MA3077 (DLI) Operational Research

Lecture 10&11– Networks

Dr Neslihan Suzen

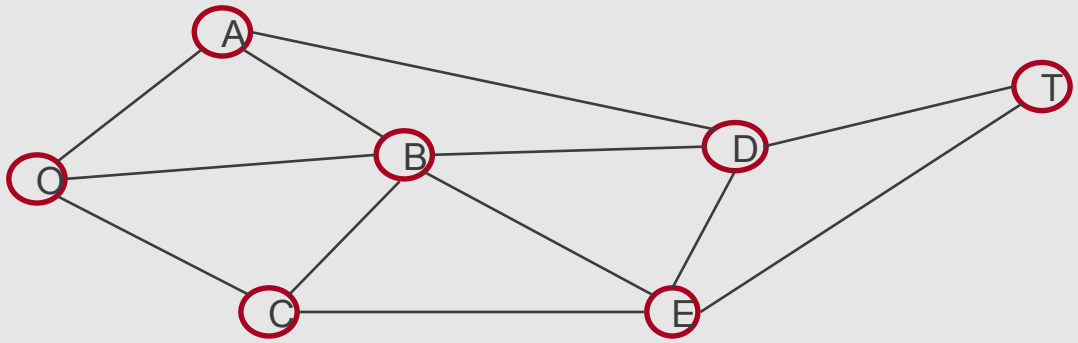
Recap and lecture outline

Summary: We have learnt:

- about sensitivity analysis
- how to model mixed-integer linear programming problems,
- and how to solve them in Matlab using `intlinprog`.

Today: Networks, following loosely Ch. 10 of the book by Hillier and Lieberman.

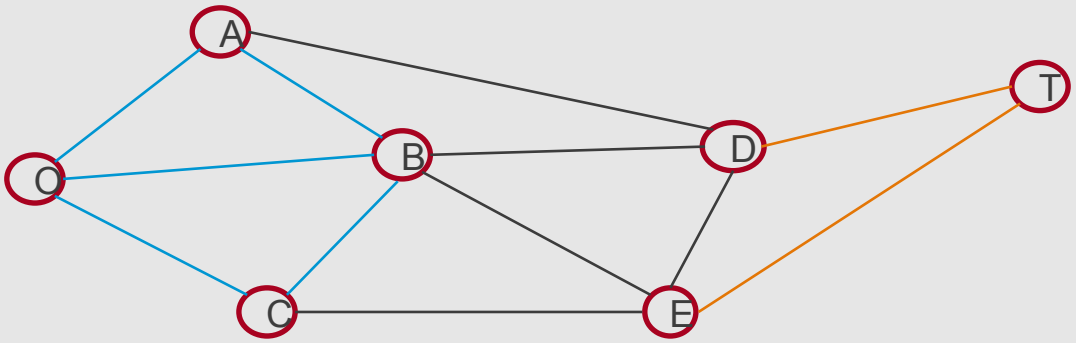
Graphs, nodes, and edges



Definition: A *graph* is a set of *nodes* (depicted as circles) and a set of *edges* (depicted as lines and also known as links, arcs or branches) that connect certain pairs of nodes.

We assume that at most one edge connects a given pair of nodes and that edges are non-oriented. Then every edge is uniquely determined by the pair of nodes it connects.

Subgraphs and full subgraphs

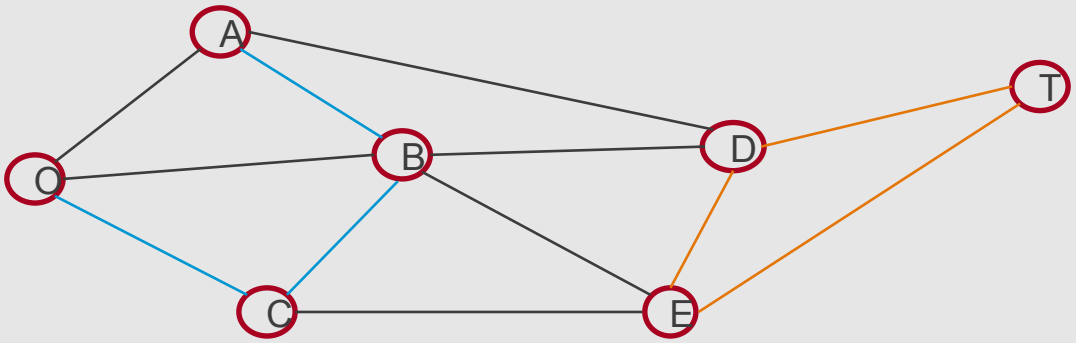


Definition: A *subgraph* is a graph with and

A subgraph is *full* if an edge connecting two nodes in that are in is also in (in other words, if pairs of nodes enter the subgraph along with their edges).

Example: The blue subgraph is full, whereas the orange one is not.

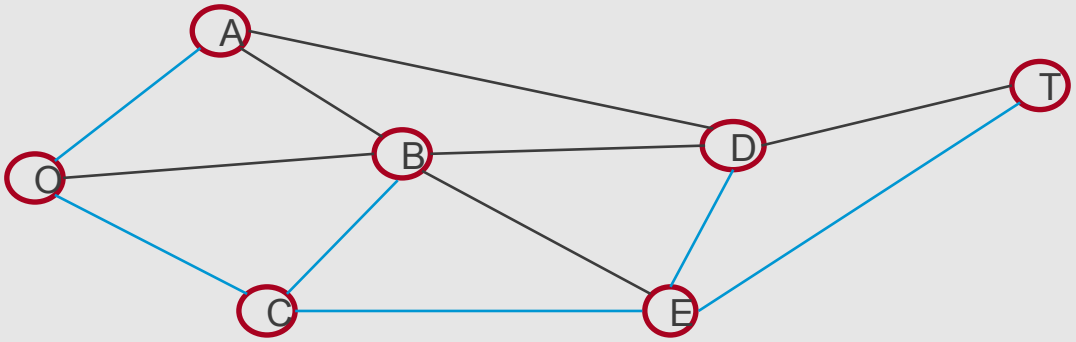
Paths and cycles



Definitions:

- Two edges are *connected* if adjacent to a common node
- A *path* is a (unique) sequence of connected edges with each node being adjacent to at most two edges in the path (e.g. blue line)
- A *cycle* is a path that begins and ends at the same node (e.g. orange line).

Connected graphs and trees



Definitions:

- A graph is connected if each pair of nodes is connected by a path.
- A connected graph is a tree if the connecting paths are unique.
- A spanning tree of a connected graph is a subgraph with that is also a tree.

Euler formula for trees

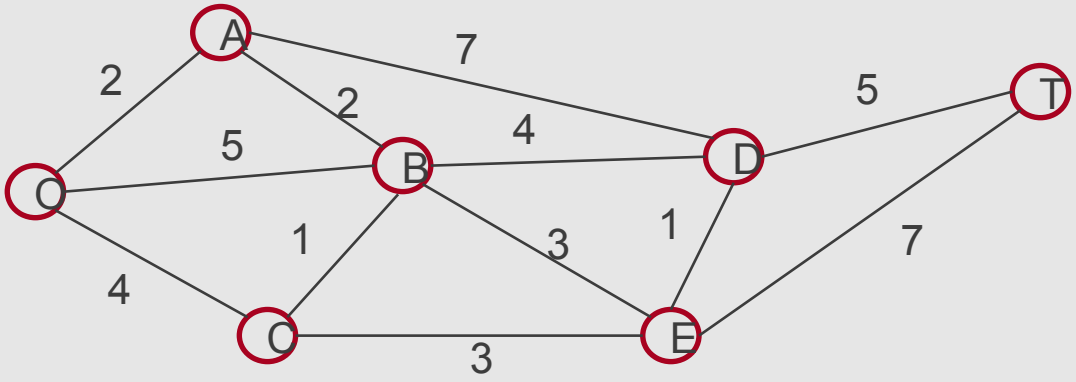
Proposition: Let be a connected graph with nodes and edges. Then, is a tree if and only if .

Proof:

Let be a tree with nodes. If , the statement is obviously true. If , remove from the tree a node with a single connection (so-called *leaf*) as well as its connection. The result is a tree with nodes and edges. Repeating this procedure additional times (for a total of removals) lead to a tree with 2 nodes, and necessarily at most one edge. This implies that the original tree had edges.

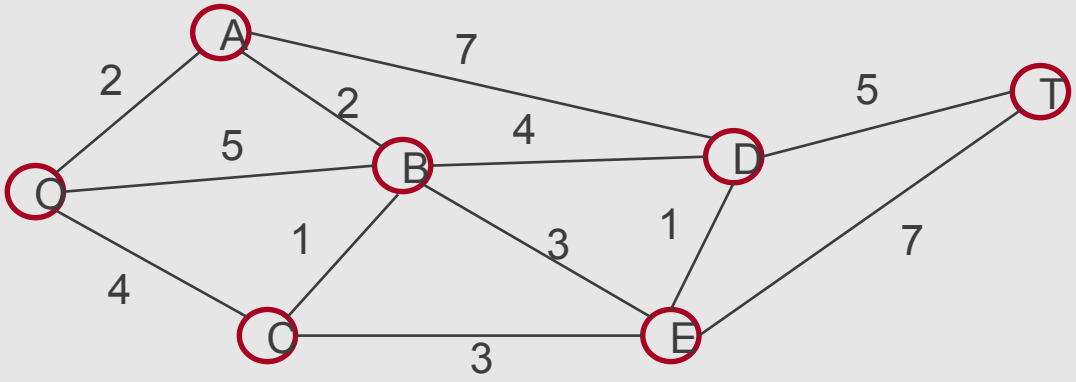
Let be a connected graph with nodes and edges. Assume that two nodes are connected by two different paths. Then, the union of these paths contains a cycle, with nodes and edges. Connecting the remaining nodes to this cycle requires at least additional edges. This is a contradiction. □

Networks



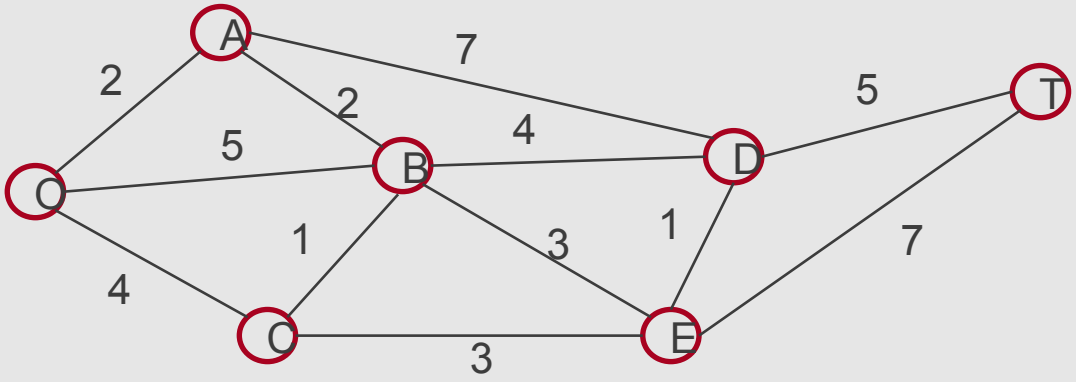
Definition: a *network* (aka *weighted graph*) is a graph with a weight associated to each edge . In most applications, the weights are positive. In this module, we assume that the weights are indeed positive.

Example of a network - scenario



Scenario: A park has a narrow, winding road system for trams (shown in the figure). Location O is the entrance into the park; other letters designate the locations of ranger stations (and other limited facilities). The numbers are lengths of roads in miles. The park contains a scenic wonder at station T. A small number of trams are used to transport sightseers from the park entrance to station T and back.

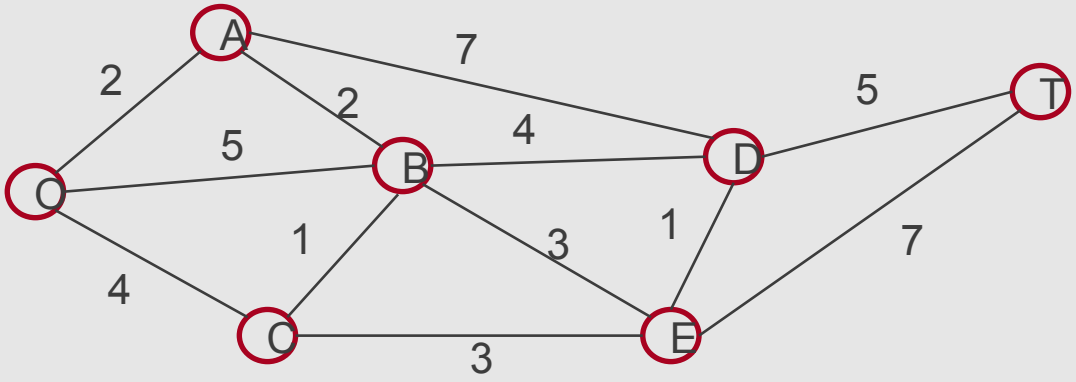
Example of a network – typical problems



Problem 1: The park manager wants to install telephone lines under the roads to establish telephone communication among all the stations. To reduce costs, lines will be installed under just enough roads to ensure every station is connected to the web. Where shall they lay the lines to minimize the total length of lines installed?

Problem 2: What is the shortest route from the entrance to the scenic station T?

Minimal spanning trees and shortest paths



Minimal spanning tree problem: Identify a connected subnetwork that contains all the nodes and such that the sum of the weights of the edges included (the total weight) is minimal.

Shortest path problem: Determine a path to join two nodes such that the sum of the weights of the edges included is minimal.

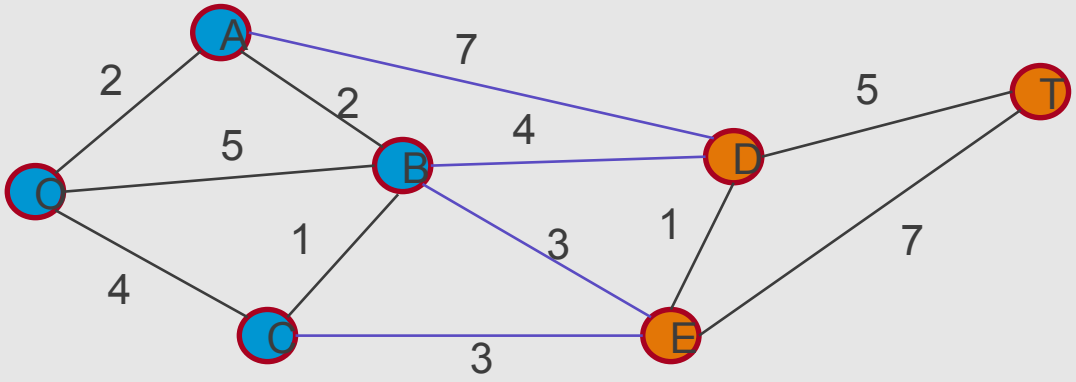
Minimal spanning tree via linear programming

Let denote the cardinality of a generic set . Let . Then,

The second constraint is known as *subtour elimination constraint*: any subset of vertices must have at most edges contained in that subset. This ensures that there are no cycles.

Remark: this formulation enjoys the integer solution property!

Network cuts



Definition:

- A *cut* of a network is a partition such that
- The *cut-set* of a cut is

Example: K={O,A,B,C}, ={E,D,T} is a cut, and the set of edges *{(C,E), (B,E), (B,D), (A,D)}* is the cut-set.

Shortest path via linear programming

You can determine the shortest path between two nodes O and T by solving

The first constraint ensures that the resulting paths connects the source to the target .

A key property of cuts

Proposition: Let be a cut of a network and let satisfy

Then, belongs to a minimal spanning tree.

Proof: Assume that is a minimal spanning tree of , and that . Then, the network contains a cycle that connects and via and another edge (otherwise would be disconnected). Since , the network remains connected, it is a tree by Euler's formula, and its total weight is not bigger than .
□

Prim's algorithm (1930, Jarník)

Let be a connected network with nodes.

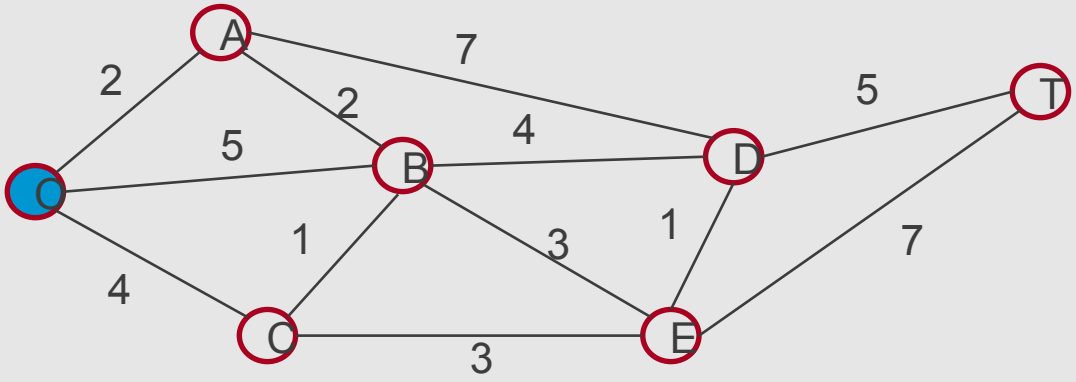
Initialization: pick , build the cut , and set

Then, for

a. determine the cut-set
b. find }
c. let
d. define
e. define

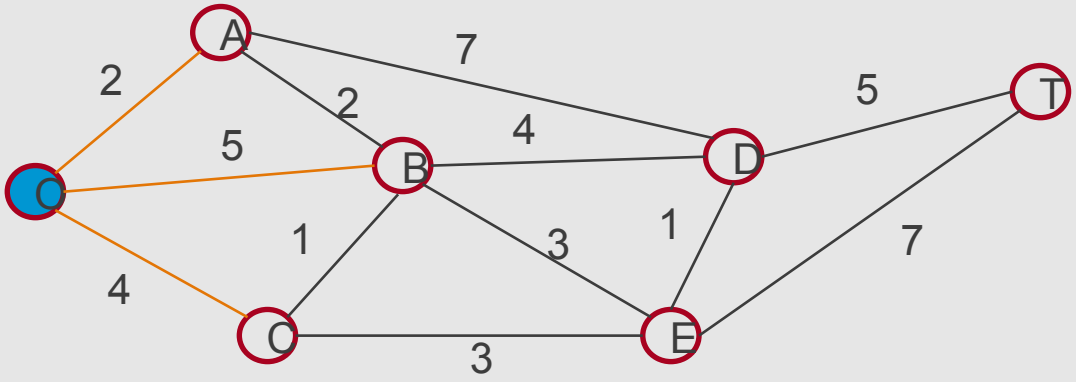
Output: is a minimal spanning tree.

Prim's algorithm – example (step 1)

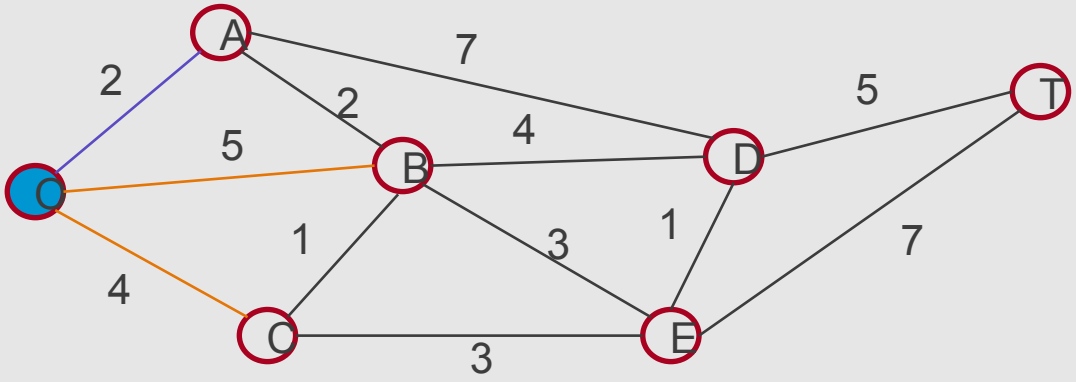


$$K_0 = \{O\}, E'_0 = \emptyset$$

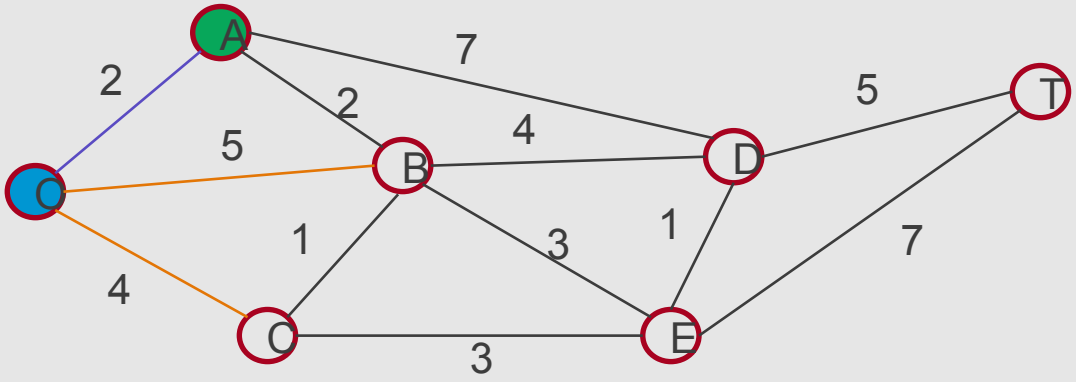
Prim's algorithm – example (step 2)



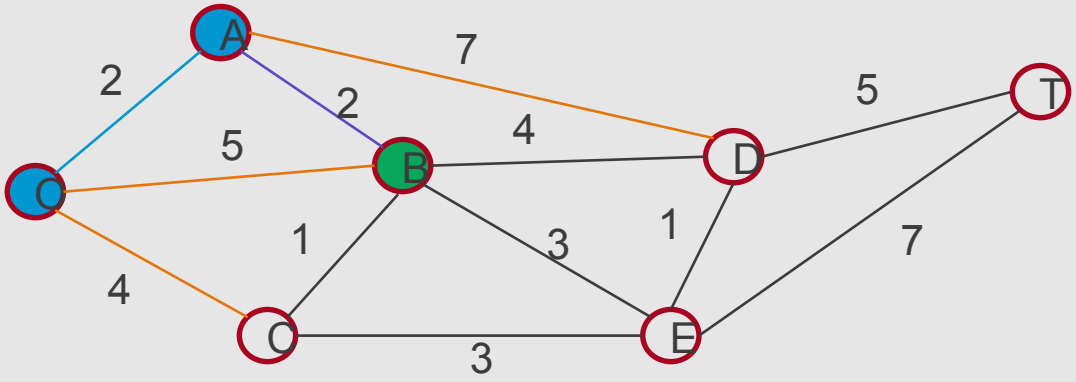
Prim's algorithm - example (step 3)



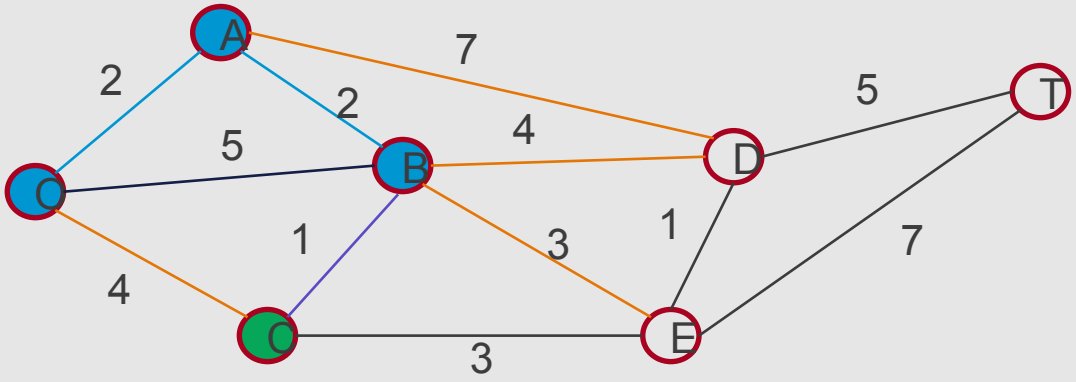
Prim's algorithm - example (step 4)



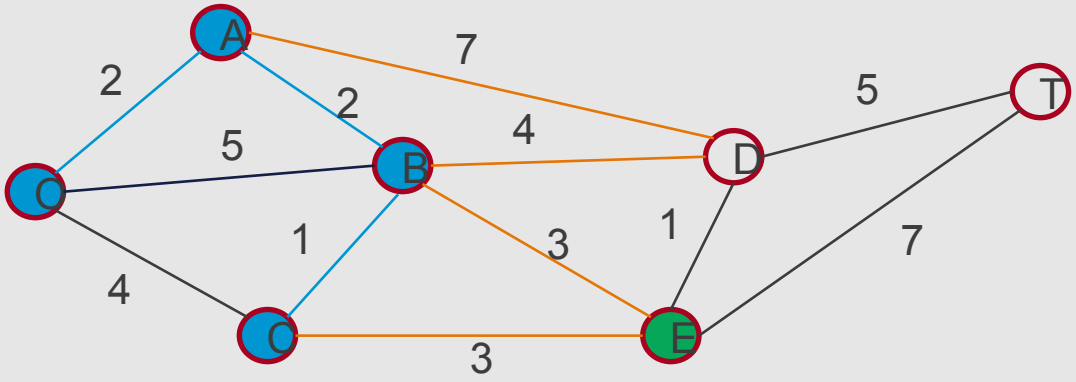
Prim's algorithm - example (step 5)



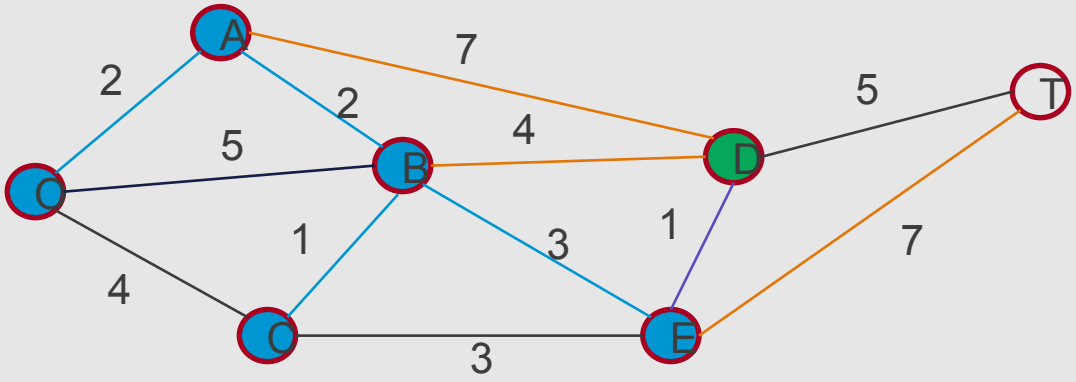
Prim's algorithm - example (step 6)



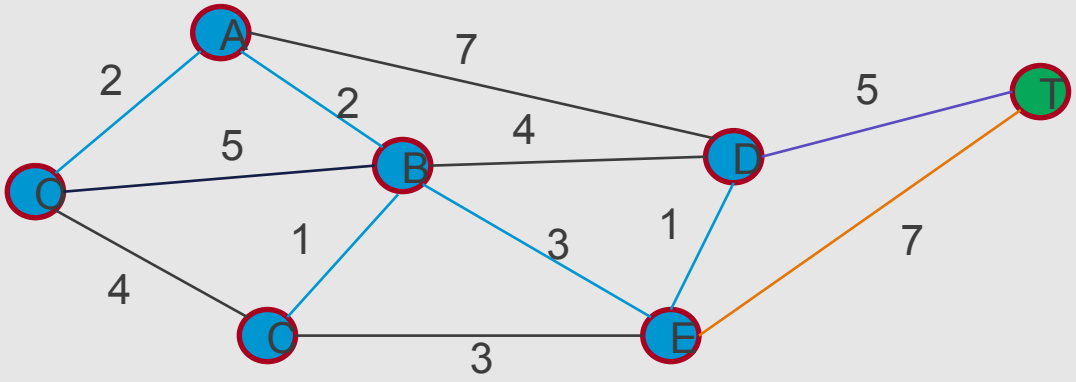
Prim's algorithm - example (step 7)



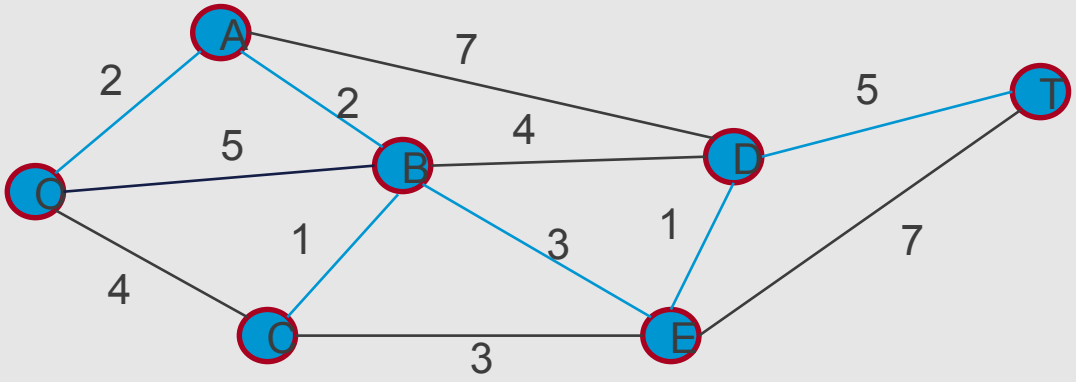
Prim's algorithm - example (step 8)



Prim's algorithm - example (step 9)



Prim's algorithm – example (step 10)



Dijkstra's algorithm (1956)

Let be a connected network with nodes numbered from to . Dijkstra's algorithms finds all shortest paths from the first node to every other node in .

Let

- , with and for ,
- , for
- , for

While

pick one

set

for all

if

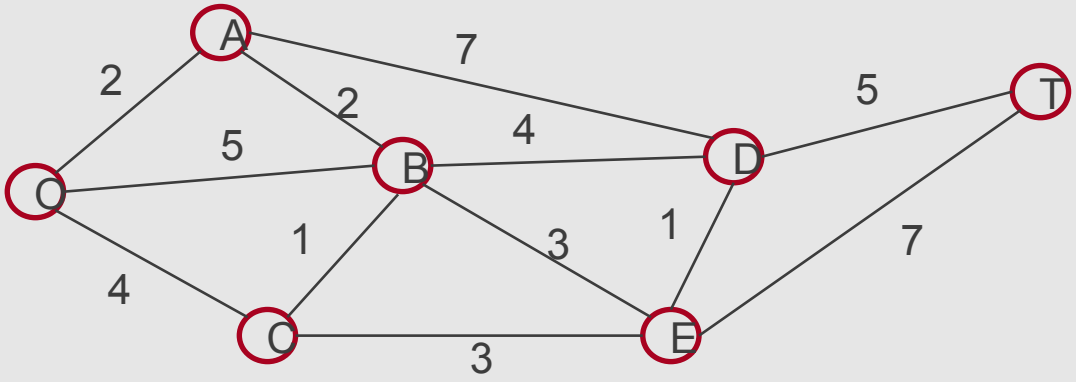
Dijkstra's algorithm (1956) in lay terms

Let ,, and , denote the distance, previous-node, and visited-node vectors, respectively.

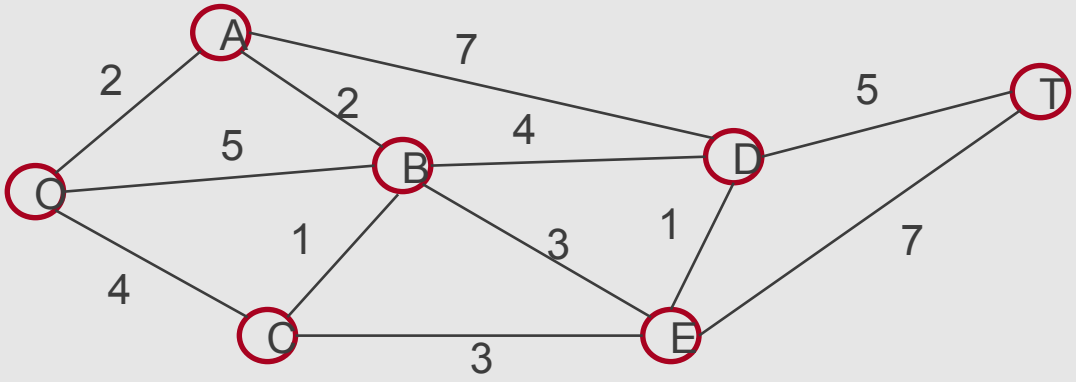
At each iteration:

1. pick the most recently visited node (say node)
2. find its neighbouring nodes that have not been visited yet
3. compute the distances to these neighbours assuming the preceding node is
4. update the distance and preceding node of these neighbours if passing through node is a shorter path
5. choose a node whose distance is a minimum among the un-visited nodes and mark it as visited. This is the new most recently visited node. Return to step 1 until all nodes are marked as visited.

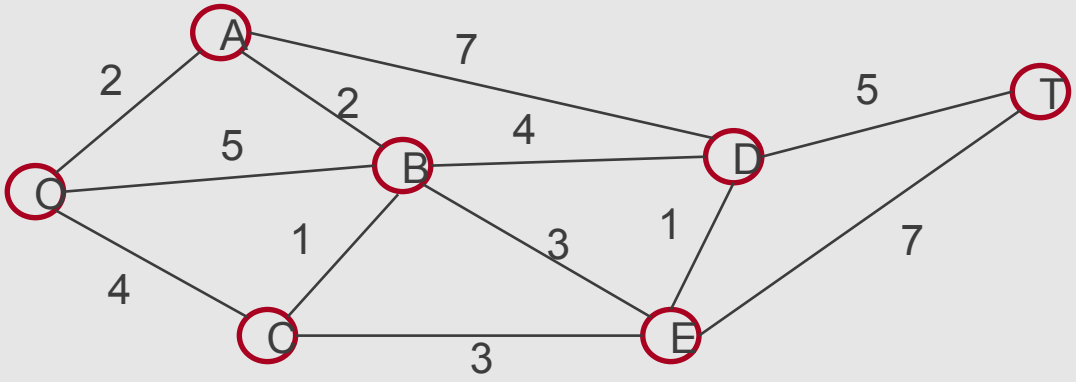
Dijkstra's algorithm – example (step 1)



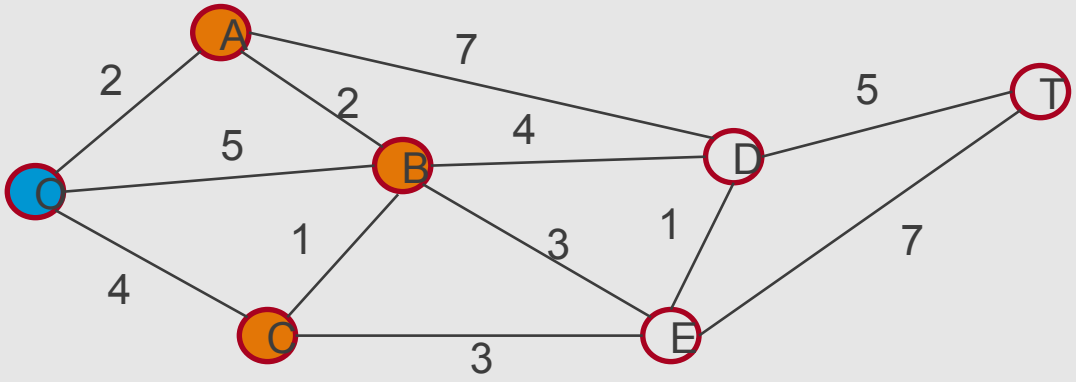
Dijkstra's algorithm – example (step 2)



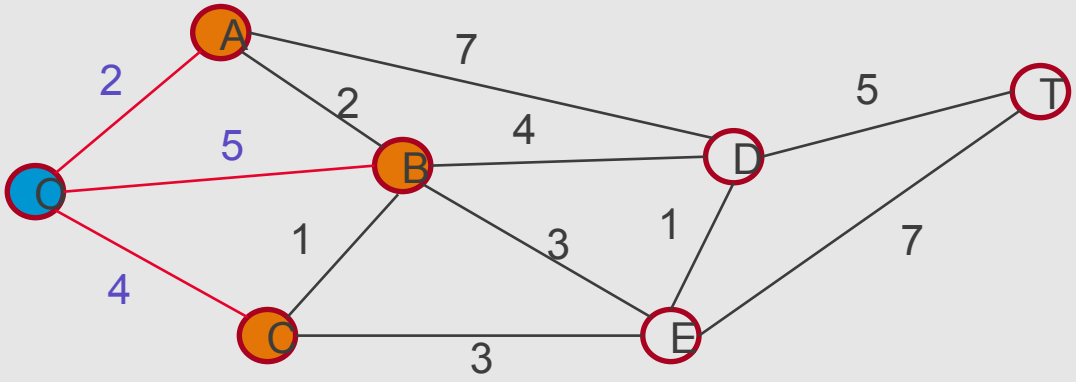
Dijkstra's algorithm – example (step 3)



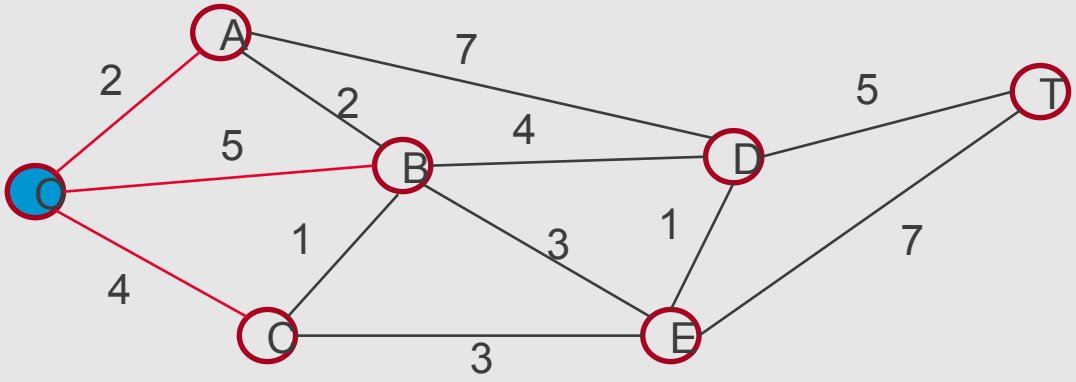
Dijkstra's algorithm – example (step 4)



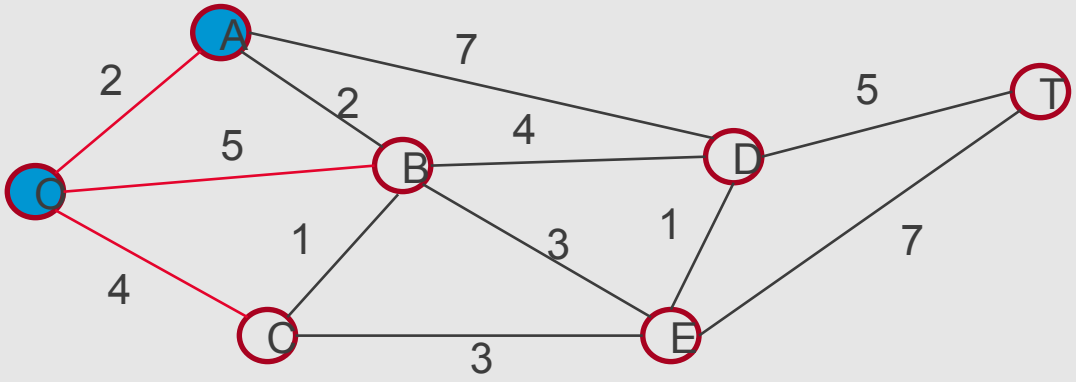
Dijkstra's algorithm – example (step 5)



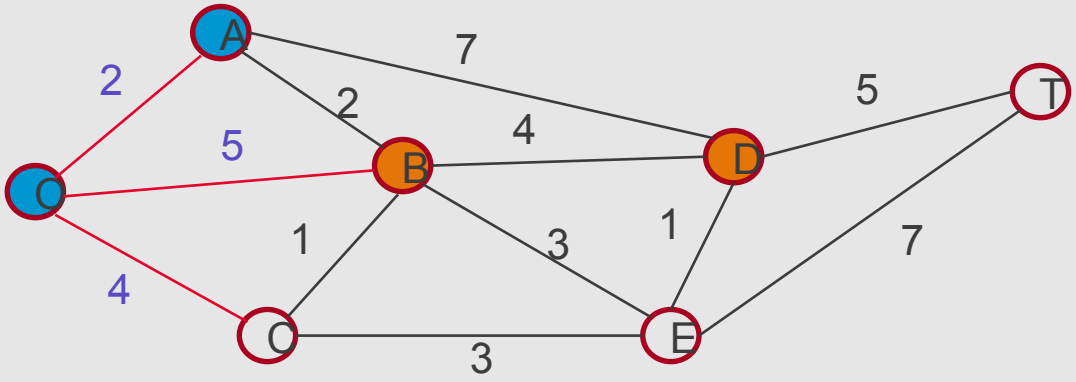
Dijkstra's algorithm – example (step 6)



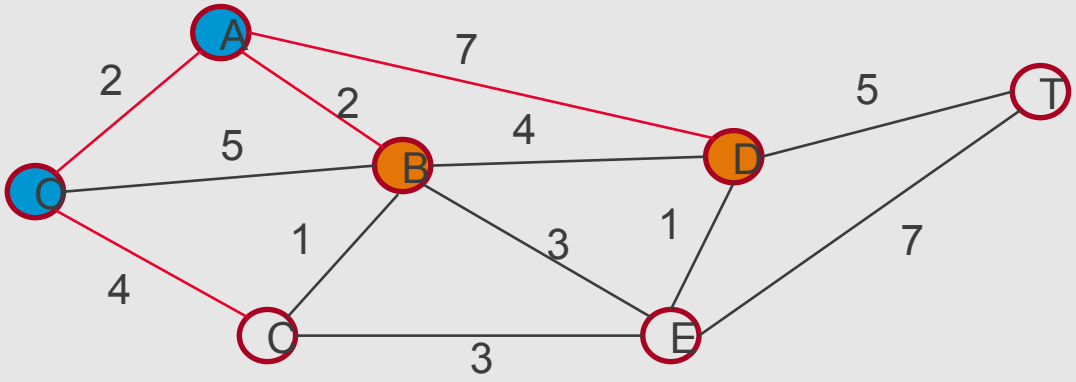
Dijkstra's algorithm – example (step 7)



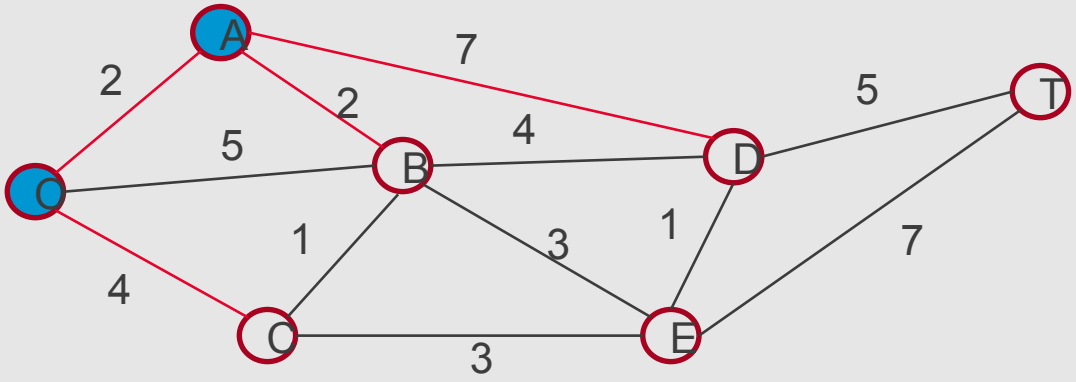
Dijkstra's algorithm – example (step 8)



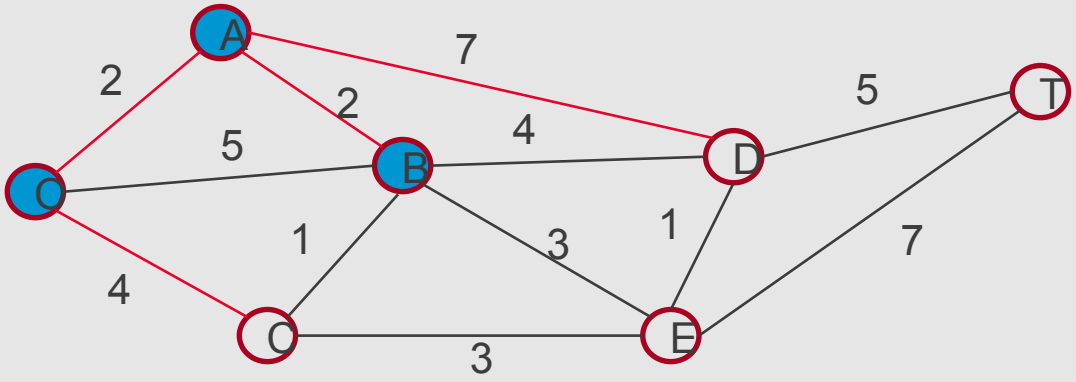
Dijkstra's algorithm – example (step 9)



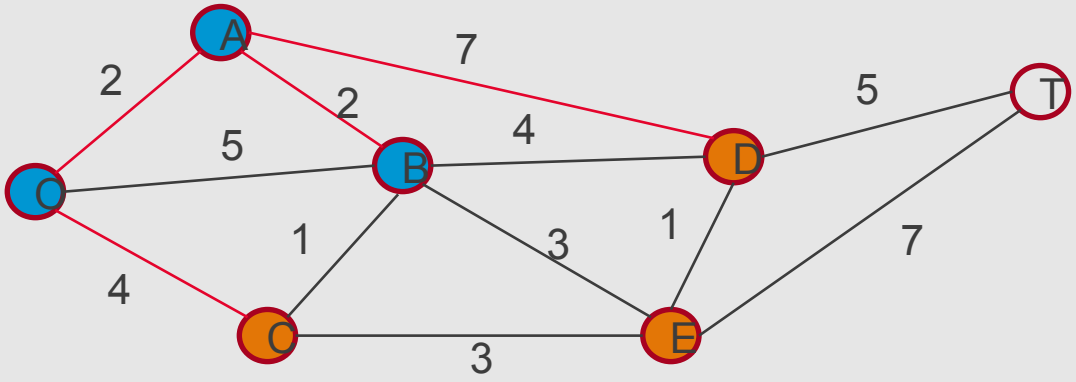
Dijkstra's algorithm – example (step 10)



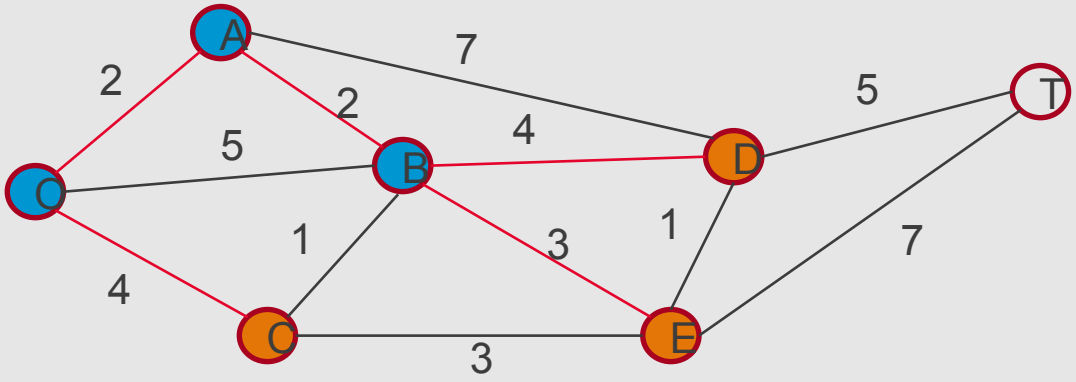
Dijkstra's algorithm – example (step 11)



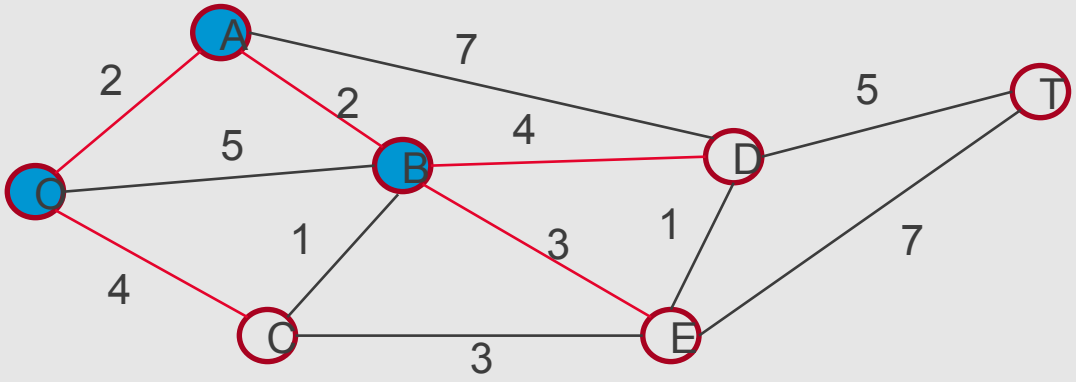
Dijkstra's algorithm – example (step 12)



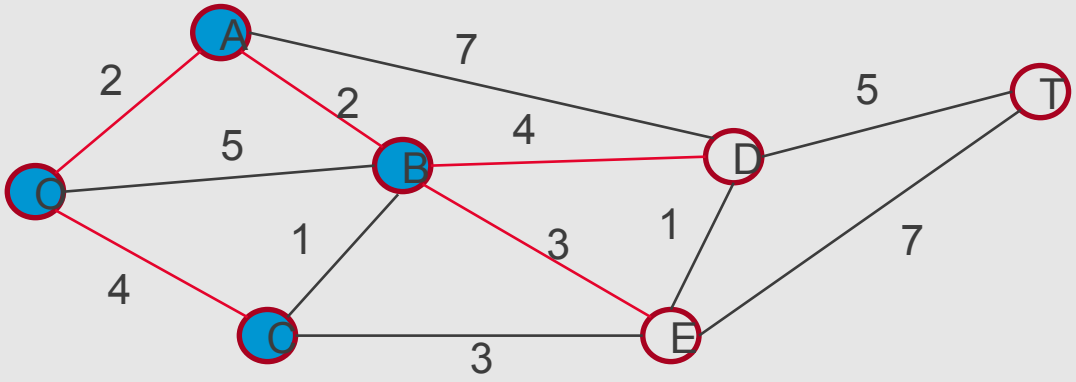
Dijkstra's algorithm – example (step 13)



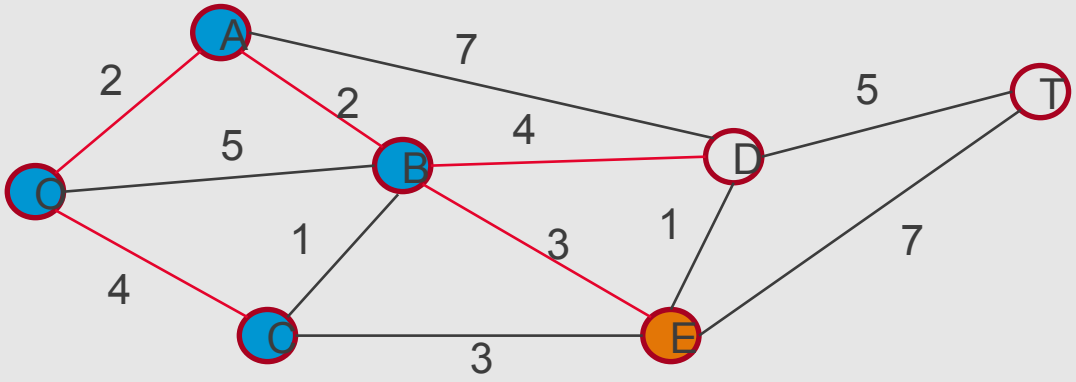
Dijkstra's algorithm – example (step 14)



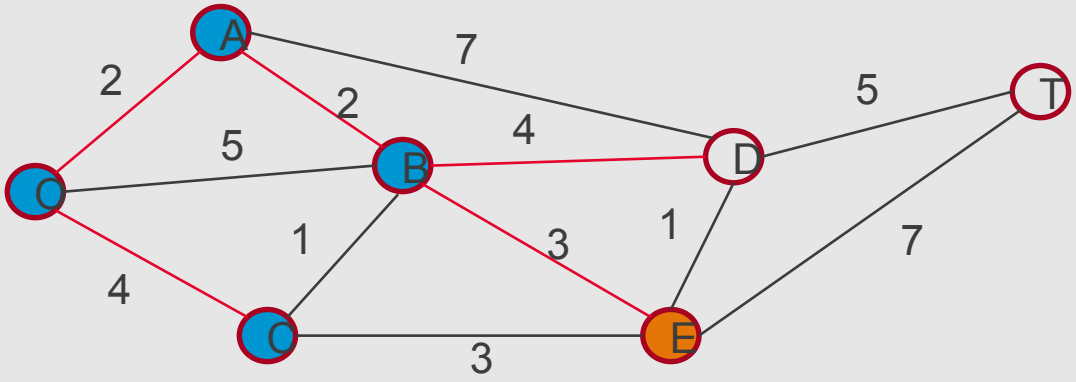
Dijkstra's algorithm – example (step 15)



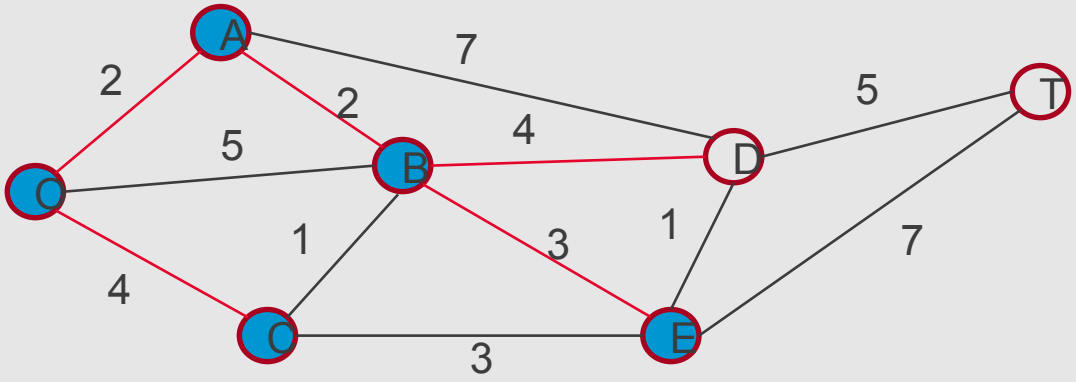
Dijkstra's algorithm – example (step 16)



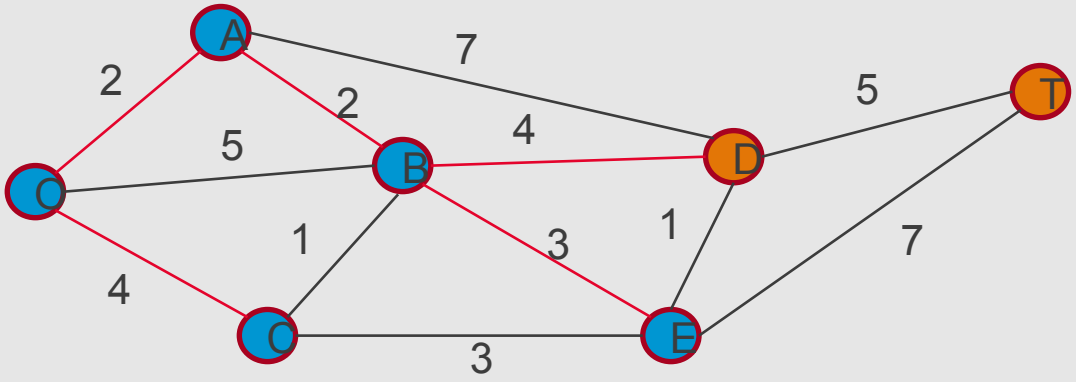
Dijkstra's algorithm – example (step 17)



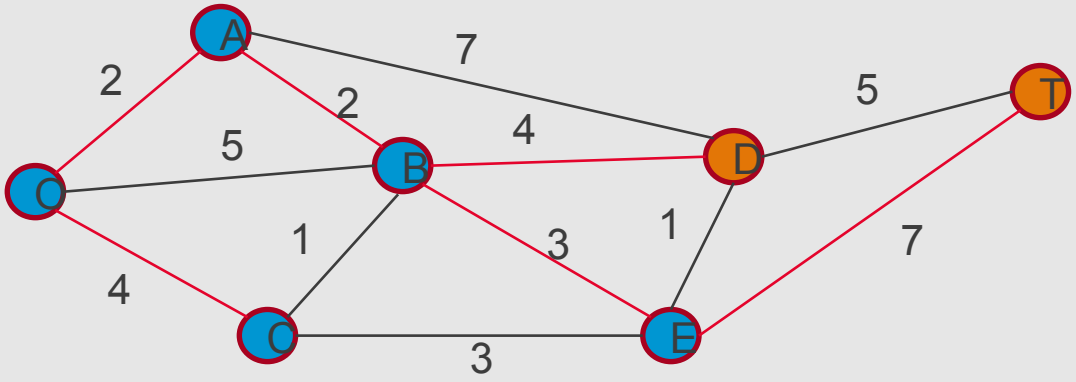
Dijkstra's algorithm – example (step 18)



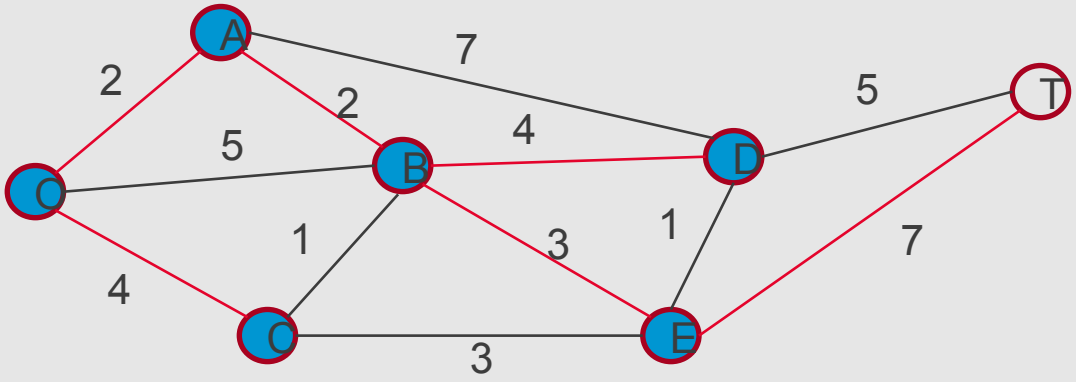
Dijkstra's algorithm – example (step 19)



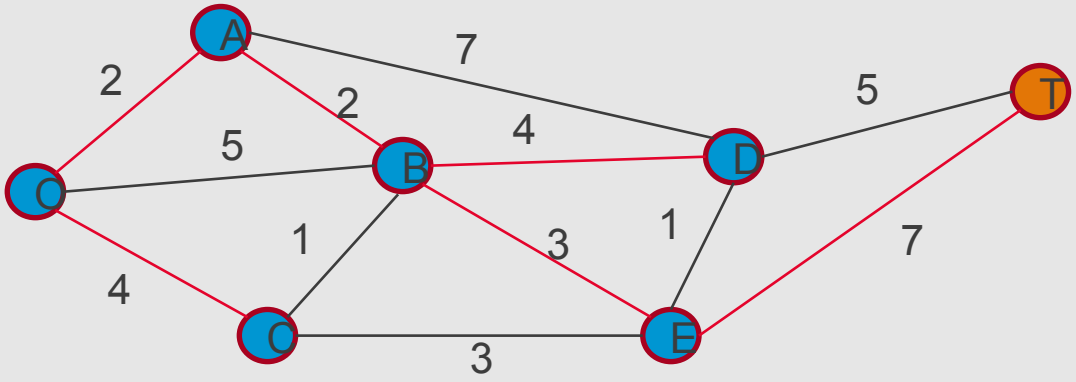
Dijkstra's algorithm – example (step 20)



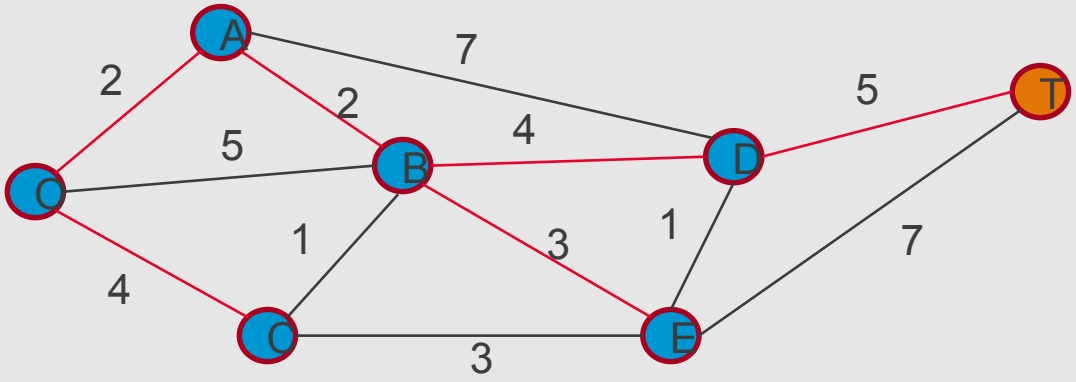
Dijkstra's algorithm – example (step 21)



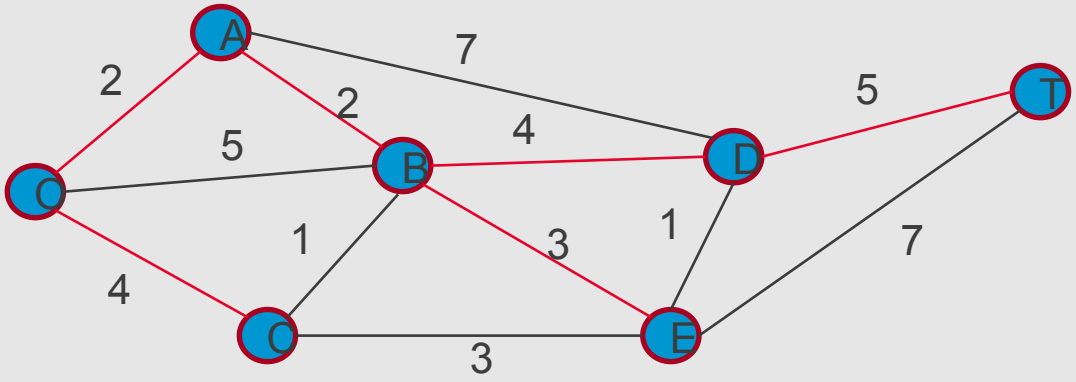
Dijkstra's algorithm – example (step 22)



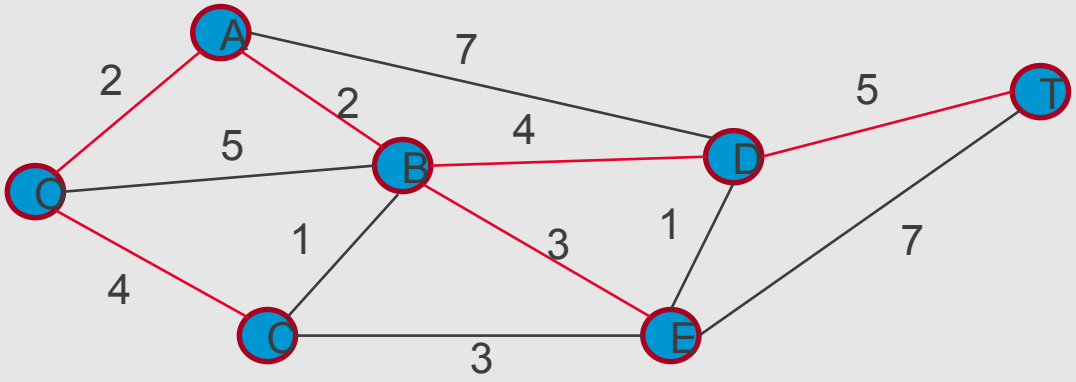
Dijkstra's algorithm – example (step 23)



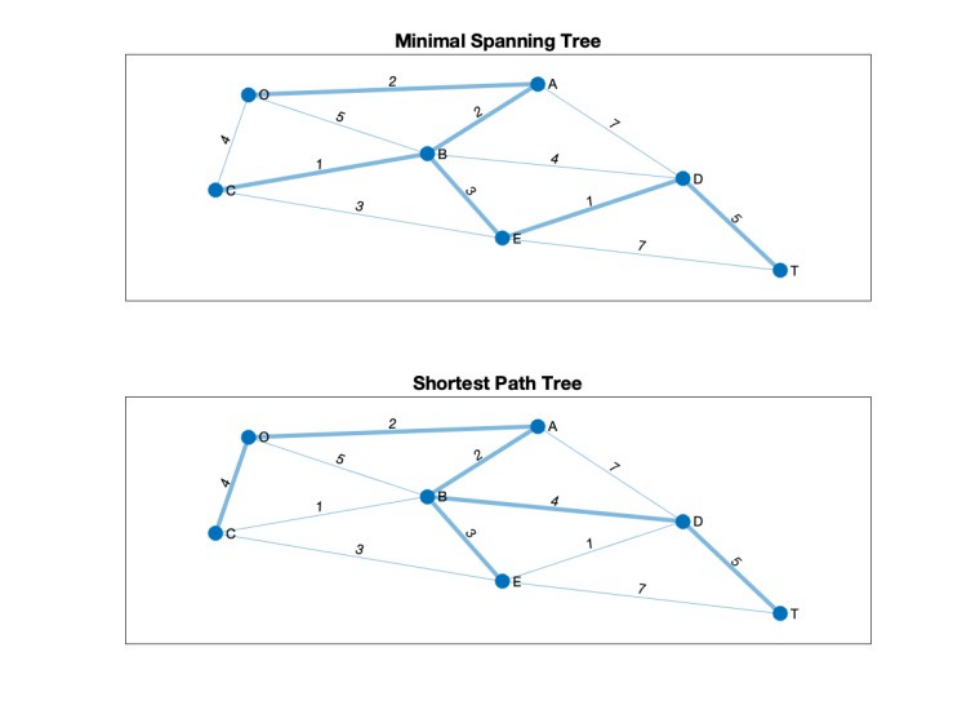
Dijkstra's algorithm – example (step 24)



Dijkstra's algorithm – example (step 25)



Minimal spanning and shortest path trees in Matlab
(see OR11_networks.m)



Summary and self-study

Summary: today we have learnt:

- about graphs and networks,
- how to determine the minimal spanning tree of a network,
- how to determine the shortest path tree of a network.

Self-study: Find the minimal spanning tree and the shortest path tree starting at node O of the network on the right.