# Integration Testing for Floyd Warshall Algorithm - Recursive Version

This is an integration testing plan for the Floyd Warshall Algorithm using recursion. The purpose of this testing plan is to ensure that the algorithm is working correctly and consistently.

## Test Cases

### Test 1: Small Graph

This test case will test the algorithm on a small graph with four vertices. The graph is the same as the example in the code:

```

10

(0)------->(3)

| /|\\

5 | |

| | 1

\|/ |

(1)------->(2)

3

```

#### Input

```python

graph = [[0, 7, INF, 8],

[INF, 0, 5, INF],

[INF, INF, 0, 2],

[INF, INF, INF, 0]

]

V = 4

dist = graph

```

#### Expected Output

```

Following matrix shows the shortest distances between every pair of vertices:

0 7 12 8

INF 0 5 10

INF INF 0 2

INF INF INF 0

```

### Test 2: Large Graph

This test case will test the algorithm on a larger graph with eight vertices. The graph is a random weighted graph.

#### Input

```python

graph = [[0, 1, 3, INF, INF, INF, INF, INF],

[1, 0, INF, 2, INF, INF, INF, INF],

[3, INF, 0, 1, INF, INF, 5, INF],

[INF, 2, 1, 0, 7, INF, INF, INF],

[INF, INF, INF, 7, 0, 2, INF, 3],

[INF, INF, INF, INF, 2, 0, 6, INF],

[INF, INF, 5, INF, INF, 6, 0, 4],

[INF, INF, INF, INF, 3, INF, 4, 0]

]

V = 8

dist = graph

```

#### Expected Output

```

Following matrix shows the shortest distances between every pair of vertices:

0 1 3 3 10 3 8 7

1 0 4 2 9 2 7 6

3 4 0 1 8 5 5 4

3 2 1 0 7 4 5 4

10 9 8 7 0 2 7 3

3 2 5 4 2 0 6 5

8 7 5 5 7 6 0 4

7 6 4 4 3 5 4 0

```

## Conclusion

This integration testing plan covers both small and large graphs to ensure that the algorithm is working correctly and consistently. By following this plan, we can ensure that the Floyd Warshall Algorithm using recursion is implemented correctly and will produce accurate results.