Floyd Warshall’s Algorithm

check this out https://www.baeldung.com/cs/floyd-warshall-shortest-path

```
Create a |V| x |V| matrix, M, that will describe the distances between vertices
For each cell (i, j) in M:
    if i == j:
        M[i][j] = 0
    if (i, j) is an edge in E:
        M[i][j] = weight(i, j)
    else:
        M[i][j] = infinity
for k from 1 to |V|:
    for i from 1 to |V|:
        for j from 1 to |V|:
            if M[i][j] > M[i][k] + M[k][j]:
                M[i][j] = M[i][k] + M[k][j]
```

The strategy adopted by the Floyd-Warshall algorithm is **Dynamic Programming**

Given a directed or an undirected weighted graph G with n vertices. The task is to find the length of the shortest path dij between each pair of vertices i and j.

The graph may have negative weight edges, but no negative weight cycles.

We’re taking a directed weighted graph G(V,E) as an input.

And first, we construct a graph matrix from the given graph.

This matrix includes the edge weights in the graph.

Next, we insert 0 in the diagonal positions in the matrix.

The rest of the positions are filled with the respective edge weights from the input graph.

Then, we need to find the distance between two vertices. While finding the distance, we also check if there’s any intermediate vertex between two picked vertices. If there exists an intermediate vertex then we check the distance between the selected pair of vertices which goes through this intermediate vertex.

If this distance when traversing through the intermediate vertex is less then the distance between two picked vertices without going through the intermediate vertex, we update the shortest distance value in the matrix.

The number of iterations is equal to the cardinality of the vertex set. The algorithm returns the shortest distance from each vertex to another in the given

graph.