# DM865 – Heuristics (Marco) & Approximation Algorithms (Lene)

---

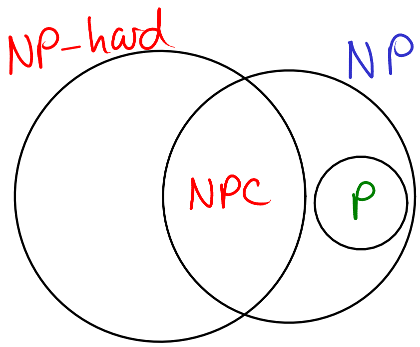
Combinatorial problems:

First 2 weeks → Traveling Salesman (TSP)
MAX SAT
Set Cover
Knapsack
Bin packing
Scheduling

decision version $\in$ NPC

**Polynomial algorithm:**

algo. with running time $O(n^c)$, for some constant $c$.




**P:** The set of decision problems that allow for a poly. algo.

**NP:** A problem belongs to NP, if solutions can be verified in poly. time.

If <u>any</u> NP-hard problem has a poly. algo., then <u>all</u> problems in NPC have poly. algo.s.

$\underbrace{\text{Optimal solutions}}_{(1)}$ $\underbrace{\text{in poly. time}}_{(2)}$ $\underbrace{\text{for all instances}}_{(3)}$

- Choose two! ((2) & (3))

An approximation algorithm comes with a performance guarantee:

**Def 1.1: $\alpha$-approximation algorithm**

An $\alpha$-approximation algorithm for an optimization problem $P$ is a poly. time algo. ALG s.t. for any instance $I$ of $P$,

- $\frac{ALG(I)}{OPT(I)} \leq \alpha$, if $P$ is a <u>min</u>imization problem
- $\frac{ALG(I)}{OPT(I)} \geq \alpha$, if $P$ is a <u>max</u>imization problem

Thus, for max. problems, $0 \leq \alpha \leq 1$, and, for min. problems, $\alpha \geq 1$.

The approximation factor / approximation ratio is
- the smallest possible $\alpha$ (for min. problems)
- the largest possible $\alpha$ (for max. problems)

More precisely, the approx. factor $R$ is

$$R = \inf \left\{ \alpha \;\middle|\; \forall I : \frac{ALG(I)}{OPT(I)} \leq \alpha \right\} \text{ for min. problems}$$

$$R = \sup \left\{ \alpha \;\middle|\; \forall I : \frac{ALG(I)}{OPT(I)} \geq \alpha \right\} \text{ for max. problems}$$

# Section 2.4: TSP

## The Traveling Salesman Problem (TSP)

**Input:** Weighted complete graph $G$

$$c_{ij} = c_{ji}, \qquad i,j \in V$$
$$c_{ii} = 0, \qquad i \in V$$
$$c_{ij} \geq 0, \qquad i,j \in V$$

**Output:** $\underbrace{\text{Hamiltonian cycle}}_{\text{Cycle visiting each vertex exactly once.}}$ of min. total weight

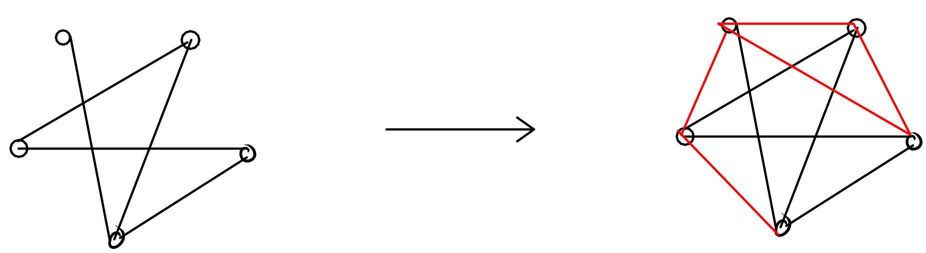
**Decision version of TSP:**

Does there exist a tour of cost $\leq X$ ?

**Decision version of TSP is NP-hard**

Reduction from Hamilton Cycle:

Ham. cycle $\qquad\qquad\qquad$ TSP



$c_{ij} = 1$

$c_{ij} = 2$

$\exists$ ham. cycle $\iff$ $\exists$ tour of cost $n$

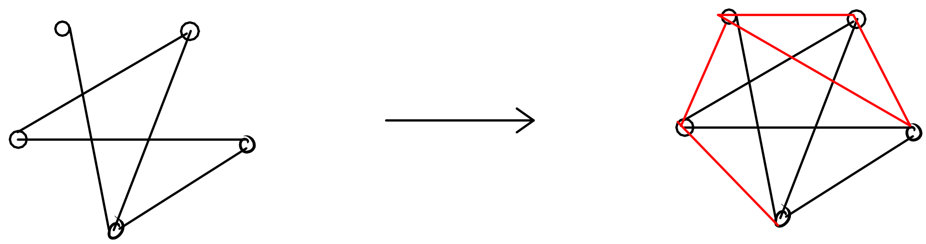
Even worse, no approximation guarantee possible:

**Theorem 2.9**

$\forall \alpha > 1$, $\nexists$ $\alpha$-approx alg. for TSP

__Proof__: Reduction from Hamilton Cycle:

Ham. cycle $\qquad\qquad\qquad\qquad$ TSP



$c_{ij} = 1$

$c_{ij} = \alpha n + 1$

$\exists$ ham. cycle $\iff$ $\exists$ tour of cost $n$

$\iff$ $\alpha$-approx. alg. gives tour of cost $\leq \alpha n$

$\iff$ $\alpha$-approx. alg. returns a tour with no red edges, i.e., a tour of cost $n$.

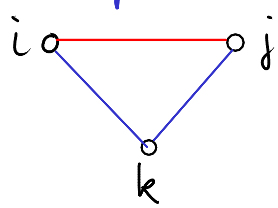
$\square$

Note: The proof does not require $\alpha$ to be a constant. In fact, it could be $2^n$, or __any__ function computable in poly. time.

Metric TSP:

The edge weights satisfy the triangle inequality:

$$c_{ij} \leq c_{ik} + c_{kj}, \quad \text{for all } i,j,k \in V$$




For metric TSP, the proof of Thm 2.9 does not work (the max. possible cost of the red edges would be 2).

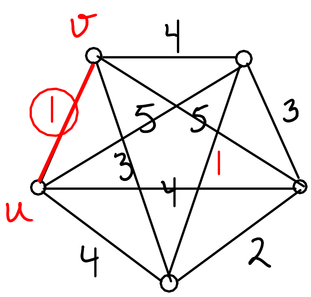
For the metric TSP problem, we will consider three algorithms:

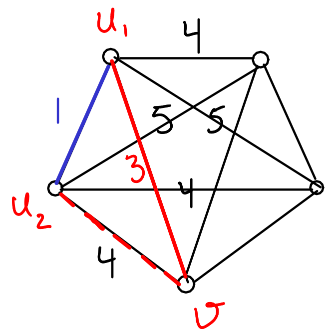
- The Nearest Addition algorithm — 2-approx.
- The Double Tree algorithm — 2-approx.
- Christofide's Algorithm — 3/2-approx

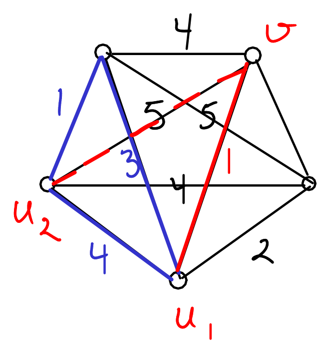
## Nearest Addition (NA)

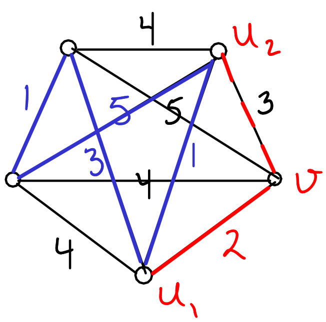
$u, v \leftarrow$ two nearest neighbors in $V$
Tour $\leftarrow \langle u, v, u \rangle$
For $i \leftarrow 1$ to $n-2$
    $v \leftarrow$ nearest neighbour of Tour
    $u_1 \leftarrow$ nearest neighbor of $v$ in Tour
    $u_2 \leftarrow$ $u_1$'s successor in Tour
    Add $v$ to Tour between $u_1$ and $u_2$

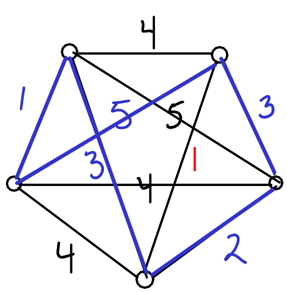
Ex:



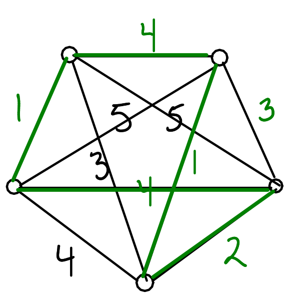








$$C_{NA} = 1 + 3 + 2 + 3 + 5 = 14$$

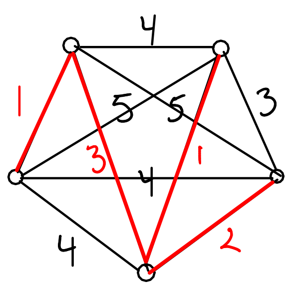


$$C_{OPT} \leq 1 + 4 + 1 + 2 + 4 = 12$$

Nearest Neighbor is a 2-approx. alg.:
We will prove that

$$\boxed{\begin{array}{ll} (1) & C_{NA} \leq 2 \cdot c(MST) \\ (2) & c(MST) \leq C_{OPT} \end{array}}$$ (Lemma 2.10)

(1): The solid red edges are exactly those chosen by Prim's Algorithm:

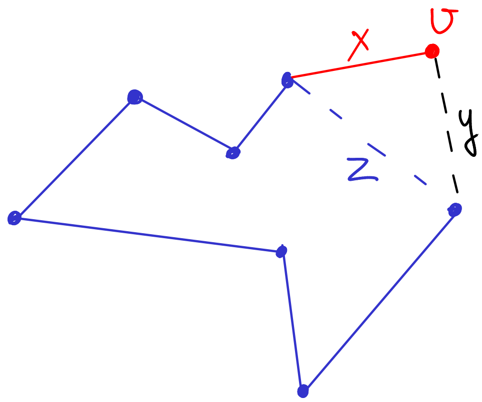



$C = 1 + 3 + 1 + 2 = 7$

Thus, the total cost $C$ of these edges is that of a minimum spanning tree:

$$\boxed{C = c(MST)}$$

Adding a new vertex v to the tour, we add two edges and delete one:




By the $\Delta$-ineq., $y \leq x + z$

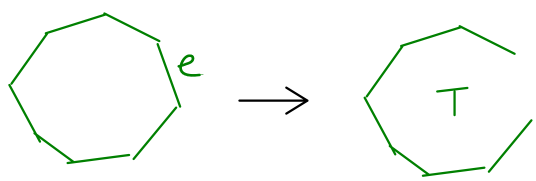
Hence, adding $v$ costs

$$x + y - z \leq x + (x+z) - z = 2x$$

Thus,

$$\boxed{C_{NA} \leq 2C = 2c(MST)}$$

(2): Deleting any edge from a tour, we get a spanning tree:




For any spanning tree $T$ obtained by deleting an edge $e$ from an optimal tour,

$$\boxed{C_{OPT}} \geq c(T), \quad \text{since } w(e) \geq 0$$

$$\geq \boxed{c(MST)}$$

Now, (1) & (2) $\Longrightarrow$ $C_{NA} \leq 2\,c(MST) \leq 2\,C_{OPT}$

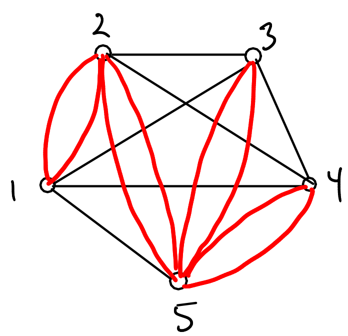
This proves:

**Theorem 2.11**

**Nearest Addition is a 2-approx. alg.**

# Double Tree algorithm

Noting that NA adds the edges of a MST one by one, we could also make a MST T and traverse T, making short cuts whenever we would otherwise visit a node for the second time:

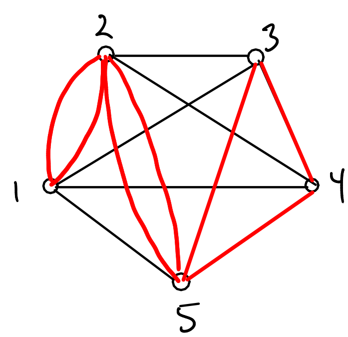


By the triangle inequality, this distance is no longer

$\langle 1, 2, 5, 3, \cancel{5}, 4, 5, 2, 1 \rangle$

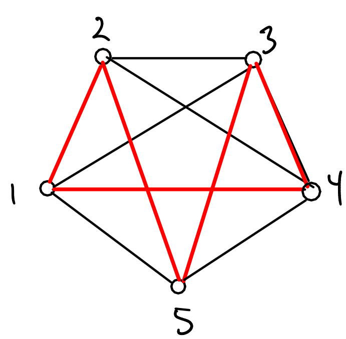
than this total distance

$\downarrow$ shortcut 5



$\langle 1, 2, 5, 3, 4, 5, 2, 1 \rangle$

$\downarrow$ shortcut 2 and 5 (using $\Delta$-ineq. twice)



$\langle 1, 2, 5, 3, 4, 1 \rangle$

An Euler tour is a traversal of a graph that traverses each edge exactly once.

A graph that has an Euler tour is called eulerian.

A graph is eulerian if and only if all vertices have even degree.

Constructive proof of „if" in exercises for Thursday.

**Double Tree Algorithm (DT)**

$T \leftarrow$ MST

$DT \leftarrow T$ with all edges doubled

Etour $\leftarrow$ Euler tour in DT

Tour $\leftarrow$ vertices in order of first appearance in ETour

Same analysis as for NA:

$$c_{DT} \leq 2\, c(MST) \leq 2 \cdot c_{OPT}$$

Hence:

**Theorem 2.12**

Double Tree is a 2-approx. alg