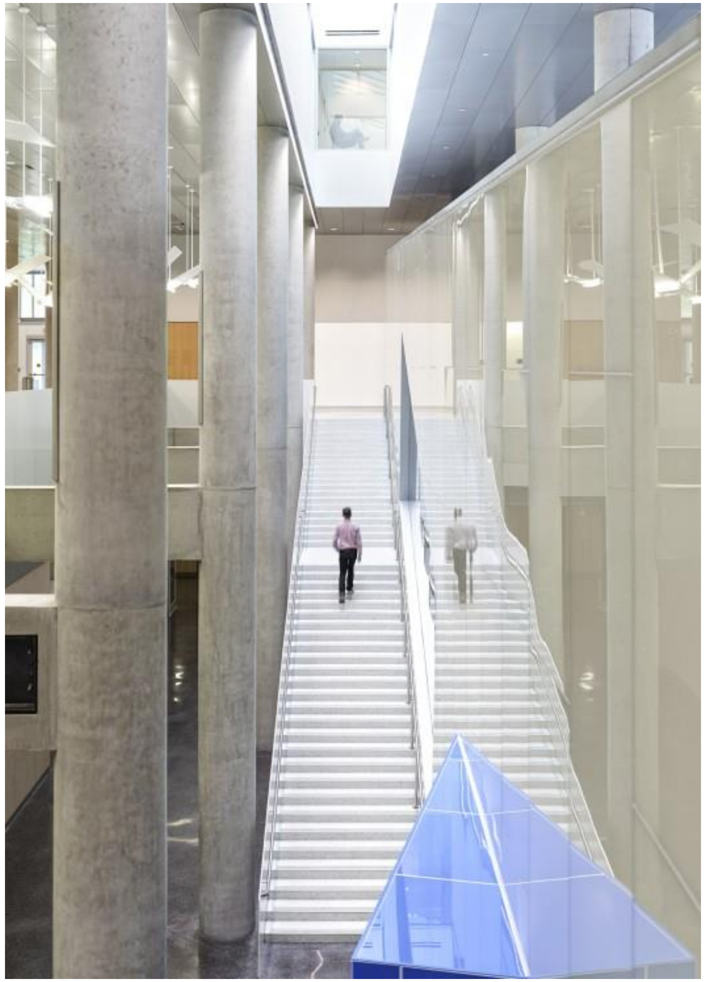
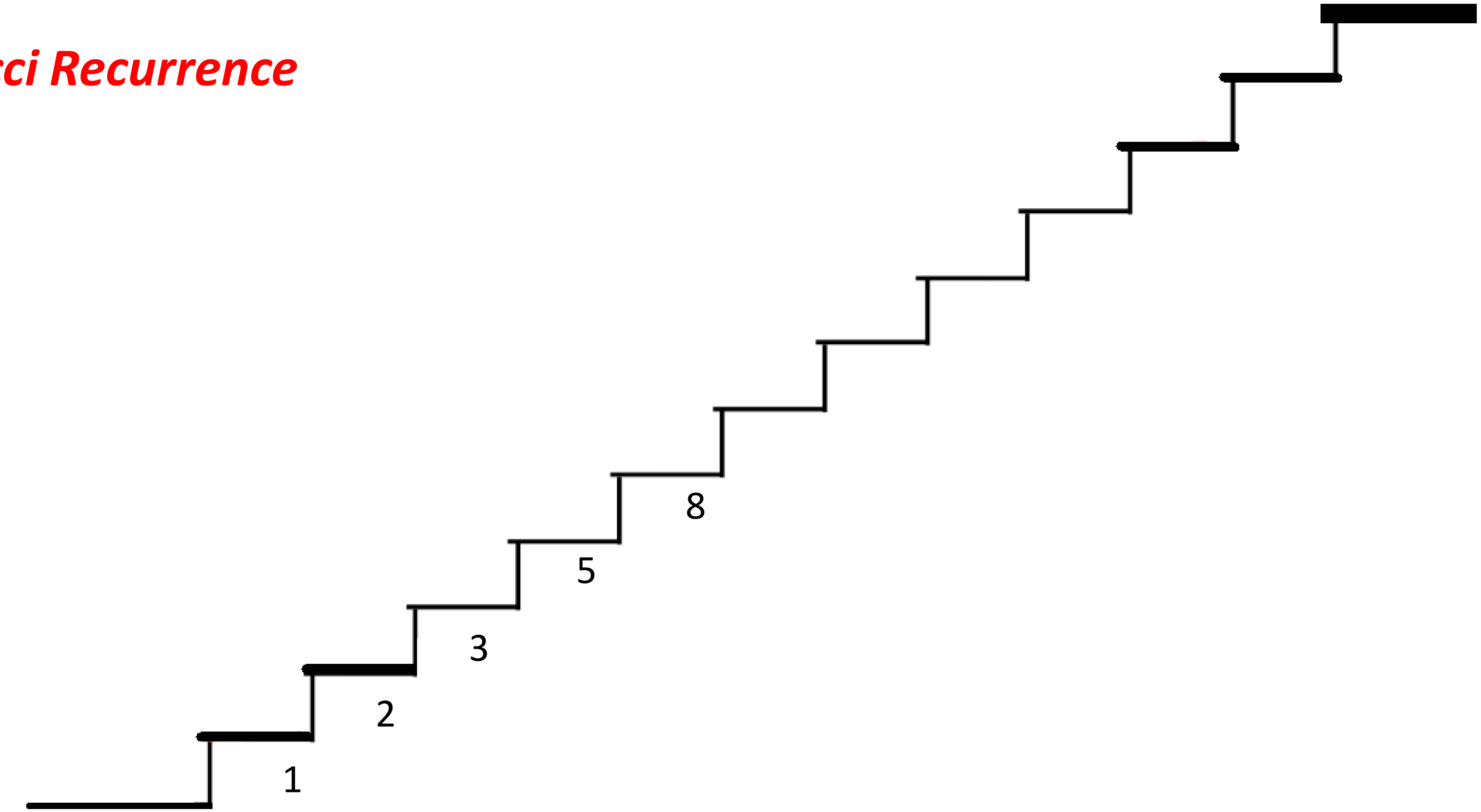
Dynamic Programming


alamy stock photo
P1NYBK
www.alamy.com



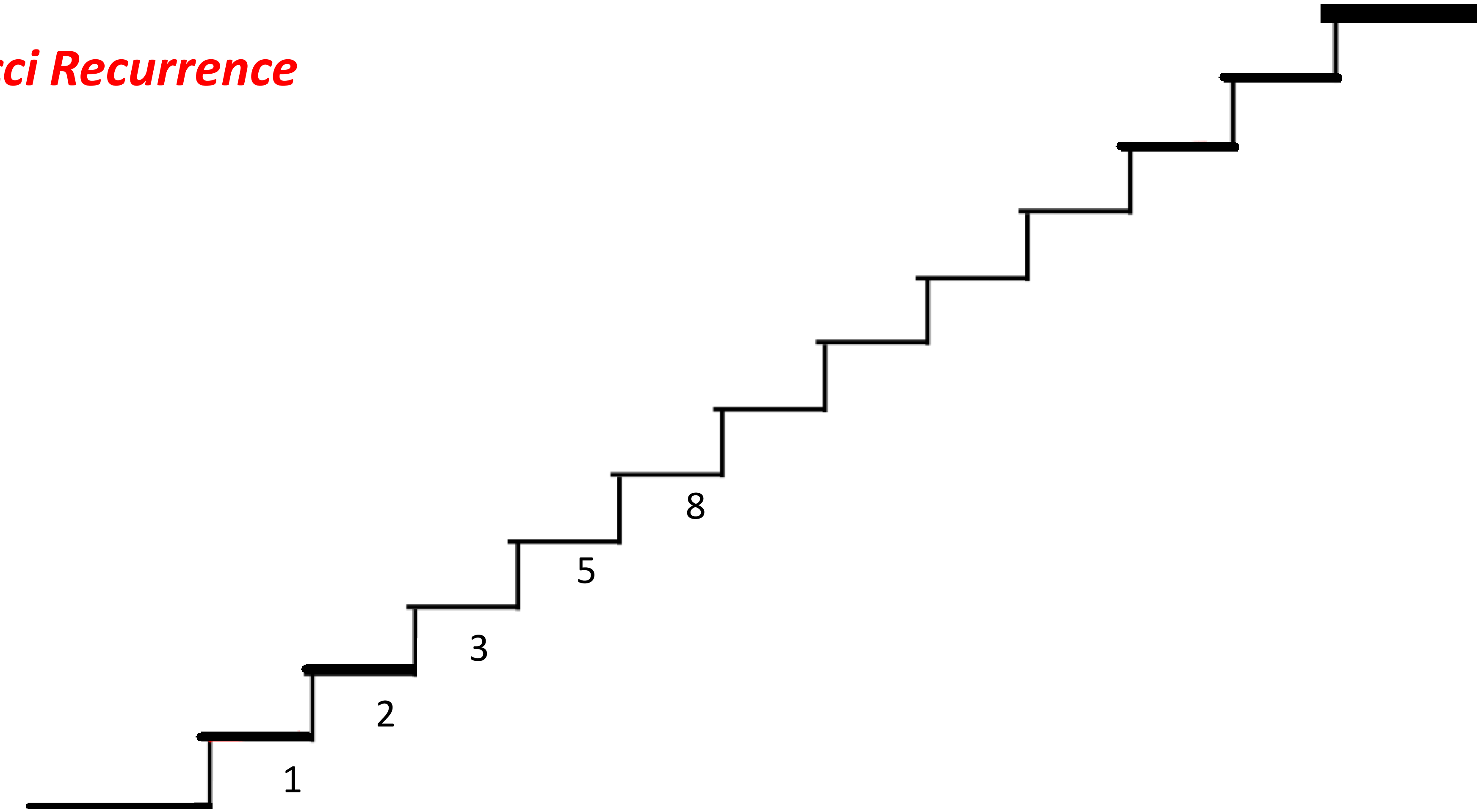
$T(n)=T(n-2)+T(n-2)$

Fibonacci Recurrence



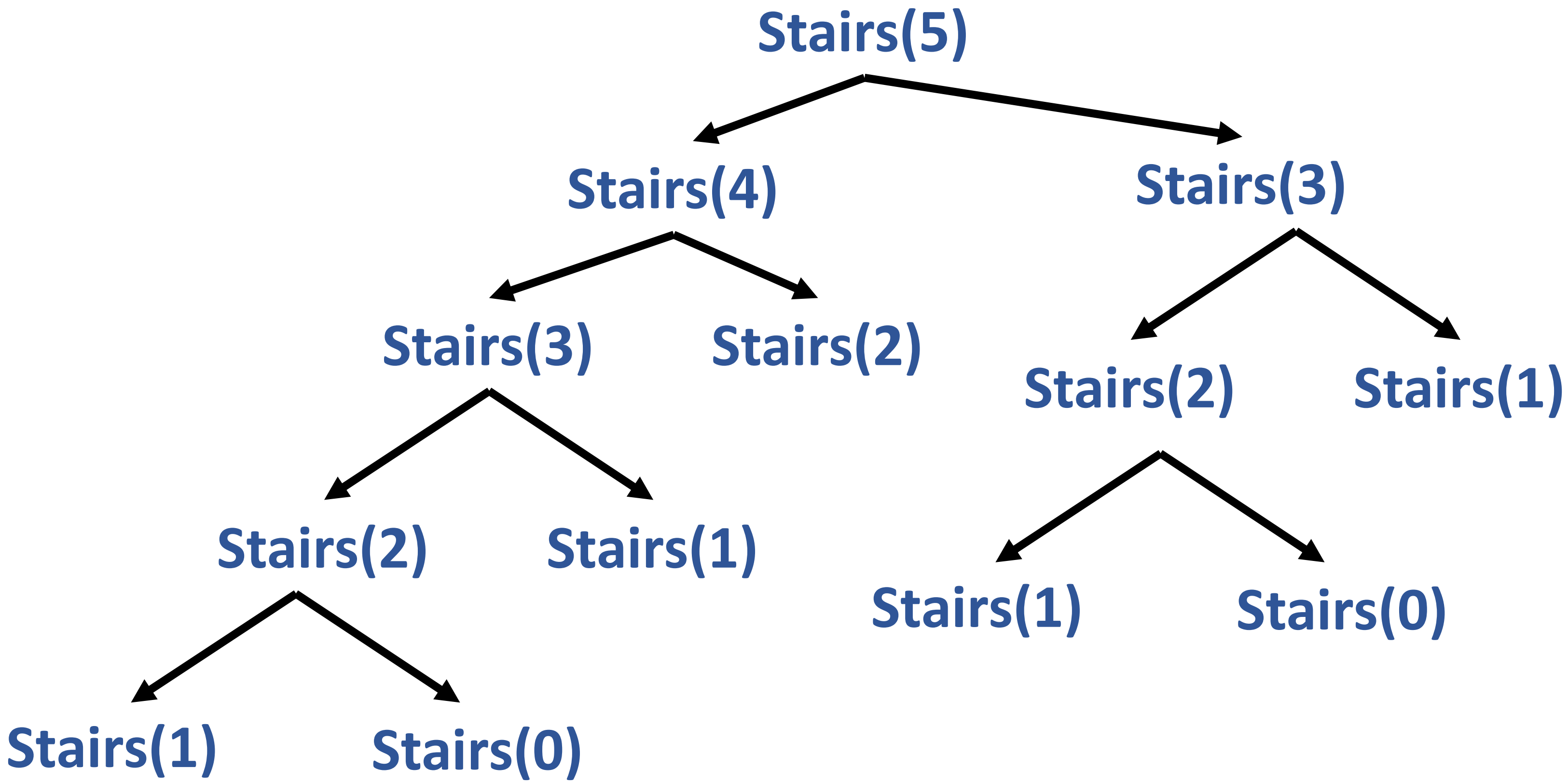
$T(n)=T(n-2)+T(n-2)$

Fibonacci Recurrence



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

We are calling same thing several times



Initialize memory M

Initialize memory M

Stairs(n)

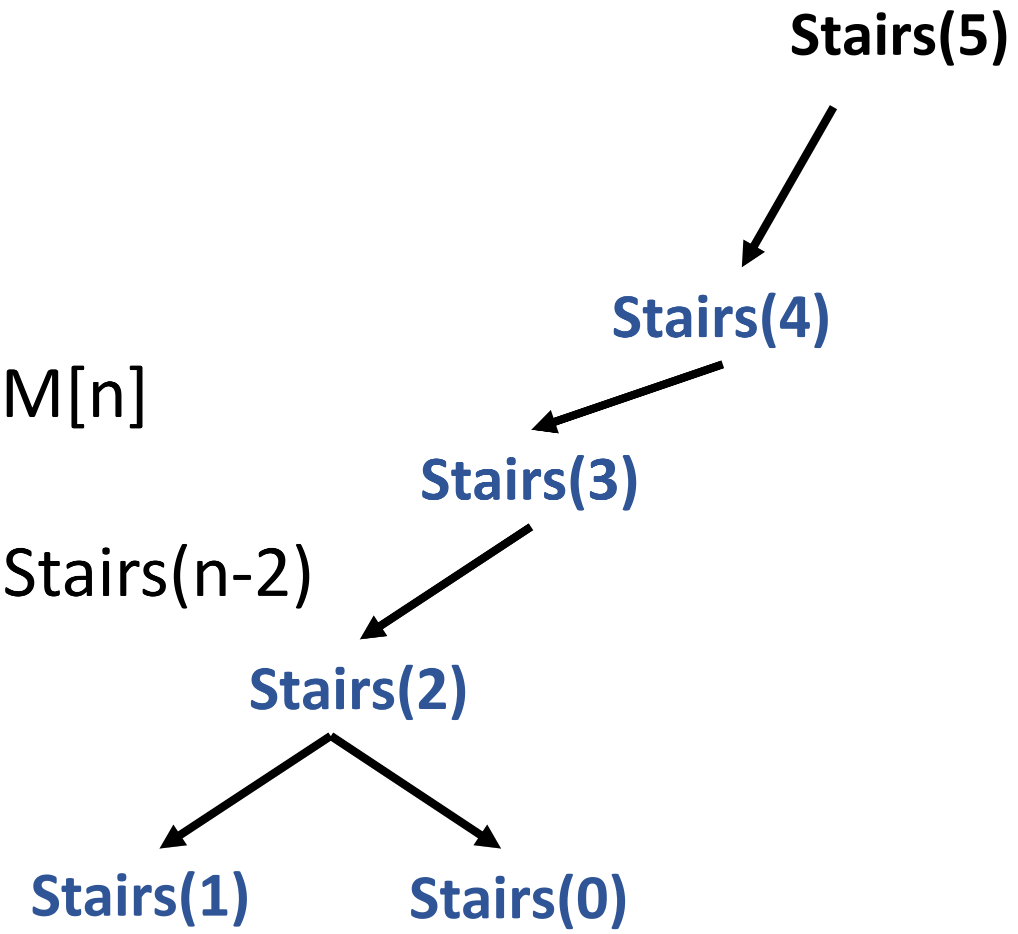
If n<=1, return 1

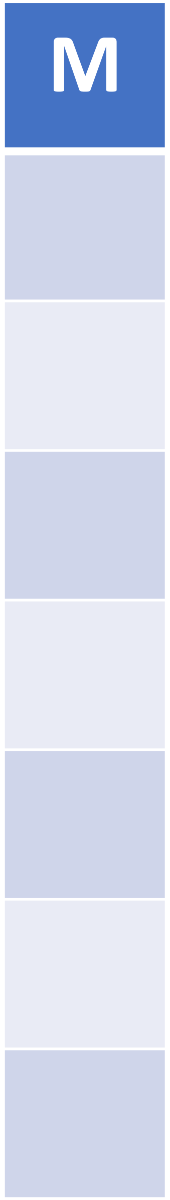
If n is in M[n] , return M[n]

Answer= Stairs(n-1) + Stairs(n-2)

M[n]=answer

Return Answer





Initialize memory M

Stairs(n)

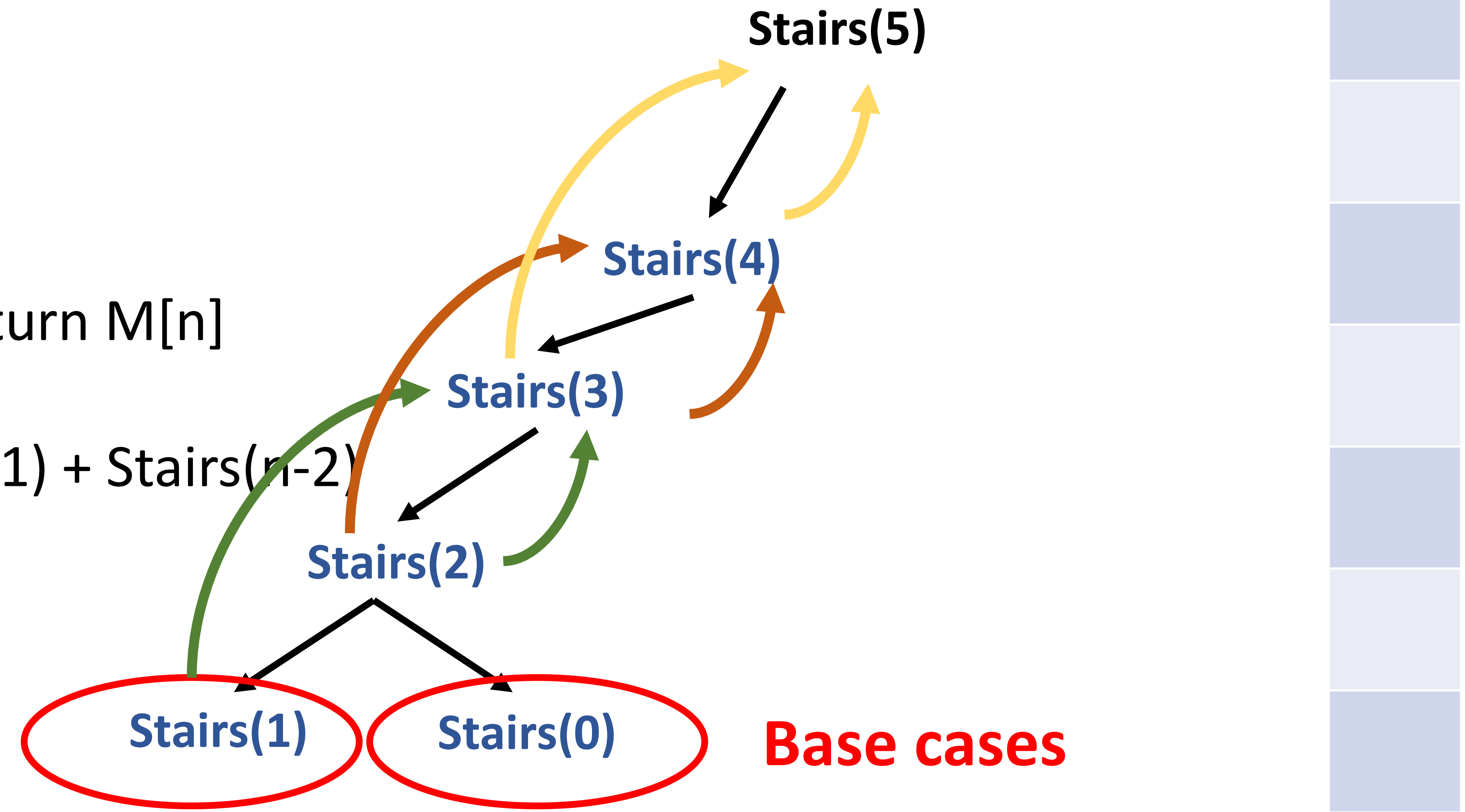
If n<=1, return 1

If n is in M[n] , return M[n]

Answer= Stairs(n-1) + Stairs(n-2)

M[n]=answer

Return Answer



θ(n) running time

θ(n) Memory

Initialize memory M

Stairs(n)

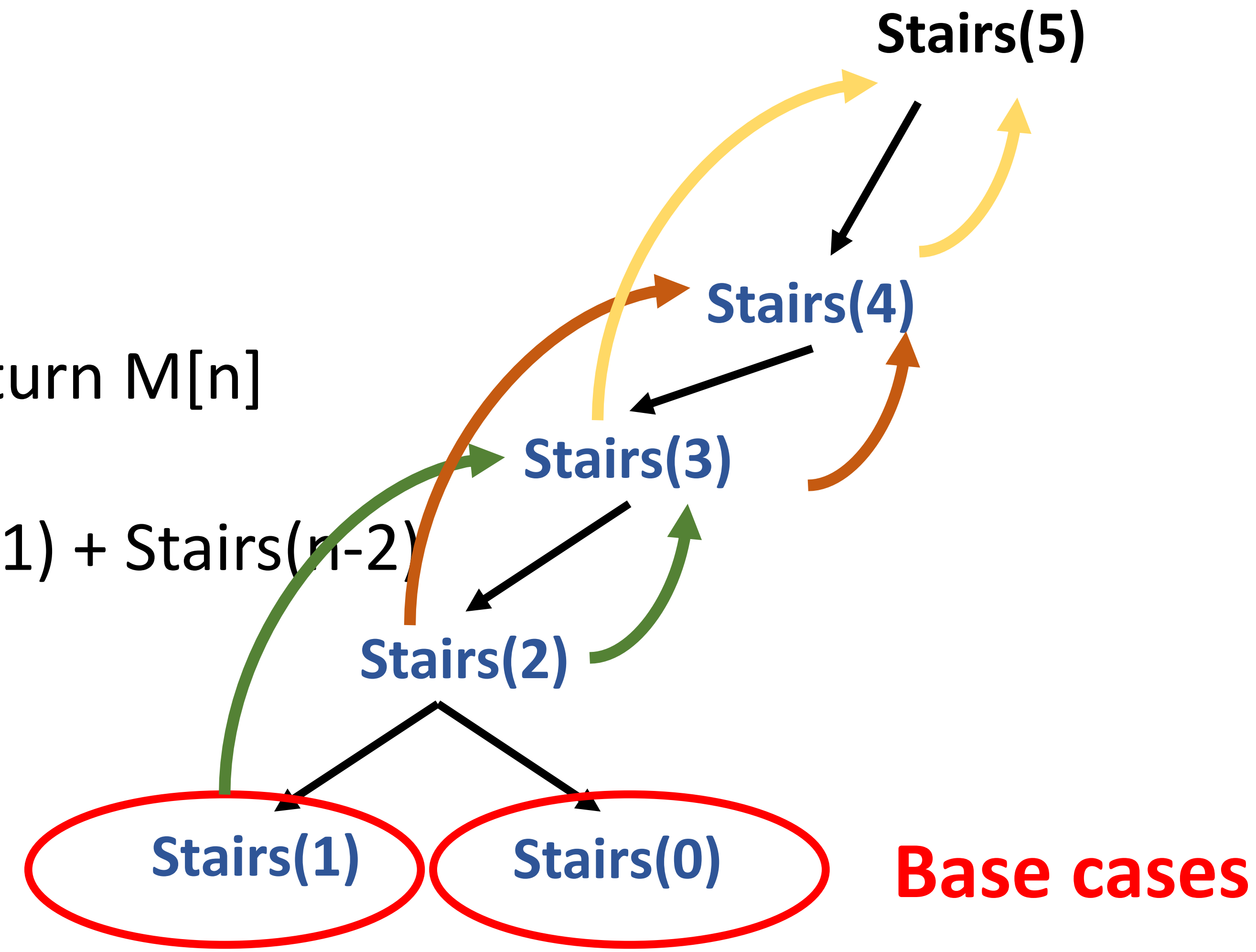
If n<=1, return 1

If n is in M[n] , return M[n]

Answer= Stairs(n-1) + Stairs(n-2)

M[n]=answer

Return Answer



M
1
1
2
3
5
8

Initialize memory M

$\theta(n)$ running time

$\theta(n)$ Memory

Stairs(n)

Stairs(5)

If

If

An

M[n]=answer

Return Answer

Stairs(1)

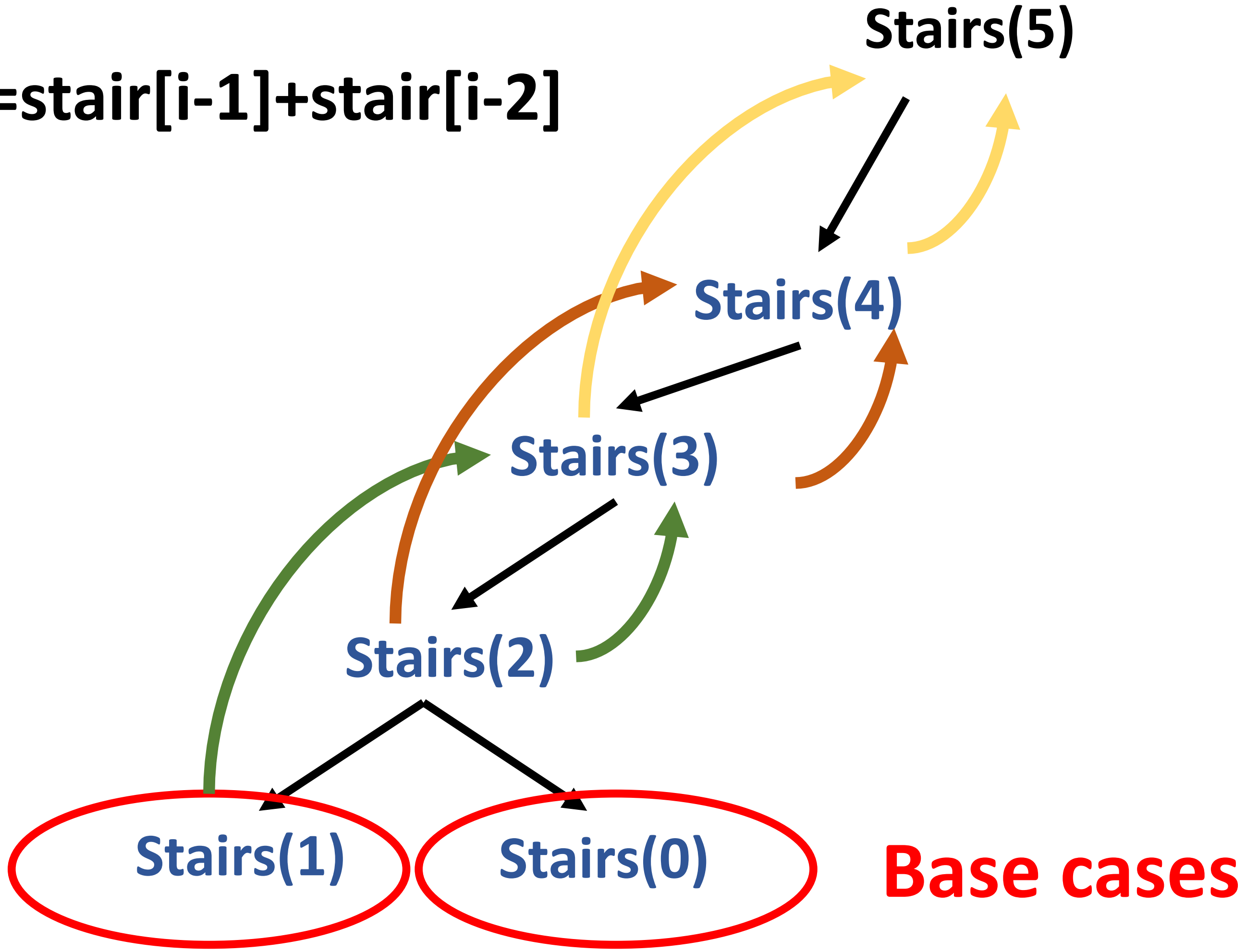
Stairs(0)

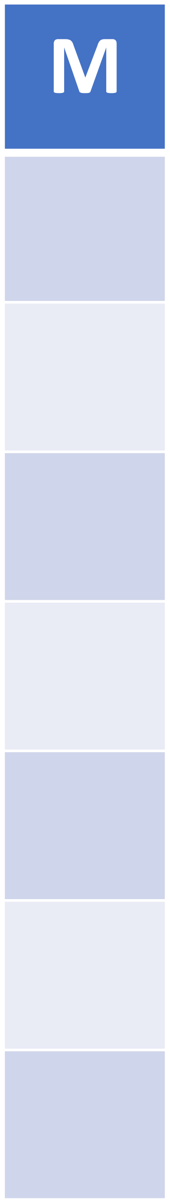
M

What are the big ideas?

- Memorize
- Start from small problems

```
Stair(n)
    Stair[0]=1
    Stair[1]=1
    for i=2 to n:
        Stair[i]=stair[i-1]+stair[i-2]
    return stair[i]
```





Stair(n)

3 ways of the solution

```
Stair[0]=1
Stair[1]=1

for i=2 to n:
    Stair[i]=stair[i-1]+stair[i-2]

return stair[i]
```

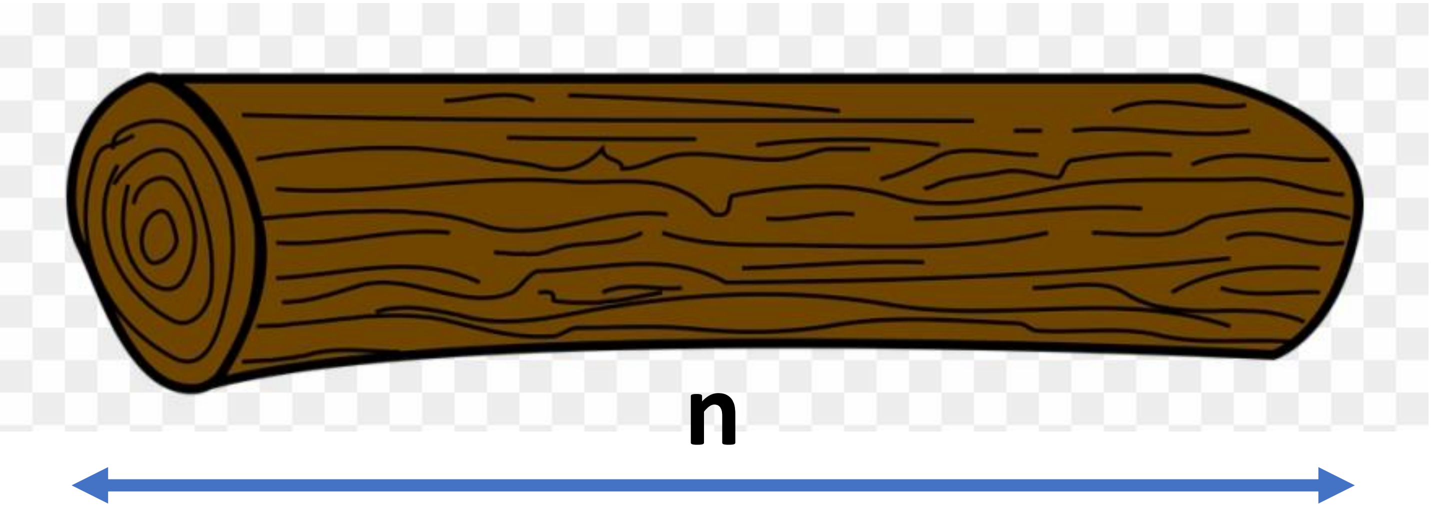
Dynamic Programming

1. Has a **recursive solution** to the problem
2. Has **memory**
3. Pick the **correct order** for evaluating the smaller problems

Wood Cutting Problem

Wood Cutting Problem

1''	2''	3''	4''	5''	6''	7''	8''
10$	16$	27$	48$	50$	90$	100$	130$



Wood Cutting Problem

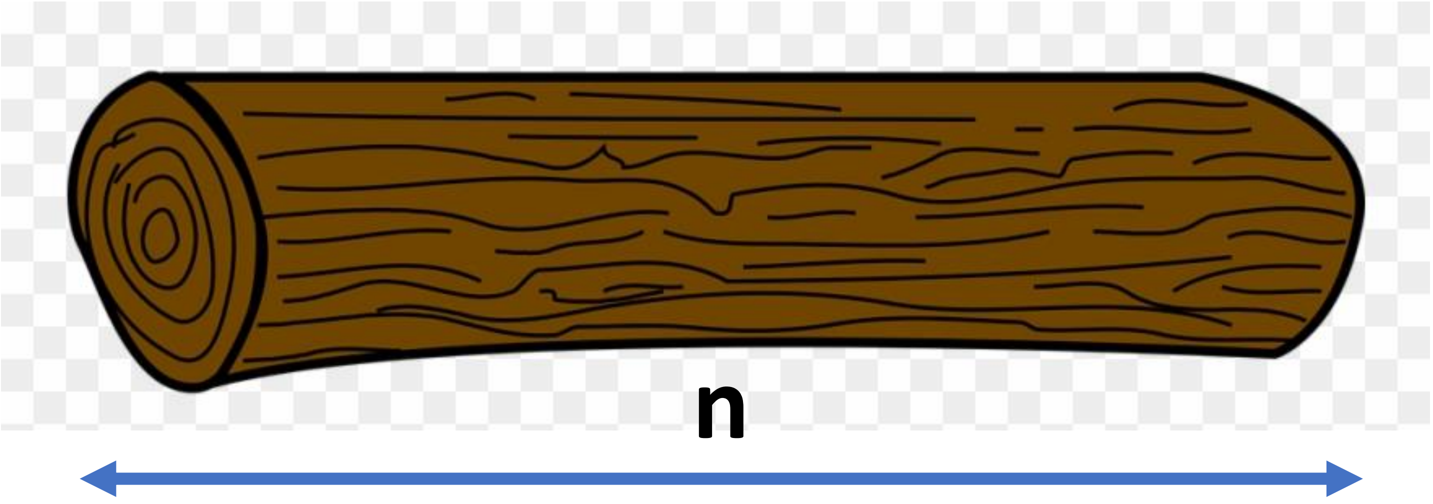
Input to the problem : n, $(p_1, p_2, \ldots\ldots\ldots, p_n)$:

n is total dimension of the wood log,
P_i prices of an 'i' length wood plank

Goal: Max profil, i.e.,

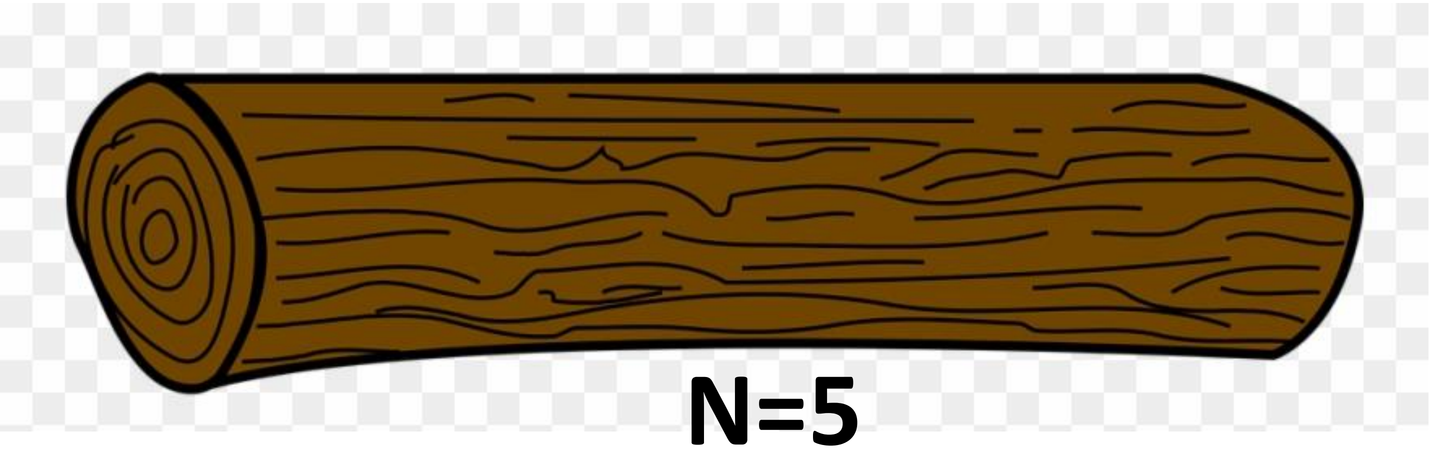
Output $(c_1, c_2, \ldots, c_k)$ the width of cuts to make
Subject to:

$$\sum_{j=1}^{k} c_j = n$$



Can we try Greedy approach?

1’’	2’’	3’’	4’’	5’’
1$	6$	7$	8$	10$



N=5

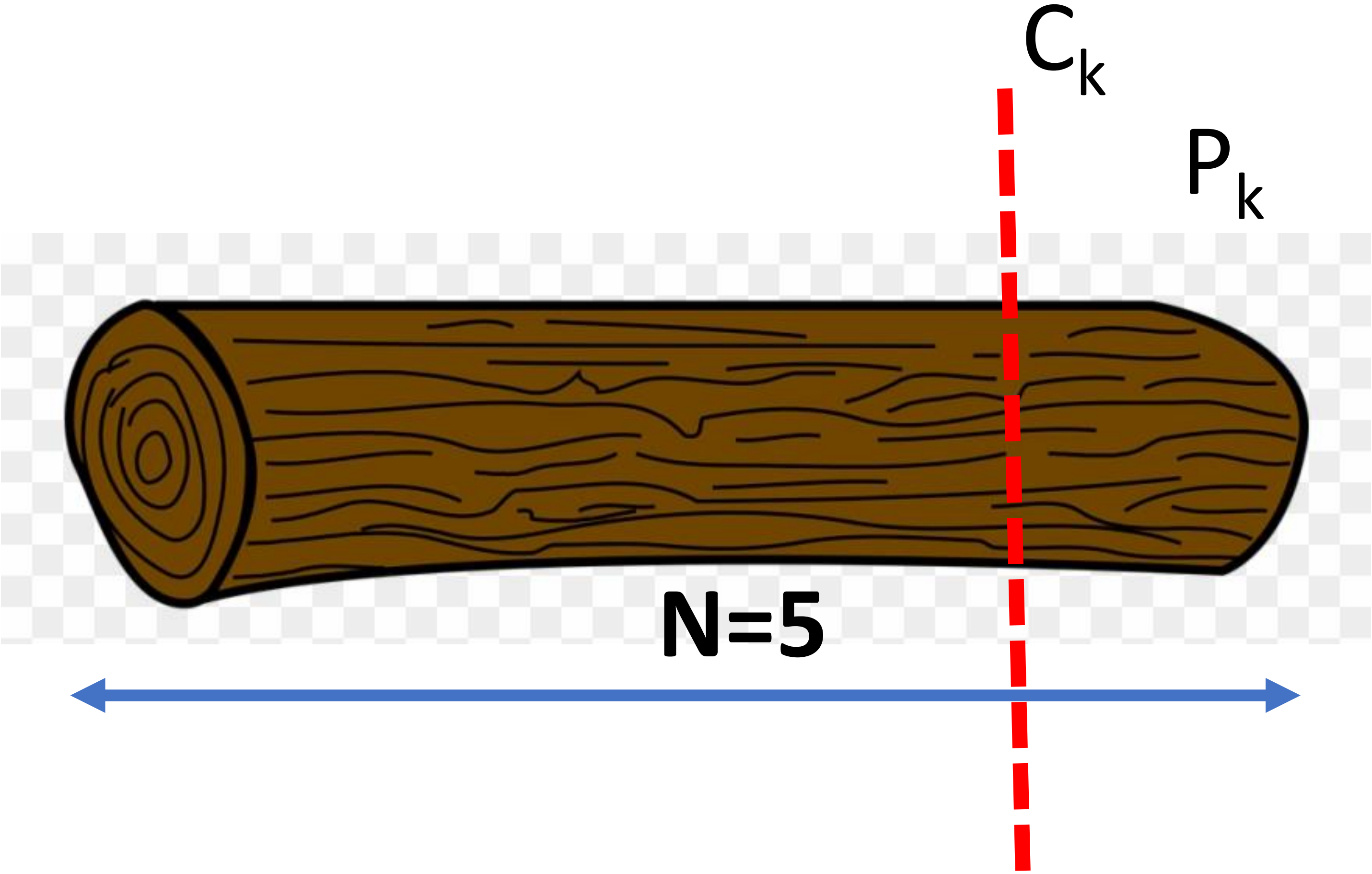
How about average?

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

N=5

Observation

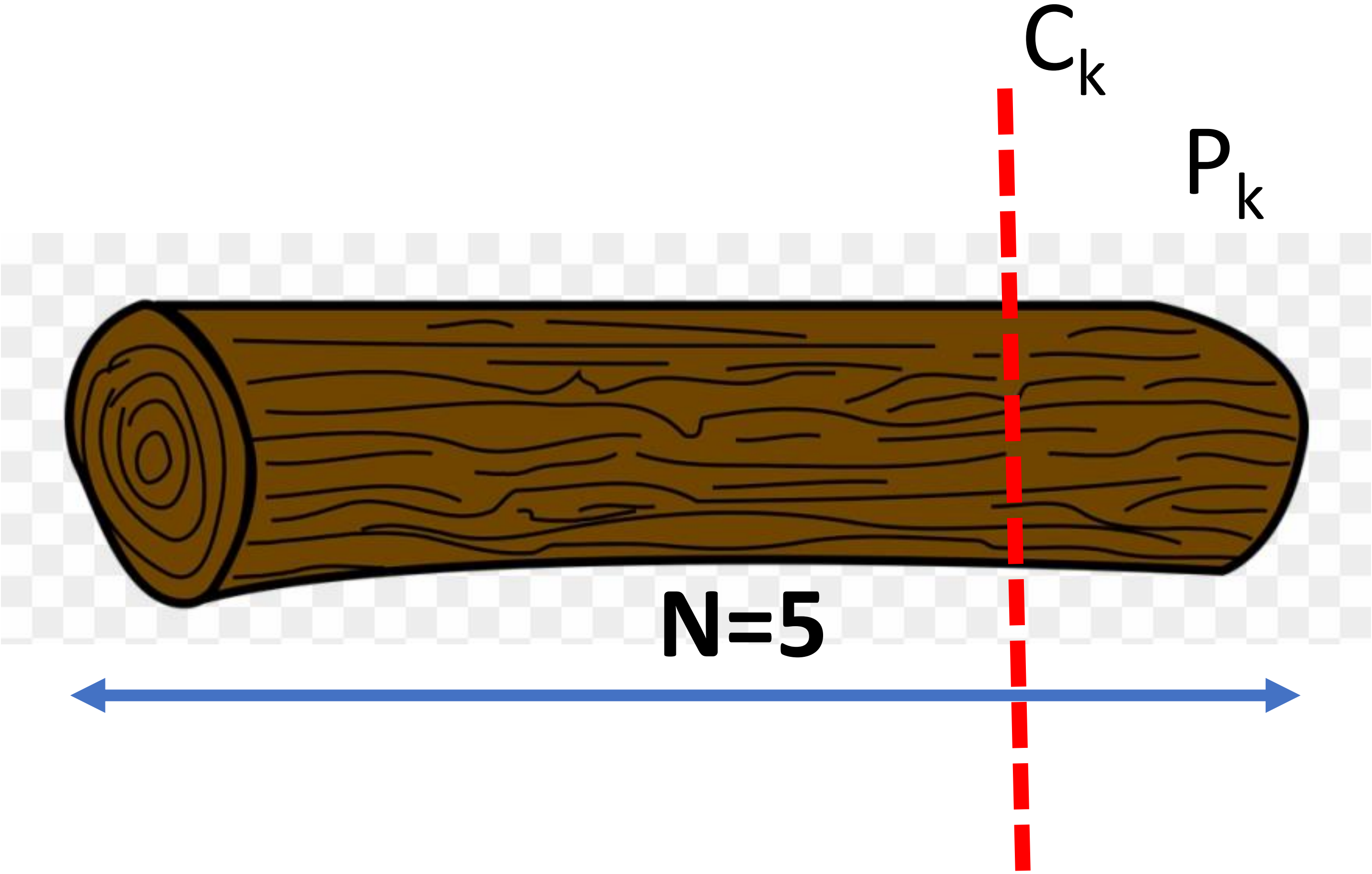
Best_N : Best profit for an ‘n’ thick wood log



$\text{Best}_N = P_k + \text{Best}_{N-Pk}$

Observation

Best $_N$: Best profit for an 'n' thick wood log

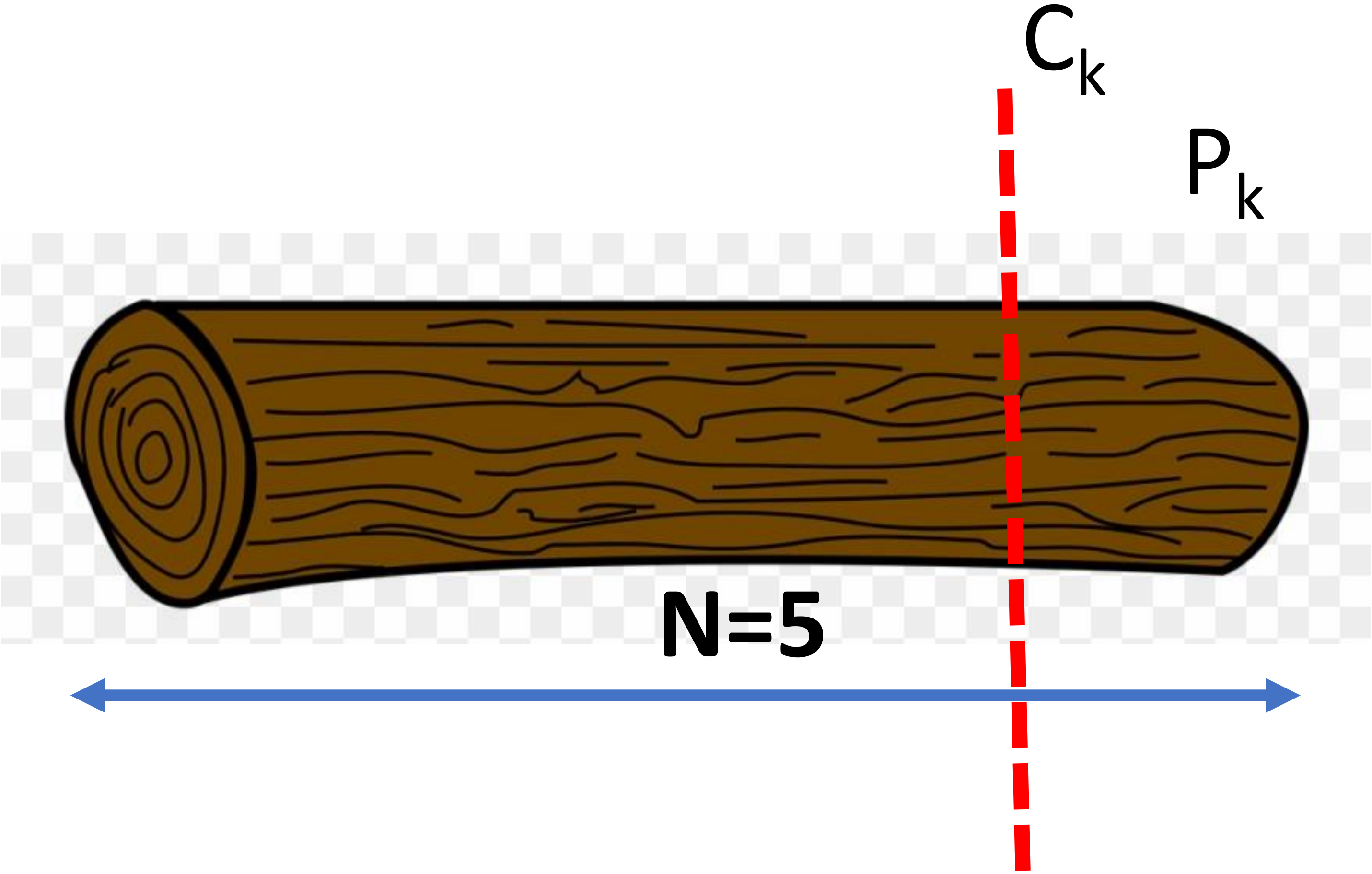


$$Best_N = P_k + Best_{N-Pk}$$

How many choices are there for this P_k? **N**

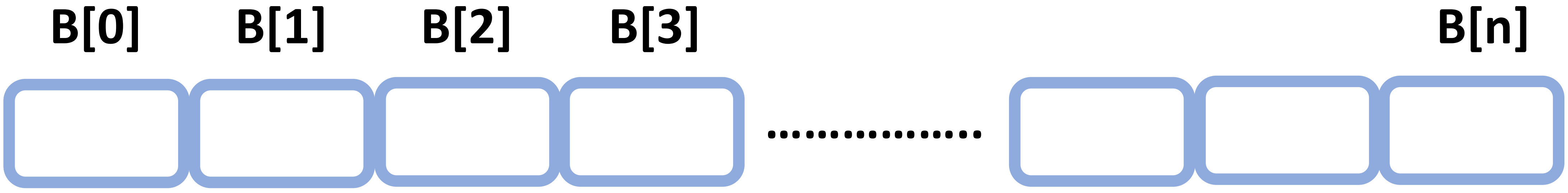
Solution <-> Observation

$Best_N$: Best profit for an ‘n’ thick wood log



$$Best_N = \text{Max} \begin{cases} P_1 + Best_{N-1} \\ P_2 + Best_{N-2} \\ P_3 + Best_{N-3} \\ \vdots \\ P_k + Best_{N-Pk} \end{cases}$$

The best price is stored at B[i], if the wooden log size is i



Picked the order of evaluation of the smaller problems

BestLogs(n,(p_1,p_2,p_3,……,p_n))

if n<=0 return 0

Initiate B[0…..n]

B[0]=0

for i=1 to n: **$\theta(n)$**

Set B[i]= Max $_{j=1}^{i}$ { P_j+B[i-j] } **$\theta(n)$**

Return B[n]

$\theta(n^2)$

Which Cuts?

BestLogs(n,(p_1,p_2,p_3,……,p_n))

if n<=0 return 0

Initiate B[0…..n]

Initiate Choice[1….n]

B[0]=0

for i=1 to n:

Set B[i]= Max $_{j=1}^{i}$ { P_j+B[i-j] }

Choice[i]= the best of j

Return B[n] , Choice[i]

Which Cuts?

BestLogs(n,(p_1,p_2,p_3,……,p_n))

if n<=0 return 0

Initiate B[0…..n]

Initiate Choice[1….n]

B[0]=0

for i=1 to n:

Set B[i]= Max $_{j=1}^{i}$ { P_j+B[i-j] }

Choice[i]= the best of j

Return B[n] , Choice[i]