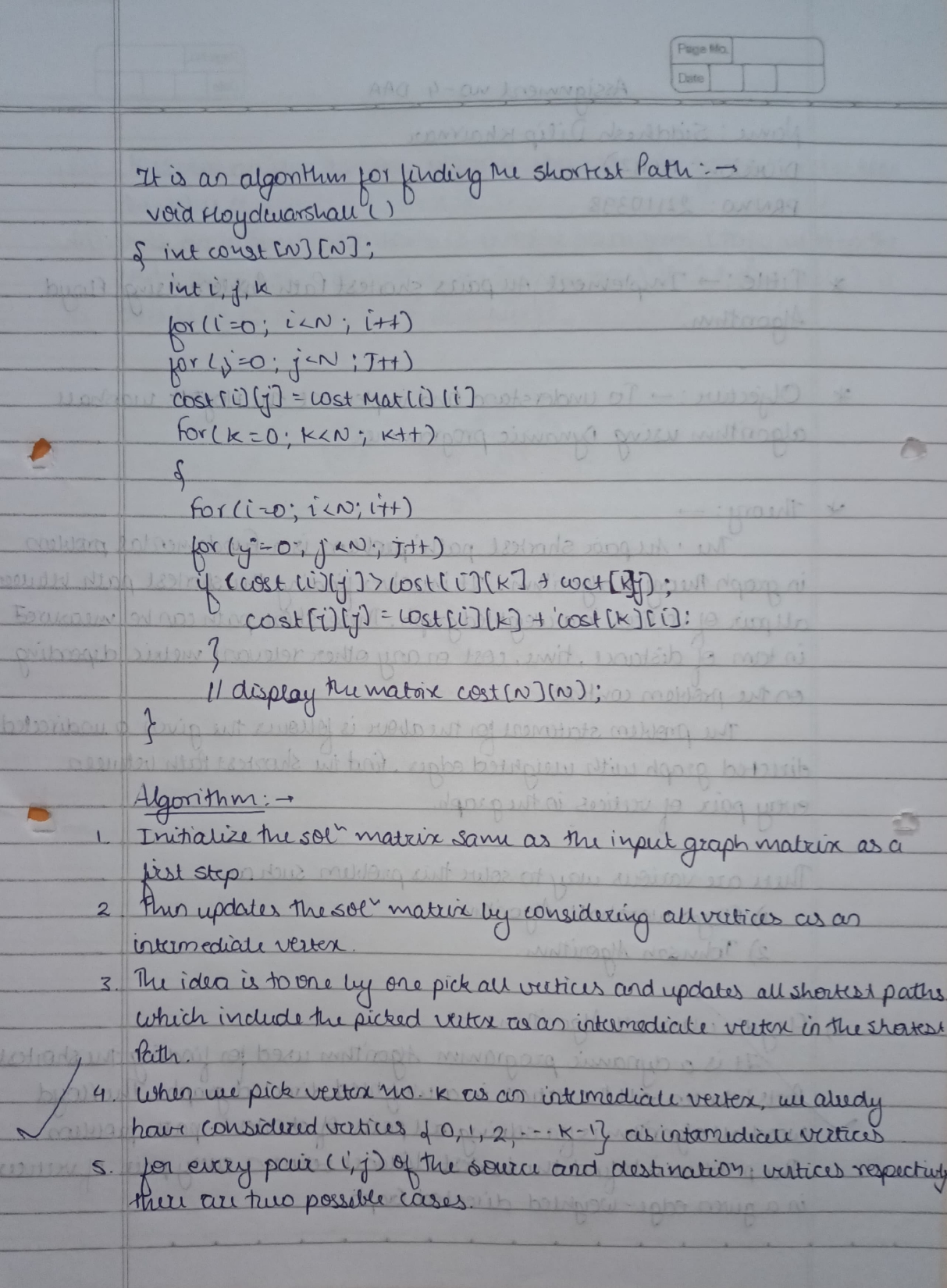
|  |  |  |  |
| --- | --- | --- | --- |
|  | Bansilal Ramnath Agarwal Charitable Trust's  Vishwakarma Institute of Information Technology  **Department of**  **Artificial Intelligence and Data Science** | | |
| Name: Siddhesh Dilip Khairnar | | | |
| Class: TY | Division: B | | Roll No: 372028 |
| Semester: V | | Academic Year: 2023-2024 | |
| Subject Name & Code: Design and Analysis of Algorithm: ADUA31202 | | | |
| Title of Assignment: Implement All Pair Shortest paths problem using Floyd's Algorithm. | | | |
| Date of Performance: 16-09-2023 | | Date of Submission: 22-09-2023 | |

**ASSIGNMENT NO. 4**

A piece of paper with writing on it

Description automatically generated

A piece of lined paper with writing on it

Description automatically generated

**Aim:** Implement All Pair Shortest paths problem using Floyd's Algorithm.

**Program Code:**

def floyd\_warshall(graph):

    num\_vertices = len(graph)

    # Initialize the distance matrix with the same values as the input graph

    dist = [[float('inf')] \* num\_vertices for \_ in range(num\_vertices)]

    for i in range(num\_vertices):

        for j in range(num\_vertices):

            if i == j:

                dist[i][j] = 0

            elif graph[i][j] != 0:

                dist[i][j] = graph[i][j]

    # Update the distance matrix using intermediate vertices

    for k in range(num\_vertices):

        for i in range(num\_vertices):

            for j in range(num\_vertices):

                if dist[i][k] != float('inf') and dist[k][j] != float('inf') and dist[i][k] + dist[k][j] < dist[i][j]:

                    dist[i][j] = dist[i][k] + dist[k][j]

    return dist

# Example usage

inf = float('inf')

graph = [

    [0, 3, inf, 7],

    [8, 0, 2, inf],

    [5, inf, 0, 1],

    [2, inf, inf, 0]

]

result = floyd\_warshall(graph)

for row in result:

    print(row)

**Result:**

A screen shot of a computer

Description automatically generated

Conclusion: Floyd's Algorithm finds shortest paths between all pairs of vertices in a weighted directed graph, suitable for medium-sized graphs (O(V^3) complexity). It detects negative weight cycles but becomes inefficient for large graphs. It has applications in network routing and transportation optimization but requires consideration of computational resources and negative cycles.