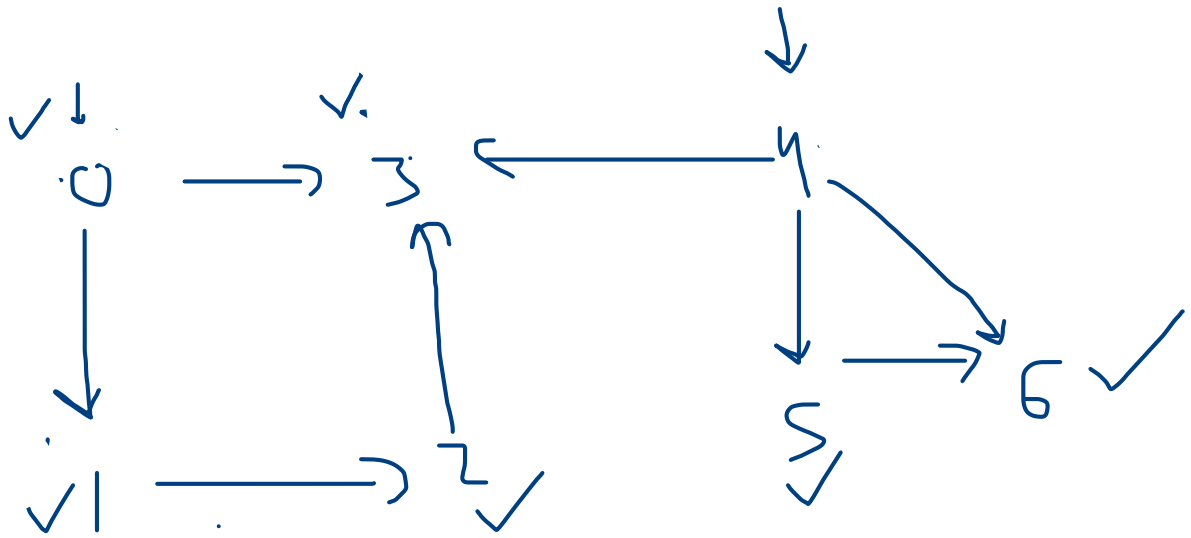
→ 0 1 2 4 5 6 3 ✓

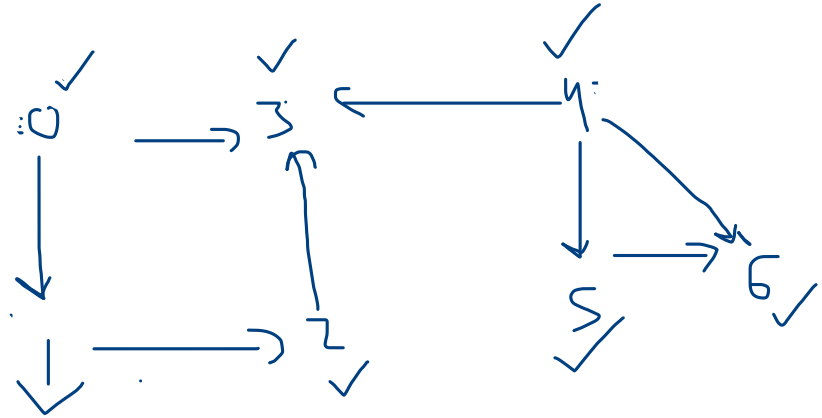
Topological Sort (DFS)

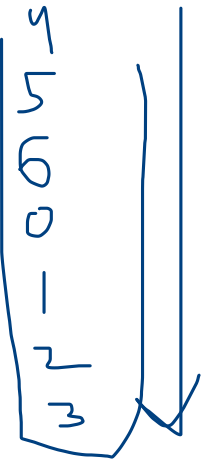
DAG



4
5
6
0
1
2
3

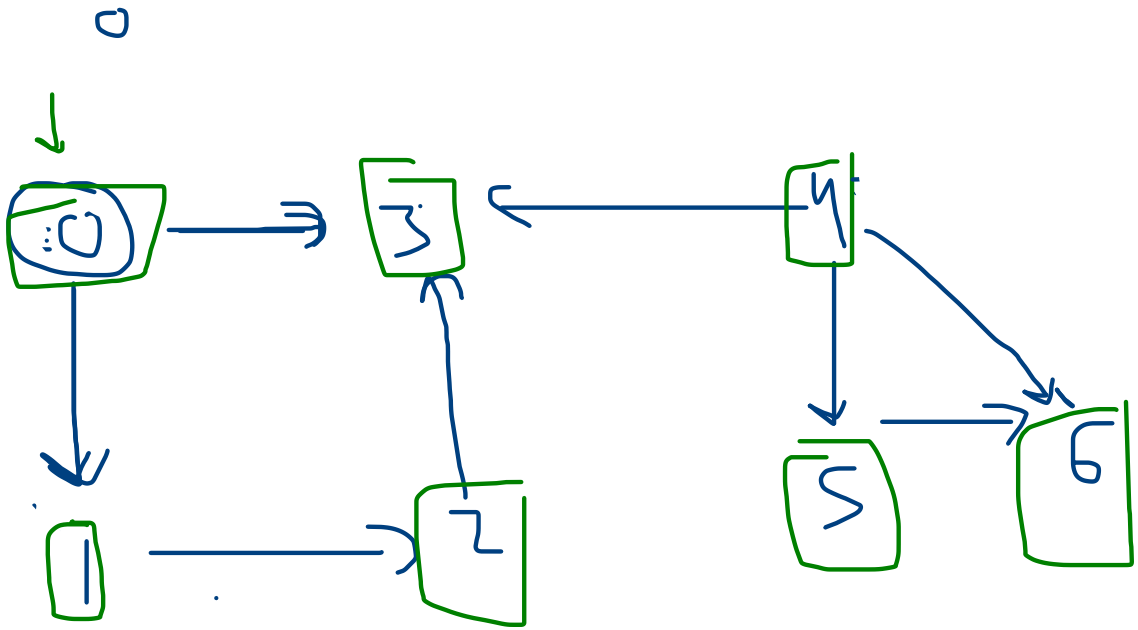
4 5 6 0 1 2 3





```java
public static void dfs(ArrayList<Edge>[] graph, int v) {

    boolean[] visited = new boolean[v];
    Stack<Integer> topologicalOrder = new Stack<>();
    for (int i = 0; i < v; i++) {
        if (! visited[i]) {
            dfsUtil(i, graph, visited, topologicalOrder);
        }
    }

    while(! topologicalOrder.isEmpty()) {
        System.out.print(topologicalOrder.pop() + " ");
    }

}

public static void dfsUtil(int src, ArrayList<Edge>[] graph, boolean[] visited, Stack<Integer> topologicalOrder) {

    visited[src] = true;

    for (Edge e: graph[src]) {
        if (!visited[e.nbr]) {
            dfsUtil(e.nbr, graph, visited, topologicalOrder);
        }
    }

    topologicalOrder.add(src);
}
```

Kahn's Algorithm (Topological Sort using BFS)

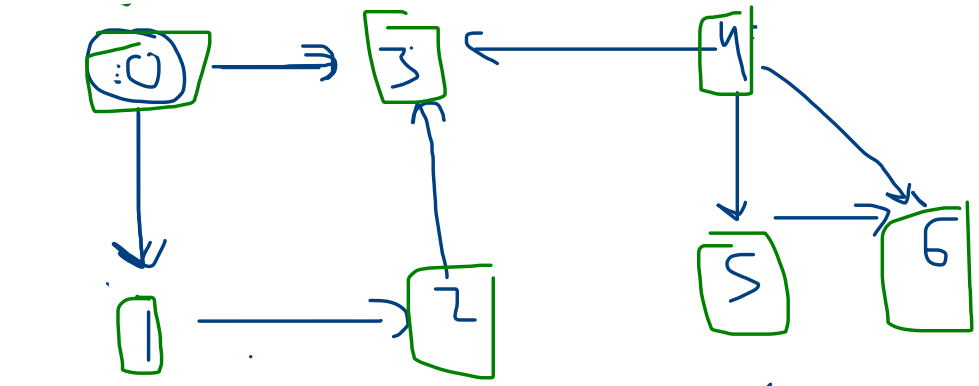


indegree =

0	1	2	3	4	5	6
0	1	1	3	0	1	2

outdegree =

0	1	2	3	4	5	6
2	1	1	0	3	1	0



indeg = 0

queue

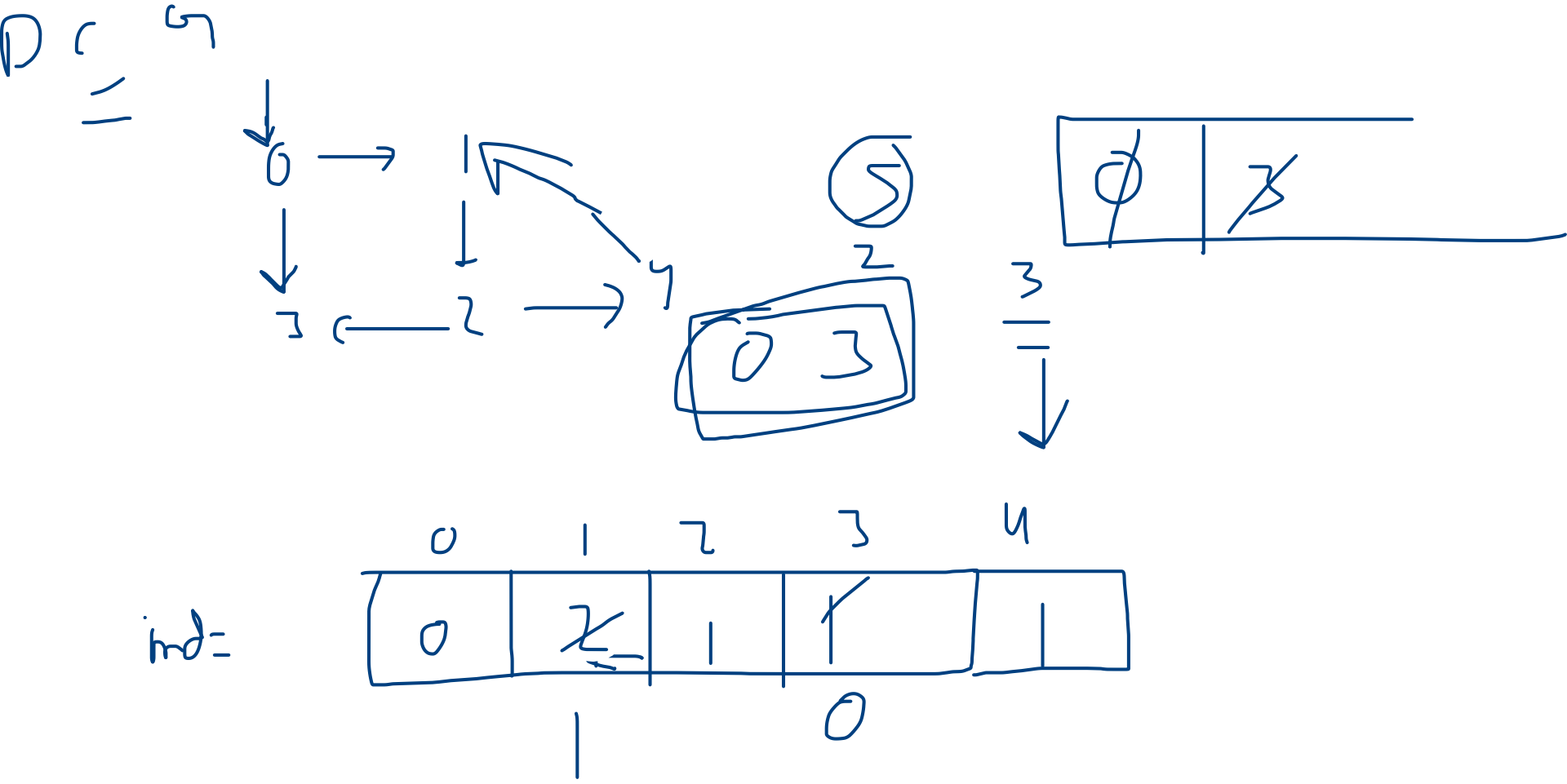
0 4 1 5 2

0 4 1 5 2 6 3

remove que
print
loop in nbr and decrement indegree
add vertex with indeg = 0 in que

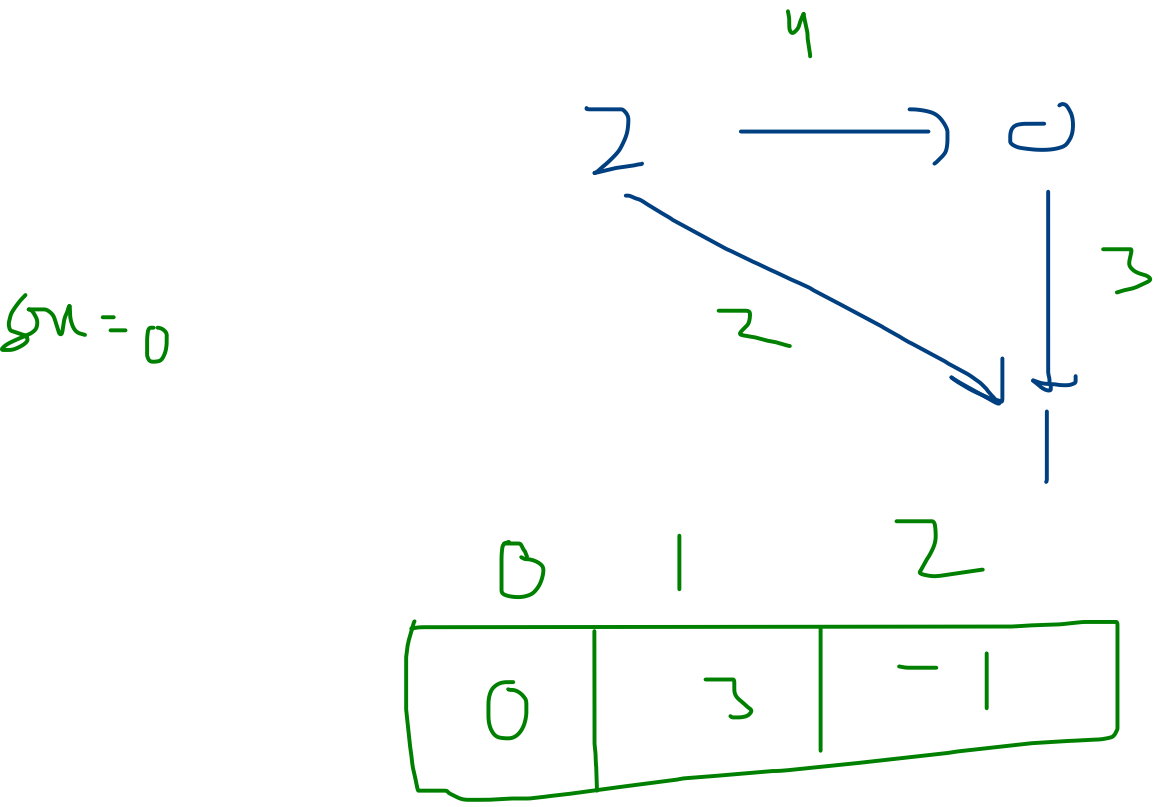
indeg = 0

0	1	2	3	4	5	6
0	1	1	3	0	1	2
	0	0	2	0	0	1


0
1
2
3
4
5
0 3
0 2
ind=
0 1 2 3 4
0 2 1 1 1
1
0

Course Schedule

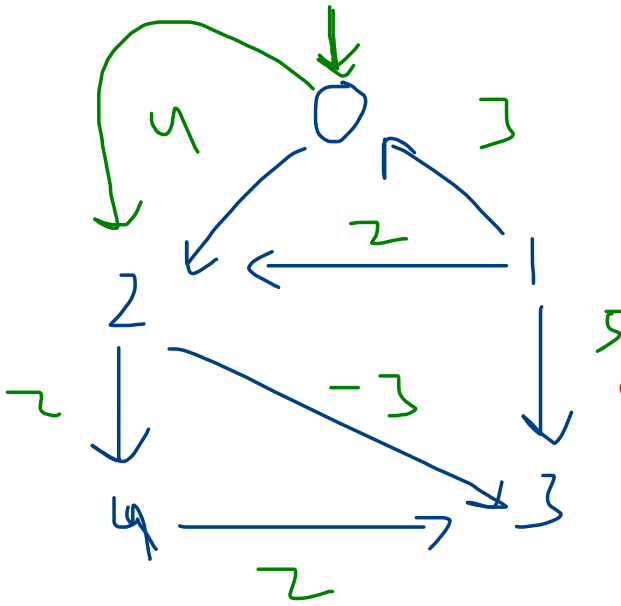
Shortest Path in DAG



0	2	3	4

src = 0

BFS




```
if( wt[u,v] + dist[u] < dist[v] )
{
  dist[v]
     = wt[u,v] + dist[u]
}
```

0	1	2	3	4
dist	0	~~1~~	~~4~~	~~1~~	~~6~~