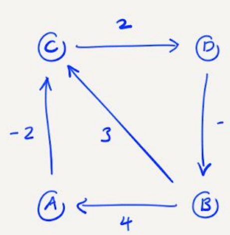
Floyd-Warshall (All-pair shortest distance)

Floyd-Warshall relies on the observation that the shortest path from u -> v that has at least 2 edges can be decomposed into two shortest paths u -> k and k -> v for some intermediate vertex k (the jump vertex). The concept of transitive closure.

1. Find all-pairs shortest paths using Floyd-Warshall.

Floyd - Warshall

A → B , B → C , ∴ A → C



initialize

	A	B	C	D
A	0	i	-2	i
B	4	0	3	i
C	i	i	0	2
D	i	-1	i	0

k = A

	A	B	C	D
A	0	i	-2	i
B	4	0	2	i
C	i	i	0	2
D	i	-1	i	0

k = B

	A	B	C	D
A	0	i	-2	i
B	4	0	2	i
C	i	i	0	2
D	3	-1	2	0

k = C

	A	B	C	D
A	0	i	-2	0
B	4	0	2	5
C	i	i	0	2
D	3	-1	2	0

k = D

	A	B	C	D
A	0	i	-2	0
B	4	0	-2	3
C	i	1	0	2
D	3	-1	2	0

i = infinite

2. Explain how you will return the paths (all-pairs) using the Floyd-Warshall algorithm.

- Maintain a V x V matrix L. L[u][v] stores which node to go next for the shortest path from u to v.
- For every vertex v, L[v][v] = v
- For every edge (u, v), L[u][v] = v
- During every relaxation, if dist[u][v] > dist[u][k] + dist[k][v], then we modify the matrix L, L[u][v] = L[u][k]
- To obtain the shortest path from vertex u to v. We can iteratively find the next node to go by referring to the matrix element, L[curr][v] where curr is the current node.

3. How do we find negative cycles?

- Look at the diagonal of your 2D array going from source vertex back to itself.
- If the value is negative, it means there's a negative cycle.

4. Discuss pseudocode.

Comparison of finding all-pair shortest distance with different algorithms we learnt

1. Dijkstra
   a. Original O(E log V)
   b. Dijkstra from every vertex:
      i. Sparse graph O(VE log V)
      ii. Dense graph (E ≈ V^2) O(V^3 log V)

2. Bellman-Ford
   a. Original O(VE)
   b. Bellman-Ford from every vertex:
      i. Sparse graph O(V^2 E)

ii. Dense graph ($E \approx V^2$) $O(V^4)$

3. Floyd-Warshall
   a. Original $O(V^3)$