Topological Sort

Adapted from the CLRS book slides

TOPOLOGICAL SORT

Directed acyclic graph (dag)

A directed graph with no cycles.

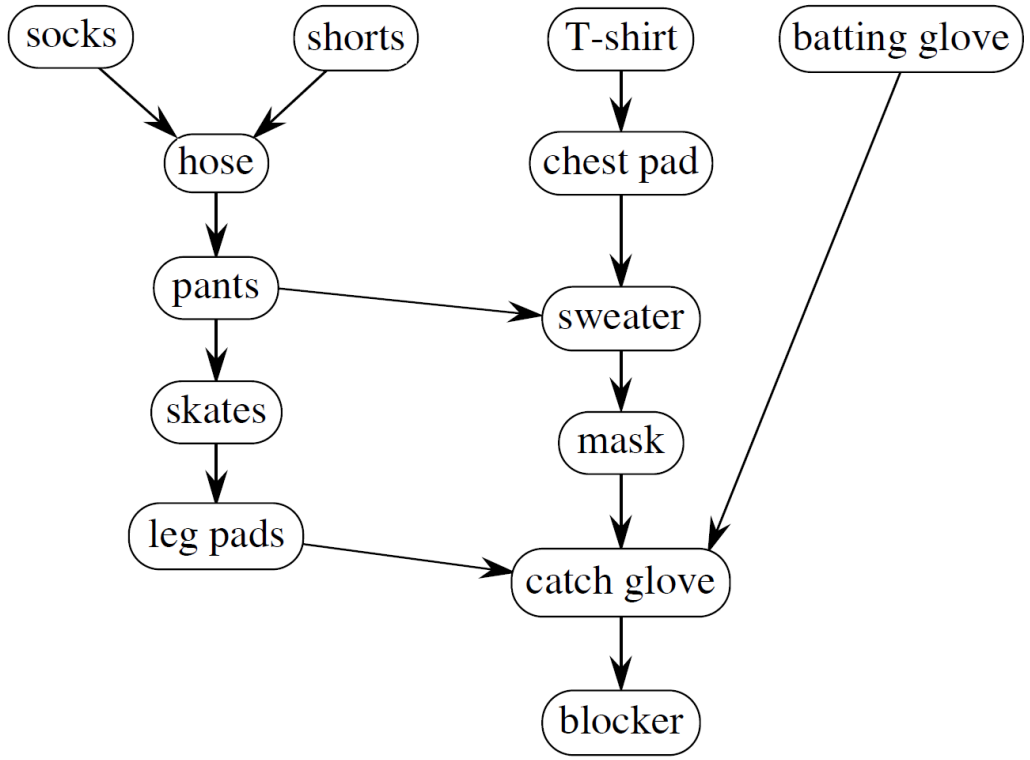
Good for modeling processes and structures that have a ***partial order:***

- $a > b$ and $b > c \Rightarrow a > c$.
- But may have a and b such that neither $a > b$ nor $b > a$

Can always make a ***total order*** (either $a > b$ or $b > a$ for all $a \neq b$) from a partial order. In fact, that's what a topological sort will do.

EXAMPLE

Dag of dependencies for putting on goalie equipment for ice hockey:

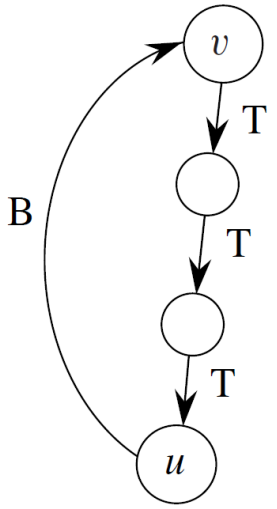


EXAMPLE (continued)

A directed graph G is acyclic if and only if a DFS of G yields no back edges.

Proof $\Rightarrow$: Show that back edge $\Rightarrow$ cycle.

Suppose there is a back edge (u, v). Then v is ancestor of u in depth-first forest.



Therefore, there is a path $v \rightsquigarrow u$, so $v \rightsquigarrow u \rightarrow v$ is a cycle.

EXAMPLE (continued)

Topological sort of a dag: a linear ordering of vertices such that if $(u, v) \in E$, then u appears somewhere before v. (Not like sorting numbers.)

TOPOLOGICAL-SORT(G)

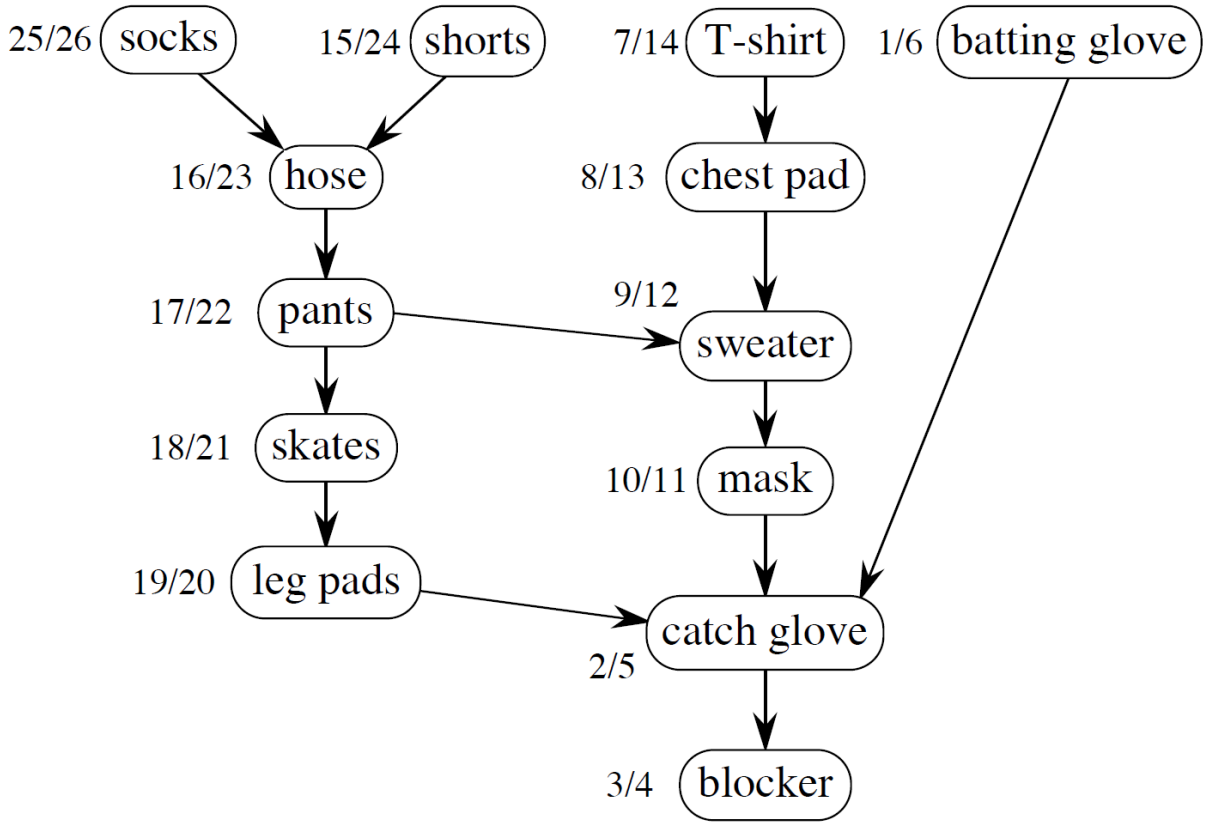
1 call DFS(G) to compute finish times $v.f$ for each vertex v
2 as each vertex is finished, insert it onto the front of a linked list
3 **return** the linked list of vertices

- Can just output vertices as they're finished and understand that we want the *reverse* of this list.
- Or put them onto the *front* of a linked list as they're finished. When done, the list contains vertices in topologically sorted order.

Time $\Theta(V + E)$.

EXAMPLE (continued)

Let's run DFS on this example to find finish times.



EXAMPLE (continued)

Correctness

Just need to show if $(u, v) \in E$, then $v.f < u.f$.
When edge (u, v) is explored, what are the colors of u and v?

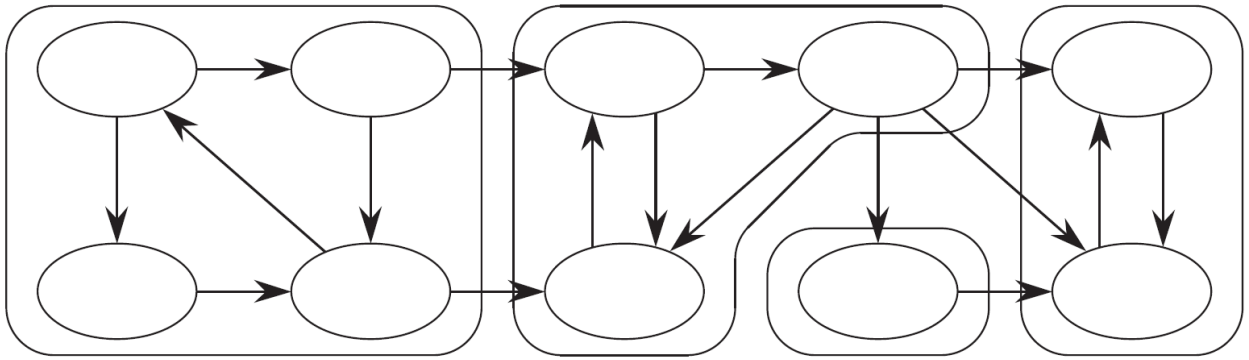
- u is gray.
- Is v gray, too?
  - *No*, because then v would be ancestor of u.
    $\Rightarrow (u, v)$ is a back edge.
    $\Rightarrow$ contradiction of previous lemma (dag has no back edges).
- Is v white?
  - Then becomes descendant of u.
    By parenthesis theorem, $u.d < v.d < v.f < u.f$.
- Is v black?
  - Then v is already finished.
    Since exploring (u, v), u is not yet finished.
    Therefore, $v.f < u.f$.

STRONGLY CONNECTED COMPONENTS

Given directed graph $G = (V, E)$.

A ***strongly connected component*** (***SCC***) of G is a maximal set of vertices $C \subseteq V$ such that for all $u, v \in C$, both $u \rightsquigarrow v$ and $v \rightsquigarrow u$.

Example



EXAMPLE (continued)

Algorithm uses $G^{\mathrm{T}} =$ ***transpose*** of G.

- $G^{\mathrm{T}} = (V, E^{\mathrm{T}}), E^{\mathrm{T}} = \{(u, v) : (v, u) \in E\}$.
- G^{T} is G with all edges reversed.

Can create G^{T} in $\Theta(V + E)$ time if using adjacency lists.

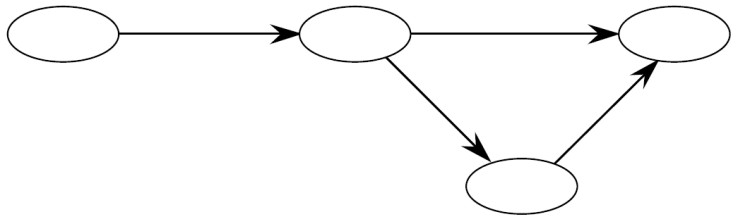
Observation

G and G^{T} have the *same* SCC's. (u and v are reachable from each other in G if and only if reachable from each other in G^{T}.)

EXAMPLE (continued)

- $G^{SCC} = (V^{SCC}, E^{SCC})$.
- V^{SCC} has one vertex for each SCC in G.
- E^{SCC} has an edge if there's an edge between the corresponding SCC's in G.

For our example:

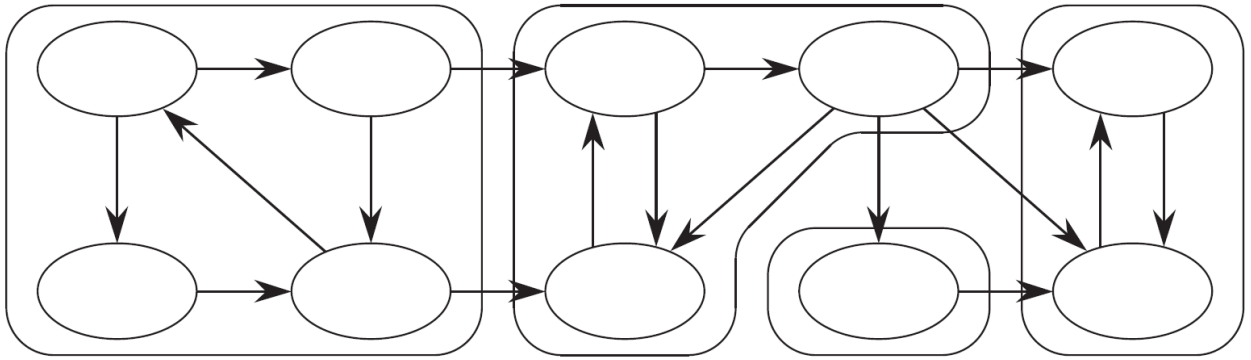


Lemma

G^{SCC} is a dag. More formally, let C and C' be distinct SCC's in G, let $u, v \in C$, $u', v' \in C'$, and suppose there is a path $u \rightsquigarrow u'$ in G. Then there cannot also be a path $v' \rightsquigarrow v$ in G.

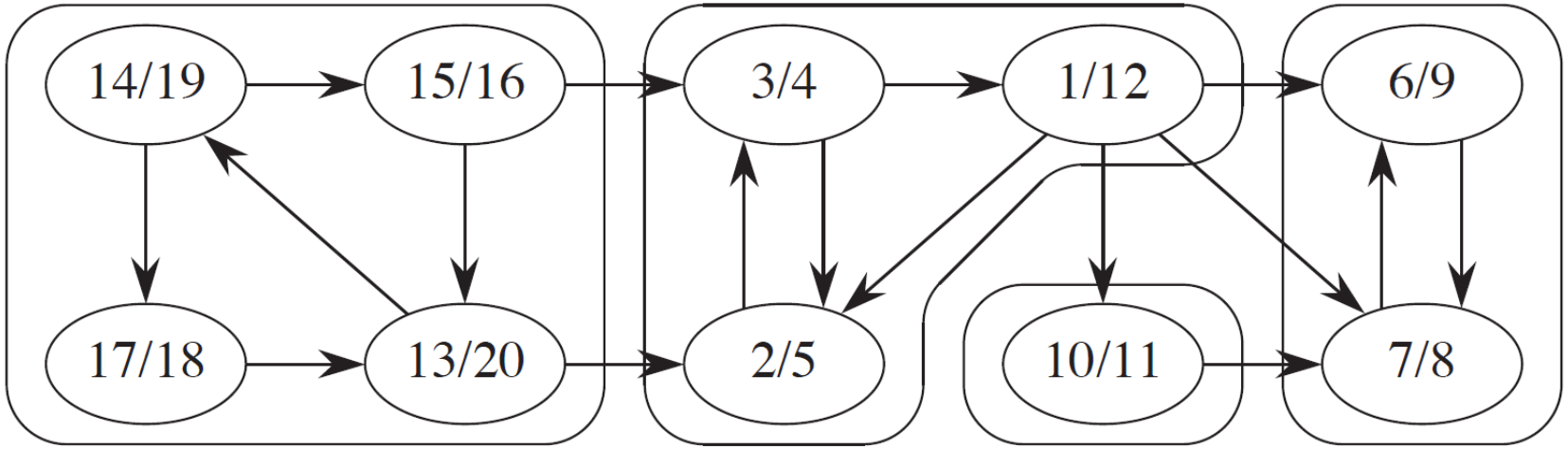
EXAMPLE (continued)

SCC(G)
  call DFS(G) to compute finish times $u.f$ for each vertex u
  create G^T
  call DFS(G^T), but in the main loop, consider vertices in order of decreasing $u.f$
      (as computed in first DFS)
  output the vertices in each tree of the depth-first forest formed in second DFS
      as a separate SCC



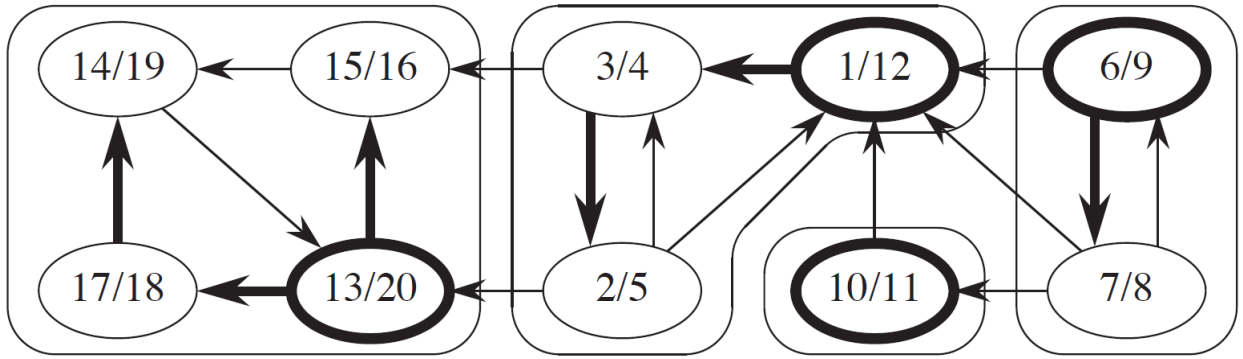
EXAMPLE (continued)

Run DFS.



EXAMPLE (continued)

DFS in G^T.



Time: $\Theta(V + E)$.

Each top level DFS-Visit will discover it's component.

Correctness Argument

How can this possibly work?

Idea

By considering vertices in second DFS in decreasing order of finish times from first DFS, visiting vertices of the component graph in topological sort order.

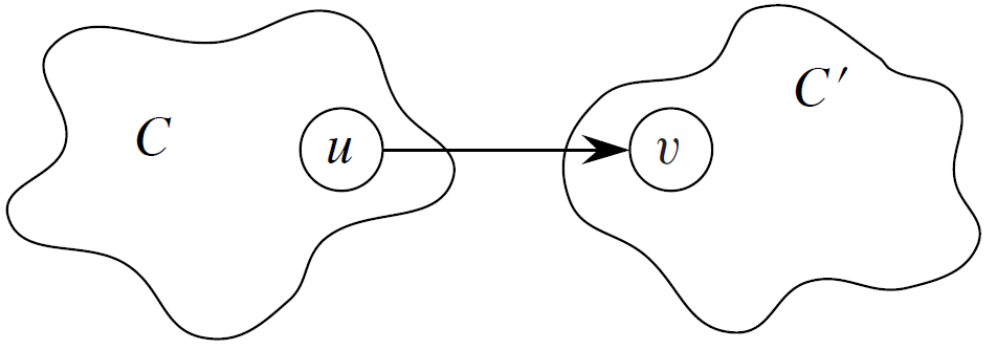
To prove that it works, first deal with 2 notational issues:

- Will be discussing $u.d$ and $u.f$. These always refer to *first* DFS.
- Extend notation for d and f to sets of vertices $U \subseteq V$:
  - $d(U) = \min\{u.d : u \in U\}$ (earliest discovery time in U)
  - $f(U) = \max\{u.f : u \in U\}$ (latest finish time in U)

Correctness Argument (continued)

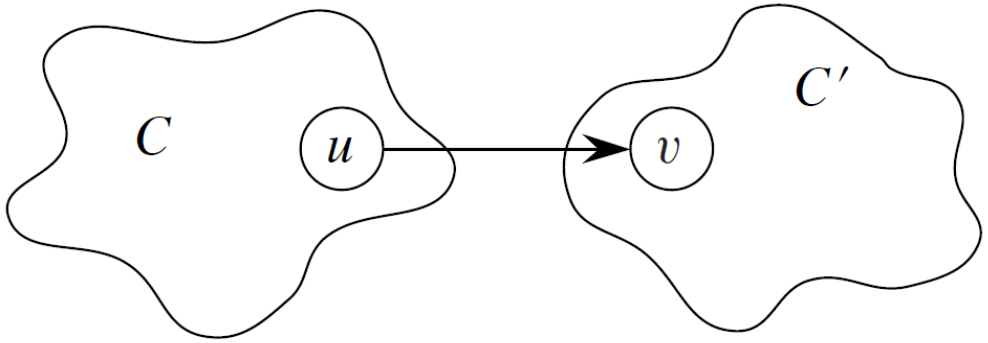
Lemma

Let C and C' be distinct SCC's in $G = (V, E)$. Suppose that there is an edge $(u, v) \in E$ such that $u \in C$ and $v \in C'$.



Then $f(C) > f(C')$.

Correctness Argument (continued)



Each root chosen for the second DFS can reach only

- vertices in its SCC—get tree edges to these,
- vertices in SCC's *already visited* in second DFS—get *no* tree edges to these.

Visiting vertices of $(G^{\mathrm{T}})^{\mathrm{SCC}}$ in reverse of topologically sorted order.