L10

4102

Feb 23 2016

abhi shelat

Dynamic programming: matrix chains, seam carve

LUNCH

2 pages

Diagrams

February 2016

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24 H3	25	26	27
28	29	1	2 H4	3	4 midterm	

February 2016

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	

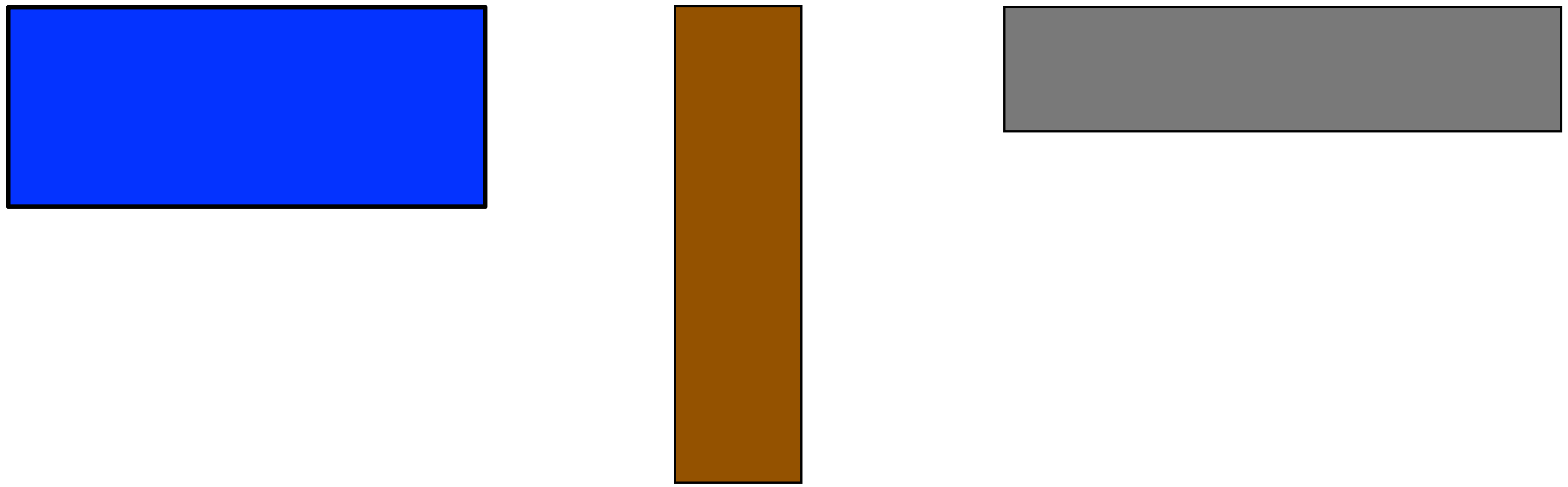
HW3

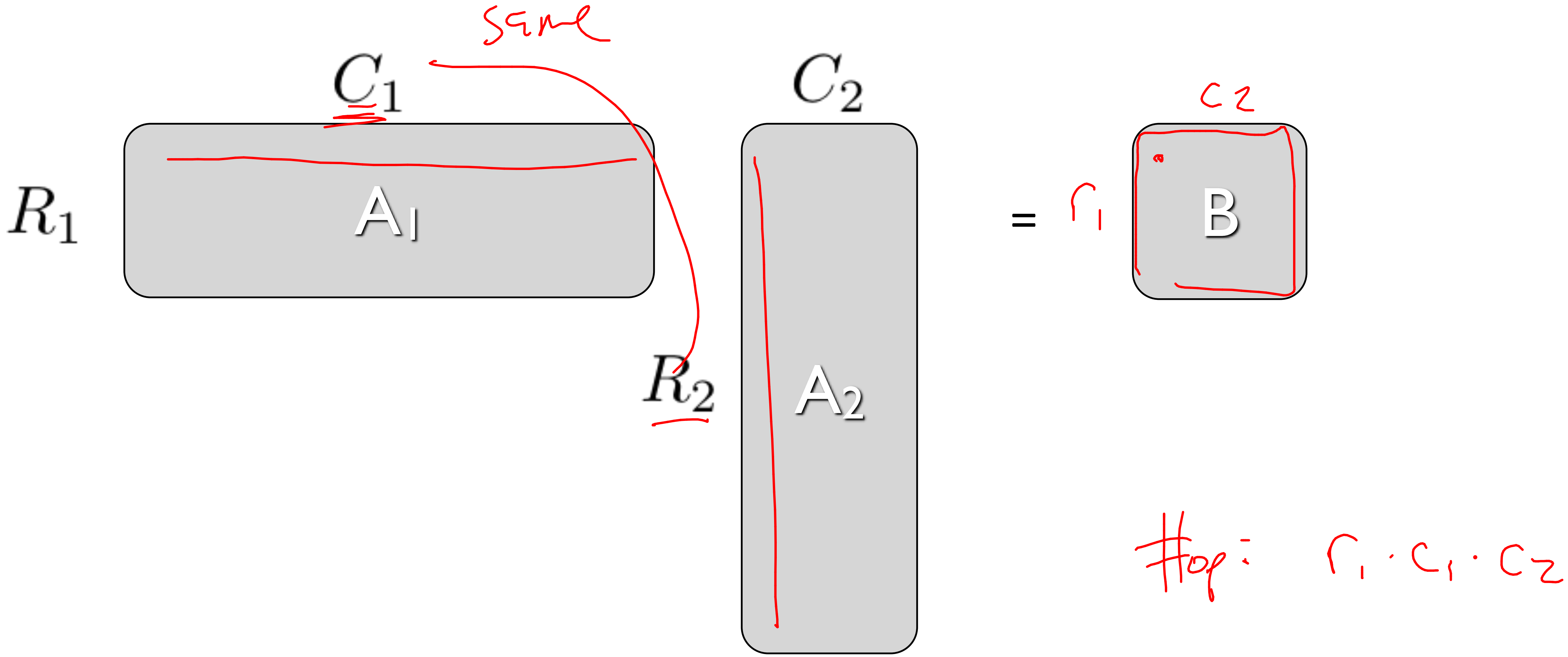
HW4

Spring break!!

midterm due

Matrix



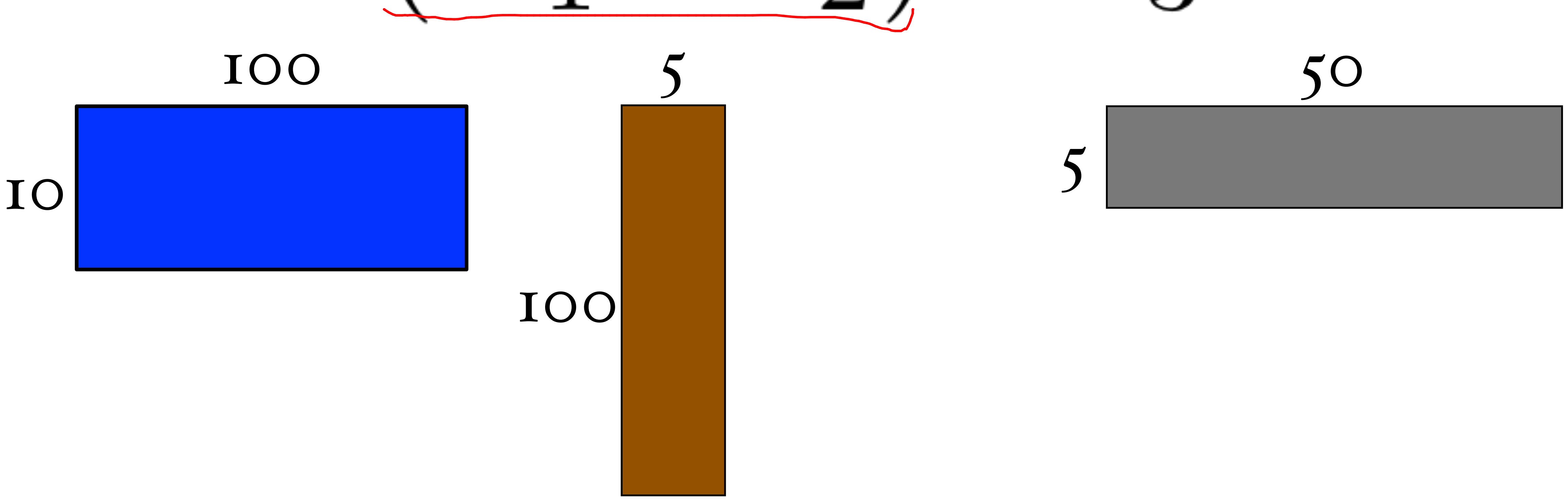


same
C_1
C_2
c_2
R_1
A_1
R_2
A_2
=
r_1
B
#op: $r_1 \cdot c_1 \cdot c_2$

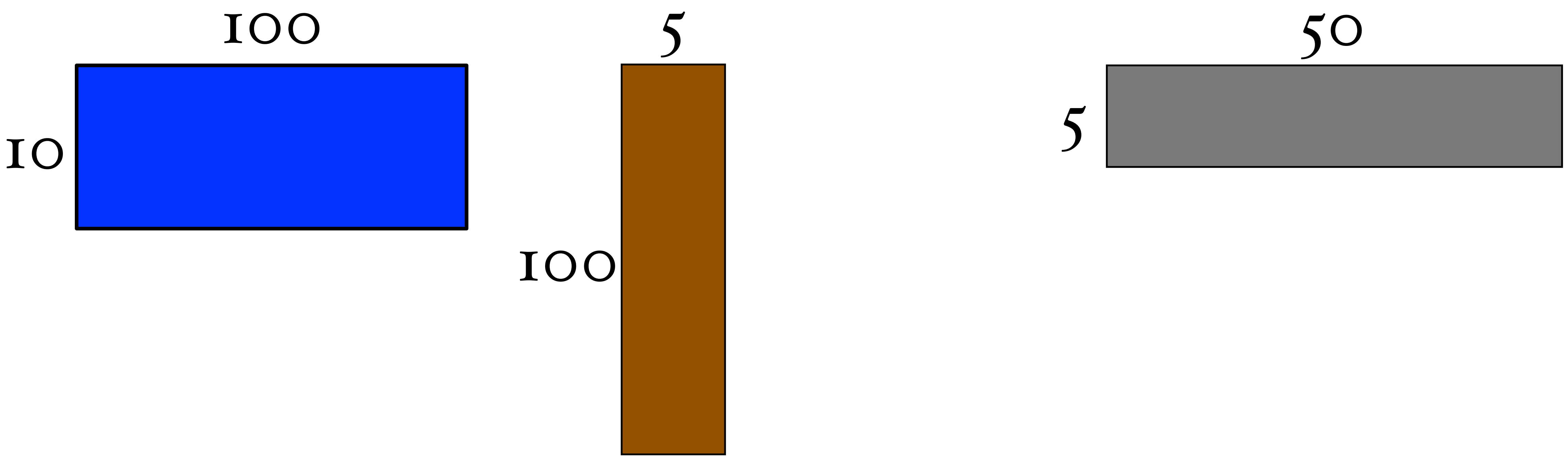
$$A_1 \cdot A_2 \cdot A_3$$

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

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



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

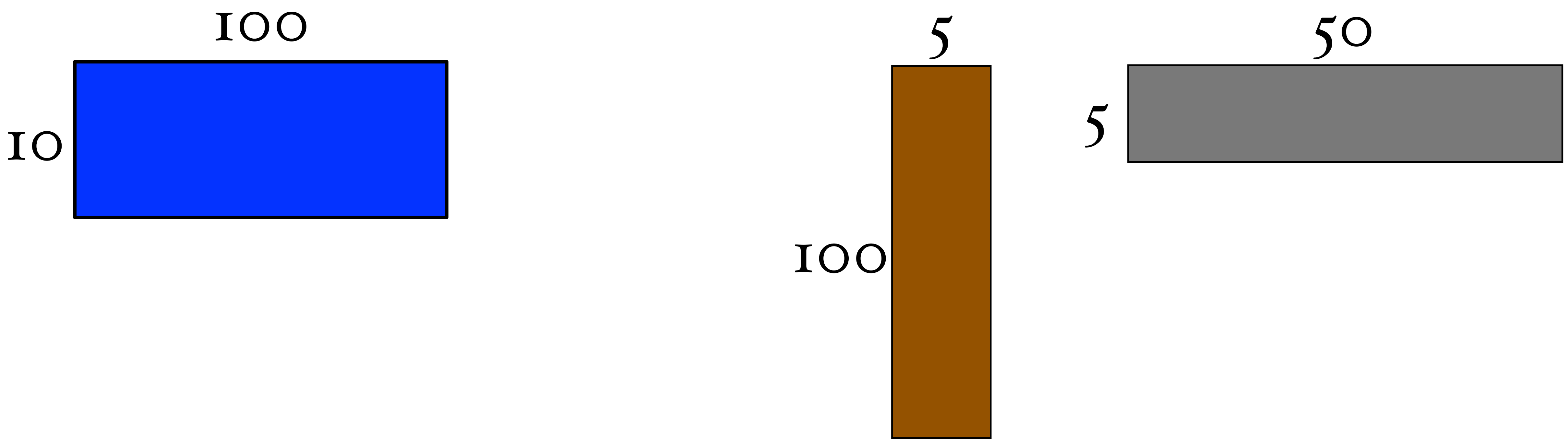


$$10 \cdot 100 \cdot 5 + 10 \cdot 5 \cdot 50$$

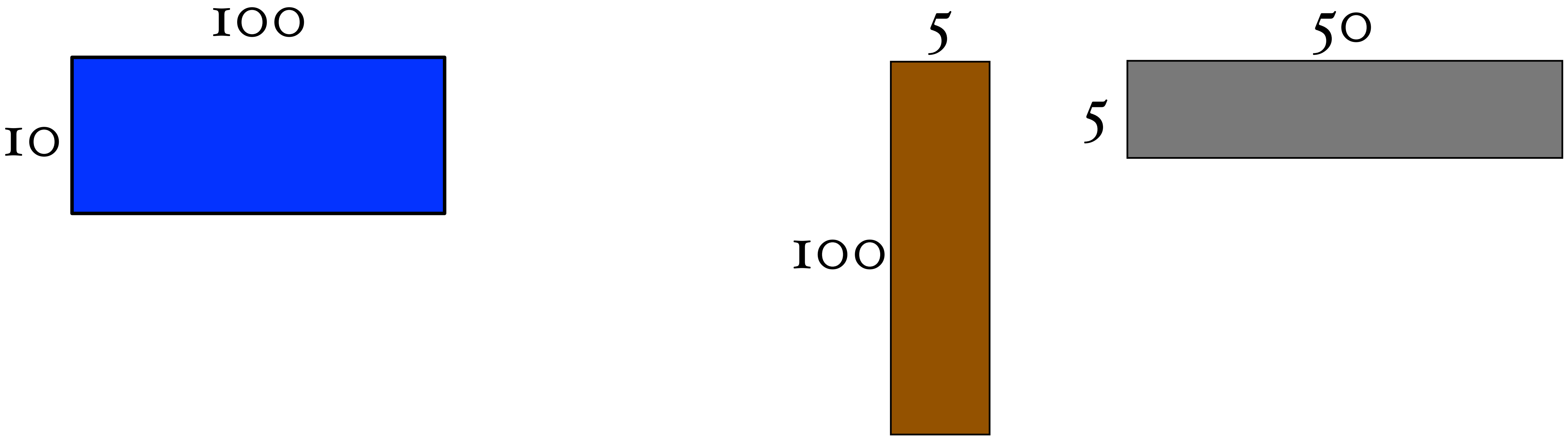
operations

7500

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



$$A_1 \cdot A_2 \cdot A_3$$



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

operations

75,000

ORDER MATTERS

(for efficiency)

HOW MANY WAYS TO

$$A_1 A_2 A_3 \ldots A_n$$

P(n): # of ways to multiply the n matricies.

$$P(n) = P(1) \cdot P(n-1) + P(2) \cdot P(n-2) + P(3) P(n-3) + \ldots + P(n-1) P(1)$$

$$= \sum_{i=1}^{n-1} P(i) P(n-i) \approx 4^n$$

HOW MANY WAYS TO

$$A_1 \; A_2 A_3 \ldots A_n$$

$$A_1 A_2 A_3 \ldots A_n$$

HOW MANY WAYS TO

$$A_1 \; A_2 A_3 \ldots A_n$$

$$A_1 A_2 \; A_3 \ldots A_n$$

$$A_1 A_2 A_3 \ldots A_n$$

HOW MANY WAYS TO

$$A_1 \; A_2 A_3 \ldots A_n$$

$$A_1 A_2 \; A_3 \ldots A_n$$

$$A_1 A_2 A_3 \; \ldots A_n$$

OPTIMAL WAY TO COMPUTE

$\ell \to$ optimal last step

$$\overset{c_1}{{}^{r_1}A_1}\ \overset{c_2}{{}^{r_2}A_2}\ A_3 \ldots \overset{c_\ell}{{}^{r_\ell}A_\ell} \cdot A_{\ell+1} \ldots A_n$$

$B[1,n]$: smallest # of operations needed to multiply $A_1 \ldots A_n$.

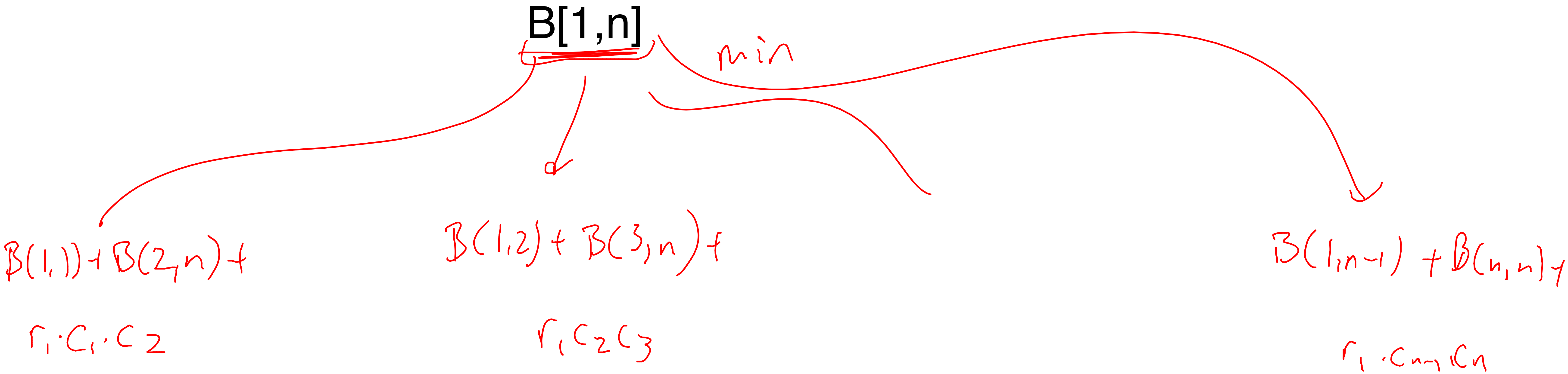
$B[1,n] = B[1,\ell] + B[\ell+1,n] + r_1 \cdot c_\ell \cdot c_{\ell+1}$ if ℓ is optimal

how many choices are there for ℓ ??

$\ell \in [1,n]$

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]

B[1,1]

B[2,n]

$R_1 C_1 C_n$

OPTIMAL WAY TO COMPUTE

$$A_1 A_2 A_3 A_4 \cdots A_n$$

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

$$B(i,i) = 0$$

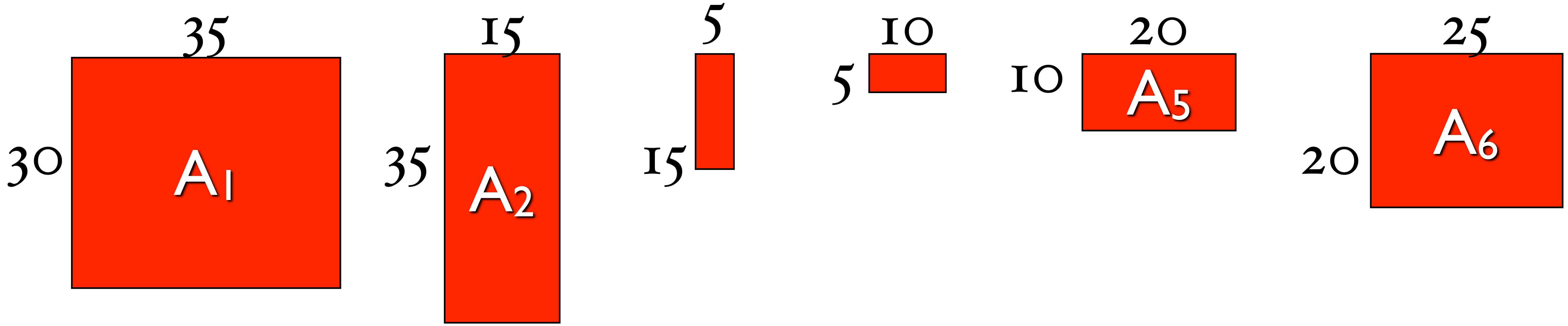
$$B(i,j) = \min_{k=i}^{j-1} \left\{ B(i,k) + B(k+1,j) + r_i c_k c_j \right.$$

prefix

suffix

→ # of operations to multiply the prefix & suffix

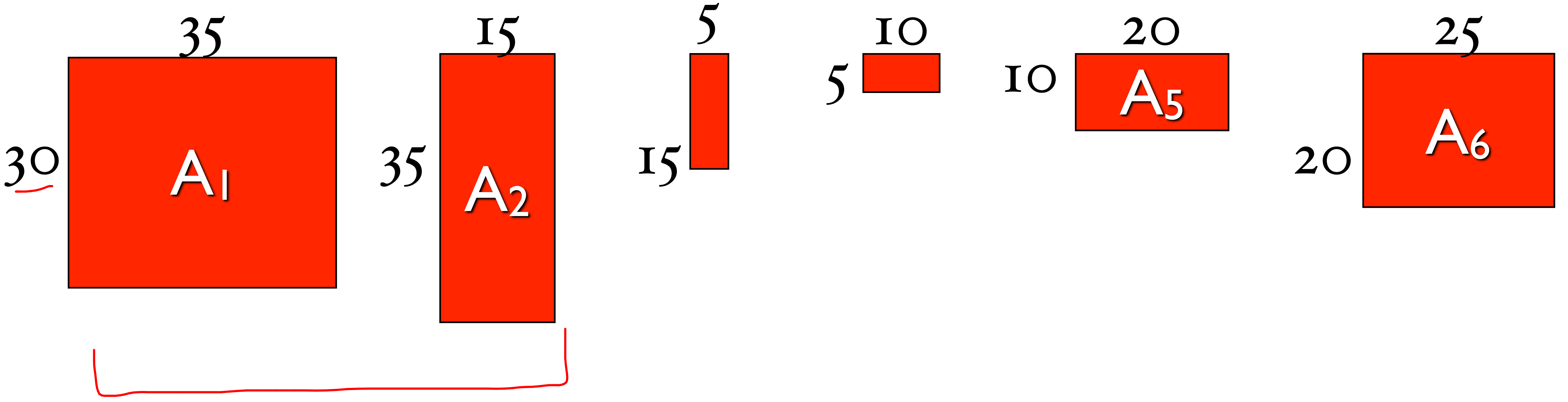
WHICH ORDER TO SOLVE?



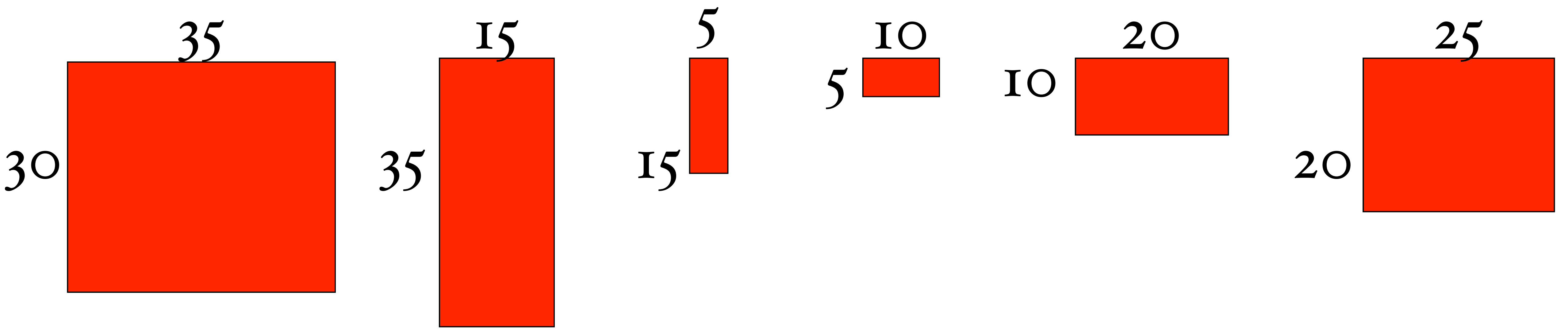
6						0
5					0	
4				0		
3			0			
2		0				
1	0					
	1	2	3	4	5	6

$$B(i,i) = 0$$

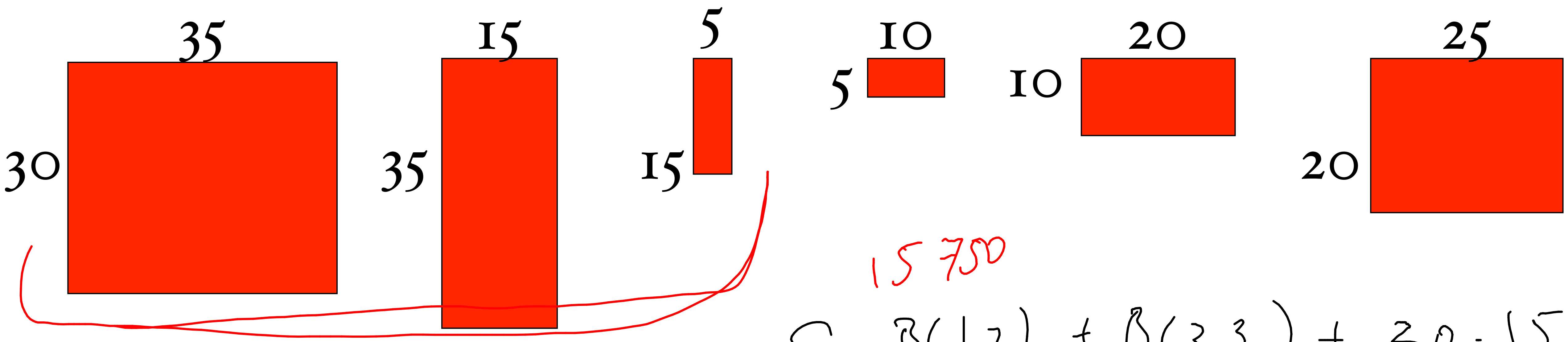
$$B(i,j) = \min_{k=i}^{j} \left\{ B(i,k) + B(k+1,j) + r_i c_k c_j \right.$$



$$B(1,2) = \text{Best}(1,1) + B(2,2) + 30 \cdot 35 \cdot 15 = 15750$$



	1	2	3	4	5	6
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					



15 750

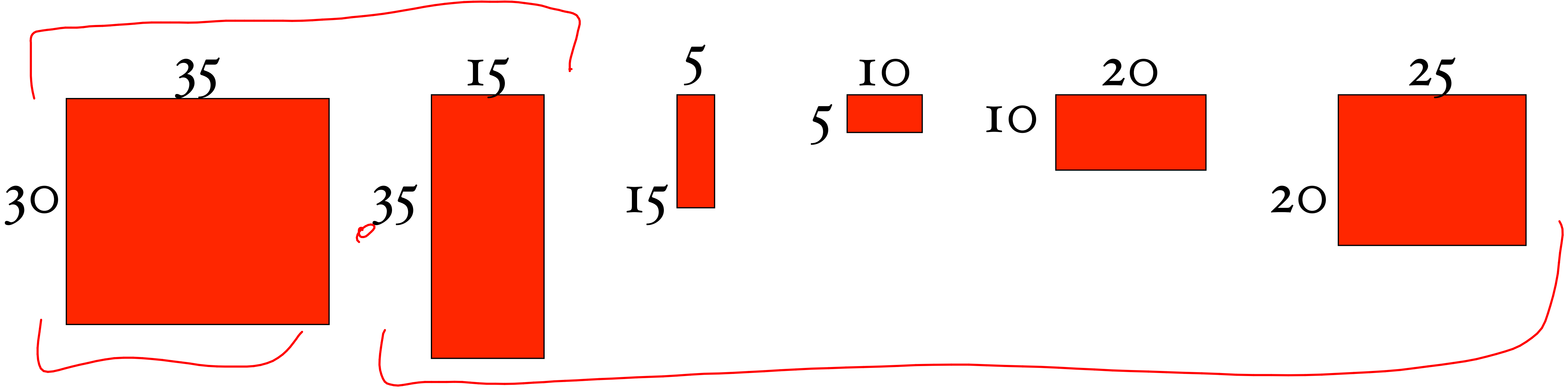
Best(1,3) = min

$$B(1,2) + B(3,3) + 30 \cdot 15 \cdot 5$$

$$B(1,1) + B(2,3) + 30 \cdot 35 \cdot 5$$

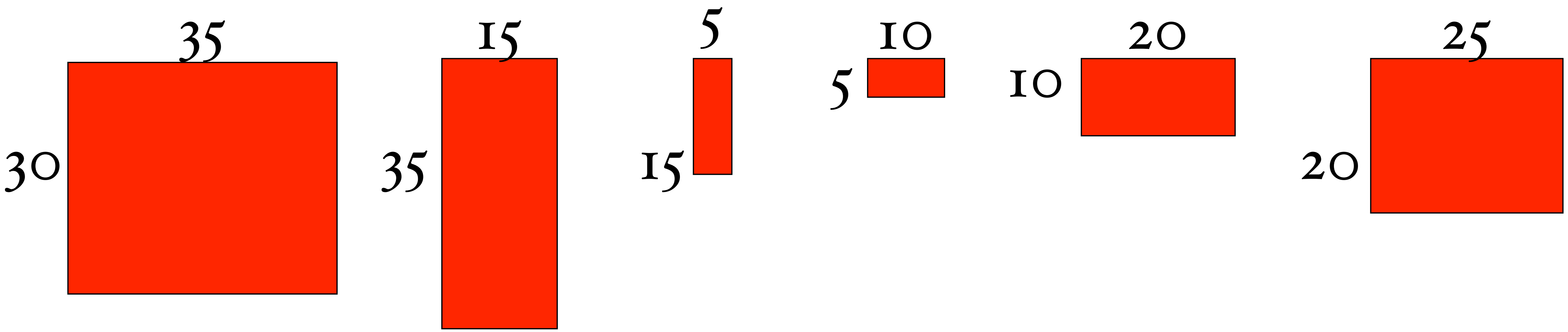
$$0 \quad 2625 + 5250 = 7875$$

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



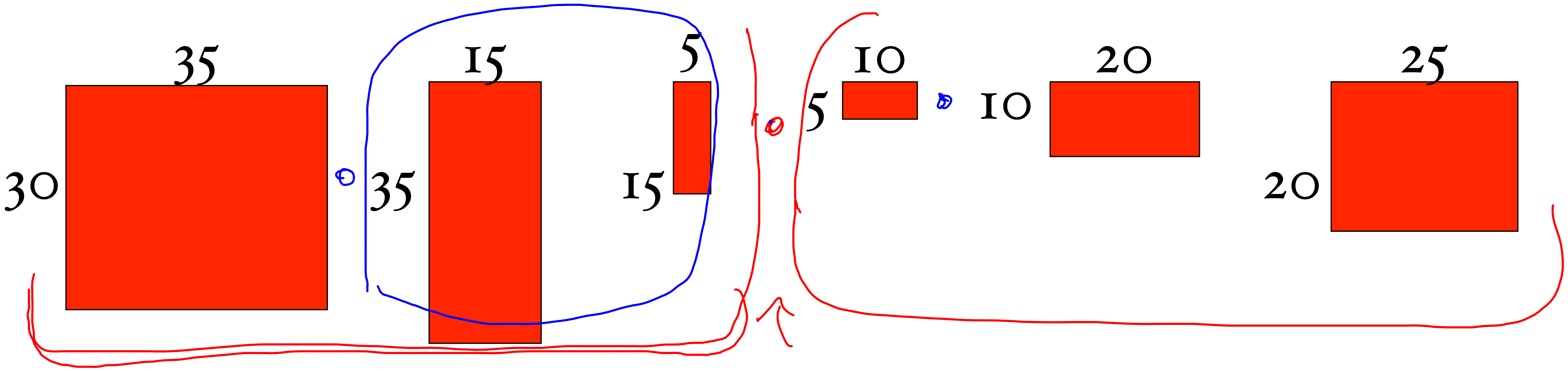
6		10500	5375	3500	10*20*25 = 5000	0
5	11875	7125	2500	5*10*20 = 1000	0	
4	9375	4375	15*5*10 = 750	0		
3	7875	35*15*5 = 2625	0			
2	30*35*15 = 15750	0				
1	0					
	1	2	3	4	5	6

$$B(i, j) =$$

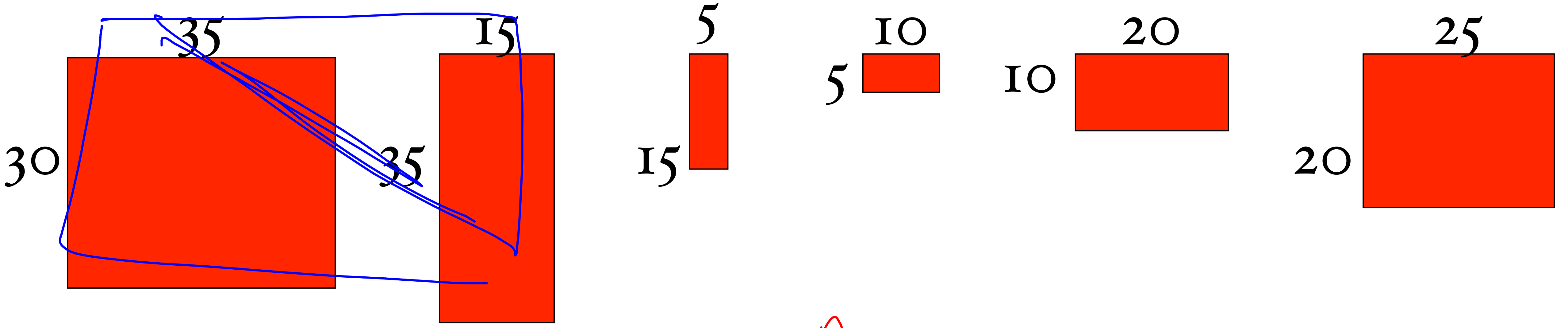


6 

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



6	15125	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



	1	2	3	4	5	6
6	15125	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					

MATRIX-CHAIN-MULT(P)

initialize array m[x,y] to zero

RUNNING TIME?

initialize array B[x,y] to zero

starting at diagonal, working towards upper-left

compute B[i,j] according to

$$B(i,i) = 0$$

$$B(i,j) = \min_{k=i}^{j-1} \left\{ B(i,k) + B(k+1,j) + r_i c_k c_j \right.$$

how many elements $\Theta(n^2)$

each loop takes $\Theta(n)$

$\Theta(n^3)$

⇒ there is a better way $\Theta(n \log n)$

PROBLEM: REDUCE IMAGE WIDTH

scaling: distortion

deleting column: distortion

delete the most invisible seam

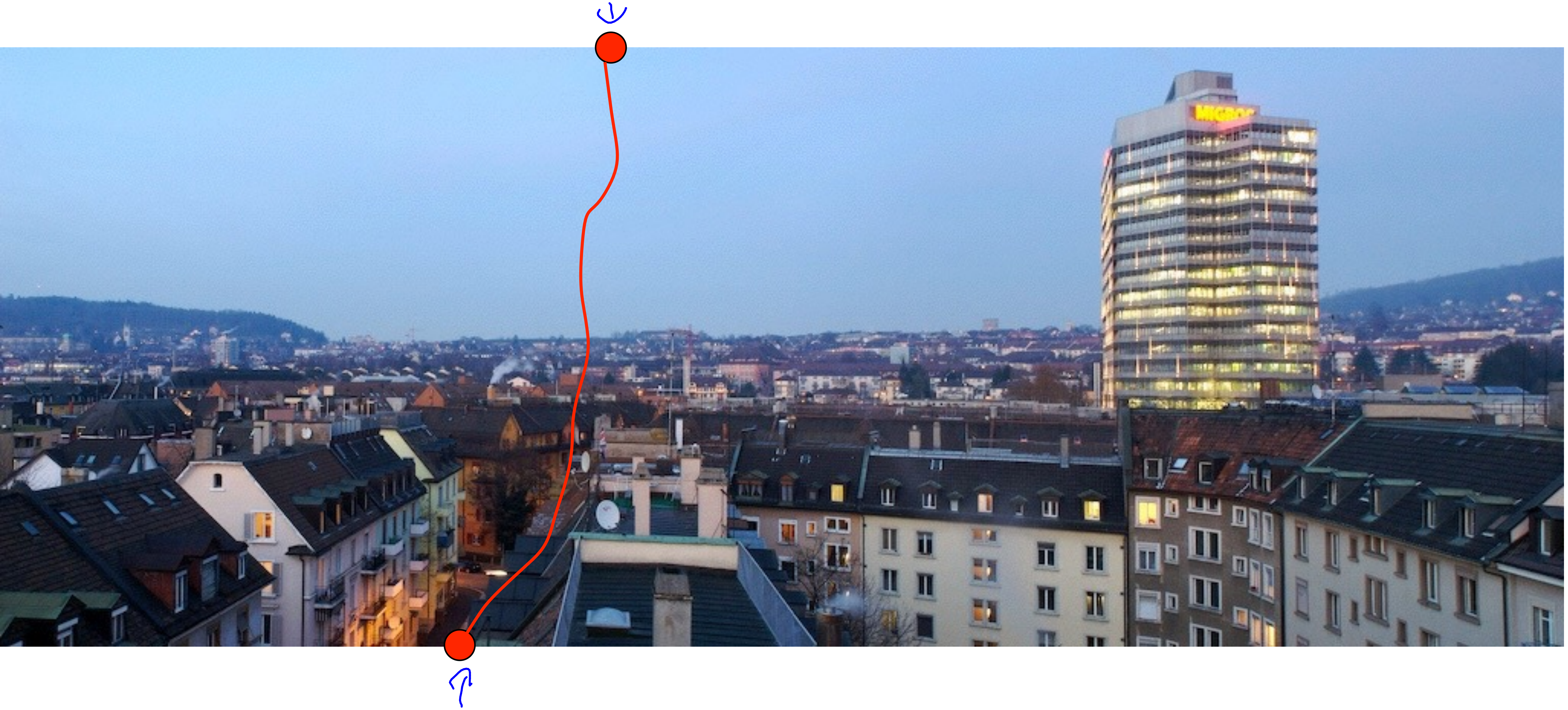
http://www.youtube.com/watch?v=qadw0BRKeMk

http://www.youtube.com/watch?v=qadw0BRKeMk

DEMO?

http://rsizr.com/

WHICH SEAM TO DELETE?

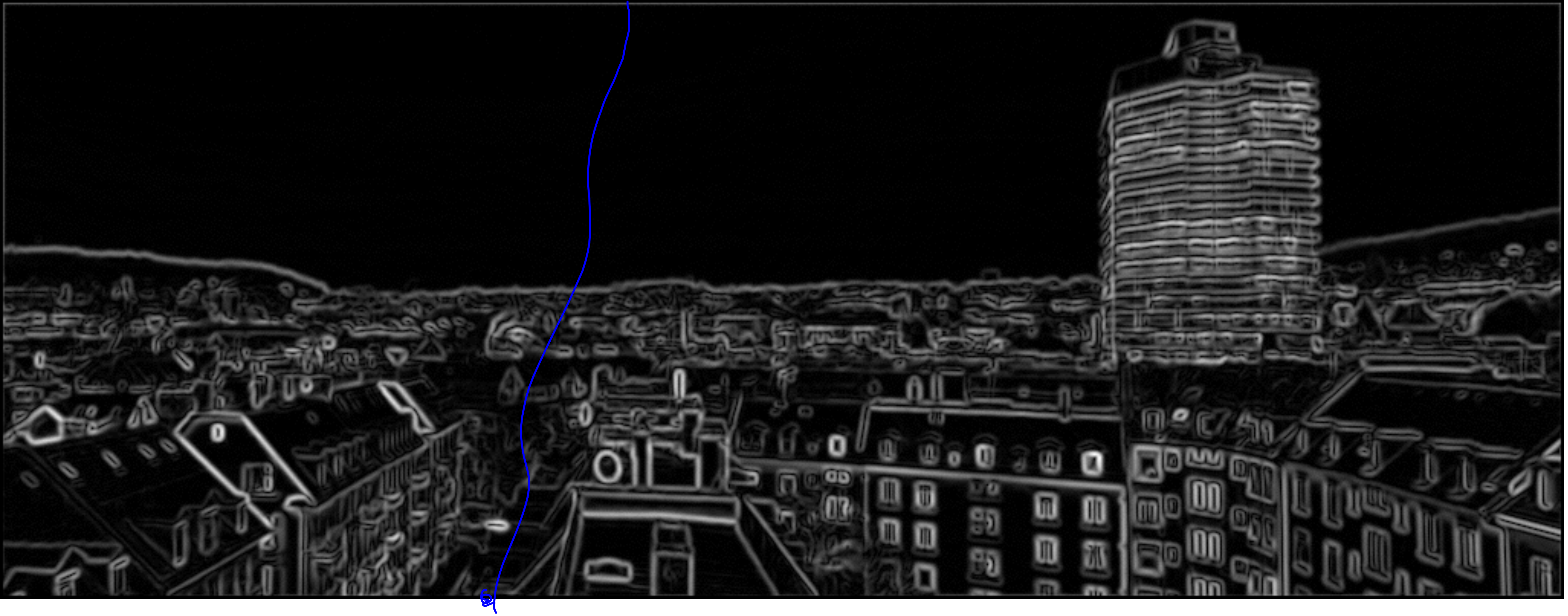


ENERGY OF AN IMAGE

$$e(\mathbf{I}) = \left|\frac{\partial}{\partial x}\mathbf{I}\right| + \left|\frac{\partial}{\partial y}\mathbf{I}\right|$$

"magnitude of gradient at a pixel"

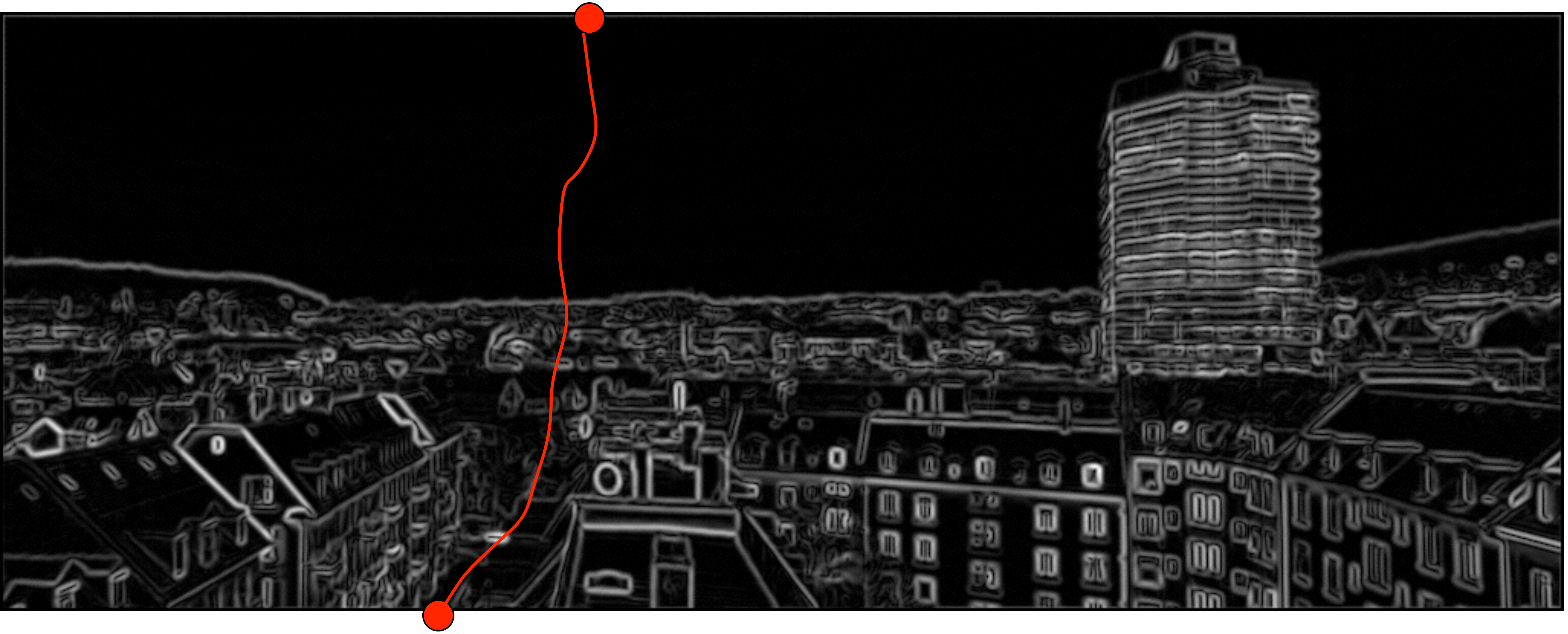
$$\frac{\partial}{\partial x} I_{x,y} = I_{x-1,y} - I_{x+1,y}$$



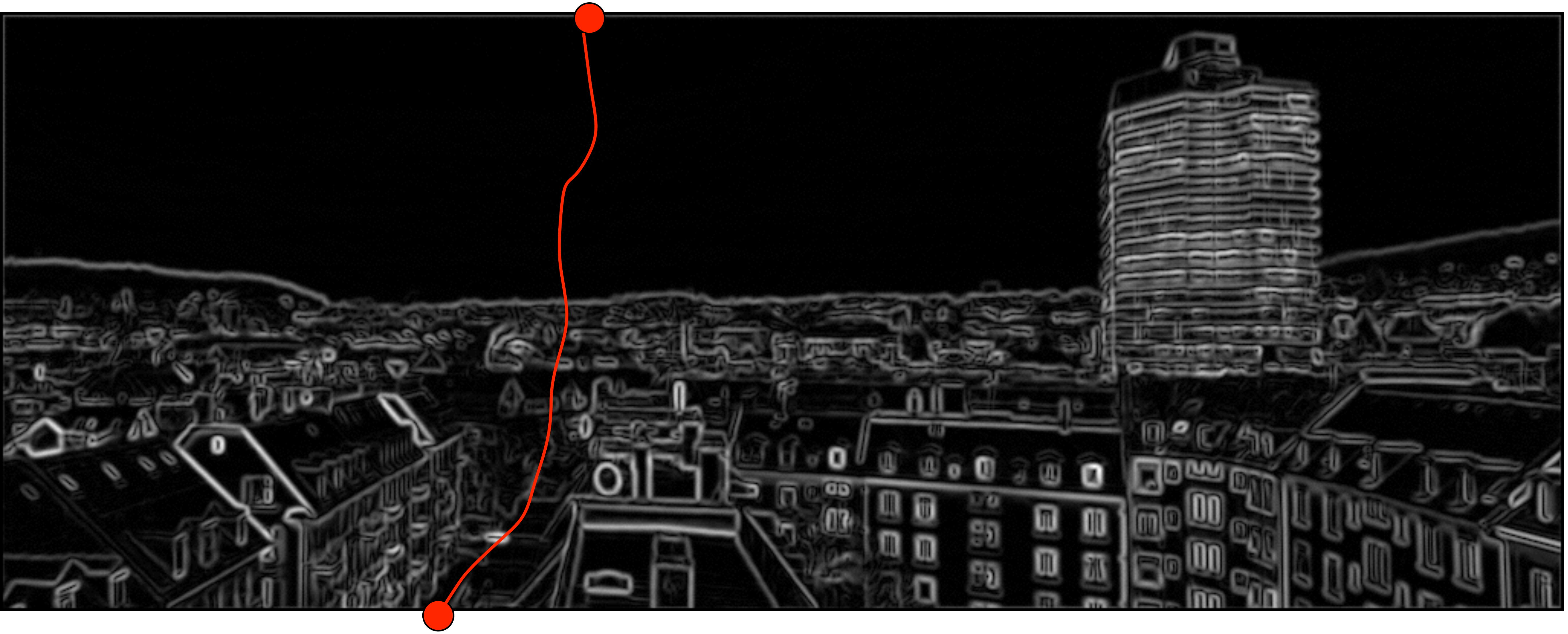
energy of sample image

thanks to Jason Lawrence for gradient software

BEST SEAM HAS LOWEST ENERGY



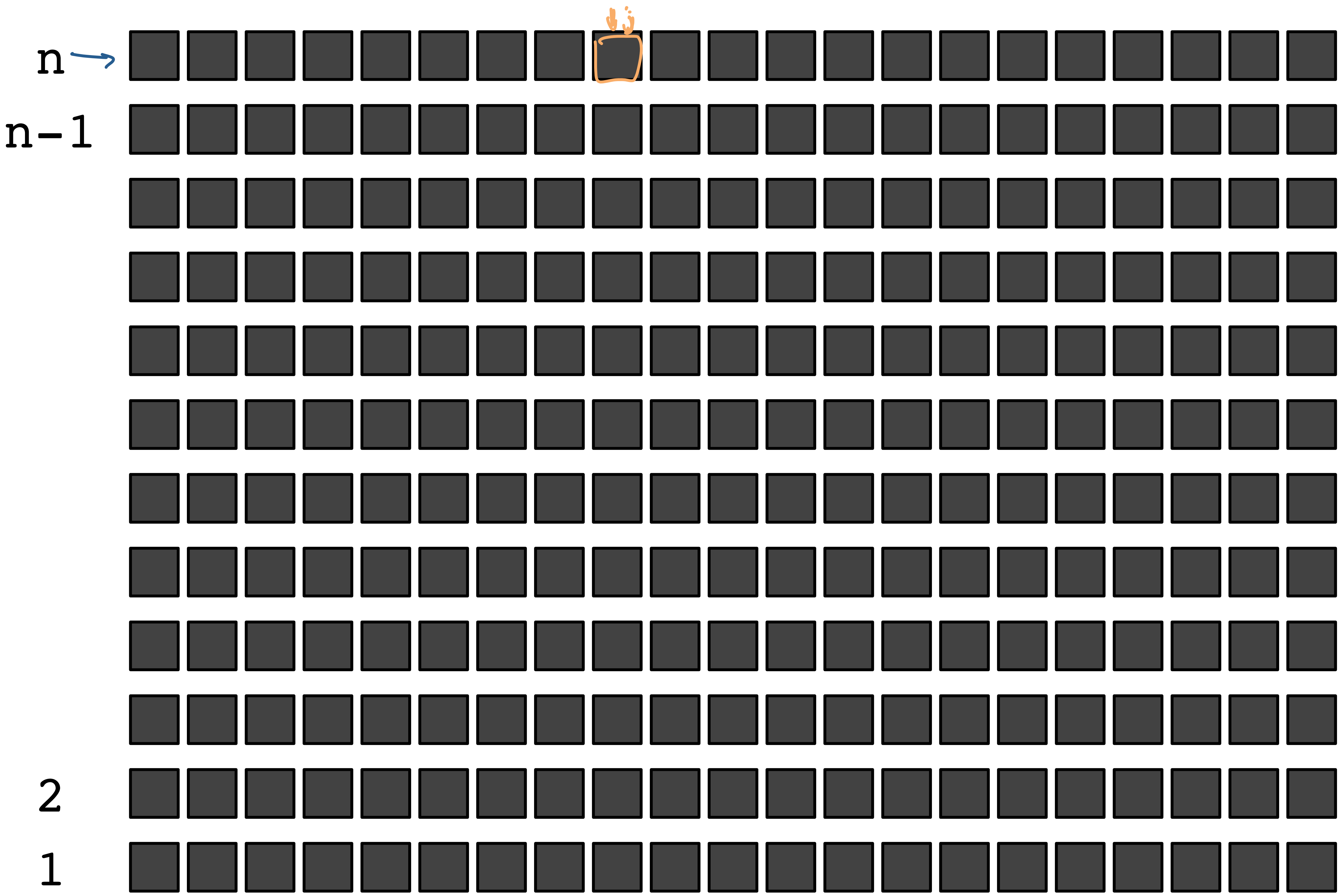
FINDING LOWEST ENERGY SEAM?



DEFINE A VARIABLE:

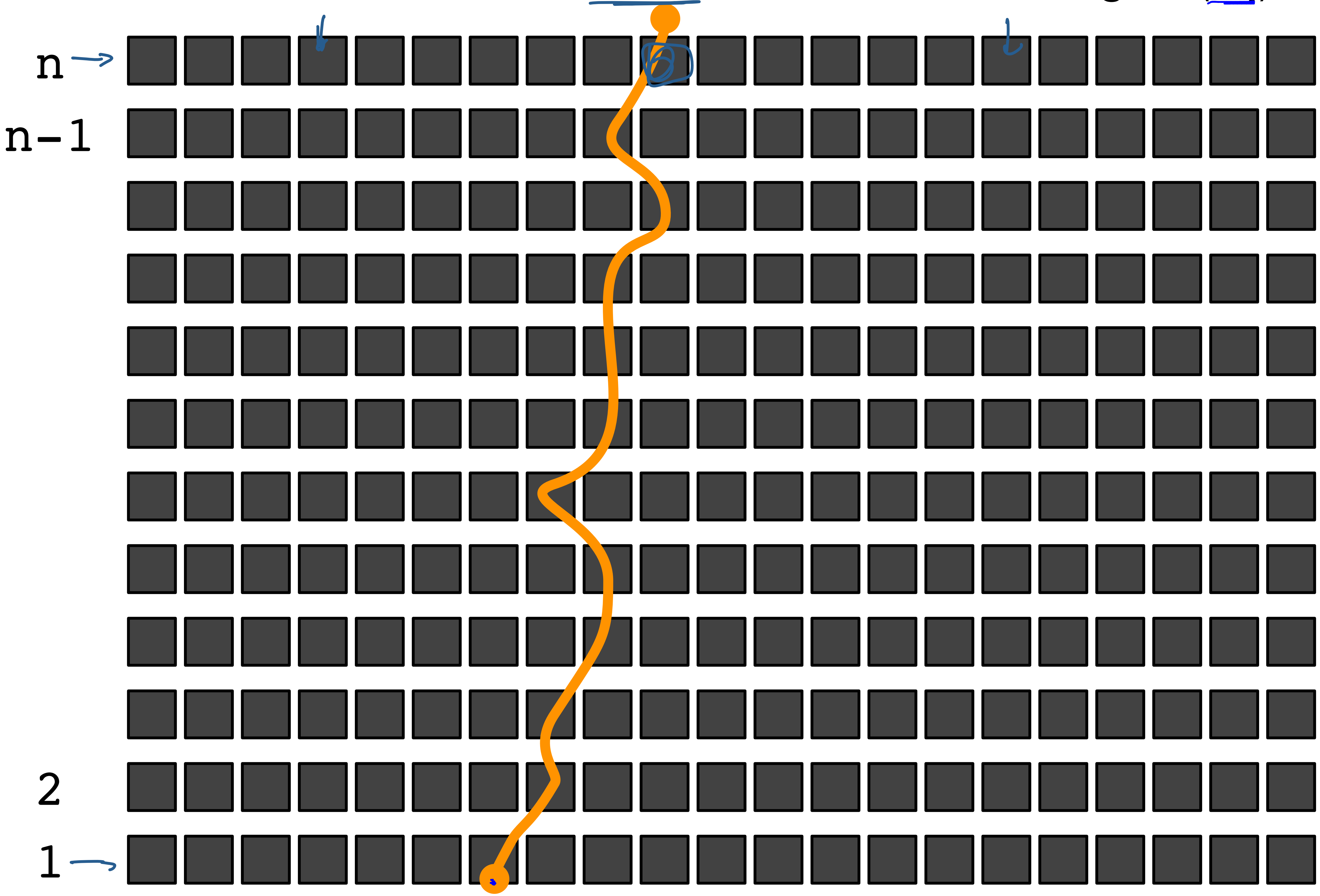
$S_i(j)$: least energy path from the bottom row to pixel (i, j)

definition: $S_n(j)$

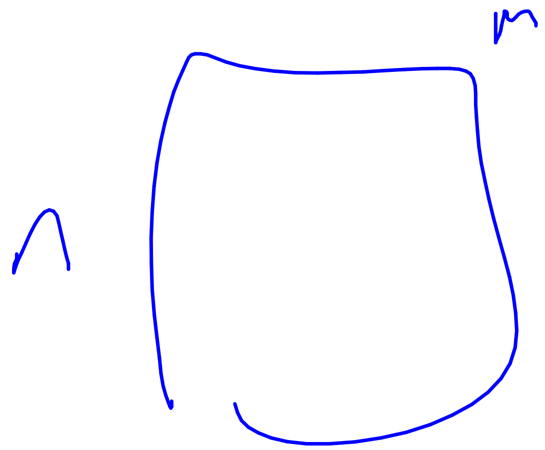


definition:

$S_n(j)$ best seam ending at (n,j)

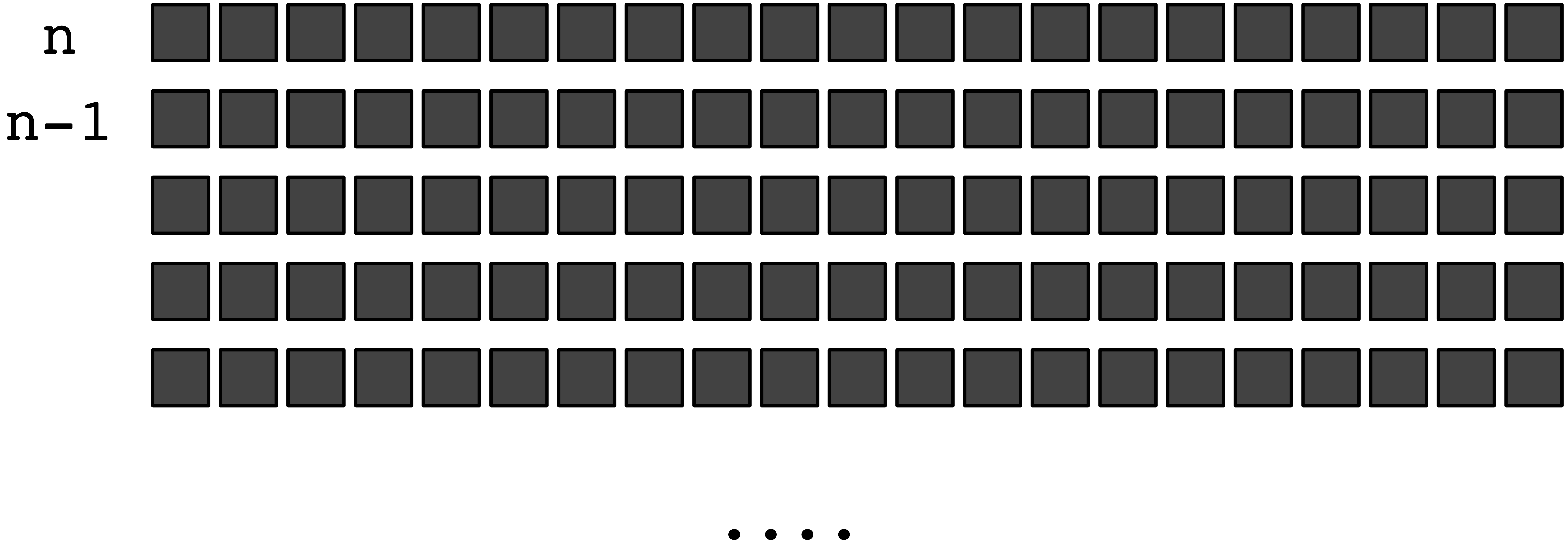


BEST SEAM TO DELETE HAS TO BE THE BEST AMONG

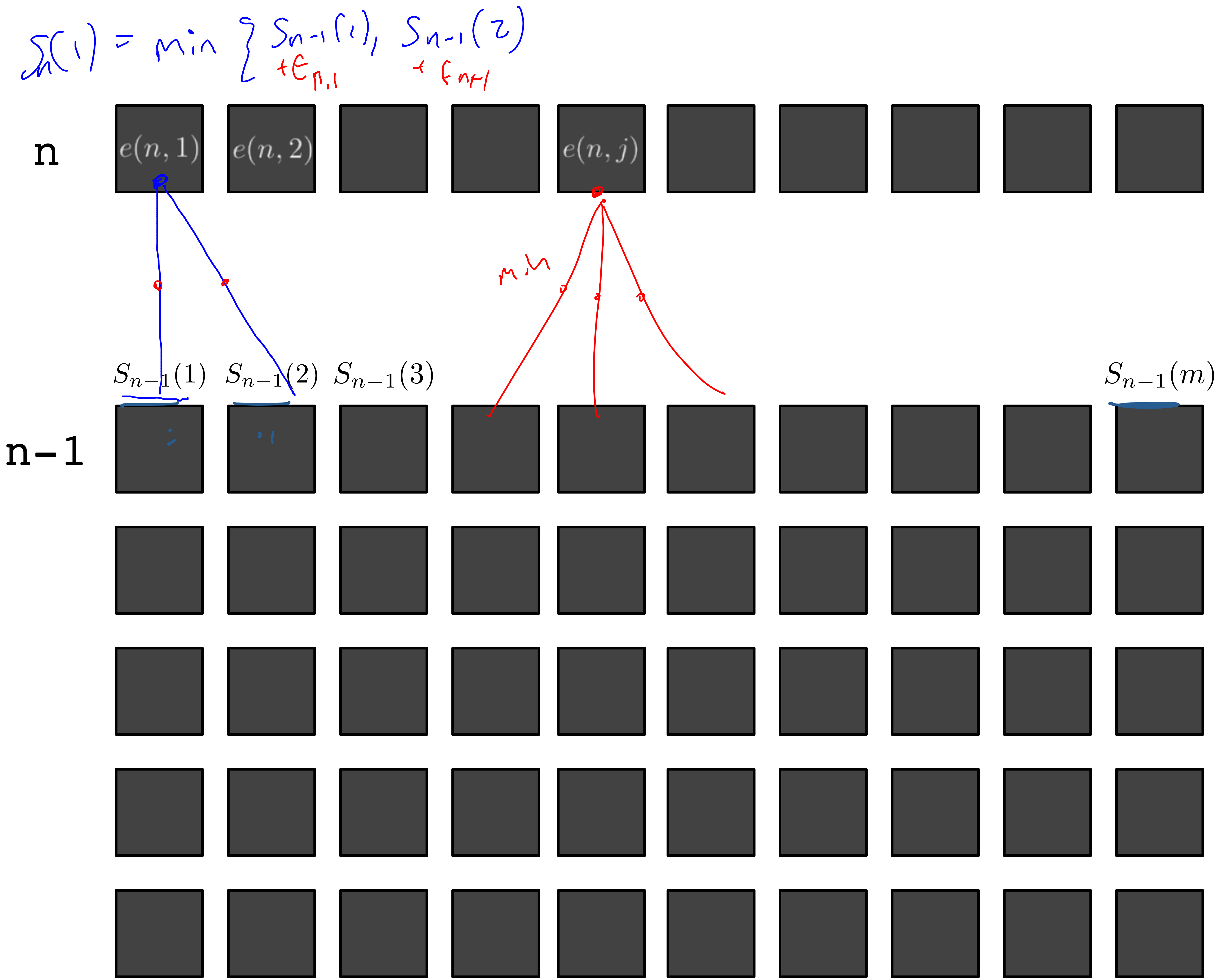


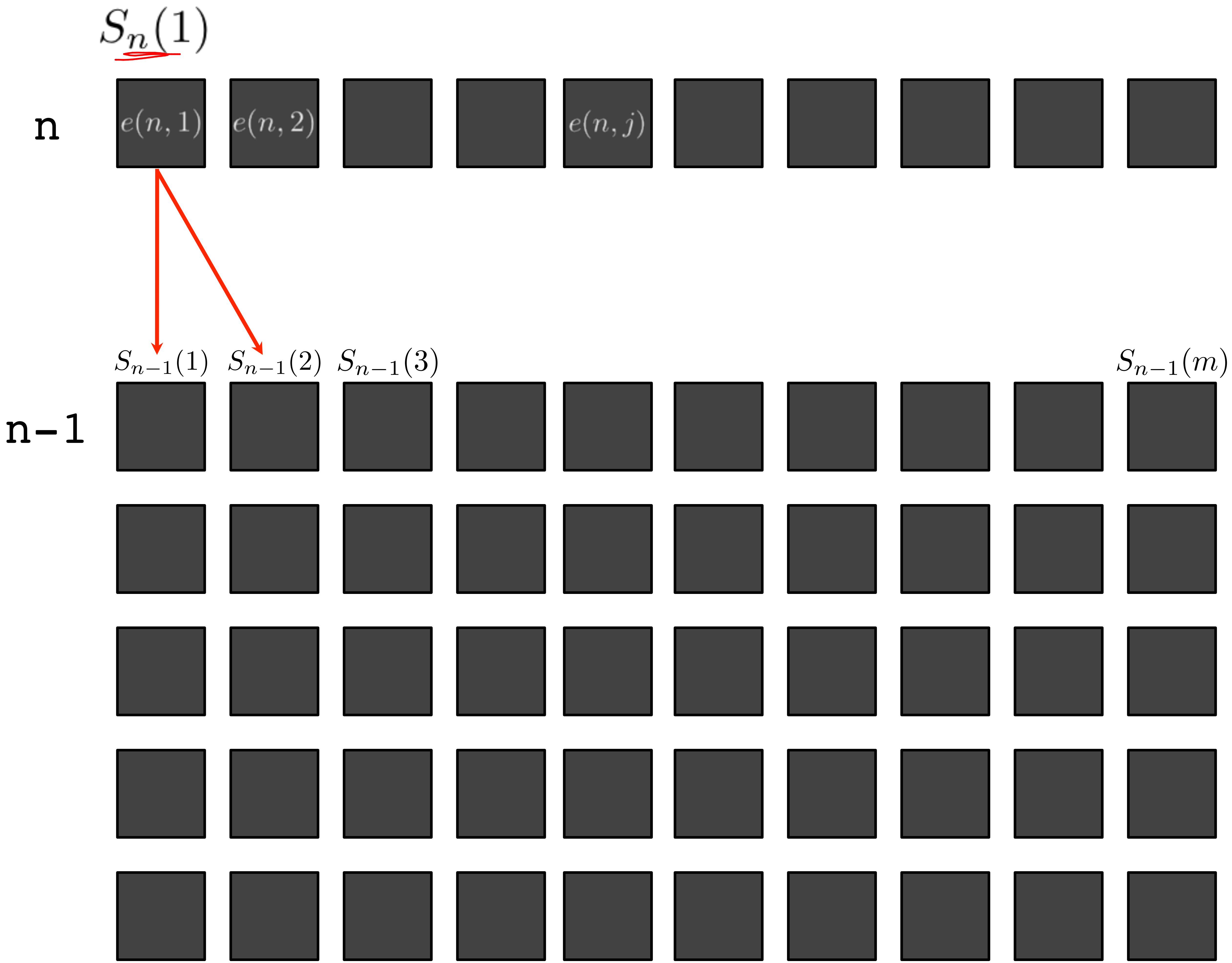
$$S_n(1), S_n(2), \ldots, S_n(m)$$

IDEA: COMPUTE + COMPARE

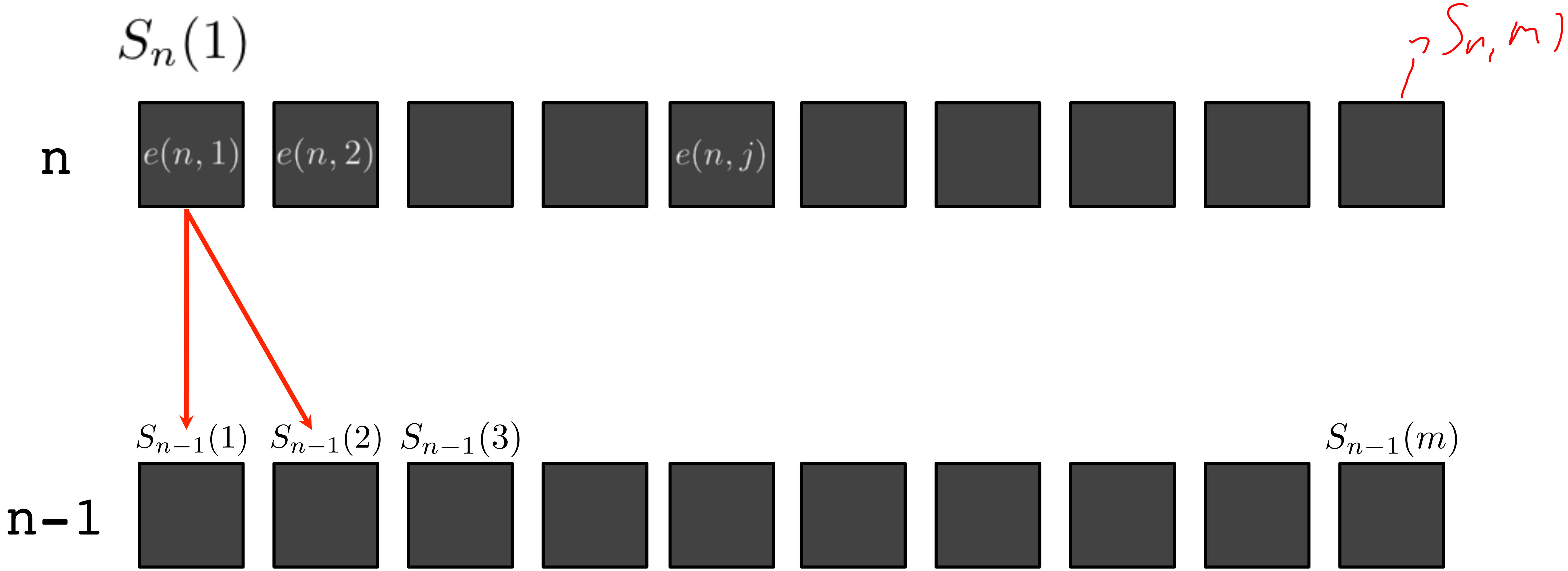


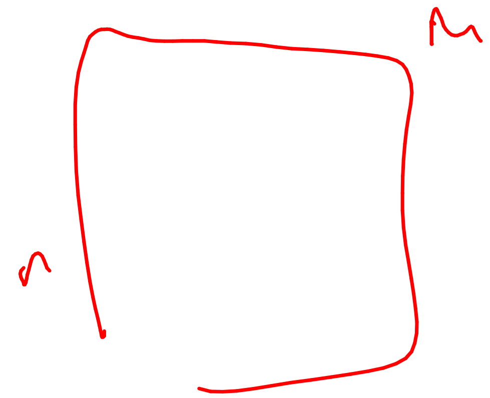
IMAGINE YOU HAVE THE SOLUTION TO THE FIRST N-1 ROWS


$S_n(1) = \min\{S_{n-1}(1) + e_{n,1}, S_{n-1}(2) + e_{n,1}\}$
n
n-1
$e(n,1)$
$e(n,2)$
$e(n,j)$
min
$S_{n-1}(1)$
$S_{n-1}(2)$
$S_{n-1}(3)$
$S_{n-1}(m)$

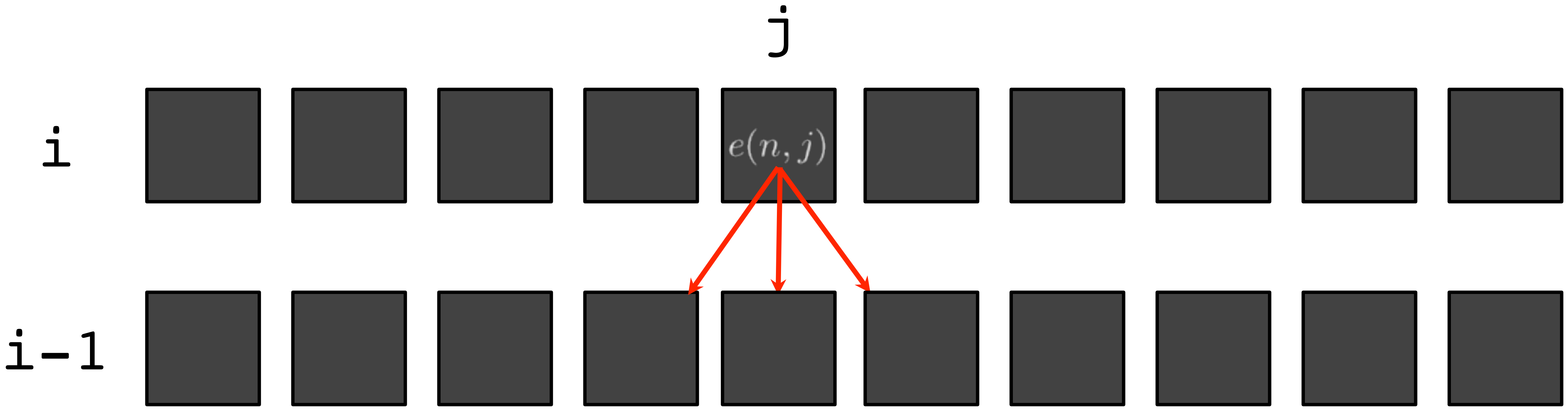


$S_n(1)$
n
$e(n,1)$
$e(n,2)$
$e(n,j)$
$S_{n-1}(1)$
$S_{n-1}(2)$
$S_{n-1}(3)$
$S_{n-1}(m)$
n-1

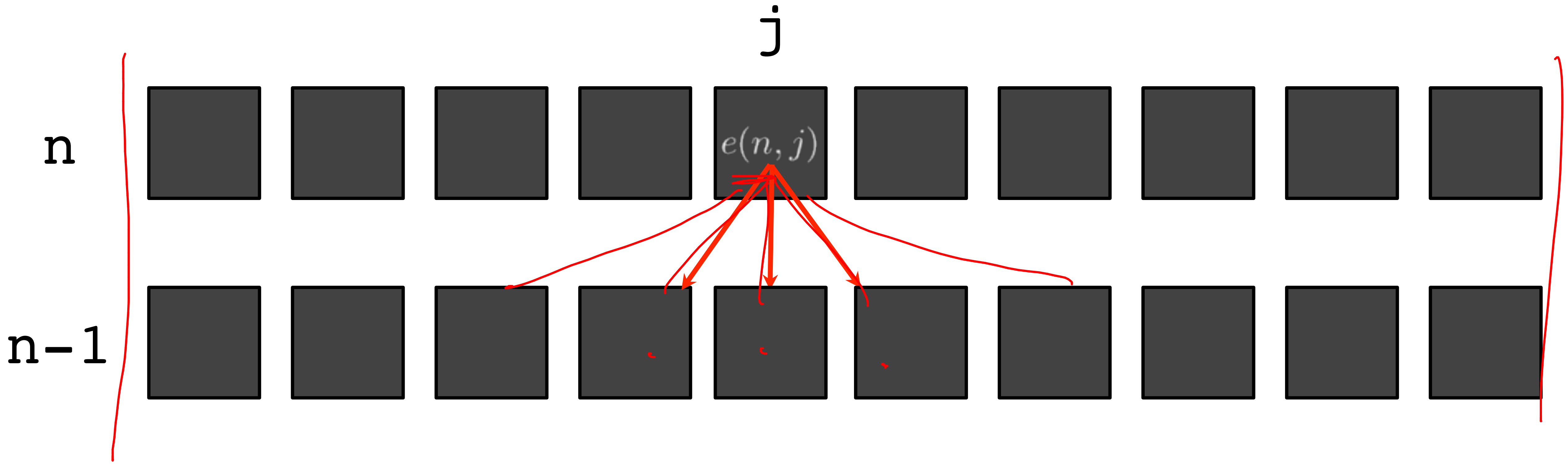




$$S_n(1) = e(n,1) + \min\{S_{n-1}(1), S_{n-1}(2)\}$$



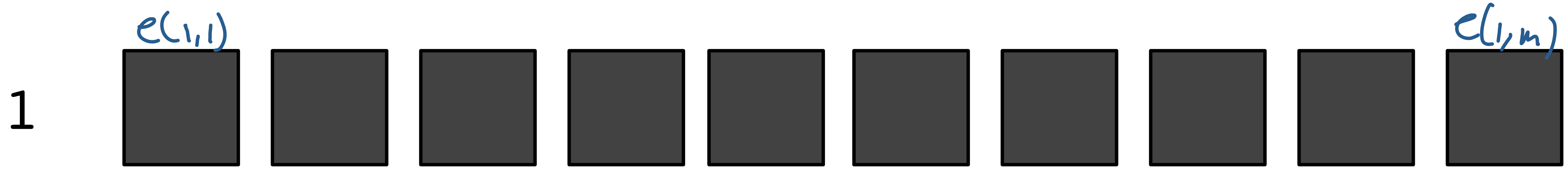
$$S_i(j) =$$



$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$

ALGORITHM

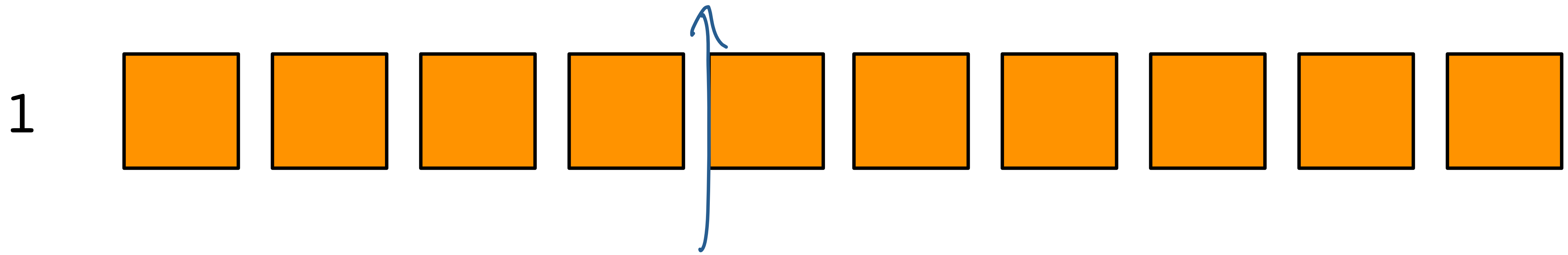
start at bottom of picture



$S_{1,j} = e_{1,j}$

ALGORITHM

start at bottom of picture. initialize $S_1(i) = e(1, i)$

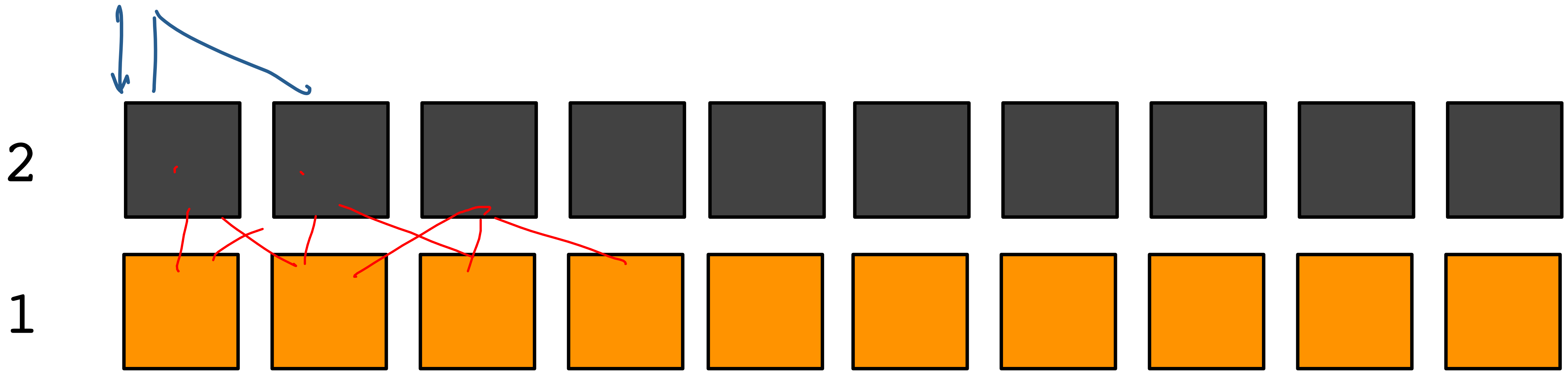


ALGORITHM

start at bottom of picture. initialize $S_1(i) = e(1,i)$

for `i=2 to n` use formula to compute $S_{i+1}(\cdot)$

$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$

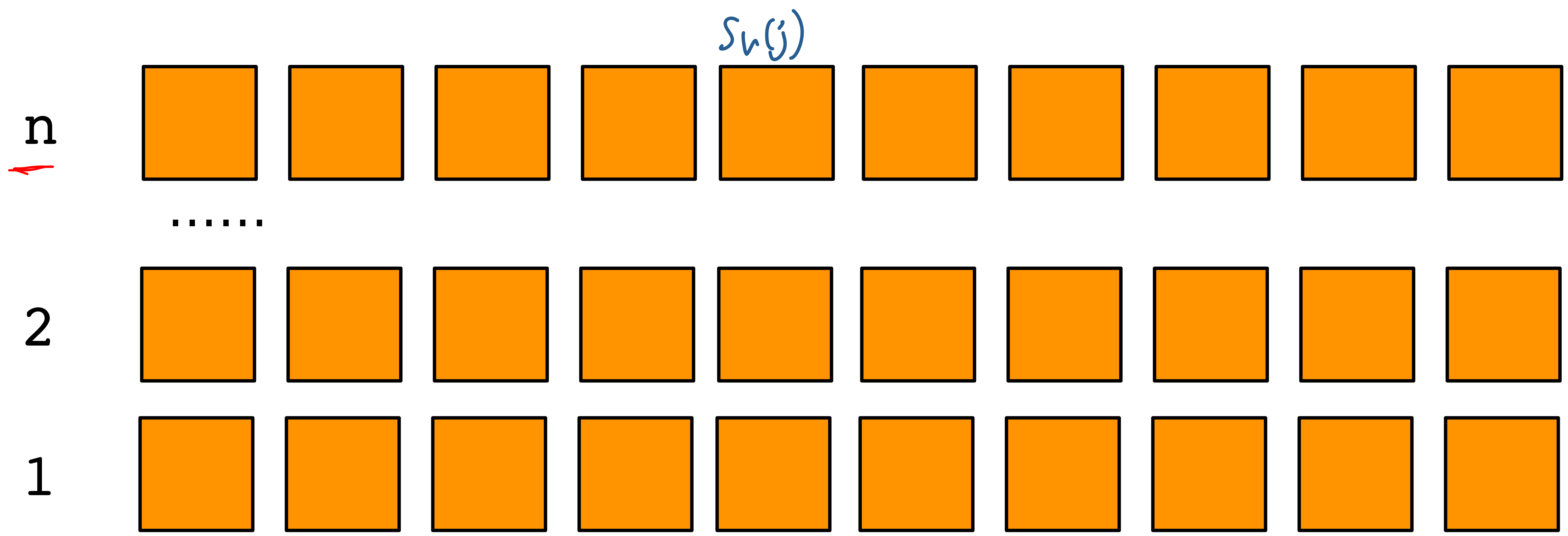


ALGORITHM

start at bottom of picture. initialize $S_1(i) = e(1, i)$

for `i=2,n` use formula to compute $S_{i+1}(\cdot)$

$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$



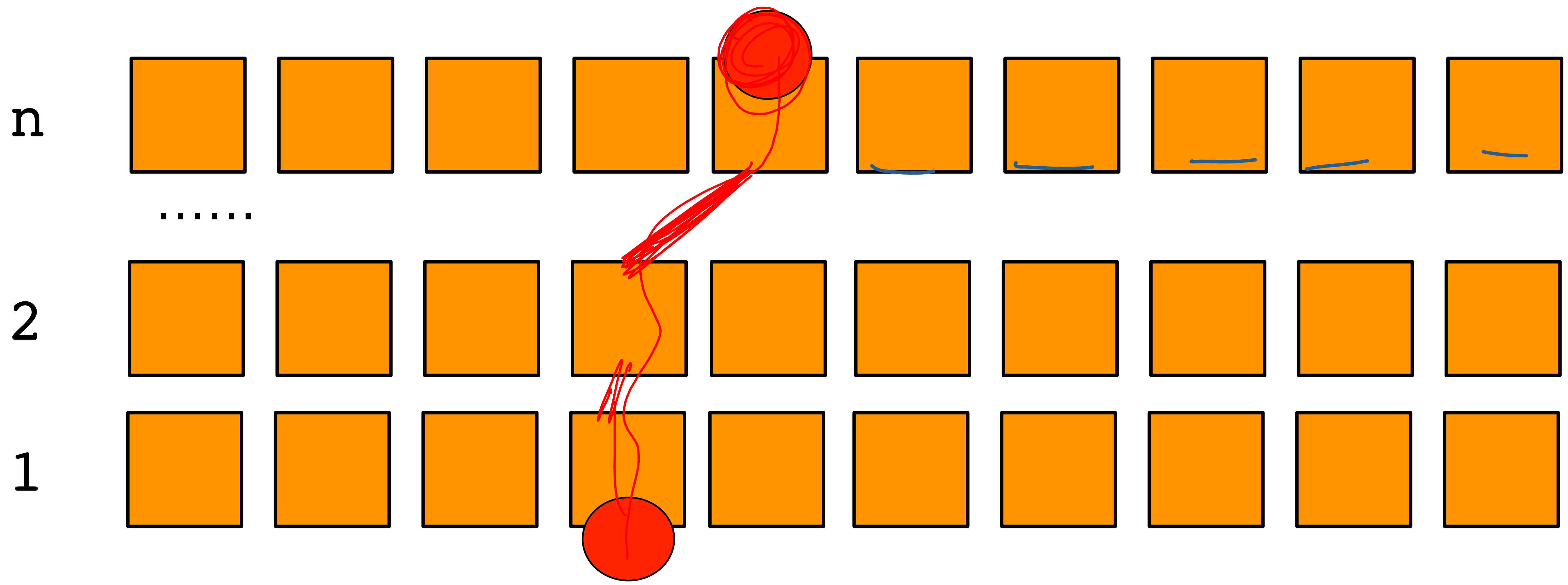
ALGORITHM

start at bottom of picture. initialize $S_1(i) = e(1, i)$

for `i=2,n` use formula to compute $S_{i+1}(\cdot)$

$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$

pick best among top row, backtrack.



RUNNING TIME

start at bottom of picture. initialize $S_1(i) = e(1, i)$

for `i=2,n` use formula to compute $S_{i+1}(\cdot)$

$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$

pick best among top row, backtrack.

RUNNING TIME

start at bottom of picture. initialize $S_1(i) = e(1, i)$

for `i=2,n` use formula to compute $S_{i+1}(\cdot)$

$$S_i(j) = e(i,j) + \min \begin{cases} S_{i-1}(j-1) \\ S_{i-1}(j) \\ S_{i-1}(j+1) \end{cases}$$

pick best among top row, backtrack.