## Elementary Graph Algorithms

Pierre Flener
(version of 2011-02-02)

## Undirected Graph

## Information Technology

Example: A city map for pedestrians
(edges are streets, nodes are intersections)


## Directed Graph (Digraph)

## Information Technology

Example: A city map for cars
(edges are one-way streets, nodes are intersections)


## Directed \& Undirected Graphs

■ Node, vertex (plural: vertices)

- Edge
- Self-loop (edge from a node to itself)
- Adjacent nodes (connected by an edge)
- Degree (\#edges from or to a node)
- In-degree (\#edges to a node)
- Out-degree (\#edges from a node)
- Path (sequence of adjacent nodes)


## Representations of Graphs

- Adjacency-matrix representation: a 2-dimensional array $A$ of $0 / 1$ values, with $A[i, j]$ containing the number of arcs between nodes $i$ and $j$ (undirected graph), or from node $i$ to node $j$ (digraph)
- Adjacency-list representation: a 1-dimensional array Adj of adjacency lists with $\operatorname{Adj}[i]$ containing the list of nodes adjacent to node $i$




## Graph Traversals

- by breadth-first search (BFS)

■ by depth-first search (DFS)

BFS and DFS work on directed graphs as well as on undirected graphs (the examples below are for digraphs). BFS and DFS assume the given graph is represented using adjacency lists.


| BFS order (black) | Queue (gray) |
| :--- | :--- |
|  | A |

Given a source node, A. Paint the source gray. Paint other nodes white. Add the source to the initially empty first-in first-out (FIFO) queue of gray nodes.

| A: | B | F |  |
| :--- | :--- | :--- | :--- |
| B: | C |  |  |
| C: | G | J |  |
| D: | C | E | J |
| E: | N |  |  |
| F: | H |  |  |
| G: | A | K |  |
| J: | I | L | M |
| I: | E |  |  |
| H: | A | K |  |
| K: | G | H | J |
| L: |  |  |  |
| M: | N |  |  |
| N: | M |  |  |



| BFS order (black) | Queue (gray) |
| :--- | :--- |
| A | BF |

Dequeue head node, A. Paint its undiscovered (=white) adjacent nodes gray and enqueue them. Paint it black (all adjacent nodes are discovered).











$1 \mid 1\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \| \|$

$1 \mid 1\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \|\| \| \|$



|  | BFS order (black) | Queue (gray) |
| :---: | :---: | :---: |
|  | A | BF |
|  | AB | FC |
|  | ABF | CH |
| ì | ABFC | HGJ |
|  | ABFCH | GJK |
|  | ABFCHG | JK |
|  | ABFCHGJ | KILM |
|  | ABFCHGJK | ILM |
|  | ABFCHGJKI | LME |
|  | ABFCHGJKIL | ME |
|  | ABFCHGJKILM | EN |
|  | ABFCHGJKILME | N |
|  | ABFCHGJKILMEN |  |


| A: | B | F |  |
| :--- | :--- | :--- | :--- |
| B: | C |  |  |
| C: | G | J |  |
| D: | C | E | J |
| E: | N |  |  |
| F: | H |  |  |
| G: | A | K |  |
| J: | I | L | M |
| I: | E |  |  |
| H: | A | K |  |
| K: | G | H | J |
| L: |  | May restart from |  |
| m: | N | May |  |
| N: | M | White source |  |



# Analysis of BFS for graph (V,E 

 (without any restarts)- The initialisations take $\Theta(\mathrm{V})$ time.
- Each node is enqueued (in $\mathrm{O}(1)$ time) and dequeued (in $\mathrm{O}(1)$ time) at most once (because no node is ever repainted white), hence $O(V)$ time for all queue operations.
- Each adjacency list is scanned at most once, and their total length is $\Theta(\mathrm{E})$, hence $O(E)$ time for scanning adjacency lists. - Aggregate analysis: $\mathrm{O}(\mathrm{V}+\mathrm{E})$ time.

Refinements of BFS (at no increase in time complexity)

- Store the predecessor (or parent) of each newly discovered node (initially NIL, as undiscovered = white): breadth-first tree, rooted at source $s$.
- Store the distance (in \#edges) from the source $s$ to each newly discovered node (by adding 1 to the distance of its parent, with $s$ being at distance 0 ): lengths of shortest paths from $s$ to all reachable nodes

Paint all nodes white.

| DFS finish order (black) | Stack (gray) | Rest (white) |
| :--- | :--- | :--- |
|  |  | ABCDEFGHIJKLMN |




# Continue to A's first undiscovered child, B. 

| DFS finish order (black) | Stack (gray) | Rest (white) |  |
| :--- | :--- | :--- | :--- |
|  |  | A | BCDEFGHIJKLMN |
|  | BA | CDEFGHIJKLMN |  |



## Continue to B's first undiscovered child, C.

| DFS finish order (black) | Stack (gray) | Rest (white) |
| :--- | :--- | :--- |
|  |  | A |
|  | BCDEFGHIJKLMN |  |
|  |  | BA |
|  |  | CDEFGHIJKLMN |
|  |  | CBA |
|  | DEFGHIJKLMN |  |

## Information Technology



## Continue to C's first undiscovered child, G.

| DFS finish order (black) | Stack (gray) | Rest (white) |
| :--- | :---: | :--- |
|  |  | A BCDEFGHIJKLMN |
|  |  | BA CDEFGHIJKLMN |
|  |  | CBA |
|  | GEFGHIJKLMN |  |
|  |  | GCBA |
|  |  |  |



## Continue to G's first undiscovered child, K.




## Continue to K's first undiscovered child, H.

| DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: |
|  |  | A | BCDEFGHIJKLMN |
|  |  | BA | CDEFGHIJKLMN |
|  |  | CBA | DEFGHIJKLMN |
|  |  | GCBA | DEFHIJKLMN |
|  |  | KGCBA | DEFHIJLMN |
|  |  | HKGCBA | DEFIJLMN |



## H has no undiscovered child, backtrack until we find node $K$, with undiscovered child J.

| DFS finish order (black) | Stack (gray) | Rest (white) |
| :---: | :---: | :---: |
|  | A | BCDEFGHIJKLMN |
|  | BA | CDEFGHIJKLMN |
|  | CBA | DEFGHIJKLMN |
|  | GCBA | DEFHIJKLMN |
|  | KGCBA | DEFHIJLMN |
|  | HKGCBA | DEFIJLMN |
| H | JKGCBA | DEFILMN |



## Continue to J's first undiscovered child, I.

| DFS finish order (black) | Stack (gray) | Rest (white) |
| :---: | :---: | :---: |
|  | A | BCDEFGHIJKLMN |
|  | BA | CDEFGHIJKLMN |
|  | CBA | DEFGHIJKLMN |
|  | GCBA | DEFHIJKLMN |
|  | KGCBA | DEFHIJLMN |
|  | HKGCBA | DEFIJLMN |
| H | JKGCBA | DEFILMN |
| H | IJKGCBA | DEFLMN |



## Continue to I's first undiscovered child, E.

| DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: |
|  |  | A | BCDEFGHIJKLMN |
|  |  | BA | CDEFGHIJKLMN |
|  |  | CBA | DEFGHIJKLMN |
|  |  | GCBA | DEFHIJKLMN |
|  |  | KGCBA | DEFHIJLMN |
|  |  | HKGCBA | DEFIJLMN |
| H |  | JKGCBA | DEFILMN |
| H |  | IJKGCBA | DEFLMN |
| H |  | EIJKGCBA | DFLMN |



## Continue to E's first undiscovered child, N.

| DFS finish order (black) | Stack (gray) | Rest (white) |
| :---: | :---: | :---: |
|  | A | BCDEFGHIJKLMN |
|  | BA | CDEFGHIJKLMN |
|  | CBA | DEFGHIJKLMN |
|  | GCBA | DEFHIJKLMN |
|  | KGCBA | DEFHIJLMN |
|  | HKGCBA | DEFIJLMN |
| H | JKGCBA | DEFILMN |
| H | IJKGCBA | DEFLMN |
| H | EIJKGCBA | DFLMN |
| H | NEIJKGCBA | DFLM |



## Continue to N's first undiscovered child, M.

| DFS finish order (black) | Stack (gray) | Rest (white) |
| :---: | :---: | :---: |
|  |  | BCDEFGHIJKLMN |
|  | BA | CDEFGHIJKLMN |
|  | CBA | DEFGHIJKLMN |
|  | GCBA | DEFHIJKLMN |
|  | KGCBA | DEFHIJLMN |
|  | HKGCBA | DEFIJLMN |
| H | JKGCBA | DEFILMN |
| H | IJKGCBA | DEFLMN |
| H | EIJKGCBA | DFLMN |
| H | NEIJKGCBA | DFLM |
| H | MNEIJKGCBA | DFL |



## M has no undiscovered child, backtrack until we find node $J$, with undiscovered child $L$.

| DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: |
|  |  | A | BCDEFGHIJKLMN |
|  |  | BA | CDEFGHIJKLMN |
|  |  | CBA | DEFGHIJKLMN |
|  |  | GCBA | DEFHIJKLMN |
|  |  | KGCBA | DEFHIJLMN |
|  |  | HKGCBA | DEFIJLMN |
| H |  | JKGCBA | DEFILMN |
| H |  | IJKGCBA | DEFLMN |
| H |  | EIJKGCBA | DFLMN |
| H |  | NEIJKGCBA | DFLM |
| H |  | MNEIJKGCBA | DFL |
| HMNEI |  | LJKGCBA | DF |



## L has no undiscovered child, backtrack until we find node $A$, with undiscovered child $F$.

| DFS finish order (black) | Stack (gray) | Rest (white) |
| :---: | :---: | :---: |
|  | A | BCDEFGHIJKLMN |
|  | BA | CDEFGHIJKLMN |
|  | CBA | DEFGHIJKLMN |
|  | GCBA | DEFHIJKLMN |
|  | KGCBA | DEFHIJLMN |
|  | HKGCBA | DEFIJLMN |
| H | JKGCBA | DEFILMN |
| H | IJKGCBA | DEFLMN |
| H | EIJKGCBA | DFLMN |
| H | NEIJKGCBA | DFLM |
| H | MNEIJKGCBA | DFL |
| HMNEI | LJKGCBA | DF |
| HMNEILJKGCB | FA | D |



## F has no undiscovered child;

 the backtrack clears the stack.| DFS finish order (black) | Stack (gray) | Rest (white) |
| :--- | ---: | :--- |
|  | A | BCDEFGHIJKLMN |
|  | BA | CDEFGHIJKLMN |
|  | CBA | DEFGHIJKLMN |
|  | GCBA | DEFHIJKLMN |
|  | KGCBA | DEFHIJLMN |
|  | HKGCBA | DEFIJLMN |
| H | JKGCBA | DEFILMN |
| H | IJKGCBA | DEFLMN |
| H | EIJKGCBA | DFLMN |
| H | NEIJKGCBA | DFLM |
| H | MNEIJKGCBA | DFL |
| HMNEI | LJKGCBA | DF |
| HMNEILJKGCB | FAA | D |
| HMNEILJKGCBFA |  | D |



Restart from next white node, $D$, as source.

| DFS finish order (black) | Stack (gray) | Rest (white) |
| :--- | ---: | :--- |
|  | A | BCDEFGHIJKLMN |
|  | BA | CDEFGHIJKLMN |
|  | CBA | DEFGHIJKLMN |
|  | GCBA | DEFHIJKLMN |
|  | KGCBA | DEFHIJLMN |
|  | HKGCBA | DEFIJLMN |
| H | JKGCBA | DEFILMN |
| H | IJKGCBA | DEFLMN |
| H | EIJKGCBA | DFLMN |
| H | NEIJKGCBA | DFLM |
| H | MNEIJKGCBA | DFL |
| HMNEI | LJKGCBA | DF |
| HMNEILJKGCB | FAA | D |
| HMNEILJKGCBFA | D |  |



## D has no undiscovered child; backtrack; done.



## DFS order (of pushing): ABCGKHJIENMLFD



Refinements of DFS (at no increase in time complexity)

- Store the predecessor (or parent) of each newly discovered node
(initially NIL, as undiscovered = white): depth-first forest of depth-first trees that are rooted at the chosen source nodes.
- Store the timestamps of discovering (graying) and finishing (blackening) each node: useful in many graph algorithms (for example: topological sort, see below).


## Depth-first forest (of depth-first trees)

| DFS finish order (black) | Stack (gray) | Rest (white) |
| :---: | :---: | :---: |
|  | A | BCDEFGHIJKLMN |
|  | BA | CDEFGHIJKLMN |
|  | CBA | DEFGHIJKLMN |
|  | GCBA | DEFHIJKLMN |
|  | KGCBA | DEFHIJLMN |
|  | HKGCBA | DEFIJLMN |
| H | JKGCBA | DEFILMN |
| H | IJKGCBA | DEFLMN |
| H | EIJKGCBA | DFLMN |
| H | NEIJKGCBA | DFLM |
| H | MNEIJKGCBA | DFL |
| HMNEI | LJKGCBA | DF |
| HMNEILJKGCB | FA | D |
| HMNEILJKGCBFA | D |  |
| HMNEILJKGCBFAD |  |  |




Depth-first forest


Another depth-first forest, starting from $D$


Starting from $L$, then $M$, then $J$, then $K$, then $D$


Strongly Connected Components of a Digraph

■ Strongly connected component (SCC): maximal set of nodes where there is a path from each node to each other node.
■ Many algorithms first divide a digraph into its SCCs, then process these SCCs separately, and finally combine the subsolutions. (This is not divide-\&-conquer!)

- An undirected graph can be decomposed into its connected components.


## Strongly Connected Components of a Digraph

■ Strongly connected component (SCC): maximal set of nodes where there is a path from each node to each other node.


## Strongly Connected Components of a Digraph

■ Strongly connected component (SCC): maximal set of nodes where there is a path from each node to each other node.


Computing the Strongly Connected Components of a Digraph G

1. Enumerate the nodes of $G$ in DFS finish order, starting from any node
2. Compute the transpose $\mathrm{G}^{\top}$ (that is, reverse all edges)
3. Make a DFS in $\mathrm{G}^{\top}$, considering nodes in reverse finish order from original DFS
4. Each tree in this depth-first forest is a strongly connected component



| DFS finish order (black) | Stack (gray) | Rest (white) |
| :--- | :--- | :--- |
|  |  | D AFBCGKJLIENMH |






|  | DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | D | AFBCGKJLIENMH |
|  | D |  | A | FBCGKJLIENMH |
|  | D |  | GA | FBCKJLIENMH |
|  | D |  | CGA | FBKJLIENMH |
|  | D |  | BCGA | FKJLIENMH |



|  | DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | D | AFBCGKJLIENMH |
|  | D |  | A | FBCGKJLIENMH |
|  | D |  | GA | FBCKJLIENMH |
|  | D |  | CGA | FBKJLIENMH |
|  | D \| |  | BCGA | FKJLIENMH |
|  | D \| BC |  | KGA | FJLIENMH |



|  | DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | D | AFBCGKJLIENMH |
|  | D |  | A | FBCGKJLIENMH |
|  | D |  | GA | FBCKJLIENMH |
|  | D |  | CGA | FBKJLIENMH |
|  | D 1 |  | BCGA | FKJLIENMH |
|  | D $\mathrm{BC}^{\text {B }}$ |  | KGA | FJLIENMH |
| $\geqslant$ | D $\mathrm{BC}^{\text {d }}$ |  | HKGA | FJLIENM |


|  | DFS finish order (black) | Stack (gray) | Rest (white) |
| :---: | :---: | :---: | :---: |
|  |  | D | AFBCGKJLIENMH |
|  | D | A | FBCGKJLIENMH |
|  | D | GA | FBCKJLIENMH |
|  | D | CGA | FBKJLIENMH |
| © | D 1 | BCGA | FKJLIENMH |
|  | D $\mathrm{BC}^{\text {d }}$ | KGA | FJLIENMH |
|  | D $\mathrm{BC}^{\text {D }}$ | HKGA | FJLIENM |
|  | D $\mathrm{BC}^{\text {d }}$ | FHKGA | JLIENM |


|  | DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | D | AFBCGKJLIENMH |
|  | D |  | A | FBCGKJLIENMH |
|  | D |  | GA | FBCKJLIENMH |
|  | D |  | CGA | FBKJLIENMH |
|  | D 1 |  | BCGA | FKJLIENMH |
|  | D \| BC |  | KGA | FJLIENMH |
|  | D \| BC |  | HKGA | FJLIENM |
|  | D 1 BC |  | FHKGA | JLIENM |
|  | D ${ }^{\text {BCFHKGA }}$ |  | J | LIENM |


|  | DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | D | AFBCGKJLIENMH |
|  | D |  | A | FBCGKJLIENMH |
|  | D 1 |  | GA | FBCKJLIENMH |
|  | D |  | CGA | FBKJLIENMH |
| $8$ | D 1 |  | BCGA | FKJLIENMH |
|  | D \| BC |  | KGA | FJLIENMH |
|  | D $\mathrm{BC}^{\text {B }}$ |  | HKGA | FJLIENM |
|  | D $\mathrm{BC}^{\text {D }}$ |  | FHKGA | JLIENM |
|  | D\|BCFHKGA |  | J | LIENM |
| $\bigcirc$ | D \| BCFHKGA ${ }^{\text {d }}$ |  | L | IENM |


|  | DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | D | AFBCGKJLIENMH |
|  | D |  | A | FBCGKJLIENMH |
|  | D |  | GA | FBCKJLIENMH |
|  | D 1 |  | CGA | FBKJLIENMH |
| $8$ | D 1 |  | BCGA | FKJLIENMH |
|  | D $\mathrm{BC}^{\text {C }}$ |  | KGA | FJLIENMH |
|  | D $\mathrm{BC}^{\text {B }}$ |  | HKGA | FJLIENM |
|  | D $\mathrm{BC}^{\text {C }}$ |  | FHKGA | JLIENM |
|  | D ${ }^{\text {BCFHKGA }}$ |  | J | LIENM |
| $\stackrel{\ominus}{C}$ | D \| BCFHKGA \| J |  |  | IENM |
|  | D\|BCFHKGA ${ }^{\text {d }}$ IL |  | I | ENM |


|  | DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | D | AFBCGKJLIENMH |
|  | D |  | A | FBCGKJLIENMH |
|  | D |  | GA | FBCKJLIENMH |
|  | D |  | CGA | FBKJLIENMH |
|  | D |  | BCGA | FKJLIENMH |
|  | D \| BC |  | KGA | FJLIENMH |
|  | D ${ }^{\text {B }}$ C |  | HKGA | FJLIENM |
|  | D $\mathrm{BC}^{\text {C }}$ |  | FHKGA | JLIENM |
|  | D ${ }^{\text {BCFHKGA }}$ |  |  | LIENM |
|  | D ${ }^{\text {BCFFHKGA }}$ J |  |  | IENM |
|  | D ${ }^{\text {BCFFHKGA }}$ J 1 L |  |  | ENM |
|  |  |  |  | NM |


|  | DFS finish order (black) | Stack (gray) |  | Rest (white) |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | D | AFBCGKJLIENMH |
|  | D |  | A | FBCGKJLIENMH |
|  | D |  | GA | FBCKJLIENMH |
|  | D |  | CGA | FBKJLIENMH |
|  | D |  | BCGA | FKJLIENMH |
|  | D ${ }^{\text {BC }}$ |  | KGA | FJLIENMH |
|  | D $\mathrm{BC}_{\text {B }}$ |  | HKGA | FJLIENM |
|  | D $\mid$ BC |  | FHKGA | JLIENM |
|  | D ${ }^{\text {BCFHKGA }}$ |  |  | LIENM |
|  | D\|BCFHKGA|J |  |  | IENM |
|  |  |  |  | ENM |
|  |  |  |  | NM |
|  | D ${ }^{\text {BCFHKGA }}$ J $1 \mathrm{~L}\|\mathrm{I}\| \mathrm{E}$ |  |  | M |


|  | DFS finish order (black) | Stack (gray) | Rest (white) |
| :---: | :---: | :---: | :---: |
|  |  | D | AFBCGKJLIENMH |
|  | D | A | FBCGKJLIENMH |
|  | D | GA | FBCKJLIENMH |
|  | D | CGA | FBKJLIENMH |
| ৪ios | D 1 | BCGA | FKJLIENMH |
|  | D $\mathrm{BC}^{\text {B }}$ | KGA | FJLIENMH |
|  | D $\mathrm{BC}^{\text {B }}$ | HKGA | FJLIENM |
|  | D $\mathrm{BC}^{\text {C }}$ | FHKGA | JLIENM |
|  | D ${ }^{\text {BCFHKGA }}$ | J | LIENM |
|  | D\|BCFHKGA ${ }^{\text {d }}$ | L | IENM |
|  |  | I | ENM |
|  | D\|BCFHKGA ${ }^{\text {J }}$ / $\mathrm{L} \mid \mathrm{I}$ | E | NM |
|  | D/BCFHKGA ${ }^{\text {J }}$ \|L|I|E | N | M |
|  | D\|BCFHKGA ${ }^{\text {d }}$ \|L|I|E | MN |  |



|  | DFS finish order (black) | Stack (gray) | Rest (white) |
| :---: | :---: | :---: | :---: |
|  |  | D | AFBCGKJLIENMH |
|  | D | A | FBCGKJLIENMH |
|  | D | GA | FBCKJLIENMH |
|  | D | CGA | FBKJLIENMH |
| $\stackrel{\otimes}{8}$ | D 1 | BCGA | FKJLIENMH |
|  | D $\mathrm{BC}^{\text {C }}$ | KGA | FJLIENMH |
|  | D $\mathrm{BC}^{\text {B }}$ | HKGA | FJLIENM |
|  | D $\mathrm{BC}^{\text {B }}$ | FHKGA | JLIENM |
|  | D\|BCFHKGA| | J | LIENM |
| $\begin{aligned} & \text { O} \\ & \underset{6}{C} \\ & \hline 0 \\ & \hline 1 \end{aligned}$ | D\|BCFHKGA ${ }^{\text {d }}$ | L | IENM |
|  |  | I | ENM |
|  | D\|BCFHKGA ${ }^{\text {d }}$ ILII | E | NM |
|  | D ${ }^{\text {BCFHKGA }}$ / $\mathrm{J}\|\mathrm{L}\| \mathrm{I} \mid \mathrm{E}$ | N | M |
|  | D/BCFHKGA ${ }^{\text {d }}$ \|L|I|E | MN |  |
| C | D\|BCFHKGA ${ }^{\text {d }}$ \|L|I|E|MN |  |  |






## Analysis of decomposition into SCCs of digraph $\mathbf{G}=(\mathrm{V}, \mathrm{E})$

Assume $G$ is represented by adjacency lists: 1. DFS in $G: \Theta(V+E)$ time, including reversal 2. Compute transpose $\mathrm{G}^{\top}: \Theta(\mathrm{V}+\mathrm{E})$ time 3. $D F S$ in $G^{\top}: \Theta(V+E)$ time 4. Output SCCs: $\Theta(V+E)$ time Total: $\Theta(V+E)$ time

A strongly connected digraph has one SCC.

## TOPOLOGICAL SORT

- A topological sort is a linear ordering of all nodes of a directed acyclic graph (DAG). - If there is a path from node $n_{i}$ to node $n_{j}$ then $n_{i}$ appears before $n_{j}$ in the ordering. Note on DAGs:
- There is no cycle (path from node to itself)
- There is at least one node of in-degree 0
- There is at least one node of out-degree 0


## TOPOLOGICAL SORT

## 잉 Example:

- We are given courses at a university and the prerequisite relations between.
- These can be modelled as a DAG.
- A topological sort of these courses gives a valid sequence of taking them.








## IntroIT

## TOPOLOGICAL SORT



## IntroIT



## TOPOLOGICAL SORT



## IntroIT

## TOPOLOGICAL SORT



## IntroIT, PKD

## TOPOLOGICAL SORT



## IntroIT, PKD






## TOPOLOGICAL SORT



## IntroIT, PKD, DArkDig



## TOPOLOGICAL SORT



IntroIT, PKD, DArkDig, AI


## TOPOLOGICAL SORT



## IntroIT, PKD, DArkDig, AI

## TOPOLOGICAL SORT



## IntroIT, PKD, DArkDig, AI



## TOPOLOGICAL SORT



IntroIT, PKD, DArkDig, AI, AD2

## TOPOLOGICAL SORT

## Av Algo

- Select AD2
- Print AD2

■ Remove it

IntroIT, PKD, DArkDig, AI, AD2

## TOPOLOGICAL SORT

## IntroIT, PKD, DArkDig, AI, AD2

## TOPOLOGICAL SORT

IntroIT, PKD, DArkDig, AI, AD2, AvAlgo

## TOPOLOGICAL SORT

- Select AvAlgo
- Print AvAlgo
- Remove it

IntroIT, PKD, DArkDig, AI, AD2, AvAlgo

## TOPOLOGICAL SORT

- A topological sort is not necessarily unique:
* At any step, we may have more than one node with in-degree 0 , so we have to choose one among them.
* IntrolT, PKD, AI, DArkDig, AD2, AvAlgo is also a valid sequence.

Other Algorithm and Analysis of Topological Sort

- Algorithm: Compute DFS finish time of each node; as each node is finished (blackened), insert it onto the front of an initially empty linked list; return that list.
- The total running time is thus $\Theta(V+E)$.

