Enumerative Combinatoric Algorithms

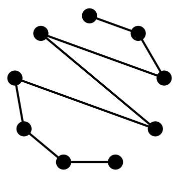
Contents

1 Enumerating vs. Counting

1.1 Example 1: Spanning Paths

Given: n points in convex position

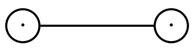
Question: How many straight line planar (crossing-free) spanning paths (all points are used) are there?



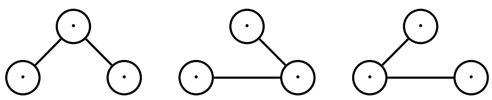
- $n = 1$

  □

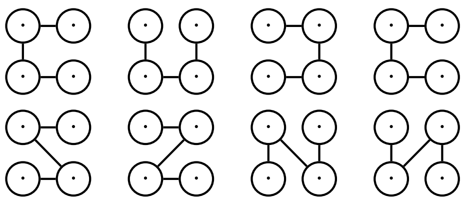
- $n = 2$

- $n = 3$: 4 paths

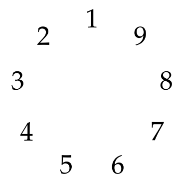
  

- $n = 4$: 8 paths

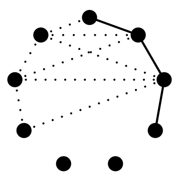
  

- $n = 5$:
  - more possibilites $\rightarrow$ we go towards counting rather than enumerating
  - look at possible types of paths, in CW and CCW direction for each of the five starting points. Divide by 2 for identical paths obtained with different starting points (since paths have 2 ends)

Counting



- # of paths starting at $\circ = \#P_\circ$
- $\#P_n = \frac{n \cdot \#p_\circ}{2} = n \cdot \frac{2^{n-1}}{2} = n \cdot 2^{n-2}$



- set of not yet used points must not be split
- continue: always two options, 1 for the last

1.2 Example 2: Blokus

Definition

- consists of unit squares connected via edges (not just connected through vertices)
- n-polyomino has n unit squares
- Animal: does not contain holes
- free: rotation/mirroring is not taken into account
- fixed: rotated/mirrored polyominoes are regarded as being distinct

Question: How many n-polyominoes are there?

- $n = 1$:

  $\square$

  1

- $n = 2$:
  - $\square\square$
  - $\begin{array}{c}\square\\\square\end{array}$

fixed: 2, free: 1

- $n = 3$:
  - $\begin{array}{ccc}\square & \square & \square\end{array}$
  - $\begin{array}{c}\square \\ \square \\ \square\end{array}$
  - $\begin{array}{cc}\square & \square \\ \square & \end{array}$ $\times 4$

  fixed: 6, free: 2

- $n = 4$:
  - $\begin{array}{cccc}\square & \square & \square & \square\end{array}$ $\times 2$
  - $\begin{array}{cc}\square & \\ \square & \\ \square & \square\end{array}$ $\times 8$
  - $\begin{array}{ccc}\square & \square & \square \\ & \square & \end{array}$ $\times 4$ (reflection has same effect as rotation)
  - $\begin{array}{cc} & \square \\ \square & \square \\ \square & \end{array}$ $\times 4$
  - $\begin{array}{cc}\square & \square \\ \square & \square\end{array}$ $\times 1$

  fixed: 19, free: 5

Enumerate/Construct all polyominoes

- construct the $(n+1)$-ominoes from the n-ominoes. This works because we can show that from each $(n+1)$-ominoes you can remove a square and get a valid n-omino (idea: construct an arbitrary spanning tree, remove leaf)
- find and remove duplicates
- how many $(n+1)$-ominoes are generated?

maximum number: $2n+2$ or $4n - 2|E|$
Why?
$\square$: 4 possibilities to add a square (at each edge)

$$\begin{array}{cccc} & & \square & \\ \square & \square & \square & \square \\ & \square & & \end{array}$$

$4n - 2|E| \to 4n - 2(n-1) = 2(n+2)$
How to find duplicates? compare every pair $\to O(k^2 n)$

1.2.1 Fingerprints

- maps an object (here: polyomino) to a *unique* representation, independent of translation, rotation, reflection
- sort the fingerprints (insertion sort)
- binary search

One possibility for polyominoes: use vector representations on a coordinate system

Examples:

- $\begin{array}{cccc} 3 & & & \square \\ 2 & & & \square \\ 1 & \square & \square & \square \\ & 1 & 2 & 3 \end{array}$ has vector representation $< (1,1),(2,1),(3,1),(3,2),(3,3) >$

  $\begin{array}{cccc} 3 & \square & & \\ 2 & \square & & \\ 1 & \square & \square & \square \\ & 1 & 2 & 3 \end{array}$ has vector representation $< (1,1),(2,1),(3,1),(1,2),(1,3) >$

  This is lexicographically smaller than the first one

- $\begin{array}{cccc} 3 & \square & & \\ 2 & \square & \square & \\ 1 & & \square & \square \\ & 1 & 2 & 3 \end{array} \quad \begin{array}{cccc} 3 & & \square & \square \\ 2 & \square & \square & \\ 1 & \square & & \\ & 1 & 2 & 3 \end{array} \quad < (1,1),(1,2),(2,2),(2,3),(3,3) >$

- $\begin{array}{cccc} 3 & & & \\ 2 & \square & \square & \square \\ 1 & \square & & \square \\ & 1 & 2 & 3 \end{array} \quad \begin{array}{cccc} 3 & \square & \square & \\ 2 & \square & & \\ 1 & \square & \square & \\ & 1 & 2 & 3 \end{array} \quad < (1,1),(3,1),(1,2),(1,3),(2,3) >$

Generally:

1. generate all 8 transformations (4 rotations, 2 reflections)
2. take the lexicographically smallest one $\rightarrow O(1) \cdot O(n)$ for an n-omino

$f'(n)$ fingerprints generated in total $\rightarrow f'(n)\,(O(n) + \log(f'(n) \cdot O(n))$

$\lim\limits_{n\to\infty} \frac{f(n+1)}{f(n)} = \lambda$, $3.98 < \lambda < 4.65$

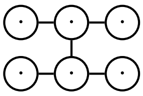
$f(n)$ is known up to $n = 56$.

1.3 Example Spanning Trees on Ladders

with n rungs

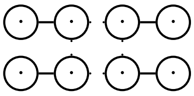
Question: Number of spanning trees on a ladder with n rungs

- $n = 1$

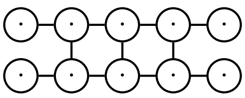
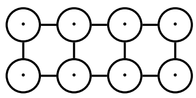
  # = 1

- $n = 2$

  # = 4

- $n = 3$

   $\Leftrightarrow$ 

  # = 15

Answer: Counting spanning trees on a ladder (with n rangs) $\Leftrightarrow$ counting spanning trees on a $2n$ grid
$n \to n + 1$

1. The 2 rightmost vertices for n share a connected component $\to A(n)$
2. The 2 rightmost vertices for n are in different connected components $\to B(n)$

Consider case 1
$A(n + 1) = 3A(n) + 1B(n)$

Consider case 2

$$A(1) = \frac{1}{2\sqrt{3}}\left((2+\sqrt{3})^1 - (2-\sqrt{3})^1\right) = \frac{2\sqrt{3}}{2\sqrt{3}}$$

$$A(2) = \cdots = \frac{8\sqrt{3}}{2\sqrt{3}} = 4$$

$$A(n) = 4A(n-1) - A(n-2)$$

$$= \frac{4}{2\sqrt{3}}\left((2+\sqrt{3})^{n-1} - (2-\sqrt{3})^{n-1}\right) - \frac{1}{2\sqrt{3}}\left((2+\sqrt{3})^{n-2} - (2-\sqrt{3})^{n-2}\right)$$

2 Inclusion – Exclusion Principle

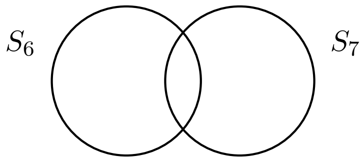
2.1 Example: Multipliers

$S = \{1, \ldots, 100\}$

Question: How many numbers in S are multipliers of 6 or 7?

$|S_6| = \lfloor \frac{100}{6} \rfloor$ = 16, $|S_6| = 14$

$|S_{6,7}| \neq |S_6| + |S_6|$:



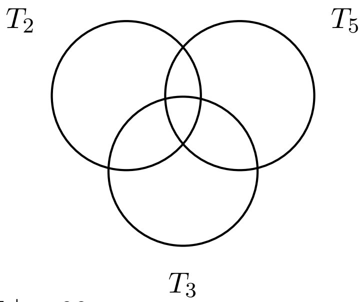
$|S_{6,7}| = |S_6| + |S_6| - |S_{6\&7}| = 16 + 14 - 2 = 28$

2.2 Example: Relatively Prime Numbers

$S = \{1, \ldots, 180\}$

Question: How many numbers in S are relatively prime to 180?

Idea: Count all numbers *not* relatively prime to 180 $\rightarrow T$.
$|T| =?$



$|T_2| = 90$
$|T_2| = 60$
$|T_2| = 36$

$|T| = |T_2| + |T_3| + |T_5| - (|T_2 \cap T_3| + |T_2 \cap T_5| + |T_3 \cap T_5|) + |T_2 \cap T_3 \cap T_5|$
$|T| = 90 + 60 + 36 - (30 + 18 + 12) + 6 = 132$

$|S| - |T| = 48$

In general:

$$\mathcal{A} = \{A_i\}_{i=1}^{n}, I = \{1, \ldots, n\}$$
$$|_{i \in I} A_i| = \sum_{i=1}^{n} \left((-1)^{i-1} \sum_{J \subseteq I, |J|=i} |\bigcap_{j \in J} A_j| \right)$$

Proof: $x \in A_1 \cup \cdots \cup A_n$, x is contained in m sets A_i.

$$|A_1 \cup \cdots \cup A_n| = \sum(.) - \sum(. \cap .) + \sum(. \cap . \cap .) - \sum(. \cap . \cap . \cap .) \ldots$$

1. for (single) sets *not* containing $x \to +0$
2. if in any $A_i \cap \cdots \cap A_j$ at least one of the sets does *not* contain $x \to \pm 0$
3. $\to$ only consider intersections of sets where all sets contain x

   $\binom{m}{k}$ intersections of k sets, all sets containing $x \to \pm 1$

$$\Rightarrow + \binom{m}{1} - \binom{m}{2} + \cdots + (-1)^{m-1} \binom{m}{m} = 1$$
$$\binom{m}{0} + \binom{m}{1} - \binom{m}{2} + \cdots + (-1)^{m} \binom{m}{m} = 0$$

   Pascal's triangle

$$\begin{array}{ccccccccc} & 1 & & 3 & & 3 & & 1 & \\ \swarrow & & \searrow\swarrow & & \searrow\swarrow & & \searrow\swarrow & & \searrow \\ 1 & & 4 & & 6 & & 4 & & 1 \end{array}$$

2.3 Example: Number of Words

Given: Language L with alphabet $\Sigma = \{A, B, C\}$ and every word of L consists of 10 characters.

Question: How many words of L contain each character at least once?

$$|G| = |L| - |X| = 3^{10} - (3 \cdot 2^{10} - 3 + 0) = 55980$$

$$|S_A| = |S_B| = |S_C| = 2^{10}$$
$$|S_A \cap S_B| = |S_A \cap S_C| = |S_B \cap S_C| = 1$$
$$|S_A \cap S_B \cap S_C| = 0$$
$$|X| = |S_A| + |S_B| + |S_C| - |S_A \cap S_B| - |S_A \cap S_C| - |S_B \cap S_C| + |S_A \cap S_B \cap S_C|$$

3 The Pigeon Hole Principle

If n elements are distributed to k sets, then there exists at least one set containing at least $\lceil \frac{n}{k} \rceil$ elements.

3.1 Example: People in Vienna

Claim: In Vienna there exist at least two persons with the exact same number of hairs on their head.

- human: at most $\approx 500,000$ hairs
- Vienna: $\geq 1.7 \cdot 10^6$ citizens

Idea: Take 500,000 people. If 2 of them have the same amount of hair, then we are done. If not, add one more person, whose amount of hair must then already be present.

3.2 Example: Division of odd numbers

Given: $1, 3, 7, 15, 31, \ldots ; a_i := 2^i - 1, i \geq 1$ and $q > 0$ arbitrary odd number.

Claim: At least one of the a_i's is (integer) divisible by q.

Proof: Consider all a_i, $1 \leq i \leq q$

- If one of these a_i is divisible by q, then we are done
- otherwise: $a_i = d_i \cdot q + r_i$, $0 < r_i < q$; $q \geq n > m > 0$

  $\Rightarrow (q-1)$ remainders $\overset{\text{p.h.p.}}{\Rightarrow}$ $\exists r_m, r_n$ such that $r_m = r_n$

  Then $(a_n - a_m) = (d_n - d_m)q + \underbrace{(r_n - r_m)}_{0}$

  $(2^n - 1) - (2^m - 1) = 2^n - 2^m = 2^m(\underbrace{2^{n-m} - 1}_{=a_{n-m}})$.

  2^m is even and thus not divisible by q.

In general: if $A_1, \ldots A_k$ are finite (pairwise disjoint) sets *and* $|A_1 \cup \cdots \cup A_k| > kr$, then $\exists i$ such that $|A_i| > r$.

Proof: Assume for the sake of contradiction that all $|A_i| \leq r$

$k \cdot r \geq \sum_{i=1}^{k} |A_i| \geq |A_1 \cup \cdots \cup A_k| > k \cdot r$

3.3 Example: Lossless Data Compression

. . . cannot guarantee compression for all input data!

- file $\rightarrow$ string of bits
- assume that each file is transformed into a *distinct* file that is not larger
- let F be the file with the least number of bits (M) that compresses $\rightarrow$ N bits
- $N < M$: $2^N + 1$ – cannot be all distinct

4 Reverse Search

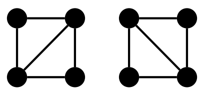
- Avis, Fukuda 1992
- enumerating data/objects/elements with almost no structure
- algorithmic
- count all objects *exactly* once, no overcounting

4.1 Example: Triangulations

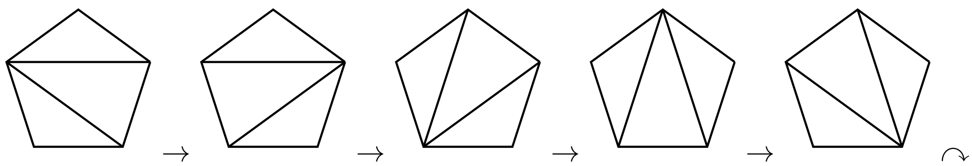
Given: set S of labelled points in convex position

Question: # of triangulations on S.

- $n = 4$



- $n = 5$



Flip on triangulations: 4 points, remove the diagonal and add the other diagonal

4.2 Basic Idea

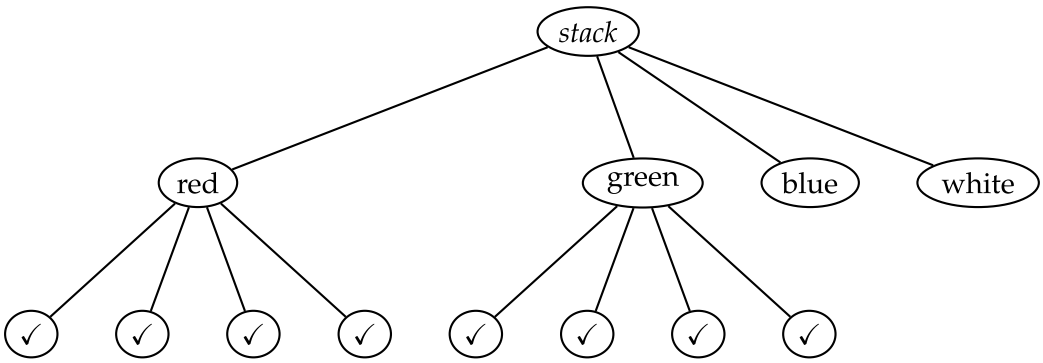
- abstract graph $G(V, E)$
- vertices V: elements to be enumerated, $|V|$ is very large
- edges $e \in E : e = v_1, v_2; v_1, v_2 \in V$
- $\Rightarrow G$ is an undirected graph
- G must be connected
- task: enumerate all vertices of G
- idea: breadth-first or depth-first search: $O(|V| + |E|)$ which is ok, but too much space necessary

4.3 Example: Poker Chips Stacking

Given: four colours (red, green, blue, white) of poker chips

Task: build stacks of height $\leq$ 30 such that at most 3 chips of equal colour appear directly after another

- v_Φ: empty stack
- neighbours:
  - 'successors': add a chip on top of the stack
  - 'predecessors': remove the topmost chip from the stack
- depth-first search



- store current stack.

  Nodes that have already been visited can be deduced from the stack composition.

Requirements:

- unique root
- successors and predecessors in arbitrary but fixed order
  neighbour relation $\gamma(v,k) \in \Gamma(v)$ (kth neighbour)
- a unique predecessor function $f(v), f : v \to v$ in $G(V,E)$
  1. $\exists v_0 \in V : f(v_0) = v_0$: root
  2. $\forall v \in V \backslash \{v_0\} : f(v) \in \Gamma(v)$: exactly one root
  3. $\forall v \in V \backslash \{v_0\} \forall x \in \mathbb{N} : f^x(v) \neq v$: cycle-free ($f^x = f(f \dots f(v) \dots)$) x times
     $\to$ reaching the root in finite time $f^{|V|}(v) = v_0$

4.4 Example: Tic Tac Toe

Given:

Task: Enumerate all valid game positions (not considering symmetries)

- V: game positions
- labelled board cells
- neighbours $\gamma(v,k)$: legally add k-th mark or remove one
- predecessor function: empty board as root, otherwise remove highest legal mark
- when iterating over the possible positions, check each node's predecessor to see if it has been visited before

4.5 Example: Connect Four (4 Gewinnt)

$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
			X		
			O		

too complex for reverse search, NP-hard

4.6 Algorithm

4.7 Example: Triangulations

4.7.1

4.7.2

4.8 Complexity Analysis

I. While loop (amortised analysis)

(r1)
- look at each neighbour of $v \Leftrightarrow$ each edge incident to v: $2 \cdot |E|$
- $t(\gamma)$: the time needed for $\gamma(v, j)$

$\Rightarrow \quad \approx t(\gamma) \cdot |E|$

(r2)
- $t(f)$: the time needed for $f(w)$ (predecessor)

$\Rightarrow \quad \approx t(f) \cdot |E|$

II. If Block

(f1) once per vertex $\Rightarrow \; t(f) \cdot |V|$

(f2) as often as $(f1)$

Let $\delta_{\max} \geq \delta(v) \; \forall_{v \in V}$ be the max. vertex degree of $G(V, E)$.
Repeat-until has at most $\delta_{\max}$ loops.
$\Rightarrow \quad \approx t(\gamma) \cdot |V| \cdot \delta_{\max}$

5 Pólya-Redfield Enumeration Theorem

5.1 Definitions

Main question: how many different objects exist?

Objects

- set of objects X
- e.g. Strings, coloured grids, Tic Tac Toe board,…

Operations

- set of n operations $R = \{R_i, 0 \leq i \leq n-1\}$
- R forms a group (operation $\circ$)

  Axioms ($\forall R_i, R_j, R_k$):

  - Closure: $\exists R_k : R_i \circ R_j = R_k$
  - Associativity: $(R_i \circ R_j) \circ R_k = R_i \circ (R_j \circ R_k)$
  - Inverse element $\exists R_k : R_i \circ R_k = R_k \circ R_i = R_0$
  - Identity element R_0: $R_i \circ R_0 = R_0 \circ R_i = R_i$

  Commutativity is in general not given.

- e.g. rotations, reflections

Orbits

- set of objects X which can be transformed into one another with an operation R_i.
- length: number of items within

Main question reformulated: how many orbits exist?

Stabilisers

- R_i stabiliser for object $x \Leftrightarrow R_i(x) = x$
- m_x: number of stabilisers for x
- r_i: number of objects for which R_i is a stabiliser (invariance number)
- it holds that $\sum\limits_{i=0}^{n-1} r_i = \sum\limits_{x \in X} m_x$

5.2 Counting Orbits

- Idea: specific case where $\sum_{x \in \text{orbit}} m_x = n$
  - go through graph of possible objects – unify those, that are one operation apart
  - in this case $\sum\limits_{x \in X} m_x = \sum\limits_{\text{all orbits}} \sum\limits_{x \in \text{orbit}} m_x$
- then number of orbits $= \frac{\sum\limits_{x \in X} m_x}{n}$

- invariance numbers r_i usually easier to obtain than all m_x
- therefore use number of orbits $= \frac{\sum\limits_{i=0}^{n-1} r_i}{n}$

5.3 Algorithm

1. identify all operations R_0 to R_{n-1}. Check for group properties.
2. compute all invariance numbers r_i, $0 \leq i \leq n-1$
3. compute the number of orbits as $\frac{\sum\limits_{i=0}^{n-1} r_i}{n}$

5.4 Example: Cyclic Shift

Strings of length 4 with characters $\{A, B, C\}$. Cyclic shift by i positions ($0 \leq i \leq 3$). How many different strings exist?

Invariance numbers

- $R_0 : r_0 = 3^4 = 81$
- $R_1 : r_1 = 3$, since a shift by 1 position means that all characters need to be the same if the result must be the same.
- $R_2 : r_2 = 3^2 = 9$
- $R_3 : r_3 = 3$

$$\#\text{orbits} = \frac{81 + 3 + 9 + 3}{4} = \frac{96}{4} = 24$$

Variation: every character has to be used at least once

$|X| = 3 \cdot \binom{2}{2} \cdot 2 = 3 \cdot 6 \cdot 2 = 36$

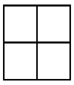
Invariance numbers

- $r_0 = 3^4 = 36$
- $r_1 = r_3 = 0$
- $r_2 = 0$

$$\#\text{orbits} = \frac{36 + 0 + 0 + 0}{4} = 9$$

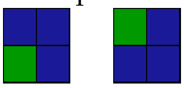
5.5 Example: Grid Colouring

5.5.1 2×2 Grid, 2 Colours



$|X| = 2^4 = 16$

Example for rotation ($90°$) invariance:



Number of orbits when..

1. only rotation is allowed

   Invariance numbers

   - $R_0 : r_0 = 16$ ($0°$ rotation)
   - $R_1 : r_1 = 2$ ($90°$ rotation)
   - $R_2 : r_2 = 4$ ($180°$ rotation)
   - $R_3 : r_3 = 2$ ($270°$ rotation)

   $$\#\text{orbits} = \frac{16 + 2 + 4 + 2}{4} = \frac{24}{4} = 6$$

2. rotation and reflection are allowed

   The grid can be reflected horizontally, vertically and diagonally

   - r_0 to r_3 as above
   - $R_4 : r_0 = 4$ (vertical reflection)
   - $R_5 : r_1 = 4$ (horizontal reflection)
   - $R_6 : r_2 = 8$ (top left to bottom right diagonal reflection)
   - $R_7 : r_3 = 8$ (top right to bottom left diagonal reflection)

   $$\#\text{orbits} = \frac{16 + 2 + 4 + 2 + 4 + 4 + 8 + 8}{8} = 6$$

Note that with more operations, the number of orbits never increases.

5.5.2 3×3 Grid

3×3 grid: how many different colourings exist, if we allow rotation/rotation and reflection?

5.6 Example: Cube Colouring

6 sides, k colours. How many different colourings exist if you can rotate the cube?

5.7 Example: Tic Tac Toe

How many different positions can we get after k half-moves, considering symmetries?

6 Generating Sequences

Example: $\{1,2,3\},\{1,3,2\},\{2,1,3\},\{2,3,1\},\{3,1,2\},\{3,2,1\} \to 6 = 3!$ permutations

6.1 Transition

Exchange two elements
$\{1,2,3,4\} \longrightarrow \{4,2,3,1\}$

6.2 Cycle Notation

$\{4,2,1,3\} \longrightarrow (1,4,3)(2) \overset{\text{can. rep.}}{\longrightarrow} (2),(1,4,3)$ $1 \to 4 \to 3 \to 1$

6.3 Derangements

Derangement: permutation where every object ends up in a new position.

Count using Inclusion-Exclusion Principle

$$|\text{derangements}| := \ !n = |\text{all perms.}| - |\#\text{perms. with } \geq 1 \text{ fixed number}| = \lfloor \frac{n!}{e} + \frac{1}{2} \rfloor$$

6.3.1 Example: Strings of Numbers

1. $\{1,2,3\}$, $(1)(2)(3)$
2. $\{1,3,2\}$, $(1)(2,3)$
3. $\{2,1,3\}$, $(1,2)(3)$
4. $\{2,3,1\}$, $(1,2,3)$
5. $\{3,1,2\}$, $(2,3,1)$
6. $\{3,2,1\}$, $(2)(3,1)$

4. and 5. are derangements.

6.4 Sorting

Comparison-based sorting is not possible any faster than $O(n \log n)$:

$$O(\log(n!)) = O(\log \underbrace{\frac{n}{e}^{n}}_{\text{Sterling's Approximation}}) = O(n \log n)$$

6.5 n-Queens Problem

Given: $n \times n$ chess board, n queens

Task: Position queens such that none threatens any other.

- Option 1: backtracking
- Option 2: permutation generator π
  - $\pi[i]$ specifies the column of the queen in the i-th row.
  - from one permutation to next: exchange 2 queens, check 4 diagonals

6.6 Generating all Permutations

How to generate permutations for problems as the n-queens problem?

6.6.1 Heap's Algorithm

c from 1 to n, two different steps:

- recursive computation of permutations $n - 1$
- oracle call to exchange two elements with a given oracle function

In Heap's algorithm, the oracle function is defined as $i = 1$ for current n odd, $i = c$ else.

Advantage: slightly more efficient than Steinhaus-Johnson-Trotter

6.6.2 Steinhaus-Johnson-Trotter Algorithm

Idea: from $n - 1$ to n by inserting the new element into all possible positions

Advantage: circular, only neighbouring elements are exchanged

6.6.3 Lexicographic Algorithms

None of the two algorithms produces permutations in lexicographic order.
(Reverse) lexicographic algorithms: loop

1. invert the order of $\pi[1], \dots, \pi[i]$
2. recursively generate permutations in reverse lexicographic order

6.6.4 Random permutations

as the name suggests :)

7 Gray Code – a 'reflected binary code'

- subsequent sequences only differ in one bit
- circular
- can also be used for puzzle solving, e.g. Towers of Hanoi

For n bits:

- $n = 1$

  0
  1

- $n = 2$

  00
  01
  11
  10

- $n = 3$

  000
  001
  011
  010
  110
  111
  101
  100

7.1 Conversion

$b_{n-1}b_{n-2}\dots b_1b_0 \Longleftrightarrow g_{n-1}g_{n-2}\cdots g_1g_0$
$b_{n-1} = g_{n-1}$

for $i \neq n-1$:

- $b_i = b_{i+1}\ XOR\ g_i$
- $g_i = b_{i+1}\ XOR\ b_i$

Examples Conversion:

- $1010_b \rightarrow 1111_g$
- $0100_g \rightarrow 01110_b$

7.2 Example: Towers of Hanoi

Given: three poles, n disks of different sizes. Stacked by size, largest at the bottom

Task: reverse the order of the disks. No disk must be placed on top of a smaller one.
Use a Gray code with as many bits as rings. Move the disk i when bit i changes. Start CW or CCW depending on the pole you want the rings to be on in the end and whether n is even or odd.

7.3 Example: Chinese Rings

7.4 Example: Subset Generation

8 Combinatorial 2 Player Games

Components

- state storage
- move generator
- identification of lose/win/draw states
- possibly backwards move generator

8.1 Game-Tree vs. State-Space Complexity

Game-Tree: consider all possible boards.
Complexity: number of nodes of the complete game's decision tree

State-Space: consider all different boards
Complexity: number of states which can be reached from the start by valid moves

Equivalence of Game States: they allow the same moves, result in the same successor.
Typically rotation, reflection, colour inversion possible.
Storing one of these states is sufficient.

8.2 Levels of Game Solution

- **ultra-weakly solved:** it is known who can win, but not how
- **weakly solved:** a winning strategy from the starting position is known
- **strongly solved:** a strategy from all positions is known
- **ultra-strongly solved:** for any position it is known what the final result will be and after how many half-moves this will happen

8.3 Enumerate all States