# Dynamische Programmierung

Richard Bellman 1957

Mächtiges Werkzeug zum Lösen interessanter (Optimierungs)-Probleme

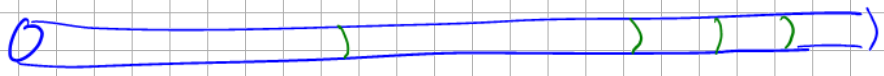
DP ≈ cleveres Brute-Force

DP ≈ Rekursion + Wiederverwendung

## Beispiel: Schneiden von Eisenstäben (Rod-Cutting)

gegeben:

Stab der Länge $n \in \mathbb{N}$



Preistabelle

| Länge $\ell$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... | $n$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Preis $v_\ell$ | 0 | 1 | 5 | 8 | 9 | 10 | 17 | 17 | 20 | ... | $v_n$ |

gesucht:

Bester Preis $r_n$ einer zerschnittenen Stange der Länge $n$

erlaubt: Längen $\ell_i \in \mathbb{N}$, $i = 1..k$

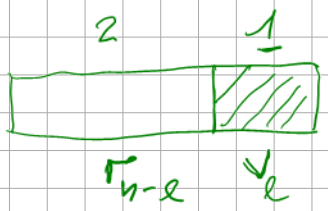
mit $\ell_1 + \ell_2 + \dots + \ell_k = n$

Wert $v_{\ell_1} + v_{\ell_2} + \dots + v_{\ell_k}$

| Länge $\ell$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... | n |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Preis $v_\ell$ | 0 | 1 | 5 | 8 | 9 | 10 | 17 | 17 | 20 | ... | $v_n$ |

$r_0 = 0$

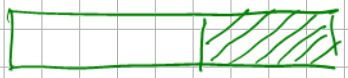
$r_1 = 1$




$r_2 = \max\{5, 2\}$

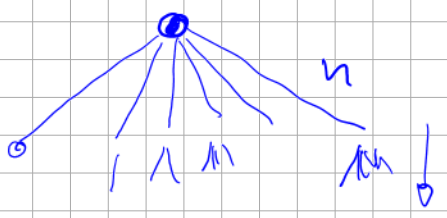
$r_3 = \max\{\underbrace{0+8}_{r_0+v_3}, \underbrace{1+5}_{r_1+v_2}, \underbrace{5+1}_{r_2+v_1}\}$

$r_4 = \max\{r_0+v_4, r_1+v_3, r_2+v_2, r_3+v_1\}$



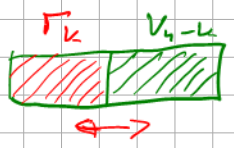

$r_k = \max\{r_i + v_{k-i}, 0 \le i < k\}$

## Rekursiv Algorithmus




```
RodCut(n):
    if n>0:
        return min{RodCut(i)+v_{n-i}, i=0..n-1}
    else:
        return 0
```

# Formulierung eines DP-Algorithmus




(0) Formulierung des Problems / Zielwertes

$r_n$ = max. Wert eines geschnittenen Stabes der Länge $n$

(1) Definiere Teilprobleme + deren Anzahl

$r_k$, $k \leq n$ $\quad\quad$ $\#TP = n$

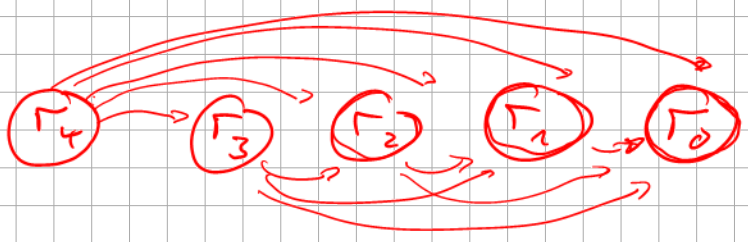
(3) Raten / Aufzählen pro Teilproblem

Länge des letzten Stückes $\quad\quad$ $\#$ Schritte : $n$

(4) Rekursion : Verbinde Teilprobleme

$$r_0 = 0$$
$$r_k = \max \{ r_i + v_{k-i} \;,\; 0 \leq i < k \}$$

(5) Abhängigkeiten → Tabelle / Memoisieren




(6) Lösung und Laufzeit

beste Welt in $r_n$

Laufzeit : $\#TP \cdot \frac{\#\text{ Schritte}}{\text{Teilproblem}}$ $\quad\quad$ $n \cdot n \in \boxed{\Theta(n^2)}$

# Mergesort