Thursday, March 24, 2022 10:54 AM

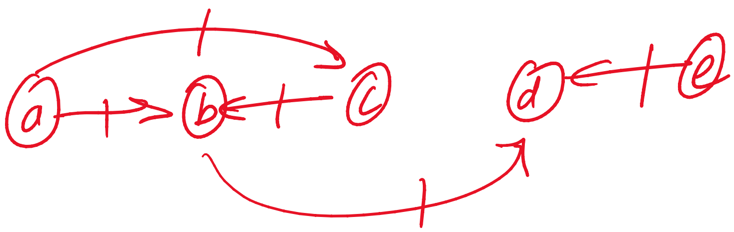
Applications of DFS:

☆ Topological Sort (TS):

Given a dir graph G,
Find ordering of vertices s.t.
$(u,v) \in E \Rightarrow u$ comes before v.

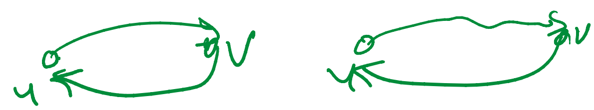
```
DFSALL(G) {
1. unmark all vertices
2. for u ∈ V
     if u is unmarked
       DFS(G, u)
```

eg.



a c b e d   a e c b d

Q: Does TS always exist? NO!

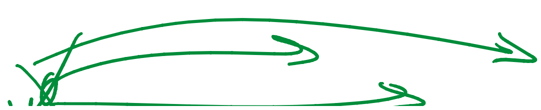


If $\exists$ a cycle then no TS.

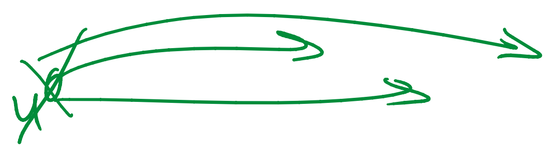
☆ So lets assume that G is acyclic (DAG).

Aside: A DAG always has a source node (in-deg(u)=0)

& a sink node (out-deg=0)



First Algo :

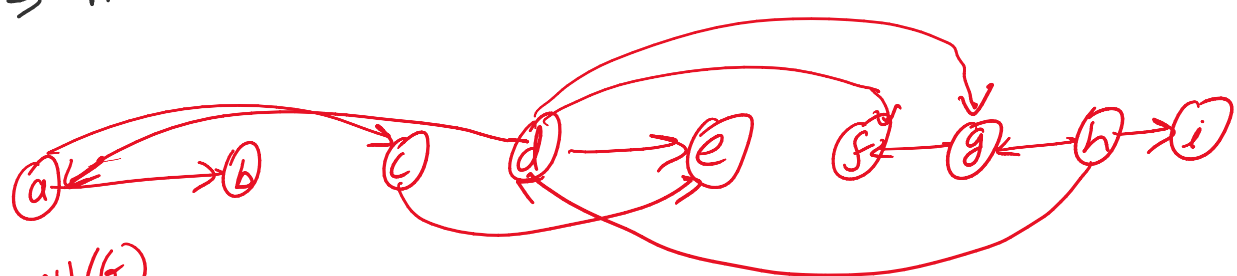


→ Find a source vertex u.
Output u.
Remove u & it's edges. Repeat.

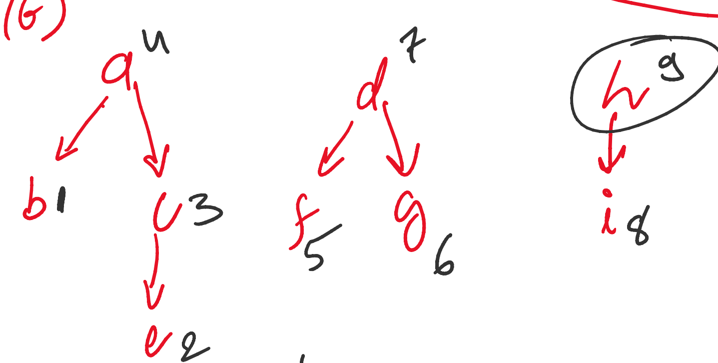
Q : How to find a source vertex ?

→ Run DFSAll(G)
→ Pick u = vertex w/ largest finish time.

eg.



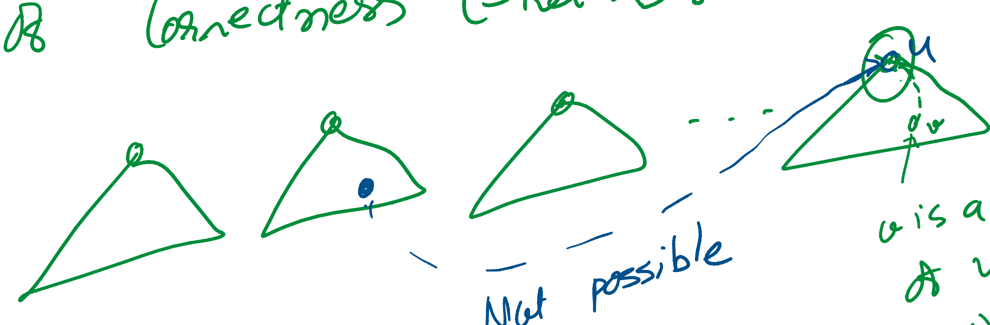
DFSAll(G)



DFS(G, u){
- Mark u
⋮
- Finish(u) = time++
} output u.

b e c a f g d i h

Proof of Correctness (Sketch):

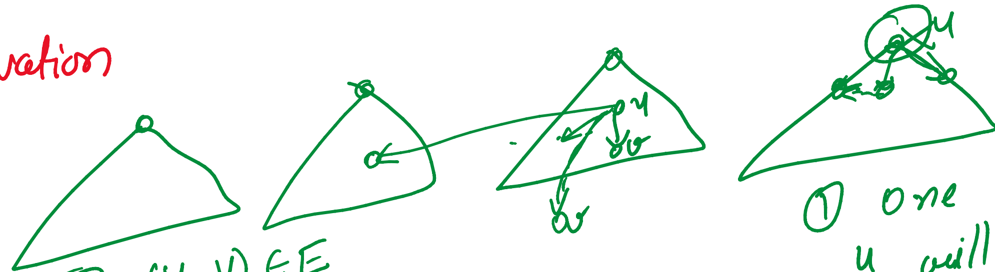


v is a descendant of u.
⇒ (v, u) is a back edge
⇒ ∃ a cycle in G!

back edge

$\Rightarrow \exists$ a cycle in G!

Observation



① One of child of u will have second largest finish time.

Assume G is a DAG ← ② $(u, v) \in E$

finish(u) > finish(v)

Q: How to remove u & repeat?

Do nothing! Just pick the vertex w/ the next largest finish time.

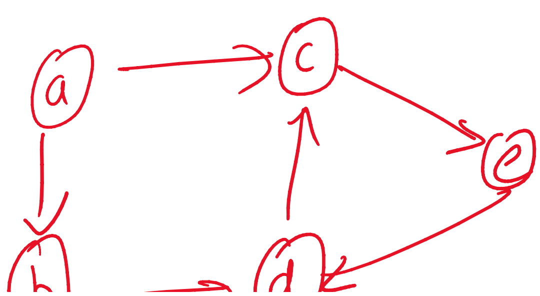
Final Algo:

1. Run DFSAll(G) & Compute finish times.
2. Output vertices in the decreasing order of finish time

Running time $O(n+m)$

Corollary: $\exists$ a topological sort $\Leftrightarrow$ G is acyclic (DAG).

★ Strongly Connected Component. (SCC)

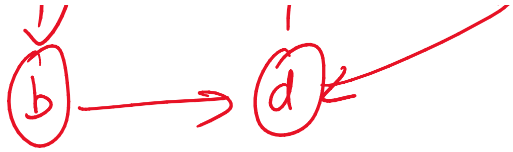


$a \rightsquigarrow e$

$e \not\rightsquigarrow a$

$c \rightsquigarrow e$ $e \rightsquigarrow c$

... strongly connected



$c \rightsquigarrow e \quad e \rightsquigarrow c$

e, c are strongly connected

Given a dir graph G, we say $u, v \in V$ are s.c. iff $\exists\ u \rightsquigarrow v$ & $v \rightsquigarrow u$.

s.c. Relation: symmetric, reflexive, $u \leftrightarrow v \leftrightarrow w$ transitive

is an equivalence Relation

$\Downarrow$

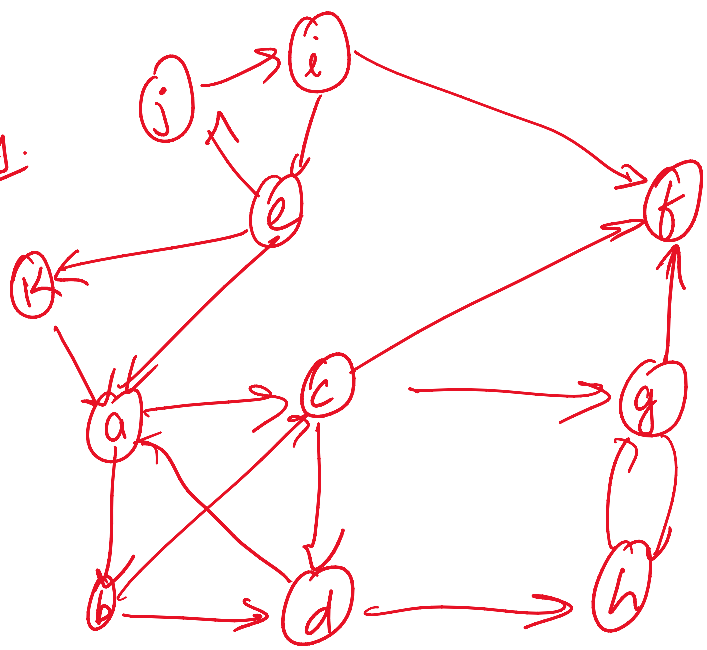
Decomposes V into disjoint components/sets.

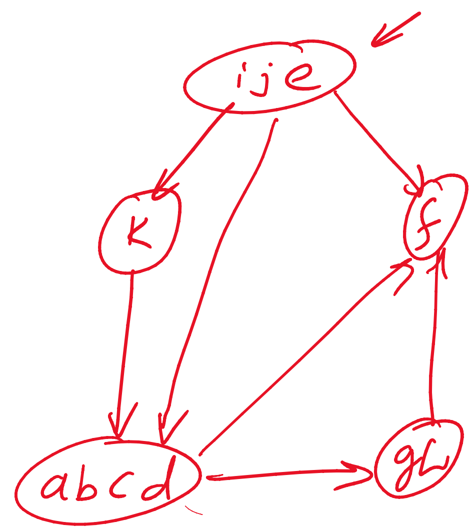
Find SCC of dir graph G:

Partition V into components s.t.

u, v are in the same component $\Leftrightarrow$ $u \rightsquigarrow v$ & $v \rightsquigarrow u$ (u, v are s.c.)

eg.





Meta-graph is acyclic (DAG).

(DAG).

$\{j, l, e\}$, $\{a\ b\ c\ d\}$, $\{g, h\}$, $\{f\}$, $\{K\}$

within a S.C.C we can go from anywhere to anywhere

Appl'n - Control flow in program, ...
Simplify a dir graph into a DAG

Naive Approach:
- test readability of every pair of vertices.
& find S.C.C.
for a node, just run a DFS/BFS.
$O(n) * O(m+n) = O(n(m+n))$

- Find a cycle, shrink it, repeat.

History:

Purdom '68 $O(n^2)$
Munro '71 $(m + n \log n)$
Tarjan '72 $O(m+n)$ ← complex.
Kosaraju '78 } $O(m+n)$ simple.
⇒ Sharir '81

First idea:

First idea:

1. Find a vertex u in the source component of the meta-graph
2. Find u's component
3. Remove & repeat.

Q: How to find a vertex in the source component?

1. Run DFSAll(G)
2. Pick u = vertex of largest finish order.

Proof sketch: