Lecture 3:
Combinatorial Generation

Steven Skiena

Department of Computer Science
State University of New York
Stony Brook, NY 11794–4400

http://www.cs.stonybrook.edu/~skiena

Contest Results

Winner: Overflowed (10 problems, 1552 minutes)

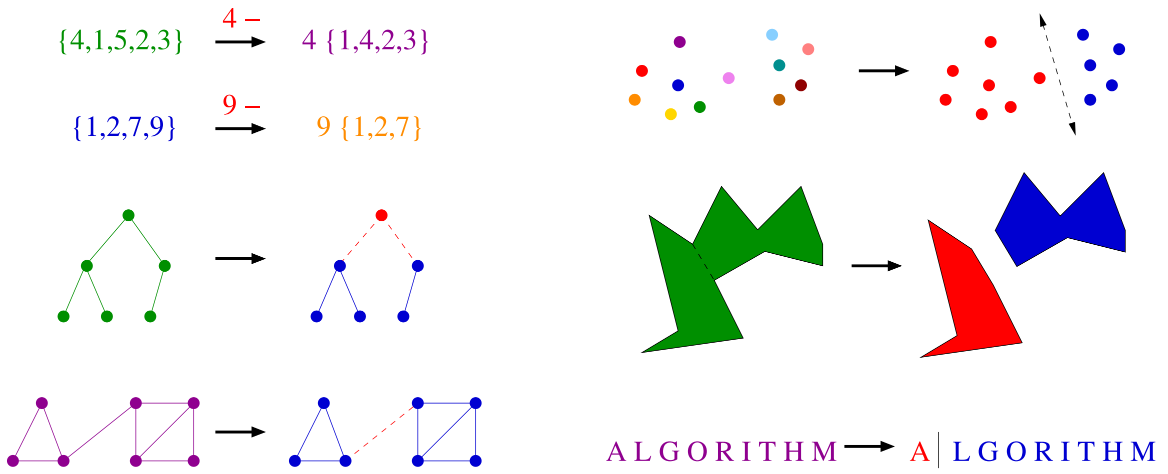
Problems

#	Name				
A	Words from cubes	standard input/output 0.25 s, 64 MB		x41	
B	Delivery Bears	standard input/output 2 s, 256 MB		x33	
C	Olympiad in Programming and Sports	standard input/output 2 s, 512 MB		x35	
D	Maximize Mex	standard input/output 2 s, 256 MB		x25	
E	Stock Exchange	standard input/output 6 s, 16 MB		x2	
F	Bits of merry old England[1]	standard input/output 2 s, 256 MB		x15	
G	Armchairs	standard input/output 2 s, 512 MB		x30	
H	Buying Sets[1]	standard input/output 2 s, 256 MB		x5	
I	New Year and Forgotten Tree	standard input/output 7 s, 256 MB		x2	
J	Anti-Palindromize	standard input/output 2 s, 256 MB		x31	

Topic: Combinatorial Objects

- Combinatorial Objects
- Ranking and Unranking
- Subsets
- Permutations
- Integer Partitions
- Trees and Graphs

Recursive Decompositions of Combinatorial Objects



Classical Combinatorial Objects

- Permutations and Strings
- Subsets and k-Subsets
- Set Partitions, Integer Partitions and Young Tableaux
- Trees and Graphs

Properties of Combinatorial Objects

- There are a discrete number of them for any given size, so they can be counted.
- The number of distinct objects typically grow exponentially with size.
- They can typicially be generated by backtracking, but …
- There are more interesting ways to work with them.

Generation by Backtracking

```
void backtrack(int a[], int k, data input) {
    int c[MAXCANDIDATES];       /* candidates for next position */
    int nc;                     /* next position candidate count */
    int i;                      /* counter */

    if (is_a_solution(a, k, input)) {
        process_solution(a, k,input);
    } else {
        k = k + 1;
        construct_candidates(a, k, input, c, &nc);
        for (i = 0; i < nc; i++) {
            a[k] = c[i];
            make_move(a, k, input);
            backtrack(a, k, input);
            unmake_move(a, k, input);

            if (finished) {
                return;         /* terminate early */
            }
        }
    }
}
```

Questions?

Topic: Ranking and Unranking

- Combinatorial Objects
- Ranking and Unranking
- Subsets
- Permutations
- Integer Partitions
- Trees and Graphs

Operations on Combinatorial Objects

- Count(n) – how many objects are there of size n?
- Rank(x,n) – What number or position is object x in an ordering of all objects of size n?
- Unrank(i,n) – Construct the ith object in the ordering of all objects of size n.
- Next(x) – Return the object appearing directly after x in the ordering of all objects of size n.
- RandomGen(n) – Return an object selected uniformly at random from all objects of size n.

Permutations Example

{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}

- Count[3] = 6
- Rank[{1, 3, 2}, 3] = 1
- Unrank[3, 3] = {2, 3, 1}
- Next[{2, 3, 1}] = {3, 1, 2}
- Previous[{1, 2, 3}] = {3, 2, 1}
- RandomGen[3] = {1, 3, 2}

Everything Follows from Count, Rank, and Unrank

- x = Unrank(Rank(x,n), n)
- Next(x) $\rightarrow$ Unrank(Rank(x,n) + 1, n)
- Previous(x) $\rightarrow$ Unrank(Rank(x,n) + 1, n)
- RandomGen(n) $\rightarrow$ Unrank(RandInt(0:Count(n)-1), n)

Better is to do Next/Previous mod $Count(n)$ to get a cyclic order.

Further, Rank and Unrank follow from Count

Natural Generation Orders

- Lexicographic or sorted order:

  permutations: 123, 132, 213, 231, 312, 321

  subsets: {} {1} {12} {123} {13} {2} {23} {3}

- Minimum change order: (one swap or insertion/deletion)

  permutations: 123, 321, 231, 132, 312, 213

  subsets: {} {1} {12} {2} {23} {123} {13} {3}

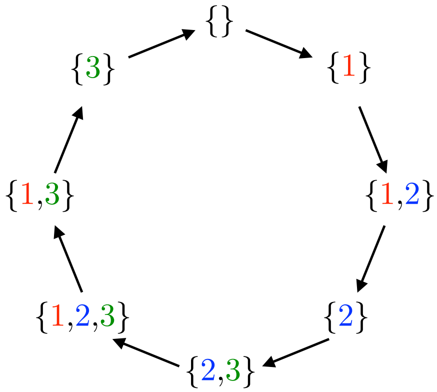
Rank and Unrank depend upon which generation order is used.

Questions?

Topic: Subsets

- Combinatorial Objects
- Ranking and Unranking
- Subsets
- Permutations
- Integer Partitions
- Trees and Graphs

Counting Subsets



The recursive formula is:

$$Count(n) = 2 \times Count(n-1)$$

$$Count(0) = 1$$

Binary Counting Representation

Thus $Count(n) = 2^n$, the same as the number of strings of boolean true/false or the number of bit patterns of length n.
Need arbitrary precision arithmetic to count for large n.
The bijection between length-n binary strings and the set of integers $\{0, 1, \ldots, 2^n - 1\}$ given by

$$b_{n-1}b_{n-2}\ldots b_2 b_1 b_0 \iff \sum_{i=0}^{n-1} 2^i b_i,$$

is well-known because it is the standard way of representing integers in computers.

Ranking Subsets

Lexicographic order is hard to rank/generate, so use binary counting:

```
 0     1     2     3     4     5     6     7
000,  001,  010,  011,  100,  101,  110,  111
```

- if (Head(x)==0) Rank(x,n) = Rank(Rest(x),n-1)
- if (Head(x)==1) Rank(x,n) = 2^{n-1} + Rank(Rest(x),n-1)
- Rank(1,3,5,6) = $2^5 + 2^3 + 2^1 = 42$

Unranking Subsets

If $i > 2^{n-1}$, the first bit is zero and item 1 is not in the subset.
SubsetUnrank[i,n] = Unrank[i,1,n]

- If $(i \geq Count(n-1))$

  $$Unrank(i,j,n) = \{j\} \cup Unrank(i - 2^{n-1}, j+1, n-1)$$

- If $(i < Count(n-1))$

  $$Unrank(i,j,n) = Unrank(i,j,n-1)$$

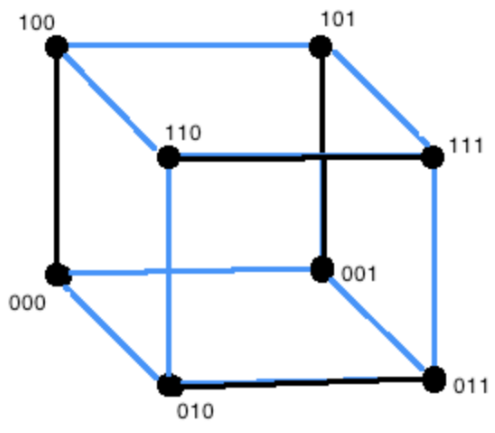
Unrank(6,1,3) = $\{1\} \cup Unrank(2,2,2) = \{1,2\} \cup Unrank(0,3,1) = \{1,2\}$

Gray Codes

Gray codes are minimum change orderings for subsets of n items.
The neighbor of each subset is constructed by adding or deleting a single element.
A Gray code for $n = 4$ is: {}, {4}, {3, 4}, {3}, {2, 3}, {2, 3, 4}, {2, 4}, {2}, {1, 2}, {1, 2, 4}, {1, 2, 3, 4}, {1, 2, 3}, {1, 3}, {1, 3, 4}, {1, 4}, {1}

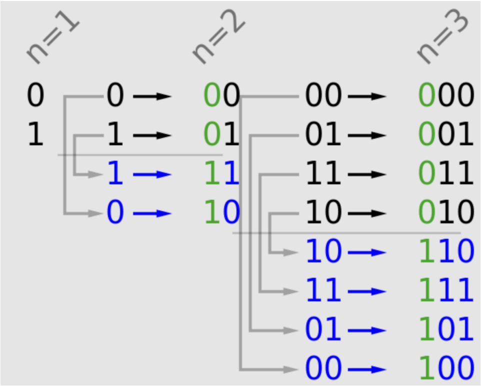
Gray Codes and Hamiltonian Cycles

Each Gray code ordering corresponds to a Hamiltonian cycle on the minimum change graph for subsets, which is a hypercube.



Binary Reflected Gray Codes

There is a nice recursive construction. Build a Gray code of size $n-1$, concatenate it to its reverse, and add n to each member of the reversed copy.



There are ranking and unranking methods for Gray codes, but binary counting is easier in our counting-based approach.

Strings

There are α^n strings of length n built from an alphabet of size α, because there are α choices for each position.

For $n = \alpha = 3$: 000 001 002 010 011 012 020 021 022 100 101 102 110 111 112 120 121 122 200 201 202 210 211 212 220 221 222

To rank string S,

$$Rank[S, n] = S[1] * Count[n-1, \alpha] + Rank[Rest[S], n-1]$$

Unranking is finding the first character by $\lfloor i/Count[n-1] \rfloor$ and then recurring.

De Bruijn Sequences

The shortest circular string which contains all strings of length k exactly once.

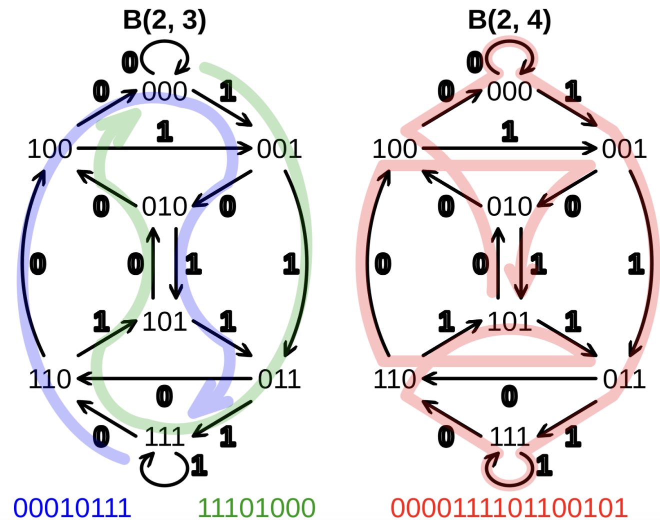
00010111 = 000, 001, 010, 101, 011, 111, 110, 100

11101000 = 111, 110, 101, 010, 100, 000, 001, 011

These are “safecracker” sequences, the most efficient way to try all possible combinations.

Eulerian Cycles and De Bruijn Sequences

Construct a directed graph where each vertex represents a $(k-1)$-mer, and each directed edge is tables with a symbol such that each edge represents a k-mer.



A Hamiltonian cycle on this graph defines a de Bruijn sequence.

But even better, an Eulerian cycle on this graph defines a longer de Bruijn sequence.

Note indegree = outdegree.

K-Subsets

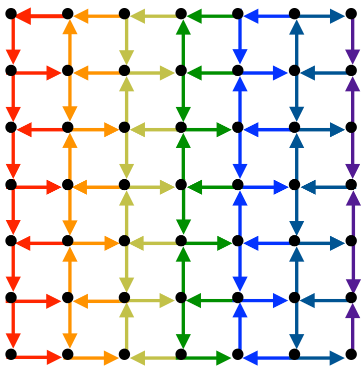
A *k-subset* is a subset with exactly k elements in it.
For $n = 5$, $k = 3$: $\{123\}$ $\{124\}$ $\{125\}$ $\{134\}$ $\{135\}$ $\{145\}$ $\{234\}$ $\{235\}$ $\{245\}$ $\{345\}$
A simple recursive construction starts from the observation that each k-subset on n elements either contains the first element of the set or it does not.
Thus the number of k-subsets of $1 \ldots n$ is:

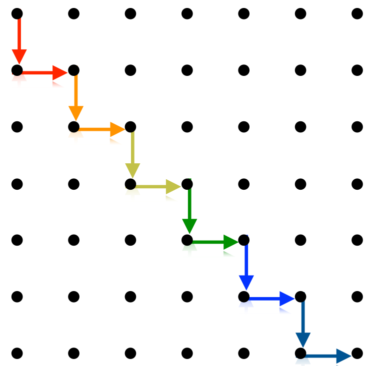
$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}.$$

Prepending the first element to each $(k-1)$-subset of the other $n-1$ elements gives the former, and building all the k-subsets of the other $n-1$ elements gives the latter.

The first element appears in Unrank[i,n] iff $i < \binom{n-1}{k-1}$

Grid Paths and k-Subsets





Any shortest path across an $n + 1 \times m + 1$ grid consists of n down hops and m right hops.
Each such path is defined by picking the positions of the n down hops as an n-subset of $[1, \ldots, n + m]$.

Questions?

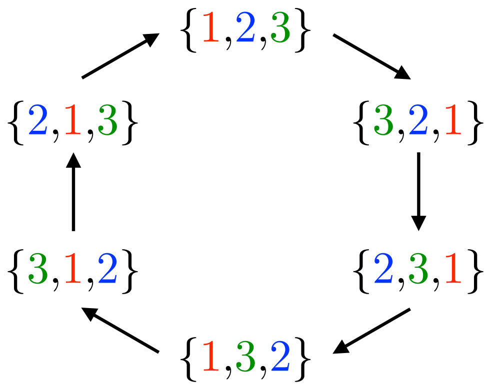
Topic: Permutations

- Combinatorial Objects
- Ranking and Unranking
- Subsets
- Permutations
- Integer Partitions
- Trees and Graphs

Permutations

A permutation is an ordering or arrangment of $1, \ldots, n$.

$\{1234\}$, $\{1243\}$, $\{1324\}$, $\{1342\}$, $\{1423\}$, $\{1432\}$, $\{2134\}$, $\{2143\}$, $\{2314\}$, $\{2341\}$, $\{2413\}$, $\{2431\}$, $\{3124\}$, $\{3142\}$, $\{3214\}$, $\{3241\}$, $\{3412\}$, $\{3421\}$, $\{4123\}$, $\{4132\}$, $\{4213\}$, $\{4231\}$, $\{4312\}$, $\{4321\}$



Counting Permutations

The first element of permutation p can be anything from $1, \ldots, n$, and then recur with any arrangement of the other $n-1$ elements:

$$Count[n] = n \times Count[n-1] = n!$$

$$Count[1] = 1$$

Ranking Permutations

For permutations in lexicographic order, the rank of permutation p is

$$Rank[p, n] = (p[1] - 1) \times Rank[Rest[p], n - 1]$$

Note that the permutation must be renormalized after removing the head:
Rest[3,1,4,2,5] = [1,3,2,4]

Unranking Permutations

The first element of $Unrank[i, n]$ is given by $\lceil i + 1/Count[n-1] \rceil$, then recur — but adjust for missing elements

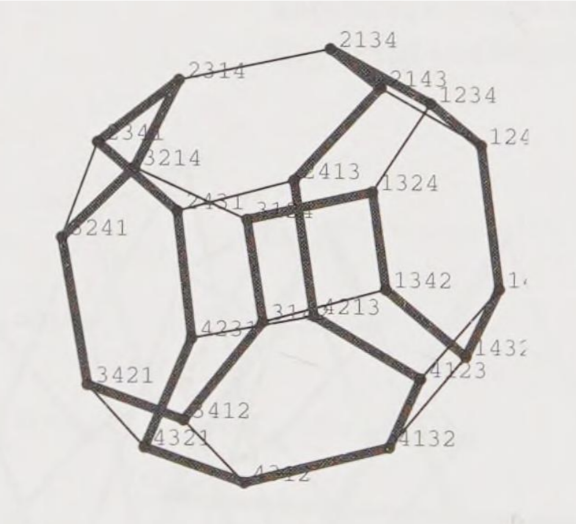
$$Unrank[14, 4] \to [3] + Unrank[14 - 2 \times (3!), 3] \to$$

$$[2] + Unrank[0, 2] \to [1, 4]$$

The reason it is [1,4] instead of [1,2] is that 2 and 3 have been used so far.

Minimum Change Ordering

The minimum possible change between permutations is a swap of a pair of elements e.g. 3,1,4,2,5 and 3,1,4,2,5. Minimum change or maximum change orders can be found through Hamiltonian cycles on the appropriate graph.



Special permutation generation algorithms (e.g. Johnson-Trotter) can generate permutations which differ in one neighboring transposition.

$$\{1234\}\{2134\}\{3124\}\{1324\}\{2314\}\{3214\}$$

$$\{4213\}\{2413\}\{1423\}\{4123\}\{2143\}\{1243\}$$

$$\{1342\}\{3142\}\{4132\}\{1432\}\{3412\}\{4312\}$$

$$\{4321\}\{3421\}\{2431\}\{4231\}\{3241\}\{2341\}$$

Derangements

Derangements are permutations where there is no element i in position i.

$\{4213\}\{2413\}\{4123\}\{2143\}\{3142\}\{3412\}\{4312\}\{4321\}\{3421$

In any derangement, either n swaps position with i (leaving the remaining $n-2$ items to be deranged) or n is in position i such that i cannot be in position n (leaving $n-1$ elements to be deranged). So:

$$D[n] = (n-1)D[n-2] + D[n-1]$$

Questions?

Topic: Integer Partitions

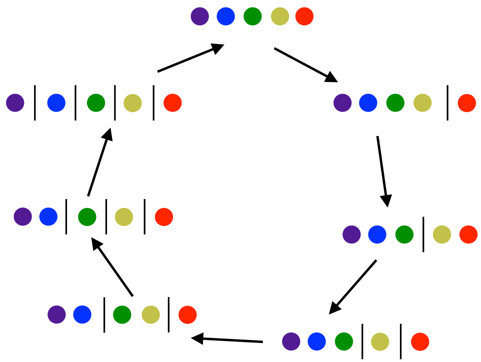
- Combinatorial Objects
- Ranking and Unranking
- Subsets
- Permutations
- Integer Partitions
- Trees and Graphs

Integer Partitions

An *integer partition* (in short, *partition*) of a positive integer n is a set of strictly positive integers which sum up to n.

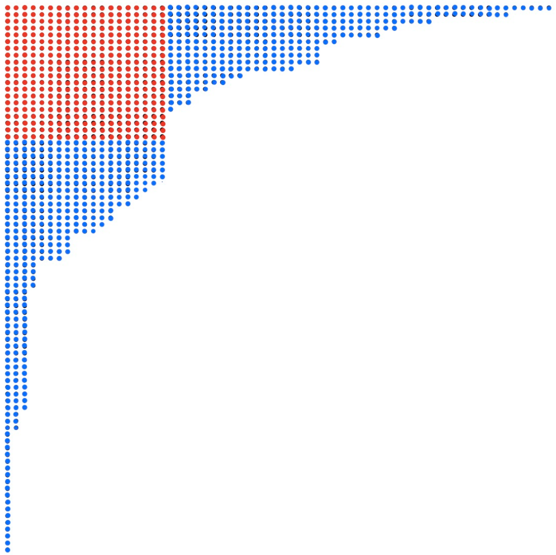
By convention, partitions are listed in non-increasing order.

$\{\{6\}, \{5, 1\}, \{4, 2\}, \{4, 1, 1\}, \{3, 3\}, \{3, 2, 1\}, \{3, 1, 1, 1\}, \{2, 2, 2\}, \{2, 2, 1, 1\}, \{2, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1\}\}$



Ferrer's Diagrams of Integer Partitions

My citations on Google Scholar can be represented by an integer partition, with my H-index being the red square (at least h papers each with at least h citations).



Counting Integer Partitions

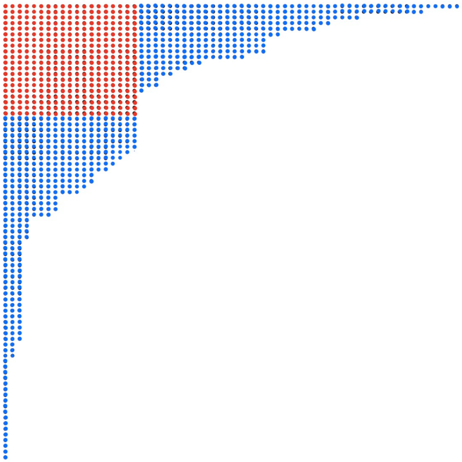
Counting partitions is best done by solving a more general problem, where Count[n,k] gives the number of n with a largest part of at most k.
The largest part of any such partition is either k or $< k$, so:

$$p_{n,k} = p_{n-k,k} + p_{n,k-1}, \text{ for } n \geq k > 0.$$

Letting $p_{n,0} = 0$ for all $n > 0$ and $p_{0,k} = 1$ for all $k \geq 0$, we get a recurrence relation for $p_{n,k}$.
The total number of partitions of n, p_n, is equal to $p_{n,n}$, and so this recurrence can be used to compute p_n as well.

Bijections on Integer Partitions

Flipping the dots across the main diagonal proves that $CountKParts(n, k) = CountMaxPart(n, k)$.



Ranking and Unranking Integer Partitions

The number of partitions with largest part exactly k is Count[n,k]-Count[n,k-1]

Lexicographic order sorts the partitions based on the size of the largest part, so for a given integer partition p, we can find the rank of the first partition with $k = p[1]$ and recur.

Unranking naturally inverts this procedure.

Questions?

Topic: Set Partitions

- Combinatorial Objects
- Ranking and Unranking
- Subsets
- Permutations
- Integer Partitions
- *Set Partitions*
- Trees and Graphs

Set Partitions

A *set partition* is a partition of a set into disjoint subsets.

$[\{1, 2, 3, 4\}]$,
$[\{1\}, \{2, 3, 4\}]$, $[\{1, 2\}, \{3, 4\}]$, $[\{1, 3, 4\}, \{2\}]$, $[\{1, 2, 3\}, \{4\}]$, $[\{1, 4\}, \{2, 3\}]$, $[\{1, 2, 4\}, \{3\}]$, $[\{1, 3\}, \{2, 4\}]$,
$[\{1\}, \{2\}, \{3, 4\}]$, $[\{1\}, \{2, 3\}, \{4\}]$, $[\{1\}, \{2, 4\}, \{3\}]$, $[\{1, 2\}, \{3\}, \{4\}]$, $[\{1, 3\}, \{2\}, \{4\}]$, $[\{1, 4\}, \{2\}, \{3\}]$,
$[\{1\}, \{2\}, \{3\}, \{4\}]$

Assuming a total order on a set X, a canonical way of writing a set partition of X is this: write each subset in increasing order and write the subsets themselves in increasing order of their minimum elements.

Set Partitions in Action

The vertex coloring of a graph is a set partition: each part is the subset of vertices of a given color.

A clustering is a set partition: the items in one cluster appear as one part in the partition

A set packing is a set partition: each item belongs to exactly one set in the packing.

Counting Set Partitions

The number of set partitions of $\{1, 2, \ldots, n\}$ having k blocks is a fundamental combinatorial number called the *Stirling number of the second kind*.

We use $\left\{ {n \atop k} \right\}$ to denote the number of set partitions of $\{1, 2, \ldots, n\}$ having k blocks.

The largest element n is either it is own part or at the end of one of k existing parts, so:

$$\left\{ {n \atop k} \right\} = \left\{ {n-1 \atop k-1} \right\} + k \left\{ {n-1 \atop k} \right\}$$

The Bell Numbers

The total number of set partitions of $\{1, 2, \ldots, n\}$ is the nth *Bell number*, denoted B_n, another fundamental combinatorial number.

Clearly, $B_n = \Sigma_{k=1}^{n} \left\{ {n \atop k} \right\}$

However, the identity

$$B_n = \sum_{k=0}^{n-1} \binom{n-1}{k} B_{n-(k+1)}$$

provides an alternate way.

The first part in a set partition contains 1 and k other elements, each choice of which which leaves $n-(k+1)$ items to partition in the other parts.

Summed over all possible k and simplified using the symmetry of binomial numbers we get:

$$B_n = \sum_{k=0}^{n-1} \binom{n-1}{k} B_{n-(k+1)} = \sum_{k=0}^{n-1} \binom{n-1}{k} B_k.$$

Ranking and Unranking Set Partitions

Lexicographic ordering is by the number of parts.

So $i < \Sigma_{k=1}^{x} \left\{ {n \atop k} \right\}$ with the smallest x tells us the number of parts.

The Stirling number recurrence then tells us whether n is in its own part, or if not what part it is in.

Then recur to unrank the remaining elements.

Young Tableaux

A *Young tableau of shape* $(n_1, n_2, \ldots, n_m)$ where $n_1 \geq n_2 \geq \cdots \geq n_m > 0$ is an arrangement of $n_1 + n_2 + \cdots + n_m$ distinct integers in an array of m rows with n_i elements in row i such that each row and in each column are in increasing order.

```
In[101]:= TableForm[FirstLexicographicTableau[{4,3,3,2}]]

Out[101]//TableForm= 1    5    9    12

                     2    6    10

                     3    7    11

                     4    8

In[102]:= TableForm[ LastLexicographicTableau[{4,3,3,2}] ]

Out[102]//TableForm= 1    2    3    4

                     5    6    7

                     8    9    10

                     11   12
```

Sequencing Young Tableaux

Young tableau are set partitions with the shape of integer partitions, with rows and columns that are ordered:

{{1, 2, 3, 4}},
{{1, 3, 4}, {2}},
{{1, 2, 4}, {3}},
{{1, 2, 3}, {4}},
{{1, 3}, {2, 4}},
{{1, 2}, {3, 4}},
{{1, 4}, {2}, {3}},
{{1, 3}, {2}, {4}},
{{1, 2}, {3}, {4}},
{{1}, {2}, {3}, {4}}

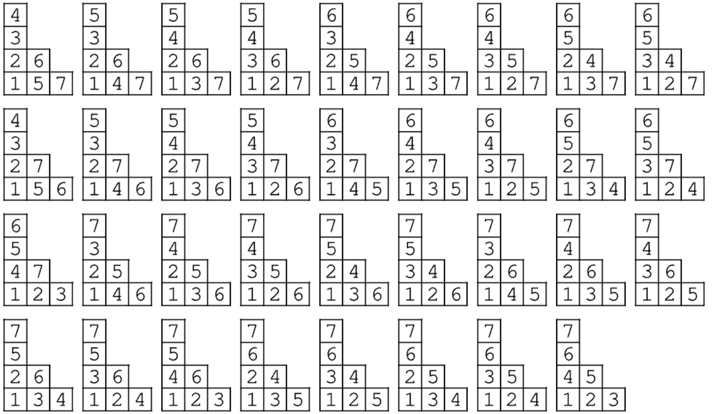
Counting Young Tableaux

Each position p within a Young tableau defines an L-shaped *hook*, consisting of p, all the elements below p, and all the elements to the right of p.

The *hook length formula* gives the number of tableaux of a given shape as $n!$ divided by the product of the hook length of each position, where n is the number of positions in the tableau.

A convincing argument that the formula works is, of the $n!$ ways to label a tableau of given shape, only those where the minimum element in each hook is in the corner

Hook Length Example



The hook length formula tells us there are $7!/(6 \times 3 \times 4 \times 2) = 35$ tableaux of this shape.

Parenthesizations

A well-formed formula is a legal sequence of n sets of parentheses.
For $n = 3$ there are five parenthesizations: ()()(), ()(()), (())(), (()()), ((())).
How many parenthesizations of n sets of parenthesis?

Catalan Numbers

Since any balanced set of parentheses has a leftmost point $k+1$ at which the number of left and right parentheses are equal, peeling off the first left parenthesis and the $k+1$th right parenthesis leaves two balanced sets k and $n-1-k$ parentheses, which leads to the following recurrence:

$$C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k} = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$$

The *Catalan numbers* also count the number of triangulations of a convex polygon and the number of paths across a lattice which do not rise above the main diagonal.

Counting Parenthesizations as Young Tableaux

The number of parenthesizations is equal to the number of $\{n, n\}$/ Young tableaux.
When filled with the numbers from 1 to $2n$, the top row gives the positions of of the left paren and the bottom row the positions of the right parens.
Thus the ith (must appear before the ith).

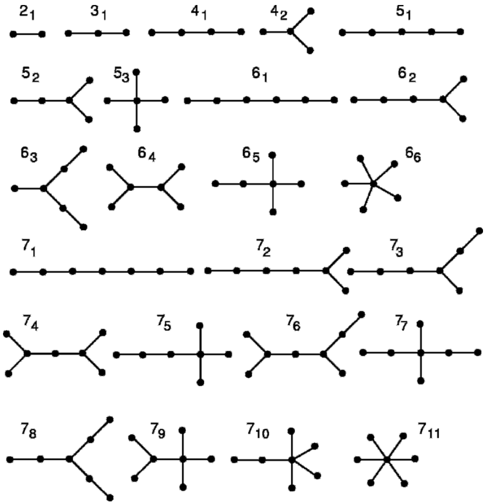
Questions?

Topic: Trees and Graphs

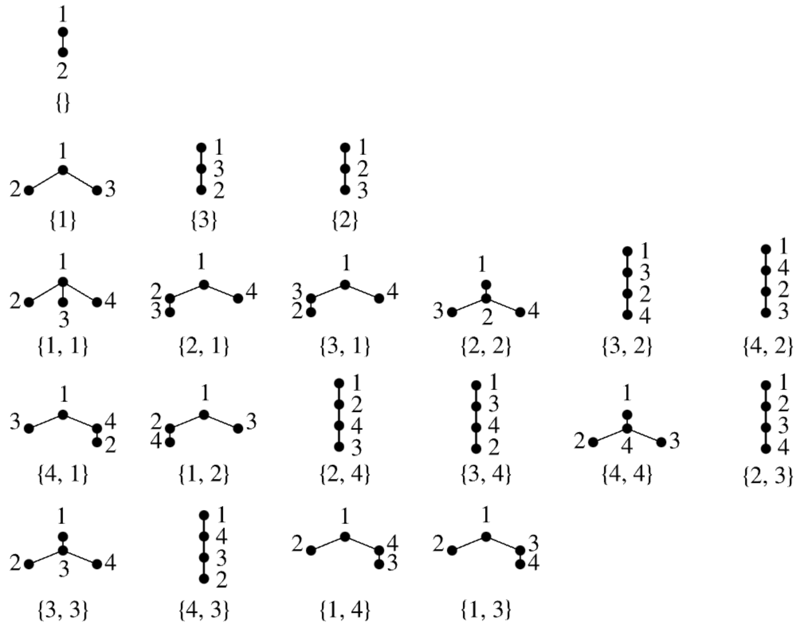
- Combinatorial Objects
- Ranking and Unranking
- Subsets
- Permutations
- Integer Partitions
- Set Partitions
- Trees and Graphs

Counting Unlabeled Trees/Graphs is Hard

Testing whether two unlabeled graphs are the same (isomorphic) is challenging, which implies there is no easy way to get an exact count of them.



Listing Labeled Trees



Counting Labeled Trees

That there are n^{n-2} distinct labeled trees on n vertices is shown by Prüfer codes, a bijection between such trees and strings of $n-2$ integers between 1 and n.

The key to Prüfer's bijection is the observation that for any tree there are always at least two vertices of degree one.

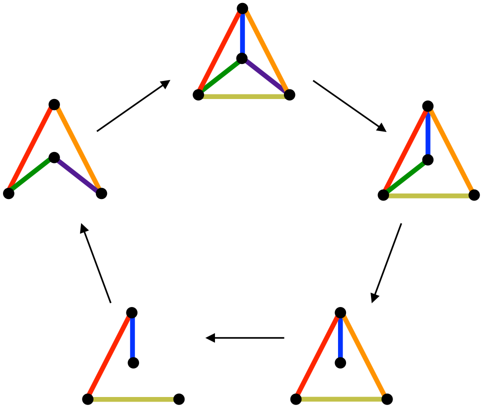
Start with an n-vertex tree T, whose vertices are labeled 1 through n. Let u be the leaf with smallest label and let v be the neighbor of u. Note that u and v are uniquely defined. We now let v be the first symbol in our string, or Prüfer code.

After deleting vertex u we have a tree on $n-1$ vertices, and repeating this operation until only one edge is left gives us $n-2$ integers between 1 and n.

Ranking/Unranking Labeled Trees

The Prüfer codes imply we can rank and unrank labeled trees exactly how we rank $n-2$ length strings on an alphabet of size n.

Labeled vs. Unlabeled Graphs



Dealing with unlabeled graphs gets into challenging problems of graph isomorphism: are two graphs the same?

Counting Labeled Graphs

Every *simple* undirected labeled graph on n nodes and m edges represents a selection of m edges from the $\binom{n}{2} = \frac{n(n+1)}{2}$ possible edges.
Thus these can be ranked/unranked like k-subsets, or just subsets if m is not given.

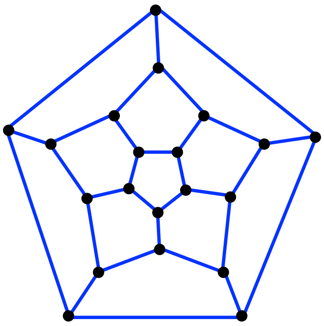
Questions?

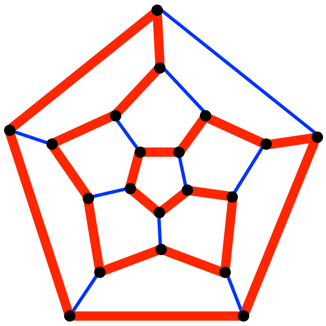
Topic: Conclusion

You Can Count On This

For any type of combinatorial object you can count, you can rank/unrank, next/previous, or randomly select.

Even if you can't count them, if you can build a graph of related objects, you can sequence them by finding a Hamiltonian path





For Further Reading

- Donald Knuth, *Combinatorial Algorithms, Part I*, Volume 4a of the *Art of Computer Programming*, 2011.
- Frank Ruskey, *Combinatorial Generation*, Working Draft, `https://page.math.tu-berlin.de/~felsner/SemWS17-18/Ruskey-Comb-Gen.pdf`, 2003
- Pemmaraju and Skiena, *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, 2003
- The Online Encyclopedia of Integer Sequences, https://oeis.org/

Questions?