PROBLEMA DEL LADRÓN
(CON PROGRAMACIÓN DINÁMICA)

Algoritmos y Programación
Javier Miranda

Escuela de Ingeniería Informática
Universidad de Las Palmas de Gran Canaria

25 de octubre de 2023

Problema del ladrón

- Nuestro ladrón debe elegir qué casas debe robar para conseguir el máximo beneficio.

- Para que no se activen las alarmas si roba en una casa no puede robar en la siguiente.

¿ Qué casas debe elegir ?

Problema del ladrón

¿ Recurrencia ?

V = [3, 10, 3, 1, 2]
N = 1, 2, 3, 4, 5

V = [3, 10, 3, 1, 2]

N = 1, 2, 3, 4, 5

V = [3, 10, 3, 1, 2]

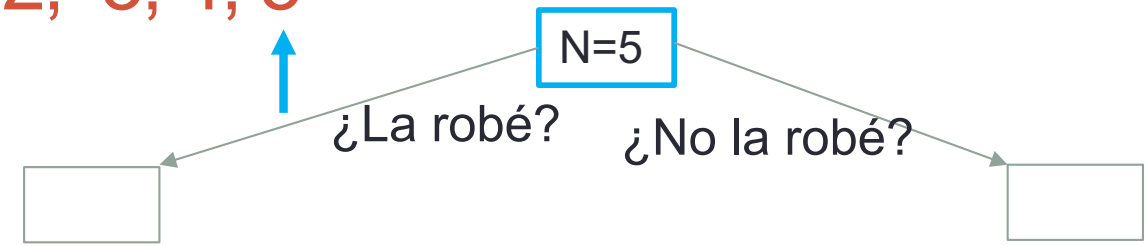
N = 1, 2, 3, 4, 5

N=5

Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

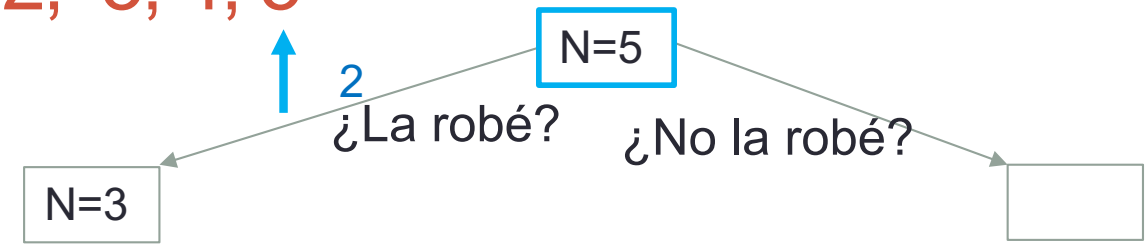
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

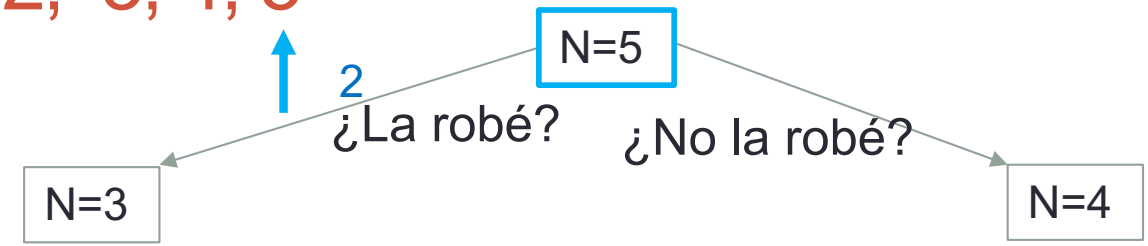
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

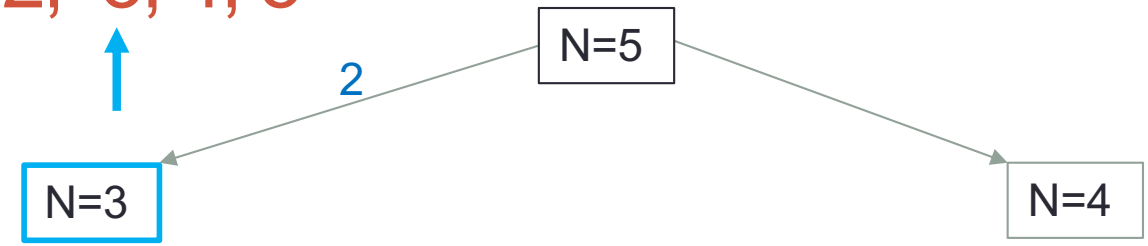
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

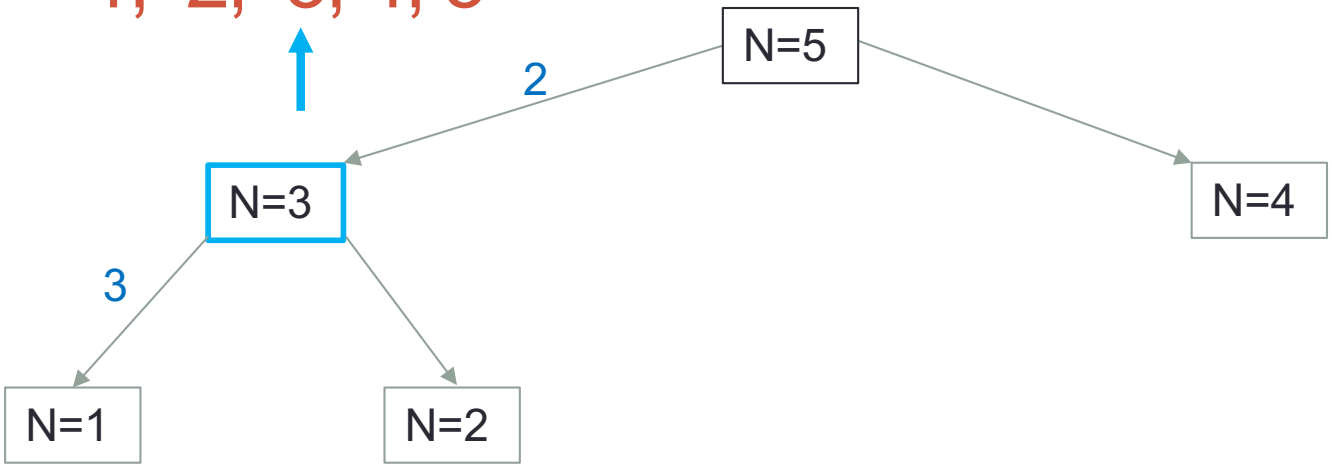
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

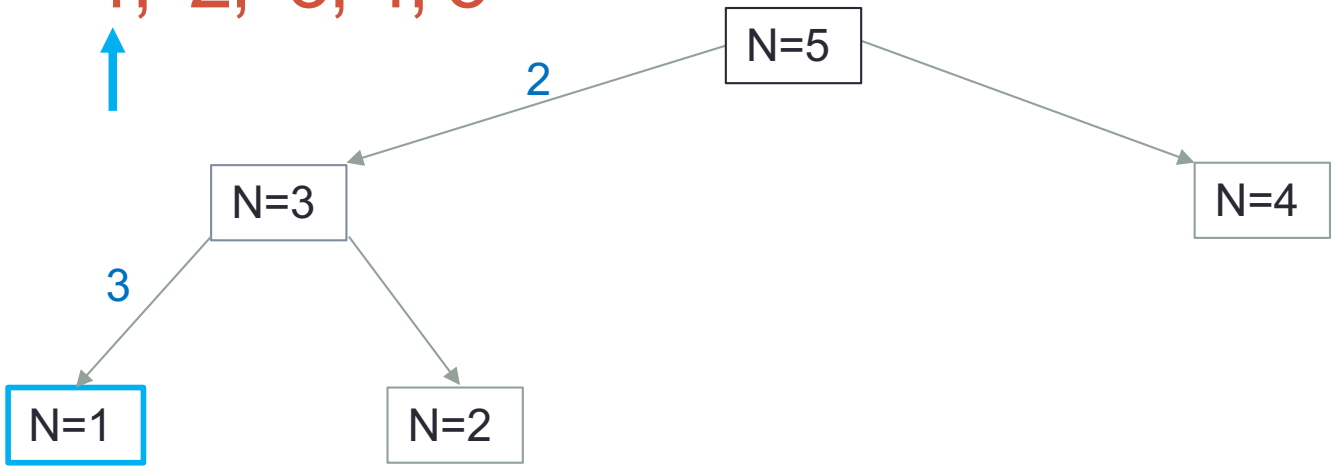
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

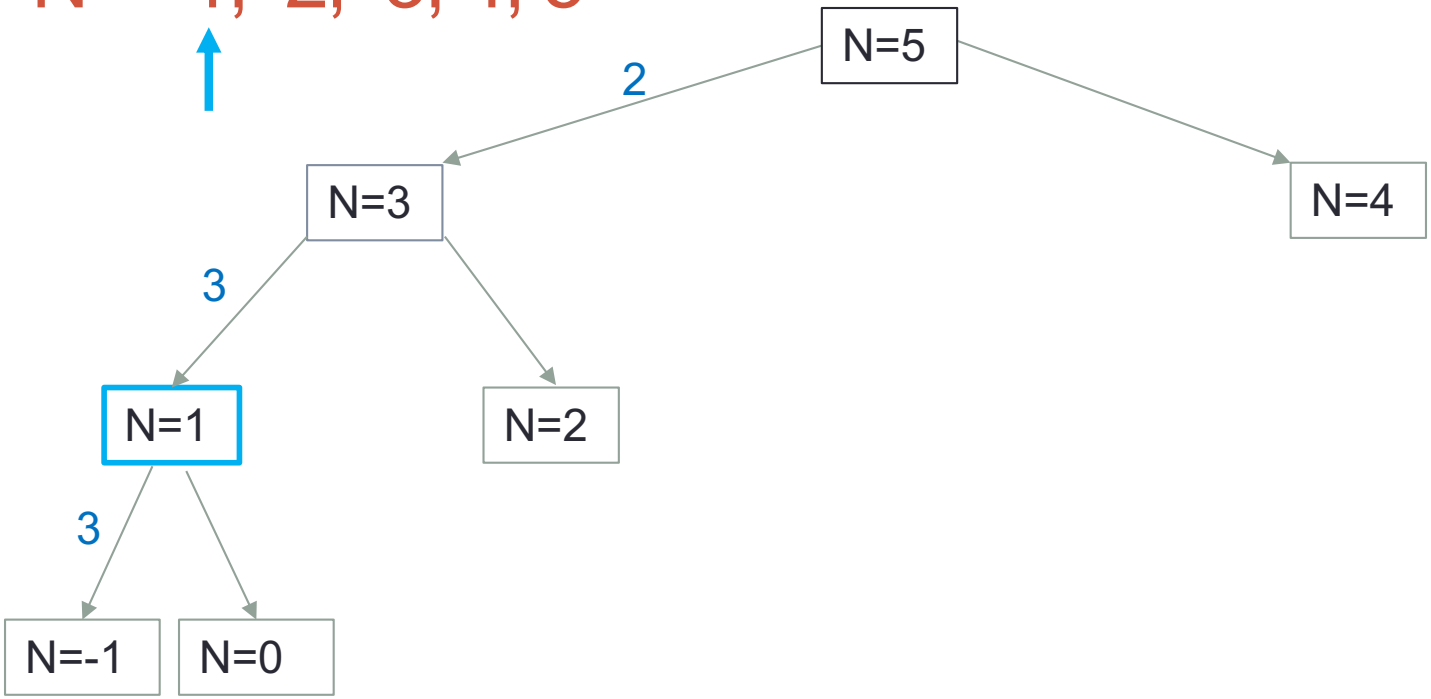
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

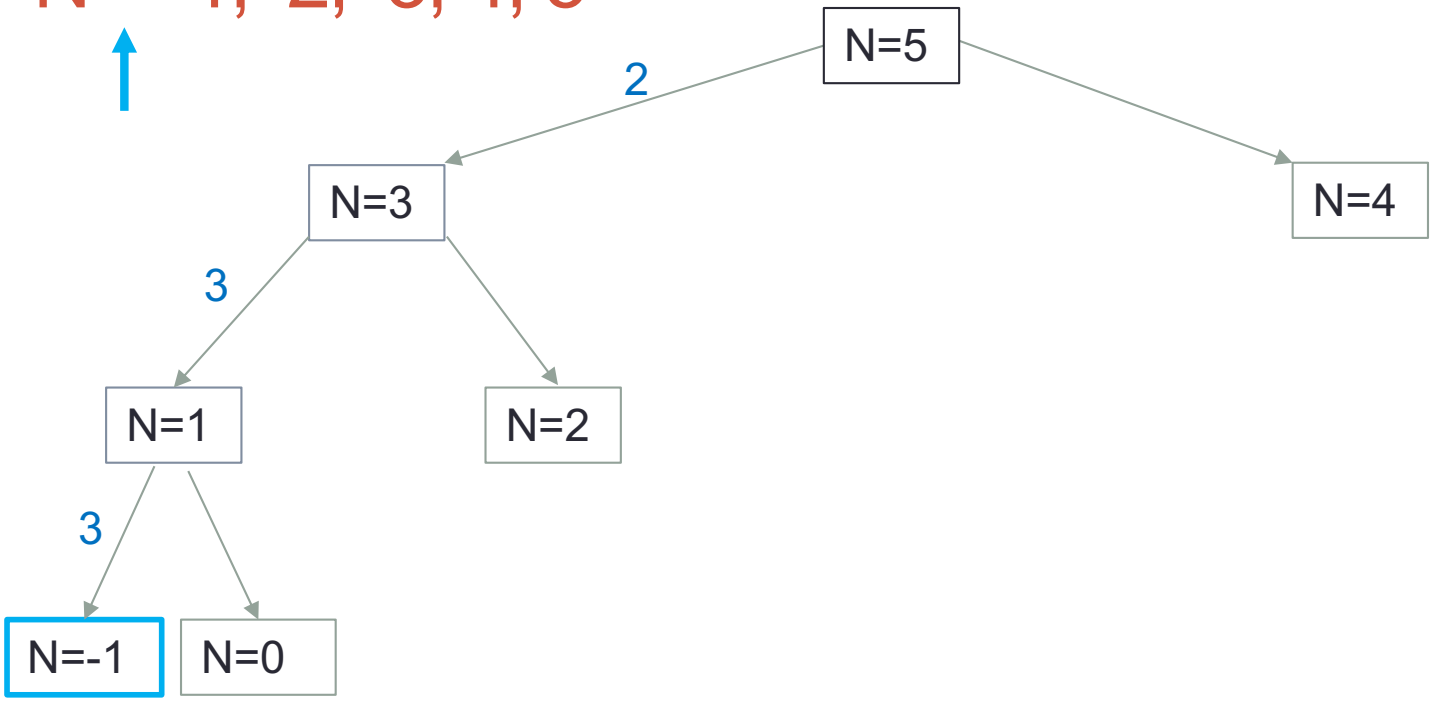
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

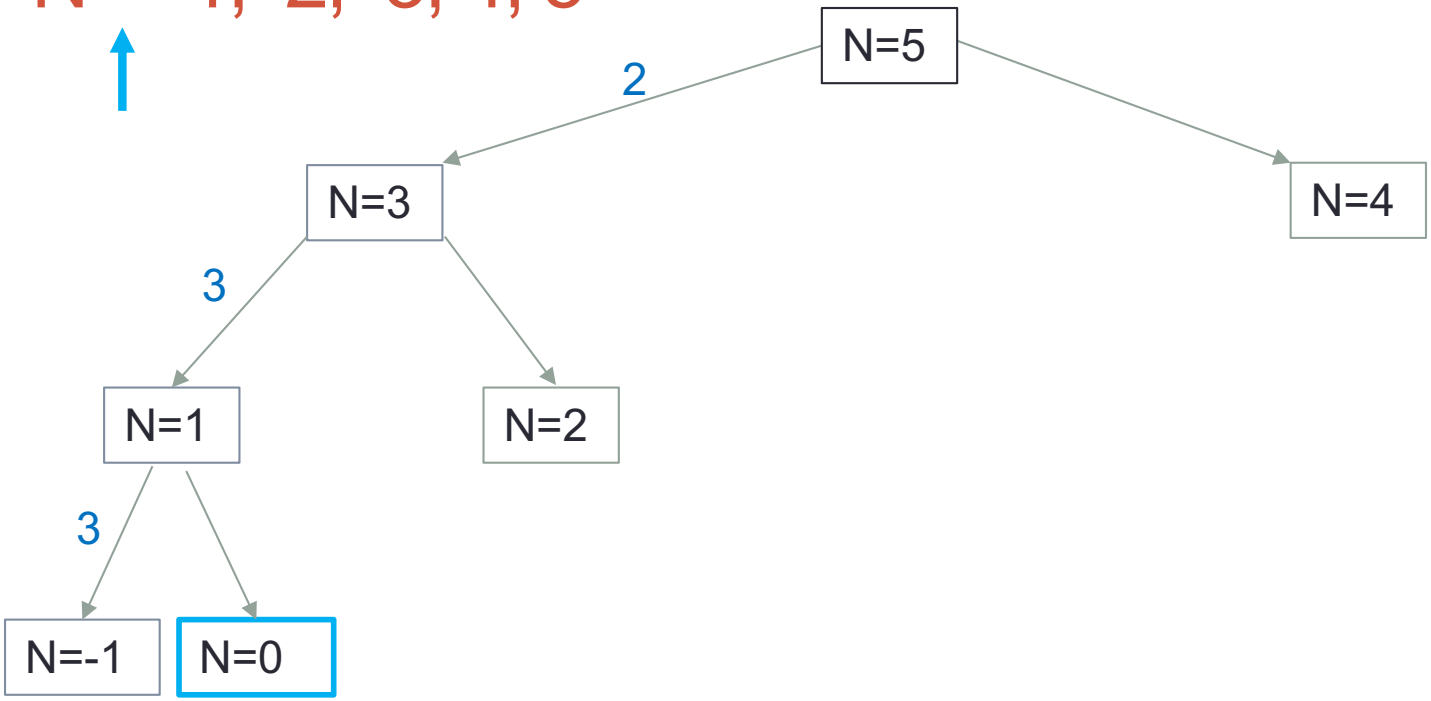
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

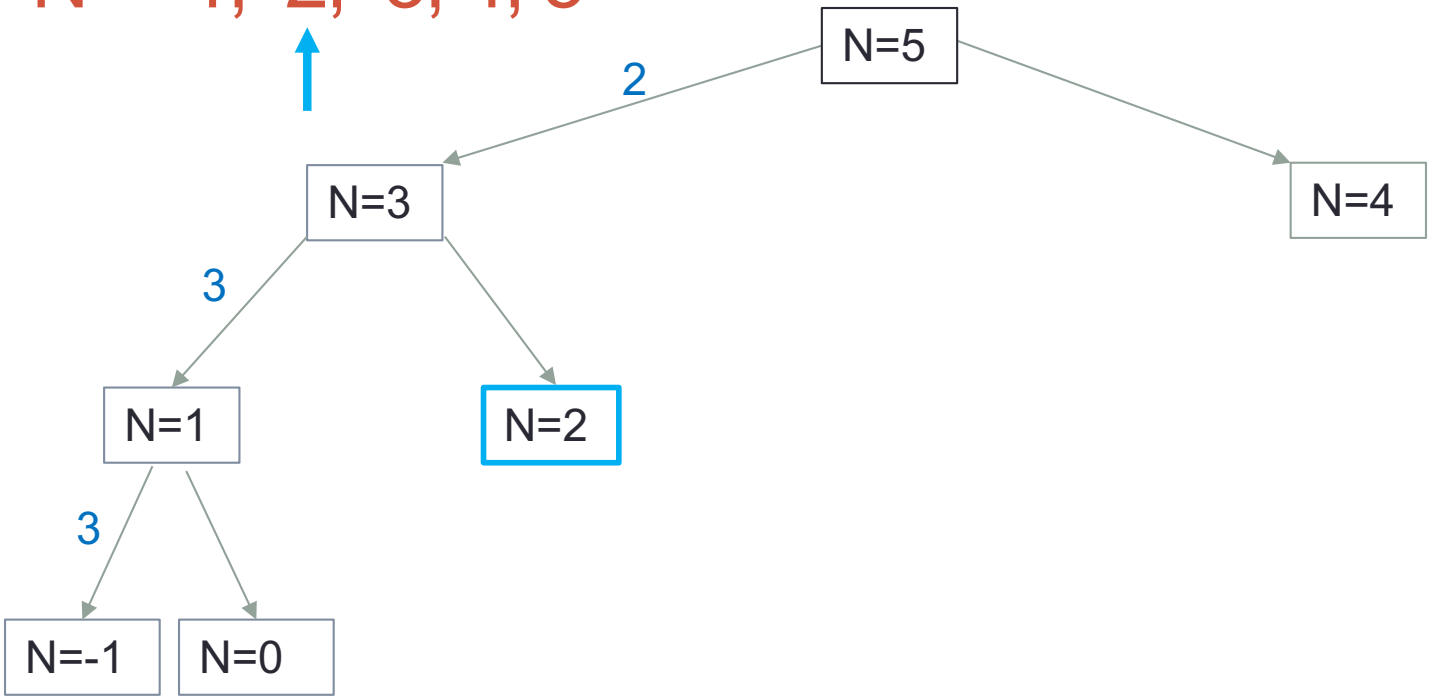
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

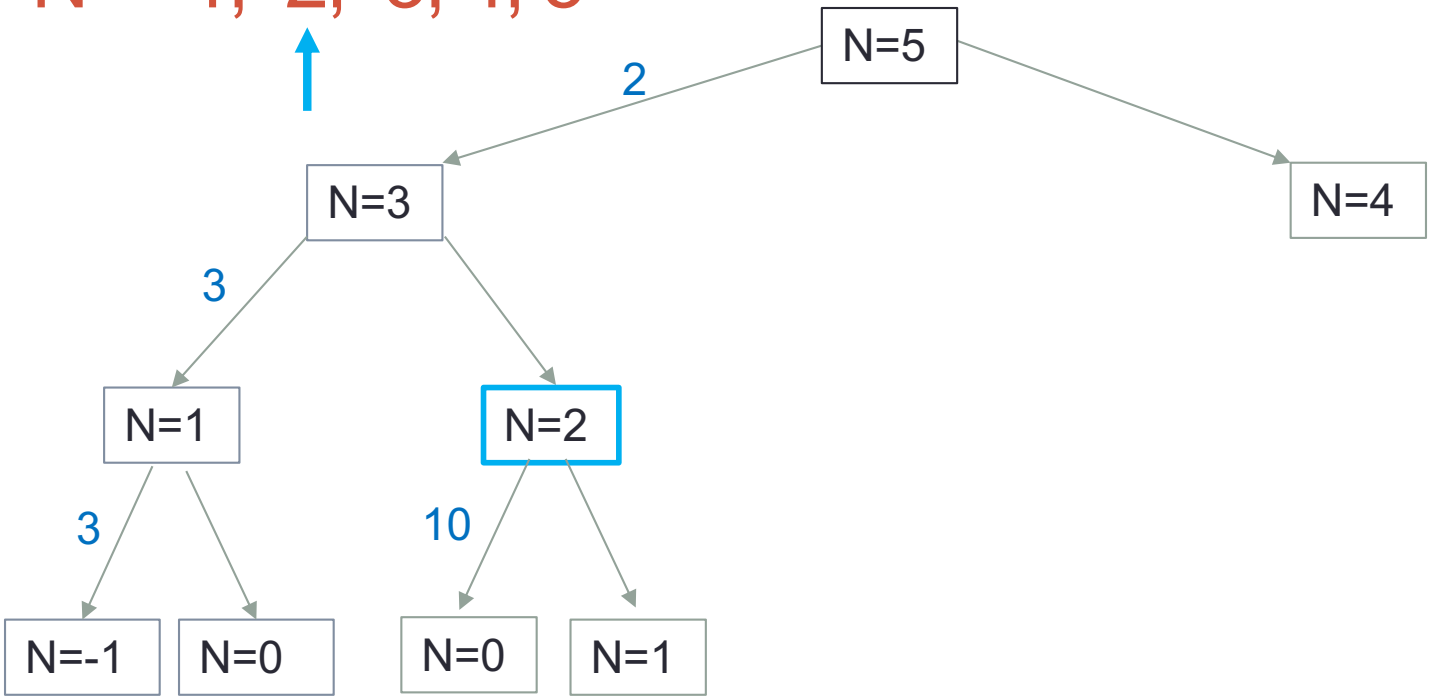
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

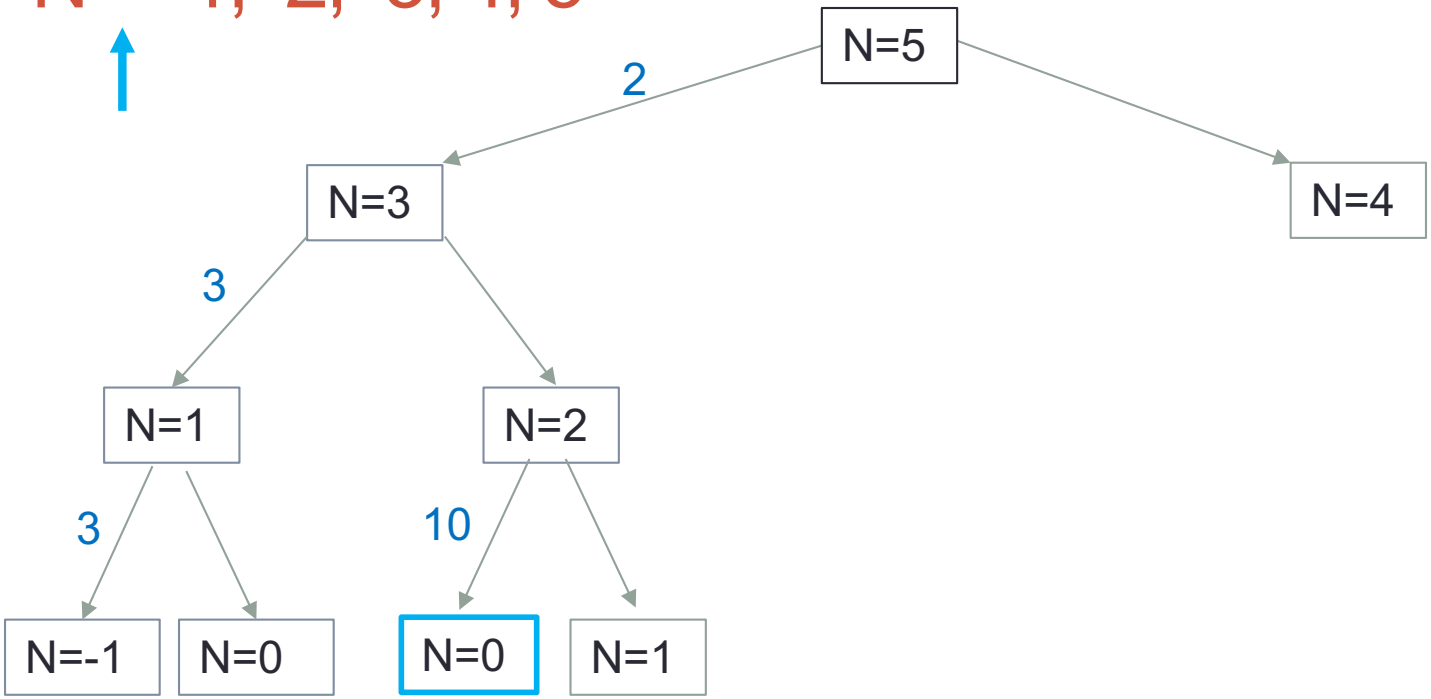
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

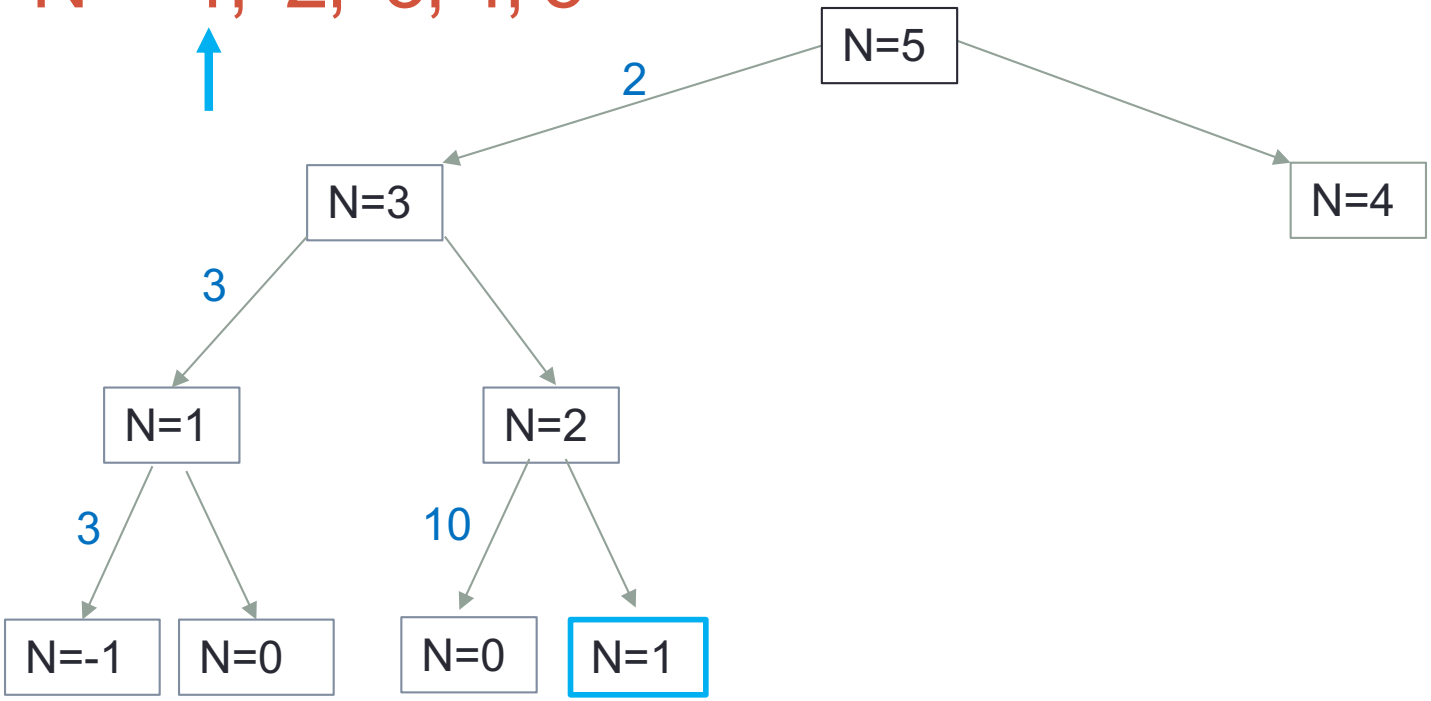
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

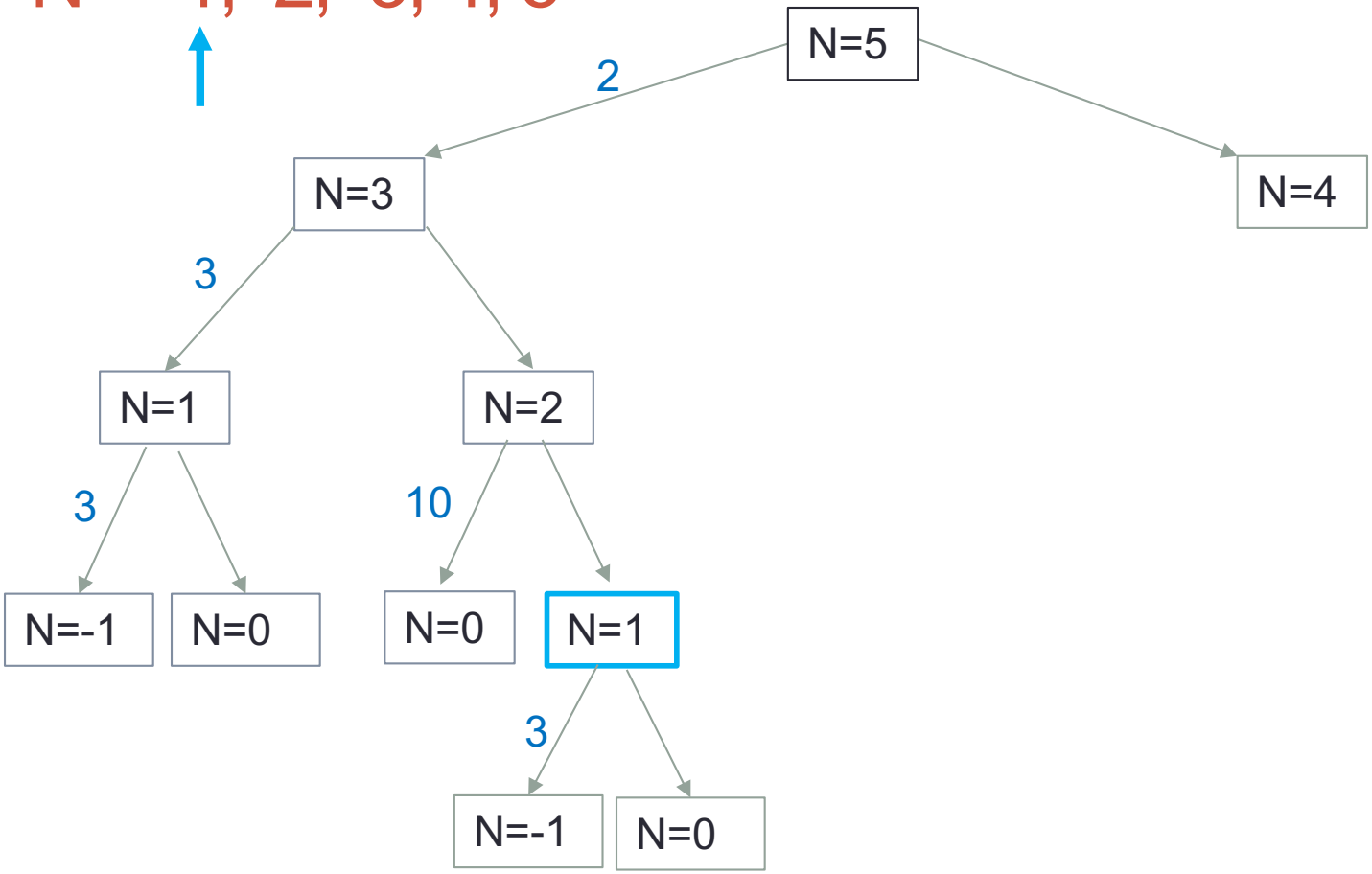
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

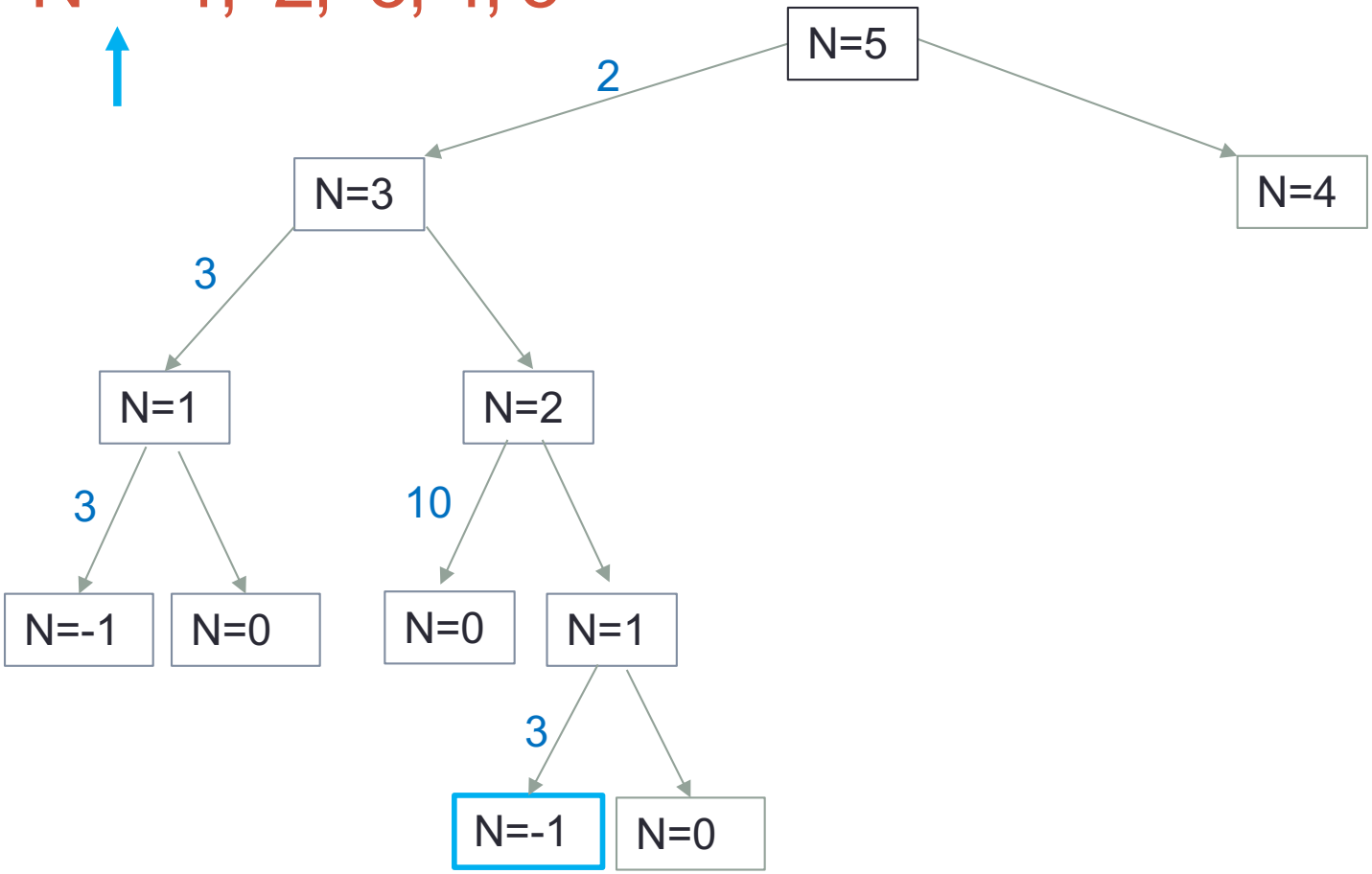
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

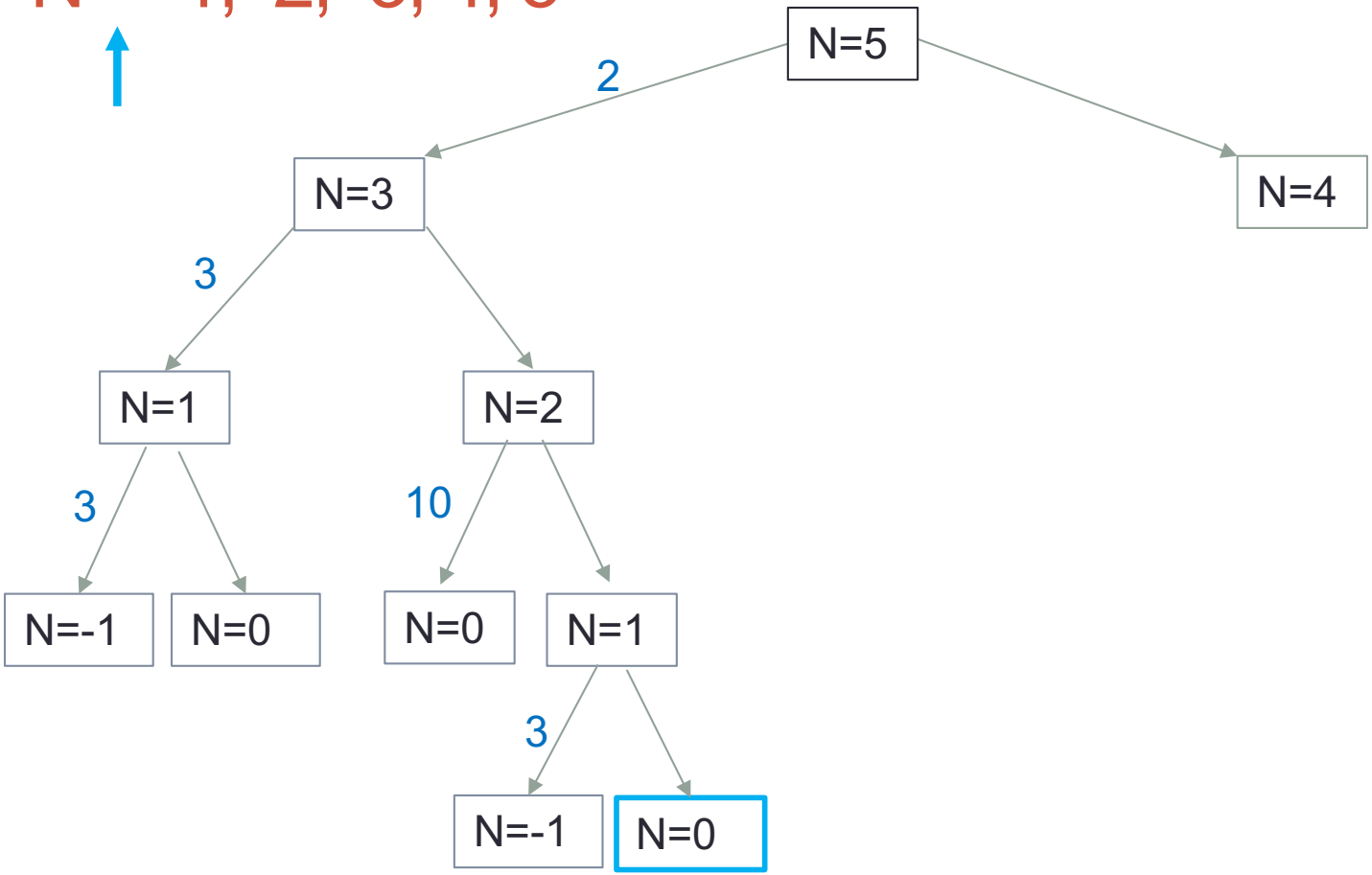
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

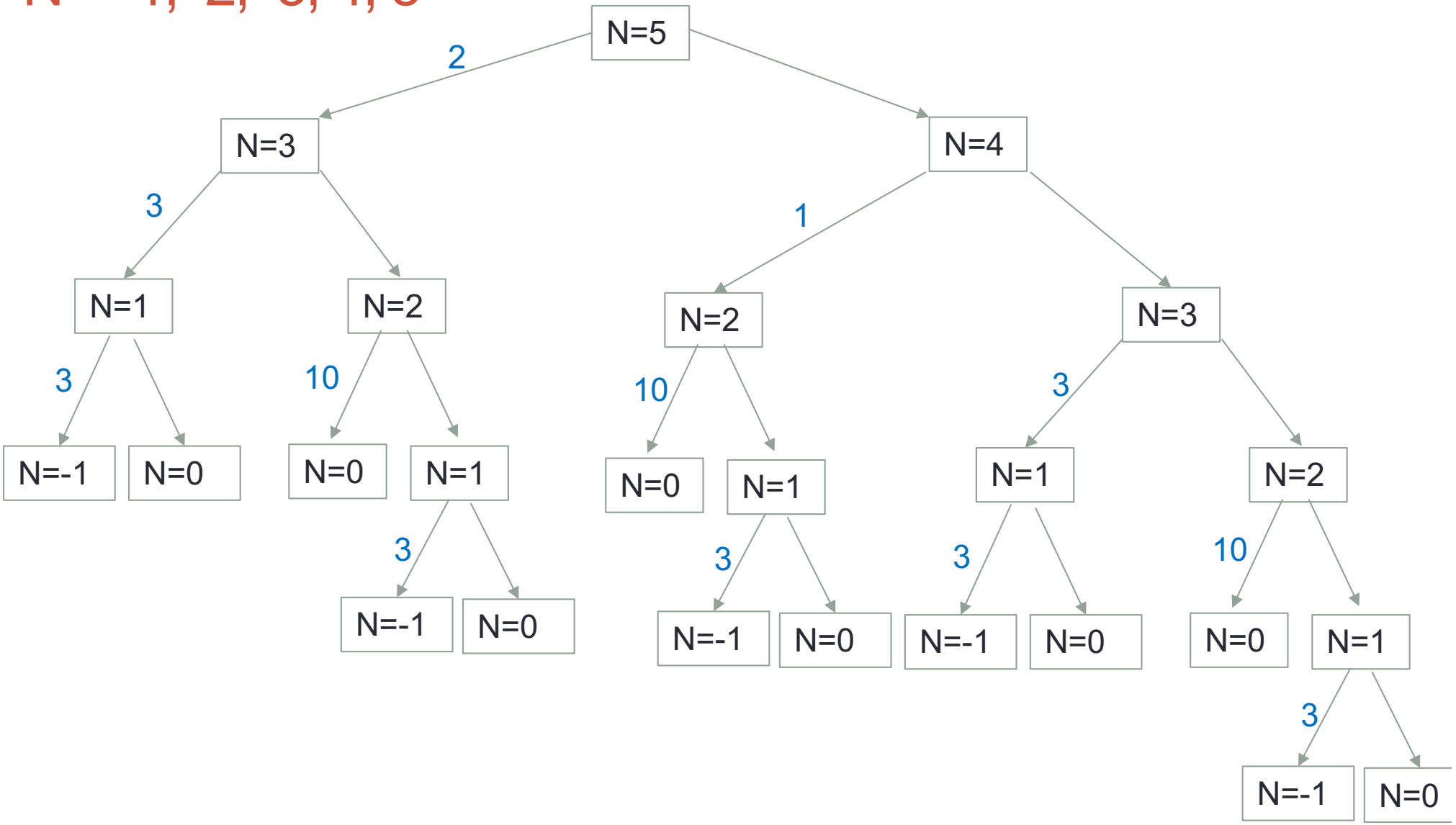
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

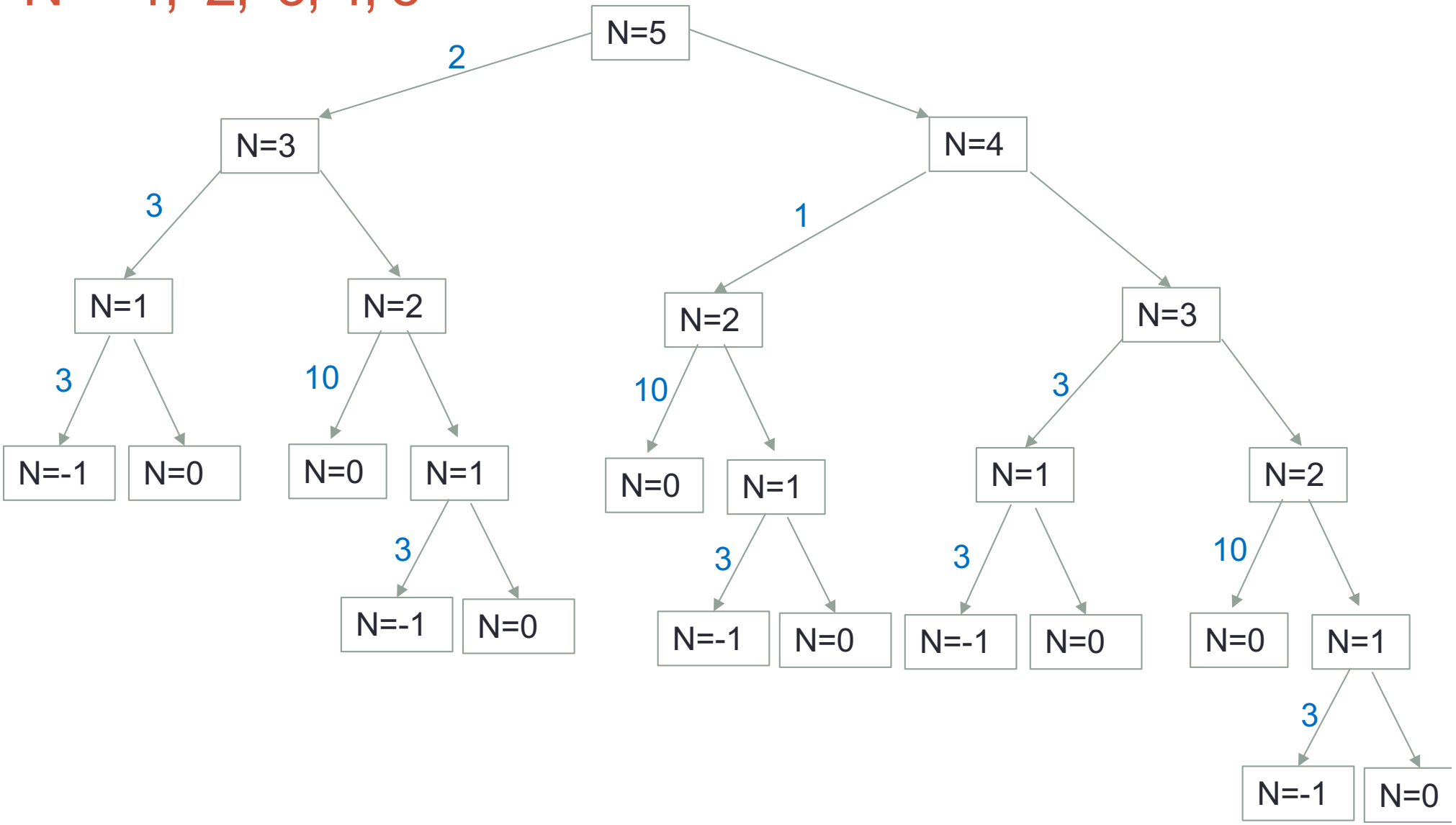
N = 1, 2, 3, 4, 5



Construyendo el árbol de todas las posibles llamadas recursivas …

V = [3, 10, 3, 1, 2]

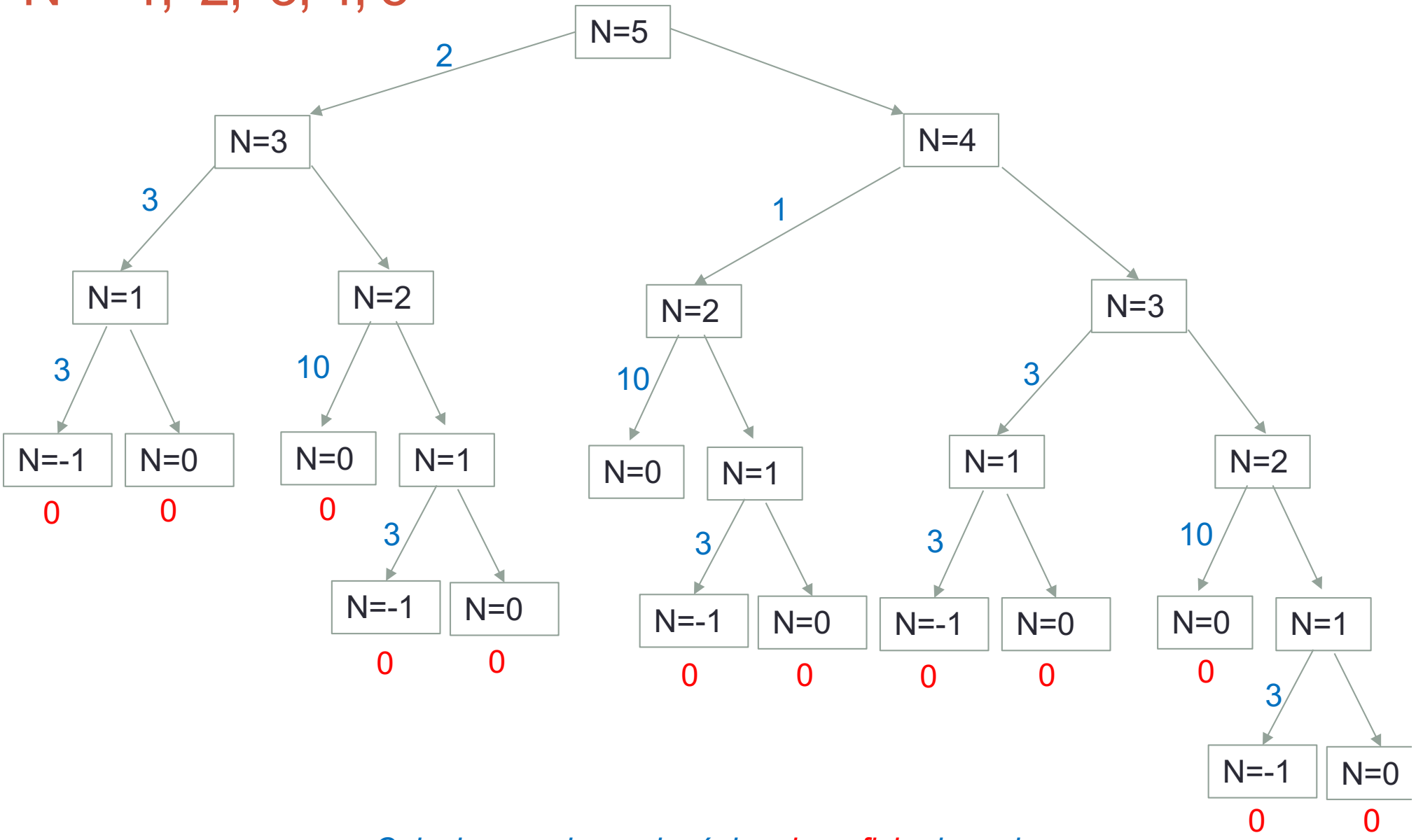
N = 1, 2, 3, 4, 5



Calculemos ahora el máximo beneficio de cada rama

V = [3, 10, 3, 1, 2]

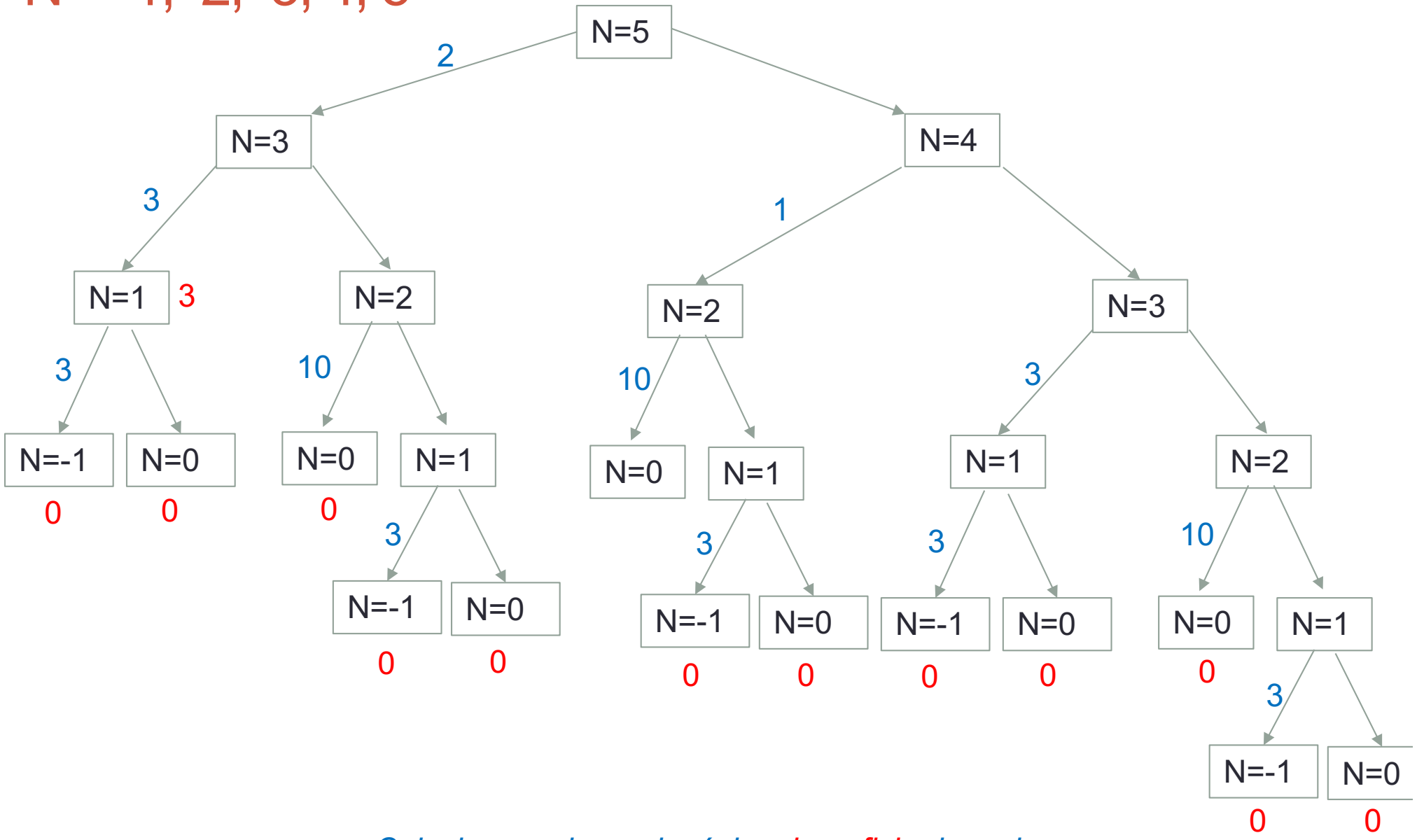
N = 1, 2, 3, 4, 5



Calculemos ahora el máximo beneficio de cada rama

V = [3, 10, 3, 1, 2]

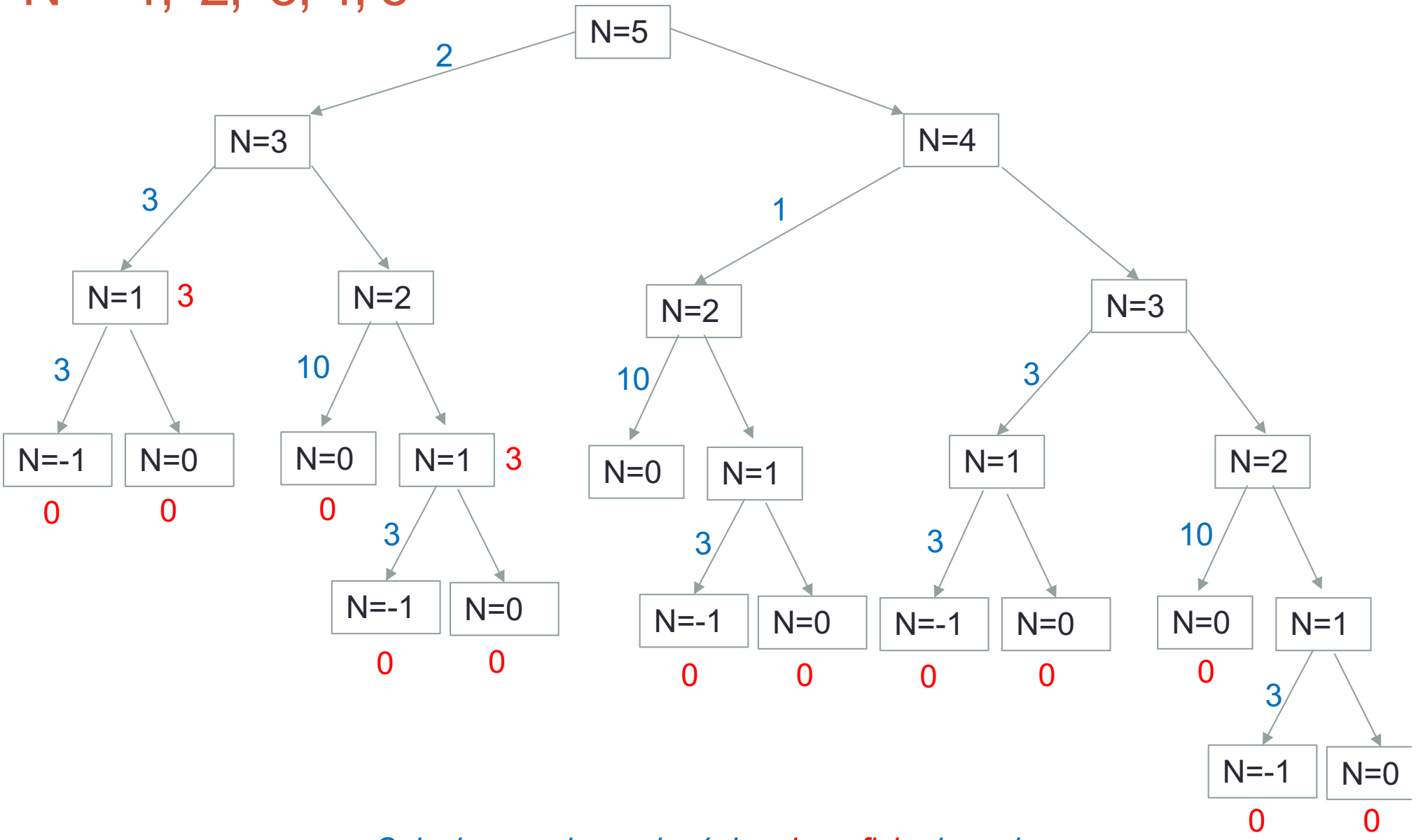
N = 1, 2, 3, 4, 5



Calculemos ahora el máximo beneficio de cada rama

V = [3, 10, 3, 1, 2]

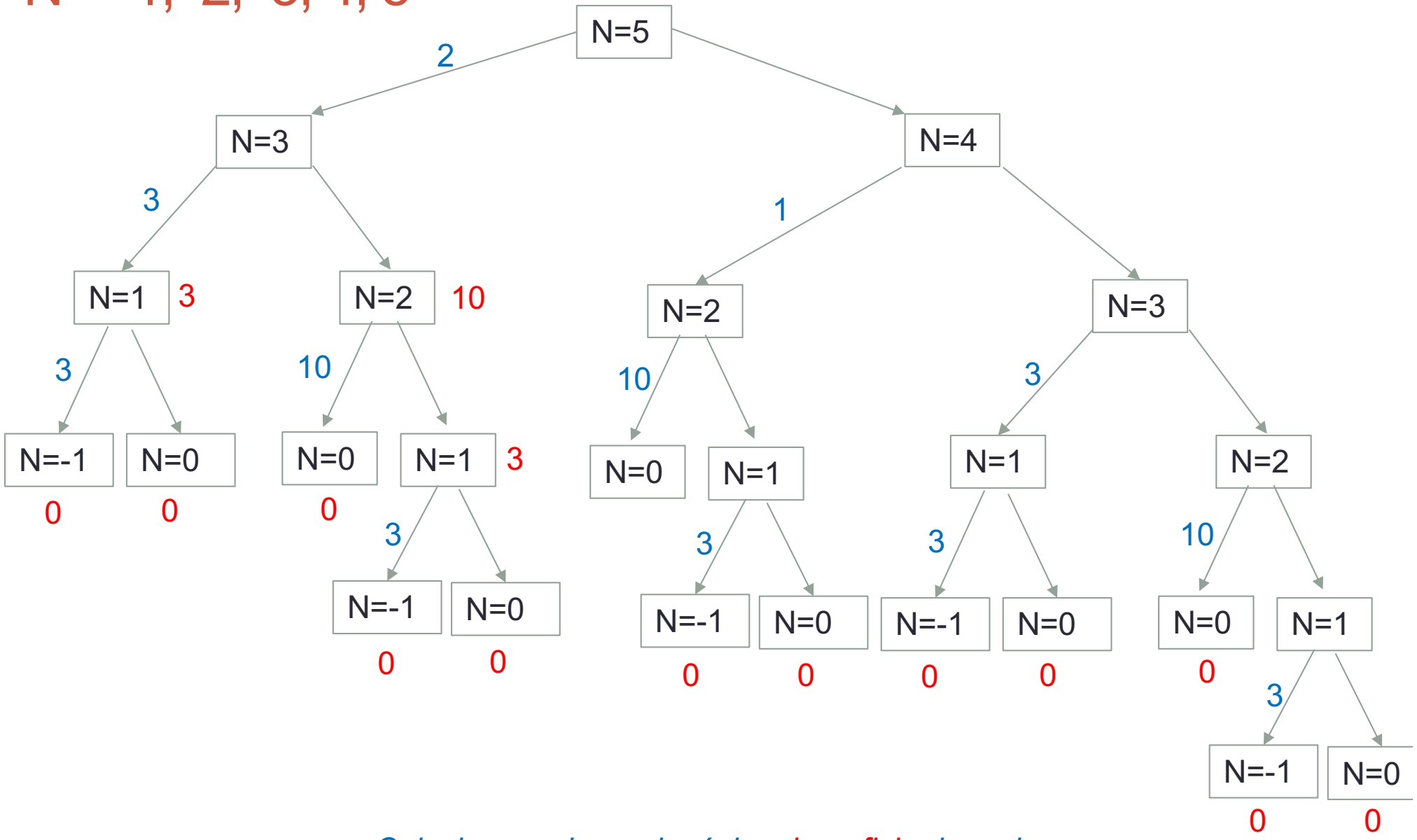
N = 1, 2, 3, 4, 5



Calculemos ahora el máximo beneficio de cada rama

V = [3, 10, 3, 1, 2]

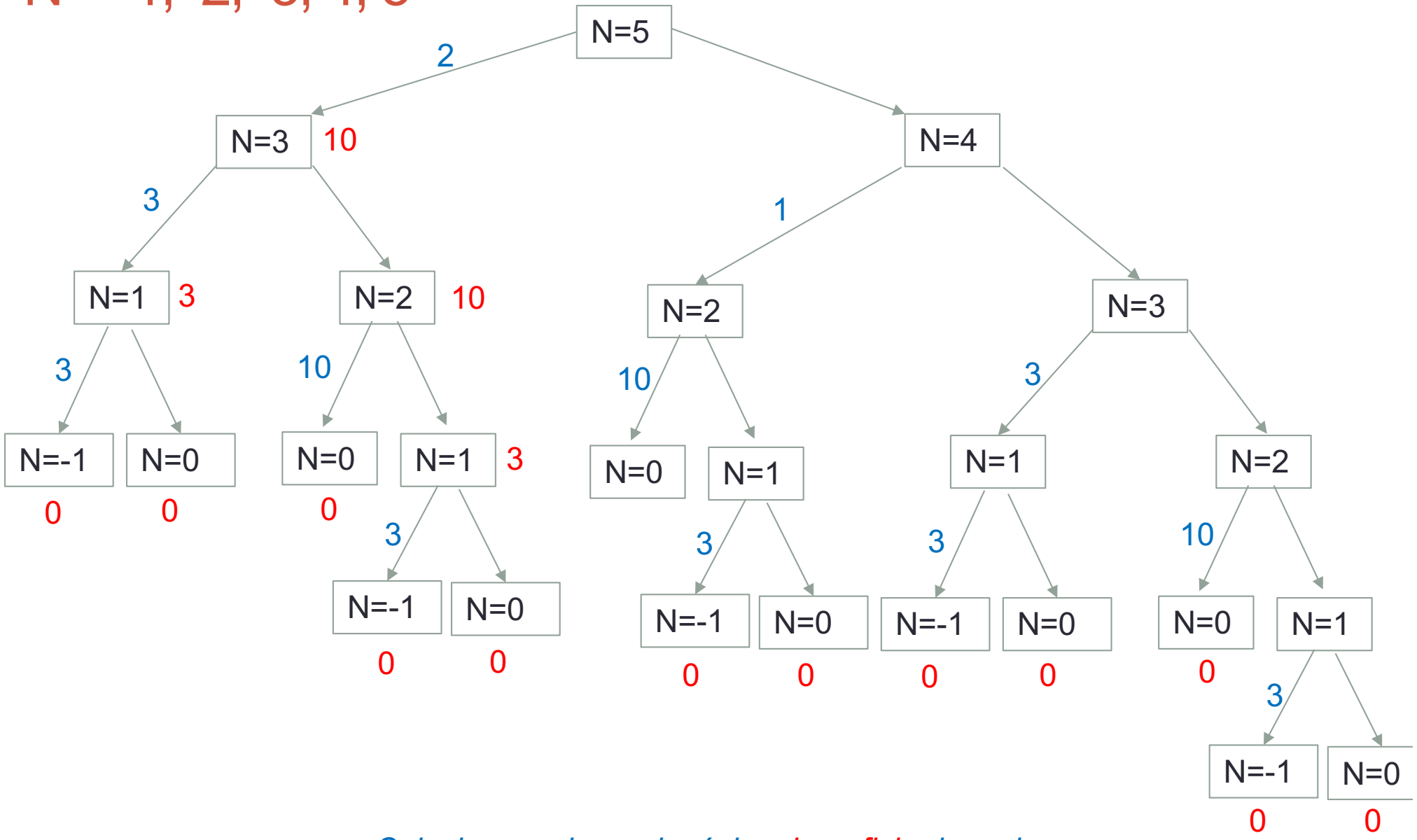
N = 1, 2, 3, 4, 5



Calculemos ahora el máximo beneficio de cada rama

V = [3, 10, 3, 1, 2]

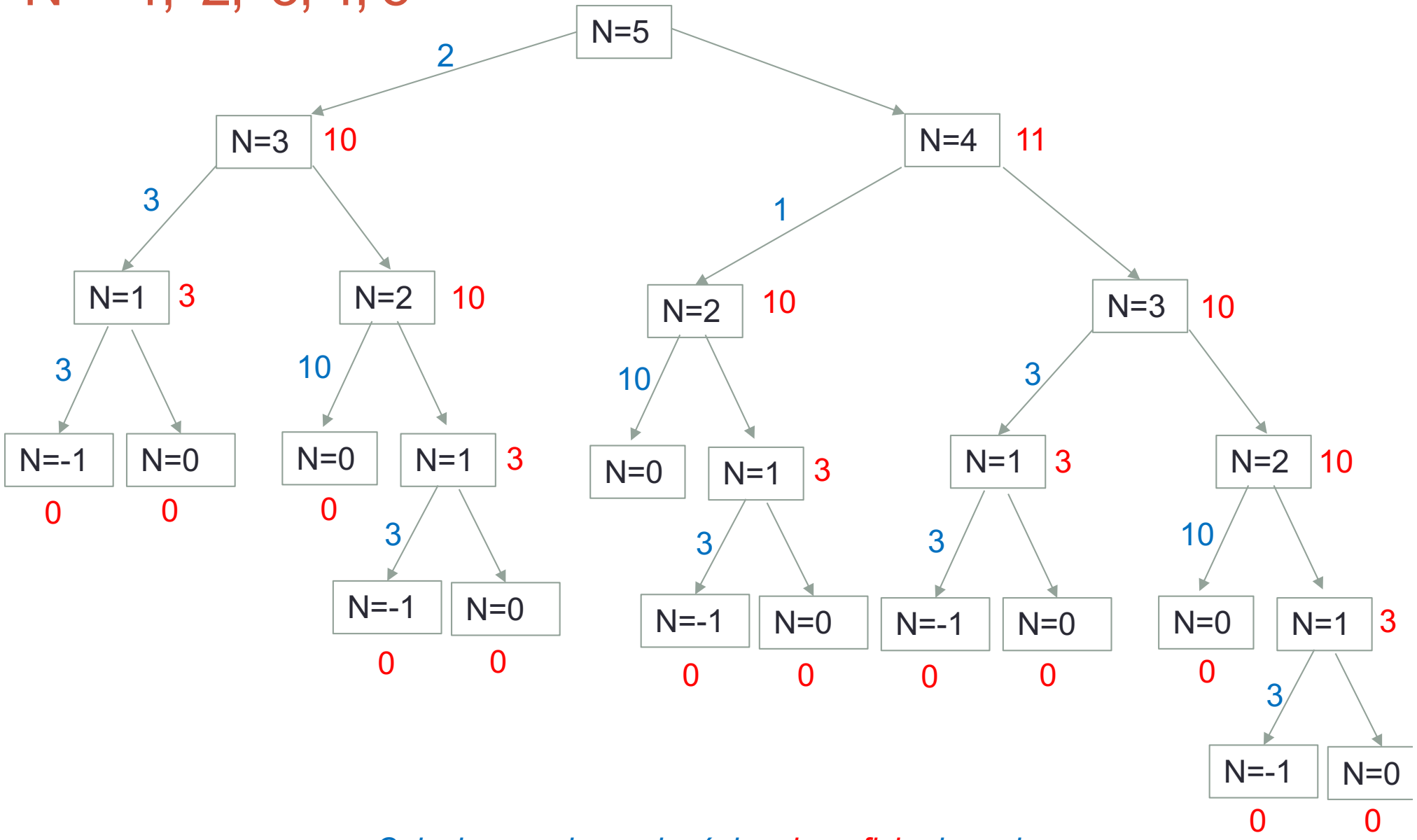
N = 1, 2, 3, 4, 5



Calculemos ahora el máximo beneficio de cada rama

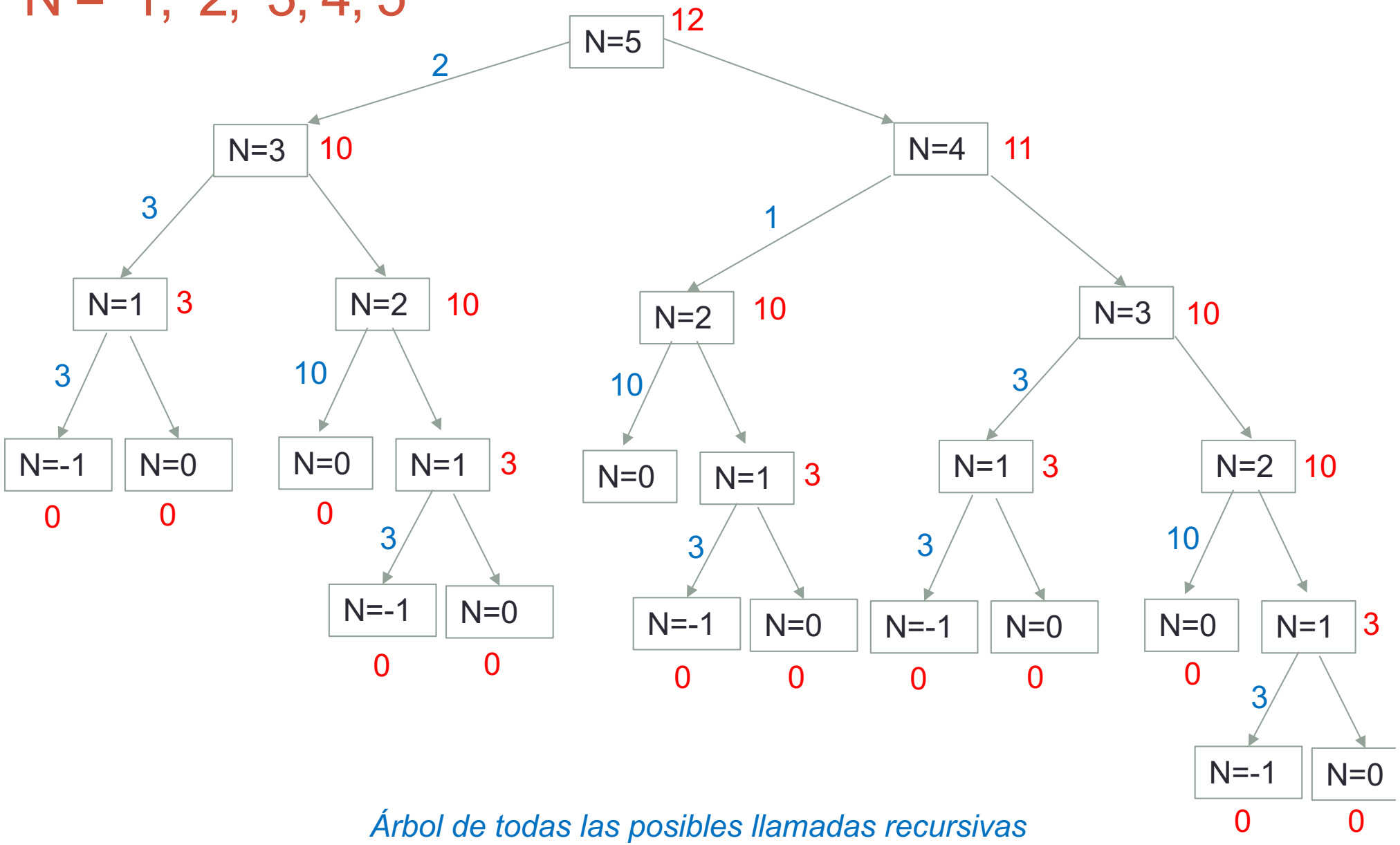
V = [3, 10, 3, 1, 2]

N = 1, 2, 3, 4, 5



Calculemos ahora el máximo beneficio de cada rama

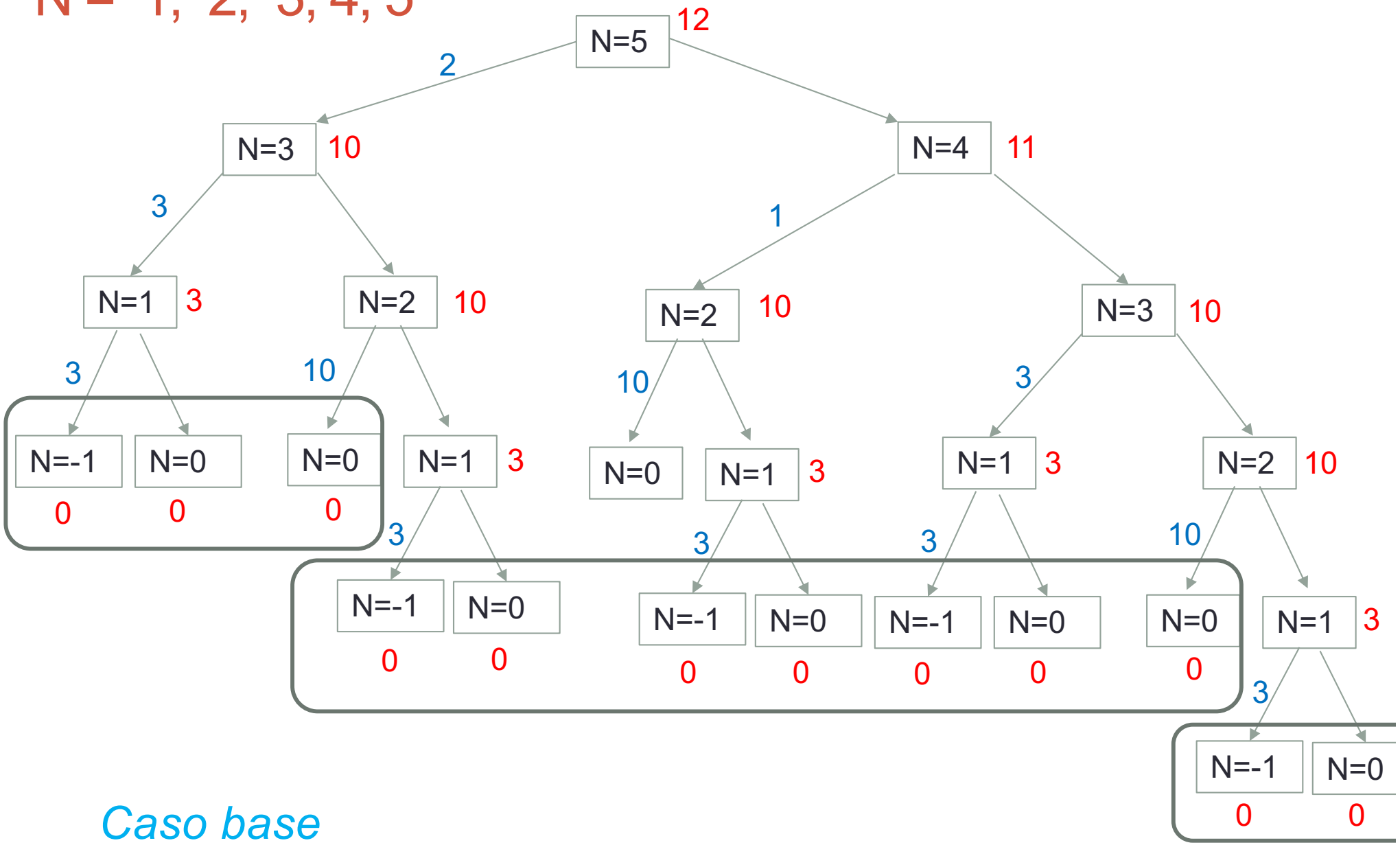
V = [3, 10, 3, 1, 2]
N = 1, 2, 3, 4, 5



Árbol de todas las posibles llamadas recursivas con el máximo beneficio de cada rama

V = [3, 10, 3, 1, 2]

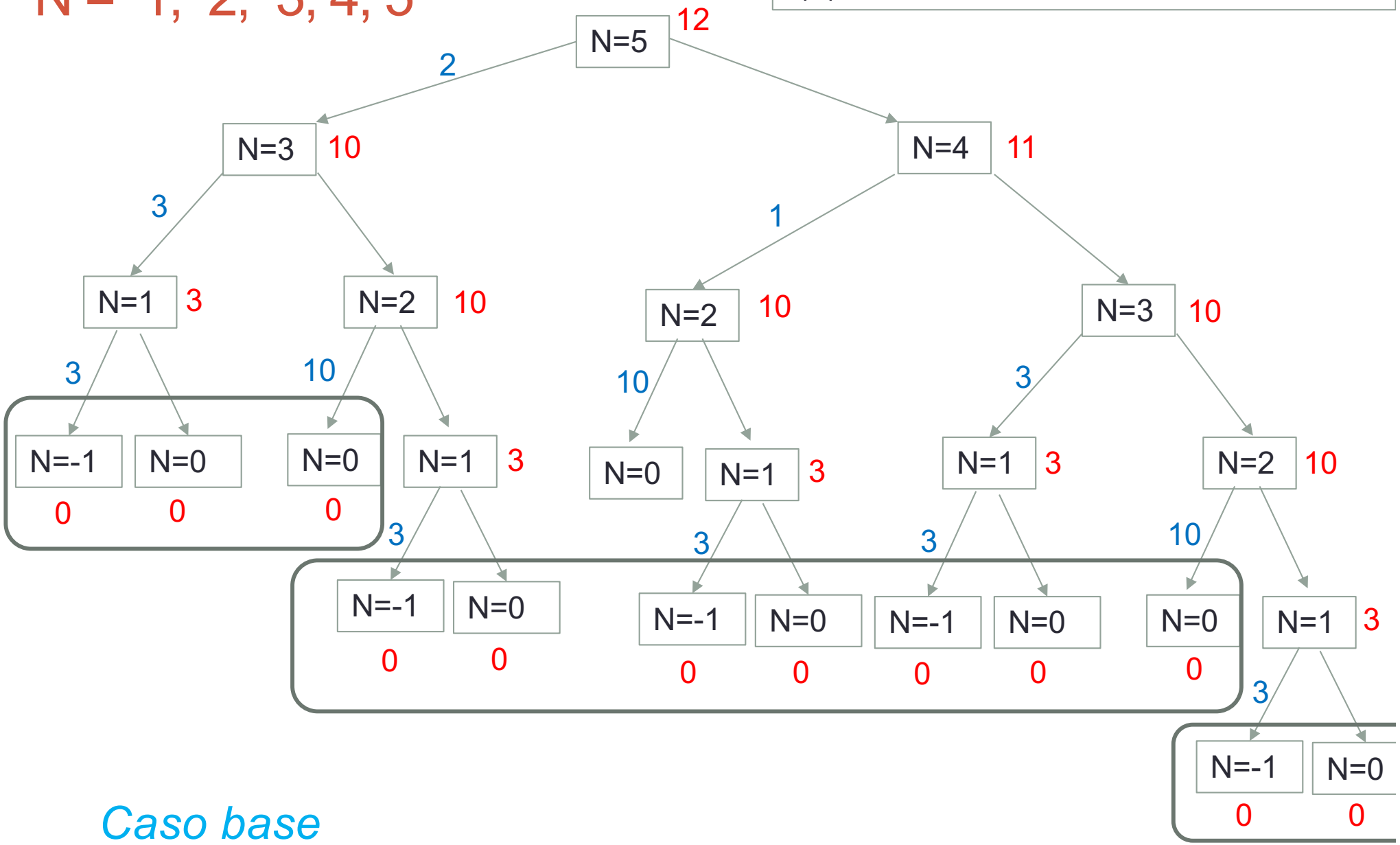
N = 1, 2, 3, 4, 5



Caso base

V = [3, 10, 3, 1, 2]
N = 1, 2, 3, 4, 5

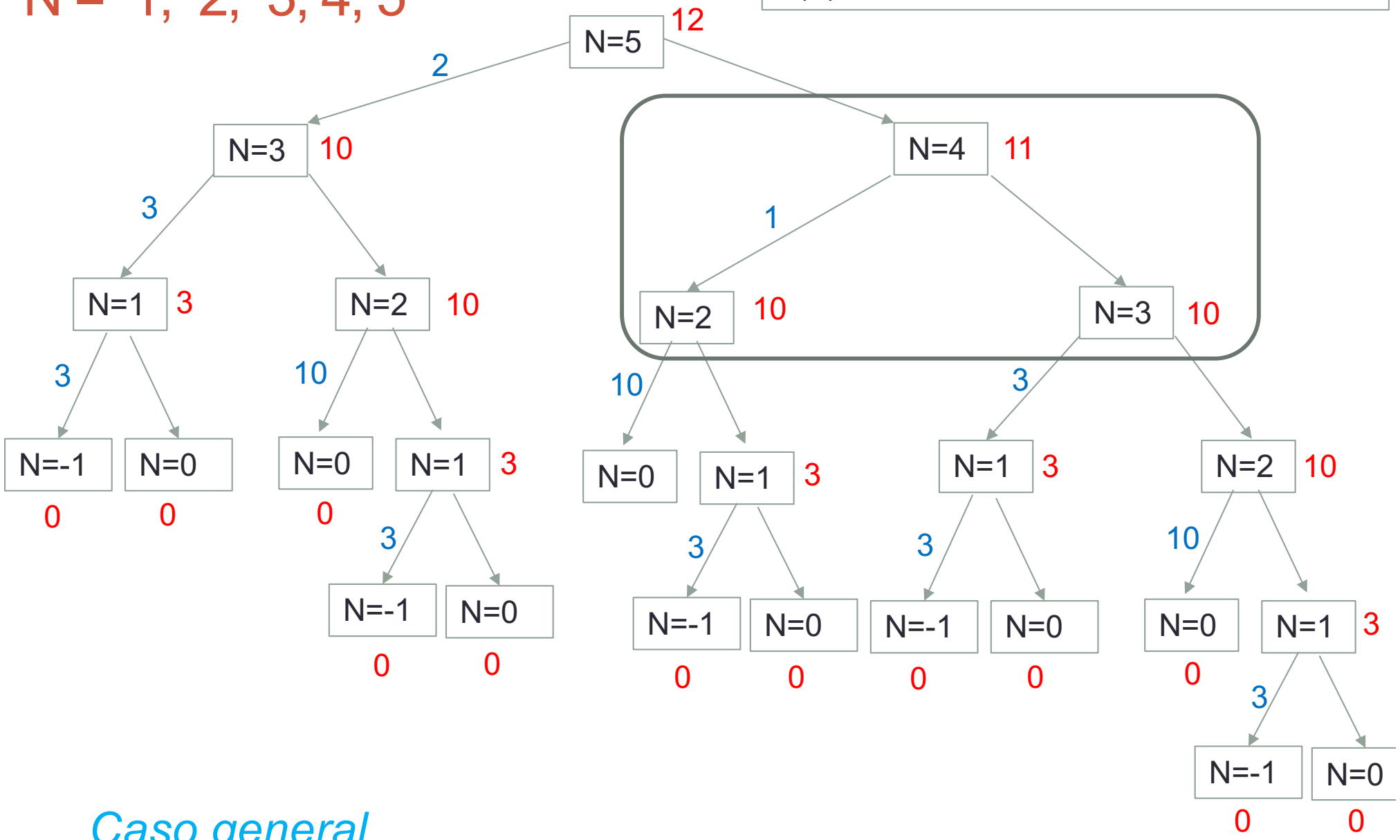
t(n) = 0 : n<=0



Caso base

V = [3, 10, 3, 1, 2]
N = 1, 2, 3, 4, 5

t(n) = 0 : n<=0



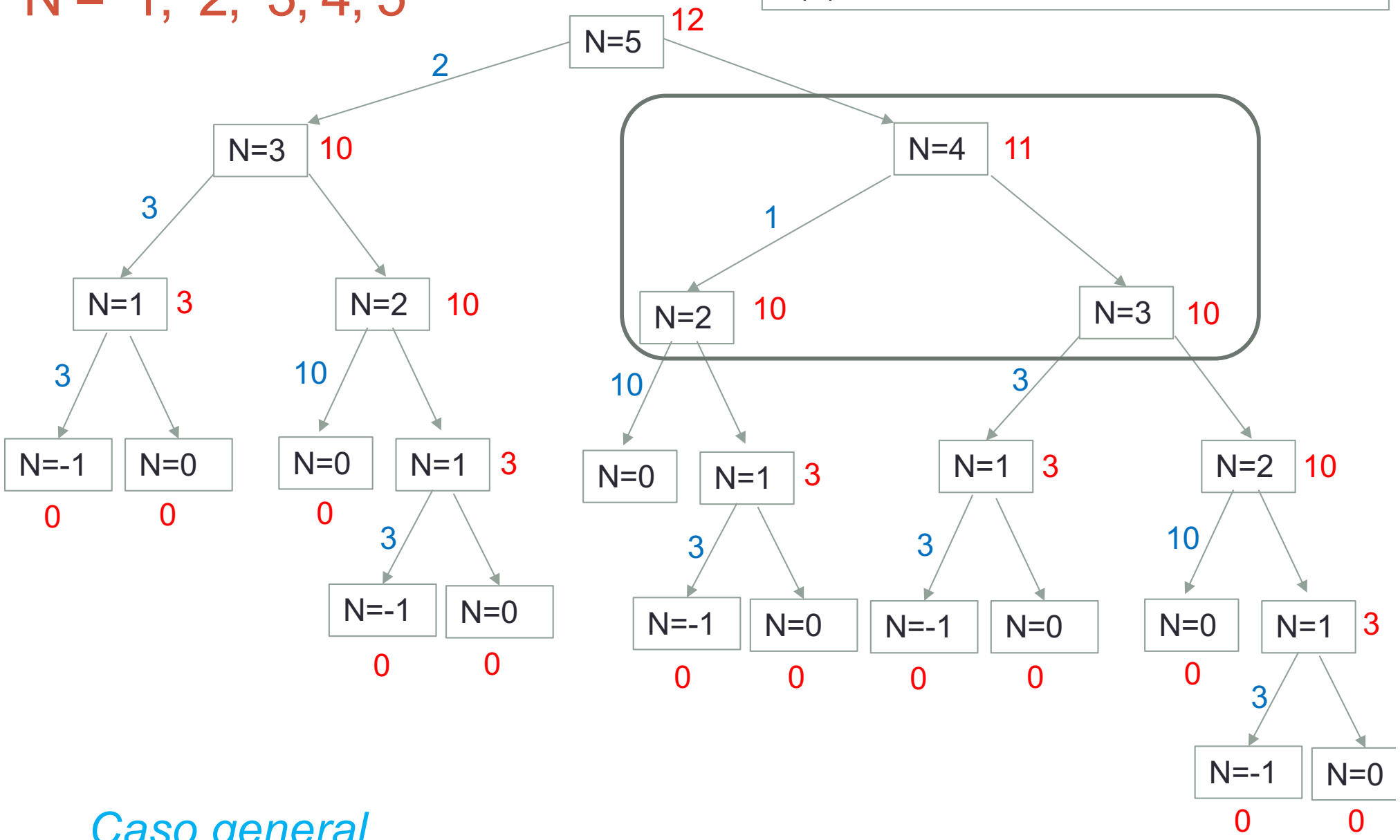
Caso general

V = [3, 10, 3, 1, 2]

N = 1, 2, 3, 4, 5

t(n) = max (t(n-2) + v[n], t(n-1))

t(n) = 0 : n<=0



Caso general

Problema del ladrón de casas

Recurrencia:

$$t(n) = \max(t(n-2) + v[n], t(n-1))$$

$$t(n) = 0 \quad : \text{si } n<=0$$

Dependiendo del lenguaje de programación que utilicemos para programar nuestra recurrencia, ajustamos nuestro caso base para que no se nos salga de la lista.

Solución en Python utilizando Programación Dinámica (*Tabulation*)

Recurrencia:

t(n) = max (t(n-2) + v[n], t(n-1))

t(n) = 0 : si n<=0

V = [3, 10, 3, 1, 2]

Table = [3, 10, 10, 11, 12]

¡ No hemos terminado !

- Nos falta completarlo para que calcule qué casas forman la mejor elección
- En este ejemplo debe indicar que son la segunda y la quinta.

¡ No hemos terminado !

Hemos visto que utilizando *tabulation* este es el resultado que queda en su tabla

V = [3, 10, 3, 1, 2]

Table = [3, 10, 10, 11, 12]

¡ No hemos terminado !

Hemos visto que utilizando *tabulation* este es el resultado que queda en su tabla

V = [3, 10, 3, 1, 2]

Table = [3, 10, 10, 11, 12]

… sólo tenemos que recorrer la tabla desde el final hasta el principio para saber qué casas eligió el ladrón !