Comienza a aparecer un patrón. Dada una cadena $\alpha = s_1 \cdots s_r$ que represente a la $r$-combinación $\{s_1, \ldots, s_r\}$, para determinar la siguiente cadena $\beta = t_1 \cdots t_r$, buscamos el elemento de más a la derecha que no tenga su máximo valor ($s_r$ puede tener el valor máximo $n$, $s_{r-1}$ puede tener el valor máximo $n - 1$, y así sucesivamente). Entonces

$$t_i = s_i \quad \text{para } i = 1, \ldots, m - 1.$$

El elemento $t_m$ es igual a $s_m + 1$. Para el resto de la cadena $\beta$ tenemos

$$t_{m+1} \cdots t_r \ (s_m + 2)(s_m + 3) \cdots .$$

A continuación damos el algoritmo.

**ALGORITMO 4.3.9** ***Generación de combinaciones***

Este algoritmo enumera todas las $r$-combinaciones de $\{1, 2, \ldots, n\}$ en orden lexicográfico creciente.

Entrada: $r, n$

Salida: Todas las $r$-combinaciones de $\{1, 2, \ldots, n\}$ en orden lexicográfico creciente.

```
1.  procedure combination(r, n)
2.    for i := 1 to r do
3.      s_i := i
4.    print s_1, ..., s_r   // se imprime la primera r-combinación
5.    for i := 2 to C(n, r) do
6.      begin
7.      m := r
8.      max_val := n
9.      while s_m = max_val do
10.       // se determina el elemento más a la derecha, que no tenga su máximo valor
11.       begin
12.       m := m - 1
13.       max_val := max_val - 1
14.       end
15.     // se incrementa el elemento más a la derecha
16.     s_m := s_m + 1
17.     // el resto de los elementos son los sucesores de s_m
18.     for j := m + 1 to r do
19.       s_j := s_{j-1} + 1
20.     print s_1, ..., s_r   // se imprime la i-ésima combinación
21.     end
22.   end combination
```

**EJEMPLO 4.3.10**

Mostramos la forma en que el algoritmo 4.3.9 genera la 5-combinación de {1, 2, 3, 4, 5, 6, 7} posterior a 23467. Estamos suponiendo que

$$s_1 = 2, \quad s_2 = 3, \quad s_3 = 4, \quad s_4 = 6, \quad s_5 = 7.$$

**EJEMPLO 4.3.13**

El método del ejemplo 4.3.12 permite enumerar las permutaciones de $\{1, 2, 3, 4\}$ en orden lexicográfico como

1234, 1243, 1324, 1342, 1423, 1432, 2134, 2143,
2314, 2341, 2413, 2431, 3124, 3142, 3214, 3241,
3412, 3421, 4123, 4132, 4213, 4231, 4312, 4321. □

A continuación damos el algoritmo.

**ALGORITMO 4.3.14** ***Generación de permutaciones***

Este algoritmo enumera todas las permutaciones de $\{1, 2, \ldots, n\}$ en orden lexicográfico creciente.

Entrada: $n$

Salida: Todas las permutaciones de $\{1, 2, \ldots, n\}$ en orden lexicográfico creciente.

```
1.  procedure permutation(n)
2.    for i := 1 to n do
3.      s_i := i
4.    print s_1, ..., s_n // se imprime la primera permutación
5.    for i := 2 to n! do
6.      begin
7.      m := n − 1
8.      while s_m > s_{m+1} do
9.        // se determina la primera disminución trabajando desde la derecha
10.       m := m − 1
11.     k := n
12.     while s_m > s_k do
13.       // se determina el elemento más a la derecha s_k con s_m < s_k
14.       k := k − 1
15.     swap(s_m, s_k)
16.     p := m + 1
17.     q := n
18.     while p < q do
19.       // se intercambian s_{m+1} y s_n, se intercambian s_{m+2} y s_{n−1}, y así sucesivamente
20.       begin
21.       swap(s_p, s_q)
22.       p := p + 1
23.       q := q − 1
24.       end
25.     print s_1, ..., s_n      // se imprime la i-ésima permutación
26.   end
27. end permutation
```