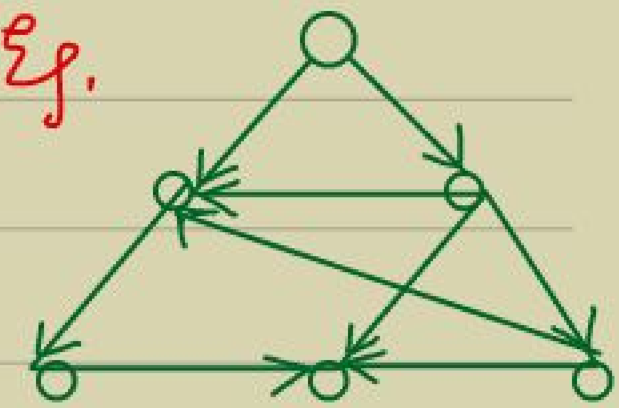
Directed Acyclic Graph (dag)

- **Defn:** A directed graph $G=(V,E)$ is acyclic if there is no cycle in it.

E.g.



τ:

- **Defn:** A topological ordering is a map $\tau: V \to [n]$ s.t. $\forall (u,v) \in E$, $\tau(u) < \tau(v)$.

- Questions: Does τ exist?
  Computable efficiently?
  Its use?

▷ τ is used to compute the shortest paths, & even count the number of paths, in a DAG in linear time!

Lemma: A dag has a vertex of in-deg $= 0$.

Pf:

- Let G be a dag.
- Start with any $v \in V$.
- If it has a pre-neighbor u, i.e. $(u,v) \in E$ then move to u.
  Else v has in-deg $= 0$ & done.
- Continuing this way the process stops in $\leq |V|-1$ steps with an in-deg $= 0$ vertex.
- Note: The process cannot get in a cycle.

$\square$

Theorem: Dag has a top. ordering τ.

Pf:

- Let $v \in V$ be an in-deg $= 0$ vertex.
- Define $\tau(v) := i := 1$.
- Remove v from V & its outgoing edges from E. Graph G remains a dag.
- Repeat the process with the least label $i \leftarrow i+1$.
- At $i = |V|$ all the vertices are labelled.

- Moreover, $\forall (u,v) \in E$, $\tau(u) < \tau(v)$ is maintained in every iteration, for the labelled vertices u, v. $\square$

▷ Naively, τ takes $O(|V|^2)$ time to compute.

– Can this be improved?

– Yes: Find the in-deg $= 0$ vertex faster using a queue data structure. (array)

– Given dag $G = (V, E)$.
- Create a queue $Q \leftarrow \phi$;
- For each $x \in V$
  if (in-deg$(x) = 0$) Enqueue(x, Q);
- $i \leftarrow 1$;
- While $(Q \neq \phi)$ {
  $v \leftarrow$ Dequeue(Q);
  $\tau(v) \leftarrow i$; $i \leftarrow i+1$;

(contd.)

- For each $(v, x) \in E$ {
    $\text{in-deg}(x) \leftarrow \text{in-deg}(x) - 1$;
    if ($\text{in-deg}(x) = 0$) Enqueue(x, Q);
  }

  return τ;

– Correctness: It is labelling in-deg $= 0$ vertex & removing it.

– Time: The two loops have number of runs $\leq |V| + \sum_{v \in V} \deg(v) = |V| + 2|E|$.

Theorem (Knuth 1968): Topological sorting can be done in $O(|V| + |E|)$ time.

Theorem: Single-source shortest paths, in a dag, can be found in linear time.

Proof:

- Given a dag $G = (V, E, w, s)$ do topological sorting.

- Initialize $L(s) \leftarrow 0$ & $L(v) \leftarrow \infty$, $\forall v \in V$.
- $\forall u \in V$ in order & not behind s {

$\quad\quad \forall (u,v) \in E$ {

$\quad\quad\quad$ if $(L(v) > L(u) + w(u,v))$

$\quad\quad\quad\quad L(v) \leftarrow L(u) + w(u,v)$;

$\quad\quad$ }

}

- This works because, inductively, we are correctly computing $L(u)$ for nodes closer to s in the topological order.

- Time: $O\left(|V| + \sum_{u \in V} deg(u)\right) = O(|V| + |E|)$. $\square$

Theorem: Number of paths $s \rightsquigarrow t$, in a dag, can be computed in linear time.

Proof:

- Given a dag $G = (V, E, s)$ do topological sorting.

- Initialize $L(v) \leftarrow 0$, $\forall v \in V$;
- $\forall u \in V$ in order & not behind s {

  $\forall (u,v) \in E$ {

  $L(v) \leftarrow L(v) + L(u)$;

  }

  }

- By induction on the #hops from s, we can see that $L(u) =$ #paths $s \rightsquigarrow u$.

- Time: $O(|V| + |E|)$. $\square$

Theorem: Given dag $G = (V,E)$ & vertices $x_1, \ldots, x_k$ we can compute the number of paths of the form $x_1 \rightsquigarrow x_2 \rightsquigarrow \cdots \rightsquigarrow x_{k-1} \rightsquigarrow x_k$ in linear time.

Proof:

- Do a topological sort on G.
- Check that $x_1 < x_2 < \cdots < x_k$.
  Otherwise the answer is zero.
- Let n_i be the number of vertices between x_i to x_{i+1}, $\forall i \in [k-1]$.

- Let m_i be the respective number of edges.
- We can compute $p_i := \#\text{paths } x_i \rightsquigarrow x_{i+1}$ in $O(n_i + m_i)$ time.

$\Rightarrow \{p_i \mid i \in [k-1]\}$ can be computed in $\sum_{i \in [k-1]} O(n_i + m_i) = O(|V| + |E|)$ time.

$\Rightarrow$ Number of paths of the form $x_1 \rightsquigarrow x_2 \rightsquigarrow \cdots \rightsquigarrow x_k$ is $\prod_{i \in [k-1]} p_i$.

$\square$