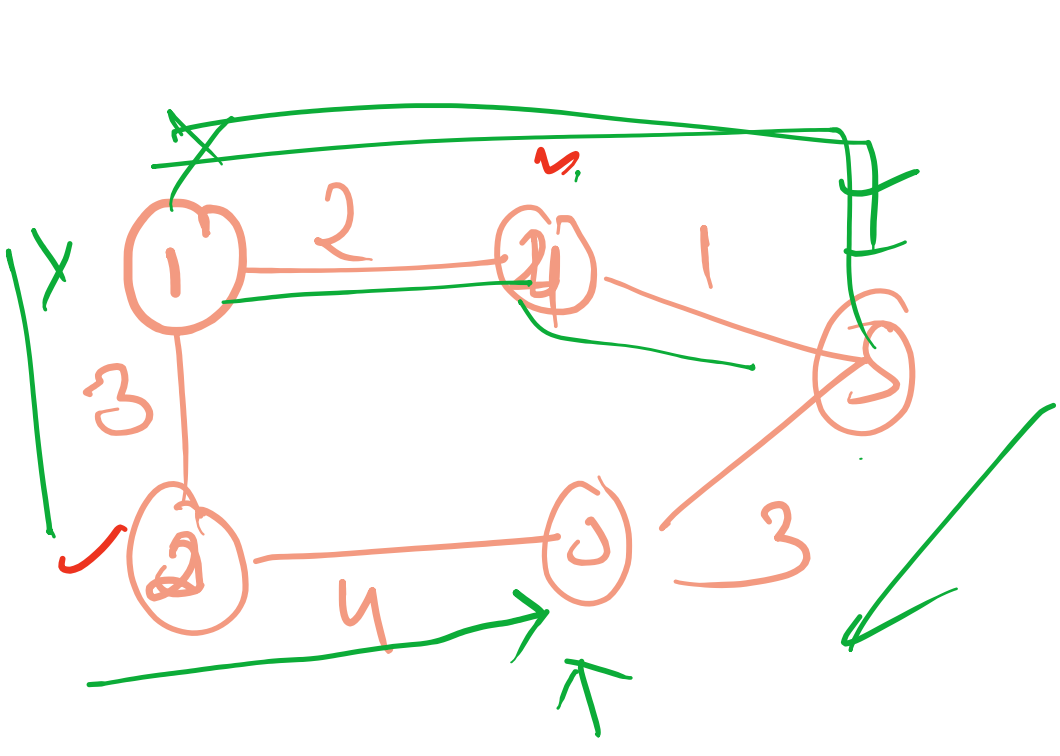


1 4 2  
3 3

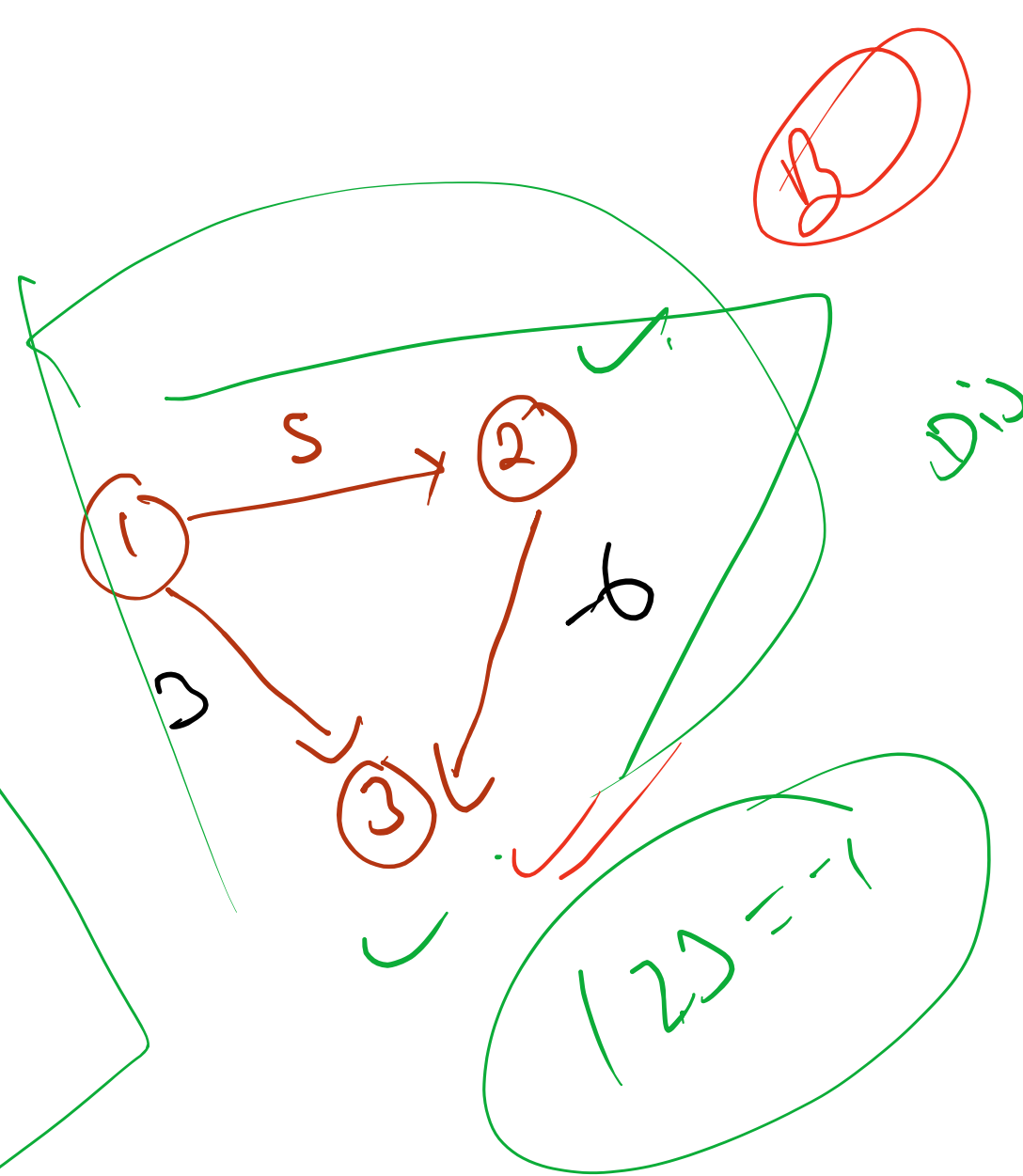


① → 2 ✓  
1 → 3 ✓  
1 → 4 ✓  
1 → 5 ✓

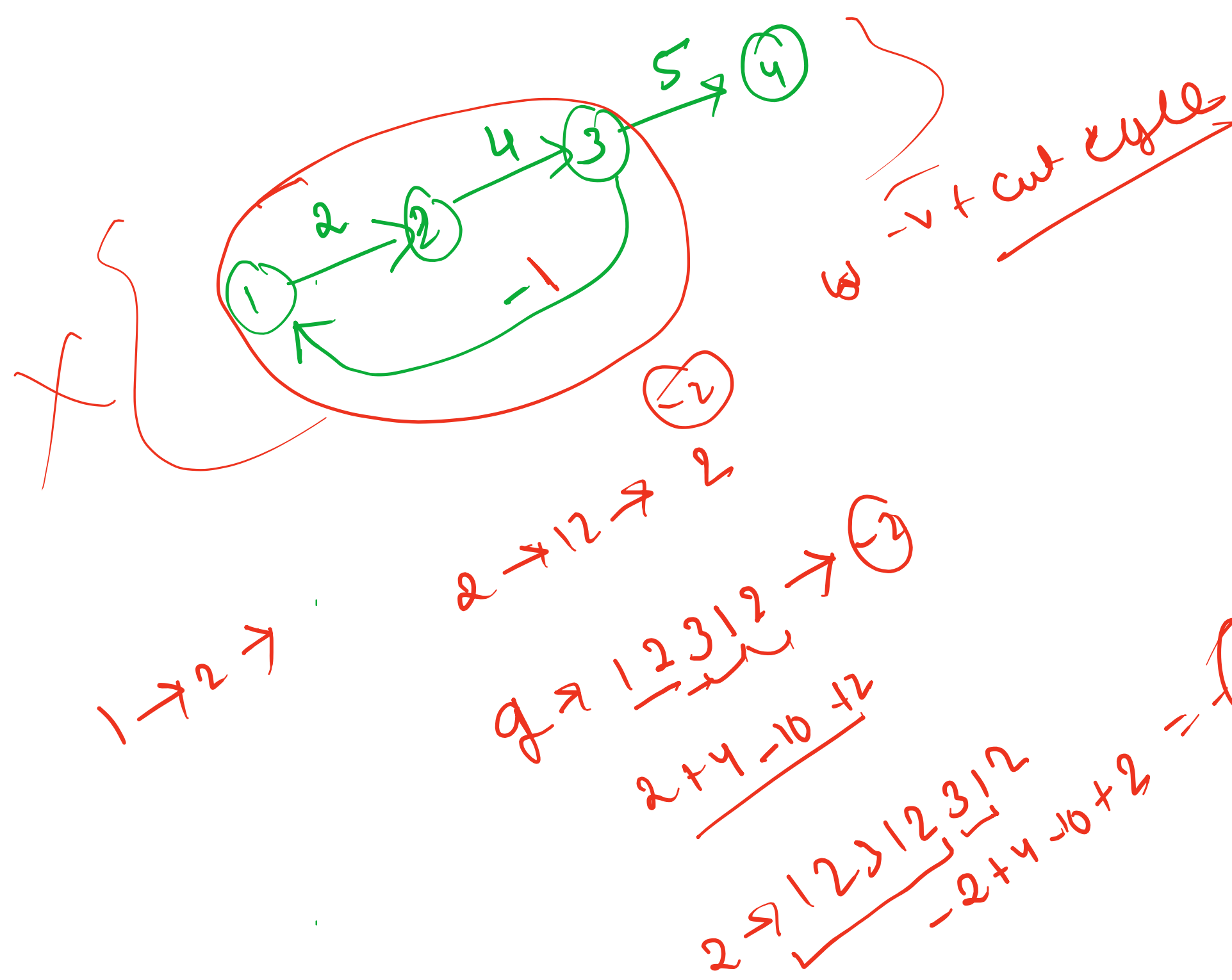
1	1	1	0
2	12	3	
4	14	2	
3	14	3	
3	123	7	
3	145	6	

1 1 0  
4 14 0 2  
2 12 0 3  
5 14 0 3  
3 14 5 0 6

1 3 2

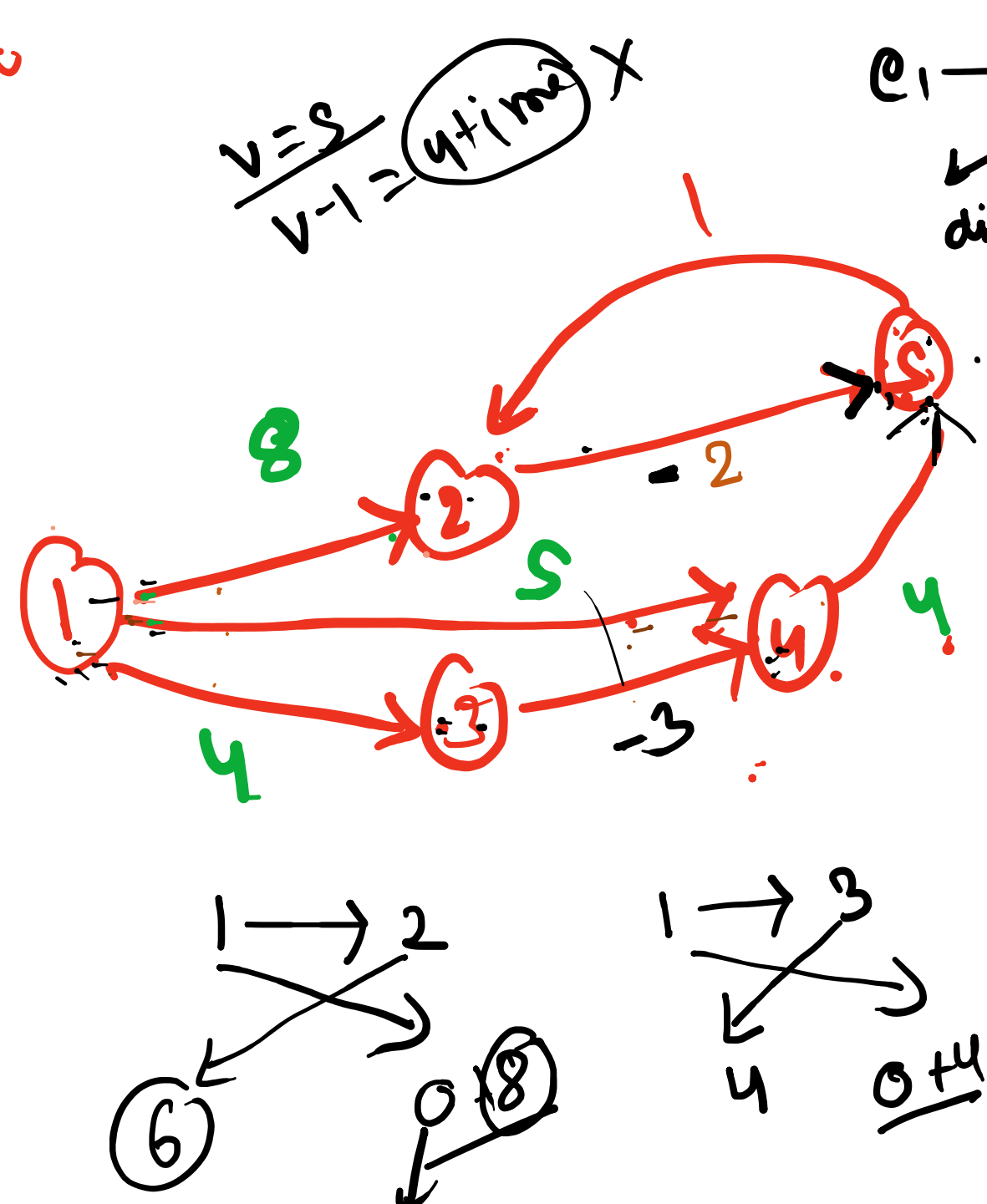


1	1	1	0
2	12	5	
3	13	3	



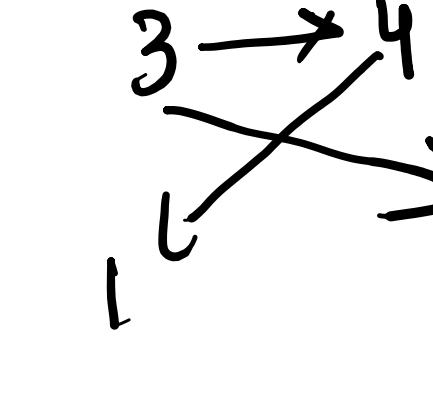
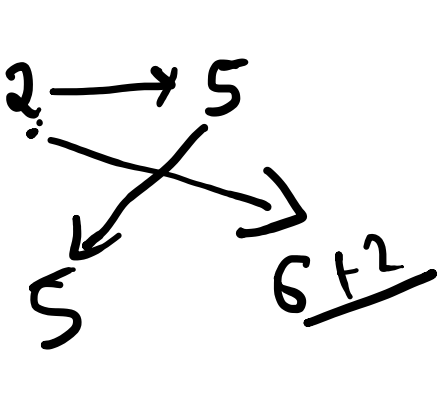
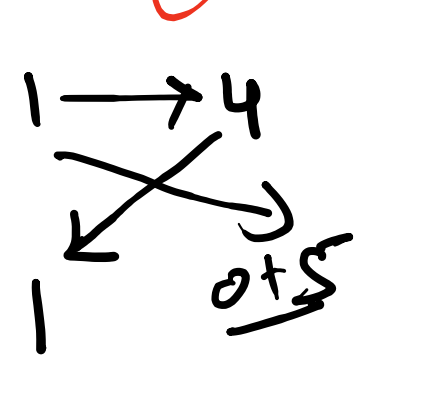
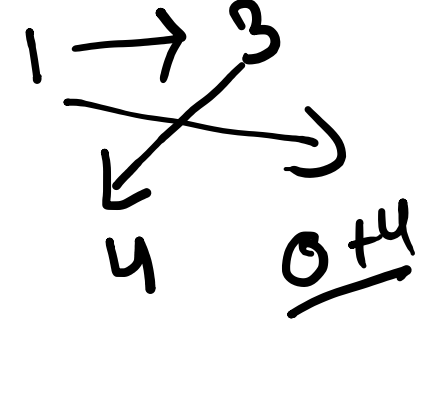
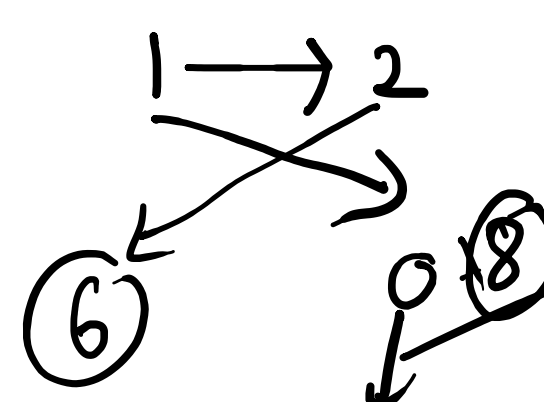
① -v+ → c  
② wt.  
→ Bellman Ford

Class EdgePair

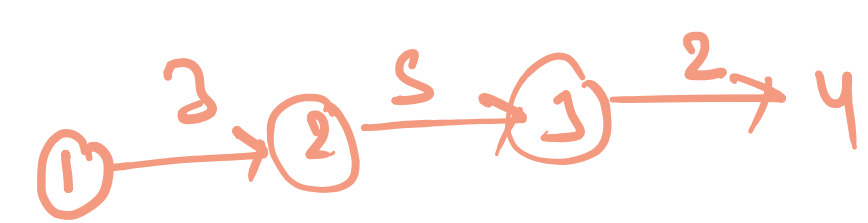
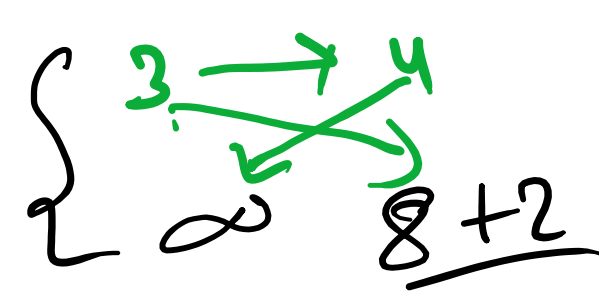


Class EdgePair  
e1 → e2  
discuss  
given test

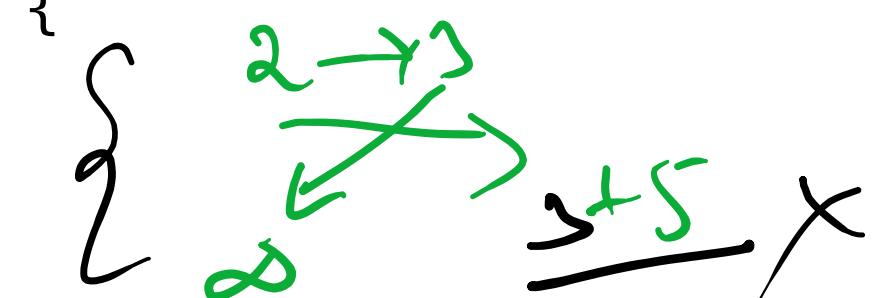
1	0
2	6
3	4
4	1
5	5



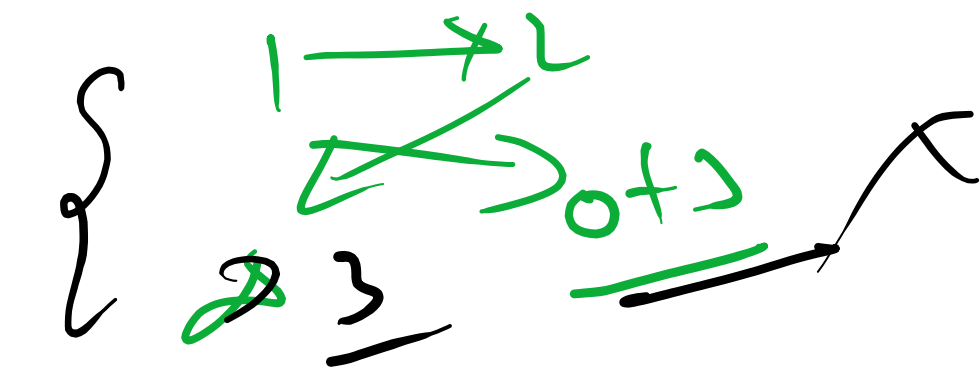
```
public void BellmanFord() {  
    int v = map.size();  
    int[] dis = new int[v + 1];  
    for (int i = 2; i < dis.length; i++) {  
        dis[i] = 999989;  
    }  
    List<Edgepair> ll = GetAllEdge();  
    for (int i = 1; i < v; i++) {  
        for (Edgepair e : ll) {  
            if (dis[e.e2] > dis[e.e1] + e.cost) {  
                dis[e.e2] = dis[e.e1] + e.cost;  
            }  
        }  
        for (int i = 1; i < dis.length; i++) {  
            System.out.println(i + " " + dis[i]);  
        }  
    }  
}
```



3 → 4 2 ✓  
2 → 3 5 ✓  
1 → 2 3 ✓



1	0
2	3
3	8
4	6



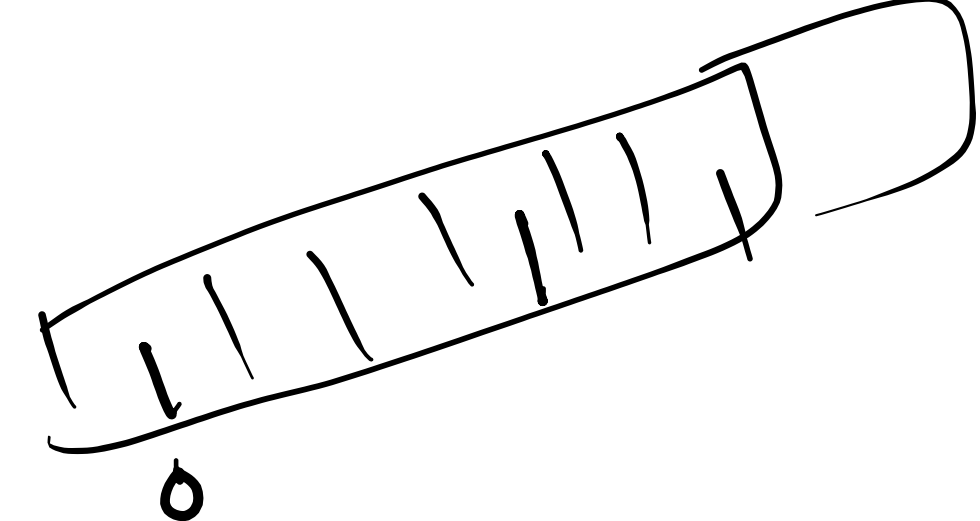
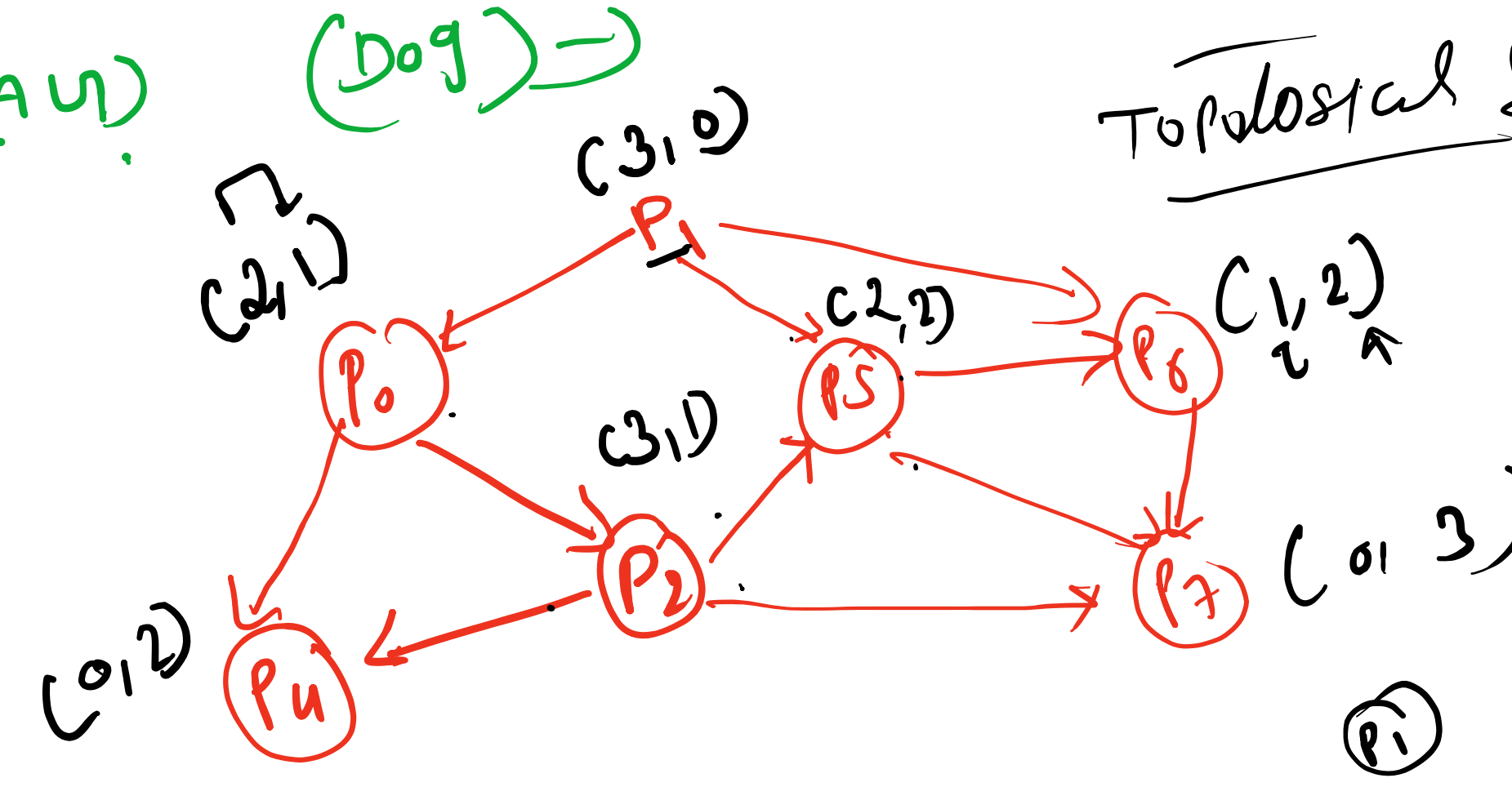
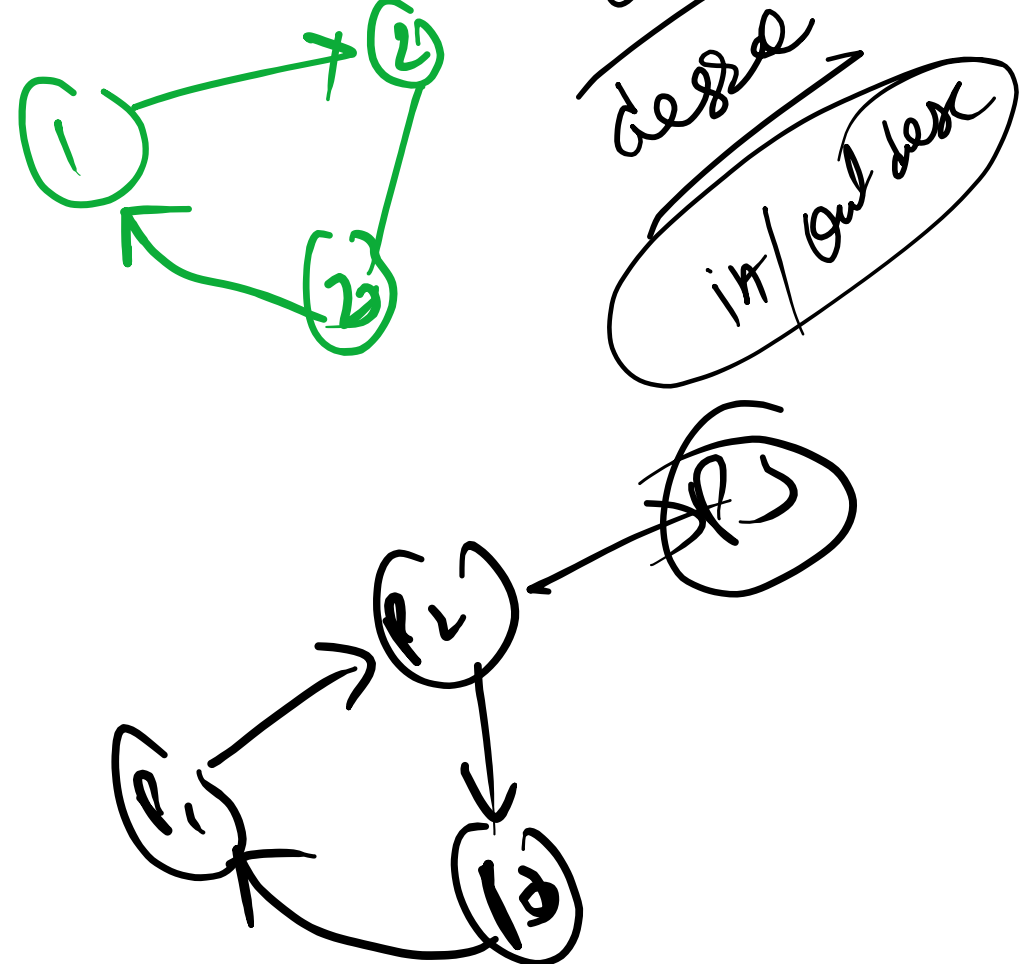
Topological Sort

(D.A.U)

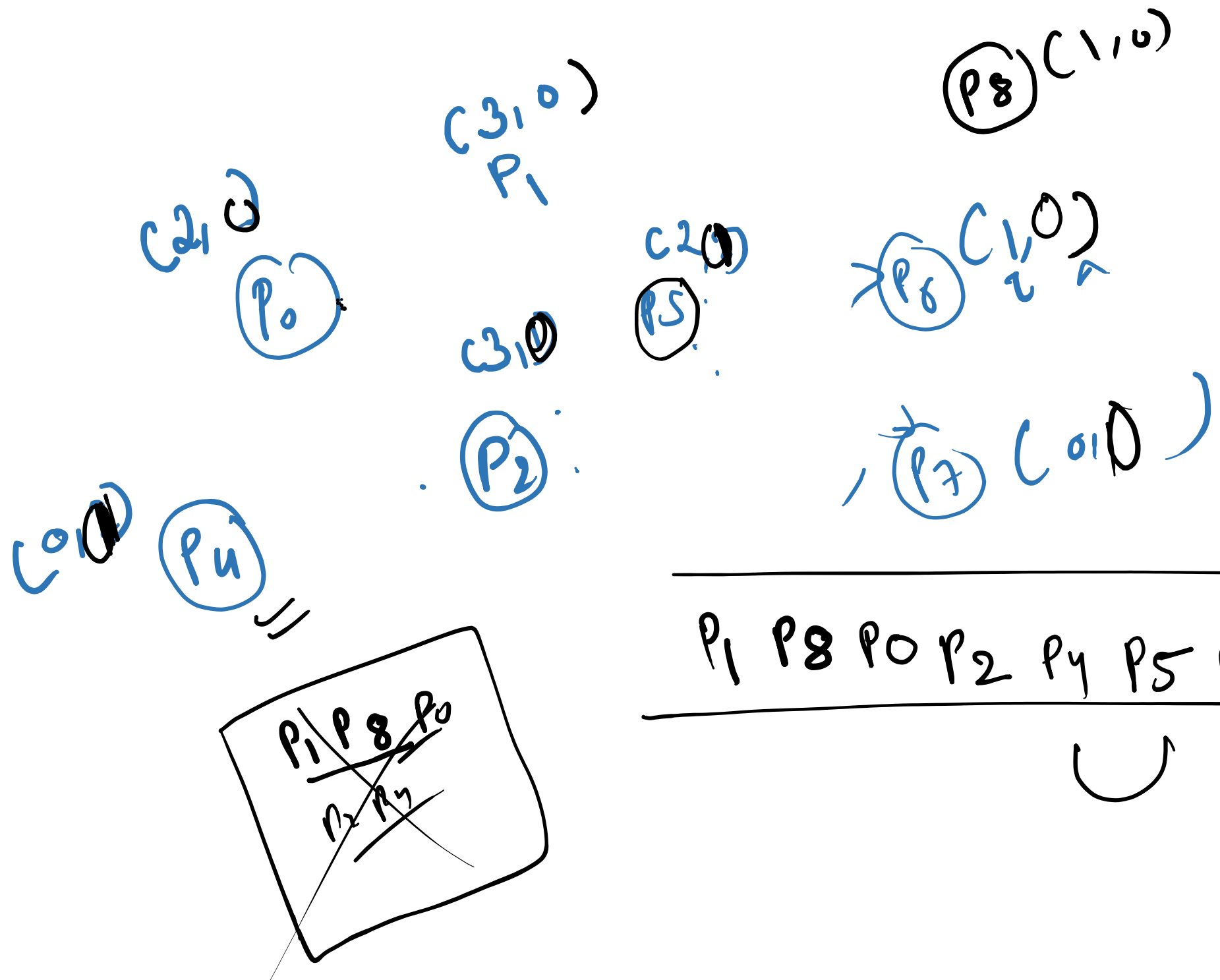
(Dog)

Topological Sort

1 2 3



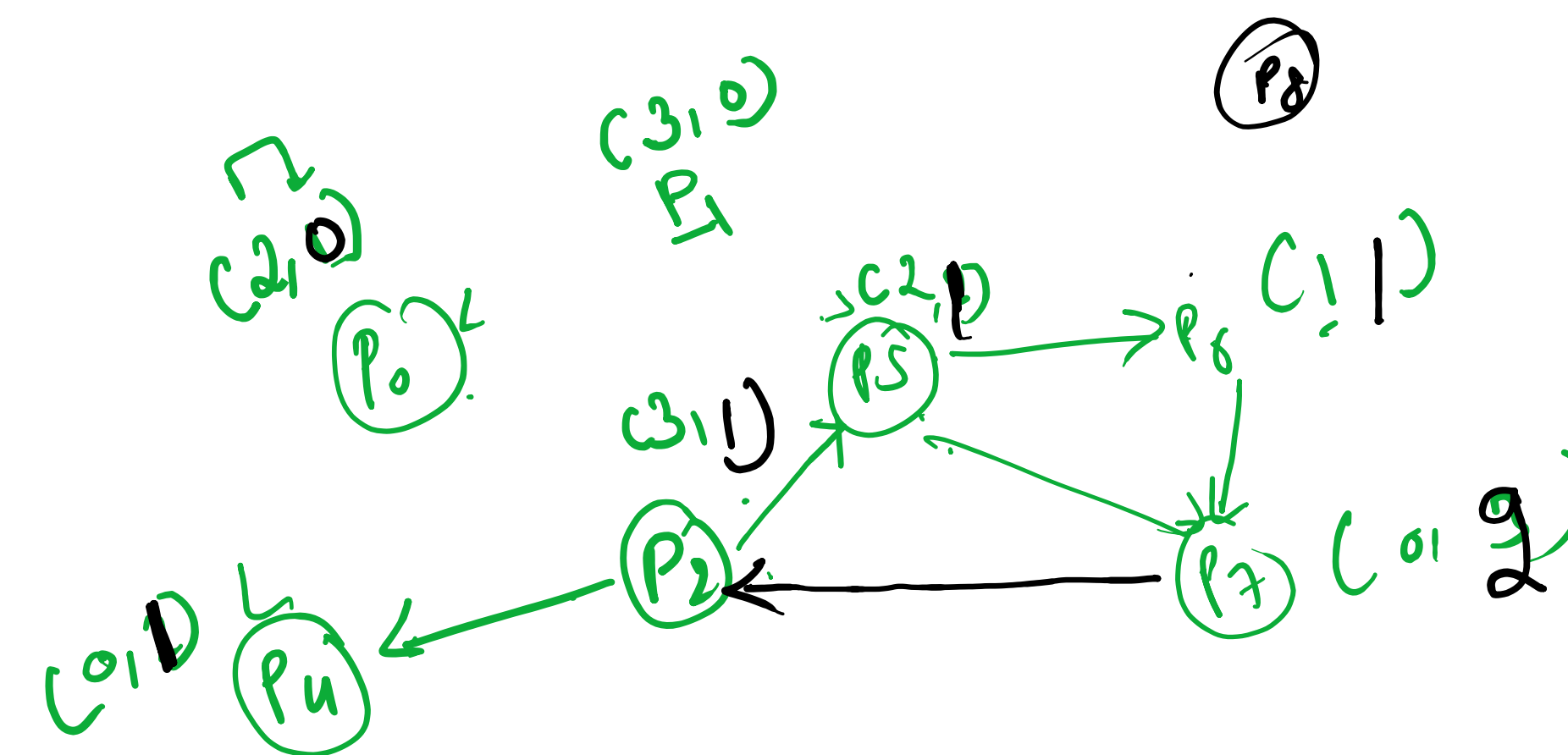
For c in e1.map.keySet() {  
 For (int e2) {  
 map.add(e1, e2);  
 }  
}



1	0
2	3
3	8
4	6

P1 P8 P0 P2 P4 P5 P6 P7

```
public void Topological() {  
    int[] in = Indegree();  
    Queue<Integer> q = new LinkedList<>();  
    for (int i = 0; i < in.length; i++) {  
        if (in[i] == 0) {  
            q.add(i);  
        }  
    }  
    while (!q.isEmpty()) {  
        //1. remove  
        int r = q.poll();  
        System.out.print(r + " ");  
        for (int nbrs : map.get(r).keySet()) {  
            in[nbrs]--;  
            if (in[nbrs] == 0) {  
                q.add(nbrs);  
            }  
        }  
    }  
}
```



1	0
2	3
3	8
4	6