Algorithm 2.5: GRAYCODEUNRANK (n, r)

$T \leftarrow \emptyset$
$b' \leftarrow 0$
for $i \leftarrow n-1$ **downto** 0
do
$b \leftarrow \lfloor \frac{r}{2^i} \rfloor$
if $b \neq b'$
then $T \leftarrow T \cup \{n-i\}$
$b' \leftarrow b$
$r \leftarrow r - b\,2^i$
return (T)

Let's work out a couple of examples to illustrate Algorithms 2.4 and 2.5. Suppose that $n = 8$ and $T = \{1,2,3,4,5,7,8\}$. We first use Algorithm 2.4 to compute $\mathsf{rank}(T)$.

i	2^i	$n-i \in T$?	b	r
7	128	yes	1	128
6	64	yes	0	128
5	32	yes	1	160
4	16	yes	0	160
3	8	yes	1	168
2	4	no	1	172
1	2	yes	0	172
0	1	yes	1	173

Thus, $\mathsf{rank}(T) = 173$. The inverse algorithm, Algorithm 2.5, can be used to compute $\mathsf{unrank}(173)$. It executes as follows:

b'	r	i	2^i	b	T
0	173	7	128	1	$\{1\}$
1	45	6	64	0	$\{1,2\}$
0	45	5	32	1	$\{1,2,3\}$
1	13	4	16	0	$\{1,2,3,4\}$
0	13	3	8	1	$\{1,2,3,4,5\}$
1	5	2	4	1	$\{1,2,3,4,5\}$
1	1	1	2	0	$\{1,2,3,4,5,7\}$
0	1	0	1	1	$\{1,2,3,4,5,7,8\}$

Hence, $\mathsf{unrank}(173) = \{1,2,3,4,5,7,8\}$.

2.3 k-Element subsets

2.3.1 Lexicographic ordering

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

A k-element subset $T \subseteq S$ can be written in a natural way as a list

$$\overrightarrow{T} = [t_1, t_2, \ldots, t_k],$$

where

$$t_1 < t_2 < \cdots < t_k.$$

The lexicographic ordering on $\mathcal{S}$ is induced by the lexicographic ordering on the sequences $\overrightarrow{T}$ $(T \in \mathcal{S})$.

We illustrate with a small example. Let $n = 5$ and $k = 3$. The lexicographic ordering of the ten 3-element subsets $T \subseteq \{1,\ldots,5\}$ is as follows:

T	$\overrightarrow{T}$	$\mathsf{rank}(T)$
$\{1,2,3\}$	$[1,2,3]$	0
$\{1,2,4\}$	$[1,2,4]$	1
$\{1,2,5\}$	$[1,2,5]$	2
$\{1,3,4\}$	$[1,3,4]$	3
$\{1,3,5\}$	$[1,3,5]$	4
$\{1,4,5\}$	$[1,4,5]$	5
$\{2,3,4\}$	$[2,3,4]$	6
$\{2,3,5\}$	$[2,3,5]$	7
$\{2,4,5\}$	$[2,4,5]$	8
$\{3,4,5\}$	$[3,4,5]$	9

It is fairly straightforward to describe a successor algorithm for $\mathcal{S}$. This algorithm is presented in Algorithm 2.6.

Algorithm 2.6: KSUBSETLEXSUCCESSOR $(\overrightarrow{T}, k, n)$

$\overrightarrow{U} \leftarrow \overrightarrow{T}$
$i \leftarrow k$
while $(i \geq 1)$ **and** $(t_i = n - k + i)$
do $i \leftarrow i - 1$
if $i = 0$
then return ("undefined")
else
for $j \leftarrow i$ **to** k
do $u_j \leftarrow t_i + 1 + j - i$
return $(\overrightarrow{U})$