ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Algorithm Theory

9 Greedy Algorithms

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08

Greedy Algorithms

1. **Introduction**
2. **Basic examples**
   - Coin changing
   - Traveling salesman
3. **Activity selection problem**

Greedy Algorithms for Optimization Problems

- **In each step make the choice looking best!**
- **Possible results**
  - The computed solution is in fact optimal
  - The computed solution may not be optimal, but it never differs much from the optimum
  - The computed solution can be arbitrarily bad

Basic Example: Coin Changing

- **Denominations of Euro coins and bank notes**
  - 500, 200, 100, 50, 20, 10, 5, 2, 1
- **Observation**
  - Any amount in Euro can be paid using coins and banknotes
- **Goal**
  - Pay an amount *n* using the smallest number of coins and banknotes

Basic Example: Coin Changing

- **Greedy algorithm**
  - Repeatedly choose the maximum number of coins or banknotes of the largest possible value until the desired sum *n* is obtained
- **Example: n= 487**
  - 500 200 100 50 20 10 5 2 1
  - 487 = 200 (287) +
    200 (87) +
    50 (37) +
    20 (17) +
    10 (7) +
    5 + 2

General Coin Changing Problem

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**

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

   $w = w - m_i \cdot n_i$

Any amount can be paid!

Is the greedy algorithms always optimal?

Absurd Coin Systems

Three coins

$n_3 = 1,\ n_2 > 1$ *arbitrary*, $n_1 = 2\,n_2 + 1$

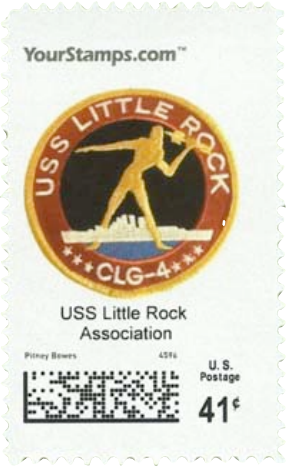
Example: 41, 20, 1

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

Optimal method of payment:

3 coins: 3 x n_2

Greedy method:



Traveling Salesman Problem (TSP)

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

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

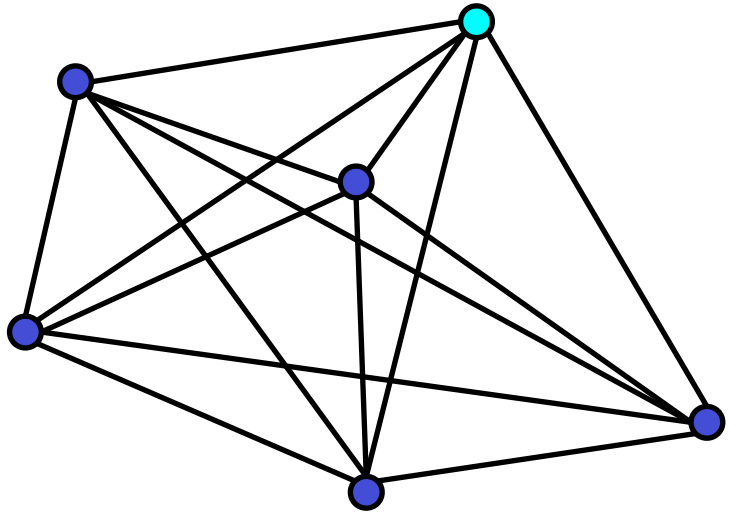
Formally: Find a permutation p of $\{1, 2, ..., n\}$, such that

$$c(p(1), p(2)) + ... + c(p(n-1), p(n)) + c(p(n), p(1))$$

is minimized.

Traveling Salesman Problem (TSP)

- **A greedy algorithms for solving 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.



Traveling Salesman Problem (TSP)

Example:

$c(i, i+1) = 1$ for $i = 1, ..., n - 1$

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

$c(i, j) = 2$ otherwise

Solution of the greedy algorithm: cost = n-1+M

Optimal tour: cost = n+O(1)

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
    $$[s_i, f_i) \cap [s_j, f_j) = \varnothing$$
- **Goal:**
  - Select a maximum size subset of mutually compatible activities

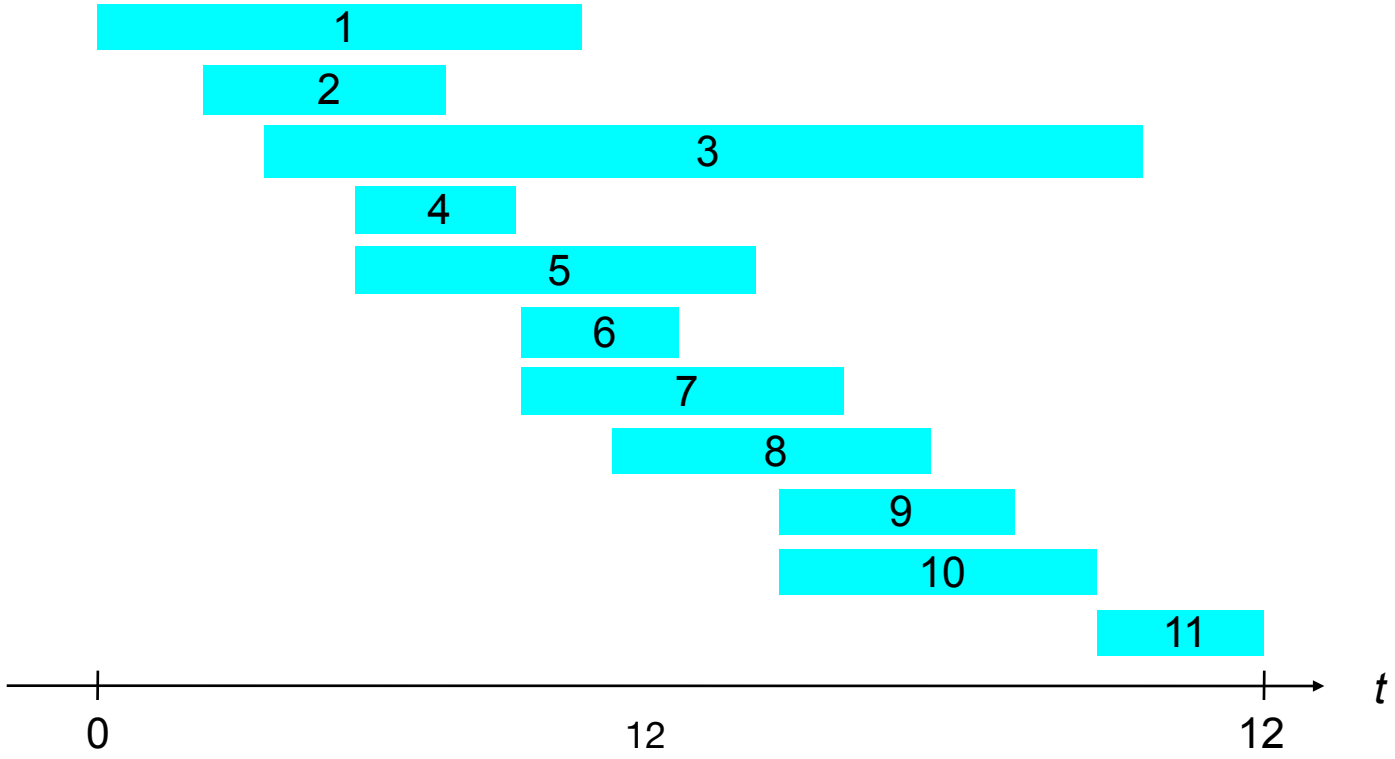
1st Greedy Strategy

Sort activities according to start times

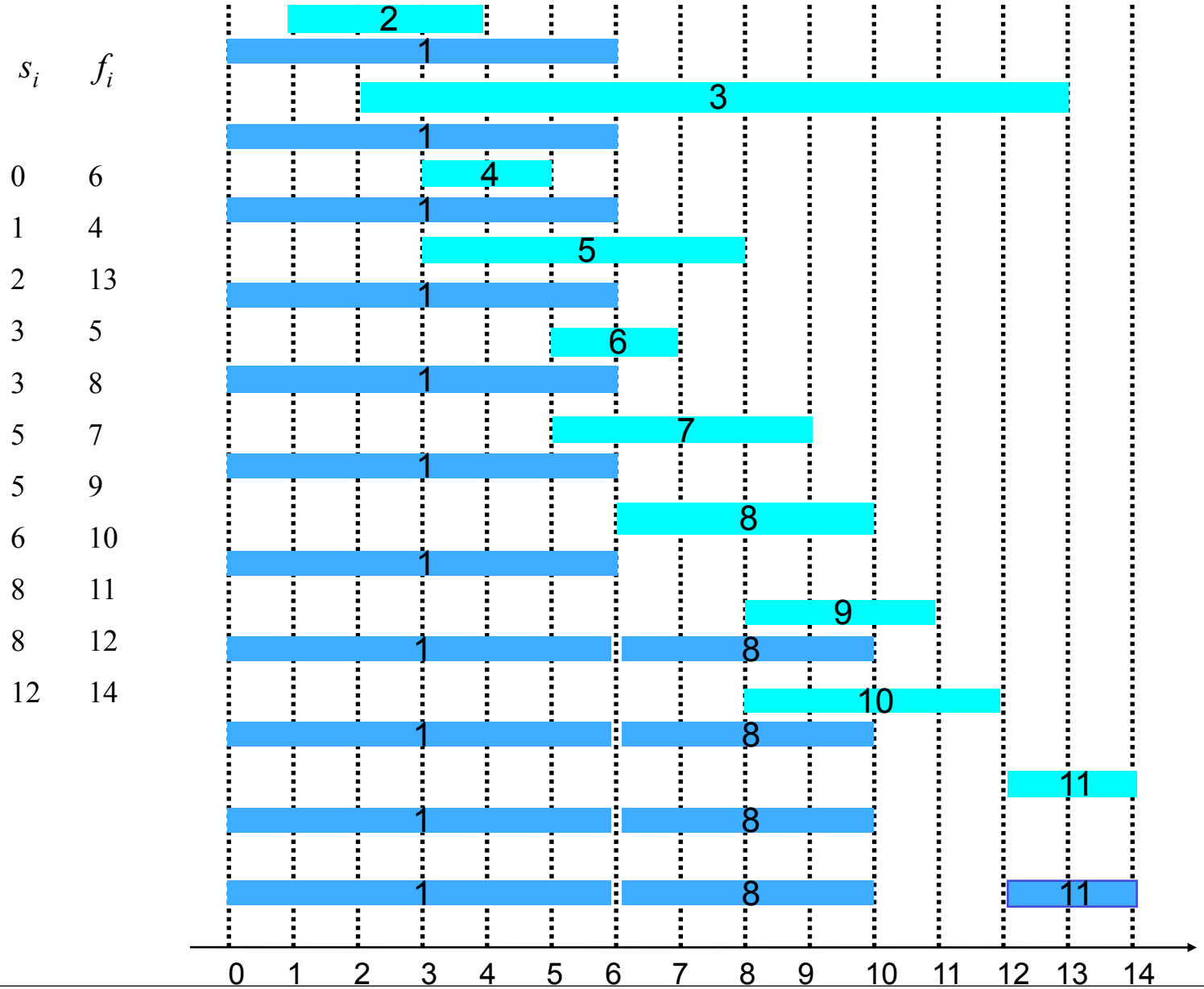
$s_1 \leq s_2 \leq s_3 \leq \ldots \leq s_n$

and choose the next activity which is compatible with the current set

Example:



Activity Selection Problem



1st Greedy Algorithm

- Sort activities according to start times
  - $s_1 \leq s_2 \leq s_3 \leq \ldots \leq s_n$
- and choose the next activity which is compatible with the current set
- **Problem:**
  - Algorithm does not yield the optimal solution

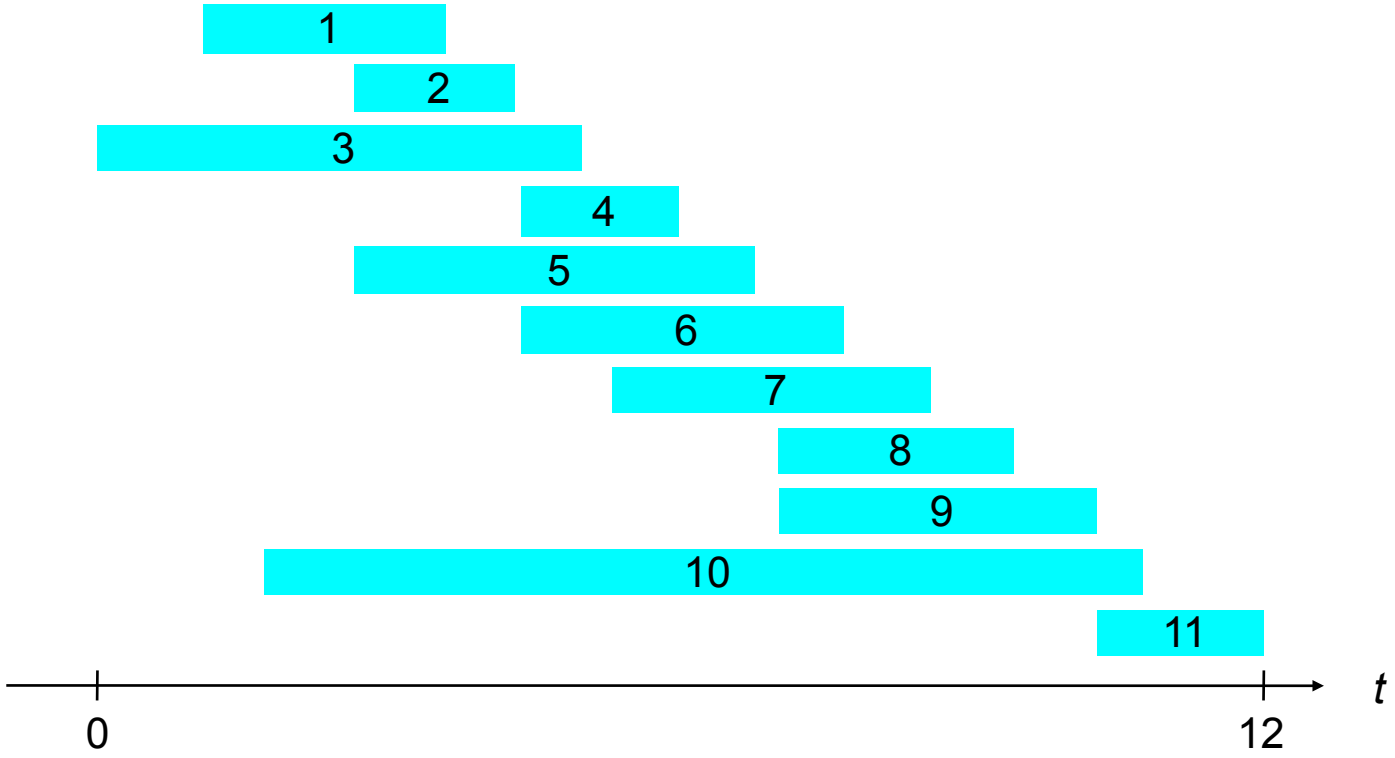
2nd Greedy Algorithms for Activity Selection

- **Assumption:**
  - Activities are sorted of non-decreasing finish times

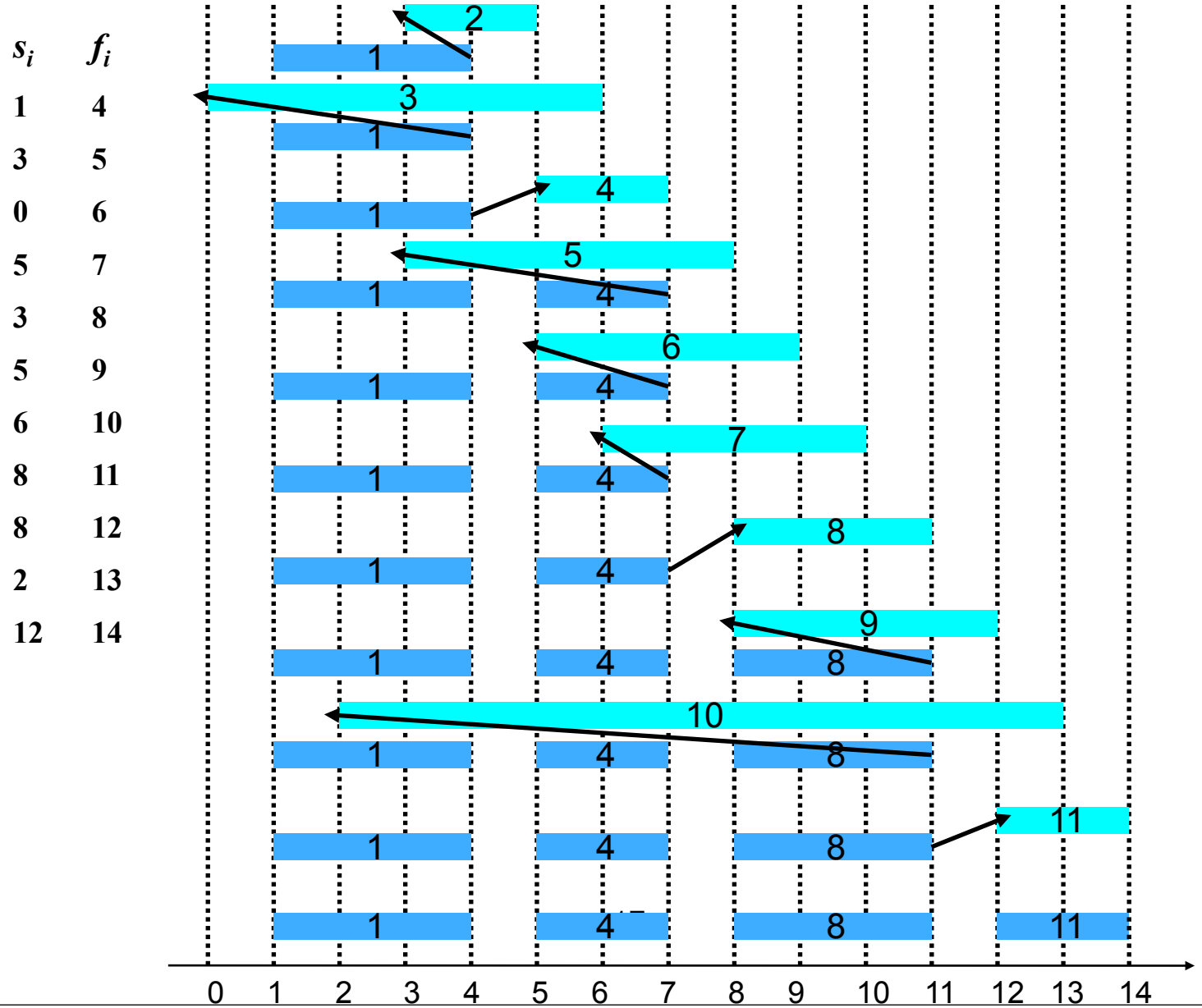
    $$f_1 \leq f_2 \leq f_3 \leq \ldots \leq f_n$$
- **Always choose the activity with the earliest finish time that is compatible with all previously selected activities!**
  - In particular the activity chosen first is the one with the earliest finish time.
- **Theorem**
  - The greedy strategy for selecting activities yields an optimal solution to the activity selection problem.

2nd Greedy Algorithms for Activity Selection

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



2nd Greedy Algorithm



Activity Selection

Algorithm *Greedy-Activity-Selector*

Input: n activity intervals $[s_i, f_i)$, $1 \le i \le n$ with $f_i \le f_{i+1}$

Output: a maximum size set of mutually compatible activities

```
1  A_1 = {a_1}
2  last = 1        /* last indexes the activity added most recently */
3  for i = 2 to n do
4    if s_i < f_last
5      then A_i = A_{i-1}
6      else  /* s_i ≥ f_last */
7            A_i = A_{i-1} ∪ {a_i}
8            last = i
9  return A_n
```

Runtime: $O(n)$

Optimality of the Greedy Algorithm

Theorem

The greedy algorithm yields an optimal solution.

Proof Show that for all $1 \le i \le n$ there exists an optimal solution A^* with $A^* \cap \{a_1, \ldots, a_i\} = A_i$

i = 1:

Case 1: a_1 is in A^*. Then A_1 will contain a_1

Case 2: a_1 is not in A^* then let $A^* \subseteq \{a_1, \ldots, a_n\}$ be some optimal solution

$A^* = \{a_{i_1}, \ldots, a_{i_k}\}$ without a_1

$A^* =$ a_{i_1} a_{i_2} a_{i_3} ..... a_{i_k}

a_1

Then, there exists some other optimal set A'* containing a_1

Optimality of the Greedy Algorithm

$i-1 \rightarrow i$:

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

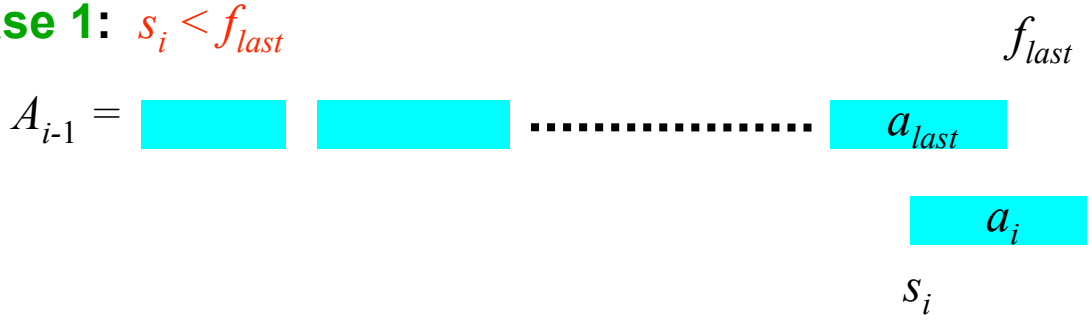
Consider $R = A^* \setminus A_{i-1}$

Observation:

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

Optimality of the Greedy Algorithm

Case 1: $s_i < f_{last}$



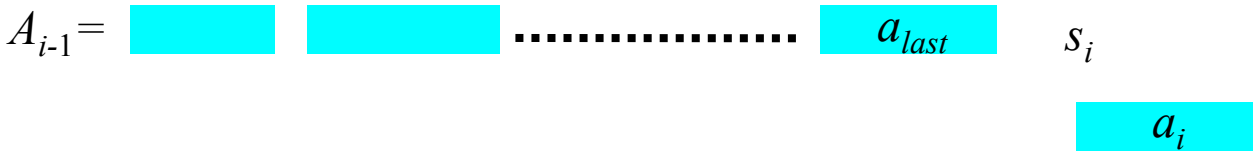
a_i is not compatible with $A_{i\text{-}1}$ ⇒ $A_i = A_{i\text{-}1}$

a_i is not in A^*

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

Optimality of the Greedy Algorithm

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

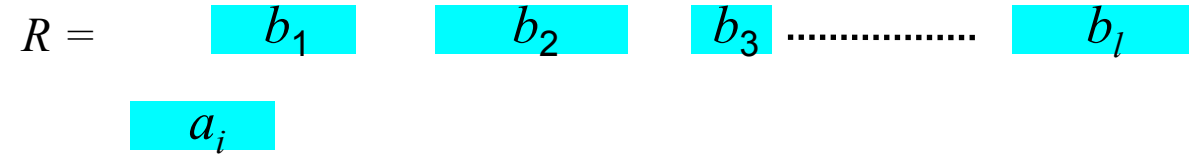


a_i is compatible with A_{i-1}

Case a: $a_i \in A^*$ (o.k.)

Case b: $a_i \notin A^*$

We have: $R \subseteq \{a_i, ..., a_n\}$



$A^*_{new} = A_{i-1} \cup (R \setminus \{b_1\}) \cup \{a_i\}$ is optimal

$A^*_{new} \cap \{a_1,...,a_i\} = A_{i-1} \cup \{a_i\} = A_i$

Greedy Algorithms

Greedy choice property

A globally optimal solution can be attained by a series of locally optimal (greedy) choices.

Optimal sub-structure property:

An optimal solution to the problem contains optimal solutions to its sub-problems.

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

ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Algorithm Theory

9 Greedy Algorithms

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08