Generating Permutations and Combinations

March 10, 2005

1 Generating Permutations

We have learned that there are $n!$ permutations of $\{1, 2, \ldots, n\}$. It is important in many instances to generate a list of such permutations. For example, for the permutation 3142 of $\{1, 2, 3, 4\}$, we may insert 5 in 3142 to generate five permutations of $\{1, 2, 3, 4, 5\}$ as follows:

$$53142, \quad 35142, \quad 31542, \quad 31452, \quad 31425.$$  

If we have a complete list of permutations for $\{1, 2, \ldots, n-1\}$, then we can obtain a complete list of permutations for $\{1, 2, \ldots, n\}$ by inserting $n$ in $n$ ways to each permutation of the list for $\{1, 2, \ldots, n-1\}$.

For $n = 1$, the list is just

$$1$$

For $n = 2$, the list is

$$\begin{array}{cc}
2 & 1 \\
1 & 2
\end{array} \quad \Rightarrow \quad 
\begin{array}{cc}
2 & 1 \\
1 & 2
\end{array}$$

For $n = 3$, the list is

$$\begin{array}{ccc}
1 & 2 & 3 \\
1 & 3 & 2 \\
3 & 1 & 2 \\
3 & 2 & 1 \\
2 & 3 & 1 \\
2 & 1 & 3
\end{array}$$

To generate a complete list of permutations for the set $\{1, 2, \ldots, n\}$, we assign a direction to each integer $k \in \{1, 2, \ldots, n\}$ by writing an arrow above it pointing to the left or to the right:

$$\left\downarrow \begin{array}{c} k \end{array} \right\downarrow \quad \text{or} \quad \begin{array}{c} k \end{array} \left\downarrow \right.$$  

We consider permutations of $\{1, 2, \ldots, n\}$ in which each integer is given a direction; such permutations are called directed permutations. An integer $k$ in a directed permutation is called mobile of its arrow points to a smaller integer adjacent to it. For example, for $3 \ 2 \ 5 \ 4 \ 6 \ 1$, the integers 3, 5, and 6 are mobile. It follows that 1 can never be mobile since there is no integer in $\{1, 2, \ldots, n\}$ smaller than 1. The integer $n$ is mobile, except two cases: (i) $n$ is the first integer and its arrow points to the left, i.e., $\left\downarrow n \right\downarrow \cdots$; (ii) $n$ is the last integer and its arrow points to the right, i.e., $\cdots \left\downarrow n \right\downarrow$.  

$1$
For \( n = 4 \), we have the list

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & 2 & 4 & 3 \\
1 & 4 & 2 & 3 \\
4 & 1 & 2 & 3 \\
4 & 1 & 3 & 2 \\
1 & 4 & 3 & 2 \\
1 & 3 & 4 & 2 \\
1 & 3 & 2 & 4 \\
3 & 1 & 2 & 4 \\
3 & 1 & 4 & 2 \\
3 & 4 & 1 & 2 \\
3 & 4 & 2 & 1 \\
4 & 3 & 1 & 2 \\
4 & 3 & 2 & 1 \\
4 & 1 & 2 & 3 \\
4 & 1 & 3 & 2 \\
3 & 1 & 2 & 4 \\
3 & 1 & 4 & 2 \\
3 & 4 & 1 & 2 \\
3 & 4 & 2 & 1 \\
2 & 3 & 1 & 4 \\
2 & 3 & 2 & 1 \\
2 & 4 & 3 & 1 \\
2 & 4 & 1 & 3 \\
2 & 1 & 3 & 4 \\
2 & 1 & 4 & 3 \\
1 & 3 & 2 & 4 \\
1 & 3 & 4 & 2 \\
1 & 4 & 3 & 2 \\
1 & 4 & 2 & 3 \\
\end{array}
\]

\[
\rightarrow 1 2 3 4 \\
1 2 4 3 \\
1 4 2 3 \\
4 1 2 3 \\
4 1 3 2 \\
1 4 3 2 \\
1 3 4 2 \\
1 3 2 4 \\
3 1 2 4 \\
3 1 4 2 \\
3 4 1 2 \\
3 4 2 1 \\
1 3 4 2 \\
1 3 2 4 \\
1 4 3 2 \\
1 4 2 3 \\
\]

**Algorithm 1.1.** Algorithm for Generating Permutations of \( \{1, 2, \ldots, n\} \):

Step 0. Begin with \( \overleftarrow{1 2 \cdots n} \).

Step 1. Find the largest mobile integer \( m \).

Step 2. Switch \( m \) and the adjacent integer its arrow points to.

Step 3. Switch the directions for all integers \( p > m \).

Step 4. Write down the resulting permutation with directions and return to Step 1.

For example, for \( n = 2 \), we have \( \overleftarrow{1 2} \) and \( \overleftarrow{2 1} \). For \( n = 3 \), we have

\[
\overleftarrow{1 2 3}, \overleftarrow{1 3 2}, \overleftarrow{3 1 2}, \overleftarrow{3 2 1}, \overleftarrow{2 3 1}, \overleftarrow{2 1 3}.
\]

For \( n = 4 \), the algorithm produces the list

\[
\begin{array}{cccc}
\overleftarrow{1} & \overleftarrow{2} & \overleftarrow{3} & \overleftarrow{4} \\
\overleftarrow{1} & \overleftarrow{2} & \overleftarrow{4} & \overleftarrow{3} \\
\overleftarrow{1} & \overleftarrow{4} & \overleftarrow{2} & \overleftarrow{3} \\
\overleftarrow{4} & \overleftarrow{1} & \overleftarrow{2} & \overleftarrow{3} \\
\overleftarrow{4} & \overleftarrow{1} & \overleftarrow{3} & \overleftarrow{2} \\
\overleftarrow{1} & \overleftarrow{4} & \overleftarrow{3} & \overleftarrow{2} \\
\overleftarrow{1} & \overleftarrow{3} & \overleftarrow{4} & \overleftarrow{2} \\
\overleftarrow{1} & \overleftarrow{3} & \overleftarrow{2} & \overleftarrow{4} \\
\overleftarrow{4} & \overleftarrow{3} & \overleftarrow{1} & \overleftarrow{2} \\
\overleftarrow{4} & \overleftarrow{3} & \overleftarrow{2} & \overleftarrow{1} \\
\overleftarrow{1} & \overleftarrow{3} & \overleftarrow{4} & \overleftarrow{2} \\
\overleftarrow{1} & \overleftarrow{3} & \overleftarrow{2} & \overleftarrow{4} \\
\overleftarrow{1} & \overleftarrow{4} & \overleftarrow{3} & \overleftarrow{2} \\
\overleftarrow{1} & \overleftarrow{4} & \overleftarrow{2} & \overleftarrow{3} \\
\overleftarrow{1} & \overleftarrow{3} & \overleftarrow{4} & \overleftarrow{2} \\
\overleftarrow{1} & \overleftarrow{3} & \overleftarrow{2} & \overleftarrow{4} \\
\end{array}
\]

**Proof.** Observe that when \( n \) is not the largest mobile the direction of \( n \) must be either like

\[
\overleftarrow{n \cdots} \text{ or } \overrightarrow{\cdots n}
\]
in the permutation. When the largest mobile $m$ (with $m < n$) is switched with its target integer to produce a new permutation, the direction of $n$ will be changed simultaneously, and the permutation with direction becomes
\[ \overline{n} \cdots \text{ or } \cdots \overline{n} \].

Now $n$ is the largest mobile. Switching $n$ with its target integer for $n - 1$ times to produce $n - 1$ more permutations, we obtain exactly $n$ new permutations (including the one before switching $n$). The algorithm stops at the permutation
\[ \overline{2} \ 1 \ \overline{3} \ 4 \cdots \overline{n}. \]

2 Inversions of Permutations

Let $u_1 u_2 \ldots u_n$ be a permutation of $S = \{1, 2, \ldots, n\}$. We can view $u_1 u_2 \ldots u_n$ as as a bijection $\pi : S \to S$ defined by
\[ \pi(1) = u_1, \ \pi(2) = u_2, \ \ldots, \ \pi(n) = u_n. \]
If $u_i > u_j$ for some $i < j$, then $(u_i, u_j)$ is called an inversion of $\pi$. The number of inversions of $\pi$ is denoted by $\text{inv}(\pi)$. For example, the permutation 3241765 of $\{1, 2, \ldots, 7\}$ has the inversions:
\[ (2, 1), (3, 1), (4, 1), (3, 2), (6, 5), (7, 5), (7, 6). \]

For $k \in \{1, 2, \ldots, n\}$ and $u_j = k$, let $a_k$ be the number of integers that precede $k$ in the permutation $u_1 u_2 \ldots u_n$ but greater than $k$, i.e.,
\[ a_k = \# \{ u_i : u_i > u_j = k, \ i < j \} = \# \{ \pi(i) : \pi(i) > \pi(j) = k, \ i < j \}. \]

It measures how much $k$ is out of order by counting the numbers of integers larger than $k$ but located before $k$. The tuple $(a_1, a_2, \ldots, a_n)$ is called the inversion sequence (or inversion table) of the permutation $\pi = u_1 u_2 \ldots u_n$. The sum $a_1 + a_2 + \cdots + a_n$ measures the total disorder of a permutation.

Example 2.1. The inversion sequence of the permutation 3241765 of $\{1, 2, \ldots, 7\}$ is $(3, 1, 0, 2, 1, 0)$.

It is clear that for any permutation $\pi$ of $\{1, 2, \ldots, n\}$, the inversion sequence $(a_1, a_2, \ldots, a_n)$ of $\pi$ satisfies
\[ 0 \leq a_1 \leq n - 1, \ 0 \leq a_2 \leq n - 2, \ \ldots, \ 0 \leq a_{n-1} \leq 1, \ a_n = 0. \tag{1} \]

It is easy to see that the number of sequences $(a_1, a_2, \ldots, a_n)$ satisfying (1) equals $n \cdot (n-1) \cdots 2 \cdot 1 = n!$. This suggests that the inversion sequences may be characterized by (1).

Theorem 2.1. Let $(a_1, a_2, \ldots, a_n)$ be an integer sequence satisfying
\[ 0 \leq a_1 \leq n - 1, \ 0 \leq a_2 \leq n - 2, \ \ldots, \ 0 \leq a_{n-1} \leq 1, \ a_n = 0. \]

Then there is a unique permutation $\pi$ of $\{1, 2, \ldots, n\}$ whose inversion sequence is $(a_1, a_2, \ldots, a_n)$.

Proof. We give two algorithms to uniquely construct the permutation whose inversion sequence is $(a_1, a_2, \ldots, a_n)$.

Algorithm I. Construction of a Permutation from Its Inversion Sequence:

Step 0. Write down $n$.
Step 1. If $a_{n-1} = 0$, place $n - 1$ before $n$; if $a_{n-1} = 1$, place $n - 1$ after $n$.
Step 2. If $a_{n-2} = 0$, place $n - 2$ before the two members $n$ and $n - 1$; if $a_{n-2} = 1$, place $n - 2$ between $n$ and $n - 1$; if $a_{n-2} = 2$, place $n - 2$ after both $n$ and $n - 1$.

\vdots

Step $k$. If $a_{n-k} = 0$, place $n - k$ to the left of the first position; if $a_{n-k} = 1$, place $n - k$ to the right of the 1st existing number; if $a_{n-k} = 2$, place $n - k$ to the right of the 2nd existing number; \ldots; if $a_{n-k} = k$, place $n - k$ to the right of the last existing number. In general, insert $n - k$ to the right of the $a_{n-k}$th existing number.
Step $n - 1$. If $a_1 = 0$, place 1 before all existing numbers; otherwise, place 1 to the right of the $a_1$th existing number.

For example, for the inversion sequence $(a_1, a_2, \ldots, a_8) = (4, 6, 1, 0, 3, 1, 1, 0)$, its permutation can be constructed by Algorithm I as follows:

8 Write down 8.
87 Since $a_7 = 1$, insert 7 to the right of the first number 8.
867 Since $a_6 = 1$, insert 6 to the right of the first number 8.
8675 Since $a_5 = 3$, insert 5 to the right of the third number 7.
48675 Since $a_4 = 0$, insert 4 to the left of the first number 8.
438675 Since $a_3 = 1$, insert 3 to the right of the first number 4.
4386752 Since $a_2 = 6$, insert 2 to the right of the sixth number 5.
43861752 Since $a_1 = 4$, insert 1 to the right of the fifth number 6.

Algorithm II. Construction of a Permutation from Its Inversion Sequence:

Step 0. Mark down $n$ empty spaces $\square\square\cdots\square\square$.
Step 1. Put 1 into the $(a_1 + 1)$th empty space from left.
Step 2. Put 2 into the $(a_2 + 1)$th empty space from left.
... Step $k$. Put $k$ into the $(a_k + 1)$th empty space from left.
... Step $n$. Put $n$ into the $(a_n + 1)$th empty space (the last empty box) from left.

For example, the permutation for the inversion sequence $(a_1, a_2, \ldots, a_8) = (4, 6, 1, 0, 3, 1, 1, 0)$ can be constructed by Algorithm II as follows:

\[
\begin{array}{c}
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square
\end{array}
\]

Mark down 8 empty spaces.

\[
\begin{array}{c}
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square
\end{array}
\]

Since $a_1 = 4$, put 1 into the 5th empty space.

\[
\begin{array}{c}
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square
\end{array}
\]

Since $a_2 = 6$, put 2 into the 7th empty space.

\[
\begin{array}{c}
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square
\end{array}
\]

Since $a_3 = 1$, put 3 into the 2nd empty space.

\[
\begin{array}{c}
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square
\end{array}
\]

Since $a_4 = 0$, put 4 into the 1st empty space.

\[
\begin{array}{c}
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square
\end{array}
\]

Since $a_5 = 3$, put 5 into the 4th empty space.

\[
\begin{array}{c}
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square
\end{array}
\]

Since $a_6 = 1$, put 6 into the 2nd empty space.

\[
\begin{array}{c}
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square
\end{array}
\]

Since $a_7 = 1$, put 7 into the 2nd empty space.

\[
\begin{array}{c}
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square \\
\square\square\square\square\square\square\square
\end{array}
\]

Since $a_8 = 0$, put 8 into the 1st empty space.

3 Generating Combinations

Let $S$ be an $n$-set. For convenience of generating combinations of $S$, we take $S$ to be the set

$$S = \{x_{n-1}, x_{n-2}, \ldots, x_2, x_1, x_0\}.$$ 

Each subset $A$ of $S$ can be identified as a function $\chi_A : S \rightarrow \{0, 1\}$, called the characteristic function of $A$, defined by

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A. \end{cases}$$

In practice, $\chi_A$ is represented by a 0-1 sequence or a base 2 numeral. For example, for $S = \{x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0\}$,

\[
\begin{array}{c}
\emptyset \\
\{x_7, x_5, x_2, x_1\} \\
\{x_6, x_5, x_3, x_1, x_0\} \\
\{x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0\}
\end{array}
\]

00000000
10100110
01101011
11111111
Algorithm 3.1. The algorithm for Generating Combinations of \( \{x_{n-1}, x_{n-2}, \ldots, x_2, x_1, x_0\} \):

Step 0. Begin with \( a_{n-1} \cdots a_1 a_0 = 0 \cdots 0 \).

Step 1. If \( a_{n-1} \cdots a_1 a_0 = 1 \cdots 11 \), stop.

Step 2. If \( a_{n-1} \cdots a_1 a_0 \neq 1 \cdots 1 \), find the smallest integer \( j \) such that \( a_j = 0 \).

Step 3. Change \( a_j, a_{j+1}, \ldots, a_0 \) (from 0 to 1 or from 1 to 0), write down \( a_{n-1} \cdots a_1 a_0 \), and return to Step 1.

For \( n = 4 \), the algorithm produces the list

\[
\begin{array}{cccc}
0000 & 0100 & 1000 & 1100 \\
0001 & 0101 & 1001 & 1101 \\
0010 & 0110 & 1010 & 1110 \\
0011 & 0111 & 1011 & 1111 \\
\end{array}
\]

The unit \( n \)-cube \( Q_n \) is a graph whose vertex set is the set of all 0-1 sequences of length \( n \), and two sequences are adjacent if they differ in only one place. A Gray code of order \( n \) is a path of \( Q_n \) that visits every vertex of \( Q_n \) exactly once, i.e., a Hamilton path of \( Q_n \). For example,

\[
000 \to 001 \to 101 \to 100 \to 110 \to 010 \to 011 \to 111
\]

is a Gray code of order 3. It is obvious that this Gray code can not be a part of any Hamilton cycle since 000 and 111 are not adjacent. A cyclic Gray code of order \( n \) is a Hamilton cycle of \( Q_n \). For example, the closed path

\[
000 \to 001 \to 011 \to 010 \to 110 \to 111 \to 101 \to 100 \to 000
\]

is a cyclic Gray code of order 3.

For \( n = 1 \), we have the Gray code 0 \( \to \) 1.

For \( n = 2 \), we use 0 \( \to \) 1 to produce the path 00 \( \to \) 01 by adding a 0 in the front, and use 1 \( \to \) 0 to produce 11 \( \to \) 10 by adding a 1 in the front, then combine the two paths to produce the Gray code

\[
00 \to 01 \to 11 \to 10.
\]

For \( n = 3 \), we use the Gray code (of order 2) 00 \( \to \) 01 \( \to \) 11 \( \to \) 10 to produce the path 000 \( \to \) 001 \( \to \) 011 \( \to \) 010 by adding 0 in the front, and use the Gray code 10 \( \to \) 11 \( \to \) 01 \( \to \) 00 (the reverse of 00 \( \to \) 01 \( \to \) 11 \( \to \) 10) to produce the path 110 \( \to \) 111 \( \to \) 101 \( \to \) 100. Combine the two paths to produce the Gray code of order 3

\[
000 \to 001 \to 011 \to 010 \to 110 \to 111 \to 101 \to 100
\]

The Gray codes obtained in this way are called reflected Gray codes.

Algorithm 3.2. Algorithm for Generating reflected Gray codes of order \( n \):

Step 0. Begin with \( a_{n-1} a_{n-2} \cdots a_0 = 00 \cdots 0 \).

Step 1. If \( a_{n-1} a_{n-2} \cdots a_0 = 10 \cdots 0 \), stop.

Step 2. If \( a_{n-1} + a_{n-2} + \cdots + a_0 = \text{even} \), then change \( a_0 \) (from 0 to 1 or 1 to 0).

Step 3. If \( a_{n-1} + a_{n-2} + \cdots + a_0 = \text{odd} \), find the smallest \( j \) such that \( a_j = 1 \) and change \( a_{j+1} \) (from 0 to 1 or 1 to 0).

Step 3. Write down \( a_{n-1} a_{n-2} \cdots a_0 \) and return to Step 1.

We note that if \( a_{n-1} a_{n-2} \cdots a_0 \neq 10 \cdots 0 \) and \( a_{n-1} + a_{n-2} + \cdots + a_0 = \text{odd} \), then \( j \leq n - 2 \) so that \( j + 1 \leq n - 1 \) and \( a_{j+1} \) is defined. We also note that the smallest number \( j \) in Step 3 may be 0, i.e., \( a_0 = 1 \); if so there is no \( i < j \) such that \( a_i = 0 \) and we change \( a_{j+1} = a_1 \) as instructed in Step 3.

Proof. We proceed by induction on \( n \). For \( n = 1 \), it is obviously true. For \( n = 2 \), we have 00 \( \to \) 01 \( \to \) 11 \( \to \) 10. Let \( n \geq 3 \) and assume that it is true for \( 1, 2, \ldots, n-1 \).

(1) When the algorithm is applied, by the induction hypothesis the first resulted \( 2^{n-1} \) words form the reflected Gray code of order \( n - 1 \) with a 0 attached to each word; the \( 2^{n-1} \)th word is 010 \( \cdots \) 0.

(2) Continuing the algorithm, we have

\[
010 \cdots 0 \to 110 \cdots 0.
\]
Now for each word of the form \(11b_{n-3} \cdots b_0\), the parity of \(11b_{n-3} \cdots b_0\) is the same as the parity of \(b_{n-3} \cdots b_0\). Continuing the algorithm, the next \(2^{n-2}\) words (including \(110 \cdots 0\)) form a reflected Gray code (of order \(n-2\)) with a 11 attached at the beginning; the last word is \(1110 \cdots 0\).

(3) Continuing the algorithm, we have

\[
1110 \cdots 0 \rightarrow 1010 \cdots 0 \rightarrow \cdots.
\]

The next \(2^{n-3}\) words (including \(1010 \cdots 0\)) form a reflected Gray code (of order \(n-3\)) with a 101 attached at the beginning; the last word is \(10110 \cdots 0\).

(4) \(10110 \cdots 0 \rightarrow 10010 \cdots 0 \rightarrow \cdots\); there are \(2^{n-4}\) words with 1001 attached at the beginning.

\[
\vdots
\]

\((n-2)\) Continuing the algorithm, we have

\[
10 \cdots 01100 \rightarrow 10 \cdots 00100 \rightarrow .
\]

The next \(2^2\) words (including \(10 \cdots 0100\)) form a reflected Gray code (of order 2) with 10 \cdots 01 attached at the beginning; the last word is \(10 \cdots 0110\).

\((n-1)\) \(10 \cdots 0110 \rightarrow 10 \cdots 0010 \rightarrow 10 \cdots 0011\); there are \(2^1\) words \(10 \cdots 0010\) and \(10 \cdots 0011\) (of order 1) with \(10 \cdots 001\) attached at the beginning.

\((n)\) \(10 \cdots 0011 \rightarrow 10 \cdots 0001\). The algorithm produces only 1 word \(10 \cdots 0001\).

\((n+1)\) Finally, the algorithm ends at \(10 \cdots 0001 \rightarrow 10 \cdots 0000\).

Note that all words produced in Steps \((k)\) \(- (n+1)\) are distinct from the words produced in Step \((k-1)\), where \(2 \leq k \leq n+1\). Thus the words produced by the algorithm are distinct and the total number of words is

\[
1 + 1 + 2 + 2^2 + \cdots + 2^{n-1} = 2^n.
\]

This implies that the sequence of the words produced forms the reflected Gray code of order \(n\).

\[\square\]

### 4 Generating r-Combinations

Let \(S = \{1, 2, \ldots, n\}\). When an \(r\)-combination or \(r\)-subset \(A = \{a_1, a_2, \ldots, a_r\}\) of \(S\) is given, we always assume that \(a_1 < a_2 < \cdots < a_r\). For two \(r\)-combinations \(A = \{a_1, a_2, \ldots, a_r\}\) and \(B = \{b_1, b_2, \ldots, b_r\}\) of \(S\), if there is an integer \(k\) \((1 \leq k \leq r)\) such that

\[
a_1 = b_1, \quad a_2 = b_2, \quad \ldots, \quad a_{k-1} = b_{k-1}, \quad a_k < b_k,
\]

we say that \(A\) precedes \(B\) in the lexicographic order, written \(A < B\). Then the set \(P_r(S)\) of all \(r\)-subsets of \(S\) is linearly ordered by the lexicographic order. For simplicity, we write an \(r\)-combination \(\{a_1, a_2, \ldots, a_r\}\) as an \(r\)-permutation

\[
a_1a_2 \cdots a_r \quad \text{with} \quad a_1 < a_2 < \cdots < a_r.
\]

**Theorem 4.1.** Let \(a_1a_2 \cdots a_r\) be an \(r\)-combination of \(\{1, 2, \ldots, n\}\). The first \(r\)-combination in lexicographic order is \(12 \cdots r\), and the last \(r\)-combination in lexicographic order is

\[
(n-r+1) \cdots (n-1)n.
\]

If \(a_1a_2 \cdots a_k \cdots a_r \neq (n-r+1) \cdots (n-1)n\) and \(k\) is the largest integer such that \(a_k \neq n-r+k\), then the successor of \(a_1a_2 \cdots a_r\) is

\[
a_1a_2 \cdots a_{k-1}(a_k + 1)(a_k + 2) \cdots (a_k + r - k + 1).
\]

**Proof.** Since \(a_i \leq (n-r+i)\) for all \(1 \leq i \leq r\), then \(a_k \neq n-r+k\) implies \(a_k < n-r+k\).

\[\square\]

**Algorithm 4.2.** Algorithm for Generating \(r\)-Combinations of \(\{1, 2, \ldots, n\}\) in Lexicographic Order:

- **Step 0.** Begin with the \(r\)-combination \(a_1a_2 \cdots a_r = 12 \cdots r\).
- **Step 1.** If \(a_1a_2 \cdots a_r = (n-r+1) \cdots (n-1)n\), stop.
- **Step 2.** If \(a_1a_2 \cdots a_r \neq (n-r+1) \cdots (n-1)n\), find the largest \(k\) such that \(a_k < n-r+k\).
- **Step 3.** Change \(a_1a_2 \cdots a_r\) to \(a_1 \cdots a_{k-1}(a_k + 1)(a_k + 2) \cdots (a_k + r - k + 1)\), write down \(a_1a_2 \cdots a_r\), and return back to Step 1.
Example 4.1. The collection of all 4-combinations of \{1, 2, 3, 4, 5, 6\} are listed by the algorithm:

\begin{align*}
1234 & \quad 1245 & \quad 1345 & \quad 1456 & \quad 2356 \\
1235 & \quad 1246 & \quad 1346 & \quad 2345 & \quad 2456 \\
1236 & \quad 1256 & \quad 1356 & \quad 2346 & \quad 3456
\end{align*}

Theorem 4.3. Let \(a_1 a_2 \cdots a_r\) be an \(r\)-combination of \(\{1, 2, \ldots, n\}\). Then the number of \(r\)-combinations up to the place \(a_1 a_2 \cdots a_r\) in lexicographic order equals

\[
\binom{n}{r} - \binom{n-a_1}{r} - \binom{n-a_2}{r-1} - \cdots - \binom{n-a_{r-1}}{2} - \binom{n-a_r}{1}.
\]

Proof. The \(r\)-combinations \(b_1 b_2 \cdots b_r\) after \(a_1 a_2 \cdots a_r\) can be classified into \(r\) kinds:

1. \(b_1 > a_1\); there are \(\binom{n-a_1}{r}\) such \(r\)-combinations.
2. \(b_1 = a_1, b_2 > a_2\); there are \(\binom{n-a_2}{r-1}\) such \(r\)-combinations.
3. \(b_1 = a_1, b_2 = a_2, b_3 > a_3\); there are \(\binom{n-a_3}{r-2}\) such \(r\)-combinations.

\vdots

\(r\) \(b_1 = a_1, \ldots, b_{r-1} = a_{r-1}, b_r > a_r\); there are \(\binom{n-a_r}{2}\) such \(r\)-combinations.

Since the number of \(r\)-combinations of \(\{1, 2, \ldots, n\}\) is \(\binom{n}{r}\), the conclusion follows immediately.

Example 4.2. The 3-combinations of \(\{1, 2, 3, 4, 5\}\) are as follows:

\[123, 124, 125, 134, 135, 145, 234, 235, 245, 345\]

The 3-permutations of \(\{1, 2, 3, 4, 5\}\) can be obtained by making 3! permutations for each 3-combination:

\[123, 124, 125, 134, 135, 145, 234, 235, 245, 345\]

\[132, 142, 152, 143, 153, 154, 234, 235, 245, 254, 345\]

\[213, 214, 215, 314, 315, 415, 234, 235, 245, 345\]

\[231, 241, 251, 341, 351, 451, 342, 352, 452, 453\]

\[312, 412, 512, 413, 513, 514, 423, 523, 524, 534\]

\[321, 421, 521, 431, 531, 541, 432, 532, 542, 543\]

5 Partially Ordered Sets

Definition 5.1. A relation on a set \(X\) is a subset \(R\) of the product set \(X \times X\). A relation \(R\) on \(X\) is called

1. reflexive if \(xRx\) for all \(x \in X\);
2. irreflexive if \(x \not\in R\) for all \(x \in X\);
3. symmetric provided that if \(xRy\) for some \(x, y \in X\) then \(yRx\);
4. antisymmetric provided that if \(xRy\) and \(yRx\) for some \(x, y \in X\) then \(x = y\);
5. transitive provided that if \(xRy\) and \(yRz\) for some \(x, y, z \in X\) then \(xRz\).

Example 5.1. (1) The relation of subset, \(\subseteq\), is a reflexive and transitive relation on the power set \(P(X)\). (2) The relation of divisibility, \(\mid\), is a reflexive and transitive relation on the set of positive integers.

A partial order on a set \(X\) is a reflexive, antisymmetric, and transitive relation. A strict partial order on a set \(X\) is an irreflexive, antisymmetric, and transitive relation. If a relation \(R\) is a partial order, we usually denote \(R\) by \(\leq\); the relation \(<\) is defined by \(a < b\) if and only if \(a \leq b\) but \(a \neq b\). Conversely, for a strict partial order \(<\) on a set \(X\), the relation \(\leq\) defined by \(a \leq b\) if and only if \(a < b\) or \(a = b\) is a partial order. A set \(X\) with a partial order \(\leq\) is called a partially ordered set (or poset for short) and is denoted by \((X, \leq)\). A linear order or total order on a set \(X\) is a strict order \(<\) such that for any two distinct elements \(a\) and \(b\), either \(a < b\) or \(b < a\).
Let \( \leq_1 \) and \( \leq_2 \) be two partial orders on a set \( X \). The poset \((X, \leq_2)\) is called an extension of the poset \((X, \leq_1)\) if, whenever \( a \leq_1 b \), then \( a \leq_2 b \). In particular, an extension of a partial order has more compatible pairs. We show that every finite poset has a linear extension, that is, an extension which is a linearly ordered set.

**Theorem 5.2.** Let \((X, \leq)\) be a finite partially ordered set. Then there is a linear order \( \preceq \) such that \((X, \preceq)\) is an extension of \((X, \leq)\).

**Proof.** We need to show that the elements of \( X \) can be listed in some order \( x_1, x_2, \ldots, x_n \) in such a way that if \( x_i \leq x_j \) then \( x_i \) comes before \( x_j \) in this list, i.e., \( i \leq j \). The following algorithm does the job.

**Algorithm 5.3.** Algorithm for a Linear Extension of an \( n \)-Poset:

1. Choose a minimal element \( x_1 \) from \( X \) (with respect to the ordering \( \leq \)).
2. Delete \( x_1 \) from \( X \); choose a minimal element \( x_2 \) from \( X - \{x_1\} \).
3. Delete \( x_2 \) from \( X - \{x_1\} \); choose a minimal element \( x_3 \) from \( X - \{x_1, x_2\} \).
   
4. \vdots
5. Delete \( x_{n-1} \) from \( X - \{x_1, \ldots, x_{n-2}\} \) and choose the only element \( x_n \) in \( X - \{x_1, \ldots, x_{n-1}\} \).

\( \square \)

A relation \( R \) on \( X \) is called an equivalence relation if \( R \) is reflexive, symmetric, and transitive. For an equivalence relation \( R \) on a set \( X \) and an element \( x \in X \), we call the set \([x] = \{y \in X : xRy\}\) an equivalence class of \( R \) and \( x \) a representative of the equivalence class \([x]\).

**Theorem 5.4.** Let \( R \) be an equivalence relation on a set \( X \). Then for any \( x, y \in X \), the following are logically equivalent: (i) \([x] \cap [y] \neq \emptyset\); (ii) \([x] = [y]\); and (iii) \( xRy \).

A collection \( P = \{A_1, A_2, \ldots, A_k\} \) of nonempty subsets of a set \( X \) is called a partition of \( X \) if \( A_i \cap A_j = \emptyset \) for \( i \neq j \) and \( X = \bigcup_{i=1}^{k} A_i \). We will show below that if \( R \) is an equivalence relation on a set \( X \), then the collection

\[ P_R = \{[x] : x \in X\} \]

is a partition of \( X \). Conversely, if \( P = \{A_1, A_2, \ldots, A_k\} \) is a partition of \( X \), then the relation

\[ R_P = \bigcup_{i=1}^{k} A_i \times A_i \]

is an equivalence relation on \( X \).

**Theorem 5.5.** Let \( R \) be an equivalence relation on a set \( X \), and let \( P = \{A_1, A_2, \ldots, A_k\} \) be a partition of \( X \). Then

1. \( P_R \) is a partition of \( X \);
2. \( R_P \) is an equivalence relation on \( X \);
3. \( R_{P_R} = R, P_{R_P} = P \).