Depth-First Search

Adapted from the CLRS book slides

DEPTH-FIRST SEARCH

Input: $G = (V, E)$, directed or undirected. No source vertex given.

Output:

- 2 ***timestamps*** on each vertex:
  - $v.d$ = ***discovery time***
  - $v.f$ = ***finish time***

  These will be useful for other algorithms later on.
- $v.\pi$ is v's predecessor in the ***depth-first forest*** of ≥ 1 ***depth-first trees***. If $u = v.\pi$, then (u, v) is a ***tree edge***.

DEPTH-FIRST SEARCH (continued)

Methodically explores *every* edge.

- Start over from different vertices as necessary.

As soon as a vertex is discovered, explore from it.

- Unlike BFS, which puts a vertex on a queue so that it's explored from later.

As DFS progresses, every vertex has a ***color***:

- WHITE = undiscovered
- GRAY = discovered, but not finished (not done exploring from it)
- BLACK = finished (have found everything reachable from it)

Discovery and finish times:

- Unique integers from 1 to $2\,|V|$.
- For all v, $v.d < v.f$.

In other words, $1 \le v.d < v.f \le 2\,|V|$.

PSEUDOCODE

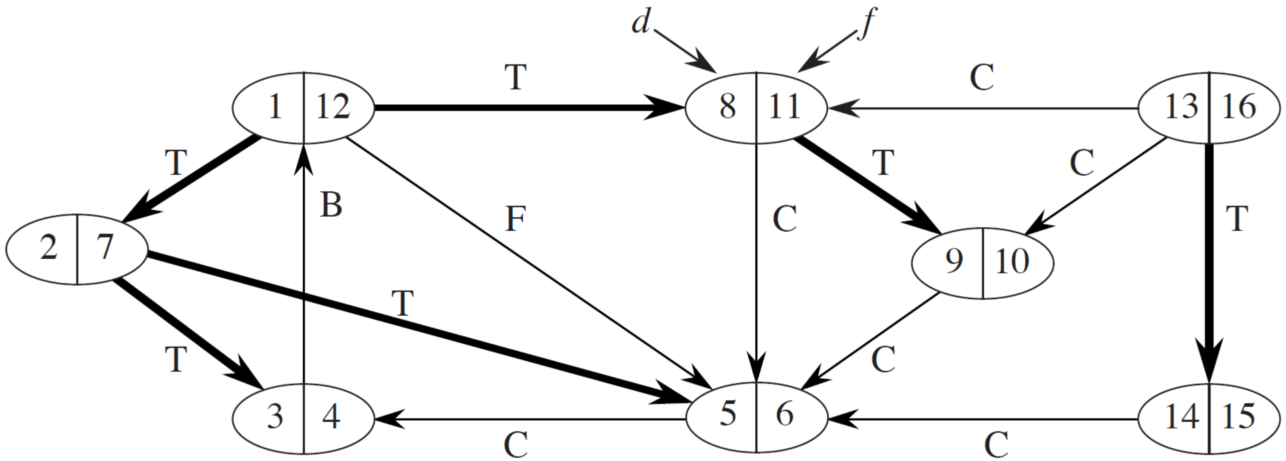
Global search starts a local search on each vertex to explore entire graph.

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

PSEUDOCODE (continued)

```
DFS-VISIT(G, u)
time = time + 1                          // white vertex u has just been discovered
u.d = time
u.color = GRAY
for each vertex v in G.Adj[u]            // explore each edge (u, v)
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
time = time + 1
u.f = time
u.color = BLACK                          // blacken u; it is finished
```

EXAMPLE



Time $= \Theta(V + E)$.

- Similar to BFS analysis.
- Θ, not just O, since guaranteed to examine every vertex and edge.

Each depth-first tree is made of edges (u, v) such that u is gray and v is white when (u, v) is explored.

CLASSIFICATION OF EDGES

- ***Tree edge:*** in the depth-first forest. Found by exploring (u, v).
- ***Back edge:*** (u, v), where u is a descendant of v.
- ***Forward edge:*** (u, v), where v is a descendant of u, but not a tree edge.
- ***Cross edge:*** any other edge. Can go between vertices in same depth-first tree or in different depth-first trees.

In an undirected graph, there may be some ambiguity since (u, v) and (v, u) are the same edge. Classify by the first type above that matches.

THEOREM (PARENTHESIS THEOREM)

For all u, v, exactly one of the following holds:

1. $u.d < u.f < v.d < v.f$ or $v.d < v.f < u.d < u.f$ (i.e., the intervals $[u.d, u.f]$ and $[v.d, v.f]$ are disjoint) and neither of u and v is a descendant of the other.
2. $u.d < v.d < v.f < u.f$ and v is a descendant of u. (v is discovered after and finished before u.)
3. $v.d < u.d < u.f < v.f$ and u is a descendant of v. (u is discovered after and finished before v.)

So $u.d < v.d < u.f < v.f$ (v is both discovered and finished after u) *cannot* happen.