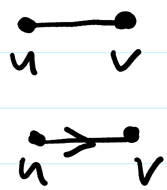
Review: Graphs, DFS, Topological search

Graph $G=(V,E)$

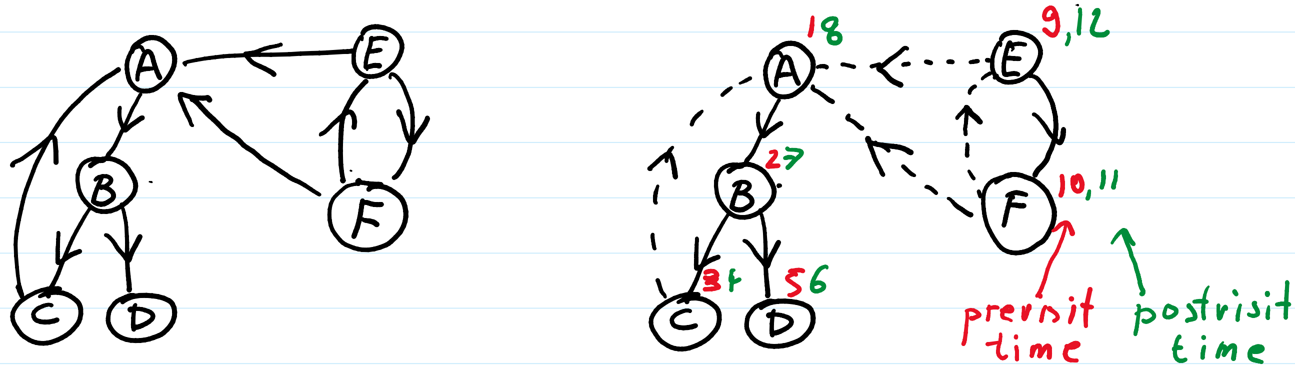
edges are undirected, $\{u,v\} \in E$

or directed, $(u,v) \in E$

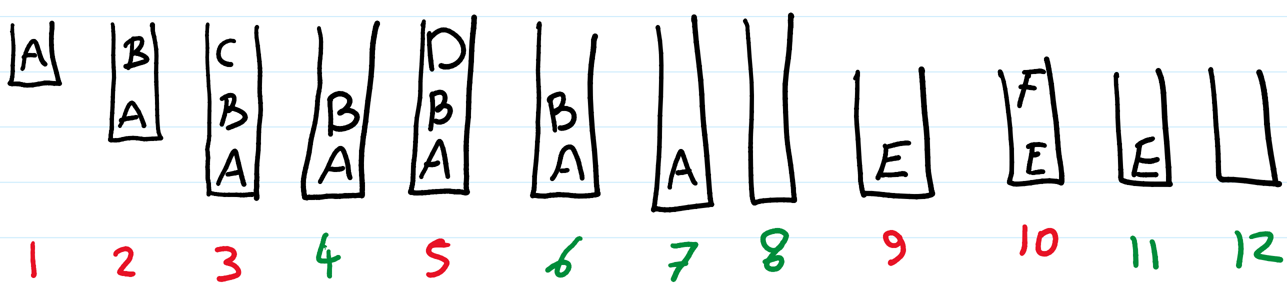


Oriented Graph

DFS-tree



Stack keeping track of the inductive calls of Explore(v)



Back edges $[3,4] \subseteq [1,8]$ $[10,11] \subseteq [9,12]$

Finding Cycles

G has cycle $\iff$ $\exists$ back edge in DFS

$\Leftrightarrow \exists\ uv$ s.th $[pre(u), post(u)] \subseteq [pre(v), post(v)]$

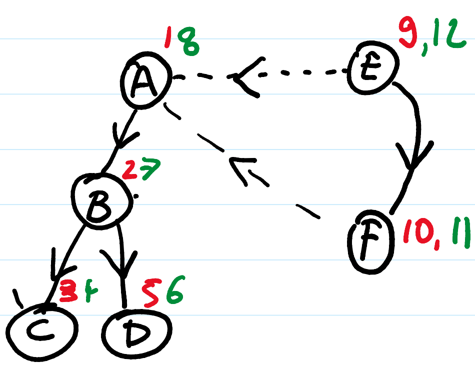
Topological Search

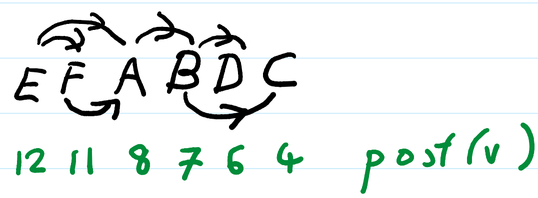
DAG (Directed Acyclic Graph) : no cycle

Task: Find liear order s.th. all edges point Forward,

i.e. $uv \in E \Rightarrow u < v$

Algorithm : Reverse sort by post(v)





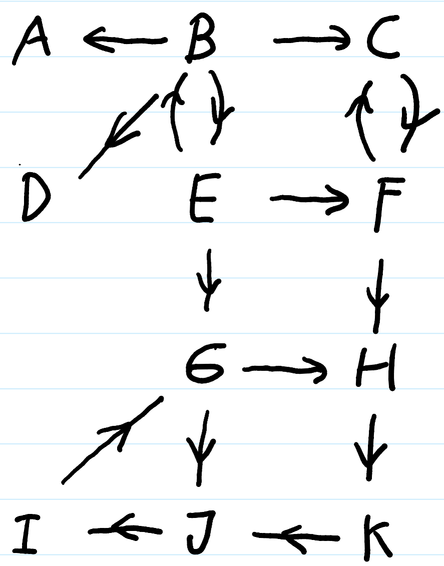
Property:

the last vertex is a sink (no out-link)

the first ——〃—— source (no in-link)

$\Rightarrow$ every DAG has at least one source and one sink

Strongly Connected Components



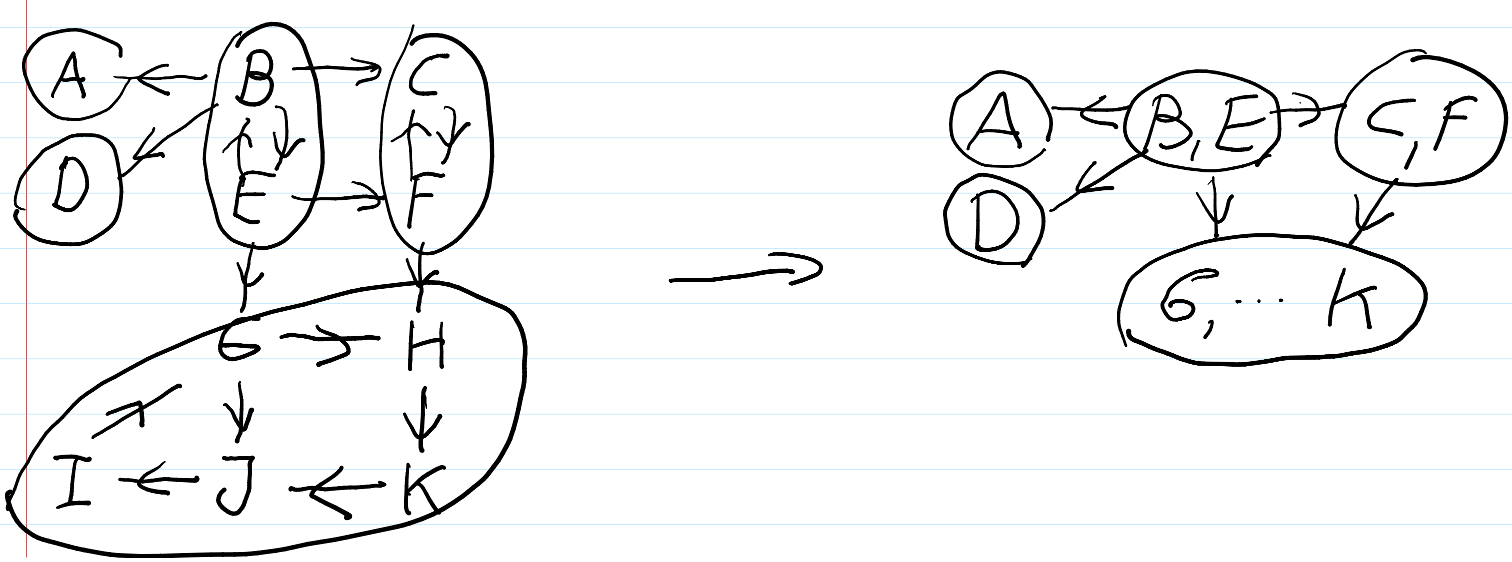
What is analog of connectivity in oriented graphs?

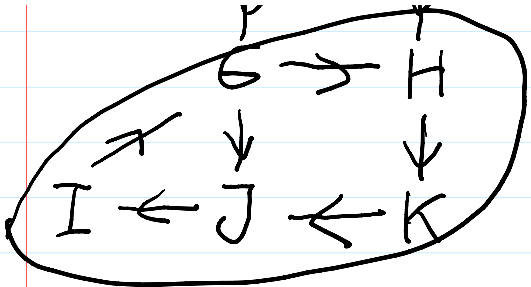
Def: $u, v \in V$ are strongly connected $\Leftrightarrow$ $\exists$ path $u \to v$ and $\exists$ path $v \to u$

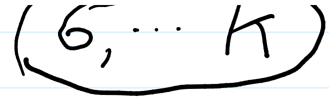
$\Rightarrow$ strongly connected components SCC

$\Leftrightarrow$ $\forall u, v \in SCC$ $\exists$ path $u \to v \to u$

Claim: Every directed graph is a DAG of its SCCs







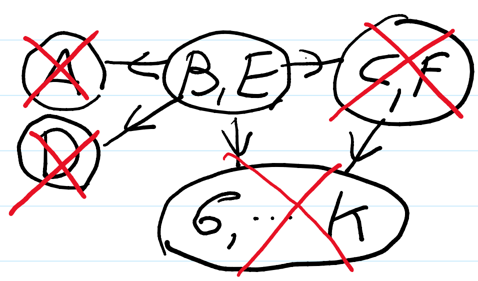
Pf: Assume not $\Rightarrow$ $\exists$ cycle involving different SCC $\Rightarrow$ $\lightning$

Finding the SCC of G

Property 1: Explore(G, v) terminates exactly when all nodes reachable from v have been visited

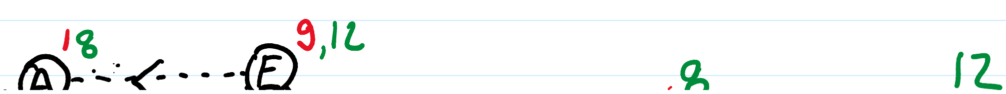
Alg. Idea: start BFS in Sink-SCC

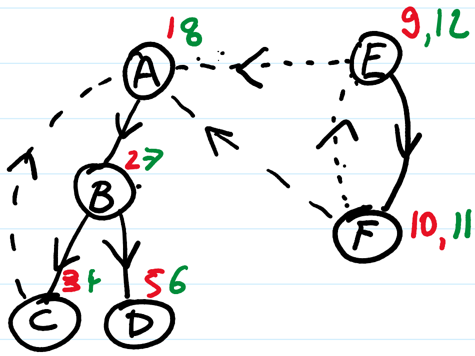
- Find SCC
- Remove SCC
- Iterate

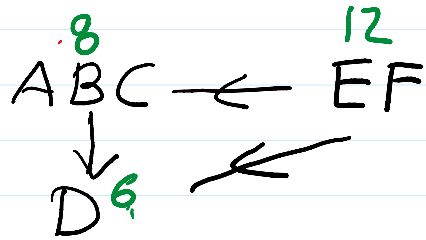


Prop 2: The node with highest post(v) must lie in a Source-SCC

Ex:







In general, this a corollary of

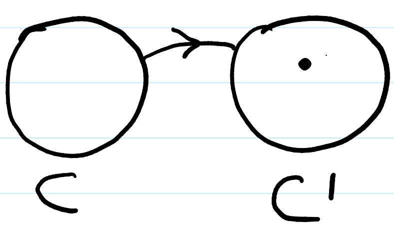
Property 3: If we reverse sort SCCs by highest post in each SCC $\Rightarrow$ all edges between SCC point forward

Proof: Consider SCCs C, C' with an edge from $C \to C'$

- Need to prove

$$\text{highest post}(C) > \text{highest post}(C')$$

Case 2: DFS visits C' before C



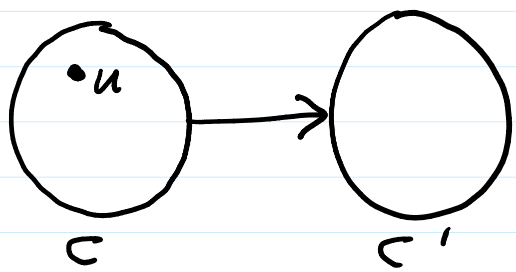
- Explore reaches all of C'
- Dies before reaching C

C $\quad$ C'

- Dies before reaching C

$\Rightarrow$ claim

<u>Case 1:</u> DFS visits C before C'



Explore(u) reaches all of $C \cup C'$

$\Rightarrow$ u stays in stack longest $\forall\, v \in C \cup C'$

$\Rightarrow$ $post(v) \leq post(u)$ $\Rightarrow$ claim ■

Algorithm:

<u>Property 1:</u> If we start Explore in a vertex v in a sink component, it finds the SCC of v

<u>Property 2:</u> If we find v with highest

post(v) ⇒ v ∈ source component

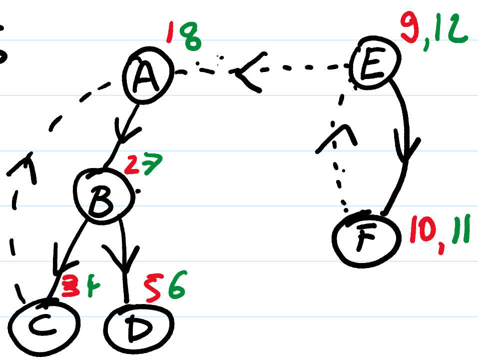
<u>Solution:</u> Look at G^R with all edges reversed

- G^R has same SCC
- source comp. of G^R = sink ~~comp.~~ of G

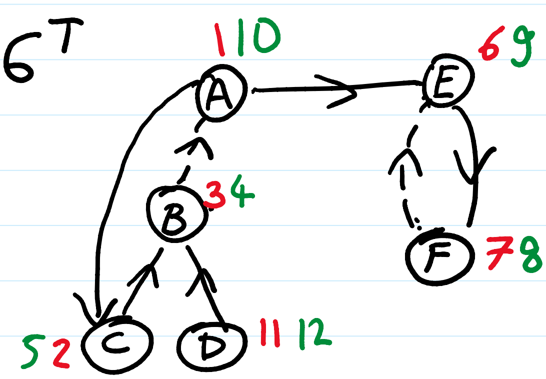
Iterate!

<u>Example</u>

G



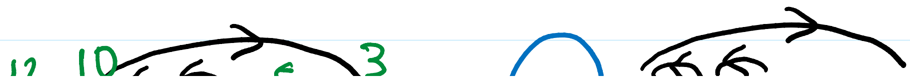
G^T



Order by post(·) in G^T

12 10 9 8 5 3
D A E F C B

BFS in G:



Final Algorithm

1) Run DFS on G^R
2) Order vertices in decreasing post(v) of G^R
3) Run DFS on G using the order in 2 for any new explore proceedure
4) Augment SCC-number by 1

after every restart of explore

Why does this work

- We use the topological search for the SCCs of G^T to make sure that for each restart, we start in a sink SCC of the truncated Graph

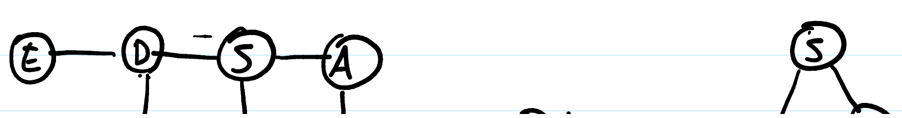
12	10	9	8	5	3
D	A	E	F	C	B

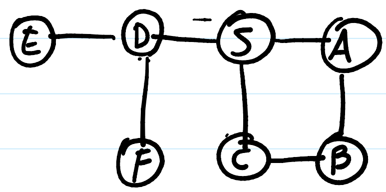
becomes

12	10 5 3	9 8
D	A C B	E F

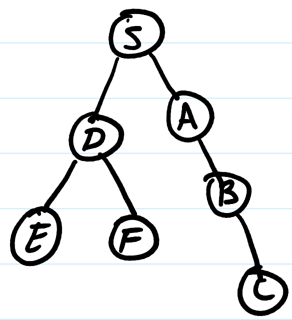
Breadth-First-Search

Goal: Determine distances from source node S

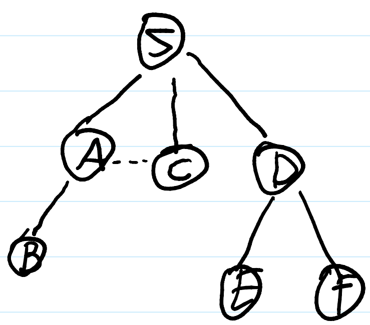




DFS



<u>BFS:</u> First explore neighbors distance 1, then 2, ...



$\left.\begin{matrix} 0 \\ 1 \\ 2 \end{matrix}\right\}$ dist. from S

<u>Procedure BFS(G, s)</u>

Input G, v

Output $dist(u) = dist(s, u)$

For all $u \in V$

$\quad dist(u) = \infty$

$dist(s) = 0$

$Q = \{s\}$

While $Q \neq \emptyset$

$u = \text{eject } Q$

For all edge $\{u, v\} \in E$

If $\text{dist}(v) = \infty$

$\text{inject}(v, Q)$

$\text{dist}(v) = \text{dist}(u) + 1$

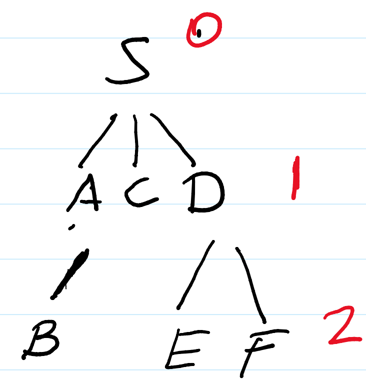
S

DCA

BDC

BD

FEB



Correctness Proof:

Show that at some time t_d,

Q contains all vertices at $\text{dist}(u) = d$