```
BFS (startV) {
    startV.distance = 0
    queue.enqueue(startV)
    discoveredSet.add(startV)
    while (queue not empty) {
        curV = queue.dequeue()
        curV.visited = true
        for (adjV in adjacent vertices to curV) {
            if (adjV is not in discoveredSet) {
                queue.enqueue(adjV)
                discoveredSet.add(adjV)
                adjV.distance = curV.distance + 1
            }
        }
    }
}
```

```
DFS (startV) {
    stack.push(startV)
    while (stack is not empty){
        curV = stack.pop()
        if (curV is not in VisitedSet) {
            curV.visited = true
            visitedSet.add(curV)
            for (adjV in adjacent Vertices to curV){
                stack.push(adjV)
            }
        }
    }
}
```

```
Dijkstra ShortestPath(startV) {
    for (curV in allVertices) {
        curV.distance = ∞
        curV.pred = 0
        unvisitedQ.enqueue(curV)
    }
    startV.distance = 0
    while (unvisitedQ not empty) {
        curV = unvisitedQ.dequeue(minV from startV)
        for (adjV in adjacent to curV) {
            edgeW = weight of edge from curV to adjV
            altDistance = curV.distance + edgeW
            if (altDistance < adj.distance) {
                adjV.distance = altDistance
                adjV.pred = curV
            }
        }
    }
}
```