Lecture 07: Dynamic Programming - I
Cutting Rods

Swakkhar Shatabda

CSI 227: Algorithms, Summer 2014
Department of Computer Science and Engineering
United International University

Dynamic Programming

Four basic 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, typically in a bottom-up fashion.
4. Construct an optimal solution from computed information

Rod Cutting

Price of pieces of rods are as follows:

length i	1	2	3	4	5	6	7	8	9	10
price p_i	1	5	8	9	10	17	17	20	24	30

Now one has to cut the rod into number of pieces so that the selling price is maximized.

How many ways you can cut a rod of length n inches? 2^{n-1}

Example: 4 inches rod

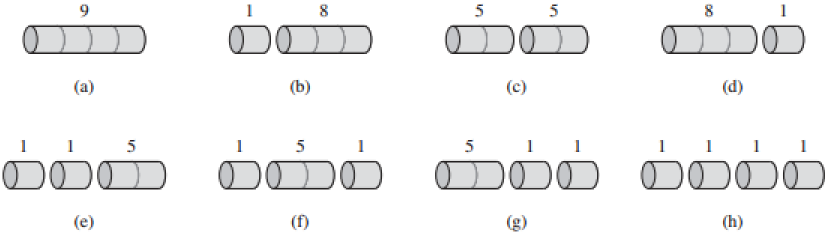


Figure 15.2 The 8 possible ways of cutting up a rod of length 4. Above each piece is the value of that piece, according to the sample price chart of Figure 15.1. The optimal strategy is part (c)—cutting the rod into two pieces of length 2—which has total value 10.

Different Lengths

length	price	cuts
1	1	no cuts
2	5	no cuts
3	8	no cuts
4	10	2+2
5	13	2+3
6	17	no cuts
7	18	1+6 or 2+2+3
8	22	2+6
9	25	3+6
10	30	no cuts

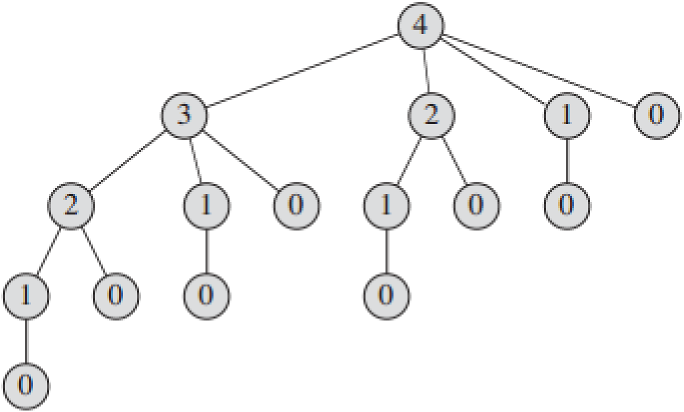
How to cut?

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

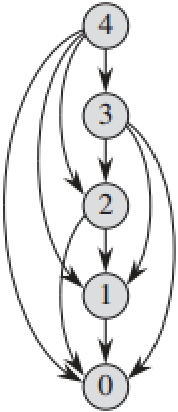
$$r_n = \max_{1 \le i \le n}(p_i + r_{n-i})$$

```
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
```

Recursive Tree



Sub Problem Graph



Memoization

```
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
```

Memoization

$\textsc{Memoized-Cut-Rod}(p[], n)$

let $r[0 \cdots n]$ be a new array
for $i = 0$ **to** n
    $r = -\infty$
return $\textsc{Memoized-Cut-Rod-Aux}(p, n, r)$

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]
```

Reconstructing Solutions

EXTENDED-BOTTOM-UP-CUT-ROD($p[], n$)

```
let r[0 ··· n] and s[0 ··· n] be new arrays
r[0] = 0
for j = 1 to n
    q = −∞
    for i = 1 to j
        if q < p[i] + r[j − i])
            q = p[i] + r[j − i])
            s[j] = i
    r[j] = q
return r and s
```

Print Solutions

```
PRINT-CUT-ROD-SOLUTIONS(p[], n)
(r, s) = EXTENDED-BOTTOM-UP-CUT-ROD(p, n)
while n > 0
    print s[n]
    n = n - s[n]
```

Reading

Chapter 15

Thats it!

Thank you