Notes on ~~Greedy~~ Make Change:

→ example where greedy doesn't work:

coins: 1¢, 3¢, 4¢

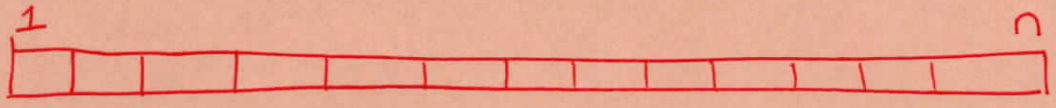
counter example 6¢ greedy: 4¢, 1¢, 1¢ (3 coins)

opt: 3¢, 3¢ (2 coins)

→ we defined this recursively by thinking about the last coin to add. What could it be?

11 Oct 2021

Rod Cutting



we can cut any size... as long as it is an integer. Each size has some price we can sell it for (given as an array of length n).

Q: How do we cut the rod in order to make the most money?

1st step: to think of this recursively.

input: n, the size of the rod
prices, an array of length n of prices

e.g., n = 3

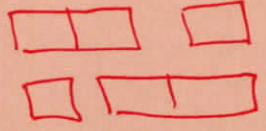
prices =

1	2	3
$2	$5	$1

ways we can sell this rod:

- 3 1-unit pieces: $3 \cdot \$2 = \$6$
- 1 2-unit + 1 1-unit: $\$5 + \$2 = \$7$ Best choice!
- 1 3-unit piece: $1

↑ 2 ways to do this



last piece can be

$ earned from selling

1 unit — BestPrice(n-1, prices) + prices[1]

2 units — BestPrice(n-2, prices) + prices[2]

3 units — BestPrice(n-3, prices) + prices[3]

⋮

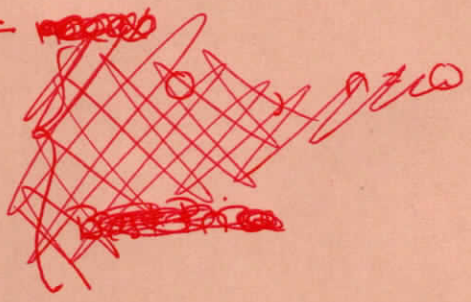
⋮

n units — BestPrice(1, prices) + prices[n-1]

prices[n]

↑ What if BestPrice(0, -) = 0

BestPrice(n, prices) =



↑ size of rod

↑ array of prices, length n

$$BestPrice(n, prices) = \begin{cases} 0, & n = 0 \\ \max_{j=1 \cdots n} \{ BestPrice(n-j, prices) + prices[j] \} \end{cases}$$

recursion is awful!

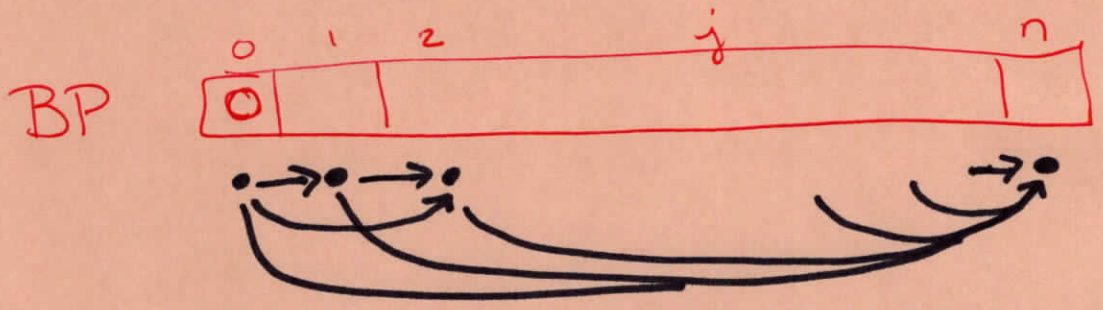
$$T(n) = \sum_{j=1}^{n} T(n-j) + \Theta(1)$$

Note: There are really only $n+1$ problems

BestPrice (x, prices)

where $x \in \{0, 1, \ldots n\}$

Let's store these solutions in an array



All arrows go from left to right!

$\Rightarrow$ we can solve the subproblems from left to right!

Solution:

```
BestPrice (n, prices)
    BP ← an array of length n+1, indexed from 0
    BP[0] ← 0 ; i ← 1
    while i ≤ n
        BP[i] ← max_{j=1,2...,i} BP[i-j] + prices[j]
        i++
    end while
    return BP[n]
```

n times { n-i { BP[i] ← ... }

a hidden while/for loop!

* no more recursion here! (yet alone an ugly recursion!)

Groups: #

1. Walk through BestPrice(3, [$2, $5, $1])
2. Runtime? $\Theta(n^2)$ $\quad \sum_{i=1}^{n} \sum_{j=1}^{i} 1 = \sum_{i=1}^{n} i = \Theta(n^2)$
3. Space complexity?

$\Theta(n)$ to store: BP ← array of length n, $\Theta(n)$

n prices } input, also $\Theta(n)$

---

the hidden for loop:

TO compute max(A), ← an array of length k, here is one way:

```
max(A)
    tempmax ← A[1]
    i ← 2
    while i ≤ k
    |   if A[i] > tempmax
    |   |   tempmax ← A[i]
    |   end if
    |   i++
    end while
```

Time: $\Theta(k)$

Space: $\Theta(n)$ input

$\Theta(1)$ additional space

$\Theta(n)$ total

Loop Invariants: a statement that is a fcn of the variables avaible to us that remains true each time we enter the loop & helps us get closer to the sol'n.

- while loop in max(A): tempmax holds the max of A[1, ... i-1]
- BestPrice while loop: BP[t] holds best price we can get for rod of length t ∀ t = 1, 2, ..., i-1