Design and Analysis of Algorithms

Mohammad GANJTABESH

mgtabesh@ut.ac.ir

School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

Graph Theoretical Problems

- Basec Definitions
- Graph Representation
- Graph Traversal (BFS, DFS)
- Topological Sort
- Strongly Connected Components
- Shortest Paths
  - Single-Source All Destination (Bellman-Ford, Dijkestra)
  - All-Pairs (Floyd-Warshall, Johnson)
- Minimum Spanning Tree (Kruskal, Prim)

Graph Traversal: Depth First Search

Depth First Search

Given a graph $G = (V, E)$ and a distinguished source vertex s, the strategy followed by **depth-first search** is to search deeper in the graph whenever possible. It produce a depth-first forest.

Graph Traversal: Depth First Search

Depth First Search

Given a graph $G = (V, E)$ and a distinguished source vertex s, the strategy followed by **depth-first search** is to search deeper in the graph whenever possible. It produce a depth-first forest.

For each vertex $v \in V$, we store the following information during the execution of DFS:

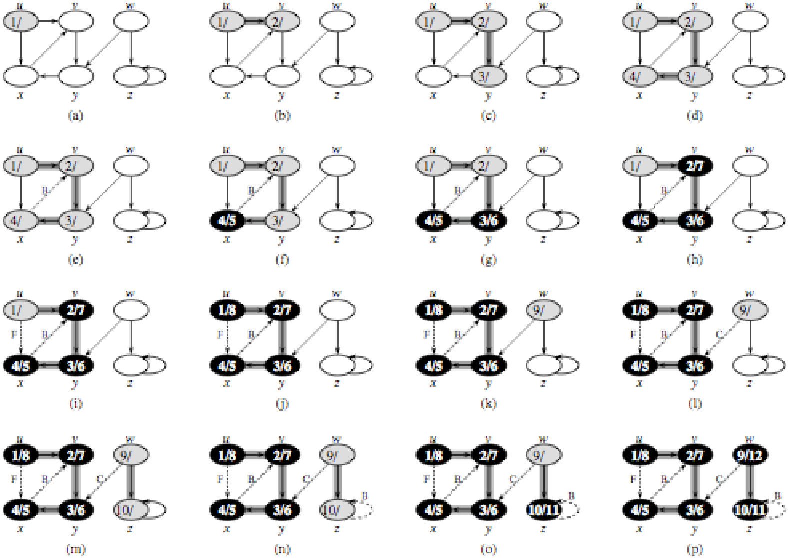
- $color[v] \in \{white, gray, black\}$: Same as bFS.
- $d[v]$: the discovery time of vertex v (when $color[v]$ becomes gray).
- $f[v]$: the finishing time of vertex v (when $color[v]$ becomes black).
- $p[v]$: the parent of v in search tree.

Graph Traversal: Depth First Search

```
DFS(G)
for each vertex u ∈ V[G]
        do color[u] ← WHITE
           π[u] ← NIL
time ← 0
for each vertex u ∈ V[G]
        do if color[u] = WHITE
              then DFS-VISIT(u)
```

```
DFS-VISIT(u)
color[u] ← GRAY              ▷ White vertex u has just been discovered.
time ← time + 1
d[u] ← time
for each v ∈ Adj[u]          ▷ Explore edge (u, v).
        do if color[v] = WHITE
              then π[v] ← u
                   DFS-VISIT(v)
color[u] ← BLACK             ▷ Blacken u; it is finished.
f[u] ← time ← time + 1
```

Graph Traversal: Depth First Search



Classification of edges in Depth First Search

- **Tree edges** are edges in the depth-first forest. Edge (u, v) is a tree edge if v was first discovered by exploring edge (u, v).

Classification of edges in Depth First Search

- **Tree edges** are edges in the depth-first forest. Edge (u, v) is a tree edge if v was first discovered by exploring edge (u, v).
- **Back edges** are those edges (u, v) connecting a vertex u to an ancestor v in a depth-first tree.

Classification of edges in Depth First Search

- **Tree edges** are edges in the depth-first forest. Edge (u, v) is a tree edge if v was first discovered by exploring edge (u, v).
- **Back edges** are those edges (u, v) connecting a vertex u to an ancestor v in a depth-first tree.
- **Forward edges** are those non-tree edges (u, v) connecting a vertex u to a descendant v in a depth-first tree.

Classification of edges in Depth First Search

- **Tree edges** are edges in the depth-first forest. Edge (u, v) is a tree edge if v was first discovered by exploring edge (u, v).
- **Back edges** are those edges (u, v) connecting a vertex u to an ancestor v in a depth-first tree.
- **Forward edges** are those non-tree edges (u, v) connecting a vertex u to a descendant v in a depth-first tree.
- **Cross edges** are all other edges. They can go between vertices in the same depth-first tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different depth-first trees.

Classification of edges in Depth First Search

- **Tree edges:** $gray \longrightarrow white$.
- **Back edges** $gray \longrightarrow gray$.
- **Forward edges** $gray \longrightarrow black$.
- **Cross edges** $gray \longrightarrow black$.

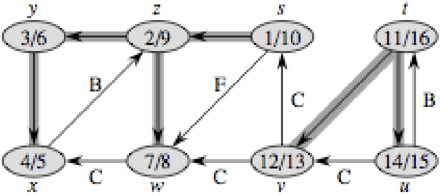
Classification of edges in Depth First Search

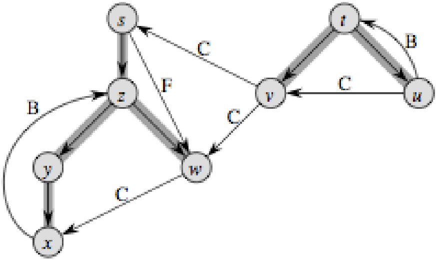
- **Tree edges:** *gray* $\longrightarrow$ *white*.
- **Back edges** *gray* $\longrightarrow$ *gray*.
- **Forward edges** *gray* $\longrightarrow$ *black*.
- **Cross edges** *gray* $\longrightarrow$ *black*.

Since the Forward edges and Cross edges are not distinguishable from the colors, we use discovery times and finishing times of vertices of edges (u, v) as follows :

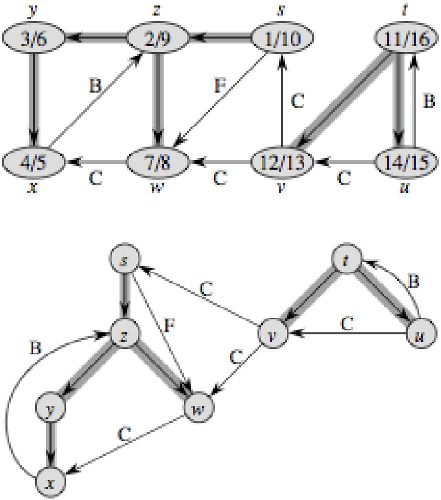
- **Tree edges:** $d[u] < d[v] < f[v] < f[u]$.
- **Back edges** $d[v] < d[u] < f[u] < f[v]$.
- **Forward edges** $d[u] < d[v] < f[v] < f[u]$.
- **Cross edges** $d[v] < f[v] < d[u] < f[u]$.

Classification of edges in Depth First Search





Classification of edges in Depth First Search



Which kind of edges do not appear in undirected graph?

Exercises

1. Classify the edges in BFS.
2. Change the DFS algorithm in order to perform the edge classification.
3. Rewrite the procedure DFS, using a stack to eliminate recursion.
4. Give a counterexample to the conjecture that if there is a path from u to v in a directed graph G, and if $d[u] < d[v]$ in a depth-first search of G, then v is a descendant of u in the depth-first forest produced.
5. Give a counterexample to the conjecture that if there is a path from u to v in a directed graph G, then any depth-first search must result in $d[v] \leq f[u]$.
6. A directed graph $G = (V, E)$ is singly connected if $u \leadsto v$ implies that there is at most one simple path from u to v for all vertices $u, v \in V$. Give an efficient algorithm to determine whether or not a directed graph is singly connected.