

PROGRAMACIÓN DINÁMICA

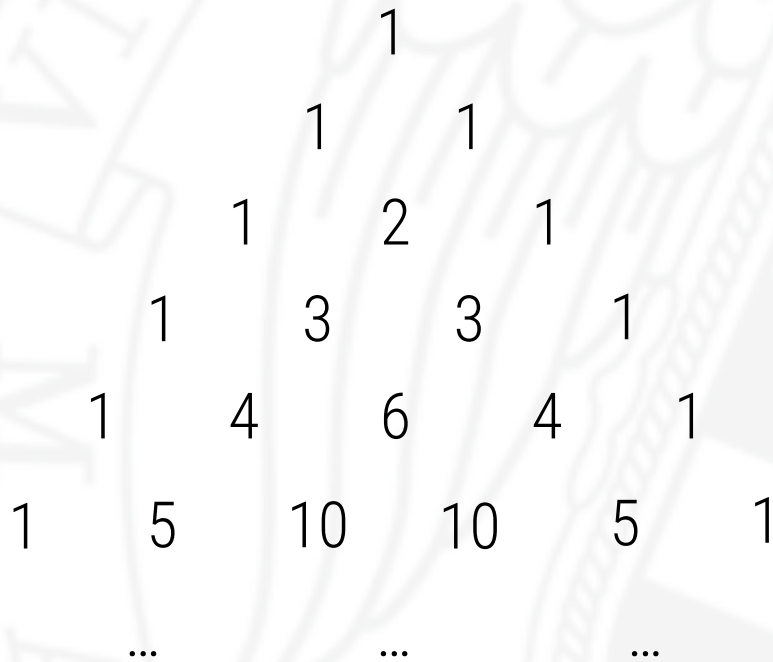


U N I V E R S I D A D
COMPLUTENSE
M A D R I D

ALBERTO VERDEJO

Números combinatorios

- ▶ $\binom{n}{r}$ es el número de subconjuntos de cardinal r que tiene un conjunto de n elementos.

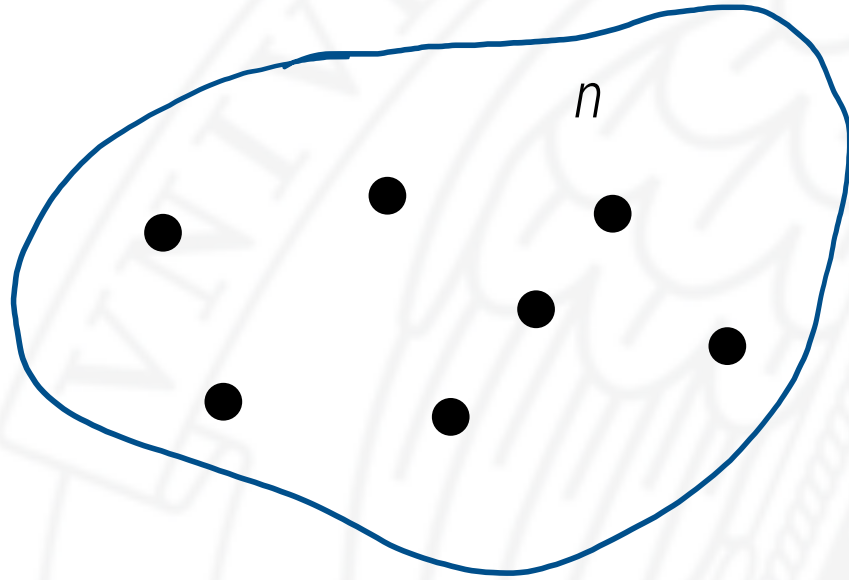


A Pascal's triangle for n=5, showing the values of the binomial coefficients $\binom{n}{r}$ for r from 0 to 5. The triangle is symmetric and each entry is the sum of the two entries directly above it.

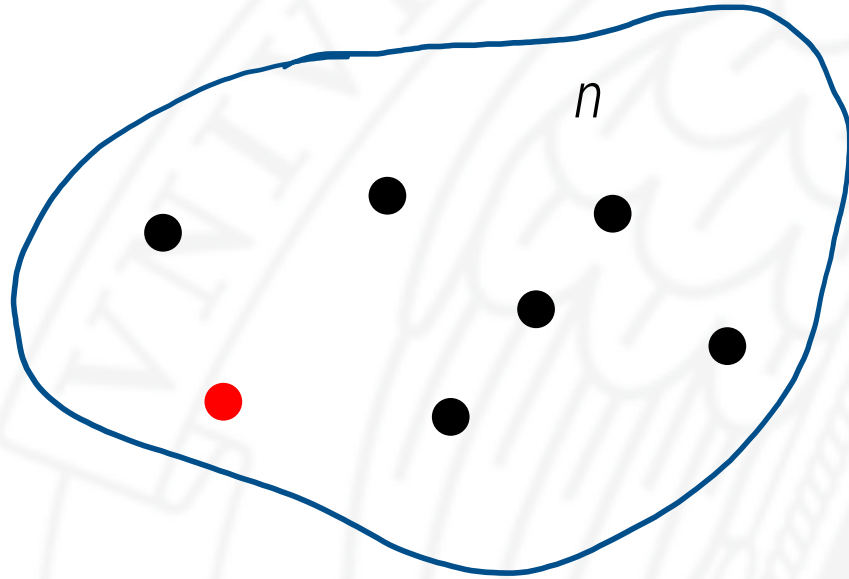
			1		
		1		1	
	1		2		1
	1	3		3	1
	1	4	6	4	1
1	5	10	10	5	1
...

$$\binom{n}{r} = \begin{cases} 1 & \text{si } r = 0 \vee r = n \\ \binom{n-1}{r-1} + \binom{n-1}{r} & \text{si } 0 < r < n \end{cases}$$

Números combinatorios



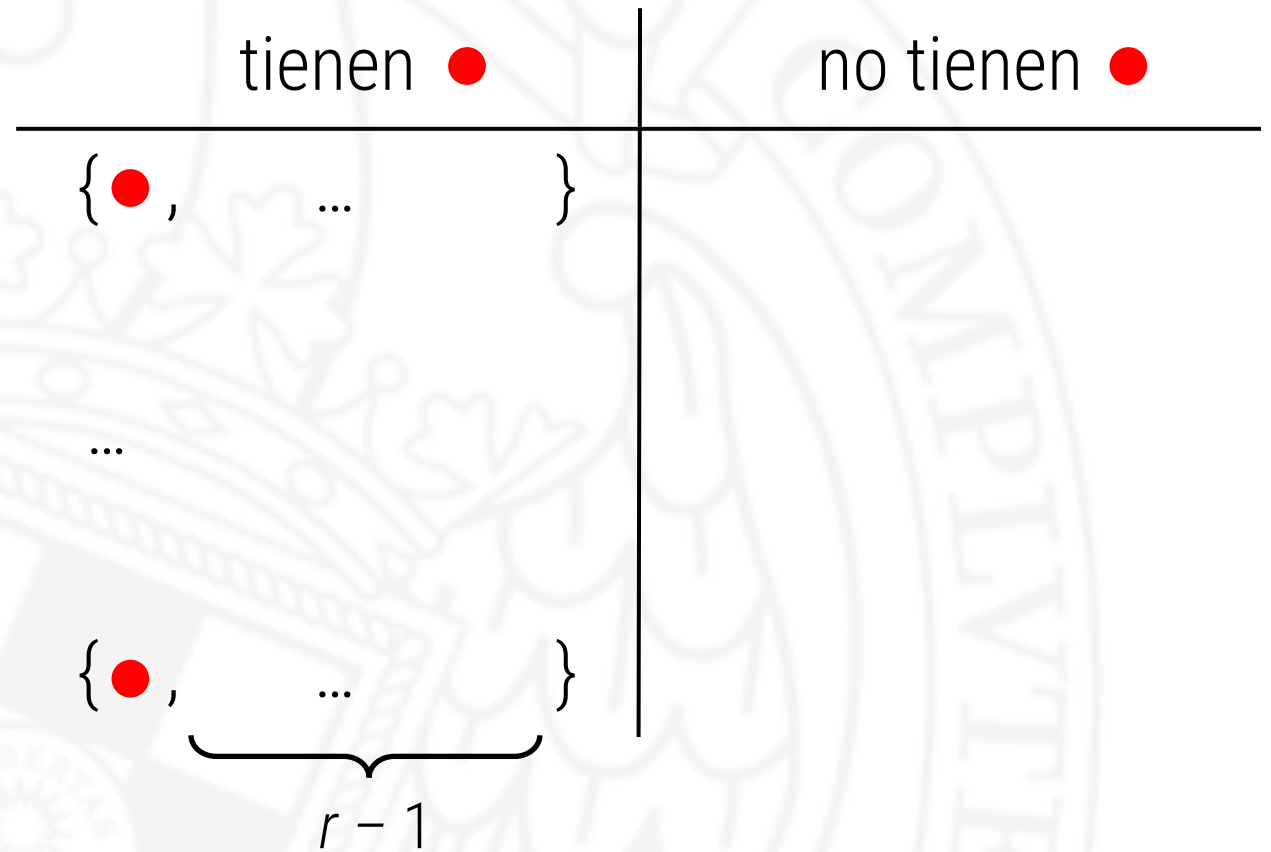
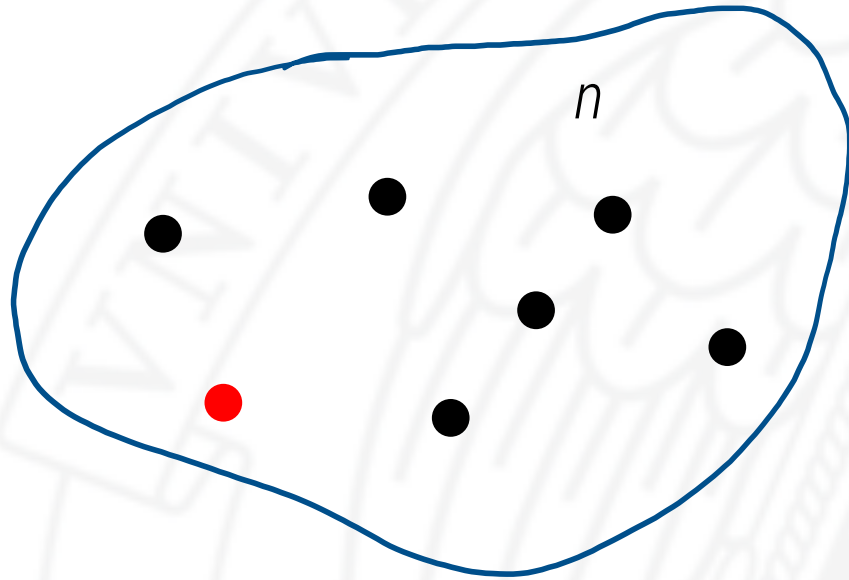
Números combinatorios



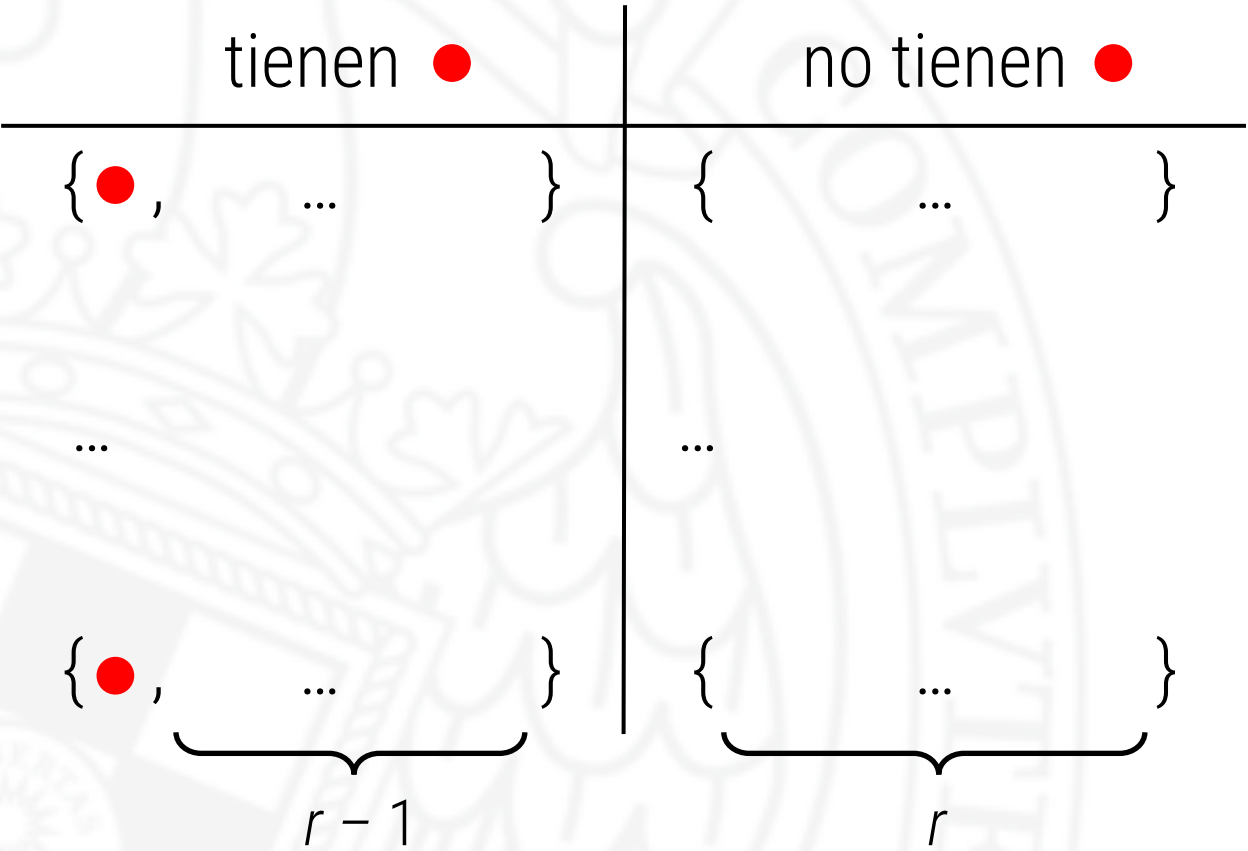
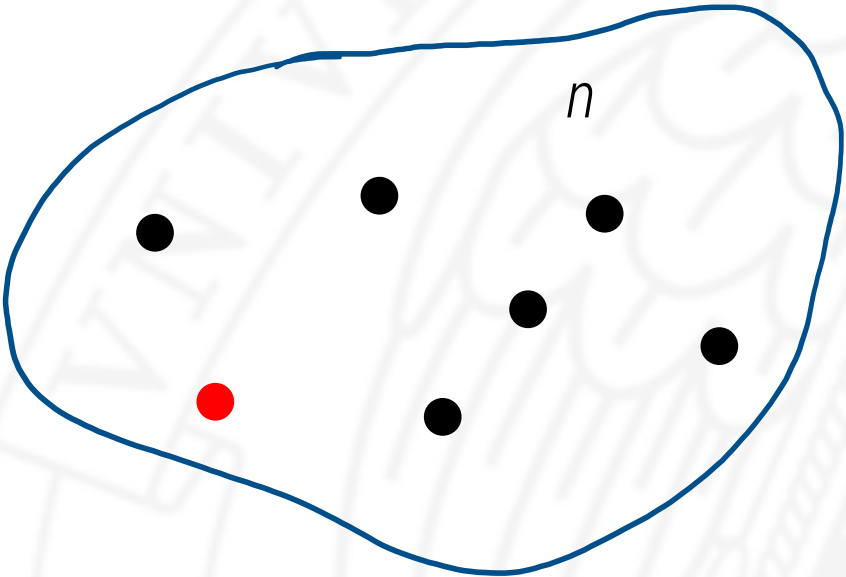
tienen ●

no tienen ●

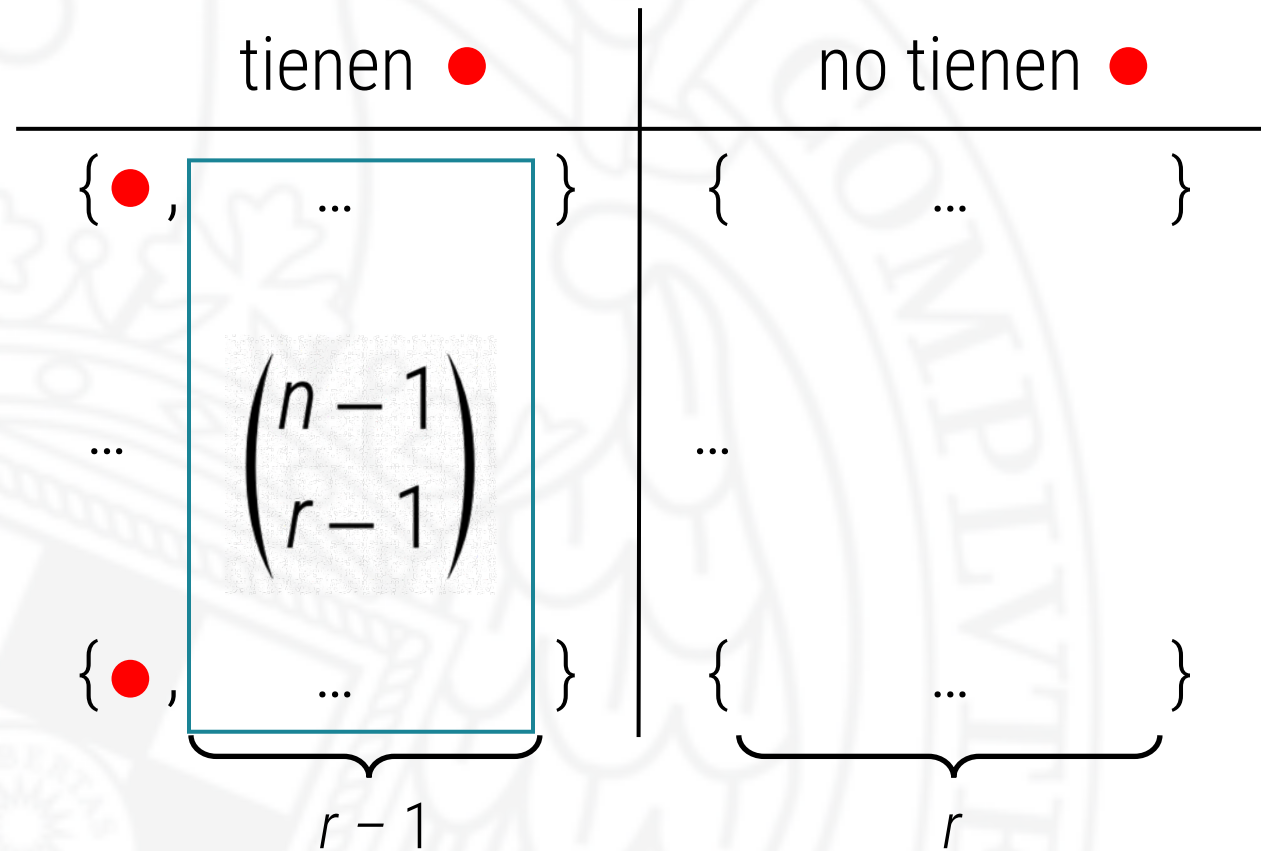
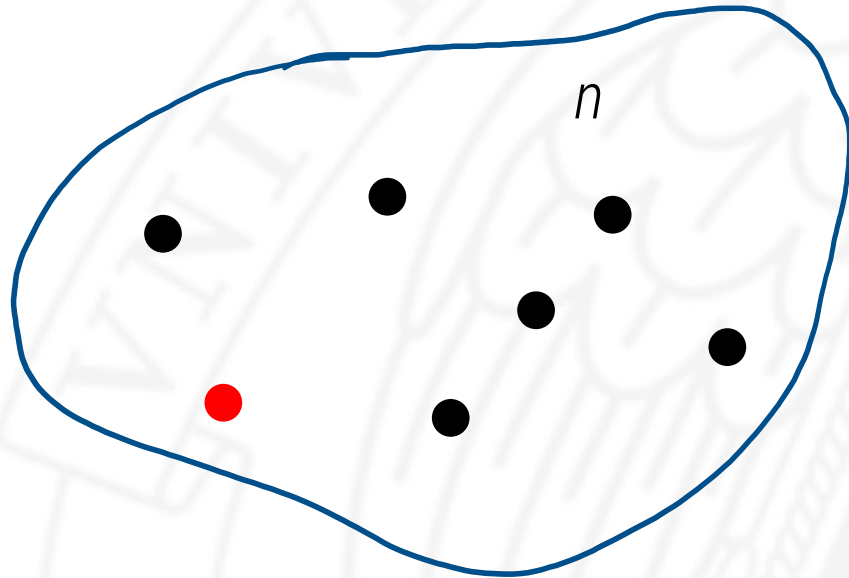
Números combinatorios



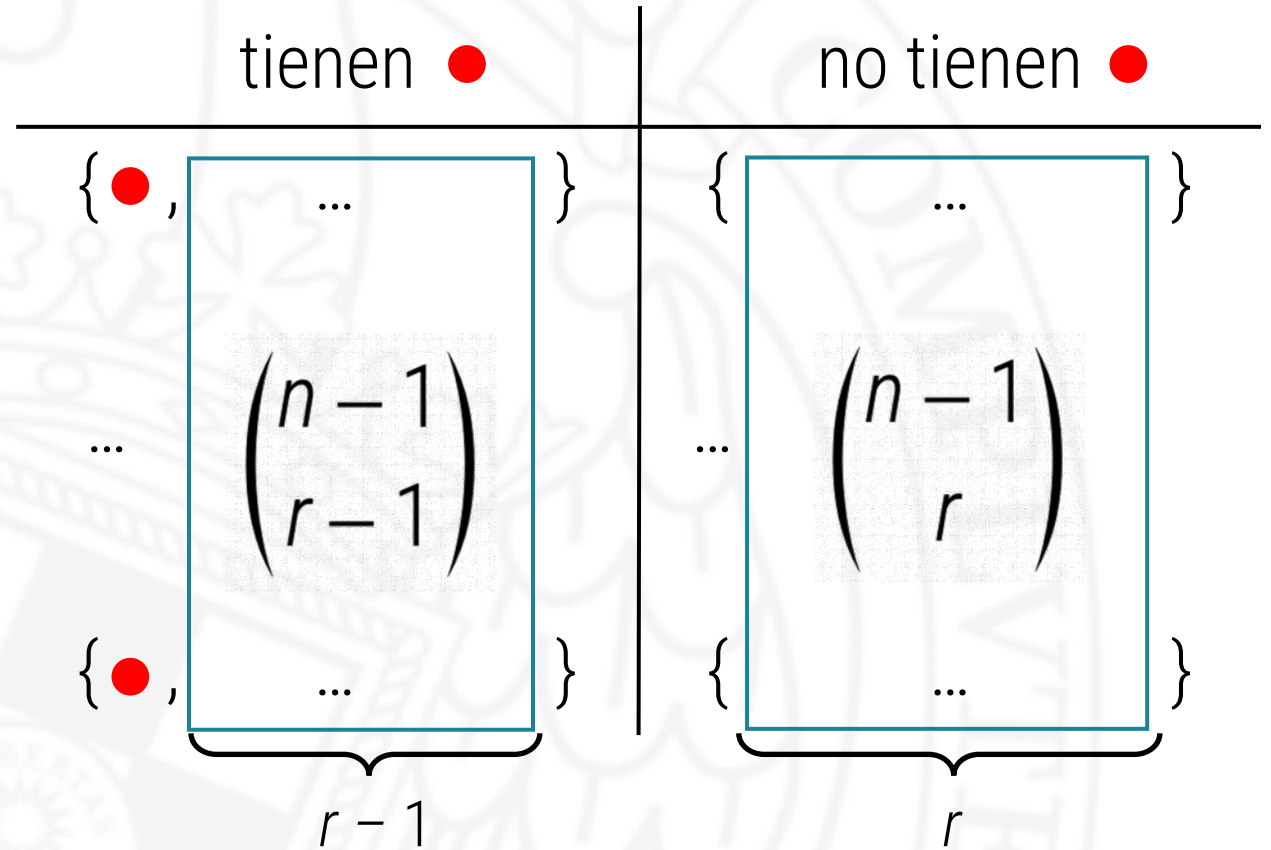
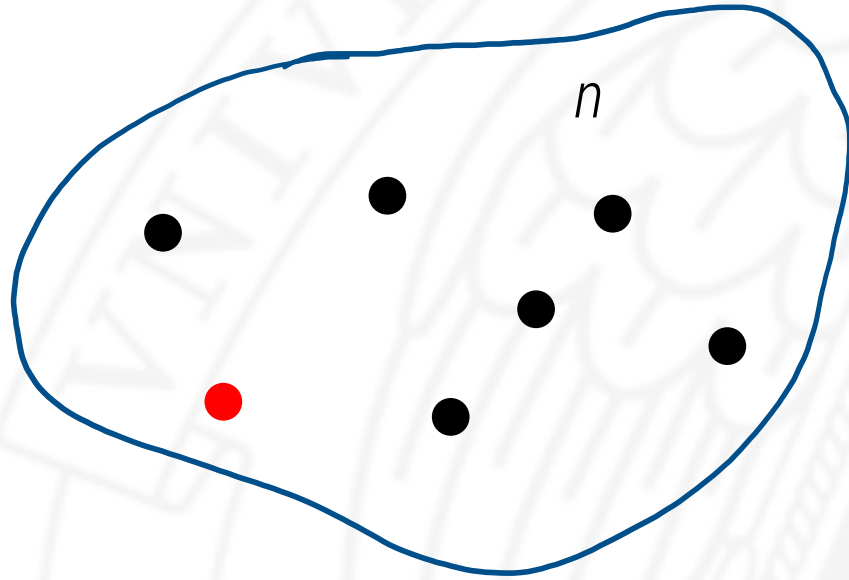
Números combinatorios



Números combinatorios



Números combinatorios



$$\binom{n-1}{r-1} + \binom{n-1}{r}$$

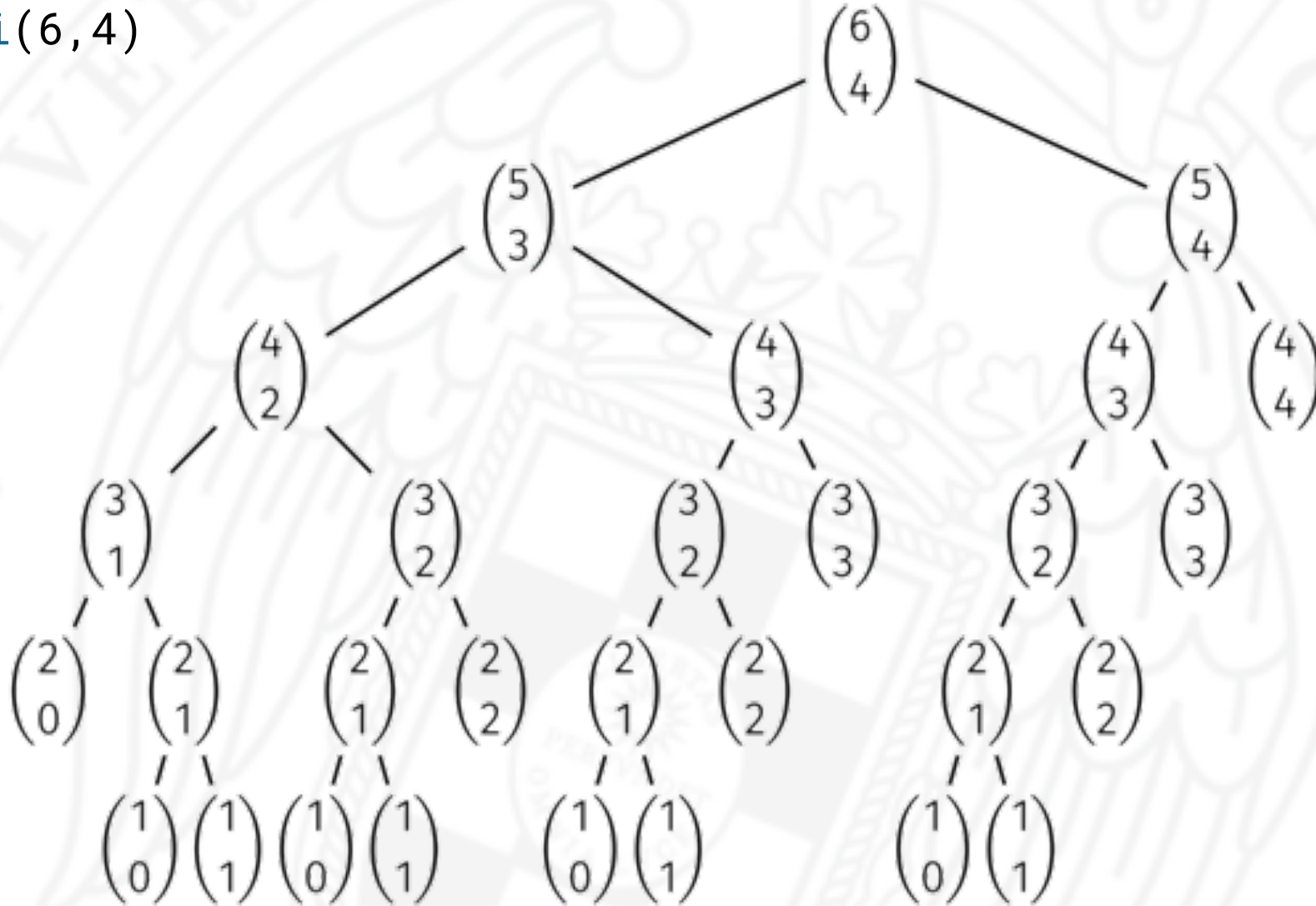
Implementación recursiva

$$\binom{n}{r} = \begin{cases} 1 & \text{si } r = 0 \vee r = n \\ \binom{n-1}{r-1} + \binom{n-1}{r} & \text{si } 0 < r < n \end{cases}$$

```
int num_combi(int i, int j) {  
    if (j == 0 || j == i)  
        return 1;  
    else  
        return num_combi(i-1, j-1) + num_combi(i-1, j);  
}
```

Implementación recursiva, solapamiento de subproblemas

`num_combi(6, 4)`



Implementación recursiva, solapamiento de subproblemas

- Número de veces que se calcula $\binom{i}{j}$ para calcular $\binom{10}{5}$.

	j						
	5	4	3	2	1	0	
10	1						
9	1	1					
8	1	2	1				
7	1	3	3	1			
6	1	4	6	4	1		
5	1	5	10	10	5	1	
4		5	15	20	15	5	
3			15	35	35	15	
2				35	70	35	
1					70	70	

En general,

$$2\binom{n}{r} - 1 \text{ términos}$$

$$(r+1)(n-r+1) - 1$$

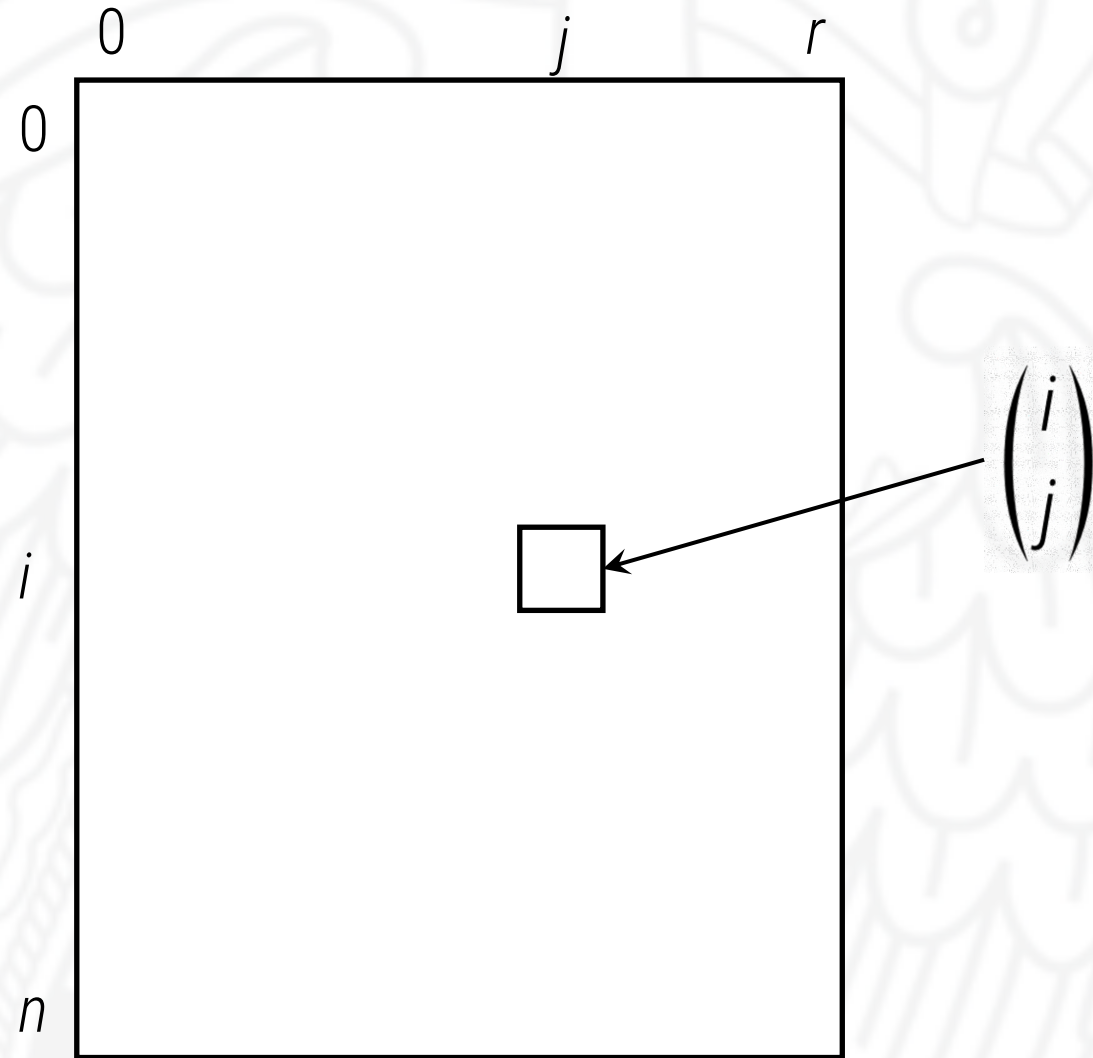
términos distintos

Programación dinámica

- ▶ Utilización de una tabla (array multidimensional) donde se almacenan los resultados a subproblemas ya resueltos.
- ▶ La tabla tiene tantas dimensiones como argumentos tiene la recurrencia.
- ▶ El tamaño de cada dimensión coincide con los valores que puede tomar el argumento correspondiente.
- ▶ Cada subproblema se asocia a una posición de la tabla.

Tabla para los números combinatorios

$$\binom{n}{r}$$



Programación dinámica descendente

- ▶ Mantiene el diseño recursivo.
- ▶ La función recibe como parámetro de entrada/salida, la tabla donde se almacenan las soluciones a subproblemas ya resueltos.
- ▶ Antes de resolver de manera recursiva un subproblema, se mira en la tabla por si ya se hubiera resuelto.
- ▶ Tras resolver un subproblema recursivo, su solución se almacena en la tabla.
- ▶ Necesidad de saber si un subproblema está resuelto o no.

Programación dinámica descendente

```
int num_combi(int i, int j, Matriz<int> & C) {  
    if (j == 0 || j == i) return 1;  
    else if (C[i][j] != -1) return C[i][j];  
    else {  
        C[i][j] = num_combi(i-1, j-1, C) + num_combi(i-1, j, C);  
        return C[i][j];  
    }  
}
```

```
Matriz<int> C(n+1, r+1, -1);
```

```
cout << num_combi(n, r, C) << '\n';
```

Programación dinámica ascendente

- ▶ Cambiar el orden en el que se resuelve los subproblemas.
- ▶ Comenzar por resolver todos los subproblemas más pequeños que se puedan necesitar, para ir combinándolos hasta llegar a resolver el problema original.
- ▶ Los subproblemas se van resolviendo recorriéndolos de menor a mayor tamaño.
- ▶ Todos los posibles subproblemas de tamaño menor tienen que ser resueltos antes de resolver uno de tamaño mayor.

Programación dinámica ascendente

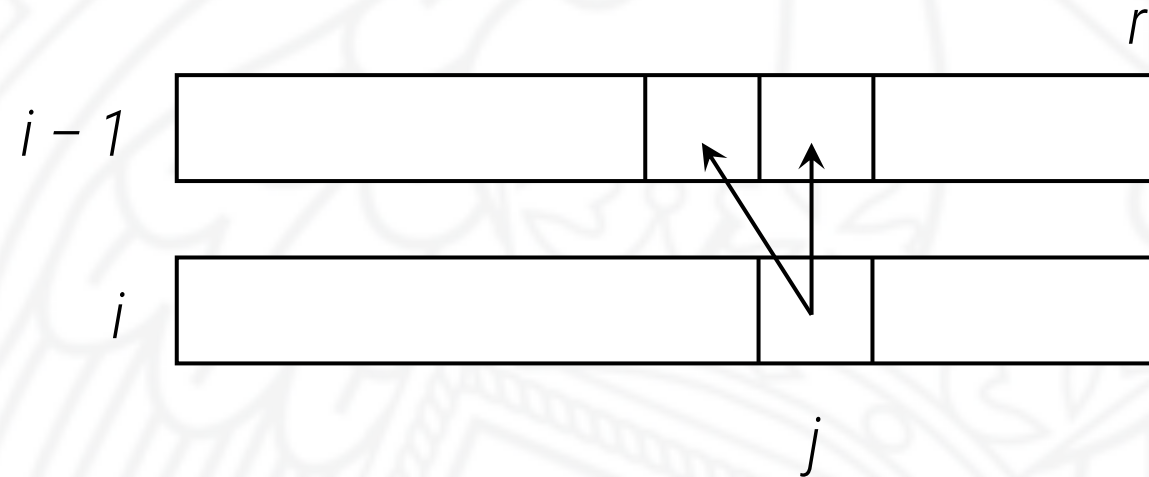
	0	1	2	...	r
0	1	0	0	...	0
1	1	1	0	...	0
2	1	2	1	...	0
\vdots	\vdots	\vdots	\vdots	\ddots	\vdots
n	1	n	$\binom{n}{2}$...	$\binom{n}{r}$

```
int pascal(int n, int r) {  
    Matriz<int> C(n+1, r+1, 0);  
    C[0][0] = 1;  
    for (int i = 1; i <= n; ++i) {  
        C[i][0] = 1;  
        for (int j = 1; j <= r; ++j)  
            C[i][j] = C[i-1][j-1] + C[i-1][j];  
    }  
    return C[n][r];  
}
```

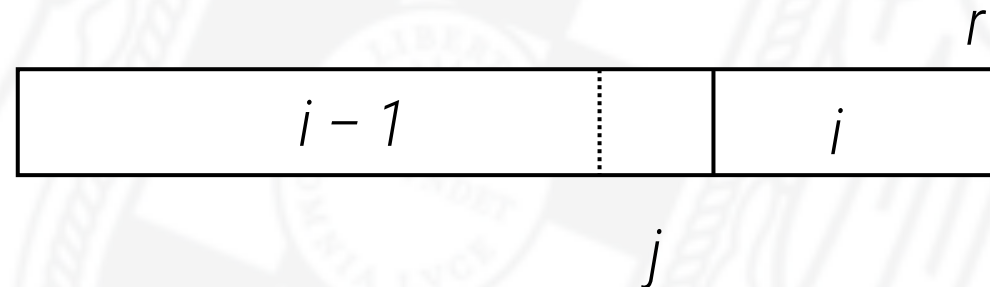
$$\binom{i}{j} = \binom{i-1}{j-1} + \binom{i-1}{j}$$

Programación dinámica ascendente

- Reducción del espacio necesario para la tabla.

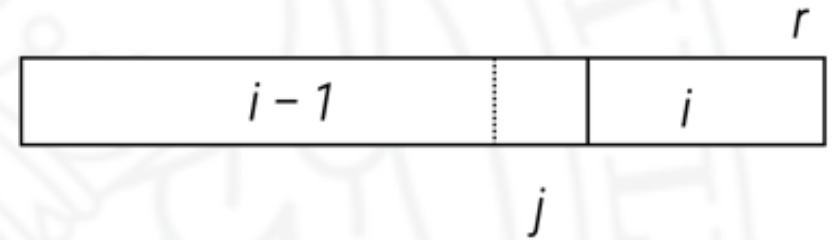


- Actualización sobre el propio vector.



Programación dinámica ascendente

```
int pascal2(int n, int r) {  
    vector<int> C(r+1,0);  
    C[0] = 1;  
    for (int i = 1; i <= n; ++i)  
        for (int j = r; j >= 1; --j)  
            C[j] = C[j-1] + C[j];  
    return C[r];  
}
```



$$\binom{i}{j} = \binom{i-1}{j-1} + \binom{i-1}{j}$$