# Depth-First Search

Andreas Klappenecker

# Depth-First Search

```
Input:  G = (V,E)

for each node u in V do

     mark u as unvisited

od;

for each unvisited node u do

    recursiveDFS(u);

od;
```

```
recursiveDFS(u):
  mark u as visited;
  for each unvisited neighbor v of u do
     recursiveDFS(v)
  od
```

Purpose of this loop:
Create DFS forest. Graph can be directed or undirected.

# DFS Forest

By keeping track of parents, we want to construct a forest resulting from the DFS traversal.

# Depth-First Search

```
Input:   G = (V,E)
for each node u in V do
      parent[u] = nil;
      mark u as unvisited
od;
for each unvisited node u do
      parent[u] = u;
      recursiveDFS(u);
od;
```

```
recursiveDFS(u):
   mark u as visited;
   for each unvisited neighbor v of u do
      parent[v] = u; recursiveDFS(v)
   od
```

# Refining DFS

Let us keep track of some interesting information for each node. We will timestamp the steps and record the

- discovery time, when the recursive call starts
- finish time, when its recursive call ends

# Depth-First Search

```
Input:   G = (V,E)
for each node u in V do
        parent[u] = nil;
        mark u as unvisited
od;
time = 0;
for each unvisited node u do
        parent[u] = u;
        recursiveDFS(u);
od;
```

```
recursiveDFS(u):
   mark u as visited;
   disc[u] = ++time;
   for each unvisited neighbor v of u do
      parent[v] = u; recursiveDFS(v)
   od;
   fin[u] = ++time;
```

# Running Time of DFS

The first for-loop for initialization takes O(V) time.

The second for-loop in non-recursive wrapper considers each node, so O(V) iterations.

One recursive call is made for each node. In a recursive call for the node u, all its neighbors are checked; so the total time in all recursive calls is O(E).

Total time is O(V+E).

# Nested Intervals

Define [disc[v],fin[v]] to be the interval for node v.

Claim: For any two nodes, either one interval precedes the other or one is enclosed in the other

Indeed, the recursive calls are nested.

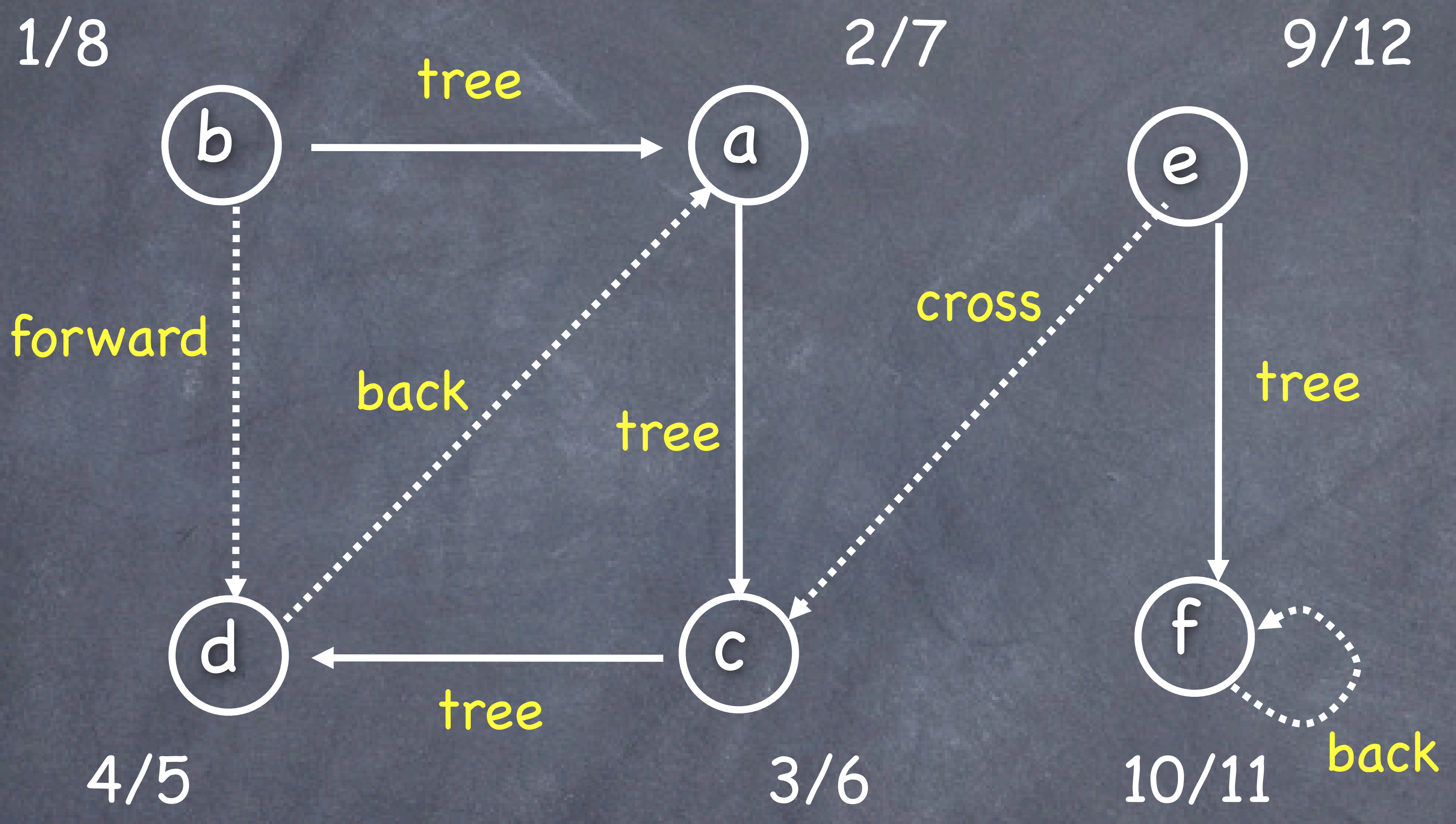
Corollary: v is a descendant of u in the DFS forest iff the interval of v is inside the interval of u.

# Classifying Edges

Consider edge (u,v) in a directed graph G = (V,E) with respect to its DFS forest

- tree edge:  v is a child of u
- back edge: v is an ancestor of u
- forward edge: v is a descendant of u but not a child
- cross edge: none of the above

# Example of Classifying Edges




in DFS forest

not in DFS forest

a/b disc./finish. time

tree edge: v child of u
back edge: v ancestor of u
forward edge: v descendant of u, but not child
cross edge: none of the above