COMPLEXITY ANALYSIS

TERMINOLOGIES

GRAPHICAL REPRESENTATION:

The algorithm operates on a directed graph, often referred to as a flow network.

The graph consists of vertices (nodes) and directed edges connecting these vertices. Each edge has a capacity indicating the maximum flow that can pass through it.

The graph is typically represented using an adjacency list or an adjacency matrix.

RESIDUAL GRAPH:

The algorithm maintains a residual graph, which is derived from the original flow network.

The residual graph represents the remaining capacity in each edge after flow has been sent along some paths.

For each edge in the original graph, there may be a corresponding backward edge in the residual graph to allow for flow in the opposite direction.

BREADTH FIRST SEARCH (BFS):

The main loop of the algorithm performs a BFS traversal on the residual graph starting from the source node.

BFS explores vertices level by level, ensuring that the shortest path from the source to any vertex is found first.

During BFS, the algorithm marks visited vertices and tracks the path from the source to the sink.

AUGMENTING PATHS:

BFS identifies augmenting paths, which are paths from the source to the sink with available capacity.

An augmenting path allows additional flow to be sent from the source to the sink.

The algorithm finds the minimum capacity along the augmenting path, which determines the maximum flow that can be sent along that path.

UPDATING FLOW:

Once an augmenting path is found, the algorithm updates the flow along that path.

Flow is added along forward edges and subtracted along backward edges in the residual graph.

This update process ensures that the flow satisfies the capacity constraints of the original graph.

TERMINATION:

The algorithm terminates when no augmenting paths can be found from the source to the sink.

This indicates that the maximum flow has been achieved, and further iterations will not increase the flow.

ANALYSIS:

Breadth-First Search (BFS) in the Residual Graph: The main loop of the algorithm performs a BFS in the residual graph to find an augmenting path from the source to the sink. The BFS takes O(V + E) time, where V is the number of vertices and E is the number of edges in the graph.

Augmenting Path Update: After finding an augmenting path, the algorithm updates the flow along the path. Updating the flow takes O(V) time, where V is the number of vertices along the path.

Overall Complexity: In the worst case, the Edmonds-Karp algorithm iterates O(VE) times, where each iteration takes O(V + E) time for BFS and O(V) time for updating the flow. Hence, the overall time complexity is O(VE^2).

In terms of space complexity, the algorithm uses additional data structures to store capacities, flow, the graph, and other auxiliary information. These data structures typically require O(V^2) space.

CONCLUSION:

Time Complexity: Dominated by the BFS traversal, which takes O(V + E) time, where V is the number of vertices and E is the number of edges. In the worst case, O(VE) iterations may be required, resulting in a time complexity of O(VE^2).

Space Complexity: O(V^2) due to the storage of the graph and auxiliary data structures, such as the residual graph, flow, and capacities.

In summary, the Edmonds-Karp algorithm efficiently finds the maximum flow in a flow network by iteratively identifying augmenting paths using BFS, updating flow along these paths, and maintaining flow conservation. It terminates when no more augmenting paths can be found, achieving maximum flow.