- [[Shortest Path Algorithms]] for all vertex pairs
- distance matrix is calculated directly
- [[Dynamische Programmierung]]
- compute a sequence of distance matrices $w_1, ..., w_n$
  - initial weight matrix w as input

    weight matrix $w(i,j)$, $1 \leq i,j \leq n$, defined by
    $$w(i,j) = \begin{cases} w(v_i, v_j) & \text{if } (v_i, v_j) \in E \\ 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$$
    - $w_k(i,j) = \min\{w_{k-1}(i,j), w_{k-1}(i,k) + w_{k-1}(k,j)\}$
  - $w_0 = w$.
- $w_n(i,j)$ is the distance from v_i to v_j in G.
- proof by induction
  - **Proof.** We show by induction on k that $w_k(i,j)$ is the length of the shortest path from v_i to v_j via $\{v_1, ..., v_k\}$.

    Induction base: For $k = 0$ the statement is true:
    - if $i \neq j$ and $v_i v_j \in E$ then $w_0(i,j) = w(v_i, v_j)$;
    - if $i \neq j$ and $v_i v_j \notin E$ then $w_0(i,j) = \infty$;
    - $w_0(i,i) = 0$.

    In all cases, $w_0(i,j)$ is the shortest path from v_i to v_j without intermediate vertices.

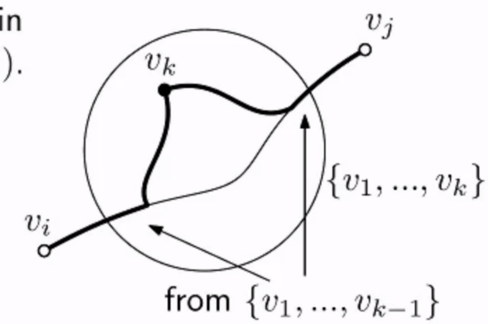
    Induction step: Assume the statement is correct up to $k-1$ and consider w_k.
  - w_{k-1} may use v_k as start or end point but not as intermediate

Observation: The shortest path π from v_i to v_j via vertices from $\{v_1, ..., v_k\}$ may or may not contain v_k.

- If π contains v_k, then the parts of π from v_i to v_k and from v_k to v_j go only via $\{v_1, ..., v_{k-1}\}$.

$\Rightarrow$ By induction, the lengths of those parts are stored in $w_{k-1}(i, k)$ and $w_{k-1}(k, j)$.

$\Rightarrow$ Hence the length of π is $w_{k-1}(i, k)+w_{k-1}(k, j)$.



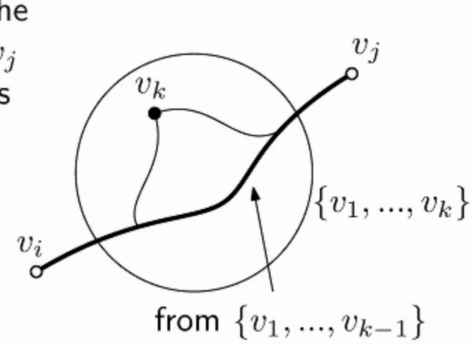
–

- If π does not contain v_k then π goes via $\{v_1, ..., v_{k-1}\}$.

$\Rightarrow$ By induction, the length of π is stored in $w_{k-1}(i, j)$.

- The algorithm takes the minimum of the two considered possibilities $\Rightarrow$ $w_k(i, j)$ is the length of π in both cases.

$\Rightarrow$ $w_n(i, j)$ is the length of the shortest path from v_i to v_j that can go via all vertices of V and hence $w_n(i, j) = d(v_i, v_j)$.



–

- pseudo code

$w_0 = w$
for $k = 1$ **to** n **do**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do**
      $w_k(i, j) = \min\{w_{k-1}(i, j), w_{k-1}(i, k) + w_{k-1}(k, j)\}$

–

Requirements for G with n vertices and m edges:

- Runtime: $\Theta(n^3)$
- Memory: $\Theta(n^2)$

•

- The Floyd-Warshall algorithm also works if the graph is disconnected (if not every vertex can be reached from every other vertex). The distance between such vertices is set to ∞ in the matrix w_n.
- With a small adaption, the Floyd-Warshall algorithm can also be used for graphs with negative edge weights: Then an additional check for the existence of (possibly trivial) cycles with negative length is needed. A graph has a (possibly trivial) cycle with negative length if and only if the matrix w_n contains negative entries in its diagonal.