Directed Graphs

$G = (V, E)$, $E \subseteq V \times V$ (not necessarily symmetric)

data structure: 2 incidence lists for
- incoming edges → # indegree
- outgoing " → # out degree

Def: G acyclic if it does not contain a (directed) cycle

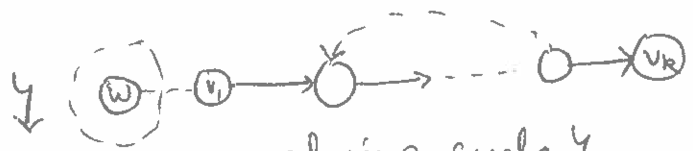
DAG if acyclic & simple

V: jobs, E: dependencies

Wanted: Total order on V s.t. $\forall$ edges $e = (v, w) \in E$: $v \leq w$

Thm: Every DAG G contains a vertex (source) with indegree 0, (target) with outdegree 0.

Proof: Let $p = v_1, \ldots, v_k$ be a longest path in G.



closing cycle ↯

Alg [Topological Sort]

1. Compute indegree $\forall v \in V$ $O(n+m)$
2. Maintain a List of indegree 0-vertices. Let $v \in L$
3. Delete v, update L/neighbours of v.

4. goto 2.

Strongly connected Components

$v, w \in V$ <u>strongly connected</u> if w reachable from v &
v " " w.

<u>Def</u>: <u>Strongly connected component</u> of G: maximal vertex sets of G s.t $\forall v, w \in S$; v, w are strongly connected.

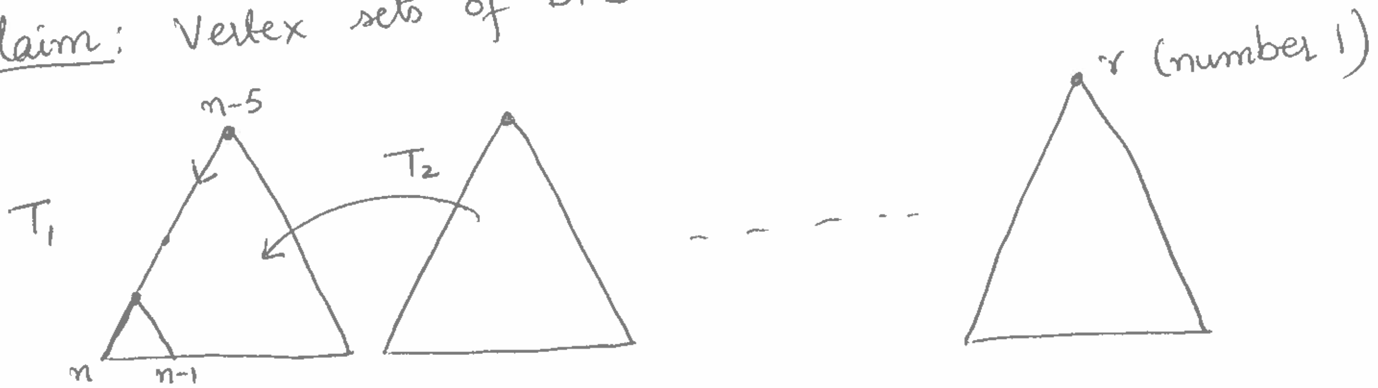
<u>SCC partition V!</u>

<u>Algorithm [SCC]</u> [Ingo Wegener 2002]

1. Do DFS and compute numbering on V s.t a vertex v whose DFS-call is finished <u>later</u> than w gets a <u>smaller</u> number.
2. Obtain G' of G by reversing all edges
3. Do DFS G' where V is sorted by numbering of 1.

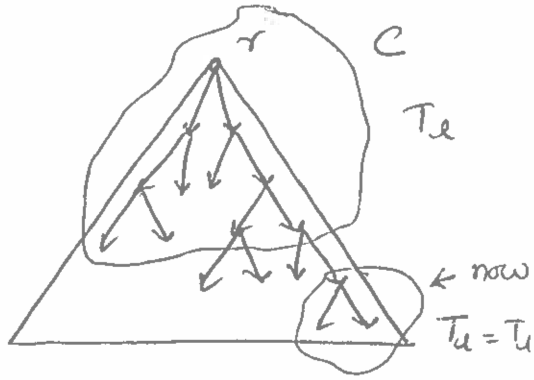
Running time: $O(n+m)$.

<u>Claim</u>: Vertex sets of DFS-trees in 3 are SCC's



Proof of Correctness :-

- Each SCC is contained in some T_i.
- Let C be the SCC containing r.
- $\forall v \in T_l : r \to v \Rightarrow V(C)$ are exactly the vertices in T_l with $v \to r$.



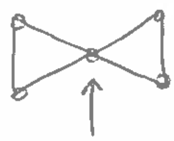
- T_l, because no incoming edges.
- if $v \in C$, $v_1, v_2, \ldots,$ are in C
- induction:
  - If C is only SCC in G, correct
  - otherwise we have trees $T_1, T_2, \ldots, T_{l-1}, \ldots, T_l$

  $\Rightarrow$ induction hypothesis

2-connectivity / 2-edge connectivity (G simple, connected)

Def v is a cut-vertex in G if $G \setminus v$ is disconnected.

$S \subseteq V$ is vertex-cut if $G \setminus S$ is disconnected.



cut-vertex.

Def: G is 2-connected $\Leftrightarrow$ $n > 2$ and G does not contain a cut-vertex.

G is 2-edge-connected $\Leftrightarrow$ $n > 1$ and G does not contain a bridge.

k-connected $\Leftrightarrow$ $n > k$ and G does not contain a vertex cut of size $= k-1$

block = maximal subgraph without cut-vertices and bridges

2-conn.	no	no	yes	no	yes
2-edge conn.	no	no	yes	no	yes
block	yes	yes	yes	no	yes
					

Thm 2-connectivity $\Rightarrow$ 2-edge-connectivity

Thm (a) any two blocks of G share at most one vertex

(b) the blocks partition E

(c) each cycle of G is contained is contained in a block of G.

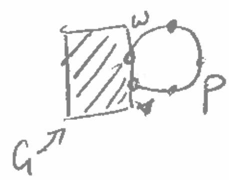
Proof: a) Assume blocks $B_1 \neq B_2$ that both contain v and w

claim: B_1 or B_2 is not maximal

Lemma [The coffee-mug Lemma]

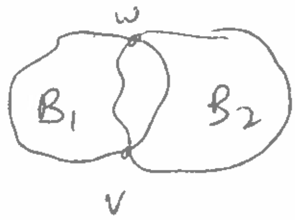
Let G be a block and let P be a path intersecting exactly in its end-points with G.

Then $G \cup P$ is a block.



$|V(G \cup P)| \geq 3$

– no cut-vertex in G and no cut-vertex in P



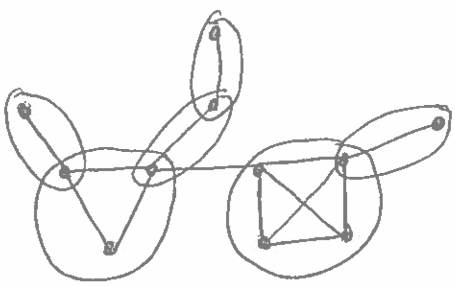
$B_1 \cup P$ is block $\Rightarrow$ ↯ maximality

(b) 

$\Rightarrow$ a) would not be true ↯

(c) 

$\Rightarrow$ ↯ maximality of B_1

G: 

The encircled components are blocks of Graph G.

<u>Block-cut tree</u> of a graph has vertex-set consisting of all block and cut vertices of G.