Floyd Warshall algorithm

Algorithm:

1. Input: Adjacency matrix 'a' representing the weighted graph and the number of vertices 'n'.

2. Initialize: the adjacency matrix with the weights of edges between vertices. If there is no direct edge between vertices, represent it with a large value.

3. Perform: the Floyd-Warshall algorithm:
(a) For each intermediate vertex 'k' from 0 to 'n-1':
(i) For each pair of vertices 'i' and 'j' from 0 to 'n-1':
- If the distance from 'i' to 'j' through 'k' is shorter than the current distance from 'i' to 'j', update the distance from 'i' to 'j'.

4. Output: the resultant adjacency matrix, which contains the shortest distances between all pairs of vertices.

Code:

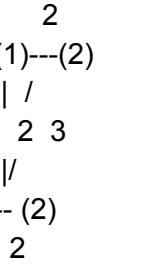
```
#include<stdio.h>
void floyd(int cost[10][10], int n) {
    int i, j, k;
    for (k = 0; k < n; k++) {
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                if (cost[i][j] > cost[i][k] + cost[k][j]) {
                    cost[i][j] = cost[i][k] + cost[k][j];
                }
            }
        }
    }
    printf("The shortest path matrix is:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", cost[i][j]);
        }
        printf("\n");
    }
}

int main() {
```

```
    int cost[10][10], n, i, j;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the cost matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &cost[i][j]);
        }
    }
    floyd(cost, n);
    return 0;
}
```

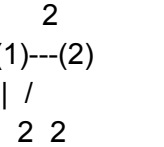
Input:

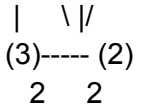
```
Enter the number of vertices: 4
Enter the cost matrix:
0 1 3 4
1 0 2 1
3 2 0 2
4 1 2 0
```



Output:

```
The shortest path matrix is:
0 1 3 2
1 0 2 1
3 2 0 2
2 1 2 0
```




\ |/
(3)----- (2)
2 2