Final Lab Report

Topic: Analysis of the Ford-Fulkerson Algorithm in Solving the Maximum Flow
Course name: Data Structure and Algorithm-2 Laboratory
Course Code: CSE 2218
Section: G

Submitted to: Charles Aunkan Gomes
Lecturer, Department of CSE
Submitted by: Group-4
Mania Sultana Id 011212038
Taspia Akter Epou Id 011212163
Sumaiya Islam Ety Id 011212164
Joyasree   Bardhan Id 011221189
Date :28-4-24

Introduction

Introduce the concept of the maximum flow problem, highlighting its relevance in network optimization. Discuss its applications in various fields such as transportation, communication networks, and resource management. This section sets the stage for the detailed analysis of solving methods by presenting the basic definitions and importance of network flow optimization.

Theoretical Background

Delve into the mathematical formulation of the flow network. Define terms like capacity, flow, source, and sink. Explain the conservation of flow and the capacity constraint in a comprehensible manner to ensure readers without a deep mathematical background can follow.

Why does this problem solve computer science?

Efficiently computing the maximum flow in a network is essential for optimising resource utilisation, minimising congestion, and improving overall network performance. It forms the backbone of various algorithms and applications in network design, routing protocols, and logistics management.

Brute Force Approach:

Brute Force is a straightforward method used in algorithmic problem-solving that checks every possible solution until the correct one is found. Brute Force Algorithms function by searching each element sequentially until the desired result is found or all options are exhausted.

Brute Force Approach to the Maximum Flow Problem:

The brute force method for solving the maximum flow problem involves generating and evaluating every possible flow configuration in a network to identify the one that maximises the flow from the source to the sink. This method calculates the flow for every possible combination of flow values on all network edges, ensuring that none of these values exceed the capacities of their respective edges.

Steps of the Brute Force Method:

Enumerate all feasible flows: Generate every combination of flows on all edges under the constraints of edge capacities.

Check flow conservation: Ensure that each flow configuration adheres to the conservation of flow principle at every node except the source and sink.

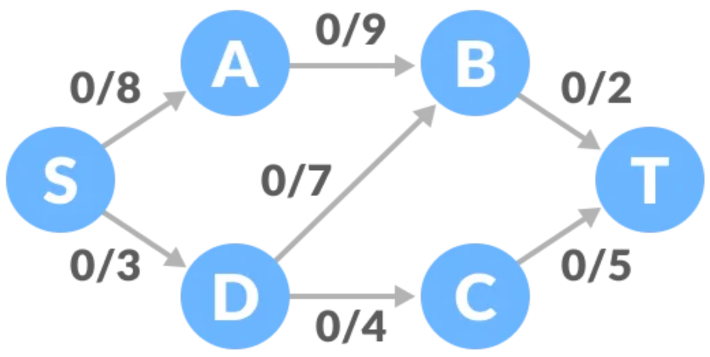
Identify maximum flow: From the set of feasible flows that satisfy the conservation laws, identify the configuration where the flow at the sink node is maximised.

Simulation :

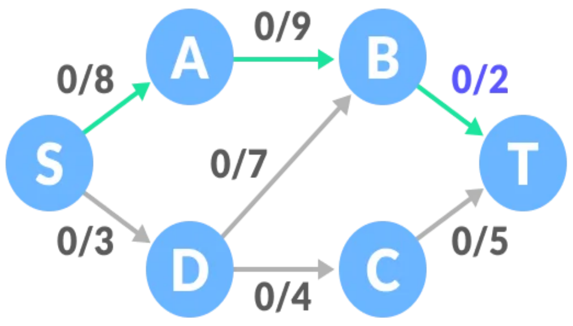
1.Initialise the flow in all the edges to 0.

2.While there is an augmenting path between the source and the sink, add this path to the flow.

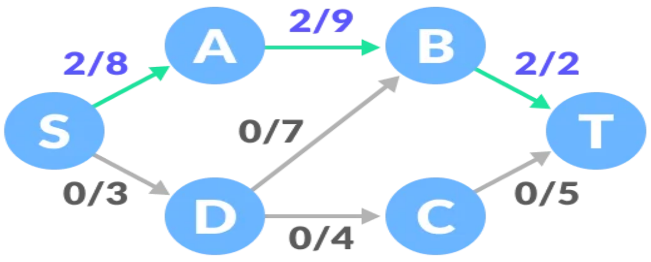
3.Update the residual graph.



1.Select any arbitrary path from S to T. In this step, we have selected path S-A-B-T

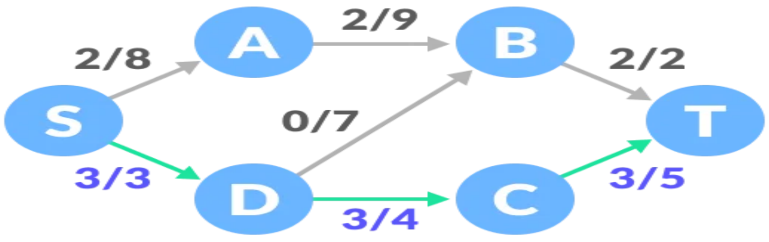


The minimum capacity among the three edges is 2 (B-T). Based on this, update the flow/capacity for each path.

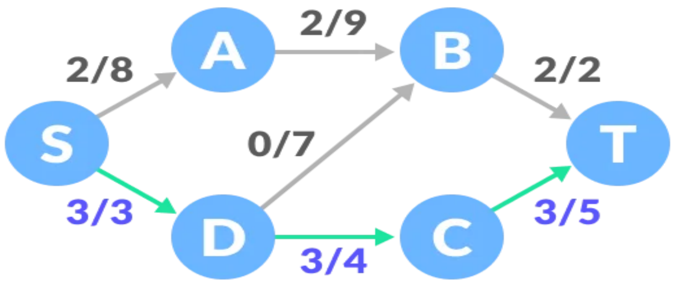


2.The minimum capacity among the three edges is 2 (B-T). Based on this, update the flow/capacity for each path.

Select another path S-D-C-T. The minimum capacity among these edges is 3 (S-D).



Update the capacities according to this



Pseudocode:

```
flow = 0
for each edge (u, v) in G:
    flow(u, v) = 0
while there is a path, p, from s -> t in residual network G_f:
    residual_capacity(p) = min(residual_capacity(u, v) : for (u, v) in p)
    flow = flow + residual_capacity(p)
    for each edge (u, v) in p:
        if (u, v) is a forward edge:
            flow(u, v) = flow(u, v) + residual_capacity(p)
        else:   flow(u, v) = flow(u, v) - residual_capacity(p)
return flow
```

<u>**Differences between Brute force and fulkerson algorithm:**</u>

<u>**Brute Force Algorithm;**</u>

<u>**Approach:**</u>

This method entails generating and evaluating every possible flow configuration within the network. It systematically calculates the flow for every possible combination of flow values on all edges, ensuring that they adhere to capacity constraints.

<u>**Evaluation:**</u> Each flow configuration must also satisfy the flow conservation principle at every node (excluding the source and sink). The maximum flow is determined by selecting the configuration where the total flow reaching the sink is maximised.

<u>**Ford-Fulkerson Algorithm:**</u>

<u>**Approach:**</u>

Rather than evaluating all possible configurations, the Ford-Fulkerson method iteratively finds paths from the source to the sink where additional flow can be pushed.These paths are known as augmenting paths.

<u>**Evaluation:**</u> The flow is increased along these paths based on the smallest capacity (bottleneck capacity) remaining on the path until no more augmenting paths are found, which means that the maximum flow is achieved.

Efficiency and Complexity:

Brute Force Approach:

Complexity: The time complexity of the brute force approach can be as bad as O((C+1)^E),where C,C is the maximum capacity on any edge, and E,E is the number of edges. This exponential growth in complexity makes it impractical except for very small networks.

Efficiency: Brute force is highly inefficient for larger networks due to the sheer number of potential configurations that need to be examined.

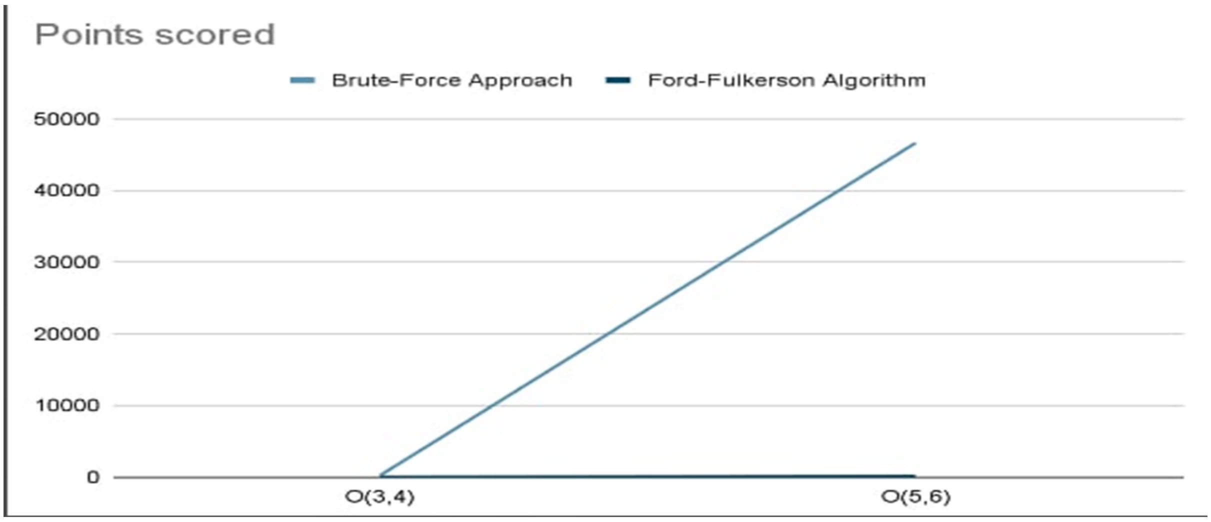
Ford-Fulkerson Algorithm:

Complexity: The efficiency of Ford-Fulkerson depends on how the augmenting paths are found. In the worst-case scenario using simple depth-first search, the time complexity can be$O(E\times f)$

O(E×f), where f,f is the maximum flow in the network. However, if the Edmonds-Karp implementation is used, which employs breadth-first search to find augmenting paths, the complexity becomes $O(V\times E$^2), where V,V is the number of vertices

Efficiency: Ford-Fulkerson is more efficient as it iteratively improves the flow, often requiring significantly fewer evaluations compared to brute force. The use of augmenting paths ensures that each increment operation is optimal and contributes directly to increasing the overall flow.

Brute Force and fulkerson algorithm time complexity graph:



Conclusion:

In summary, while the brute force approach guarantees finding a solution by exhaustively checking all possibilities, its use is limited due to extreme inefficiencies and impractical time complexities for larger networks. In contrast, the Ford-Fulkerson algorithm provides a dynamic and efficient method for increasing network flow iteratively using augmenting paths, making it more suitable for practical applications. This adaptability, coupled with more manageable computational demands, makes Ford-Fulkerson the preferred approach in most scenarios involving the maximum flow problem.