# GRAPH TRAVERSALS.

# How to traverse the graph we created?

**DEPTH FIRST SEARCH (DFS)**

It is a form of graph traversal in which we start from a node and keep exploring the depth of the graph as far below as we can go. Recursion uses this DFS.

**BREADTH-FIRST DEARCH**

It is a form of graph traversal in which we first explore all the immediate neighbours of a particular node and then move on to explore the other nodes.

# DEPTH FIRST SEARCH

Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

Stacks are used for DFS.

# Important terms for DFS

### ENTRY/EXIT TIMES

These are self explanatory. They'll be very useful when studying about trees.

### TREE EDGES

The edges traversed during the DFS traversal.

### BACK EDGES

The remaining edges. Tree and back edges will be very useful when learning about bridges and articulation points.

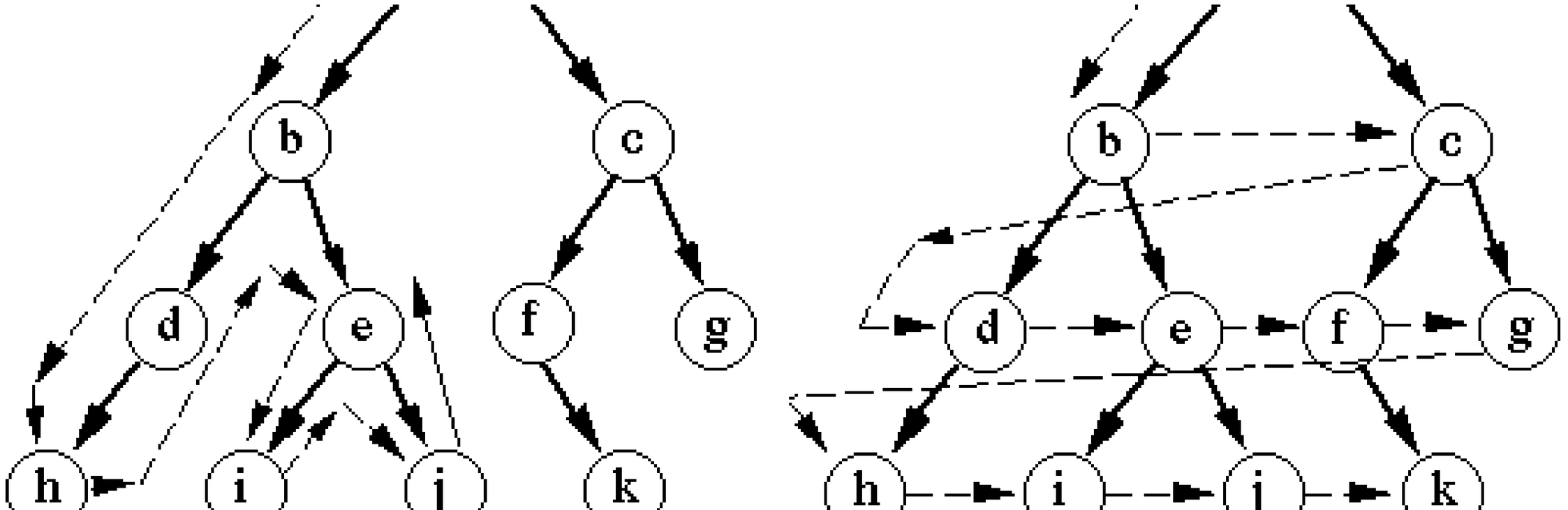
# Breadth First Search

The algorithm can be visualized as a fire originating from some node and spreading throughout the graph using the edges.

Queues are used for the same.

# WHEN TO USE WHAT?

Both the traversal mechanisms have their own advantages and disadvantages. The following set of problems will get you acquainted with what to use where.



# Problem 1

Given an undirected graph determine the minimum distance each node has from the source node.

# Problem 2

Given an undirected graph where each node has a letter associated with it. Determine the lexicographically minimum path of the graph from the root to each of the vertices.

# Problem 3

Given a graph find if it contains any cycles.

# Problem 4

Given a tree determine the minimum distance between the given root and any leaf.

# Problem 5

Given an undirected graph, figure out if it's bi partite.

# Problem 6 (0/1 BFS)

Given a graph such that each edge has either weight of 0 or 1. Find out the minimum distance each node has from the root, where distance refers to the sum of weights of the edges in the path from the root to the node.

# Practice Problems

SPOJ ABCPATH

SPOJ KOZE