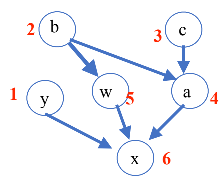
Topological order of a directed graph.
Kuhn Algorithm

- **Definition:** A **directed** graph G(V,E) that contains no cycles is called a directed acyclic graph (DAG).
- Given a DAG, assume that we are given also a unique label ***lbl[v]***, with each vertex ***v***. The label is a unique number between 1…n (n - number of vertices)
- Think about these labels as **order** by which we could visit the vertices: (first, second…last).
- **Definition:** we say that the vertices are **topologically sorted** if for every edge ***(u,v)***, the label of ***u*** is smaller than the label of ***v.***
  $(u,v) \in E$ implies $lbl((\mathbf{u}) < lbl(\mathbf{v})$

Given the DAG (without the labels, we want to find such an order if possible, or to be informed that no such order exists.

Example; The vertices are {a,b,c,x,y,w}
labels are the red numbers

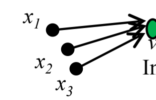
lbl[b]=2
lbl[c]=3
lbl[y]=1
….



Topological order of a directed graph.

Def: *InDegree(v, E),* be the number of edges that *"enter" v.*

InDegree(x_1,E)=0



InDegree(v,E)=3

Kuhn Algorithm:

Input: A directed graph G(V,E).
Output: a label for each vertex that is a topological order (if exists)

Algorithm: for every node v set lbl[**v**]=NULL
S ← Set of all nodes with no incoming edge in E. // (InDegree=0)
cnt=1 ;
while S is non-empty **do**
    remove a node u from ***S***
    lbl[u] =cnt ; cnt++ ;
    for each node v with an edge *(u,v) in E (each nbr of u)* **do**
        If lbl(v) is not NULL – Error. There are cycles. Else
        remove ***(u,v)*** from ***E***
        Indegree(v) --
        if v has no other incoming edges **then**
            insert v into S
if E is not empty **then** return error (graph has at least one cycle) **else** return the labels of all vertices.

Kuhn Algorithm:

Input: A directed graph G(V,E).
Output: a label for each vertex that is a topological order (if exists)

Algorithm: for every node v set lbl[**v**]=NULL
S ← Set of all nodes with no incoming edge in E. // (InDegree=0)
cnt=1 ;

Running time O(|V|+|E|)

while S is non-empty **do**
    remove a node u from ***S***
    lbl[u] =cnt ; cnt++ ;
    for each node v with an edge *(u,v) in E (each nbr of u)* **do**
        If lbl(v) is not NULL – Error. There are cycles. Else
        remove ***(u,v)*** from ***E***
        Indegree(v) --
        if v has no other incoming edges **then**
            insert v into S
if E is not empty **then** return error (graph has at least one cycle) **else** return the labels of all vertices.