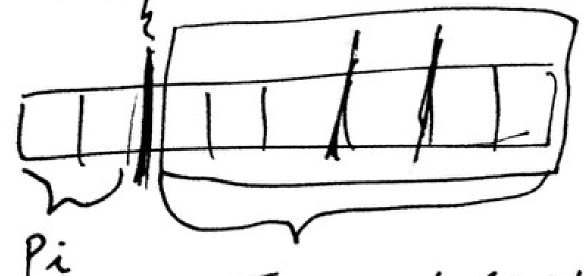
Rod cutting.

Input: a length n, and an array of prices $p_1, \dots, p_n$.

Output: Find a way to cut the rod to maximize your profit. Profit is the sum of the prices for the broken up rods.

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

best profit possible for an input of size n



I can still this rod!

r_{n-i}

- Optimal substructure property.
  - The solution to sub-problems used within an optimal sol is itself optimal.

Theorem: Rod-cutting exhibits the opt. substructure property. Specifically, in the optimal solution to a ~~p~~ rod of length n that contains a first cut at i, there is the opt. sol to the rod of length $n-i$.

Proof: - lets consider an opt sol. S.
- Suppose that a ~~sol~~ solution to a subproblem is not optimal.
- Create a new solution S' by cutting out the subopt. sol. to the subpr. and replace it by the opt sol to that subproblem.
- S' is now a better solution than S. This is a contradiction b/c S was optimal.

MEM-CUT-ROD-AX (p, n, r) table of vals that you've computed so far.

if $r(n) \geq 0$ then

$\Theta(1)$ return $r(n)$.

if $(n == 0)$ then

$\Theta(1)$ $q = 0$

else $q = -\infty$

$\max_{1 \leq i \leq n} p_i + r_{n-i}$

$\Theta(n)$ for $i = 1$ to n {

$q = \max(q, p(i) +$ MEM-CUT-ROD-AUX $(p, n-i, r)$

}

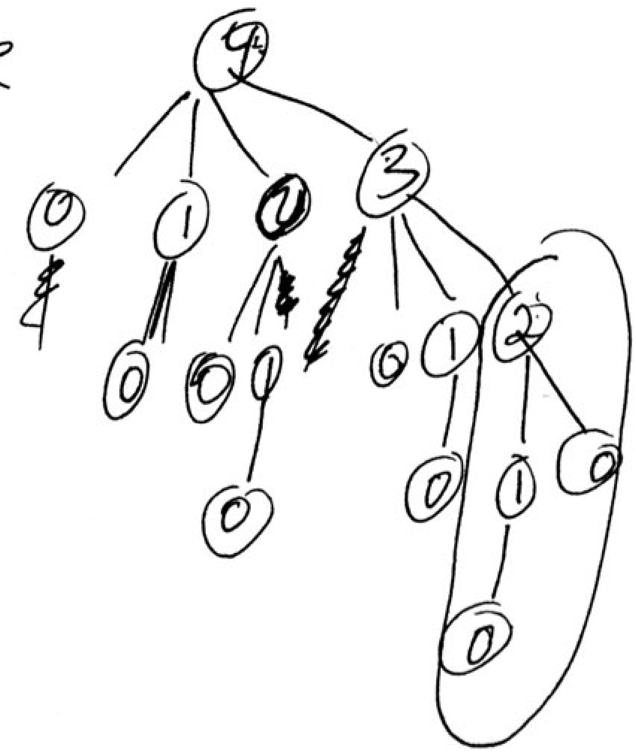
$r(n) = q$

return q.

Memoized Dynamic Programming sol to Rod-cutting.

Run time:

Basic Recursive alg!

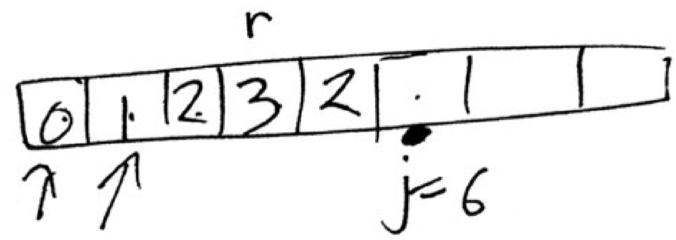


- Solves each subproblem just once!
- There are n subproblems.
- For each subproblem, it can be solved in $\underline{\Theta(n)}$ time

---

$O(n^2)$ worst case runtime of memoized DP for rod cutting.

Bottom-Up DP alg for Rod Cutting



Solve the subproblems one-by-one, starting w/ the smallest, and put the results in a table. No need for recursion.

Bot-Up-Cut-Rod(p, n)

```
Let r[0...n] be a new array.
r(0)=0
for j = 1 to n                          compute r(j).
    q = -∞
    for i = 1 to j {
        q = max(q, p(i) + r(j-i))
    }
    r(j) = q.
return r(n)
```

$\Theta(n^2)$ $\Theta(n)$

Runtime $= \Theta(n^2)$.

Bot-Up-cut-Rod(5).

$i =$	1	2	3	4	5	
$p =$	1.	5.	8	9	10	

$p_3 = 8$ $\quad p_1 + r_2 = 6$
$p_2 + r_1 = 6$

r	1	5	8	10	13	

$p_2 = 5$
or $p_1 + r_1 = 2$

$p_4 = 9$
$p_3 + r_1 = 9$
$p_2 + r_2 = 10$
$p_1 + r_3 \; 9$

$p_5 = 10$
$p_4 + r_1 = 10$
$p_3 + r_2 = 10$
$p_2 + r_3 = 13$
$p_1 + r_4 = 11$

return $r(5) = 13.$