Design and Analysis of Algorithms

L15: Graphs
DFS & BFS

Dr. Ram P Rustagi
Sem IV (2019-H1)
Dept of CSE, KSIT/KSSEM
rprustagi@ksit.edu.in

Resources

- Text book 1: Sec 5.1-5.3 - Levitin

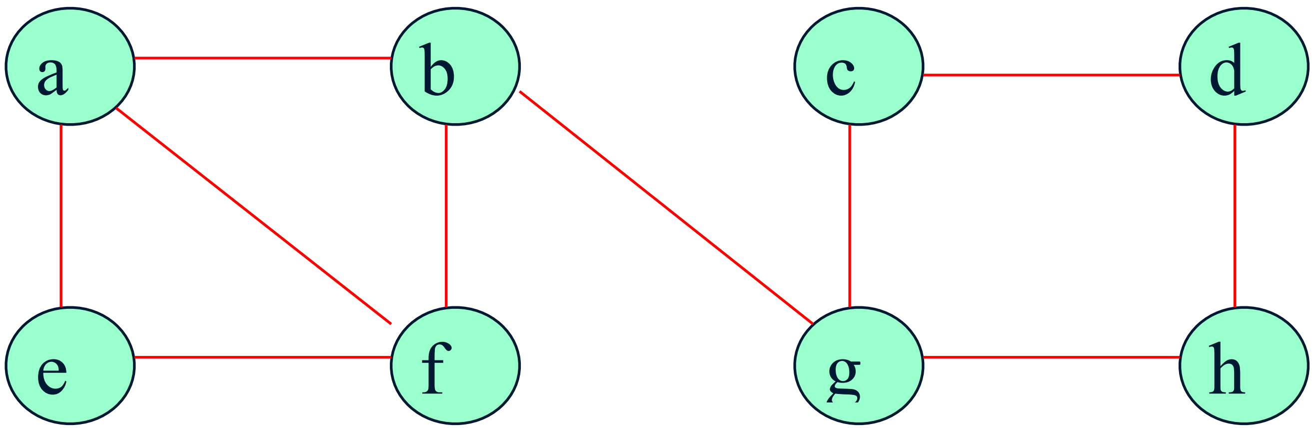
Review of DFS and BFS

- Graph
  - Set of nodes (vertices) connected by edges
  - Max number of edges are `n(n-1)/2`
  - Assumption: no multiple edges b/w any two nodes.
  - Some pair of nodes may not have any edge
- Directed Graph
  - When edges are directed
    - A→B is different than B→A
- Implementation
  - Adjancey (Linked) list
  - Adjacency Matrix
    - Symmetric for undirected graph
    - Asymmetric for directed graph

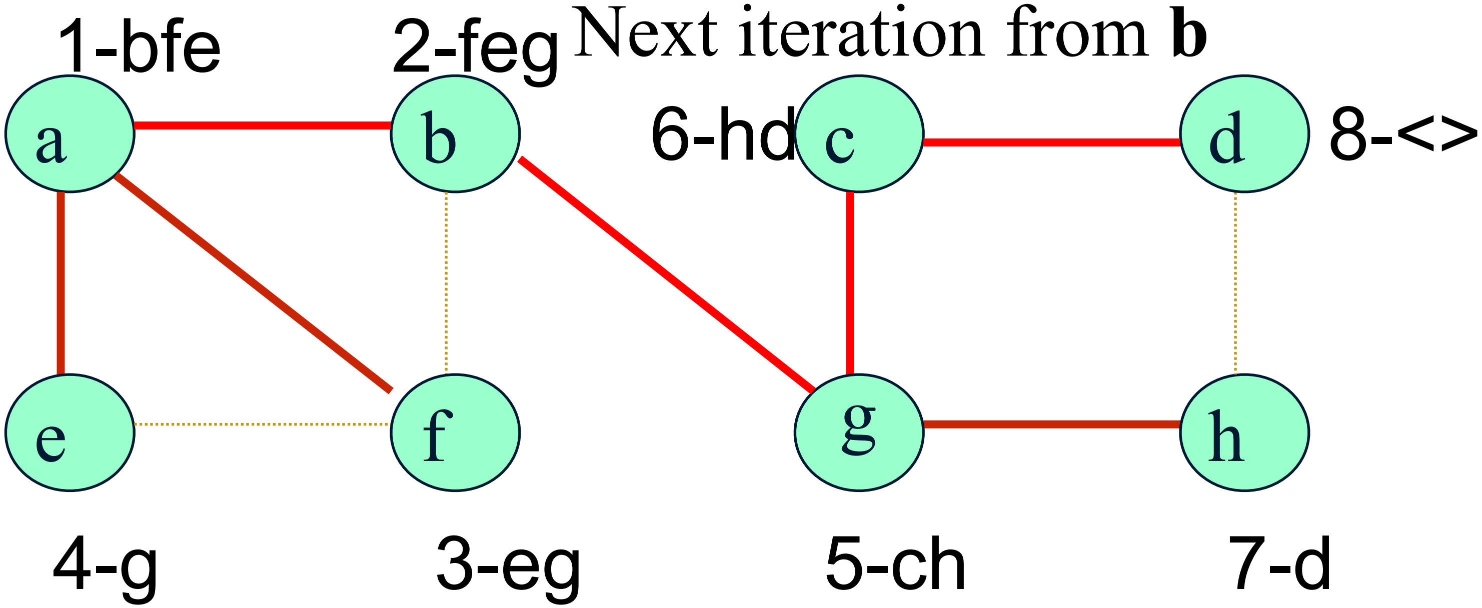
BFS Algo (Undirected Graph)

```
proc BFS(v)
   mark(v) ← ++count
   Add v to queue.
   while queue is not empty do
      remove front vertex (i.e. v) from queue
      for each vertex w ∈ adjacency(v) do
         if w is marked with 0
            mark(w) ← ++count
            add w to the queue
#main
count←0; Initialize queue;
for each vertex v∈V do
   mark(v) ← 0
for each vertex v∈V do
   if mark(v) is 0
      BFS(v)
```

BFS Traversal



Next iteration from **b**



BFS Time Complexity

- Same efficiency as DFS
  – Adjacency matrices: $\Theta(|V|^2)$?
  – Adjacency lists: $\Theta(|V|+|E|)$?
- Vertices ordering
  – Single ordering of vertices
- Applications
  – Similar to DFS
  – Finding shortest path from a vertex to another becomes easier

BFS Traversal

- Visits graph vertices by
  - visiting all neighbours of last visited node
- Instead of a stack based implementation
  - Uses queue based implementation

DFS

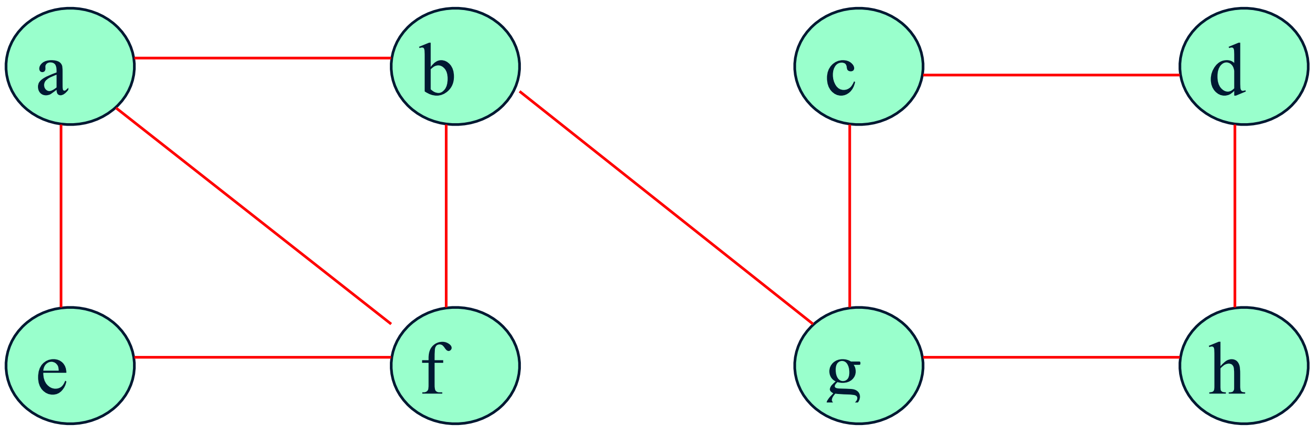
- DFS:
  - Start from a vertex (called root), mark it visited
  - Repeat the following
    - Find an unvisited vertex (not marked) connected by current node under consideration.
      - Mark this node as visited.
    - If there is no unvisited (unmarked) node connected to current node, backtrack.
- DFS Implementation
  - Using recursion (and stack)

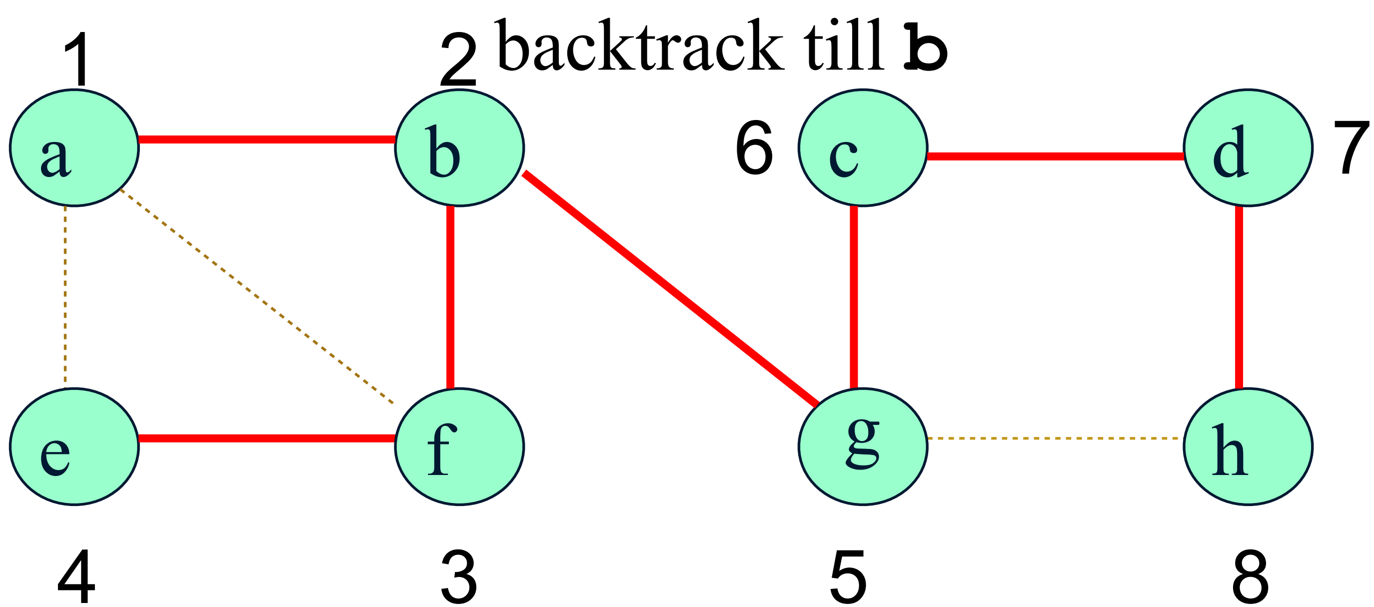
DFS Algo

```
# Input: G=(V, E)
# o/p: nodes V marked in the order these are visited.
# mark of 0 implies unvisited.
proc dfs(v)
   mark(v) ← ++count; // perform any Prework
  for each vertex w ∈ V adjacent to v do
     if w is marked with 0, then
        dfs(w); // perform any Postwork
#end proc dfs(v)

for each vertex v ∈ V do
   mark(v) ← 0
count ← 0
for each vertex v ∈ V do
   if v is marked with 0, then
      dfs(v)
```

DFS Traversal





DFS Traversal: Time Complexity

- DFS implementation by Adjacency Matrix

  $\Theta(|V|^2)$

- DFS implementation by Adjacency Lists

  $\Theta(|V|+|E|)$

- Applications
  - Connected components
  - Checking for connected graph
  - Checking for acyclicity
  - Finding bi-connected components

Tree Traversal

- Forward Edge
- Cross Edge
- Back edge (Cycle)

Summary

- Advantages and disadvantages of Divide and Conquer
- Decrease and conquer approach
- DFS traversal
- BFS traversal