All-pairs shortest path

$D^{(0)}$:

	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

$D^{(1)}$:

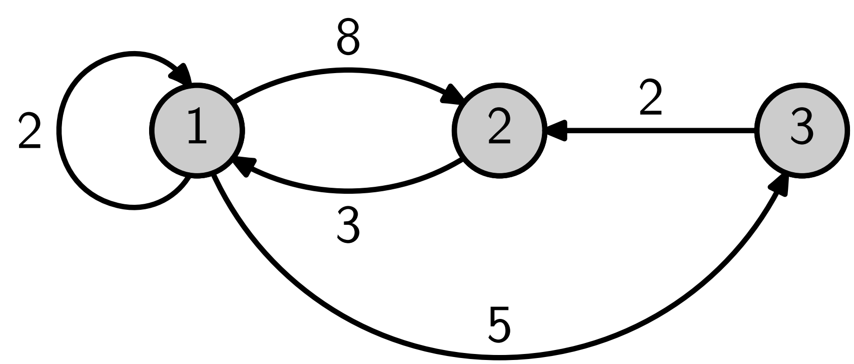
	1	2	3
1	0	8	5
2	3	0	8
3	∞	2	0

$D^{(2)}$:

	1	2	3
1	0	8	5
2	3	0	8
3	5	2	0

$D^{(3)}$:

	1	2	3
1	0	7	5
2	3	0	8
3	5	2	0

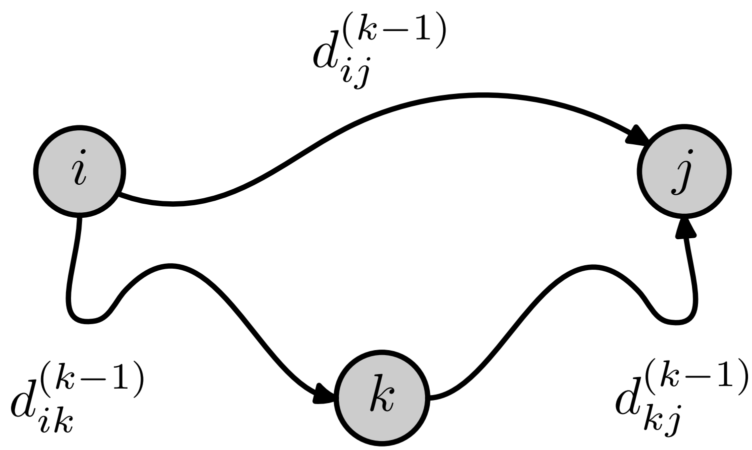


In each iteration $k = 1 \ldots n$, see if it's cheaper to go from i to j via k.

After the kth iteration, $d_{ij}^{(k)}$ will be the shortest path from i to j that does not pass through any vertex $> k$.

Recurrence

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min(d_{ij}^{(k-1)}, d_{ij}^{(k-1)} + d_{kj}^{(k-1)}) & \text{if } k \geq 1 \end{cases}$$



Floyd-Warshall's algorithm

Since $d_{ik}^{(k)} = d_{ik}^{(k-1)}$ and $d_{kj}^{(k)} = d_{kj}^{(k-1)}$, no entry with either subscript $= k$ changes during the kth iteration.

Therefore we can perform the computation with only copy of D.

$\textsc{Floyd-Warshall}(W)$

1. $n \leftarrow rows[W]$
2. $D \leftarrow W$
3. **for** $k \leftarrow 1$ **to** n
4.   **do for** $i \leftarrow 1$ **to** n
5.   **do for** $j \leftarrow 1$ **to** n
6.   **do** $d_{ij} \leftarrow \min(d_{ij}, d_{ik} + d_{kj})$

Analysis and comparison

- Clearly, FLOYD-WARSHALL is an $O(n^3)$ algorithm.
- An adjacency-list implementation of Dijkstra's algorithm is $O(n^2)$.
- Calling Dijkstra's algorithm n times would cost $O(n^3)$.
- Why use Floyd-Warshall's algorithm then?
- The constant inside $O(n^3)$ is much smaller for Floyd-Warshall.
- For sparse graphs, however, there is a $O(n\ e \lg n)$ implementation of Dijkstra's algorithm, where $e =$ number of edges.