## 2.6 Network Optimization

Networks are everywhere: transportation, electricity, communication/internet, roads/trains

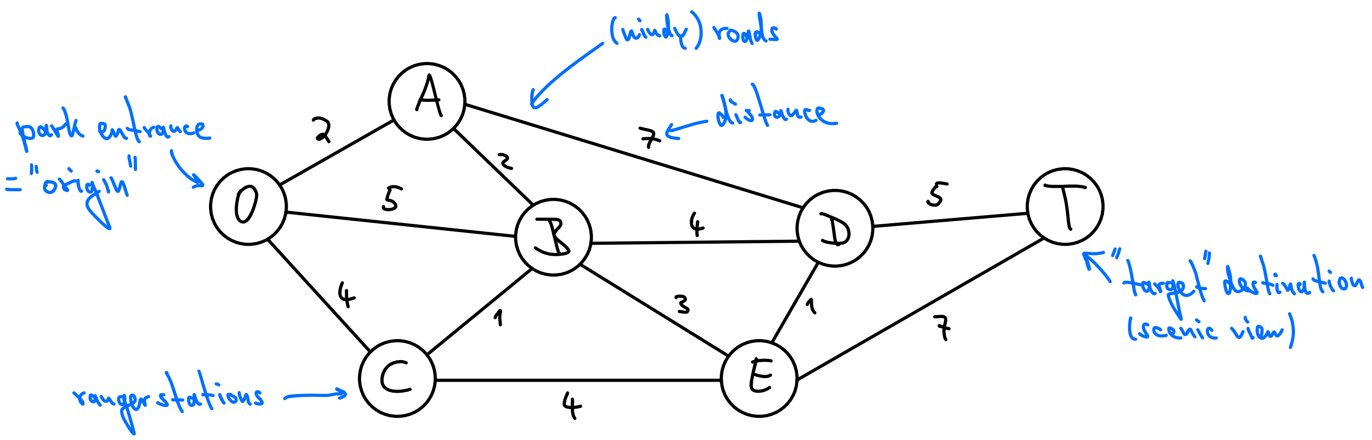
optimize costs, time

optimize coverage with links

optimize length of path, optimize capacities so flow is maximal

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, Lieberman Chapter 9)



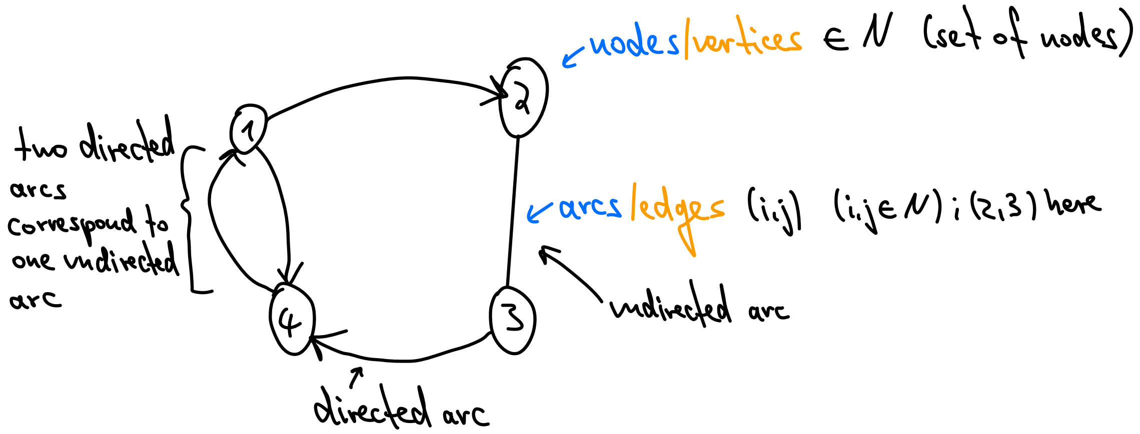

Types of problems:

- Shortest path problem: Which route from O to T has smallest distance?
- Minimum spanning tree problem: Install communication 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.

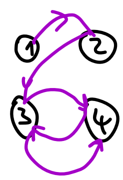
First: some network terminology

OR language ↓ math language ↙

Network / Graph



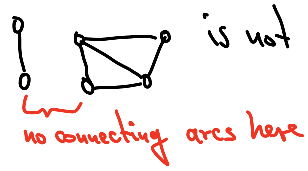

- **Path:** sequence of matching arcs, e.g.




$(1,2),(2,3),(3,4),(4,3),(3,4)$

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

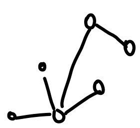
- A network is **connected** if there is an undirected path between any two nodes; e.g., Seervada park above is connected,



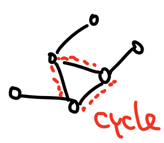

is not connected.

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

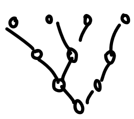
E.g.:



tree



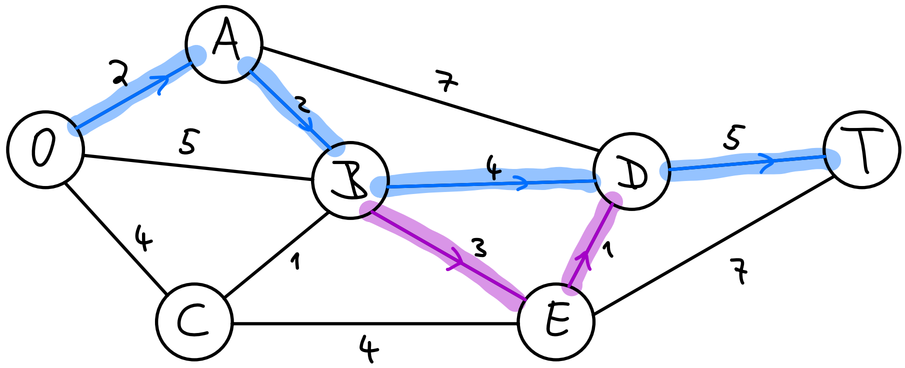

not a tree



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:

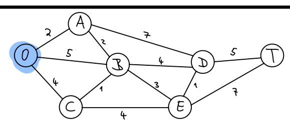
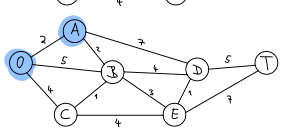
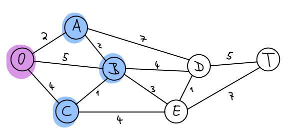
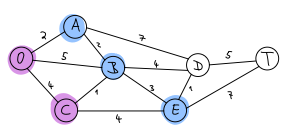
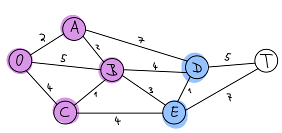



Goal: Find shortest path from O to T.

Algorithm (Dijkstra's algorithm):

- Start with O as a "solved node"; all others are "unsolved nodes".
- List "candidates": For each solved node, select the shortest connection to any unsolved node.
  Out of all these, select the shortest total connection, this becomes a new solved node.
  (In case of a tie, we have several new solved nodes.)
- Repeat with new set of solved nodes until T is reached (i.e., until T is a solved node).

Table for our example:

| Iteration step n (total # of solved nodes) | Solved nodes (directly connected to unsolved nodes) | Closest unsolved node | Total distance | n-th nearest node (=new solved node in next step) | Min. distance | Last connection |
|---|---|---|---|---|---|---|
| 1 | O  | A | 2 | A | 2 | O-A |
| 2,3 | O<br>A  | C<br>B | 4<br>2+2=4 | C<br>B | 4<br>4 | O-C<br>A-B |
| 4 | A<br>B<br>C  | D<br>E<br>E | 2+7=9<br>4+3=7<br>4+4=8 | E | 7 | B-E |
| 5 | A<br>B<br>E  | D<br>D<br>D | 2+7=9<br>4+4=8<br>7+1=8 | D<br>D | 8<br>8 | B-D<br>E-D |
| 6 | D<br>E  | T<br>T | 8+5=13<br>7+7=14 | T | 13 | D-T |

Purple: solved nodes that are not connected to any unsolved nodes (and can thus be neglected in further steps

Blue: solved nodes that are connected to unsolved nodes

=> Shortest paths: O-A-B-E-D-T and O-A-B-D-T with total distance 13