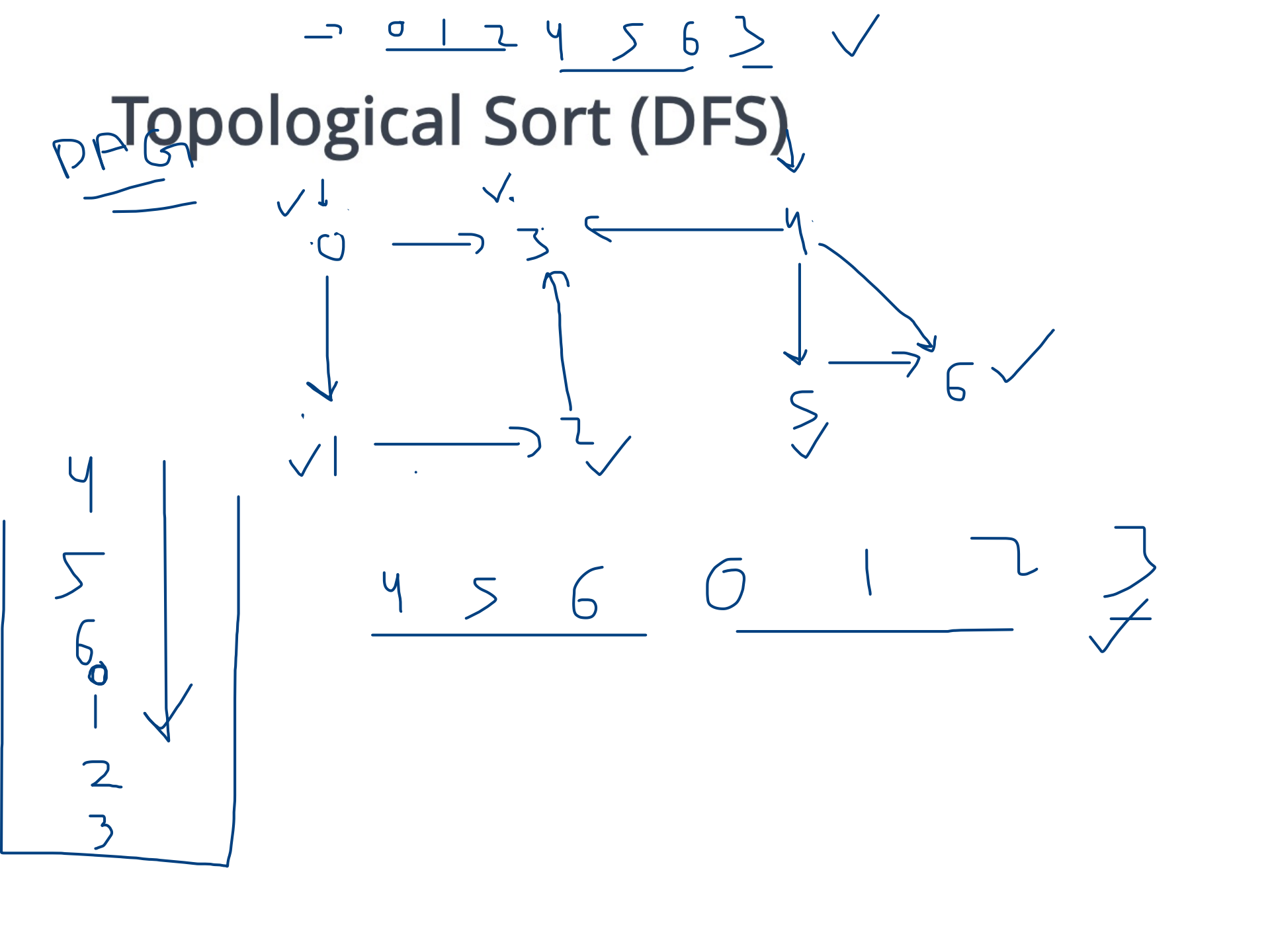
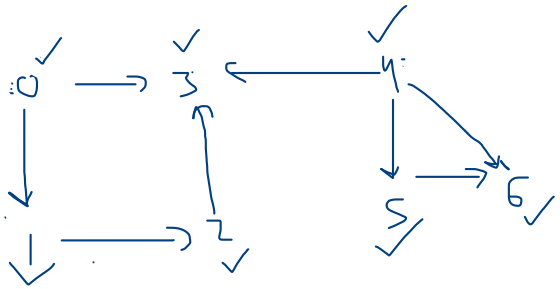
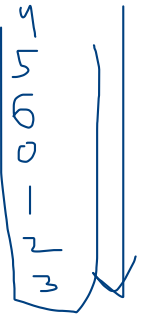
Topological Sort (DFS)

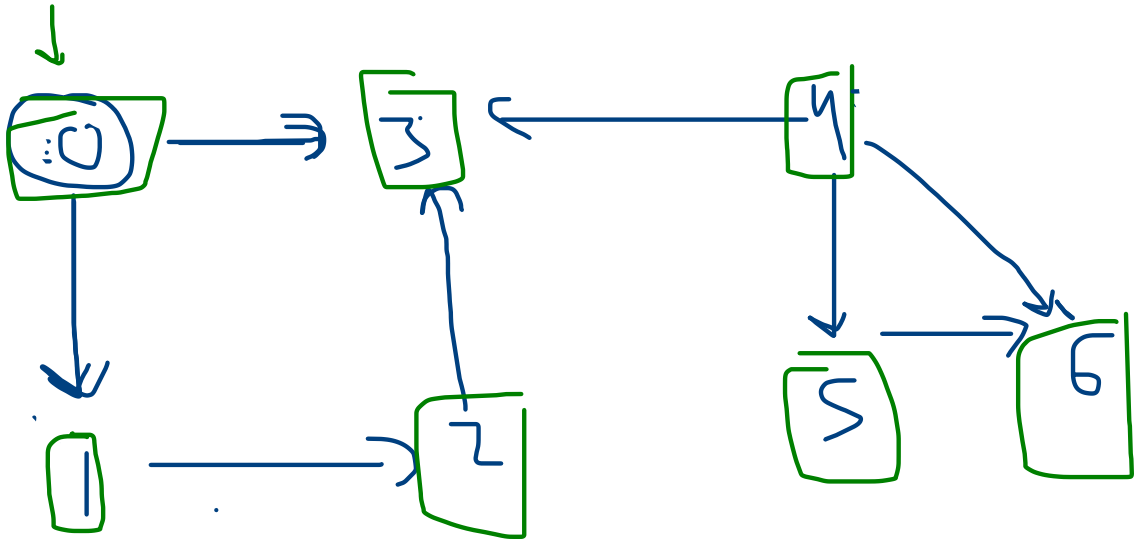






```java
public static void dfs(ArrayList<Edge>[] graph, int v) {

    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)

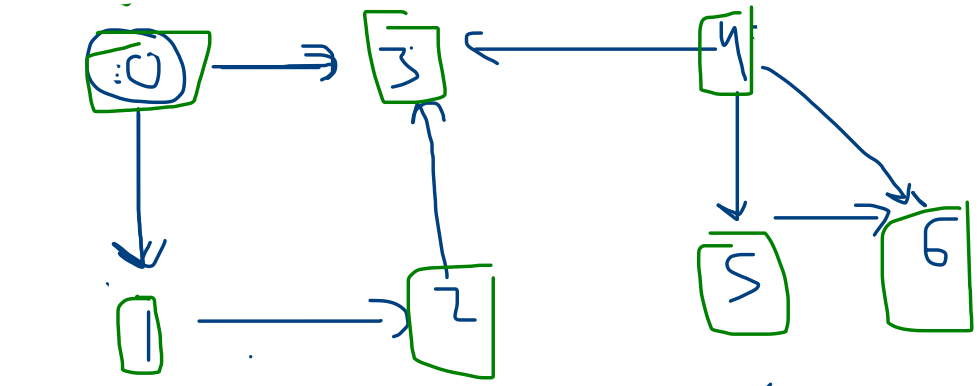


indegree =

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

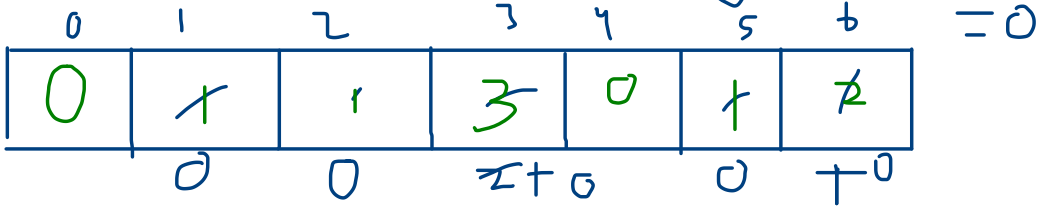
outdegree =

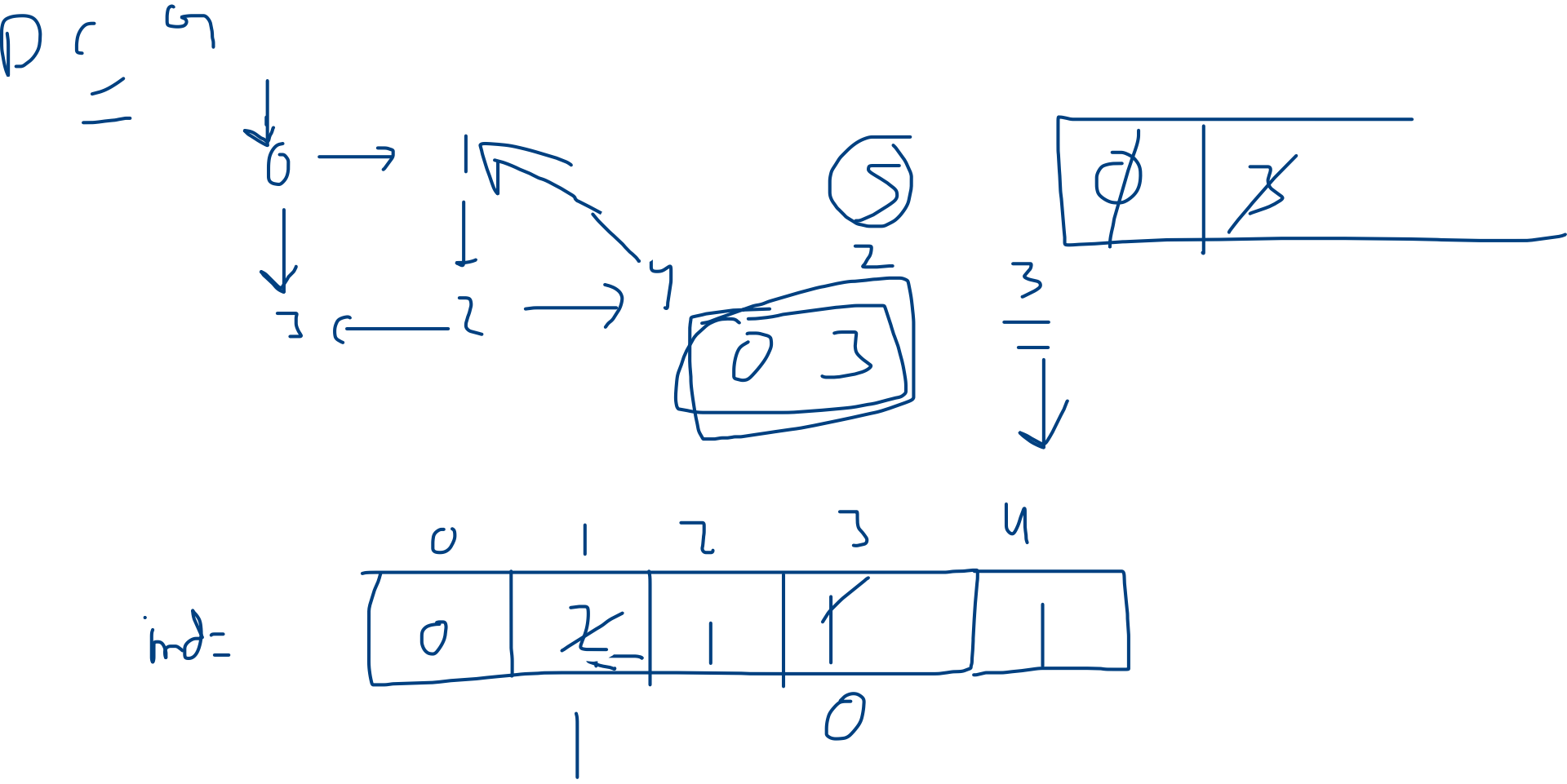
0	1	2	3	4	5	6
2	1	1	0	3	1	0



0 4 1 5 2 6 3

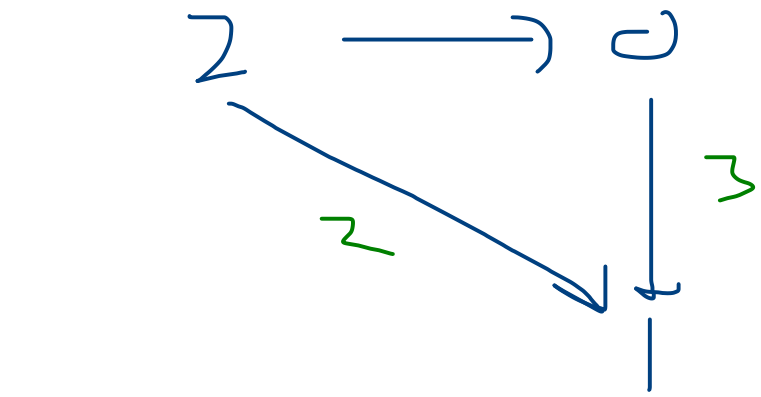
remove que
print
loop in nbr and decrement indegree
add vertex with indeg = 0 in que



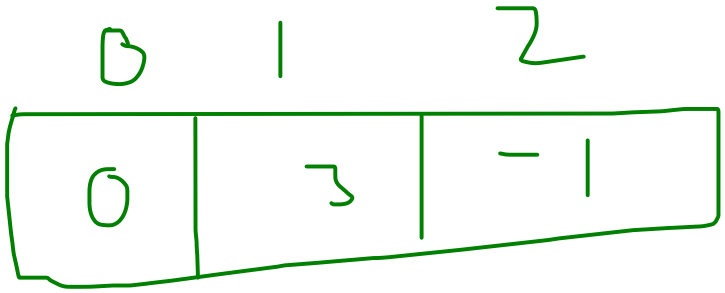

0 1 2 3 4
0 3
0 2
0 1 2 3 4
ind=
0 2 1 1 1
1 0

Course Schedule

Shortest Path in DAG

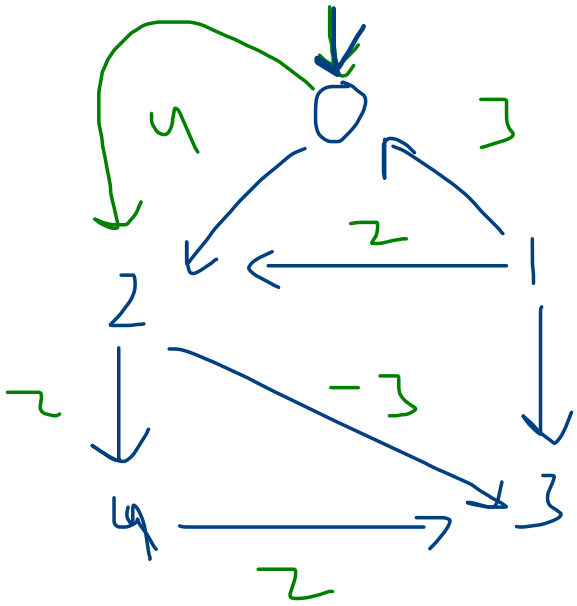


src = 0



Src : 0

BFS

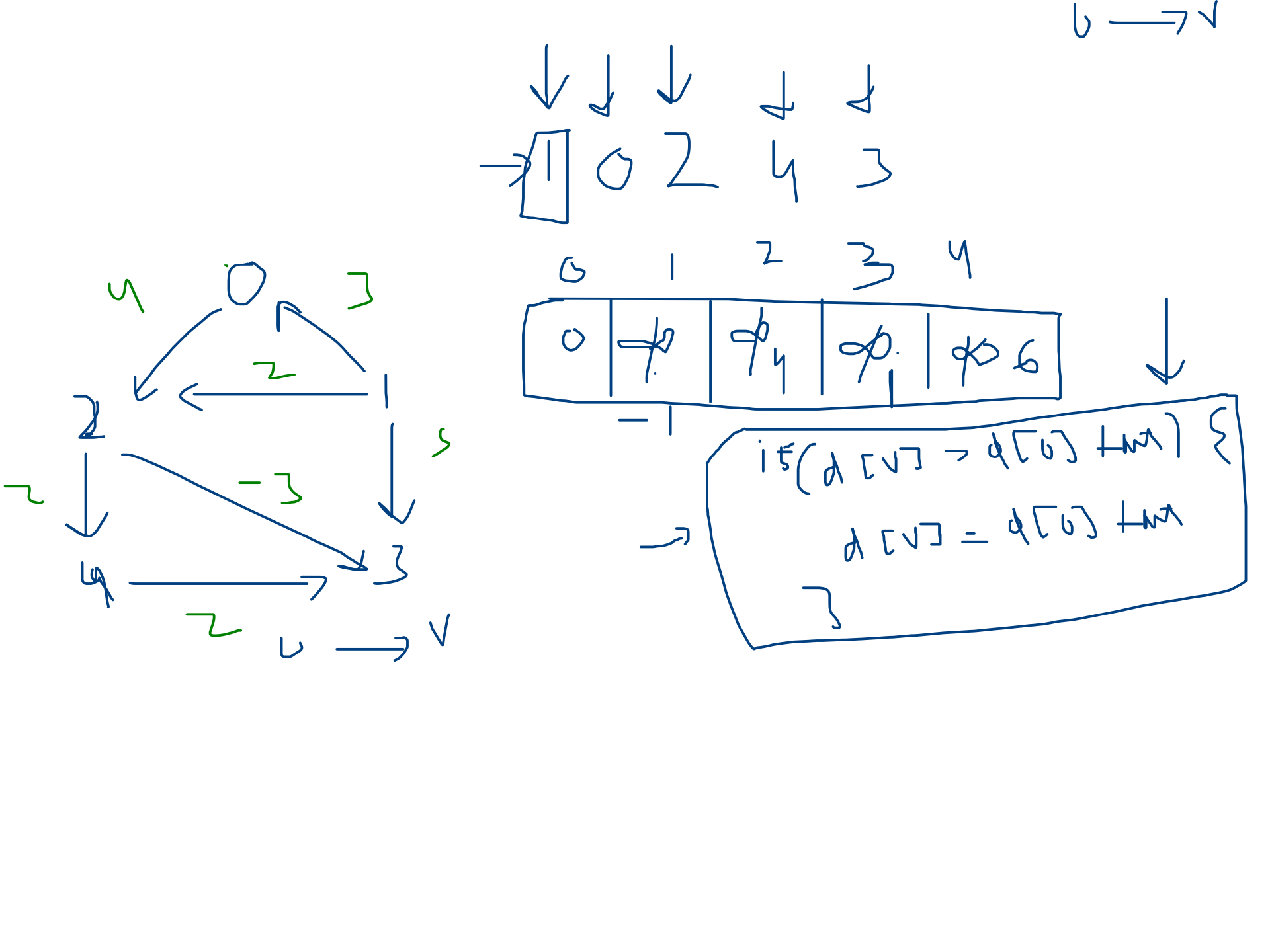


∅	2	3	4

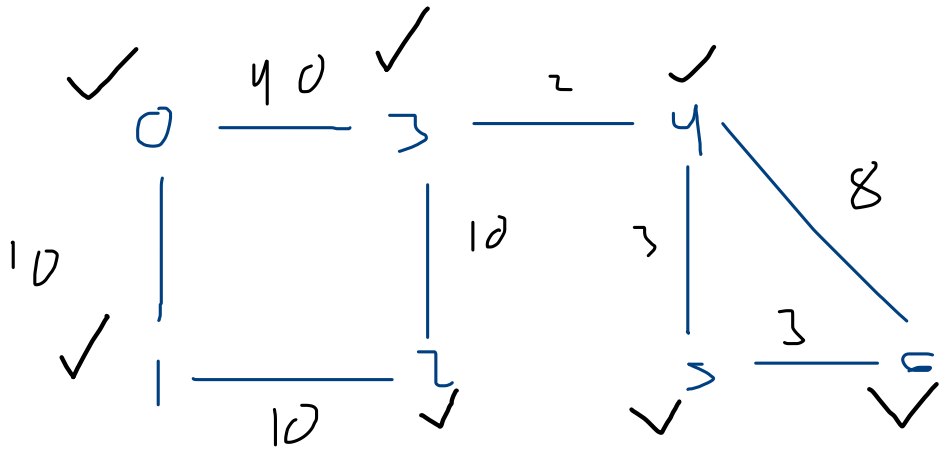
```
if( wt[u,v] + dist[u] < dist[v] )
{
  dist[v]
     = wt[u,v] + dist[u]
}
```

dist

0	1	2	3	4
0	~~∞~~ 1	~~∞~~ 4	~~∞~~ 1	~~∞~~ 6


0 1 2 3 4
0 ∞ ∞ ∞ ∞
-1
if(d[v] > d[u] + w
d[v] = d[u] + w
}

Dijkstra's shortest path

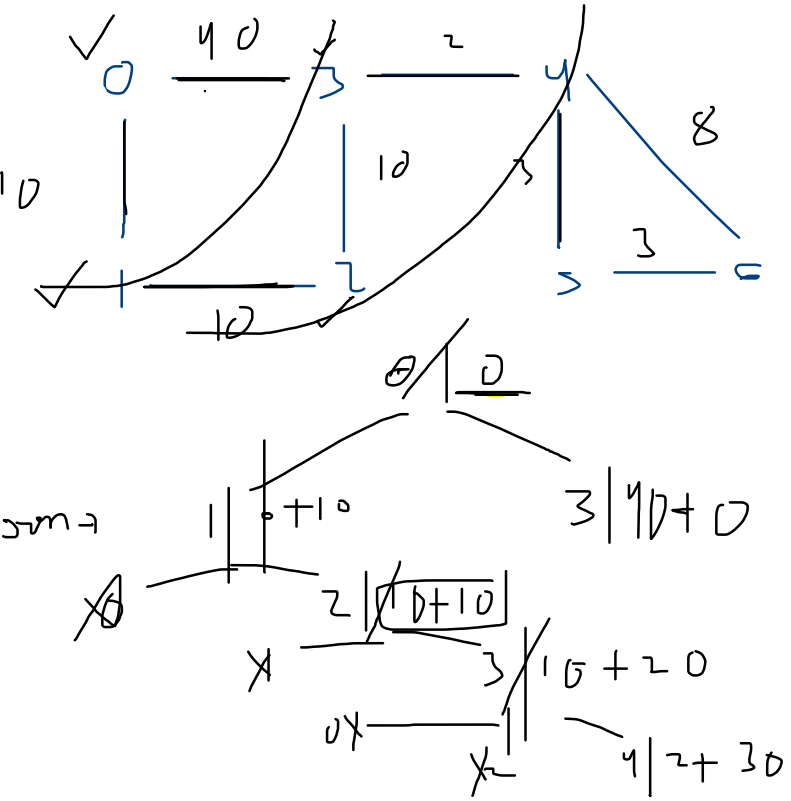


$V + E \log V$

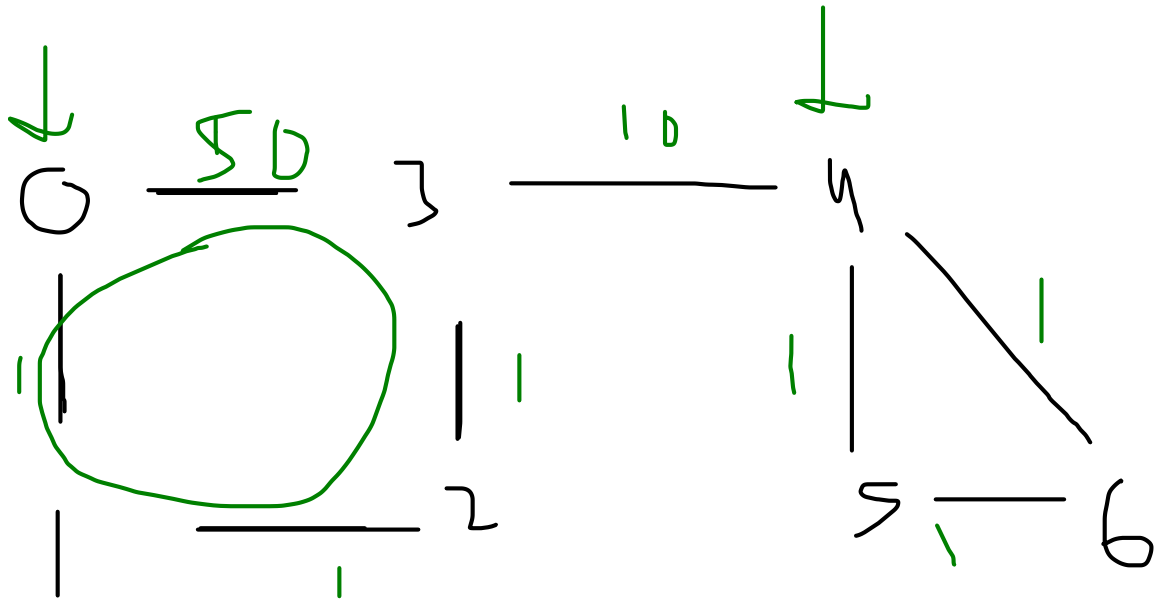
PQ

BFS

0	PSF	U SF
0	0	0
1	0-1	10
3	0-3	40
2	0-1-2	20
3	0-1-2-3	30
4	0-1-2-3-4	32
5	0-1-2-4-5	35
6	0-1-2-3-6	40
6	0-1-2-4-5-6	38

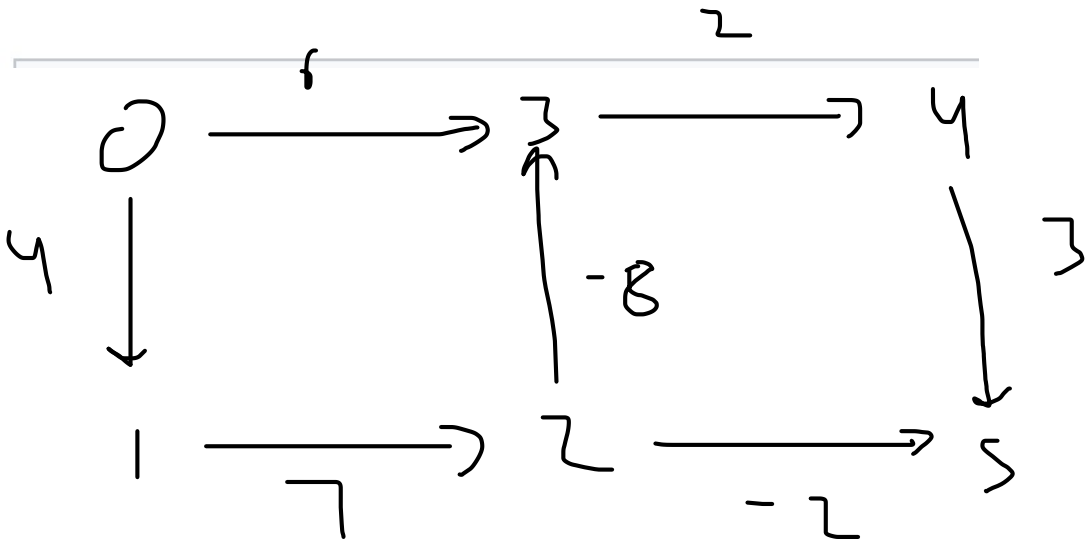


```
pq.add(new Triplet(src, 0));

boolean[] visited = new boolean[v];

int[] ans = new int[v];
while(pq.size() > 0) {

    Triplet rem = pq.remove();

    if (visited[rem.vtx]) {
        continue;
    }

    visited[rem.vtx] = true;
    ans[rem.vtx] = rem.cost;

    for (Edge e: graph[rem.vtx]) {
        if (!visited[e.nbr]) {
            Triplet temp = new Triplet(e.nbr, e.wt + rem.cost);
            pq.add(temp);
        }
    }

}

return ans;
```

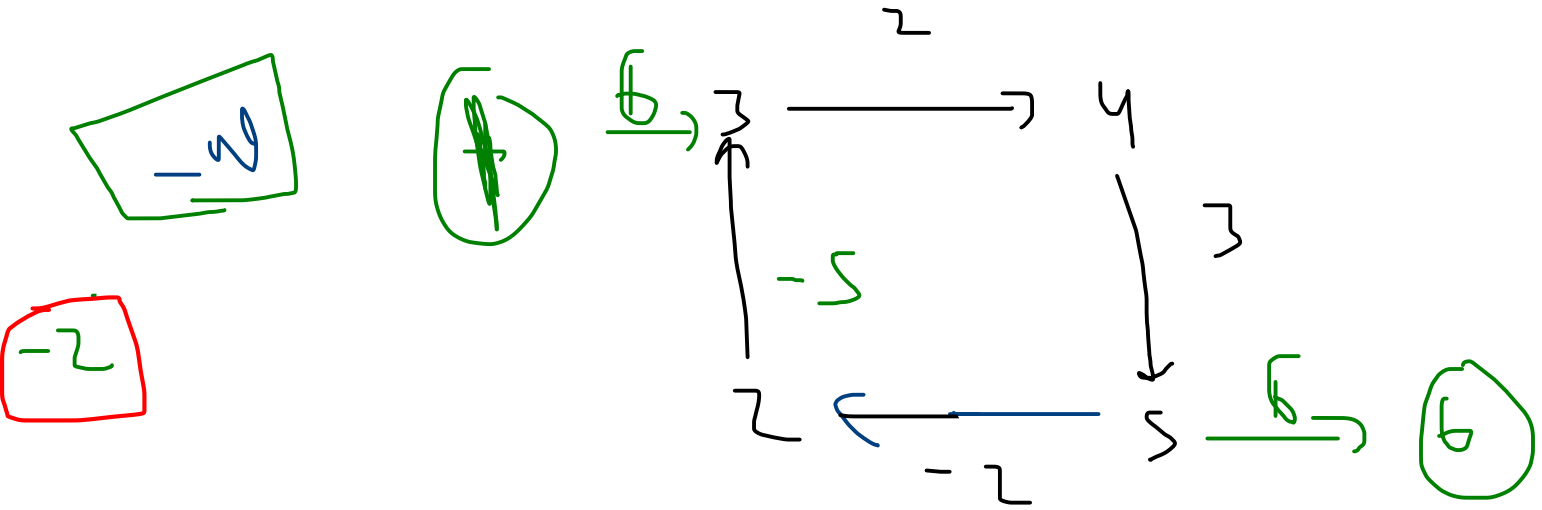


we will never get shorted distance b/w two vertices by looping into cycle

Bellman Ford



negative weight cycle is never possible

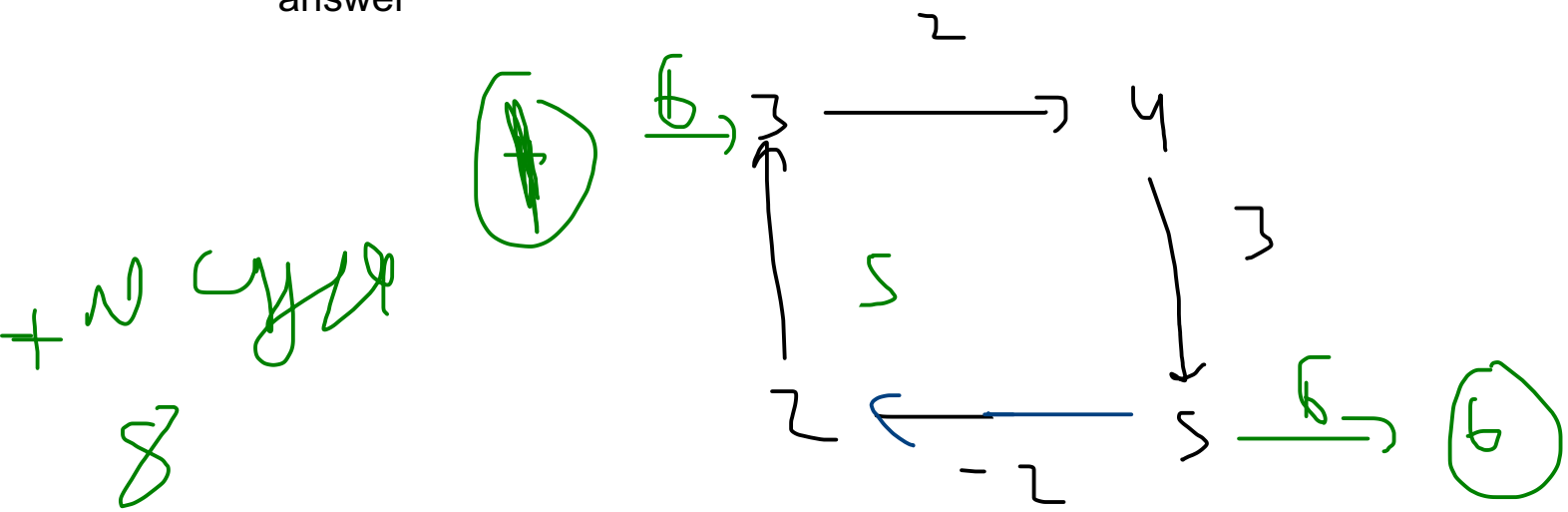


$6 + 2 + 3 + 6 = 17$

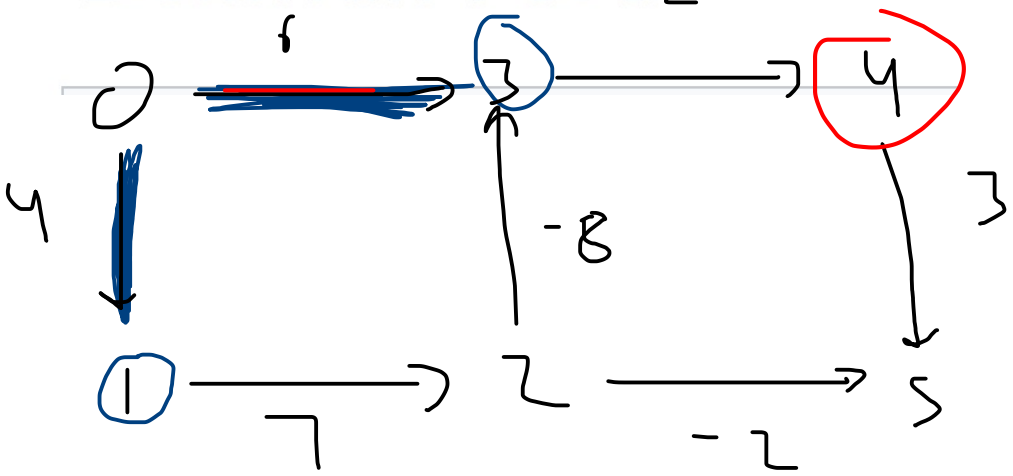
$6 + (-2) + 2 + 3 + 6 = 15$

$6 + (-2) \times 2 + 2 + 3 + 6 = 13$

for positive wt cycle you will never going to complete one cycle to get the answer



Bellman Ford



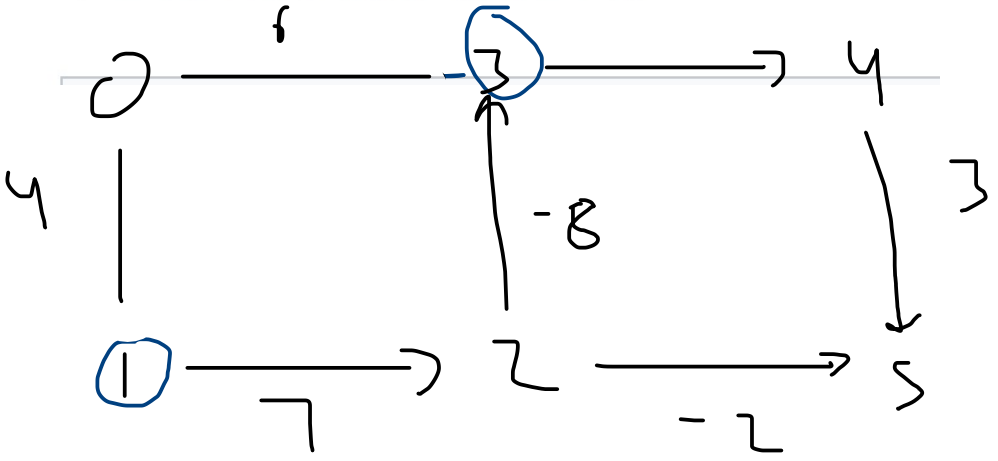
u	v	w
Random → 4	5	3
→ 3	4	2
→ 2	3	-8
→ 1	2	7
→ 0	3	6
→ 0	1	4

0	1	2	3	4	5
dist	0	~~∞~~ 4	∞	~~∞~~ 6	∞	∞

v - 1 iterations

6 - 1 = 5

Bellman Ford



u → v	w
→ 4 5	3
→ 3 4	7
→ 2 3	-8
→ 1 2	7
→ 0 3	6
→ 0 1	4

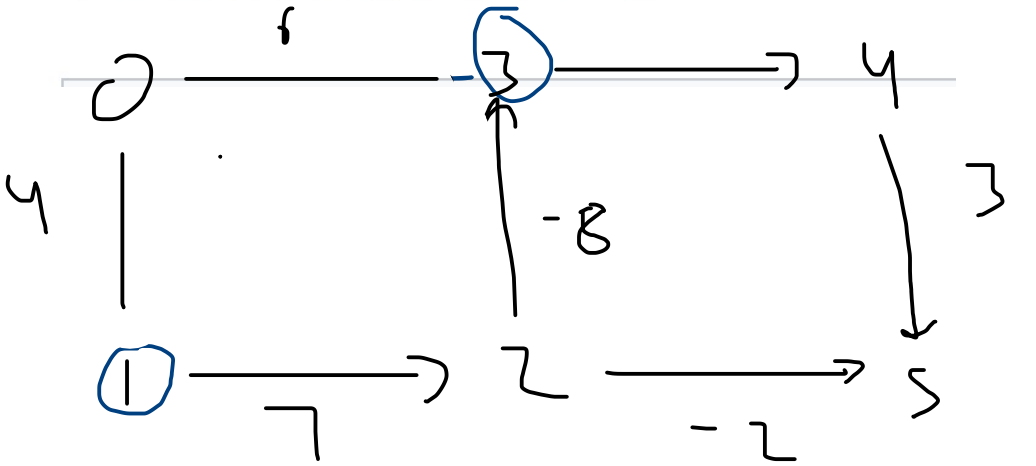
dist

0	1	2	3	4	5
0	~~∞~~ 4	~~∞~~ 11	~~∞~~ 6	~~∞~~ 8	∞

v - 1 iterations

6 - 1 = 5

Bellman Ford



dist[v] > dist[u] + wt(u,v)

u → v	w
4 → 5	3
3 → 4	2
2 → 3	-8
1 → 2	7
0 → 3	6
0 → 1	4

dist

0	1	2	3	4	5
0	~~∞~~ 4	~~∞~~ 11	~~∞~~ 6 3	~~∞~~ 8 5	∞ 11 8

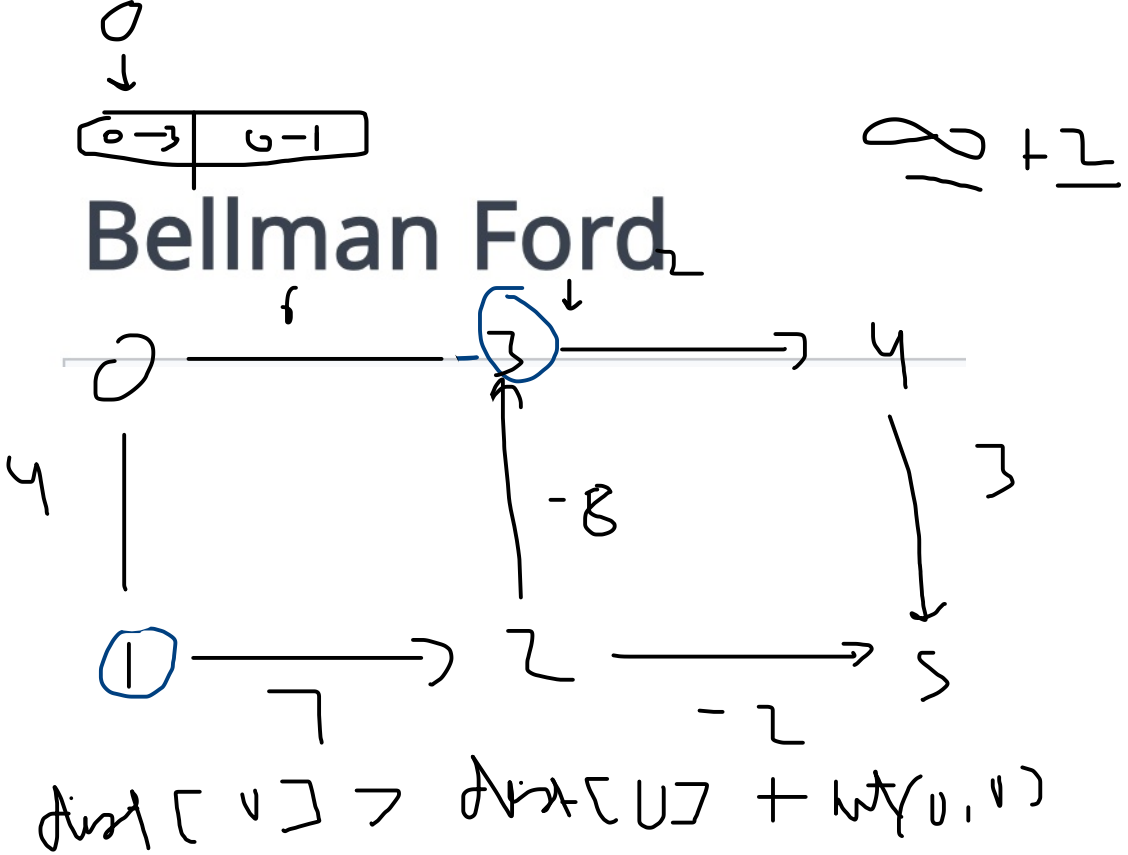
v - 1 iterations

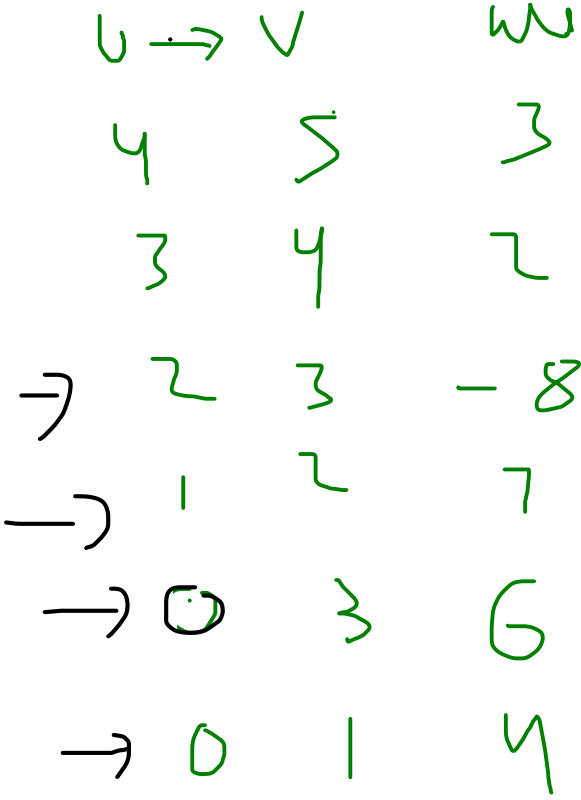
6 - 1 = 5

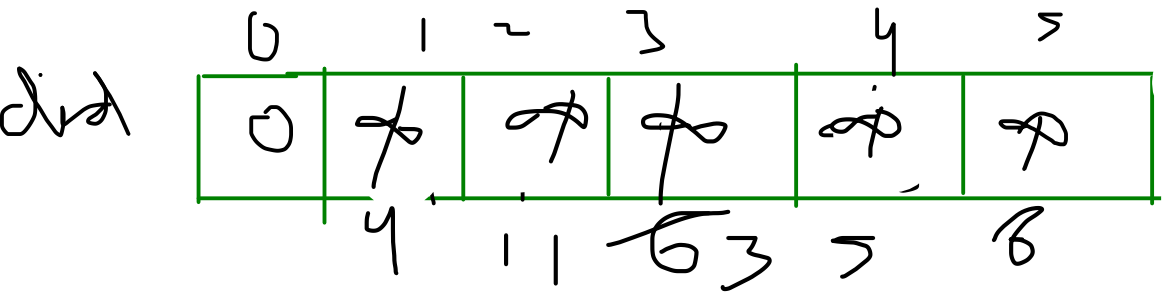
ans of a particular vertice is made in ith iteration whose path is also i edge path away

ith edge path ans ----> in <= ith iteration

Bellman Ford







v - 1 iterations

$V - 1$

Acyclic : V V to r

$V-1$ edge

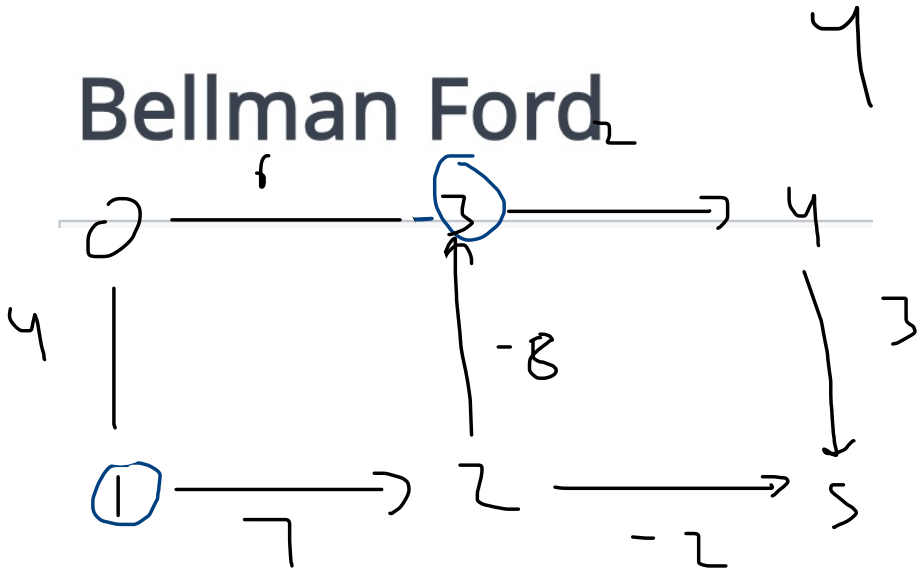
cyclic

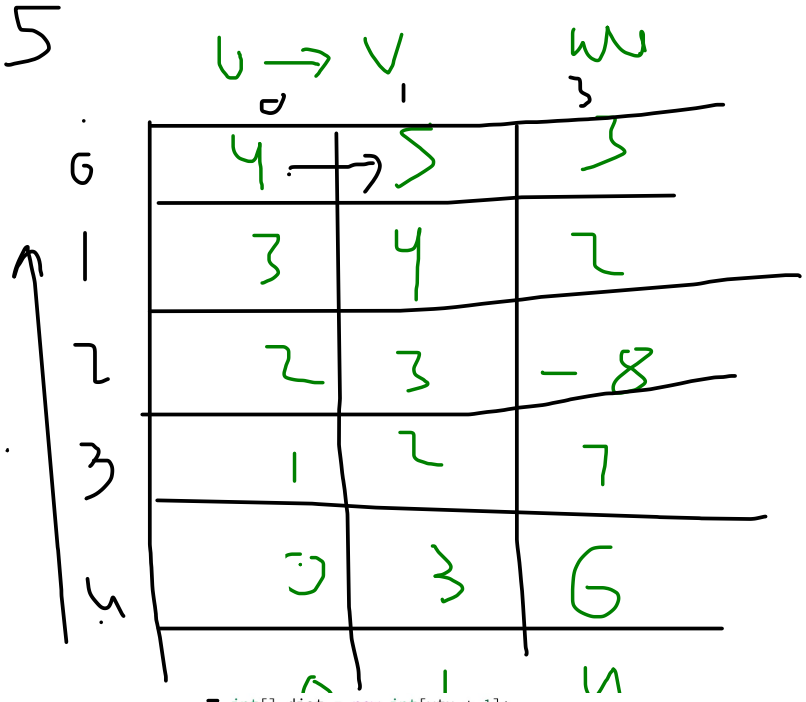
$6 = V$

$= 5 = V - 1$

+ve

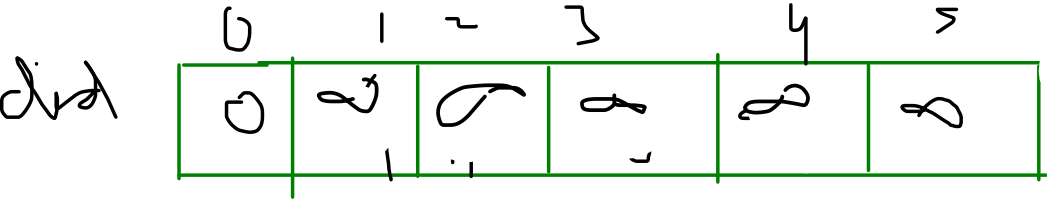
Bellman Ford





dist[v] > dist[u] + wt(u,v)

∞ + 3 = ∞

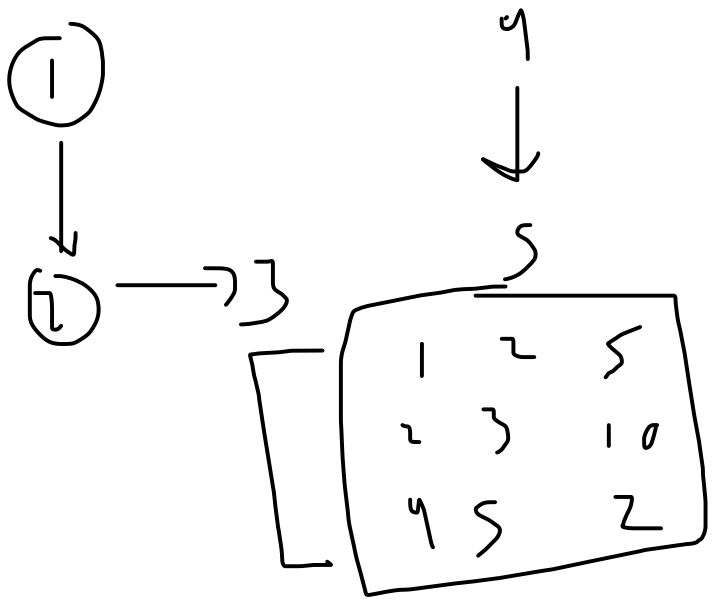


v - 1 ite

```
int[] dist = new int[vtx + 1];
Arrays.fill(dist, Integer.MAX_VALUE);
dist[src] = 0;


for (int i = 1; i < vtx; i++) {
    for (int j = 0; j < arr.length; j++) {

            int u = arr[j][0];
            int v = arr[j][1];
            int wt = arr[j][2];

            if (dist[u] == Integer.MAX_VALUE) {
                continue;
            }

            if (dist[v] > dist[u] + wt) {
                dist[v] = dist[u] + wt;
            }


    }
}

for (int i = 1; i < vtx + 1; i++) {
    if (dist[i] == Integer.MAX_VALUE) {
        System.out.print(M + " ");
    } else {
        System.out.print(dist[i] + " ");
    }
}
```

Time Comp of Bellman Ford

```
for (int i = 1; i < vtx; i++) {
    for (int j = 0; j < arr.length; j++) {

        int u = arr[j][0];
        int v = arr[j][1];
        int wt = arr[j][2];

        if (dist[u] == Integer.MAX_VALUE) {
            continue;
        }

        if (dist[v] > dist[u] + wt) {
            dist[v] = dist[u] + wt;
        }
    }
}
```



$T(n) = V \times E$ → worst Best Comments

$S(n) = O(V)$

Shortest Path In A Binary Maze