EXPERIMENT NO. 08

Roll No.:24141001
Batch: I1

Title: All Pairs Shortest Paths Problem Using Floyd's Algorithm.

Objectives:

- Understand the concept of shortest paths in weighted graphs.
- Find the shortest distance between every pair of vertices in a graph.
- Implement Floyd–Warshall algorithm using dynamic programming.
- Analyze time and space complexity.

Theory:

The All Pairs Shortest Paths (APSP) problem finds the minimum distance between every pair of vertices in a weighted graph.

Floyd–Warshall Algorithm

Floyd's algorithm is a Dynamic Programming algorithm used to compute shortest paths between every pair of nodes in a weighted graph.

Works for:

- Directed and undirected graphs
- Positive and negative edge weights
- Does NOT work for negative cycles

Formula Used

For each pair of vertices *(i, j)*, check if path through vertex *k* is shorter:

dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])

This update is repeated for all vertices k = 1 to n.

Algorithm:

```
1. Input number of vertices n
2. Input adjacency matrix dist[][]
   - If no edge, use a large value (∞)
   - dist[i][i] = 0
3. For k = 1 to n:
       For i = 1 to n:
           For j = 1 to n:
               If dist[i][j] > dist[i][k] + dist[k][j]:
                   dist[i][j] = dist[i][k] + dist[k][j]
4. Print the final distance matrix
```

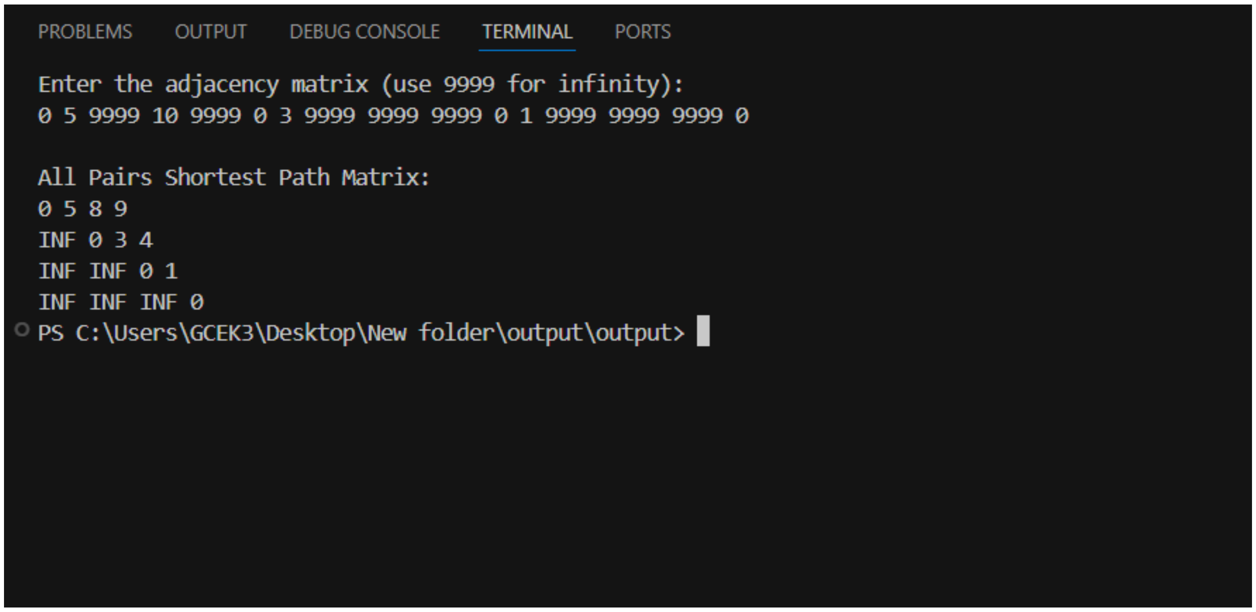
Program:

```c
#include <stdio.h>
#define INF 9999

int main() {
    int n, i, j, k;
    int dist[20][20];

    printf("Enter number of vertices: ");
    scanf("%d", &n);
```

```
printf("Enter the adjacency matrix (use 9999 for infinity):\n");
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
        scanf("%d", &dist[i][j]);
    }
}

// Floyd–Warshall Algorithm
for (k = 0; k < n; k++) {
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (dist[i][j] > dist[i][k] + dist[k][j])
                dist[i][j] = dist[i][k] + dist[k][j];
        }
    }
}

printf("\nAll Pairs Shortest Path Matrix:\n");
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
        if (dist[i][j] == INF)
            printf("INF ");
        else
            printf("%d ", dist[i][j]);
    }
```

```
        printf("\n");
    }

    return 0;
}
```

OUTPUT:



```
Enter the adjacency matrix (use 9999 for infinity):
0 5 9999 10 9999 0 3 9999 9999 9999 0 1 9999 9999 9999 0

All Pairs Shortest Path Matrix:
0 5 8 9
INF 0 3 4
INF INF 0 1
INF INF INF 0
PS C:\Users\GCEK3\Desktop\New folder\output\output>
```

Applications of Floyd–Warshall Algorithm

- Shortest routes in transport networks
- Routing protocols in computer networks
- Finding transitive closure of a graph
- Social network analysis (degrees of separation)
- Traffic optimization

- Optimal communication paths

Time and Space Complexity:

Time complexity: O(n^3)

Space Complexity: O(n^2)

Conclusion:

The Floyd–Warshall algorithm is a powerful dynamic programming technique used to compute shortest paths between every pair of vertices in a weighted graph. Although its time complexity is O(n^3), it is simple and very useful for dense graphs and routing problems.