Data Structures and Algorithms

Tutorial 5. Dynamic programming with examples

Today's topic is covered in detail in

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. **Introduction to Algorithms.** The MIT Press 2009.

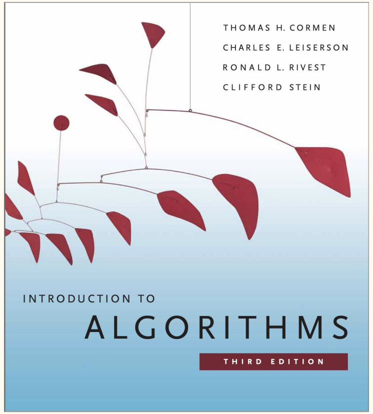
IV ***Advanced Design and Analysis Techniques***

Introduction ***357***

15 Dynamic Programming ***359***
- 15.1 Rod cutting *360*
- 15.2 Matrix-chain multiplication *370*
- 15.3 Elements of dynamic programming *378*
- 15.4 Longest common subsequence *390*
- 15.5 Optimal binary search trees *397*

16 Greedy Algorithms ***414***
- 16.1 An activity-selection problem *415*
- 16.2 Elements of the greedy strategy *423*
- 16.3 Huffman codes *428*
- ★ 16.4 Matroids and greedy methods *437*
- ★ 16.5 A task-scheduling problem as a matroid *443*

17 Amortized Analysis ***451***
- 17.1 Aggregate analysis *452*
- 17.2 The accounting method *456*
- 17.3 The potential method *459*
- 17.4 Dynamic tables *463*



Objectives

- Overview of dynamic programming
- DP example: the rod cutting problem
- DP example: weighted interval scheduling

Dynamic Programming in general

1. Can be thought of as optimization technique (in terms of performance) for divide-and-conquer algorithms where **subproblems overlap**:
    - DP makes sure that each subproblem is solved at most once

2. Is usually applied to optimization problems:
    - To such problems multiple solutions are possible.
    - Each solution has a certain value.
    - We are interested in finding
        - either an optimal solution (one with optimal value)
        - or just the the optimal value

Dynamic Programming: steps

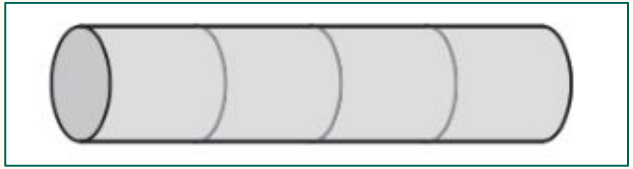
1. Characterize the structure of an optimal solution
2. Recursively define the value of an optimal solution
3. Compute the value of an optimal solution
4. Construct an optimal solution from computed information

The rod cutting problem

Problem 5.1. Serling Enterprises buys long steel rods and cuts them into shorter rods, which it then sells. Each cut is free. The management of Serling Enterprises wants to know the best way to cut up the rods.

Length	Price
1м	1 ₽
2м	5 ₽
3м	8 ₽
4м	9 ₽
5м	10 ₽
6м	17 ₽
7м	17 ₽
8м	20 ₽
9м	24 ₽
10м	30 ₽

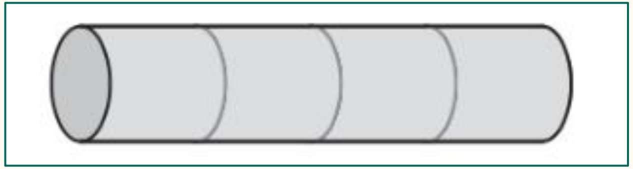
The rod cutting problem



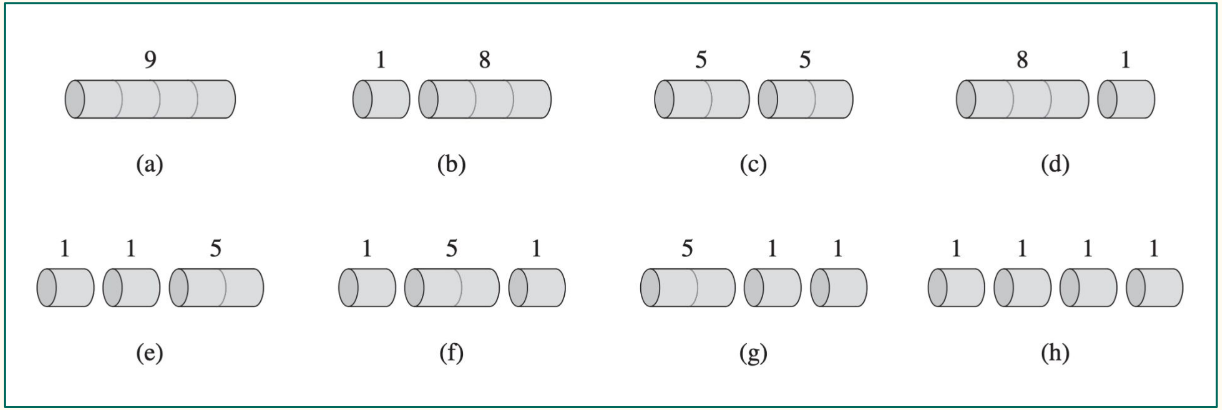
How many ways can we cut a rod of length 4?

Length	Price
1м	1 ₽
2м	5 ₽
3м	8 ₽
4м	9 ₽
5м	10 ₽
6м	17 ₽
7м	17 ₽
8м	20 ₽
9м	24 ₽
10м	30 ₽

The rod cutting problem

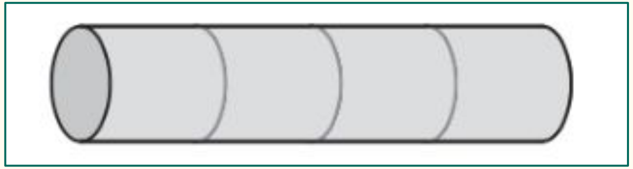


How many ways can we cut a rod of length 4?

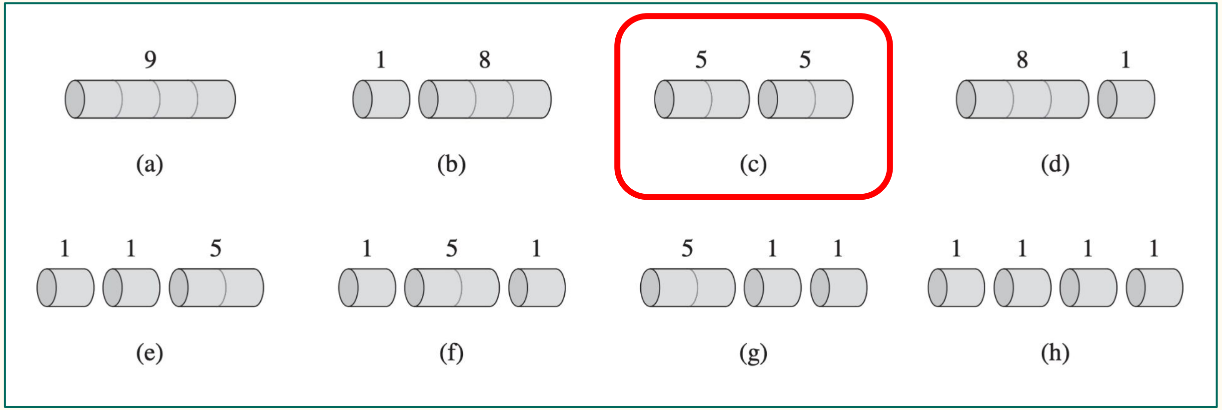


Length	Price
1м	1 ₽
2м	5 ₽
3м	8 ₽
4м	9 ₽
5м	10 ₽
6м	17 ₽
7м	17 ₽
8м	20 ₽
9м	24 ₽
10м	30 ₽

The rod cutting problem

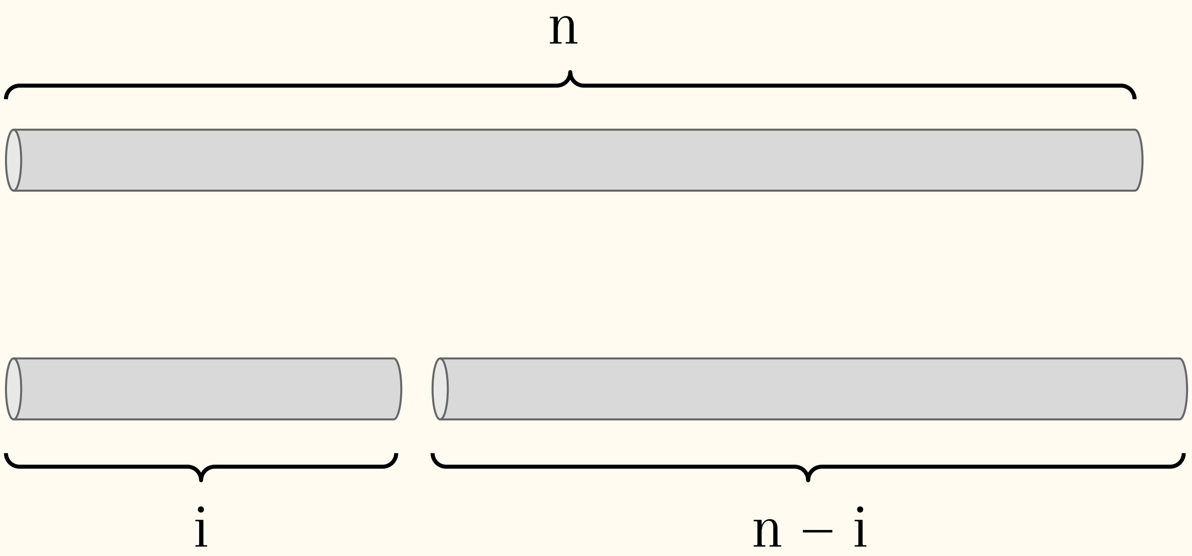


How many ways can we cut a rod of length 4?



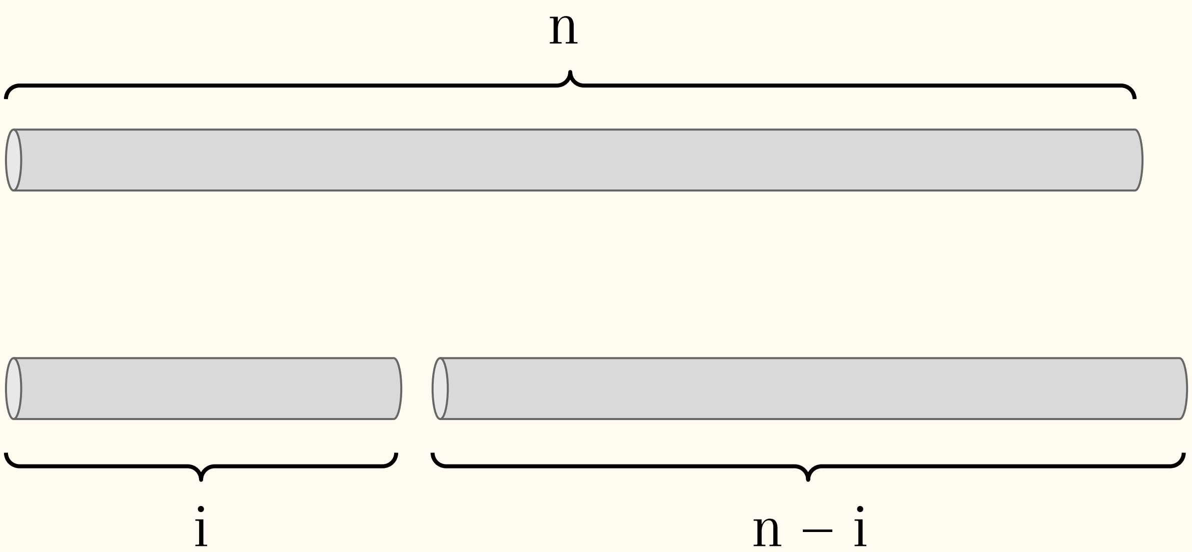
Length	Price
1м	1 ₽
2м	5 ₽
3м	8 ₽
4м	9 ₽
5м	10 ₽
6м	17 ₽
7м	17 ₽
8м	20 ₽
9м	24 ₽
10м	30 ₽

The rod cutting problem: recursive formulation



$$r_n = max(p_n, r_1 + r_{n-1}, r_2 + r_{n-2}, \ldots, r_{n-1} + r_1)$$

The rod cutting problem: recursive formulation 2



$$r_n = max_{1 \leq i \leq n}(p_i + r_{n-i})$$

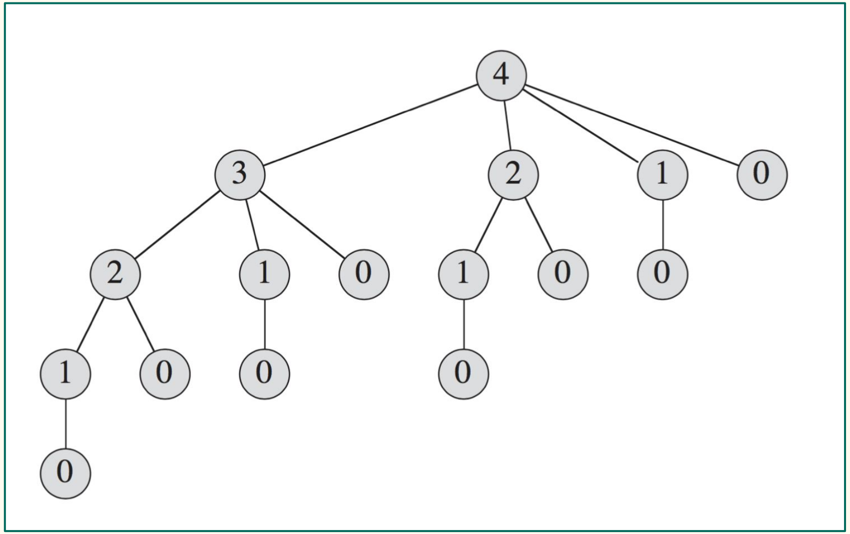
The rod cutting problem: recursive implementation

```
CUT-ROD(p, n)
if n == 0
    return 0
q = −∞
for i = 1 to n
    q = max(q, p[i] + CUT-ROD(p, n − i))
return q
```

$$r_n = max_{1 \leq i \leq n}(p_i + r_{n-i})$$

The rod cutting problem: recursive implementation

```
CUT-ROD(p, n)
if n == 0
      return 0
q = −∞
for i = 1 to n
      q = max(q, p[i] + CUT-ROD(p, n − i))
return q
```



$$r_n = max_{1 \leq i \leq n}(p_i + r_{n-i})$$

The rod cutting problem: DP with memoization

```
MEMOIZED-CUT-ROD(p, n)
let r[0..n] be a new array
for i = 0 to n
    r[i] = -∞
return MEMOIZED-CUT-ROD-AUX(p, n, r)
```

```
MEMOIZED-CUT-ROD-AUX(p, n, r)
if r[n] ≥ 0
    return r[n]
if n == 0
    q = 0
else q = -∞
    for i = 1 to n
        q = max(q, p[i] + MEMOIZED-CUT-ROD-AUX(p, n - i, r))
r[n] = q
return q
```

The rod cutting problem: bottom-up DP

```
BOTTOM-UP-CUT-ROD(p, n)
let r[0..n] be a new array
r[0] = 0
for j = 1 to n
    q = -∞
    for i = 1 to j
        q = max(q, p[i] + r[j - i])
    r[j] = q
return r[n]
```

Rod cutting exercises

Exercise 5.2. Consider a modification of the rod-cutting problem in which each cut incurs a fixed cost of **c**. The revenue associated with a solution is now the sum of the prices of the pieces minus the costs of making the cuts. Give a dynamic-programming algorithm to solve this modified problem.

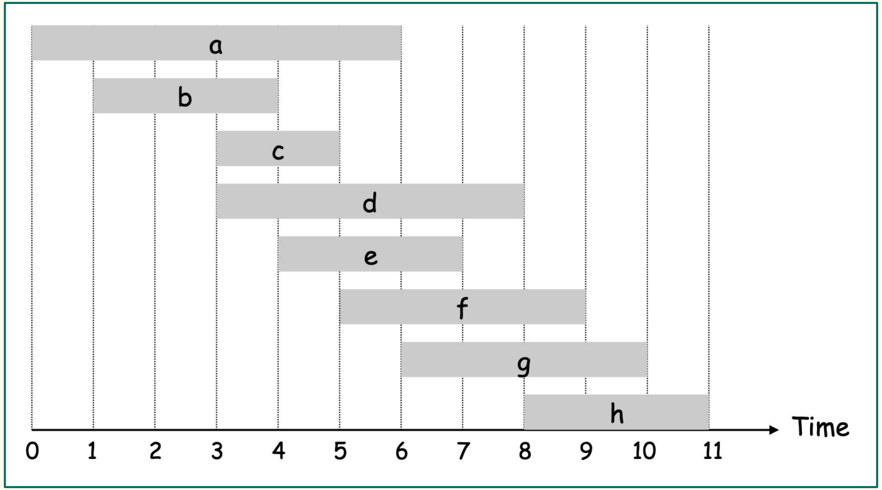
Exercise 5.3. Consider a greedy approach to solving the rod cutting problem. At each step cut a portion of the rod of length i, such that it has the most value per meter. Then cut recursively the rest of the rod. Show that such algorithm does not always find an optimal solution by providing a counterexample.

Attendance

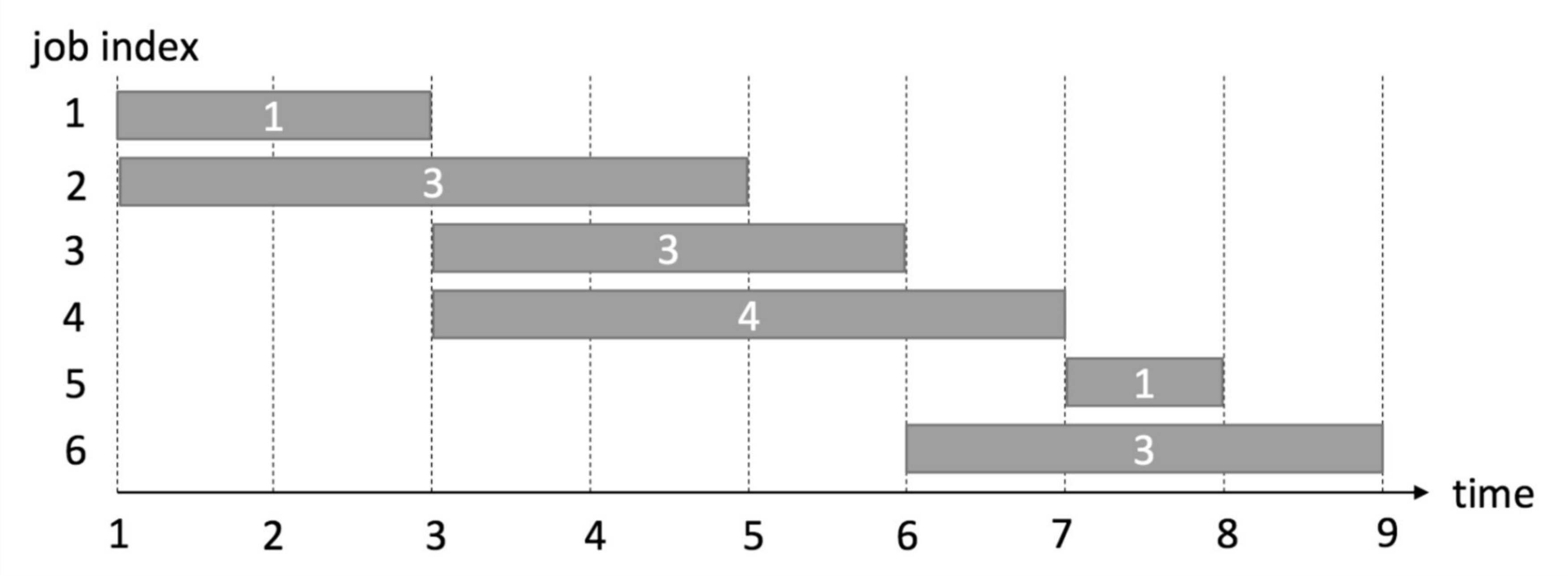
https://baam.duckdns.org

Weighted interval scheduling

Problem 5.3. Multiple people want to use a common resource. Each job (request for access) has a start time s_i and a finish time f_i, and an associated value v_i. Two jobs are compatible if they do not overlap. Find the subset of jobs, that are all mutually compatible, and such that the total value is maximized.



Weighted interval scheduling: practice



Weighted interval scheduling: recursive formulation

First, we sort all jobs by their finish time.

For each job (request) we either accept or deny it. If we accept it, then we can only consider jobs that finish before the accepted job. We denote by **p(n)** the largest index of a job that finishes before job **n**.

$$v_n = max(v_n + v_{p(n)}, v_{n-1})$$

Summary

- Dynamic Programming is an optimization technique that is usually applied to speed up divide-and-conquer algorithms for optimization problems.

- See more examples and exercises in Chapter 15 of Cormen et al.

Summary

- Dynamic Programming is an optimization technique that is usually applied to speed up divide-and-conquer algorithms for optimization problems.

- See more examples and exercises in Chapter 15 of Cormen et al.

See you next week!