23 Sept 2019

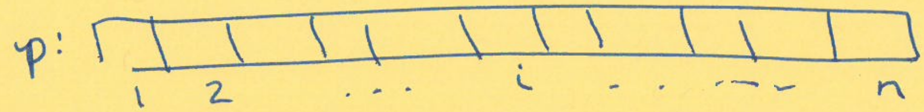
Dynamic Programming

- related to D&C: we solve a problem by breaking down into simpler problems & then combining the simpler problems.
- DP helps when we have independent overlap between the subproblems/ simpler problems

```
DPStockExchange(p, n)
        soln ← array of size n
Θ(n)    ~~[illegible]~~
        minidx ← 1       } the sol'n for p[1...1]
        sol[1] ← 0       }
        for i = 2... n
 Θ(1)       if(soln[i-1] < p[i] - p[minidx])
repeated        sol'n[i] ← p[i] - p[minidx]
n-times     else
   ⇓            soln[i] ← soln[i-1]
Θ(n) for    endelse
the for lop.  if (p[i] < p[minidx])
                minidx ← i
            endif
        end for
        return soln[n]
```

Runtime: $\Theta(n)$ Linear!

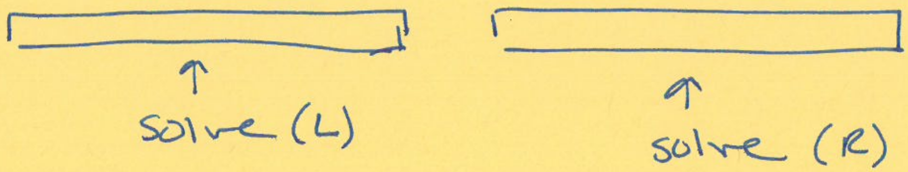
Recall the stock market problem:



← price on each day i

goal: find $i < j$ to max $p(j) - p(i)$

Sol'n:



$$\max(\text{solve}(L), \text{solve}(R), \max(R) - \min(L))$$

time complexity:

$$T(n) = 2T(\tfrac{n}{2}) + \Theta(n)$$
$$= \Theta(n \log n)$$

Another sol'n

	1	2	3	4	5
p:	5	2	8	6	13
best:					

→

$i = 3$

best(i) holds sol'n for $p[1 \ldots i]$ and any info needed to solve $p[1 \ldots i+1]$

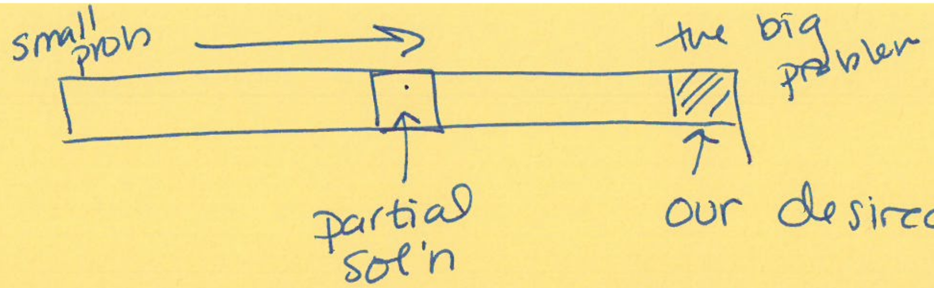
$$\text{best}[3] = \begin{cases} \text{solution} = 8 - 2 = 6 \\ \text{pair } (2,3) \\ \text{minidx} = 2 \end{cases}$$

best[4].sol'n = max (best[3].sol'n, p[4] − p[minidx])

↑ this update is now constant time!

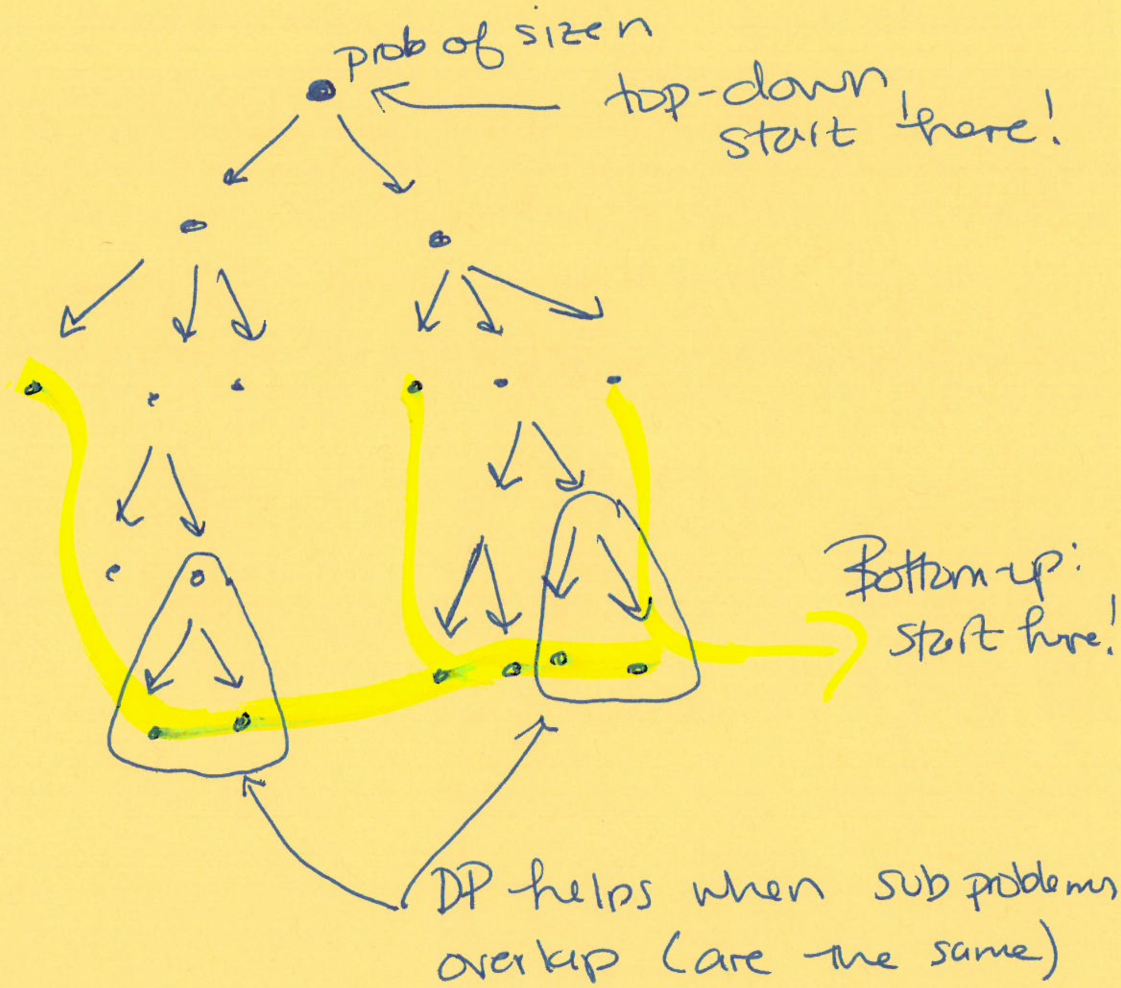
Another ex: p = [2, 8, 1, 4, ...]

Array



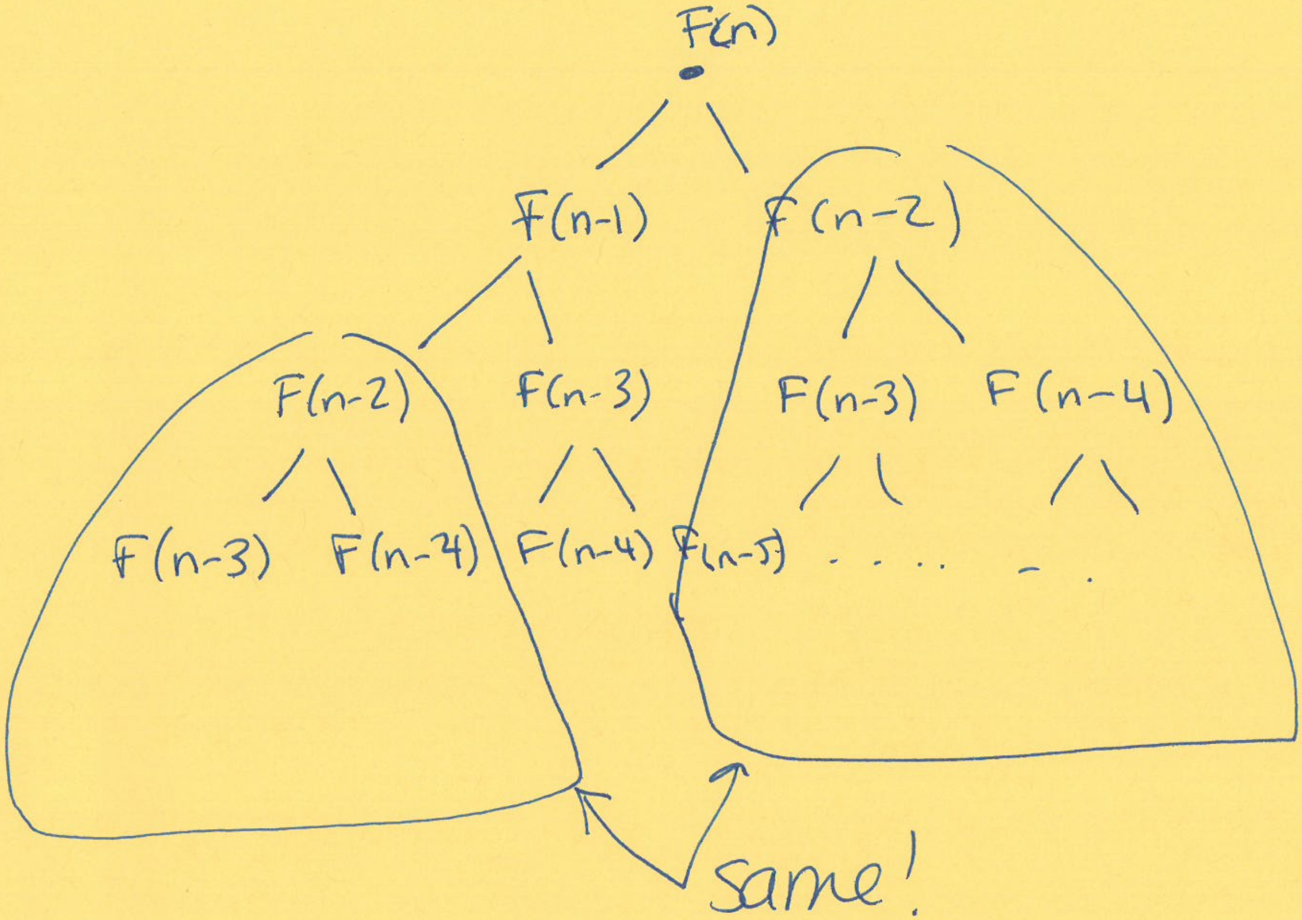
"bottom-up" approach: start w/ the small & grow until you solve the big problem.

schematically
Think of recursion tree



$F(0) = F(1) = 1$

$F(n) = F(n-1) + F(n-2), \quad \forall n \geq 2$



Bottom-up:

0	1			
1	1	2	3	. . .

Top-down:

0	1	2	3	4	5
1	1	2	3		

memoization

"take a note"

$F(5) = F(4) + F(3)$

$F(4) = F(3) + F(2)$

$F(3) = F(2) + F(1)$

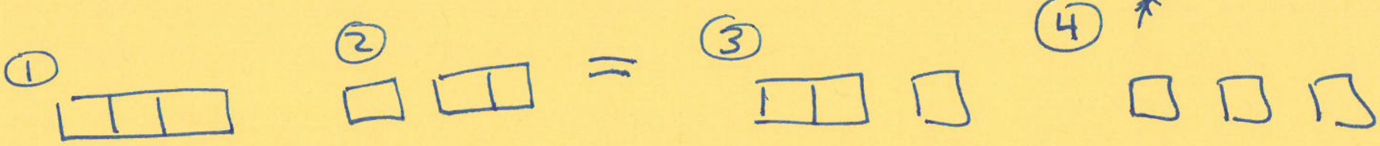
$F(2) = F(1) + F(0)$

Rod Cutting

- given: a rod of length n, profit array
- want: cut into pieces (of unit multiples) to maximize profit

> - profit[i] = profit of having a rod of length = i
> - profit is length n

e.g., $n = 3$
profit = [6, 2, 17]



total profit: 17 | $6 + 2 = 8$ | $6 + 6 + 6 = 18$

The optimal sol'n (note: sometimes opt is not unique!)

in groups: ① what is a DP sol'n to this?

② what is the $\Theta(2^n)$ brute force approach?

(Answers next time)