Basic Graph Algorithms

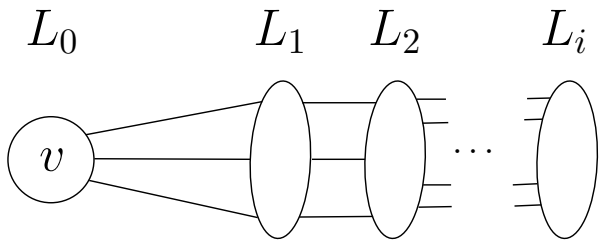
- Exploring Graphs
- Depth First Search
- DFS Forest - Start and Finish Time
- DAG, Topological Sorting
- Strongly Connected Components
- Breadth First Search
- Bipartite Graphs

Imdad ullah Khan

Breadth First Search (BFS)

- Breadth First Search is an alternative algorithm to explore graphs
- Starting from a vertex v, BFS explores all vertices that are reachable from v
- BFS first explores neighbors of v, then neighbors of neighbors of v and so on

Starting from v, explore outwards, adding nodes to one "layer" at a time



- $L_0 := \{v\}$
- $L_1 :=$ neighbors of L_0
- $L_2 :=$ those unvisited nodes having an edge from L_1
- $L_{i+1} :=$ those unvisited nodes having an edge from L_i

BFS: Algorithm

Algorithm BFS(G)

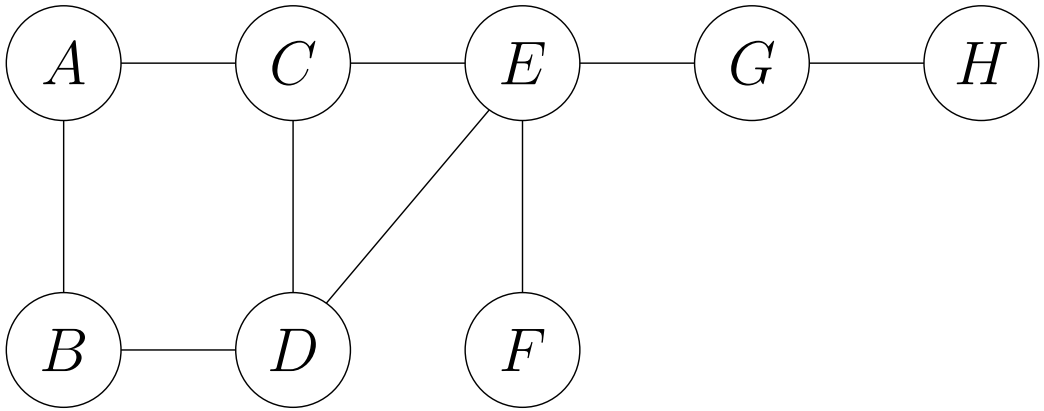
```
visited ← ZEROS(n)
for v ∈ V do
    if visited[v] = 0 then
        BFS-EXPLORE(v)
```

Algorithm BFS-EXPLORE(s)

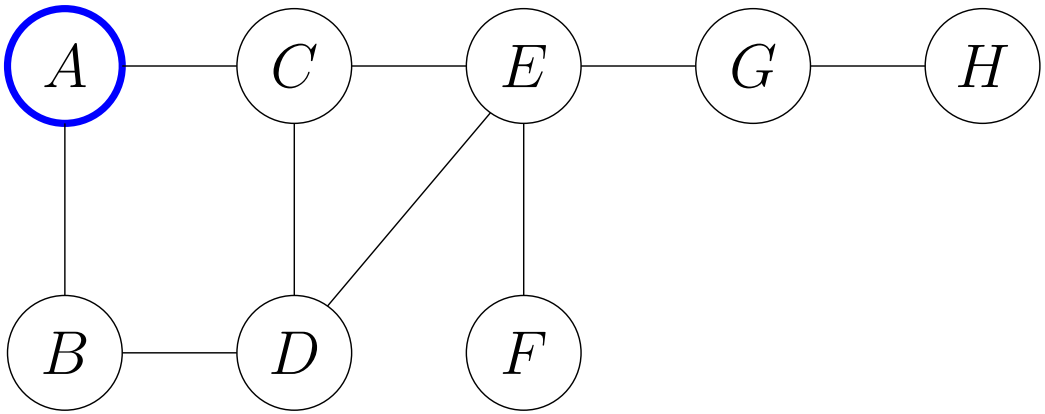
```
visited[v] ← 1
ENQUEUE(Q, s)
while Q ≠ ∅ do
    v ← DEQUEUE(Q)
    for u ∈ N(v) do
        if visited[u] = 0 then
            visited[u] ← 1
            ENQUEUE(Q, u)
```

Running time: $O(|E| + |V|) = O(n + m)$

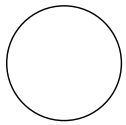
BFS: Example



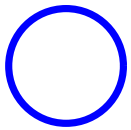
BFS: Example



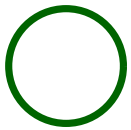
L_0



Not Started

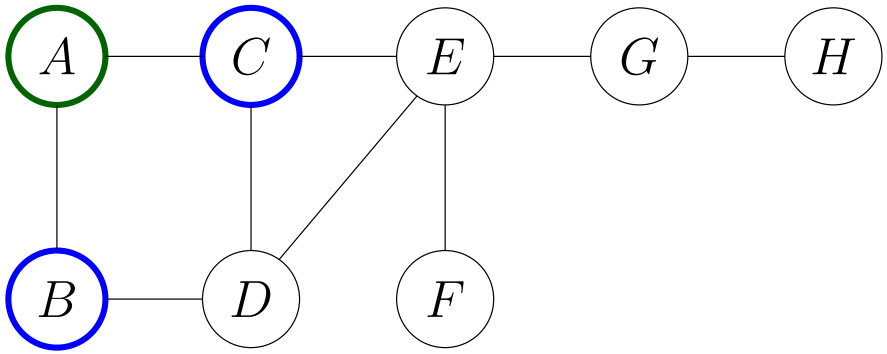


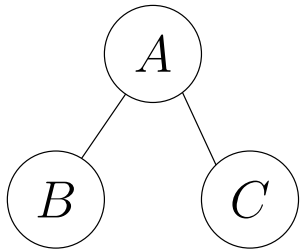
Started



Finished

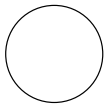
BFS: Example



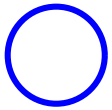


L_0

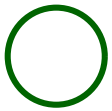
L_1



Not Started

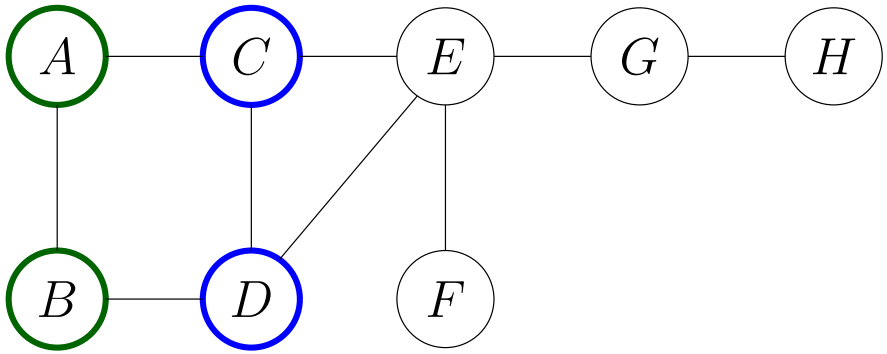


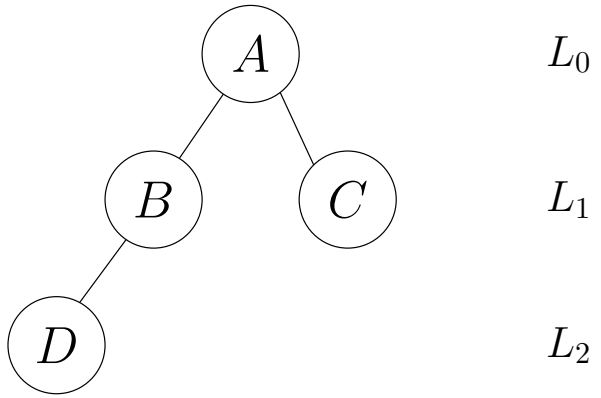
Started

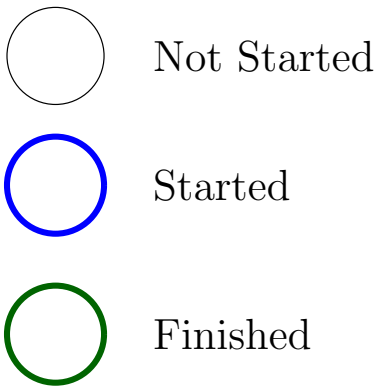


Finished

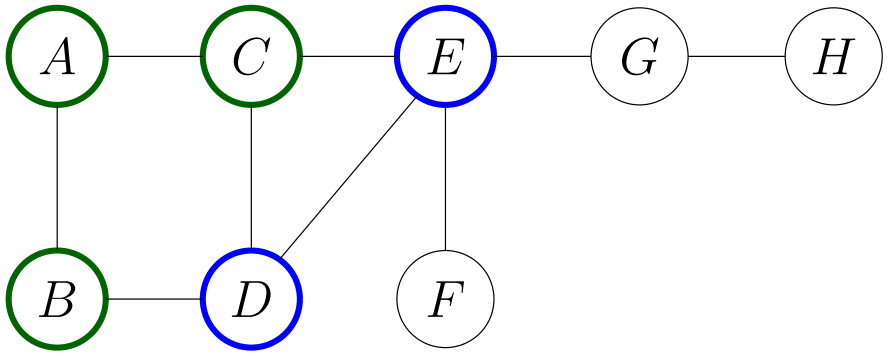
BFS: Example

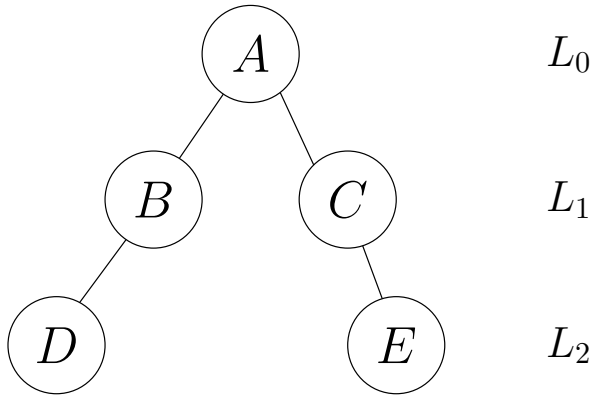


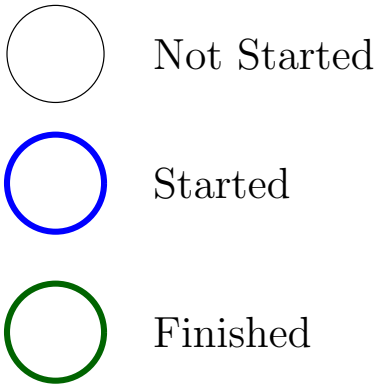




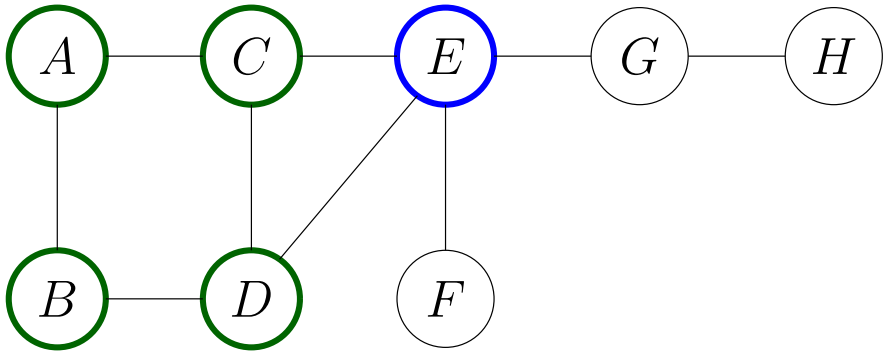
BFS: Example

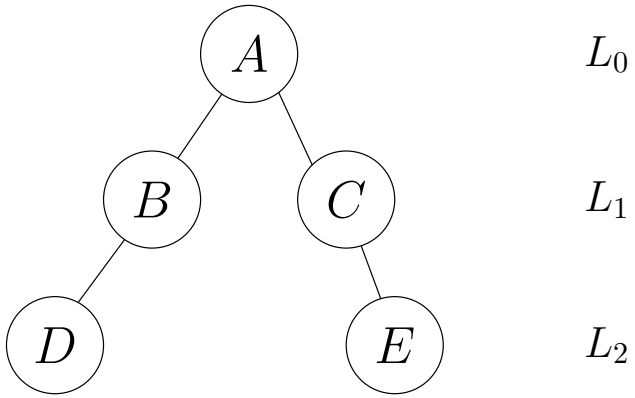


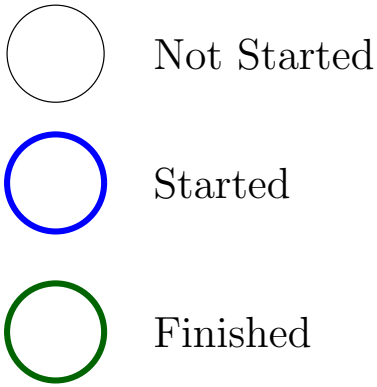




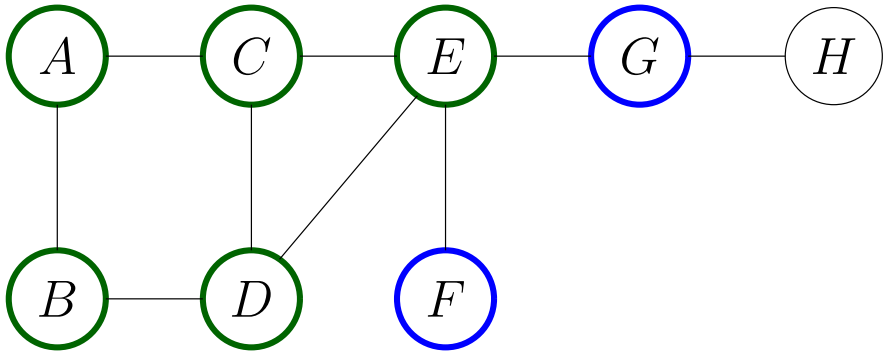
BFS: Example

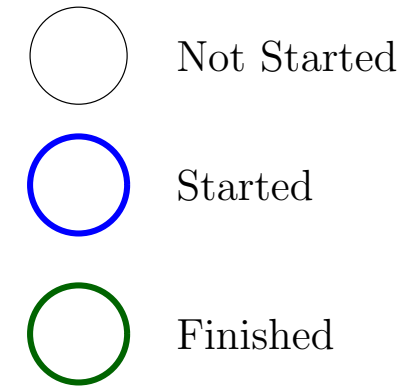


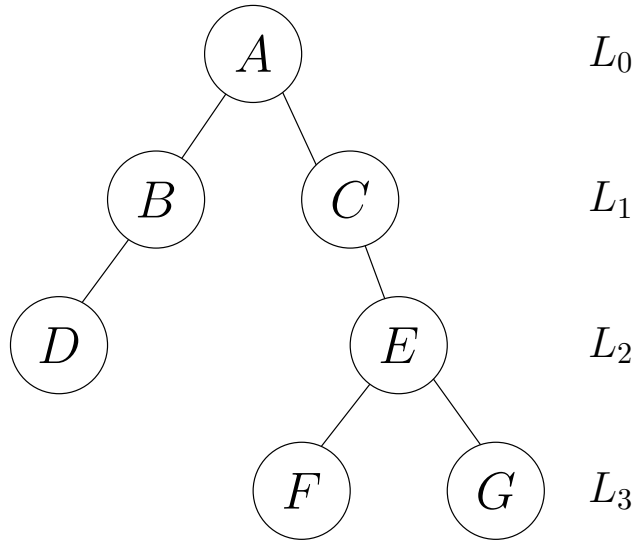




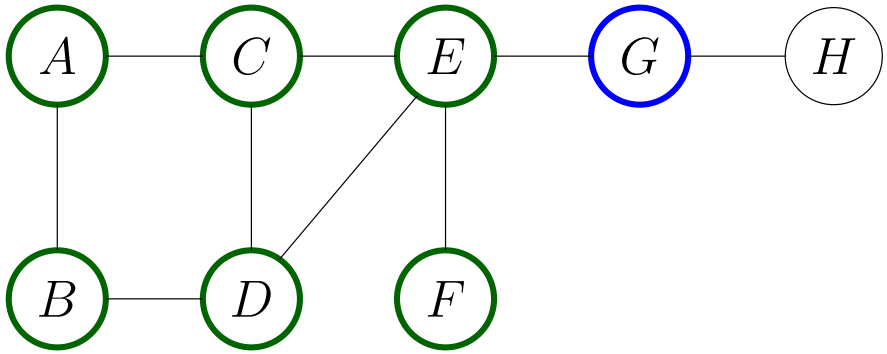
BFS: Example

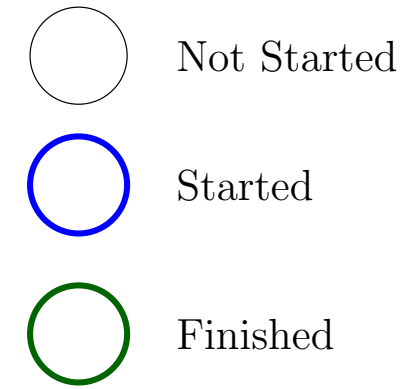


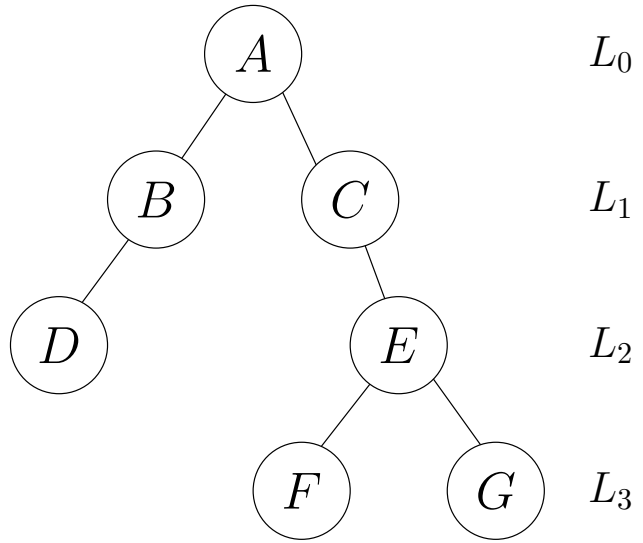




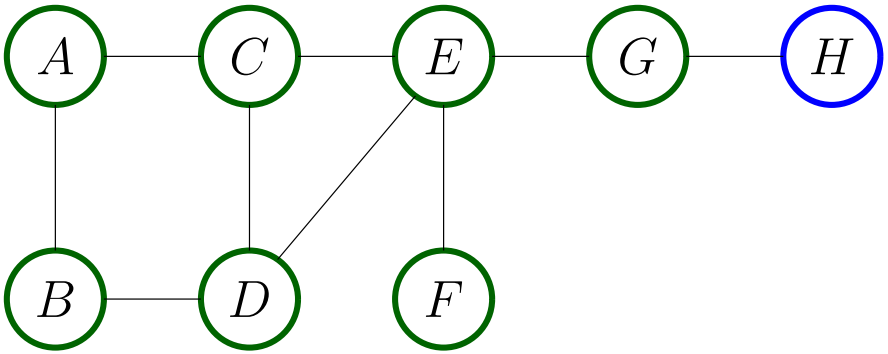
BFS: Example

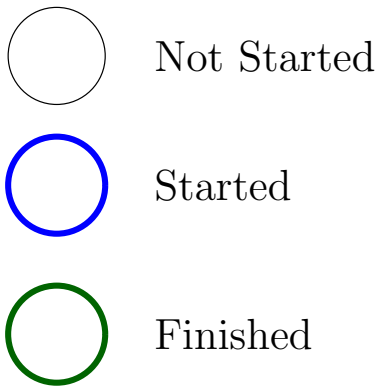


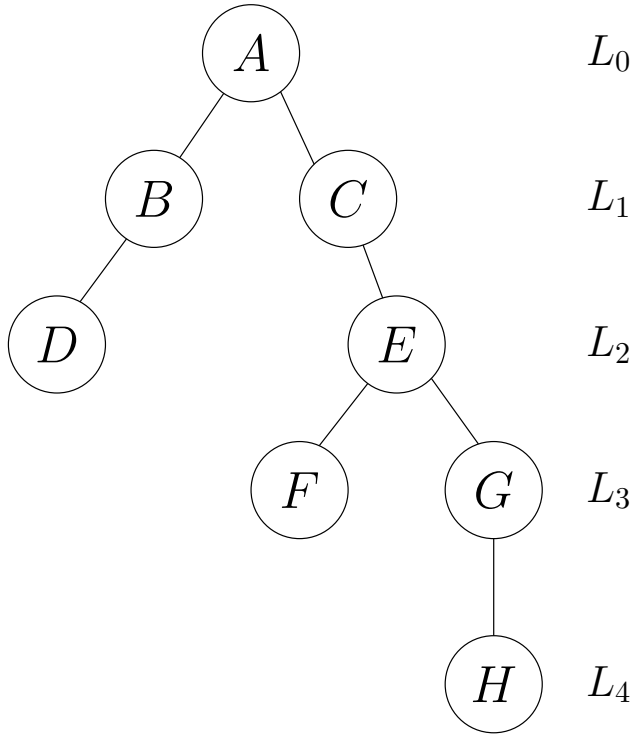




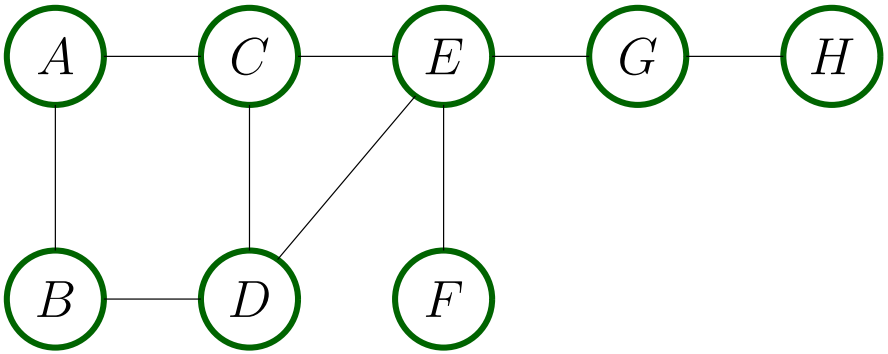
BFS: Example

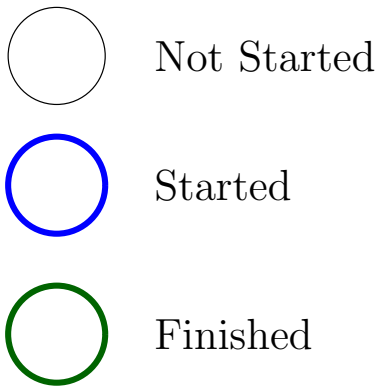

A
C
E
G
H
B
D
F

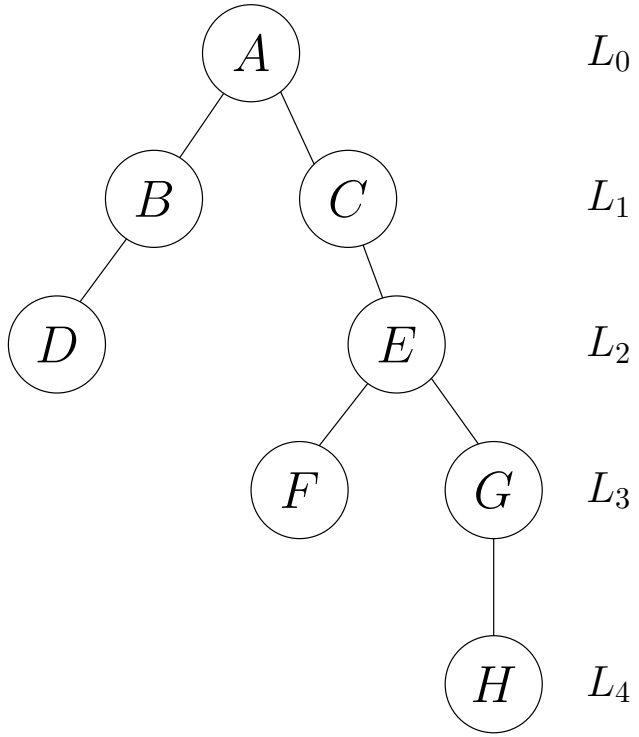

Not Started
Started
Finished


A
B
C
D
E
F
G
H
L_0
L_1
L_2
L_3
L_4

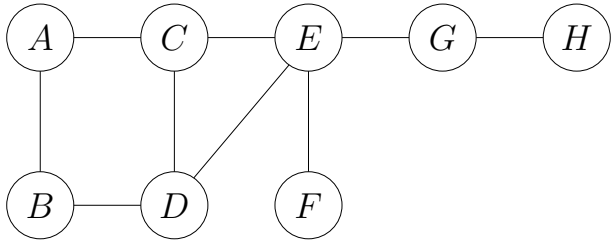
BFS: Example


A
C
E
G
H
B
D
F

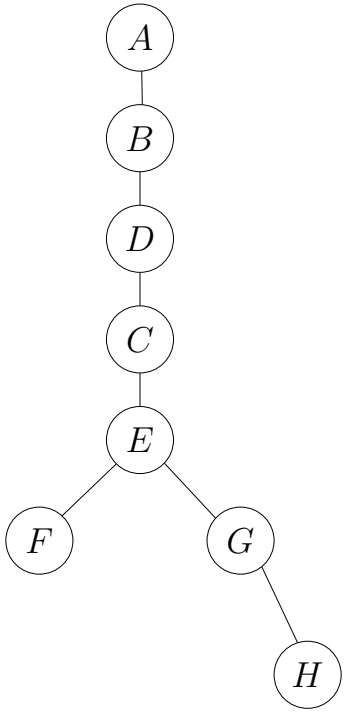

Not Started
Started
Finished


A
B
C
D
E
F
G
H
L_0
L_1
L_2
L_3
L_4

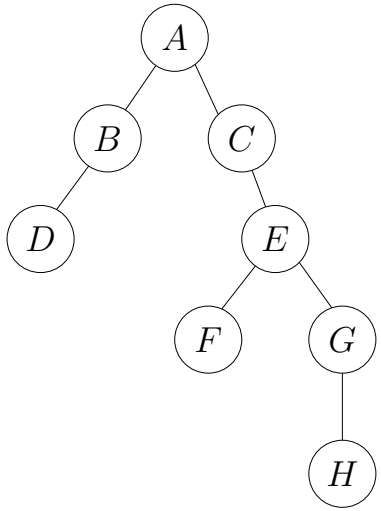
DFS and BFS



(a) Graph G



(a) DFS Tree of G



(b) BFS Tree of G

Connected Components via BFS

Find number of connected components in graph G

Algorithm BFS(G)

```
visited ← ZEROS(n)
compID ← 0
for v ∈ V do
    if visited[v] = 0 then
        compID ← compID + 1
        BFS-EXPLORE(v)
```

Algorithm BFS-EXPLORE(s)

```
visited[v] ← 1
ENQUEUE(Q, s)
while Q ≠ ∅ do
    v ← DEQUEUE(Q)
    for u ∈ N(v) do
        if visited[u] = 0 then
            visited[u] ← 1
            ENQUEUE(Q, u)
```

This finds just the number of connected components

- Can be extended to find nodes in all components, components sizes, the largest/smallest components sizes etc.

Running Time: $O(|V| + |E|)$

Applications of BFS

- Superimpose all graph edges onto the BFS tree
- What can we say about edges?
- (forward edges, tree edges, backward edges, cross edges)
- This observation is the key to many applications of BFS
- Shortest Paths, Bipartite testing, Betweenness Centrality