tonces el problema es elegir un subconjunto $\{i_1, \ldots, i_k\}$ de $\{1, 2, \ldots, n\}$ tal que la suma

$$\sum_{j=1}^{k} t_{i_j} \tag{6.3.1}$$

no exceda $C$ y sea tan grande como sea posible. Un enfoque directo consiste en examinar todos los subconjuntos de $\{1, 2, \ldots, n\}$ y elegir un subconjunto de manera que la suma (6.3.1) no exceda $C$ y sea lo más grande posible. Para llevar a la práctica este enfoque, se necesita un algoritmo que genere todas las combinaciones posibles de un conjunto de $n$ elementos. En esta sección se desarrollan algoritmos para generar combinaciones y permutaciones.

Como existen $2^n$ subconjuntos de un conjunto de $n$ elementos, el tiempo de corrida de un algoritmo que examina todos los subconjuntos es $\Omega(2^n)$. Como se vio en la sección 4.3, es poco práctico correr estos algoritmos excepto para valores pequeños de $n$. Desafortunadamente, existen problemas (un ejemplo de ellos es el problema de llenado de cinta de video que se describió) para los cuales no se conoce un método mucho mejor que el enfoque de "listar todos".

Nuestros algoritmos listan las permutaciones y combinaciones en **orden lexicográfico**. El orden lexicográfico generaliza el orden común del diccionario.

Se tienen dos palabras diferentes y, para determinar si una precede a la otra en el diccionario, se comparan las letras de las palabras. Existen dos posibilidades:

1. Las palabras tienen longitudes diferentes y cada letra en la palabra más corta es idéntica a la letra correspondiente en la palabra más larga.
2. Las palabras tienen la misma longitud o diferente y en alguna posición, las letras de las palabras difieren. **(6.3.2)**

Si se cumple el número 1, la palabra más corta precede a la más larga. (Por ejemplo, "can" precede a "canino" en el diccionario). Si se cumple el número 2, se localiza la posición $p$ más a la izquierda en la que las letras difieren. El orden de las palabras se determina por el orden de las letras en la posición $p$. (Por ejemplo, "gladiador" precede a "gladiolo" en el diccionario. En la posición más a la izquierda en que las letras difieren, encontramos "a" en "gladiador" y "o" en "gladiolo"; "a" precede a "o" en el alfabeto).

El orden lexicográfico generaliza el orden común del diccionario sustituyendo el alfabeto por cualquier conjunto de símbolos donde se ha definido un orden. Se estudiarán las cadenas de enteros.

**Definición 6.3.1 ▶**

Sean $\alpha = s_1 s_2 \cdots s_p$ y $\beta = t_1 t_2 \cdots t_q$ cadenas en el conjunto $\{1, 2, \ldots, n\}$. Se dice que $\alpha$ es *menor que* $\beta$, *en el orden lexicográfico*, y se escribe $\alpha < \beta$ si ocurre

*a)* $p < q$ y $s_i = t_i$ para $i = 1, \ldots, p$,

o

*b)* para alguna $i$, $s_i \neq t_i$, y para la más pequeña de esas $i$, se tiene $s_i < t_i$. ◀

En la definición 6.3.1, el caso *a)* corresponde a la posibilidad 1 de (6.3.2) y el caso *b)* corresponde a la posibilidad 2 de (6.3.2).

**Ejemplo 6.3.2 ▶**

Sean $\alpha = 132$ y $\beta = 1324$ dos cadenas en el conjunto $\{1, 2, 3, 4\}$. En la notación de la definición 6.3.1, $p = 3$, $q = 4$, $s_1 = 1$, $s_2 = 3$, $s_3 = 2$, $t_1 = 1$, $t_2 = 3$, $t_3 = 2$, y $t_4 = 4$. Como $p = 3 < 4 = q$ y $s_i = t_i$ para $i = 1, 2, 3$, se cumple la condición *a)* de la definición 6.3.1. Por lo tanto, $\alpha < \beta$. ◀

**Ejemplo 6.3.3 ▶**

Sean $\alpha = 13246$ y $\beta = 1342$ dos cadenas en $\{1, 2, 3, 4, 5, 6\}$. En la notación de la definición 6.3.1, $p = 5$, $q = 4$, $s_1 = 1$, $s_2 = 3$, $s_3 = 2$, $s_4 = 4$, $s_5 = 6$, $t_1 = 1$, $t_2 = 3$, $t_3 = 4$, y $t_4 = 2$. La $i$ más pequeña para la que $s_i \neq t_i$ es $i = 3$. Como $s_3 < t_3$, por la condición *b)* de la definición 6.3.1, $\alpha < \beta$. ◀

**Ejemplo 6.3.4 ▶**

Sean $\alpha = 1324$ y $\beta = 1342$ dos cadenas en $\{1, 2, 3, 4\}$. En la notación de la definición 6.3.1, $p = q = 4$, $s_1 = 1$, $s_2 = 3$, $s_3 = 2$, $s_4 = 4$, $t_1 = 1$, $t_2 = 3$, $t_3 = 4$, y $t_4 = 2$. La $i$ más pequeña para la que $s_i \neq t_i$ es $i = 3$. Como $s_3 < t_3$, por la condición *b)* de la definición 6.3.1, $\alpha < \beta$. ◀

**Ejemplo 6.3.5 ▶**

Sean $\alpha = 13542$ y $\beta = 21354$ dos cadenas en $\{1, 2, 3, 4, 5\}$. En la notación de la definición 6.3.1, $s_1 = 1$, $s_2 = 3$, $s_3 = 5$, $s_4 = 4$, $s_5 = 2$, $t_1 = 2$, $t_2 = 1$, $t_3 = 3$, $t_4 = 5$, y $t_5 = 4$. La $i$ más pequeña para la que $s_i \neq t_i$ es $i = 1$. Como $s_1 < t_1$, por la condición *b)* de la definición 6.3.1, $\alpha < \beta$. ◀

Para cadenas de la misma longitud en $\{1, 2, \ldots, 9\}$, el orden lexicográfico es el mismo que el orden numérico en los enteros positivos si se interpretan las cadenas como números decimales (vea los ejemplos 6.3.4 y 6.3.5). Para cadenas de diferente longitud, el orden lexicográfico es diferente del orden numérico (vea el ejemplo 6.3.3). En el resto de esta sección, la palabra *orden* se referirá al orden lexicográfico.

Primero se considera el problema de listar todas las combinaciones $r$ de $\{1, 2, \ldots, n\}$. En nuestro algoritmo, se lista la combinación $r$ $\{x_1, \ldots, x_r\}$ como la cadena $s_1 \cdots s_r$, donde $s_1 < s_2 < \cdots < s_r$ y $\{x_1, \ldots, x_n\} = \{s_1, \ldots, s_r\}$. Por ejemplo, la combinación de 3 $\{6, 2, 4\}$ quedará en la lista como 246.

Se presentarán las combinaciones $r$ de $\{1, 2, \ldots, n\}$ en orden lexicográfico. Así, la primera cadena en la lista será $12 \cdots r$ y la última cadena listada será $(n - r + 1) \cdots n$.

**Ejemplo 6.3.6 ▶**

Considere el orden en el que se listan las combinaciones de 5 elementos de $\{1, 2, 3, 4, 5, 6, 7\}$. La primera cadena es 12345, que va seguida de 12346 y 12347. La siguiente cadena es 12356, seguida de 12357. La última cadena será 34567. ◀

**Ejemplo 6.3.7 ▶**

Encuentre la cadena que sigue a 13457 cuando se listan las combinaciones de 5 de $X = \{1, 2, 3, 4, 5, 6, 7\}$.

Ninguna cadena que comienza con 134 y represente una combinación de 5 de $X$ excede a 13467. Así, la cadena que sigue a 13467 debe comenzar con 135. Como 13567 es la cadena más pequeña que comienza con 135 y representa una combinación de 5 elementos de $X$, la respuesta es 13567. ◀

**Ejemplo 6.3.8 ▶**

Encuentre la cadena que sigue a 2367 cuando se listan las combinaciones de 4 de $X = \{1, 2, 3, 4, 5, 6, 7\}$.

Ninguna cadena que comienza con 23 y represente una combinación de 4 de $X$ excede a 2367. Entonces, la cadena que sigue a 2367 debe comenzar con 24. Como 2456 es la cadena más pequeña que comienza con 24 y representa una combinación de 4 elementos de $X$, la respuesta es 2456. ◀

Se está desarrollando un patrón. Dada una cadena $\alpha = s_1 \cdots s_r$, que representa la combinación $r$ $\{s_1, \ldots, s_r\}$, para encontrar la siguiente cadena $\beta = t_1 \cdots t_r$, primero se encuentra el elemento más a la derecha $s_m$ que no tiene su valor máximo. ($s_r$ puede tener el valor máximo $n$, $s_{n-1}$ puede tener el valor máximo $n - 1$, etcétera) 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 $\beta$ se tiene

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

El algoritmo es el siguiente.

**Algoritmo 6.3.9**

**Generación de combinaciones**

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

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

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

**Ejemplo 6.3.10 ▶**

Se mostrará la manera en que el algoritmo 6.3.9 genera la combinación de 5 elementos de $\{1, 2, 3, 4, 5, 6, 7\}$ que sigue a 23467. Se supone que

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

En la línea 13, se encuentra que $s_3$ es el elemento más a la derecha que no tiene su valor máximo. En la línea 14, $s_3$ se iguala a 5. En las líneas 16 y 17, $s_4$ se iguala a 6 y $s_5$ se hace igual a 7. En este punto

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

Se ha generado la combinación de 5, 23567, que sigue a 23467. ◀

**Ejemplo 6.3.11 ▶**

Las combinaciones de 4 de $\{1, 2, 3, 4, 5, 6\}$ según las lista el algoritmo 6.3.9 son

1234, 1235, 1236, 1245, 1246, 1256, 1345, 1346,
1356, 1456, 2345, 2346, 2356, 2456, 3456. ◀

Igual que el algoritmo para generar las combinaciones $r$, el algoritmo para generar las permutaciones lista las permutaciones de $\{1, 2, \ldots, n\}$ en orden lexicográfico. (El ejercicio 16 pide un algoritmo que genere todas las permutaciones $r$ de un conjunto de $n$ elementos).

**Ejemplo 6.3.12 ▶**

Para construir la permutación de $\{1, 2, 3, 4, 5, 6\}$ que sigue a 163542, debemos conservar iguales el mayor número posible de dígitos de la izquierda.

¿Puede la permutación que sigue a la permutación dada tener la forma 1635_ _? Como la única permutación de la forma 1635_ _ distinta de la permutación dada es 163524, y 163524 es menor que 163542, la permutación que sigue a la permutación dada no es de la forma 1635_ _.

¿Puede la permutación que sigue a la permutación dada tener la forma 163_ _ _? Los últimos tres dígitos deben ser una permutación de {2, 4, 5}. Como 542 es la permutación más grande de {2, 4, 5}, cualquier permutación que comience con 163 es más pequeña que la permutación dada. Entonces, la permutación que sigue a la que se da no es de la forma 163_ _ _.

La razón para que la permutación que sigue a la permutación dada no pueda comenzar con 1635 o 163 es que, en cualquiera de los dos casos, el resto de los dígitos de la permutación dada (42 y 542, respectivamente) *decrece*. Por lo tanto, al trabajar desde la derecha debemos encontrar el primer dígito $d$ cuyo vecino de la derecha $r$ satisface $d < r$. En nuestro caso, el tercer dígito, 3, tiene esta propiedad. Entonces, la permutación que sigue a la permutación dada comenzará con 16.

El dígito que sigue a 16 debe exceder a 3. Como se desea la siguiente permutación más pequeña, el siguiente dígito es 4, el dígito más pequeño disponible. Entonces, la permutación deseada comienza con 164. Los dígitos restantes 235 deben estar en orden creciente para lograr el valor mínimo. Por lo tanto, la permutación que sigue a la permutación dada es 164235. ◀

Se puede ver que para generar todas la permutaciones de $\{1, 2, \ldots, n\}$, se comienza con la permutación $12 \cdots n$ y después se usa el método del ejemplo 6.3.12 repetidas veces para generar la siguiente permutación. El proceso termina cuando se genera la permutación $n(n - 1) \cdots 21$.

**Ejemplo 6.3.13 ▶**

Usando el método del ejemplo 6.3.12, se pueden listar 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. ◀

El algoritmo es el siguiente.

**Algoritmo 6.3.14**

**Generación de permutaciones**

Este algoritmo lista 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.  permutación(n) {
2.      for i = 1 to n
3.          s_i = i
4.      println(s_1, . . . , s_n) // imprime la primera permutación
5.      for i = 2 to n! {
6.          m = n − 1
7.          while (s_m > s_{m+1})
8.              // encuentra el primer decremento trabajando
                // desde la derecha
9.              m = m − 1
10.         k = n
11.         while (s_m > s_k)
12.             // encuentra el elemento s_k más a la derecha
                // con s_m < s_k
13.             k = k − 1
14.         swap(s_m, s_k)
15.         p = m + 1
16.         q = n
17.         while (p < q) {
18.             // cambia s_{m+1} y s_n, cambia s_{m+2} y s_{n-1}, etc.
19.             swap(s_p, s_q)
20.             p = p + 1
```

```
21.              q = q − 1
22.          }
23.          println(s_1, . . . , s_n)//imprime la i-ésima permutación
24.      }
25.  }
```

**Ejemplo 6.3.15 ▶**

Se mostrará la manera en que el algoritmo 6.3.14 genera la permutación que sigue a 163542. Suponga que

$$s_1 = 1, \quad s_2 = 6, \quad s_3 = 3, \quad s_4 = 5, \quad s_5 = 4, \quad s_6 = 2$$

y que estamos en la línea 6. El índice mayor $m$ que satisface $s_m < s_{m+1}$ es 3. En las líneas 10 a la 13, se encuentra que el índice más grande $k$ que satisface $s_k > s_m$ es 5. En la línea 14, se intercambian $s_m$ y $s_k$. En este punto, se tiene $s = 164532$. En las líneas 15 a la 22, se invierte el orden de los elementos $s_4s_5s_6 = 532$. Se obtiene la permutación deseada, 164235. ◀

## Sección de ejercicios de repaso

***1.*** Defina *orden lexicográfico*.

***2.*** Describa el algoritmo para generar combinaciones $r$.

***3.*** Describa el algoritmo para generar permutaciones.

## Ejercicios

*En los ejercicios 1 al 3, encuentre la combinación r que generará el algoritmo 6.3.9 con n = 7 después de la combinación r dada.*

***1.*** 1356 **2.** 12367 **3.** 14567

*En los ejercicios 4 al 6, encuentre la permutación que generará el algoritmo 6.3.14 después de la permutación dada.*

***4.*** 12354 **5.** 625431 **6.** 12876543

***7.*** Para cada cadena en los ejercicios 1 al 3, explique (como en el ejemplo 6.3.10) exactamente la forma en que el algoritmo 6.3.9 genera la siguiente combinación $r$.

**8.** Para cada cadena en los ejercicios 4 al 6, explique (como en el ejemplo 6.3.15) exactamente la forma en que el algoritmo 6.3.14 genera la siguiente permutación.

***9.*** Muestre la salida del algoritmo 6.3.9 cuando $n = 6$ y $r = 3$.

**10.** Muestre la salida del algoritmo 6.3.9 cuando $n = 6$ y $r = 2$.

**11.** Muestre la salida del algoritmo 6.3.9 cuando $n = 7$ y $r = 5$.

***12.*** Muestre la salida del algoritmo 6.3.14 cuando $n = 2$.

**13.** Muestre la salida del algoritmo 6.3.14 cuando $n = 3$.

***14.*** Modifique el algoritmo 6.3.9 de manera que la línea 5

5. for $i = 2$ to $C(n, r)$ {

se elimine. Base la condición de terminación en el hecho de que la última combinación $r$ tiene todos los elementos $s_i$ iguales a su valor máximo.

**15.** Modifique el algoritmo 6.3.14 de manera que la línea 5

5. for $i = 2$ to $n!$ {

se elimine. Base la condición de terminación en el hecho de que la última permutación tiene los elementos $s_i$ en orden decreciente.

**16.** Escriba un algoritmo que genere todas las permutaciones $r$ de un conjunto de $n$ elementos.

***17.*** Escriba un algoritmo cuya entrada es una combinación $r$ de $\{1, 2, \ldots, n\}$. La salida es la siguiente combinación $r$ (en orden lexicográfico). La primera combinación $r$ sigue a la última combinación $r$.

**18.** Escriba un algoritmo cuya entrada es una permutación de $\{1, 2, \ldots, n\}$. La salida es la siguiente permutación (en orden lexicográfico). La primera permutación sigue a la última permutación.

***19.*** Escriba una algoritmo cuya entrada es una combinación $r$ $\{1, 2, \ldots, n\}$. La salida es la combinación $r$ anterior (en orden lexicográfico). La última combinación $r$ precede a la primera combinación $r$.

**20.** Escriba un algoritmo cuya entrada es una permutación de $\{1, 2, \ldots, n\}$. La salida es la permutación anterior (en orden lexicográfico). La última permutación precede a la primera.

***★21.*** Escriba un algoritmo recursivo que genere todas las combinaciones $r$ del conjunto $\{s_1, s_2, \ldots, s_n\}$. Divida el problema en dos subproblemas:

- Elabore una lista de las combinaciones $r$ que contienen a $s_1$.
- Elabore una lista de las combinaciones $r$ que no contienen a $s_1$.

**22.** Escriba un algoritmo recursivo que genere todas las permutaciones del conjunto $\{s_1, s_2, \ldots, s_n\}$. Divida el problema en $n$ subproblemas:

- Elabore una lista de las permutaciones que comienzan con $s_1$.
- Elabore una lista de las permutaciones que comienzan con $s_2$.

  .
  .
  .

- Elabore una lista de las permutaciones que comienzan con $s_n$.