MTech CSE – 1st Semester
Student ID: **A125023**
Student Name: **SRIJITA VERMA**

I I I T Bhubaneswar
Imagine, Innovate, Inspire

Question

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

Answer

Short Answer

Breadth First Search (BFS) should be applied when finding augmenting paths in the context of network flow algorithms, particularly in the Edmonds–Karp algorithm.

Although Depth First Search (DFS) can be used to find augmenting paths, BFS is preferred because it guarantees shortest augmenting paths, ensures polynomial-time complexity, and avoids pathological performance cases.

Background: Augmenting Paths in Flow Networks

An augmenting path is a path from the source to the sink in the residual graph along which additional flow can be pushed.

Augmenting paths are central to:

- the Ford–Fulkerson method,
- the Edmonds–Karp algorithm,
- maximum flow computation.

The choice of how augmenting paths are found (BFS vs. DFS) directly affects correctness guarantees and time complexity.

BFS vs DFS: Conceptual Comparison

Depth First Search (DFS)

- Explores one path as deeply as possible before backtracking.
- May find very long or inefficient augmenting paths.

- The choice of path depends heavily on traversal order.
- Does not guarantee the shortest augmenting path.

Breadth First Search (BFS)

- Explores paths in increasing order of length (number of edges).
- Always finds the shortest augmenting path.
- Uses systematic, level-by-level exploration.
- Exhibits predictable and stable behavior.

Why BFS Is Preferred for Finding Augmenting Paths

1. Shortest Augmenting Path Property

When BFS is used on the residual graph, the first time the sink is reached, the path found has the minimum number of edges. Shorter augmenting paths reduce the number of augmentations required.

This property is essential in the Edmonds–Karp algorithm.

2. Guaranteed Polynomial-Time Complexity

Using DFS in the Ford–Fulkerson method can lead to exponential time complexity in the worst case. In contrast, using BFS guarantees a polynomial-time bound.

Specifically, the Edmonds–Karp algorithm runs in:

$$O(VE^2)$$

time, where V is the number of vertices and E is the number of edges.

3. Avoidance of Bad Augmenting Paths

DFS may:

- repeatedly push small amounts of flow,
- undo previous flow decisions,
- become trapped in inefficient augmentations.

BFS avoids these issues by:

- making steady progress using shortest paths,
- ensuring that the distance from source to sink never decreases.

4. Theoretical Guarantee of Termination

With BFS:

- each edge can become critical only a bounded number of times,
- the level of the sink strictly increases after enough augmentations.

This provides a provable upper bound on the number of iterations.

Illustrative Example: BFS vs DFS for Augmenting Paths

Consider the following flow network:

$$s \to v_1 \to v_2 \to t$$
$$s \to v_3 \to t$$

Assume all edges have unit capacity.

Using DFS

A depth-first search may discover the augmenting path:

$$s \to v_1 \to v_2 \to t$$

which has length 3.

After pushing one unit of flow, DFS may repeatedly explore long paths or undo previous choices before discovering alternative shorter routes.

Using BFS

Breadth-first search explores the graph level by level and finds the augmenting path:

$$s \to v_3 \to t$$

which has length 2.

This is the shortest augmenting path in terms of number of edges and leads to faster convergence.

Observation

Although both searches find valid augmenting paths, BFS consistently selects shorter paths, whereas DFS may choose longer and inefficient paths depending on the traversal order.

Formal Justification (Edmonds–Karp Insight)

In the Edmonds–Karp algorithm:

- BFS is used to find augmenting paths,
- each BFS takes $O(O(E))$ time,
- the number of augmentations is $O(VE)$.

Hence, the total running time is:

$$O(VE^2).$$

This guarantee does not hold if DFS is used instead.

Can DFS Ever Be Used?

Yes, DFS can be used:

- in the basic Ford–Fulkerson algorithm,
- for small graphs,
- when capacities are integers and inputs are well-behaved.

However:

- it offers no worst-case time guarantees,
- it is unsuitable for large or adversarial inputs.

Relation to Ford–Fulkerson and Edmonds–Karp Algorithms

The choice between BFS and DFS directly distinguishes the Ford–Fulkerson and Edmonds–Karp algorithms.

Ford–Fulkerson Method

The Ford–Fulkerson method allows augmenting paths to be found using *any* graph traversal strategy, including DFS.

As a result:

- the algorithm is correct,
- but the running time depends on the choice of augmenting paths,
- and in the worst case, it may take exponential time.

Edmonds–Karp Algorithm

The Edmonds–Karp algorithm is a specific implementation of Ford–Fulkerson that uses BFS to find augmenting paths.

By always selecting the shortest augmenting path:

- the number of augmentations is bounded,
- the distance from source to sink never decreases,
- the algorithm runs in polynomial time.

Specifically, Edmonds–Karp runs in:

$$O(VE^2)$$

time.

Key Distinction

Thus, while DFS can be used in Ford–Fulkerson, BFS is essential in Edmonds–Karp to guarantee efficiency and predictable performance.

Final Conclusion

Breadth First Search (BFS) should be used to find augmenting paths.

Reasons:

- BFS finds the shortest augmenting paths,
- guarantees polynomial-time performance,
- prevents inefficient or cyclic augmentations,
- forms the basis of the Edmonds–Karp algorithm.

Practical Consideration. In practice, BFS-based augmenting path selection leads to more stable flow growth and fewer iterations, making it suitable for large networks.

Intuition

Augmenting flow is like sending water through pipes. BFS finds the shortest route to the sink. Shorter routes fill faster and stabilize sooner, whereas DFS may wander unnecessarily and slow convergence.