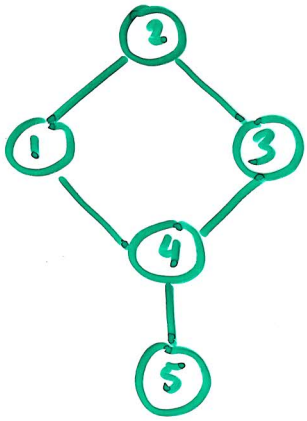
Graph Algorithms

Graphs are nodes connected with edges. Trees are graphs.

Adjacency Matrix representation

The vertices are labeled 1 through n. Each edge corresponds to one entry (or two) of the matrix.

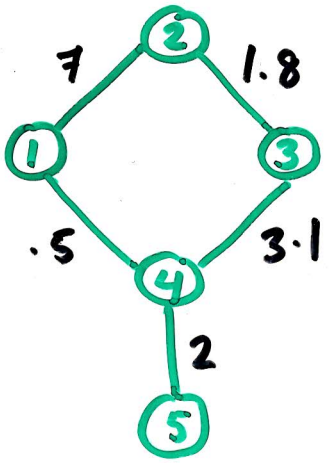


$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

(V, E) is <u>undirected</u> if $E \subseteq \binom{V}{2}$ in which case $A = A^T$.

Each undirected edge is equivalent to two directed edges (u, v) and (v, u).

A graph (V, E) together with a weight function $w: E \longrightarrow \mathbb{R}$ is a weighted graph.



$$A = \begin{bmatrix} 0 & 7 & \infty & .5 & \infty \\ 7 & 0 & 1.8 & \infty & \infty \\ \infty & 1.8 & 0 & 3.1 & \infty \\ .5 & \infty & 3.1 & 0 & 2 \\ \infty & \infty & \infty & 2 & 0 \end{bmatrix}$$

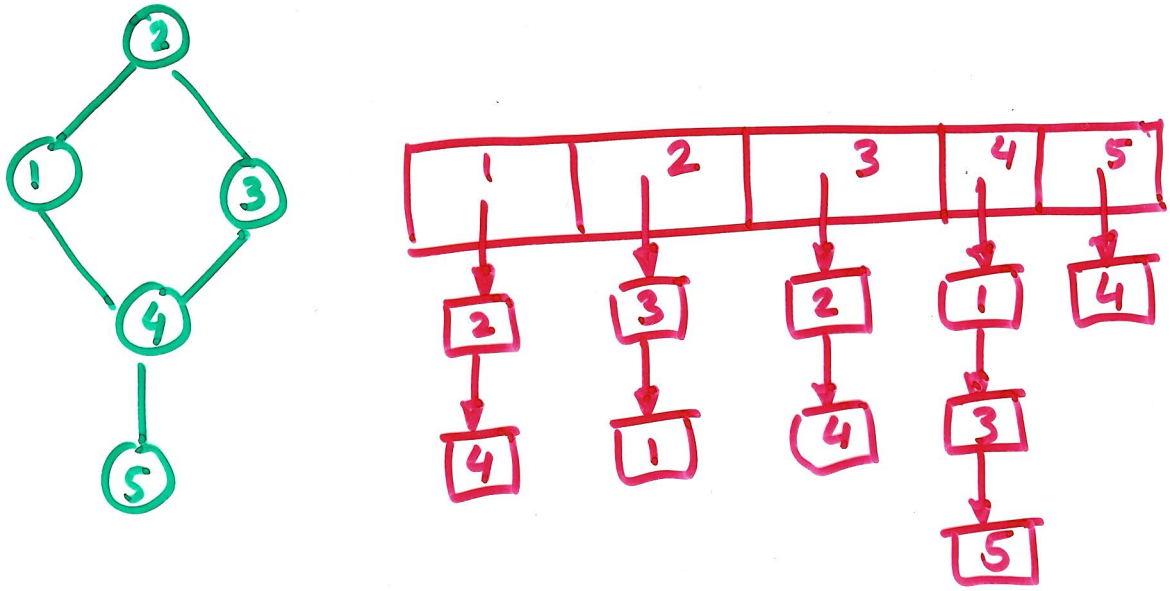
Non-existing edges are identified with weight ∞. The diagonal entries are put as 0. This is suitable when the weights are thought as some kind of distances. But, other variations are possible.

Adjacency Structure

A disadvantage of the adjacency matrix is that it occupies $O(n^2)$ space where $n = |V|$, the number of vertices. A more economic representation is a linked list representation.

The <u>adjacency structure</u> of (V, E) consists of a linear array for V plus a linked list of neighbors for each vertex i (i's adjacency list)

Ex.



It takes $O(n+m)$ space when

$$n = |V|, \quad m = |E|.$$

Variations

(1) If (V,E) is undirected, each edge is represented twice. It might be convenient to link the two corresponding nodes.

(2) If (V,E) is directed, then i's adjacency list stores only its successors.

(3) If (V,E) is weighted, then the weight of an edge is stored in the node representing the edge.

Depth First Search

We use the following data structure

```
Vertex = record  d, f : integer
                 π : index
         end;
Edge = record    v: index;
                 next: ↑Edge
       end;
Adj-struct = array [1...n] of Vertex;

V : Adj-struct;
```

Depth-first-Search (DFS): Traverses all nodes exactly once. The algorithm time stamps each vertex using d (when it is first visited) and f (when all neighbors are finished)

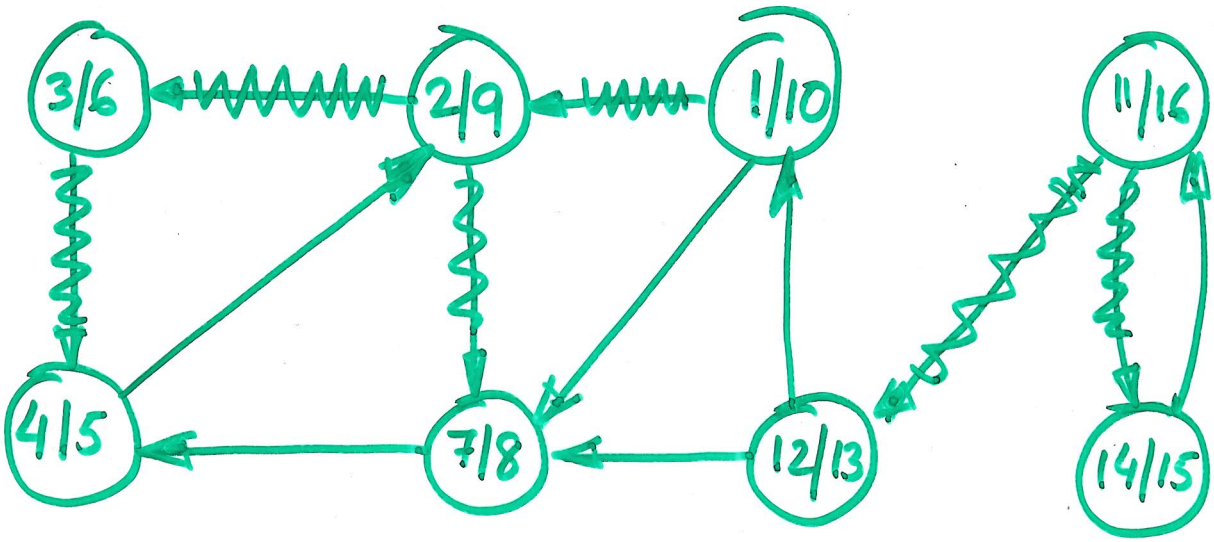
```
for i:=1 to n do V[i].d:=0 endfor ; time:=0;
for i:=1 to n doifV[i].d=0 then
                V[i].π:=nil ;
                Visit(i)
               endif
endfor
```

```
procedure Visit(i);
time:=time+1;  V[i].d:=time;
t:= V[i].adj;
while t≠nil do
      if V[t.v].d=0 then V[t.v].π:=i;
                         visit(t.v)
       endif;
       t:= t.next;
endwhile
time := time+1; V[i].f:=time;
```

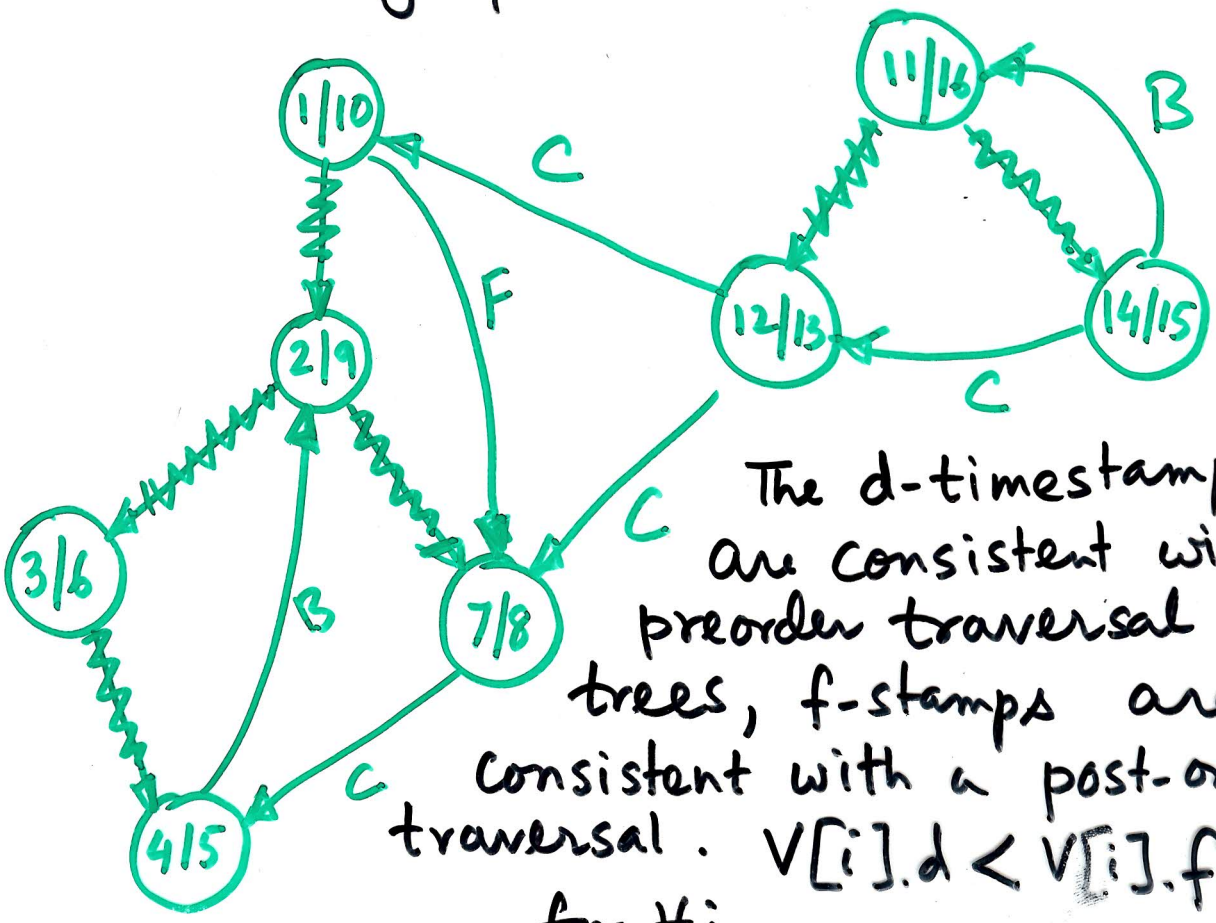
(V and ~~time~~ time are global variable)

$T(n,m) = O(n+m)$

Ex.



The DFS implies a spanning forest of the graph that can be used to redraw the graph.



The d-timestamps are consistent with a preorder traversal of trees, f-stamps are consistent with a post-order traversal. $V[i].d < V[i].f$ for $\forall i$.

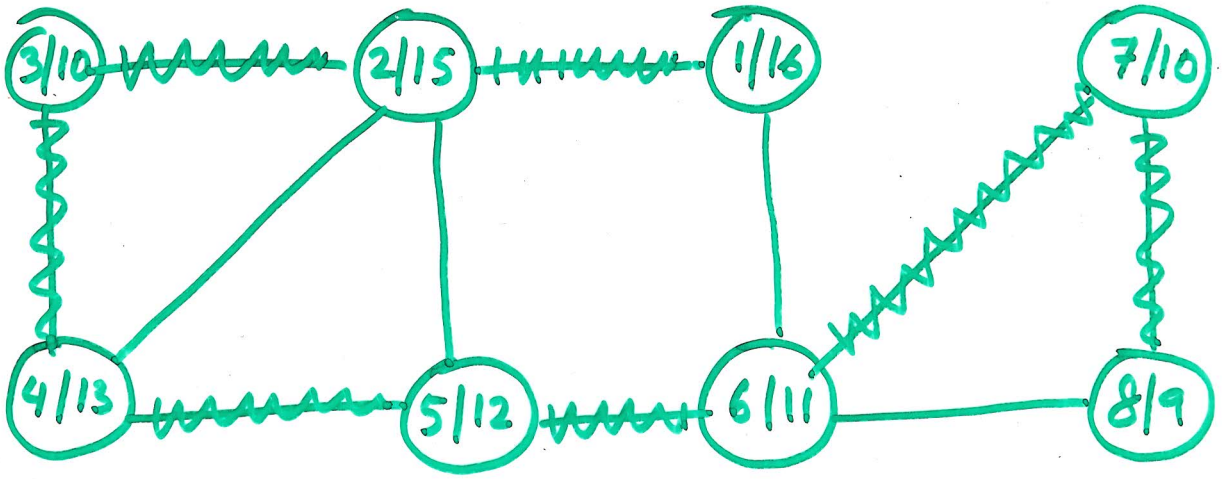
Nesting property.

Vertex j is a proper descendant of vertex i in the DF-forest of (V,E) iff $v[i].d < v[j].d < v[j].f < v[i].f$.

The edge in E can be classified as tree edges (if (i,j) is a tree edge then $v[i].\pi = i$), back edges (edges (i,j) so that j is ancestor of i), forward edges (non-tree edges (i,j) s.t. j is descendant of i), and cross edges (the rest).

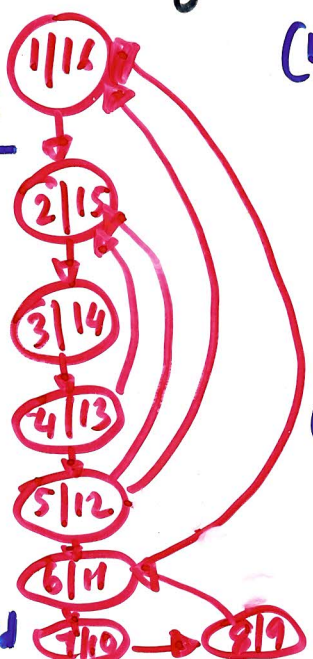
Claim If (V,E) is undirected then there are no cross edges in the DFS of the graph.

Ex.



DFS "imposes" a direction on each undirected edge, because the first time an edge is used, it is done from one end to the other. All edges thus are either tree or back edges.

Vertex u is connected to v if u=v or a neighbor of u is connected to v. A connected component is a maximal subset $U \subseteq V$ s.t. vertices in U are pairwise connected



(1) DFS can be used for connected component determinati

(2) DFS finds cycle in a graph. Every back edge forms a cycle.