Depth First Search

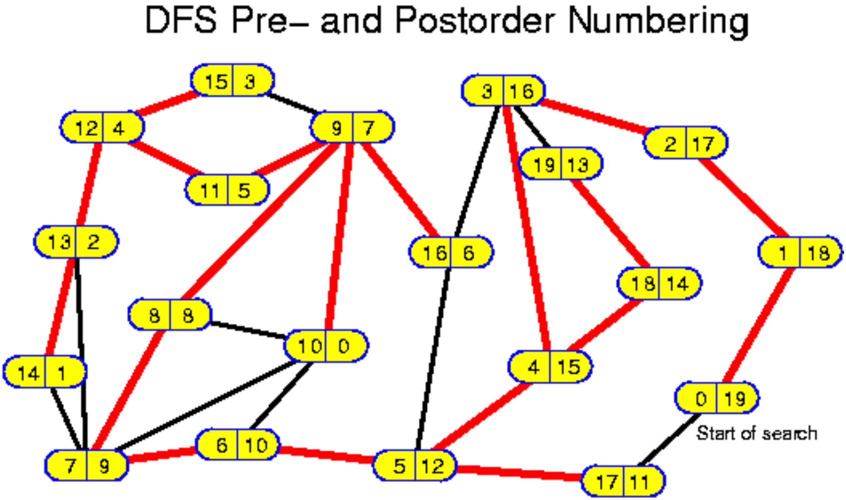
- Basic form of processing graphs is traversal.
- DFS and BFS are two important traversal techniques.

```
// Initializations
index = 0;
for all (v ∈ V) {
    mark[v] = "unvisited";
    T = Φ; // Tree edges
}
choose(s); // Start vertex
DFS(s,G);
```

Depth First Search

```
procedure DFS(G, v) {
    mark[v] = "visited";
    dfn[v] = ++index; // DFS numbers
    for all (w ∈ ADJ_G(v)) {
        if (mark[w]=="unvisited") {
            T = T ∪ {(v, w)}; // Update T
            DFS(G, w); // Recursive call
        }
    }
}
```

Depth First Search Example



Correctness of DFS

Lemma

DFS procedure is called exactly once for each vertex.

Proof.

- Once DFS is called for a particular vertex v, it is marked as "visited".
- DFS is never called out on "visited" vertices.

$\square$

Running Time of DFS

Lemma

Running time of DFS procedure is $(V + E)$.

Proof.

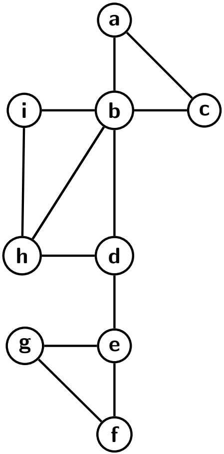
- Also obvious from algorithm's pseudo code.
- Procedure is called once for each vertex.
- But it traverses each edge exactly twice: once in forward direction and once in reverse direction.

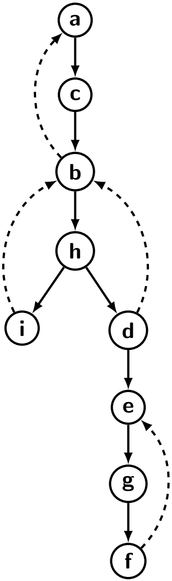
□

Classification of Edges

- DFS gives an orientation to edges of an undirected graph.
- Traversing some edges lead to unvisited vertices.
- While the remaining edges lead to visited vertices.
- If a vertex w is found visited during DFS(v), then w must be an ancestor of v in the DFS tree.
  - DFS(v) must have been called during the time DFS(w) call itself.
  - In other words, DFS(w) is still incomplete when DFS(v) was called.
- So edges are classified into two types: tree edges, and back edges.

Edge Types in DFS of Undirected Graphs

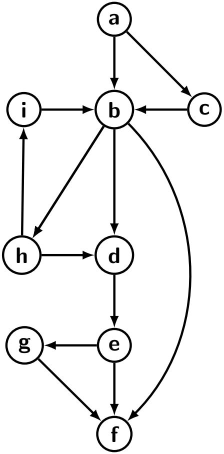


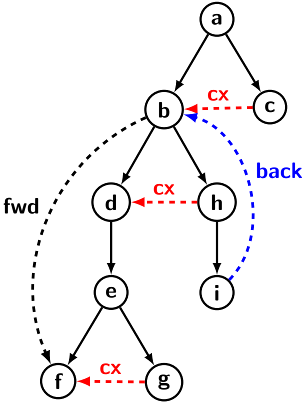


DFS of Directed Graphs

- DFS of directed graphs must explore the edges by respecting the direction of orientation of the edges.
- As usual, tree edges are those edges that always lead to new (unvisited) vertices.
- Remaining edges are partitioned into three other types.
  - Back edges: which lead from a descendant to an ancestor.
  - Forward edges: which lead from a proper ancestor to a descendant
  - Cross edges: connects two unrelated vertices w and v. If the orientation is $v \to w$, then v is visited aftr w.

Edge Types in DFS of Directed Graphs





DFS of Disconnected Graphs

- Algorithm we have presented works for connected graph.
- For DFS of disconnected graphs, we need to change initial calling of procedure a bit.

```
// Initializations
index = 0;
for all (v ∈ V) {
    mark[v] = "unvisited";
    T = Φ; // Tree edges
}
for all (v ∈ V) {
     if (mark[v] == "unvisited")
          DFS(v, G);
}
```