```
BFS(graph, start_vertex):
    let queue be a queue
    queue.enqueue(start_vertex)
    visited[start_vertex] = true
    while queue is not empty:
        vertex = queue.dequeue()
        for each neighbor in graph[vertex]:
            if not visited[neighbor]:
                queue.enqueue(neighbor)
                visited[neighbor] = true
```

```
DFS(graph, start_vertex):
    let stack be a stack
    stack.push(start_vertex)
    while stack is not empty:
        vertex = stack.pop()
        if vertex is not visited:
            visited[vertex] = true
            for each neighbor in graph[vertex]:
                if not visited[neighbor]:
                    stack.push(neighbor)
```

```
Dijkstra algorithm:
    for each vertex v:
        dist[v] = ∞
        prev[v] = none
    dist[source] = 0
    set all vertices to unexplored
    while destination not explored:
        v = least-valued unexplored vertex
        set v to explored
        for each edge (v,w)
            if dist[v] + len(v,w) < dist[w]:
                dist[w] = dist[v] + len(v,w)
                prev[w] = v
```

```
Bell-Ford Algorithm:
    For each vertex v
        dist[v] = ∞
        dist[source] = 0
    for i = 1 to |V|-1
        for each edge (u,v)
            if d[v] > d[u] + w(u,v)
                then d[v] = d[u] + w(u,v)
    For each edge (u,v)
        if d[v] > d[u] + w(u,v)
            then a negative-weight cycle exists
```