RFDP - Class 9 : Routing problems

→ 10 am : plenary presentation in Cauchy (LocalSolver)

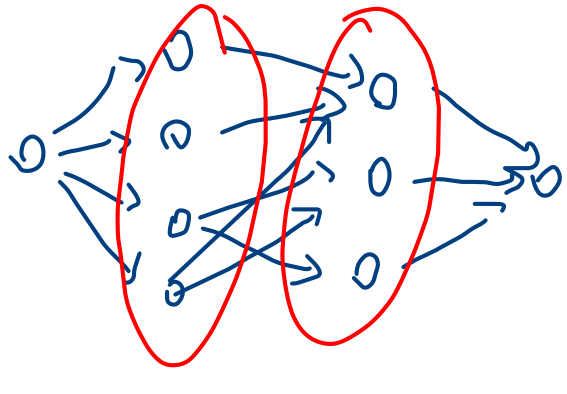
I/ Homework

Ex 13.4 : Bin packing (put items with sizes $a_1, \dots, a_n$ in a minimum number of bins with capacity W)

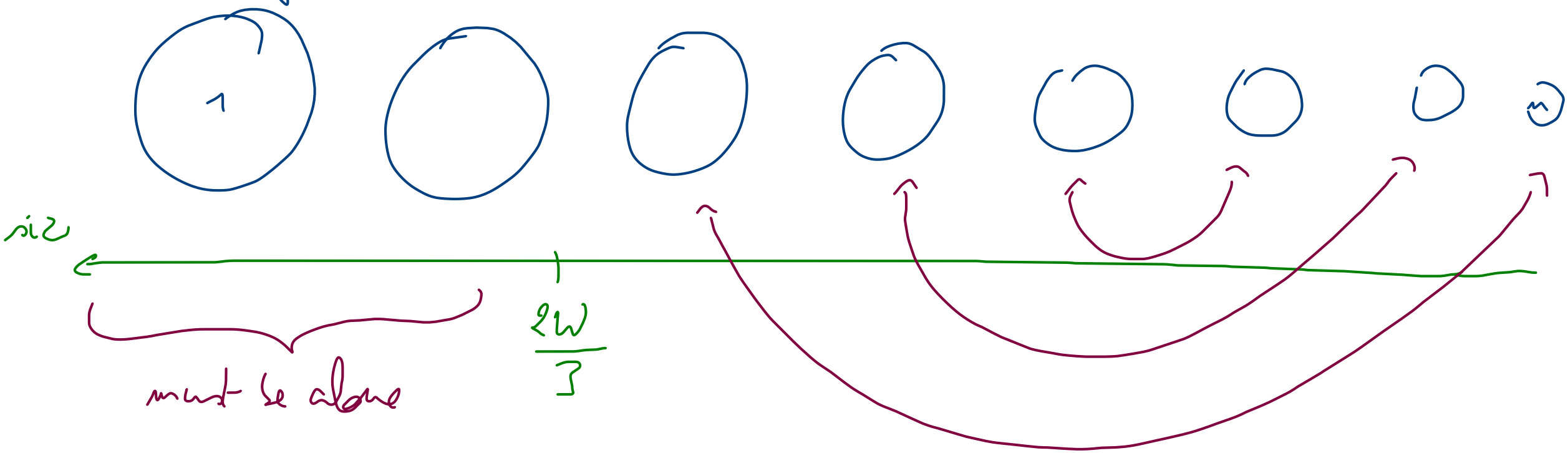
Hypothesis : "large items" $\forall i,\ a_i > \frac{1}{3}W$ (in the textbook sometimes $W=1$)

1) Build a graph $G = (V, E)$ with one vertex per item i : $V = [1, n]$
For two items i, j we add an edge $\{i, j\}$ if $a_i + a_j \leq W$
Since all item sizes are $> W/3$, there can be no more than 2 items per bin. Finding an optimal packing is equivalent to finding a maximum cardinality matching M in G. If an item i has no edge of the matching incident to it, then i is alone in a bin. Edges of M correspond to bins with 2 items.

2) To use flow techniques for matchings we built a graph of the form
The crucial assumption was a bipartite graph :
not the case here



We can actually solve this with sorting: we sort the items by decreasing size

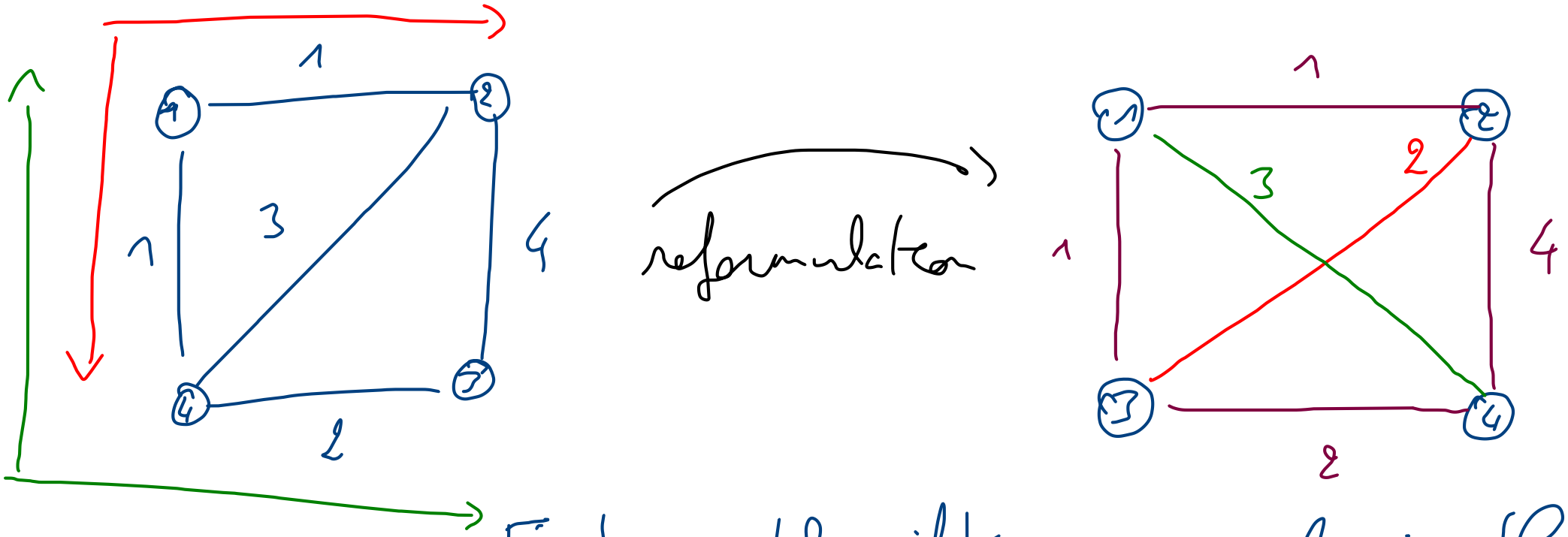


we can iteratively try to match big items with small ones ... TBC

II/ The Traveling Salesperson Problem (TSP)

For the rest of this class we focus on problems with
- a single vehicle / person
- an undirected graph $G = (V, E)$ with an edge cost function $c : E \to \mathbb{R}_+$

Goal: find a cycle C of minimum cost $\sum_{e \in C} c(e)$ that visits every vertex $\geq$ once



Reformulation : complete graph K_n (where all edges exist) and edge costs $\tilde{c}$ correspond to shortest path lengths in the initial graph G with costs c

This is the case we study, it is called the metric TSP because the new costs satisfy

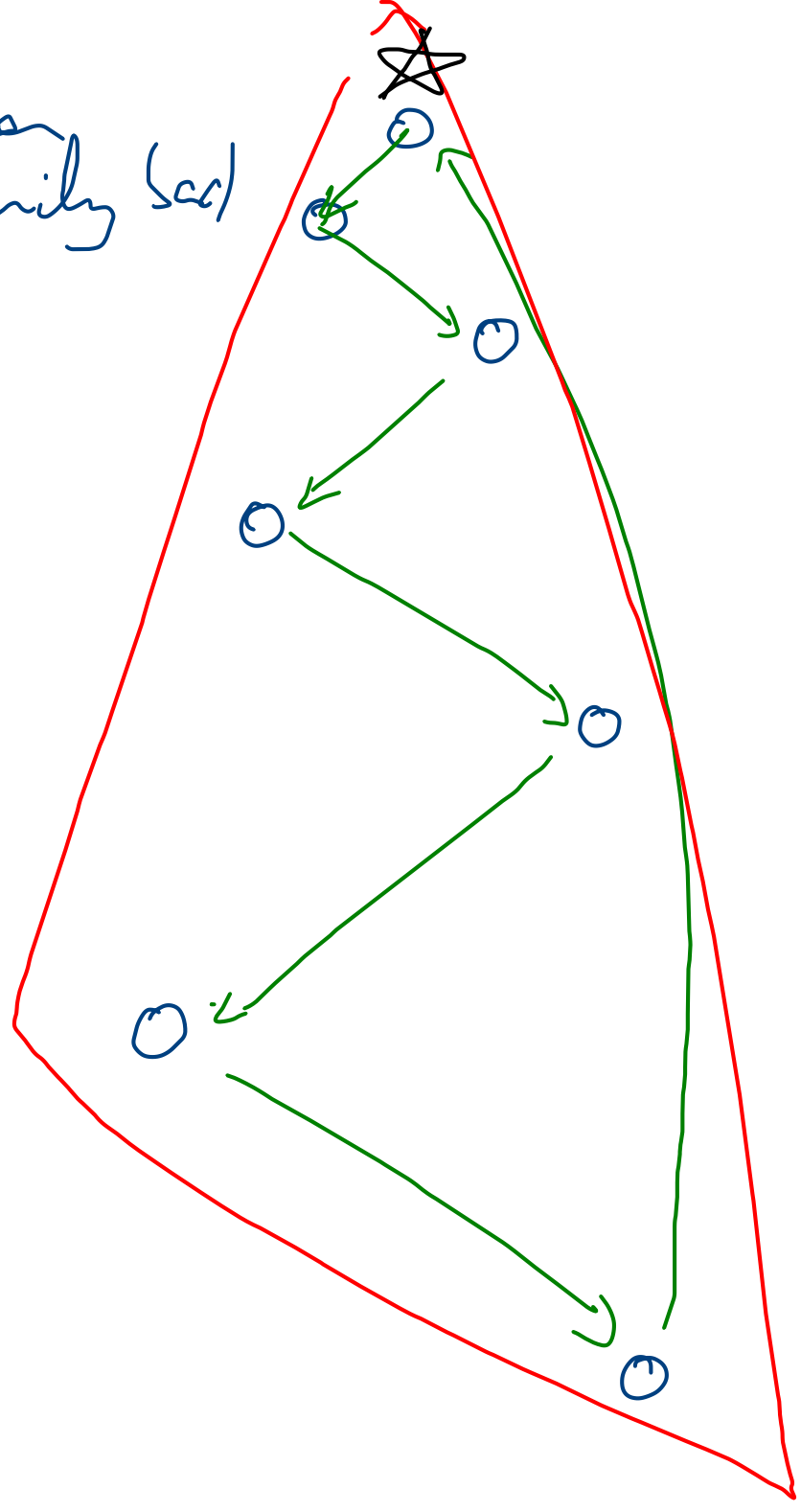
$$\tilde{c}(u,v) \leq \hat{c}(u,w) + \tilde{c}(w,v)$$ Triangle Inequality

A) Heuristics

Greedy heuristic : nearest neighbor

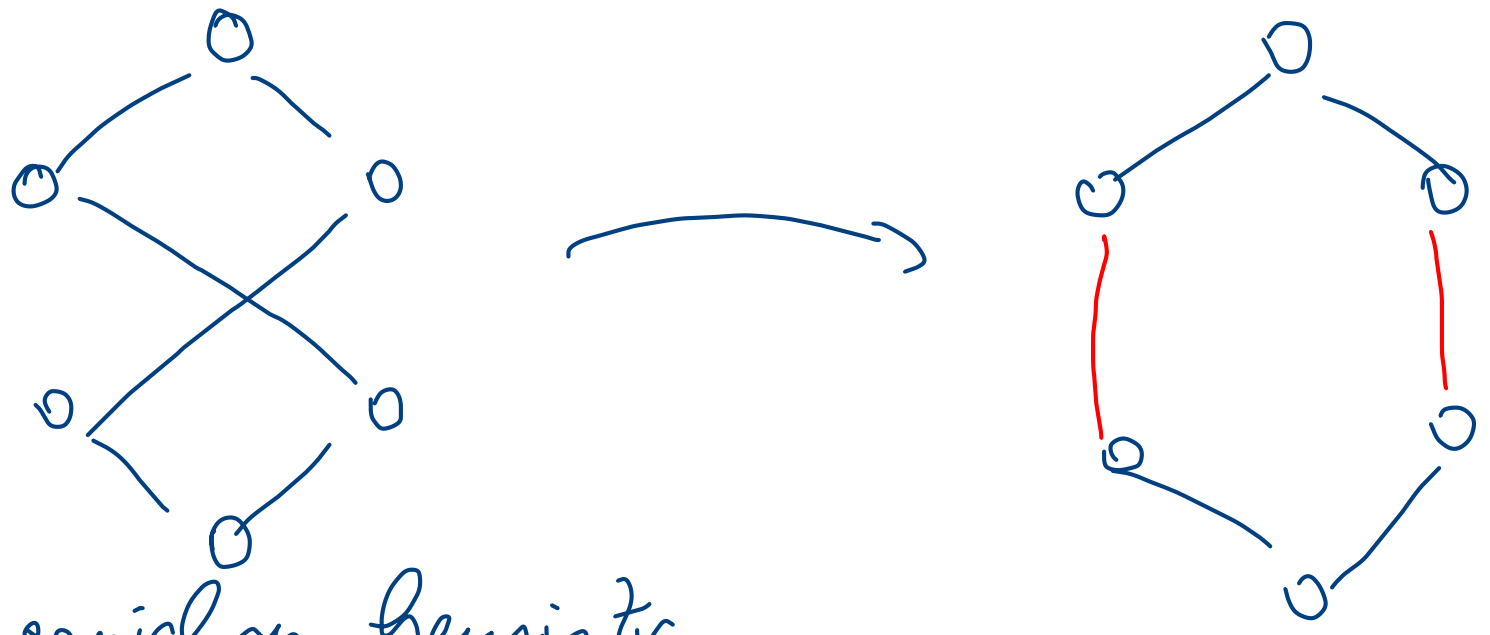
nearest neighbor
can be arbitrarily bad

vs



Local search heuristics : find good neighborhoods / moves

- 2-OPT
  swap 2 edges
- 3-OPT
  k-OPT



Combining this with a few clever ideas → Lin-Kernighan heuristic close to the state of the art on large instances

Does not provide performance guarantees

B) Approximation algorithms ?

$\tilde{c}$

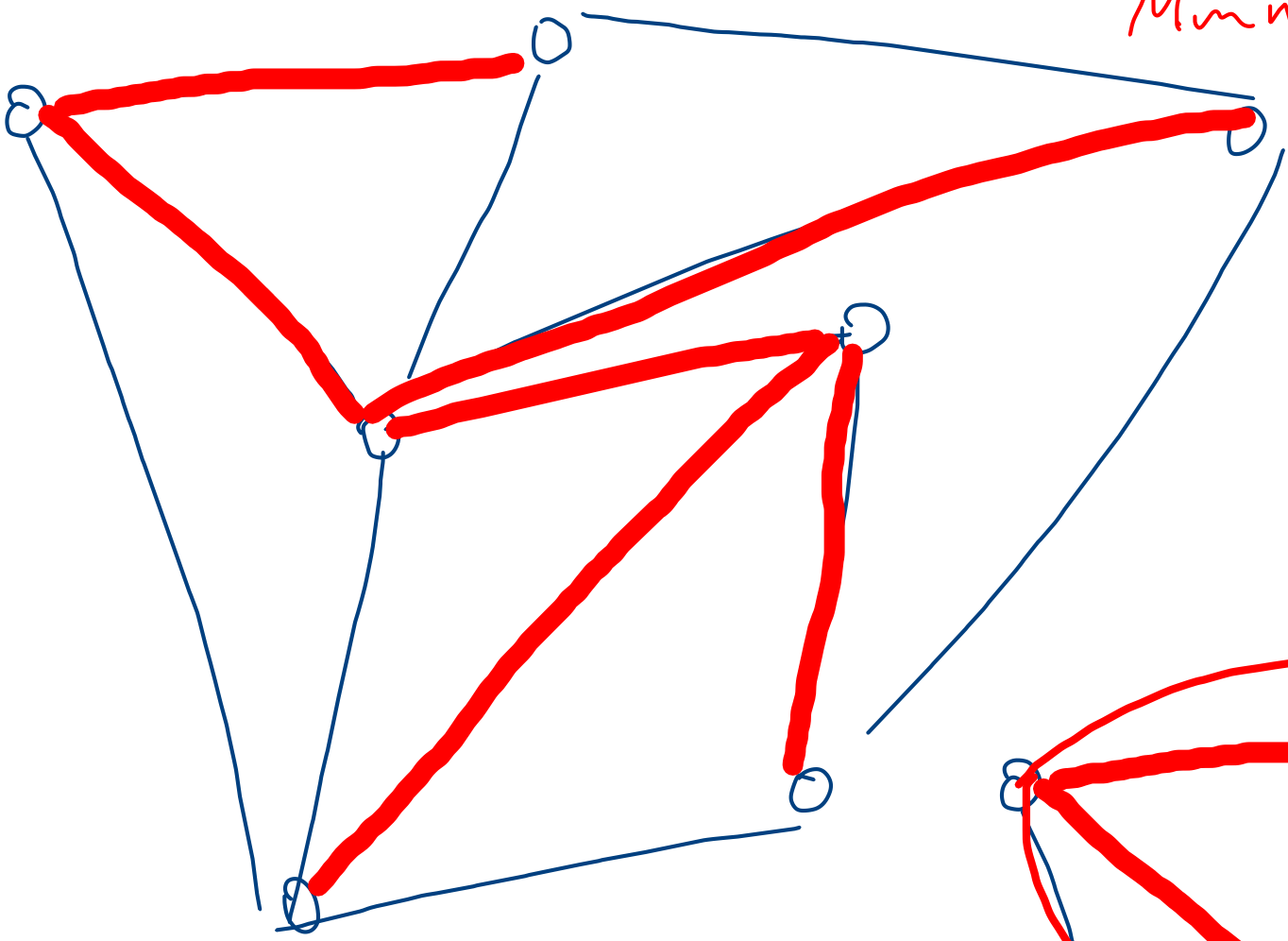
Double tree algorithm

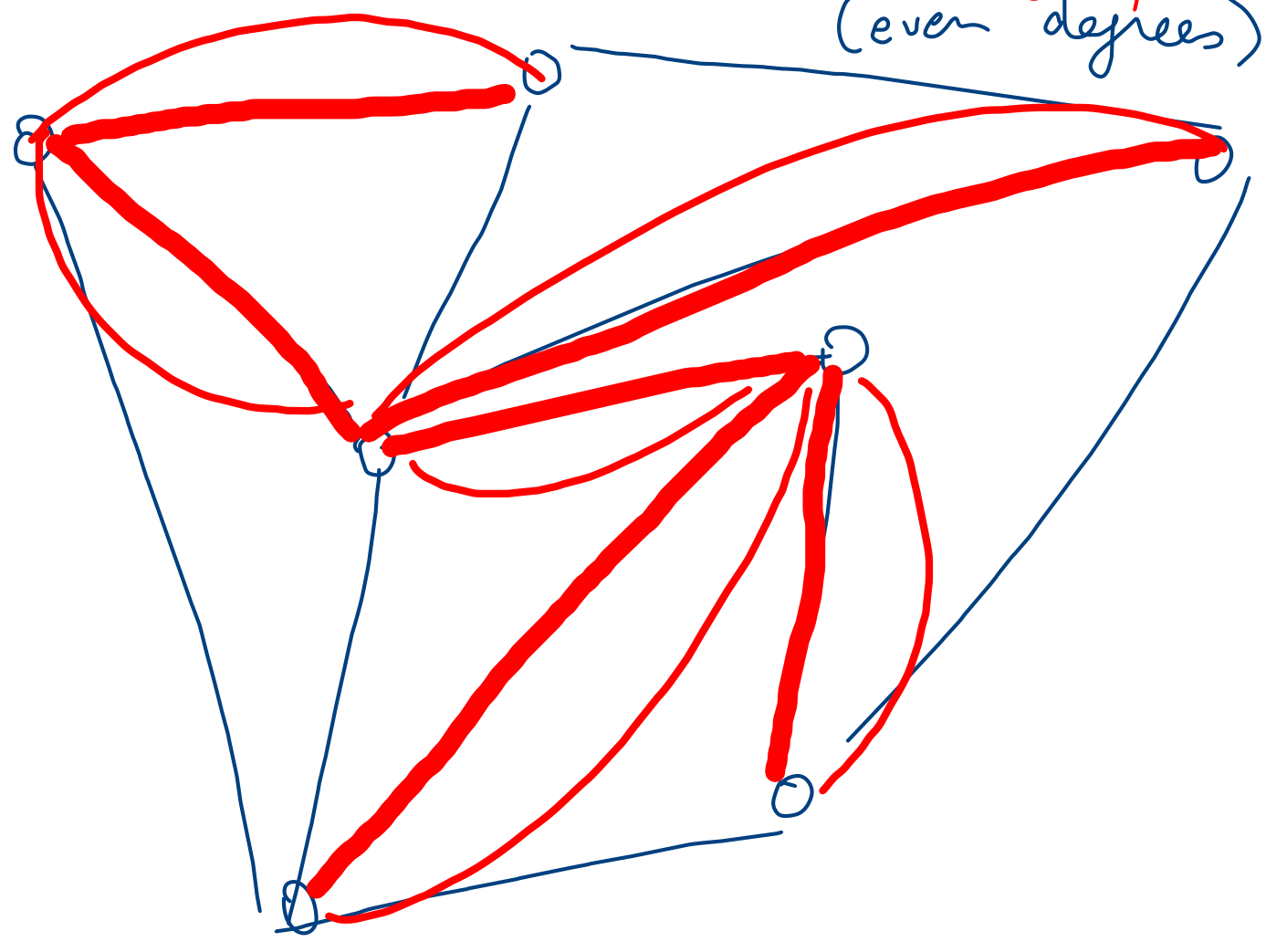
1. Find a Minimum Spanning Tree on (K_n, c)
2. Duplicate every edge of the MST → multigraph
3. Find a Eulerian path in the multigraph (visits each edge exactly once)
4. Follow this path and skip vertices you have already seen, to visit each vertex exactly once

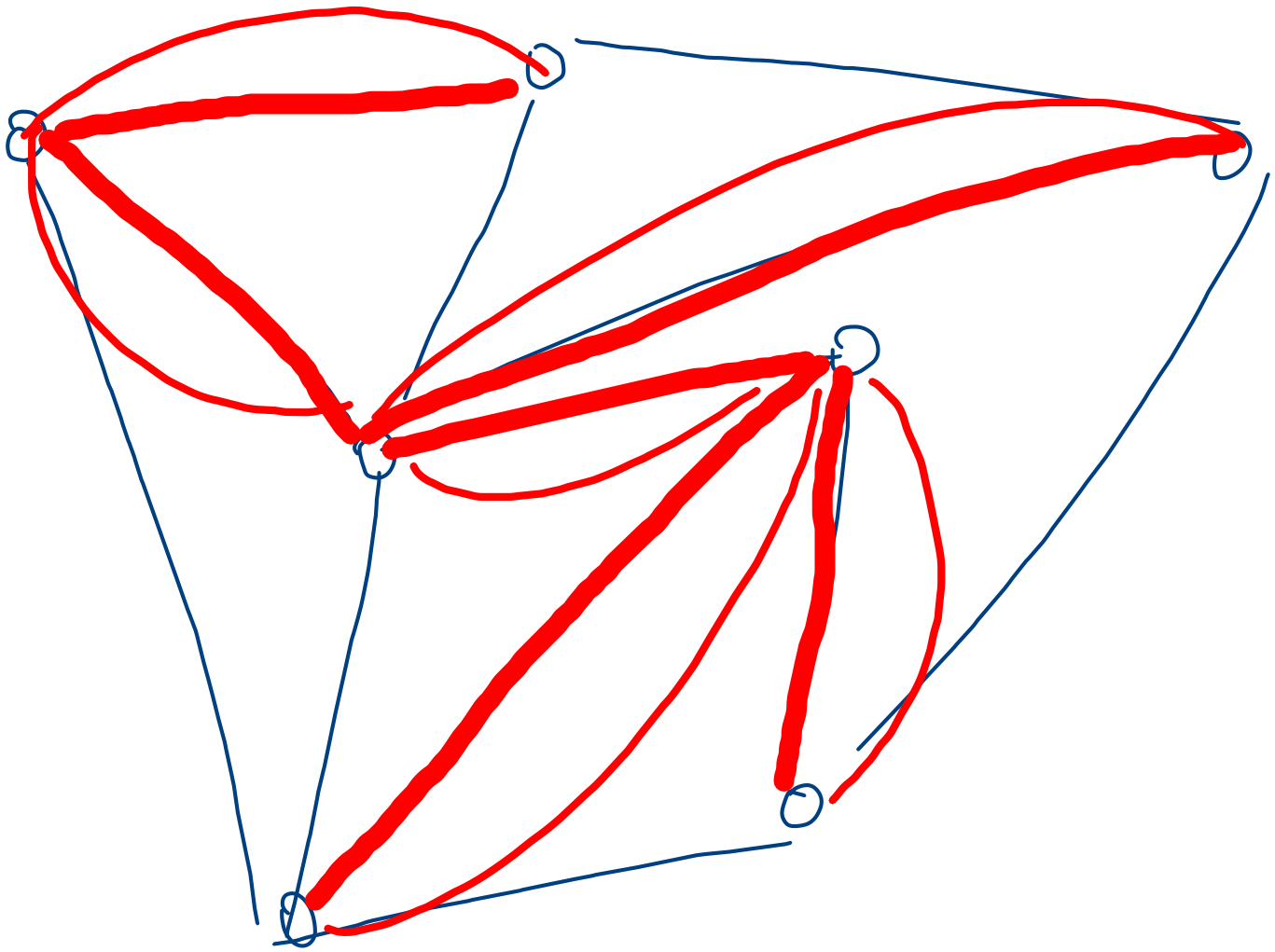
Minimum spanning tree (1)

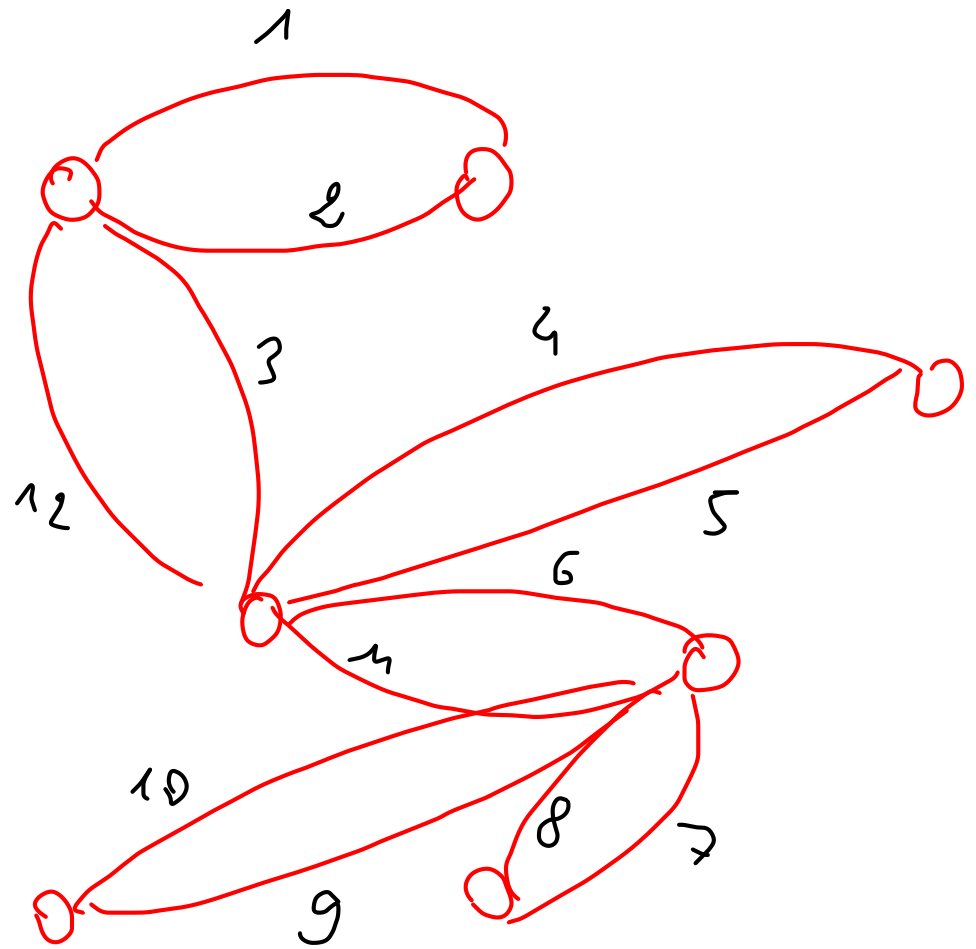
duplicate edges

Eulerian multigraph (2)
(even degrees)



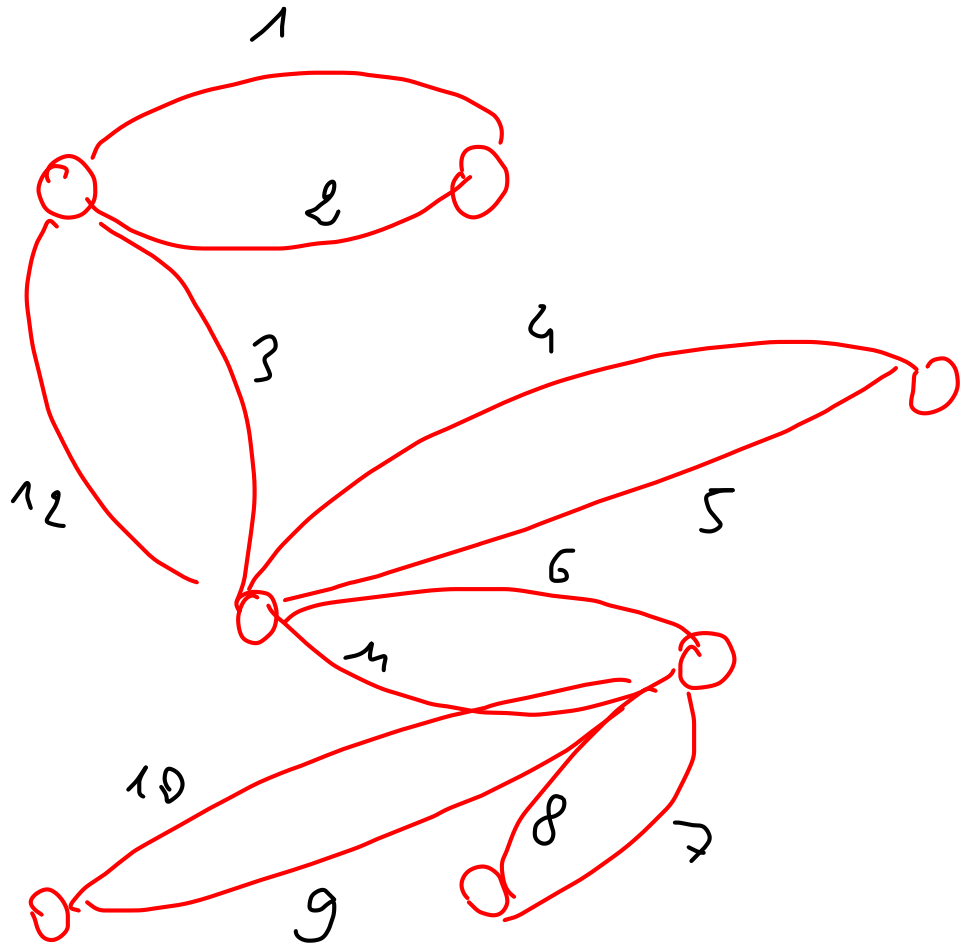




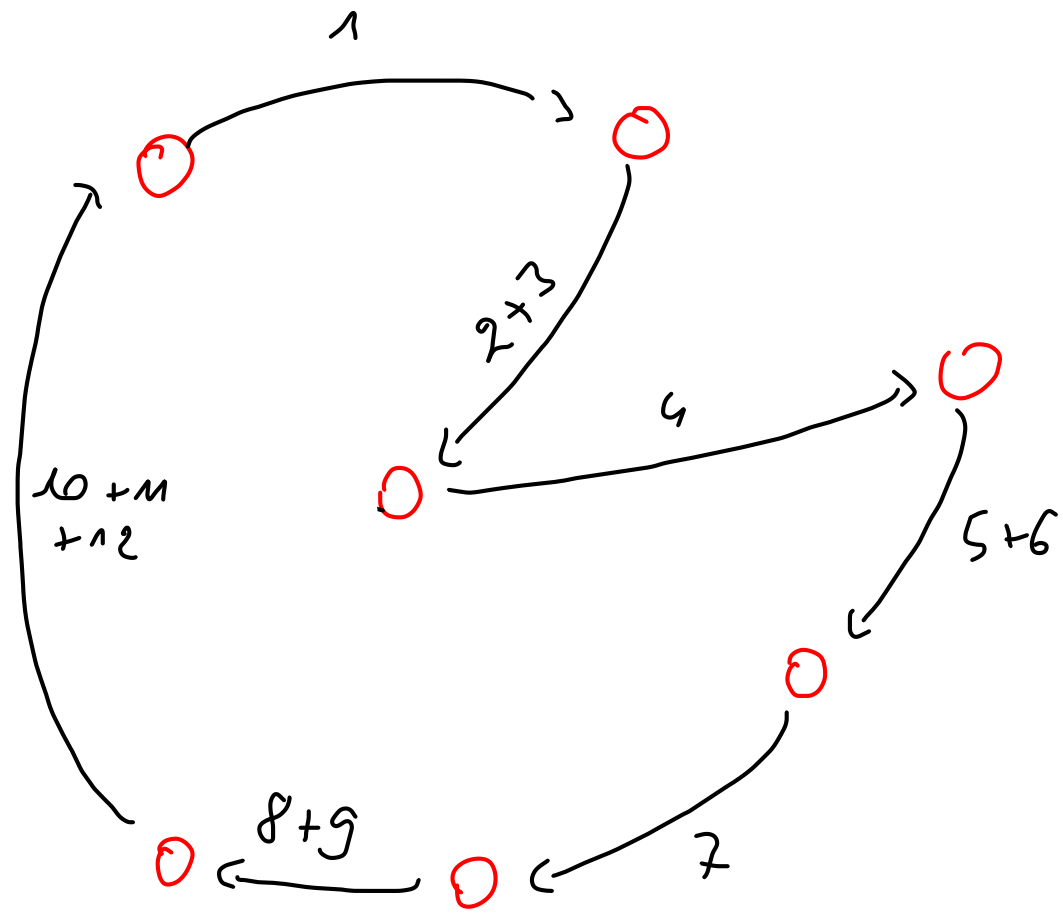


Eulerian path (3)

visits each edge exactly one



Taking shortcuts in the Eulerian path to avoid repeating vertices (shortcuted edges exist bc complete graph) (4)



Thm: The double tree algorithm has an approximation factor of 2

Proof: 1. A MST has lower cost than any (optimal) Hamiltonian cycle (going through every vertex exactly once)
Why? Take a Ham. cycle, remove any edge: you get a spanning tree, which is larger than a MST

$cost(MST) \leq OPT$

2.&3. Double the edges of the MST & find a Eulerian cycle

$cost(\text{Euler. cycle}) = cost(\text{duplicated MST}) \leq 2\,OPT$

4. Take shortcuts through the Eulerian cycle to turn it into a Hamiltonian cycle

T1: $c(u,v) \leq c(u,w) + c(w,v)$ → taking shortcuts does not increase the cost!

$cost(\text{Ham. cycle}) \leq cost(\text{Euler. cycle}) \leq 2\,OPT$

Christofides algorithm:

1. Find an MST called T
2. Find a minimum weight matching M between the vertices with odd degree in T
3. Duplicate the edges of M & find a Eulerian cycle in $T \cup M$
4. Take shortcuts to get a Hamiltonian cycle

Thm: The Christofides algorithm has an approximation ratio of $3/2$

Proof: in the textbook

B) Integer programming

$x_e = \begin{cases} 1 & \text{if edge } e \text{ is part of the selected cycle} \\ 0 & \text{otherwise} \end{cases}$

$$\min \sum_{e \in E} c(e)\, x_e \quad \text{subject to}$$

$$\begin{cases} x_e \in \{0,1\} & \forall e \in E \\ \sum_{e \in \delta(v)} x_e = 2 & \forall v \in V \\ \sum_{e \in \delta(X)} x_e \geq 2 & \forall X \subset V,\ X \neq \emptyset,\ X \neq V \end{cases}$$

in a cycle, every vertex has 2 incident edges, but this also allows unions of cycles

The last constraint is called "subtour elimination": each nontrivial subset of vertices must have at least one way in & one way out (very similar to the MST formulation)

There are $2^n - 2$ subsets X to handle

Branch & Bound → Branch & Cut

When we solve the continuous relaxation at a B&B node, we start with a small set of constraints and add violated constraints iteratively until we can prove that no more constraints are violated

This works because we can find such a "violated X" in poly time (separation pb)

HW: how?

Remark: Here the subproblems on B&B are a little harder than the root (constrained TSP) → textbook / my notes

III / Postperson Tour Problem

Goal: find a cycle C of minimum cost that crosses every edge at least once

Why: imagine you're delivering mail

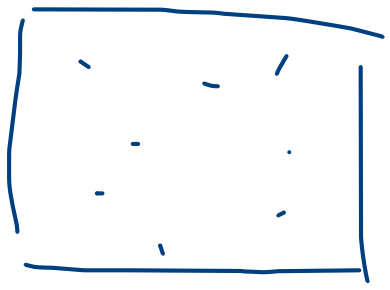
G = city road network V = intersections E = streets

Unlike the TSP, this is a polynomial problem (see notes) with the same idea as Christofides

Based on Eulerian cycles

IV / A bit of theory on TSP approximation

1972: The TSP is NP-hard

1956: 

Random points in the unit square

The length of the shortest tour satisfies

$$\frac{L_n}{\sqrt{n}} \xrightarrow{a.s.} \beta \quad (\text{TSP constant}) \sim 0.7$$

1976: It is NP-hard to approximate the non-metric TSP within any constant factor

1976: The metric TSP has a polynomial $\frac{3}{2}$-approx algorithm

1998: The Euclidean TSP has a poly. $(1+\varepsilon)$-approx alg for any ε

2015: It is NP-hard to approximate the metric TSP within a factor of $123/122$

2018: The metric asymmetric (directed) TSP has a poly 506-approx alg.

2020: The metric TSP has a poly$\left(\frac{3}{2} - 10^{-36}\right)$ approx. algo