

Algorithms Algorithms

Rod Cutting

Algorithms

School of Computing and Data Science

- 1/10 -

Frank Kreimendahl | kreimendahlf@wit.edu



Algorithms

Beyond craftsmanship lies invention, and it is here that lean, spare, fast programs are born. Almost always these are the result of strategic breakthroughs rather than tactical cleverness. Sometimes the strategic breakthrough will be a new algorithm, such as the Cooley-Tukey Fast Fourier Transform or the substitution of an $n \log n$ sort of an n^2 set of comparisons. ... Representation *is* the essence of programming.

-Fred Brooks, 1974 (lead on IBM/360, Turing Award)



Rod Cutting

Algorithms: Modern Version

Smart data structures and dumb code works a lot better than the other way around.

-Guy Steele, 2002 (inventor of Scheme)



Algorithms Algorithms Types of Algos

Rod Cutting

Types of Algorithms

- divide and conquer
- dynamic programming
- greedy
- backtracking
- reduction



Algorithms

Rod Cutting

Problem Optimal Value An Algorithm Solution Recove Properties

Rod Cutting

School of Computing and Data Science

Frank Kreimendahl | kreimendahlf@wit.edu



Problem

Optimal Value An Algorithm Solution Recovery Properties

Problem Description

Given table of profits p_i for each possible integer length *i*, find the best way to cut a rod of length *n*. Cuts are free, but must be integer lengths.

length <i>i</i>										
price p_i	1	5	8	9	14	15	17	20	24	30

 2^{n-1} possible solutions. Can we solve it faster?



Problem

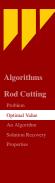
Optimal Value An Algorithm Solution Recovery Properties

Problem Description

Given table of profits p_i for each possible integer length *i*, find the best way to cut a rod of length *n*. Cuts are free, but must be integer lengths.

length <i>i</i>										
price p_i	1	5	8	9	14	15	17	20	24	30

 2^{n-1} possible solutions. Can we solve it faster?



Optimal Value

Step 1: write down value of optimal solution

$$best(n) = best profit achievable for length n$$
$$= \max_{1 \le first \le n} (p_{first} + best(n - first))$$
$$best(0) = 0$$

What is the complexity of the naive recursive algorithm? How can we make this efficient?



An Algorithm

Algorithms Rod Cutting Problem Optimal Value An Algorithm Solution Recovery Properties

Step 2: compute optimal value

$\operatorname{CUTROD}(p, n)$

- 1: best[0] $\leftarrow 0$
- 2: **for** len from 1 to *n* **do**
- 3: best[len] $\leftarrow \max_{1 \le first \le len} (p[first] + best[len first])$
- 4: **return** best[*n*]



Algorithms Rod Cutting Problem

Optimal Val

An Algorithm

Solution Recovery

roperties

Solution Recovery

CUTROD(p,n)

- 1: best[0] $\leftarrow 0$
- 2: **for** len from 1 to *n* **do**
- 3: best[len] $\leftarrow -\infty$
- 4: **for** first from 1 to len **do**
- 5: this \leftarrow p[first]+best[len-first]
- 6: **if** this > best[len] **then**
- 7: best[len]←this
- 8: cut[len]←first
- 9: $n_0 \leftarrow n$
- 10: **while** $n_0 > 0$ **do**
- 11: print $\operatorname{cut}[n_0]$
- 12: $n_0 \leftarrow n_0 \operatorname{cut}[n_0]$
- 13: **return** best[*n*]

Properties

- Algorithms Rod Cutting Problem Optimal Value An Algorithm Solution Recovery
- Properties

- optimal substructure: global optimum uses optimal solutions of subproblems
- ordering of subproblems: solve 'smallest' first, build larger solutions from smaller
- 'overlapping' subproblems: polynomial number of subproblems, used multiple times
- independent subproblems: optimal solution of one subproblem doesn't affect optimality of another