Problem definition

Let G =(V,E,w) be a directed Graph with the set of nodes V{\displaystyle V}VVV and the set of edges E Consisting of V x V.

Each edge ( e ) belong to E has a weight w( e ) assigned.

The goal of the all-pair-shortest-paths problem is to find the shortest path between **all** pairs of nodes of the graph.

For this path to be unique it is required that the graph does not contain cycles with a negative weight.

the graph is represented using an adjacency matrix We expect the output of the algorithm to be a distance matrix {\displaystyle D}.

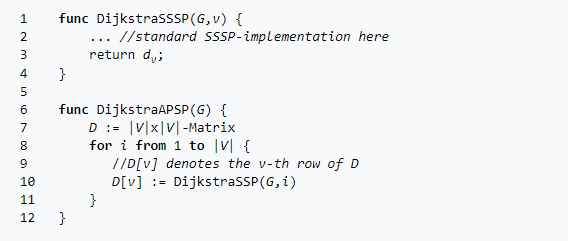
in D, every entry d – i,j is the weight of the shortest path in G from node i to node j.

whereas the Dijkstra algorithm requires all edges to have a positive weight.

## Dijkstra algorithm

The [Dijkstra algorithm](https://en.wikipedia.org/wiki/Dijkstra_algorithm" \o "Dijkstra algorithm) originally was proposed as a solver for the single-source-shortest-paths problem. However, the algorithm can easily be used for solving the All-Pair-Shortest-Paths problem by executing the Single-Source variant with each node in the role of the root node.

In pseudocode such an implementation could look as follows:



In this example we assume that DijkstraSSSP  takes the graph G and the root node v as input.

The result of the execution in turn is the distancelist dv .

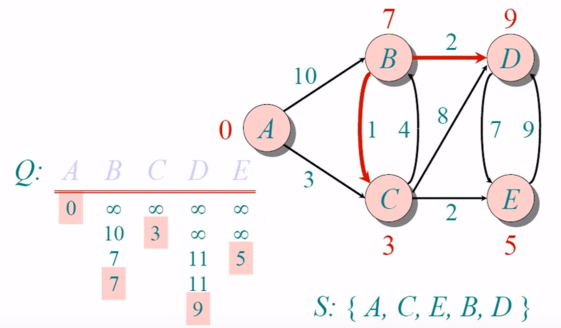
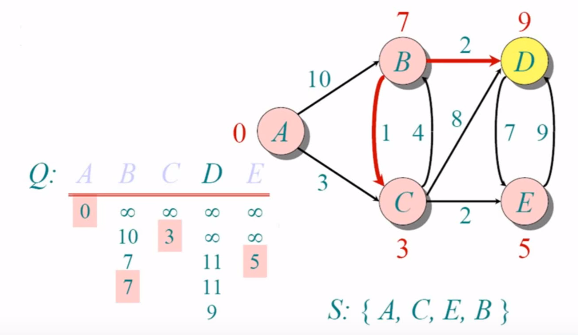
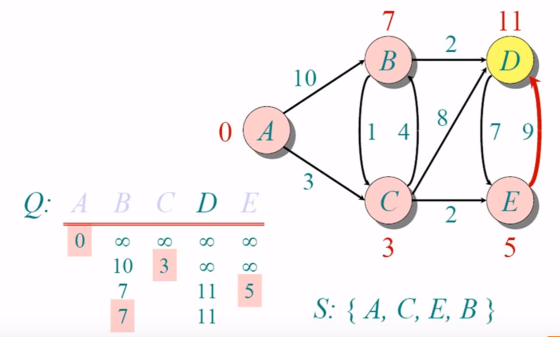
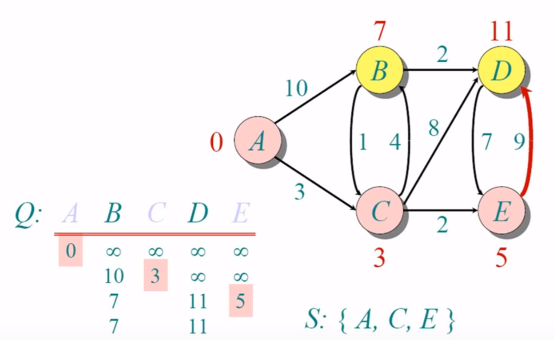
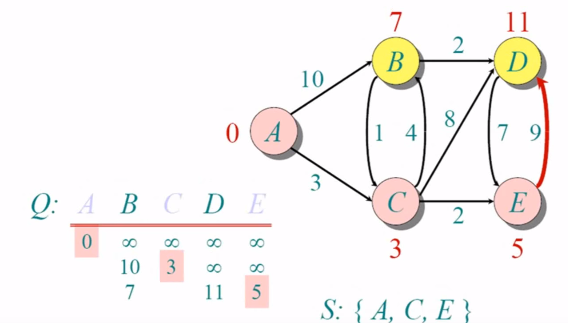
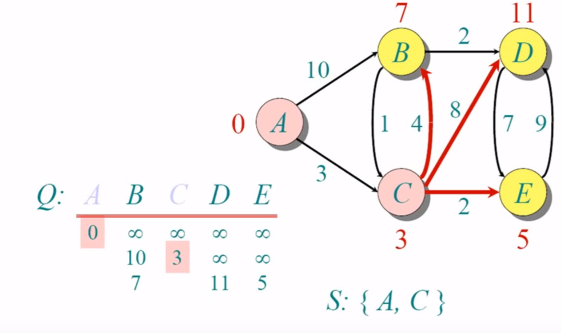
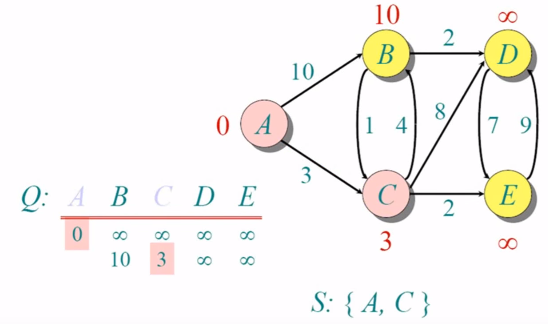
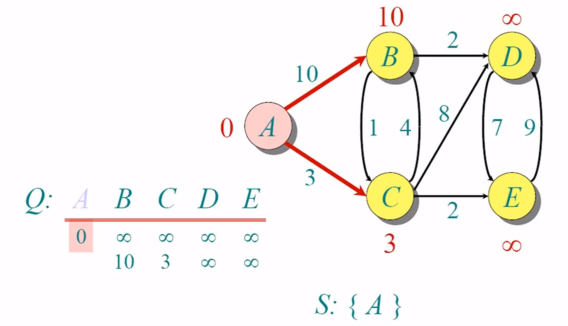
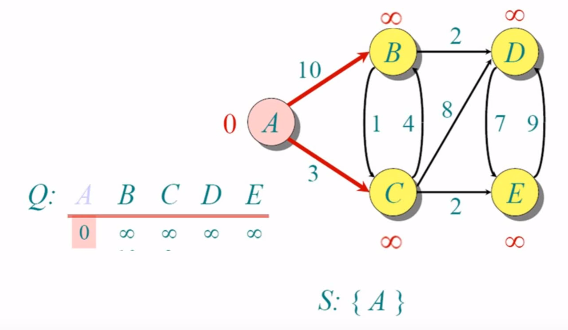
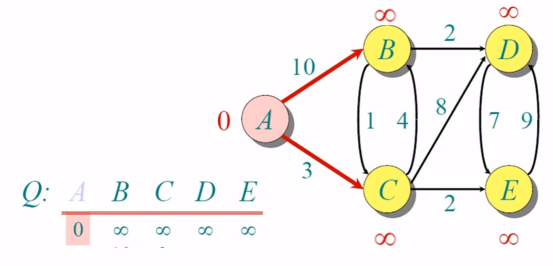
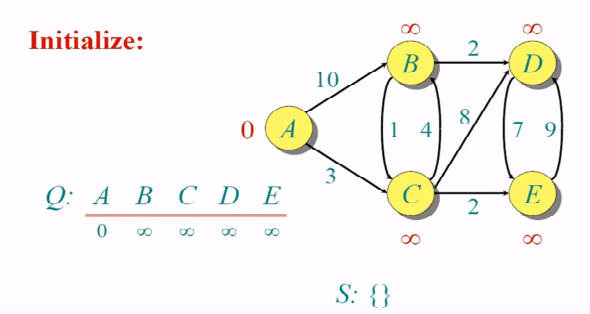
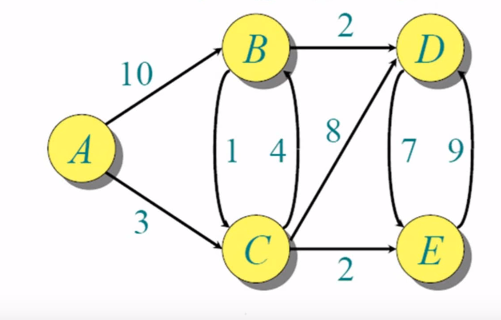
The i-th element stores the distance from the root node v to the node i .

Therefore the list  dv corresponds exactly to the v-th  row of the APSP distancematrix D .

For this reason DijkstraAPSP iterates over all nodes of the graph G and executes DijkstraSSSP   with each as root node while storing the results in D .

The runtime of DijkstraSSSP is O(|V|^2)  as we expect the graph to be represented using an [adjacency matrix](https://en.wikipedia.org/wiki/Adjacency_matrix).

Therefore DijkstraAPSP has a total sequential runtime of O(|V|^3).



### Parallelization for up to |*V*| processors

A trivial parallelization can be obtained by parallelizing the loop of DijkstraAPSP in line*8*.

However, when using the sequential DijkstraSSSP  this limits the number of processors to be used by the number of iterations executed in the loop.

Therefore, for this trivial parallelization |V| is an upper bound for the number of processors.

For example, let the number of processors P be equal to the number of nodes |V| .

This results in each processor executing DijkstraSSSP exactly once in parallel.    
 However, when there are only for example P = |V|/2  processors available, each processor has to execute DijkstraSSSP twice.

In total this yields a runtime of O(|V|^2 . |V|/P) , when  |V|  is a multiple of P.

 Consequently, the efficiency of this parallelization is perfect: Employing P

processors reduces the runtime by the factor P.

{\displaystyle |V|}