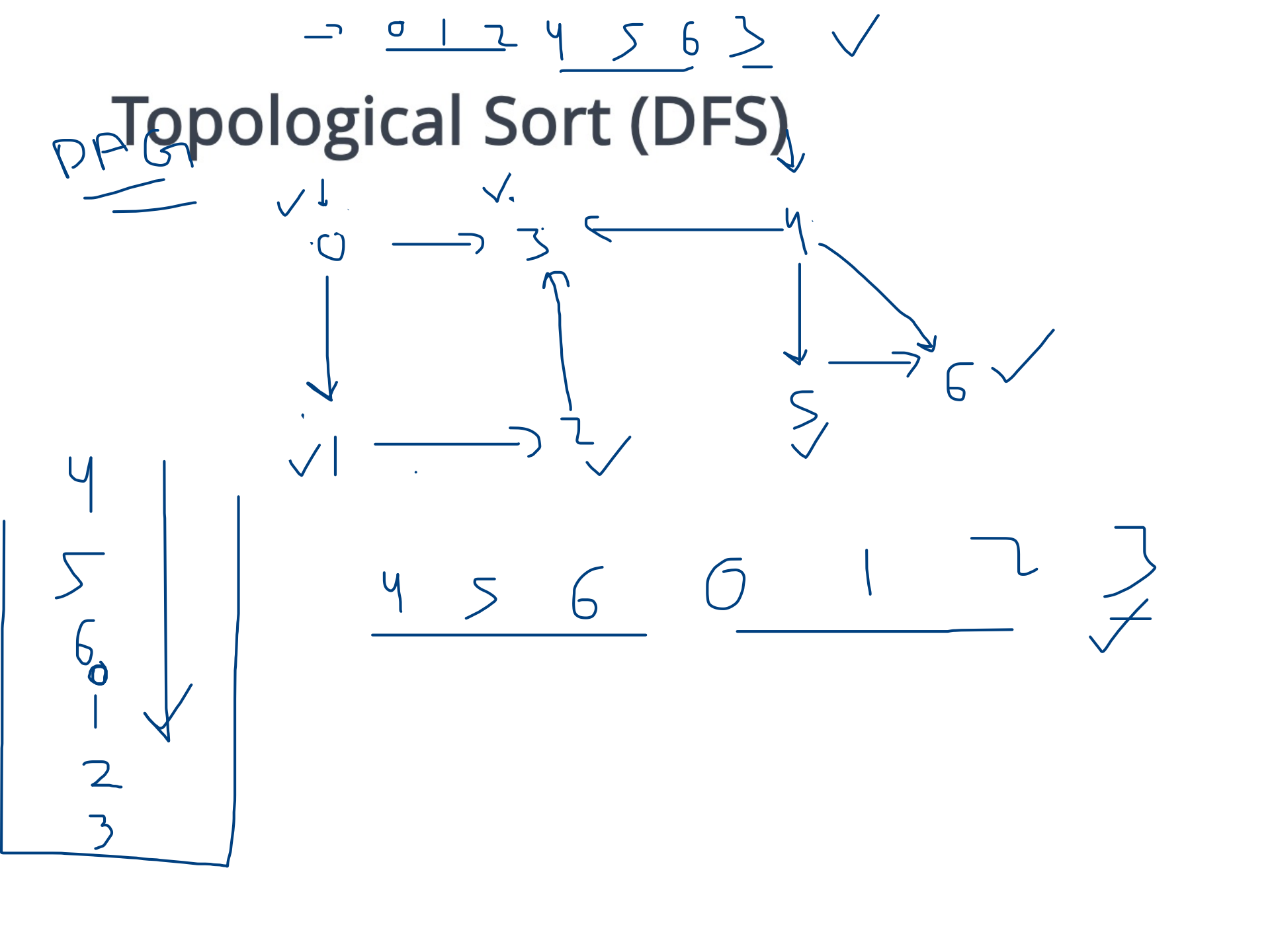
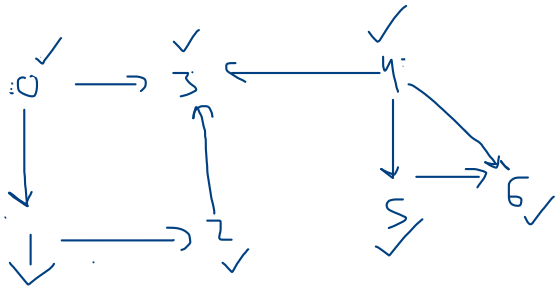
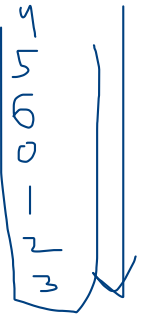
Topological Sort (DFS)

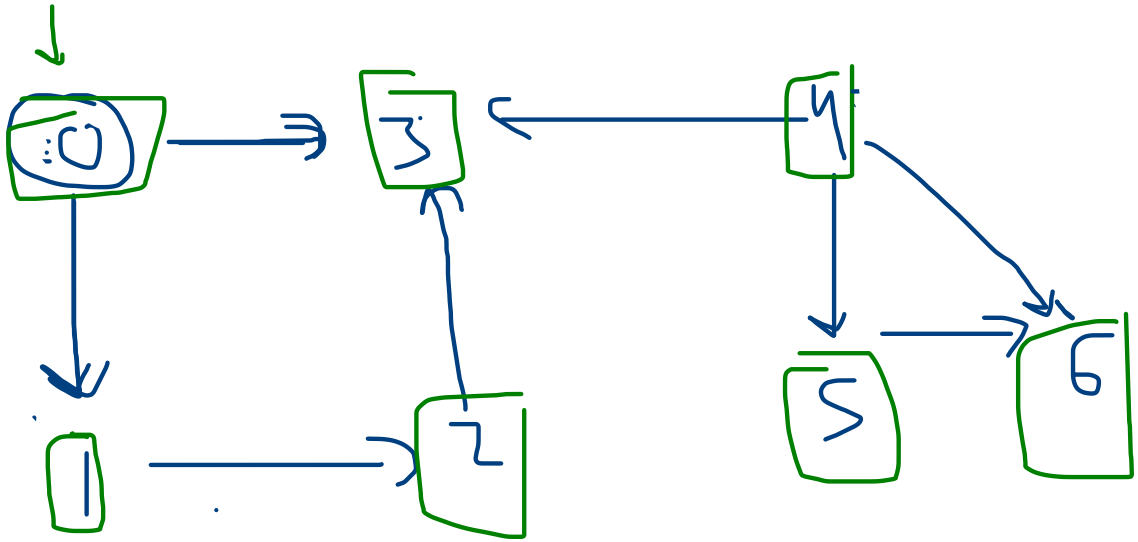






```java
public static void dfs(ArrayList<Edge>[] graph, int v) {

    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)

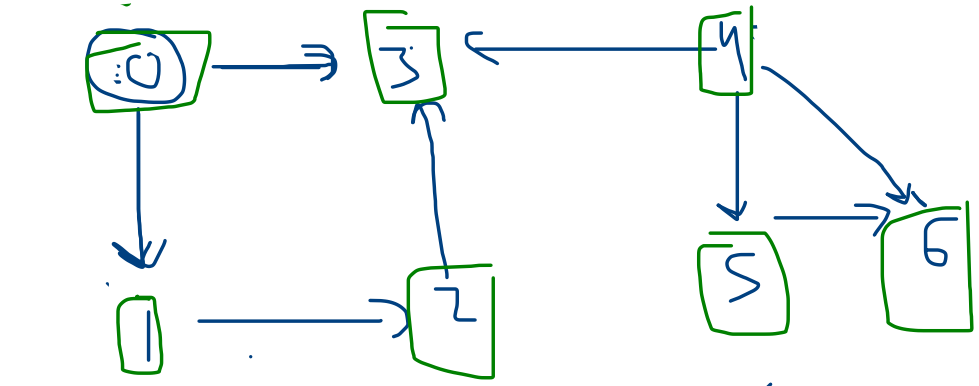


indegree =

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

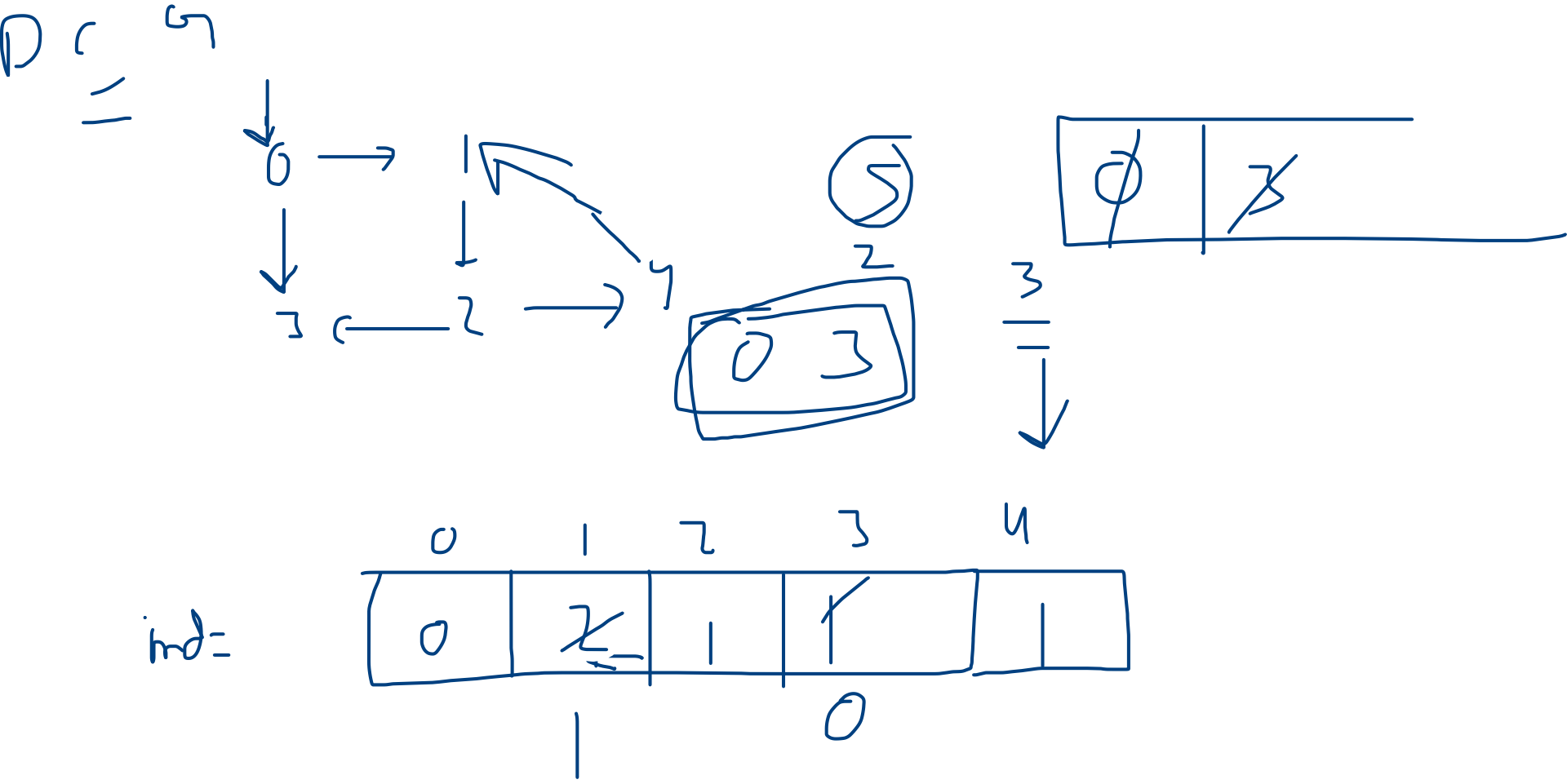


0 4 1 5 2 6 3

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

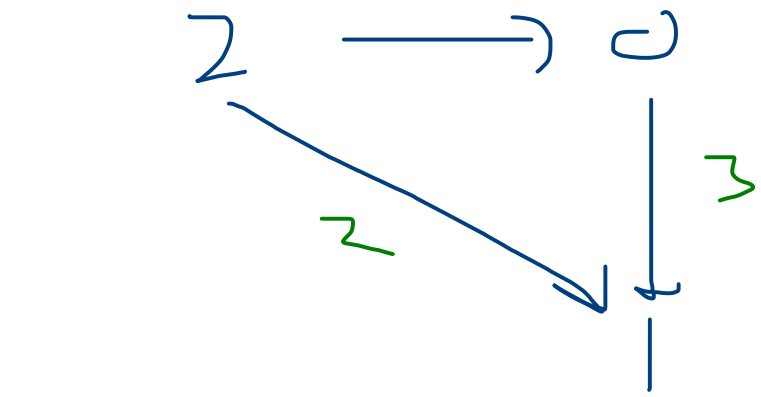
indegree =

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

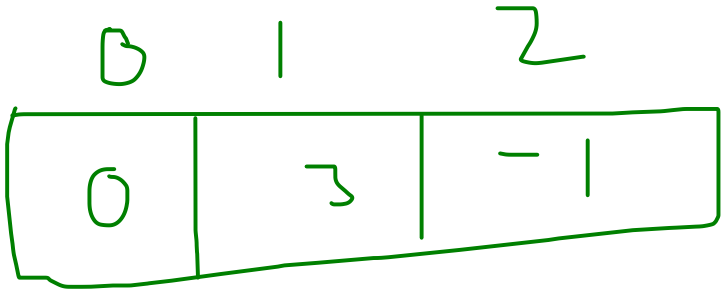

0 1 2 3 4
0 3
0 2
0 1 2 3 4
ind=
0 2 1 1 1
1 0

Course Schedule

Shortest Path in DAG



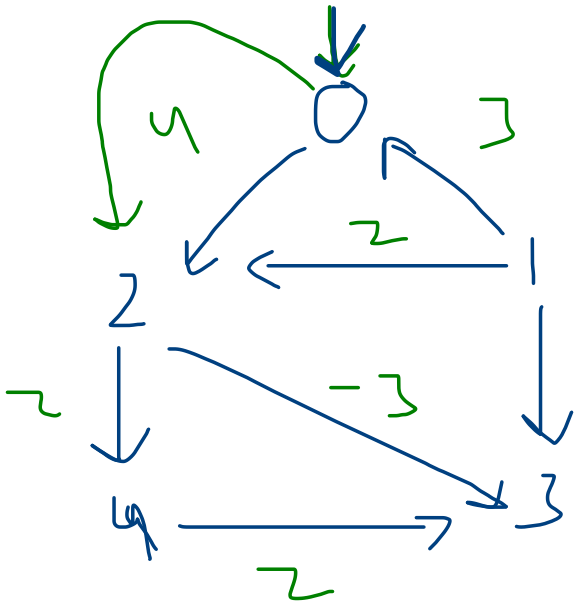
src = 0



0	2	3	4

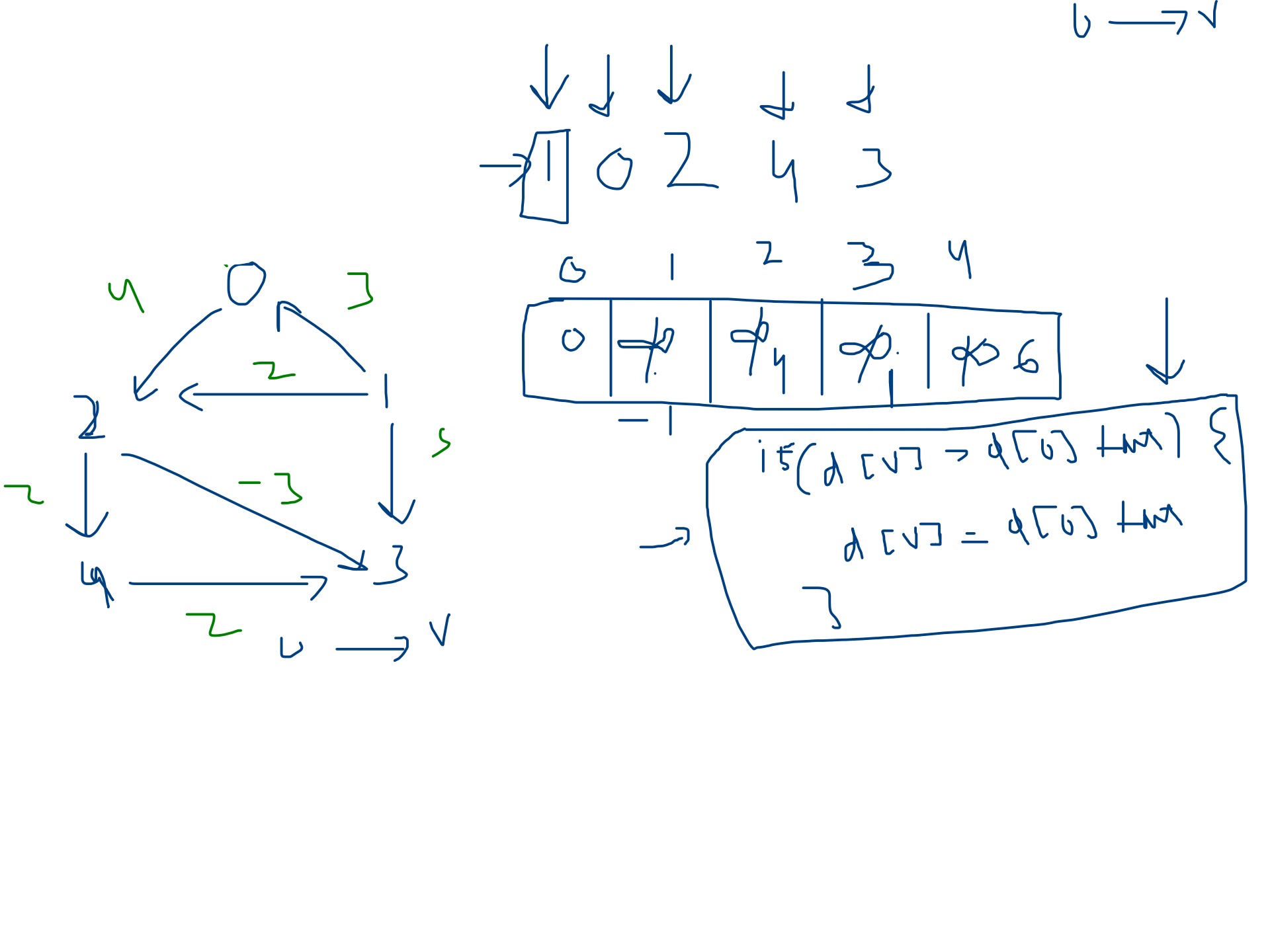
src = 0

BFS




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

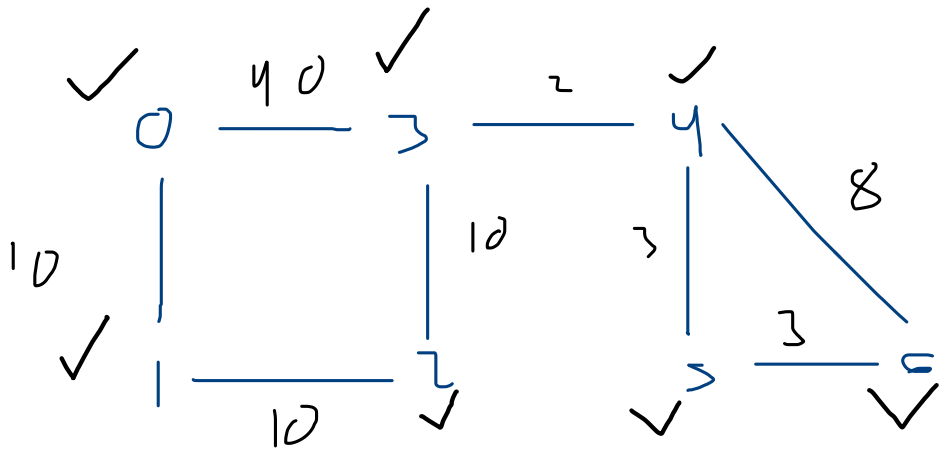
0	1	2	3	4
dist	0	~~1~~ 1	~~4~~ 4	~~1~~ 1	~~6~~ 6


if(d[v] > d[u] + w)
d[v] = d[u] + w

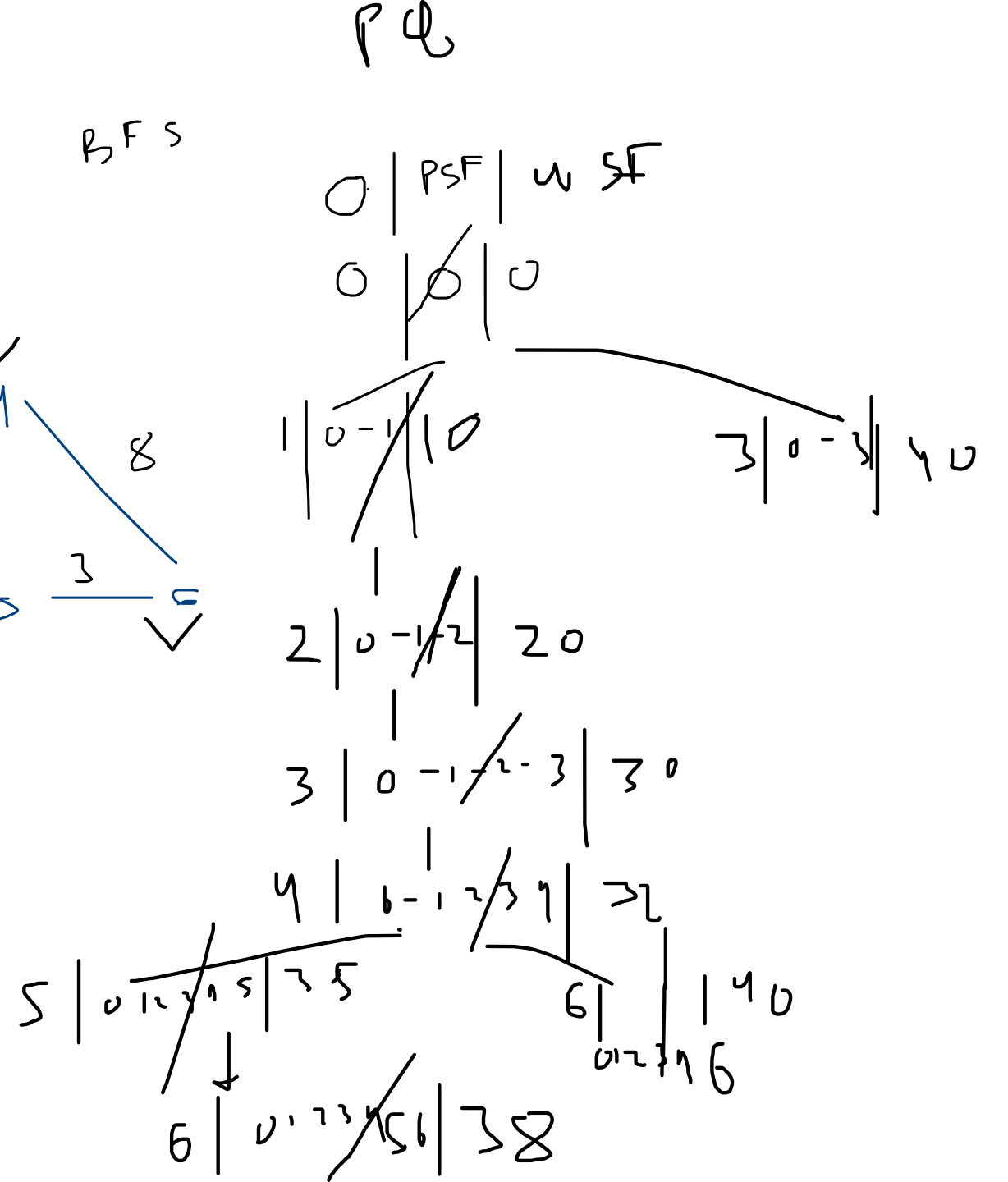
PQ

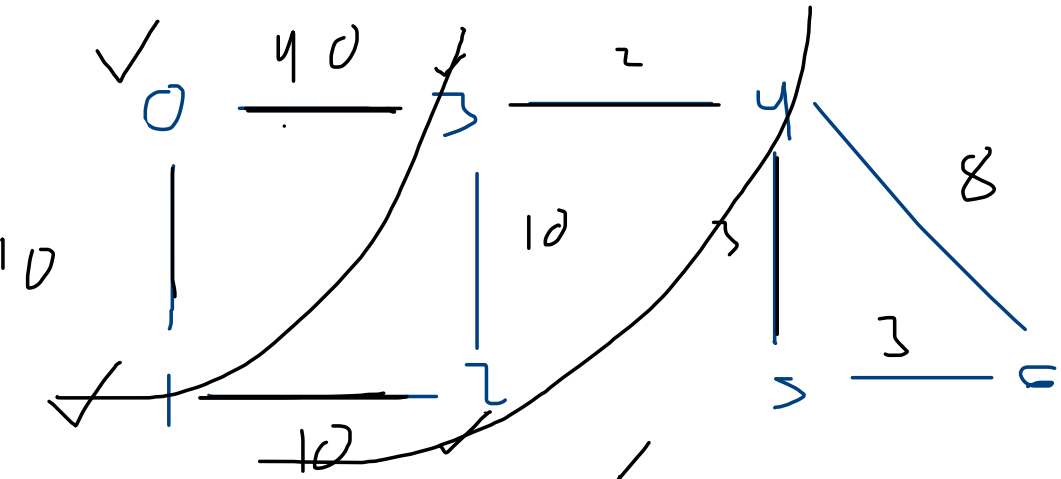
Dijkstra's shortest path

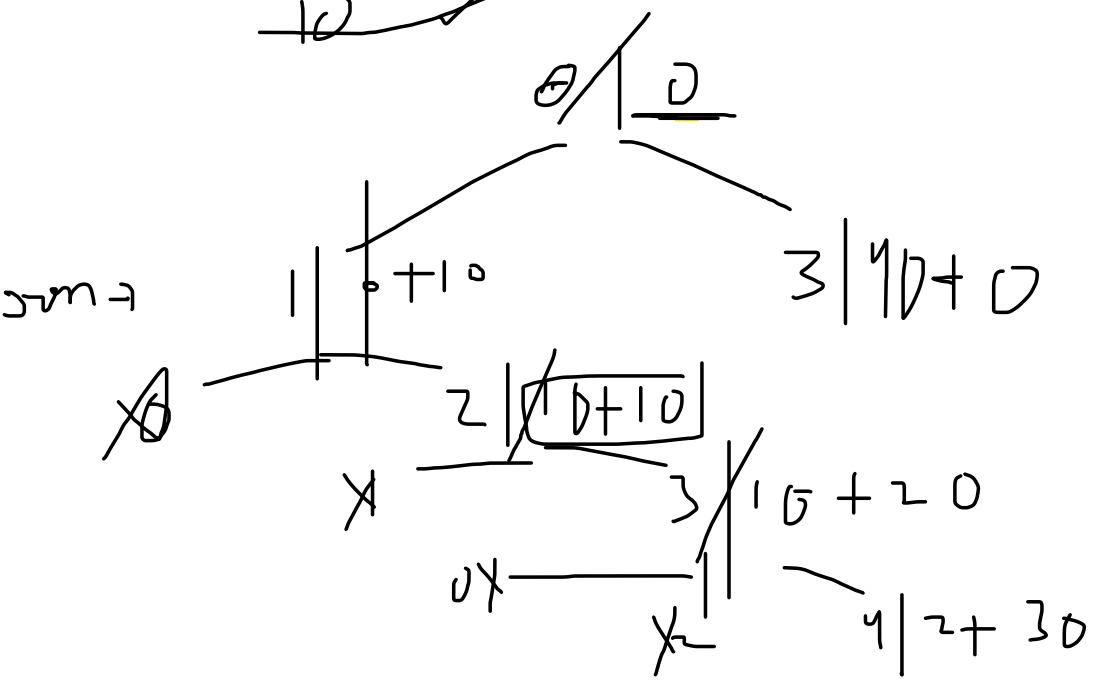
BFS



$V + E \log V$

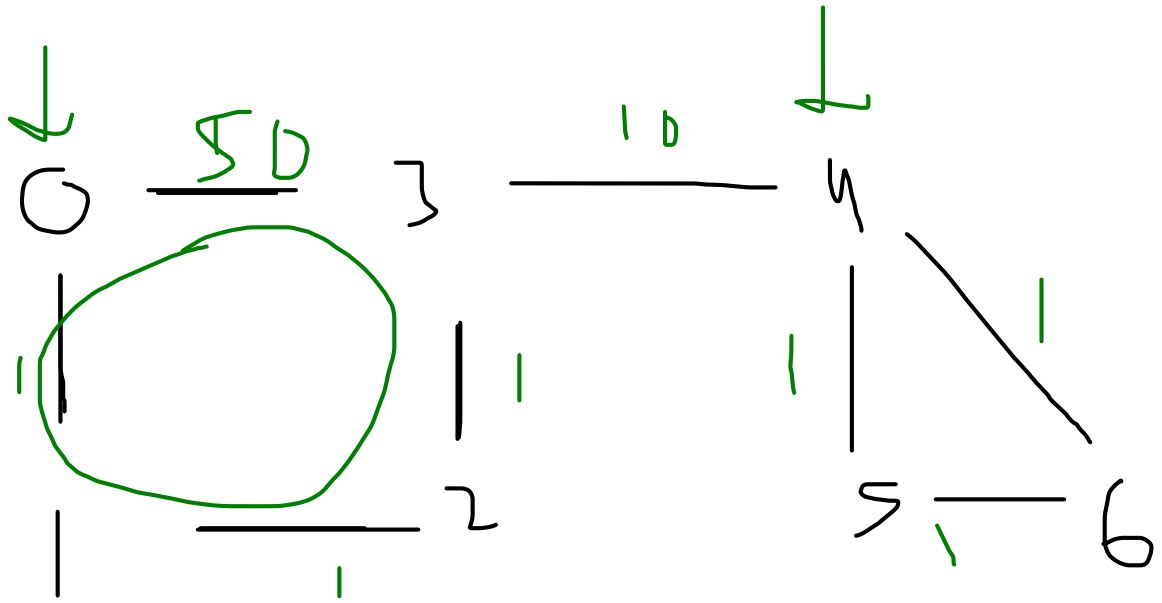






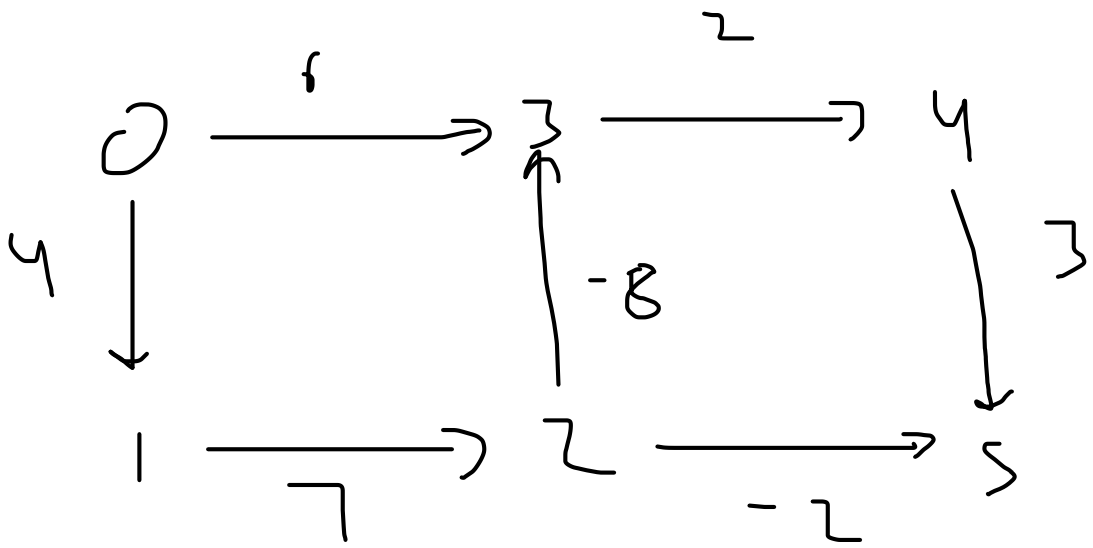
```
pq.add(new Triplet(src, 0));

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;
```

0	1	2	3
0	10	20	30

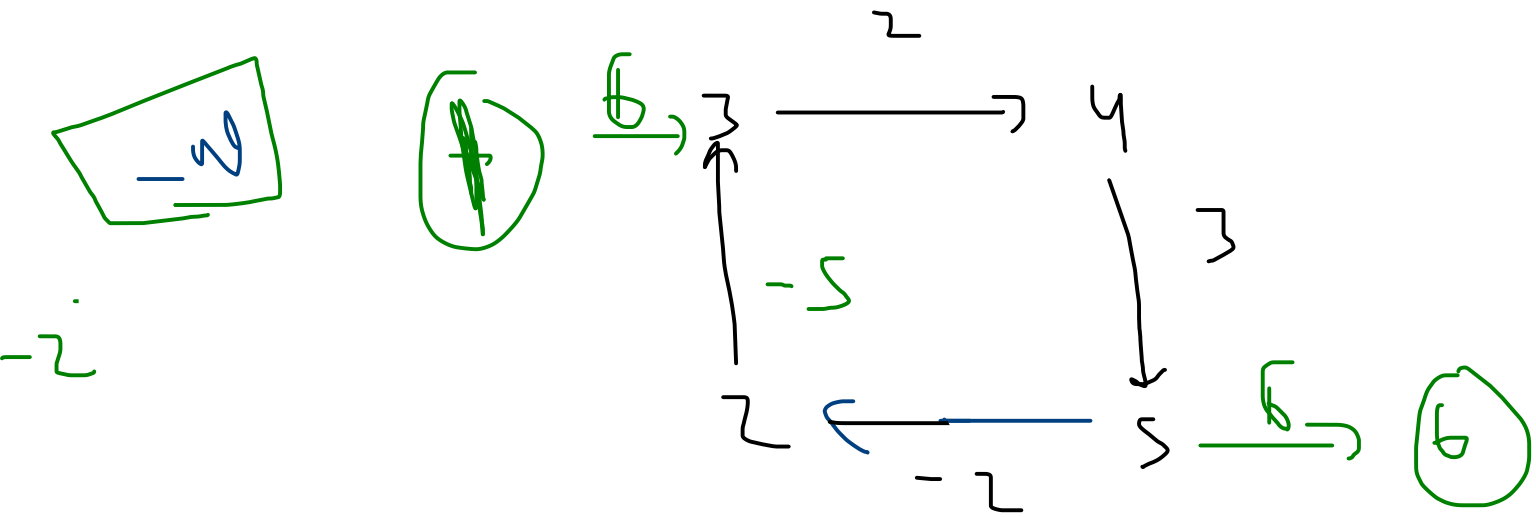


we will never get shorted distance b/w two vertices by looping into cycle

Bellman Ford



negative weight cycle is never possible

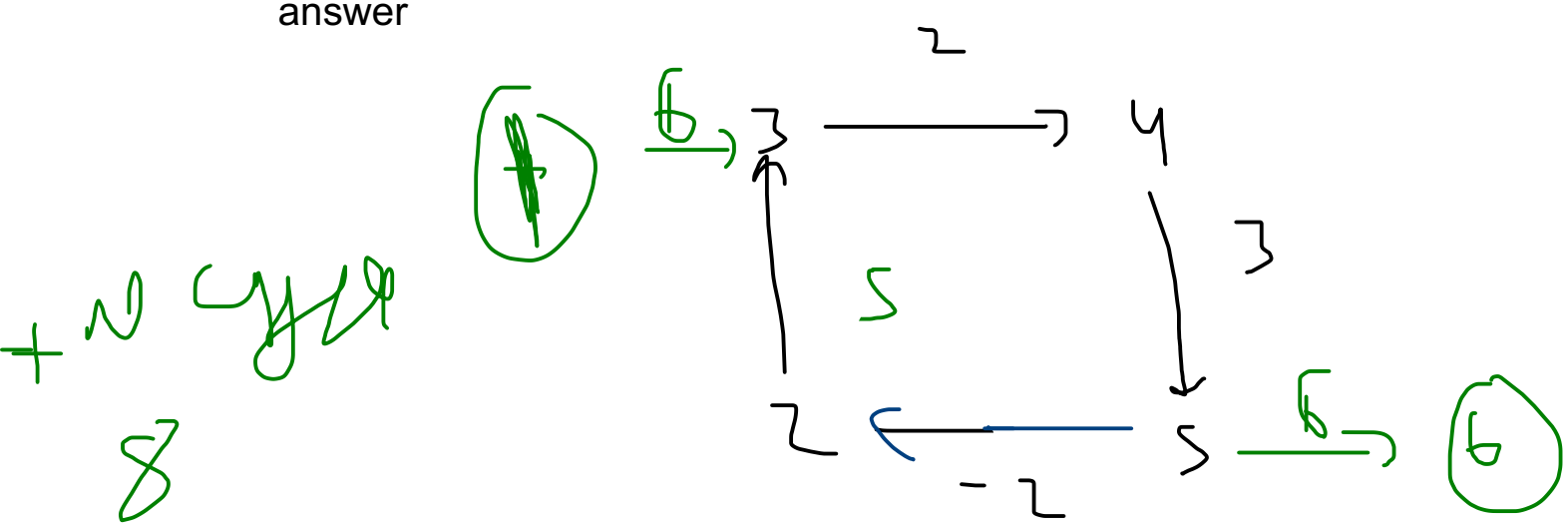


$6 + 2 + 3 + 6 = 17$

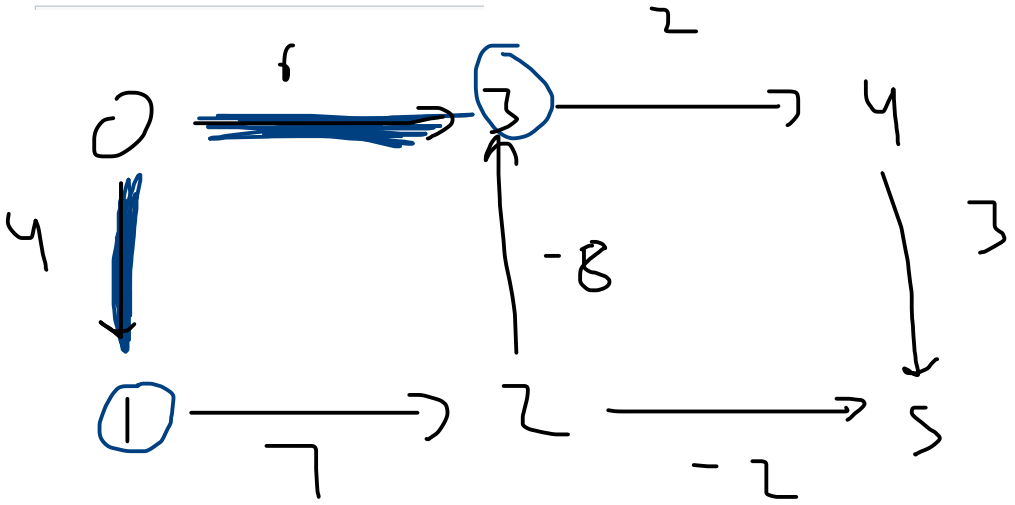
$6 + (-2) + 2 + 3 + 6 = 15$

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

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



Bellman Ford



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

u → v w

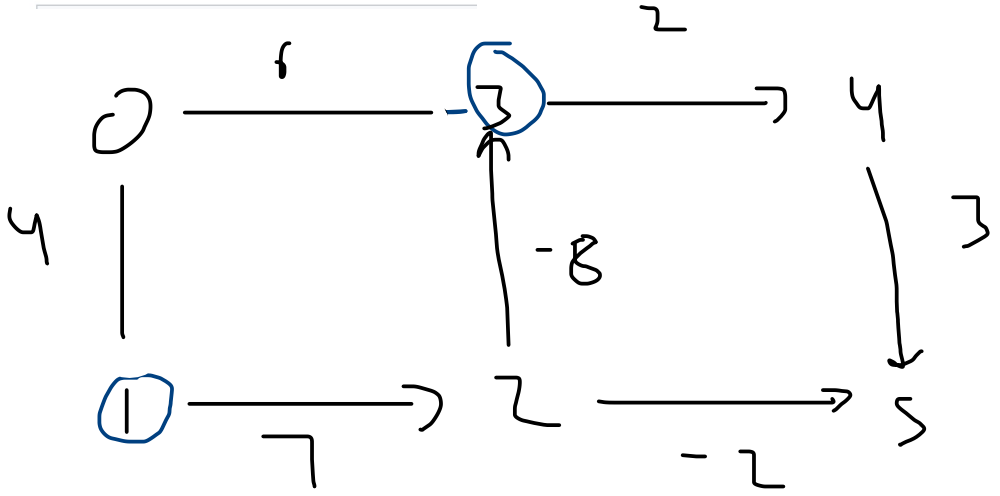
Random →
4 5 3
→ 3 4 2
→ 2 3 -8
→ 1 2 7
→ 0 3 6
→ 0 1 4

dist

0	1	2	3	4	5
0	~~∞~~ 4	∞	~~∞~~ 6	∞	∞

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

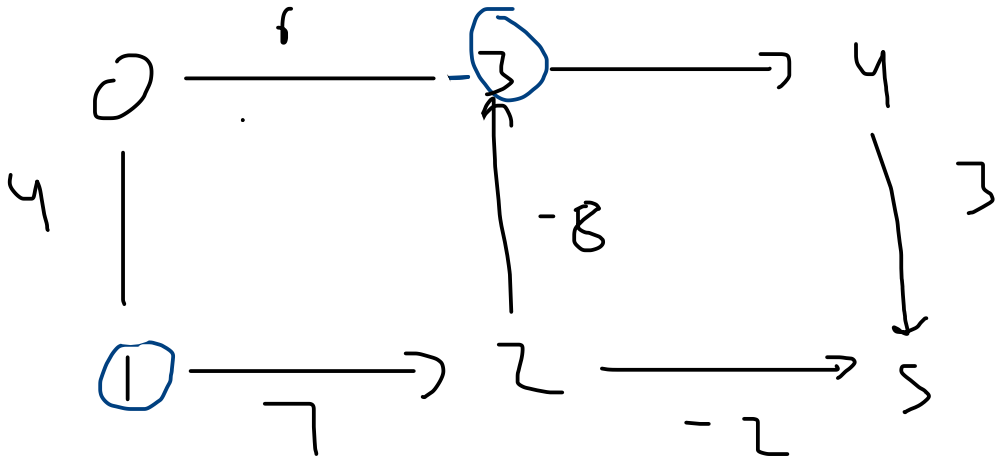
0	1	2	3	4	5
0	~~∞~~ 4	~~∞~~ 11	~~∞~~ 6	~~∞~~ 8	∞

v - 1 iterations

6 - 1 = 5

Bellman Ford

ith edge path shortest distance we will get in <= ith iteration



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	~~∞~~ 5 8	~~∞~~ 8 11

v - 1 iterations

6 - 1 = 5