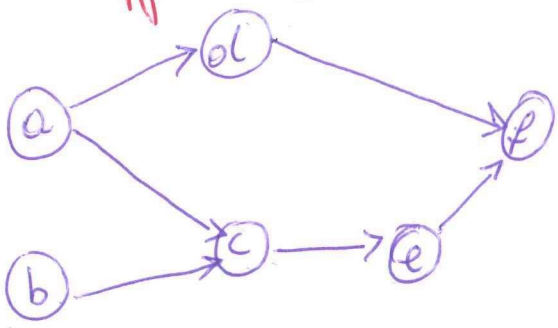
LAB 3

1. TOPOLOGICAL ORDER

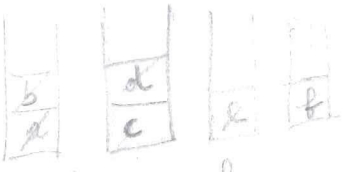
1.1. Modify the Algorithm in the slides (3.1) adding the array count[w] and a Stack S containing the nodes without incoming edges.
Assume we can check, given vertices w, v, whether $(v, w) \in G$.

1.2 Run the algorithm on the following DAG

1.3. Modify the Algorithm to "arbitrary" Graphs.



DAG



b, a, d, c, e, f.

```
main-TO (G)
  S ← [] ;
  for all w ∈ V  TopOrder array of n elements;
      compute count[w]; the number of incoming edges to w
      if count[w] = 0 then S.push(w);
  i = 1; /* index for the Top Order array */
  TO()
```

```
TO()
  If S = [] then return
  Else v = S.pop; TopOrder[i] = v;
       i = i + 1;
       for each (v, w) ∈ G
            count[w] = count[w] - 1;
            if count[w] = 0 then S.push[w];

       TO()
```