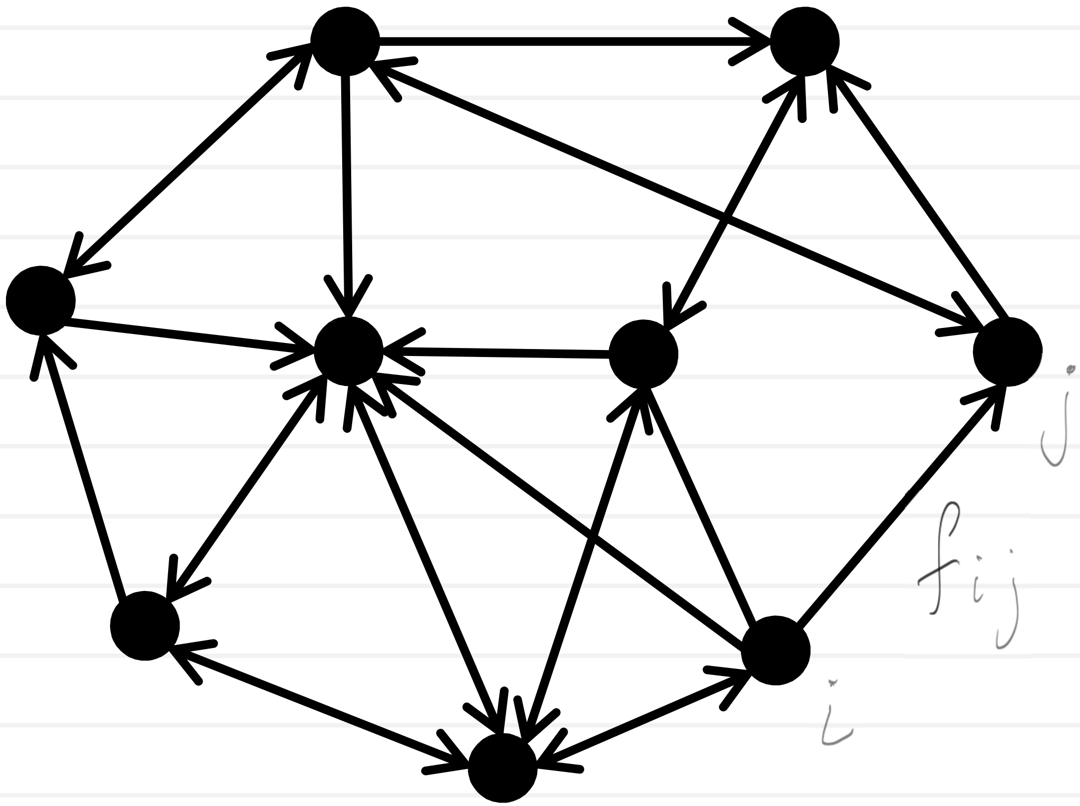
Lecture 7: Optimal network flows

Network flows



f_{ij} – the flow from i to j

$$f_{ij} \neq f_{ji}$$

u_{ij} – capacity of arc ij

b_i – inflow at vertex i (can be negative)

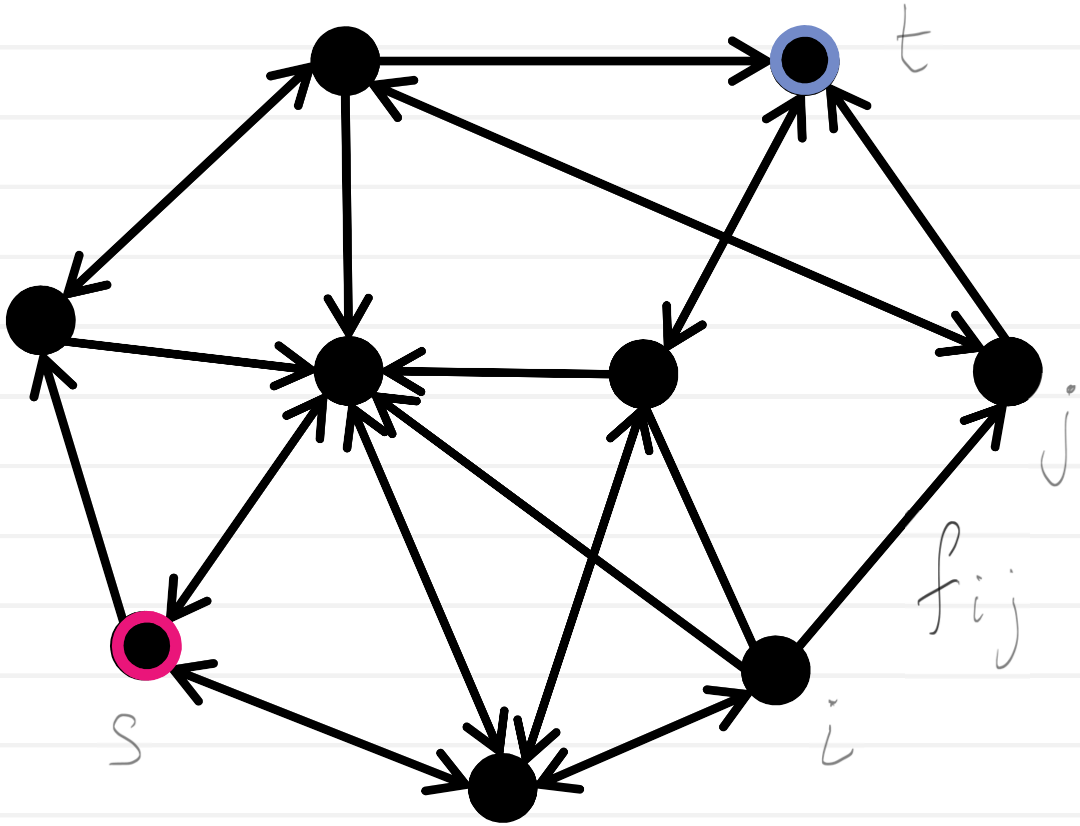
$I(i)$ – vertices with arcs directed to vertex i

$O(i)$ – vertices with arcs directed from vertex i

Definition: f is a feasible flow if

$0 \leq f_{ij} \leq u_{ij}$ and $b_i + \sum_{j \in I(i)} f_{ji} = \sum_{j \in O(i)} f_{ij}$

Total inflow



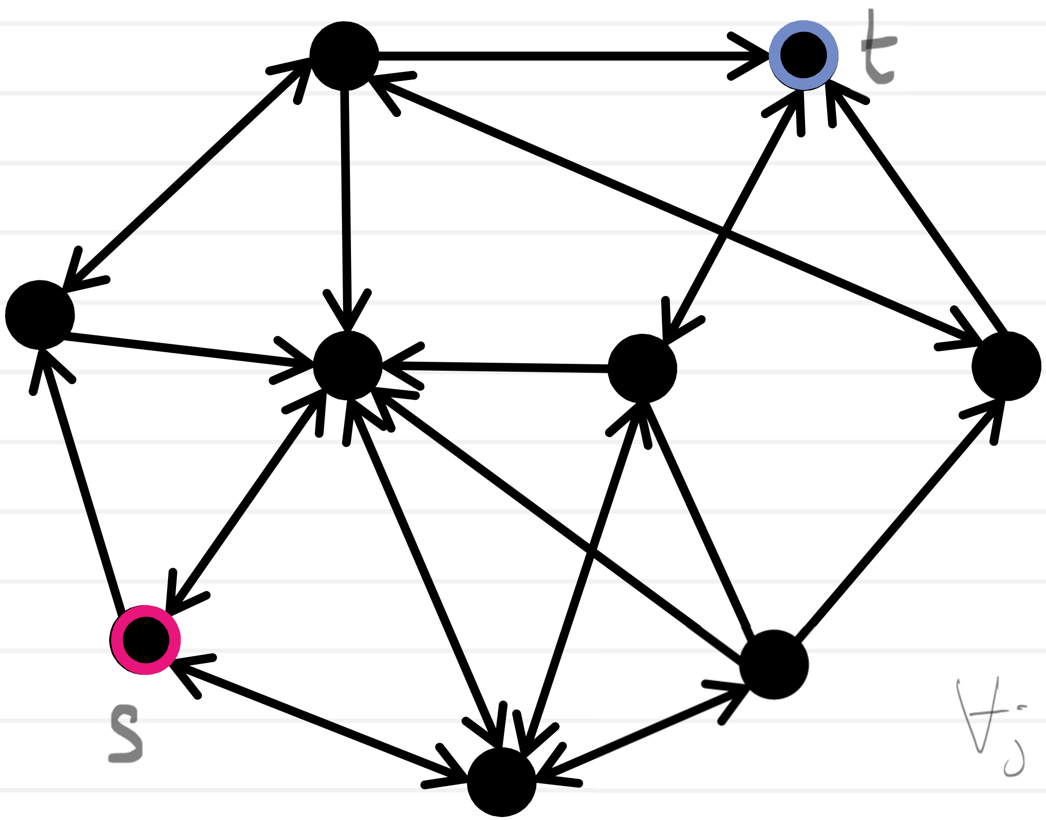
Sources are vertices with positive inflow, sinks are vertices with negative inflow.

Statement: if the network admits feasible flows then

$$\sum_{i=1}^{N} b_i = 0$$

Maximum flow (maxflow)

Maximum flow problem: given arc capacities, a single source vertex s, and a single sink vertex t, find a maximal flow (that maximizes the inflow at s and outflow at t).



$$\max_{b,f} \; b$$

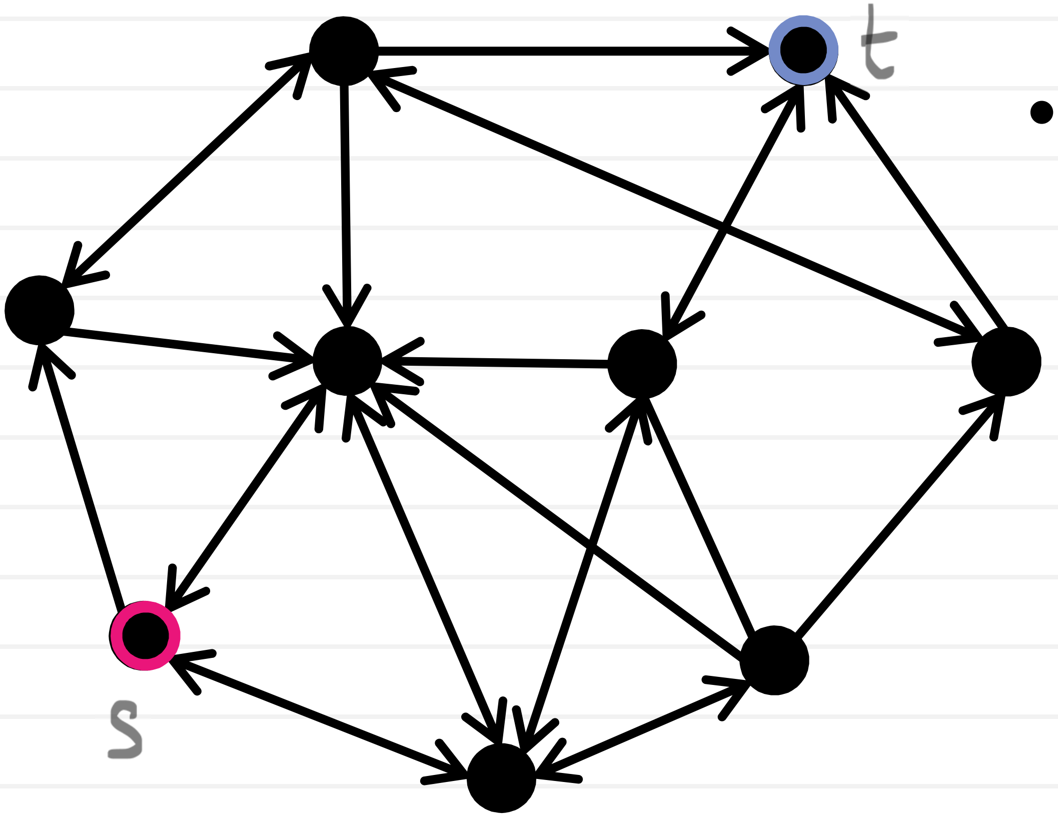
$$\text{s.t.:} \quad 0 \le f_{ij} \le u_{ij}$$

$$b + \sum_{i \in I(s)} f_{is} = \sum_{i \in O(s)} f_{si}$$

$$\sum_{i \in I(t)} f_{it} = \sum_{i \in O(t)} f_{ti} + b$$

$$\forall j \ne s,t \quad \sum_{i \in I(j)} f_{ij} = \sum_{i \in O(j)} f_{ji}$$

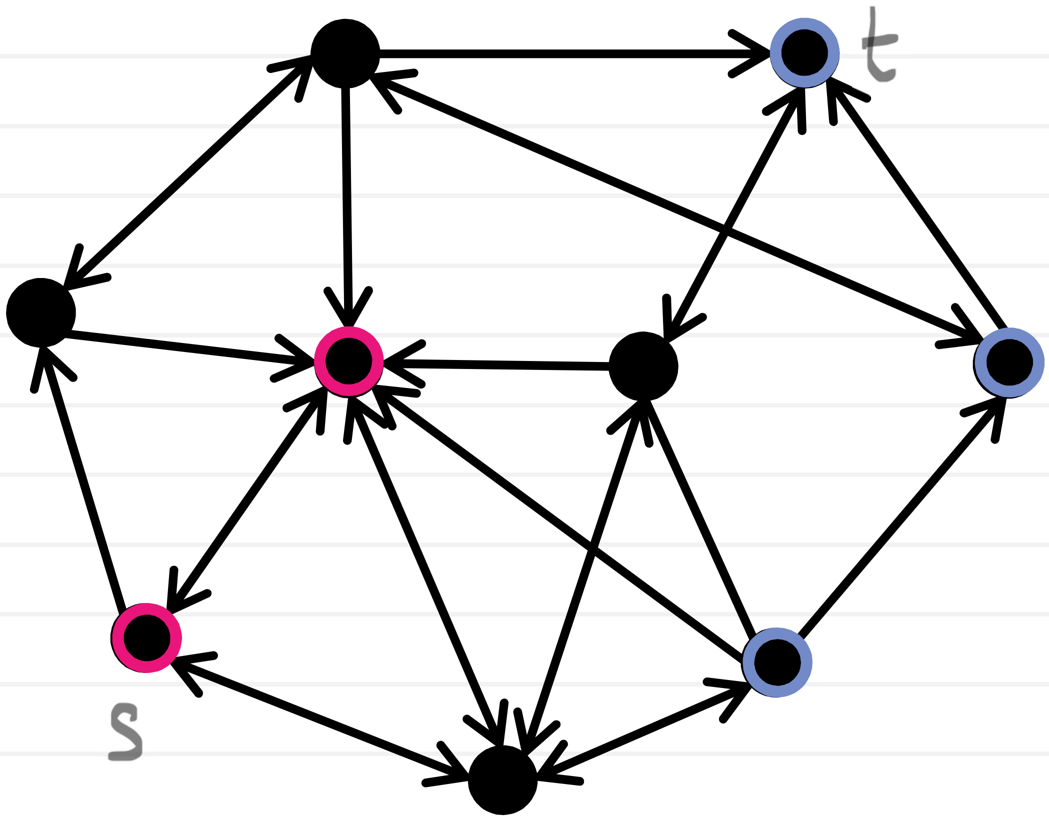
Maxflow interpretation



- Arc capacities are maximum amount of water that can be pumped.
- We try to pump as much water as possible from s to t.

Minimum cost flow (mincost flow)

Minimum cost flow: given arc capacities, a set of inflows, and *arc costs*, find a feasible flow that minimizes the total cost.

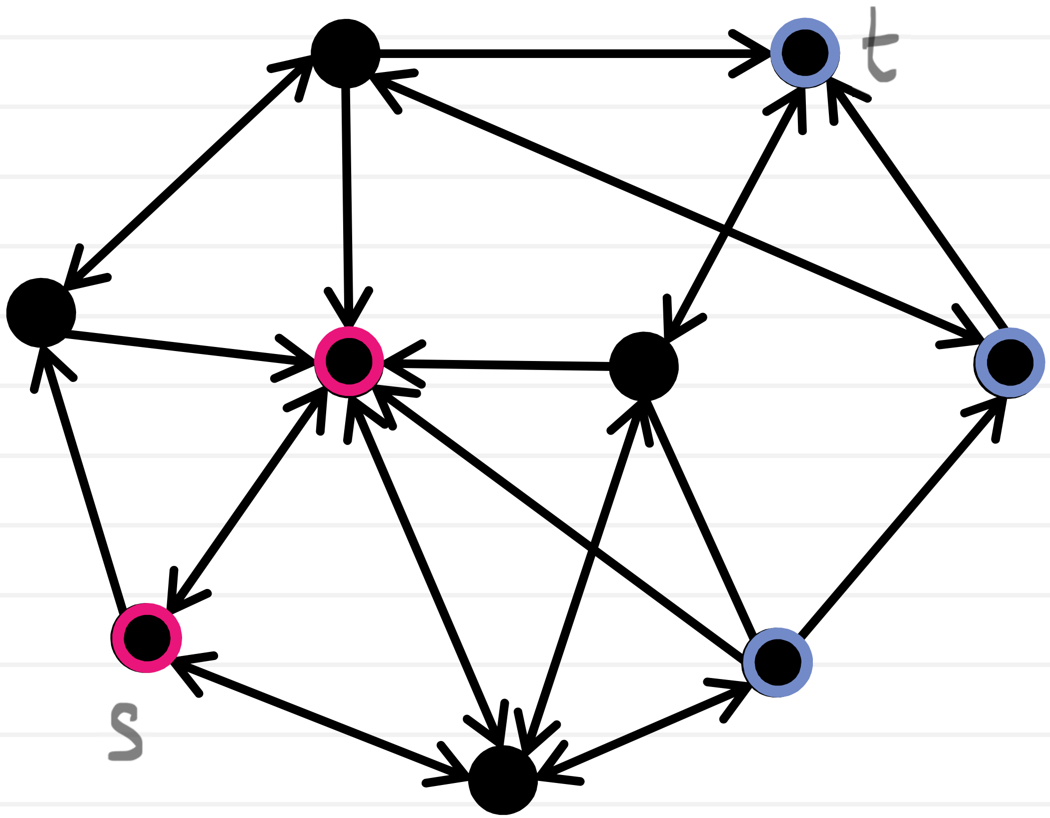


$$\min_f \; c^T f = \sum_{i,j} c_{ij} f_{ij}$$

$$\text{s.t.:} \quad 0 \le f_{ij} \le u_{ij}$$

$$b_j + \sum_{i \in I(j)} f_{ij} = \sum_{i \in O(j)} f_{ji}$$

Min-cost flow: transportation networks



$$\min_{f} c^T f = \sum_{i,j} c_{ij} f_{ij}$$

$$\text{s.t.:} \quad 0 \le f_{ij} \le u_{ij}$$

$$b_j + \sum_{i \in I(j)} f_{ij} = \sum_{i \in O(j)} f_{ji}$$

Deliver goods from producers to consumers subject to road capacities at the minimum transportation cost *(transshipment problem).*

Integrality

$$\max_{b,f} \; b$$

$$\text{s.t.:} \quad 0 \le f_{ij} \le u_{ij}$$

$$b + \sum_{i \in I(s)} f_{is} = \sum_{i \in O(s)} f_{si}$$

$$\sum_{i \in I(t)} f_{it} = \sum_{i \in O(t)} f_{ti} + b$$

$$\forall j \ne s,t \quad \sum_{i \in I(j)} f_{ij} = \sum_{i \in O(j)} f_{ji}$$

Corollary: if u_{ij} are integer, then all flow values are integer.

$$\min_{f} \; c^T f$$

$$\text{s.t.:} \quad 0 \le f_{ij} \le u_{ij}$$

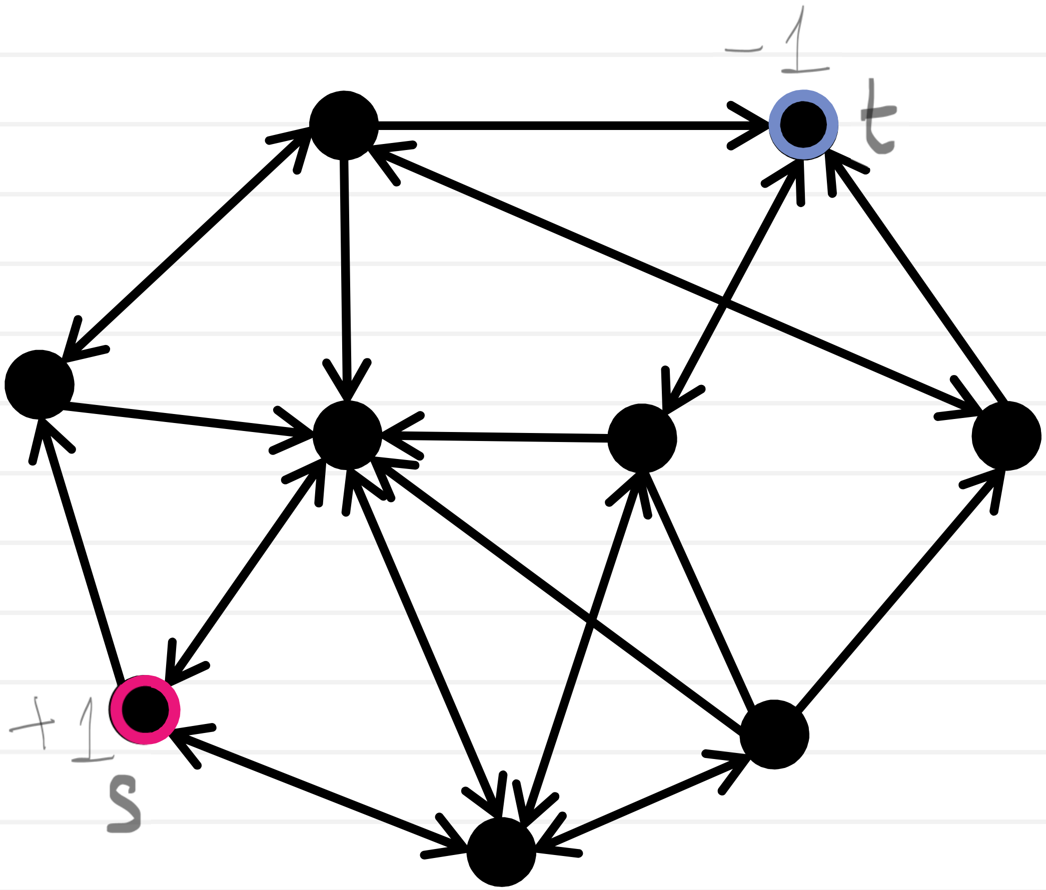
$$b_j + \sum_{i \in I(j)} f_{ij} = \sum_{i \in O(j)} f_{ji}$$

Corollary: if b_j and u_{ij} are integer, then all flow values are integer.

Shortest path

Shortest path can be cast as an instance of min-cost flow (with unit excess/demand).

arc costs without negative cycles



$$\min_f \; c^T f$$

$$\text{s.t.:} \quad 0 \le f_{ij} \le +\infty$$

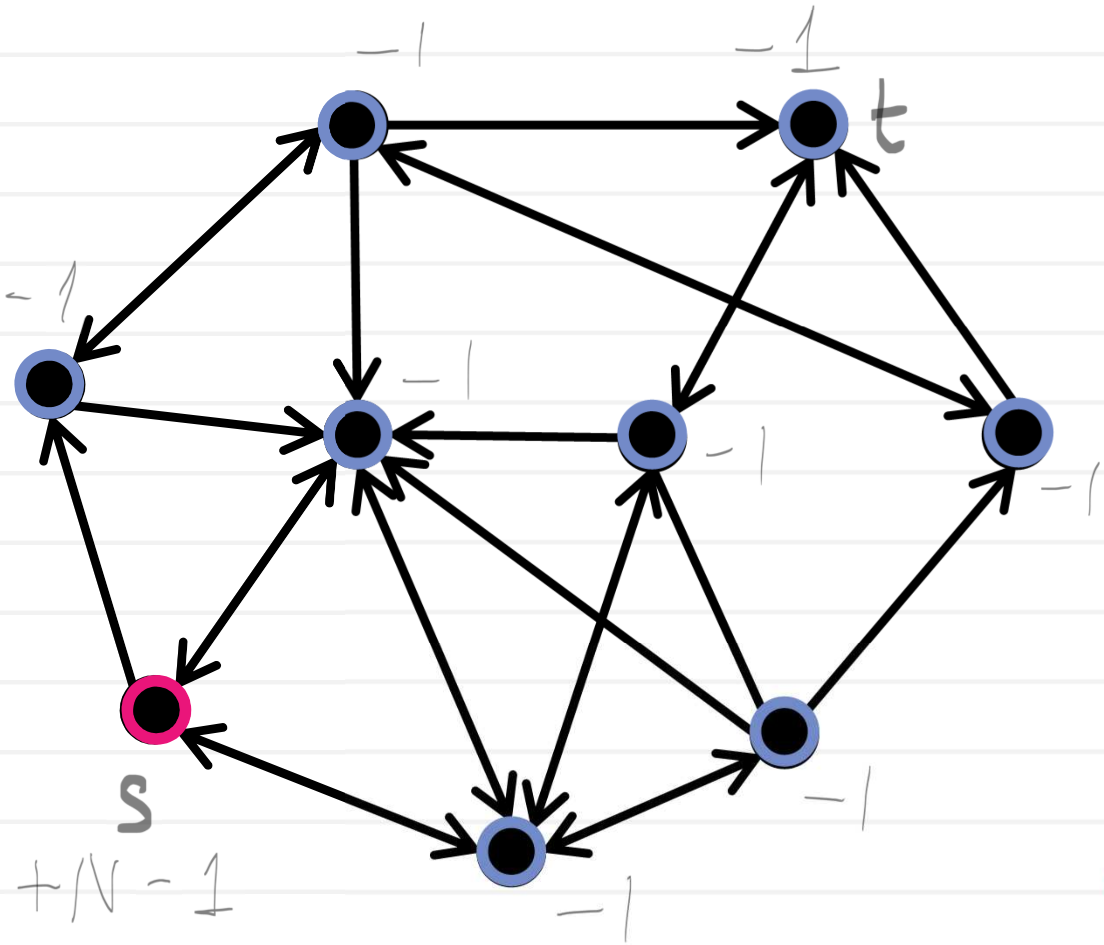
$$b_j + \sum_{i \in I(j)} f_{ij} = \sum_{i \in O(j)} f_{ji}$$

$$b_s = +1 \qquad b_t = -1$$

Shortest paths to all vertices

Shortest path can be cast as an instance of min-cost flow (with unit excess/demand).

arc costs without negative cycles



$$\min_f \; c^T f$$

$$\text{s.t.:} \quad 0 \le f_{ij} \le +\infty$$

$$b_j + \sum_{i \in I(j)} f_{ij} = \sum_{i \in O(j)} f_{ji}$$

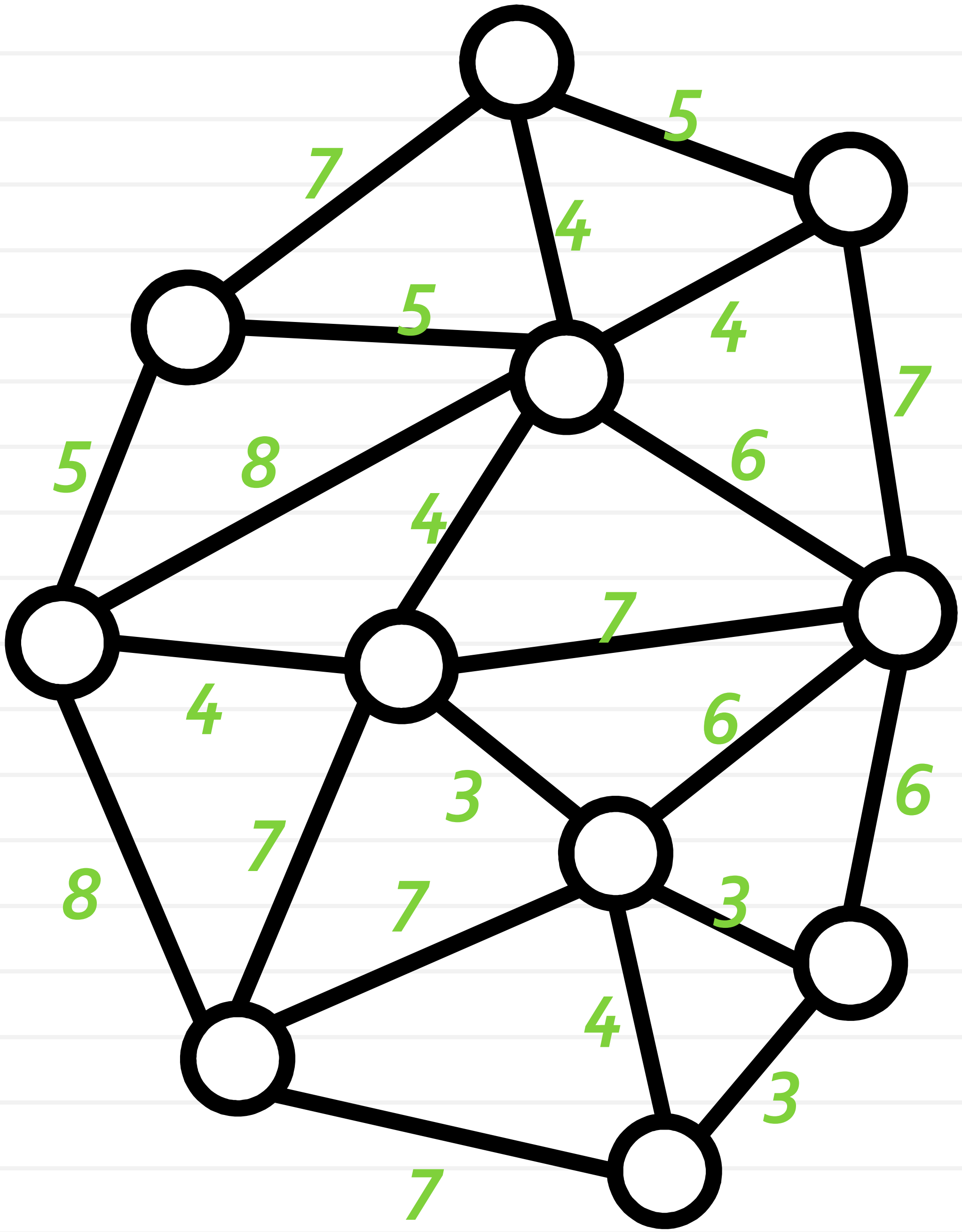
$$b_s = +N-1 \qquad \forall i \ne s \;\; b_i = -1$$

Finding optimal flows

How to solve network flow problems (min-cost flow, maxflow, shortest path)?

- Generic LP solver
- Optimized LP solver (e.g. *network simplex*) with integer arithmetics. Interestingly, basic solutions correspond to spanning trees containing all arcs with non-zero flow.
- Optimized algorithms specific to each problem (Dijkstra, Bellman-Ford, Ford-Fulkerson, push-relabel,…)

Dijkstra algorithm for shortest paths



non-negative

arc costs

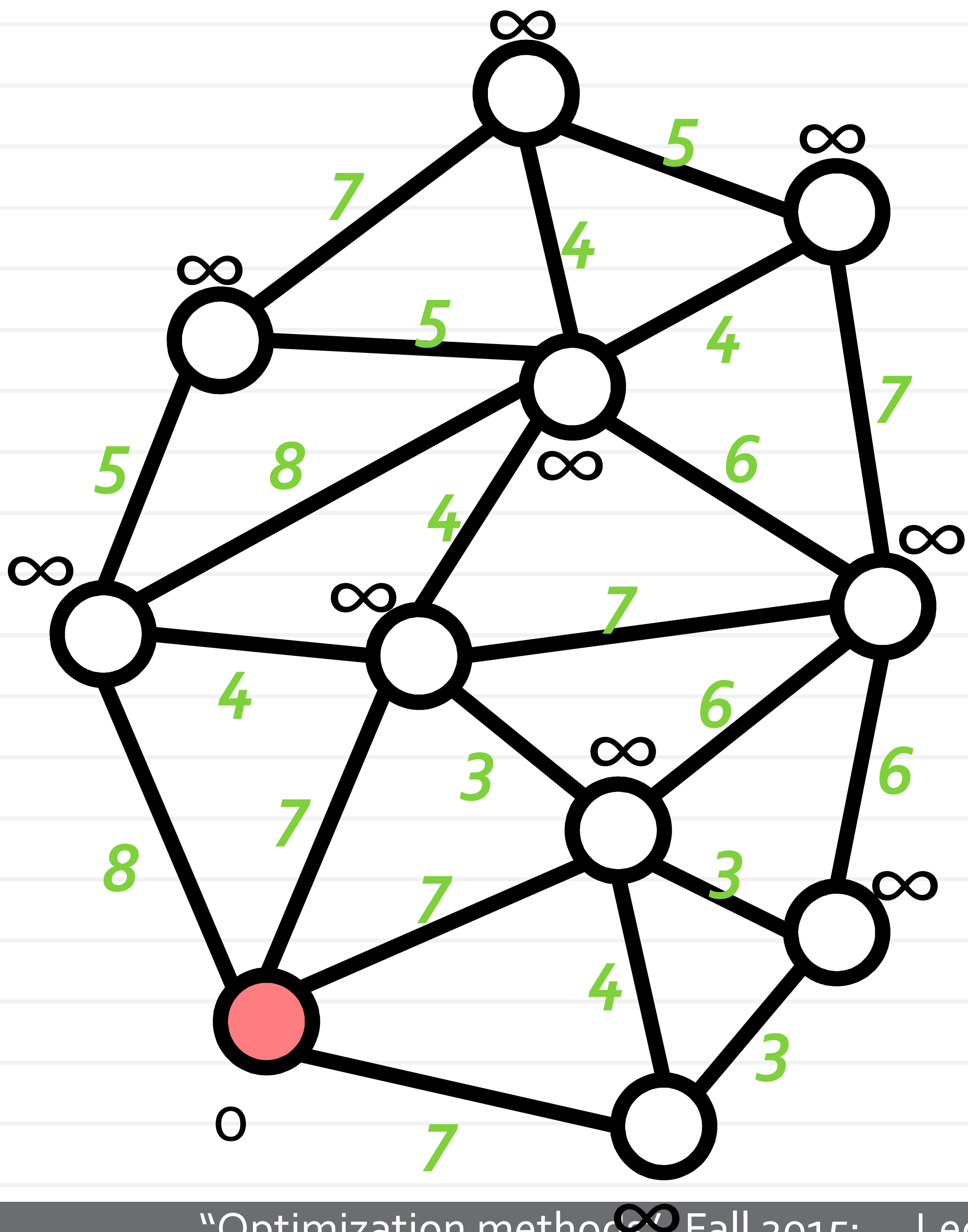
$$\min_f \; c^T f$$

$$\text{s.t.:} \quad 0 \le f_{ij} \le +\infty$$

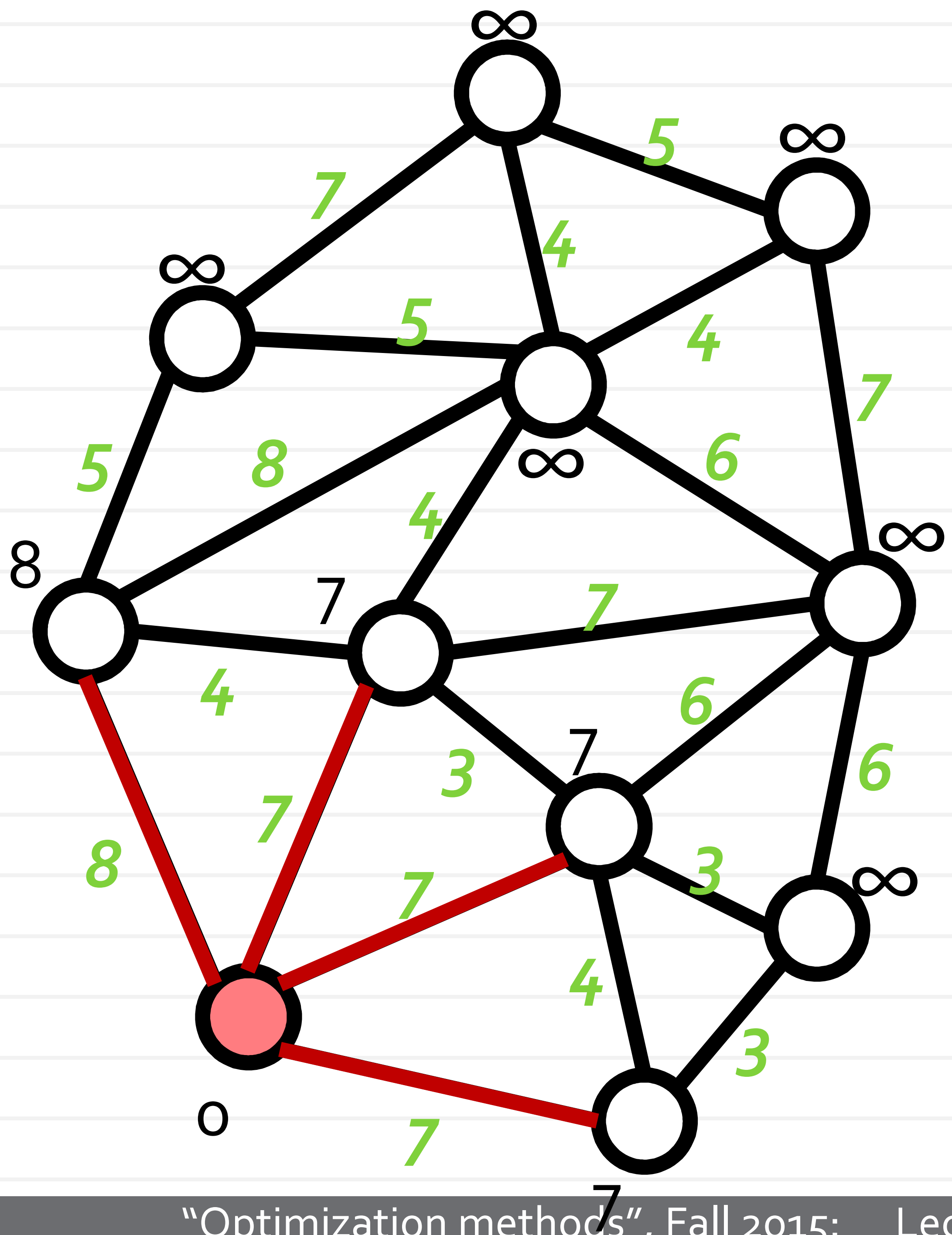
$$b_j + \sum_{i \in I(j)} f_{ij} = \sum_{i \in O(j)} f_{ji}$$

$$b_s = +N-1 \qquad \forall i \neq s \;\; b_i = -1$$

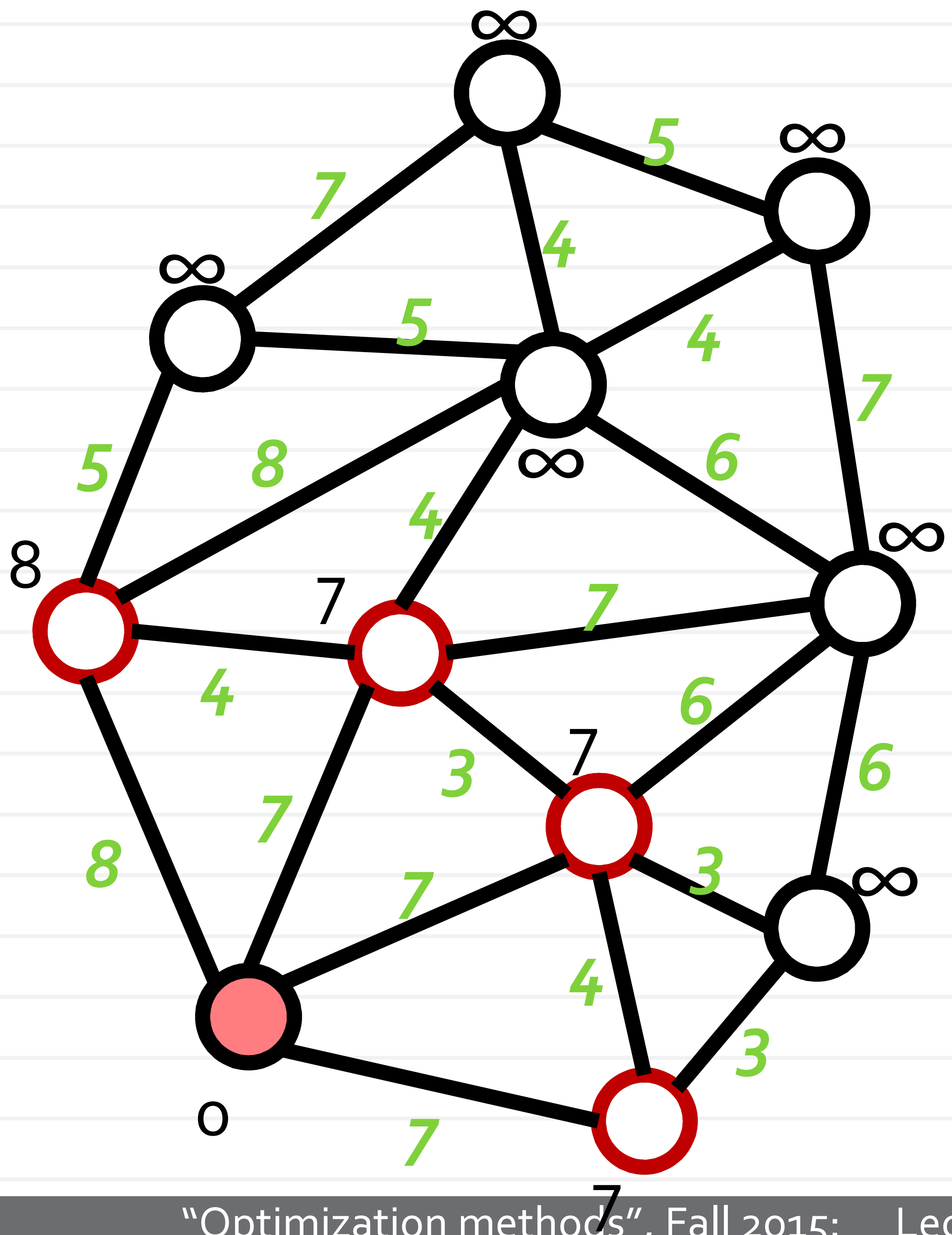
Dijkstra Algorithm



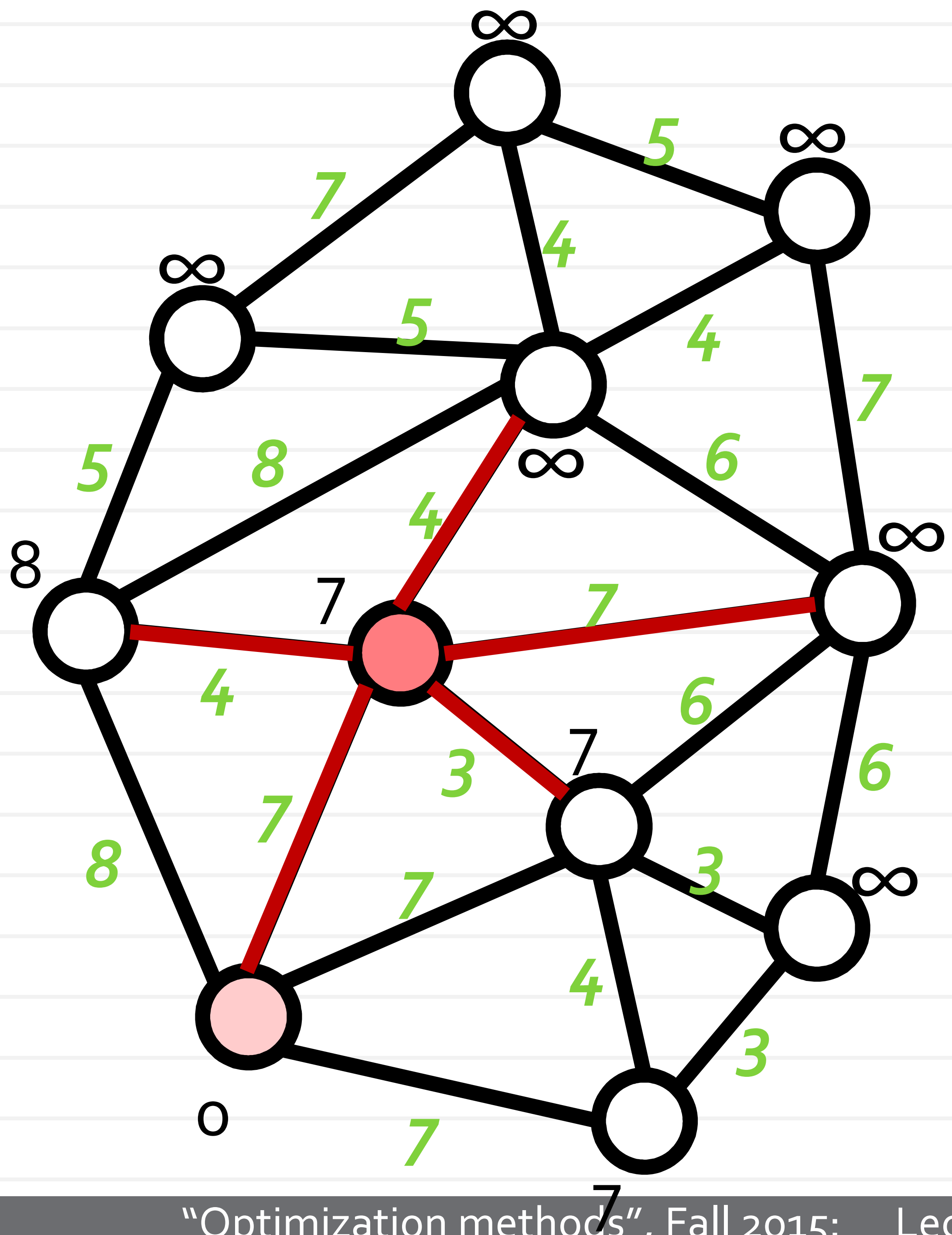
Dijkstra Algorithm



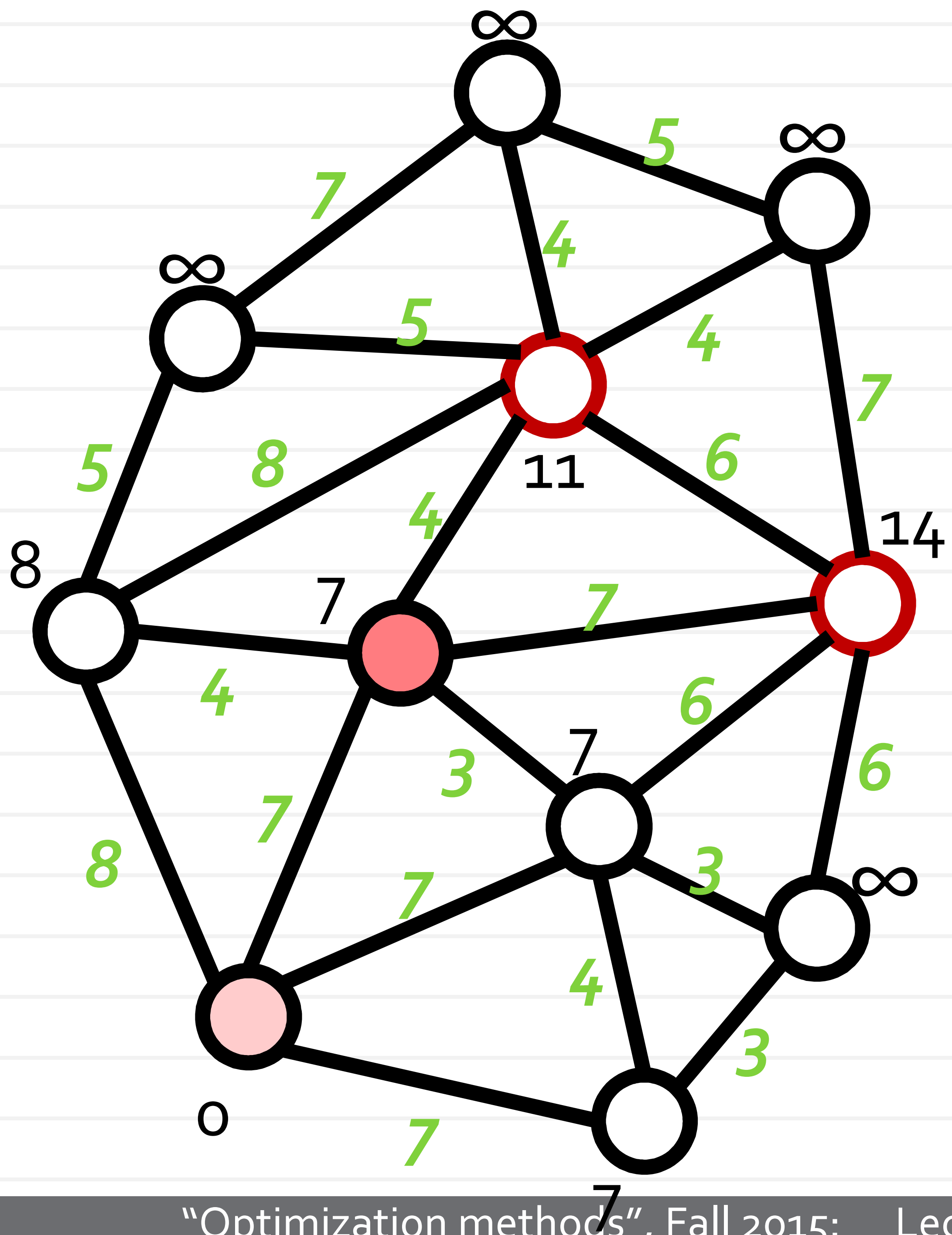
Dijkstra Algorithm



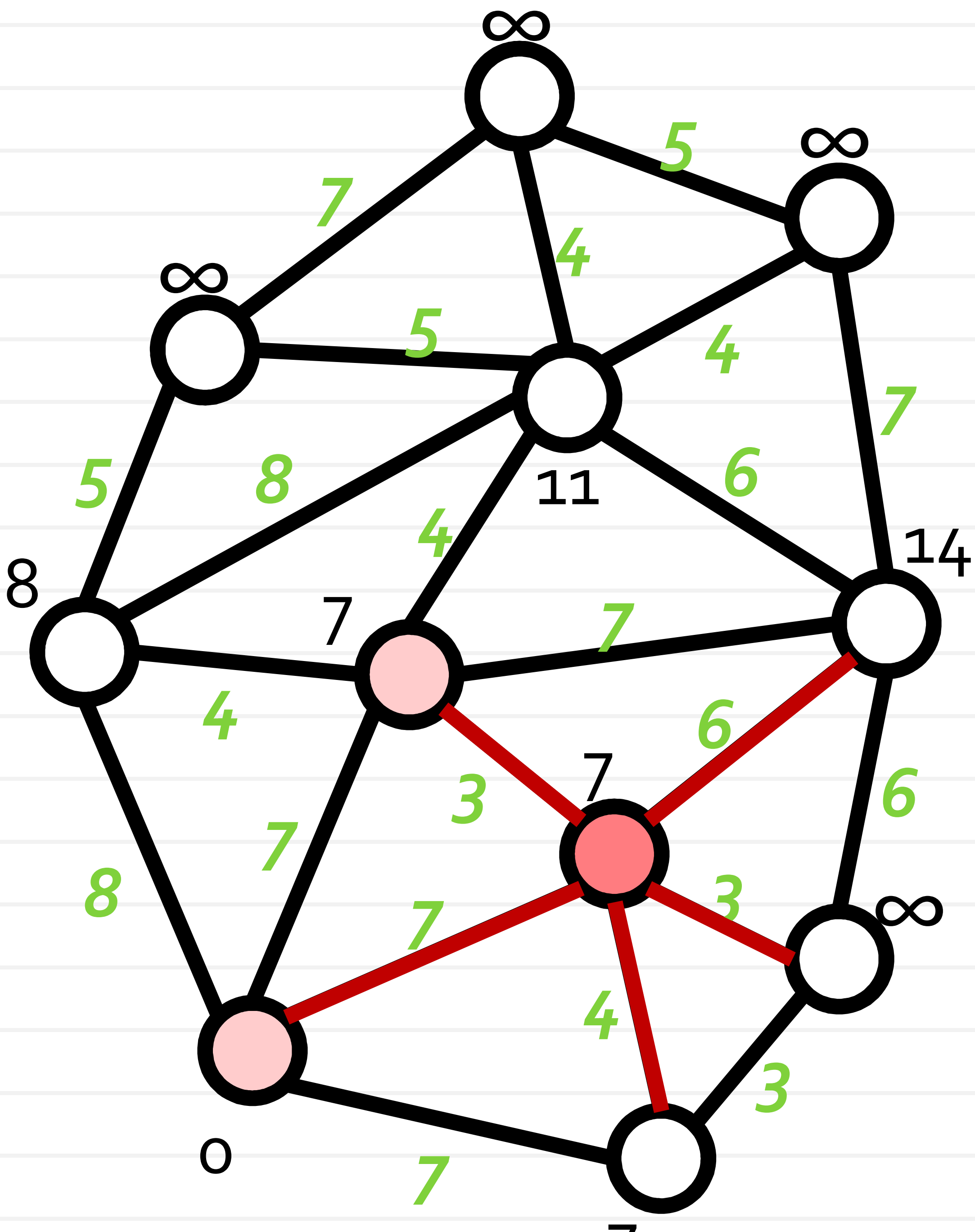
Dijkstra Algorithm



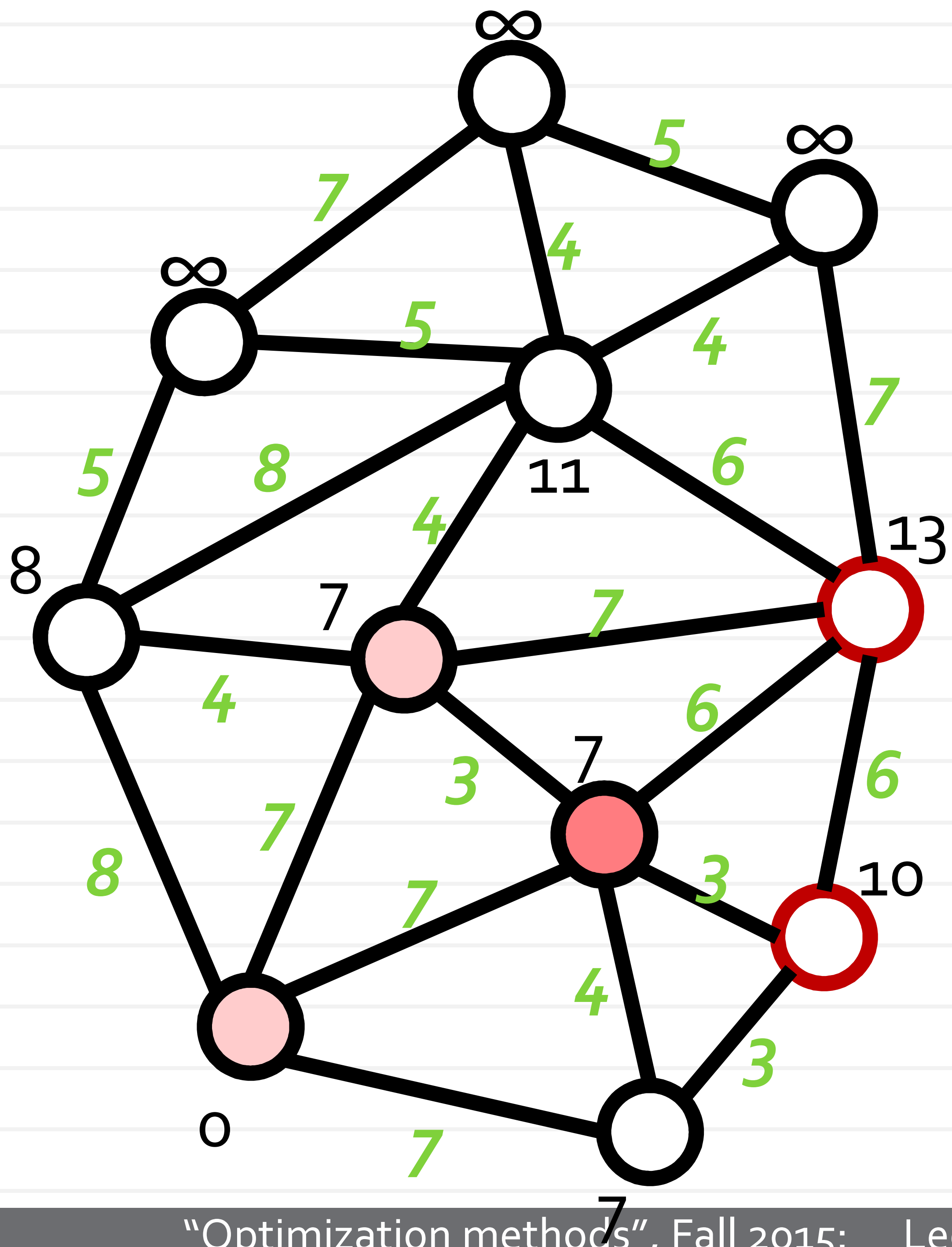
Dijkstra Algorithm



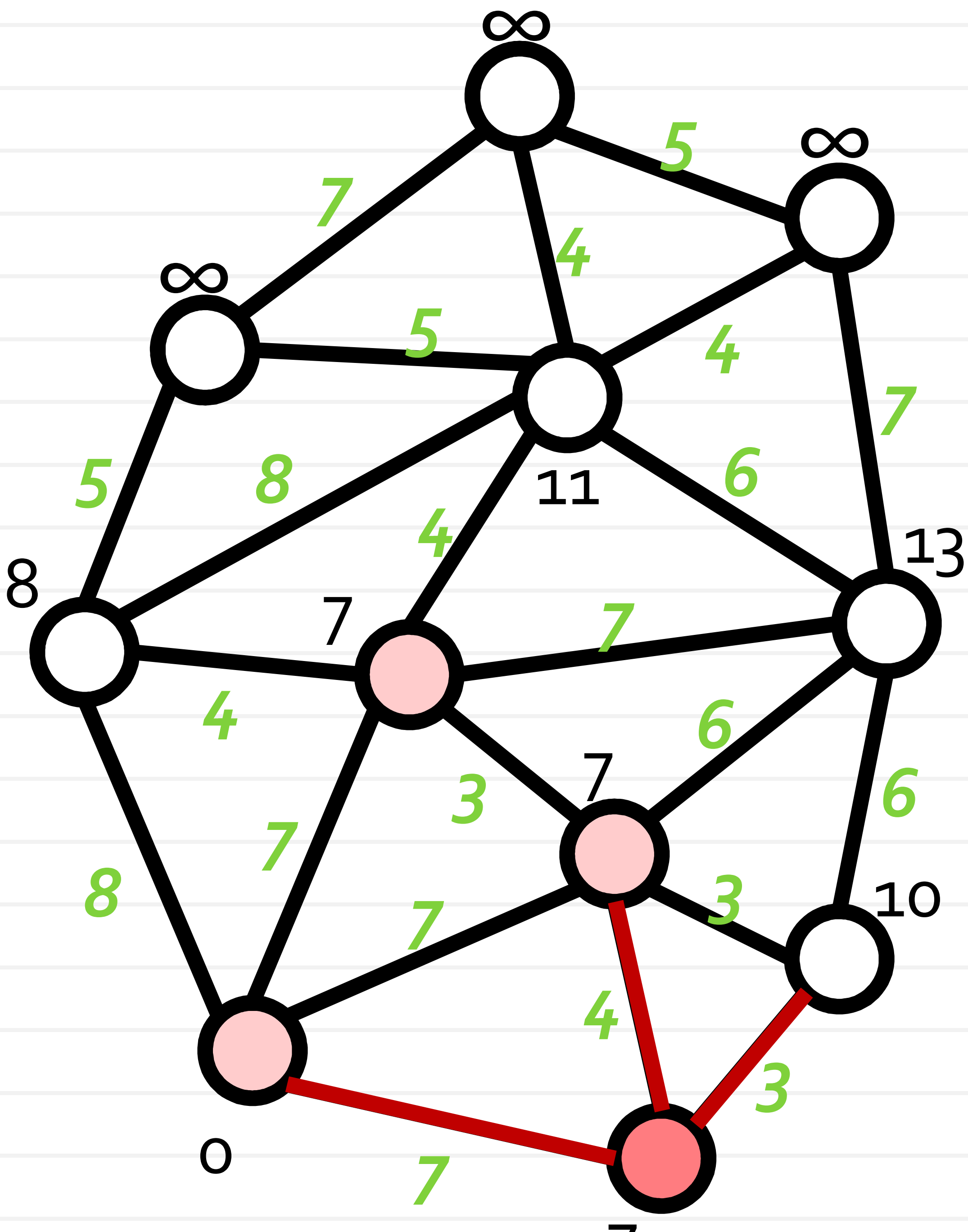
Dijkstra Algorithm



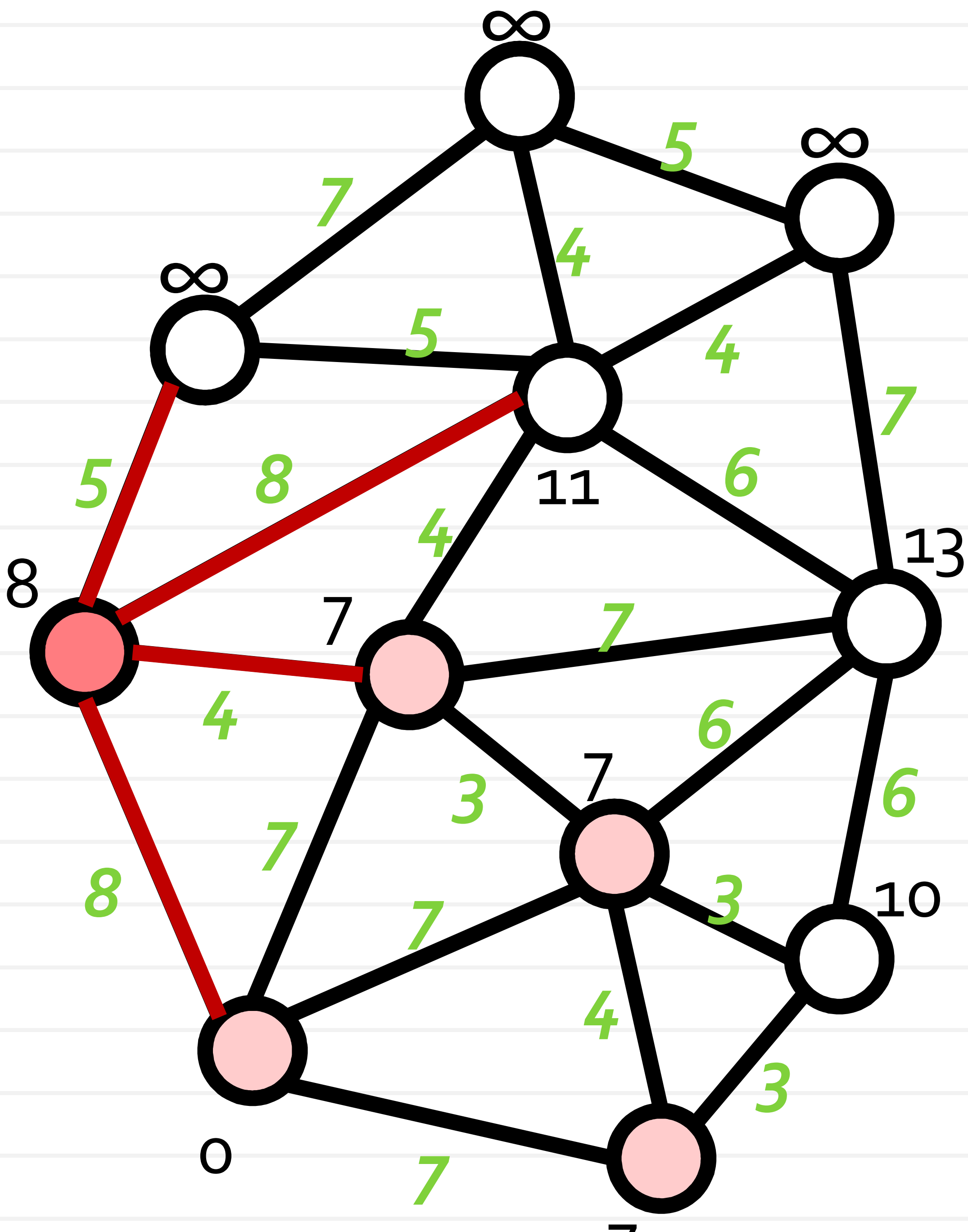
Dijkstra Algorithm



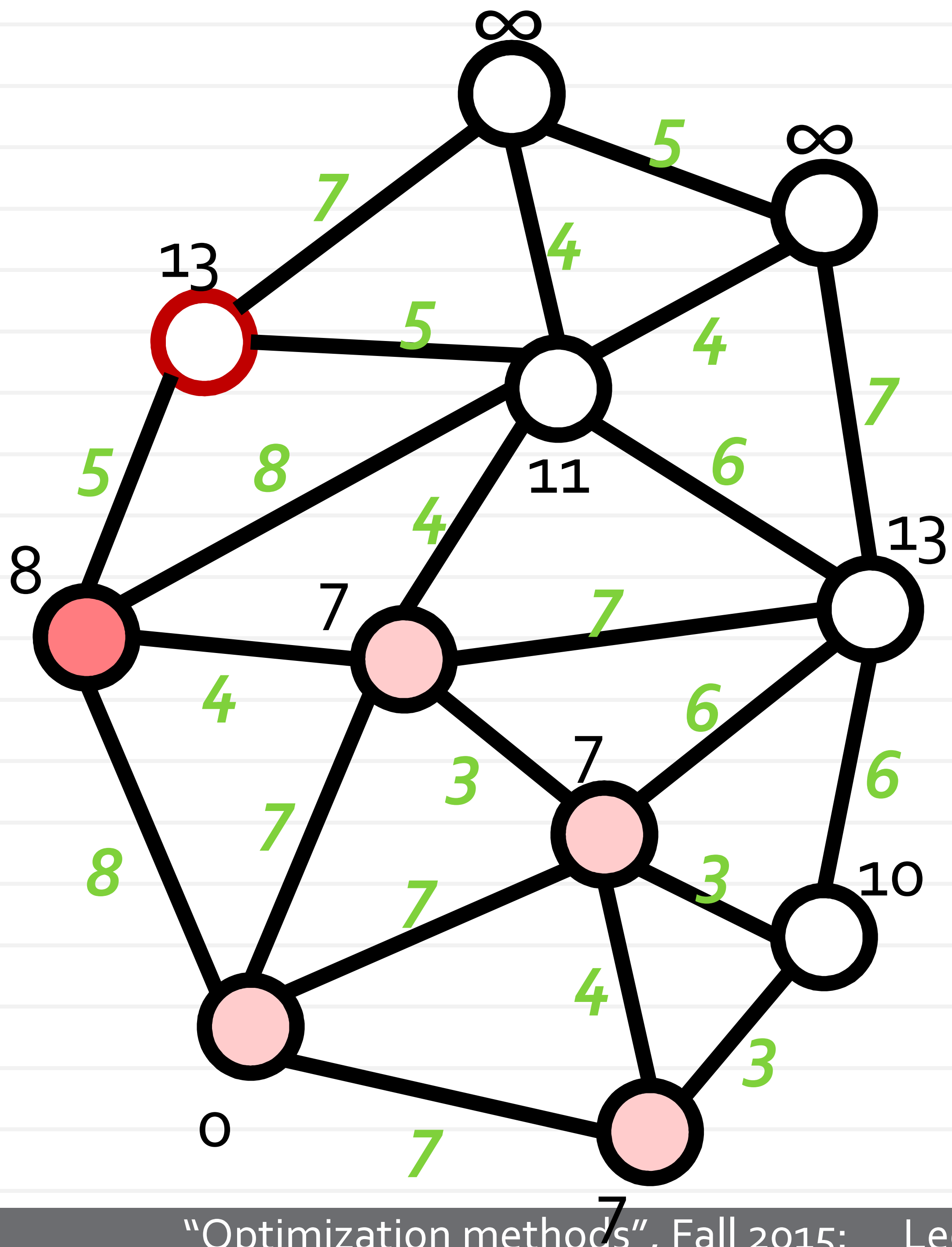
Dijkstra Algorithm



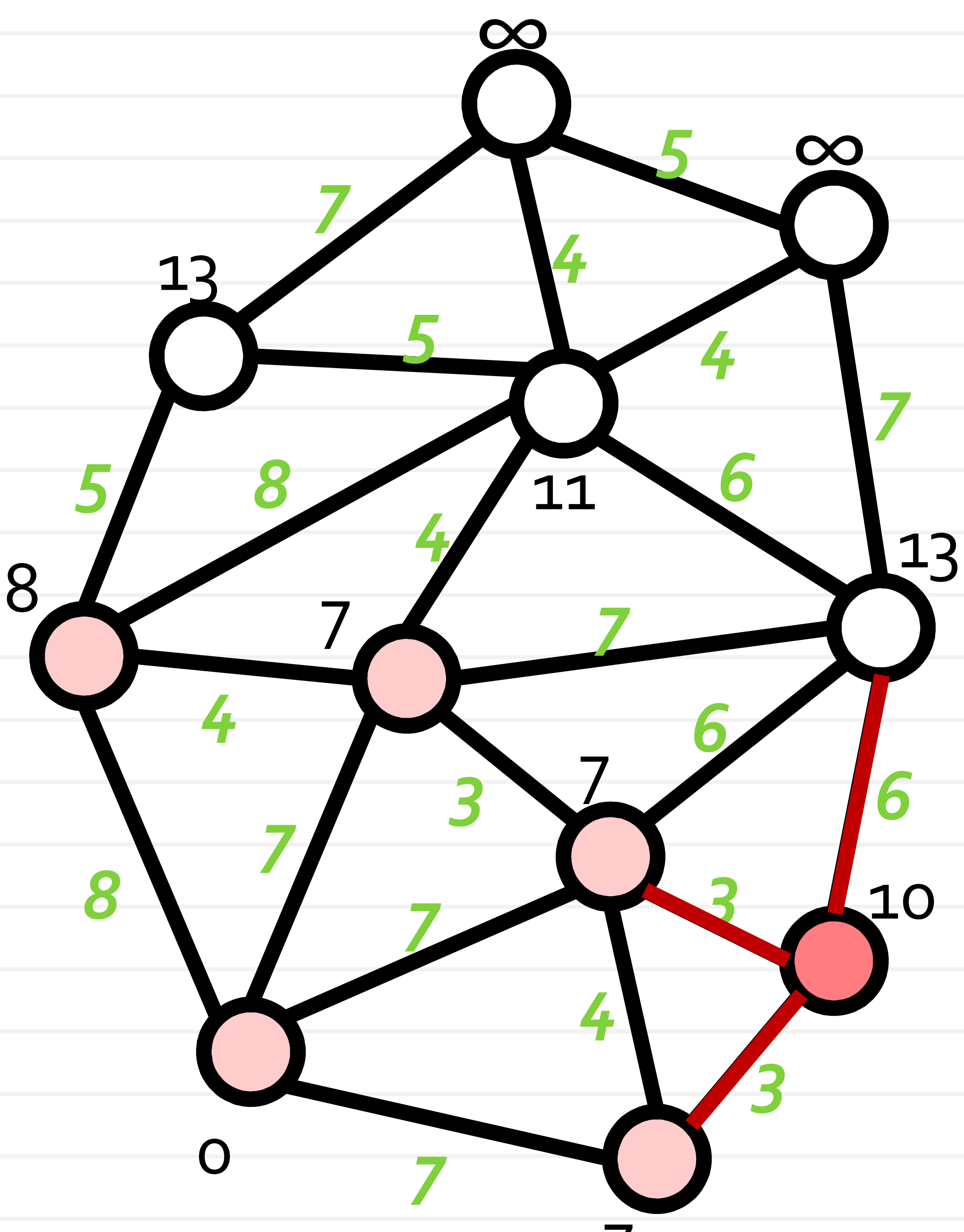
Dijkstra Algorithm



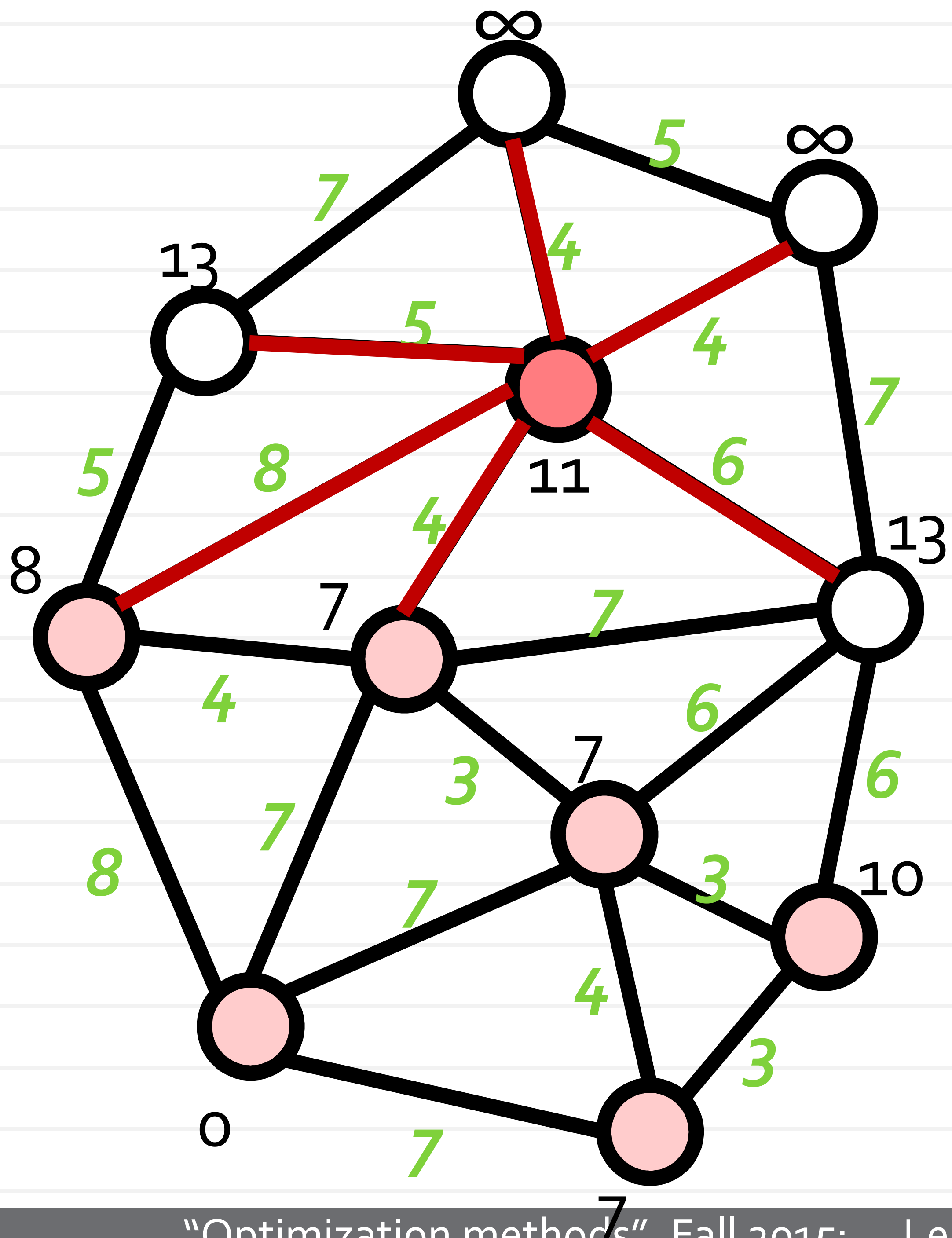
Dijkstra Algorithm



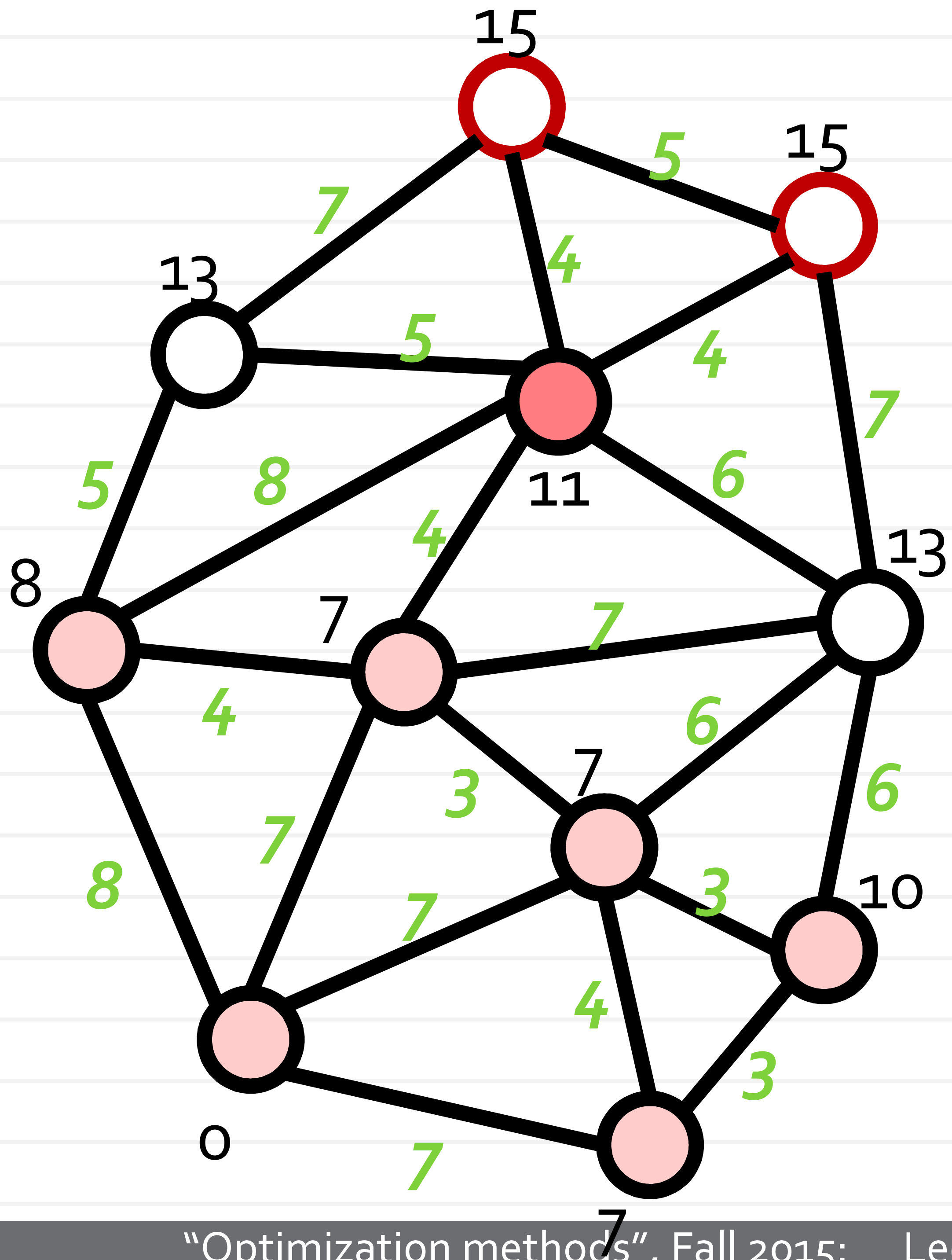
Dijkstra Algorithm



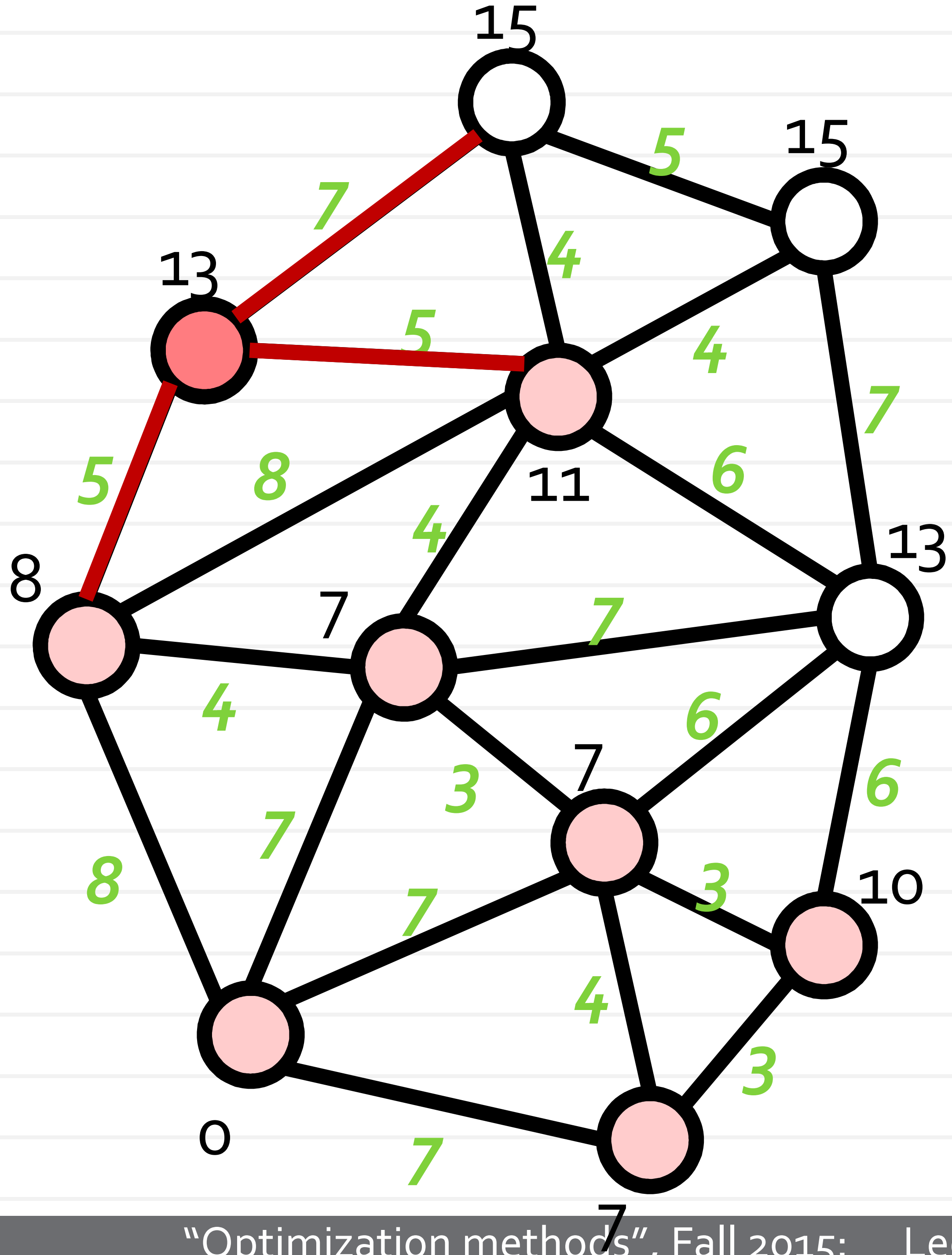
Dijkstra Algorithm



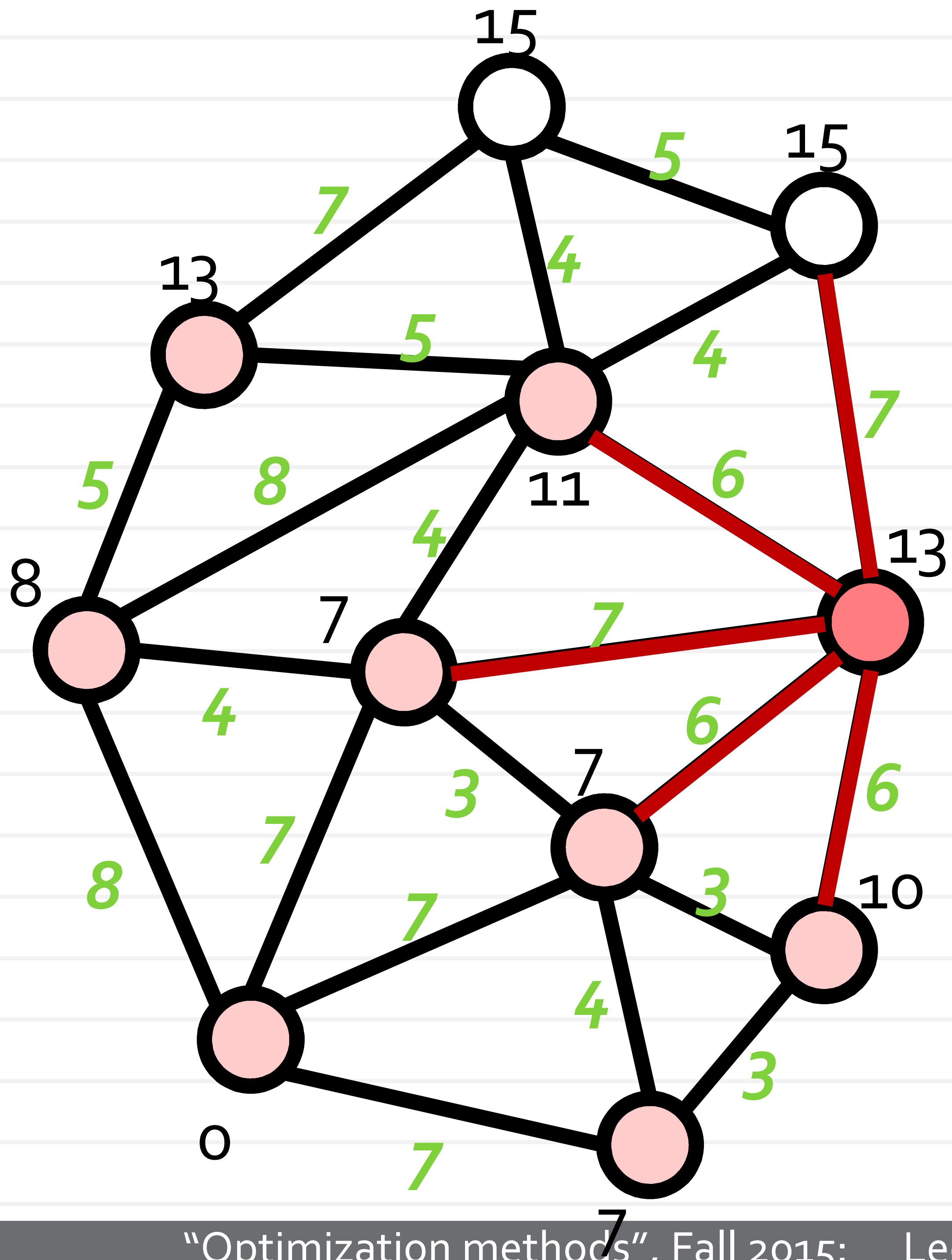
Dijkstra Algorithm



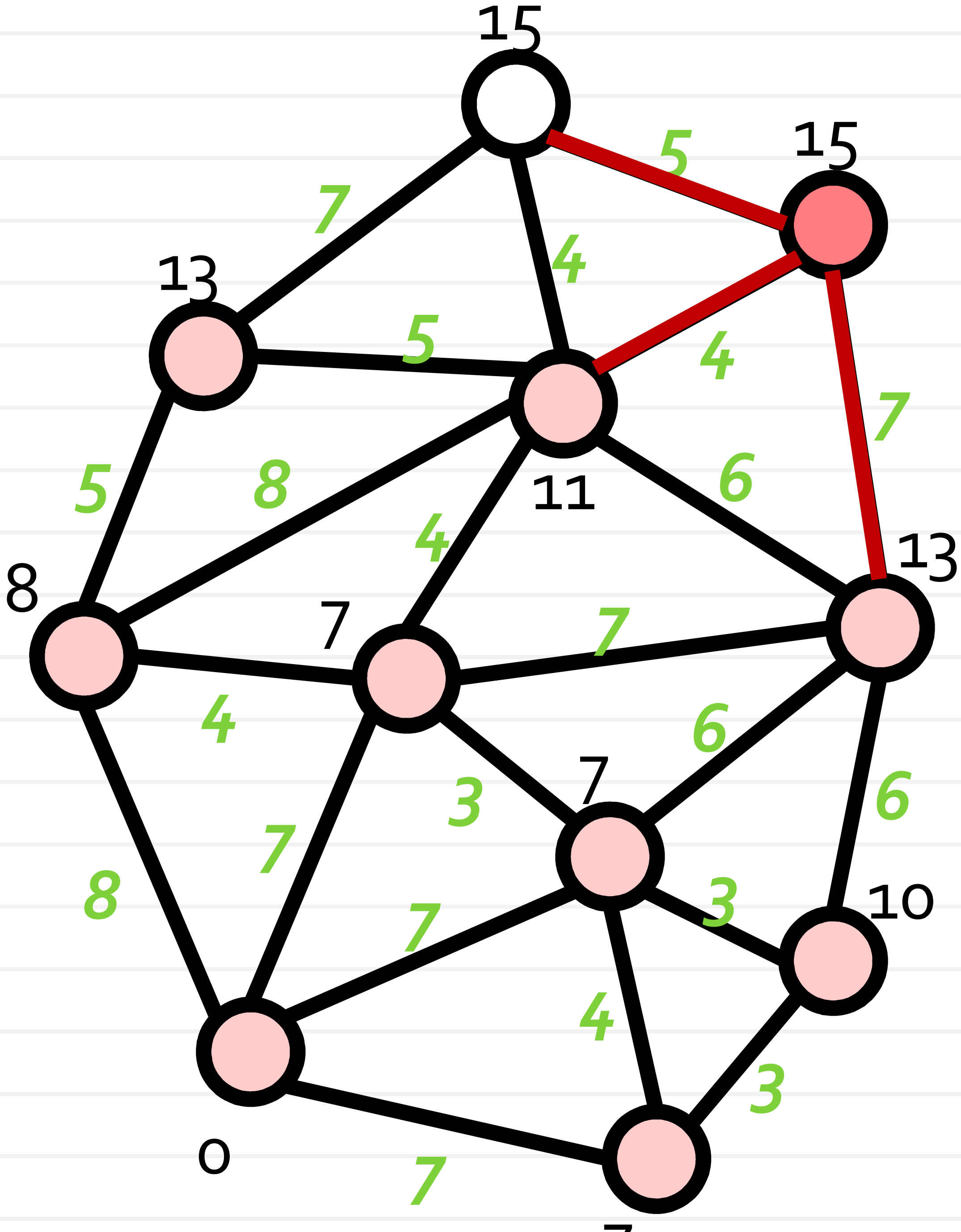
Dijkstra Algorithm



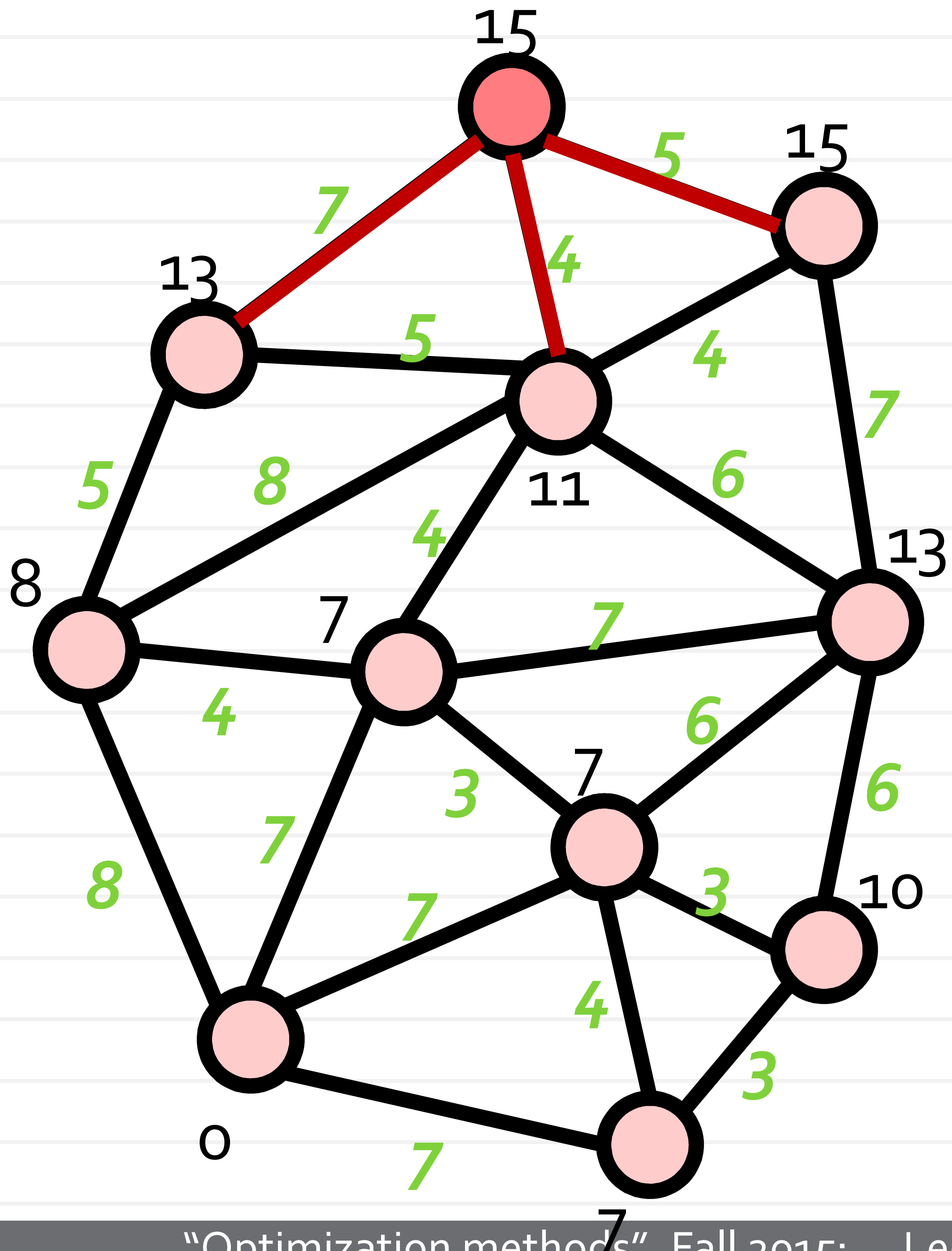
Dijkstra Algorithm



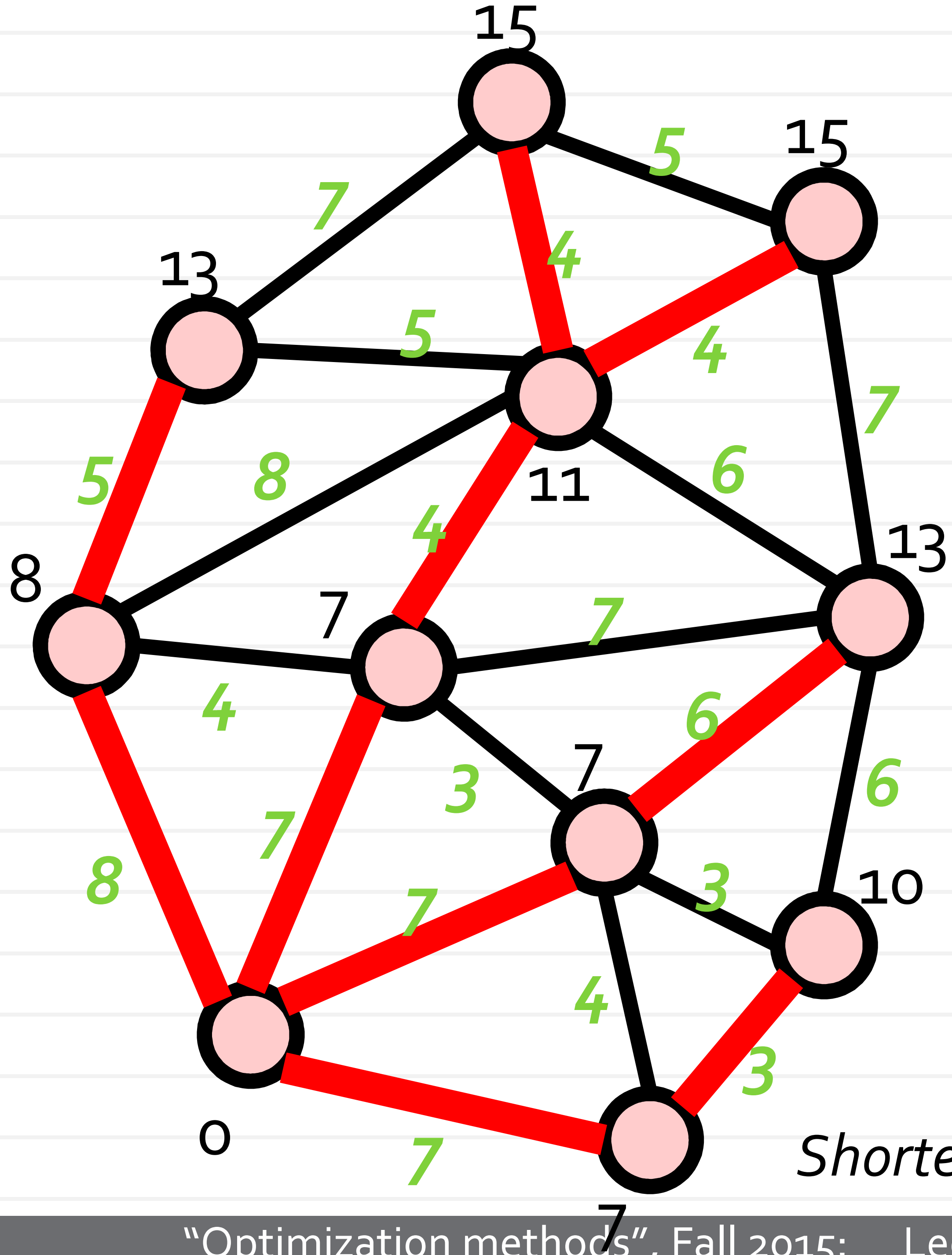
Dijkstra Algorithm



Dijkstra Algorithm

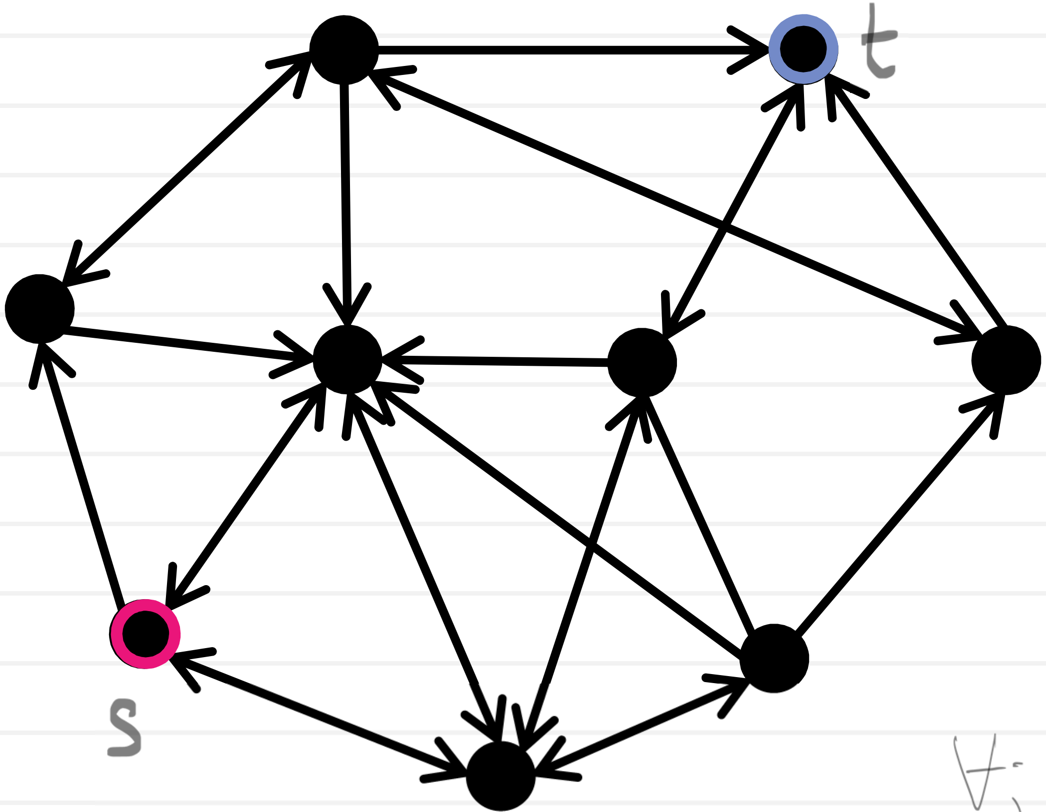


Dijkstra Algorithm



Shortest path spanning tree

Solving max-flow: Ford-Fulkerson



$$\max_{b,f} \; b$$

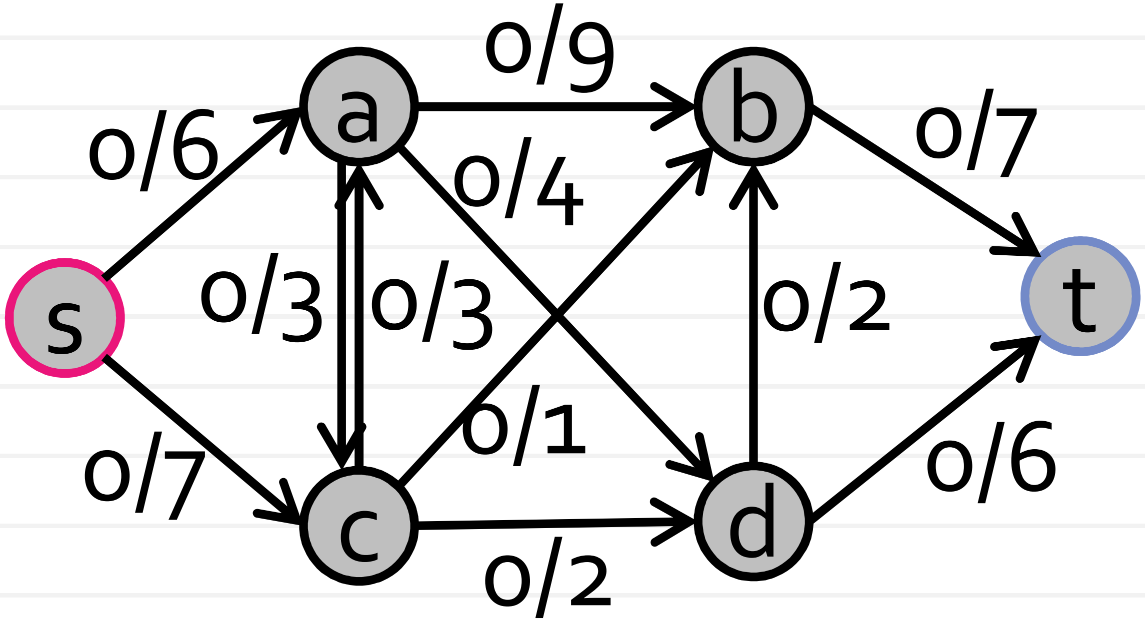
$$\text{s.t.:} \quad 0 \le f_{ij} \le u_{ij}$$

$$b + \sum_{i \in I(s)} f_{is} = \sum_{i \in O(s)} f_{si}$$

$$\sum_{i \in I(t)} f_{it} = \sum_{i \in O(t)} f_{ti} + b$$

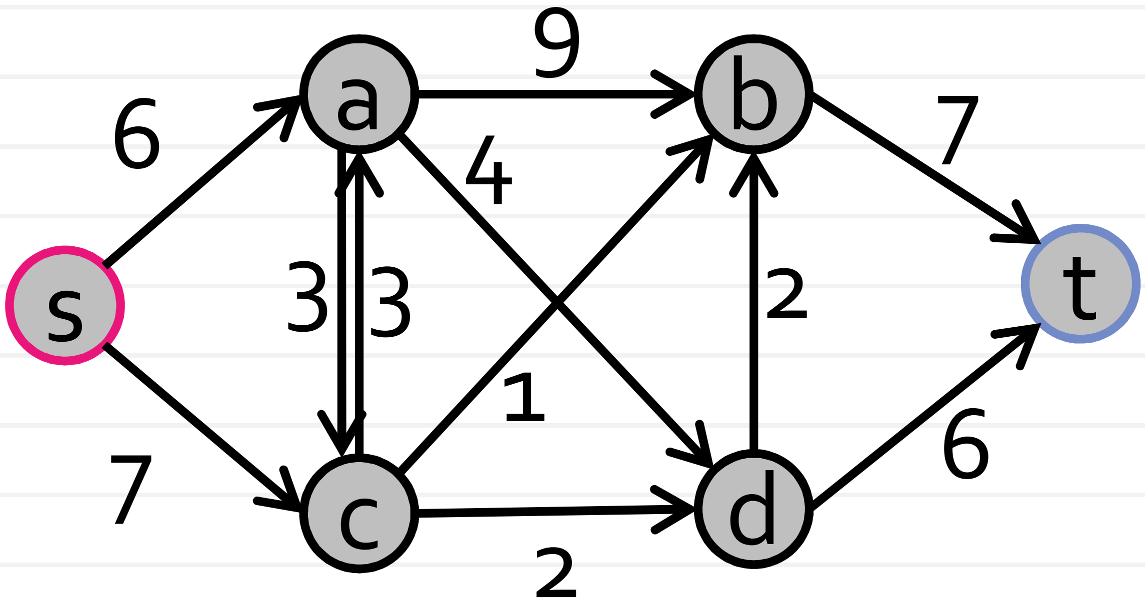
$$\forall j \ne s,t \quad \sum_{i \in I(j)} f_{ij} = \sum_{i \in O(j)} f_{ji}$$

Solving max-flow: Ford-Fulkerson



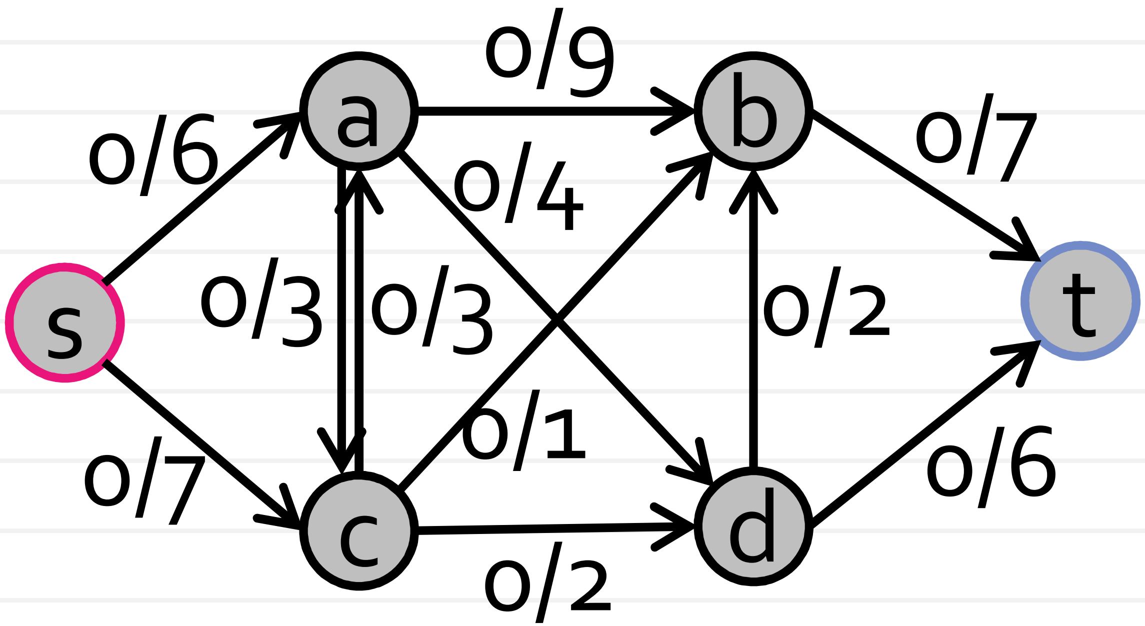
Original graph

Flow = 0



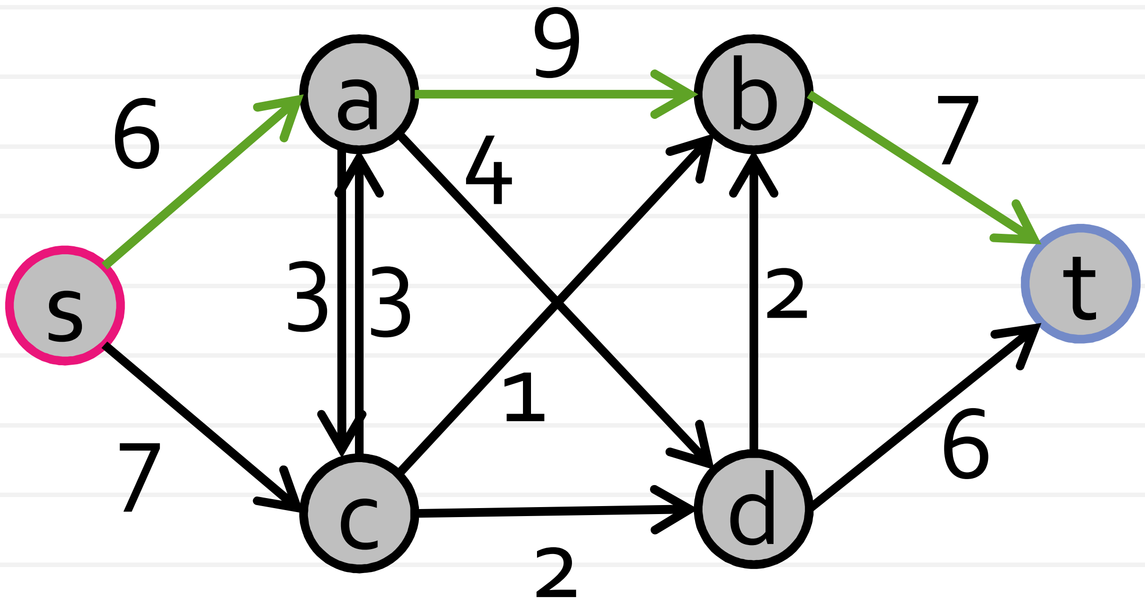
Residual graph

Solving max-flow: Ford-Fulkerson



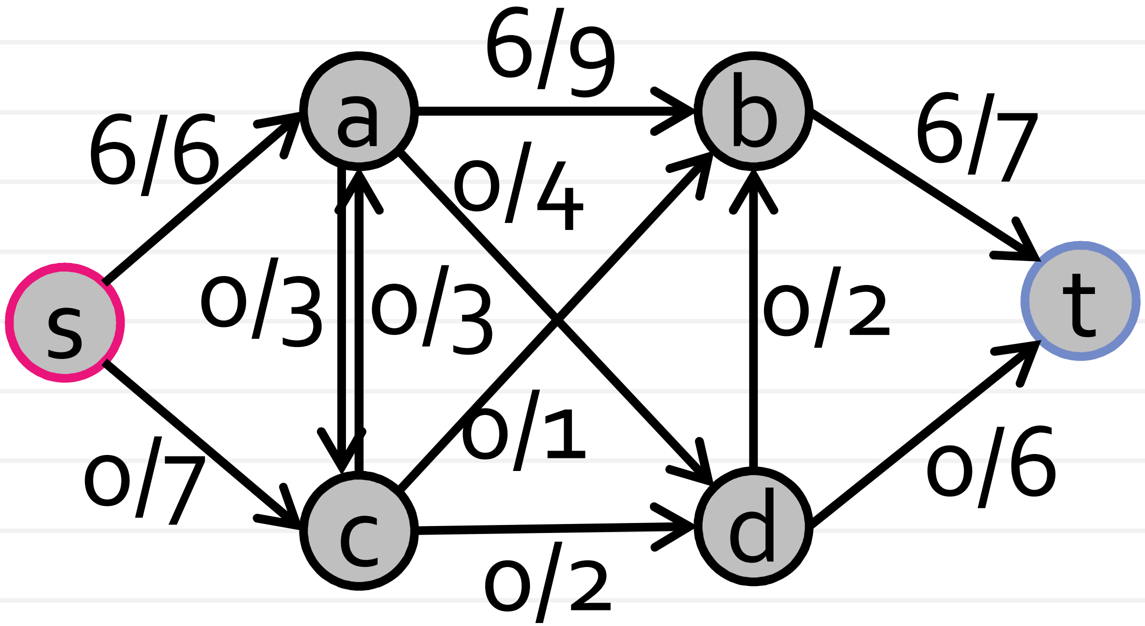
Original graph

Flow = 0



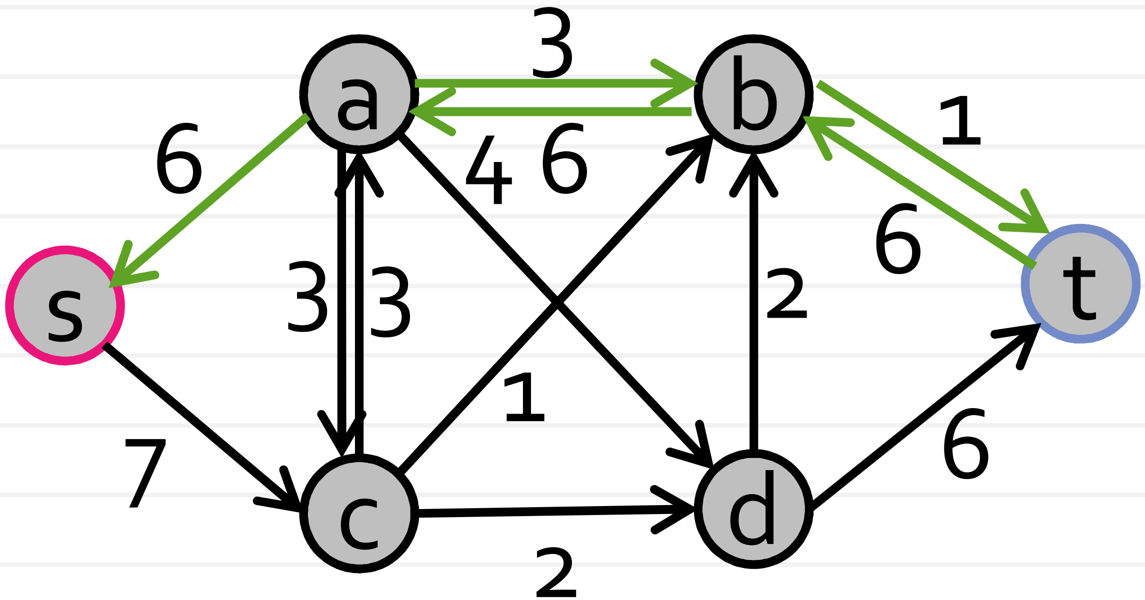
Residual graph

Solving max-flow: Ford-Fulkerson



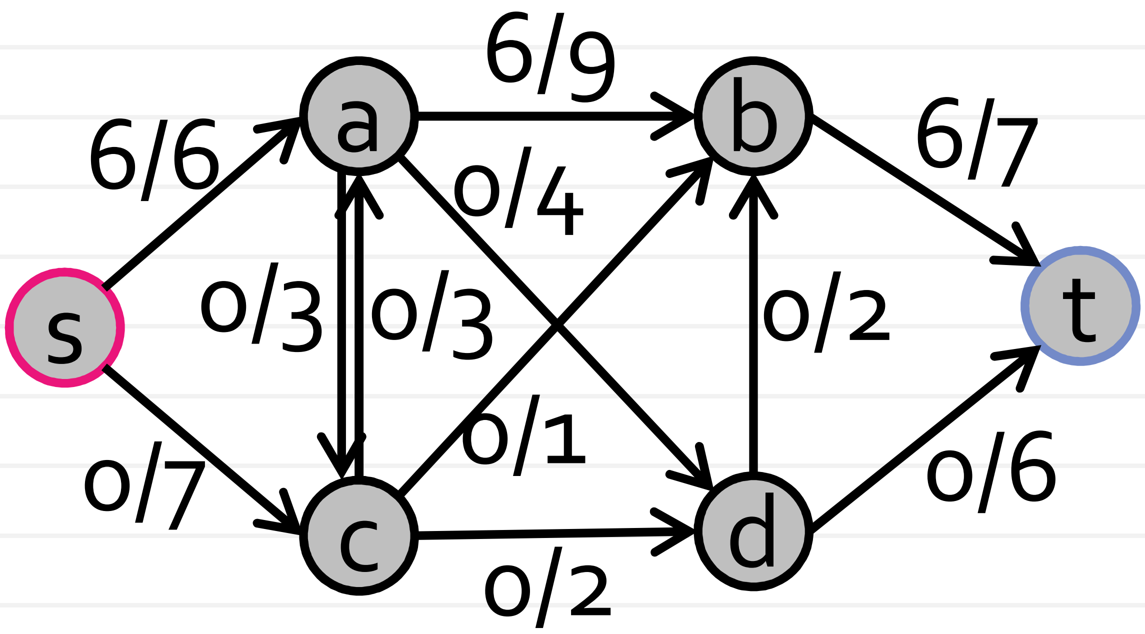
Original graph

Flow = 6



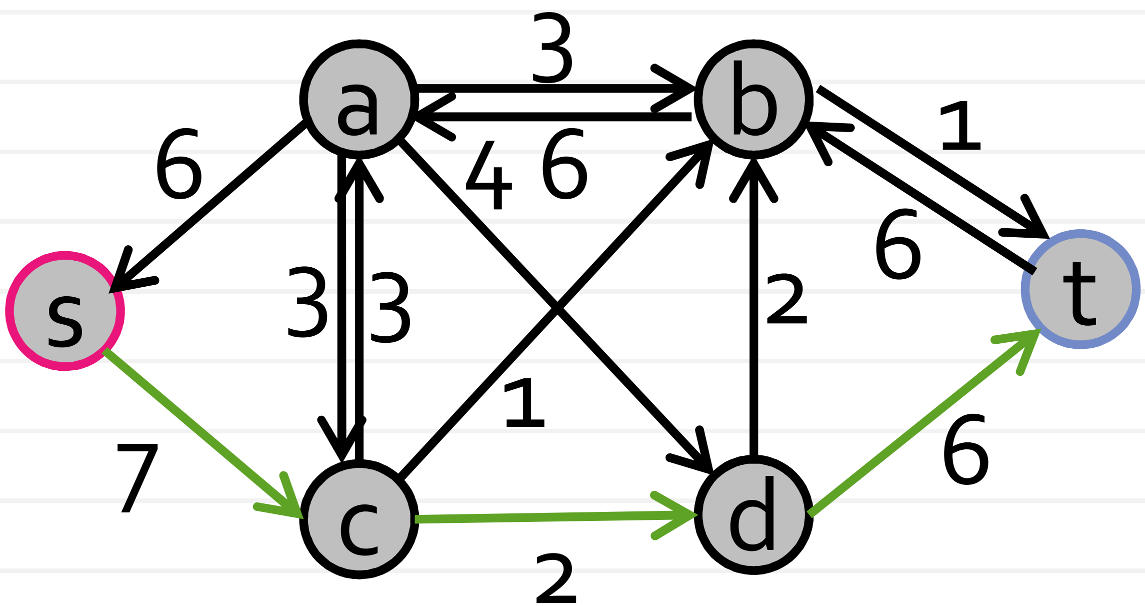
Residual graph

Solving max-flow: Ford-Fulkerson



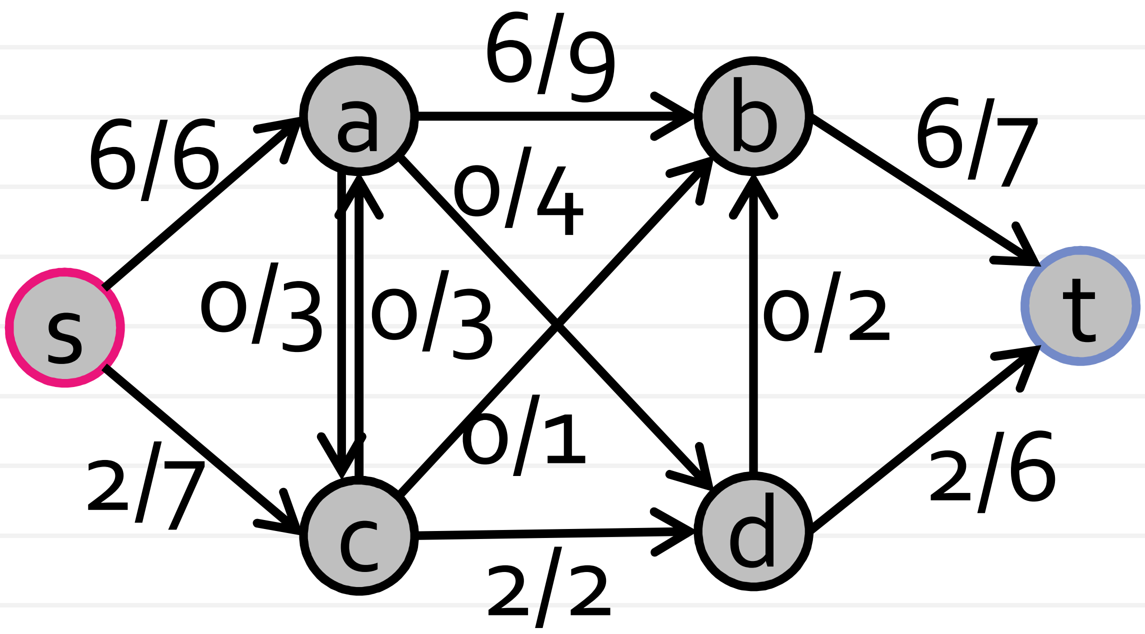
Original graph

Flow = 6



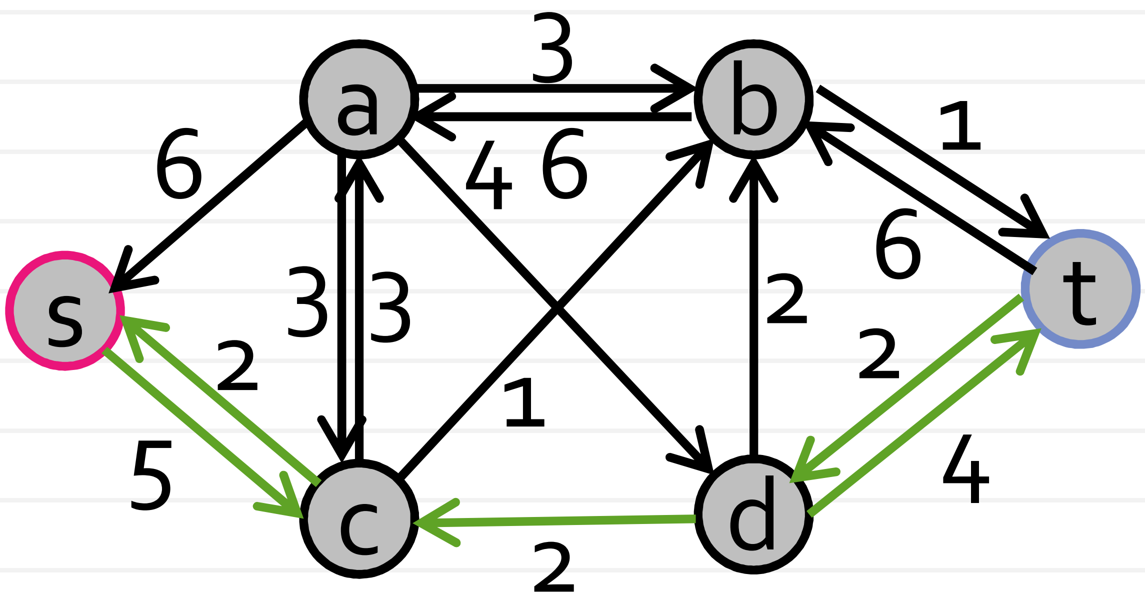
Residual graph

Solving max-flow: Ford-Fulkerson



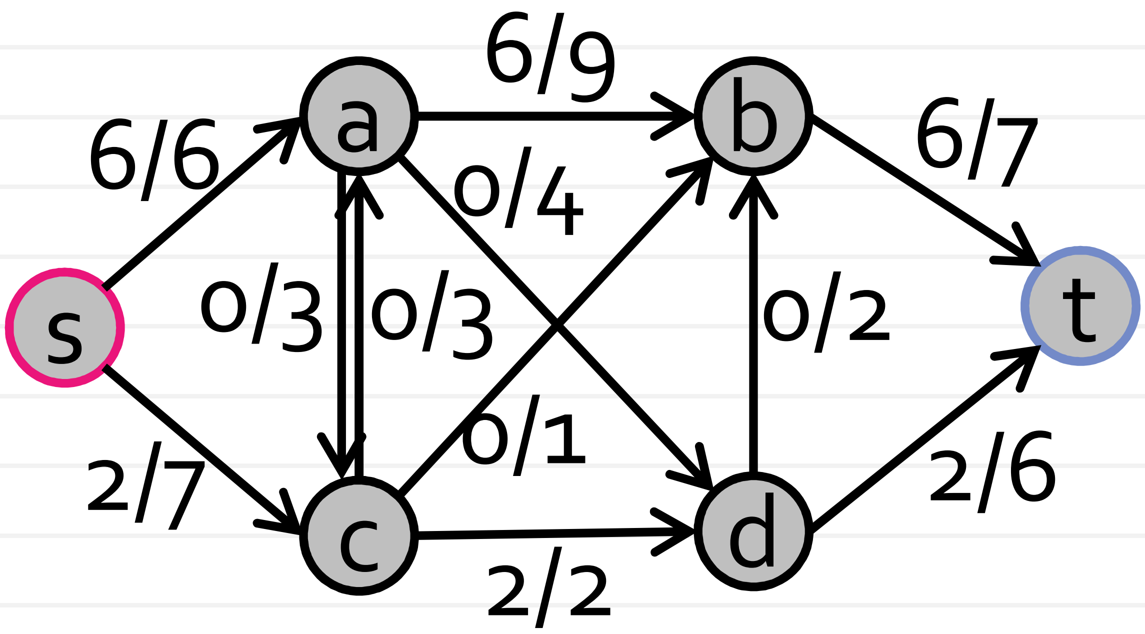
Original graph

Flow = 8



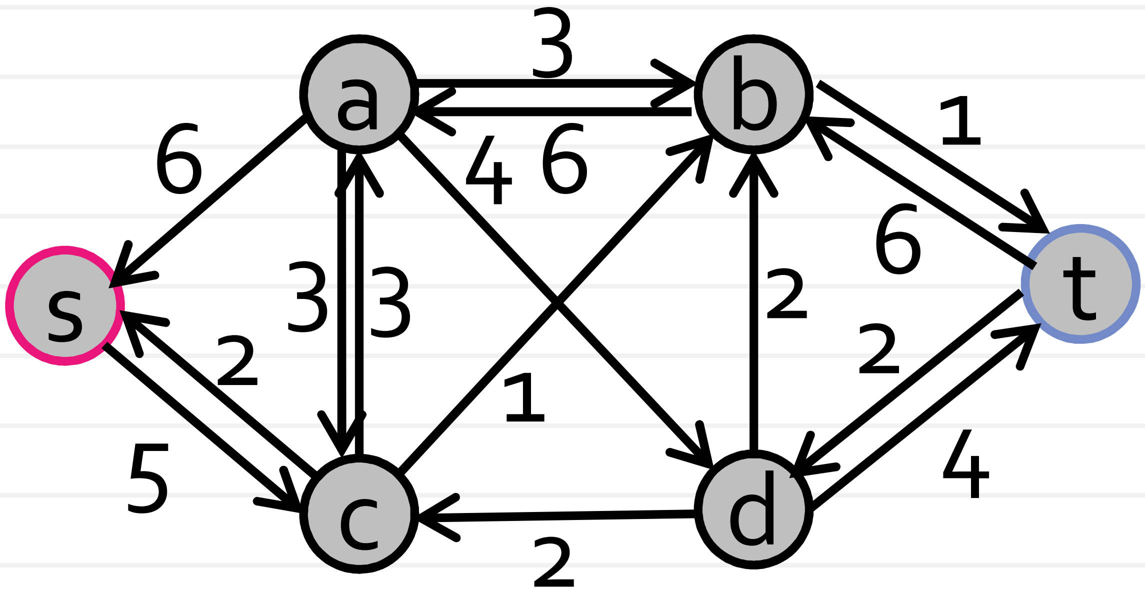
Residual graph

Solving max-flow: Ford-Fulkerson



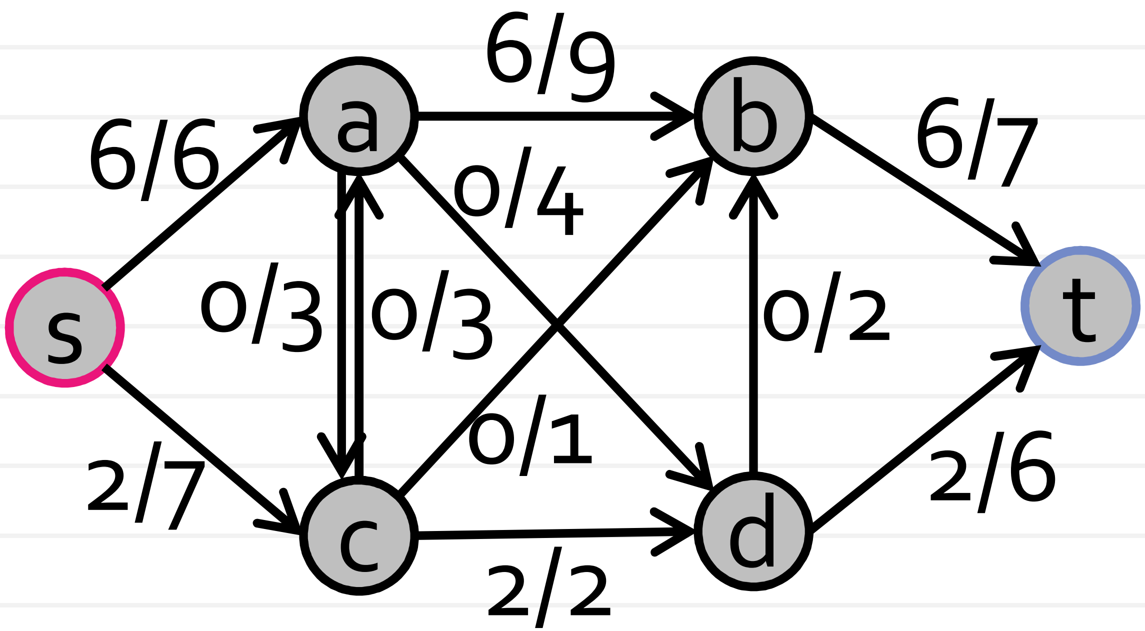
Original graph

Flow = 8



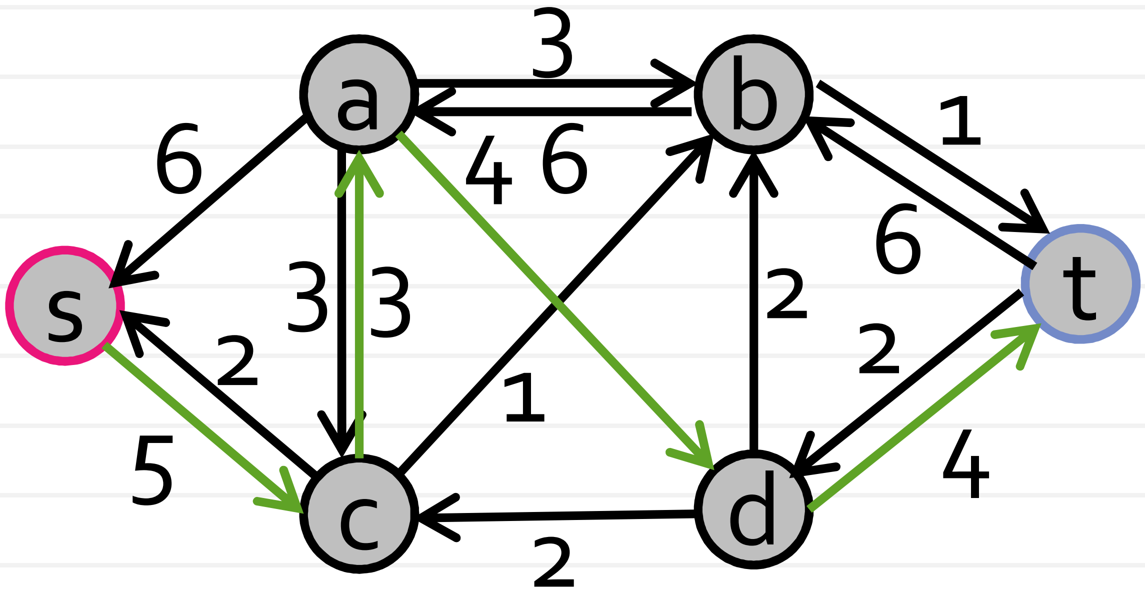
Residual graph

Solving max-flow: Ford-Fulkerson



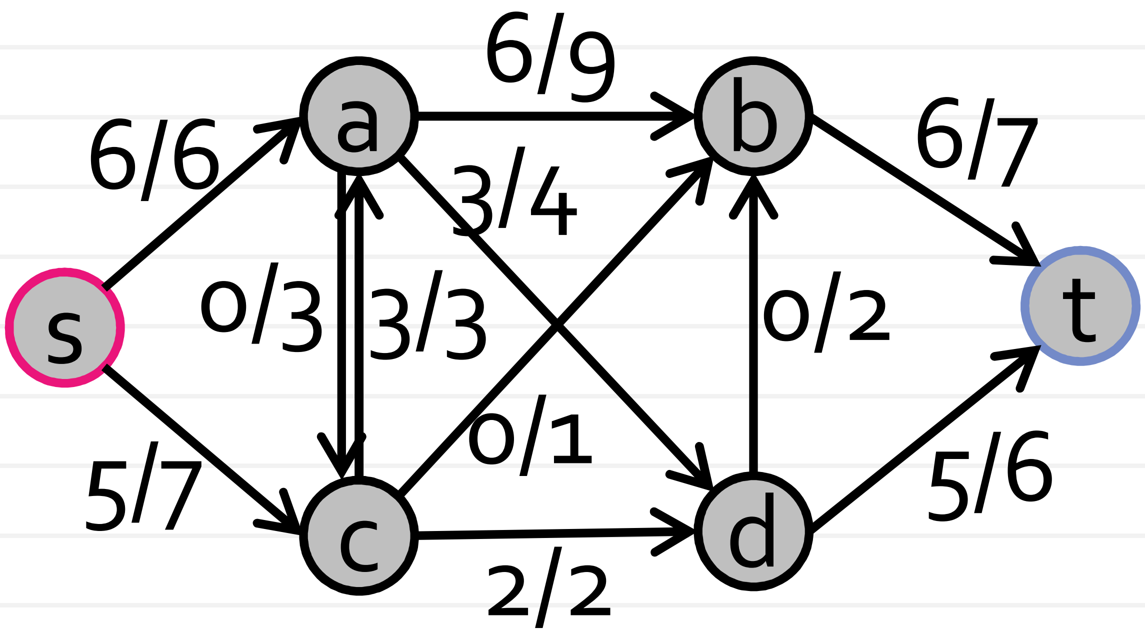
Original graph

Flow = 8



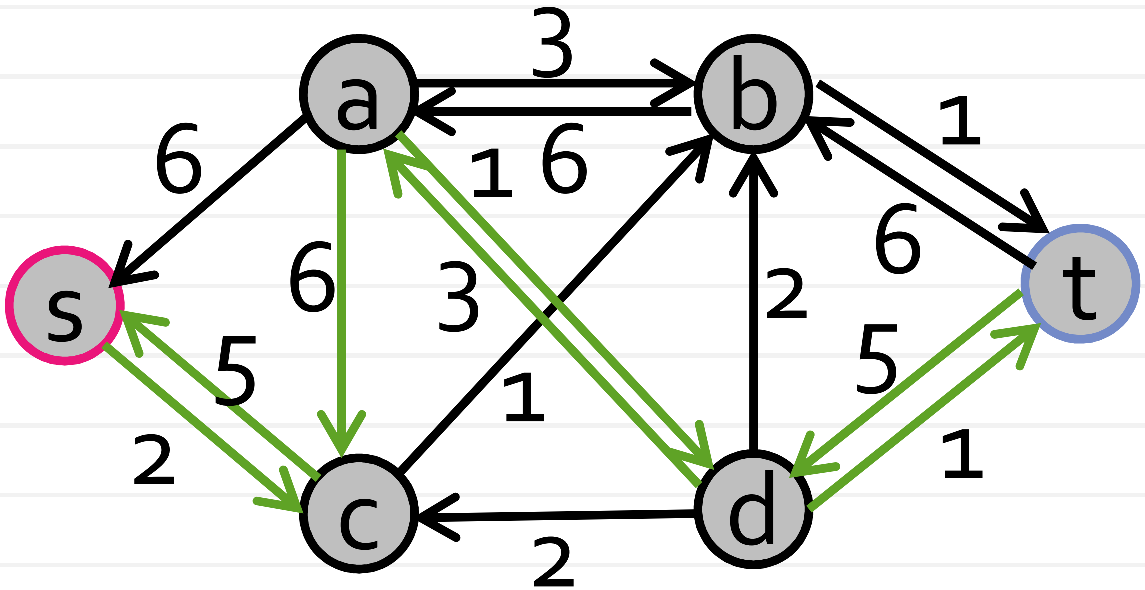
Residual graph

Solving max-flow: Ford-Fulkerson



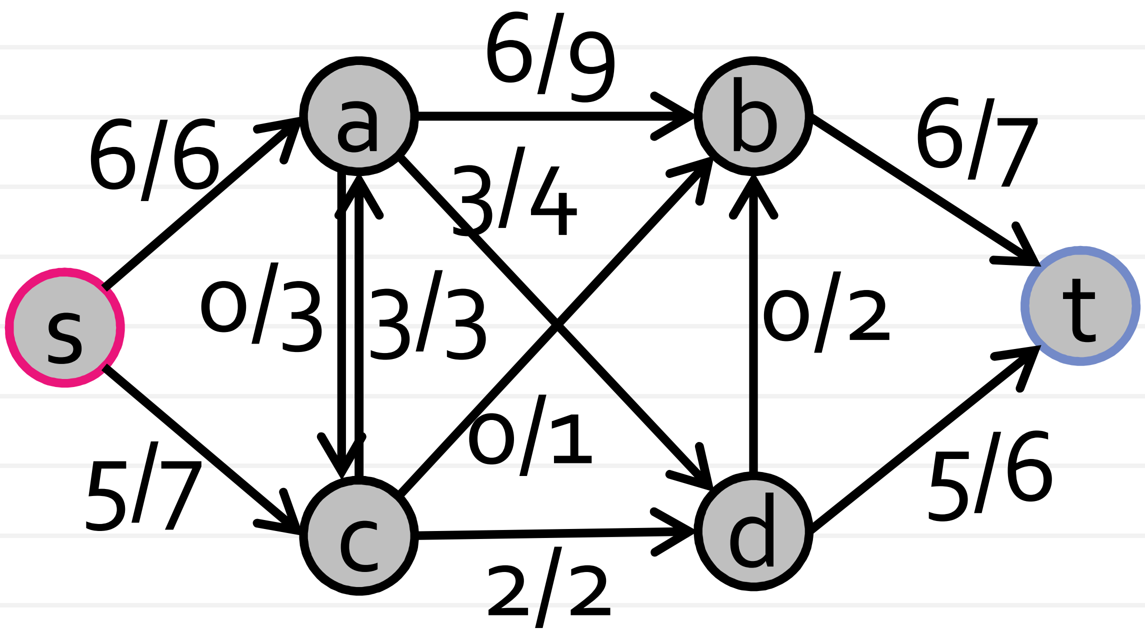
Original graph

Flow = 11



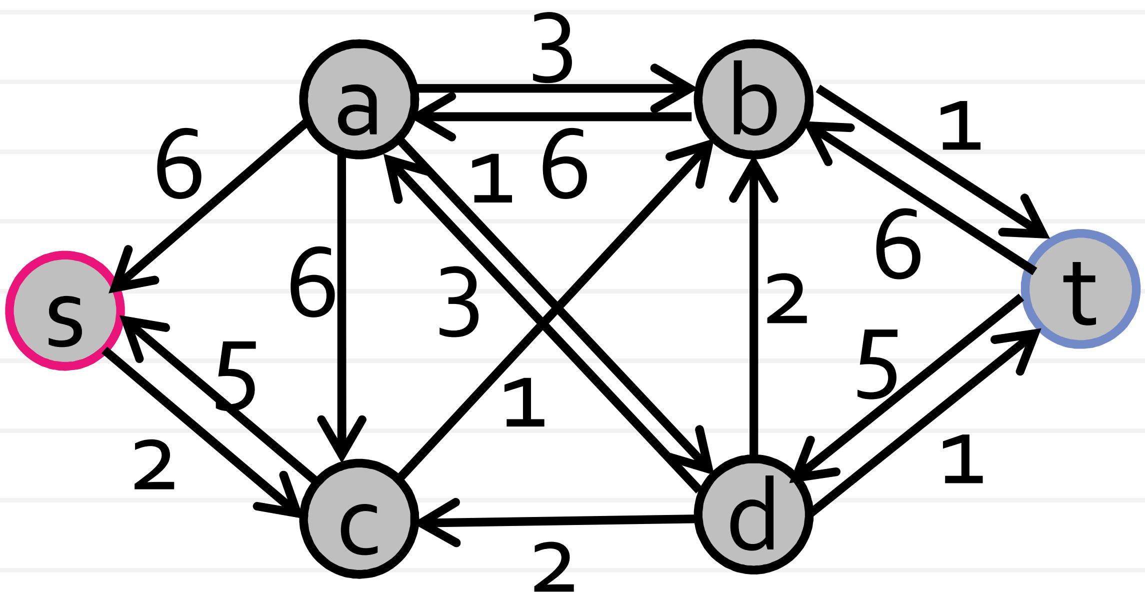
Residual graph

Solving max-flow: Ford-Fulkerson



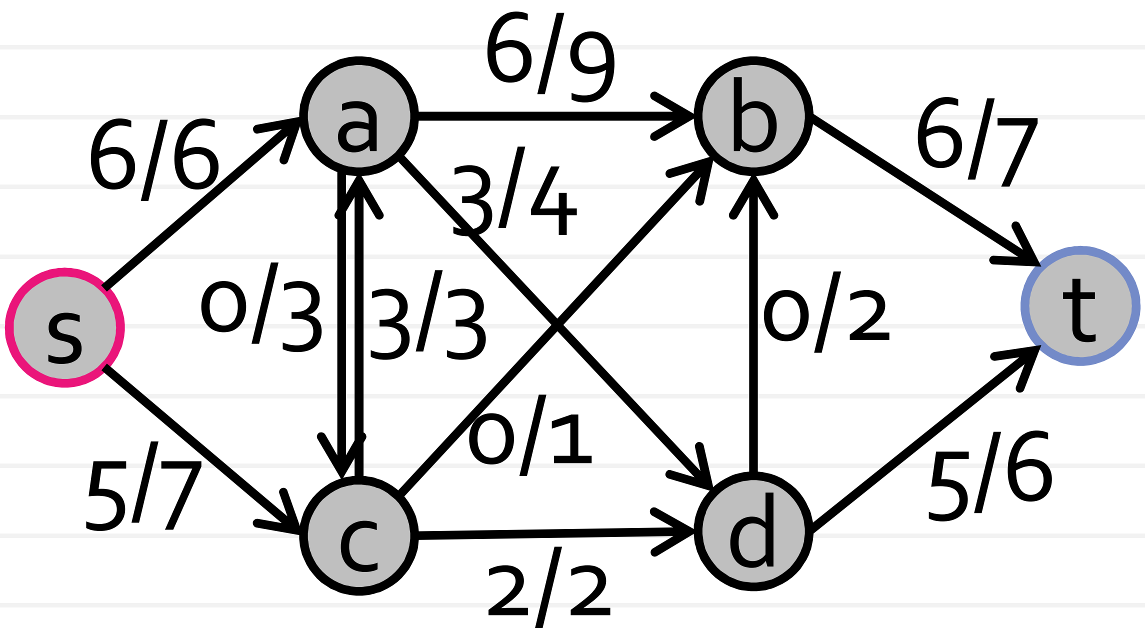
Original graph

Flow = 11



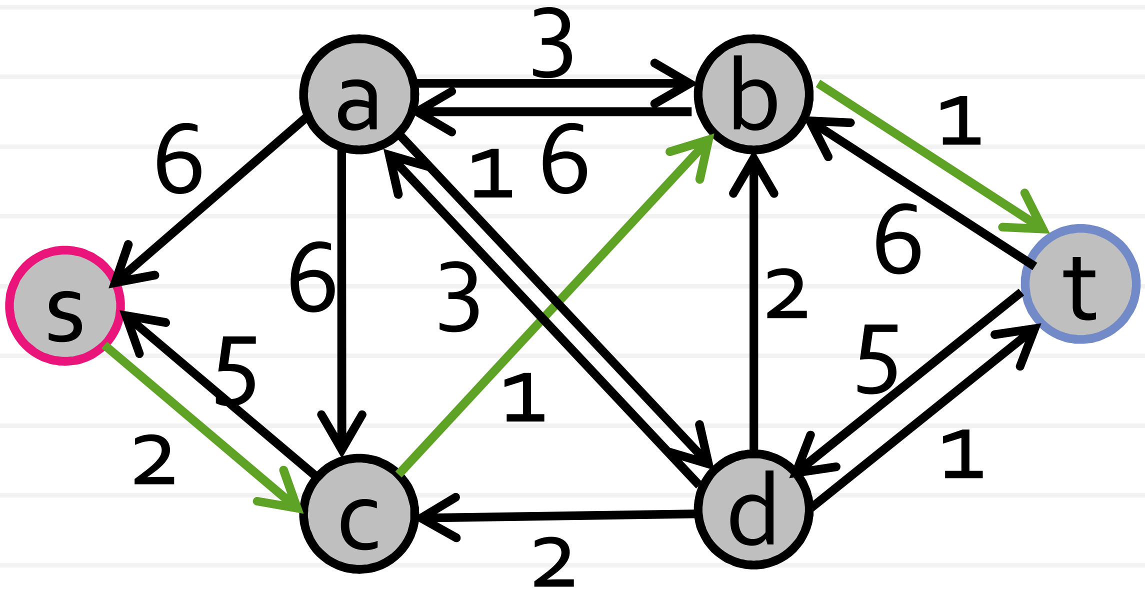
Residual graph

Solving max-flow: Ford-Fulkerson



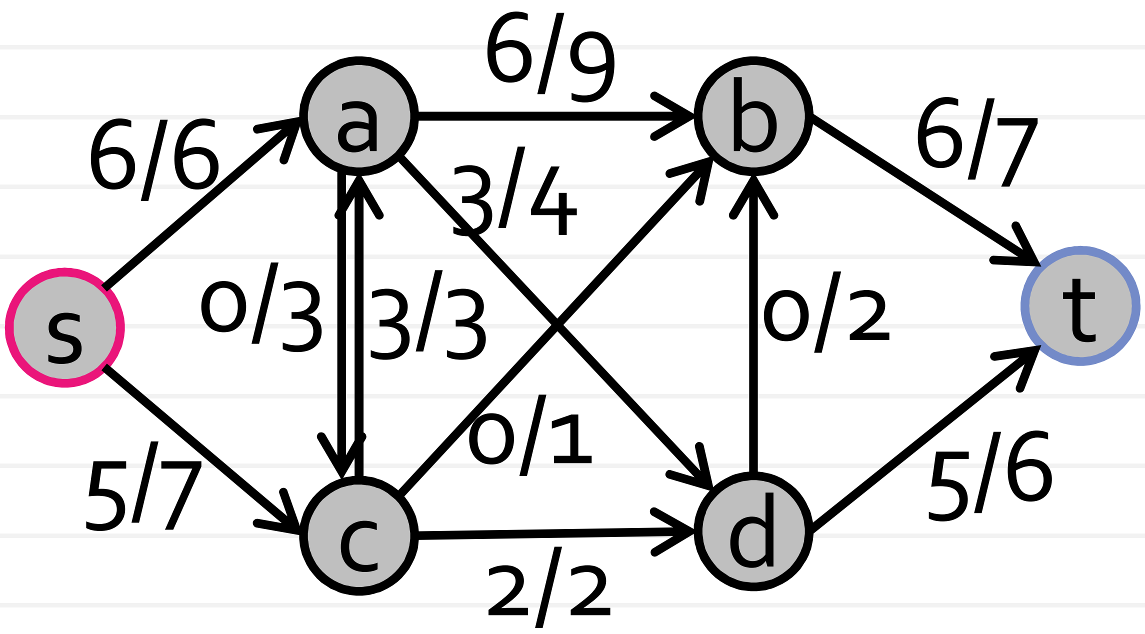
Original graph

Flow = 11



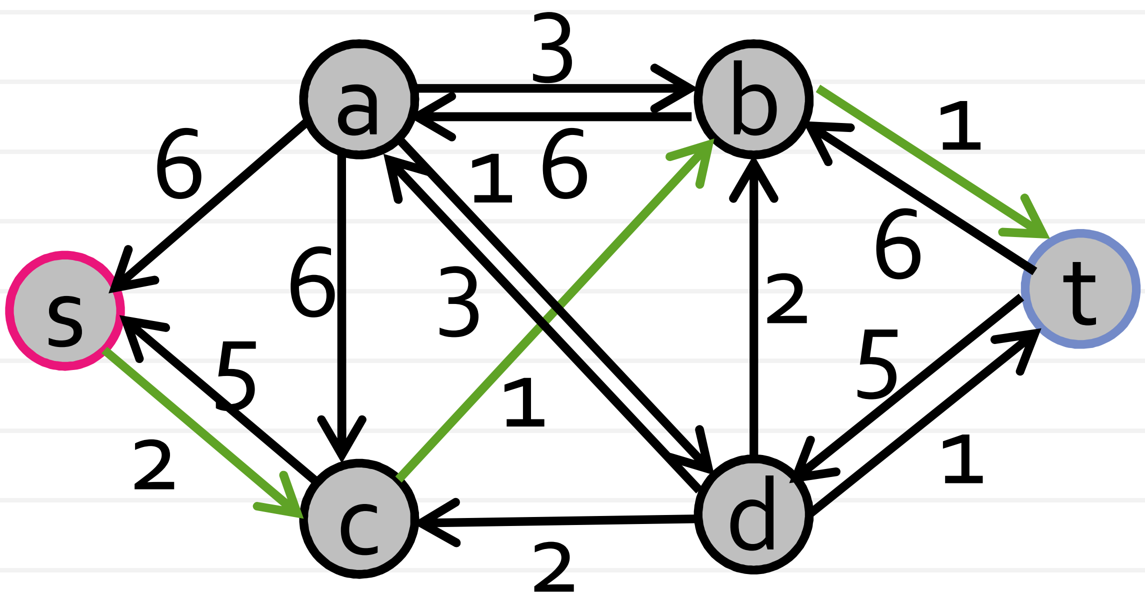
Residual graph

Solving max-flow: Ford-Fulkerson



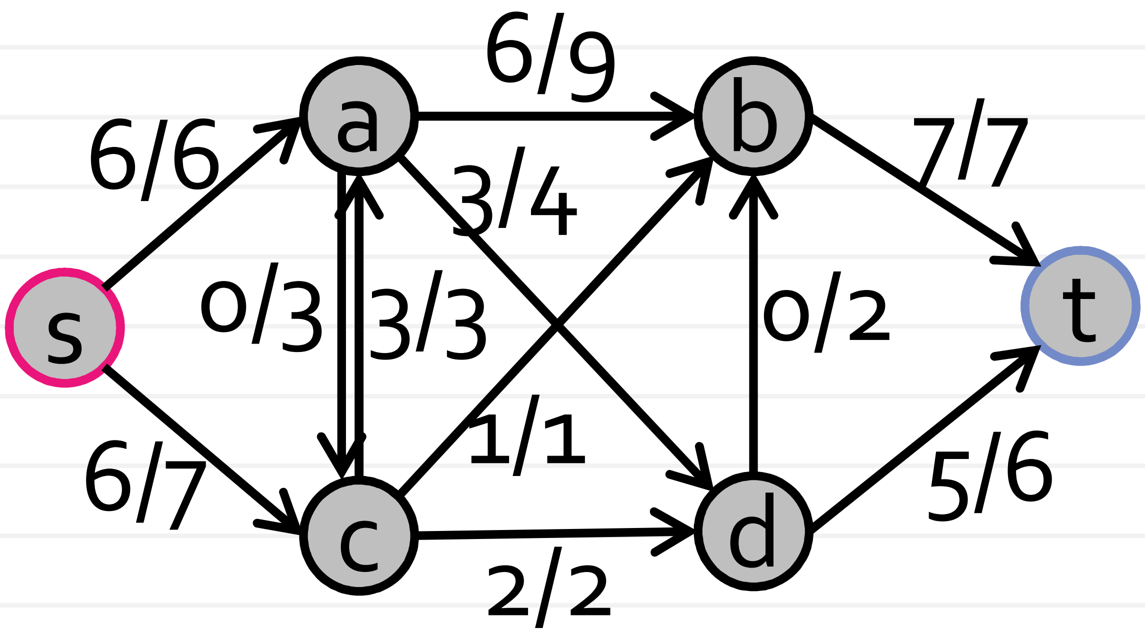
Original graph

Flow = 11



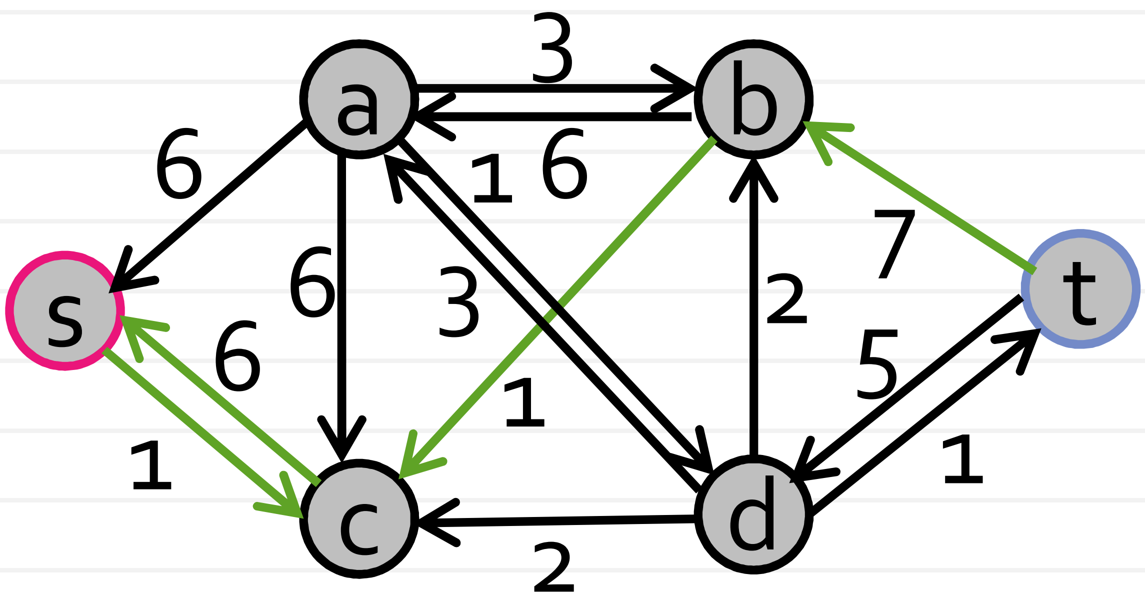
Residual graph

Solving max-flow: Ford-Fulkerson



Original graph

Flow = 12



Residual graph

Solving max-flow: Ford-Fulkerson

How to choose the augmented path?

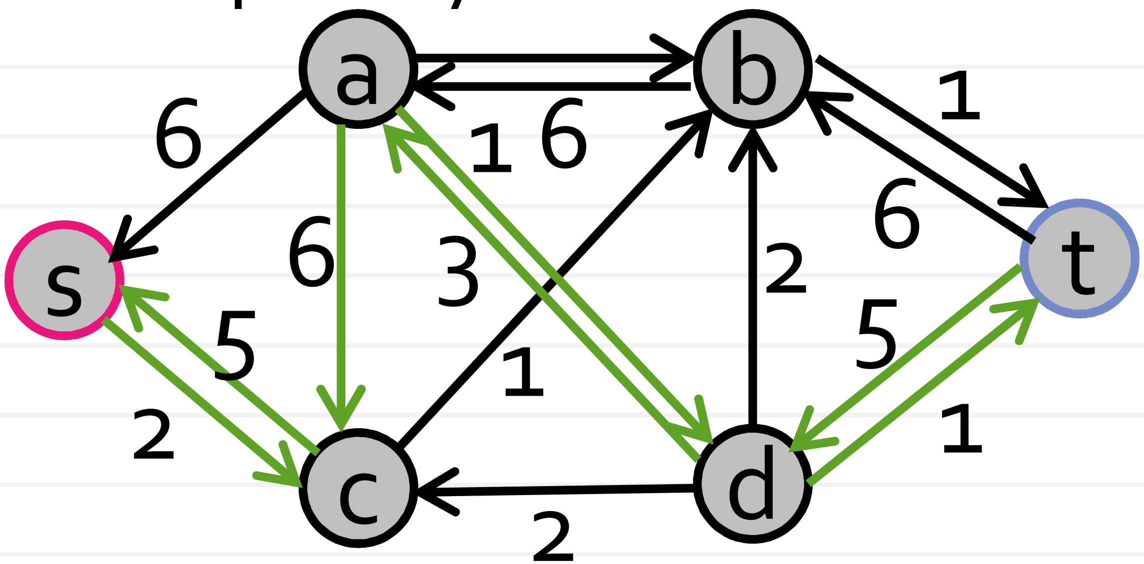
"Thickest" path: Ford-Fulkerson algorithm

- O(FE) – pseudopolynomial complexity

"Shortest" path: Edmonds-Karp algorithm

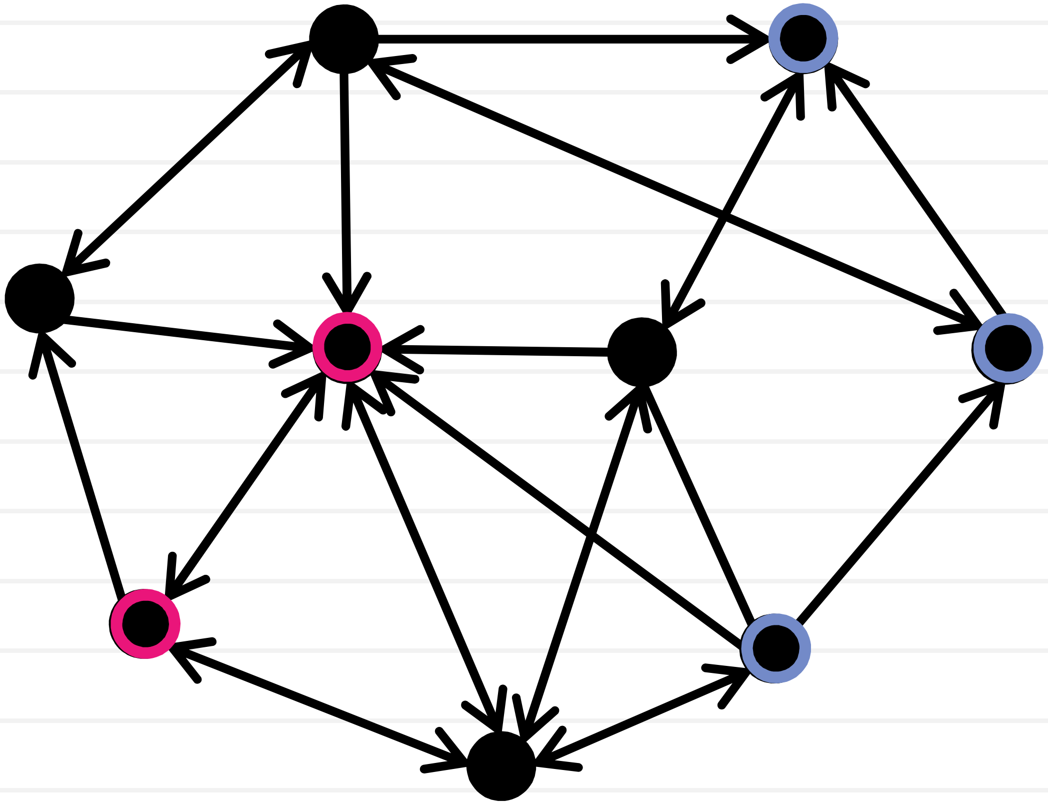
- $O(VE^2)$ – polynomial complexity

Current best: O(VE)



Network simplex for min-cost flow

Min-cost flow (*uncapacitated* version):



$$\min_f c^T f = \sum_{i,j} c_{ij} f_{ij}$$

$$\text{s.t.: } Af = b$$

$$\sum_{i \in O(j)} f_{ji} - \sum_{i \in I(j)} f_{ij} = b_j$$

$$f \geq 0$$

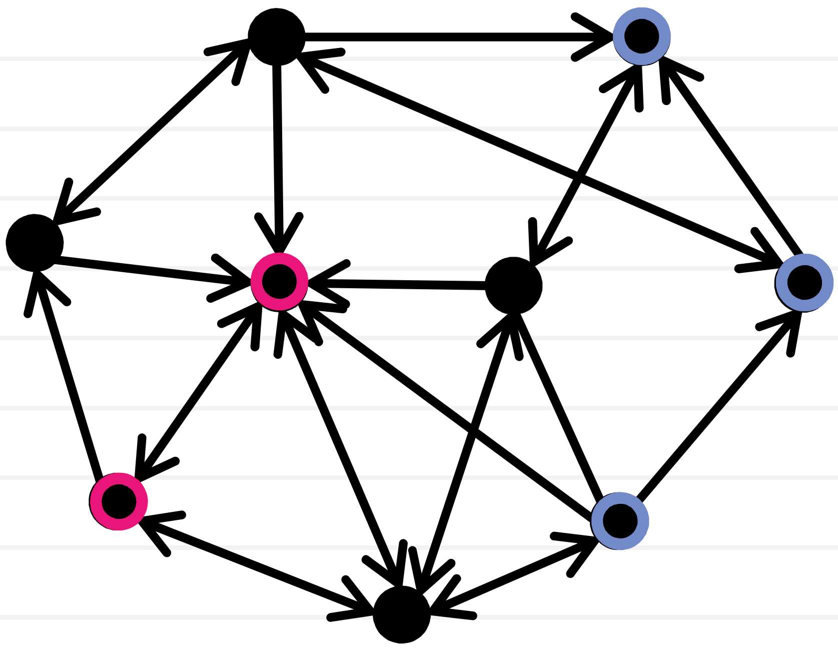
This is LP in a standard form!

|A| variables.

|V|-1 linearly-independent equations.

How do basic feasible solutions look like?

Network simplex ideas



$$\min_f c^T f = \sum_{i,j} c_{ij} f_{ij}$$

$$\text{s.t: } Af = b \quad f \geq 0$$

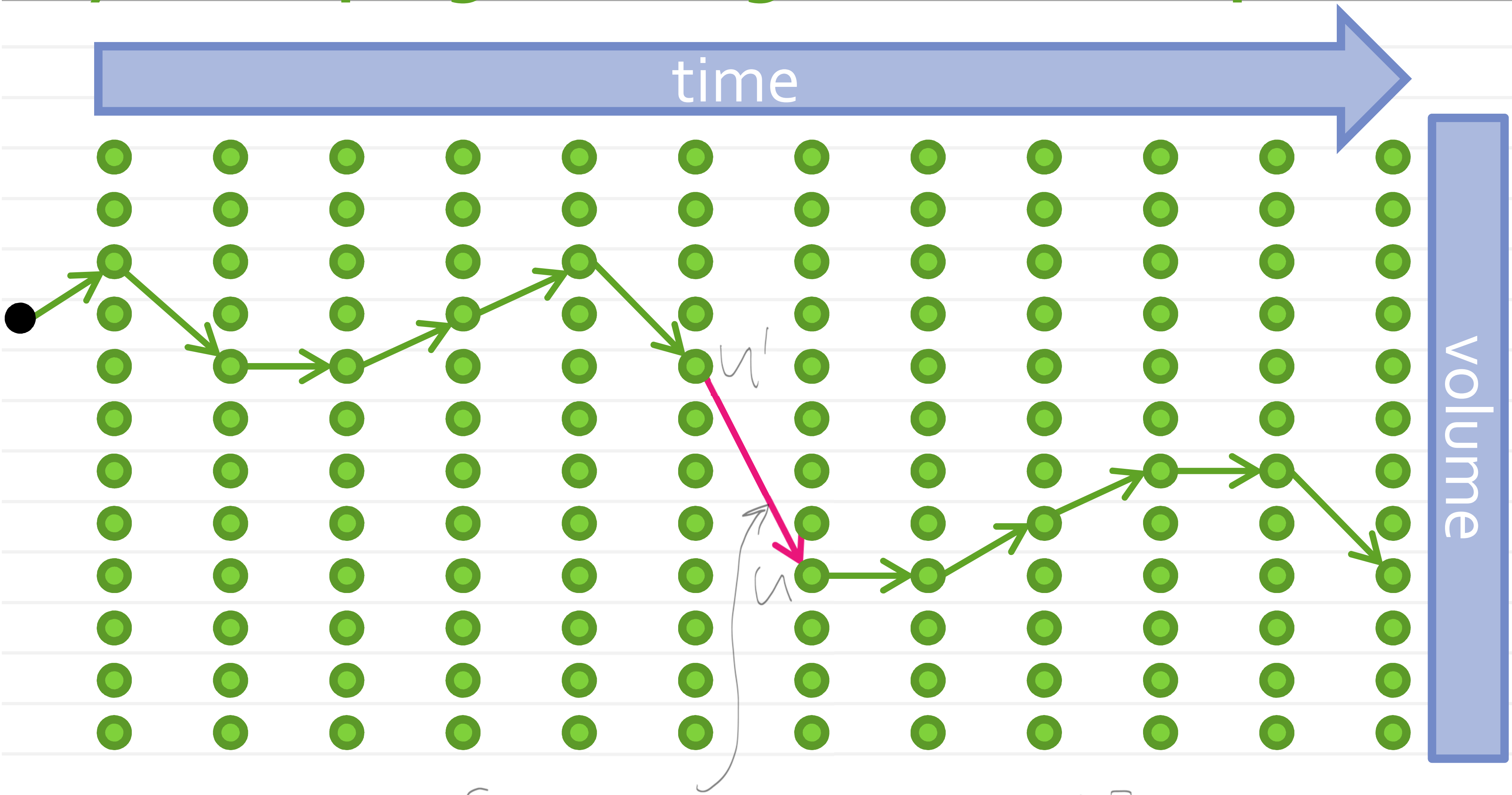
$$\sum_{i \in O(j)} f_{ji} - \sum_{i \in I(j)} f_{ij} = b_j$$

- Network simplex operates with basic solutions/canonical forms corresponding to spanning trees (of arcs with non-zero flow)
- Each time, an entering variable corresponds to an arc not in a tree
- During pivoting an arc is added to the tree creating a cycle, then one of the arcs in the cycle leaves the basic feasible solution

Networks and optimization

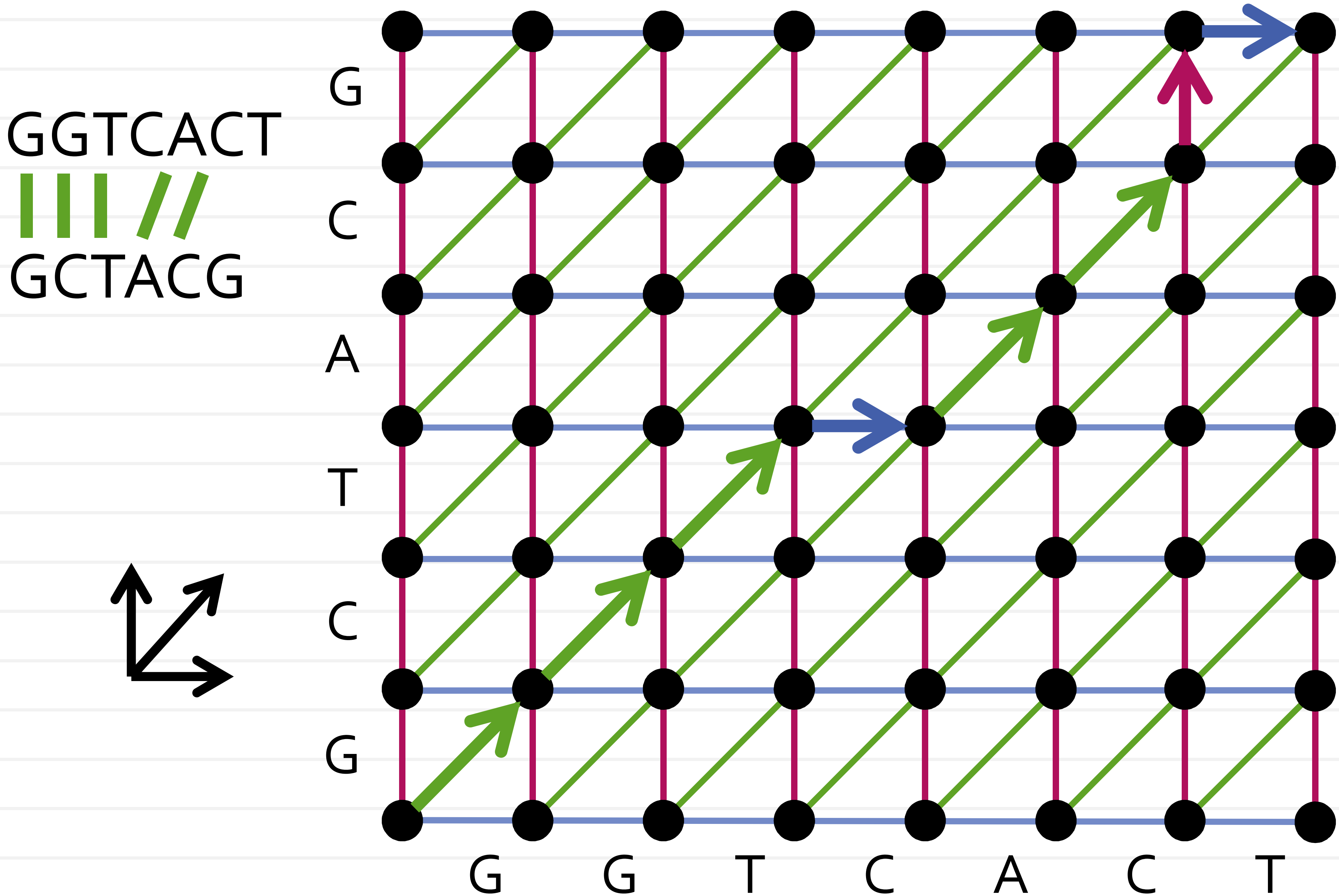
- A lot of optimization problems can be reduced to network optimization
- Due to integrality property, such reductions are very useful for integer programming

Dynamic programming and shortest paths

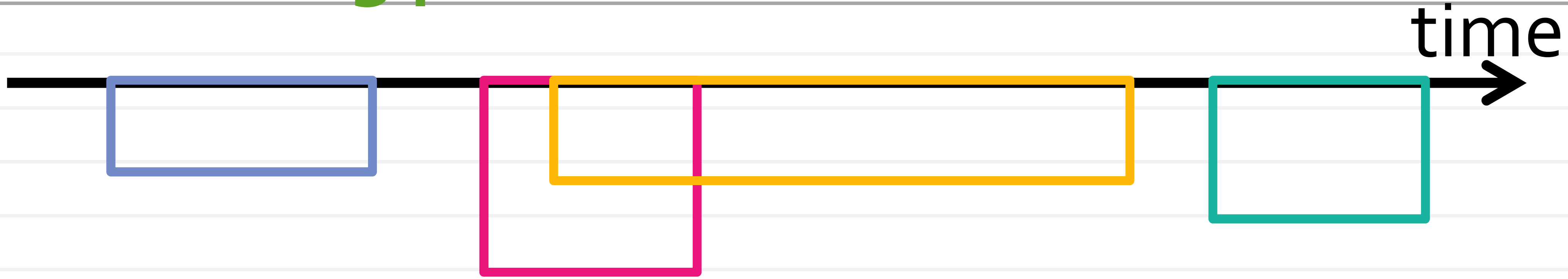


$$\varphi\left[d_n - \psi\left(u - u' + f_n - s(u')\right)\right]$$

Dynamic programming and shortest paths

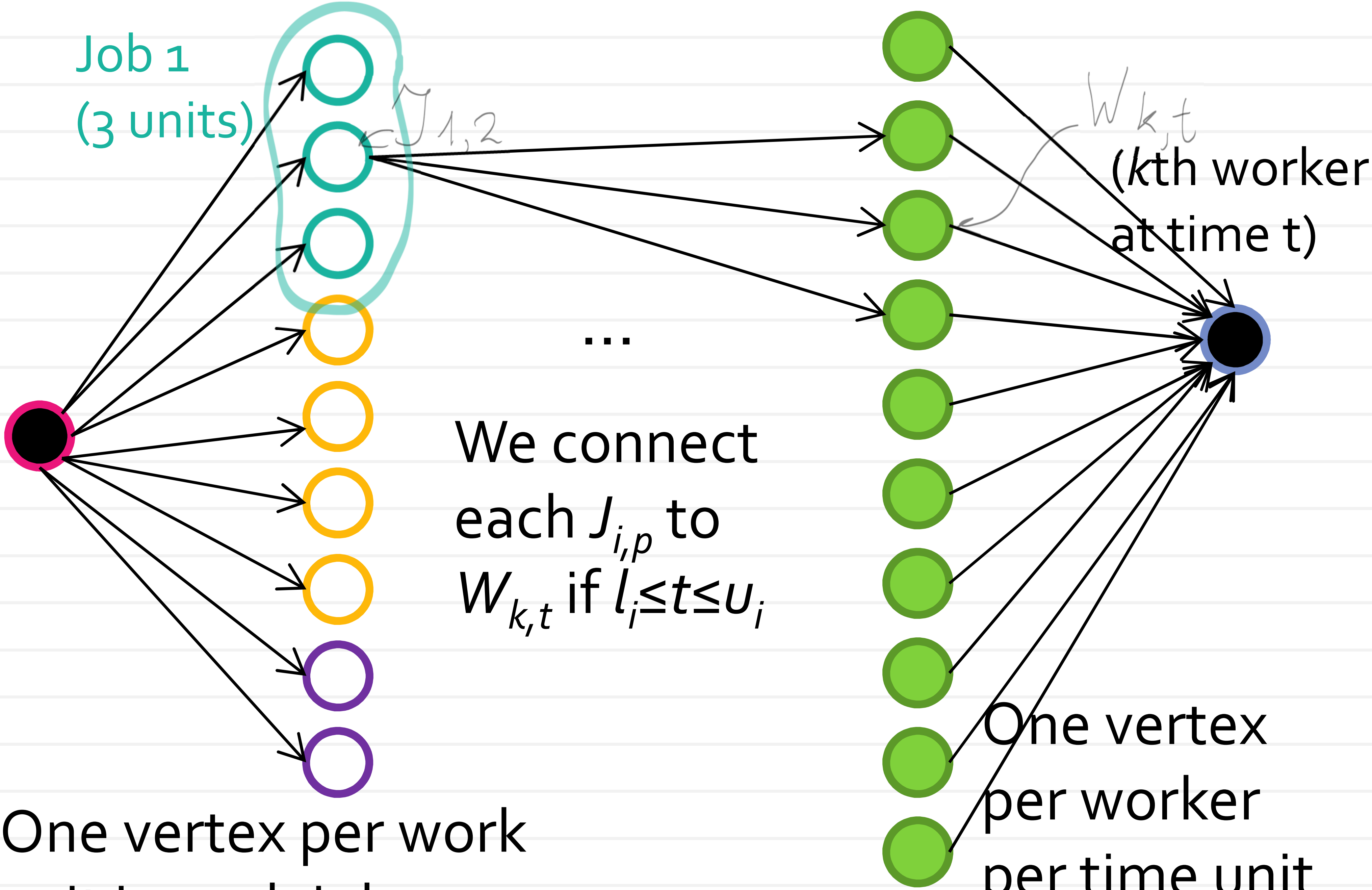


Scheduling problem



- We have a sequence of jobs
- Each job *i* is characterized by the amount of work w_i (in work units), the release date l_i, and the due date u_i
- We have workers, each of which work on a certain schedule (can be arbitrary) and can do one unit of work per unit time
- How to schedule to get as much work done as possible?

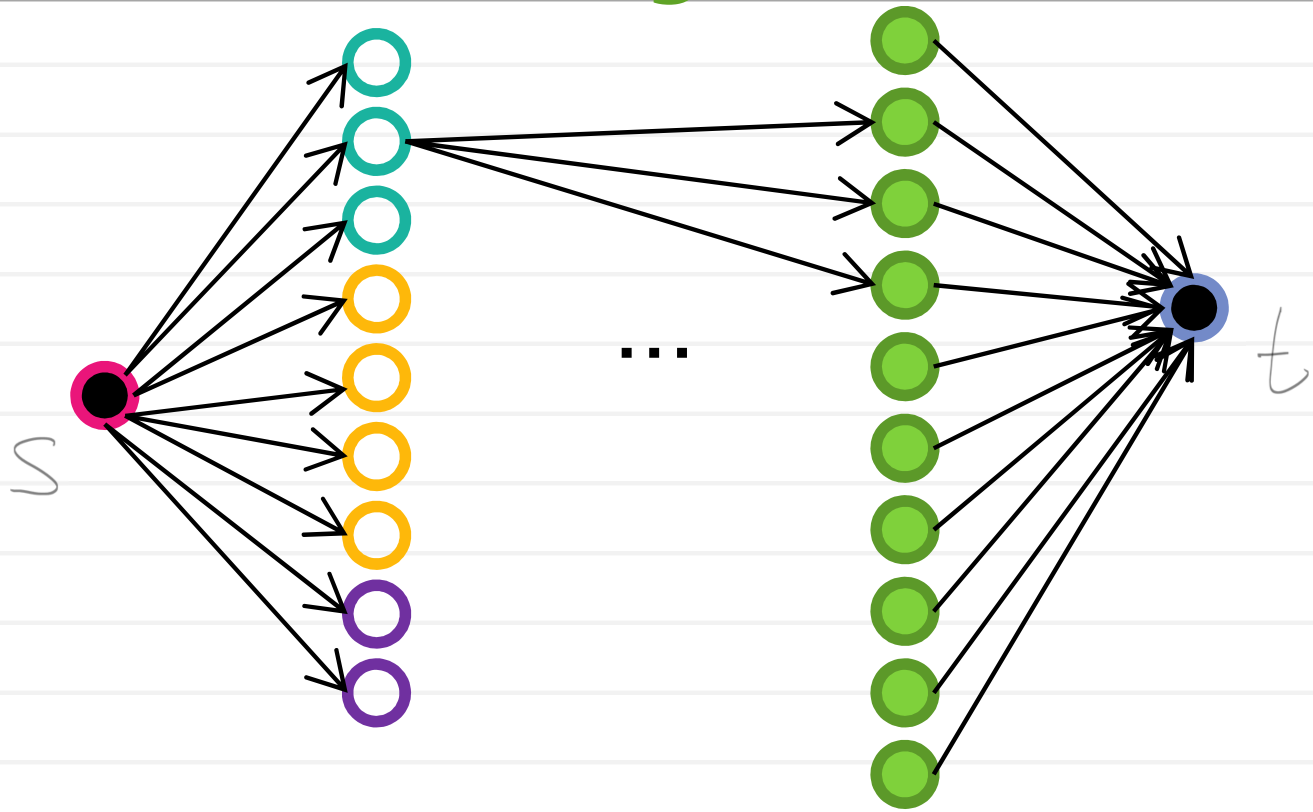
Solution to scheduling



We connect each $J_{i,p}$ to $W_{k,t}$ if $l_i \le t \le u_i$

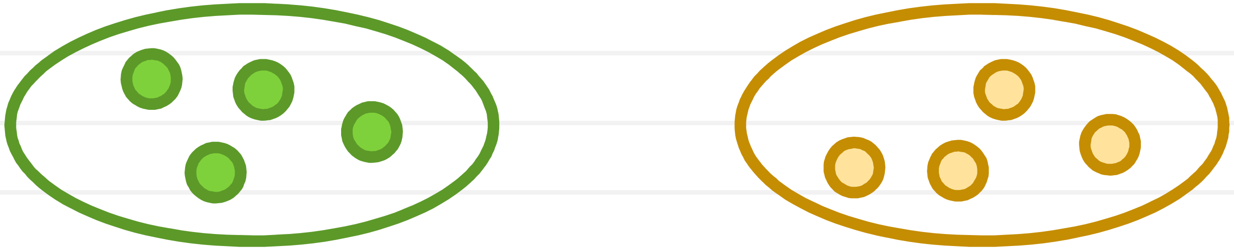
One vertex per work unit in each job

Solution to scheduling



- All arc capacities are set to one
- The maximum flow gives the schedule (each worker knows what to do at each time)
- Can maximize work for very large problems
- Cannot maximize the number of completed jobs

Linear assignment problem



- One-to-one matching two sets M and N of equal size (workers and projects, marriages, points at time t and t+1, etc.)
- a_{ij} is a cost of matching m_i and n_j

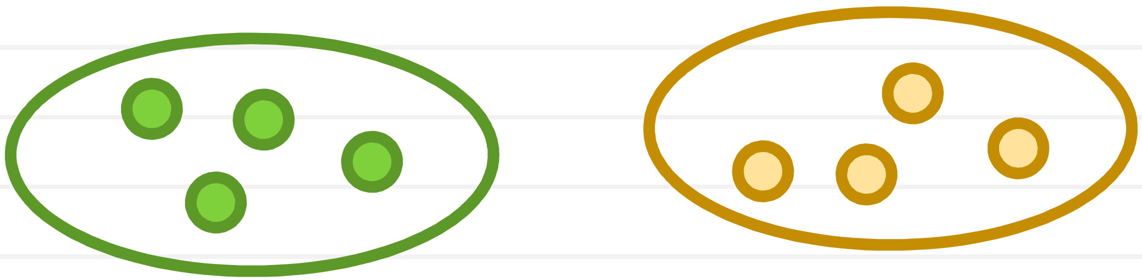
$$\min \sum_{i=1}^{|M|} a_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_i x_{ij} = 1$$

$$\sum_j x_{ij} = 1$$

$$x_{ij} \in \{0;1\}$$

Reduction to mincost flow

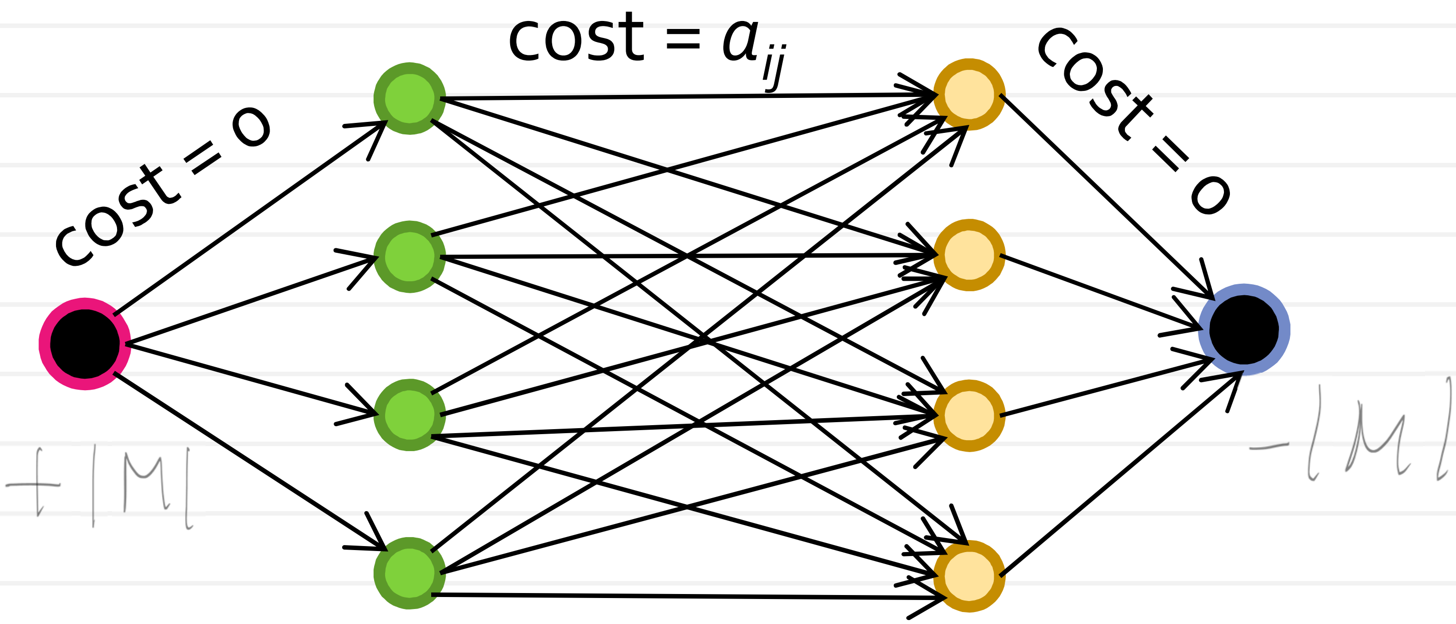


$$\min \sum_{i=1}^{|M|} a_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_i x_{ij} = 1$$

$$\sum_j x_{ij} = 1$$

$$x_{ij} \in \{0;1\}$$



(all capacities=1)

- More efficient specialized algorithms exist (Hungarian)
- What about allowing not to match (at a cost)?

Optimal Cargo Transport

(aka “hopping airplane”)

- a_{ij} containers can be delivered from i to j
- Delivering each gives a profit c_{ij}
- The ship capacity is S
- Maximize profit by choosing which containers to transport

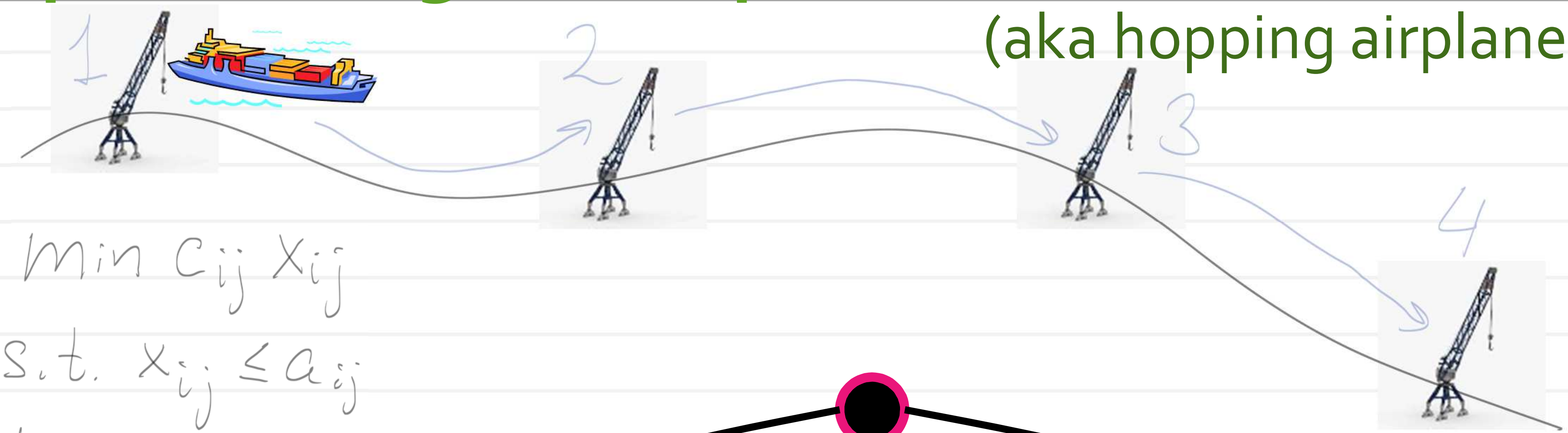
$$\min c_{ij} x_{ij}$$

$$\text{s.t. } x_{ij} \le a_{ij} \qquad x_{ij} \in \mathbb{Z}_+$$

$$\forall t \quad \sum_{\substack{i \le t \\ j \ge t+1}} x_{ij} \le S$$

Optimal Cargo Transport

(aka hopping airplane)

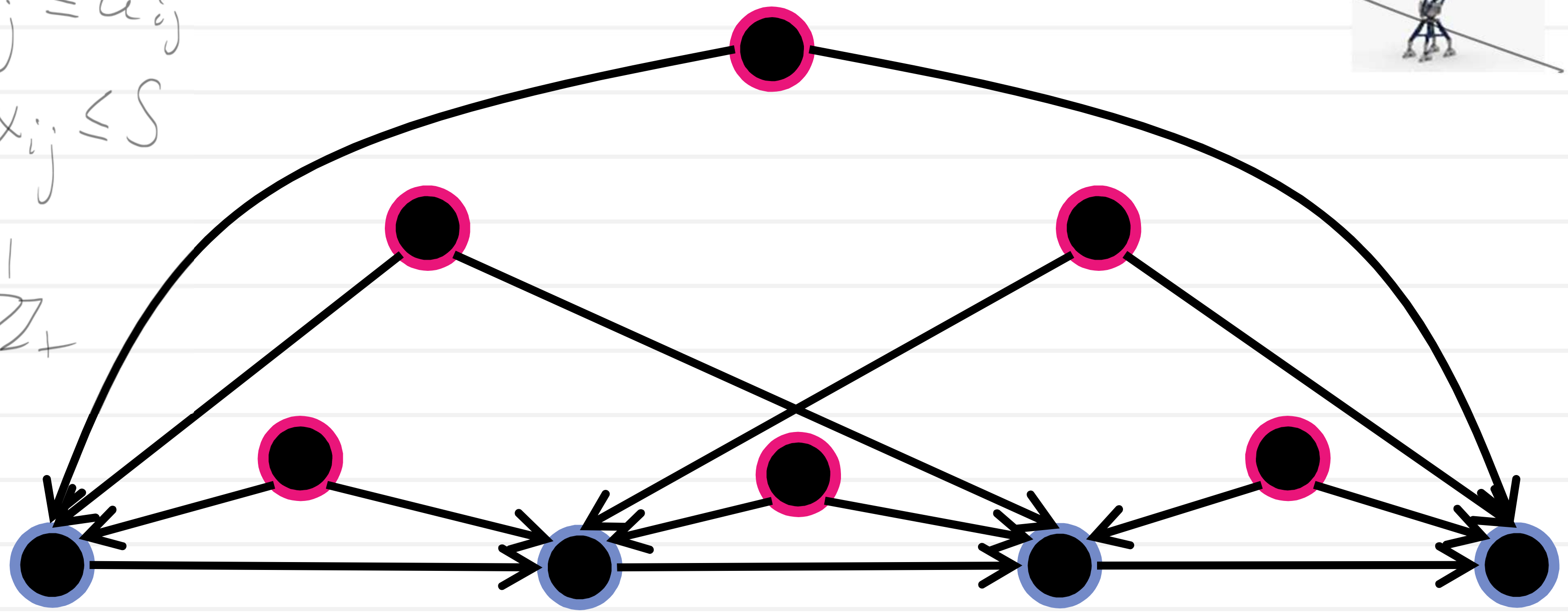


$$\min c_{ij} x_{ij}$$

$$\text{s.t. } x_{ij} \le a_{ij}$$

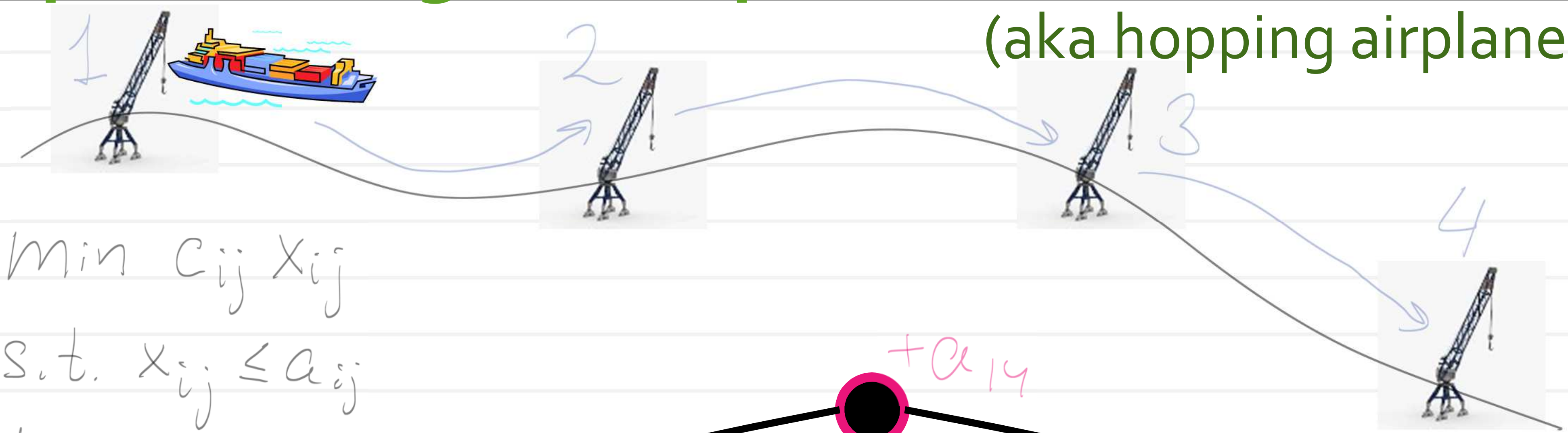
$$\forall t \quad \sum_{\substack{i \le t \\ j \ge t+1}} x_{ij} \le S$$

$$x_{ij} \in \mathbb{Z}_+$$



Optimal Cargo Transport

(aka hopping airplane)

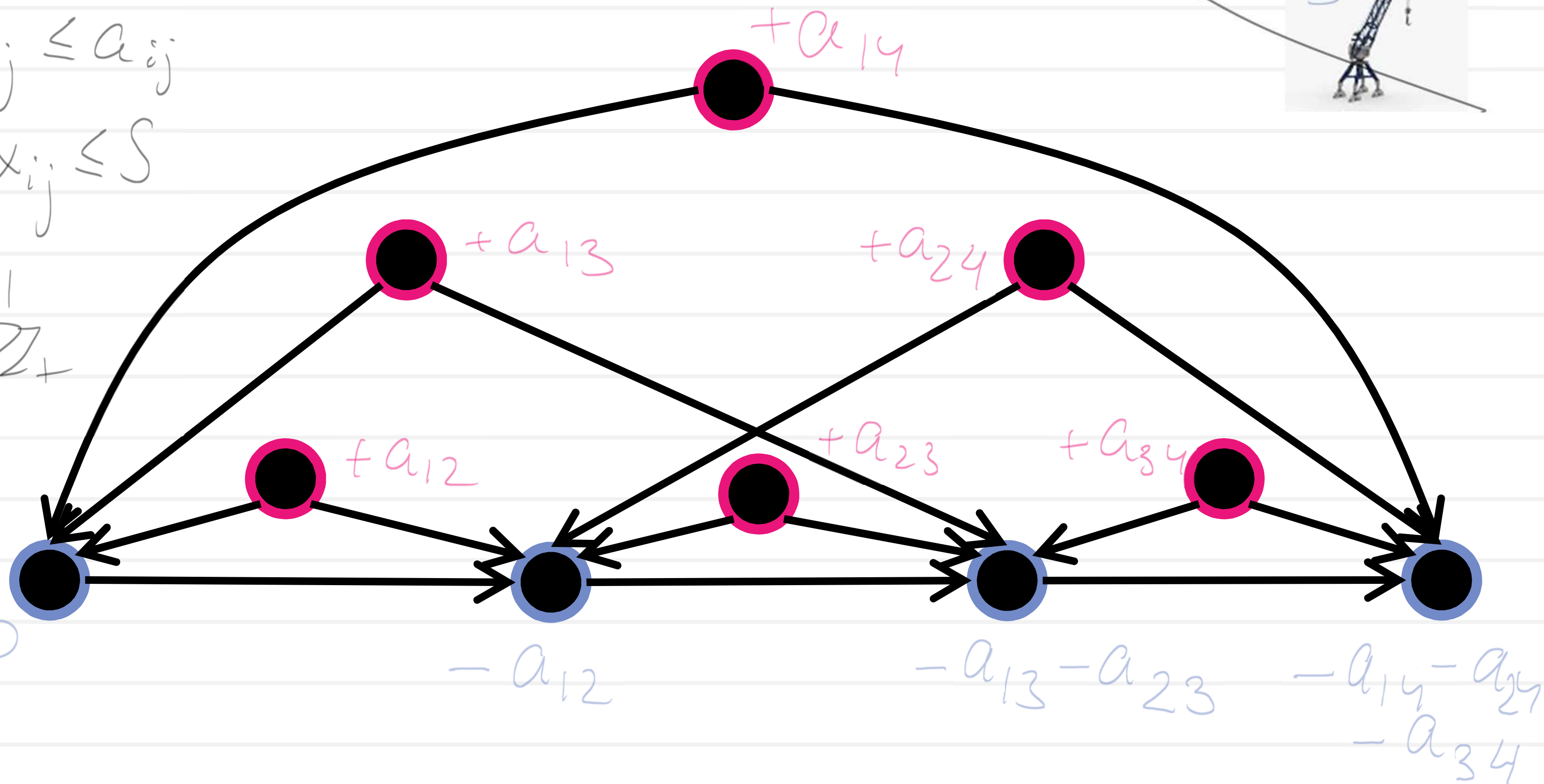


$$\min \; c_{ij} x_{ij}$$

$$\text{s.t. } x_{ij} \le a_{ij}$$

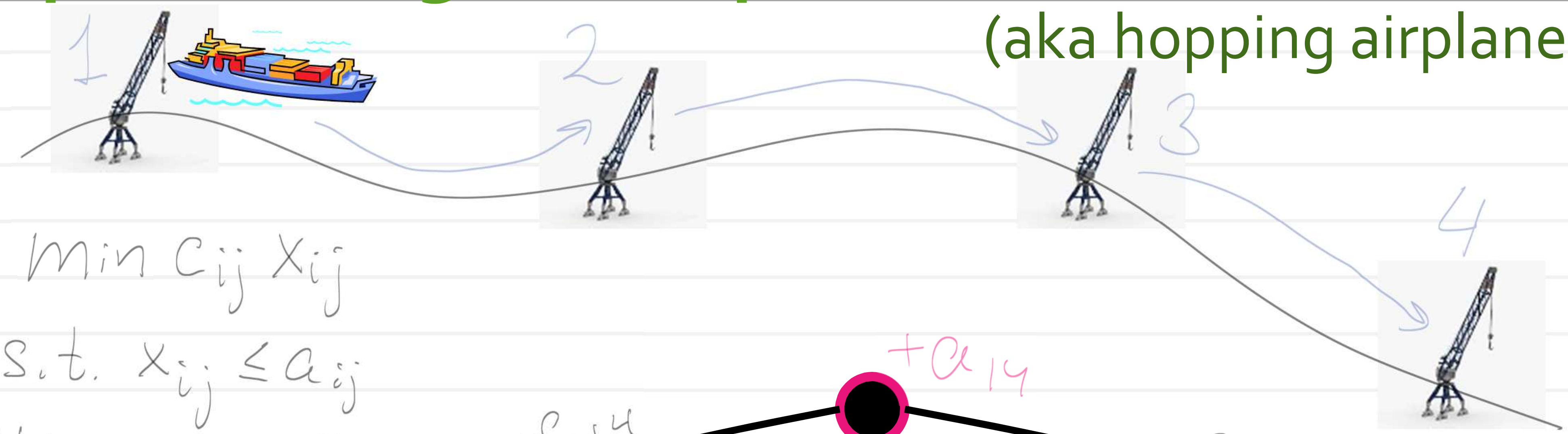
$$\forall t \quad \sum_{\substack{i \le t \\ j \ge t+1}} x_{ij} \le S$$

$$x_{ij} \in \mathbb{Z}_+$$



Optimal Cargo Transport

(aka hopping airplane)

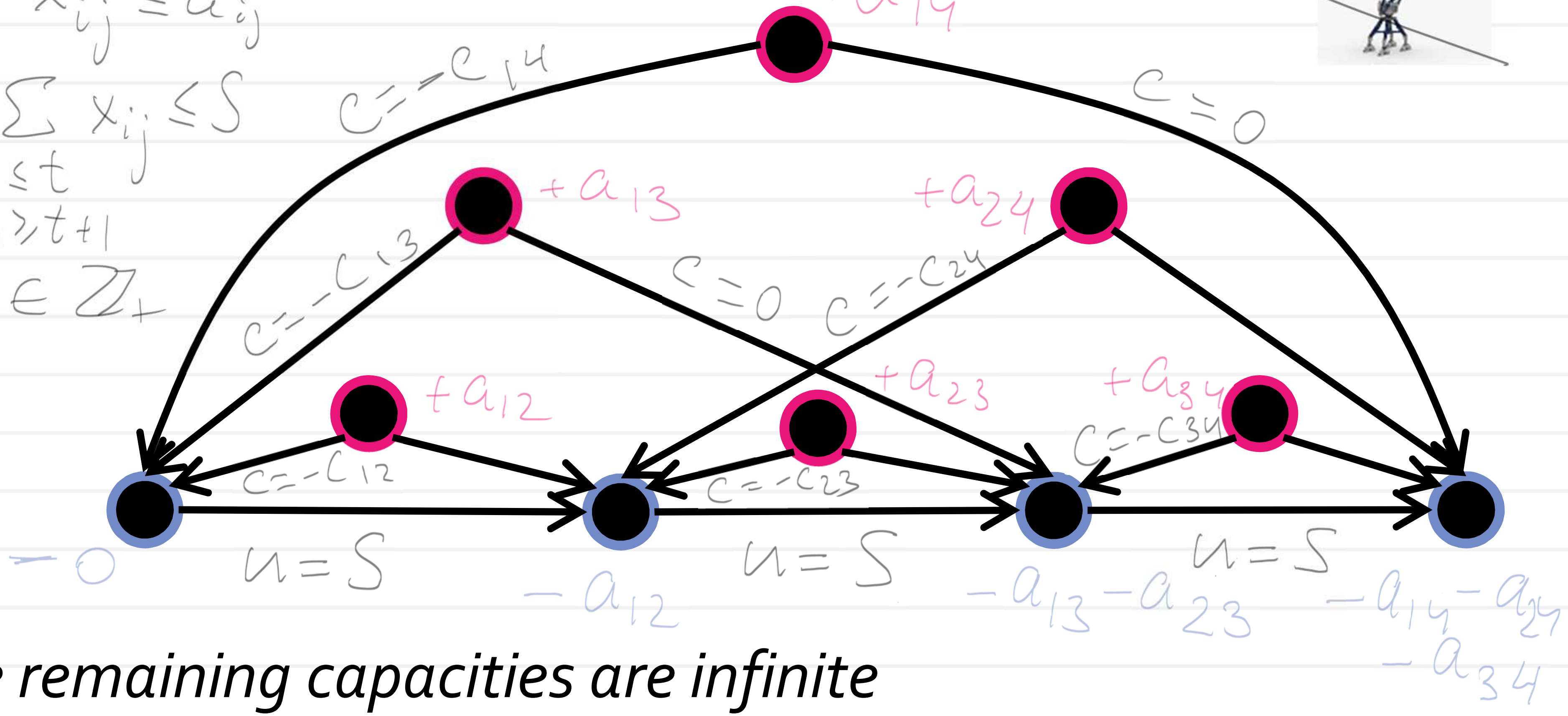


$$\min c_{ij} x_{ij}$$

$$\text{s.t. } x_{ij} \le a_{ij}$$

$$\forall t \sum_{\substack{i \le t \\ j \ge t+1}} x_{ij} \le S$$

$$x_{ij} \in \mathbb{Z}_+$$



The remaining capacities are infinite

The remaining costs are zeros