Advanced Data Structures and Algorithms

Comprehensive Assignment Solutions

Arya Raag
Roll No: A125002
M.Tech (Computer Science and Engineering)

January 5, 2026

1 Graph Algorithms

Question 1. For finding an augmenting path in a graph, should Breadth First Search (BFS) or Depth First Search (DFS) be applied? Justify your answer.

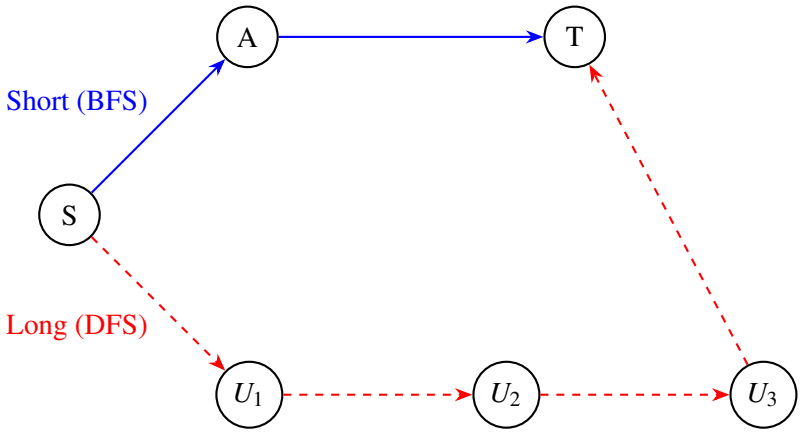
Detailed Solution:

Recommendation

Breadth First Search (BFS) should be applied. When BFS is used to find augmenting paths in the Ford-Fulkerson method, the specific implementation is known as the **Edmonds-Karp algorithm**.

Visual Comparison: BFS vs. DFS

The diagram below illustrates why BFS is superior. In the graph, DFS might explore the long path $S \to U_1 \to U_2 \to U_3 \to T$ first, whereas BFS will immediately find the direct path $S \to A \to T$.



Detailed Justification

1. **Shortest Path Guarantee:** BFS explores the graph layer by layer. Therefore, it is guaranteed to find the *shortest* augmenting path (in terms of the number of edges) from the source to the sink. DFS, on the other hand, might explore a very long, winding path before hitting the sink.
2. **Complexity Bound:** Using BFS guarantees a polynomial time complexity. It can be proven that if we always choose the shortest augmenting path, the length of the shortest path increases monotonically over time.
   - The total number of augmentations is bounded by $O(VE)$.

- Each BFS takes $O(E)$.
- Total Complexity: $O(VE^2)$.

3. **Avoiding Pathological Cases:** If DFS is used (standard Ford-Fulkerson), the algorithm depends heavily on edge capacities. In graphs with large integer capacities, DFS might choose paths that reduce flow bottlenecks by only 1 unit at a time. This can lead to pseudo-polynomial time complexity (e.g., proportional to the max capacity C), which is inefficient.

Conclusion

BFS is strictly superior to DFS for the Ford-Fulkerson method because it guarantees the shortest augmenting path property. This property transforms the algorithm from one with potentially exponential (or pseudo-polynomial) time complexity into the strongly polynomial Edmonds-Karp algorithm ($O(VE^2)$), ensuring efficient performance even for networks with large capacities.
