L9

4102

Feb 18 2016

abhi shelat

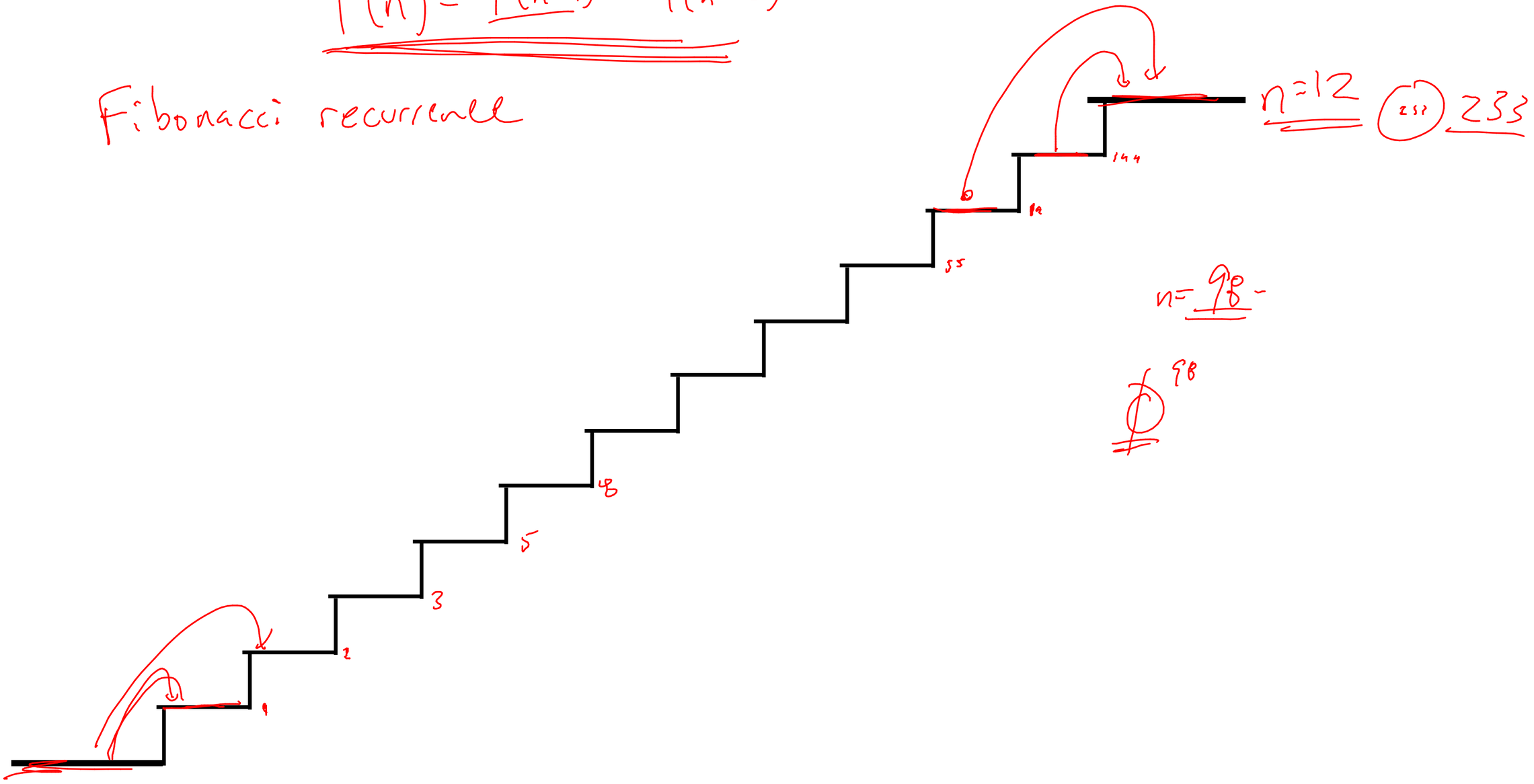
Dynamic programming: log cutter, matrix chains, typesetting


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


MIGROS

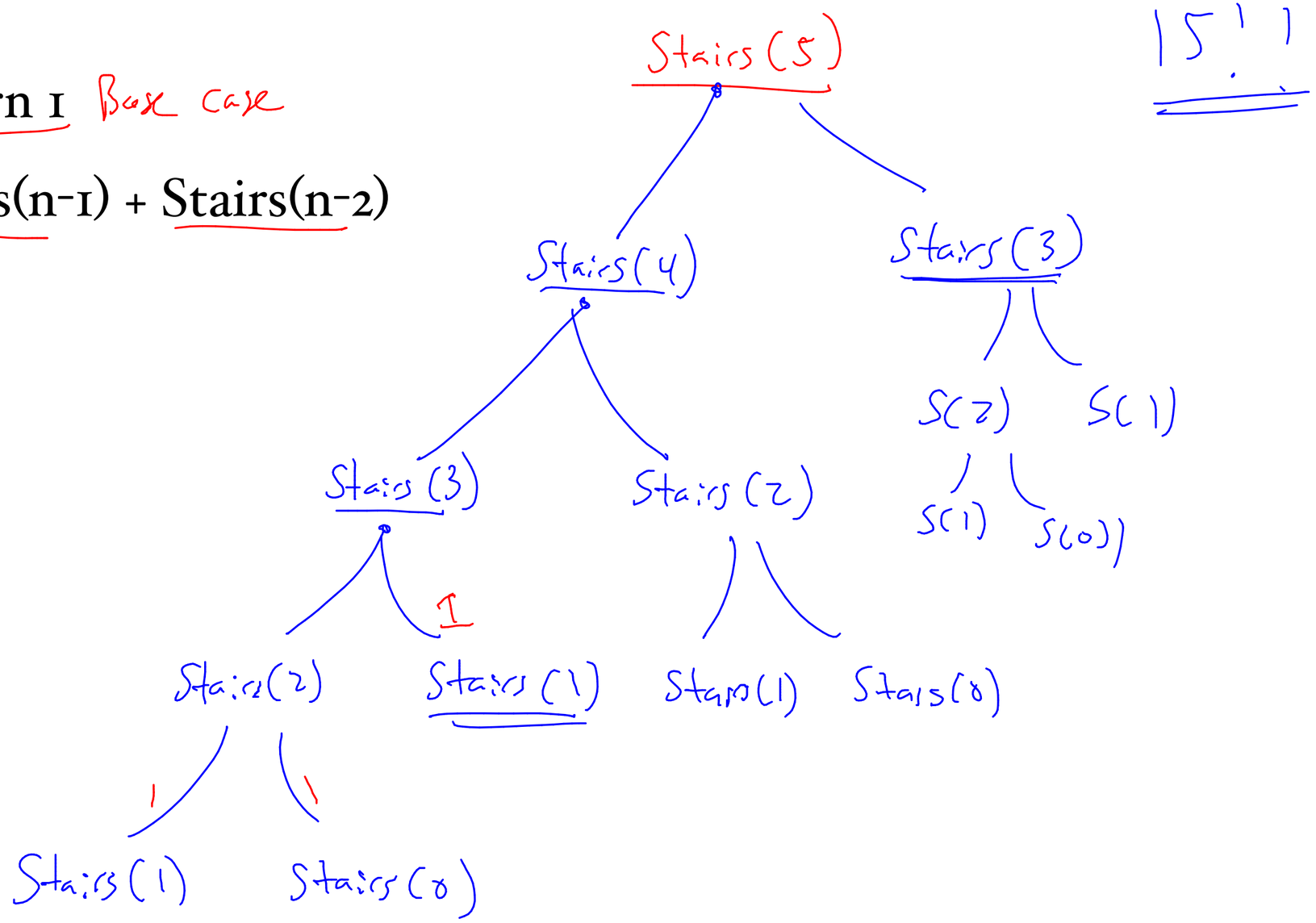
$$T(n) = T(n-1) + T(n-2)$$

Fibonacci recurrence



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

Base case



15!!

Wasteful b/c it redundantly computes many calls.

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

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)

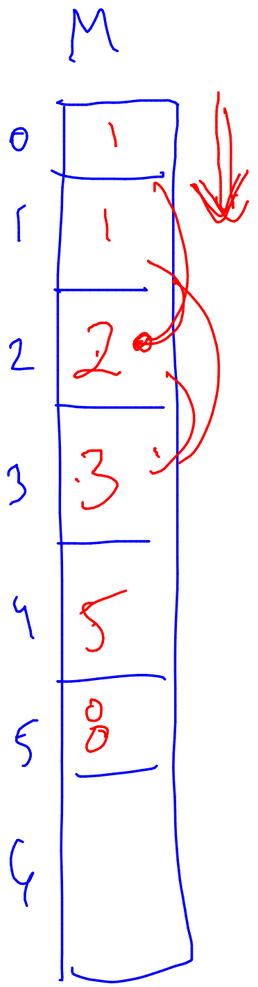
if $n \leq 1$, return 1.

if Memory[n] is already set, return M[n]

else return Stairs(n-1) + Stairs(n-2)

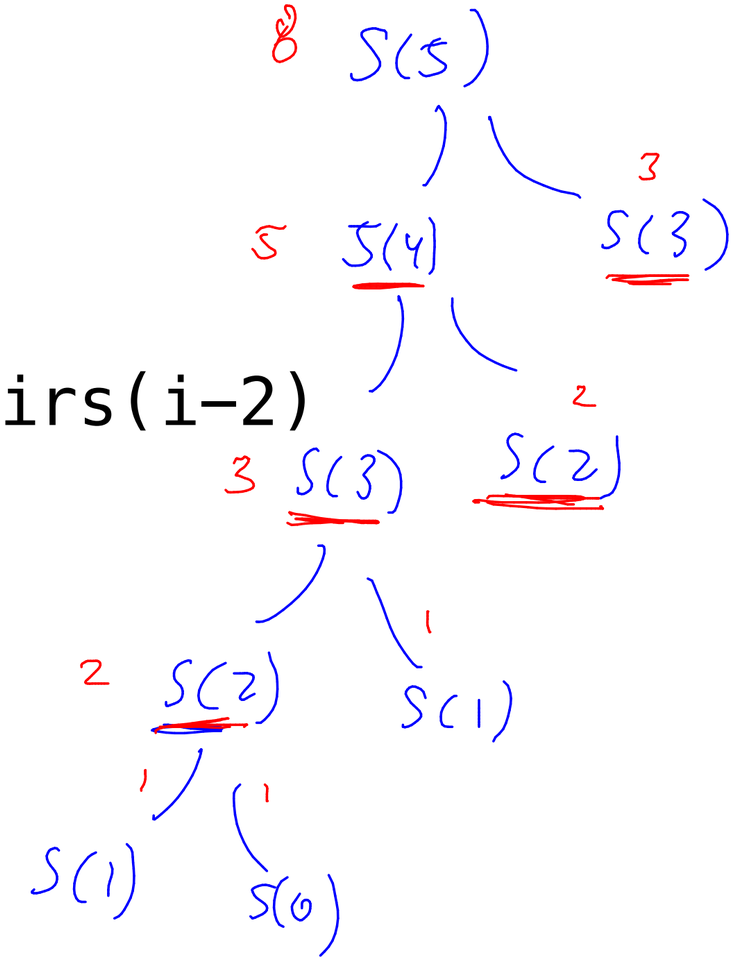
```
initialize memory M

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



$\Theta(n)$ memory

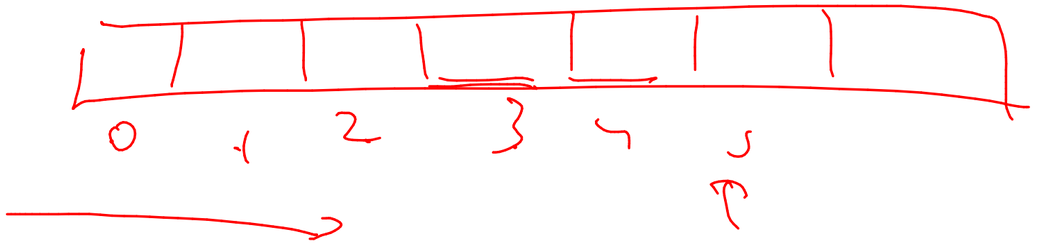
$\Theta(n)$ running time.



```
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
```

Stairs(5)

Stairs(n)




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

for i = 2 to. n

Stairs[i] = Stairs[i-1] + stairs[i-2]

return Stairs[n]

Stairs(n)

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

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

Dynamic Programming

→ (1) Recursive structure to the problem.

$T(n) = T(n-1) + T(n-2)$ for example.

→ (2) Pick the correct order for evaluating the smaller problems

two big ideas

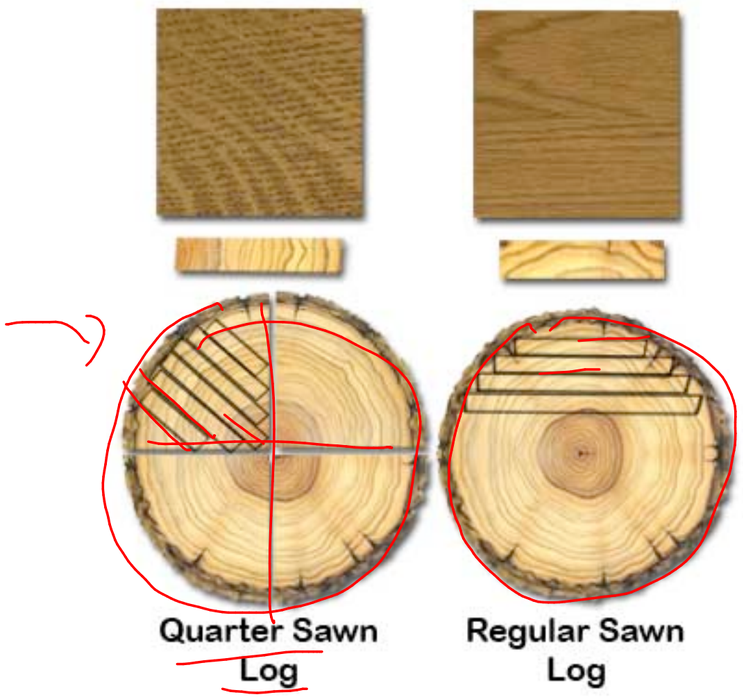
two big ideas

recursive structure

+

memoizing

wood cutting



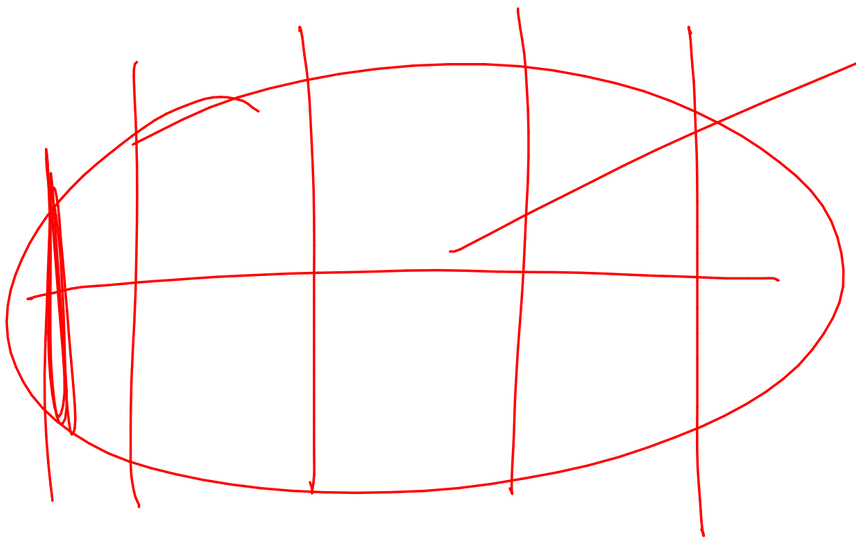
http://www.amishhandcraftedheirlooms.com/quarter-sawn-oak.htm

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

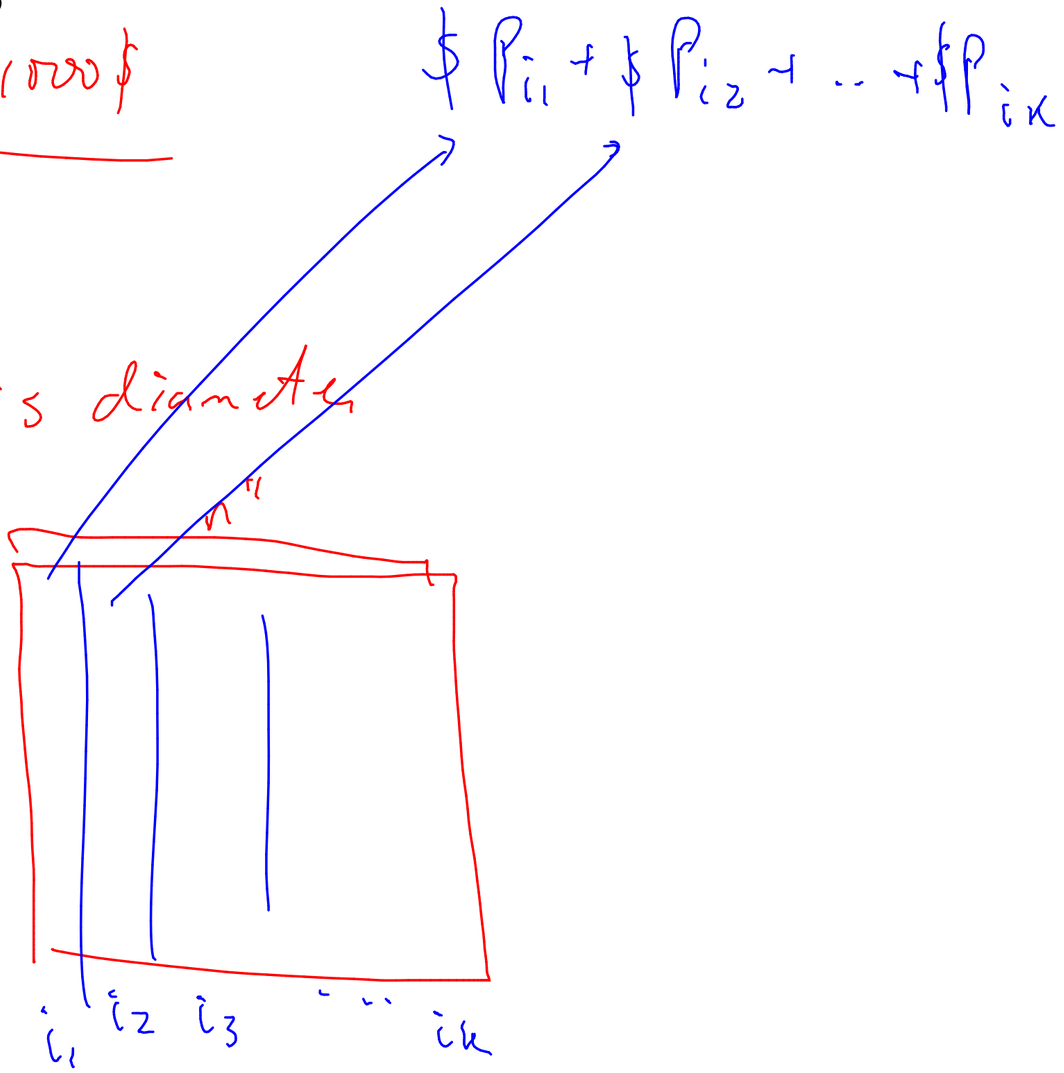
Spot price for lumber

1"	2"	3"	4"	5"	6"	7"	8"
1$	6$	7$	8$	11$	90$	200$	1000$

given a n" thick log



→ n is diameter



Log cutter dilemna

input to the problem: $n, (p_1, \ldots, p_n)$ ← spot prices for an i" board; n → dimension of timber

goal: MAX profit, i.e.

output $(i_1, \ldots i_k)$ the widths of cuts to make

subject to $\sum_{j=1}^{k} i_j = n$

$$\max \sum_{j=1}^{k} p_{i_j}$$

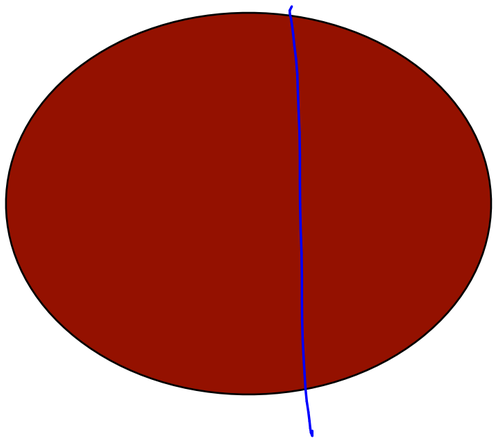
Greedy fails

1”	2”	3”	4”	5”
1$	6$	7$	8$	10$

10$ − 5“ = i_1

$i_1 = 2$, $i_2 = 3$, $13

5” thick log

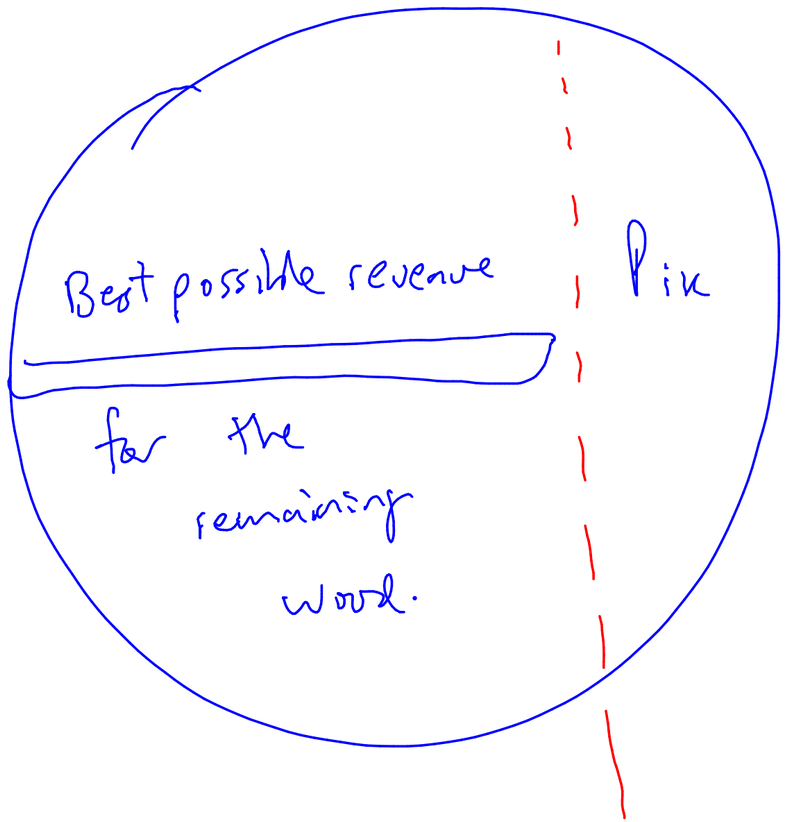


What about averaging argument ??

Observation

$Best_n$: best profit for an n" thick slab.

. very last cut. ix.



$Best_n = P_{ix} + Best_{n-ix}$

?

How many choices are there for this Pix. ??

@ most n.

Solution equation

$$Best_n = \max \begin{cases} P_1 + Best_{n-1} \\ P_2 + Best_{n-2} \\ P_3 + Best_{n-3} \\ \vdots \\ P_{n-1} + Best_1 \\ P_n + Best_0 \end{cases}$$

n possibilities, we take the max

Approach

B[0] Best[1]

4

$Best_n = \max \begin{cases} P_1 + B_{n-1} \\ \vdots \\ P_n + Best_0 \end{cases}$

$Best_1 = \max \{ P_1 + B_0 \}$

$Best_2 = \max \begin{cases} P_1 + B_1 \\ P_2 + B_0 \end{cases}$

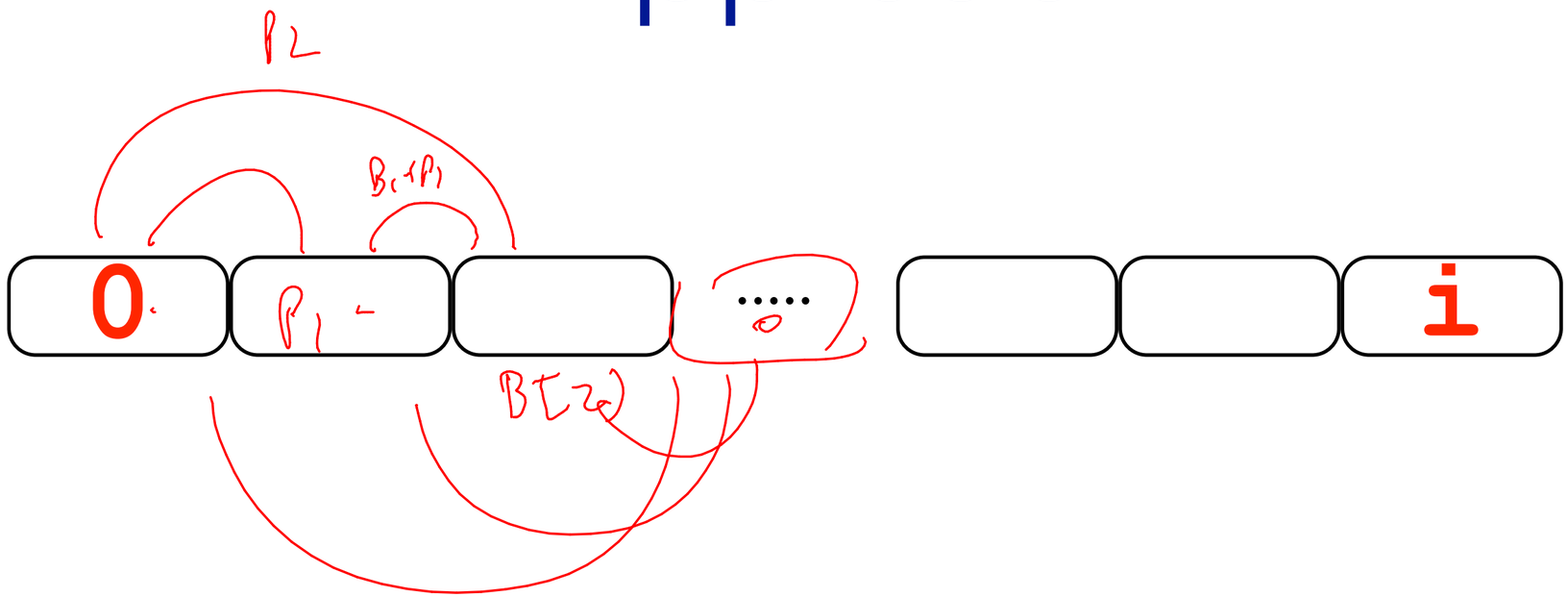
Avg example

p	1	2	3	4	5	6
	1	18	24	36	50	50
avg price	1	9	8	9	10	

$n = 6$, $5, 1 \Rightarrow 5\$$

$2, 4 \Rightarrow 5\$\$$

Approach



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

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

Initialize $B[0 \ldots n]$

$B[0] = 0$

for $i = 1$ to n

set $B[i] = \max_{j=1}^{i} \{ p_j + B[i-j] \}$

return $B[n]$

I picked the order of evaluation from $i = 1 \ldots n$

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]
```

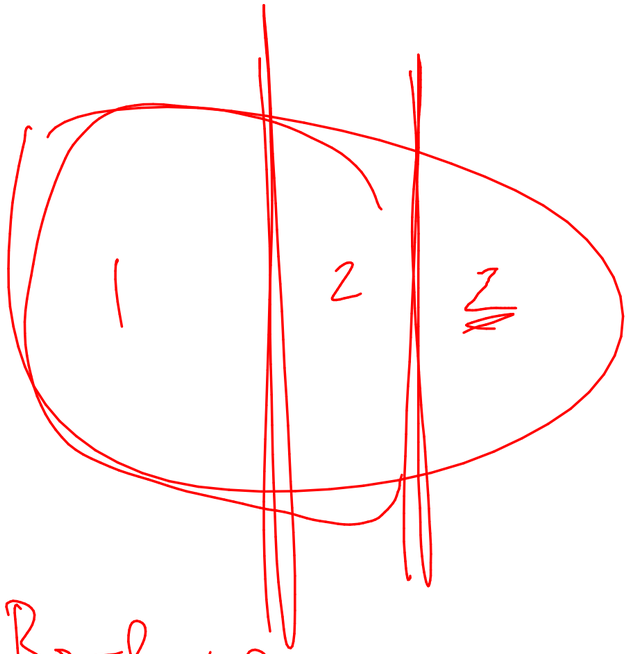
n loops.

$\Theta(k)$

$1 + 2 + 3 + \ldots + n-1 + n = \Theta(n^2)$

$0 \ldots n-1$

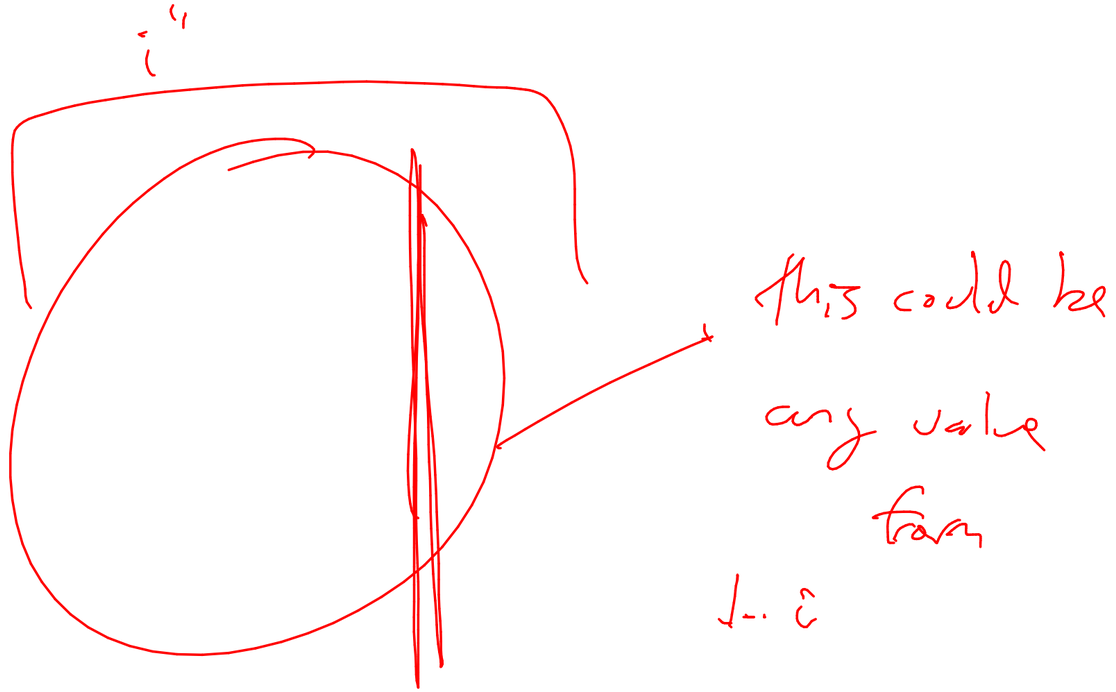
$1 \ldots n$



$B_3 = P_2 + B_1$

$B_5 = B_3 + P_2$

$$Best_n = \max \begin{cases} Best_n + P_0 \\ Best_{n-1} + P_1 \end{cases}$$

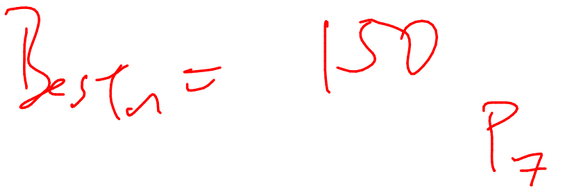


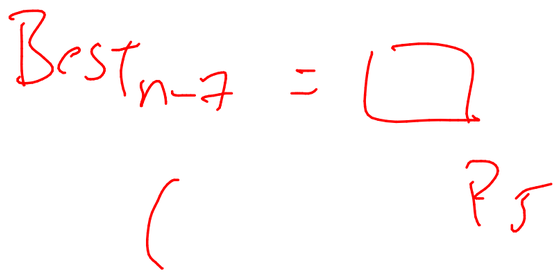

Bn
B1
B2
B3
Bn-1

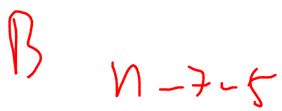
The actual cuts?

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

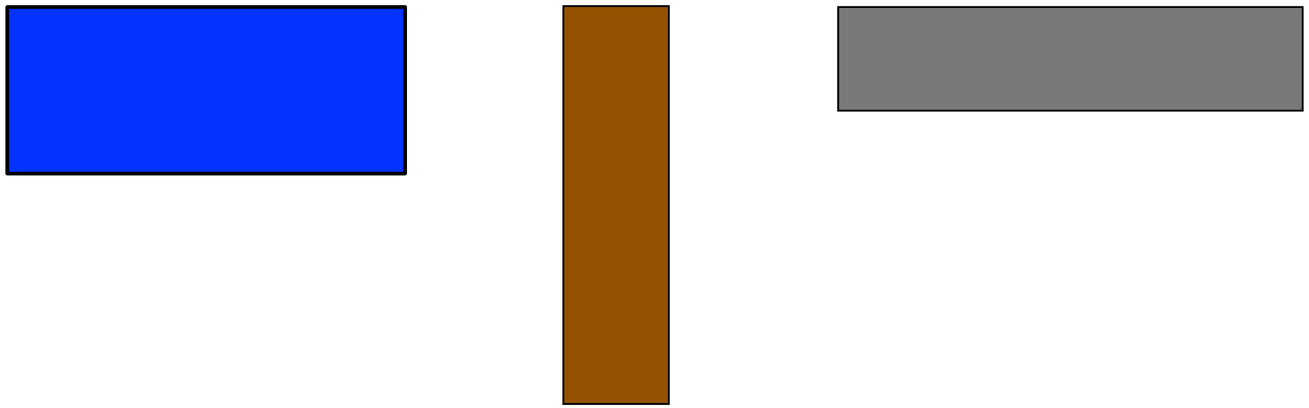
```
if n<=0 return 0
for i=1 to n
  Best[i] = max_{k=1...i} {p_k* + Best[i − k]}
  choice[i] = k*
return Best[n]
```



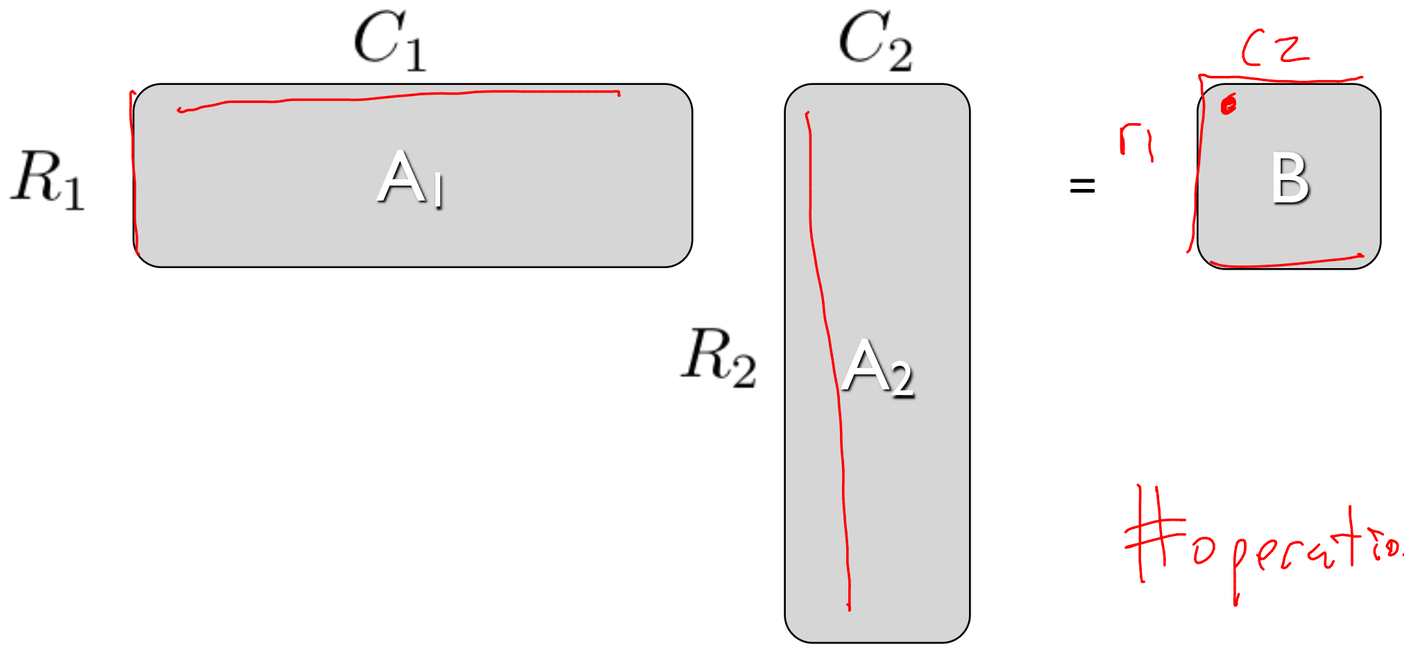




Matrix



$C_1 = R_2$



#operations: $r_1 \cdot c_2 \cdot c_1$

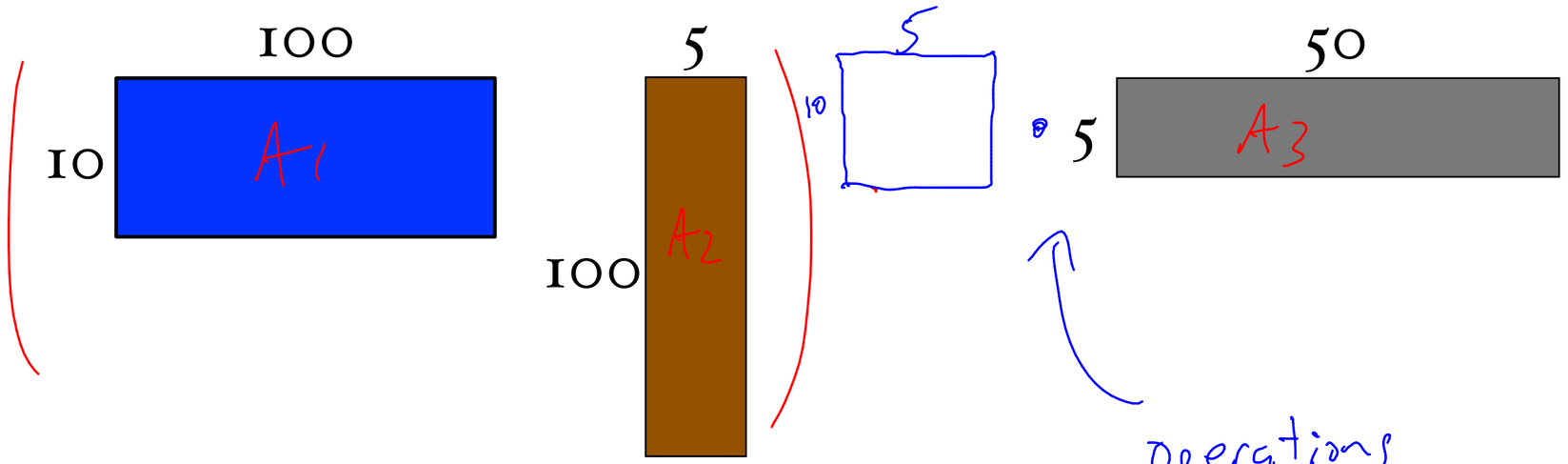
$= r_1 \cdot c_2 \cdot r_2$

$$A_1 \cdot A_2 \cdot A_3$$

2 3 5

$$(A_1 \cdot A_2) \cdot A_3 \qquad A_1 \cdot (A_2 \cdot A_3)$$

$$(A_1 \cdot A_2) \cdot A_3$$



ops

$10 \cdot 100 \cdot 5$

5000

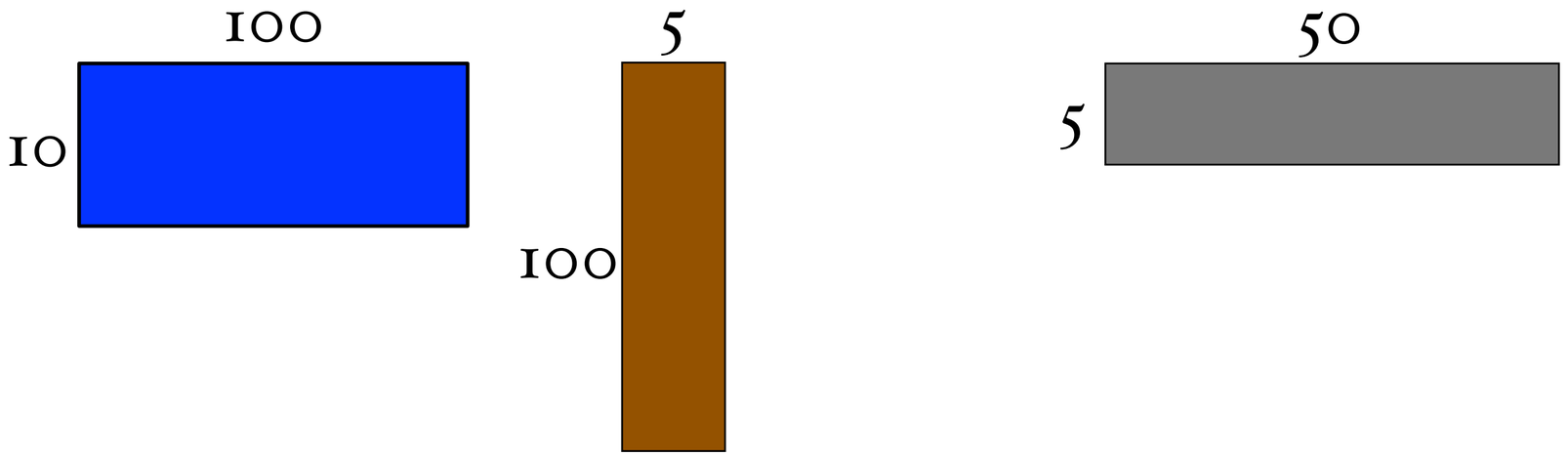
operations

$10 \cdot 5 \cdot 50$

2500

$= 7500$ ops

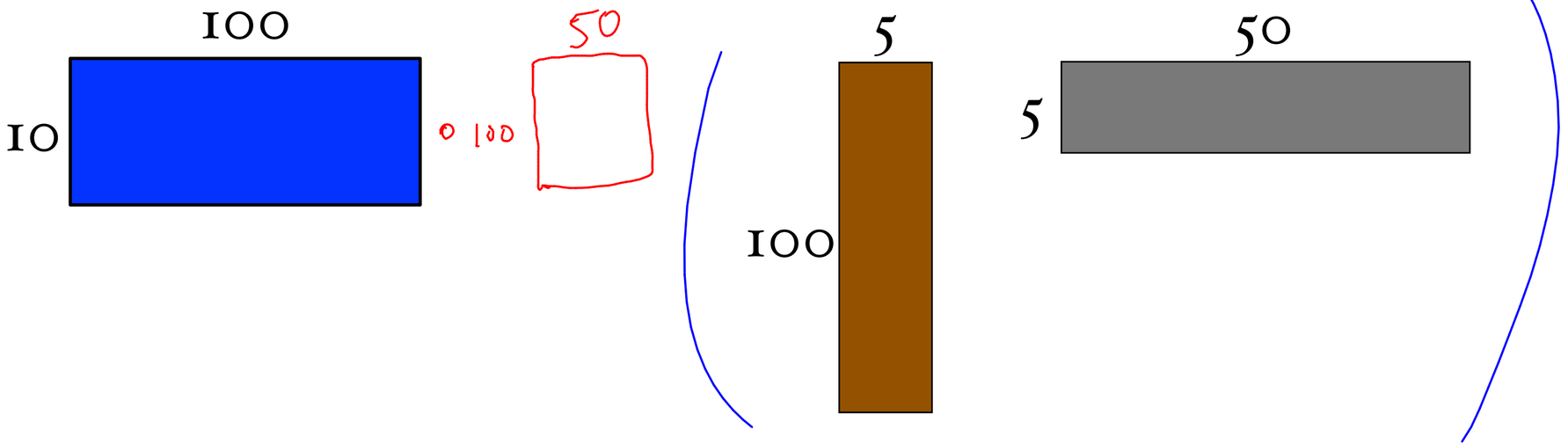
$$(A_1 \cdot A_2) \cdot A_3$$



$$10 \cdot 100 \cdot 5 + 10 \cdot 5 \cdot 50$$

operations

$$A_1 \cdot A_2 \cdot A_3$$



$10 \cdot 100 \cdot 50$

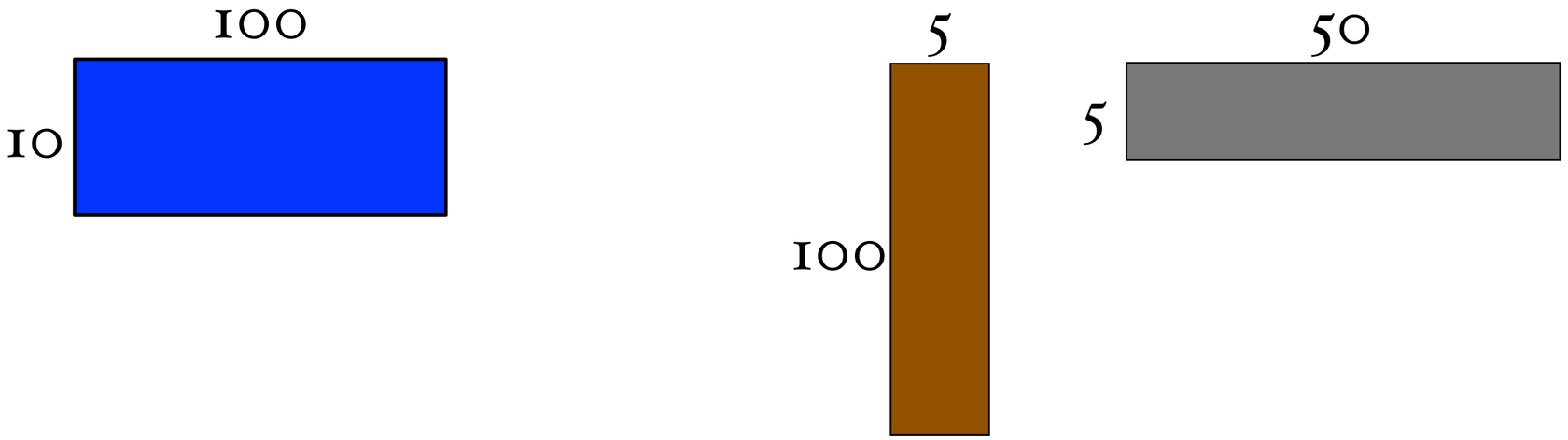
$50,000$

$100 \cdot 5 \cdot 50$

$25,000$ $=$ 75000

...

$$A_1 \cdot A_2 \cdot A_3$$



$$100 \cdot 5 \cdot 50 + 10 \cdot 100 \cdot 50$$

operations

order matters

(for efficiency)

how many ways to compute?

$$A_1 A_2 A_3 \ldots A_n$$

q: what is the best way to multiply them??

how many ways to compute?

$$A_1 \cdot A_2 A_3 \ldots A_n$$

$$A_1 A_2 \cdot A_3 \ldots A_n$$

$P(n)$ = # ways to mult n matrices

$P(n) = P(1) \cdot P(n-1) +$

$P(2) \cdot P(n-2) +$

$P(3) \cdot P(n-3) +$

$P(n/2)\, P(n/2) +$

$P(n-1)\, P(1) +$

Recurrence

$\sim 4^n$

how many ways to compute?

$A_1 A_2 A_3 \ldots A_n$

$A_1 A_2 A_3 \ldots A_n$

$A_1 A_2 A_3 \ldots A_n$

how many ways to compute?

$A_1 \; A_2 A_3 \ldots A_n$

$A_1 A_2 \; A_3 \ldots A_n$

$A_1 A_2 A_3 \; \ldots A_n$

how do we solve it?

identify smaller instances of the problem

devise method to combine solutions

small # of different subproblems

solved them in the right order

optimal way to compute

$A_1 A_2 A_3 A_4 \ldots A_n$

optimal way to compute

$$A_1 A_2 A_3 A_4 \ldots A_n$$

B[1,n]

optimal way to compute

$$A_1 A_2 A_3 A_4 \ldots A_n$$

B[1,n]

B[1,1]

B[2,n]

$R_1 C_1 C_n$

optimal way to compute

$$A_1 A_2 A_3 A_4 \ldots A_n$$

B[1,n]

B[1,1]	B[1,2]	...	B[1,n-2]	B[1,n-1]
B[2,n]	B[3,n]	...	B[n-1,n]	B[n,n]

$R_1 C_1 C_n$ $R_1 C_2 C_n$ $R_1 C_{n-2} C_n$ $R_1 C_{n-1} C_n$

$$B(i,i) = 1$$

$$B(1,n) = \min \left\{ \right.$$

$$B(i,i) = 1$$

$$B(1,n) = \min \begin{cases} B(1,1) + B(2,n) + r_1 c_1 c_n \\ B(1,2) + B(3,n) + r_1 c_2 c_n \\ \vdots \\ B(1,n-1) + B(n,n) + r_1 c_{n-1} c_n \end{cases}$$

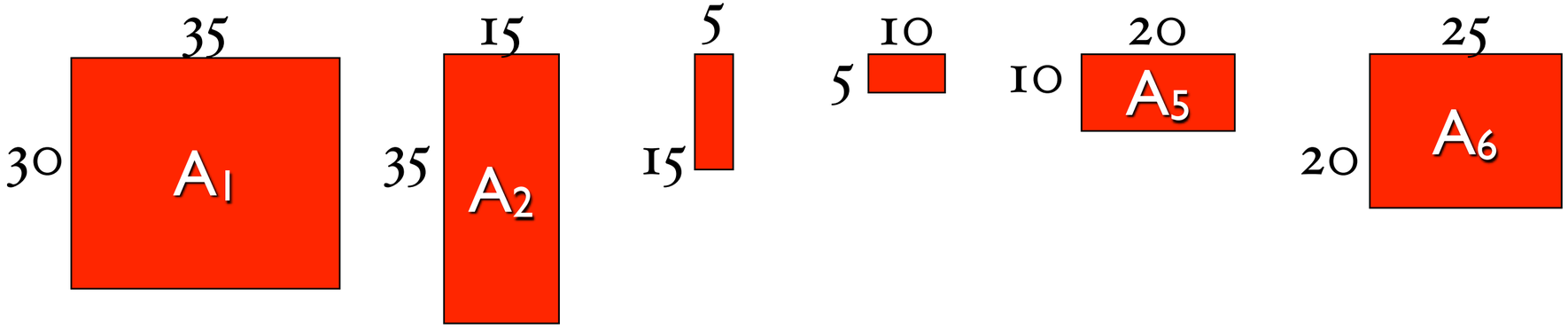
$$B(i,j) =$$

$$\begin{cases} 0 \text{ if } i = j \\ \min_k \{B(i,k) + B(k+1,j) + r_i c_k c_j \end{cases}$$

$$B(i,j) =$$

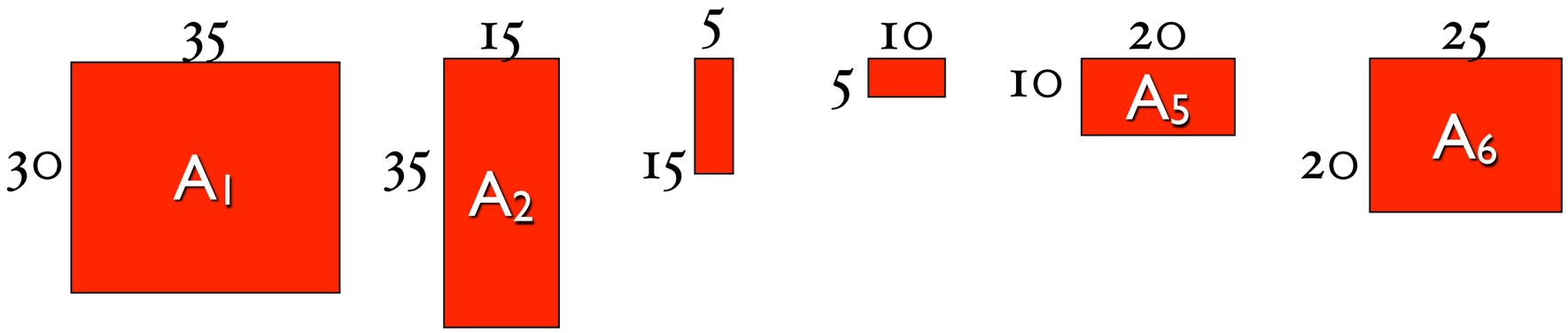
$$\begin{cases} 0 \text{ if } i = j \\ \min_k \{B(i,k) + B(k+1,j) + r_i c_k c_j \end{cases}$$

which order to solve?

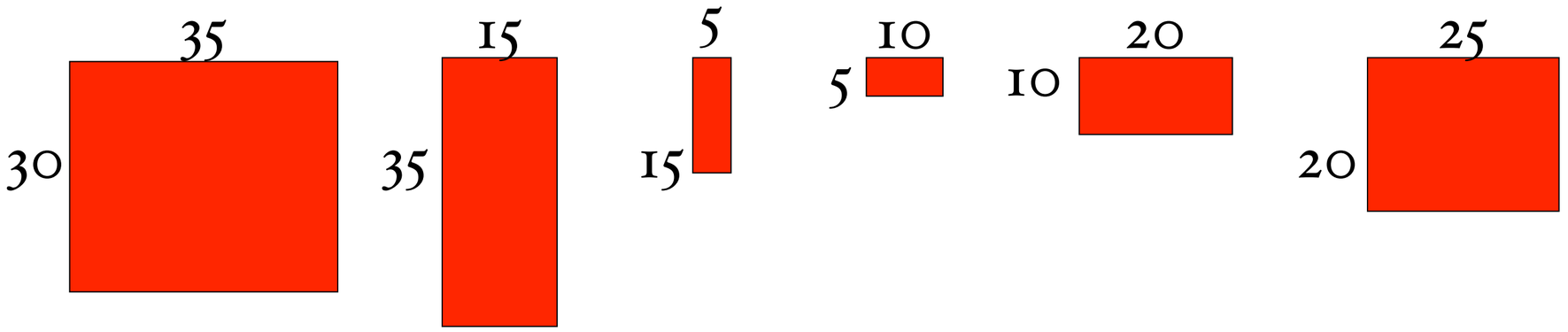


6						0
5					0	
4				0		
3			0			
2		0				
1	0					
	1	2	3	4	5	6

$$B(i,j) = \begin{cases} 0 \text{ if } i = j \\ \min_k \{B(i,k) + B(k+1,j) + r_i c_k c_j \end{cases}$$

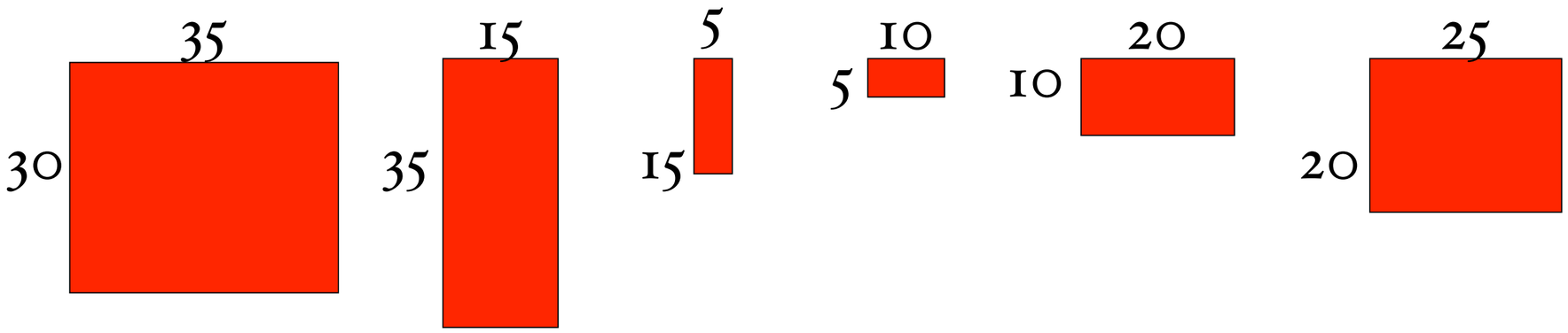


$$B(1,2) =$$



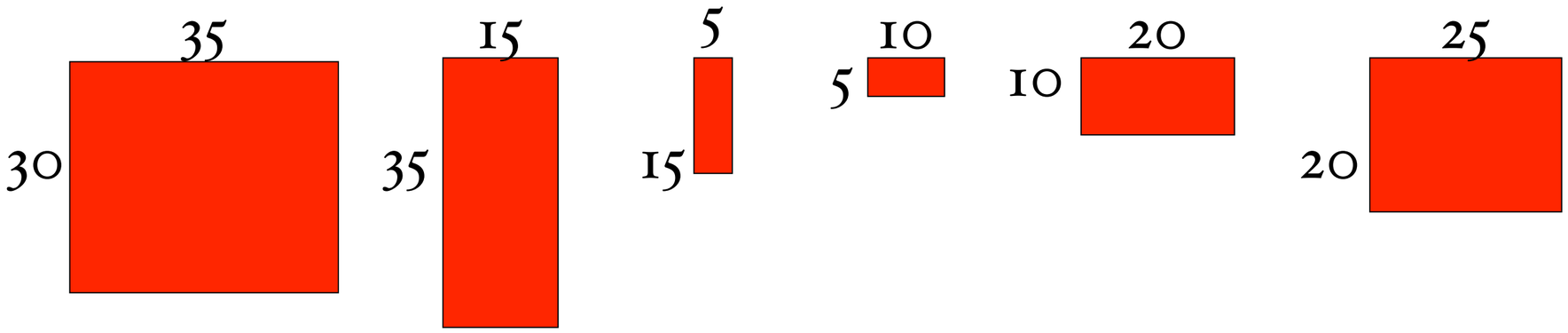
6					10*20*25 = 5000	0
5				5*10*20 = 1000	0	
4			15*5*10 = 750	0		
3		35*15*5 = 2625	0			
2	30*35*15 = 15750	0				
1	0					
	1	2	3	4	5	6

$$B(i,j) = \begin{cases} 0 \text{ if } i = j \\ \min_k \{B(i,k) + B(k+1,j) + r_i c_k c_j \end{cases}$$



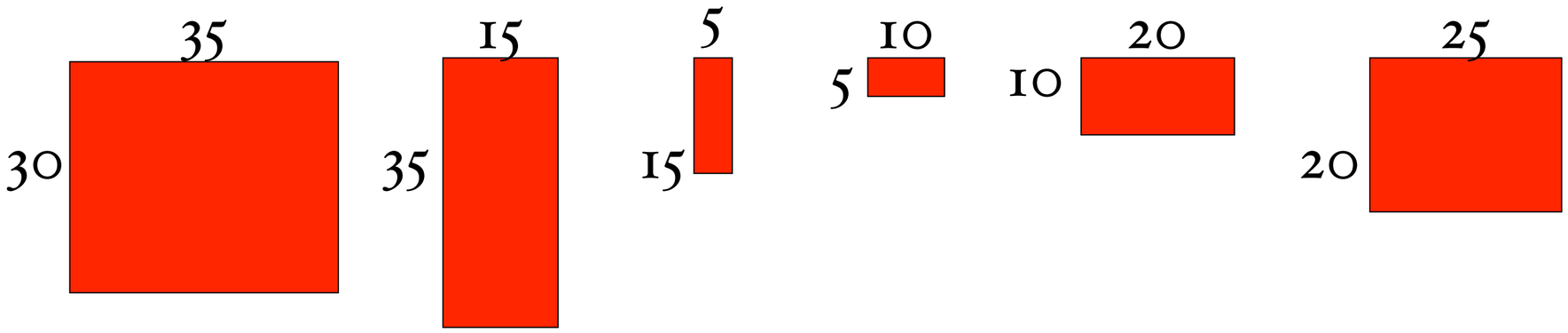
3		35*15*5 = 2625	0
2	30*35*15 = 15750	0	
1	0		
	1	2	3	4	5	6

$$B(i,j) = \begin{cases} 0 \text{ if } i = j \\ \min_k \{B(i,k) + B(k+1,j) + r_i c_k c_j \end{cases}$$



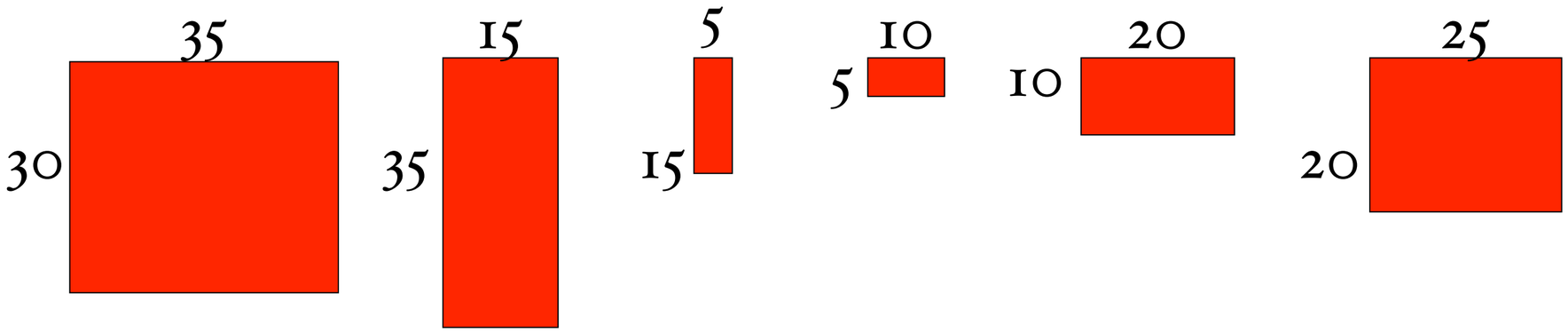
6		10500	5375	3500	10*20*25 = 5000	0
5	11875	7125	2500	5*10*20 = 1000	0	
4	9375	4375	15*5*10 = 750	0		
3	7875	35*15*5 = 2625	0			
2	30*35*15 = 15750	0				
1	0					
	1	2	3	4	5	6

$$B(i,j) = \begin{cases} 0 \text{ if } i = j \\ \min_k \{B(i,k) + B(k+1,j) + r_i c_k c_j \end{cases}$$



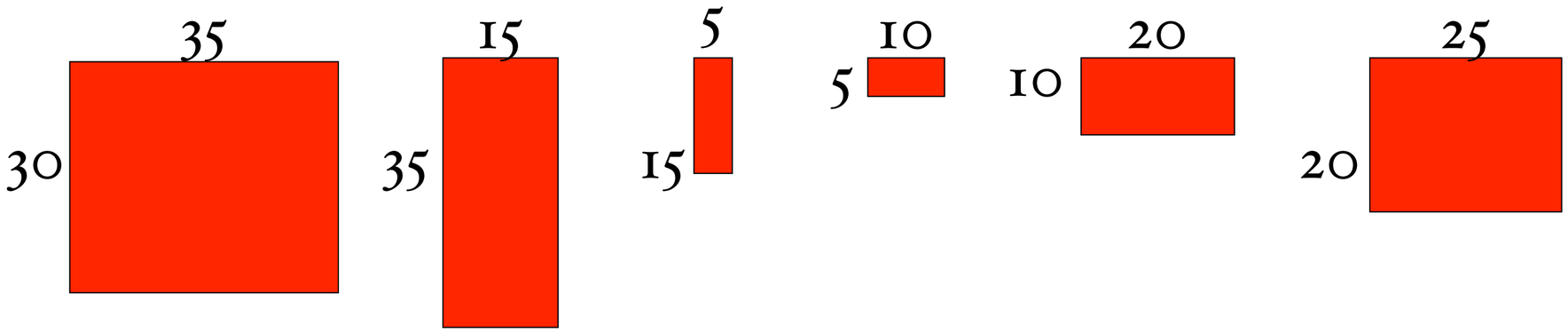
6

$$C(1,6) = \min \begin{cases} k=1 & C(1,1) + C(2,6) + r_1 c_1 c_6 \\ k=2 & C(1,2) + C(3,6) + r_1 c_2 c_6 \\ k=3 & C(1,3) + C(4,6) + r_1 c_3 c_6 \\ k=4 & C(1,4) + C(5,6) + r_1 c_4 c_6 \\ k=5 & C(1,5) + C(6,6) + r_1 c_5 c_6 \end{cases}$$



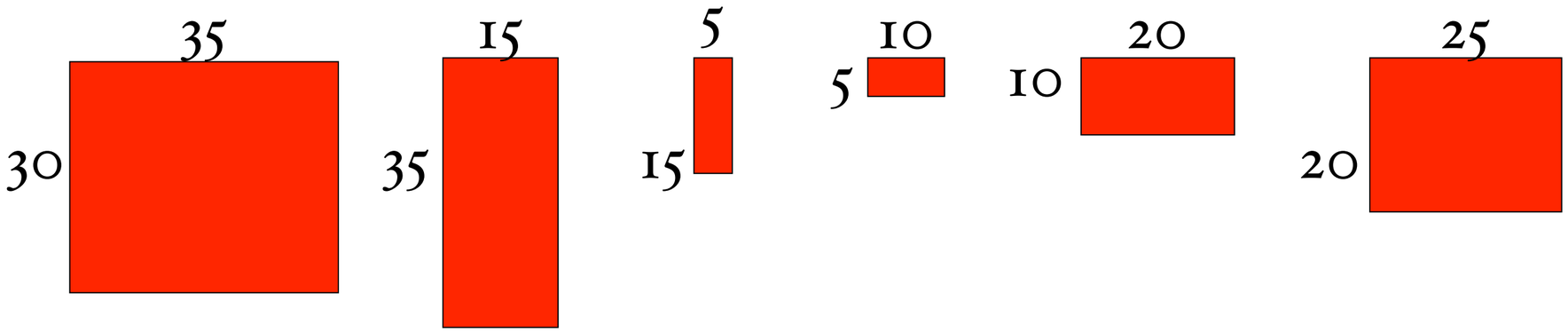
6 

$$C(1,6) = \min \begin{cases} k = 1 & 0 + 10500 + 30 \cdot 35 \cdot 25 \\ k = 2 & 15750 + 5375 + 30 \cdot 15 \cdot 25 \\ k = 3 & 7875 + 3500 + 30 \cdot 5 \cdot 25 \\ k = 4 & 9375 + 5000 + 30 \cdot 10 \cdot 25 \\ k = 5 & 11875 + 0 + 30 \cdot 20 \cdot 25 \end{cases}$$

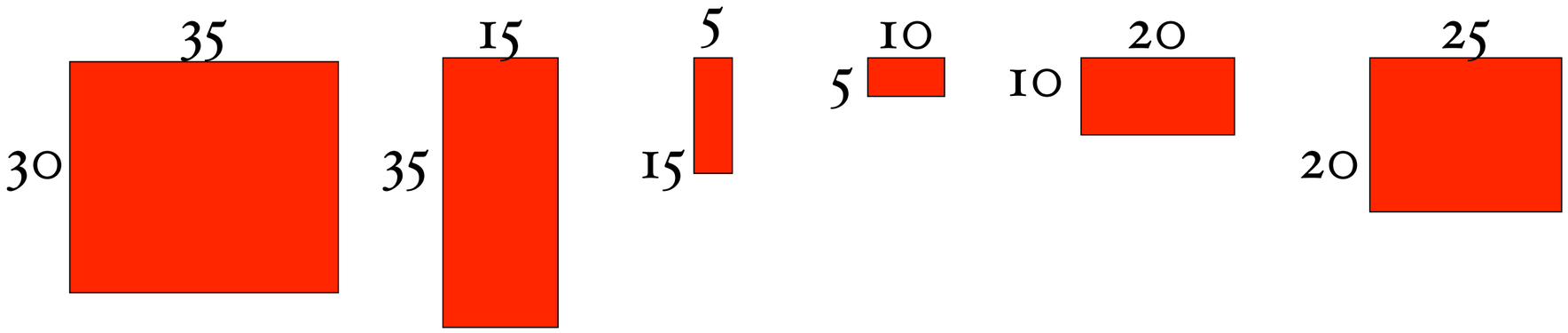


6 

$$C(1,6) = \min \begin{cases} k=1 & 0 + 10500 + 26250 \\ k=2 & 15750 + 5375 + 11250 \\ k=3 & 7875 + 3500 + 3750 \\ k=4 & 9375 + 5000 + 7500 \\ k=5 & 11875 + 0 + 15000 \end{cases}$$



6	15125 3	10500	5375	3500 ★	10*20*25 = 5000	0
5	11875	7125	2500	5*10*20 = 1000	0	
4	9375	4375	15*5*10 = 750	0		
3	7875 ★	35*15*5 = 2625	0			
2	30*35*15 = 15750	0				
1	0					
	1	2	3	4	5	6



6	15125 3	10500	5375	3500 ★	10*20*25 = 5000	0
5	11875	7125	2500	5*10*20 = 1000 ★	0	
4	9375	4375	15*5*10 = 750	0		
3	7875 ★	35*15*5 = 2625 ★	0			
2	30*35*15 = 15750	0				
1	0					
	1	2	3	4	5	6

matrix-chain-mult(p)

initialize array m[x,y] to zero

matrix-chain-mult(p)

initialize array m[x,y] to zero

starting at diagonal, working towards upper-left

compute m[i,j] according to

$$\begin{cases} 0 \text{ if } i = j \\ \min_k \{B(i,k) + B(k+1,j) + r_i c_k c_j \end{cases}$$

running time?

initialize array m[x,y] to zero

starting at diagonal, working towards upper-left

compute m[i,j] according to

$$\begin{cases} 0 \text{ if } i = j \\ \min_k \{B(i,k) + B(k+1,j) + r_i c_k c_j \end{cases}$$