▷ Existance ? Go to the first, then walk along.

Worst case: $O(V)$.

b) Adjaceney matrix.

0-1 matrix (this is pretty to understand).

1 ← with edge.

0 ← no edge

▷ Memory $O(|V|^2)$. of memory for storing.

▷ Existence of edge: $O(1)$. (query

② What about indeg and outdeg ? (Answer may vary).

3. Add extra info.

Add extra info to the node section.

4. Graph traversal.

a) BFS

white initially

gray visiting, into queue

black visited.

```
BFS(G, s)
for each vertex u ∈ V[G] − {s}
    do color[u] ← WHITE   // unvisited
       d[u] ← ∞           dist : number of edges to the source
       π[u] ← NIL         predicessor
color[s] ← GRAY  // first time seen, put in queue
d[s] ← 0
π[s] ← NIL
Q ← Ø
ENQUEUE(Q, s)
while Q ≠ Ø
    do u ← DEQUEUE(Q)
       for each v ∈ Adj[u]
           do if color[v] = WHITE
                 then color[v] ← GRAY
                      d[v] ← d[u] + 1
                      π[v] ← u
                      ENQUEUE(Q, v)
       color[u] ← BLACK
                 // last time seen, out of queue
```

$O(v)$

and numbers tracking show min.mum for 2 points.

▷ Runtime. · Each vert is enqueued at most once.
· Each vert can be at most adjacent to others. Edge touched once.

→ $O(V+E)$.

b) DFS

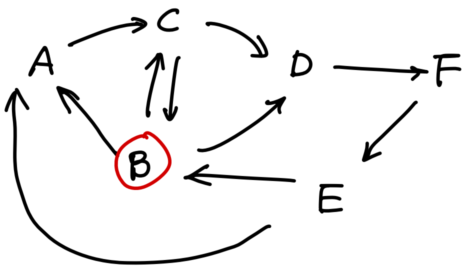
```
DFS(G)
for each vertex u ∈ V[G]
    do color[u] ← WHITE
       π[u] ← NIL
time ← 0
for each vertex u ∈ V[G]
    do if color[u] = WHITE
          then DFS-VISIT(u)
```

If we track the time,
$\forall a \rightsquigarrow b$, the time of a is in time of b.

```
DFS-VISIT(u)
color[u] ← GRAY                ▷ White vertex u has just been discovered.
time ← time + 1
d[u] ← time                    ← first time see
for each v ∈ Adj[u]            ▷ Explore edge (u, v).
    do if color[v] = WHITE
          then π[v] ← u
               DFS-VISIT(v)
color[u] ← BLACK               ▷ Blacken u; it is finished.
f[u] ← time ← time + 1
```

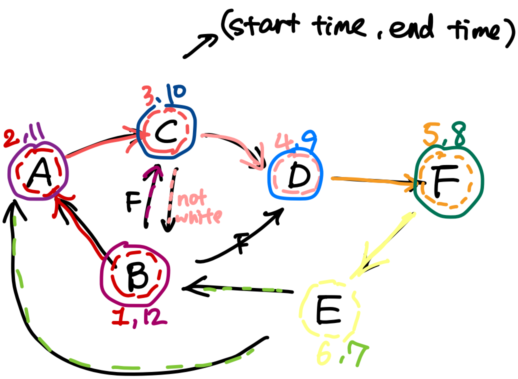
first time finish

◇ Breath first tree: edges used to visit unvisit nodes.



track by "time"

- when we reach
- leave.



Stack

This is also $O(V+E)$.

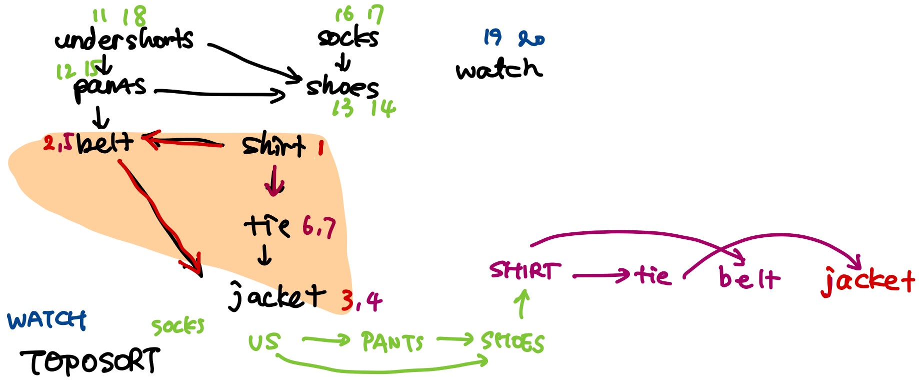
◇ Depth first Tree. Used to discover white nodes.

- Tree edges: ↗
- Back edges: connect to already-discovered node (ancestor)
  — cycles
- fwd eges: descendant of DFTree
- Cross edges. not ancestor, not descendant ones.

Topological Sort on Directed Acyclic Graphs

Example. $(u,v) :=$ u must be done before v.

a DAG, ask for possible sequence to do it



TOPOSORT

1. Call DFS(G) and compute $f[v]$ for each vtx v
2. each vertex is finished, insert it onto front of a linked list
3. return the linked list.

Why does it work?

Look at finish time, item with smallest finish time has to come with it.

by Parenthesis theorm. either of

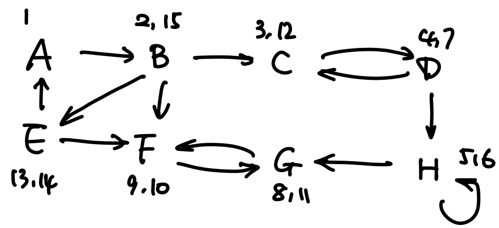
$$\begin{cases} d(u) < d(v) < f(v) < f(u) & \text{(nested)} \\ d(u) < f(u) < d(v) < f(v) & \text{(parallel).} \end{cases}$$

$\times$ $d(u) < d(v) < f(u) < f(v)$ impossible !!

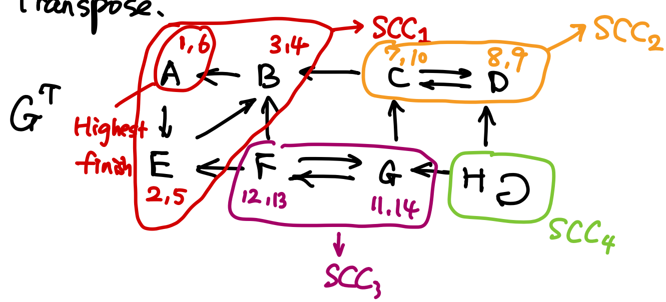
STRONGLY-CONNECTED-COMPONENT (SCC)

A graph is said to strongly conn. reachable from every other vertex

STRONGLY-CONNECTED-COMPONENTS



Transpose.



Why does this work? • G and G^T connectivity compoments dont change

• The comp. with largest finish time in the transpose can't get anywhere else.

QUESTION: How to make the transpose

Adj mat. directly get transpose

Adj.