Algorithm 2.1: SUBSETLEXRANK (n, T)

$r \leftarrow 0$
for $i \leftarrow 1$ **to** n
do { **if** $i \in T$ **then** $r \leftarrow r + 2^{n-i}$
return (r)

Algorithm 2.2: SUBSETLEXUNRANK (n, r)

$T \leftarrow \emptyset$
for $i \leftarrow n$ **downto** 1
do { **if** $r \bmod 2 = 1$ **then** $T \leftarrow T \cup \{i\}$; $r \leftarrow \lfloor \frac{r}{2} \rfloor$
return (T)

As an example, suppose that $n = 8$ and $T = \{1, 3, 4, 6\}$. Then Algorithm 2.1 computes

$$\begin{aligned} \mathsf{rank}(T) &= 2^7 + 2^5 + 2^4 + 2^2 \\ &= 128 + 32 + 16 + 4 \\ &= 180. \end{aligned}$$

Conversely, if we run Algorithm 2.2 with $n = 8$ and $r = 180$, then we obtain the following:

i	r	$r \bmod 2$	T
8	180	0	$\emptyset$
7	90	0	$\emptyset$
6	45	1	$\{6\}$
5	22	0	$\{6\}$
4	11	1	$\{4, 6\}$
3	5	1	$\{3, 4, 6\}$
2	2	0	$\{3, 4, 6\}$
1	1	1	$\{1, 3, 4, 6\}$.

This example illustrates that ranking and unranking are inverse operations, since the subset $\{1, 3, 4, 6\}$ has rank 180 and unranking 180 produces the set $\{1, 3, 4, 6\}$.

We have assumed in this section that our base set is $S = \{1, \ldots, n\}$. What would we do if we wanted to rank and unrank the subsets of some other n-element set, say S'? We could of course design algorithms for ranking and unranking subsets of S', but a different approach is usually more convenient. It suffices to construct a bijection $\phi : S' \to S$. Now we can rank any subset $X \subseteq S'$, using a rank function for subsets of S, from the following formula:

$$\mathsf{rank}(X) = \mathsf{rank}(\phi(X)).$$

Similarly, we can unrank r to a subset of S', using the following formula:

$$\mathsf{rank}(r) = \phi^{-1}(\mathsf{unrank}(r)).$$

In the formula above, ϕ^{-1} is the inverse function of ϕ, i.e., $\phi(X) = Y$ if and only if $\phi^{-1}(Y) = X$, where $X \subseteq S'$ and $Y \subseteq S$.

For example if we want to rank and unrank subsets of $S' = \{0, \ldots, n-1\}$, then we can use the bijections ϕ and ϕ^{-1} defined by the following formulas:

$$\phi(X) = \{i + 1 : i \in X\}$$

and

$$\phi^{-1}(Y) = \{i - 1 : i \in Y\}.$$

2.2.2 Gray codes

The lexicographic ordering defined above makes ranking and unranking very simple, but the ordering is not well suited to the sequential generation of all 2^n subsets of an n-set. This is because subsets that are consecutive with respect to the ordering can be very "different." For example, in the case $n = 3$ considered above, $\mathsf{rank}(\{2, 3\}) = 3$ and $\mathsf{rank}(\{1\}) = 4$. Hence, we have two consecutive subsets that are in fact complements of each other (so they are as different as they could possibly be).

Given two subsets $T_1, T_2 \subseteq S$, we define the *symmetric difference* of T_1 and T_2, denoted $T_1 \Delta T_2$, to be

$$T_1 \Delta T_2 = (T_1 \backslash T_2) \cup (T_2 \backslash T_1).$$

The *distance* between T_1 and T_2 is defined to be

$$\mathsf{dist}(T_1, T_2) = |T_1 \Delta T_2|.$$

Alternatively, $\mathsf{dist}(T_1, T_2)$ is equal to the number of coordinates in which $\chi(T_1)$ and $\chi(T_2)$ have different entries, which is called the *Hamming distance* between the vectors $\chi(T_1)$ and $\chi(T_2)$. The relevance of the distance between two subsets is that it represents the number of elements that need to be added to and/or deleted from one subset in order to obtain the other.

If we are going to generate all 2^n subsets sequentially, it might be desirable to do so in such a way that any two consecutive subsets have distance one (the smallest possible). This means that any subset can be obtained from the previous one