جامعة أم القرى
UMM AL-QURA UNIVERSITY

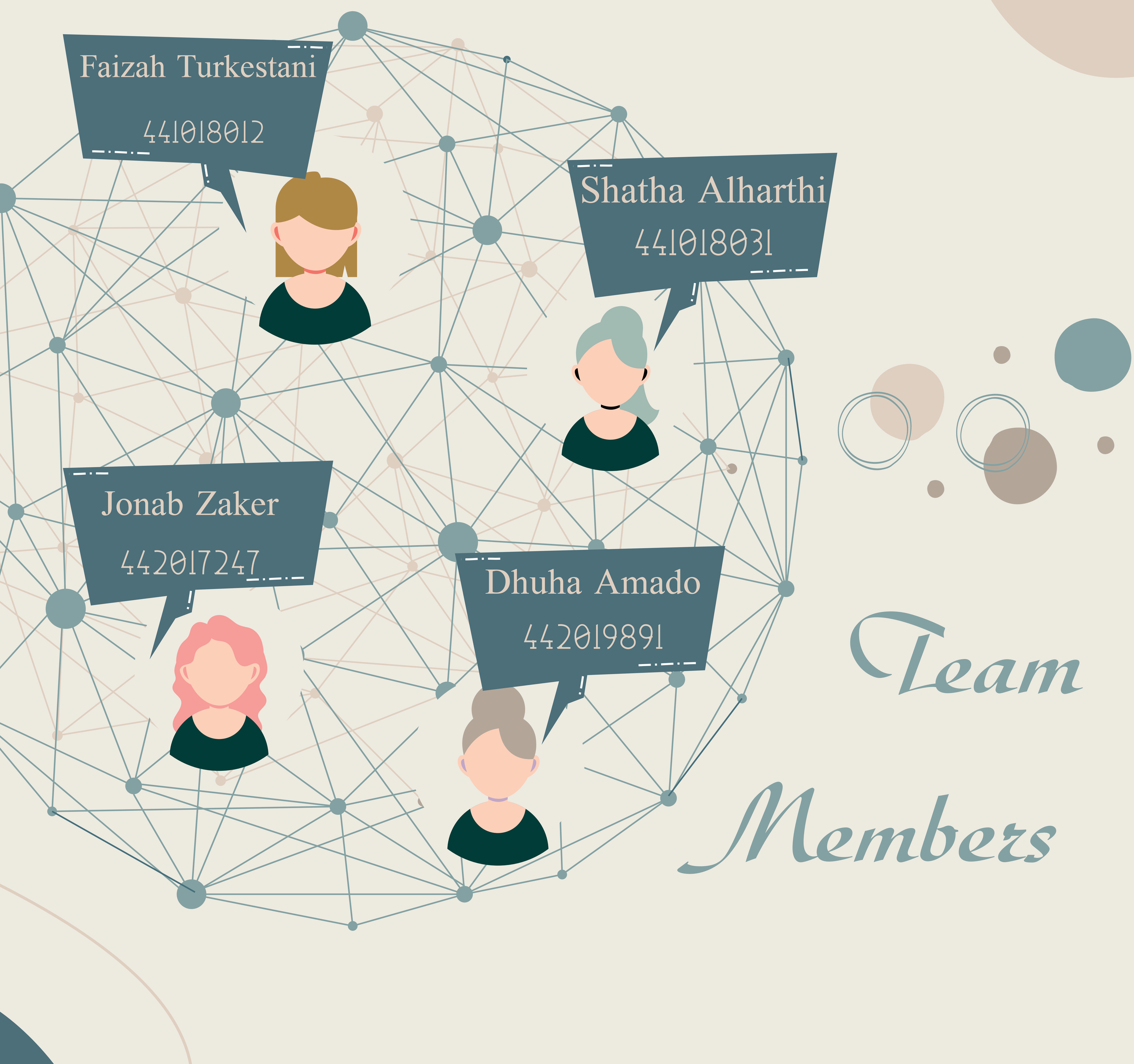
Algorithms Programming

Network

Flow

Project Report

Instructor : Areej Al-sini



Team Members

Networ Flow

is a directed graph where each edge has a capacity and each edge receives a flow. The amount of flow on an edge cannot exceed the capacity of the edge. Often in operations research, a directed graph is called a network. A flow must satisfy the restriction that the amount of flow into a node equals the amount of flow out of it, unless it is a source, which has only outgoing flow, or sink, which has only incoming flow. A network can be used to model traffic in a computer network or anything similar in which something travels through a network of nodes.

DINIC'S ALGORITHM

EDMONDS' ALGORITHM

FORD-FULKERSON ALGORITHM

Dinic's Algorithm :

is a strongly polynomial algorithm for computing the maximum flow in a flow network, The algorithm runs in O(V^{2}E) time . in that it uses .shortest augmenting paths

The data structures used are :

- GRAPH
- ADJACENCY LIST
- ADJACENCY MATRIX

Dinic's Algorithm

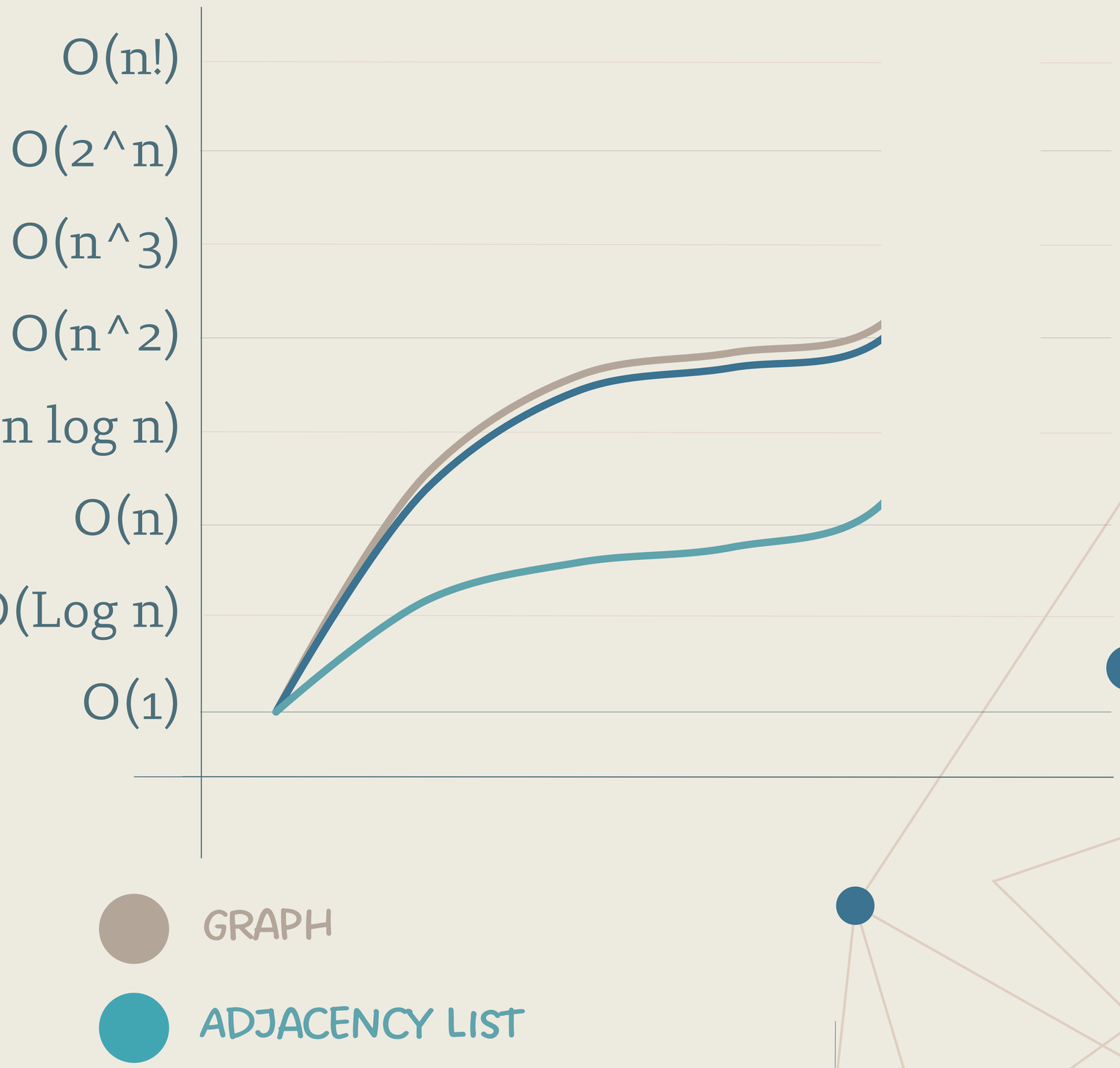
Pseudocode

```
DINIC(SOURCE, SINK, ADJACENCY_LIST)
    INITIALIZE THE LEVEL GRAPH AND THE RESIDUAL GRAPH
    WHILE BFS(SOURCE, SINK)
        WHILE THERE IS AN AUGMENTING PATH FROM SOURCE TO SINK IN THE RESIDUAL GRAPH
            CALCULATE THE BOTTLENECK CAPACITY OF THE PATH
            AUGMENT THE FLOW ALONG THE PATH BY THE BOTTLENECK CAPACITY
    RETURN THE FLOW IN THE RESIDUAL GRAPH

BFS(SOURCE, SINK)
    INITIALIZE THE LEVEL GRAPH WITH -1 FOR ALL NODES EXCEPT THE SOURCE NODE, WHICH HAS LEVEL 0
    INITIALIZE A QUEUE AND ADD THE SOURCE NODE TO IT
    WHILE THE QUEUE IS NOT EMPTY
        DEQUEUE A NODE U FROM THE QUEUE
        FOR EACH ADJACENT NODE V OF U
            IF THE LEVEL OF V IS -1 AND THERE IS AVAILABLE CAPACITY IN THE RESIDUAL GRAPH FROM U TO V
                SET THE LEVEL OF V TO THE LEVEL OF U + 1
                ADD V TO THE QUEUE
            END IF
        END FOR
    END WHILE
    RETURN THE LEVEL OF SINK NODE IS NOT -1
```

Dinic's Algorithm

Discussion

As we observation that arraylist is the best data structure for this algorithm because it takes O(n) while array matrix and graph take o(n^2)



Edmonds-Karp algorithm

The Edmonds-Karp Algorithm is a specific implementation of the Ford-Fulkerson algorithm. Like Ford-Fulkerson, Edmonds-Karp is also an algorithm that deals with the max-flow min-cut problem.

is an algorithm for finding a spanning arborescence of minimum weight (sometimes called an optimum branching). It is the directed analog of the minimum spanning tree problem.

The data structures used are :

- STACK & 2D ARRAY
- QUEUE & GRAPH
- QUEUE & 2D ARRAY

نص فقرتك

Edmonds-karp algorithm

Pseudocode

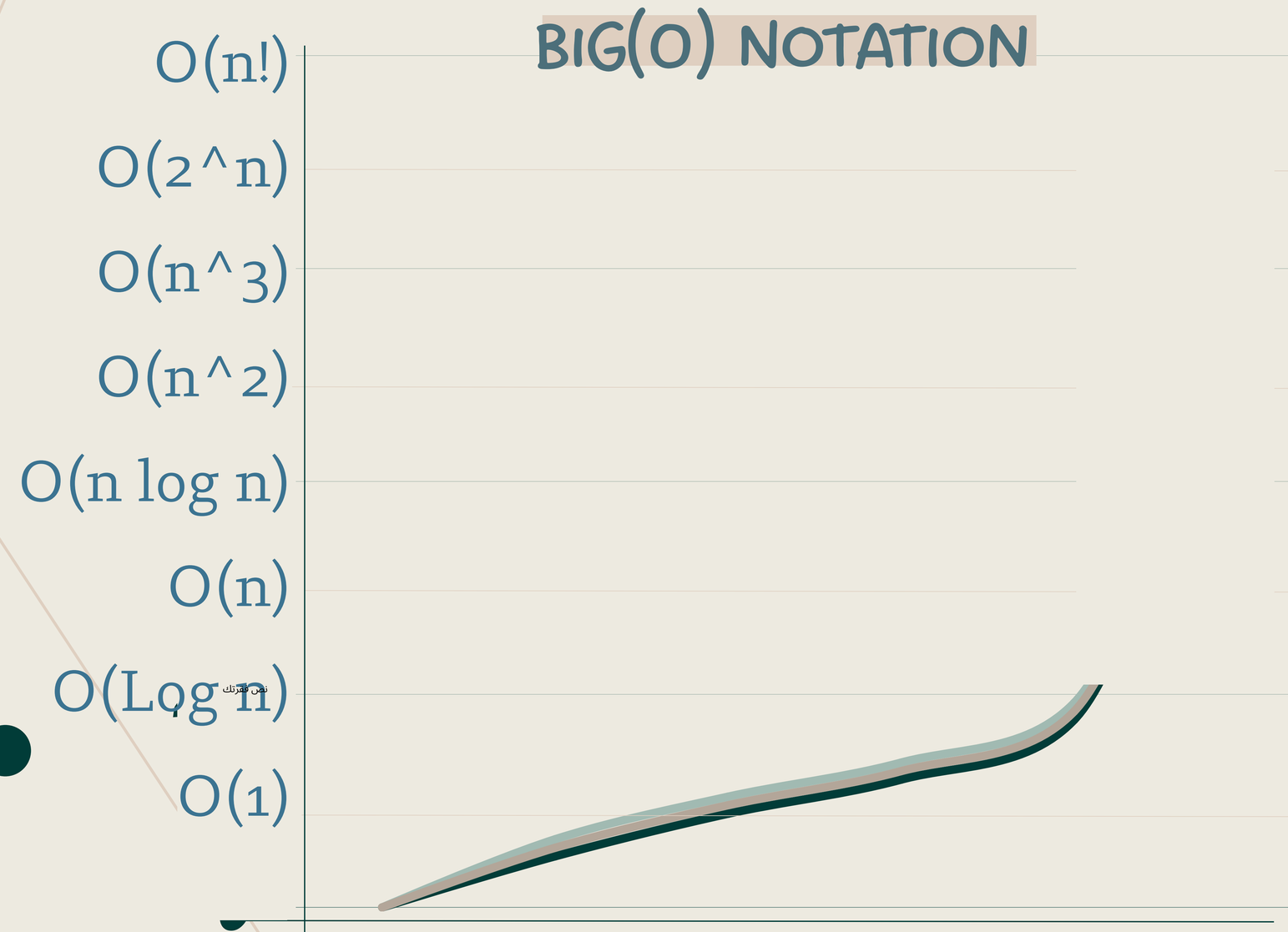
```
INPUT
    C[N x N] : CAPACITY MATRIX
    E[N x N] : ADJACENCY MATRIX
    S : SOURCE
    T : SINK
OUTPUT
    F : MAXIMUM FLOW
EDMONDS-KARP:
    F = 0
    F = [N x N]
    WHILE TRUE:
M, P = BREADTH-FIRST-SEARCH(C, E, S, T, F)
IF M = 0:
    BREAK
F = F + M
V = T
WHILE V != S:
    U = P[V]
    F[U, V] = F[U, V] - M
    F[V, U] = F[V, U] + M
    V = U
RETURN
```

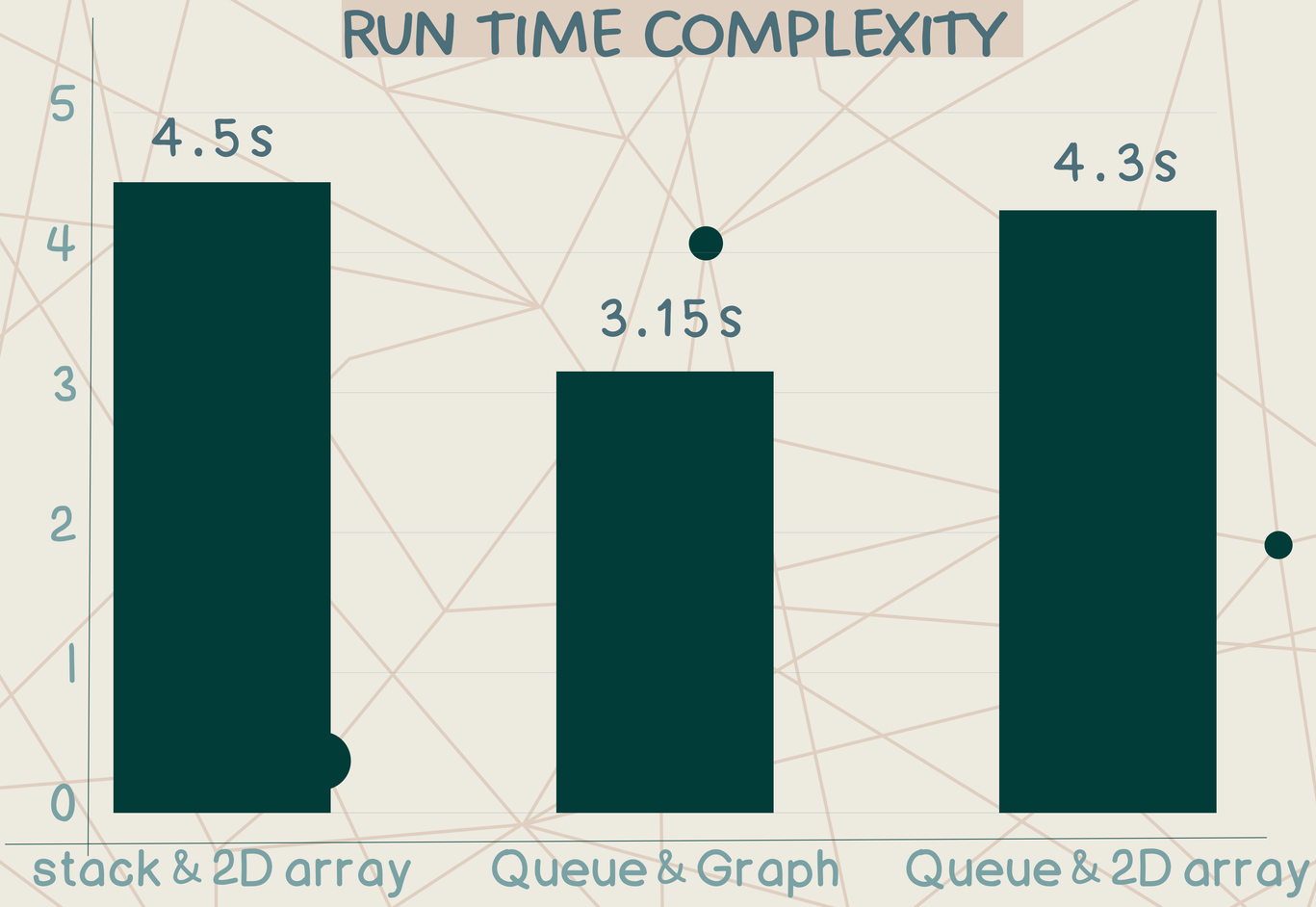
نص فقرتك

Edmonds’-karp algorithm

Discussion

We observation that all data structures take the same Big O, but (Queue with Graph) was the best of them of Runtime Complexity and we concluded that by reading 124675 data.





Ford_Fulkerson algorithm

is a greedy algorithm that computes the maximum flow in a flow network.

The idea behind the algorithm is as follows: as long as there is a path from the source (start node) to the sink (end node), with available capacity on all edges in the path, we send flow along one of the .paths. Then we find another path, and so on

The data structures used are :

- ADJACENCY LIST
- ADJACENCY GRAPH
- ADJACENCY ARRAY

Ford_Fulkerson algorithm

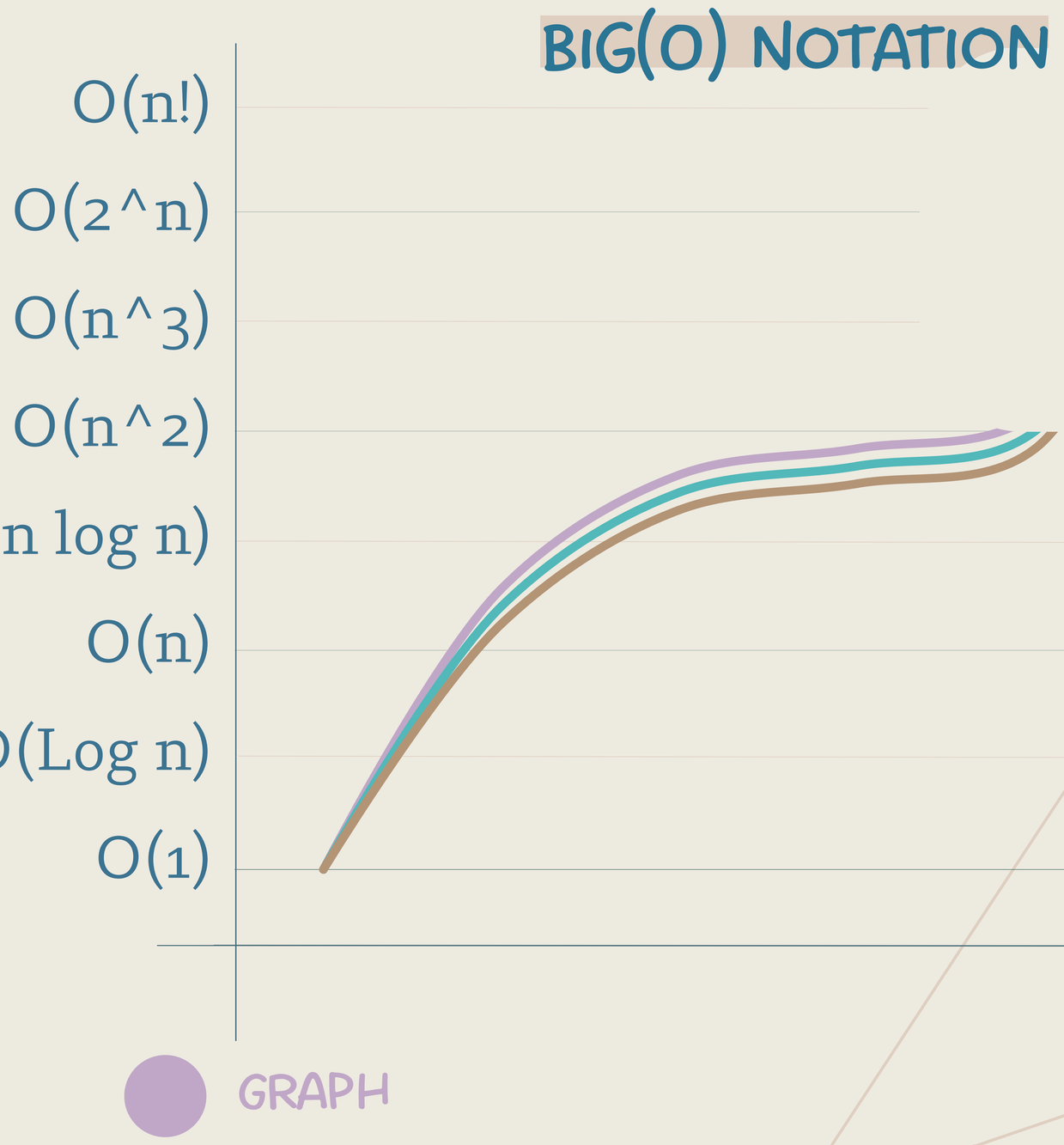
Pseudocode

```
FORD_FULKERSON(G,S,T)

FOR EACH EDGE(U,V)∈E[G]DO

    F[U,V] <— 0

    F[V,U] <— 0

WHILE THERE EXISTS A PATH P FROM S TO T IN THE RESIDUAL NETWORK GF DO

    CF(P) <— MIN{ CF(U,V) : (U,V) IS IN P }

    FOR EACH EDGE(U,V) IN P DO

        F[U,V] <— F[U,V]+CF(P)

        F[V,U] <— -F[U,V]

    END FOR

    REBUILD G BASED ON NEW FLOW F

END WHILE
```

Ford_Fulkerson algorithm

Discussion

We also observation that all data structures take the same Big O, but (Adjacency List) was the best of them of Runtime Complexity and we concluded that by reading 12213 data.



Run Time Device Specifications

LENOVO IDEAPAD

SDD: 1TB - RAM: 16GB

AMD RYZEN7 | 2.9GHZ

SPYDER - VERSION 5

PROGRAMMING LANGUAGE:PYTHON

Big(O) Notation

Data Structure	Graph	Adjacency List	Adjacency matrix
Dinic's algorithm	O(n^2)	O(n)	O(n^2)
Data Structure	Stack & 2D Array	Queue & Graph	Queue & 2D Array
Edmonds' - Karp algorithm	O(log n)	O(log n)	O(log n)
Data Structure	Adjacency Array	Graph	Adjacency List
Ford Fulkerson algorithm	O(n^2)	O(n^2)	O(n^2)

Best Algorithm

We observation the best Algorithm for Network Flow is EDMONONDS'-KARP Algorithm because it's take O(log n)

Run Time Complexity

Data Structure	Graph	Adjacency List	Adjacency matrix
Dinic's algorithm	-	-	-
Data Structure	Stack & 2D Array	Queue & Graph	Queue & 2D Array
Edmonds' algorithm	4.5 s	3.15s	4.3s
Data Structure	Adjacency Array	Graph	Adjacency List
Ford Fulkerson algorithm	52.09s	1.89s	0.06s

References

Dinic's Algorithm

https://www.geeksforgeeks.org/dinics-algorithm-maximum-flow/

https://iq.opengenus.org/dinics-algorithm/#algorithm

https://github.com/williamfiset/Algorithms/blob/master/src/main/java/com/williamfiset/algorithms

/graphtheory/networkflow/Dinics.java

https://www.topcoder.com/thrive/articles/edmonds-karp-and-dinics-algorithms-for-maximum-flow

Edmonds' - Karp Algorithm

Edmonds Karp Algorithm for maximum flow (opengenus.org)

python - Edmonds—Karp time complexity - Stack Overflow

https://brilliant.org/wiki/edmonds-karp-algorithm/

Creating capacity graph for edmonds karp maximum flow algorithm in Python - Stack Overflow

Ford_ Fulkerson Algorithm

https://www.programiz.com/dsa/ford-fulkerson-algorithm

https://www.autoscripts.net/pseudocode-of-ford-fulkerson/

I hope you like it and satisfy you.