Traveling Salesman Problem

Definition 1

Let $G = (V, E)$ be a complete, weighted, directed graph with $|V| = n$, and let $c : E \to \mathbb{R}$ be the cost function of G.

(a) The Asymmetric Traveling Salesman Problem (ATSP) is the problem of finding a Hamiltonian cycle $(v_1, v_2, \ldots, v_n, v_1)$ with minimum cost $c(v_n, v_1) + \sum_{i=1}^{n-1} c(v_i, v_{i+1})$.

(b) The Symmetric Traveling Salesman Problem (STSP) is the ATSP for the special case, that for all edges it holds that the cost of the edge and its opposite edge is equal, i.e., the graph can be assumed to be undirected.

Nearest Neighbor Heuristic for the ATSP/STSP

Algorithm 1: Nearest Neighbor Heuristic for the ATSP/STSP (Rosenkrantz, Stearns, Lewis, 1977)

Let $G = (V, E)$ be a complete, weighted graph with $|V| = n$.

```
(1) Start with an arbitrary vertex $v_1 \in V$ as path $P$.
(2) FOR $i = 1, 2, \ldots, n-1$
(3)         Choose the vertex $w \in V \setminus P$ with the smallest cost
            from $v_i$.
(4)         Append $w$ as $v_{i+1}$ to the path $P$ after $v_i$.
(5) Connect the last vertex $v_n$ with the first vertex $v_1$,
    and receive a tour.
```

Remark 1

The big disadvantage of the heuristic is that the arcs (v_{n-1}, v_n) and (v_n, v_1) are not chosen,
but they are determined by the previous choices.
In the worst case, these arc costs can become very large.

$\rightsquigarrow$ An improvement would be to go in both directions, i.e., also backwards.

Algorithm 2: Extended Nearest Neighbor Heuristic for the ATSP/STSP

Let $G = (V, E)$ be a complete, weighted graph with $|V| = n$ and cost function $c : E \to \mathbb{R}$.

```
(1) Start with an arbitrary vertex $v_1 \in V$ as path $P$.
(2) FOR $i = 1, 2, \ldots, n-1$
(3)     Choose a vertex $w' \in V \setminus P$ which has smallest
        cost to $v_1$ and a vertex $w \in V \setminus P$ which has
        smallest cost from $v_i$.
(4)     IF $c(v_i, w) < c(w', v_1)$.
(5)       THEN Append $w$ as $v_{i+1}$ to the path $P$ after $v_i$.
(6)       ELSE Shift $(v_1, v_2, \ldots, v_i)$ to $(v_2, v_3, \ldots, v_{i+1})$.
(7)            Append $w'$ as $v_1$ to the path $P$ before $v_2$.
(8) Connect the last vertex $v_n$ with the first vertex $v_1$,
    and receive a tour.
```

Remark 2

Both variants of the Nearest Neighbor Heuristic have complexity $\mathcal{O}(n^2)$.

Remark 3

The repeated versions of both the Nearest Neighbor Heuristic and the Extended Nearest Neighbor Heuristic apply all steps n times, once for each possible starting vertex.
Then they compute the tour with minimum cost of all n trials.
Of course this results in a not worse tour quality.

However, these repetition variants of the heuristics have complexity $\mathcal{O}(n^3)$.

Greedy Heuristic for the ATSP

Contraction of an Arc

Let $G = (V, E)$ be a complete graph with $V = \{v_1, v_2, \dots, v_n\}$ and $n \geq 3$ and cost function $c : E \to \mathbb{R}$.
Let an arbitrary arc be given.
W.l.o.g., let (v_{n-1}, v_n) be this arc.
Construct a new complete graph $G' = (V', E')$ with $V = \{v'_1, v'_2, \dots, v'_{n-1}\}$, where $v'_i = v_i$ for $i = 1, 2, \dots, n-2$, $v'_{n-1} = (v_{n-1}, v_n)$ and with cost function $c' : E' \to \mathbb{R}$ defined by

$$c'(v'_i, v'_j) = \begin{cases} c(v_i, v_j) & \text{for } 1 \leq i \neq j \leq n-2 \\ c(v_i, v_{n-1}) & \text{for } 1 \leq i \leq n-2,\ j = n-1 \\ c(v_n, v_j) & \text{for } i = n-1,\ 1 \leq j \leq n-2 \end{cases}$$

Algorithm 3: Greedy Heuristic for the ATSP

(Glover, Gutin, Yeo, Zverovich, 2001)

Let $G = (V, E)$ be a complete, directed, weighted graph.

(1) Determine the arc $e = (u, v)$ with smallest cost in the graph.

(2) Contract the two incident vertices u, v of this arc to a new vertex (u, v).

(3) Receive a new graph $G' = (V', E')$ with one vertex less.

(4) Apply the steps (1) to (3), until only two vertices remain.

(5) Connect the two remaining vertices to a cycle.

(6) Replace all vertices by the corresponding contracted paths, and receive a tour.

Remark 4

The Greedy Heuristic has complexity $\mathcal{O}(n^2)$.

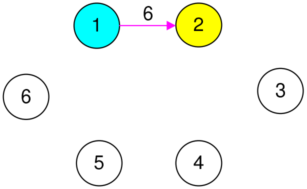
Remark 5

Consider the (Extended) Nearest Neighbor Heuristic and the Greedy Heuristic.
They give in particular for the Euclidean TSP tours of good quality.
However, there exists an instance for each n,
where all these three heuristics output the worst possible tour,
i.e., these heuristics have domination number 1.

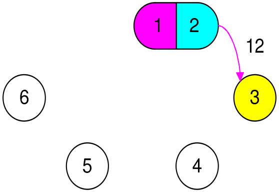
$\rightsquigarrow$ See the following examples.

Example 1: Consider a directed graph with 6 vertices and the following cost function.

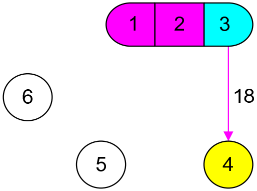
	1	2	3	4	5	6
1	–	6	7	7	7	7
2	35	–	12	13	13	13
3	35	13	–	18	19	19
4	35	13	19	–	24	25
5	35	13	19	25	–	30
6	216	13	19	25	31	–



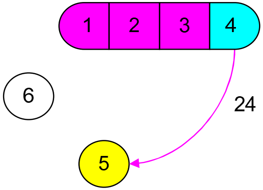
	(1,2)	3	4	5	6
(1,2)	–	12	13	13	13
3	35	–	18	19	19
4	35	19	–	24	25
5	35	19	25	–	30
6	216	19	25	31	–



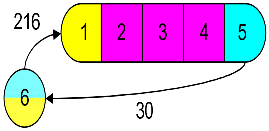
	(1,2,3)	4	5	6
(1,2,3)	–	18	19	19
4	35	–	24	25
5	35	25	–	30
6	216	25	31	–



	(1,2,3,4)	5	6
(1,2,3,4)	–	24	25
5	35	–	30
6	216	31	–



	(1,2,3,4,5)	6
(1,2,3,4,5)	–	30
6	216	–



Received tour $(1,2,3,4,5,6,1)$ with cost 306.

Nearest Neighor Algorithm (with start vertex 1):
$(1,2) \to (1,2,3) \to (1,2,3,4) \to (1,2,3,4,5)$
$\to (1,2,3,4,5,6) \to (1,2,3,4,5,6,1)$.

Extended Nearest Neighor Algorithm (with start vertex 1):
$(1,2) \to (1,2,3) \to (1,2,3,4) \to (1,2,3,4,5)$
$\to (1,2,3,4,5,6) \to (1,2,3,4,5,6,1)$.

$\Rightarrow$ Worst possible tour.

Example 2: Consider a directed graph with 6 vertices and the following cost function.

Greedy Algorithm:

	1	2	3	4	5	6
1	–	11	3	9	15	27
2	20	–	12	23	30	17
3	22	8	–	10	16	19
4	5	18	2	–	24	26
5	13	7	29	25	–	1
6	6	21	4	14	28	–

	1	2	3	4	(5,6)
1	–	11	3	9	15
2	20	–	12	23	30
3	22	8	–	10	16
4	5	18	2	–	24
(5,6)	6	21	4	14	–

	1	2	(4,3)	(5,6)
1	–	11	9	15
2	20	–	23	30
(4,3)	22	8	–	16
(5,6)	6	21	14	–

	2	(4,3)	(5,6,1)
2	–	23	30
(4,3)	8	–	16
(5,6,1)	11	9	–

	(4,3,2)	(5,6,1)
(4,3,2)	–	30
(5,6,1)	9	–

Received tour $(1, 4, 3, 2, 5, 6, 1)$ with cost 56.

Nearest Neighor Algorithm (with start vertex 1):

$(1,3)$
$\rightarrow (1,3,2)$
$\rightarrow (1,3,2,6)$
$\rightarrow (1,3,2,6,4)$
$\rightarrow (1,3,2,6,4,5) \rightarrow (1,3,2,6,4,5,1)$ with cost 79.

Extended Nearest Neighor Algorithm (with start vertex 1):

$(1,3)$
$\rightarrow (4,1,3)$
$\rightarrow (4,1,3,2)$
$\rightarrow (6,4,1,3,2)$
$\rightarrow (5,6,4,1,3,2)$
$\rightarrow (5,6,4,1,3,2,5)$
$\rightarrow (1,3,2,5,6,4,1)$ with cost 61.

Exercise 1:

(a) Create an ATSP instance with $n = 5$, where all costs in the cost matrix are different numbers and where the resulting ATSP tours for the Nearest-Neighbor Heuristic with starting vertex 1, the Extended Nearest Neighbor Heuristic with starting vertex 1 and the Greedy Heuristic are three different ATSP tours.

(b) Apply the Nearest-Neighbor Heuristic with starting vertex 1 to the instance from (a) and give the resulting ATSP tour and its cost.

(c) Apply the Extended Nearest-Neighbor Heuristic with starting vertex 1 to the instance from (a) and give the resulting ATSP tour and its cost.

(d) Apply the Greedy Heuristic with starting vertex 1 to the instance from (a) and give the resulting ATSP tour and its cost.

Exercise 2:
Implement the Nearest-Neighbor Heuristic with starting vertex 1, the Extended Nearest-Neighbor Heuristic with starting vertex 1, and the Greedy Heuristic.

The program should work with an input file, where the first line stands for the number of vertices and the last lines for the entries of the cost matrix (see attached example file for 5 vertices). However, your program should also work for larger instances, e.g., with 100 vertices, and it should output the final ATSP tour with costs.

Furthermore, add a file describing how your file(s) are organized, how your program is working and how it is compiled. You can also some comments in your programming code to improve understandability.

Remark: Preferably use as programming language C/C++, but I will also look at solutions in other programming languages.