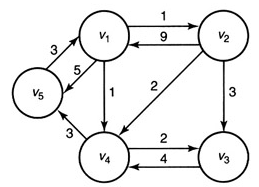
# Shortest path algorithms



Multiple paths from V1 to V3:

[v1,v2,v3]

[v1,v4,v2,v3]

[v1,v4,v3] 🡪 Shortest path.

## Brute Force Discussion:

Path from one vertex (v1) to every other vertex will have a subset where every vertex is in the path. Such a path will have exponential value as there will be

(n-2)(n-3)(n-4)…1 = (n-2)!

n-2 vertices to choose as the second vertex.

n-3 vertices to choose as the third vertex.

Now this is only a subset of all paths from vertex V1 to another vertex and this is exponential already.

Hence, brute force will be worse than exponential.

## Floyd’s algorithm

Dk[i][j] = shortest path from ‘i’ to ‘j’ using vertices in the set {v1,v2,v3…vk}

D0[i][j] = shortest path from ‘I’ to ‘j’ using vertices in the set {} i.e. direct paths = W[i][j]

Dn[i][j] = shortest path from ‘I’ to ‘j’ using any of the vertices in the set {1,2,3,…n}

Shortest path from i to j is:

Dk[i][j] = min { Dk-1[i][j], Dk-1[i][k] + Dk-1[k][j]}

Dk-1[i][j] : Shortest path from ‘i’ to ‘j’ that does not pass through vertex k

Dk-1[i][k] + Dk-1[k][j]: Shortest path from ‘i’ to ‘j’ that passes through vertex k.

### Bottoms Up Construction

Start with the base cases,

1. Calculate shortest paths from u to v using 0 intermediate vertices.
2. Calculate shortest paths from u to v using (0,1) intermediate vertices
3. Calculate shortest paths from u to v using (0, 1, 2) intermediate vertices
4. Calculate shortest paths from u to v using (0,1,2…n) intermediate vertices.