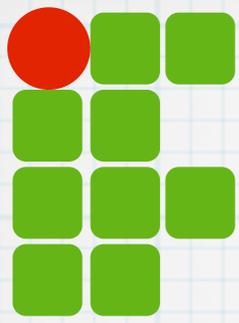


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Algoritmos

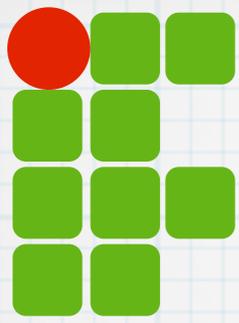
Programação dinâmica

Copyright © 2014 IFRN

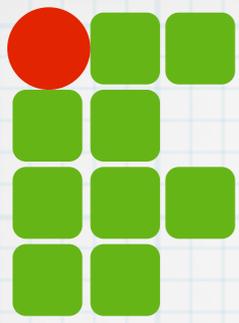


Agenda

- * Programação dinâmica
- * Exemplos
- * Exercícios

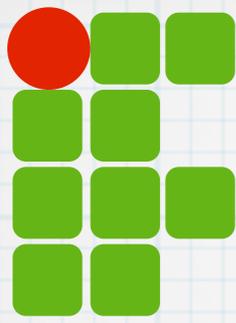


O que é?



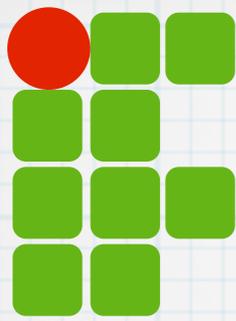
O que é?

*** Problemas resolúveis recursivamente (Combinação de soluções de subproblemas)**



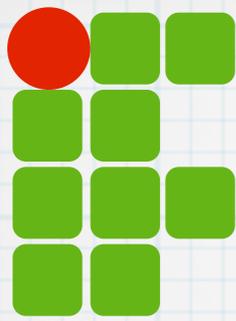
O que é?

- * Problemas resolúveis recursivamente (Combinação de soluções de subproblemas)
- * ... Resolução recursiva direta duplica trabalho (resolução repetida do mesmo subproblema)



O que é?

- * Problemas resolúveis recursivamente (Combinação de soluções de subproblemas)
 - * ... Resolução recursiva direta duplica trabalho (resolução repetida do mesmo subproblema)
- * Abordagem:



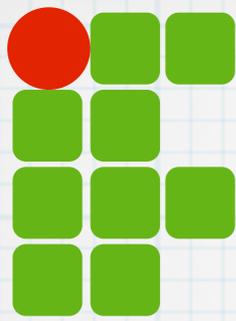
O que é?

*** Problemas resolúveis recursivamente (Combinação de soluções de subproblemas)**

*** ... Resolução recursiva direta duplica trabalho (resolução repetida do mesmo subproblema)**

*** Abordagem:**

1) Economizar tempo (evitar repetir trabalho), memorizando as soluções parciais dos sub-problemas (gastando memória!)



O que é?

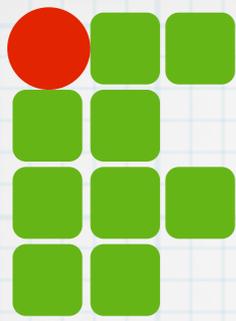
* Problemas resolúveis recursivamente (Combinação de soluções de subproblemas)

* ... Resolução recursiva direta duplica trabalho (resolução repetida do mesmo subproblema)

* Abordagem:

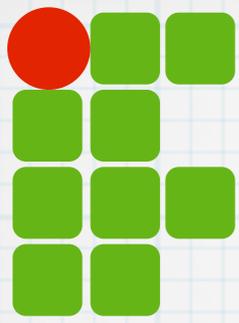
1) Economizar tempo (evitar repetir trabalho), memorizando as soluções parciais dos sub-problemas (gastando memória!)

2) Economizar memória, resolvendo sub-problemas por ordem que minimiza n° de soluções parciais a memorizar (bottom-up, começando pelos casos base)



O que é?

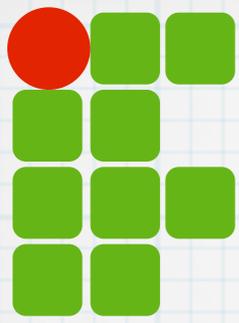
- * **Problemas resolúveis recursivamente (Combinação de soluções de subproblemas)**
 - * ... Resolução recursiva direta duplica trabalho (resolução repetida do mesmo subproblema)
- * **Abordagem:**
 - 1) Economizar tempo (evitar repetir trabalho), memorizando as soluções parciais dos sub-problemas (gastando memória!)
 - 2) Economizar memória, resolvendo sub-problemas por ordem que minimiza nº de soluções parciais a memorizar (bottom-up, começando pelos casos base)
- * **Termo “Programação” vem da Pesquisa Operacional, no sentido de formular restrições ao problema que tornam um método aplicável**



Exemplo

* Combinação de n elementos, k a k

$$C_k^n = \frac{n!}{k! \times (n - k)!}$$

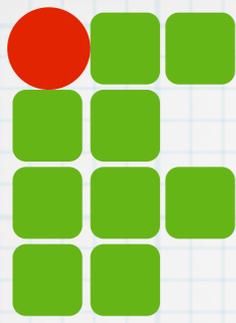


Exemplo

* Combinação de n elementos, k a k

$$C_k^n = \frac{n!}{k! \times (n - k)!}$$

$$C_k^n = \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{Se } k = n \\ C_{k-1}^{n-1} + C_k^{n-1} & \text{Se } 1 < k < n \end{cases}$$

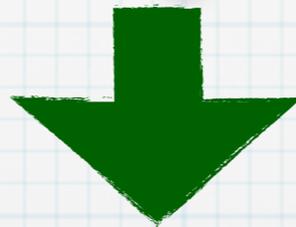


Exemplo

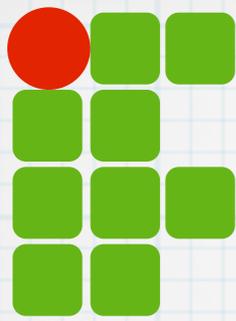
* Combinação de n elementos, k a k

$$C_k^n = \frac{n!}{k! \times (n - k)!}$$

$$C_k^n = \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{Se } k = n \\ C_{k-1}^{n-1} + C_k^{n-1} & \text{Se } 1 < k < n \end{cases}$$



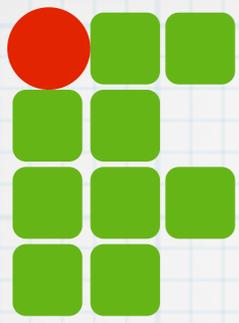
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$



Exemplo

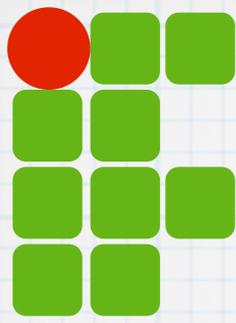
```
int comb(int n, int k) {  
    if (k == 1)  
        return n;  
    if (n == k)  
        return 1;  
    return (comb(n - 1, k - 1) + comb(n, k - 1));  
}
```

$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$



Exemplo

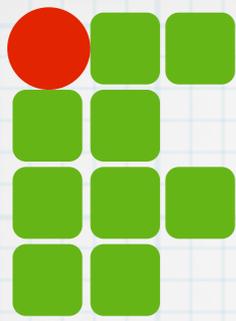
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$



Exemplo

$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

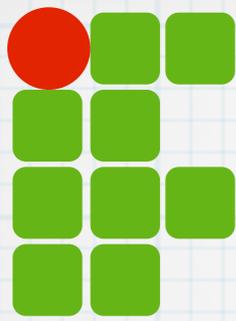
Comb(6,3)



Exemplo

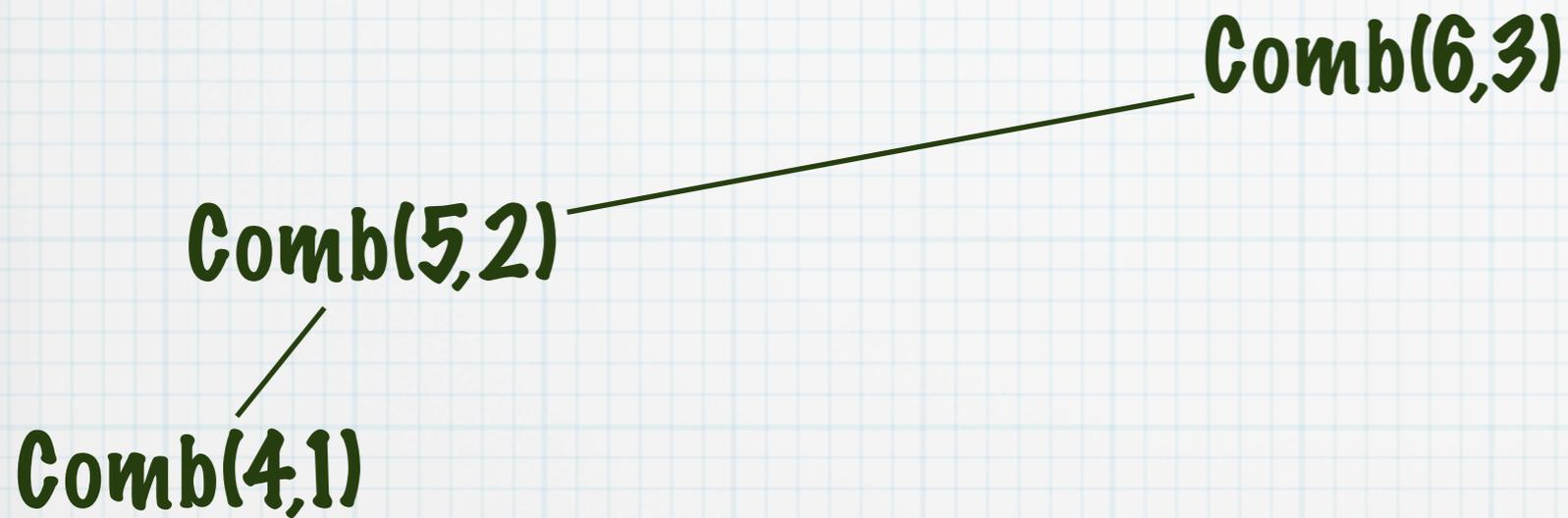
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

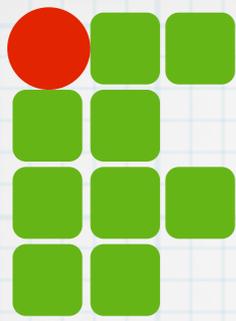
Comb(5,2) ————— **Comb(6,3)**



Exemplo

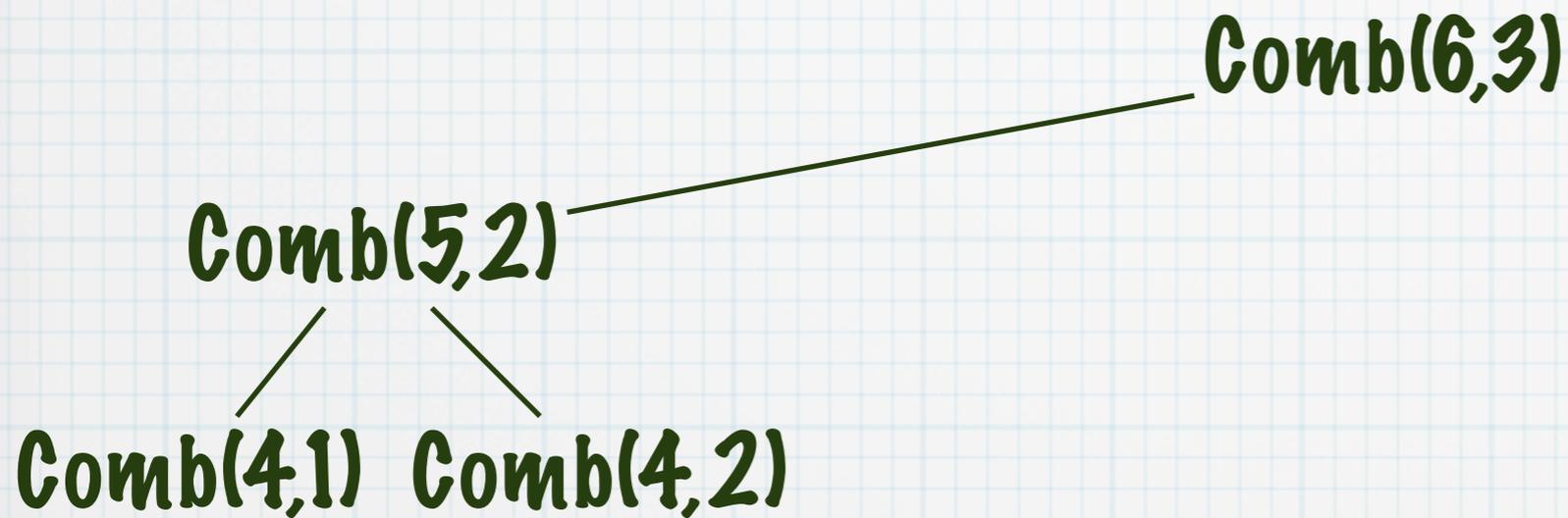
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

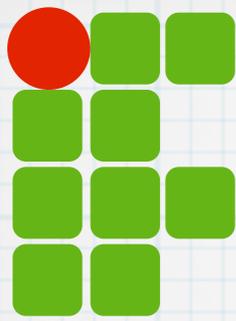




Exemplo

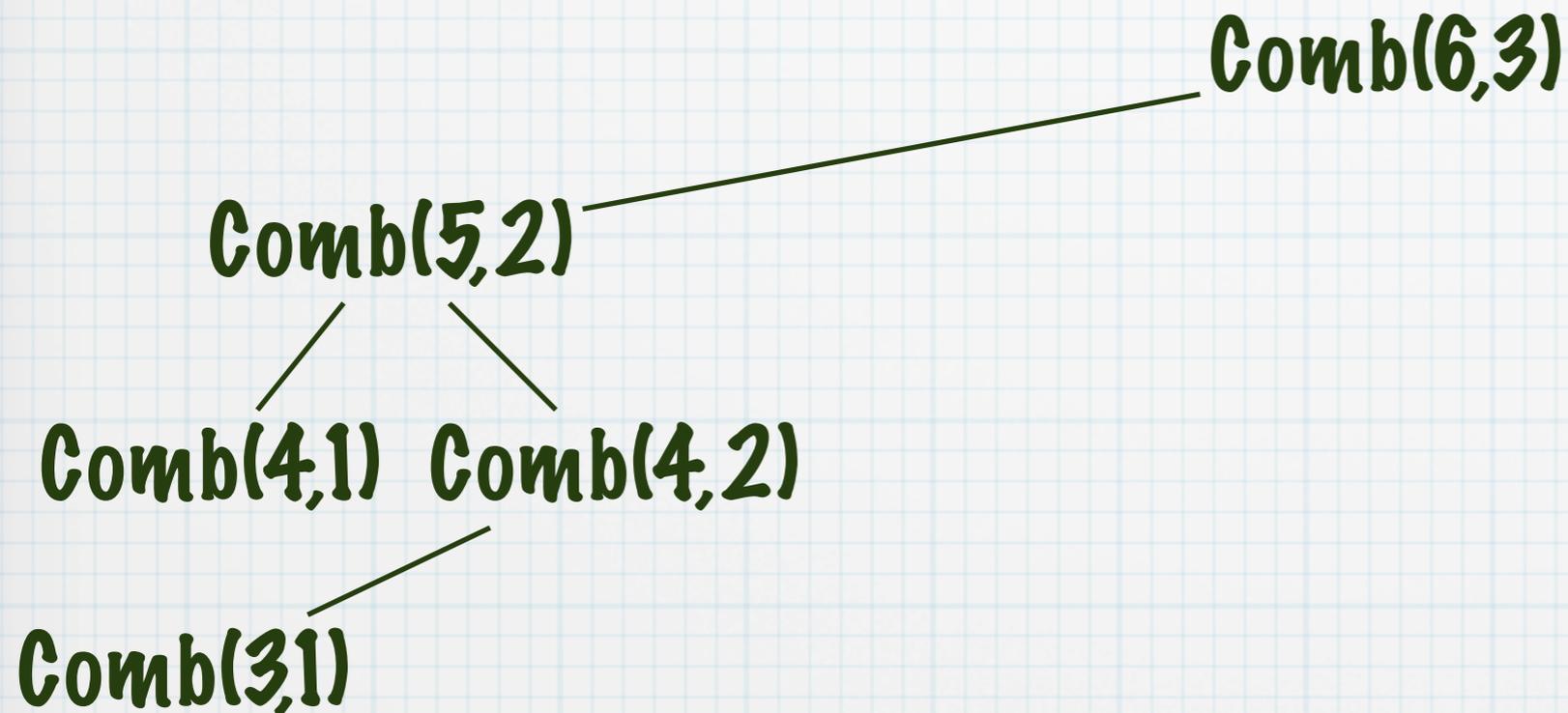
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

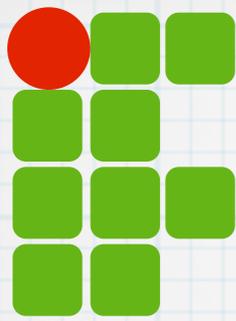




Exemplo

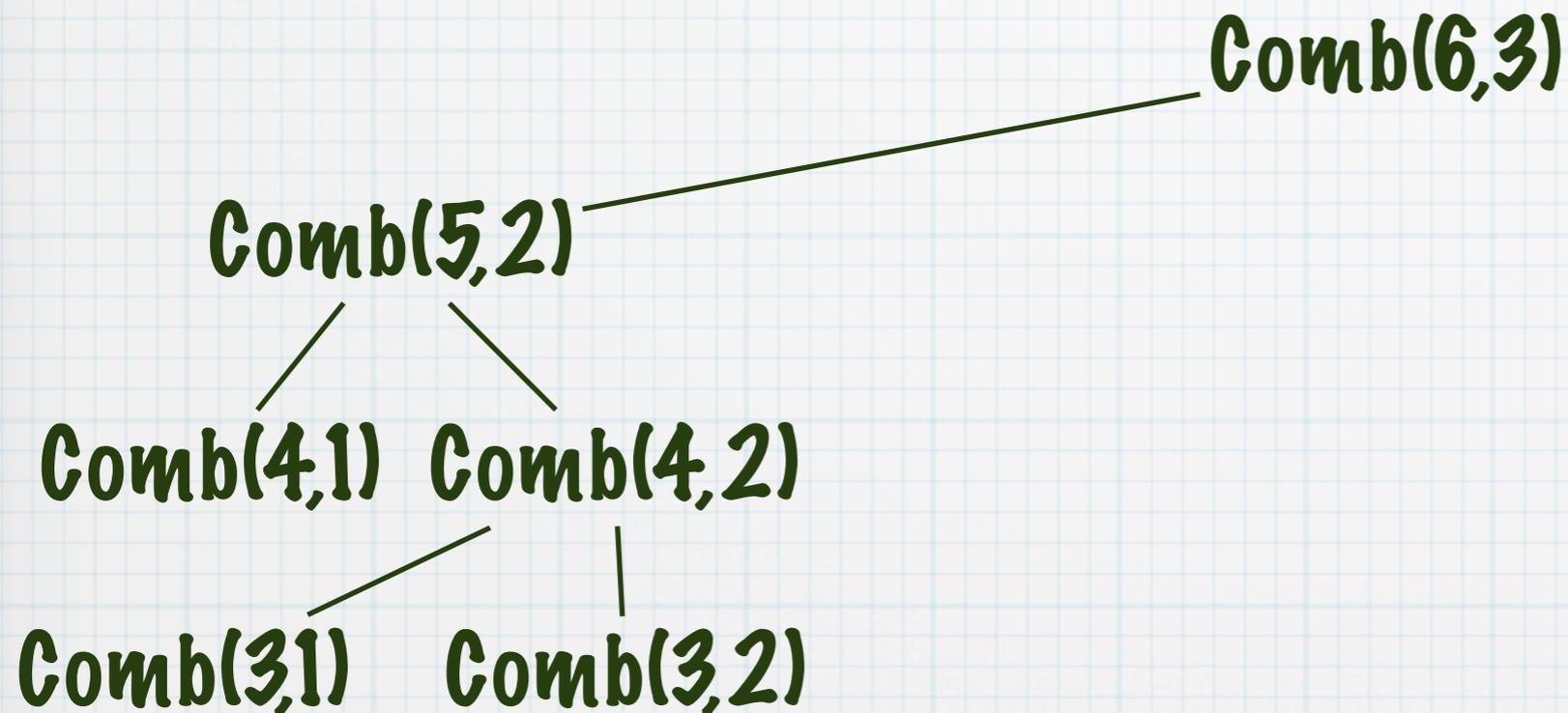
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

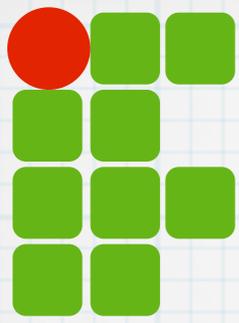




Exemplo

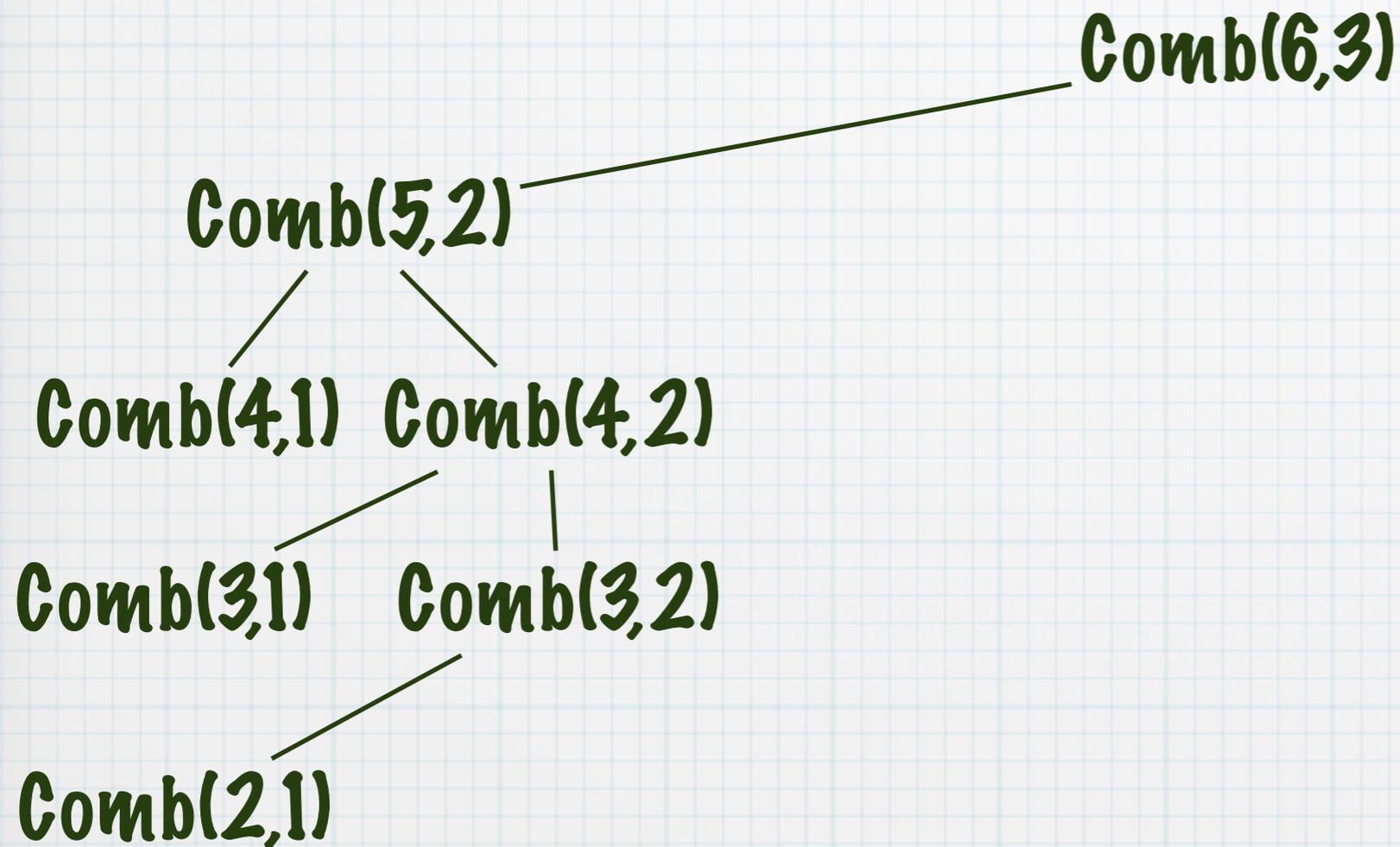
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

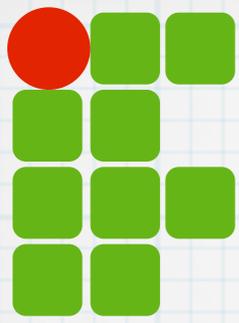




Exemplo

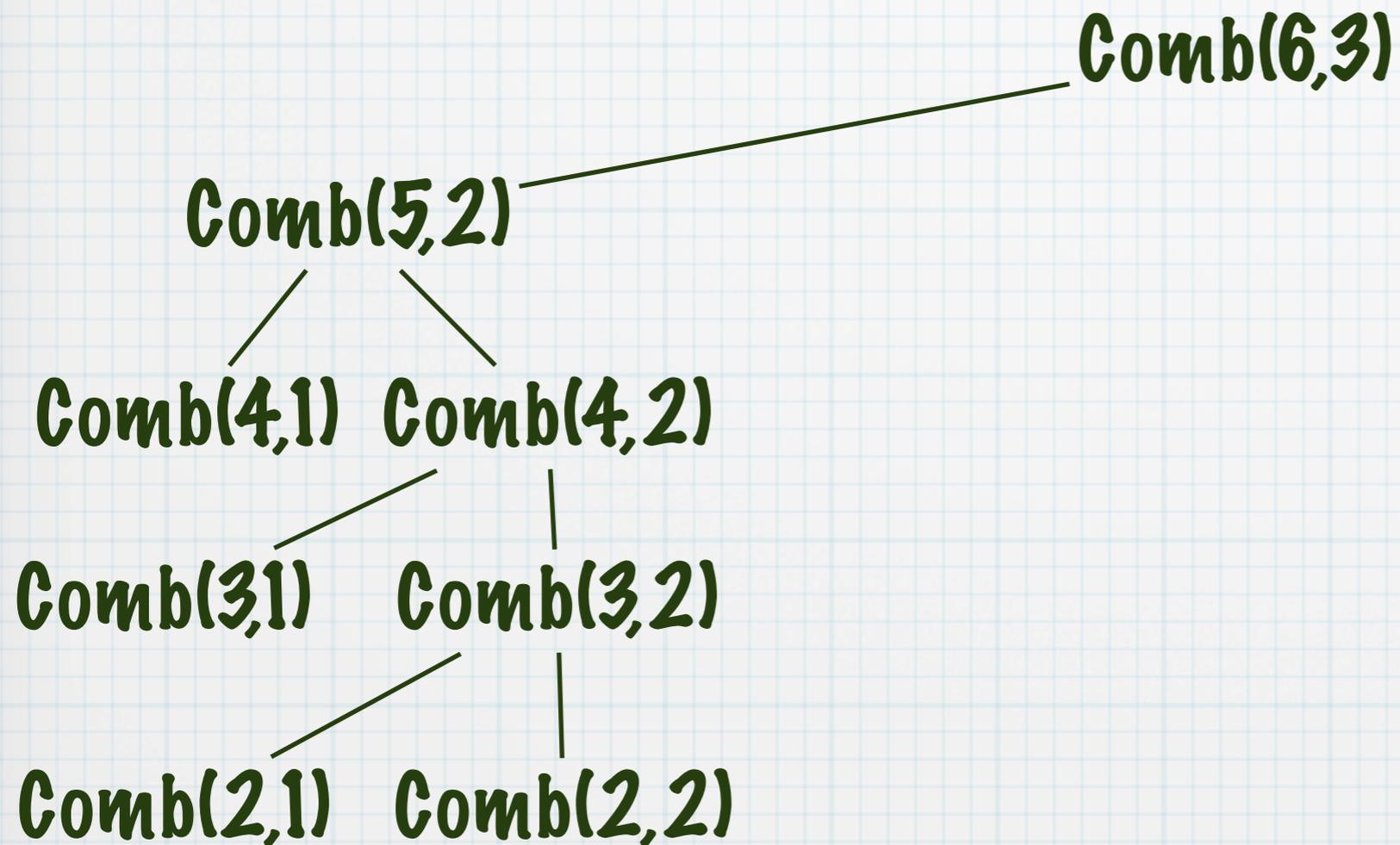
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

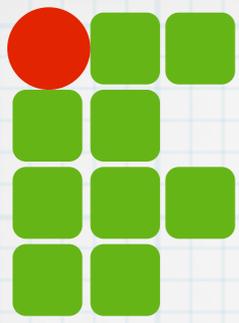




Exemplo

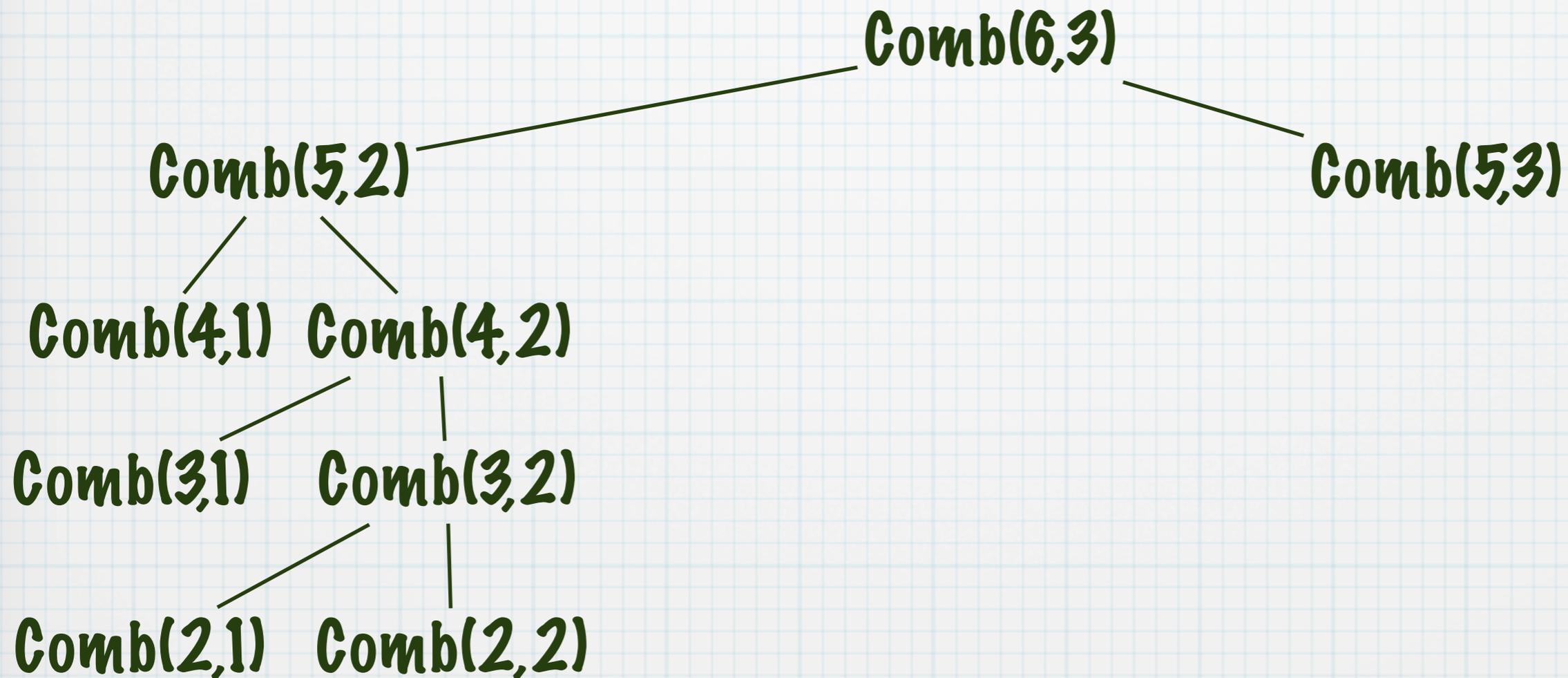
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

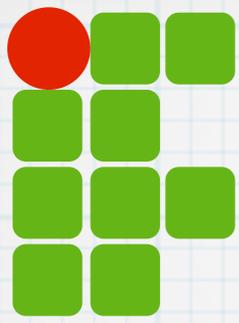




Exemplo

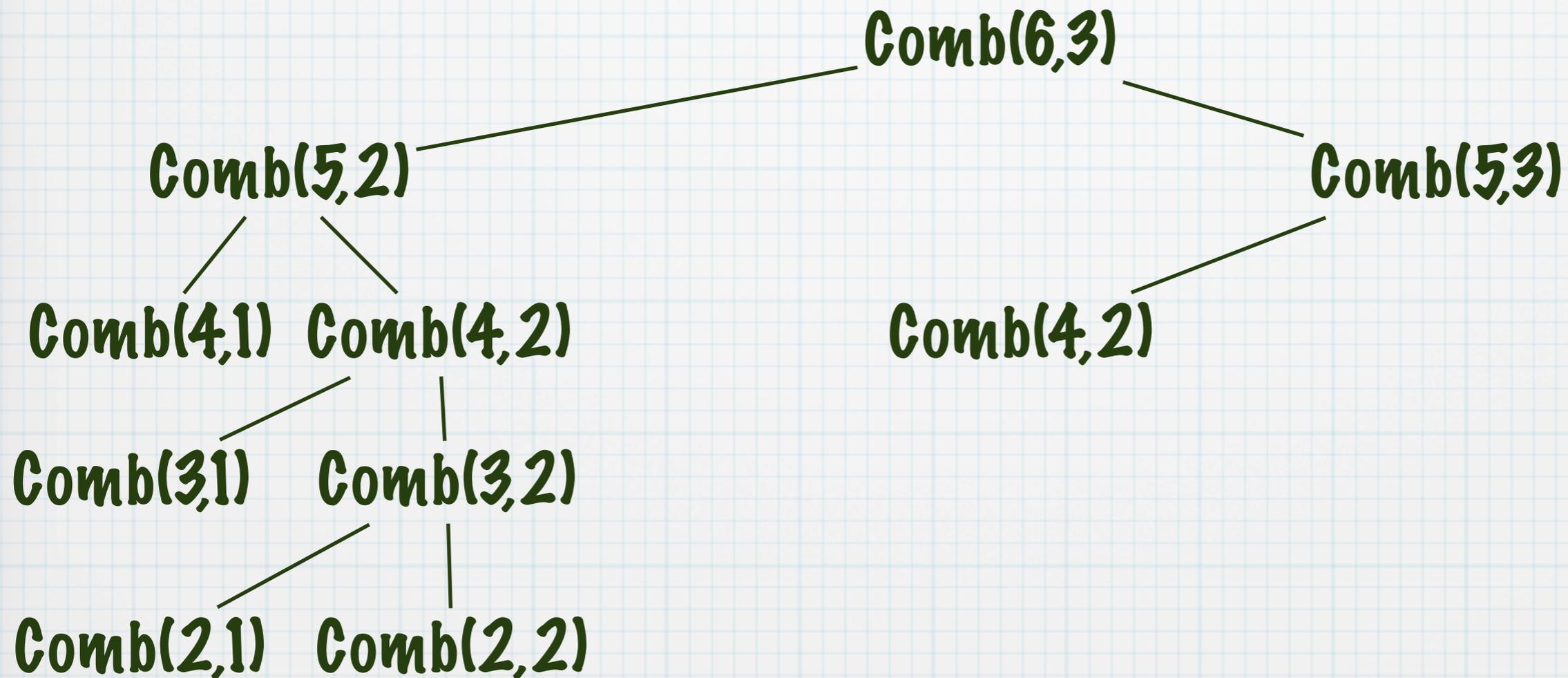
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

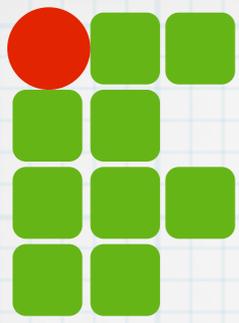




Exemplo

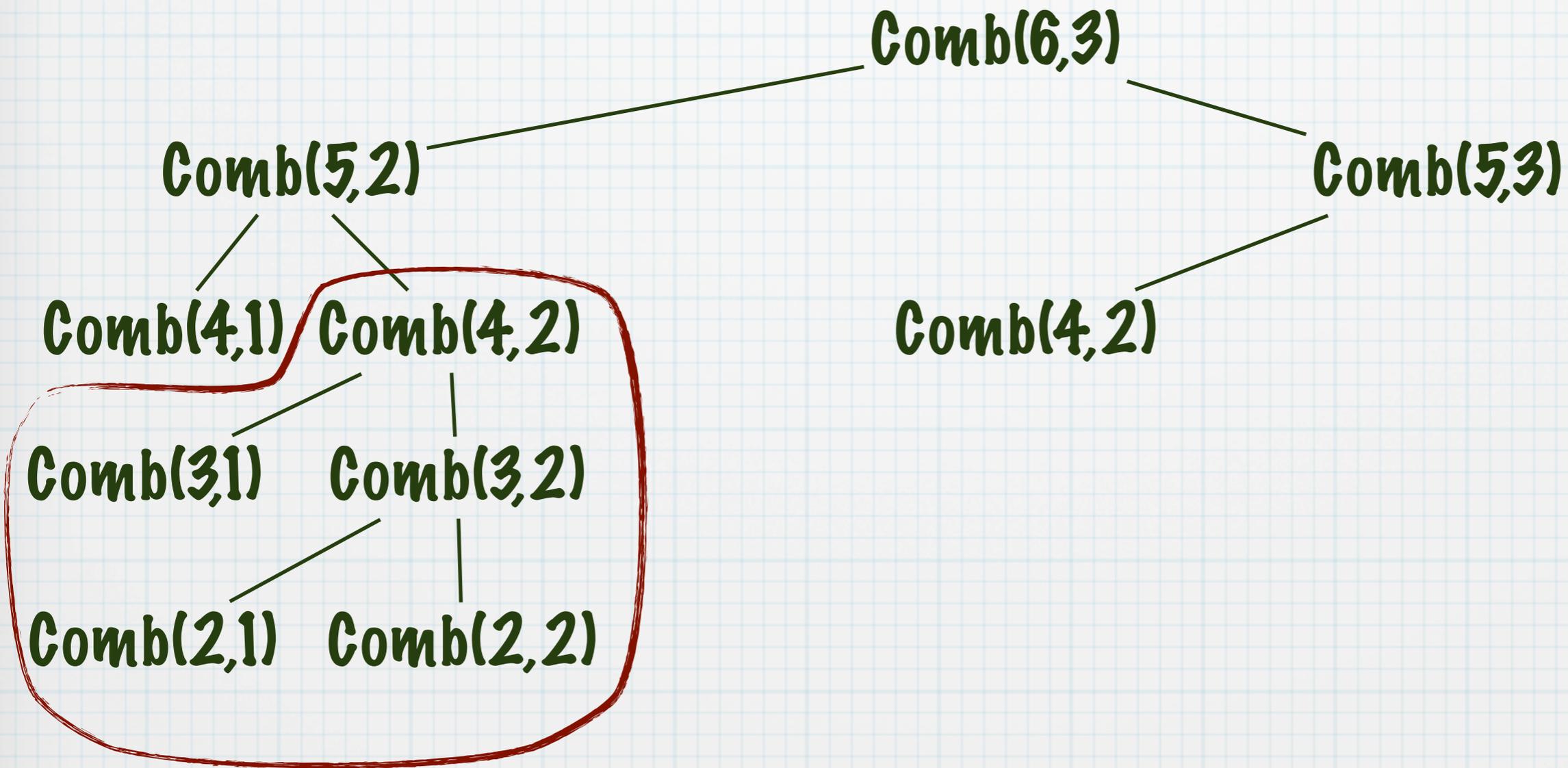
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

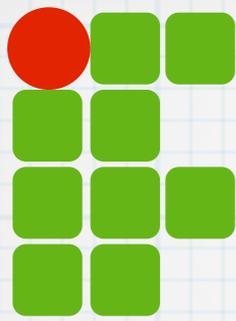




Exemplo

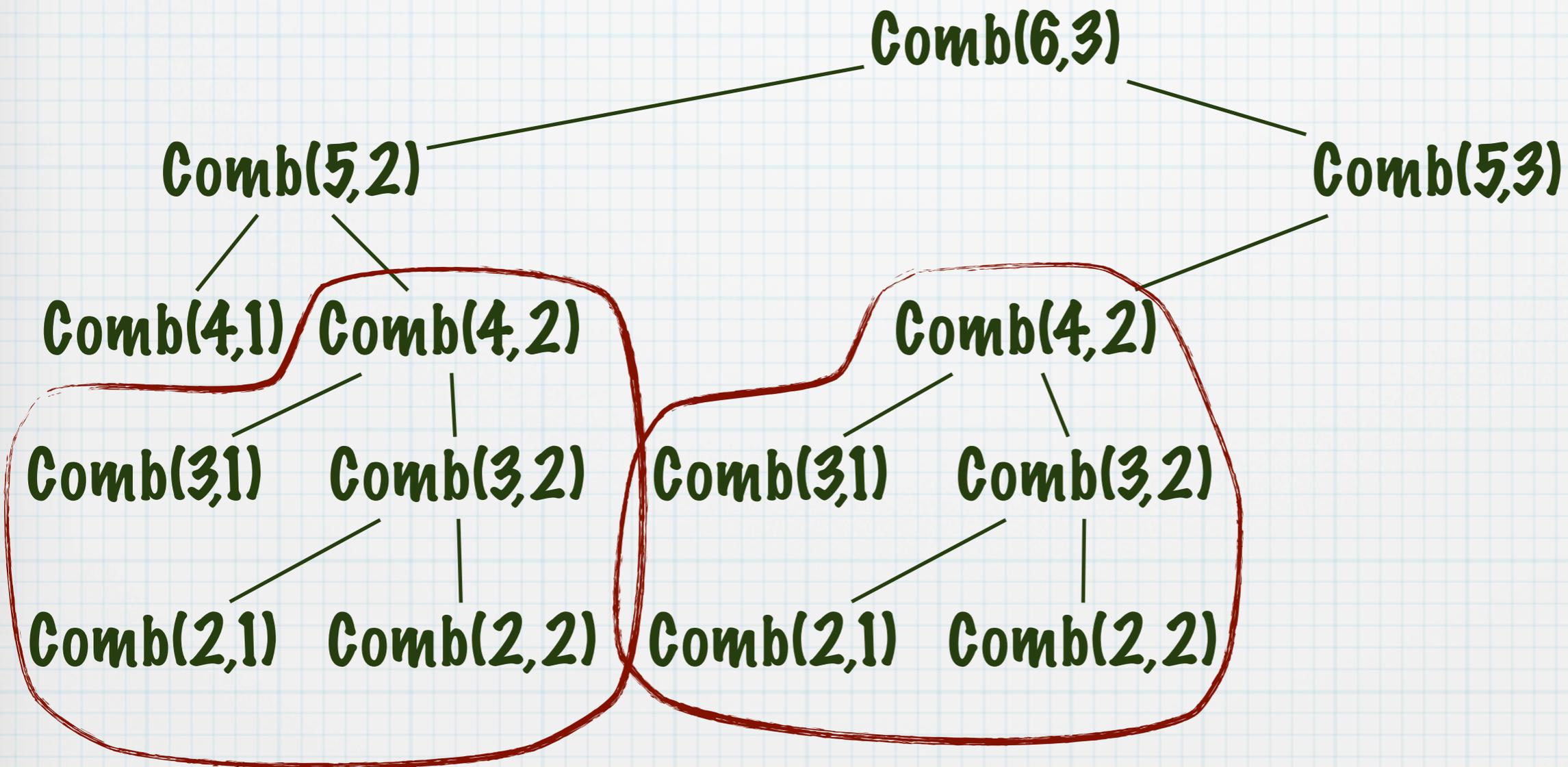
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

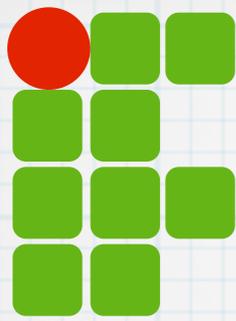




Exemplo

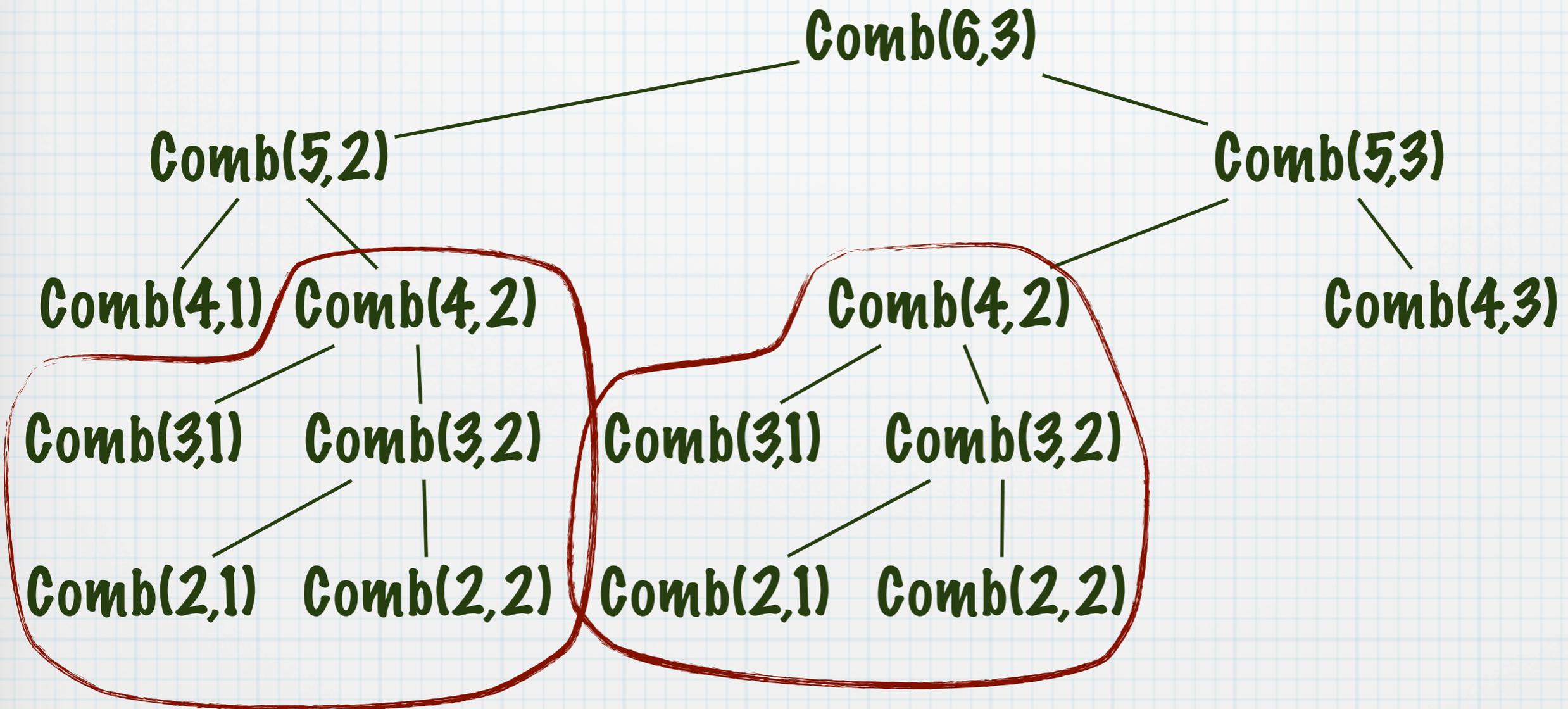
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

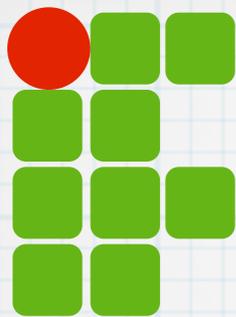




Exemplo

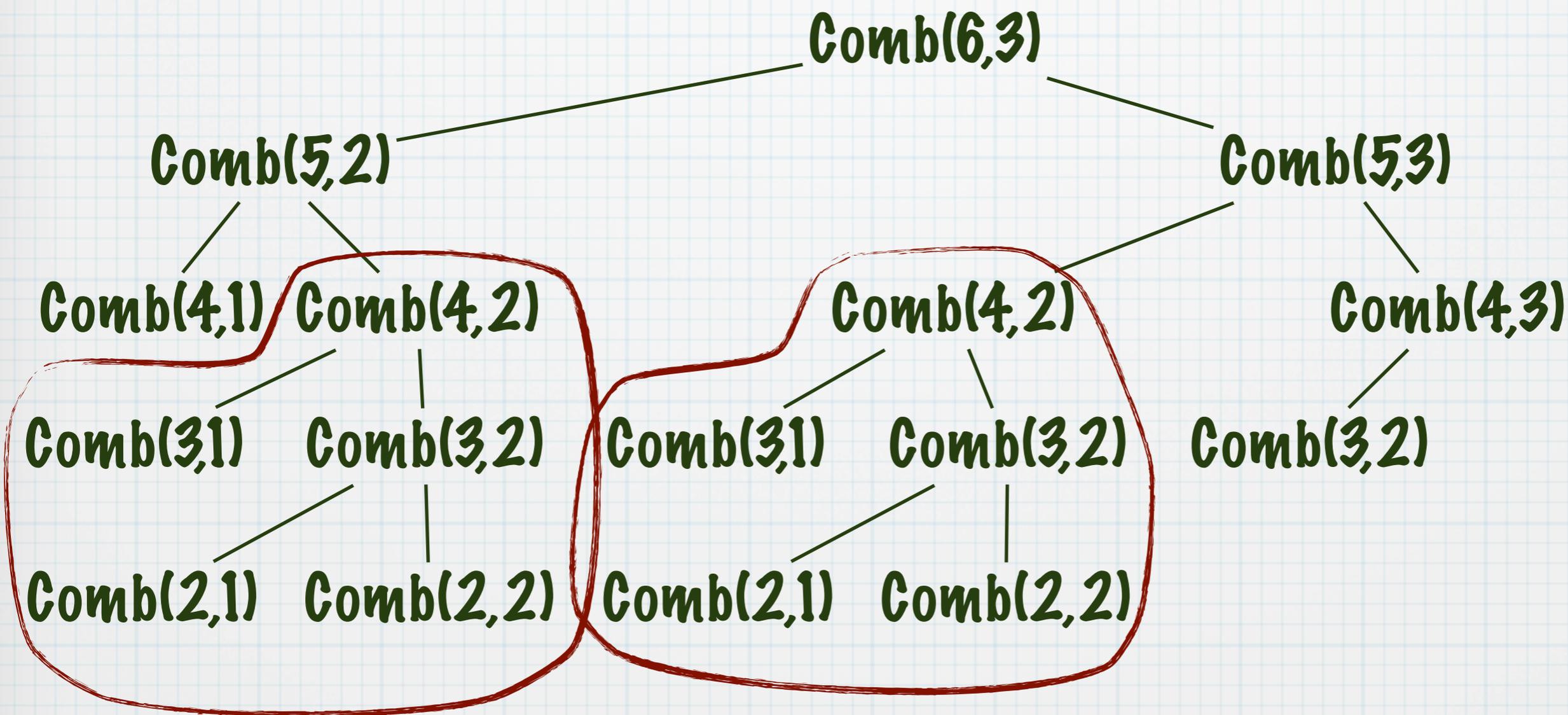
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

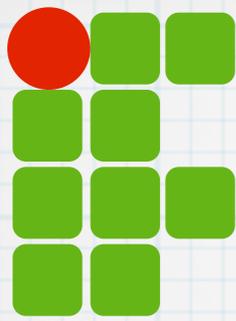




Exemplo

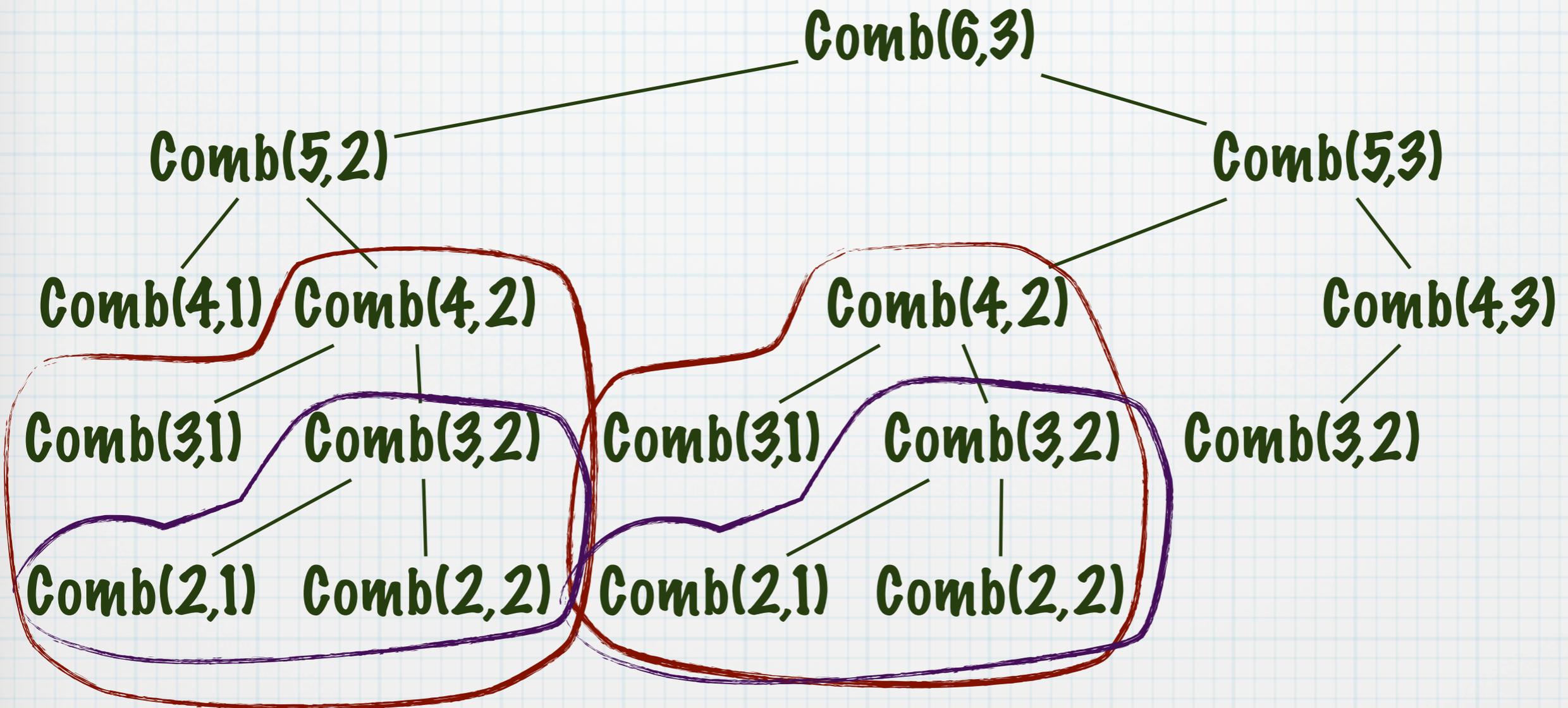
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

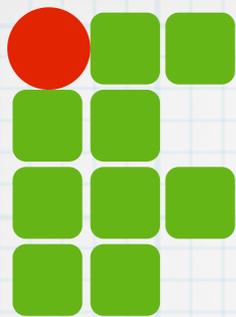




Exemplo

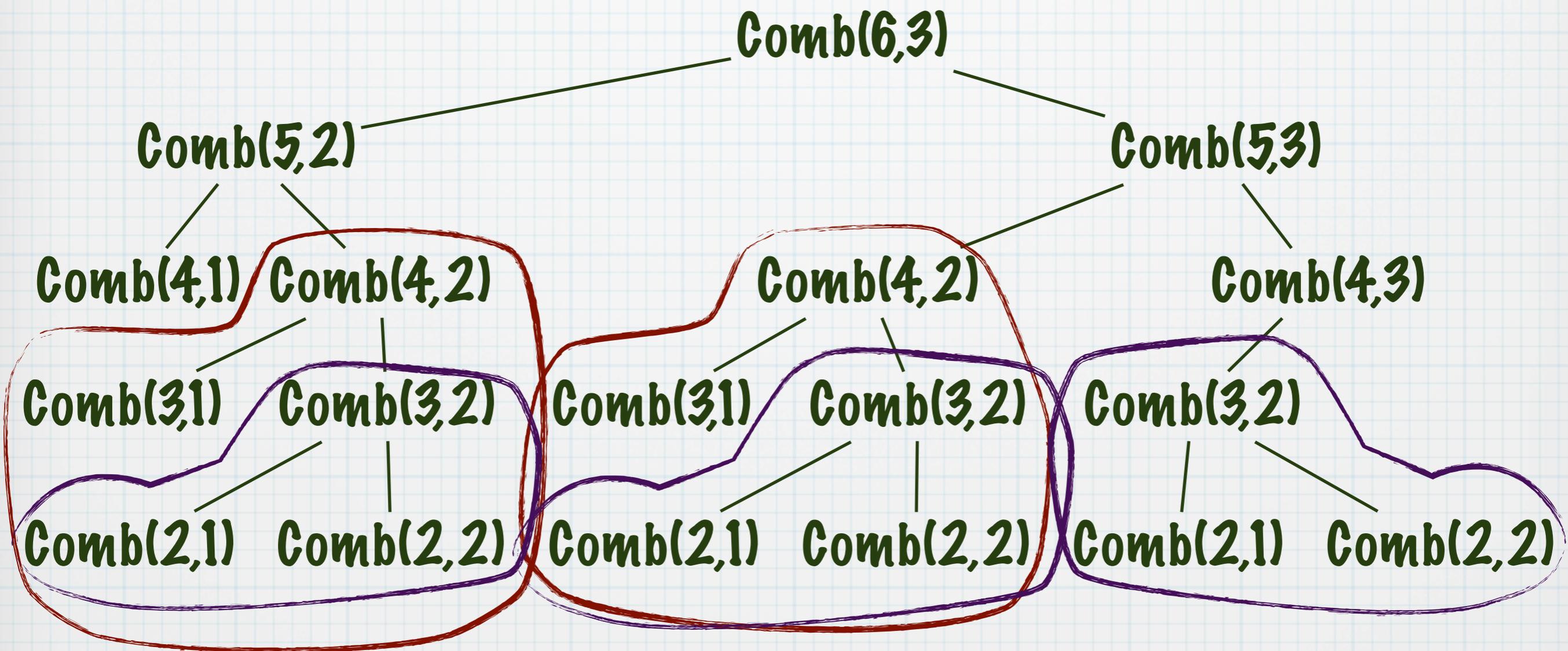
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

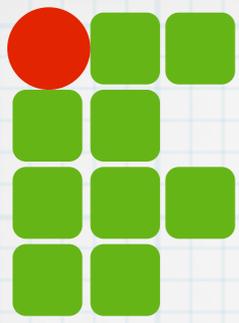




Exemplo

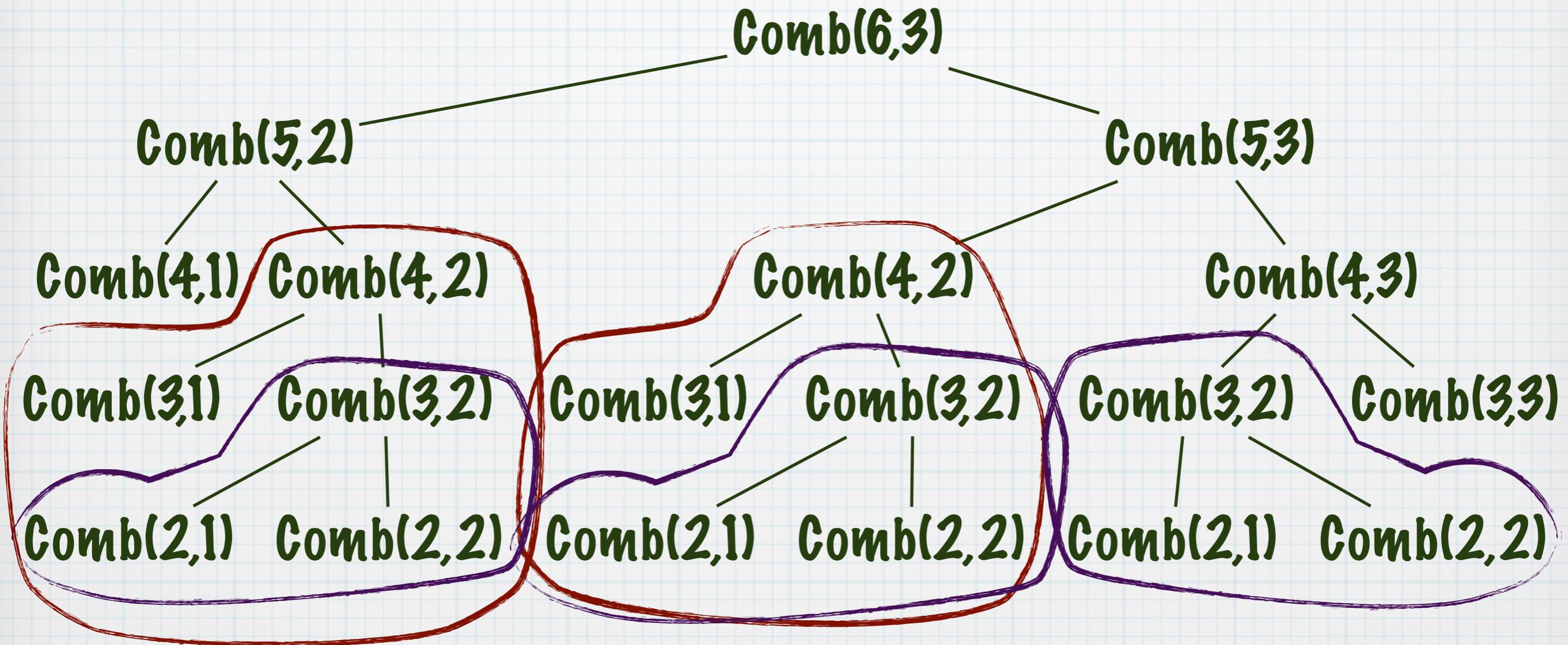
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$

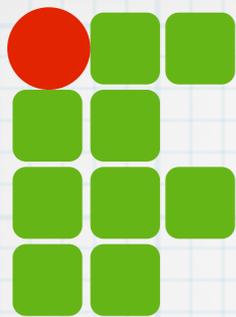




Exemplo

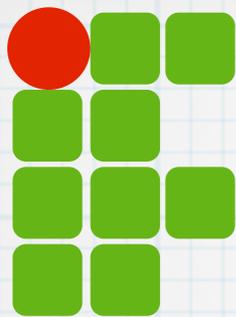
$$\text{Comb}(n, k) \begin{cases} n & \text{Se } k = 1 \\ 1 & \text{se } k = n \\ \text{Comb}(n - 1, k - 1) + \text{Comb}(n - 1, k) & \text{Se } 1 < k < n \end{cases}$$





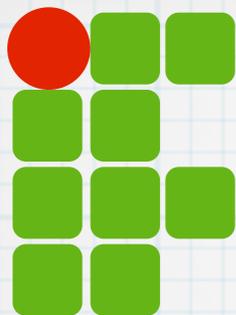
Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35



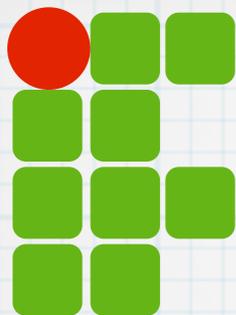
Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35



Exemplo

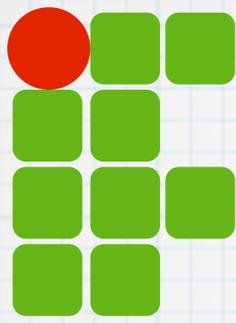
C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35



Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

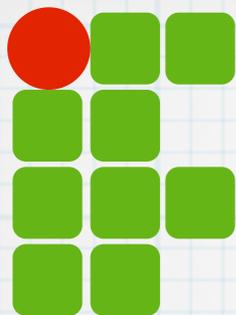
Diagram illustrating the calculation of the binomial coefficient $C_3^6 = 20$ using Pascal's triangle. Arrows show the addition of $C_2^5 = 10$ and $C_3^5 = 10$ to reach $C_3^6 = 20$. The value 20 is circled in red.



Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

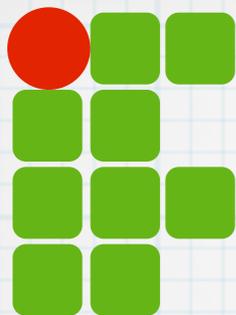
Diagram illustrating the calculation of binomial coefficients C_k^n for $n=5$ and $n=6$. Arrows show the calculation of $C_2^5 = 10$ from $C_1^4 = 4$ and $C_2^4 = 6$, and $C_3^6 = 20$ from $C_2^5 = 10$ and $C_3^5 = 10$. The value 20 is circled in red.



Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

Diagram illustrating the calculation of binomial coefficients C_k^n for $n=1$ to $n=7$ and $k=1$ to $k=4$. The values are shown in a grid. Arrows indicate the calculation path: $C_2^3 = 3$ and $C_3^4 = 4$ are used to calculate $C_3^5 = 10$; $C_2^4 = 6$ and $C_3^5 = 10$ are used to calculate $C_3^6 = 20$; $C_2^5 = 10$ and $C_3^6 = 20$ are used to calculate $C_3^7 = 35$. The value $C_3^6 = 20$ is circled in red.

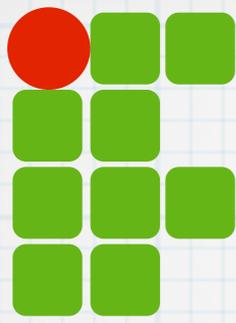


Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

Arrows indicating the calculation of C_k^n values:

- From $C_2^2=1$ to $C_3^2=3$
- From $C_3^2=3$ to $C_4^2=6$
- From $C_4^2=6$ to $C_5^2=10$
- From $C_5^2=10$ to $C_6^2=15$
- From $C_3^3=1$ to $C_4^3=4$
- From $C_4^3=4$ to $C_5^3=10$
- From $C_5^3=10$ to $C_6^3=20$
- From $C_4^4=1$ to $C_5^4=5$
- From $C_5^4=5$ to $C_6^4=15$
- From $C_6^4=15$ to $C_7^4=35$



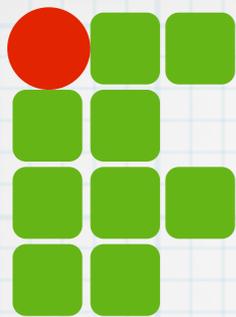
Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

Arrows indicating the calculation of values in the table:

- From (n=2, k=1) to (n=3, k=2)
- From (n=3, k=1) to (n=4, k=2)
- From (n=4, k=1) to (n=5, k=2)
- From (n=3, k=2) to (n=4, k=3)
- From (n=4, k=2) to (n=5, k=3)
- From (n=5, k=2) to (n=6, k=3)
- From (n=4, k=3) to (n=5, k=4)
- From (n=5, k=3) to (n=6, k=4)

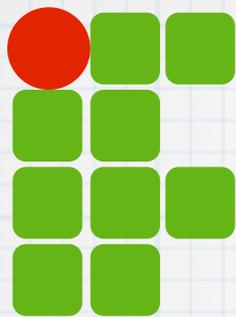
The value 20 in the cell (n=6, k=3) is circled in red.



Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

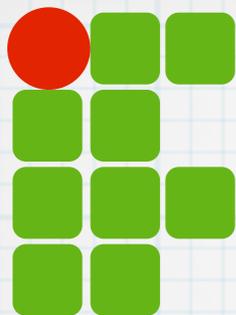
Diagram illustrating the calculation of binomial coefficients C_k^n for $n=1$ to $n=7$ and $k=1$ to $k=4$. The values are shown in a grid. Arrows indicate the calculation path: $C_2^2 = 1$ leads to $C_3^2 = 3$ and $C_3^3 = 1$; $C_3^2 = 3$ leads to $C_4^2 = 6$ and $C_4^3 = 4$; $C_4^2 = 6$ leads to $C_5^2 = 10$ and $C_5^3 = 10$; $C_5^2 = 10$ leads to $C_6^2 = 15$ and $C_6^3 = 20$. The value $C_6^3 = 20$ is circled in red.



Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

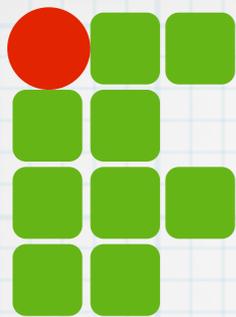
Diagram illustrating the calculation of binomial coefficients C_k^n for $n=1$ to $n=7$ and $k=1$ to $k=4$. The values are shown in a grid, with arrows indicating the calculation path from the previous row and column. The value 20 is circled in red.



Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

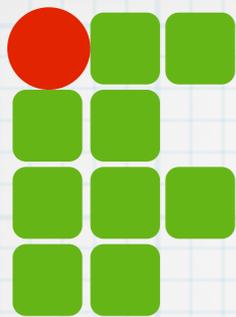
Diagram illustrating the calculation of binomial coefficients C_k^n for $n=1$ to $n=7$ and $k=1$ to $k=4$. The values are shown in a grid. Arrows indicate the calculation path: $C_1^1=1$ leads to $C_2^2=1$, $C_3^3=1$, $C_4^4=1$, and $C_5^5=1$. From $C_2^2=1$, arrows lead to $C_3^2=3$ and $C_4^2=6$. From $C_3^2=3$, arrows lead to $C_4^3=4$ and $C_5^3=10$. From $C_4^2=6$, arrows lead to $C_5^3=10$ and $C_6^3=20$. From $C_5^3=10$, an arrow leads to $C_6^4=15$. The value $C_6^3=20$ is circled in red.



Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

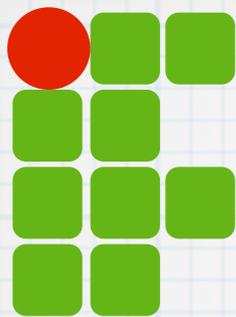
The table illustrates the values of the binomial coefficient C_k^n for n from 1 to 7 and k from 1 to 4. The values are: $C_1^n = n$, $C_2^n = \frac{n(n-1)}{2}$, $C_3^n = \frac{n(n-1)(n-2)}{6}$, and $C_4^n = \frac{n(n-1)(n-2)(n-3)}{24}$. The value 20 is circled in red, and arrows indicate the calculation path from $C_1^4 = 4$ to $C_2^5 = 10$ and $C_3^6 = 20$.



Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

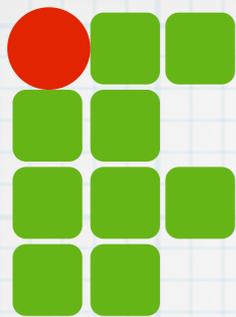
The table illustrates the calculation of binomial coefficients C_k^n for n from 1 to 7 and k from 1 to 4. The values are calculated using the recurrence relation $C_k^n = C_{k-1}^{n-1} + C_k^{n-1}$. The value 20 is circled in red, and a purple box highlights the sub-table for $k=3$ and $n \geq 3$. A red box highlights the first column ($k=1$) for $n \leq 4$. Arrows indicate the flow of information from the first column to the other columns.



Exemplo

C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

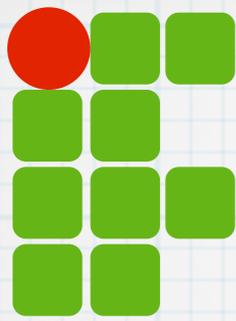
The table illustrates the calculation of binomial coefficients C_k^n for $n=1$ to $n=7$ and $k=1$ to $k=4$. The values are calculated using the recurrence relation $C_k^n = C_{k-1}^{n-1} + C_k^{n-1}$. The value 20 is circled in red, and a red box highlights the path of calculations leading to it: $C_1^2=1$, $C_1^3=3$, $C_1^4=6$, $C_1^5=10$, and $C_2^6=15$. A purple box highlights the entire row for $n=6$ and the column for $k=3$. A green arrow points from the value 1 in the $(n=1, k=1)$ cell to the value 1 in the $(n=6, k=4)$ cell.



Exemplo

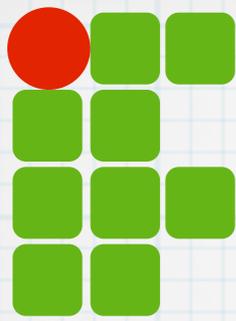
C_k^n	k=1	k=2	k=3	k=4
n=1	1			
n=2	2	1		
n=3	3	3	1	
n=4	4	6	4	1
n=5	5	10	10	5
n=6	6	15	20	15
n=7	7	21	35	35

The table illustrates the calculation of binomial coefficients C_k^n for n from 1 to 7 and k from 1 to 4. The values are calculated using the recursive formula $C_k^n = C_{k-1}^{n-1} + C_k^{n-1}$. The value 20 is highlighted with a red circle, and a red box encloses the sub-table for $n=3, 4, 5, 6$ and $k=2, 3$. Arrows indicate the flow of calculation from top-left to bottom-right.



Exemplo

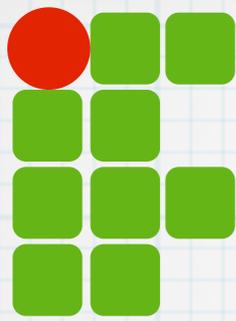
```
int comb(int n, int k) {
    int i, j;
    int maxj = n - k;
    int *c = malloc(sizeof(int) * maxj+1);
    int result = 0;
    for (j = 0; j <= maxj; j++)
        c[j] = j + 1;
    for (i = 1; i < k; i++) {
        for (j = 1; j <= maxj; j++)
            c[j] += c[j - 1];
    }
    result = c[maxj];
    free(c);
    return result;
}
```



Exemplo

1
2
3
4

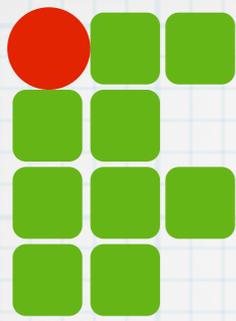
```
int comb(int n, int k) {
    int i, j;
    int maxj = n - k;
    int *c = malloc(sizeof(int) * maxj+1);
    int result = 0;
    for (j = 0; j <= maxj; j++)
        c[j] = j + 1;
    for (i = 1; i < k; i++) {
        for (j = 1; j <= maxj; j++)
            c[j] += c[j - 1];
    }
    result = c[maxj];
    free(c);
    return result;
}
```



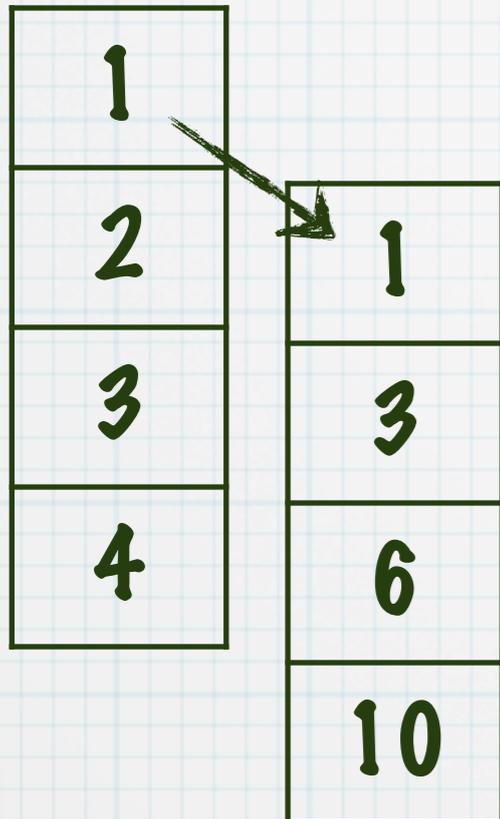
Exemplo

1	
2	1
3	3
4	6
	10

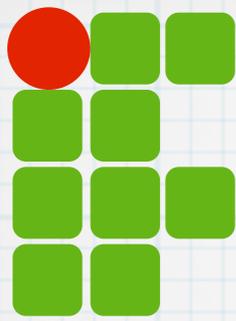
```
int comb(int n, int k) {  
    int i, j;  
    int maxj = n - k;  
    int *c = malloc(sizeof(int) * maxj+1);  
    int result = 0;  
    for (j = 0; j <= maxj; j++)  
        c[j] = j + 1;  
    for (i = 1; i < k; i++) {  
        for (j = 1; j <= maxj; j++)  
            c[j] += c[j - 1];  
    }  
    result = c[maxj];  
    free(c);  
    return result;  
}
```



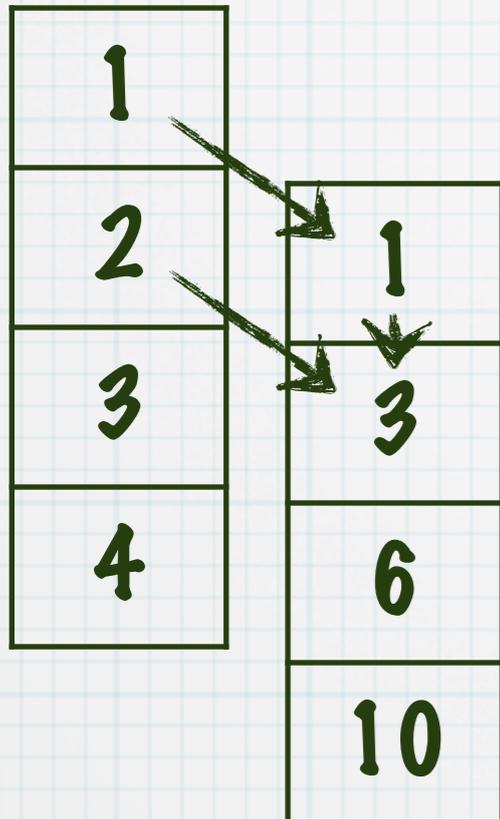
Exemplo



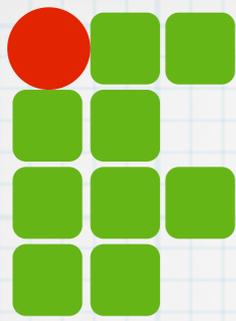
```
int comb(int n, int k) {
    int i, j;
    int maxj = n - k;
    int *c = malloc(sizeof(int) * maxj+1);
    int result = 0;
    for (j = 0; j <= maxj; j++)
        c[j] = j + 1;
    for (i = 1; i < k; i++) {
        for (j = 1; j <= maxj; j++)
            c[j] += c[j - 1];
    }
    result = c[maxj];
    free(c);
    return result;
}
```



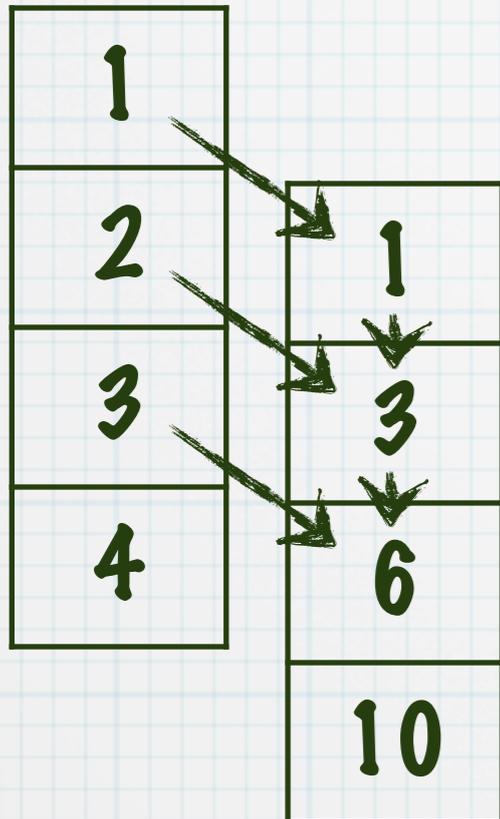
Exemplo



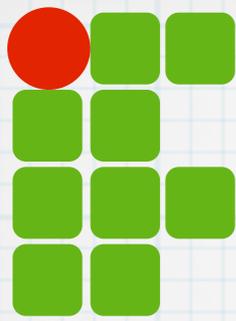
```
int comb(int n, int k) {
    int i, j;
    int maxj = n - k;
    int *c = malloc(sizeof(int) * maxj+1);
    int result = 0;
    for (j = 0; j <= maxj; j++)
        c[j] = j + 1;
    for (i = 1; i < k; i++) {
        for (j = 1; j <= maxj; j++)
            c[j] += c[j - 1];
    }
    result = c[maxj];
    free(c);
    return result;
}
```



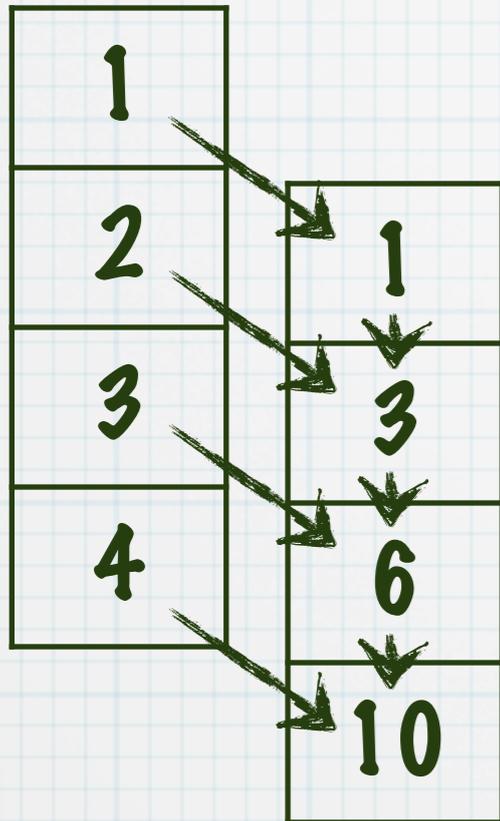
Exemplo



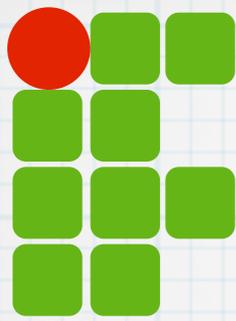
```
int comb(int n, int k) {  
    int i, j;  
    int maxj = n - k;  
    int *c = malloc(sizeof(int) * maxj+1);  
    int result = 0;  
    for (j = 0; j <= maxj; j++)  
        c[j] = j + 1;  
    for (i = 1; i < k; i++) {  
        for (j = 1; j <= maxj; j++)  
            c[j] += c[j - 1];  
    }  
    result = c[maxj];  
    free(c);  
    return result;  
}
```



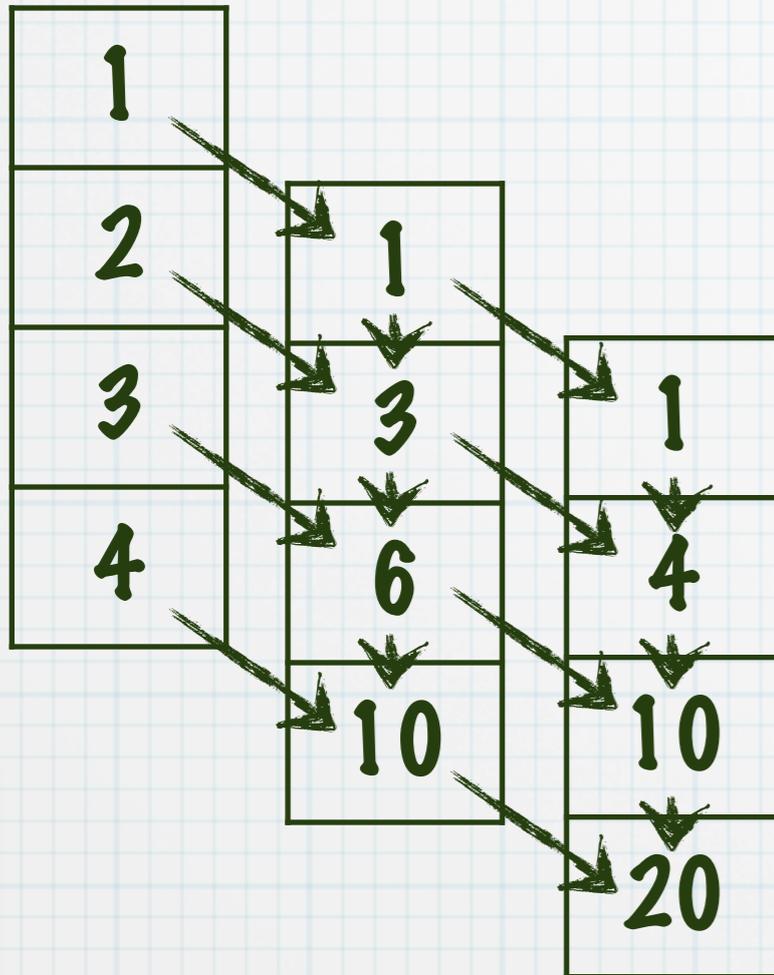
Exemplo



```
int comb(int n, int k) {  
    int i, j;  
    int maxj = n - k;  
    int *c = malloc(sizeof(int) * maxj+1);  
    int result = 0;  
    for (j = 0; j <= maxj; j++)  
        c[j] = j + 1;  
    for (i = 1; i < k; i++) {  
        for (j = 1; j <= maxj; j++)  
            c[j] += c[j - 1];  
    }  
    result = c[maxj];  
    free(c);  
    return result;  
}
```

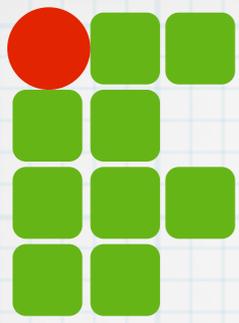


Exemplo



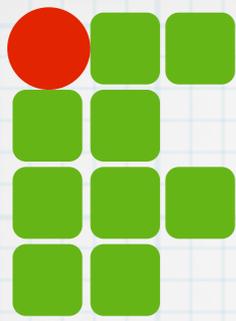
```
int comb(int n, int k) {  
    int i, j;  
    int maxj = n - k;  
    int *c = malloc(sizeof(int) * maxj+1);  
    int result = 0;  
    for (j = 0; j <= maxj; j++)  
        c[j] = j + 1;  
    for (i = 1; i < k; i++) {  
        for (j = 1; j <= maxj; j++)  
            c[j] += c[j - 1];  
    }  
    result = c[maxj];  
    free(c);  
    return result;  
}
```

i=2



Fibonacci

$$\text{fib}(n) = \begin{cases} 1 & \text{Se } n = 1 \\ 1 & \text{Se } n = 0 \\ \text{fib}(n - 1) + \text{fib}(n - 2) & \text{Se } n > 1 \end{cases}$$



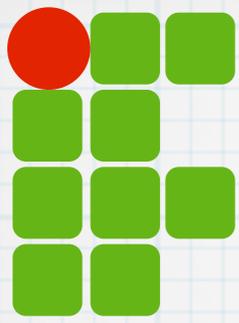
Fibonacci

$$\text{fib}(n) = \begin{cases} 1 & \text{Se } n = 1 \\ 1 & \text{Se } n = 0 \\ \text{fib}(n - 1) + \text{fib}(n - 2) & \text{Se } n > 1 \end{cases}$$

*** Para calcular fib(n)**

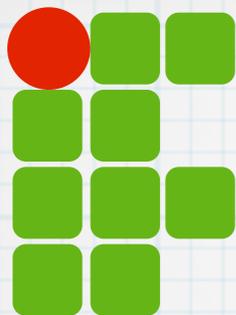
*** Memorizar os dois últimos elementos da sequencia**

```
int fib(int n) {
    int a = 1, b = 1, t;
    for (int i = 1; i <= n; i++) {
        t = a;
        a = b;
        b += t;
    }
    return b;
}
```



Conclusão

- * problemas de solução recursivas
- * Minimizar tempo e espaço
- * Induzir uma progressão iterativa de transformações sucessivas



Dúvidas?