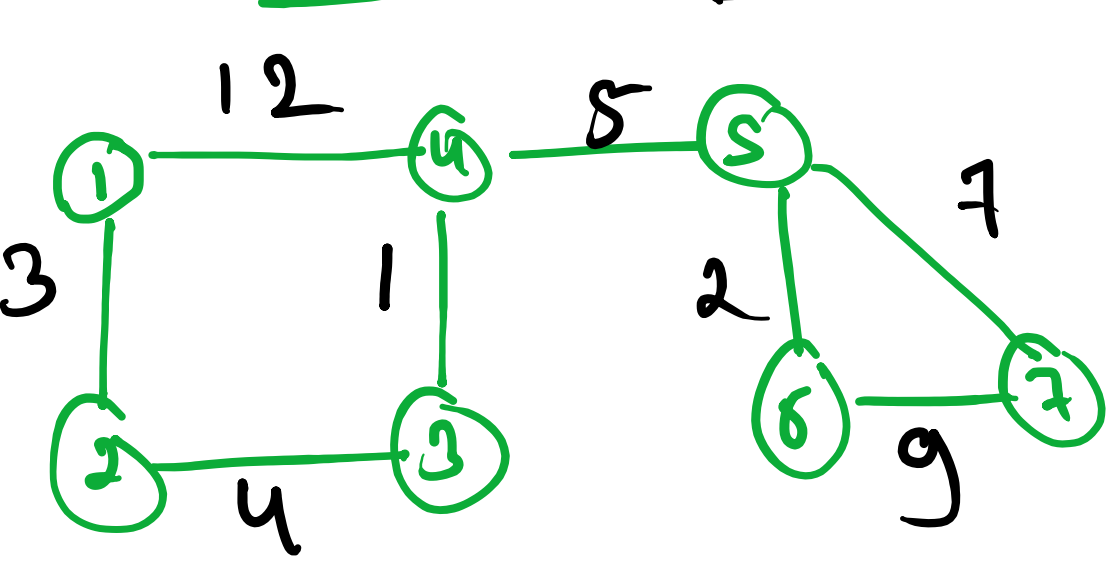


SSSP Single Source Shortest Path Also



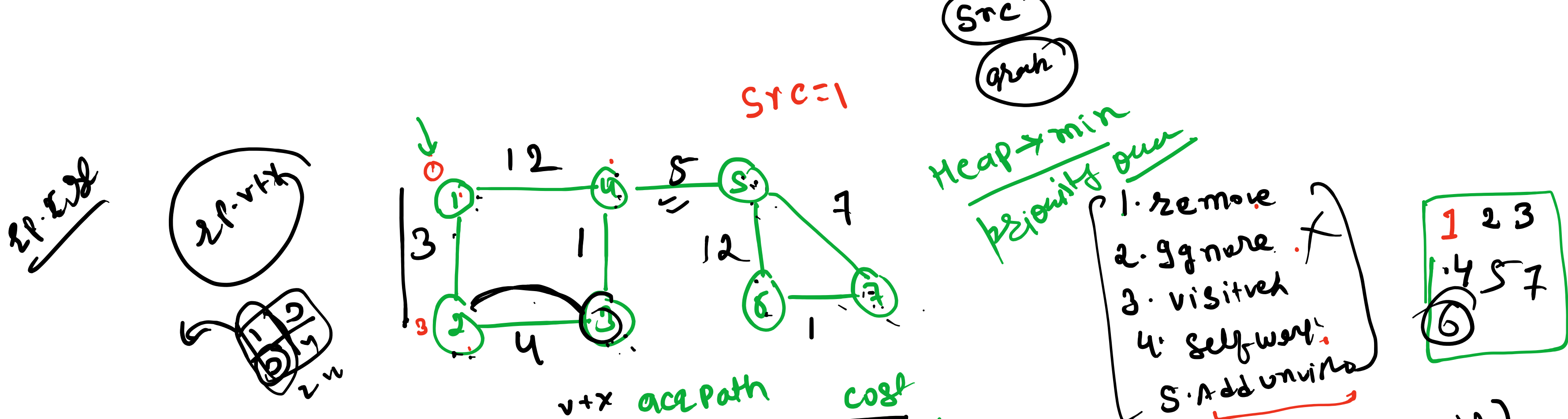
8-9

1-4 ~
1 2 3 4 → 8
14 → 12

1-2 min cost path

1-3 → min cost path
1-4 "
1-5 "
1-6 "
1-7 "

SSSP → Dijkstra
 → Bellman Ford

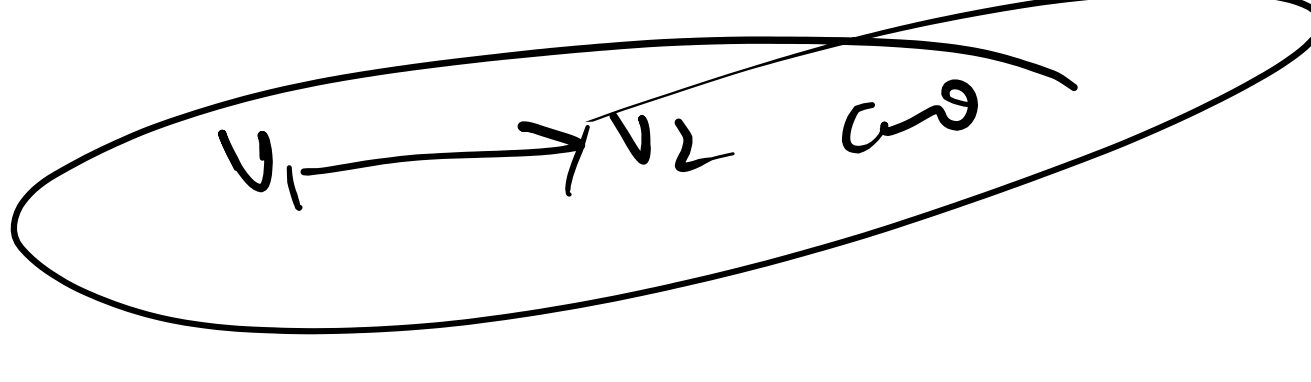


1 1 @ 0
2 12 @ 3
3 123 @ 7
4 1234 @ 8
5 12345 @ 13
6 123456 @ 20
7 1234567 @ 21

v	x	acq path	cost
1	1		0
2	14		12
3	123		7
4	1234		8
5	12345		13
6	12456		25
7	12357		20
8	1234576		21

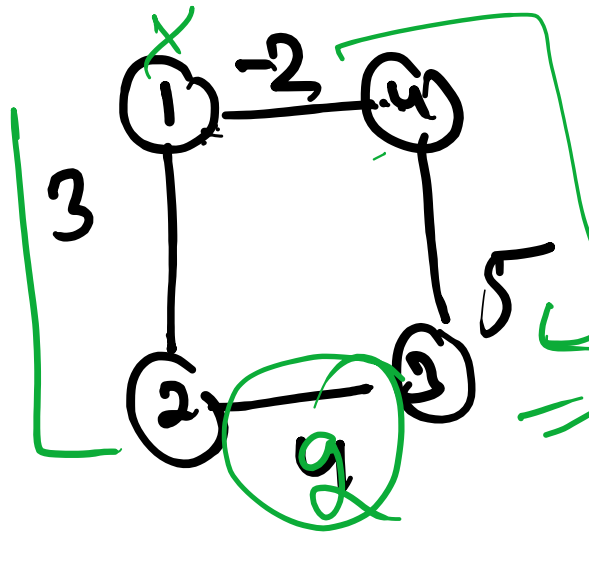
map.get(rp.vtx).keySet()

1 1 @ 0
2 12 @ 3
3 123 @ 7
4 1234 @ 8
5 12345 @ 13
6 123457 @ 20
7 1234576 @ 21



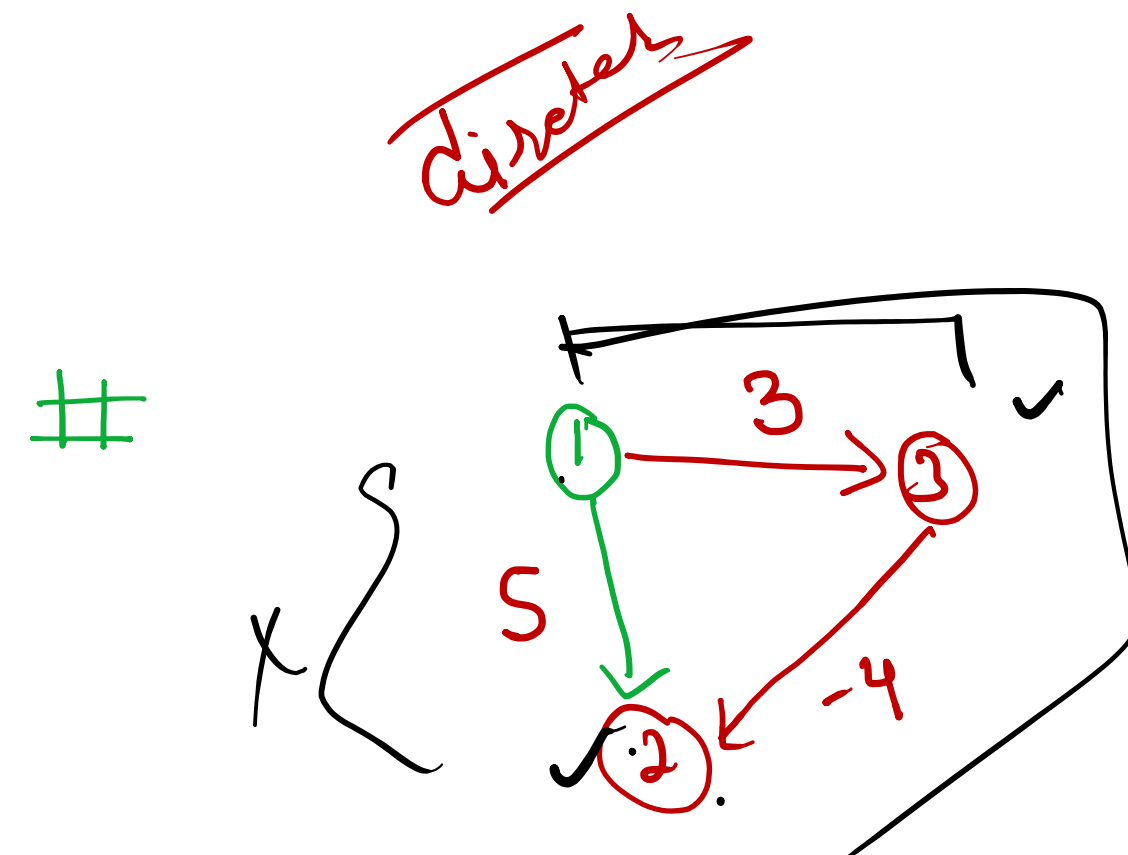
```
public void Dijkstra(int src) {
    PriorityQueue<DijkstraPair> pq = new PriorityQueue<>();
    HashSet<Integer> visited = new HashSet<>();
    pq.add(new DijkstraPair(src, "" + src, 0));
    while (!pq.isEmpty()) {
        // 1. remove
        DijkstraPair rp = pq.poll();
        // 2. Ignore if Already visited
        if (visited.contains(rp.vtx)) {
            continue;
        }
        // 3. Marked Visited
        visited.add(rp.vtx);
        // 4. Self Work
        System.out.println(rp);
        // Add Unvisited nbrs
        for (int nbrs : map.get(rp.vtx).keySet()) {
            if (!visited.contains(nbrs)) {
                int cost = map.get(rp.vtx).get(nbrs);
                pq.add(new DijkstraPair(nbrs, rp.acq_path + nbrs, rp.cost + cost));
            }
        }
    }
}
```

14



1 1 @ 0
4 14 @ 2
2 12 @ 3
3 123 @ 5

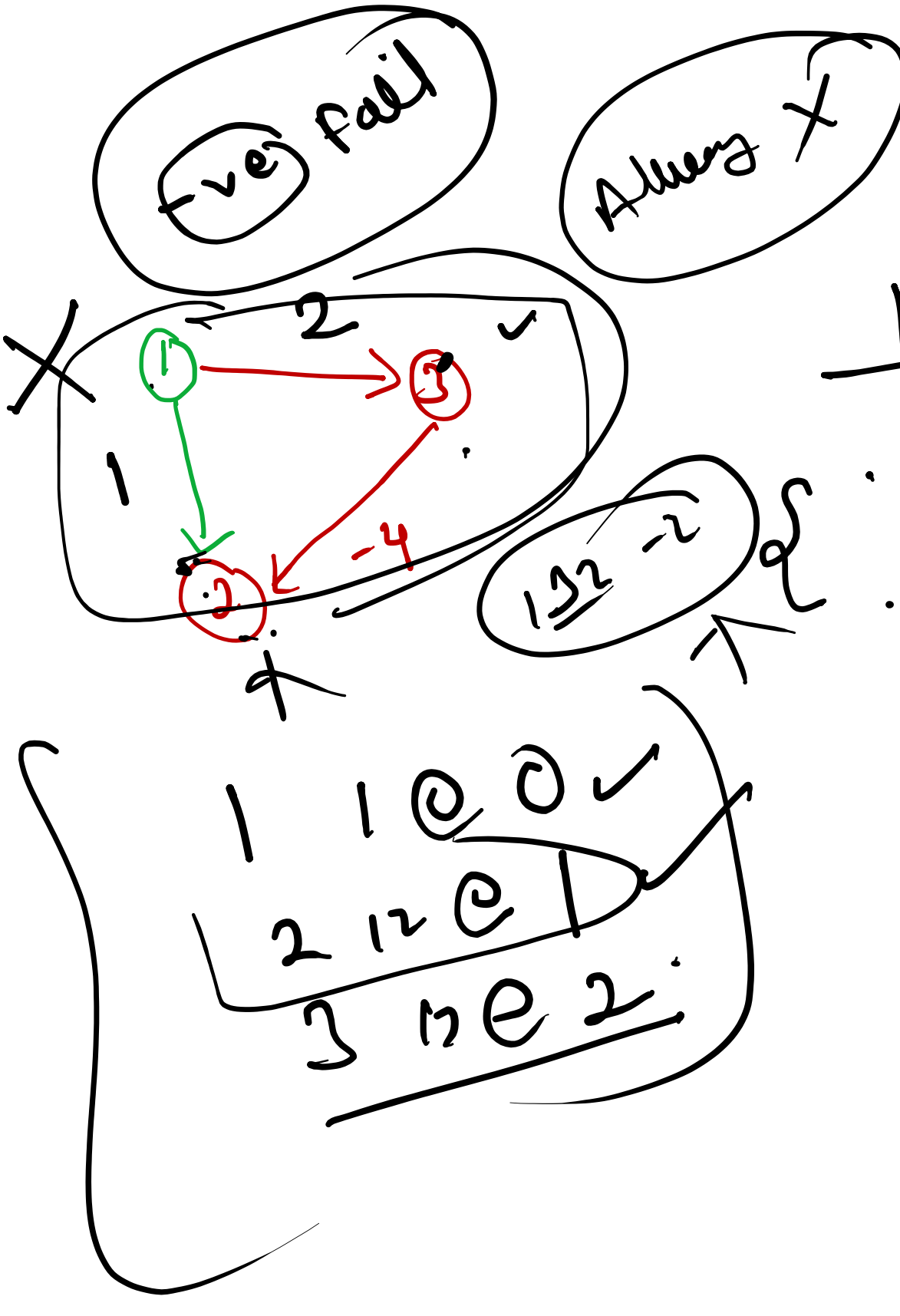
v	x	acq path	cost
1	1		0
2	14		3
3	143		7
4	1432		8



1	1	0
2	12	5
3	123	7
4	1234	8

122

1-2
1-3
1. remove
2. ignore
3. make visited
4. self work
5. add unvisited

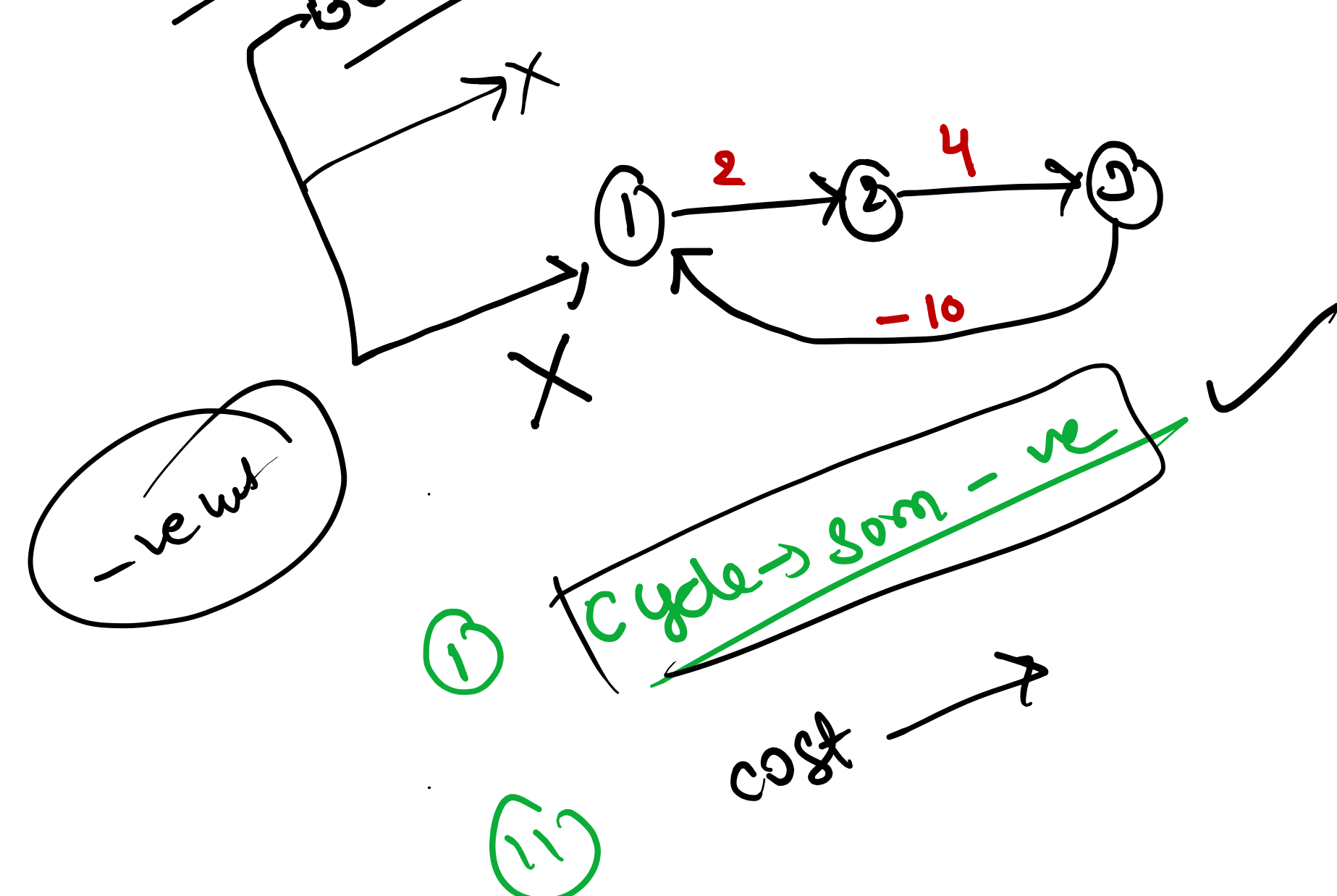


1	1	0
2	12	5
3	123	7
4	1234	8

Dijkstra
dir → va
unv → end

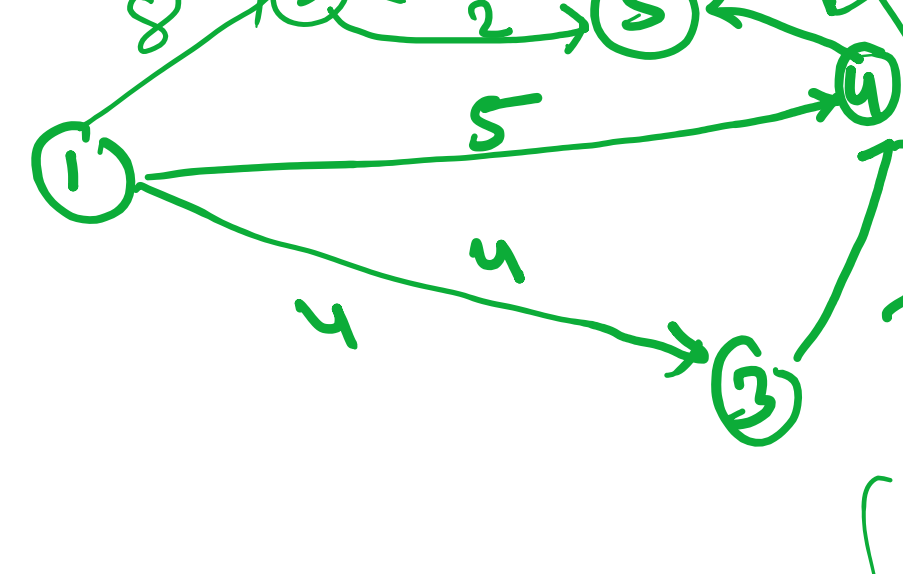
-ve wgt

-4

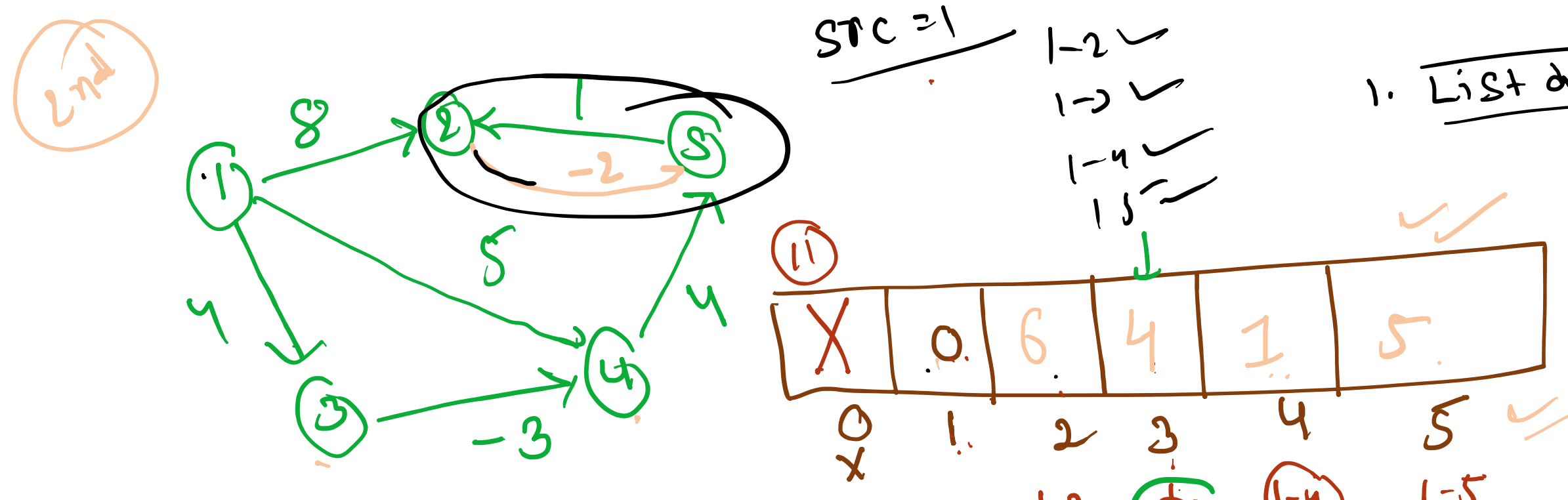
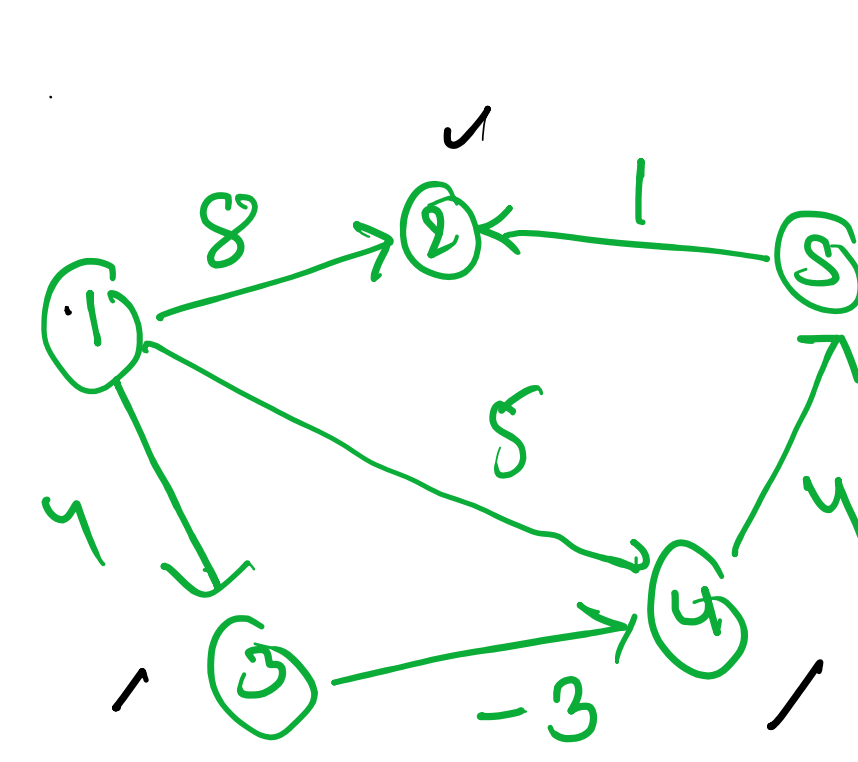
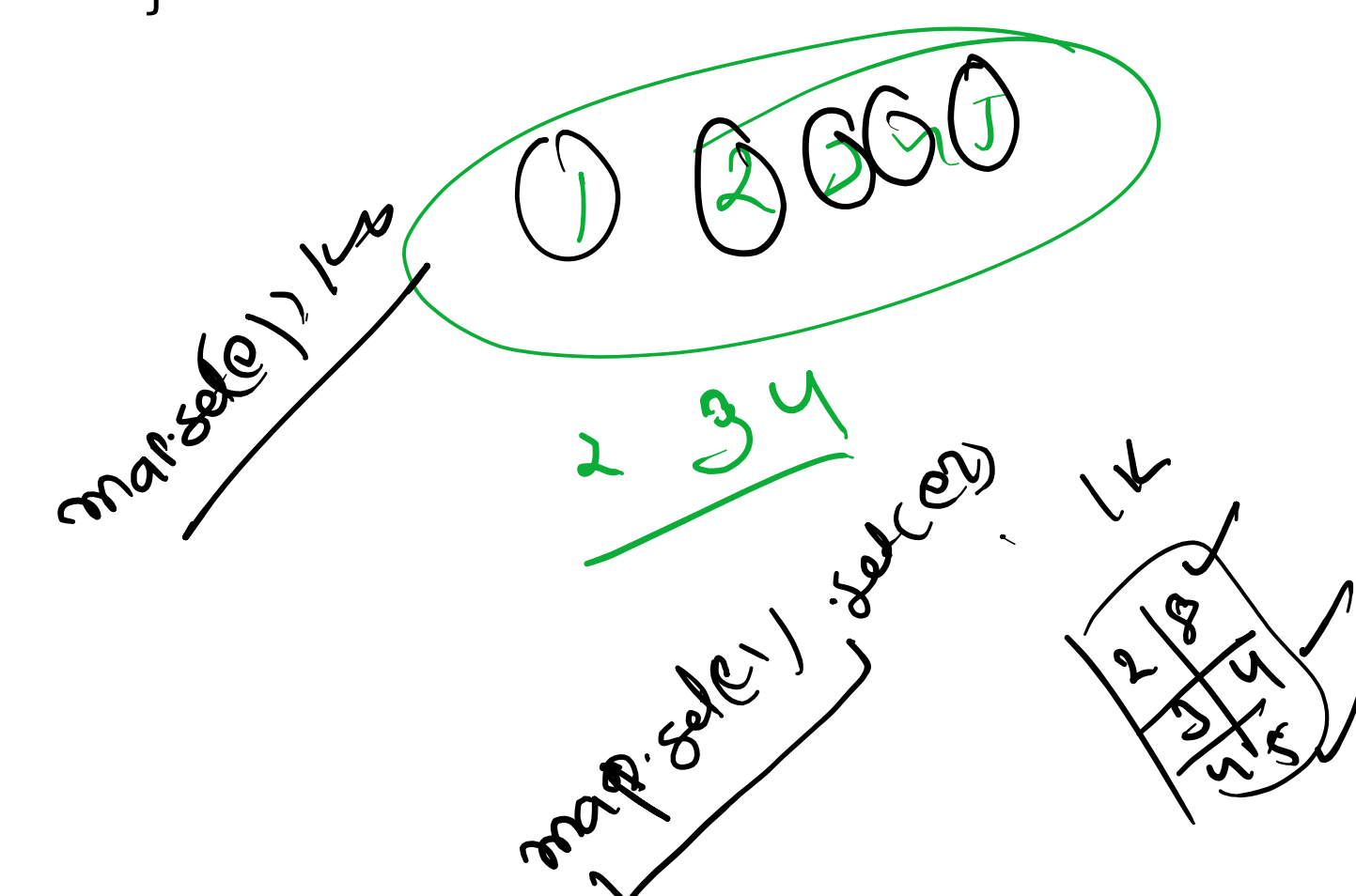


3 → 123
3 → 123123 = 8
6 - 10 + 2 + 4 = 2
3 → 123123123 = 0
2 2 6

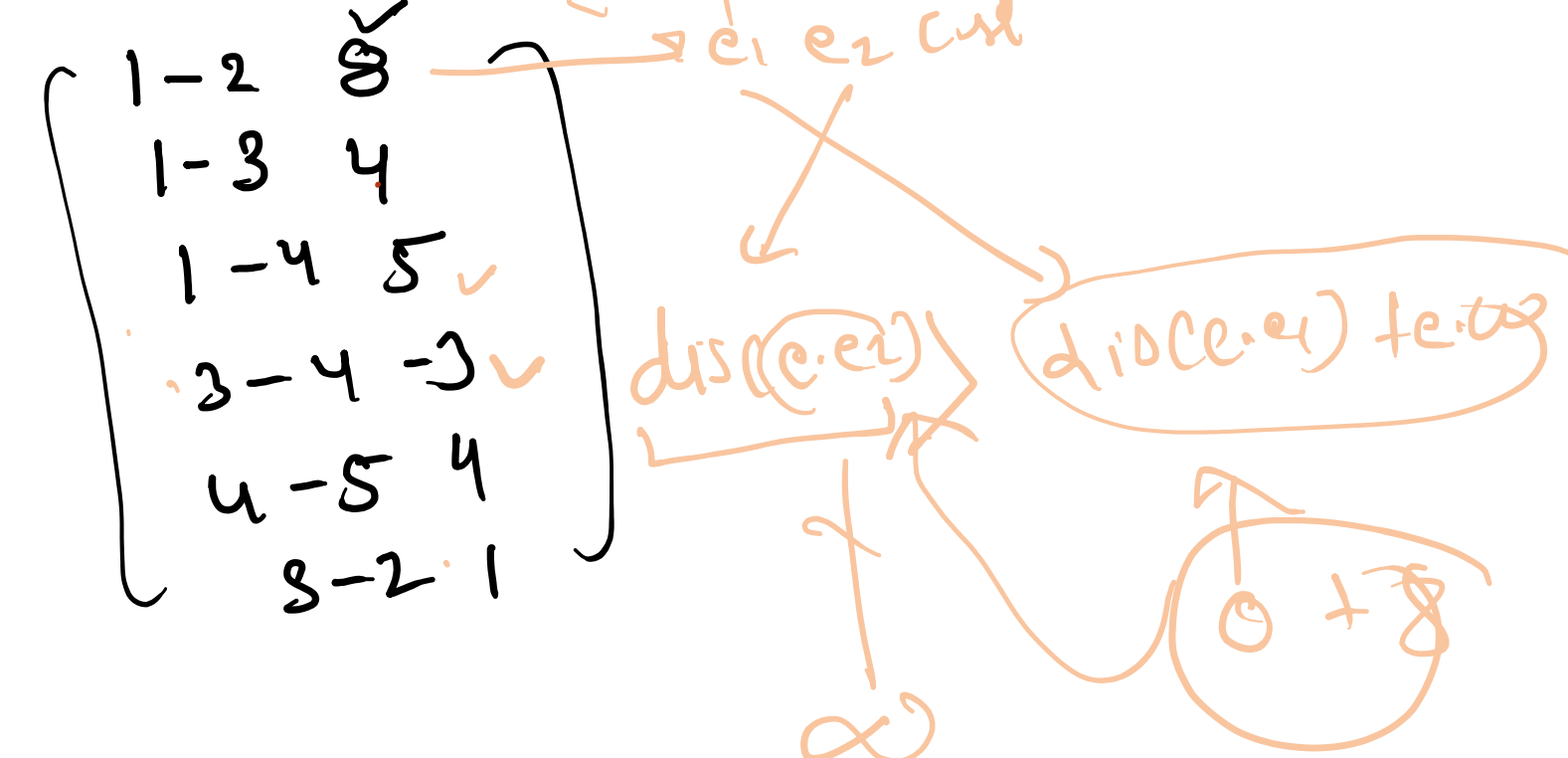
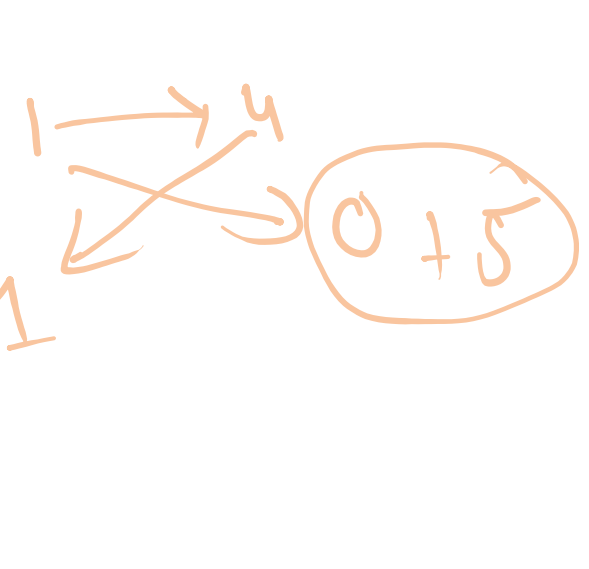
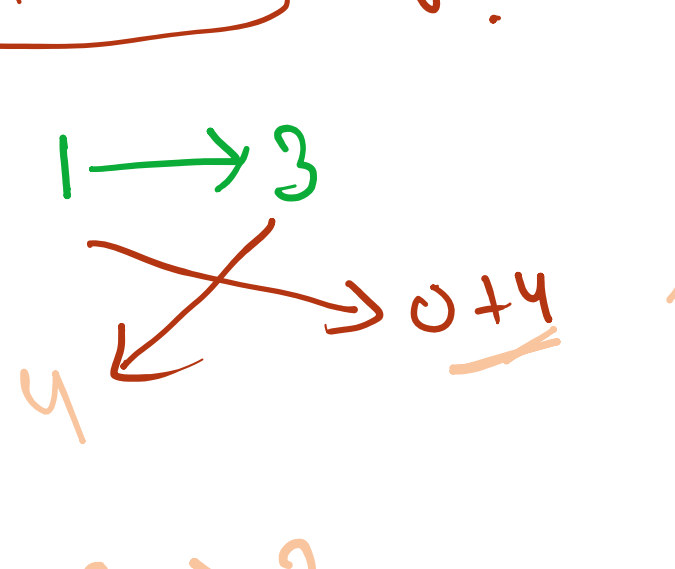
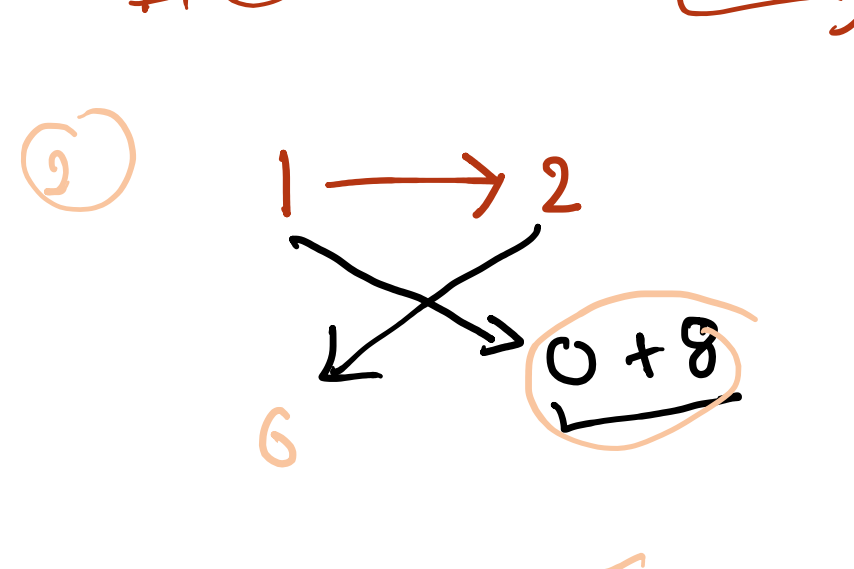
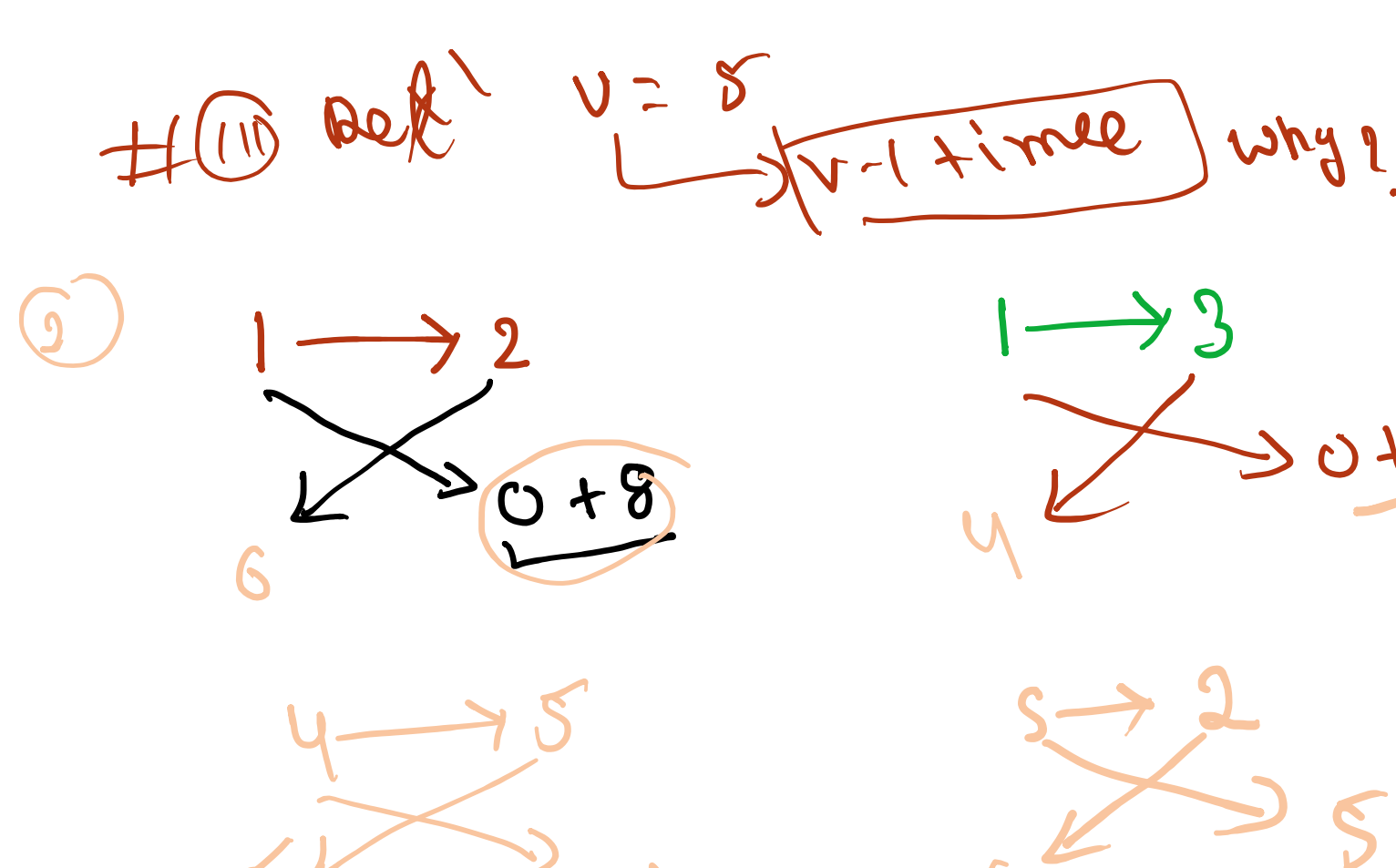
```
public static void main(String[] args) {
    BellMan_Ford bf = new BellMan_Ford(5);
    bf.AddEdge(1, 2, 8);
    bf.AddEdge(1, 3, 4);
    bf.AddEdge(1, 4, 5);
    bf.AddEdge(3, 4, -3);
    bf.AddEdge(4, 5, 4);
    bf.AddEdge(5, 2, 1);
}
```



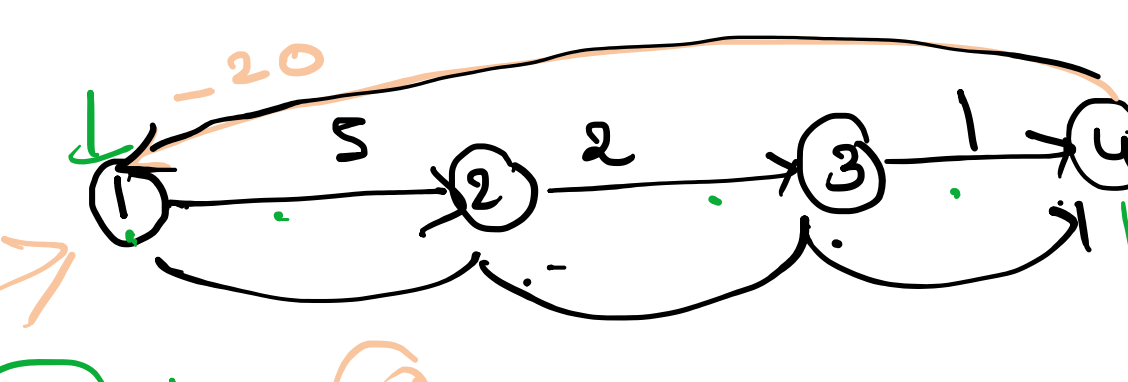
List
1 → 2 8
1 → 3 4
1 → 4 5
3 → 4 -3
4 → 5 4
5 → 2 1



1. List down
1-2 8
1-3 4
1-4 5
2-3 4
2-4 5
2-5 1
3-4 -3
3-5 4
4-5 4
5-2 1



```
public void BellmanFord() {
    // src=1
    List<EdgePair> ll = getAllEdgeList();
    int v = map.size();
    int[] dis = new int[v + 1];
    for (int i = 2; i < dis.length; i++) {
        dis[i] = 88888889;
    }
    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]);
    }
}
```



X	0	5	7	8
0	1	2	3	4

1. 3 → 4 1
2 → 3 2
1 → 2 5

MSR To Polished sol Monday