Data Structures and Algorithms

Prof. Ganesh Ramakrishnan,
Prof. Ajit Diwan,
Prof. D.B. Phatak

Department of Computer Science and Engineering
IIT Bombay

Session: Graph Traversal Algorithm (DFS)

Depth-First Search (DFS)

- $UNEXP \equiv$ Unexplored,
- $VIST \equiv$ Visited,
- $DISC \equiv$ Discovery
- $BACK \equiv$ Back edge (to an ancestor in the tree of $DISC$)

Algorithm DFS(G)
Input: Graph G
Output: Labeling of the edges of G as $DISC$ and $BACK$
for each vertex $u \in V[G]$ **do**
    $setLabel(u, UNEXP)$
end for
for each edge $e \in E[G]$ **do**
    $setLabel(e, UNEXP)$
end for
for each vertex $v \in V[G]$ **do**
    if $getLabel(v) = UNEXP)$ **then**
        DFS(G,v)
    end if
end for

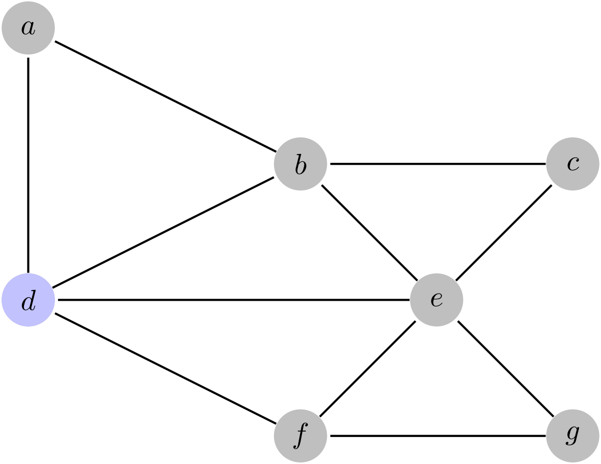
Figure: Depth-First Search: DFS(G)

Depth-First Search (DFS contd.)

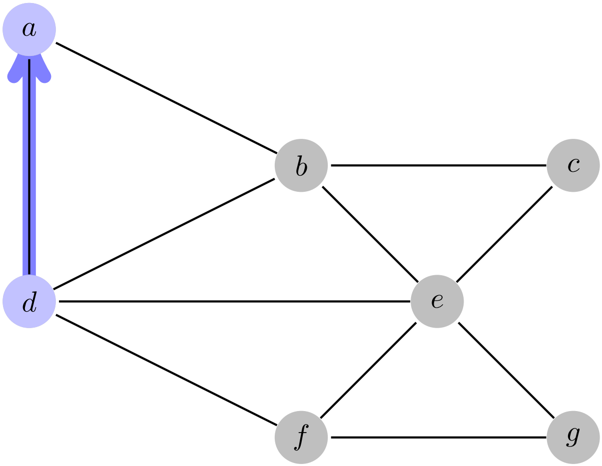
```
Algorithm DFS(G,v)
setLabel(v, VIS)
for all  e ∈ G.incidentEdges(v) do
    if getLabel(e) == UNEXP then
        w ← e.getOtherVertex(v)
        if getLabel(w) = UNEXP then
            setLabel(e, DISC)
            DFS(G, w)
        else
            setLabel(e, BACK)
        end if
    end if
end for
```

Figure: Depth-First Search: DFS(G,v) for a single connected component containing v

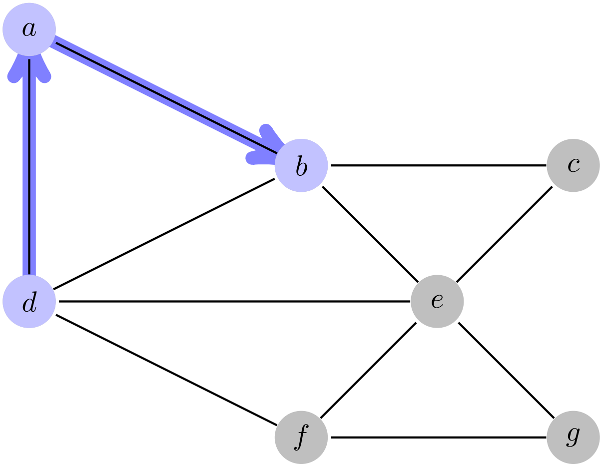
Depth First Search



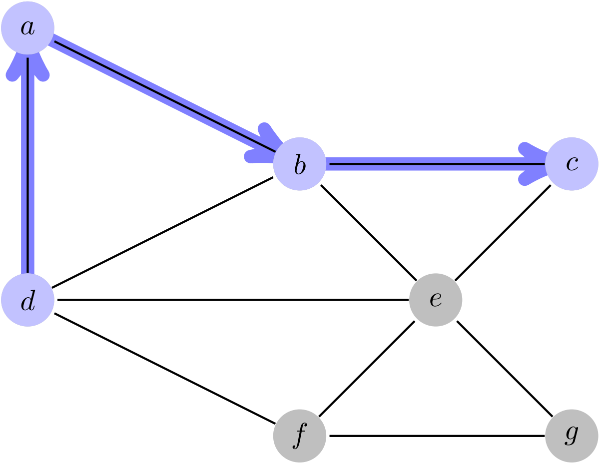
Depth First Search



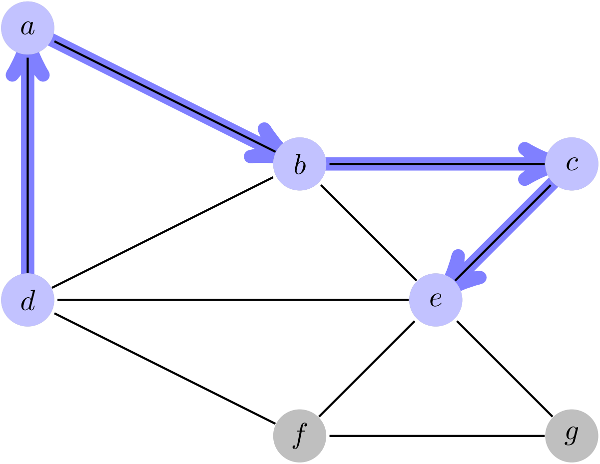
Depth First Search



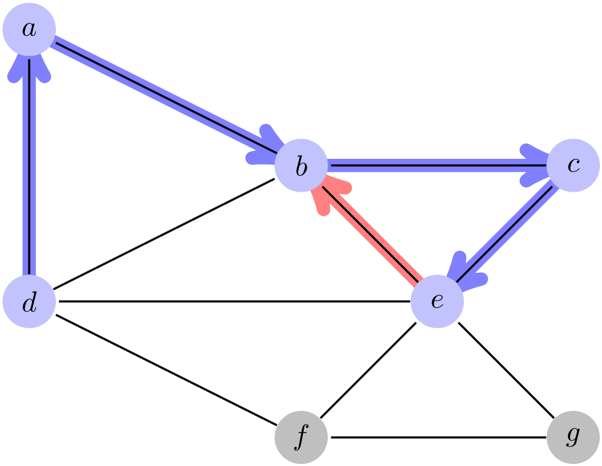
Depth First Search



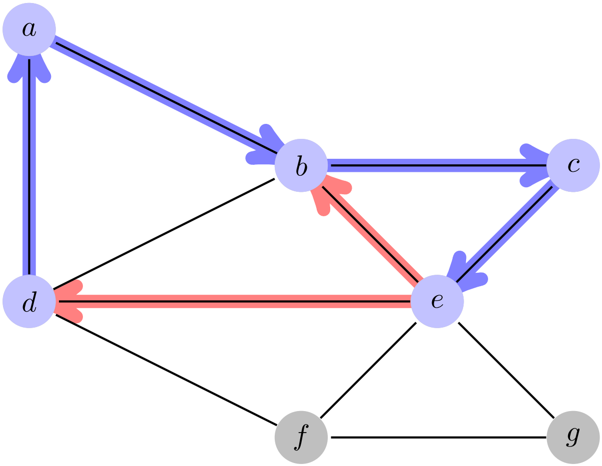
Depth First Search



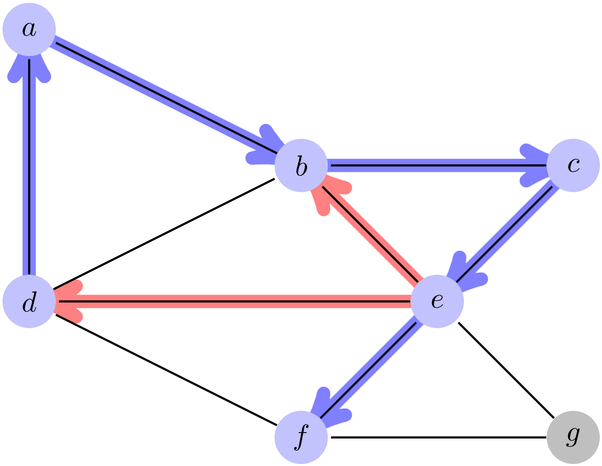
Depth First Search



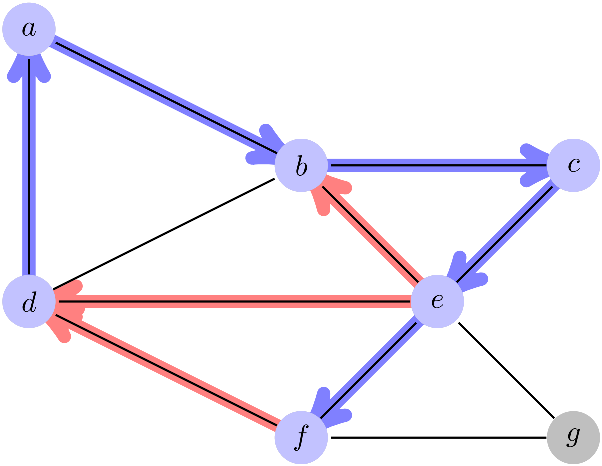
Depth First Search



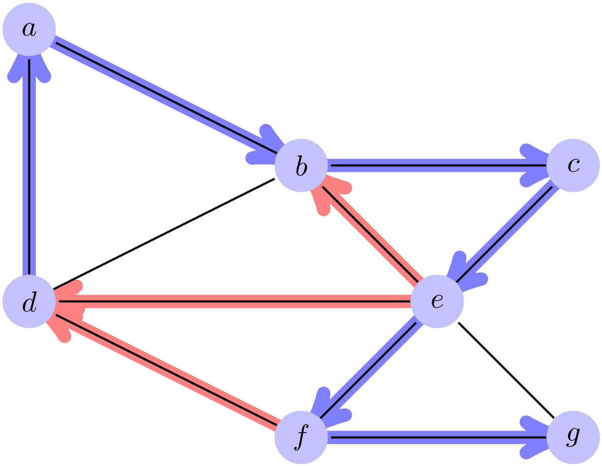
Depth First Search



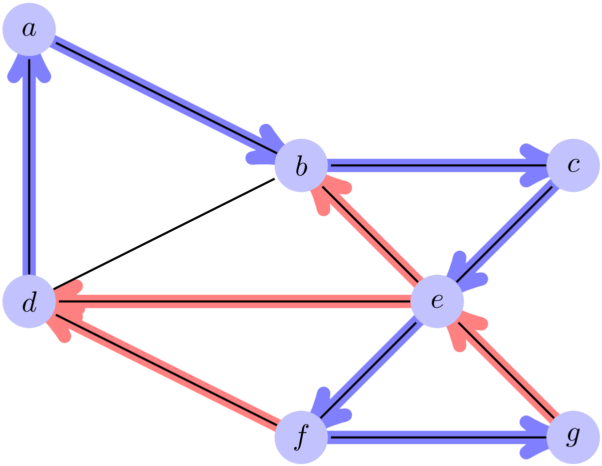
Depth First Search



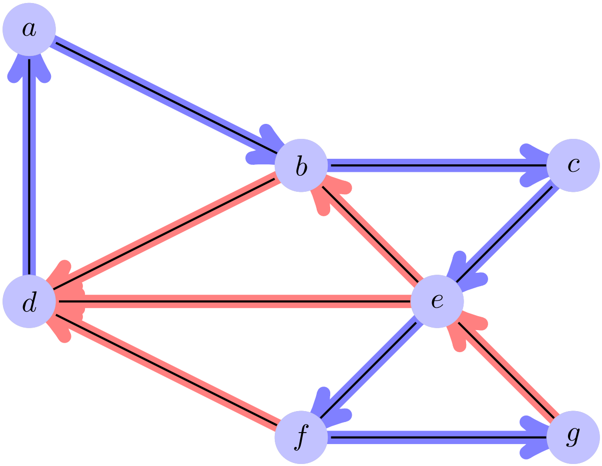
Depth First Search



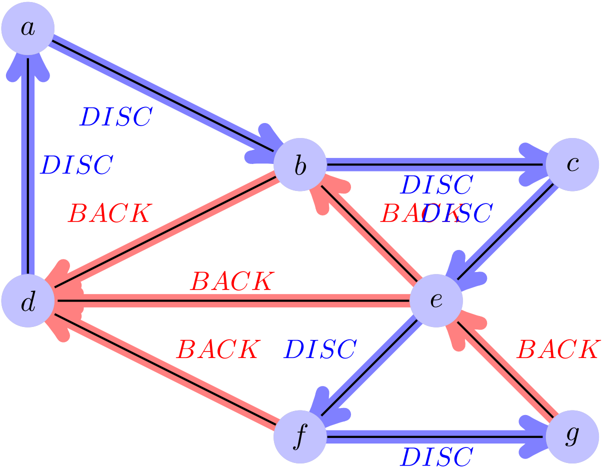
Depth First Search



Depth First Search



Depth First Search



Properties of DFS

1. $DFS(G, v)$ visits all the vertices and edges in the connected component G_v containing v.
2. The edges labeled $DISC$ by $DFS(G, v)$ form a spanning tree T_v of G_v

Analysis of DFS

1. Each vertex is labeled twice $\Longrightarrow O(V)$
   - once as $UNEXP$ (Unexplored)
   - once as VIS (Visited)
2. Each edge is labeled twice $\Longrightarrow O(E)$
   - once as $UNEXP$ (Unexplored)
   - once as $DISC$ (Discovered) or as $BACK$ (a Back going edge within the same connected component)
3. $G.incidentEdges(u)$ is called once for each vertex $u \Longrightarrow O(E)$
4. $\Longrightarrow$ BFS runs in $O(V + E)$ time

Applications of DFS

DFS traversal of a graph G can be used to solve the following problems in $O(V + E)$ time by keeping track of path from start node v to current vertex using a stack S

- Find path from vertex v to u by invoking $DFS(G, v)$ and keeping track of path using a stack S until u is reached
- Detecting cycle by invoking $DFS(G, v)$ for any v and keeping track of path using a stack S until a $BACK$ edge is found
- DFS is the classic strategy for exploring a maze

Thank you