Chapter 22 Elementary Graph Algorithms Part II

Chi-Yeh Chen

陳奇業

成功大學資訊工程學系

藏行顯光
成就共好

Achieve Securely
Prosper Mutually

國立成功大學 九十週年
90th Anniversary of NCKU

Topological sort

NCKU
National Cheng Kung University
1931

藏行顯光
成就共好
Achieve Securely
Prosper Mutually
國立成功大學 九十週年
90th Anniversary of NCKU

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.

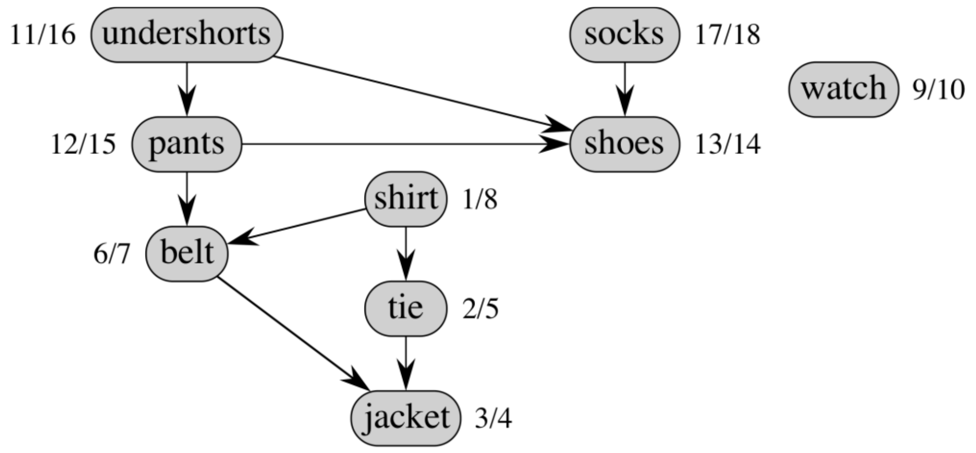
In fact, that's what a topological sort will do.

Topological sort

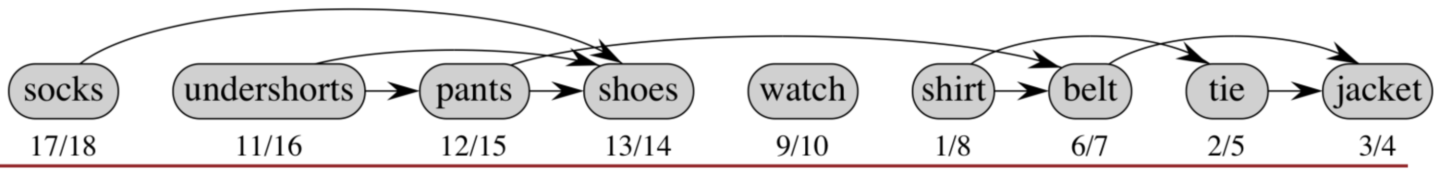
▷ Define:

A topological sort of a 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

(a)



(b)



```
TOPOLOGICAL-SORT($G$)
1  call DFS($G$) to compute finishing time $v.f$ for each vertex $v$
2  as each vertex is finished, insert it onto the front of linked list
3  **return** the linked list of vertices
```

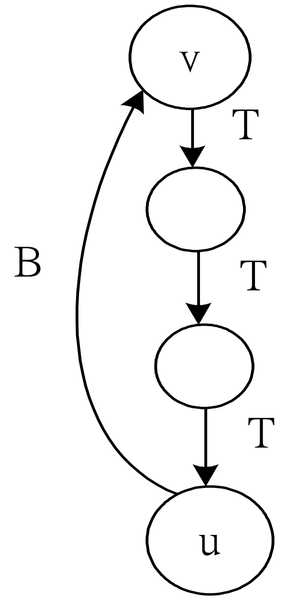
Don’t need to sort by finish times.

- 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 topological sorted order.

Time: $\Theta(V + E)$

Lemma 22.11

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



Proof

⇒: Show that back edge ⇒ cycle.

Suppose there is a back edge (u, v).

Then v is ancestor of u in depth-first forest.
Therefore, there is a path $v \rightarrow u$, so $v \rightarrow u \rightarrow v$ is a cycle.

⇐: Show that cycle ⇒ back edge.

Suppose G contains cycle c. At time $v.d$, vertices of c form a white path $v \longrightarrow u$ (since v is the first vertex discovered in c).

By white-path theorem, u is descendant of v in depth-first forest.

Therefore, (u, v) is a back edge.

Theorem 22.12

TOPOLOGICAL-SORT produces a topological sort of the directed acyclic graph provided as its input.

Proof

Correctness: Just need show if $(u, v) \in E$, then $v.f < u.f$.

When we explore (u, v), 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 22.11 (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 we're exploring (u, v), we have not yet finished u.

    Therefore, $v.f < u.f$.

藏行顯光
成就共好
Achieve Securely
Prosper Mutually

Strongly connected components

NCKU
National Cheng Kung University
1931

藏行顯光
成就共好
Achieve Securely
Prosper Mutually
國立成功大學 九十週年
90th Anniversary of NCKU

Strongly connected components

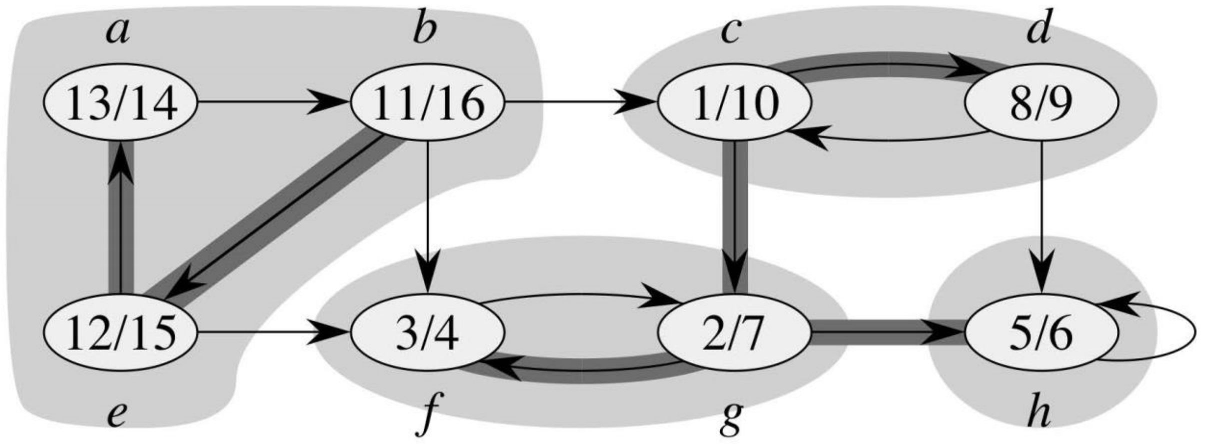
Given directed graph $G = (V, E)$.

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 $u \to v$ and $v \to u$.

Example: [Just show SCC's at first. Do DFS a little later.]

(a)



Algorithm uses $G^{\mathrm{T}} = \textbf{transpose}$ 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 $\Theta(V + \mathrm{E})$ time if using adjacency lists.

Observation: G and G^{T} have the same SCC's.

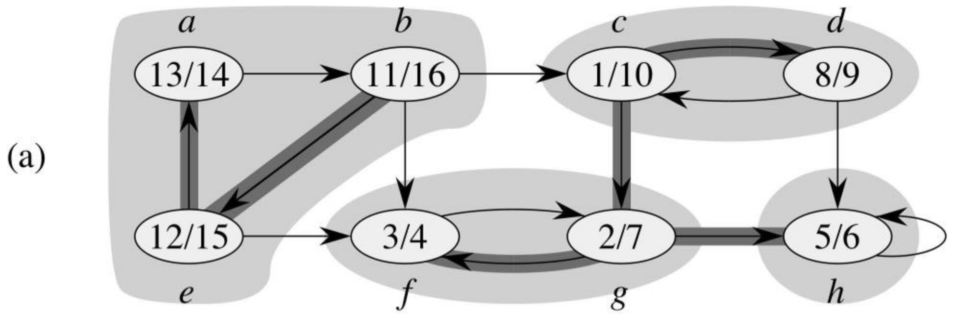
(u and v are reachable from each other in *G* if and only

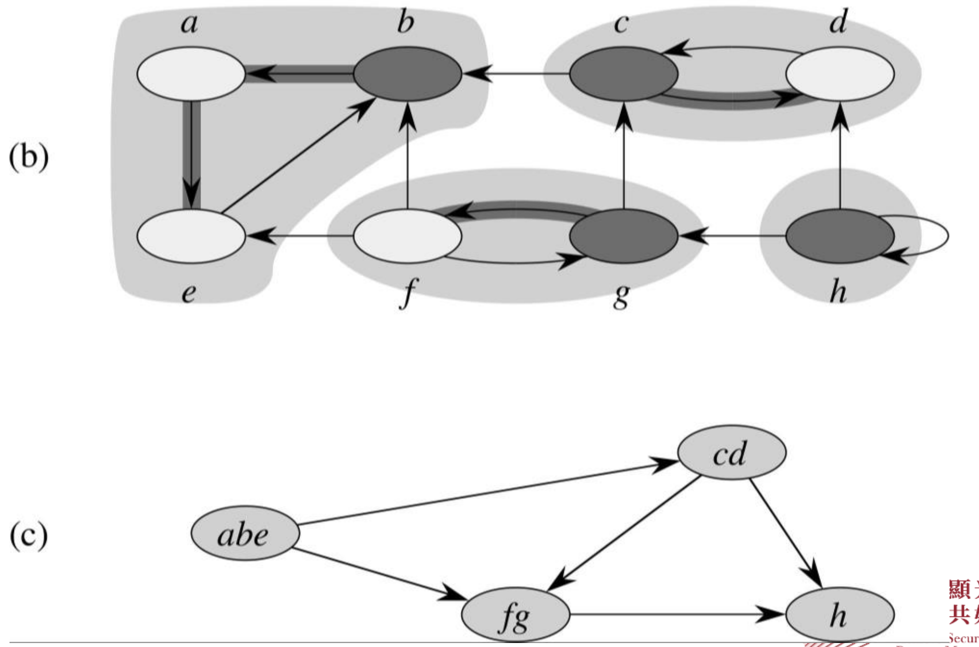
if reachable from each other in G^{T}.)

Component graph

- $G^{\text{SCC}} = (V^{\text{SCC}}, E^{\text{SCC}})$.
- 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:

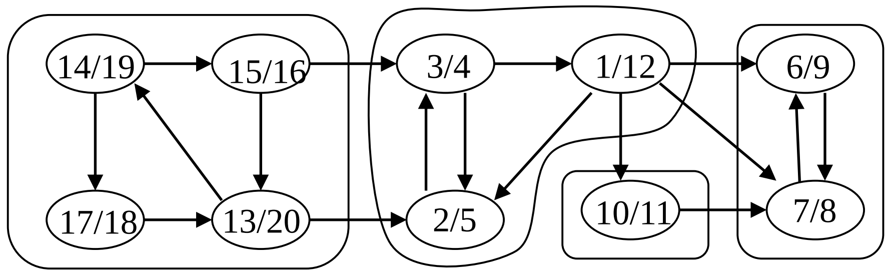


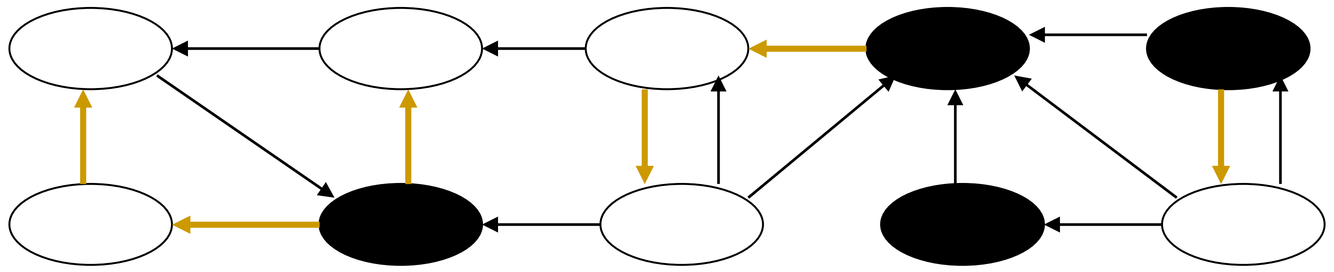


```
STRONGLY-CONNECTED-COMPONENTS($G$)
1  call DFS($G$) to compute finishing time $u.f$ for each vertex $u$
2  compute $G^T$
3  call DFS($G^T$), but in the main loop of DFS, consider the vertices in
     order of decreasing $u.f$ (as computed in line 1)
4  output the vertices of each tree in the depth-first forest formed in
     line 3 as a separate strongly connected component.
```

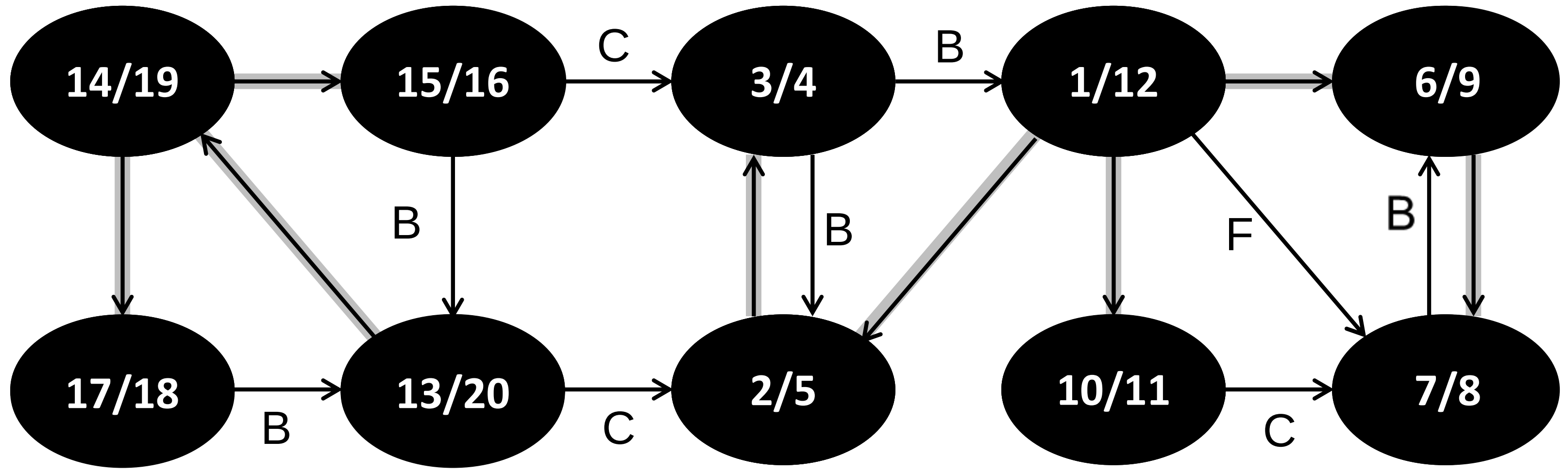
Example:

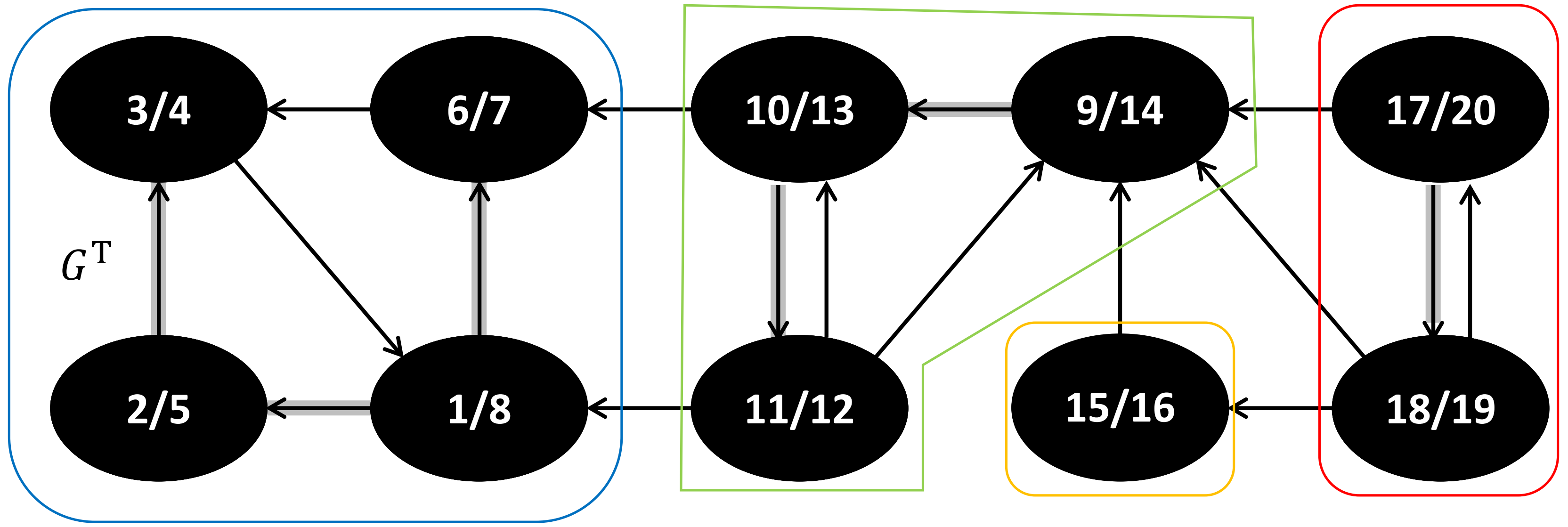
1. Do DFS
2. G^{T}
3. DFS (roots blackened)





Time: $\Theta(V+E)$


14/19
15/16
3/4
1/12
6/9
17/18
13/20
2/5
10/11
7/8
C
B
B
B
B
F
B
B
C
C


3/4
6/7
10/13
9/14
17/20
G^{T}
2/5
1/8
11/12
15/16
18/19

How can this possibly work?

Idea:
By considering vertices in second DFS in decreasing order of finishing times from first DFS, we are 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 $U \subseteq V$:
    - $d(\mathrm{U}) = \min_{u \in \mathrm{U}}\{u.d\}$ (earliest discovery time)
    - $f(\mathrm{U}) = \max_{u \in \mathrm{U}}\{u.f\}$ (latest finishing time)

Lemma 22.13

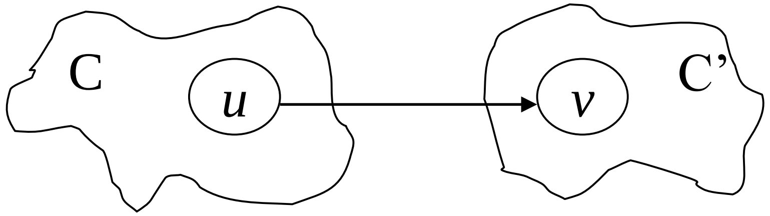
Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$, let $u, v \in C$, let $u', v' \in C'$, and suppose that G contains a path $u \leadsto u'$. Then G cannot also contain a path $v' \leadsto v$.

Proof

If G contains a path $v' \leadsto v$, then it contains paths $u \leadsto u' \leadsto v'$ and $v' \leadsto v \leadsto u$. Thus, u and v' are reachable from each other, thereby contradicting the assumption that C and C' are distinct strongly connected components.

Lemma 22.14

Let C and C' be distinct strongly connected components 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 $f(C) > f(C')$.



Proof

Two cases, depending on which SCC had the first discovered vertex during first DFS.

1. If $d(C) < d(C')$, let *x* be the first vertex discovered in C.
At time $x.d$, all vertices in C and C' are white. Because $(u, v) \in E$, there exist paths of white vertices from *x* to all vertices in C and C'.

By the white-path theorem, all vertices in C and C' are descendants of x in depth-first tree.

By the parenthesis theorem, $x.f = f(C) > f(C')$.

2. If $d(C) > d(C')$, let y be the first vertex discovered in C'.
At time $y.d$, all vertices in C' are white and there is a white path from y to each vertex in C' $\Rightarrow$ all vertices in C' become descendants of y.

Again, $y.f = f(C')$.

At time $y.d$, all vertices in C are white.

By lemma 22.13, since there is an edge (u, v), we cannot have a path from C' to C.

So no vertex in C is reachable from y.

Therefore, at time $y.f$, all vertices in C are still white.

Therefore, for all $w \in C$, $w.f > y.f$, which implies that $f(C) > f(C')$.

Corollary 22.15

Let C and C' be distinct strongly connected components 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 $f(C) < f(C')$.

Proof

Since $(u, v) \in E^T$, we have $(v, u) \in E$. Because the strongly connected components of G and G^T are the same, Lemma 22.14 implies that $f(C) < f(C')$.

Corollary 22.15.1

Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$. Suppose that $f(C) > f(C')$. Then there cannot be an edge from C to C' in G^T.

Why the SCC procedure works:

When we do the second DFS, on G^{T}, start with SCC C such that $f(C)$ is maximum.

- The second DFS starts from some $x \in C$, and it visits all vertices in C.
- Corollary says that since $f(C) > f(C')$ for all $C' \neq C$, there are no edges from C to C' in G^{T}.

Therefore, DFS will visit *only* vertices in C.

- The next root chosen in the second DFS is in SCC C' such that $f(C')$ is maximum over all SCC’s other than C.
  DFS visits all vertices in C', but the only edges out of C' go to C, *which we’ve already visited.*

- Each time we choose a root for the second DFS, it 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.

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

Theorem 22.12

The STRONGLY-CONNECTED-COMPONENTS produces correctly computes the strongly connected components of the directed graph G provided as its input.

Proof

We argue by induction on the number of depth-first trees found in the depth-first search of G^T that are vertices of each tree form a strongly connected component.

The inductive hypothesis is that the first k trees are strongly connected components.

The basis for the induction, when $k = 0$, is trivial.

In the inductive step, we assume that each of the first k depth-first trees is a strongly connected component, and we consider the $(k+1)$st tree produced.

Let the root of this tree be vertex u, and let u be in strongly connected component C.

Because of how we choose roots, $u.f = f(C) > f(C')$ for any strongly connected component C' other than C that has yet to be visited. (Corollary 22.15.1)

By the inductive hypothesis, at the time that the search visits u, all other vertices of C are white. By the white-path theorem, therefore, all other vertices of C are descendants of u in its depth-first tree.

Moreover, by the inductive hypothesis and by Corollary 22.15, any edges in G^T that leave C must be to strongly connected components that have already been visited.

Thus, no vertex in any strongly connected component other than C will be a descendant of u during the depth-first search of G^T.

Thus, the vertices of the depth-first tree in G^T that is rooted at u form exactly one strongly connected component, which completes the inductive step and the proof.

藏行顯光
成就共好

Achieve Securely
Prosper Mutually

國立成功大學 九十週年
90th Anniversary of NCKU

1931
國立成功大學
National Cheng Kung University