CS301
Data Structure and Algorithms
Lecture 10: Graph Representation and Traversal

Pandav Patel
Assistant Professor

Computer Engineering Department
Dharmsinh Desai University
Nadiad, Gujarat, India

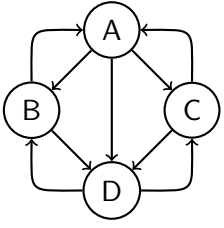
Objective

- To understand representations (storage structures) for graph data structure
- To learn Depth-First-Search (DFS) and Breadth-First-Search (BFS) traversal of graph

Overview

Adjacency Matrix And Adjacency List Representation

- How can we represent graph in memory?
- Two most common representations of graph are *adjacency matrix representation* and *adjacency list representation*



	A	B	C	D
A	0	1	1	1
B	1	0	0	1
C	1	0	0	1
D	0	1	1	0

Table: Adjacency matrix

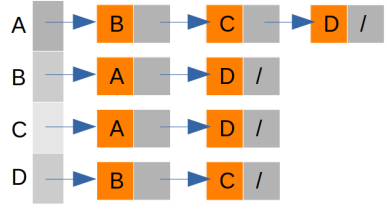


Figure: Adjacency list representation

- Can a graph be represented as list of edges (edge list)?
- Choice of representation depends on algorithm (to be applied on graph) too.

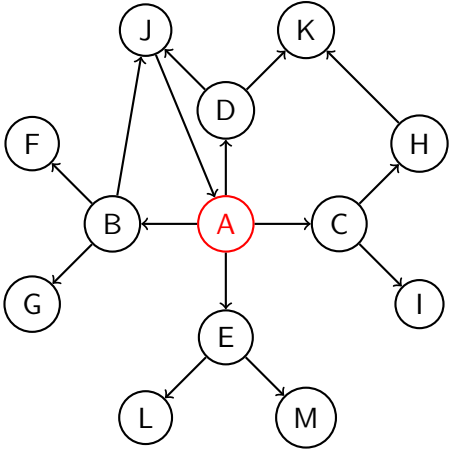
Adjacency Matrix Representation

- Number of 1's is a given row is same as *outdegree* of corresponding node
- Number of 1's is a given column is same as *indegree* of corresponding node
- Adjacency matrix for non-weighted graph is represented by bit matrix (a.k.a. Boolean matrix) as it only contains 0's and 1's
- If nodes are arranged differently (e.g. B, A, C, D) then matrix will look different
- Space complexity is $O(V^2)$, where V is number of vertices in the graph
- Well suited for dense graph
- How can we represent weighted graph with this representation?

Adjacency List Representation

- Can we use vector instead of linked list in adjacency list representation?
- Space complexity is $O(V + E)$, where V is number of vertices in the graph and E is number of edges in the graph
- Well suited for sparse graph
- How can we represent weighted graph with this representation?

Breadth First Search



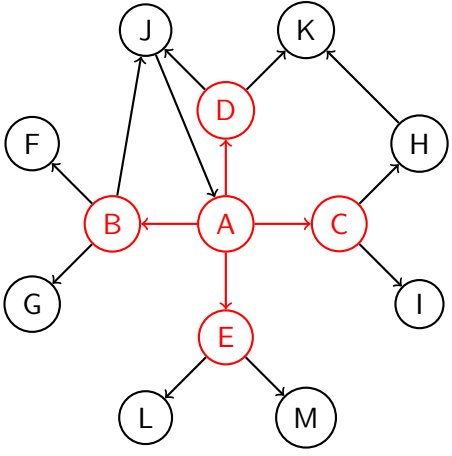
Queue

A

Output

A

Breadth First Search (cont...)



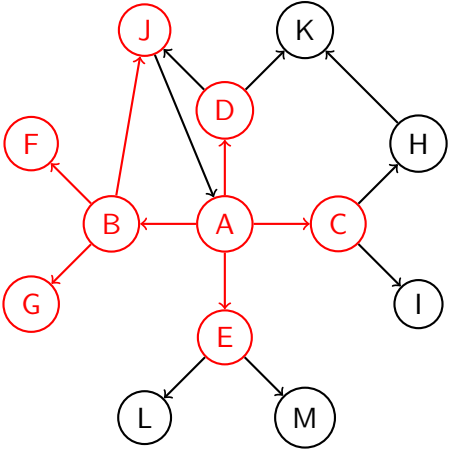
Queue

~~A~~ B C D E

Output

A B C D E

Breadth First Search (cont...)



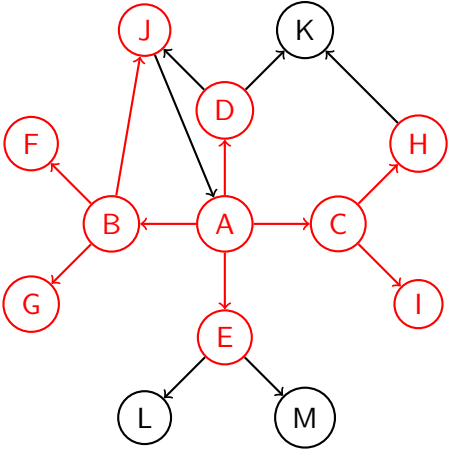
Queue

~~A~~ ~~B~~ C D E F G J

Output

A B C D E F G J

Breadth First Search (cont...)



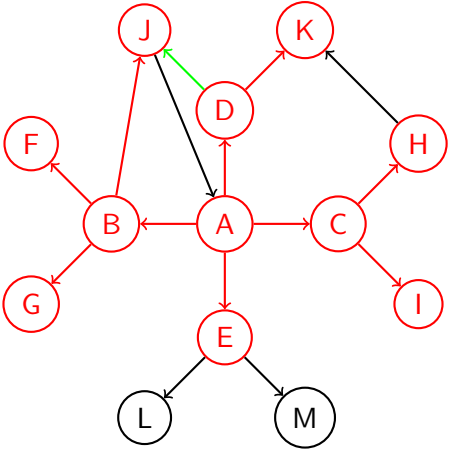
Queue

~~A~~ ~~B~~ ~~C~~ D E F G J H I

Output

A B C D E F G J H I

Breadth First Search (cont...)



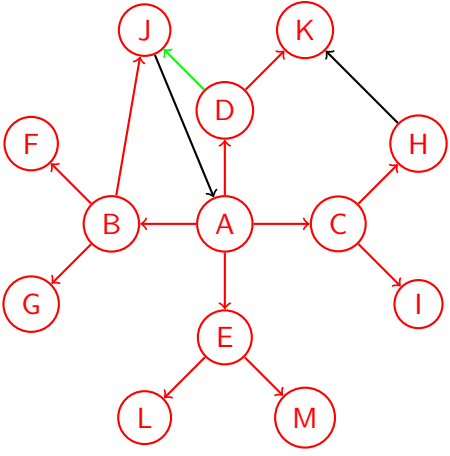
Queue

~~A~~ ~~B~~ ~~C~~ ~~D~~ E F G J H I K

Output

A B C D E F G J H I K

Breadth First Search (cont...)



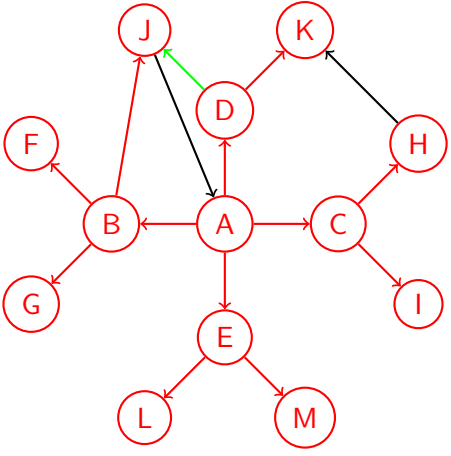
Queue

~~A~~ ~~B~~ ~~C~~ ~~D~~ ~~E~~ F G J H I K L M

Output

A B C D E F G J H I K L M

Breadth First Search (cont...)



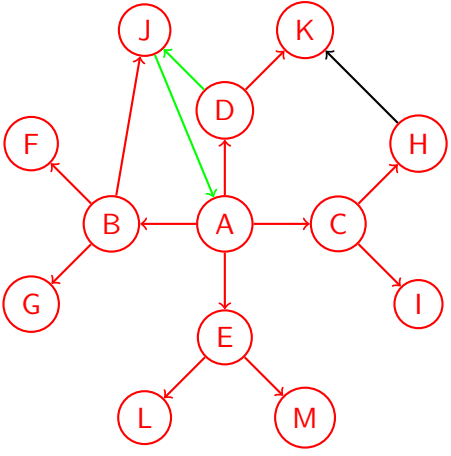
Queue

~~A~~ ~~B~~ ~~C~~ ~~D~~ ~~E~~ ~~F~~ ~~G~~ J H I K L M

Output

A B C D E F G J H I K L M

Breadth First Search (cont...)



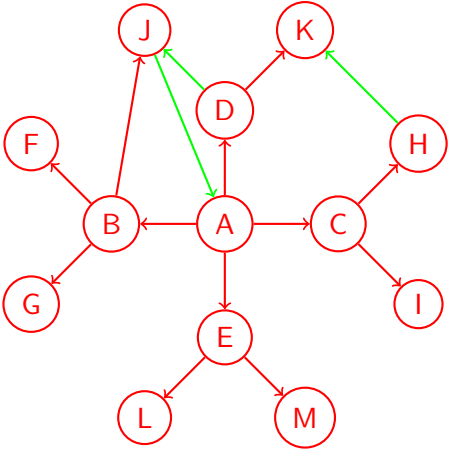
Queue

~~A~~ ~~B~~ ~~C~~ ~~D~~ ~~E~~ ~~F~~ ~~G~~ ~~J~~ H I K L M

Output

A B C D E F G J H I K L M

Breadth First Search (cont...)



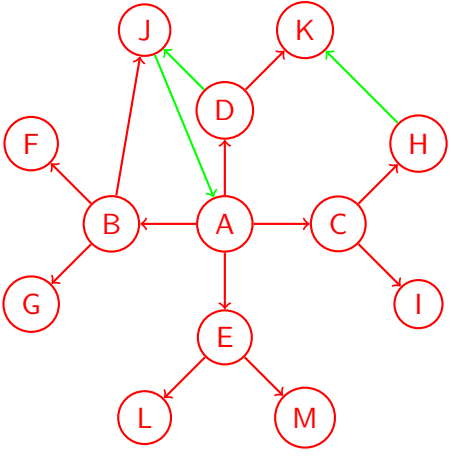
Queue

~~A~~ ~~B~~ ~~C~~ ~~D~~ ~~E~~ ~~F~~ ~~G~~ ~~J~~ ~~H~~ I K L M

Output

A B C D E F G J H I K L M

Breadth First Search (cont...)



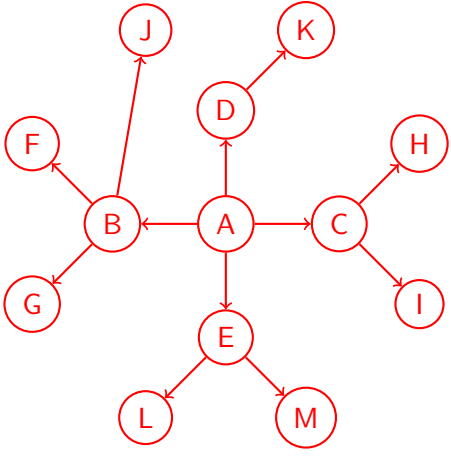
Queue

~~A~~ ~~B~~ ~~C~~ ~~D~~ ~~E~~ ~~F~~ ~~G~~ ~~J~~ ~~H~~ ~~I~~ ~~K~~ ~~L~~ ~~M~~

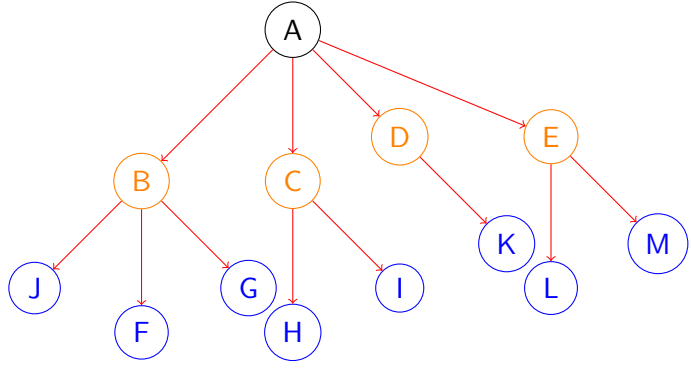
Output

A B C D E F G J H I K L M

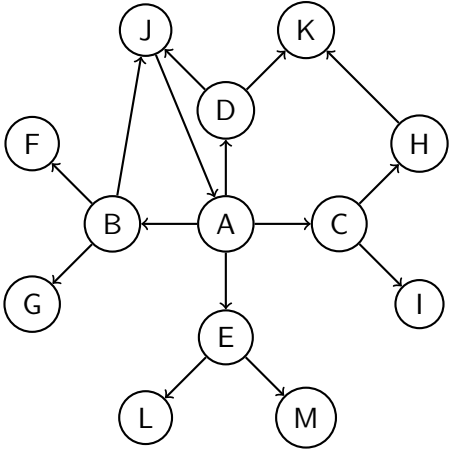
Breadth First Search (cont...)



- If we remove green edges then we get BFS tree
- Black nodes - level 0, orange nodes - level 1, blue nodes - level 2 (shortest distance from A)

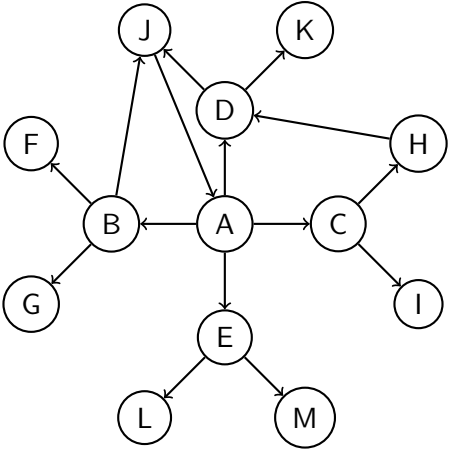


Breadth First Search (cont...)



- Time complexity of BFS traversal is $O(V + E)$
- What if we start BFS traversal from B, D, J? How many levels will be there in BFS tree in each case?
- What if we start BFS traversal from any other node except A, B, D, J?
  - Not all nodes can be reached
  - For each node X in the graph do
    If node X is not reached then
    BFS(X)

Depth First Search

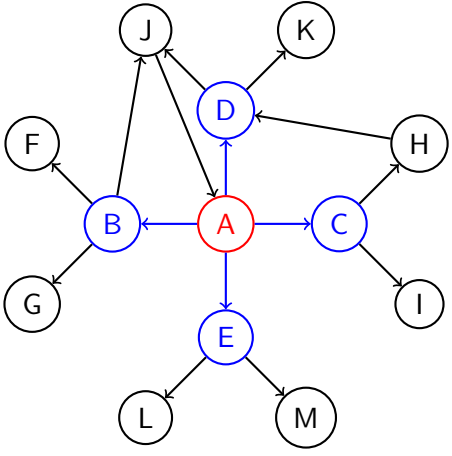


Stack

A

Output

Depth First Search (cont...)



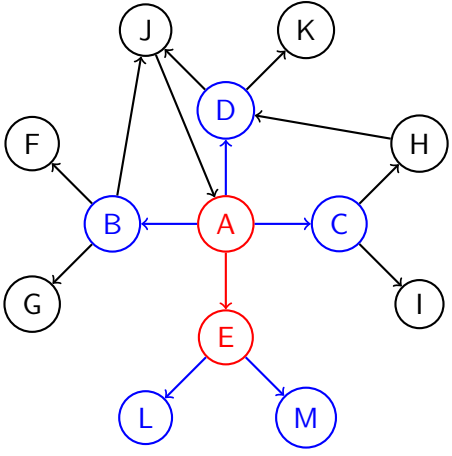
Stack

D B C E

Output

A

Depth First Search (cont...)



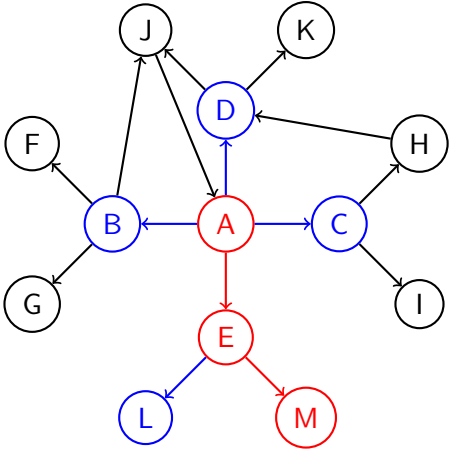
Stack

D B C L M

Output

A E

Depth First Search (cont...)



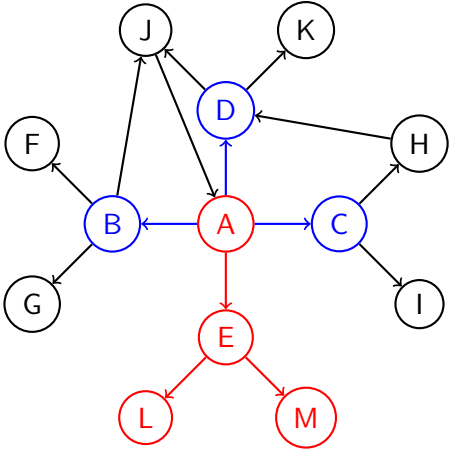
Stack

D B C L

Output

A E M

Depth First Search (cont...)



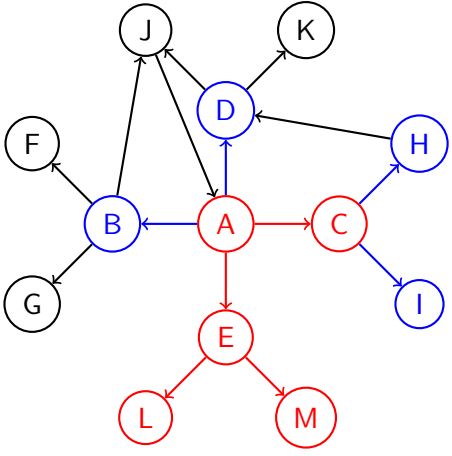
Stack

D B C

Output

A E M L

Depth First Search (cont...)



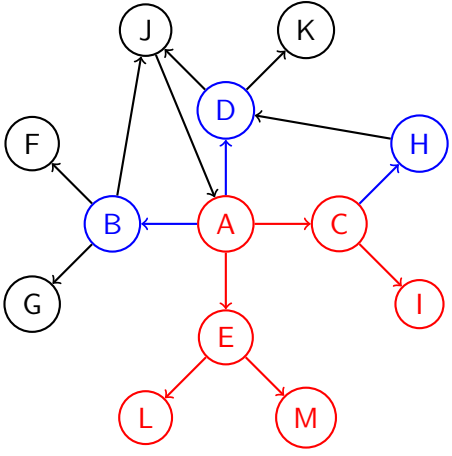
Stack

D B H I

Output

A E M L C

Depth First Search (cont...)



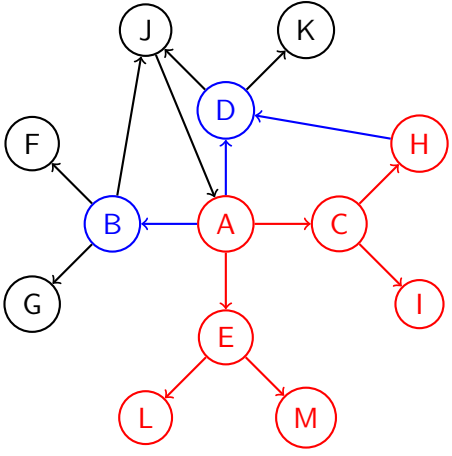
Stack

D B H

Output

A E M L C I

Depth First Search (cont...)



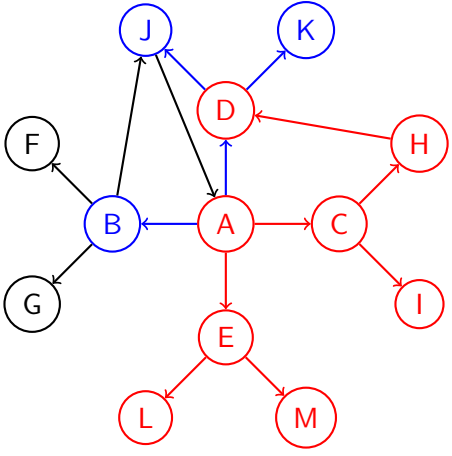
Stack

D B D

Output

A E M L C I H

Depth First Search (cont...)



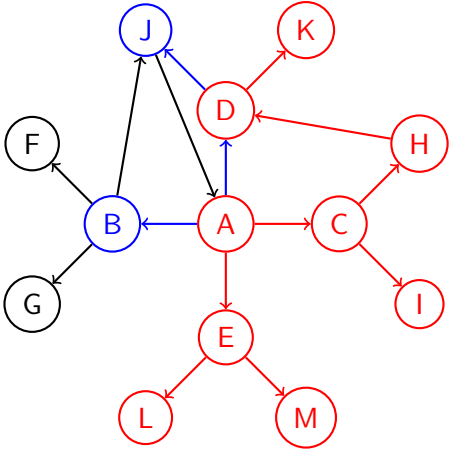
Stack

D B J K

Output

A E M L C I H D

Depth First Search (cont...)



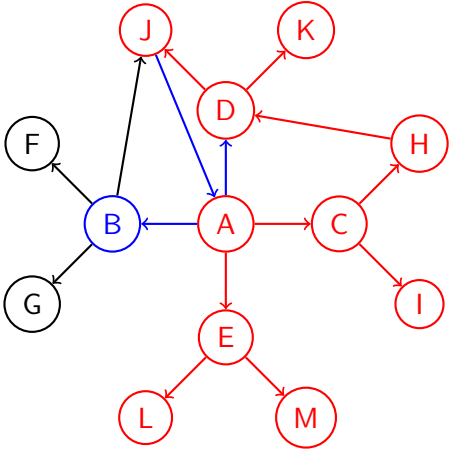
Stack

D B J

Output

A E M L C I H D K

Depth First Search (cont...)



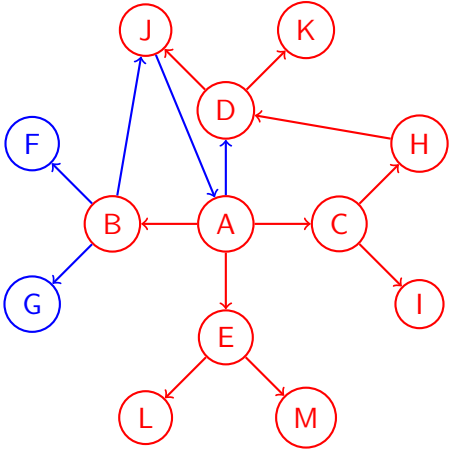
Stack

D B

Output

A E M L C I H D K J

Depth First Search (cont...)



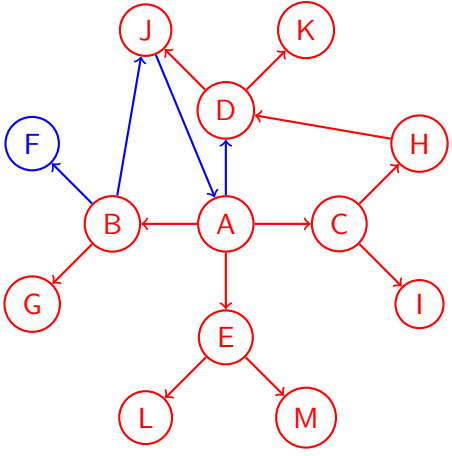
Stack

D F G

Output

A E M L C I H D K J B

Depth First Search (cont...)



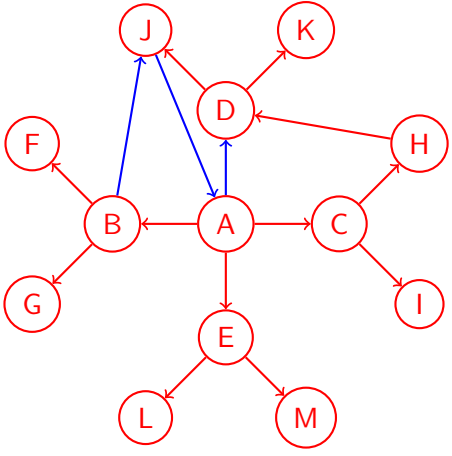
Stack

D F

Output

A E M L C I H D K J B G

Depth First Search (cont...)



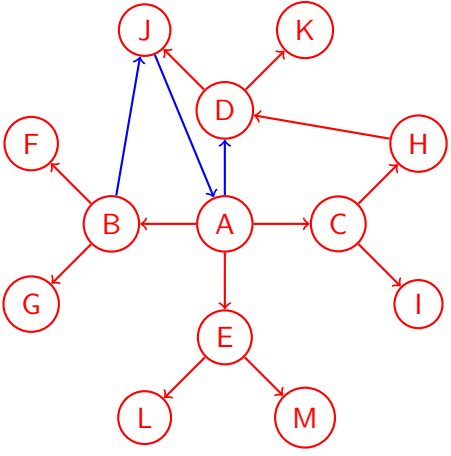
Stack

D

Output

A E M L C I H D K J B G F

Depth First Search (cont...)

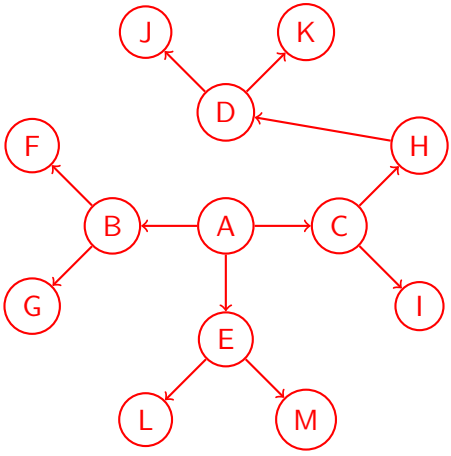


Stack

Output

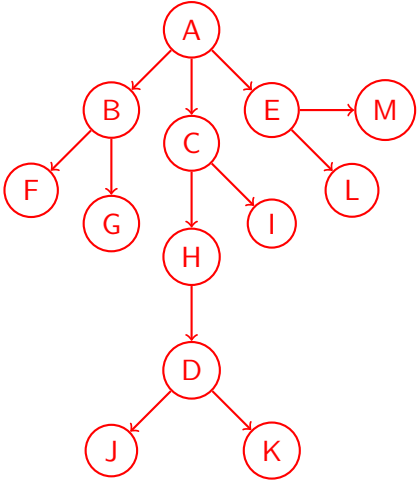
A E M L C I H D K J B G F

Depth First Search (cont...)



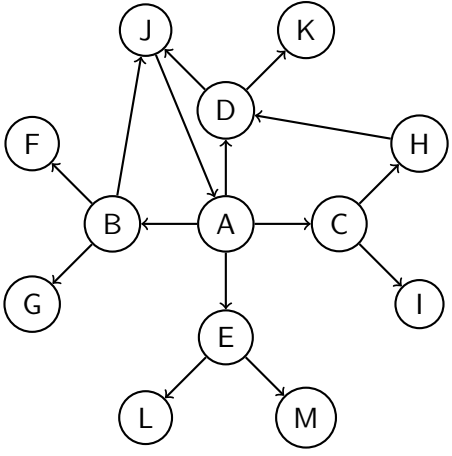
- If we remove blue edges then we get DFS tree
- If you draw this like actual tree with root at A then its height will be 4. This is deeper than BFS tree because depth is explored first

Depth First Search (cont...)



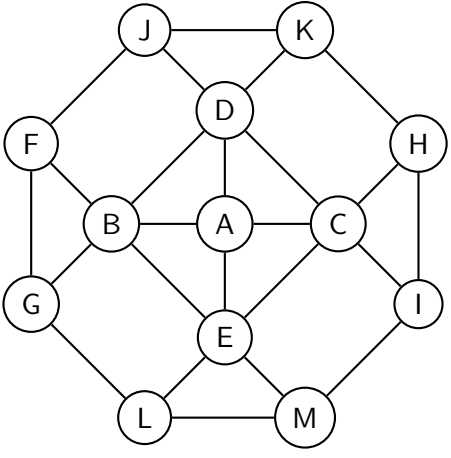
- If we remove blue edges then we get DFS tree
- If you draw this like actual tree with root at A then its height will be 4. This is deeper than BFS tree because depth is explored first
- In recursive DFS algo where choice of sequence results in same tree
  - while J is added to the output, all the nodes from root of tree (A) to J will be on the stack frame and no other node will be on the stack frame.
  - While H is added to the output, only A, C and H will be on the stack frame. Similarly, While B is added to the output, only A and B will be on the stack frame.

Depth First Search (cont...)



- Stack in recursive algo will look different (only one neighbour is pushed on stack at a time). Output may also be different (depending on sequence in which neighbours are pushed on stack).
- Time complexity of DFS traversal is $O(V + E)$
- What if we start DFS traversal from B, C, D, H, J? Or from any other node except A, B, D, J?
- Sometimes all nodes can not be reached
  - For each node X in the graph do
    If node X is not reached then
    DFS(X)

BFS and DFS Practice



- BFS and DFS on undirected graph is similar. An undirected edge between u and v is considered $u \rightarrow v$ and $v \rightarrow u$
- Try BFS and DFS traversal of given graph, starting from A, D and M.

- Output of DFS and DFS traversal differs based on node from which we start traversal and also based on sequence in which we insert neighbours in stack/queue at each step