unrank is the inverse function of the function rank, i.e., we have

$$\mathsf{rank}(s) = i \Leftrightarrow \mathsf{unrank}(i) = s,$$

for all $s \in \mathcal{S}$ and all $i \in \{0, \ldots, N-1\}$.

Efficient ranking and unranking algorithms have several potential uses. We mention a couple now. One application is the generation of random objects from a specified set $\mathcal{S}$. This can be done easily by generating a random integer $i \in \{0, \ldots, N-1\}$, where $N = |\mathcal{S}|$, and then unranking i. This algorithm ensures that every element of $\mathcal{S}$ is chosen with equal probability $1/N$, assuming that the random number generator being used is unbiased. For an example of the use of such an algorithm, see Algorithm 4.20.

Another use of ranking and unranking algorithms is in storing combinatorial objects in the computer. Instead of storing a combinatorial structure, which could be quite complicated, an alternative would be to simply store its rank, which of course is just an integer. If the structure is needed at any time, then it can be recovered by using the unranking algorithm.

Given a ranking function, rank, defined on $\mathcal{S}$, the successor function, which we name successor, satisfies the following rule:

$$\mathsf{successor}(s) = t \Leftrightarrow \mathsf{rank}(t) = \mathsf{rank}(s) + 1.$$

Thus, $\mathsf{successor}(s)$ is the next element following s in the total ordering. We will use the convention that $\mathsf{successor}(s)$ is undefined if $\mathsf{rank}(s) = N - 1$ (i.e., if s is the last (largest) element in $\mathcal{S}$).

The function successor can easily be constructed from the functions rank and unrank, according to the following formula:

$$\mathsf{successor}(s) = \begin{cases} \mathsf{unrank}(\mathsf{rank}(s)+1) & \text{if } \mathsf{rank}(s) < N-1 \\ \text{undefined} & \text{if } \mathsf{rank}(s) = N-1. \end{cases}$$

However, for a given set $\mathcal{S}$ under consideration, it may be that there is a more efficient way to construct a successor function.

Once we have constructed a successor function, it is a simple matter to generate all the elements in $\mathcal{S}$. We would do this by beginning with the first element of $\mathcal{S}$, and applying the function successor $N-1$ times.

2.2 Subsets

2.2.1 Lexicographic ordering

Suppose that n is a positive integer, and $S = \{1, \ldots, n\}$. Define $\mathcal{S}$ to consist of the 2^n subsets of $S = \{1, \ldots, n\}$. We begin by describing how to generate the subsets in $\mathcal{S}$ in lexicographic order.

Given a subset $T \subseteq S$, let us define the *characteristic vector* of T to be the n-tuple

$$\chi(T) = [x_{n-1}, \ldots, x_0],$$

where

$$x_i = \begin{cases} 1 & \text{if } n-i \in T \\ 0 & \text{if } n-i \notin T. \end{cases}$$

This method of labeling the coordinates $x_{n-1}, \ldots, x_0$ is convenient for the purposes of the algorithms we are going to describe. It is essentially the same method that we used in Section 1.7.1, except that here we are taking the base set to be $\{1, \ldots, n\}$.

Next, define the *lexicographic ordering* on the set of subsets of S to be that induced by the lexicographic ordering of the characteristic vectors. If we think of these characteristic vectors as being the binary representations of the integers from 0 to $2^n - 1$, then this ordering corresponds to the usual ordering of the integers. With respect to this ordering, the rank of a subset T, denoted $\mathsf{rank}(T)$, is just the integer whose binary representation is $\chi(T)$. That is,

$$\mathsf{rank}(T) = \sum_{i=0}^{n-1} x_i 2^i.$$

We illustrate by taking $n = 3$, and tabulating the eight subsets of $S = \{1, 2, 3\}$:

T	$\chi(T) = [x_2, x_1, x_0]$	$\mathsf{rank}(T)$
$\emptyset$	$[0,0,0]$	0
$\{3\}$	$[0,0,1]$	1
$\{2\}$	$[0,1,0]$	2
$\{2,3\}$	$[0,1,1]$	3
$\{1\}$	$[1,0,0]$	4
$\{1,3\}$	$[1,0,1]$	5
$\{1,2\}$	$[1,1,0]$	6
$\{1,2,3\}$	$[1,1,1]$	7

We now present ranking and unranking algorithms for lexicographic generation of subsets. These are very simple. As mentioned above, ranking a subset $T \subseteq \{1, \ldots, n\}$ consists of computing the integer whose binary representation is $\chi(T)$. Unranking an integer r, where $0 \leq r \leq 2^n - 1$, requires the computation of the subset T having rank r. These algorithms are described below without reference to the characteristic vectors $\chi(T)$.