Combinatorial Enumeration: Theory and Practice

Gordon Royle

Semester 1, 2004
Combinatorial Structures

**Combinatorics** is the study of finite sets of objects defined by certain specified properties – *combinatorial structures* – such as:

- **Subsets of a finite set**
  \[ \emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\} \]

- **Partitions of a number**
  \[ 4 = 1 + 1 + 1 + 1 = 1 + 1 + 2 = 1 + 3 = 2 + 2 \]

- **Words over a finite alphabet**
  \[ \text{aaa, aab, aba, abb, baa, bab, bba, bbb} \]
Graphs

Latin squares

1 2 3 4
2 1 4 3
3 4 1 2
4 3 2 1
Combinatorial Questions

For any particular combinatorial structure, a number of (related) questions can be asked:

- **Existence**
  Are there *any* combinatorial structures of this type?

- **Enumeration**
  *How many* combinatorial structures of this type are there?

- **Generation**
  *List* all the combinatorial structures of this type.
Existence

An *existence* question can be answered in different ways:

- Constructively, by giving
  - An *explicit example*, or
  - An *algorithm* to construct an example.

- Non-constructively, by giving an existence proof that does not yield an actual example.

A *constructive proof* is regarded as being “better” than a non-constructive one.
Enumeration

There are even more ways in which an enumeration question can be answered, including:

- **Exactly**
  A set of size $n$ has $\binom{n}{k}$ subsets of size $k$.

- **Approximately**
  The number $L(n)$ of Latin squares of order $n$ satisfies
  $$\log L(n) = n^2 \log n + O(n^2).$$

- **Implicitly**
  The $n$'th Fibonacci number $F_n$ is the coefficient of $x^n$ in the expansion of
  $$\frac{1}{1 - x - x^2}$$
  as a power series (about 0).
Enumeration cont.

- Bijectively
  The number of switching classes of graphs is equal to the number of Eulerian graphs.

- Computationally
  If all else fails, it may only be possible to produce a short list of the numbers of small combinatorial structures by direct computer generation. For example, the number of Steiner triple systems on \( n \) points for \( n = 1, \ldots, 19 \) is

\[
1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 2, 0, 80, 0, 0, 0, 11084874829.
\]
Generation

Algorithms for the generation of combinatorial structures are often called *combinatorial algorithms*. Such an algorithm should generate every combinatorial structure of a particular type – for example, we might want to generate all the subsets of a given set. Other considerations include:

- **Efficiency**
  The algorithm should be efficient in both space and time terms.

- **Ordering**
  The order in which the structures are output may be significant — a specific order may be required by the user, or a cleverly chosen order may reduce the amount of work required.
Ranking

Given a set $X$ of $n$ combinatorial structures, a ranking function is a bijection

$$r : X \rightarrow \{0, \ldots, n - 1\}$$

where $r(x)$ is the rank of the structure $x$.

A ranking function specifies an order from the first structure (the one with rank 0) to the last (the one with rank $n - 1$).

With this definition, counting starts at 0 which is usually the most convenient for computation. Mathematically, it is probably more natural to have a ranking function defined to be a function

$$r : X \rightarrow \{1, \ldots, n\}$$

and we will freely use this whenever appropriate.
Successors

If we have a ranking on a set $X$, then for any structure $x$ with $r(x) < n - 1$ the successor of $x$ is the structure $y$ such that

$$r(y) = r(x) + 1.$$

The predecessor of a structure $x$ is defined analogously provided $r(x) > 0$.

A generation algorithm can often be described simply by giving an explicit successor function (that is, a rule for computing the successor of $x$ directly from $x$). This is extremely useful because it means that the corresponding generation algorithm does not need to maintain large lists of structures.
Unranking

The inverse of a ranking function is called an unranking function. That is, a function

$$u : \{0, \ldots, n - 1\} \rightarrow X$$

such that $ru = e$ (where $e$ is the identity function).

Any ranking function implicitly defines an unranking function, but having an explicitly defined unranking function is extremely useful.
Sampling

For most types of combinatorial structure, there is an enormously rapid increase in their number as their size increases — the *combinatorial explosion*.

In this case, exhaustive generation is impossible, but we often wish to perform tests or collect statistical data by *sampling* the structures uniformly at random.

If we have an explicit unranking function $u$ then getting a sample structure from a set $X$ is very easy.

- Generate a (pseudo-)random number $i$ between 0 and $|X| - 1$
- Calculate $u(i)$
Subsets

The simplest of the classical combinatorial structures are the *subsets* of a set.

Recall the notation used for sets:

- \( A \subseteq B \)  \( A \) is a subset of \( B \).
- \( A \subset B \)  \( A \) is a proper subset of \( B \).
- \( \emptyset \)  The empty set.
- \( A \cap B \)  The intersection of \( A \) and \( B \).
- \( A \cup B \)  The union of \( A \) and \( B \).
- \( 2^A \)  The set of all subsets of \( A \) (the *power set* of \( A \)).
- \( \mathcal{P}(A) \)  Another common notation for \( 2^A \).
The powerset $2^{\{1,2,3,4\}}$

\[
\{1, 2, 3, 4\} \\
\{1, 2, 3\} \quad \{1, 2, 4\} \quad \{1, 3, 4\} \quad \{2, 3, 4\} \\
\{1, 2\} \quad \{1, 3\} \quad \{1, 4\} \quad \{2, 3\} \quad \{2, 4\} \quad \{3, 4\} \\
\{1\} \quad \{2\} \quad \{3\} \quad \{4\} \\
\emptyset
\]
Counting all subsets

The existence of subsets of a set is not in doubt, so we only need to consider enumeration and generation.

**Theorem**

If $A$ is a set of size $n$, then $2^A$ has size $2^n$.

**Proof.**

If $A = \{a_1, \ldots, a_n\}$ then a subset of $A$ is determined by specifying for each $i = 1, \ldots, n$ whether $a_i$ is included or excluded. These $n$ choices are independent and hence there are $2^n$ possibilities.
Representing subsets

Representing these $n$ choices with 0s (for excluded) and 1s (for included) yields a useful representation of a subset as a binary $n$-tuple.

For example, if $A = \{1, 2, 3, 4, 5\}$ then we can specify the subset $\{2, 3, 5\}$ as follows:

$$
\begin{array}{cccccc}
5 & 4 & 3 & 2 & 1 \\
5 & 4 & 3 & 2 & 1 \\
1 & 0 & 1 & 1 & 0
\end{array}
$$

If we view this binary $n$-tuple as the binary representation of an integer, then we obtain a very convenient ranking function $r$. For our example subset we have

$$
r(\{2, 3, 5\}) = 10110_2 = 16 + 4 + 2 = 22.
$$
Unranking

The corresponding unranking function merely involves computing the binary representation of an integer $i$. Each successive binary digit (from the least significant) is obtained by checking to see if $i$ is odd and then replacing $i$ by the quotient when $i$ is divided by 2. For example, if $i = 29$ and $n = 5$ then

<table>
<thead>
<tr>
<th>$i$</th>
<th>$i \mod 2$</th>
<th>$i/2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Reading the second column upwards we conclude that

\[ 29_{10} = 11101_2 \text{ corresponding to } \{1, 3, 4, 5\} \]
Ordering

The ordering on subsets determined by this representation is sometimes called *lexicographic ordering*. Unfortunately there are at least two other orderings on the set of all subsets that are also sometimes called lexicographic!

<table>
<thead>
<tr>
<th>( A )</th>
<th>n-tuple</th>
<th>( r(A) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>( {1} )</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>( {2} )</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>( {1, 2} )</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>( {3} )</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>( {1, 3} )</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>( {2, 3} )</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>( {1, 2, 3} )</td>
<td>111</td>
<td>7</td>
</tr>
</tbody>
</table>
The powerset \(2^{\{1,2,3,4\}}\)

\[
\begin{align*}
\{1, 2, 3, 4\} &\quad 15 \\
\{1, 2, 3\} &\quad 7 \\
\{1, 2, 4\} &\quad 11 \\
\{1, 3, 4\} &\quad 13 \\
\{2, 3, 4\} &\quad 14 \\
\{1, 2\} &\quad 3 \\
\{1, 3\} &\quad 5 \\
\{1, 4\} &\quad 9 \\
\{2, 3\} &\quad 6 \\
\{2, 4\} &\quad 10 \\
\{3, 4\} &\quad 12 \\
\{1\} &\quad 1 \\
\{2\} &\quad 2 \\
\{3\} &\quad 4 \\
\{4\} &\quad 8 \\
\emptyset &\quad 0
\end{align*}
\]
Minimal change algorithms

Under the lexicographic ordering the difference between successive subsets can be very large. For example,

\[ \text{succ}(\{1, 2, 3\}) = \{4\} \]

which involves altering the status of all 4 elements (that is, changing all 4 bits).

A *minimal change* algorithm is one where the successor function changes the current structure as little as possible — in the case of generating subsets, we would like to change just one bit.

A cyclic ordering of the $2^n$ binary $n$-tuples such that each differs from the previous one by a single bit is called a *Gray code*. 
Small Gray codes

We can easily find Gray codes for small values of \( n \):
For \( n = 1 \) we have
\[
0 \quad 1
\]
For \( n = 2 \) we have
\[
00 \quad 01 \quad 11 \quad 10
\]
For \( n = 3 \) we have
\[
000 \quad 001 \quad 011 \quad 010 \quad 110 \quad 111 \quad 101 \quad 100
\]
There is a pattern evident in this last Gray code:
\[
000 \quad 001 \quad 011 \quad 010 \quad 110 \quad 111 \quad 101 \quad 100
\]
Existence of Gray codes

The following theorem shows that Gray codes exist for all lengths — it is an example of a constructive proof.

**Theorem**

Let $G_n = (g_0, g_1, \ldots, g_m)$ be an $n$-bit Gray code (so $m = 2^n - 1$). Then the code

$$(0g_0, 0g_1, \ldots, 0g_m, 1g_m, \ldots, 1g_1, 1g_0)$$

obtained by listing the words of $G_n$ each preceded by 0, and then the words of $G_n$ in reverse order each preceded by 1 is an $(n + 1)$—bit Gray code.
Proof.
There is a 1-bit change between each of the first \(2^n\) words and each of the last \(2^n\) words, and so we merely need to check the halfway point, where the successor of \(0g_m\) is \(1g_m\) and that the final word \(1g_0\) is one bit different from the first word \(0g_0\), both of which are clear.

Together with the existence of the Gray code \(G_1\), this provides an inductive proof that Gray codes of all lengths exist.
Gray codes are intimately related to a series of graphs known as the $k$-cubes $Q_k$. The vertices of $Q_k$ are the $2^k$ binary $k$-tuples, where two $k$-tuples are adjacent if they differ in exactly one coordinate position.
The cube $Q_3$
A Gray code is a Hamilton cycle
Enumeration of Gray codes

An important resource for checking the status of enumeration problems is the *Online Encyclopedia of Integer Sequences* located at

http://www.research.att.com/~njas/sequences

This is an online searchable database of sequences of integers that can be used to determine if a given type of structure has been enumerated, or if a given sequence corresponds to a known formula or type of structure.

If we search for “Gray code” we discover that the numbers are known only for $n = 1, \ldots, 5$:

1, 1, 6, 1344, 906545760
The On-Line Encyclopedia of Integer Sequences

Enter a ○ sequence, ○ word, or ○ sequence number:

gray code

Clear | Hints | Advanced look-up

Other languages: Albanian Arabic Bulgarian Catalán Chinese (simplified, traditional) Croatian Czech Danish Dutch Esperanto Finnish French German Greek Hebrew Hindi Hungarian Italian Japanese Korean Polish Portuguese Romanian Russian Serbian Spanish Swedish Thai Turkish

For information about the Encyclopedia see the Welcome page.

Lookup | Welcome | Français | Demes | Index | Browse | More | WebCam
Contribute new seq. or comment | Format | Transforms | Puzzles | Hot | Classics
More pages | Superseeker | Maintained by N. J. A. Sloane (njas@research.att.com)

[Last modified Tue Feb 17 18:05:31 EST 2004. Contains 91600 sequences.]
Exercises

1. Write down the 4-bit reflected Gray code.
2. Write down the sequence corresponding to which bit is negated at each step in the reflected Gray code.
3. Write a program (in any language) that produces the \( n \)-bit reflected Gray code.
4. Find the ranking function for the reflected Gray code.
5. Find the unranking function for the reflected Gray code.
6. Find all the 3-bit Gray codes.
The mid-levels conjecture

Graph theory and combinatorics abounds with simple-to-state unsolved problems. One of these is the mid-levels conjecture:

**Conjecture (Mid-Levels Conjecture)**
The subsets of size $n$ and $n + 1$ of a set of size $2n + 1$ can be arranged in a cyclic order in such a way that successive elements differ only by a single element.

This is asking for the existence of a hamilton cycle through the subgraph of the $(2n + 1)$-cube induced by the subsets of size $n$ and $n + 1$. If we draw the $(2n + 1)$-cube with all the subsets of a given size on one horizontal line, then these form the two middle levels.