Algorithms Theory

10 – Greedy Algorithms

Prof. Dr. S. Albers

Greedy algorithms

1. Introductory remarks
2. Basic examples:
   - The coin-changing problem
   - The traveling salesman problem
3. The activity-selection problem

Greedy algorithms for optimization problems

In each step make the choice that looks best at the moment!

Depending on the problem, the outcome can be:

1. The computed solution is always optimal.
2. The computed solution may not be optimal, but it never differs much from the optimum.
3. The computed solution can be arbitrarily bad.

Basic examples: The coin-changing problem

Denominations of euro coins and banknotes (in €):

500, 200, 100, 50, 20, 10, 5, 2, 1

Observation

Any amount in € can be paid using coins and banknotes of these denominations.

Goal

Pay an amount n using the fewest number of coins and banknotes possible.

Greedy algorithm

> Repeatedly choose the maximum number of coins or banknotes of the largest feasible denomination until the desired sum n is obtained.

Example: $n = 487$

500 200 100 50 20 10 5 2 1

The coin-changing problem: formal description

Denominations of coins: $n_1, n_2, \ldots, n_k$

$n_1 > n_2 > \ldots > n_k$, and $n_k = 1$.

Greedy algorithm:

1. $w = n$

2. **for** $i = 1$ **to** k **do**

 # coins of denomination n_i **is** $m_i = \lfloor w / n_i \rfloor$

$w = w - m_i \, n_i$

Any amount can be paid!

Country 'Absurdia'

Three denominations:

$n_3 = 1, \quad n_2 > 1 \text{ arbitrary}, \quad n_1 = 2\, n_2 + 1$

Example: 41, 20, 1

Amount to pay: $n = 3\, n_2$ (e.g. $n = 60$)

Optimal method of payment:

Greedy method:

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

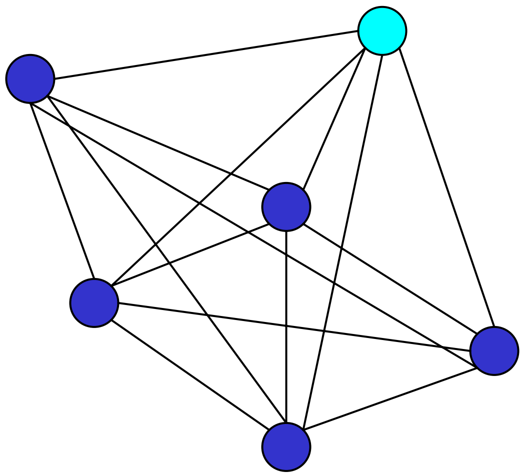
$$c(\ p(1), p(2)\ ) + \cdots + c(\ p(n-1), p(n)\ ) + c(\ p(n), p(1)\ )$$

is minimzed.

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)

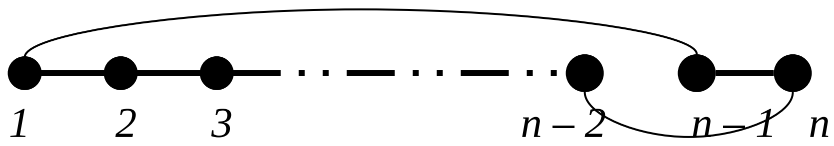
Example

$c(i, i{+}1) = 1,$ for $i = 1, \ldots, n - 1$

$c(n, 1) = M$ (for some large number M)

$c(i,j) = 2,$ otherwise

Optimal tour:



Solution of the greedy algorithm:

The activity-selection problem

Given:

A set $S = \{a_1,...,a_n\}$ 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

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 order of non-decreasing of finish time:

$$f_1 \leq f_2 \leq f_3 \leq \dots \leq f_n$$

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!

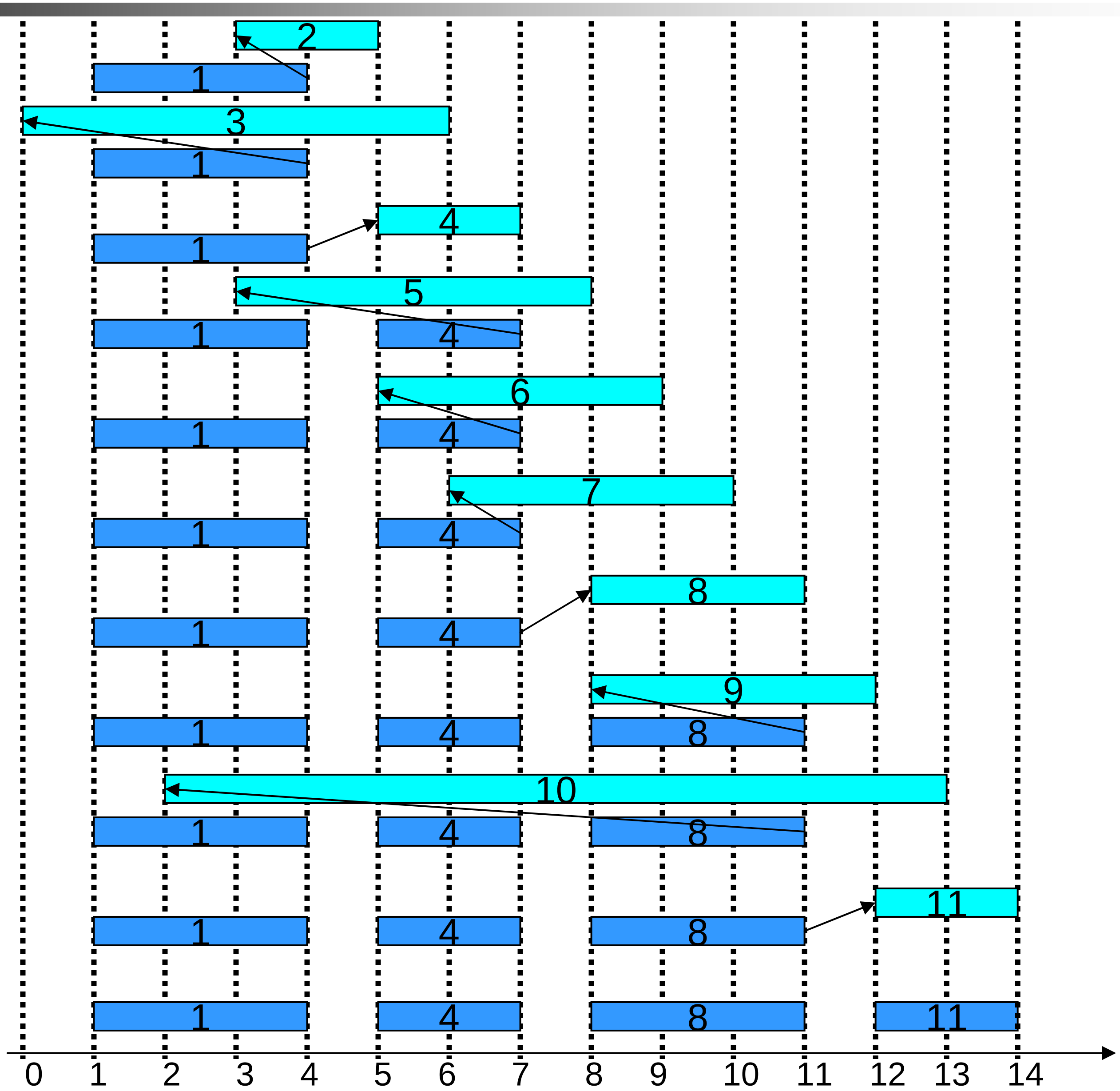
In particular, the activity chosen first is the one with the earliest finish time.

Theorem

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

The activity-selection problem

s_i	f_i
1	4
3	5
0	6
5	7
3	8
5	9
6	10
8	11
8	12
2	13
12	14



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

$A_1 = \{a_1\}$
$last = 1$ /* $last$ indexes the activity added most recently */
for $i = 2$ **to** n **do**
  if $s_i < f_{last}$
    then $A_i = A_{i-1}$
    else /* $s_i \geq f_{last}$ */
        $A_i = A_{i-1} \cup \{a_i\}$
        $last = i$
return A_n

Running time: $O(n)$

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:

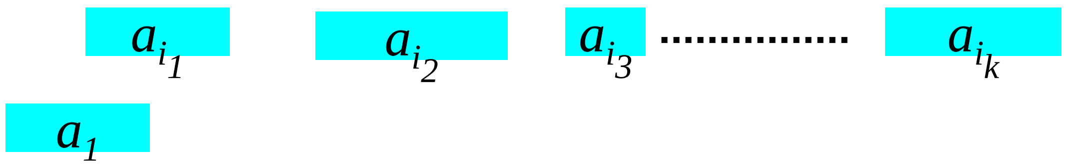
There exists an optimal solution A* with

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

$i = 1$:

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

$A^* =$



Optimality of the greedy algorithm

i - $1 \rightarrow i$:

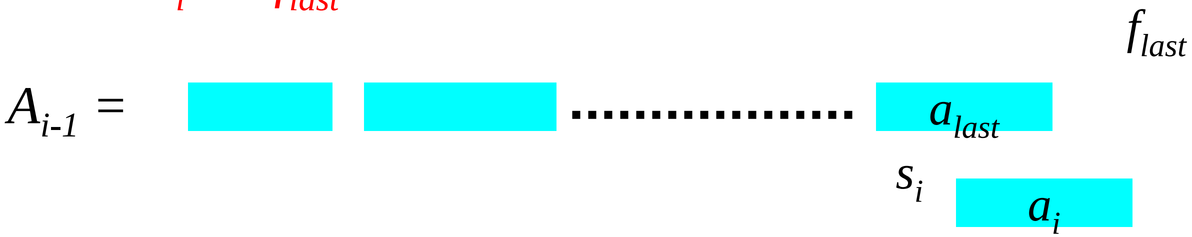
Let $A^* \subseteq \{a_1,\ldots,a_n\}$ be some optimal solution with $A^* \cap \{a_1,\ldots,a_{i-1}\} = A_{i-1}$.
Consider $R = A^* \setminus A_{i-1}$.

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

Case 1: $s_i < f_{last}$



a_i is not compatible with A_{i-1}

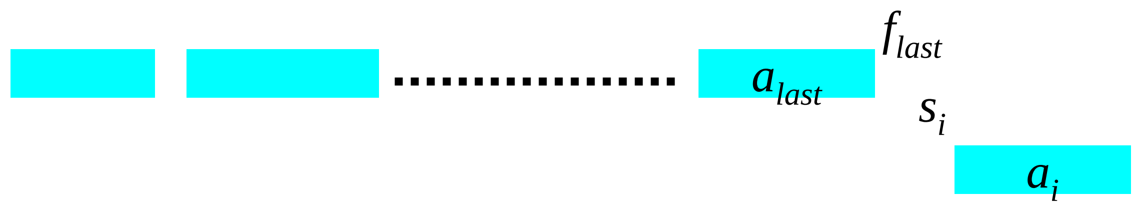
a_i is not contained in A^*

$$A^* \cap \{a_1,\ldots,a_i\} = A_{i-1} = A_i$$

Optimality of the greedy algorithm

Case 2: $s_i \geq f_{last}$

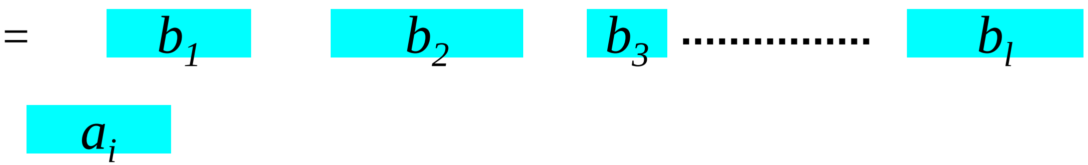
$A_{i-1} =$



a_i is compatible with A_{i-1}

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

$R =$



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