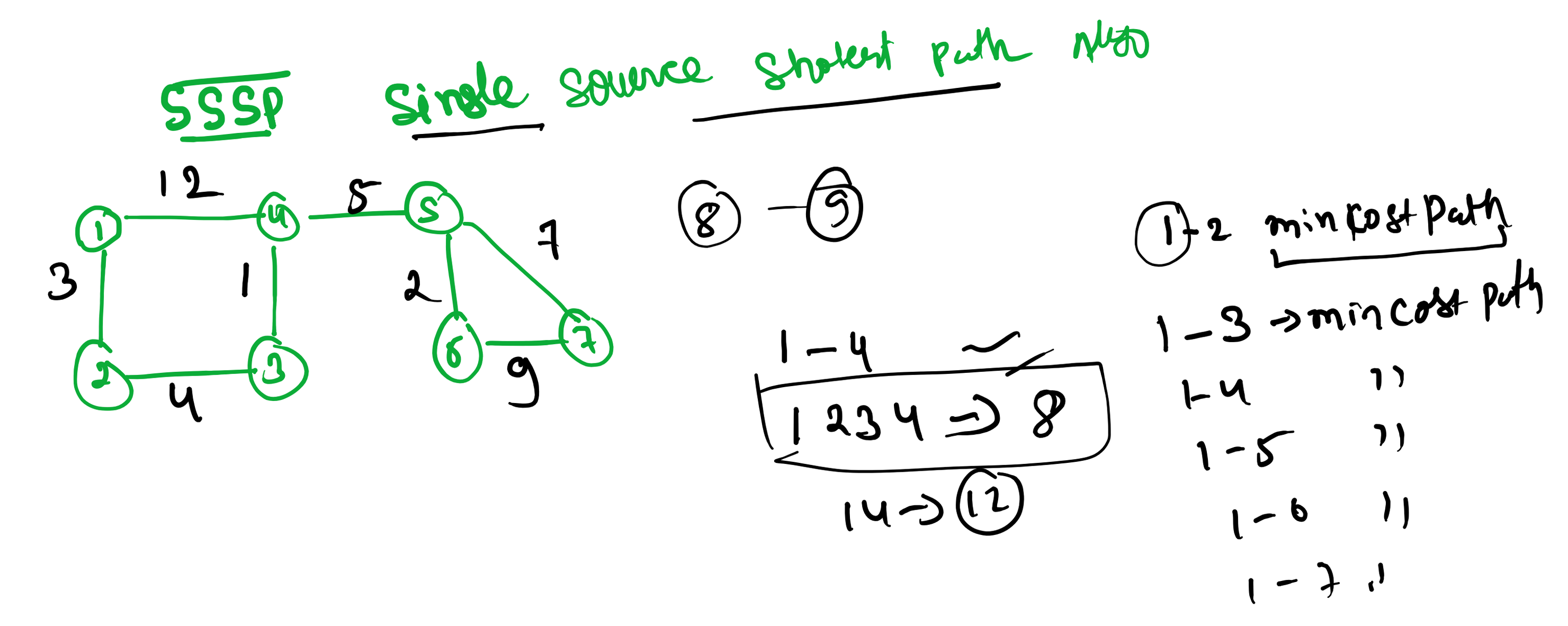
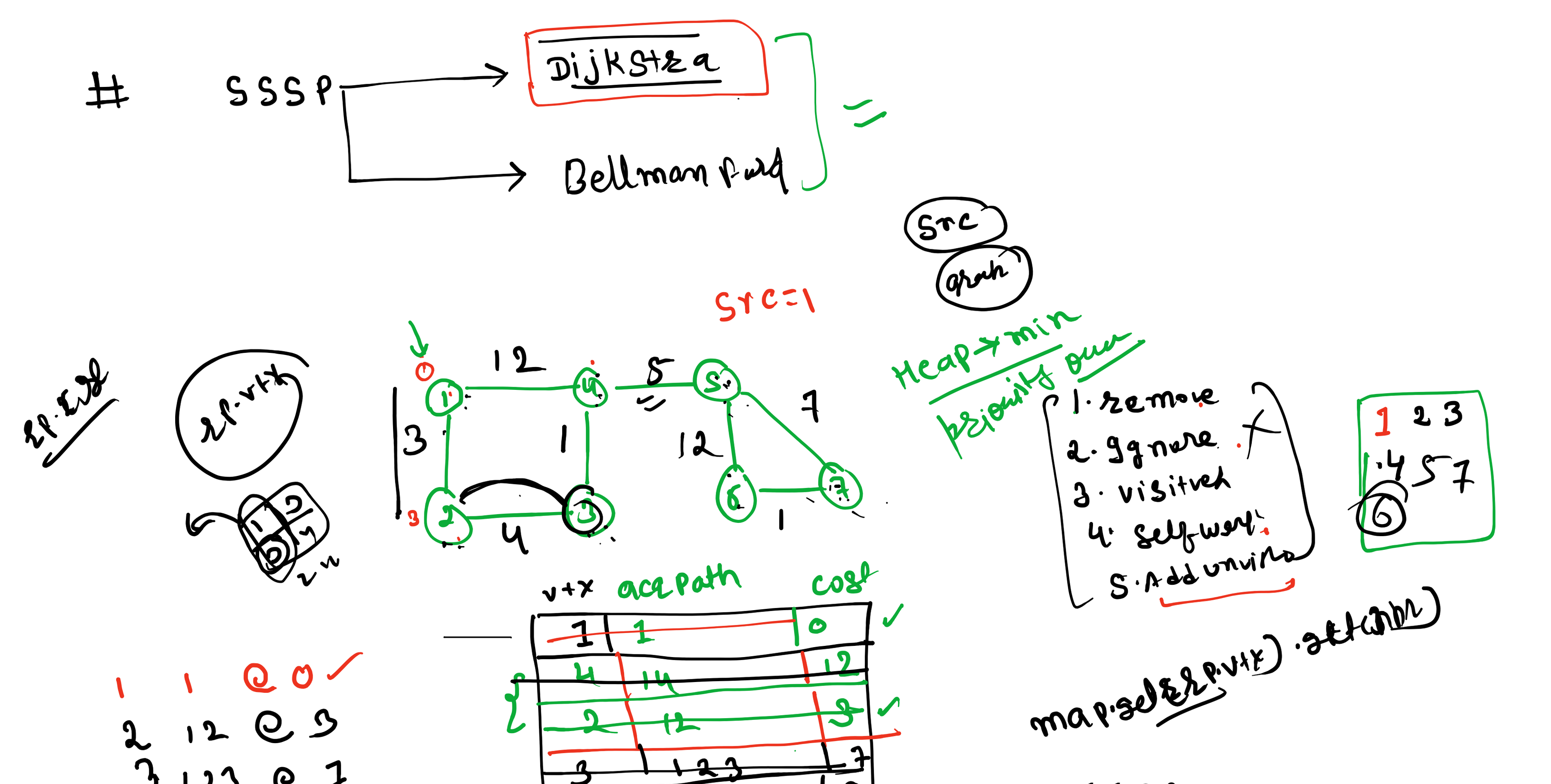
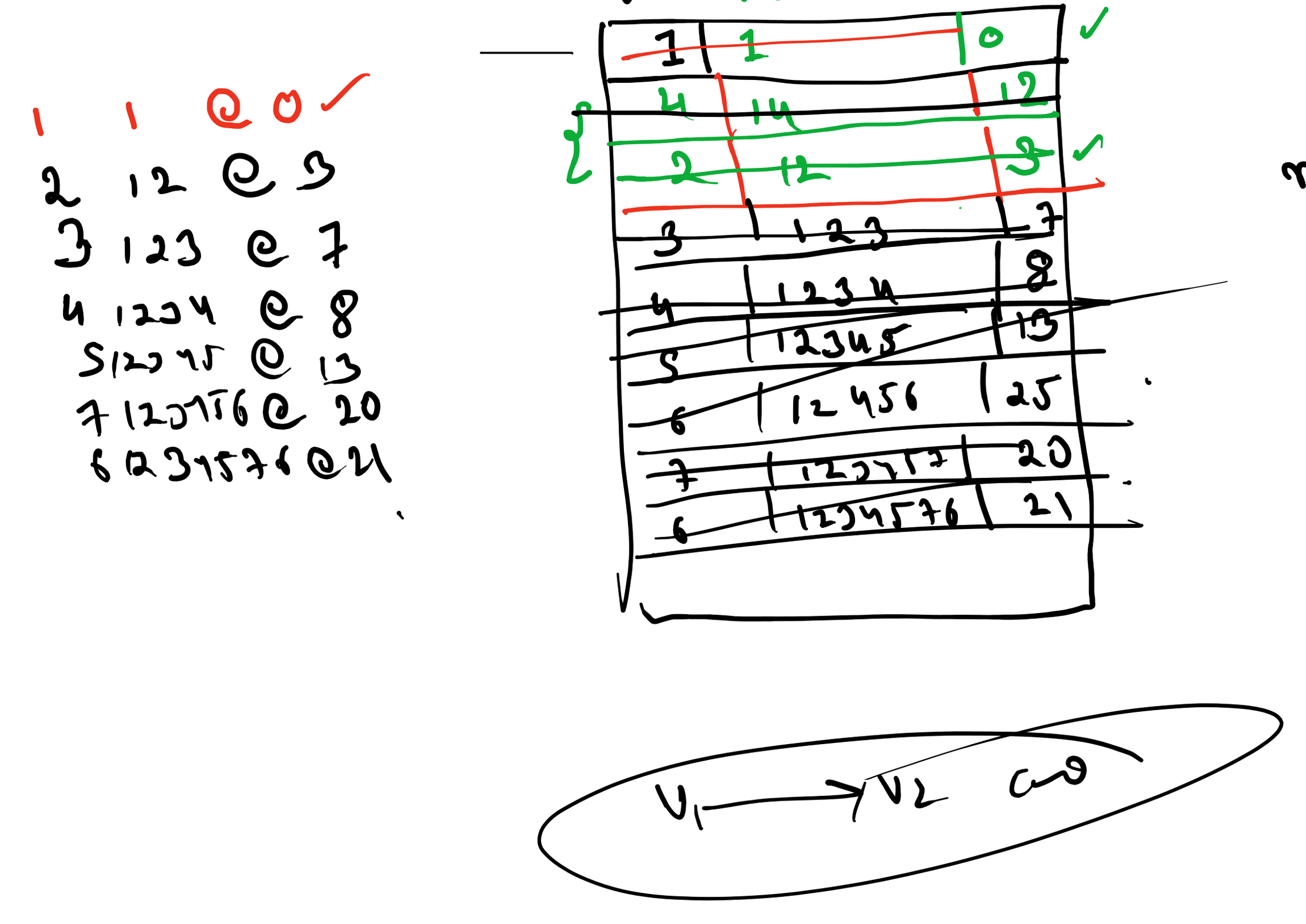
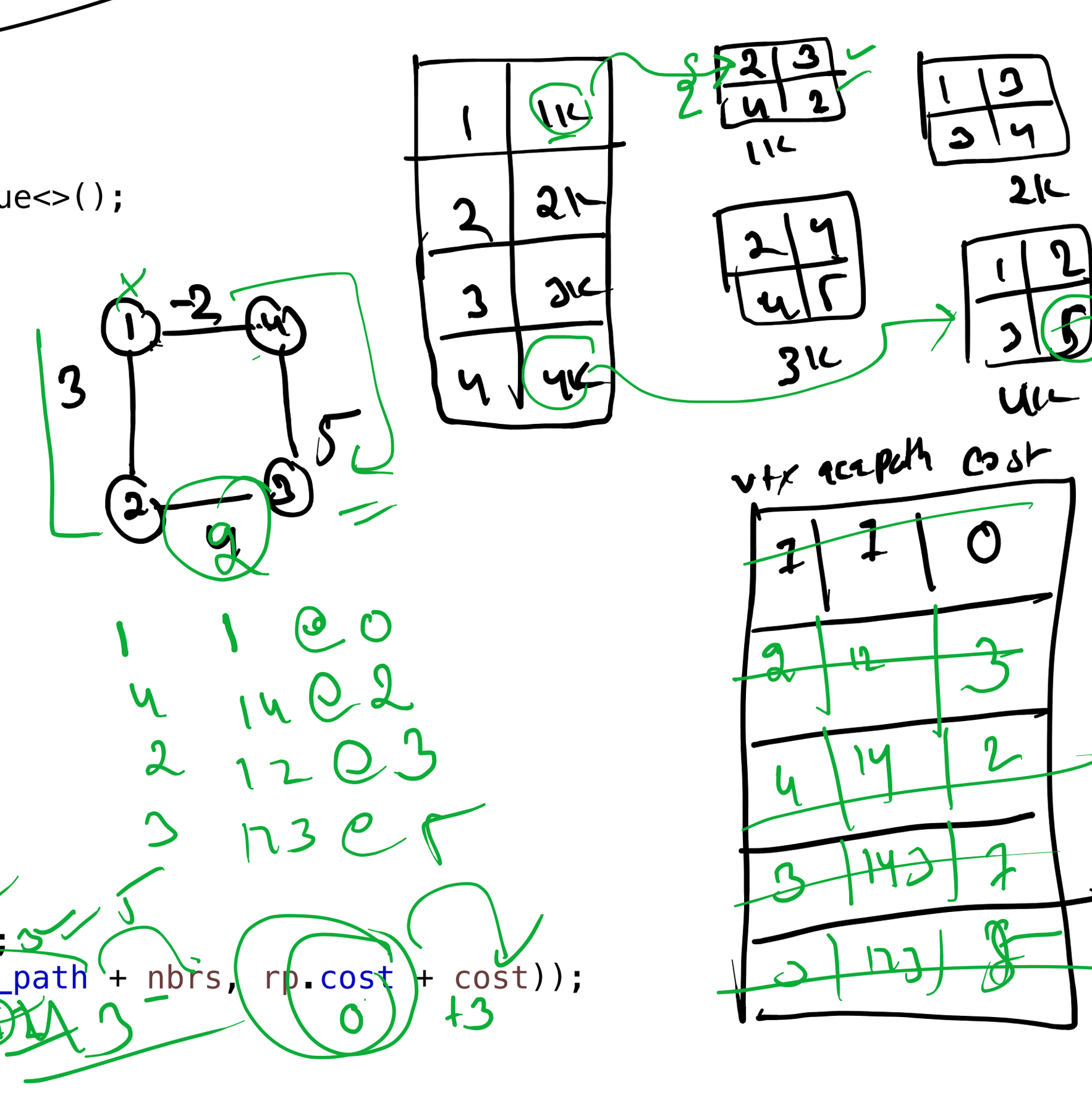


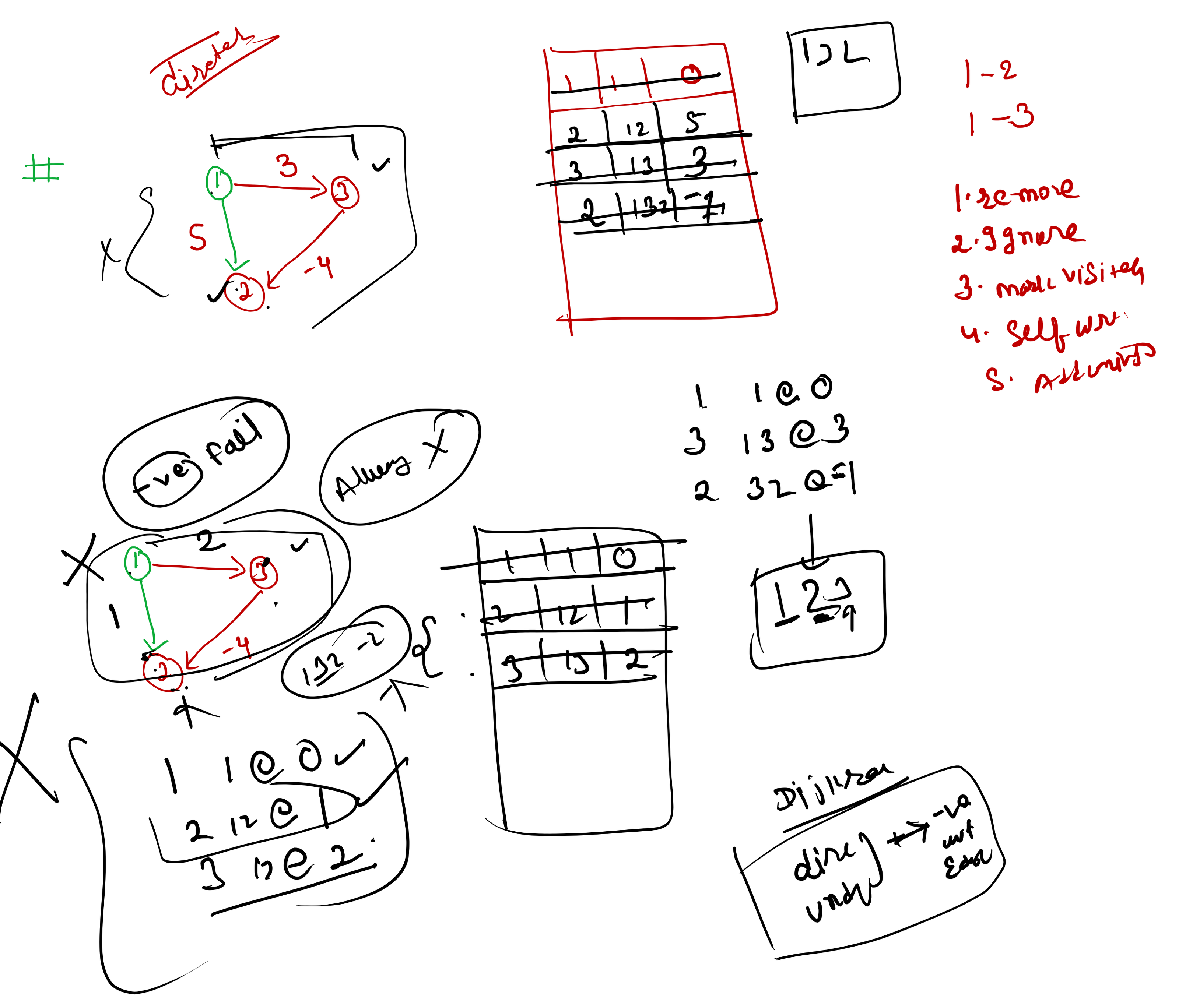


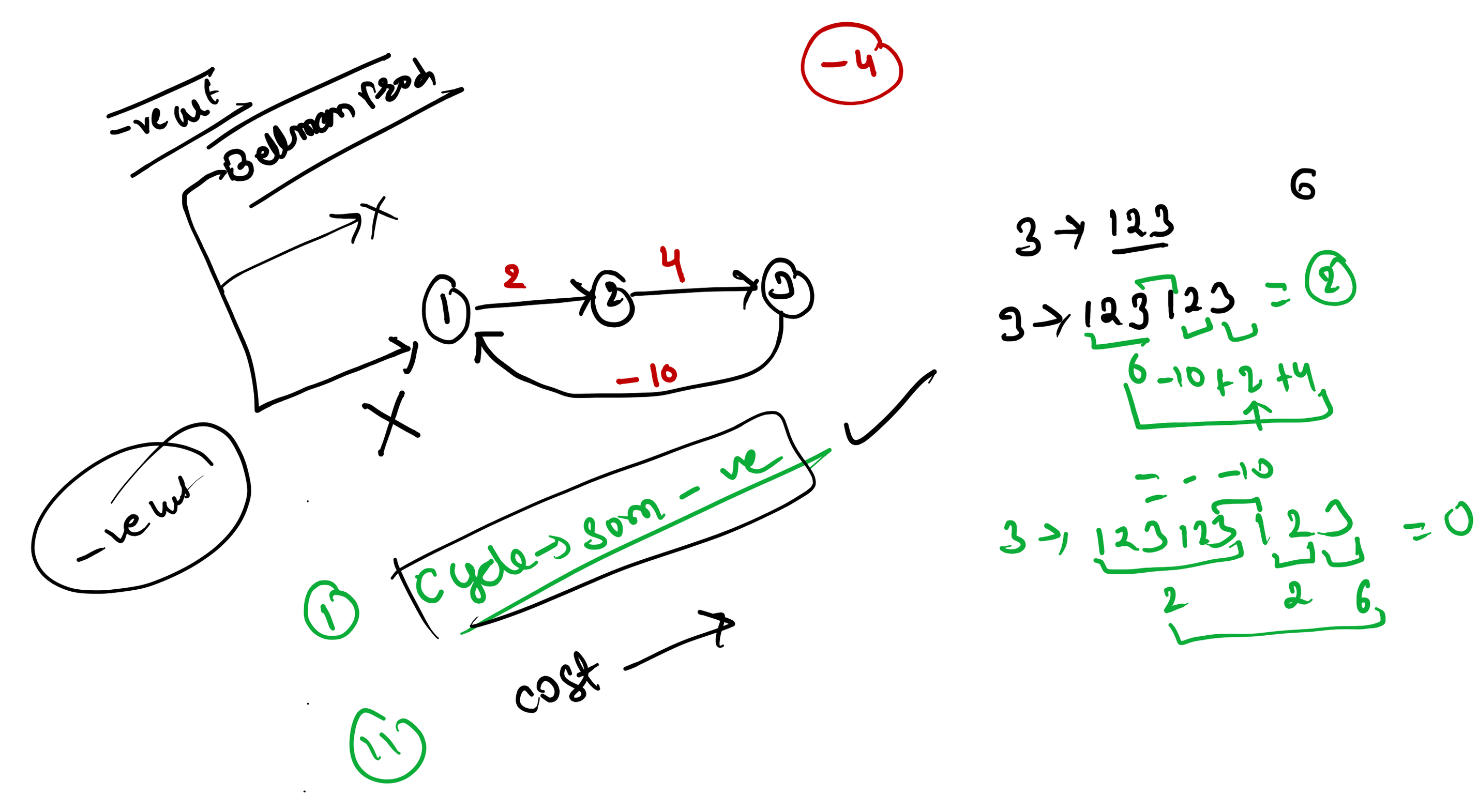


```
1 1 @ 0
2 12 @ 3
3 123 @ 7
4 1234 @ 8
5 12345 @ 13
7 123457 @ 20
6 1234576 @ 21
```

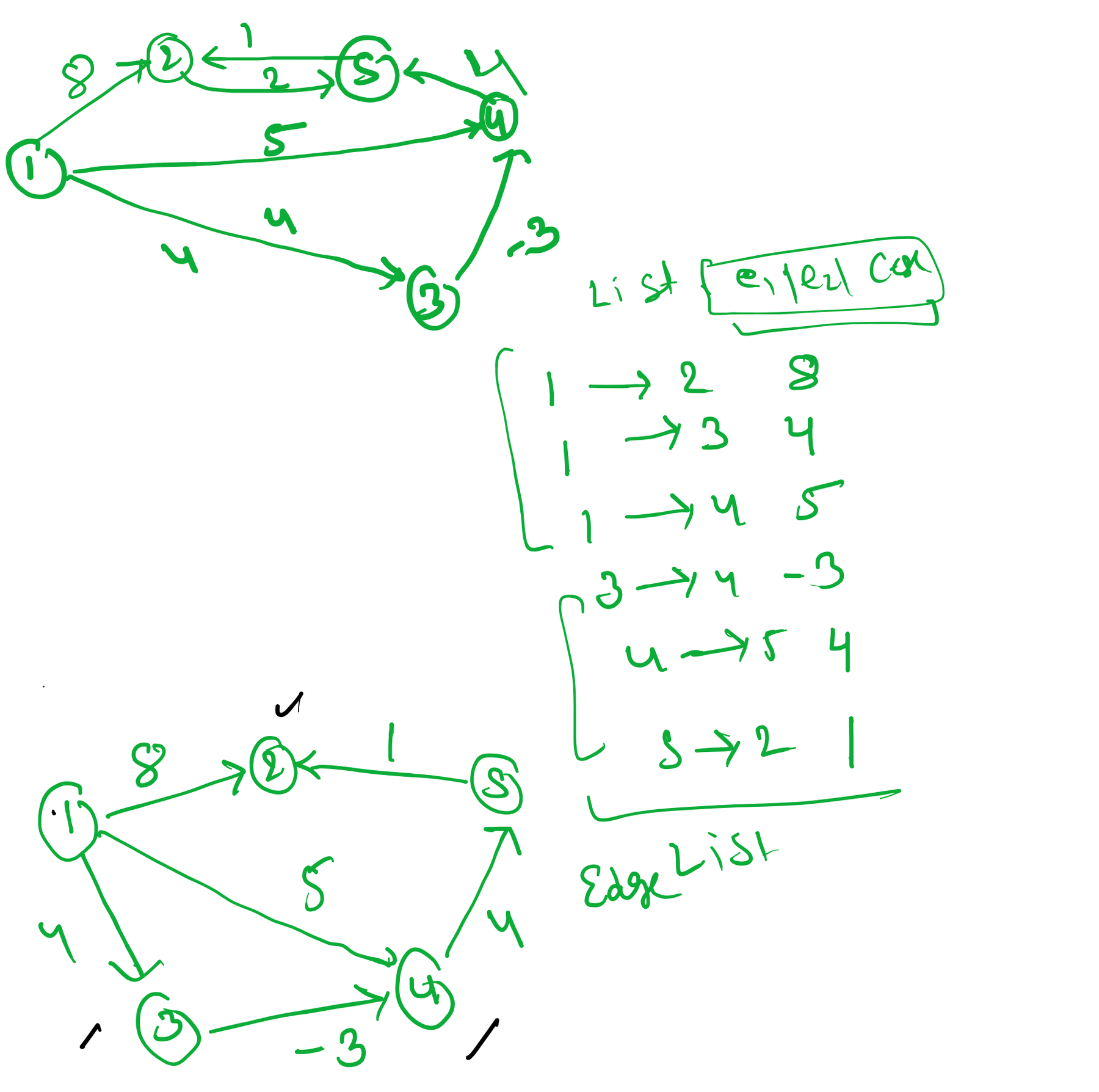
```java
public void Dijkstra(int src) {
        PriorityQueue<DijkstraPair> pq = new PriorityQueue<>();
        HashSet<Integer> visited = new HashSet<>();
        pq.add(new DijkstraPair(src, "" + src, 0));
        while (!pq.isEmpty()) {
            // 1. remove
            DijkstraPair rp = pq.poll();

            // 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));
                }
            }

        }
```

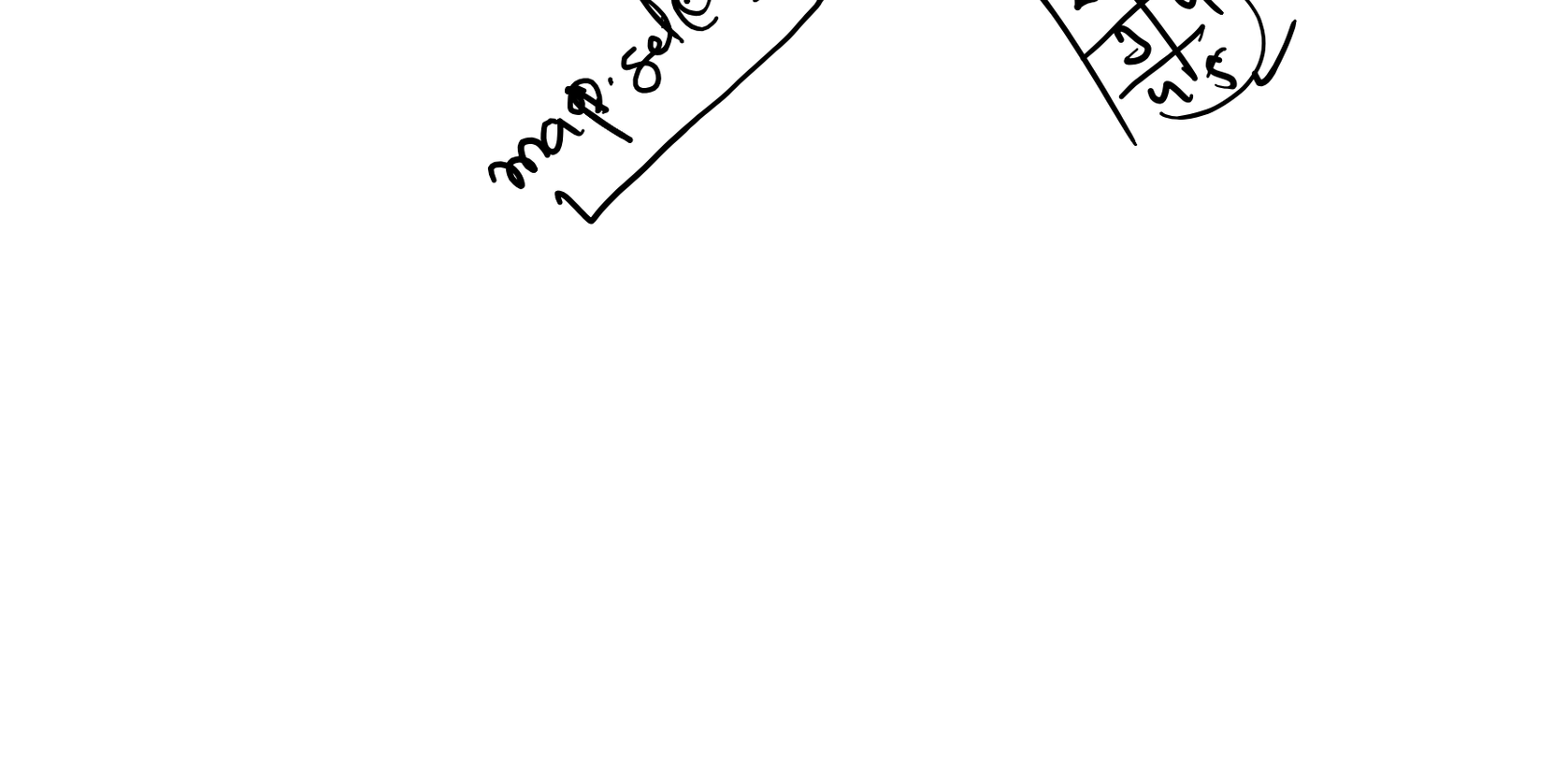


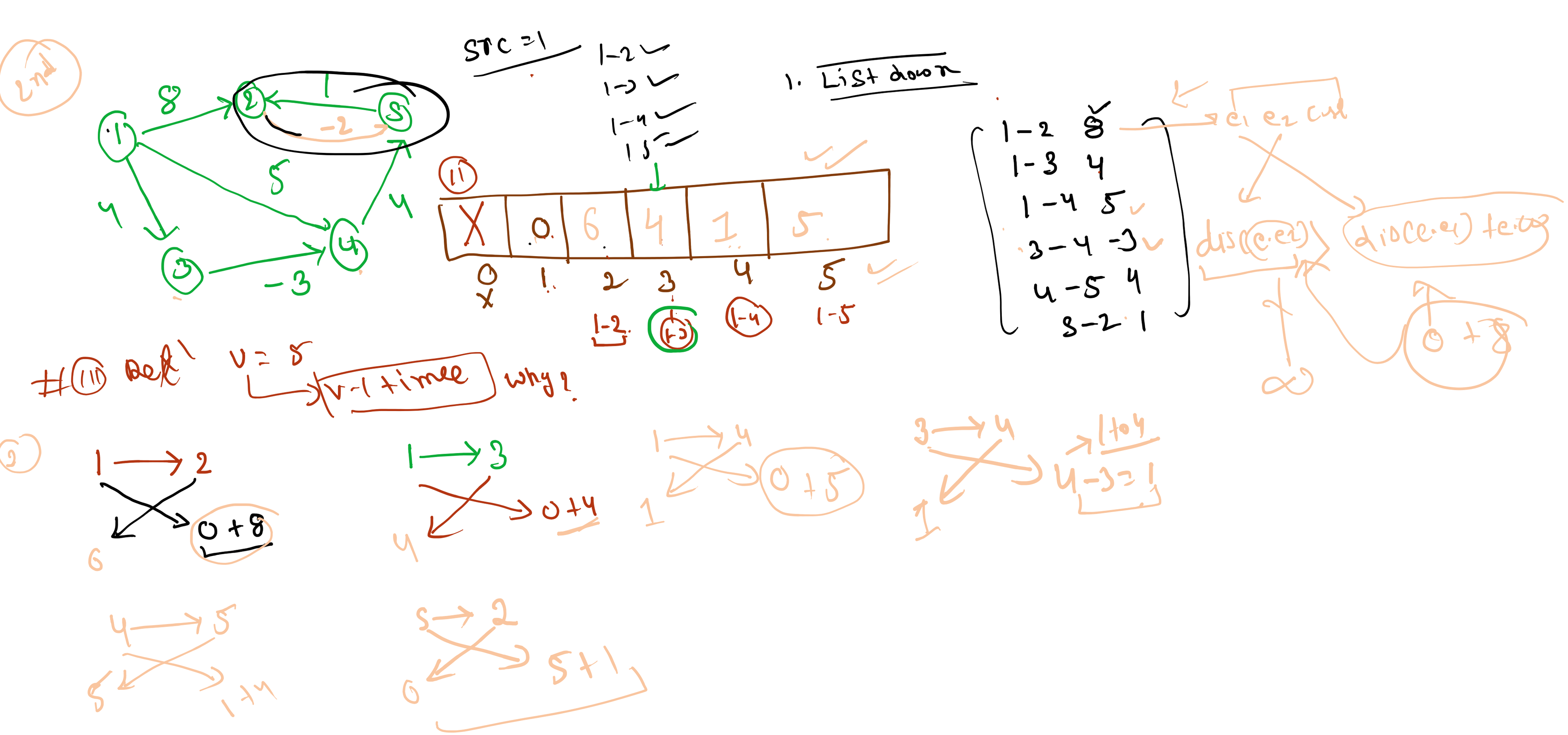




```java
public static void main(String[] args) {
        BellMan_Ford bf = new BellMan_Ford(5);
        bf.Addedge(1, 2, 8);
        bf.Addedge(1, 3, 4);
        bf.Addedge(1, 4, 5);
        bf.Addedge(3, 4, -3);
        bf.Addedge(4, 5, 4);
        bf.Addedge(5, 2, 1);

    }
```







```java
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]);
        }
    }
```