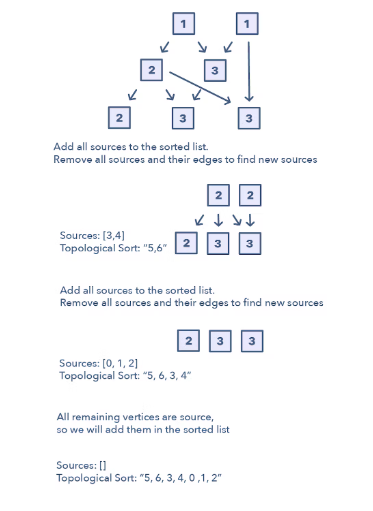
**Topological Sort**



**Ways to Identify the pattern:**

* The problem will deal with graphs that have no directed cycles
* If you’re asked to update all objects in a sorted order
* If you have a class of objects that follow a particular order

**Common problems you use the sliding window pattern with:**

* Task scheduling (medium)
* Minimum height of a tree (hard)

**Broad Categories:**

* None

**Strategies to Solve the Problem:**

* BFS Solution is simpler and more efficient to follow.

1. Initialization  
   a) Store the graph in adjacency lists by using a HashMap  
   b) To find all sources, use a HashMap to keep the count of in-degrees.
2. Build the graph and find in-degrees of all vertices

a) Build the graph from the input and populate the in-degrees HashMap.

1. Find all sources  
   a) All vertices with ‘0’ in-degrees will be sources and are stored in a Queue.
2. Sort  
   a) For each source, do the following things:  
   — i) Add it to the sorted list.  
   — ii)Get all of its children from the graph.  
   — iii)Decrement the in-degree of each child by 1.  
   — iv)If a child’s in-degree becomes ‘0’, add it to the sources Queue.  
   b) Repeat (a), until the source Queue is empty.
3. Return the Topological Order only when the length of the topological order is equal to length of vertices or in-degrees.

a) A topological ordering is possible only when the graph has no directed cycles, i.e. if the graph is a Directed Acyclic Graph (DAG).