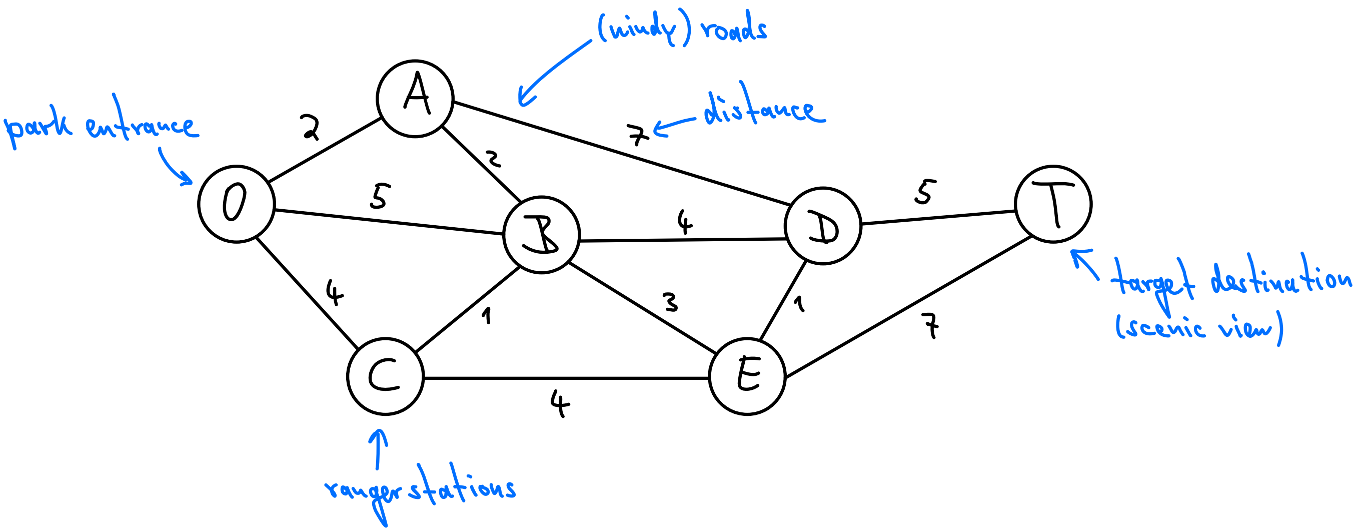
2.6 Network Optimization

Networks are everywhere: transportation, electricity, communication, ...

Network optimization problems are often special types of LP problems (as it was for the transportation problem).

Example to illustrate problem types:

Seervada Park (Hillier, Liebermann Chapter 9)



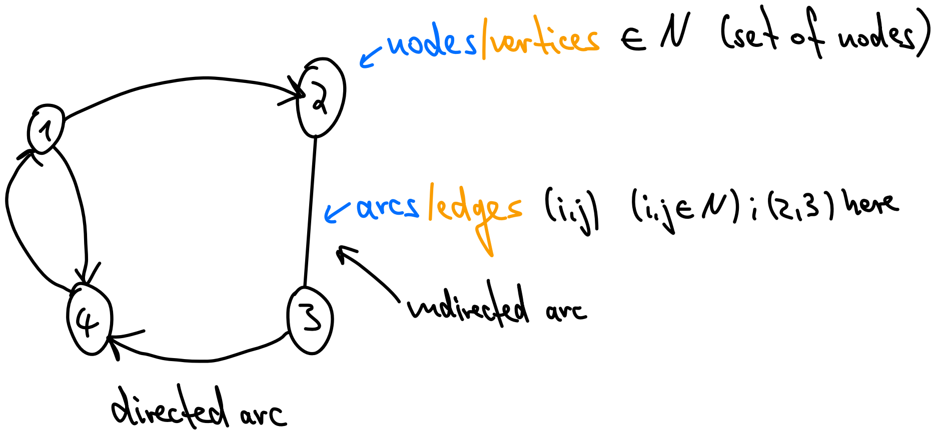
Types of problems:

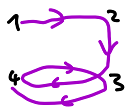
- Shortest path problem: Which route from O to T has smallest distance?
- Minimum spanning tree problem: Install telephone lines under roads so every pair of stations is connected, while minimizing the construction costs.
- Maximum flow problem: Limits are set on transportation via each road. Maximize number of trips ("visitor flow") from O to T.

First: some network terminology

OR language ↓   math language ↙

Network / Graph



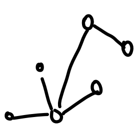
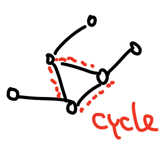
- **Path:** sequence of matching arcs, e.g.  $(1,2),(2,3),(3,4),(4,3),(3,4)$

- **Cycle:** path that begins and ends at same node, e.g., $(1,2),(2,3),(3,4),(4,1)$

- A network is **connected** if there is an undirected path between any two nodes; e.g., Seervada park above is connected,  is not connected.

- A **tree** is a connected network that has no cycles.

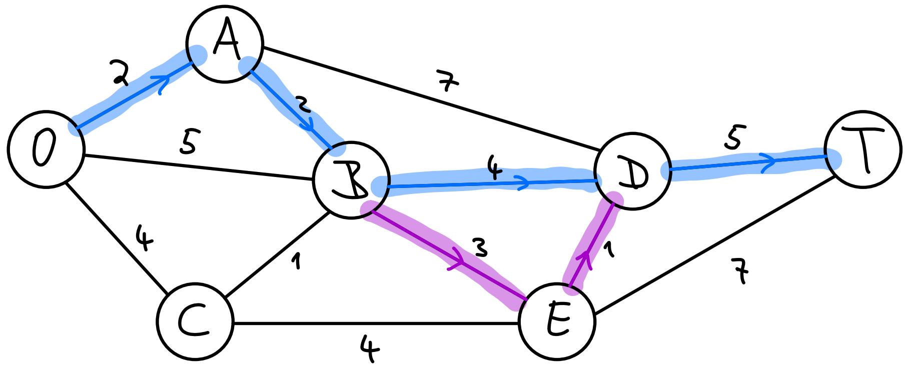
E.g.:

 tree   

not a tree

Next, we briefly discuss some special algorithms for the problem types above.
Afterwards, we discuss their LP formulation in a more general context.

• Shortest Path problem:



Goal: Find shortest path from O to T.

Algorithm: • keep a list of nodes with the shortest connection to O

• add new connections in each step

• stop once T is in the list

Here:

n (iteration step)	node	path	distance
1	A	0-A	2
2	B	0-A-B	4 → step 4
	or C	0-C	4 (⇒ 0-C-B (distance 5) excluded)
3	D	0-A-D	2+7=9 → step 4
	E	0-A-B-E	2+2+3=7 → step 4 → step 5
	E	0-C-E	4+4=8
4	D	0-A-D	9 (from before)
	D	0-A-B-D	4+4=8 → step 5
	D	0-A-B-E-D	7+1=8 → step 5
5	T	0-A-B-D-T or 0-A-B-E-D-T	$\boxed{8+5=13}$ $\boxed{8+5=13}$ two shortest paths
	T	0-A-B-E-T	7+7=14

- Minimum Spanning Tree problem:

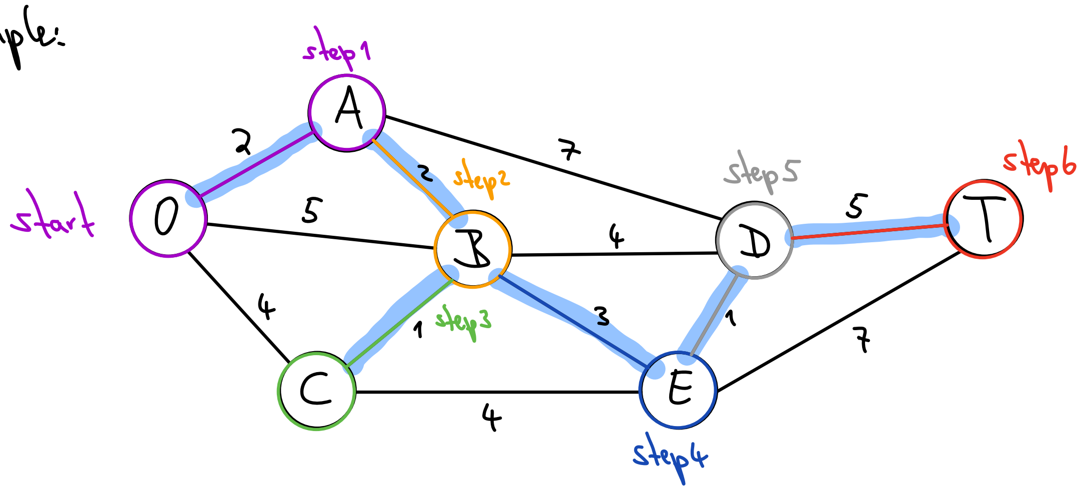
Similar to Shortest Path problem, but now path needs to connect each pair of nodes, with minimal cost (say, cost is proportional to distance here).

Note: for n nodes we need to find n-1 links, i.e., a tree connecting all nodes

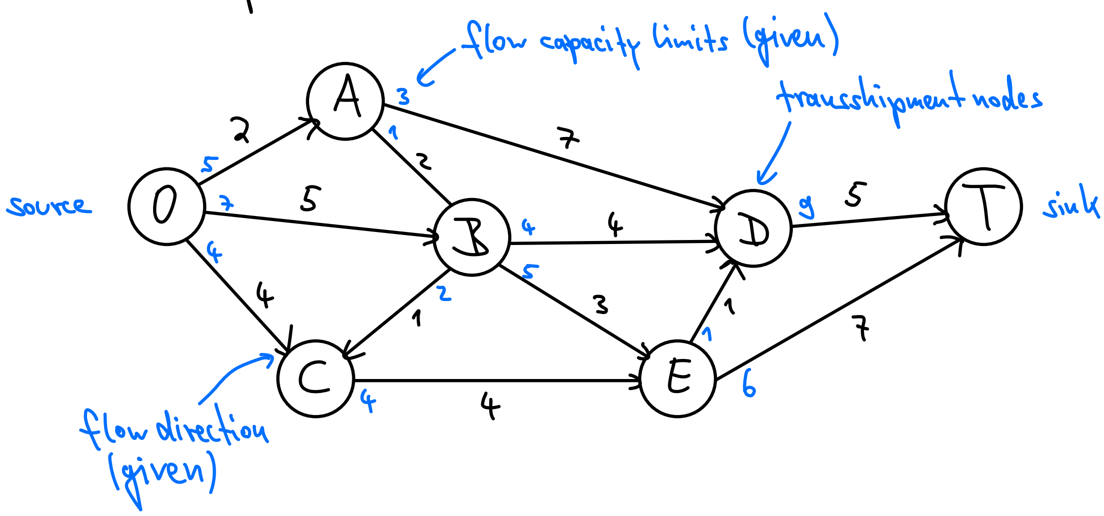
otherwise we could always delete a link to make cost smaller

Algorithm:
- start with any node, add link to nearest node
- connect linked notes to next nearest node; repeat

In our example:



- Maximum Flow problem:



Objective: maximize flow from source to sink