**SHORTEST PATH ALGORITHMS**

**Introduction:**

* We use these algorithms to solve shortest path problems.
* In general, Shortest path problem is a problem of finding the shortest path(s) between vertices or nodes in the given graph.
* **Shortest path** between two vertices is a path that has the least cost (weight /

distance) as compared to all other paths that exists in the graph.

**Applications:**

Shortest path algorithms have a wide range of applications such as in-

* Google Maps
* Road Networks
* Logistics Research

## Types of Shortest Path algorithms:

There are different types of shortest path algorithms

* Single-pair shortest path algorithm (A\* search algorithm)
* **Single-source shortest path algorithm (Dijkstra's algorithm)**
* Single-destination shortest path algorithm(Varient of Dijkstra's algorithm)
* All pairs shortest path algorithm (Floyd Warshall & Johnson's algorithm)

### Single-source shortest path algorithm:

* Single-source shortest path algorithm is a shortest path algorithm with which we find out the shortest paths from a given source vertex to all the other remaining vertices of the given graph.
* [Dijkstra’s Algorithm](https://www.gatevidyalay.com/dijkstras-algorithm-shortest-path-algorithm/) and Bellman Ford Algorithm are the famous algorithms used for solving single-source shortest path problems.

## Dijkstra’s Algorithm:

* It is used for solving the single source shortest path problem which gives the shortest paths from one particular source node to all the other nodes of the given graph.
* Conditions for Dijkstra’s Algorithm-

1. Dijkstra’s algorithm works only for those graphs that are connected.
2. Dijkstra’s algorithm works only for those graphs that do not contain any edges with negative weights.
3. Dijkstra’s algorithm works for directed as well as undirected graphs.

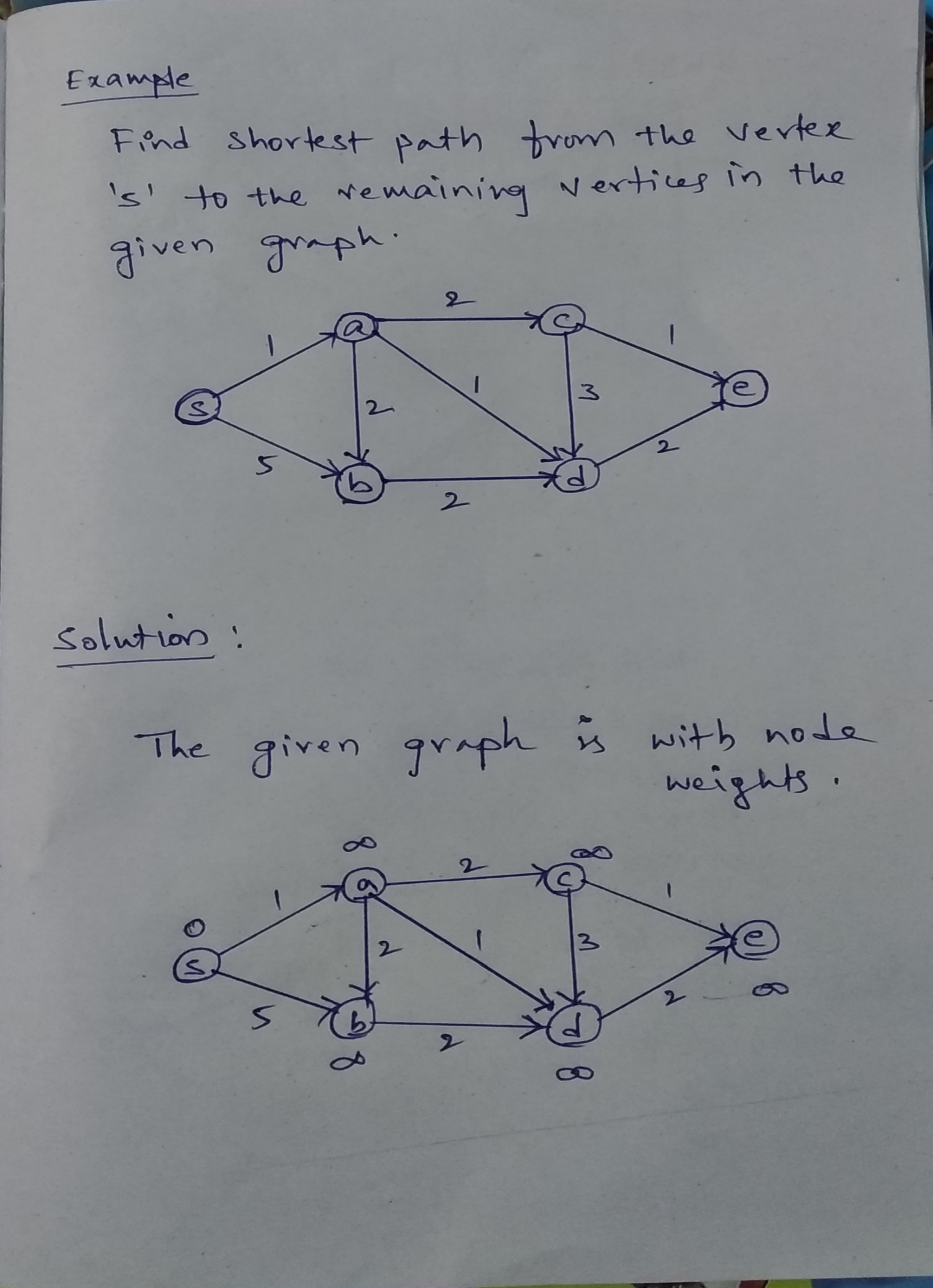
**Algorithm**

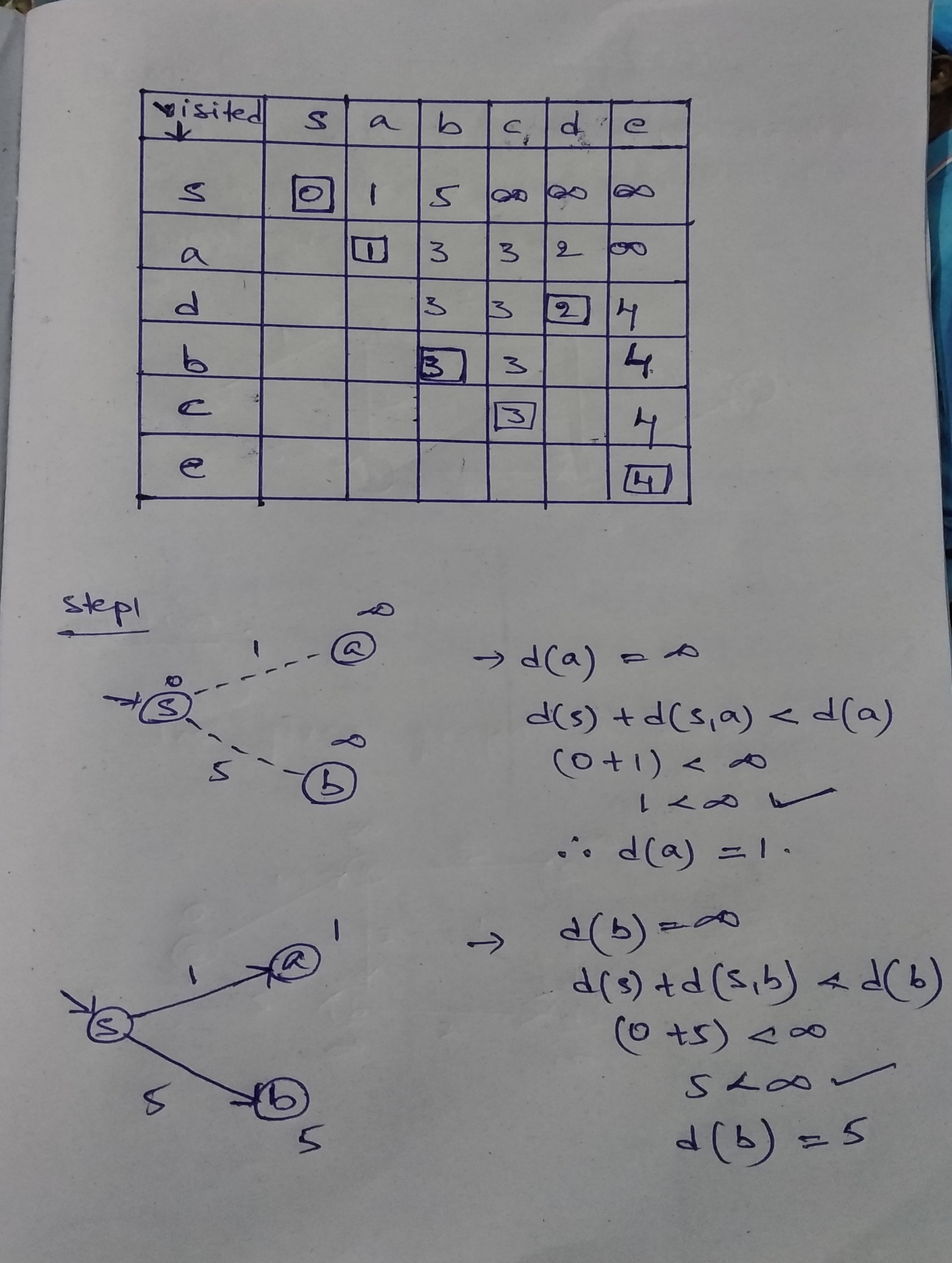
1. for each vertex v Є V // vertex set

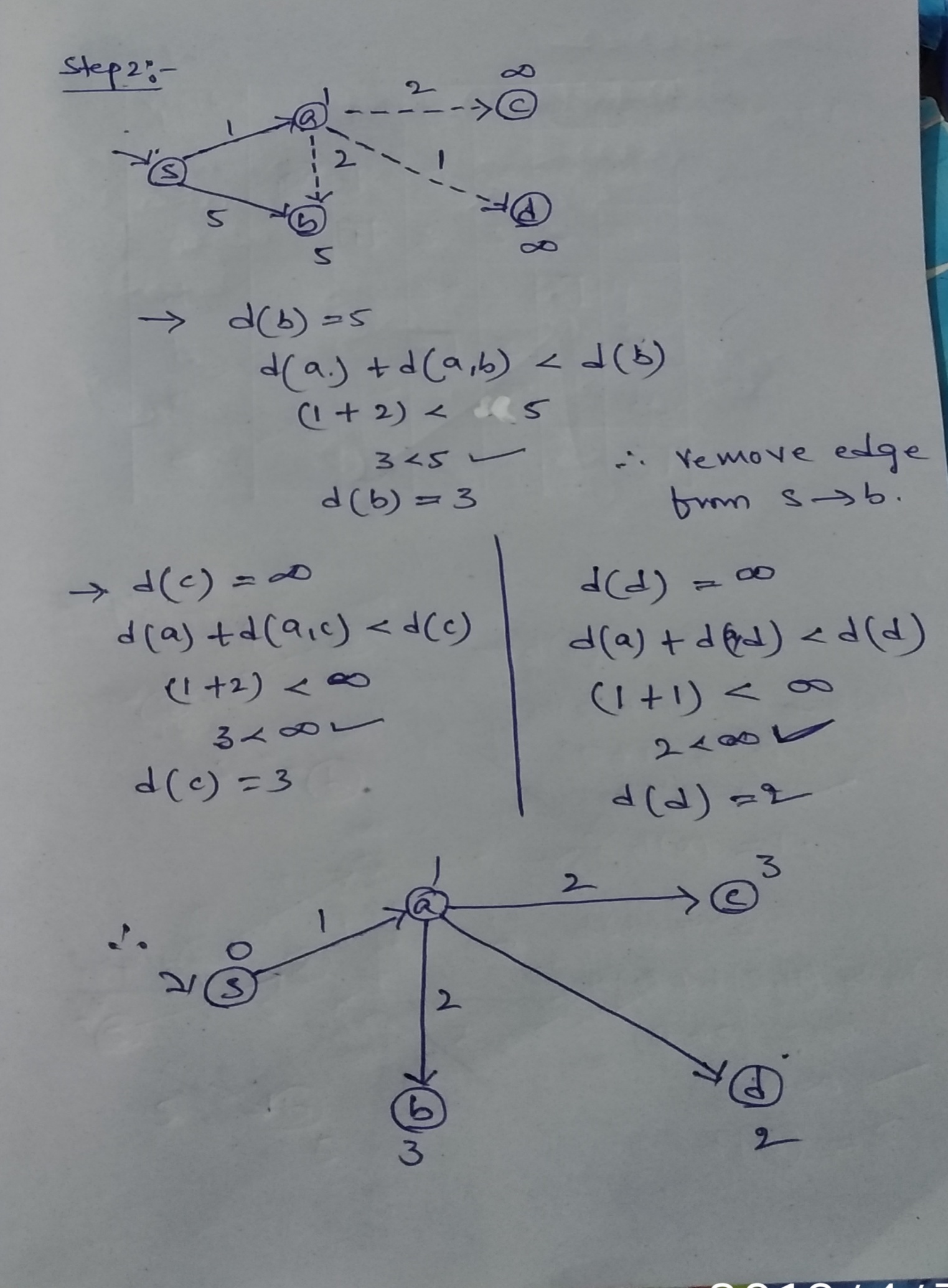
* dis(source) = 0 // distance of source is 0
* dis(v) = ∞ // distance of remaining vertexes is ∞

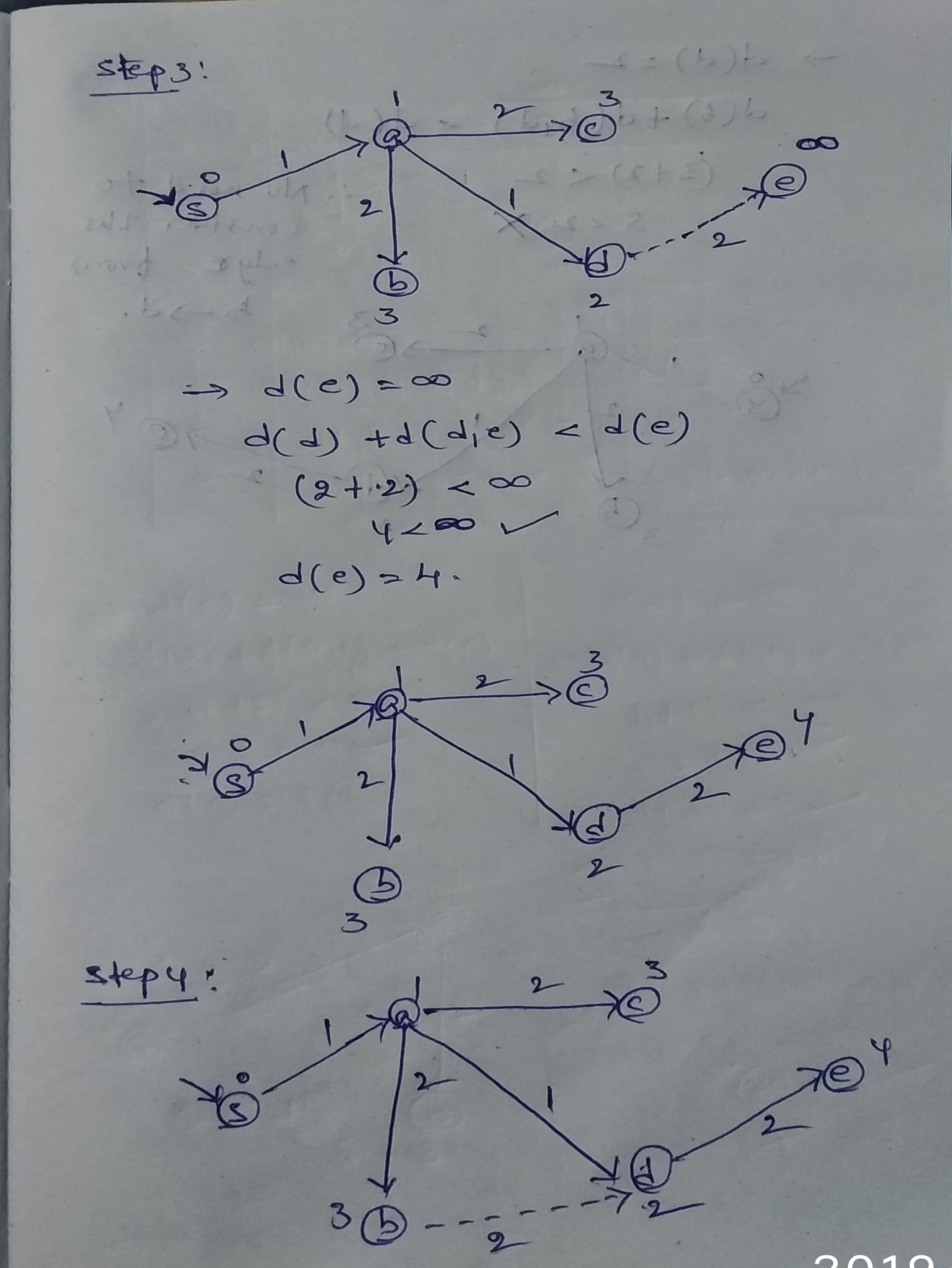
1. Q = |V| // vertex set
2. while Q ≠ Ф

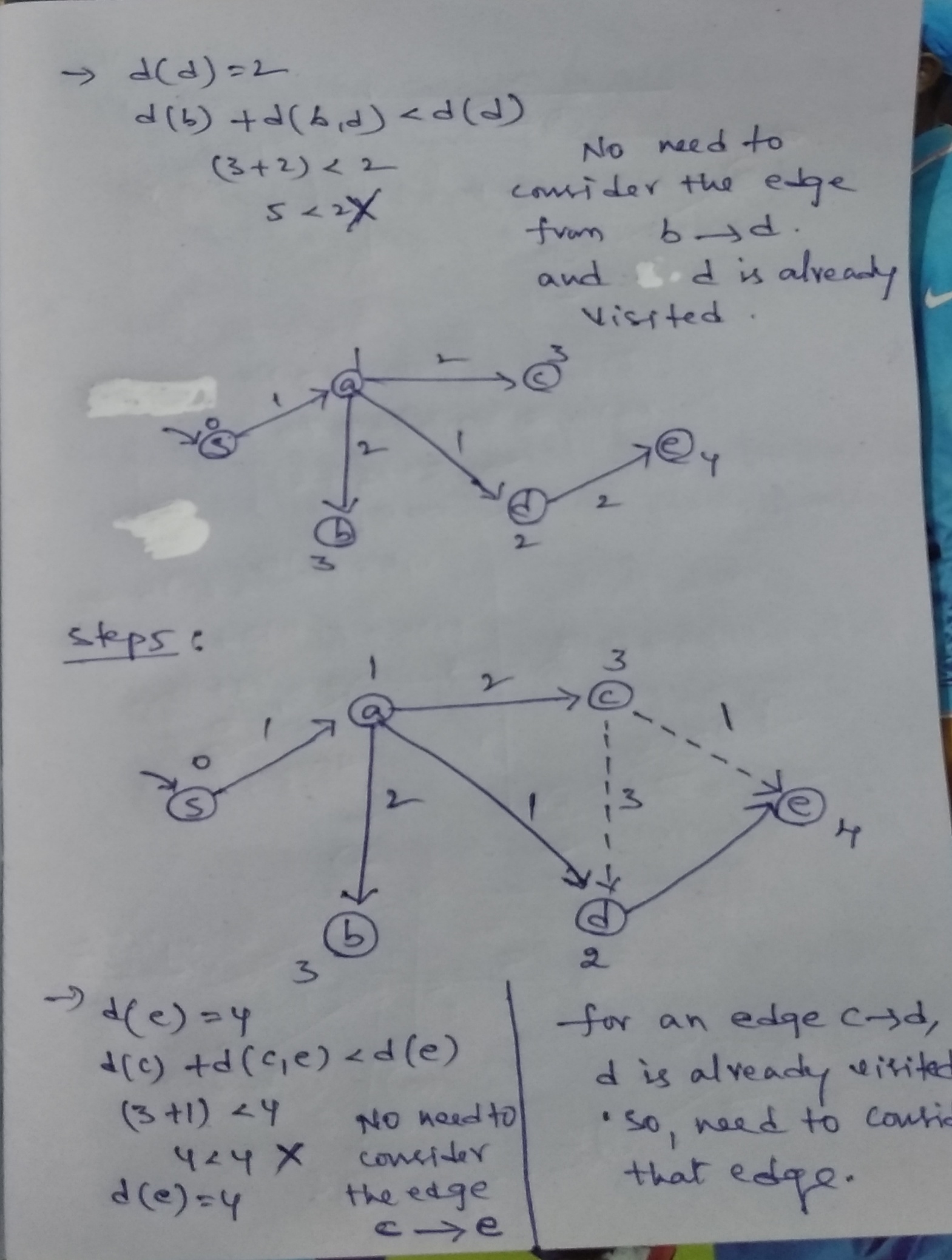
* u = Extract-Min (Q) // selecting minimum edge
* for each vertex v Є adj[u] //
  + if dis(u) + weight(u,v) < dis(v) then
    - dis(v) = dis(u) + weight(u, v)

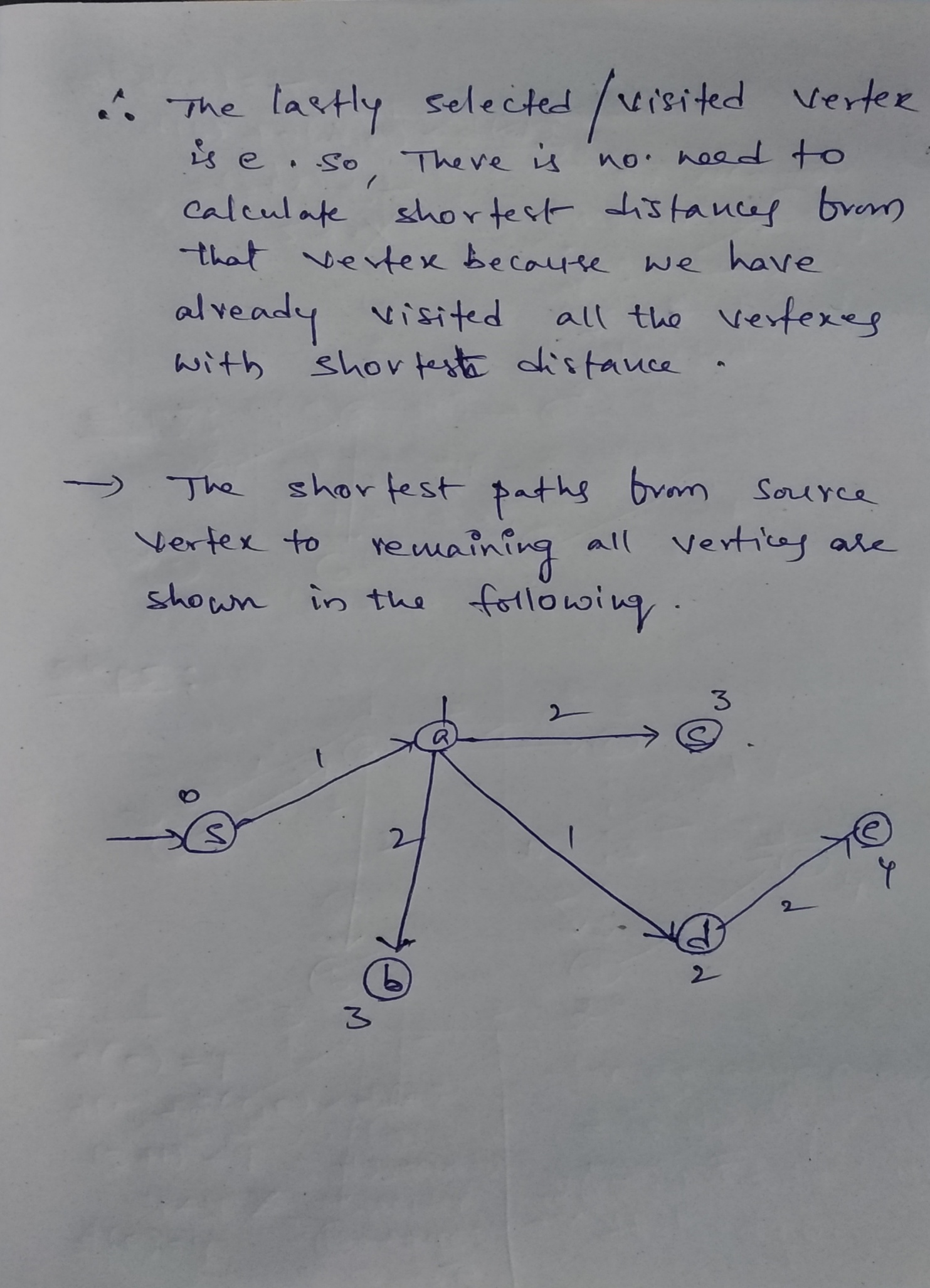












**Time Complexity:**

* Using Adjacency matrix representation is O(V^2).
* Using Adjacency list representation is O(ELogV)