**Floyd-Warshall Algorithm**

**Understanding the Problem**

The Floyd-Warshall algorithm is a graph algorithm designed to find the shortest paths between all pairs of vertices in a weighted graph. It's particularly useful when you need to find the shortest paths between multiple pairs of nodes efficiently.

**The Algorithm**

The algorithm works by iteratively improving estimates of the shortest distances between vertex pairs. It uses dynamic programming to achieve this.

**Steps:**

1. **Initialization:**
   * Create a distance matrix dist where dist[i][j] represents the shortest distance between vertices i and j.
   * Initialize the matrix with the weights of the edges. If there's no direct edge, set the distance to infinity.
   * Set the diagonal elements (distances from a vertex to itself) to 0.
2. **Iteration:**
   * Iterate over all possible intermediate vertices k.
   * For each pair of vertices i and j, check if using k as an intermediate vertex provides a shorter path:
     + dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])
3. **Result:**
   * After the iterations, the dist matrix will contain the shortest distances between all pairs of vertices.

**Pseudocode**

for k := 1 to |V|

for i := 1 to |V|

for j := 1 to |V|

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

multi source shortest path also

detect negative cycle as well

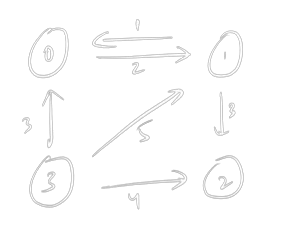
go via every vertex/nodes

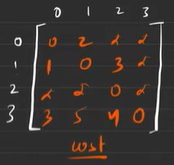
d[0][1] -> (0->1)

(0->2) + (2->1)

(0->3) + (3 -> 1)

(0->4) + (4->1)





[1][2] -> [1][0] + [0][2]

1 + inf

Inf

Dijkstra vs floyyd