Rod cutting

Decide where to cut steel rods:

- Given a rod of length n inches and a table of prices p_i, i=1,2,…,n, find the maximum revenue r_n obtainable by cutting up the rod and selling the pieces
  - Rod lengths are integers
  - For i=1,2,…,n we know the price p_i of a rod of length i inches

Example

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

- For a rod of length 4: 2+2 is optimal ($p_2+p_2=10$)
- In general, can cut a rod of length n 2^{n-1} ways

- If optimal sol. cuts rod in k pieces then
  - optimal decomposition: $n=i_1+i_2+\ldots+i_k$
  - Revenue: $r_n=p_{i1}+p_{i2}+\ldots+p_{ik}$
- In general: $r_n=\max\{p_n,r_1+r_{n-1},r_2+r_{n-2},\ldots,r_{n-1}+r_1\}$
  - Initial cut of the rod: two pieces of size i and n-I
    - Revenue r_i and r_{n-i} from those two pieces
  - Need to consider all possible values of i
  - May get better revenue if we sell the rod uncut

A different view of the problem

- Decomposition in
  - A first, left-hand piece of length i
  - A right-hand reminder of length n-i
  - Only the reminder is further divided
  - Then
    - $r_n = \max\{p_i + r_{n-i}, 1 <= i <= n\}$
  - Thus, need solution to only one subproblem

Top-down implementation

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

- Time recurrence: $T(n)=1+T(1)+T(2)+\ldots+T(n-1)$
  - $T(n)=O(2^n)$

Dynamic Programming

- Optimality of subproblems is obvious

```
DP-CUT-ROD(p,n)
    let r[0..n], 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]
                s[j]=i; q= p[i]+r[j-i]
        r[j]=q
    return r and s
```

Retrieving an optimal solution

```
PRINT-CUT-ROD
    (r,s) = DP-CUT-ROD(p,n)
    while n>0
        print s[n]
        n=n-s[n]
```

Example:

i	0	1	2	3	4	5	6	7
r[i]	0	1	5	8	10	13	17	18
s[i]	0	1	2	3	2	2	6	1