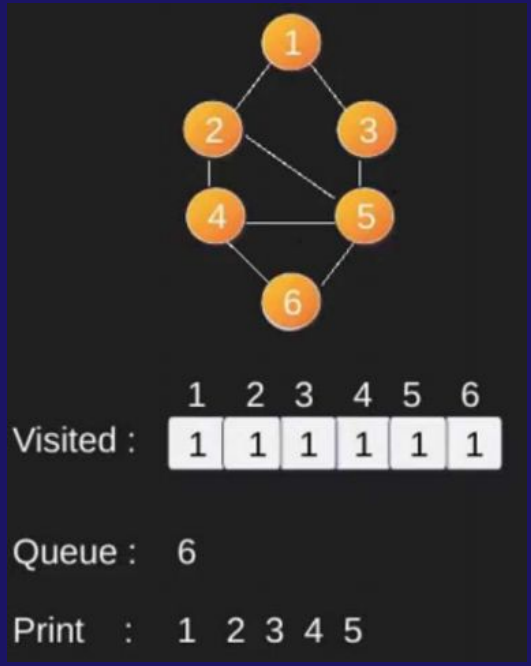
BFS Illustration



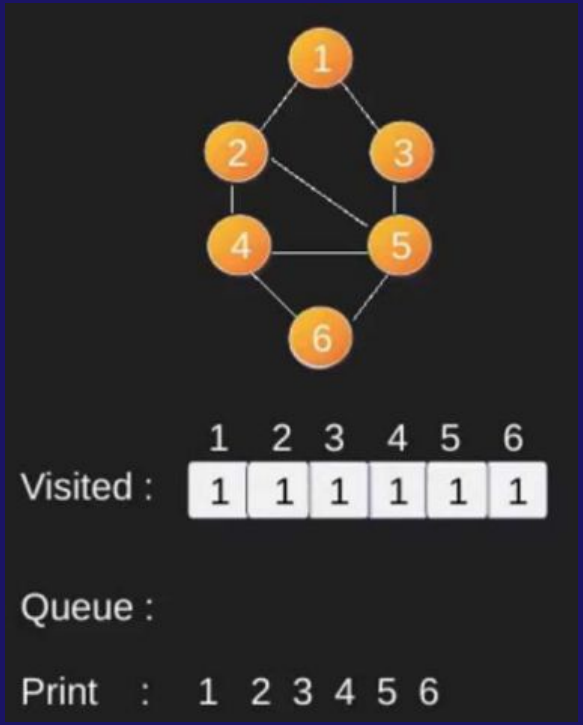
Step 7



Step 8



Step 9



Step 10

* BFS is just like Level Order & it follow 3 simple rules

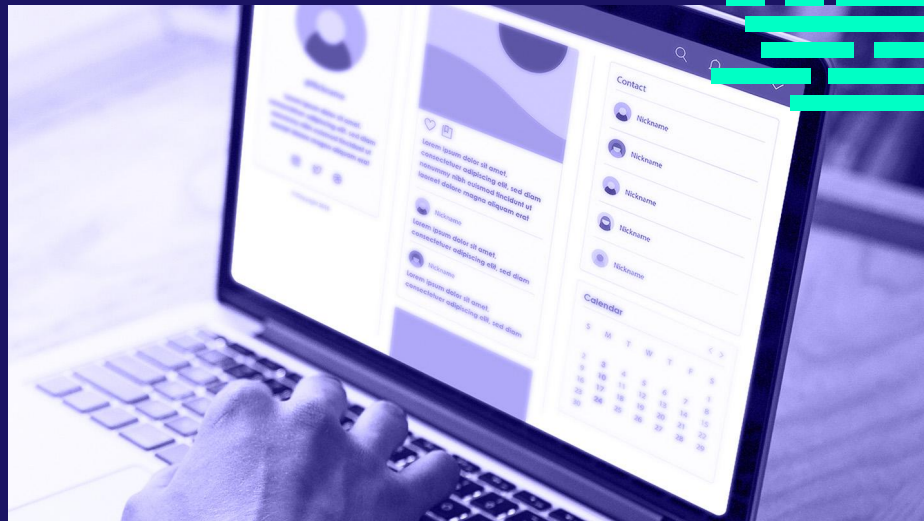
Rule 1 : Visit the adjacent unvisited vertex. Mark it as visited. Insert it in a queue & Display it.

Rule 2 : If no adjacent vertex is found, remove the first vertex from the queue.

Rule 3 : Repeat Rule 1 and Rule 2 until the queue is empty.

Code implementation

Let's look at
examples with code



Coding

```
// Java program to print BFS traversal from a given source vertex.
// BFS(int s) traverses vertices reachable from s.
import java.io.*;
import java.util.*;

// This class represents a directed graph using adjacency list
// representation
class Graph
{
    // No. of vertices
    private int V;
    //Adjacency Lists
    private LinkedList<Integer> adj[];

    // Constructor
    Graph(int v)
    {
        V = v;
        adj = new LinkedList[v];
        for (int i=0; i<v; ++i)
            adj[i] = new LinkedList();
    }

    // Function to add an edge into the graph
    void addEdge(int v,int w)
    {
        adj[v].add(w);
    }
```

Coding

```
void BFS(int s) // prints BFS traversal from a given source s
    {
        boolean visited[] = new boolean[V];

        LinkedList<Integer> queue = new
LinkedList<Integer>();                  // Create a queue for BFS

        visited[s]=true;
        queue.add(s);

        while (queue.size() != 0)
        {
            s = queue.poll();
            System.out.print(s+" ");

            Iterator<Integer> i =
adj[s].listIterator();
            while (i.hasNext())
            {
                int n = i.next();
                if (!visited[n])
                {
                    visited[n] = true;
                    queue.add(n);
                }
            }
        }
    }
```

// Mark all the vertices as not visited(By default // set as false)

// Mark the current node as visited and enqueue it

// Dequeue a vertex from queue and print it

// Mark the current node as visited and enqueue it

// Get all adjacent vertices of the dequeued vertex s
// If a adjacent has not been visited, then mark it
// visited and enqueue it

Coding

```
// Driver method to
public static void main(String args[])
        {
                Graph g = new Graph(4);

                g.addEdge(0, 1);
                g.addEdge(0, 2);
                g.addEdge(1, 2);
                g.addEdge(2, 0);
                g.addEdge(2, 3);
                g.addEdge(3, 3);

                System.out.println("Following is Breadth First
Traversal "+
                                                "(starting from
vertex 2)");

                g.BFS(2);
        }
}
```

Output

```
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
```

Applications of BFS

- Shortest Path and Minimum Spanning Tree for unweighted graph In an unweighted graph, the shortest path is the path with least number of edges. With Breadth First, we always reach a vertex from given source using the minimum number of edges. Also, in case of unweighted graphs, any spanning tree is Minimum Spanning Tree and we can use either Depth or Breadth first traversal for finding a spanning tree.

- Peer to Peer Networks. In Peer to Peer Networks like BitTorrent, Breadth First Search is used to find all neighbor nodes.

- Crawlers in Search Engines: Crawlers build index using Breadth First. The idea is to start from source page and follow all links from source and keep doing same. Depth First Traversal can also be used for crawlers, but the advantage with Breadth First Traversal is, depth or levels of the built tree can be limited.

- Social Networking Websites: In social networks, we can find people within a given distance ‘k’ from a person using Breadth First Search till ‘k’ levels.

Applications of BFS

- GPS Navigation systems: Breadth First Search is used to find all neighboring locations.
- Broadcasting in Network: In networks, a broadcasted packet follows Breadth First Search to reach all nodes.
- In Garbage Collection: Breadth First Search is used in copying garbage collection using Cheney's algorithm. Refer this and for details. Breadth First Search is preferred over Depth First Search because of better locality of reference:
- Cycle detection in undirected graph: In undirected graphs, either Breadth First Search or Depth First Search can be used to detect cycle. We can use BFS to detect cycle in a directed graph also, Ford-Fulkerson algorithm In Ford-Fulkerson algorithm, we can either use Breadth First or Depth First Traversal to find the maximum flow. Breadth First Traversal is preferred as it reduces worst case time complexity to O(VE2).

Thanks for attention

Our team:

Orynbekov Akhmet

Tasmagambetov Aslan

Nurbek Miras

Zhanibekov Niyaz