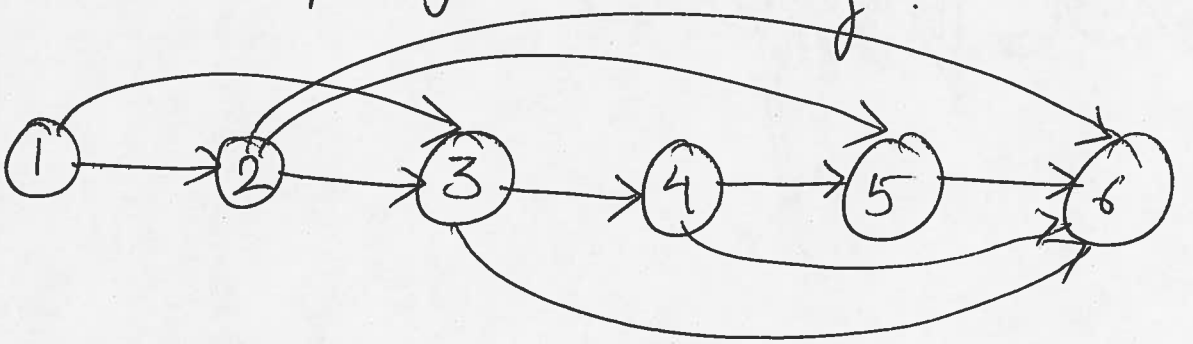
Thm: If G is a DAG $\Rightarrow$ G has a topological ordering.

Pf idea: We present an algo to compute the topological ordering.



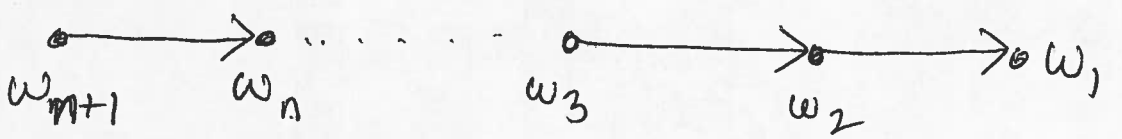
Q: How many incoming edges can the 1st ~~vertex~~ have?

A: Zero!

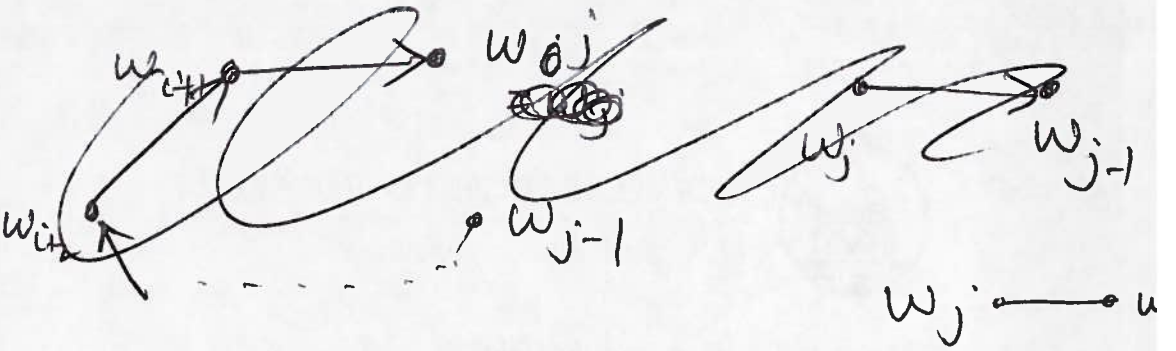
Lemma: If G is a DAG $\Rightarrow$ $\exists$ a vertex u with 0 incoming edge (0 in-degree)

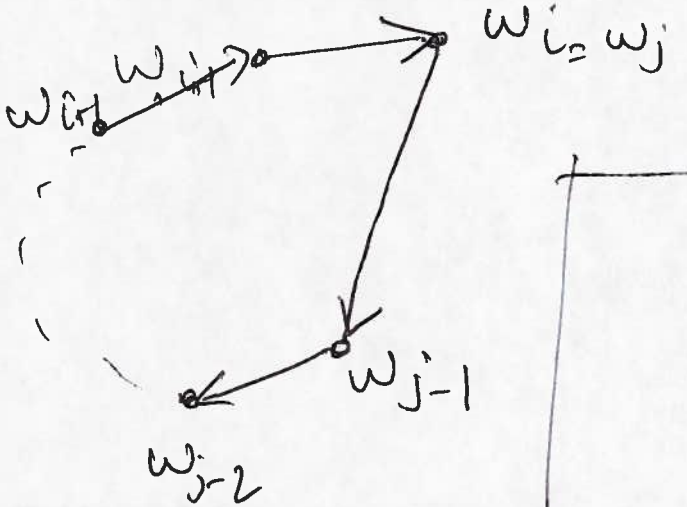
Pf: By contradiction.

Assume every vertex has ≥ 1 incoming edge.



$\Rightarrow$ (by Pigeon+hole principle) $\exists\ i \neq j$ $(i < j)$ s.t $w_i = w_j$





⇒ ∃ a directed cycle

⇒ contradiction as G is a DAG.

recursive calls $\leq n$

→ TopOrd (G)

Obs: Let G' be G without u & all it's edge

$O(n)$ → 1. Let u be a vertex with 0 incoming edges in G.

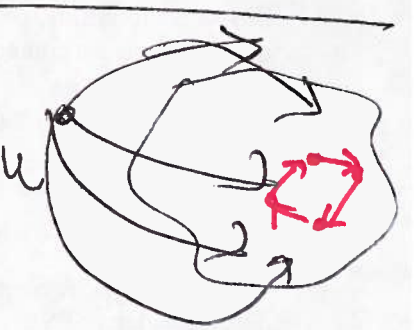
$O(n)$ → 2. Let G' be G without u & it's edges

3. Output the ordering

$O(1)$ → u, TopOrd(G')

→ Alg. is correct (by Lemma) +

→ G' is a DAG



THM: TopOrd can be implemented in $O(m+n)$ time.

$O(n^2)$

Step 1+2 : $O(n)$ time

Soln: Maintain two arrays of length n.

InDeg[v] = current indegree of v

Alive[v] = T / F

↑ If v is still in the current G.

Initialization, Update & query can be done in $O(n)$ time