The traveling salesman problem (TSP)

Given:  $n$ cities, costs $c(i,j)$ to travel from city $i$ to city $j$

Goal:  Find a cheapest round-trip route that visits each city exactly once and then returns to the starting city.

Formally: Find a permutation $p$ of $\{1, 2, ..., n\}$, such that

$$c(p(1), p(2)) + \cdots + c(p(n-1), p(n)) + c(p(n), p(1))$$

is minimized.
The traveling salesman problem (TSP)

A greedy algorithm for solving the TSP

Starting from city 1, each time go to the nearest city not visited yet. Once all cities have been visited, return to the starting city 1.
The traveling salesman problem (TSP)

Example
\[ c(i, i+1) = 1, \quad \text{for } i = 1, ..., n - 1 \]
\[ c(n, 1) = M \quad (\text{for some large number } M) \]
\[ c(i,j) = 0, \quad \text{otherwise} \]

Optimal tour:

Solution of the greedy algorithm:

\[ \frac{\text{GREEDY}}{\text{OPT}} = \frac{M-1+n}{M+2} \to \infty \quad \text{as } n \to \infty \]
The activity-selection problem

**Given:**
A set \( S = \{a_1, \ldots, a_n\} \) of \( n \) activities that wish to use a resource, e.g. a lecture hall.

**activity** \( a_i \): start time \( s_i \), finish time \( f_i \) \( \quad a_i = [s_i, f_i) \)

Activities \( a_i \) and \( a_j \) are compatible if

\[ [s_i, f_i) \cap [s_j, f_j) = \emptyset \]

**Goal:**
Select a maximum-size subset of mutually compatible activities.

**Assumption:**
Activities are sorted in non-decreasing order of finish time:

\[ f_1 \leq f_2 \leq f_3 \leq \ldots \leq f_n \]
**Greedy strategy for solving the activity-selection problem:**

Always choose the activity with the *earliest finish time* that is *compatible* with all previously selected activities!

In particular, the activity chosen first is the one with the earliest finish time.

**Theorem**

The greedy strategy for selecting activities yields an optimal solution to the activity-selection problem.
The activity-selection problem

Winter term 11/12
Activity-selection

**Algorithm**  Greedy-Activity-Selector

**Input:**  \( n \) activities represented by intervals \([s_i, f_i)\), \(1 \leq i \leq n\), where \( f_i \leq f_{i+1} \)

**Output:**  a maximum-size set of mutually compatible activities

1.  \( A_1 = \{a_1\} \)
2.  \( \text{last} = 1 \)  /* last indexes the activity added most recently */
3.  \( \text{for } i = 2 \text{ to } n \text{ do} \)
4.    \( \text{if } s_i < f_{\text{last}} \)
5.      \( A_i = A_{i-1} \)
6.    \( \text{else} \)  /* \( s_i \geq f_{\text{last}} \) */
7.      \( A_i = A_{i-1} \cup \{a_i\} \)
8.    \( \text{last} = i \)
9.  return \( A_n \)

Running time:  \( O(n) \)

provided that activities are sorted \( f_1 \leq \ldots \leq f_n \).
Optimality of the greedy algorithm

**Theorem**
The greedy algorithm yields an optimal solution.

**Proof**
Show that for all $1 \leq i \leq n$ the following holds:

There exists an optimal solution $A^*$ with

$$\{ A^* \cap \{a_1, \ldots, a_i\} = A_i \}$$

Let $A^* \subseteq \{a_1, \ldots, a_n\}$ be some optimal solution, $A^* = \{a_{i_1}, \ldots, a_{i_k}\}$

Winter term 11/12
Optimality of the greedy algorithm

\( i - 1 \rightarrow i: \)

Let \( A^* \subseteq \{a_1, \ldots, a_n\} \) be some optimal solution with \( A^* \cap \{a_1, \ldots, a_{i-1}\} = A_{i-1} \).

Consider \( R = A^* \setminus A_{i-1} \).

**Observation:**

\( R \) is an optimal solution to the problem of finding a maximum-size set of activities in \( \{a_i, \ldots, a_n\} \) that are compatible with all activities in \( A_{i-1} \).
Optimality of the greedy algorithm

\[ A^* \cap \{a_1, \ldots, a_{i-1}\} = A_{i-1} \subseteq A^* \]

Case 1: \( s_i < f_{last} \)

\[ A_{i-1} = \underbrace{\ldots}_{a_{i-1}} a_{last} \]

\( a_i \) is not compatible with \( A_{i-1} \) \( \Rightarrow \) \( a_i \) is not in \( A_i \) \( \Rightarrow \) \( A_i = A_{i-1} \).

\( a_i \) is not contained in \( A^* \)

\[ \Rightarrow A^* \cap \{a_1, \ldots, a_i\} = A_{i-1} = A_i \]
Optimality of the greedy algorithm

**Case 2:** \( s_i \geq f_{last} \)

\[
A_{i-1} = a_{i-1} \cdots a_{last} \quad f_{last} \quad s_i \quad a_i
\]

\( a_i \) is compatible with \( A_{i-1} \) \( \Rightarrow \) \( A_i = A_{i-1} \cup \{ a_i \} \)

There is: \( R \subseteq \{ a_i, \ldots, a_n \} \)

\[
R = b_1 \quad b_2 \quad b_3 \quad \cdots \quad b_4
\]

\( B^* = A_{i-1} \cup (R \setminus \{ b_1 \}) \cup \{ a_i \} \) is optimal

\[
B^* \cap \{ a_1, \ldots, a_i \} = A_{i-1} \cup \{ a_i \} = A_i
\]
Greedy algorithms

**Greedy-choice property:**
A globally optimal solution can be attained by a series of locally optimal (greedy) choices.

**Optimal substructure property:**
An optimal solution to the problem contains optimal solutions to its subproblems.

→ After making a locally optimal choice a new problem, analogous to the original one, arises.