L7 5800

feb 8/10 2022

shelat

We are introducing a new algorithmic technique


NY
West 81st Street, New York, ...
Add a photo

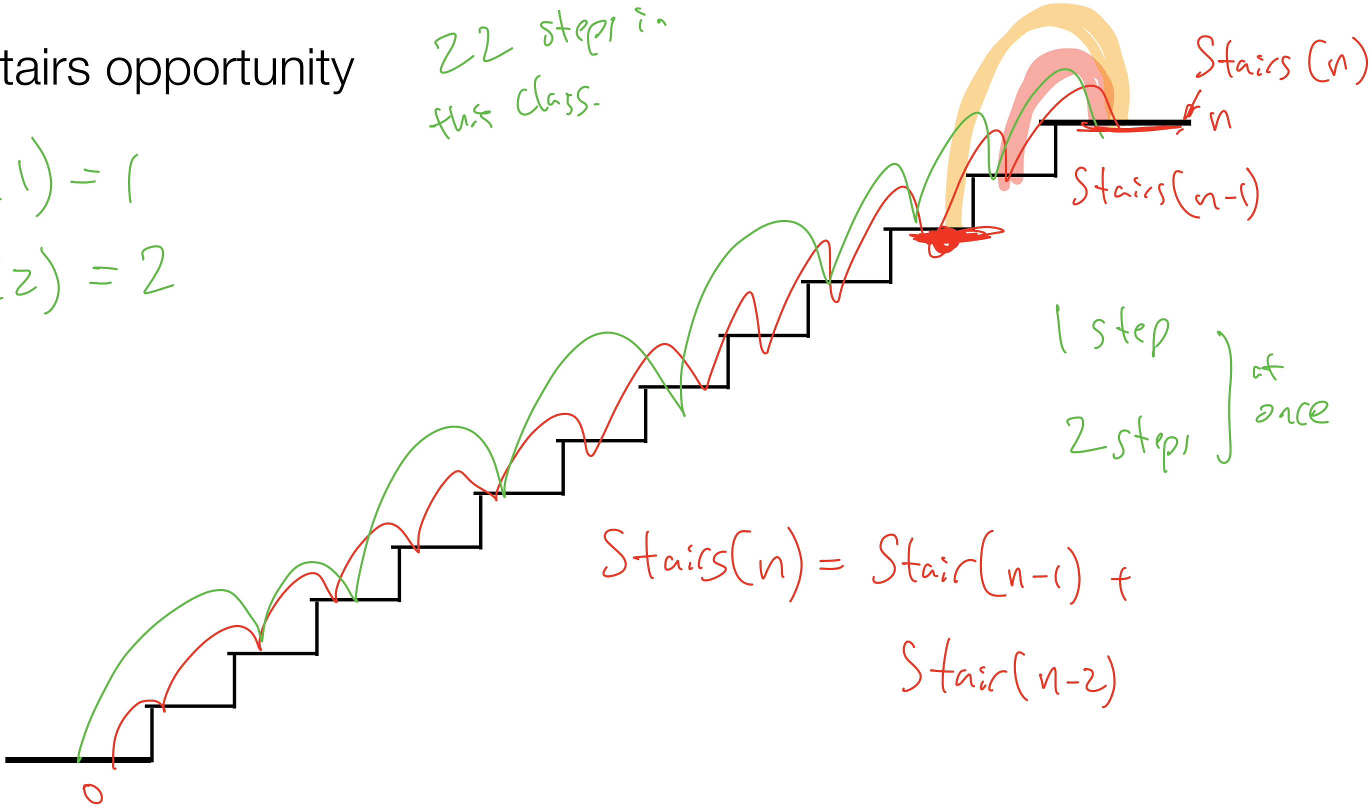

MIGROS

The Stairs opportunity

22 steps in this class.

$Stairs(1) = 1$

$Stairs(2) = 2$



1 step
2 steps
at once

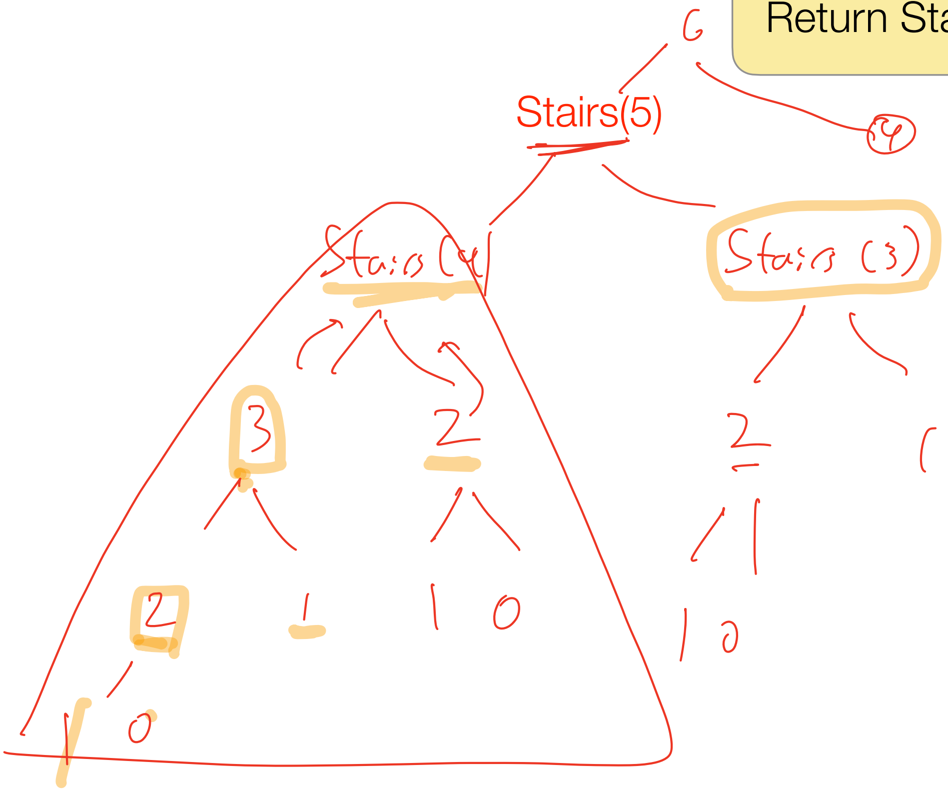
$$Stairs(n) = Stair(n-1) + Stair(n-2)$$

```
Stairs(n)
    if n<=1 return 1
    return Stairs(n-1) + Stairs(n-2)
```

Inefficient

Stairs(n)

if n<=1 return 1
Return Stairs(n-1) + Stairs(n-2)



Stairs(n)

```
if n<=1 return 1
Return Stairs(n-1) + Stairs(n-2)
```

Stairs(5)

Stairs(4)

Stairs(3)

Stairs(n)

if n<=1 return 1
Return Stairs(n-1) + Stairs(n-2)

Memorize previous answers.

Stairs(5)

Stairs(4)

Stairs(3)

Stairs(3) Stairs(2) Stairs(2) Stairs(1)

Stairs(2) Stairs(1) Stairs(1) Stairs(0) Stairs(1) Stairs(0)

```
initialize memory M

Stairs(n)
```

Stairs(n)

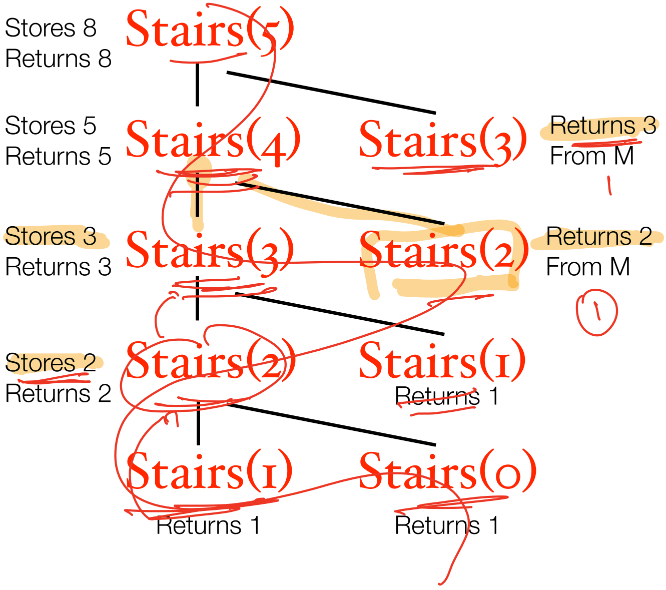
```
if n<=1 then return 1
if n is in M, return M[n]
answer = Stairs(i-1)+ Stairs(i-2)
M[n] = answer
return answer
```

look to see if the problem has been solved before.

Memorize previous answer

Stairs(5)

```
Stairs(n)
    if n<=1 then return 1
    if n is in M, return M[n]
    answer = Stairs(i-1)+ Stairs(i-2)
    M[n] = answer
    return answer
```



Observation 3:

The Order of solving problem can help.

$0 \to 1 \to 2 \to 3 \to \ldots$

Stairs(n)

```
stair[0]=1
stair[1]=1
```

Stairs(n)

$\Theta(n)$

```
stair[0]=1
stair[1]=1

for i=2 to n
   stair[i] = stair[i-1]+stair[i-2]
return stair[i]
```

For this simple example, you might have started with this structure. But the same pattern applies to more complicated examples.

Dynamic Programming

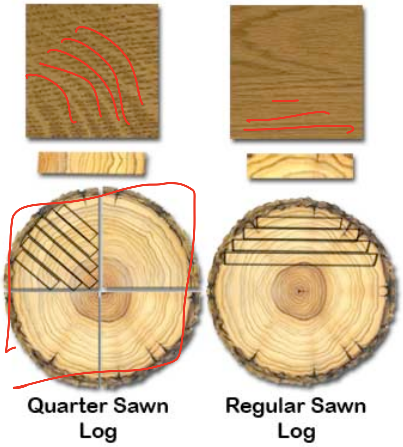
two ideas

recursive structure

memoizing

Order.

wood cutting



http://www.amishhandcraftedheirlooms.com/quarter-sawn-oak.htm

http://snlm.files.wordpress.com/2008/08/bill-wakefield-and-carl-fie.gif

Spot price for lumber

1”	2”	3”	4”	5”	6”	7”	8”
1$	3	13	19	25	35	60	80$

Log cutter dilemna

input to the problem: ~~width~~ n and prices $(p_1, \ldots, p_n)$

of a log

goal: find the right cuts $i_1 \ldots i_k$

that maximize the revenue from the log

$$\max \sum_{j=1}^{k} p_{i_j} \qquad \text{such that} \qquad \sum_{j=1}^{k} i_j \leq n$$

Log cutter dilemna

input to the problem: width n and prices $(p_1, \ldots, p_n)$

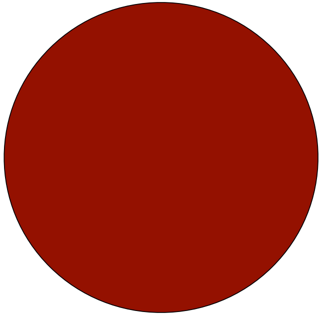
goal: Output plank widths $i_1, \ldots, i_k$ such that the sum of the plank widths is less than n which maximizes the revenue $\sum_{j=1}^{k} p_{i_j}$

Greedy fails

1"	2"	3"	4"	$\boxed{5"}$
1$	6$	7$	8$	$\boxed{10\$}$

2" + 3" = 13$ > 10$

5" log



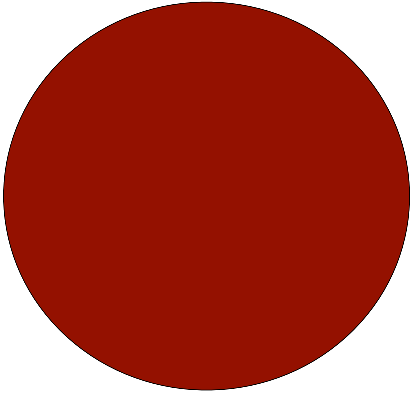
Greedy “Avg” fails

1”	2”	3”	4”	5”	6”
1$	18$	24$	36$	50$	50$
1	9	8	9	10	8.3

5” + 1” ⇒ 51$

4” + 2” ⇒ 54$

6” log



Observation

This is our log

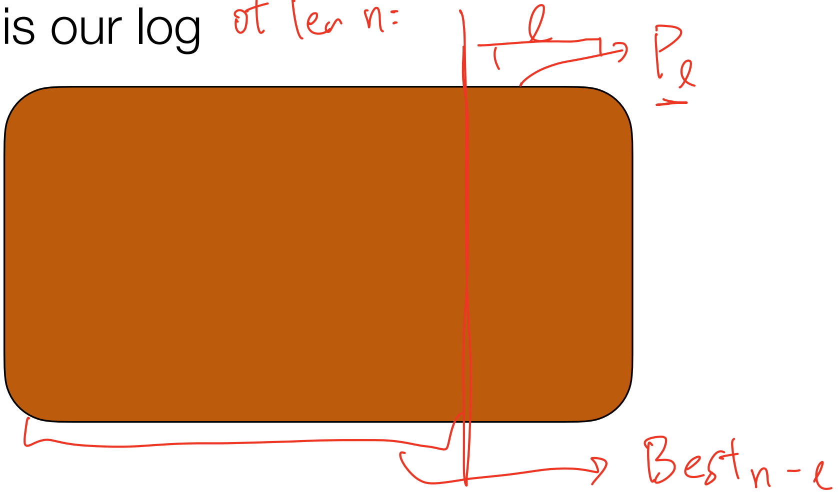
the very last cut I made was between 1...n.

$Best_n = P_{last\,cut} + B_{n-lastcut}$

Best for n-lastcut.

Observation

This is our log of len n:



Think about the very last cut that is made in the optimal solution. From this, we know that the best revenue is the price of this cut plus the best solution for the rest of the log.

Solution equation

$$\underline{Best_n} = \underset{k=1 \ldots n}{\max} \left\{ \underline{P}_k + \underline{Best_{n-k}} \right.$$

$$\Downarrow$$

$$\max \begin{cases} P_1 + Best_{n-1} \\ P_2 + Best_{n-2} \\ P_3 + Best_{n-3} \\ \ddots \\ \underline{P_n} + Best_{n-n} \end{cases}$$

Approach

Approach

BestLogs($n, (p_1, \ldots, p_n)$)

```
    if n<=0 return 0
```

BestLogs($n, (p_1, \ldots, p_n)$)

```
if n<=0 return 0
for i=1 to n
 Best[i] = 
```
$$\max_{k=1\ldots i} \{p_k + \mathrm{Best}[i-k]\}$$

```
return Best[n]
```

Example

1"	2"	3"	4"	5"	6"
1$	18$	24$	36$	50$	50$

BestLogs($n, (p_1, \ldots, p_n)$)

```
if n<=0 return 0
for i=1 to n
 Best[i] =  max_{k=1...i} {p_k + Best[i - k]}
return Best[n]
```

$$\text{Best}[i] = \max_{k=1\ldots i}\{p_k + \text{Best}[i-k]\}$$

0						

Example

1"	2"	3"	4"	5"	6"
1$	18$	24$	36$	50$	50$

$\text{BestLogs}(n, (p_1, \ldots, p_n))$

```
if n<=0 return 0
for i=1 to n
 Best[i] = max_{k=1...i} {p_k + Best[i-k]}
return Best[n]
```

$$\text{Best}[i] = \max_{k=1\ldots i} \{p_k + \text{Best}[i-k]\}$$

1 + 1   18 + 0

$$\max_{1,2} \{p_1 + B_1, p_2 + B_0\}$$

$$\max_{1,2,3,4} \{p_1 + B_3, p_2 + B_2, p_3 + B_1, p_4 + B_0\}$$

Best

0	1	18	24			

0

$$\max_{1} \{p_1 + B_0\}$$

$$\max_{1,2,3,4,5} \{p_1 + B_4, p_2 + B_3, p_3 + B_2, p_4 + B_1, p_5 + B_0\}$$

1   18   18   1   24   0

$$\max_{1,2,3} \{p_1 + B_2, p_2 + B_1, p_3 + B_0\}$$

The actual cuts?

BestLogs($n, (p_1, \ldots, p_n)$)

```
if n<=0 return 0
for i=1 to n
 Best[i] =
```
$$\max_{k=1\ldots i} \{p_k + \text{Best}[i-k]\}$$
```
  choice[i] = k*
return Best[n]
```