GENERATING ELEMENTARY COMBINATORIAL OBJECTS

Combinatorial Generation

We are going to look at combinatorial generation of:

- Subsets
- k-subsets
- Permutations

To do a sequential generation, we need to impose some order on the set of objects we are generating.

Let $\mathcal{S}$ be a finite set and $N = |\mathcal{S}|$.
A rank function is a bijection

$$\text{RANK}: \mathcal{S} \rightarrow \{0, 1, \ldots, N-1\}.$$

It has another bijection associated with it

$$\text{UNRANK}: \{0, 1, \ldots, N-1\} \rightarrow \mathcal{S}.$$

A rank function defines an ordering on $\mathcal{S}$.
Many types of ordering are possible; we will discuss two types: **lexicographical** ordering and **minimal change** ordering.

Once an ordering is chosen, we can talk about the following types of algorithms:

- Successor: given an object, return its successor.
- Rank: given an object $S \in \mathcal{S}$, return $\text{RANK}(\text{S})$
- Unrank: given a rank $i \in \{0, 1, \ldots, N-1\}$, return $\text{UNRANK}(n)$, its corresponding object.

1. Generating Subsets (of an n-set)

1.1. Generating Subsets: Lexicographical Ordering

Represent a set by its **characteristic vector**:

subset X of {1,2,3}	characteristic vector
{1,2}	[1,1,0]
{3}	[0,0,1]

The **characteristic vector** of a subset $T \subseteq X$ is a vector $\mathcal{X}(T) = [x_{n-1}, x_{n-2}, \ldots, x_1, x_0]$ where

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

Example:

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

Note that the order is lexicographical on $\mathcal{X}(T)$ and not on T.
Note that $\mathcal{X}(T)$ corresponds to the binary representation of rank!

Ranking

More efficient implementation:

```
SubsetLexRank (n, T)
    r ← 0;
    for i ← 1 to n do
        r ← 2 * r;
        if (i ∈ T) then r ← r + 1;
    return r;
```

Books'version:

```
if (i ∈ T) then
   r ← r + 2^(n−i)
```

This is like a conversion from the binary representation to the number.

Unranking

```
SubsetLexUnrank (n, r)
    T ← ∅;
    for i ← n downto 1 do
        if (r mod 2 = 1) then T ← T ∪ {i};
        r ← ⌊r/2⌋;
    return T;
```

This is like a conversion from number to its binary representation.

Successor

The following algorithm is adapted for circular ranking, that is, the successor of the largest ranked object is the object of rank 0.

$\textsc{SubsetLexSuccessor}\ (n, T)$

    $i \leftarrow 0;$

    while $(i \leq n-1)$ and $(n-i \in T)$ do

        $T \leftarrow T \setminus \{n-i\};$

        $i \leftarrow i+1;$

    if $(i \leq n-1)$ then $T \leftarrow T \cup \{n-i\};$

    return $T;$

This algorithm works like an increment on a binary number.

Examples:

1. $\textsc{SubsetLexSuccessor}(3, \{2,3\}) = \{1\}$.

   $\{2,3\}$ $\quad$ $[\overline{0}, \underline{1, 1}]$

   $\{1\}$ $\quad$ $[1, 0, 0]$

2. $\textsc{SubsetLexSuccessor}(4, \{1,4\}) = \{1,3\}$.

   $\{1,4\}$ $\quad$ $[1, 0, \overline{0}, \underline{1}]$

   $\{1,3\}$ $\quad$ $[1, 0, 1, 0]$

1.2. Generating Subsets: Minimal Change Ordering

In minimal change ordering, successive sets are as similar as possible.

The **hamming distance** between two vectors is defined as the number of bits in which the two vectors differ.

Example: $dist(\underline{0}00\underline{1}010, \underline{1}00\underline{0}010) = 2$.

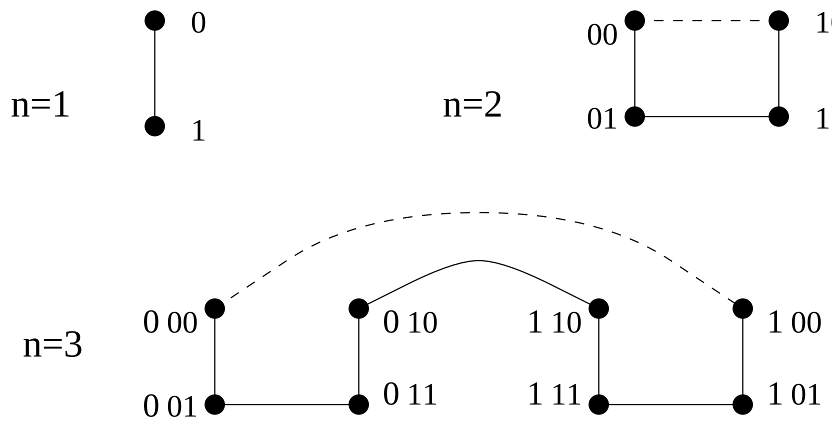
When we apply to the subsets corresponding to the binary vectors, it is equivalent to:
$dist(T_1, T_2) = |T_1 \triangle T_2| = |(T_1 \setminus T_2) \cup (T_2 \setminus T_1)|$.

A **Gray Code** is a sequence of vectors with successive vectors having hamming distance exactly 1.

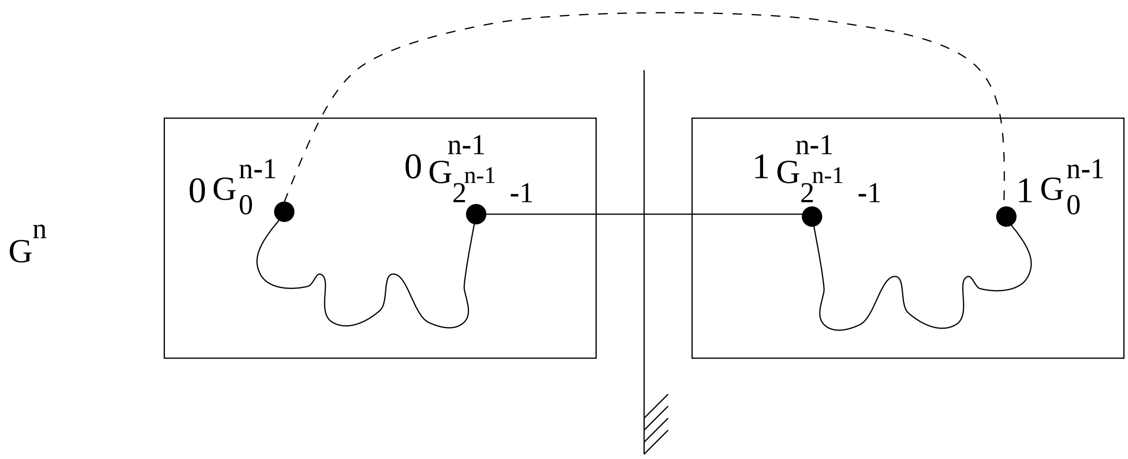
Example: $[00, 01, 11, 10]$.

We will now see a construction for one possible type of Gray Codes...

Construction for Binary Reflected Gray Codes



In general, build G_n as follows:



More formally, we define G^n inductively as follows:

$$
\begin{aligned}
G^1 &= [0, 1] \\
G^n &= [0G_0^{n-1}, \cdots, 0G_{2^{n-1}-1}^{n-1}, 1G_{2^{n-1}-1}^{n-1}, \cdots 1G_0^{n-1}]
\end{aligned}
$$

Theorem 2.1. For any $n \geq 1$, G^n is a gray code.
Exercise: prove this theorem by induction on n.

Successor

Examples:

$$
\begin{aligned}
G_3 &= [000, 001, 011, 010, 110, 111, 101, 100] \\
G_4 &= [0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, \\
&\quad 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000].
\end{aligned}
$$

Rules for calculating successor:

- If vector has even weight (even number of 1's): flip last bit.
- If vector has odd weight (odd number of 1's): from right to left, flip bit after the first 1.

GRAYCODESUCCESSOR (n, T)
    if (T is even) then
    $U \leftarrow T \triangle \{n\}$;
    else
        $j \leftarrow n$;   *(flip last bit)*
        while ($j \notin T$) and ($j > 0$) do $j \leftarrow j - 1$;
        if ($j = 1$) then $U \leftarrow \emptyset$;   *(I changed for circular order)*
                else $U \leftarrow T \triangle \{j - 1\}$;
    return U;

Ranking and Unranking

	r	0	1	2	3	4	5	6	7
$b_3b_2b_1b_0$	bin.rep. r	000	001	010	011	100	101	110	111
$a_2a_1a_0$	G^3_r	000	001	011	010	110	111	101	100

Set $b_3 = 0$ in the example above.

We need to relate $(b_n b_{n-1} \dots b_0)$ and $(a_{n-1}a_{n-2}, \dots a_0)$.

Lemma 1.
Let $P(n)$: "For $0 \leq r \leq 2^n - 1, a_j \equiv b_j + b_{j+1} \pmod 2$, for all $0 \leq j \leq n-1$". Then, $P(n)$ holds for all $n \geq 1$.

Proof: We will prove $P(n)$ by induction on n.
Basis: $P(1)$ holds, since $a_0 = b_0$ and $b_1 = 0$.

Induction step: Assume $P(n-1)$ holds. We will prove $P(n)$ holds.
Case 1. $r \leq 2^{n-1} - 1$ **(first half of G_n).**
Note that $b_{n-1} = 0 = a_{n-1}$ and $b_n = 0$, which implies

$$a_{n-1} = 0 = b_{n-1} + b_n. \tag{1}$$

By induction,

$$a_j \equiv b_j + b_{j+1} \pmod 2, \text{ for all} 0 \leq j \leq n-2. \tag{2}$$

Equations (1) and (2) imply $P(n)$.

Case 2. $2^n \leq r \leq 2^n - 1$ **(second half of** G_n**).**
Note that $b_{n-1} = 1 = a_{n-1}$ and $b_n = 0$, which implies

$$a_{n-1} \equiv 1 \equiv b_{n-1} + b_n \pmod 2. \tag{3}$$

Now, $G_r^n = 1G_{2^n-1-r}^{n-1} = 1a_{n-2}a_{n-3}\ldots a_1a_0$. The binary representation of $2^n - 1 - r$ is
$0(1-b_{n-2})(1-b_{n-3})\ldots(1-b_1)(1-b_0)$.
By induction hypothesis we know that, for all $0 \leq j \leq n-2$,

$$\begin{aligned} a_j &\equiv (1-b_j) + (1-b_{j+1}) \pmod 2 \quad (4) \\ &\equiv b_j + b_{j+1} \pmod 2 \quad (5) \end{aligned}$$

Equations (3) and (5) imply $P(n)$.

Lemma 2.
Let $n \geq 1$, $0 \leq r \leq 2^n - 1$. *Then,*

$$b_j \equiv \sum_{i=j}^{n-1} a_i \pmod 2, \quad \text{for all } 0 \leq j \leq n-1.$$

Proof:

$$\begin{aligned} \sum_{i=j}^{n-1} a_i &\equiv \sum_{i=j}^{n-1} b_i + b_{i+1} \pmod 2 \quad \text{[By Lemma 1]} \\ &\equiv b_j + 2b_{j+1} + \ldots + 2b_{n-1} + b_n \pmod 2 \\ &\equiv b_j + b_n \pmod 2 \\ &\equiv b_j \pmod 2 \qquad \text{[Since } b_n = 0\text{]}. \end{aligned}$$

Let $n \geq 1$, $0 \leq r \leq 2^n - 1$.
We haved proved the following properties hold, for all $0 \leq j \leq n-1$,

$$b_j \equiv \sum_{i=j}^{n-1} a_i \pmod 2.$$

$$a_j \equiv b_j + b_{j+1} \pmod 2,$$

The first property is used for ranking:

GrayCodeRank (n, T)

$r \leftarrow 0$; $b \leftarrow 0$;
for $i \leftarrow n-1$ downto 0 do
  if $((n-i) \in T)$ then   (*if* $a_i = 1$)
    $b \leftarrow 1 - b$;   ($b_i = \overline{b_{i+1}}$)
  $r \leftarrow 2r + b$;
return r;

The second property is used for unranking:

GrayCodeUnrank (n, r)

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

2. Generating k-subsets (of an n-set)

2.1. Generating k-subsets: Lexicographical Ordering

Example: $k = 3$, $n = 5$.

rank	T	$\vec{T}$
0	$\{1,2,3\}$	$[1,2,3]$
1	$\{1,2,4\}$	$[1,2,4]$
2	$\{1,2,5\}$	$[1,2,5]$
3	$\{1,3,4\}$	$[1,3,4]$
4	$\{1,3,5\}$	$[1,3,5]$
5	$\{1,4,5\}$	$[1,4,5]$
6	$\{2,3,4\}$	$[2,3,4]$
7	$\{2,3,5\}$	$[2,3,5]$
8	$\{2,4,5\}$	$[2,4,5]$
9	$\{3,4,5\}$	$[3,4,5]$

Successor

IDEA: $n = 10$, Successor($\{\ldots, \underline{5}, 8, 9, 10\}$)=$\{\ldots, \underline{6}, 7, 8, 9\}$

kSubsetLexSuccessor ($\vec{T}, k, n$)

$\vec{U} \leftarrow \vec{T}$; $i \leftarrow k$;
while ($i \geq 0$) and ($t_i = n - k + i$) do $i \leftarrow i - 1$;
if ($i = 0$) then $\vec{U} = [1, 2, \ldots, k]$;
else for $j \leftarrow i$ to k do
    $u_j \leftarrow (t_i + 1) + j - i$;
return $\vec{U}$;

Ranking

How many subsets preceed $\vec{T} = [t_1, t_2, \ldots, t_k]$?

all sets $[X, \ldots]$ with $1 \leq X \leq t_1 - 1$
$(\sum_{j=1}^{t_1-1} \binom{n-j}{k-1})$

all sets $[t_1, X, \ldots]$ with $t_1 + 1 \leq X \leq t_2 - 1$
$(\sum_{j=t_1+1}^{t_2-1} \binom{n-j}{k-2})$
$\vdots$
all sets $[t_1, \ldots, t_{k-1}, X, \ldots]$ with $t_{k-1} + 1 \leq X \leq t_k - 1$
$(\sum_{j=t_{k-1}+1}^{t_k-1} \binom{n-j}{k-(k-1)})$

Thus,

$$rank(T) = \sum_{i=1}^{k} \sum_{j=t_{i-1}+1}^{t_i-1} \binom{n-j}{k-i}.$$

kSubsetLexRank $(\vec{T}, k, n)$

$r \leftarrow 0$;
$t_0 \leftarrow 0$;
for $i \leftarrow 1$ to k do
  for $j \leftarrow t_{i-1} + 1$ to $t_i - 1$ do
    $r \leftarrow r + \binom{n-j}{k-i}$;
return r;

Unranking

$t_1 = x \iff \Sigma_{j=1}^{x-1} \binom{n-j}{k-1} \leq r < \Sigma_{j=1}^{x} \binom{n-j}{k-1}$
$t_2 = x \iff \Sigma_{j=t_1+1}^{x-1} \binom{n-j}{k-2} \leq r - \Sigma_{j=1}^{t_1-1} \binom{n-j}{k-1} < \Sigma_{j=t_1+1}^{x} \binom{n-j}{k-1}$
etc.

kSubsetLexUnrank (r, k, n)

```
x ← 1;
for i ← 1 to k do
    while (r ≥ C(n−x, k−i)) do
        r ← r − C(n−x, k−i);
        x ← x + 1;
    t_i ← x;
    x ← x + 1;
return T⃗ ;
```

2.2 Generating k-subsets: Minimal Change Ordering

The minimum Hamming distance possible between k-subsets is 2.

Revolving Door Ordering

It is based on Pascal's Identity: $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.
We define the sequence of k-subsets $A^{n,k}$ based on $A^{n-1,k}$ and the reverse of $A^{n-1,k-1}$, as follows:

$$\begin{aligned}
A^{n,k} &= \left[A_0^{n-1,k}, \ldots, A_{\binom{n-1}{k}-1}^{n-1,k}, |A_{\binom{n-1}{k-1}-1}^{n-1,k-1} \cup \{n\}, \ldots, A_0^{n-1,k-1} \cup \{n\}\right], \\
&\quad for\ 1 \leq k \leq n-1 \\
A^{n,0} &= [\emptyset] \\
A^{n,n} &= [\{1, 2, \ldots, n\}]
\end{aligned}$$

Example: Bulding $A^{5,3}$ using $A^{4,3}$ and $A^{4,2}$:

$$\begin{aligned}
A^{4,3} &= [\{1,2,3\}, \{1,3,4\}, \{2,3,4\}, \{1,2,4\}] \\
A^{4,2} &= [\{1,2\}, \{2,3\}, \{1,3\}, \{3,4\}, \{2,4\}, \{1,4\}]
\end{aligned}$$

$$\begin{aligned}
A^{5,3} &= [\{1,2,3\}, \{1,3,4\}, \{2,3,4\}, \{1,2,4\}, | \\
&\quad |\{1,4,\mathbf{5}\}, \{2,4,\mathbf{5}\}, \{3,4,\mathbf{5}\}, \{1,3,\mathbf{5}\}, \{2,3,\mathbf{5}\}, \{1,2,\mathbf{5}\}]
\end{aligned}$$

To see that the revolving door ordering is a minimal change ordering, prove:

1. $A^{n,k}_{\binom{n}{k}-1} = \{1, 2, \ldots, k-1, n\}$.
2. $A^{n,k}_0 = \{1, 2, \ldots, k\}$.
3. For any n, k, $1 \leq k \leq n$, $A^{n,k}$ is a minimal ordering of $\mathcal{S}^n_k$.

Ranking

The ranking algorithm is based on the following fact (prove it as an exercise):

$$rank(T) = \sum_{i=1}^{k} (-1)^{k-i} \left(\binom{t_i}{i} - 1 \right) = \begin{cases} \sum_{i=1}^{k} (-1)^{k-i} \binom{t_i}{i}, & k \text{ even} \\ \left[\sum_{i=1}^{k} (-1)^{k-i} \binom{t_i}{i} \right] - 1, & k \text{ odd} \end{cases}$$

Hint: Prove the first equality by induction and the second, directly.

KsubsetRevDoorRank($\vec{T}, k$)
  $r \leftarrow -(k \bmod 2)$;
  $s \leftarrow 1$;
  for $i \leftarrow k$ downto 1 do
    $r \leftarrow r + s\binom{t_i}{i}$
    $s \leftarrow -s$;
  return r;

Unranking

IDEA

Example: $n = 7$, $k = 4$, $r = 8$

$4 \in T$, $5, 6, 7 \notin T$	$5 \in T$, $6, 7 \notin T$	$6 \in T$, $7 \notin T$	$7 \in T$
$\binom{4}{4} = 1$	$\binom{5}{4} = 5$	$\binom{6}{4} = 15$	$\binom{7}{4} = 21$

We can determine the largest element in the set:
$r = 8$ implies $\{\_, \_, \_, 6\}$.

Now, solve it recursively for $n' = 5$, $k' = 3$, $r' = \binom{6}{4} - r - 1 = 6$.

```
KsubsetRevDoorUnrank(r, k, n)
        x ← n;
        for i ← k downto 1 do
            While (x choose i) > r do   x ← x − 1;
            t_i ← x + 1
            r ← (x+1 choose i) − r − 1;
        return T⃗;
```

$$
\begin{aligned}
&\textsc{KsubsetRevDoorUnrank}(r, k, n)\\
&\quad x \leftarrow n;\\
&\quad \text{for } i \leftarrow k \text{ downto } 1 \text{ do}\\
&\quad\quad \text{While } \binom{x}{i} > r \text{ do } \quad x \leftarrow x - 1;\\
&\quad\quad t_i \leftarrow x + 1\\
&\quad\quad r \leftarrow \binom{x+1}{i} - r - 1;\\
&\quad \text{return } \vec{T};
\end{aligned}
$$

Successor

Let $\vec{T} = [1, 2, 3, \ldots, j-1, t_j, \ldots]$, where $j = \min\{i : t_i \neq i\}$.
Consider fours cases for computing successor:

- Case A: $k \equiv j \pmod 2$
  - Case A1: if $t_{j+1} = t_j + 1$ then move j to the right, and remove $t_j + 1$.
    Example: $\textsc{Successor}(\{\underline{1, 2, 3}, \mathbf{7}, \overline{8}, 12\}) = \{\{\underline{1, 2, 3, 4}, \mathbf{7}, 12\}$.
  - Case A2: if $t_{j+1} \neq t_j + 1$ then move j to the left, and add $t_j + 1$.
    Example:
    $\textsc{Successor}(\{\underline{1, 2, 3}, \mathbf{7}, 10, 12\}) = \{\underline{1, 2}, \mathbf{7}, \overline{8}, 10, 12\}$.
- Case B: $k \not\equiv j \pmod 2$
  - Case B1: if $j > 1$ then increment t_{j-1} and (if exists) t_{j-2}.
    Example: $\textsc{Successor}(\{\underline{1, 2, 3}, \mathbf{7}, 10\}) = \{1, \underline{3, 4}, \mathbf{7}, 10\}$.
  - Case B2: if $j = 1$ then decrement t_1
    Example: $\textsc{Successor}\{7, 9, 10, 12\}) = \{6, 9, 10, 12\}$.

For each case, prove RANK(SUCCESSOR(T))−RANK(T) $= 1$.

Case A1: SUCCESSOR($\{\underline{1,2,3},\mathbf{7},\overline{8},12\}$) $= \{\{\underline{1,2,3,4},\mathbf{7},12\}$.

RANK(SUCCESSOR(T))−RANK(T) =

$$= (-1)^{k-j}\binom{j}{j} + (-1)^{k-j-1}\binom{t_j}{j+1}$$
$$-(-1)^{k-j}\binom{t_j}{j} - (-1)^{k-j-1}\binom{t_j+1}{j+1}$$
$$= \binom{j}{j} + \left(\binom{t_j+1}{j+1} - \binom{t_j}{j+1} - \binom{t_j}{j}\right) = 1 + 0 = 1.$$

Prove cases A2, B1, B2...

KSUBSETREVDOORSUCCESSOR($\vec{T}, k, n$)

$t_{k+1} \leftarrow n+1$;
$j \leftarrow 1$;
While ($j \leq k$) and ($t_j = j$) do $j \leftarrow j+1$;
if ($k \not\equiv j \pmod 2$) then
  if ($j = 1$) then $t_1 \leftarrow t_1 - 1$; (Case B2)
  else (Case B1)
    $t_{j-1} \leftarrow j$;
    $t_{j-2} \leftarrow j-1$;
else
  if ($t_{j+1} \neq t_j + 1$) then (Case A2)
    $t_{j-1} \leftarrow t_j$;
    $t_j \leftarrow t_j + 1$
  else (Case A1)
    $t_{j+1} \leftarrow t_j$;
    $t_j \leftarrow j$;
return $\vec{T}$;

3. Generating Permutations

A permutation is a bijection $\Pi : \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\}$.

We represent it by a list: $\Pi = [\Pi[1], \Pi[2], \ldots, \Pi[n]]$.

3.1. Generating Permutations:Lexicographical Ordering

$n = 3$

rank	permutation
0	$[1, 2, 3]$
1	$[1, 3, 2]$
2	$[2, 1, 3]$
3	$[2, 3, 1]$
4	$[3, 1, 2]$
5	$[3, 2, 1]$

Successor

Example: $\Pi = [3, 5, \mathbf{4}, \underline{7, \overline{6}, 2, 1}]$
Let i = index right before a decreasing suffix = 3.
Let j= index of the successor of $\pi[i] = 4$ in $\{\Pi[i+1], \ldots, \Pi[n]\} = \{7, 6, 2, 1\}$, $j = 5$.
Swap $\Pi[i]$ and $\Pi[j]$, and reverse $\{\Pi[i+1], \ldots, \Pi[n]\}$.

$$\textsc{Successor}(\Pi) = [3, 5, \mathbf{6}, \underline{1, 2, 4, 7}]$$

Note that

$$\begin{aligned} i &= \max\{l : \Pi[l] < \Pi[l+1]\} \\ j &= \max\{l : \Pi[l] > \Pi[i]\}. \end{aligned}$$

For the algorithm, we add: $\Pi[0] = 0$.

```
PermLexSuccessor(n, Π)
    Π[0] ← 0;
    i ← n − 1;
    while (Π[i] > Π[i + 1]) do  i ← i − 1;
    if (i = 0) then return Π = [1, 2, ..., n]
    j ← n;
    while (Π[j] < Π[i]) do  j ← j − 1;
    t ← Π[j]; Π[j] ← Π[i]; Π[i] ← t; (swap Π[i] and Π[j])
    // In-place reversal of Π[i + 1], ..., Π[n]:
    for h ← i + 1 to ⌊(n−i)/2⌋ do
        t ← Π[h]; Π[h] ← Π[n + i + 1 − h];
        Π[n + i + 1 − h] ← t;
    return Π;
```

Ranking

How many permutations come before
$\Pi = [3,5,1,2,4]$?

the ones of the form $\Pi = [1,\ldots]$ (there are $(n-1)! = 24$ of them)
the ones of the form $\Pi = [2,\ldots]$ (there are $(n-1)! = 24$ of them)

plus the rank of $[5,1,2,4]$ as a permutation of $\{1,2,4,5\}$, which is the standard rank of $[4,1,2,3]$.
So,
$\textsc{Rank}([3,5,1,2,4]) = 2 \times 4! + \textsc{Rank}([4,1,2,3])$
$= 2 \times 4! + 3 \times 3! + \textsc{Rank}([1,2,3])$
$= 2 \times 4! + 3 \times 3! + 0 \times 2! + \textsc{Rank}([1,2])$
$= 2 \times 4! + 3 \times 3! + 0 \times 2! + 0 \times 1! + \textsc{Rank}([1])$
$= 2 \times 4! + 3 \times 3! + 0 \times 2! + 0 \times 1! + 0 = 66$

General Formula:

$\textsc{Rank}([1],1) = 0,$
$\textsc{Rank}(\Pi,n) = (\Pi[1]-1) \times (n-1)! + \textsc{Rank}(\Pi', n-1)$, where

$$\Pi'[i] = \begin{cases} \Pi[i+1]-1, & \text{if } \Pi[i+1] > \Pi[1] \\ \Pi[i+1], & \text{if } \Pi[i+1] < \Pi[1] \end{cases}$$

$\text{PermLexRank}(n, \Pi)$

$r \leftarrow 0;$
$\Pi' \leftarrow \Pi;$
for $j \leftarrow 1$ to $n-1$ do (Note: correction from book: $n \rightarrow n-1$)
  $r \leftarrow r + (\Pi'[j] - 1) * (n-j)!$
  for $i \leftarrow j+1$ to n do
    if $(\Pi'[i] > \Pi'[j])$ then $\Pi'[i] = \Pi'[i] - 1;$
return $r;$

Unranking

Unranking uses the factorial representation of r.
Let $0 \leq r \leq n! - 1$. Then, $(d_{n-1}, d_{n-2}, \ldots, d_1)$ is the factorial representation of r if

$$r = \Sigma_{i=1}^{n-1} d_i \times i!, \text{ where } 0 \leq d_i < i.$$

(Exercise: prove that such r has a unique factorial representation.)

Examples:

1. $\text{UNRANK}(15, 4) = [3, 2, 4, 1]$
   $15 = \mathbf{2} \times 3! + \mathbf{1} \times 2! + \mathbf{1} \times 1!$, put $d_0 = \mathbf{0}$.

2	**1**	$\underline{\mathbf{1}}$	**0**
3	2	2	1

2	$\underline{\mathbf{1}}$	**1**	**0**
3	2	2	1

$\underline{\mathbf{2}}$	**1**	**1**	**0**
3	2	3	1

2	**1**	**1**	**0**
3	2	4	1

2. $\text{UNRANK}(8, 4) = [2, 3, 1, 4]$
   $8 = \mathbf{1} \times 3! + \mathbf{1} \times 2! + \mathbf{0} \times 1!$,

1	**1**	$\underline{\mathbf{0}}$	**0**
2	2	1	1

1	$\underline{\mathbf{1}}$	**0**	**0**
2	2	1	2

$\underline{\mathbf{1}}$	**1**	**0**	**0**
2	2	1	3

1	**1**	**0**	**0**
2	3	1	4

3. $\text{UNRANK}(21, 4) = [4, 2, 3, 1]$
   $21 = \mathbf{3} \times 3! + \mathbf{1} \times 2! + \mathbf{1} \times 1!$,

3	**1**	$\underline{\mathbf{1}}$	**0**
4	2	2	1

3	$\underline{\mathbf{1}}$	**1**	**0**
4	2	2	1

$\underline{\mathbf{3}}$	**1**	**1**	**0**
4	2	3	1

3	**1**	**1**	**0**
4	2	3	1

Justification: $\Pi[1] = d_{n-1} + 1$ because exactly d_{n-1} blocks of size $(n-1)!$ come before Π.
$\Pi[2], \Pi[3], \ldots, \Pi[n]$ is computed from permutation Π', in the following way:

$r' = r - d_{n-1} \times (n-1)!$
$\Pi' =$UNRANK$(r', n-1)$,

$$\Pi[i] = \begin{cases} \Pi'[i-1], & \text{if } \Pi'[i-1] < \Pi[1] \\ \Pi'[i-1]+1, & \text{if } \Pi'[i-1] > \Pi[1] \end{cases} \qquad \text{for } 2 \leq i \leq n$$

```
PermLexUnrank(r, n)
    Π[n] ← 1;
    for j ← 1 to n − 1 do
        d ← (r mod (j+1)!) / j!;   // calculates d_j
        r ← r − d ∗ j!;
        Π[n − j] ← d + 1;
        for i ← n − j + 1 to n do
            if (Π[i] > d) then Π[i] ← Π[i] + 1;
    return Π;
```

3.2. Generating permutations:Minimal Change Ordering

Minimal change for permutations: two permutatiosn must differ by adjacent transposition.

The Trotter-Johnson algorithm follows the following ordering:

$$
\begin{aligned}
T^1 &= [[1]] \\
T^2 &= [[1, \mathbf{2}], [\mathbf{2}, 1]] \\
T^3 &= [[1, 2, \mathbf{3}], [1, \mathbf{3}, 2], [\mathbf{3}, 1, 2][\mathbf{3}, 2, 1], [2, \mathbf{3}, 1], [2, 1, \mathbf{3}]]
\end{aligned}
$$

How to build T^3 using T^2:

$$
\begin{array}{cccc}
 & 1 & & 2 \;\; \mathbf{3} \\
 & 1 & \mathbf{3} & 2 \\
\mathbf{3} & 1 & & 2 \\
\hline
\mathbf{3} & 2 & & 1 \\
 & 2 & \mathbf{3} & 1 \\
 & 2 & & 1 \;\; \mathbf{3}
\end{array}
$$

See picture for T^4 in page 58 of the textbook.

Ranking

Let

$$\Pi = [\Pi[1], \ldots, \Pi[k-1], \mathbf{\Pi[k] = n}, \Pi[k+1], \ldots, \Pi[n]].$$

Thus, Π is built from Π' by inserting n, where

$$\Pi' = [\Pi[1], \ldots, \Pi[k-1], \Pi[k+1], \ldots, \Pi[n]].$$

$\textsc{Rank}(\Pi, n) = n \times \textsc{Rank}(\Pi', n-1) + E$,

$$E = \begin{cases} n-k, & \text{if Rank}(\Pi', n-1) \text{ is } \textit{even} \\ k-1, & \text{if Rank}(\Pi', n-1) \text{ is } \textit{odd} \end{cases}$$

Example:
$\textsc{Rank}([3,4,2,1],4) = 4 \times \textsc{Rank}([3,2,1],3) + E =$
$4 \times 3 + (2-1) = 13.$

```
PermTrotterJohnsonRank(Π, n)
    r ← 0;
    for j ← 2 to n do
        k ← 1; i ← 1;
        while (Π[i] ≠ j) do
            if (Π[i] < j) then k ← k + 1;
            i ← i + 1;
        if (r ≡ 0 mod 2) then r ← j * r + j − k;
                         else r ← j * r + k − 1;
    return r;
```

Unranking

Based on similar recursive principle.
Let $r' = \lfloor \frac{r}{n} \rfloor$, $\Pi' =$ UNRANK$(r', n-1)$.
Let $k = r - n \times r'$.
Insert n into Π' in position:

$$\begin{array}{ll} k+1, & \text{if } r' \text{ is odd} \\ n-k, & \text{if } r' \text{ is even} \end{array}$$

```
PermTrotterJohnsonUnrank(n, r)
        Π[1] ← 1;
        r_2 ← 0;
        for j ← 2 to n do
            r_1 ← ⌊(r*j!)/n!⌋;    // rank of Π when restricted to {1, 2, ..., j}
            k ← r_1 − j * r_2;
            if (r_2 is even) then
               for i ← j − 1 downto j − k do
                    Π[i + 1] ← Π[i];
               Π[j − k] ← j;
            else
                 for i ← j − 1 downto k + 1 do
                      Π[i + 1] ← Π[i];
                 Π[k + 1] ← j;
            r_2 ← r_1;
        return Π;
```

Successor

There are four cases to analyse:

- RANK(Π') is even
  - If possible, move left: SUCCESSOR($[1, \mathbf{4}, 2, 3]$)=($[\mathbf{4}, 1, 2, 3]$)
  - If n is in first position, get successor of the remaining permutation: SUCCESSOR($[\mathbf{4}, 1, 2, 3]$)=($[\mathbf{4}, 1, 3, 2]$),
- RANK(Π') is odd
  - If possible, move right: SUCCESSOR($[3, \mathbf{4}, 2, 1]$)=($[3, 2, \mathbf{4}, 1]$)
  - If n is in last position, get successor of the remaining permutation: SUCCESSOR($[3, 2, 1, \mathbf{4}]$)=($[2, 3, 1, \mathbf{4}]$).

We need to be able to determine the parity of RANK(Π').

The parity of a permutation is the parity of the number of interchanges necessary for transforming the permutation into $[1, 2, \ldots, n]$.
$\Pi' = [5, 1, 3, 4, 2]$ is an even permutation since 2 steps are sufficient to convert it into $[1, 2, 3, 4, 5]$.

Note that: parity of RANK(Π') = parity of Π', since in the Trotter-Johnson algorithm $[1, 2, \ldots, n]$ has rank 0, and each swap increases the rank by 1.

It is easy to compute the parity of a permutation in $\Theta(n^2)$:

$\textsc{PermParity}(n, \Pi) = |\{(i,j) : \Pi[i] > \Pi[j], 1 \leq i \leq j \leq n\}|.$

See the textbook for a $\Theta(n)$ algorithm.

```
PermTrotterJohnsonSuccessor(n, Π)
    s ← 0;
    for i ← 1 to n do   ρ[i] ← Π[i];
    done ← false;
    m ← n;
    while (m > 1) and (not done) do
        d ← 1;
        while (ρ[d] ≠ m) do  d ← d + 1;
        for i ← d to m − 1 do ρ[i] ← ρ[i + 1];
        par ← PermParity(m − 1, ρ);
        if (par = 1) then
          if (d = m) then m ← m − 1;
          else swap Π[s + d], Π[s + d + 1]
                done ← true;
        else
              if (d = 1) then m ← m − 1; s ← s + 1
              else swap Π[s + d], Π[s + d + 1]
                    done ← true;
    if (m = 1) then return [1, 2, . . . , n]
                else return Π;
```