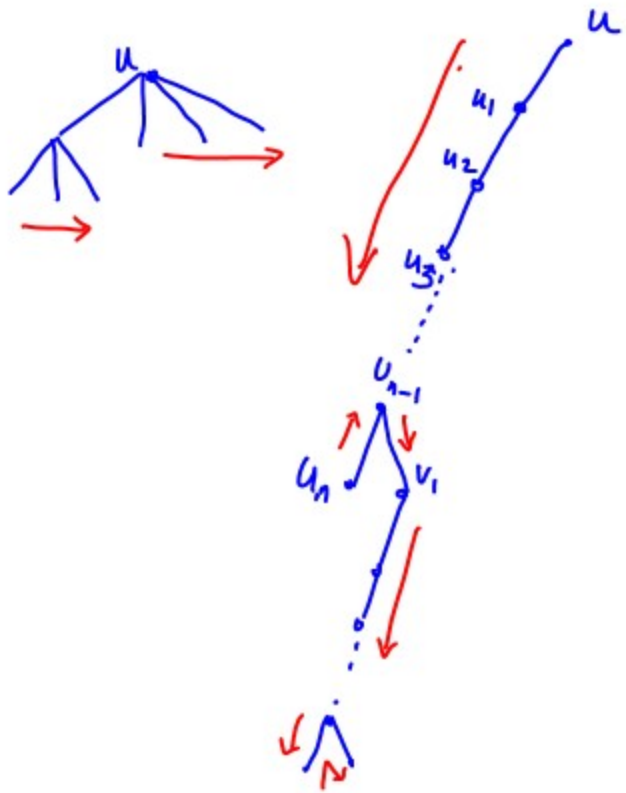
Depth First Search (DFS)



Precise description is given by means of an algorithm (as in the case of BFS)

$v.col$ — White, Grey, Black

$v.\pi$ — Set of DFS trees (DFS forest)

$v.d$, $v.f$ — Timestamps $\in \{1, \ldots, 2|V|\}$

$v.d$ → discovery of vertex v

$v.f$ → Finishing time of vertex v (Backtracking from)

White

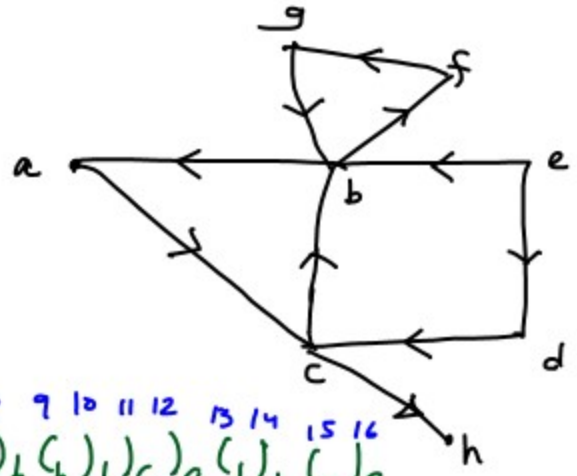
Grey

Black

```
DFS(G)        // G = (V, E)
  for v ∈ V do
      v.col = White
      v.π   = nil
  time = 0
  for v ∈ V do
     if (v.col == White)
          DFS-Visit(G, v)
```

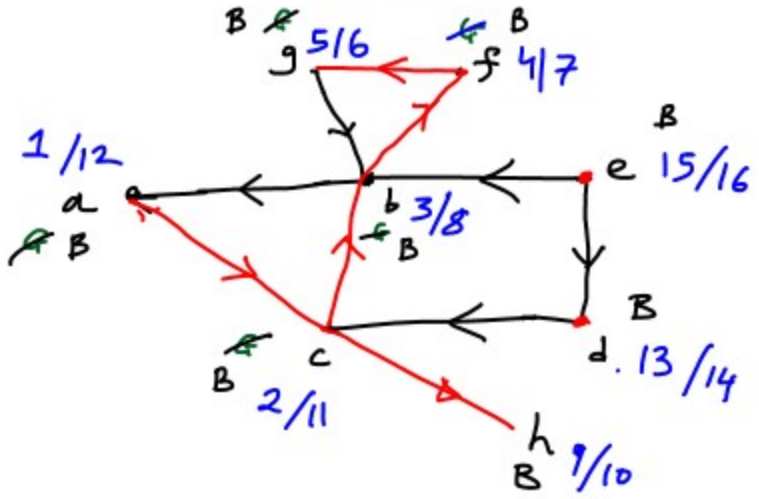
```
DFS-Visit(G, v)
     v.col = Grey
     time  = time+1
     v.d   = time
     for w s.t. (v,w) ∈ E do
        if (w.col == White)
             w.π = v
             DFS-Visit(G, w)
     v.col = Black
     time  = time+1
     v.f   = time
```

// DFS(G)



1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

$(_a(_c(_b(_f(_g)_g)_f)_b(_h)_h)_c)_a(_d)_d(_e)_e$



- b a, g b are backedges
- e b, e d, d c are cross edges
- All the other edges are red and are tree edges

Time Complexity

Aggregate Analysis

- DFS-visit(G, v) is called exactly once from each $v \in V$
- In DFS-visit(G, v), the for loop is executed

$$\sum_{v \in V} |\{(v,w) \in E\}| \text{ time}$$

$$\sum_{v \in V} \text{outdeg}(v) = |E| \quad \text{(in directed graph)}$$

$$= 2 \cdot |E| \quad \text{(in undirected graph)}$$

Total time is

$|V|$ (initialization)

$|V|$ (total DFS-visit(G, v) instructions)

$|E| + 6|V|$ (for all instructions in DFS-visit(G, w) excluding DFS-visit(G, w) instructions, summed over all $v \in V$)

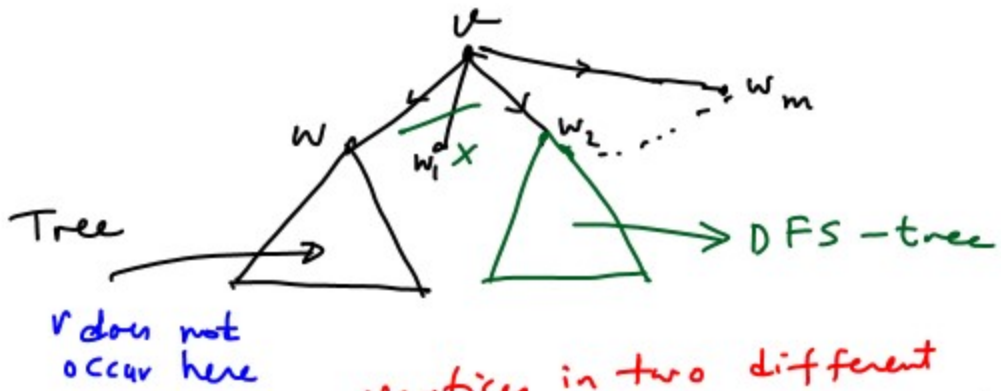
$O(|V| + |E|)$

Properties of DFS

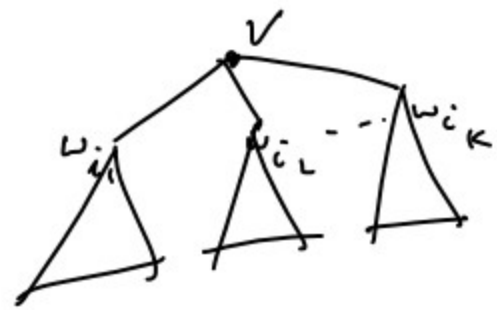
$G_\pi = (V, E_\pi)$, where $E_\pi = \{ (v.\pi, v) \mid \begin{matrix} v \in V, \\ v.\pi \neq nil \end{matrix} \}$

- G_π is a forest of trees.

Any DFS-visit(G,v) call from program DFS(G), makes v as a root ($v.\pi = nil$)



An edge (v,w) is created in G_π iff a call is made to DFS-visit(G,w) from DFS-visit(G,v).



vertices in two different subtrees w_i, w_j are disjoint because call to each DFS(G, w_i) is made exactly once.

Each call to DFS-visit(G, v) from DFS(G) results in a single DFS tree.

All calls to DFS-visit(G, v) from DFS(G) result in a forest of trees (called DFS forest)

- A vertex s is a descendent of v iff s is discovered while $v.col$ is Grey.
  (Whole construction of tree rooted at v happens when v is Grey)

- If we denote by '$(_r$' discovery of vertex r and by '$)_r$' the finishing (or backtracking) of r and list all such events in temporal order then resulting sequence is a well formed parathesized expression.

In our example this sequence is shown along with the example.

An equivalent way to formulate parenthesis property in the following Lemma.

Lemma: For any $u, v \in V$ exactly one of the following holds.

(i) $(u.d, u.f)$ is contained in $(v.d, v.f)$. In this case u is a descendent of v.

(ii) $(v.d, v.f)$ is contained in $(u.d, u.f)$. [$u.d < v.d < v.f < u.f$] In this case v is a descendent of u.

(iii) $(u.d, u.f)$ is disjoint from $(v.d, v.f)$.

Proof: Let $u.d < v.d$ (other case is symmetrical)

Now consider two subcases:

(1) $u.f < v.d$ (2) $u.f > v.d$

Subcase 1: $u.d < u.f < v.d < v.f$

Vertex v is discovered when u is finished

neither u is descendent of v nor v is descendent of u.

$(u.d, u.f) \cap (v.d, v.f)$ are disjoint.

Subcase 2: $u.d < v.d < u.f$

v is discovered while u is grey.

$\Rightarrow$ call to DFS-visit(G, v) is made while executing call to DFS-visit(G, u)

$\Rightarrow$ DFS-visit(G, v) must finish before DFS-visit(G, u)

$\Rightarrow$ $u.d < v.d < v.f < u.f$

$\Rightarrow$ $(v.d, v.f)$ is contained in $(u.d, u.f)$.

Further, v is a descendent of u.

□

DFS Properties (Continued)

u is a descendent of v in DFS forest

iff v is discovered when u is grey

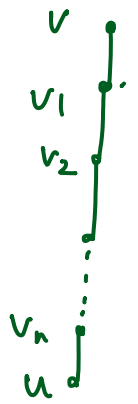
iff $(v.d, v.f) \subseteq (u.d, u.f)$

Another characterization of descendent relation.

Lemma: (White path theorem)
u is a descendent of v iff when v is discovered, there is path consisting of white vertices from v to u.

Proof: $\Rightarrow$ Let u be a descendant of v, via path

$v, v_1, v_2, \ldots, v_n, u$ in a DFS tree.



when v_1 is discovered, it is white,

v_1 is discovered after v is discovered ($\because$ v is parent of v_1)

$\Rightarrow$ v_1 is white when v was discovered

v_2 is white when v_1 was discovered

$\Rightarrow$ v_2 is white when v was discovered.

Similarly all vertices $v_3, \ldots, v_n, u$ are white when v is discovered.

So there was a white path from v to u when v was discovered.

$\Leftarrow$ Suppose when v is discovered there is a white path from v to u, say $v, v_1, \ldots, v_m, u$. $(u = v_{m+1})$

We prove the claim by induction on m.

During DFS(G, v) let v_i $(i \geq 1)$ be the first vertex discovered among $v_1, \ldots, v_m, u$.

[There always exists such vertex, because v_1 being adjacent to v is considered in the for loop of DFS-visit(G, v)]

$v_i, v_{i+1}, \ldots, v_m, u$ are all white vertices.

By I.H. u is a descendent of v_i.

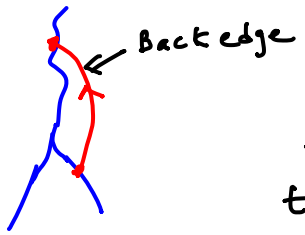
But v_i is a descendent of v.

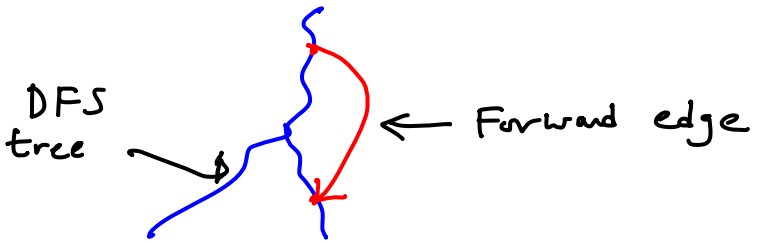
$\Rightarrow$ putting together, u is a descendent of v.

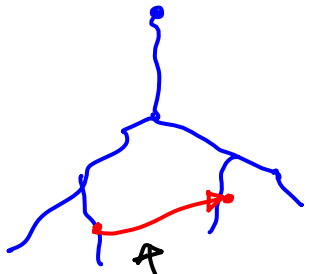
$\square$

Classification of Edges of G (w.r.t. a DFS forest)

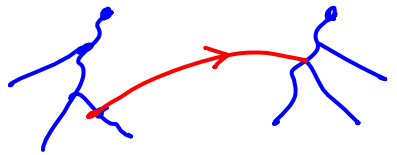
- Tree edges
- Edges (u,v) s.t. u is a descendent (in DFS forest given) of v.
  Back edges
- Edge (u,v) s.t. v is a descendent of u.
  Forward edges
- Edge (u,v) s.t. neither u is a descendent of v, nor v is a descendent of u.
  Cross edges







Cross edge (within same DFS tree)



Cross edge from one DFS tree to another DFS tree.

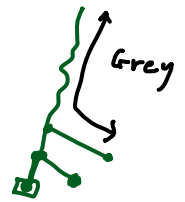
The above classification of edges can be computed during DFS itself.

When we consider edge (v, w), color of w determines the type of edge

- w is white $\Rightarrow$ (v, w) is a tree edge
- w is Grey $\Rightarrow$ (v, w) is a back-edge
- w is black $\Rightarrow$ (v, w) is a forward or a cross edge

The first case is clear.

In the second case, we observe that at any point in DFS all grey vertices are linearly ordered by the ancester relation, and edges are explored from the most recently discovered grey vertex.



$\Rightarrow$ w is an ancester of v.

$\Rightarrow$ (v, w) is a backedge (by defn)

In the third case, w is black.

So w is not an ascester of v.

Only cases remaining are the last two cases of the classification

$\Rightarrow$ (v, w) is either forward or cross edge.

Consider two subcases

(i) w is a descendent of v $\Rightarrow$ forward edge

$\Rightarrow$ v.d < w.d

(ii) v, w are not related by descendant relation. [$\Rightarrow$ cross edge]

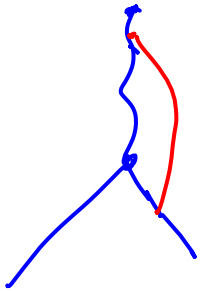
$\Rightarrow$ (v.d, v.f) (w.d, w.f) are disjoint

v is grey, w is black

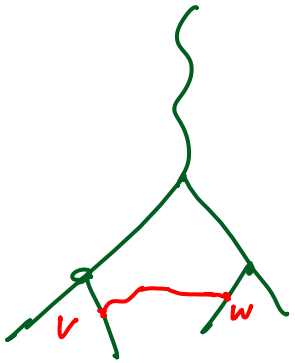
$\Rightarrow$ (w.d, w.f) < (v.d, v.f)

$\Rightarrow$ w.d < v.d

Undirected Graph



No direction
back edge



v must be grey

$\Rightarrow$ w is black (v is not a descendent of w)
v w is not a tree edge

But v must have been explored while visiting w
(as v is adjacent to w).

$(w.d, w.f) < (v.d, v.f)$ $\Rightarrow$ v must get black color before w gets black color

Contradiction.

In an undirected graph, there are only two kinds of edges.

Tree edges and back-edges.