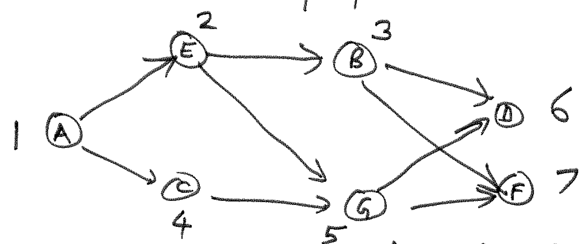
Directed Acyclic Graphs (DAG) and topological orderings

DAG $G=(V,E)$, a directed graph without directed cycles.



Def: A topological ordering of a DAG $G=(V,E)$ is a linear ordering of V such that all edges of G go from ~~sh~~ left to right in this ordering

$T: V \to \mathbb{Z}^+$ such that $(u,v) \in E \Rightarrow T(u) < T(v)$.

Theorem: A graph has a topological ordering of its vertices if and only if it is a DAG.

Algorithm 1 for top order

Idea: Repeat
- Find a node with no incoming edges
- Give it the smallest number available
- Delete the node along with its outgoing edges

Pseudocode: For each node u, ~~find~~ compute u.degree, the number of edges incoming to u.

```
Q = Queue of vertices with all vertices with degree = 0
top = 1
while Q is not empty do
    u = Q.remove()
    u.top = top++
    for each edge (u,v) ∈ E do (iterating on u.Adj)
        v.degree--
        if (v.degree == 0) Q.add(v)
```

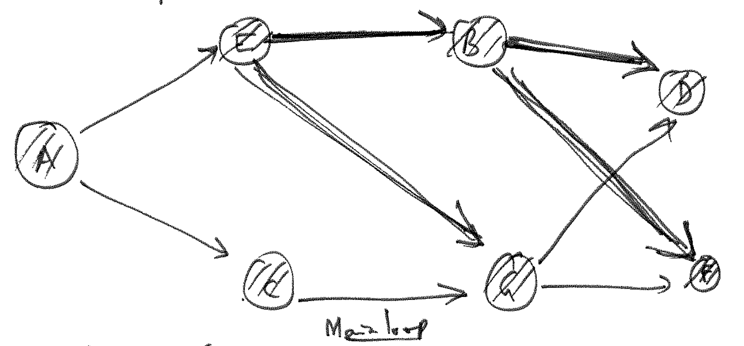
RT $= O(|V|+|E|)$

Algorithm 2 for top order: DFS = Depth-first search

Run DFS on G

At the end of DFS_Visit (u),

~~add~~ push u to a stack.



stack

A
C
E
G
B
F
D

Nodes:

~~E~~ : ~~B~~, ~~G~~ DFS.Visit(E)

~~B~~ : ~~D~~, ~~F~~

C : G DFS.Visit(C)

~~G~~ : F

~~D~~ —

~~F~~ — DFS_Visit(A)

strongly connected components

Framework: Directed graph $G = (V, E)$

Define: u and v are strongly connected if
there is a path from u to v and
" " v to u.

Nodes can be partitioned into strongly connected components

C_1 C_2 . . . . C_k

all nodes in C_i
are strongly connected
to each other

Algorithm for finding SCC:

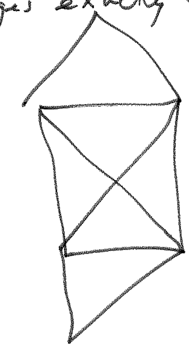
G^R = Reverse of G

1. Run DFS on G
   ↓
   Return stack of nodes in finish time order
   (DFS Top)
2. Run DFS on G^R
   ↳ use stack to decide order in outer loop

Euler tours:

Input: Undirected graph $G=(V,E)$

Q: Is there a tour that goes through all edges exactly once?



Theorem: G has an Euler tour iff

(a) G is connected

(b) degree of every node is even.

Graph has exactly $\underbrace{2 \text{ nodes}}_{u,\,v}$ of odd degree

Euler path: u ——————— v

all edges of G
exactly once

Otherwise? Find shortest tour that goes through every edge at least once — Chinese Postman Problem.