Topological Sort and SCCs

Subhabrata Samajder

IIIT, Delhi

Winter Semester,

24^{th} May, 2023

Topological Sort on a Directed Acyclic Graph

Definition

Definition (Topological Sort)

A **topological sort** of a directed acyclic graph (DAG) $G = (V, E)$ is a linear ordering of all its vertices such that if G contains an edge (u, v), then u appears before v in the ordering.

Definition

Definition (Topological Sort)

A **topological sort** of a directed acyclic graph (DAG) $G = (V, E)$ is a linear ordering of all its vertices such that if G contains an edge (u, v), then u appears before v in the ordering.

Note:

- No linear ordering is possible, if the graph is not acyclic.
- Can be viewed as an ordering of its vertices along a horizontal line so that all directed edges go from left to right.
- $\therefore$ different from the usual kind of "sorting" studied earlier.

Applications

- Scheduling a sequence of jobs or tasks based on their dependencies.
  - The jobs are represented by vertices.
  - There is an edge from x to y if job x must be completed before job y can be started.

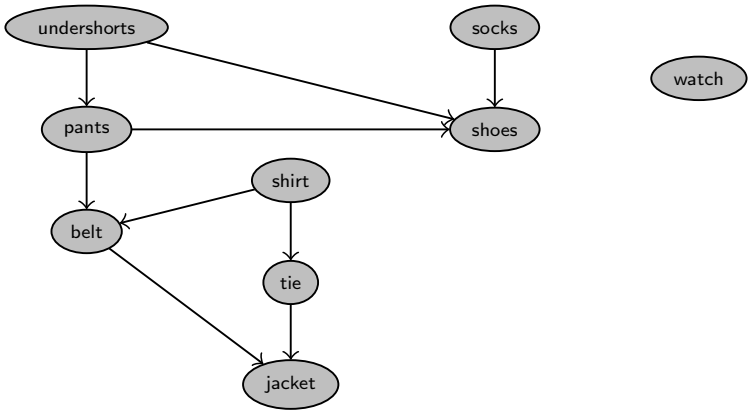
- Then, a topological sort gives an order in which to perform the jobs.

- It forms the basis of linear-time algorithms for finding the critical path of the project.

Applications

- **Examples:**
  - Instruction scheduling in compiler optimization.
  - Ordering of formula cell evaluation when recomputing formula values in spreadsheets.
  - Logic synthesis.
  - Determining the order of compilation tasks to perform in make-files.
  - Data serialization
  - Resolving symbol dependencies in linkers.
  - To decide in which order to load tables with foreign keys in databases.

An Example

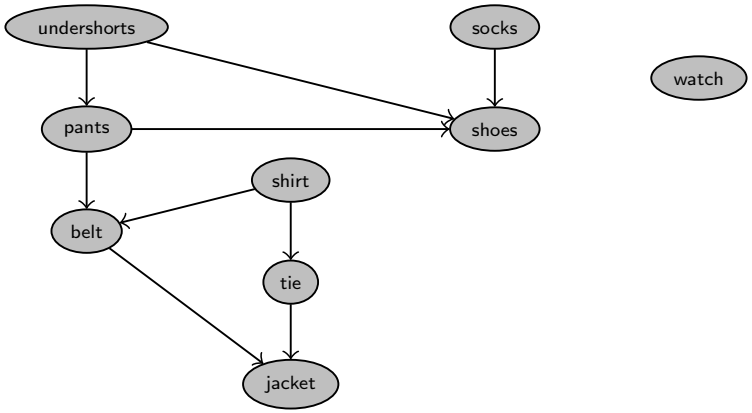


Suppose that Professor Bumstead gets dressed in the morning.

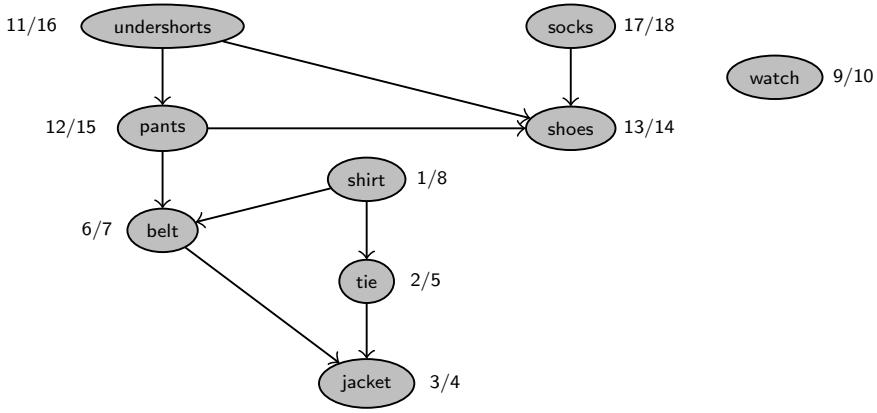
- The professor must don certain garments before others (e.g., socks before shoes).
- Other items may be put on in any order (e.g., socks and pants).
- A directed edge (u, v) in the figure indicates that garment u must be donned before garment v.

A topological sort of this DAG then gives an order for getting dressed.

Topological-Sort(G)

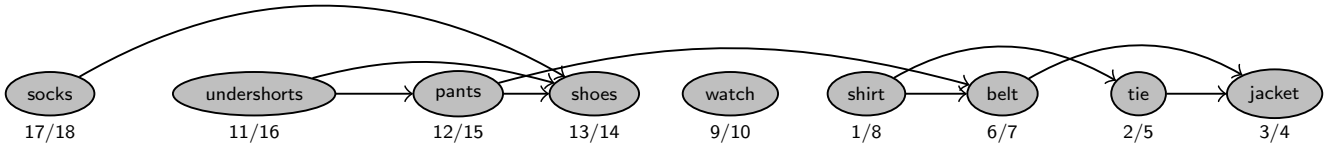


Topological-Sort(G)



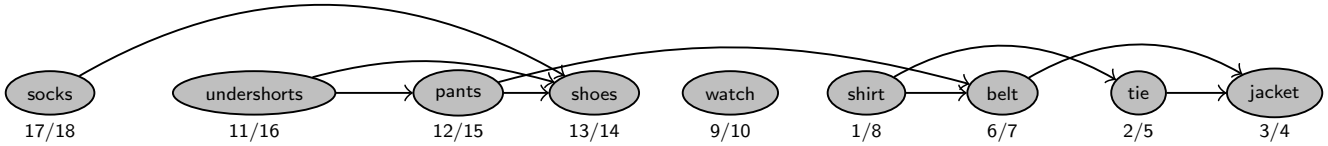
1. Call $\mathrm{DFS}(G)$ to compute finishing times $f[v]\ \forall\ v \in V$
2. As each vertex is finished, insert it onto the front of a linked list

Topological-Sort(G)



1. Call $\mathrm{DFS}(G)$ to compute finishing times $f[v]\ \forall\ v \in V$
2. As each vertex is finished, insert it onto the front of a linked list
3. **return** the linked list of vertices

Topological-Sort(G)



1. Call $\mathrm{DFS}(G)$ to compute finishing times $f[v]\ \forall\ v \in V$
2. As each vertex is finished, insert it onto the front of a linked list
3. **return** the linked list of vertices

Note: It sorts the vertices in reverse order of their finishing times.

Complexity

- Depth-first search takes $\Theta(|V| + |E|)$ time.
- It takes $\mathcal{O}(1)$ time to insert each of the $|V|$ vertices onto the front of the linked list.
- **Total Complexity:** $\Theta(|V| + |E|) + |V| \cdot \mathcal{O}(1) = \Theta(|V| + |E|)$.

Lemma

The correctness of TOPOLOGICAL-SORT uses the following key lemma.

Lemma

A directed graph G is acyclic if and only if a depth-first search of G yields no back edges.

Note: The Lemma gives a characterization of a DAG.

White-path Theorem

Gives an important characterization of when one vertex is a descendant of another in the depth-first forest.

Theorem (White-path Theorem)

In a depth-first forest of a (directed or undirected) graph $G = (V, E)$, vertex v is a descendant of vertex u if and only if at the time $d[u]$ that the search discovers u, vertex v can be reached from u along a path consisting entirely of white vertices.

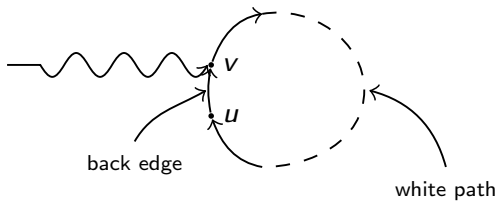
Note: Required to prove the converse of the lemma.

Proof of the Lemma

Suppose that there is a back edge (u, v), then

- vertex v is an ancestor of vertex u in the depth-first forest.
- There is thus a path from v to u in G, and the back edge (u, v) completes a cycle.
- Which is a contradiction to G begin acyclic.
- Therefore G cannot have a back edge.

Proof of the Lemma



Conversely, let, if possible, G contains a cycle C.

We show that a depth-first search of G yields a back edge.

- Let v be the first vertex to be discovered in c, and let (u, v) be the preceding edge in C.
- At time $d[v]$, the vertices of C form a path of white vertices from v to u.
- By the white-path theorem, vertex u becomes a descendant of v in the depth-first forest.
- Therefore, (u, v) is a back edge.

Correctness

Theorem

TOPOLOGICAL-SORT(G) *produces a topological sort of a directed acyclic graph* G.

Proof of the Theorem

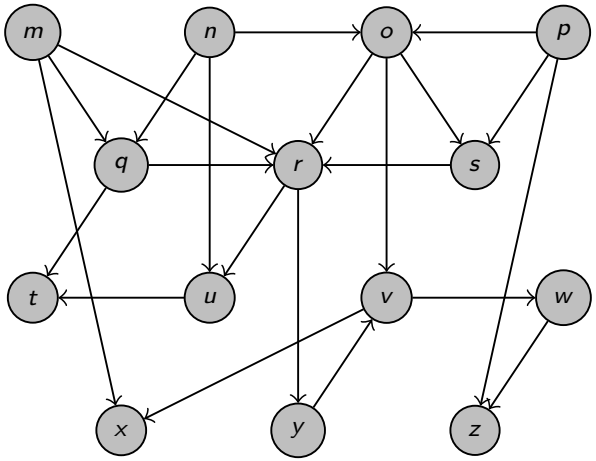
Suppose that DFS is run on a given DAG $G = (V, E)$ to determine finishing times for its vertices.

It suffices to show that for any pair of distinct vertices $u, v \in V$, if there is an edge in G from u to v, then $f[v] < f[u]$.

- Consider any edge (u, v) explored by $\mathrm{DFS}(G)$.
  - **Note:** v cannot be gray.
  - Otherwise, v would be an ancestor of u and (u, v) would be a back edge $\Rightarrow$ cycle $\Rightarrow\Leftarrow$!
- Therefore, v must be either white or black.
  - If v is white, it becomes a descendant of u, and so $f[v] < f[u]$.
  - If v is black $\Rightarrow$ v is finished $\Rightarrow$ $f[v]$ is set.
    - Still exploring from $u \Rightarrow f[u]$ is yet to be assigned.
    - Once $f[u]$ is set $\Rightarrow f[v] < f[u]$ as well.
- $\therefore$ for any edge $(u, v) \in E$, $f[v] < f[u]$.

Exercise

Show the ordering of vertices produced by TOPOLOGICAL-SORT when it is run on the following DAG assuming that the DFS procedure considers the vertices in alphabetical order, and also assume that each adjacency list is ordered alphabetically.

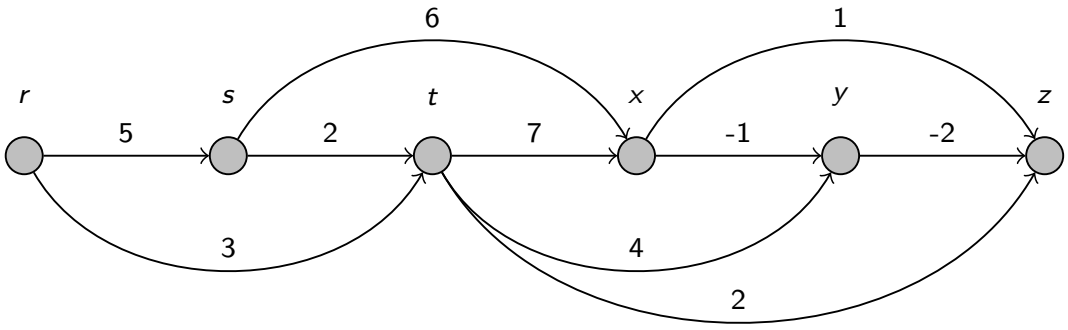


Topological Sort on a Weighted Directed Acyclic Graph

Main Idea

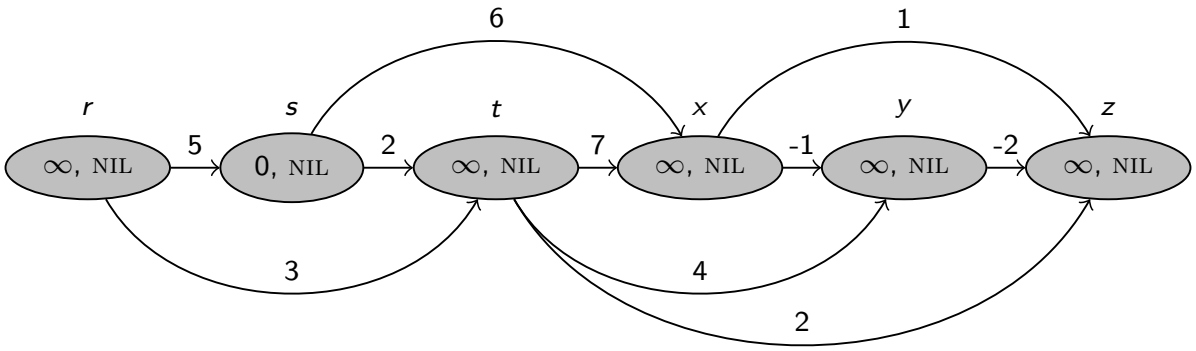
- By relaxing the edges of a weighted DAG $G = (V, E)$ according to a topological sort of its vertices, the shortest paths from a single source s can be computed.
- Shortest paths are always well defined in a DAG.
- This is because even if negative-weight edges are allowed, no negative-weight cycles can exist.
- **Main Idea:**
  - Topologically sort the DAG (as unweighted DAG) to impose a linear ordering on the vertices.
  - That is, if there is a path from vertex u to vertex v, then u precedes v in the topological sort.
  - For each vertex, relax each edge that leaves the vertex.

DAG-Shortest-Paths(G, w, s)



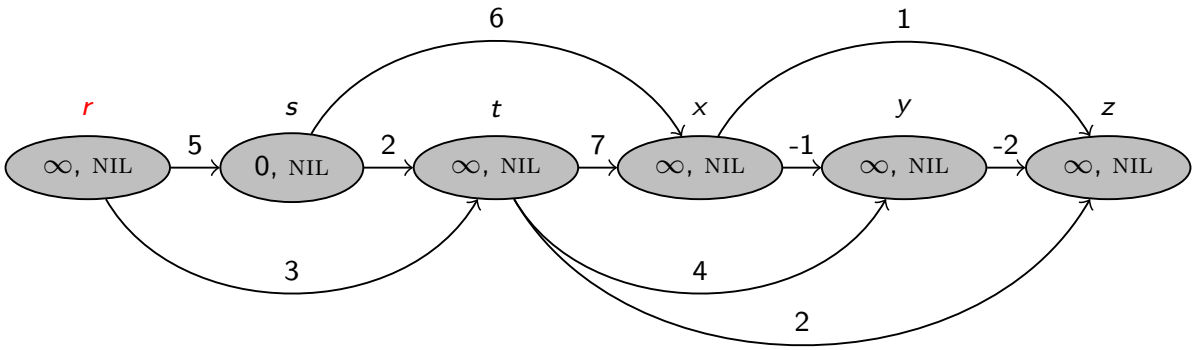
1. Topological-Sort(G)

DAG-Shortest-Paths(G, w, s)



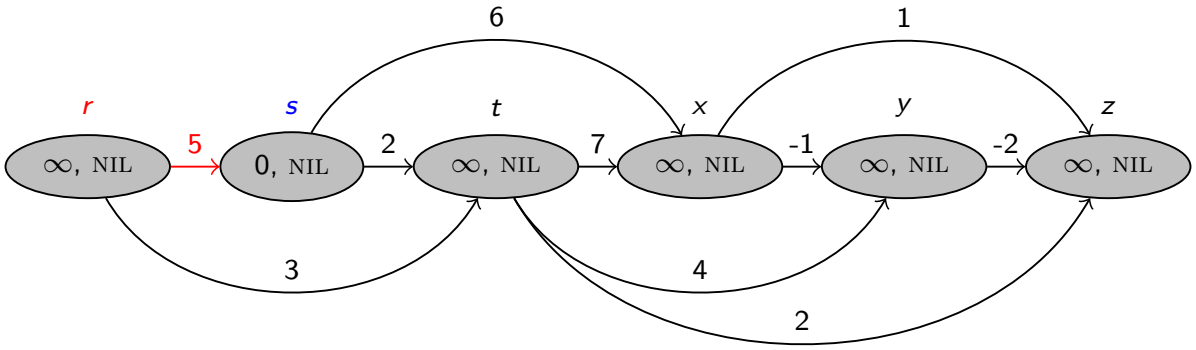
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$

DAG-Shortest-Paths(G, w, s)



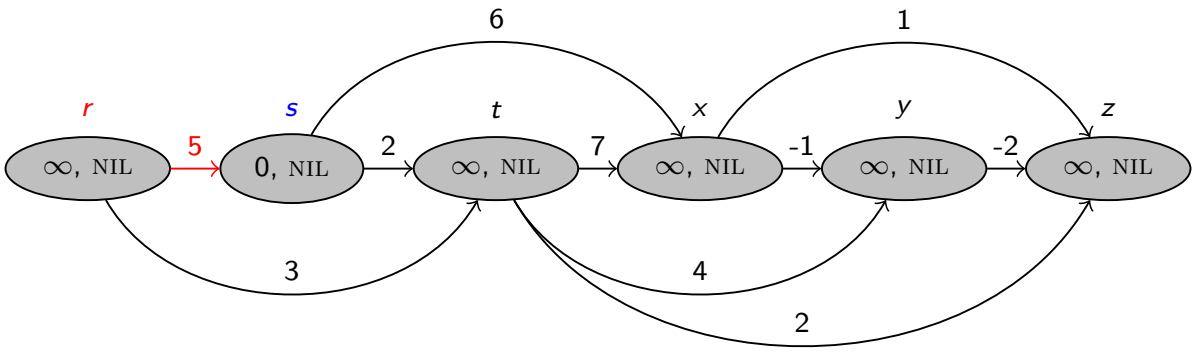
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order

DAG-Shortest-Paths(G, w, s)



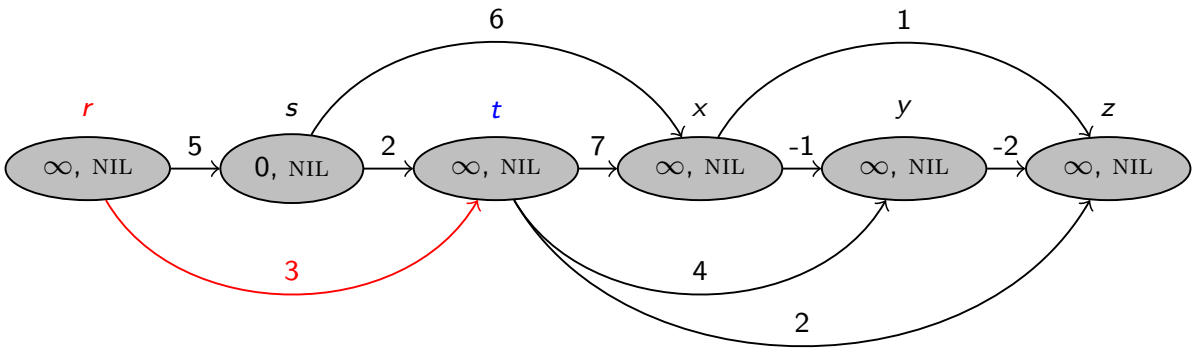
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$

DAG-Shortest-Paths(G, w, s)



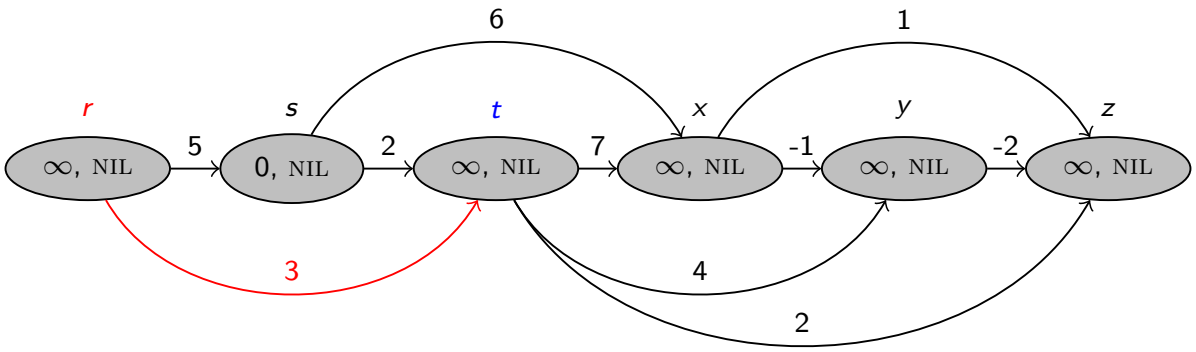
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$

DAG-Shortest-Paths(G, w, s)



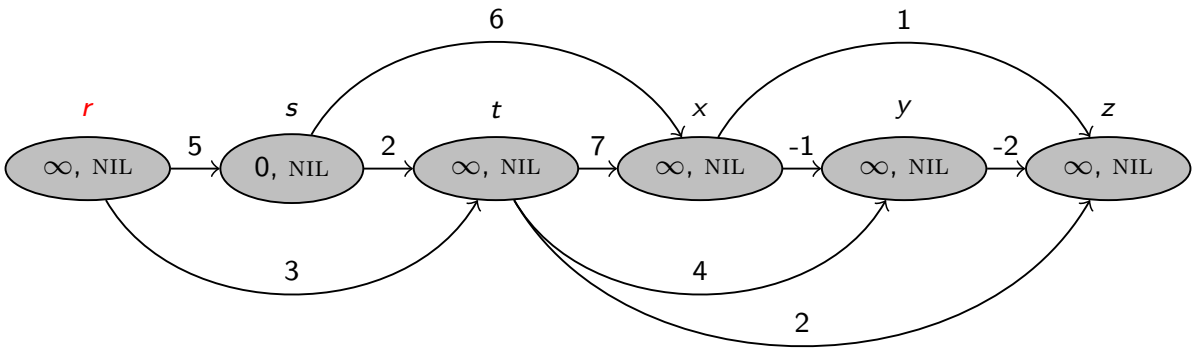
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$

DAG-Shortest-Paths(G, w, s)



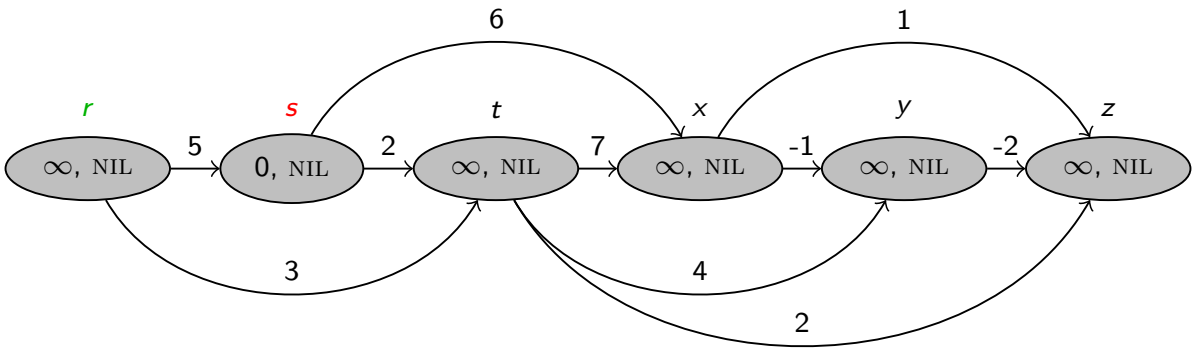
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$

DAG-Shortest-Paths(G, w, s)



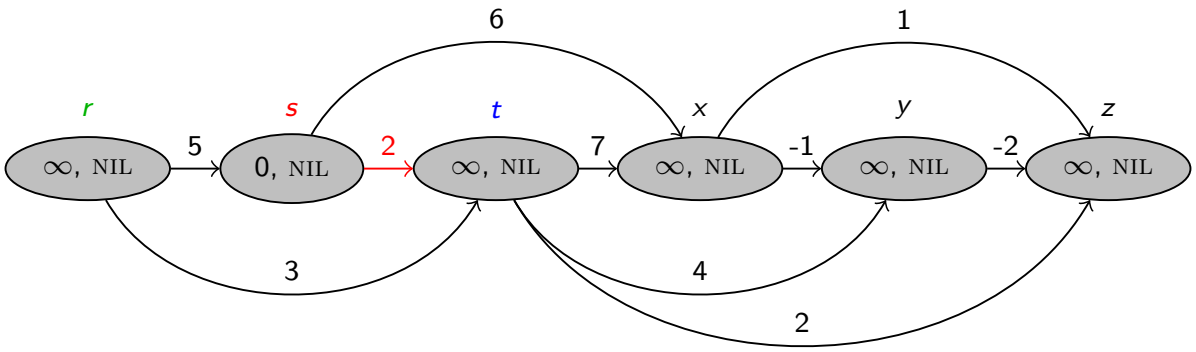
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$

DAG-Shortest-Paths(G, w, s)



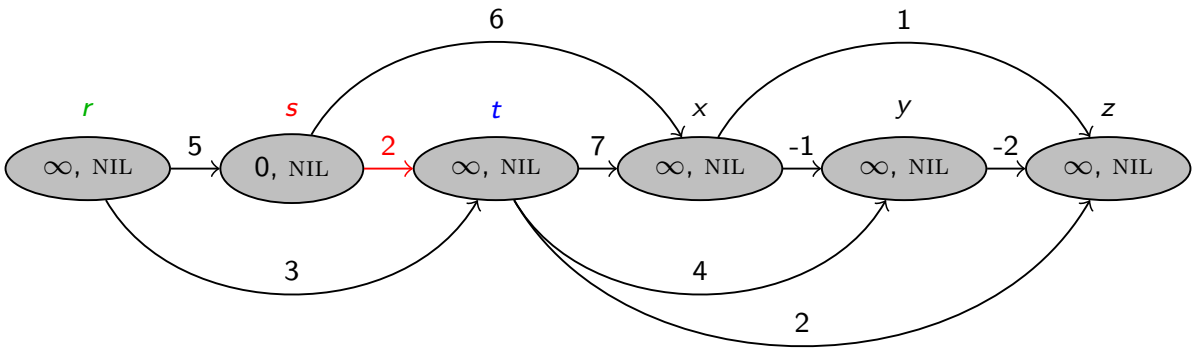
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$

DAG-Shortest-Paths(G, w, s)



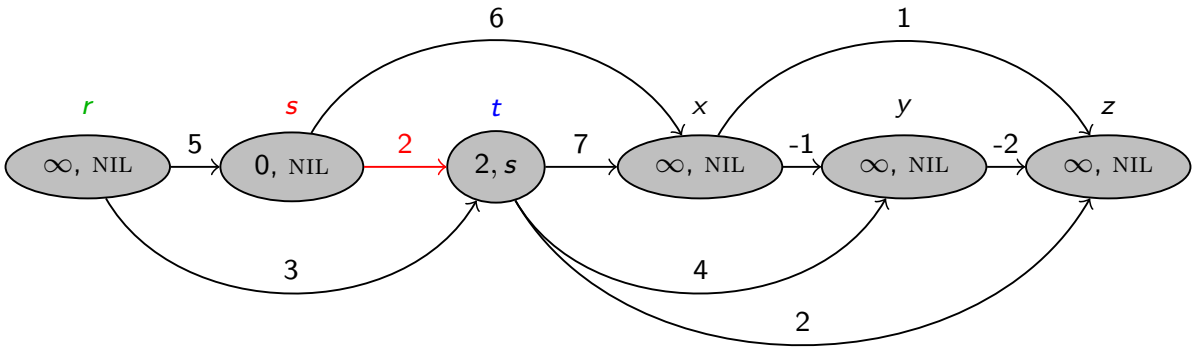
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$

DAG-Shortest-Paths(G, w, s)



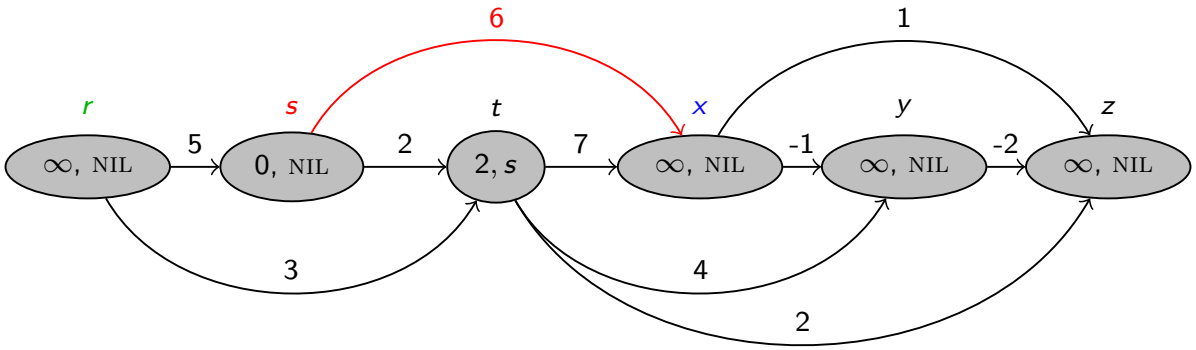
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$

DAG-Shortest-Paths(G, w, s)



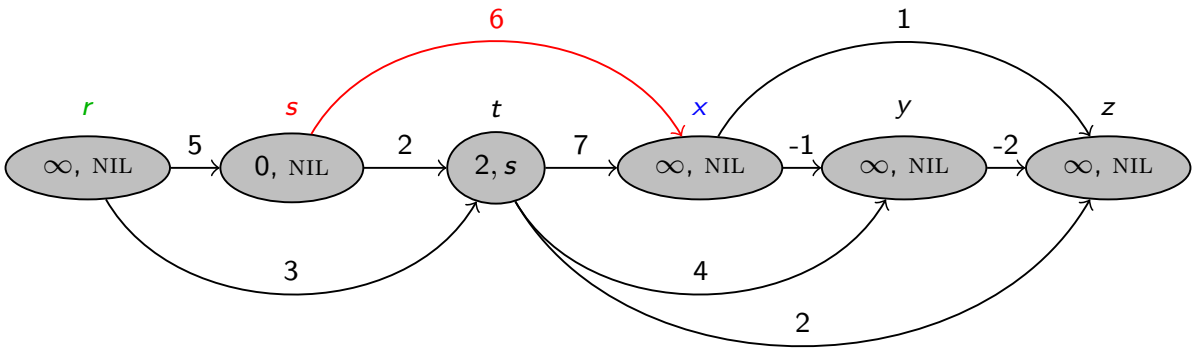
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



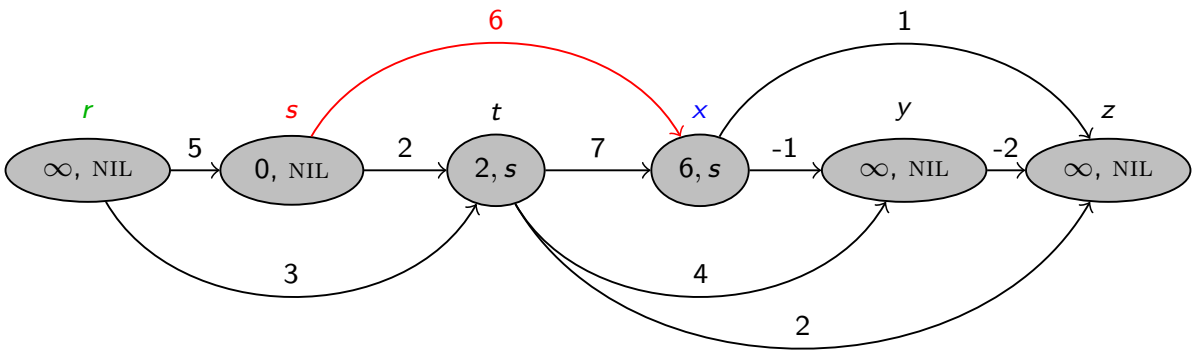
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



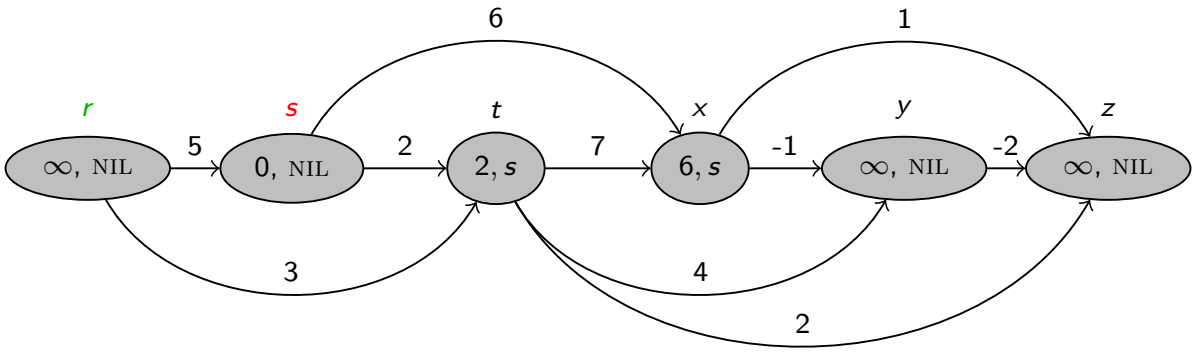
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



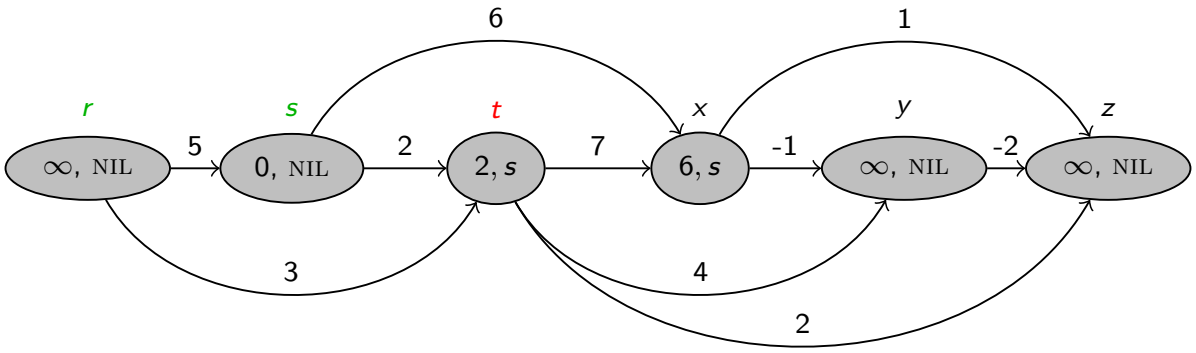
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



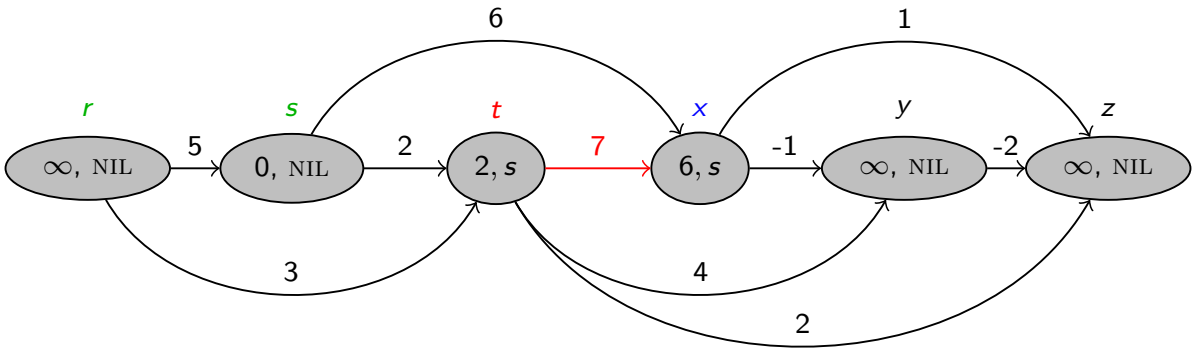
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



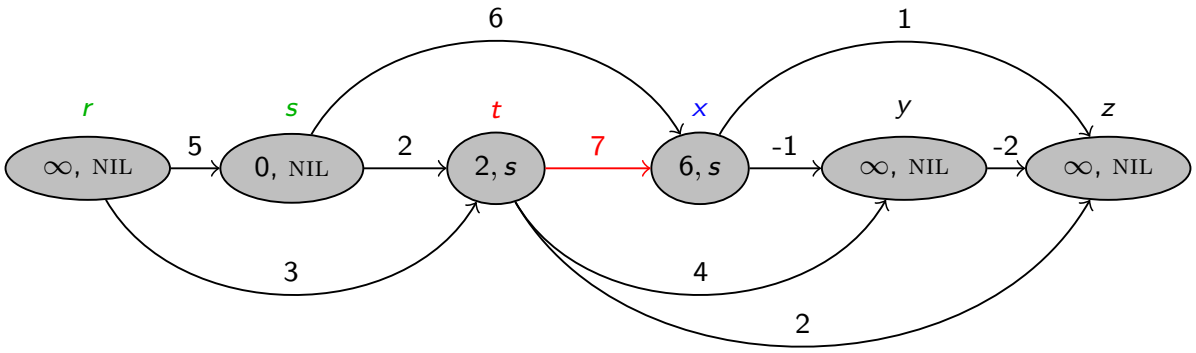
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



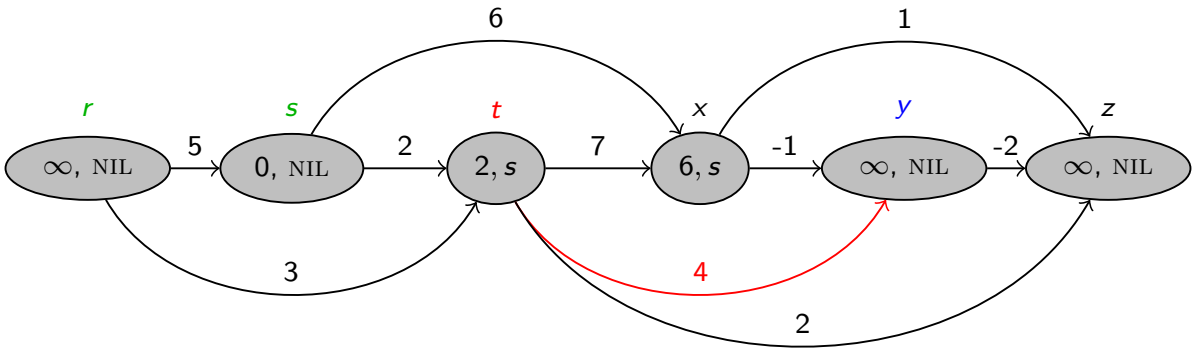
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



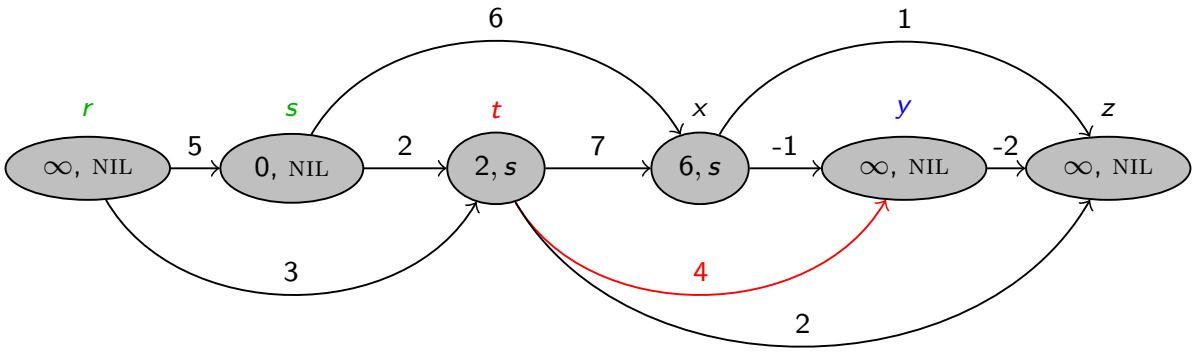
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



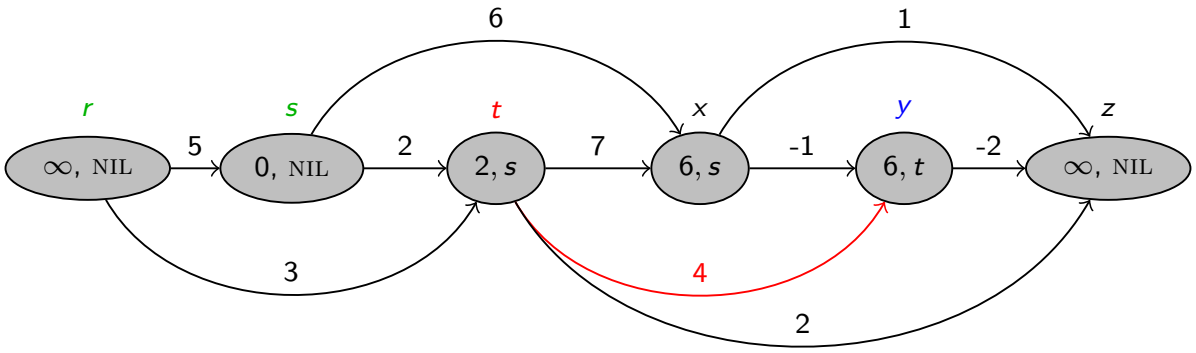
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



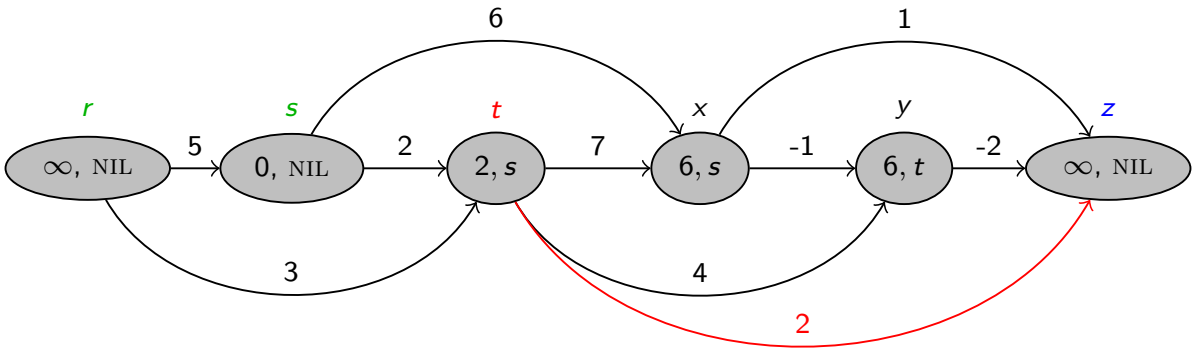
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



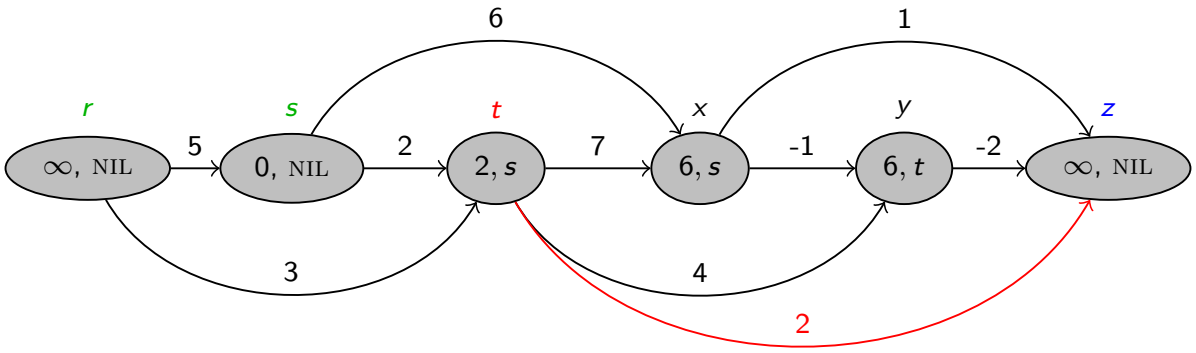
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



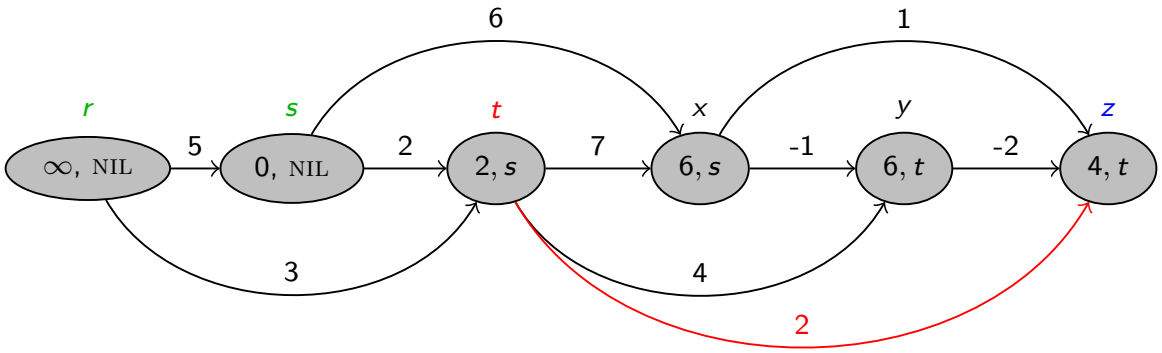
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



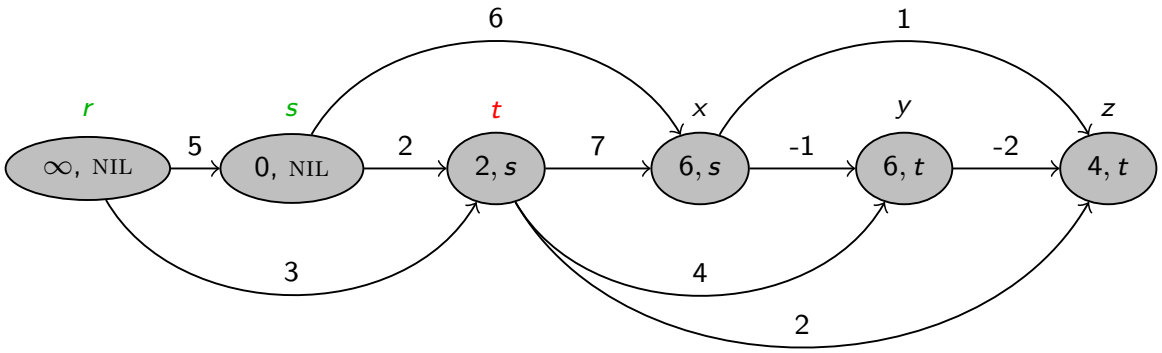
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



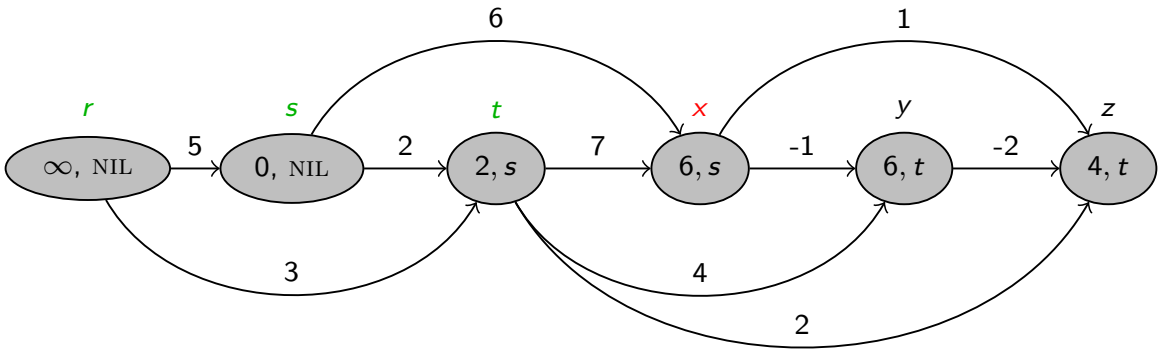
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



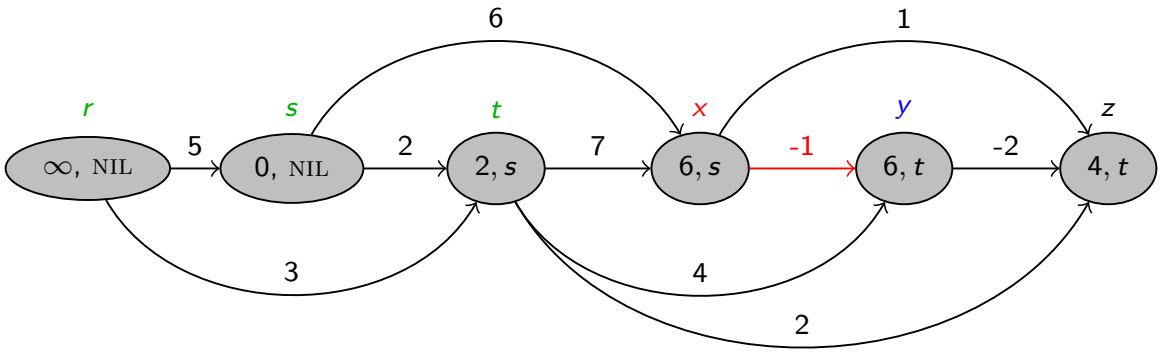
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



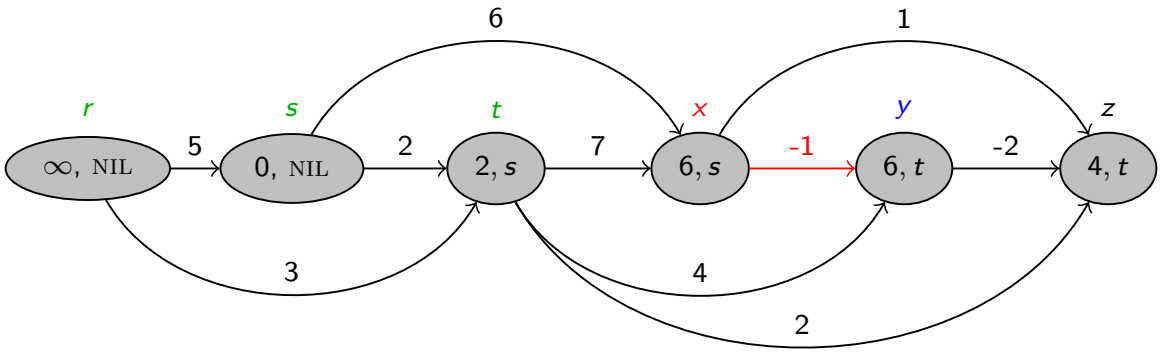
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



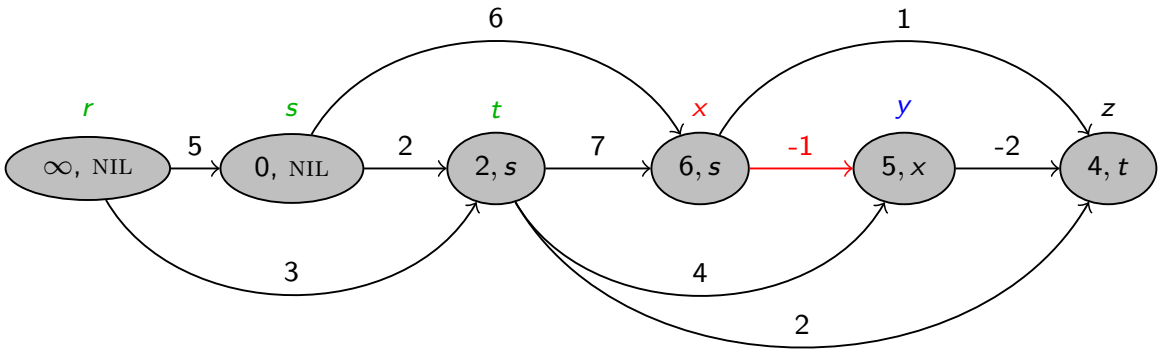
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



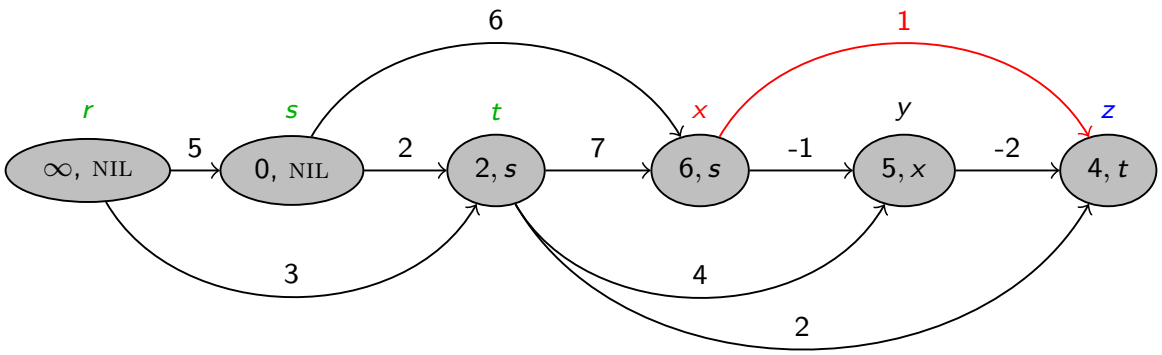
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



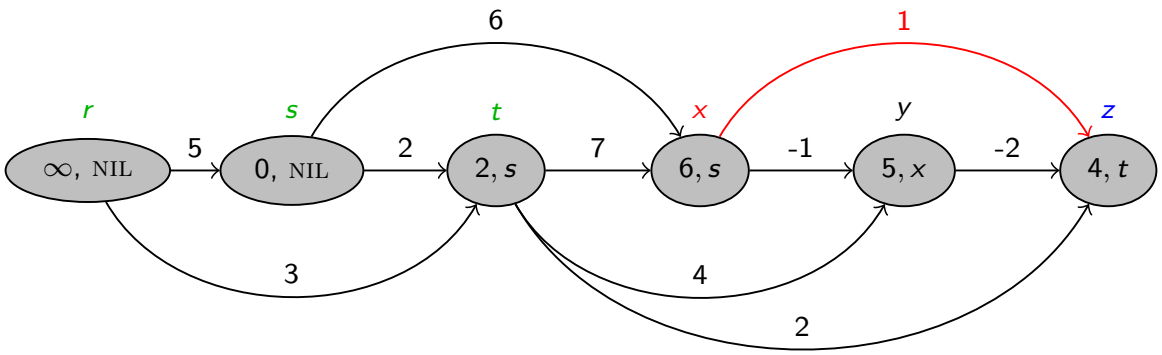
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



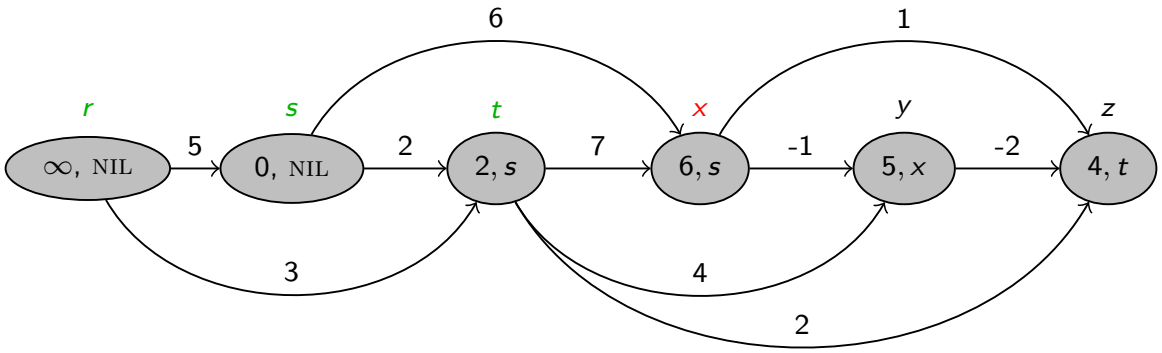
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



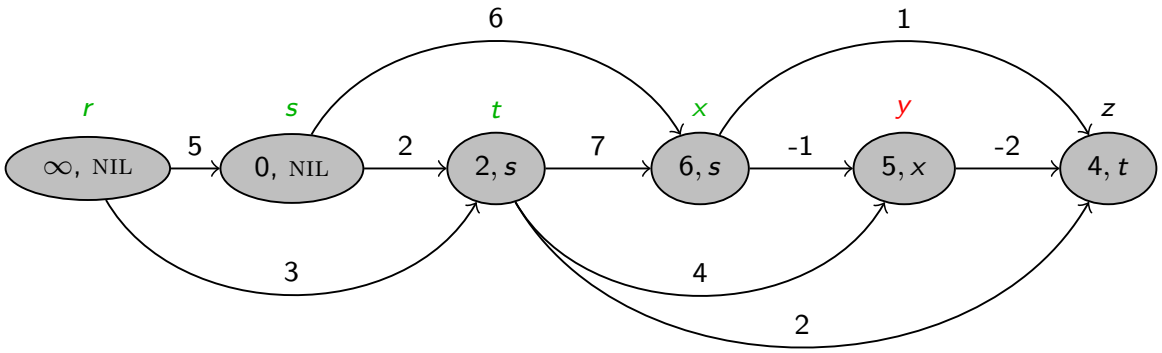
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



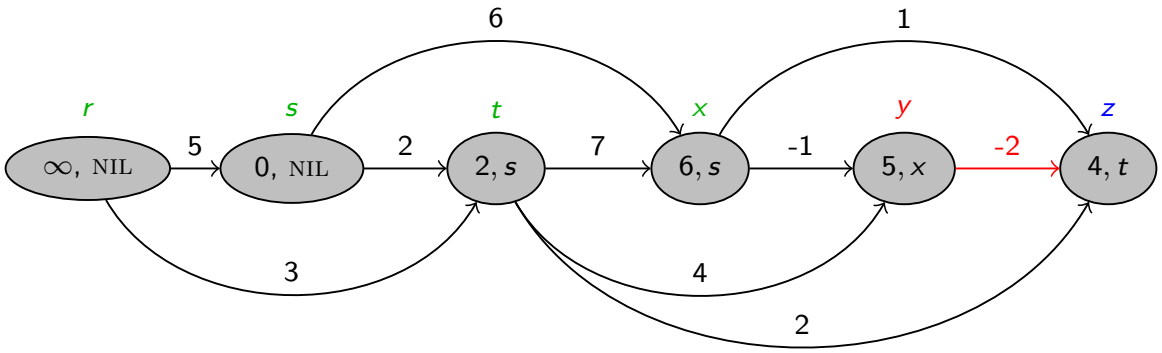
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



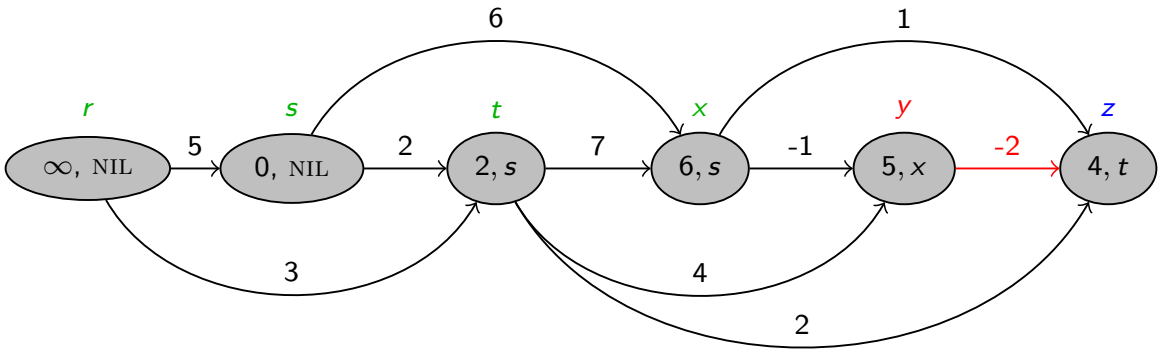
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



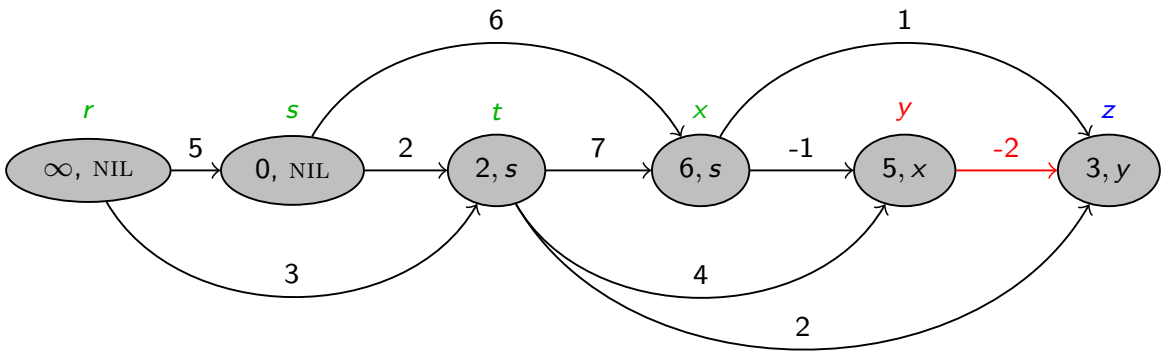
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



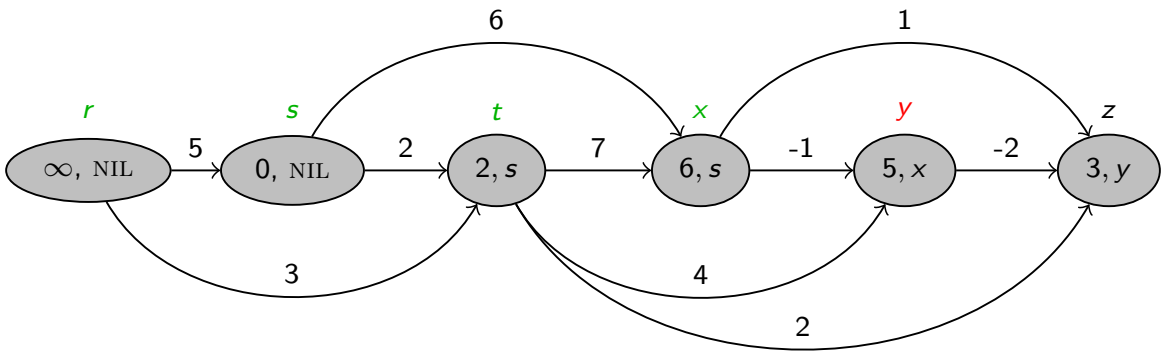
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



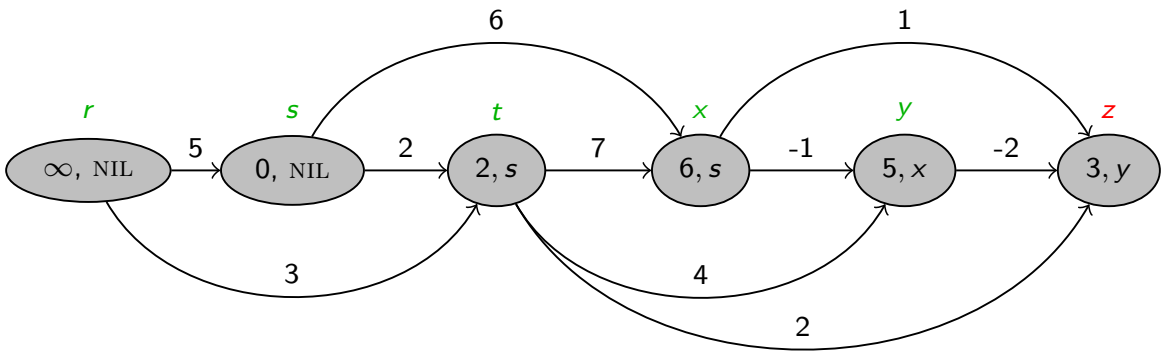
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



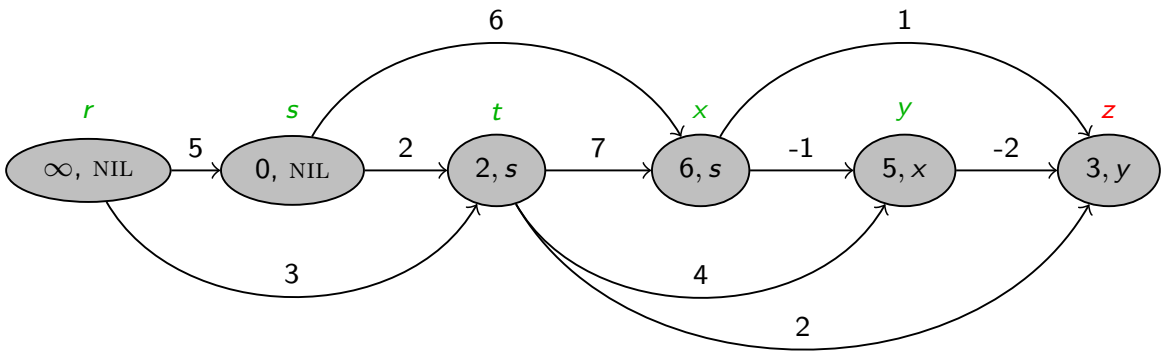
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



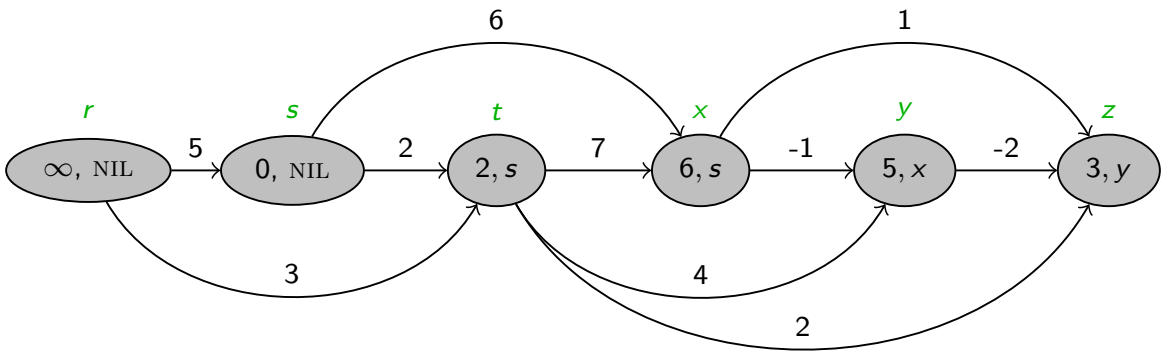
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



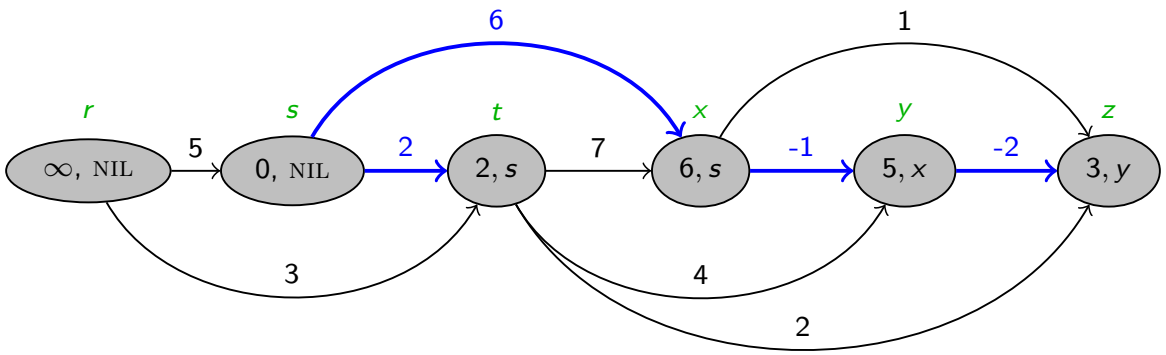
1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

DAG-Shortest-Paths(G, w, s)



1. Topological-Sort(G)
2. **for** each vertex $v \in V$
3. $\quad d[v] \leftarrow \infty$
4. $\quad \pi[v] \leftarrow$ NIL
5. $d[s] \leftarrow 0$
6. **for** each vertex $u \in V$, taken in topologically sorted order
7. $\quad$ **for** each vertex $v \in Adj[u]$
8. $\quad\quad$ **if** $(d[v] > d[u] + w(u, v))$
9. $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v)$
10. $\quad\quad\quad \pi[v] \leftarrow u$

Complexity

- **Line 1:** $\Theta(|V| + |E|)$.
- **Line 2-5:** $\Theta(|V|)$.
- **Line 7-9:** $|E| \cdot \Theta(1) = \Theta(|E|)$.
- **Total Complexity:** $\Theta(|V| + |E|)$.

Correctness

Theorem

If a weighted, directed graph $G = (V, E)$ has source vertex s and no cycles, then at the termination of the DAG-Shortest-Paths *procedure, $d[v] = \delta(s, v)$ for all vertices $v \in V$, and the predecessor sub-graph G_π is a shortest-paths tree.*

Path-relaxation Property

Lemma (Path-relaxation Property)

If $p = \langle v_0, v_1, \ldots, v_k \rangle$ is a shortest path from $s = v_0$ to v_k, and the edges of p are relaxed in the order $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, then $d[v_k] = \delta(s, v_k)$.

Predecessor-subgraph Property

Lemma (Predecessor-subgraph Property)

Once $d[v] = \delta(s, v)$ for all $v \in V$, the predecessor subgraph is a shortest-paths tree rooted at s.

Proof of the Theorem

We first show that $d[v] = \delta(s, v)\ \forall\ v \in V$ at termination.

- If v is not reachable from s, then $d[v] = \delta(s, v) = \infty$.
- Suppose that v is reachable from s, so that there is a shortest path $p = \langle v_0, v_1, \ldots, v_k \rangle$, where $v_0 = s$ and $v_k = v$.
- Because we process the vertices in topologically sorted order, the edges on p are relaxed (lines 8-10) in the order $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$.
- The path-relaxation property implies that $d[v_i] = \delta(s, v_i)$ at termination for $i = 0, 1, \ldots, k$.

Finally, by the predecessor-subgraph property, G_π is a shortest-paths tree.

Strongly Connected Components (SCC)

Introduction

- It is a classic application of DFS.
- Decomposes a directed graph into its SCCs.
- Can be done using two DFSs.
- Many algorithms that work with directed graphs begin with such a decomposition.
- After decomposition, the algorithm is run separately on each SCC.
- The solutions are then combined according to the structure of SCCs.

Transpose of a Directed Graph

Definition (Transpose of a Graph)

The transpose of a directed graph $G = (V, E)$, is the graph $G^T = (V, E^T)$, where $E^T = \{(u, v) : (v, u) \in E\}$.

Time Complexity: Given an adjacency-list representation of G, the time to create G^T is $\mathcal{O}(V + E)$.

Observations

- G and G^T has exactly the same SCCs.

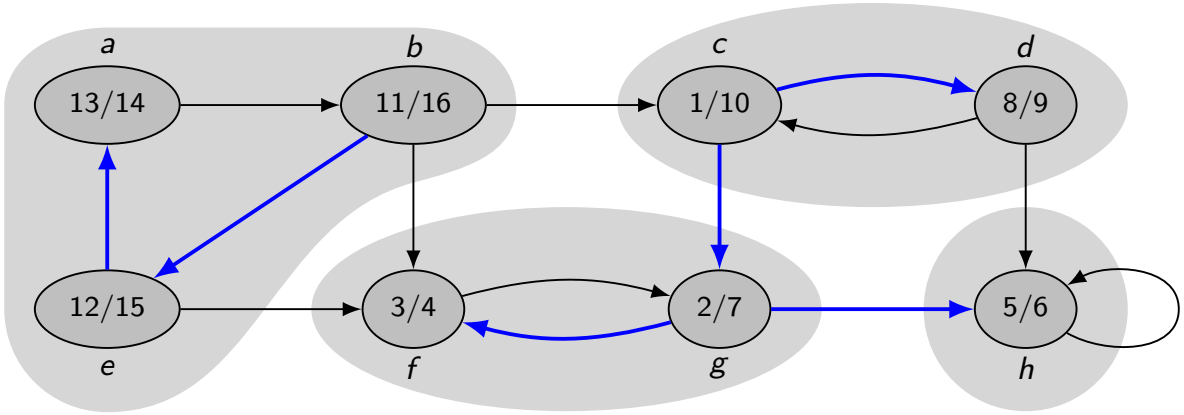
- That is, u and v are reachable from each other in G if and only if they are reachable from each other in G^T.

Stongly-Connected-Component(G)

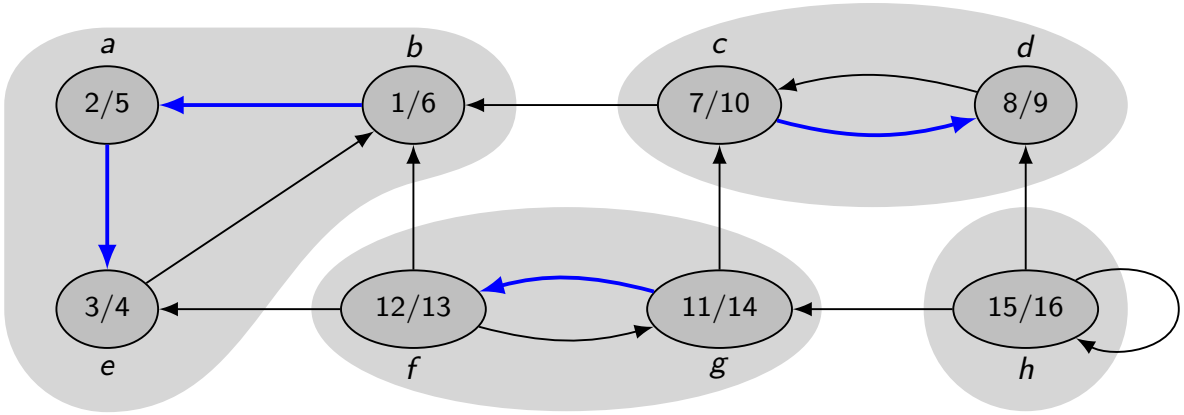
1. Call $\mathrm{DFS}(G)$ to compute finishing times $f[u]$ for each vertex u
2. Compute G^T
3. Call $\mathrm{DFS}(G^T)$, but in the main loop of DFS, consider the vertices in order of decreasing $f[u]$ (as computed in line 1)
4. Output the vertices of each tree in the depth-first forest formed in line 3 as a separate SCC

Time Complexity: $\Theta(|V| + |E|)$

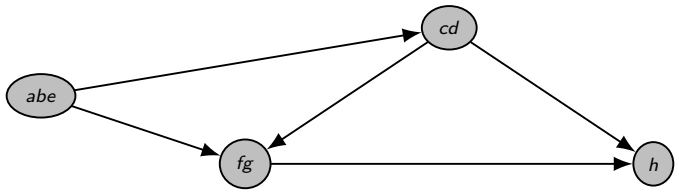
SCCs of G



SCCs of G^T



Acyclic Component Graph



The **acyclic component graph** G^{SCC} obtained by contracting all edges within each SCC of G so that only a single vertex.

Key Property: The component graph is a DAG.

Lemma

Let and $'$ be distinct SCCs in directed graph $G = (V, E)$, let $u, v \in C$, let $u', v' \in C'$, and suppose that there is a path $u \rightsquigarrow u'$ in G. Then there cannot also be a path $v' \rightsquigarrow v$ in G.

Correctness

For $U \subseteq V$, define

$$d(U) = \min_{u \in U}\{d[u]\} \quad \text{and} \quad f(U) = \max_{u \in U}\{f[u]\},$$

i.e., $d(U)$ and $f(U)$ are the earliest discovery time and latest finishing time, respectively, of any vertex in U.

Correctness

Lemma

Let C and C' be distinct SCCs in directed graph $G = (V, E)$. Suppose that there is an edge $(u, v) \in E$, where $u \in C$ and $v \in C'$. Then $f(C) > f(C')$.

Correctness

Lemma

Let C and C' be distinct SCCs in directed graph $G = (V, E)$. Suppose that there is an edge $(u, v) \in E$, where $u \in C$ and $v \in C'$. Then $f(C) > f(C')$.

Corollary

Let C and C' be distinct SCCs in directed graph $G = (V, E)$. Suppose that there is an edge $(u, v) \in E^T$, where $u \in C$ and $v \in C'$. Then $f(C) < f(C')$.

Books and Other Materials Consulted

1. *Introduction to Algorithms* by Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein.

Thank You for your kind attention!

Questions!!