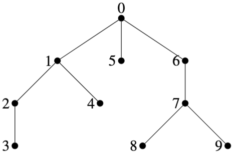
Picking, posing and attacking natural problems in Discrete Mathematics: from insightful bijections to black-box help from Machine Learning



Stoyan Dimitrov

Rutgers University

Rutgers Experimental Math Seminar, Jan 23, 2025

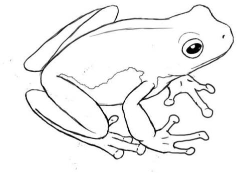
RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

What should be the goal of the math researcher?

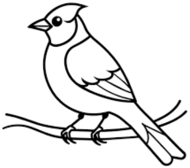
What should be the goal of the math researcher?

The famous dichotomy:

Problem-solvers
("*Frogs*")



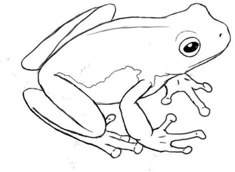
Theory-builders
("*Birds*")



What should be the goal of the math researcher?

The famous dichotomy:

Problem-solvers
("*Frogs*")



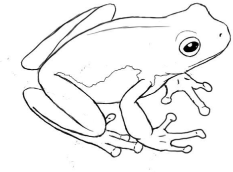
Theory-builders
("*Birds*")

Frogs see only the flowers that grow near them. They delight in the details of particular objects, and they solve problems one at a time.

– Freeman Dyson

The famous dichotomy:

Problem-solvers
("*Frogs*")



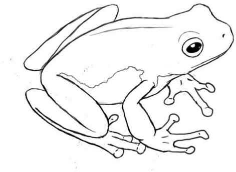
Theory-builders
("*Birds*")

Birds fly high in the air and delight in concepts that unify our thinking and bring together diverse problems from different parts of the landscape.

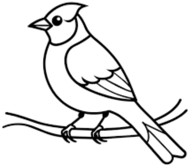
– Freeman Dyson

The famous dichotomy:

Problem-solvers
(“*Frogs*”)



Theory-builders
(“*Birds*”)



As a **combinatorialist**, I’m $\approx$ *a prototypical frog*

The importance of picking good problems

Mathematical research is not about solving problems; It is about finding the right problems.

Gian-Carlo Rota,
"Combinatorics: The Rota Way"

To select a problem for your PhD student is the same as to choose a bride for one's son.

Vladimir Arnold,
"Arnold's problems"

The importance of picking good problems

> Mathematical research is not about solving problems; It is about finding the right problems.
>
> *Gian-Carlo Rota,*
> *"Combinatorics: The Rota Way"*

> To select a problem for your PhD student is the same as to choose a bride for one's son.
>
> *Vladimir Arnold,*
> *"Arnold's problems"*

> How people ask questions is the most difficult thing to understand, in order to mirror intelligence!
>
> *Vladimir Vapnik,*
> *"Lex Fridman's podcast"*

* In the computer era, attacking problems is easier, so picking good such is even more important!

Characteristics of a good problem?

– natural (with succinct formulation)   // but why??

– well-motivated

Characteristics of a good problem?

– natural (with succinct formulation)   // but why??

– well-motivated

– surprising, as a fact

– unsolved, yet not too difficult

* Pick problems you are very excited/curious about! (why?)

Sources of problems and ideas

- ⋆ Workshops/Reading groups/Conferences/Seminars.
- ⋆ Follow the work of your research heroes!

Sources of problems and ideas

- ⋆ Workshops/Reading groups/Conferences/Seminars.
- ⋆ Follow the work of your research heroes!
- ⋆ Read the recent arXiv papers.
- ⋆ Textbooks on the fundamentals.

Two options: **Selecting** an existing problem OR **Posing** your own problem.

Problem posing is underrated!

> *I am not such a fast runner. If I am one of N people all working on the same problem, there is a very little chance I'll win.*
> *Thinking of a new problem in a new area will give me a chance!*
>
> *– Jim Simons*

Some ideas on posing new problems, when having a set of objects (examples – problems posed by H. Wilf on the set of partitions $\mathcal{P}_n$):

Problem posing is underrated!

I am not such a fast runner. If I am one of N people all working on the same problem, there is a very little chance I'll win.
Thinking of a new problem in a new area will give me a chance!

– Jim Simons

Some ideas on posing new problems, when having a set of objects (examples – problems posed by H. Wilf on the set of partitions $\mathcal{P}_n$):

a) Ask something about **parts** of the objects.

Example: Given $m \geq 0$, for how many partitions of $n = x_1 + x_2 + \ldots + x_k$, $x_1 \geq x_2 \geq \ldots \geq x_k$, we have $m = x_2 + x_4 + x_6 + \ldots$?

b) Ask something about **collections** of these objects.

Example: Is it true that for large enough n, an $n \times p(n)$ rectangle can be tiled by the Ferrers diagrams corresponding to all $p(n)$ partitions of n?

Problem posing is underrated!

I am not such a fast runner. If I am one of N people all working on the same problem, there is a very little chance I'll win. Thinking of a new problem in a new area will give me a chance!

– Jim Simons

Some ideas on posing new problems, when having a set of objects (examples – problems posed by H. Wilf on the set of partitions $\mathcal{P}_n$):

c) Ask a **probabilistic question** about a random object in the family.

   Example: What is the probability that a random partition in $\mathcal{P}_{2n}$ is the degree sequence of a simple graph, when $n \to \infty$?

d) Ask something about a **graph** defined over the set of objects.

   Example: Put and edge between the partitions p_1 and p_2 iff: increasing a part of p_1 by 1 and decreasing another part by 1 gives p_2. Is there a Hamiltonian path in this graph?

Outline

I. Enumerative and Algebraic Combinatorics (EAC)

II. Three results on EAC problems motivated by Computer Science tasks:

1. Sorting
2. Searching
3. Random sampling

III. Two results explaining *surprising* combinatorial observations.

- - - - - - - - - -

IV. Future work

– Reinforcement learning for mathematical conjectures.

Enumerative Combinatorics. What is it all about?:

Counting things that are difficult to count!

Algebraic Combinatorics:

Studies combinatorial structures using algebraic techniques.

Founding of EAC

Prior to 1960, EAC was considered to be a set of ad-hoc problems without a unifying theory!

Gian-Carlo Rota had a major contribution to the change.

Main topics developed by Rota (and students):

– partially ordered sets
– matroid theory
– symmetric functions
– finite operator calculus

EAC and TCS

Nowadays, EAC has gained traction, because of its applications to Theoretical Computer Science.

Here is a quote from Institute of Advanced Study’s website (2025):

> ***The tight connection between Discrete Mathematics and Theoretical Computer Science****, and the rapid development of the latter in recent years, led to an increased interest in Combinatorial techniques.... There are already well developed enumeration methods, some of which are based on deep algebraic tools.*

1. Application to sorting

Background: Avoiding patterns in permutations.

- S_n - the set of permutations of size n.

  Example: $4172365 \in S_7$.

Background: Avoiding patterns in permutations.

- S_n - the set of permutations of size n.

  Example: $4172365 \in S_7$.

- Permutation π contains a permutation p as a *pattern*, if there is a subsequence λ of π with elements in the same relative order as the elements of p.

  Example: 32514 contains the pattern 231.
  32514 contains the pattern 132.

Background: Avoiding patterns in permutations.

- S_n - the set of permutations of size n.

  Example: $4172365 \in S_7$.

- Permutation π contains a permutation p as a *pattern*, if there is a subsequence λ of π with elements in the same relative order as the elements of p.

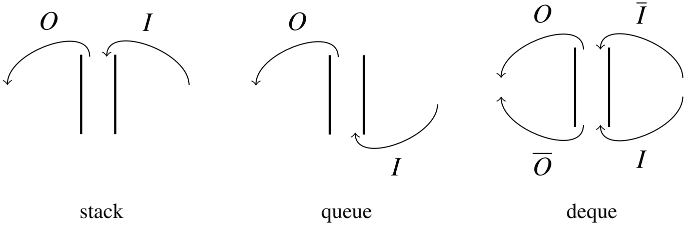
  Example: 32514 contains the pattern 231.
  32514 contains the pattern 132.

- Otherwise, π avoids p. We will write $\pi \in \mathrm{Av}(p)$.

  Example: 32514 avoids the pattern 123.

Knuth was the first to ask:

“Which input permutations can be sorted with the classical data structures *stack*, *queue* and *deque*?”



Example: stack sorting

52143

Example: stack sorting

$$\begin{array}{|c|} 5 \\ \\ \\ \end{array} \qquad 2143$$

Example: stack sorting

2 5	143

Example: stack sorting

$$\begin{array}{|c|}1\\2\\5\\ \\ \\\end{array}\qquad 43$$

Example: stack sorting

1 | 2
5 | 43

Example: stack sorting

12 | 5 | 43

Example: stack sorting

12	4	3
	5	

Example: stack sorting

$$12 \qquad \left|\begin{array}{c}3\\4\\5\\ \\ \\ \end{array}\right|$$

Example: stack sorting

12345

Example: stack sorting

12345

Which permutations are stack-sortable?

Theorem 1 (Knuth, 1968)

π is stack-sortable if and only if $\pi \in Av(231)$.

Theorem 2 (Pratt, 1973)

π is deque-sortable if and only if it avoids certain infinite set of patterns.

Theorem 1 (Knuth, 1968)

π is stack-sortable if and only if $\pi \in Av(231)$.

Theorem 2 (Pratt, 1973)

π is deque-sortable if and only if it avoids certain infinite set of patterns.

But how about sorting by a queue??

Sorting by a queue and cuts

Trivially, the only queue-sortable permutation is the identity!

But what if we consider a special queue that can do **cuts**?

Example:

213645

Sorting by a queue and cuts

Trivially, the only queue-sortable permutation is the identity!

But what if we consider a special queue that can do **cuts**?

Example:

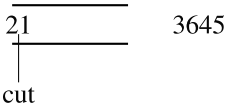
2	13645

Sorting by a queue and cuts

Trivially, the only queue-sortable permutation is the identity!

But what if we consider a special queue that can do **cuts**?

Example:



Sorting by a queue and cuts

Trivially, the only queue-sortable permutation is the identity!

But what if we consider a special queue that can do **cuts**?

Example:

12 3645

Sorting by a queue and cuts

Trivially, the only queue-sortable permutation is the identity!

But what if we consider a special queue that can do **cuts**?

Example:

12 3 645

Sorting by a queue and cuts

Trivially, the only queue-sortable permutation is the identity!

But what if we consider a special queue that can do **cuts**?

Example:

123 645

Sorting by a queue and cuts

Trivially, the only queue-sortable permutation is the identity!

But what if we consider a special queue that can do **cuts**?

Example:

123 6 45

Trivially, the only queue-sortable permutation is the identity!

But what if we consider a special queue that can do **cuts**?

Example:

123 64 5

Sorting by a queue and cuts

Trivially, the only queue-sortable permutation is the identity!

But what if we consider a special queue that can do **cuts**?

Example:

123 645

cut

Sorting by a queue and cuts

Trivially, the only queue-sortable permutation is the identity!

But what if we consider a special queue that can do **cuts**?

Example:

123456

Two results on sorting by cuts

Theorem 3 (D., 2022)

The cut-sortable permutations are the 321*-avoiding separable permutations, i.e., those in the set* $\mathrm{Av}_n(321, 2413, 3142)$.

Definition (cost of permutation)

$\mathrm{cost}(\pi) :=$ minimum number of times a queue performing cuts has to be used to sort π.

Theorem 4 (D., 2022)

$cost(\pi) \leq \lceil \frac{n}{2} \rceil$*, for any* $\pi \in S_n$.

Theorem 3 (D., 2022)

The cut-sortable permutations are the 321*-avoiding separable permutations, i.e., those in the set* $\mathrm{Av}_n(321, 2413, 3142)$.

Definition (cost of permutation)

$\mathrm{cost}(\pi) :=$ minimum number of times a queue performing cuts has to be used to sort π.

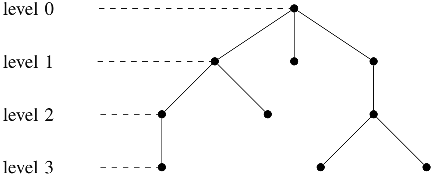
Theorem 4 (D., 2022)

cost$(\pi) \leq \lceil \frac{n}{2} \rceil$*, for any* $\pi \in S_n$.

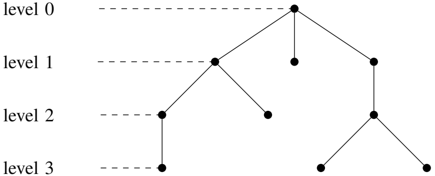
Conjecture (D., 2022)

By In-shuffles and Monge shuffles, one can sort <u>the same</u> number of permutations, given that every 'pop' unloads the entire queue.

Goal: Efficient search for random nodes in plane (ordered) trees.

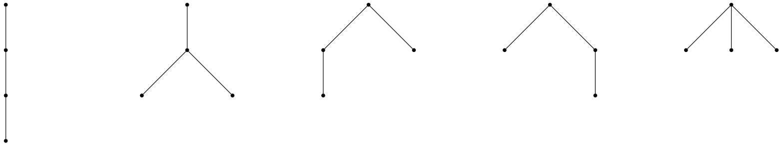


Goal: Efficient search for random nodes in plane (ordered) trees.



$\mathcal{T}_n :=$ the set of plane trees with n edges.

Example: The trees in $\mathcal{T}_3$:



The general problem

Assume that:

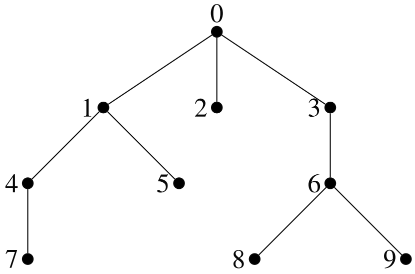
- We are at the root of some unknown $T \in \mathcal{T}_n$.
- There is a target node $x \in T$ that we want to find.

How to search for x?

BFS and DFS

Breadth-first search (BFS) and Depth-first search (DFS) are the two most popular searching algorithms.

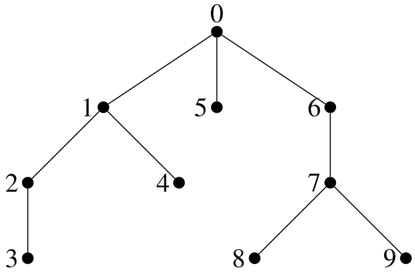
Example: BFS traversal



$\mathrm{bfsScore}(v) :=$ the number of nodes visited before v when using BFS.

Breadth-first search (BFS) and Depth-first search (DFS) are the two most popular searching algorithms.

Example: DFS traversal



$\mathrm{dfsScore}(v) :=$ the number of nodes visited before v when using DFS.

Random target

Question 1 (Easy)

Is BFS or DFS faster in expectation, if the target x is chosen uniformly at random among all nodes in $\mathcal{T}_n$?

Question 1 (Easy)

Is BFS or DFS faster in expectation, if the target x is chosen uniformly at random among all nodes in $\mathcal{T}_n$?

Answer: The expected bfsScore and dfsScore are the same!

$$\mathop{\mathbb{E}}_{\substack{x\in T\\ T\in\mathcal{T}_n}}(\mathrm{bfsScore}(x)) = \mathop{\mathbb{E}}_{\substack{x\in T\\ T\in\mathcal{T}_n}}(\mathrm{dfsScore}(x)) = \frac{1}{n}\frac{n(n+1)}{2} = \frac{n+1}{2}.$$

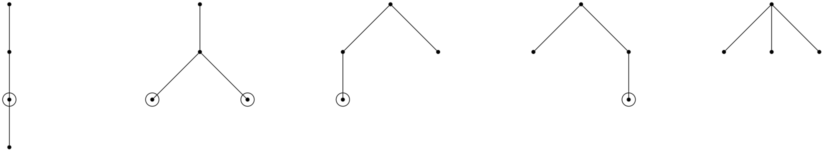
Fixed level target

Question 2

Is BFS or DFS faster in expectation, if the target x is chosen uniformly at random among all nodes ***at level*** *ℓ in $\mathcal{T}_n$? That is, can we compare*

$$\mathop{\mathbb{E}}_{\substack{x\in\mathrm{lev}(T,\ell)\\ T\in\mathcal{T}_n}}(\mathrm{bfsScore}(x)) \text{ and } \mathop{\mathbb{E}}_{\substack{x\in\mathrm{lev}(T,\ell)\\ T\in\mathcal{T}_n}}(\mathrm{dfsScore}(x))?$$

Example: $n = 3$, $\ell = 2$.



Fixed level target

Question 2

Is BFS or DFS faster in expectation, if the target x is chosen uniformly at random among all nodes ***at level*** ℓ *in* $\mathcal{T}_n$*? That is, can we compare*

$$\mathop{\mathbb{E}}_{\substack{x \in \mathrm{lev}(T,\ell) \\ T \in \mathcal{T}_n}}(\mathrm{bfsScore}(x)) \textit{ and } \mathop{\mathbb{E}}_{\substack{x \in \mathrm{lev}(T,\ell) \\ T \in \mathcal{T}_n}}(\mathrm{dfsScore}(x))?$$

– For small ℓ, BFS should be faster.

– For large ℓ, DFS should be faster.

Where is the threshold?

We have the following fact.

Theorem 5 (Dershowitz and Zaks, 1980)

The total number of nodes in $\mathcal{T}_n$ residing on level ℓ is

$$\frac{2\ell+1}{2n+1}\binom{2n+1}{n-\ell}.$$

We have the following fact.

Theorem 5 (Dershowitz and Zaks, 1980)

The total number of nodes in $\mathcal{T}_n$ residing on level ℓ is

$$\frac{2\ell+1}{2n+1}\binom{2n+1}{n-\ell}.$$

Thus, it suffices to compare

$$\mathrm{totalB}(n,\ell) \coloneqq \sum_{T\in\mathcal{T}_n}\sum_{v\in\mathrm{lev}(T,\ell)} \mathrm{bfsScore}(v),$$

and

$$\mathrm{totalD}(n,\ell) \coloneqq \sum_{T\in\mathcal{T}_n}\sum_{v\in\mathrm{lev}(T,\ell)} \mathrm{dfsScore}(v).$$

We obtained a surprisingly simple formula for $\mathrm{totalD}(n, \ell)$.

Theorem 6 (D., Minchev, Zhuang, 2024)

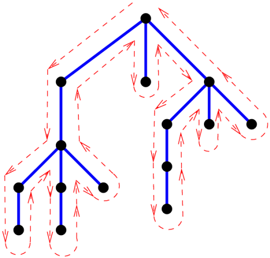
For every $n \geq 0$ *and* $\ell \in [0, n]$,

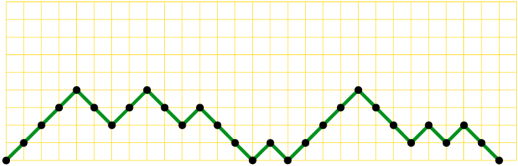
$$\mathrm{totalD}(n, \ell) = \ell \binom{2n}{n-\ell}.$$

To establish that, we used the well-known *accordion* bijection.
Then, the so-called *cycle lemma* to show that it suffice to prove that

$$\sum_{j=l}^{n} \binom{2j-l-1}{j-1} \binom{2n-2j+l+1}{n-j} \frac{l+1}{2n-2j+l+1} = \binom{2n}{n-l}.$$

The accordion bijection: plane trees and Dyck paths





Finding $\mathrm{totalB}(n, l)$ is more complicated.

First, we tried to use Generatingfunctionology!

Finding $\mathrm{totalB}(n, l)$ is more complicated.

First, we tried to use Generatingfunctionology!

For a given level l, let

$$F_l = F_l(x, y, z) := \sum_{T \in \mathcal{T}} x^{k(T)} y^{m(T)} z^{n(T)},$$

where

$\mathcal{T} := \bigsqcup_{n \geq 0} \mathcal{T}_n$,
$k = k(T)$ - nb of nodes in T at levels smaller than l,
$m = m(T)$ - nb of nodes in T at level l,
$n = n(T)$ - nb of edges in T.

totalB

We can show that

$$B_l = B_l(z) := \sum_{n=0}^{\infty} \mathrm{totalB}(n,l) z^n = \left[\left(\frac{\partial^2}{\partial x \partial y} + \frac{1}{2} \cdot \frac{\partial^2}{\partial y^2} + \frac{\partial}{\partial y} \right) F_l \right]_{x=1,\, y=1}.$$

We can show that

$$B_l = B_l(z) := \sum_{n=0}^{\infty} \mathrm{totalB}(n,l) z^n = \left[\left(\frac{\partial^2}{\partial x \partial y} + \frac{1}{2} \cdot \frac{\partial^2}{\partial y^2} + \frac{\partial}{\partial y} \right) F_l \right]_{x=1,\, y=1}.$$

It turns out that F_l and B_l can be expressed in terms of the so-called *Fibonacci polynomials*

$$f_n(z) := \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-k}{k} z^k,$$

and the Catalan generating function

$$C = C(z) := \frac{1 - \sqrt{1-4z}}{2z}.$$

Theorem 7 (D., Minchev, Zhuang, 2024)

For all $l \geq 1$,

$$F_l(x,y,z) = \frac{f_{l-1}(-xz) - f_{l-2}(-xz)yzC}{f_l(-xz) - f_{l-1}(-xz)yzC}$$

and

$$B_l(z) = z^l C^{3l+1} \left(zC \frac{d}{dz} f_l(-z) - \frac{d}{dz} f_{l+1}(-z) \right).$$

Using additional facts and Lagrange inversion, we obtain:

$$\mathrm{totalB}(n,l) = \sum_{k=1}^{\lfloor (l+1)/2 \rfloor} (-1)^{k-1} k \binom{l-k+1}{k} \frac{3l+1}{n-k+2l+2} \binom{2n-2k+l+2}{n-l-k+1} - \sum_{k=1}^{\lfloor l/2 \rfloor} (-1)^{k-1} k \binom{l-k}{k} \frac{3l+2}{n-k+2l+2} \binom{2n-2k+l+1}{n-l-k}.$$

But, this formula did NOT help us to compare totalB and totalD ...

totalB - attempt 2

Utilizing results of Takács (1999), we obtain

Theorem 8 (D., Minchev, Zhuang, 2024)

For every $n \geq 0$ and $0 \leq \ell \leq n$, we have

$$\mathrm{totalB}(n, \ell) = \frac{n(2\ell+1)}{n+\ell+1}\binom{2n}{n-\ell} - \binom{2n}{n-2\ell-1} - 2\sum_{j=\ell+1}^{2\ell}\binom{2n}{n-j}.$$

Via asymptotic bounds for $\binom{2n}{n-\ell}$ and $\sum_{j=\ell+1}^{2\ell}\binom{2n}{n-j}$, we obtain

$$\frac{\mathrm{totalB}(n, s\sqrt{n})}{4^n} = \frac{2s}{\sqrt{\pi}}e^{-s^2} - 2\left(\Phi\left(2s\sqrt{2}\right) - \Phi\left(s\sqrt{2}\right)\right) + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right),$$

where Φ denotes the c.d.f. of the standard normal distribution.

Using the latter Theorem 8, we prove our main result:

Theorem 9 (D., Minchev, Zhuang, 2024)

As $n \to \infty$, $\mathrm{totalB}(n, \ell) \le \mathrm{totalD}(n, \ell)$ if and only if $\ell \le C\sqrt{n}$, where $C \approx 0.789004$ is the unique positive root of the equation

$$xe^{-x^2} - 2\sqrt{\pi}\left(\Phi\left(2x\sqrt{2}\right) - \Phi\left(x\sqrt{2}\right)\right) = 0.$$

Moreover, there is a unique such threshold, for each $n \ge 1$.

We also managed to find an asymptotic formula for $\mathrm{totalB}(n, l)$, for an arbitrary Galton-Watson tree with n edges.

We suspected the threshold is close to the average level of a node in $\mathcal{T}_n$.

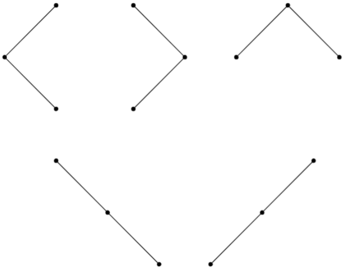
Theorem 10 (Flajolet and Sedgewick, 2009)

The average level of the nodes among all trees in $\mathcal{T}_n$ is $\frac{1}{2}\sqrt{\pi n} + o(1)$.

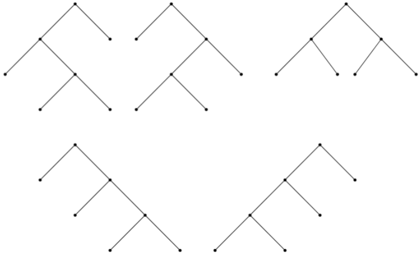
However, $\frac{1}{2}\sqrt{\pi} \approx 0.8862 > 0.789 \approx C$.

Question

How to generate trees (binary/full binary/ordered) uniformly at random?



(A) All binary trees with 3 nodes.



(B) All full binary trees with 4 leaves.

Catalan numbers

The numbers C_n count:

- Ordered trees with n edges.
- Binary trees with n nodes.
- **Full binary trees with** $n+1$ **leaves.**

  [+ more than 200 other combinatorial objects]

Remy's beautiful algorithm

We have

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

This is equivalent to

$$(n+2)C_{n+1} = 2(2n+1)C_n.$$

Remy's combinatorial proof of this identity gives a simple random sampling algorithm!

Remy's beautiful algorithm

$$(n+2)C_{n+1} = 2(2n+1)C_n.$$

You have a random full binary tree with $2n+1$ nodes (and $n+1$ leaves).

1. Pick a random node v among $\{1, 2, \ldots, 2n+1\}$.
2. Pick a random direction $d = L|R$.
3. Move v in direction d and add a leaf in the other direction.

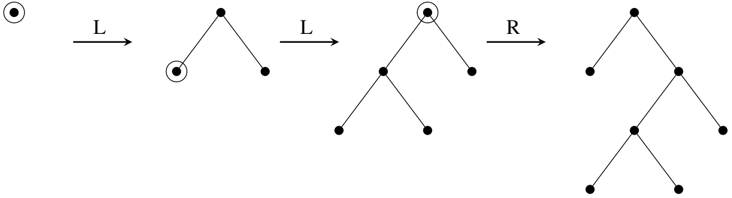
You got a random tree with $n+2$ leaves (one of them marked)!

$$(n+2)C_{n+1} = 2(2n+1)C_n.$$

You have a random full binary tree with $2n+1$ nodes (and $n+1$ leaves).

1. Pick a random node v among $\{1, 2, \ldots, 2n{+}1\}$.
2. Pick a random direction $d = L|R$.
3. Move v in direction d and add a leaf in the other direction.

You got a random tree with $n+2$ leaves (one of them marked)!



Generalization to forests

Forest in $F_{n,k} :=$ a list $[T_1, \ldots, T_k]$ of k full binary trees with n leaves in total.

Let $|F_{n,k}| = C_n^{(k)}$. Catalan himself showed:

$$C_n^{(k)} = \frac{k}{2n+k}\binom{2n+k}{n}.$$

Forest in $F_{n,k} :=$ a list $[T_1, \ldots, T_k]$ of k full binary trees with n leaves in total.

Let $|F_{n,k}| = C_n^{(k)}$. Catalan himself showed:

$$C_n^{(k)} = \frac{k}{2n+k}\binom{2n+k}{n}.$$

We prove the following in the spirit of Remy:

Theorem 11

For any $k \geq 2$ and $n \geq k$,

$$n(n-k)C_n^{(k)} = (2n-k-1)(2n-k-2)C_{n-1}^{(k)}.$$

Forest in $F_{n,k}$:= a list $[T_1, \ldots, T_k]$ of k full binary trees with n leaves in total.

Let $|F_{n,k}| = C_n^{(k)}$. Catalan himself showed:

$$C_n^{(k)} = \frac{k}{2n+k}\binom{2n+k}{n}.$$

We prove the following in the spirit of Remy:

Theorem 11

For any $k \geq 2$ and $n \geq k$,

$$n(n-k)C_n^{(k)} = (2n-k-1)(2n-k-2)C_{n-1}^{(k)}.$$

This gives a linear time and space algorithm for sampling of forests in $F_{n,k}$!

Another main theme of my research!

Two examples (work with students):

1. Chess tableaux and powers of two.
2. The 4321-avoiding involutions in S_n and
   the $(n, n+1, n+2)$ - core partitions are equinumerous!

Chess tableaux and powers of two

A *Standard Young Tableaux* (SYT) for $\lambda \vdash n$:= a filling with $1, 2, \ldots, n$ of the Young diagram for λ (s.t., the rows and columns are increasing).

A *Chess tableaux* := a SYT with alternating parity of the entries.

1	2	3	6	7	10
4	5	8	9		
11					

a Chess tableaux for the partition $(6, 4, 1)$.

Chess Tableaux arise in *representation theory* and *mathematical physics*.

Chess tableaux and powers of two

Let $\mathrm{SYT}(\lambda) :=$ the nb of SYTs for the partition λ.

Let $\mathrm{Chess}(n) :=$ the nb of Chess tableaux for the partition λ.

Famously, the RSK correspondence implies

$$\sum_{\lambda \vdash n} \mathrm{SYT}(\lambda)^2 = n!$$

Chess tableaux and powers of two

Let $\mathrm{SYT}(\lambda) :=$ the nb of SYTs for the partition λ.

Let $\mathrm{Chess}(n) :=$ the nb of Chess tableaux for the partition λ.

Famously, the RSK correspondence implies

$$\sum_{\lambda \vdash n} \mathrm{SYT}(\lambda)^2 = n!$$

Chow, Eriksson and Fan looked at the values of $\sum_{\lambda \vdash n} \mathrm{Chess}(\lambda)^2$.

They noticed that this sum is divisible by unusually high powers of 2:

$$\boxed{\begin{array}{c} 1,\ 2,\ 2,\ 2^2,\ 2^3,\ 2^4,\ 2^4\cdot 3,\ 2^5\cdot 5,\ 2^6\cdot 7,\ 2^{11},\ 2^8\cdot 5^2,\ 2^9\cdot 61,\ 2^{10}\cdot 3\cdot 41, \\ 2^{11}\cdot 5\cdot 59,\ 2^{11}\cdot 1523,\ 2^{13}\cdot 23\cdot 83,\ 2^{13}\cdot 11411,\ 2^{15}\cdot 103\cdot 163,\ \ldots \end{array}}$$

[The first few values of $\sum_{\lambda \vdash n} \mathrm{Chess}(\lambda)^2$]

Using a connection between Chess tableaux and certain representation of *Lie algebras*, we prove the following.

Theorem 12 (Labelle, D., 2023)

Let $a(n) :=$ *the nb of triangular numbers between* 1 *and* n *(equivalently,* $a(n)$ *is the largest integer* m *such that* $\frac{m(m+1)}{2} \leq n$*).*

Then, $\sum_{\lambda \vdash n} \mathrm{Chess}(\lambda)^2$ *is divisible by* $2^{n-a(n)}$.

Remark: It is known that $2^{n-b(n)}$ divides $n!$, where $b(n)$ is the number of 1s in the binary representation of n.

Since $b(n) \leq \log(n+1)$ and $a(n)$ grows as $\sqrt{n}$, the observation of Chow et al. is <u>not</u> that surprising!

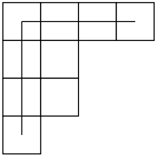
Involution – a permutation p, such that $p^2 = \mathrm{id}$ (equivalently, p has cycles lengths 1 or 2).

Example. $4261735 = (14)(2)(36)(57)$.

Involution – a permutation p, such that $p^2 = \mathrm{id}$
(equivalently, p has cycles lengths 1 or 2).

Example. $4261735 = (14)(2)(36)(57)$.

t-core partition – all of its hook numbers are <u>not</u> divisible by t.



The Young diagram of $(4, 2, 2, 1)$ and the hook number $H(1, 1) = 7$.

Involution – a permutation p, such that $p^2 = \mathrm{id}$ (equivalently, p has cycles lengths 1 or 2).

Example. $4261735 = (14)(2)(36)(57)$.

t-core partition – all of its hook numbers are not divisible by t.

7	5	2	1
4	2		
3	1		
1			

All hook numbers for the partition $(4, 2, 2, 1)$.

* core partitions were used to prove $p(5n+4) \equiv 0 \mod 5$ and other Ramanujan's congruences.

4321-avoiding involutions and core partitions

$I_n(4321) :=$ the set of involutions in S_n avoiding the pattern 4321.

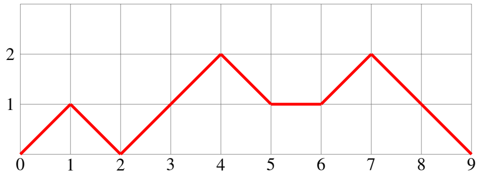
$\mathrm{CoreP}(n, n+1, n+2) :=$ partitions, which are n, $n+1$, and $n+2$-core.

Previous results show that both $|I_n(4321)|$ and $|\,\mathrm{CoreP}(n, n+1, n+2)|$ are counted by the Motzkin number M_n.

In 2022, Tewodros Amdeberhan asked for a bijective proof.

4321-avoiding involutions and core partitions

We found a direct such bijection [Schleppy, D., 2024].
Example:



The Motzkin path corresponding to $\pi = 215836947 = (12)(35)(48)(6)(79)$.

17	15	8	7	6	5	4	2	1
14	12	5	4	3	2	1		
8	6							
7	5							
6	4							
5	3							
4	2							
3	1							
1								

The $(9, 10, 11)-$core partition corresponding to π.

Future work

Directions:

- More EAC problems with applications to Computer Science.
- Explaining more surprising conjectural results on combinatorial objects.
- Applications of Machine Learning to Combinatorics.

On one exciting Reinforcement Learning idea

Reinforcement Learning (RL) $:=$ a type of machine learning, where an agent learns to make decisions in an unknown environment.
[it makes actions, get rewards and tries to maximize the long-term reward]

RL turned out to be extremely successful in games!
Here is little more about that...

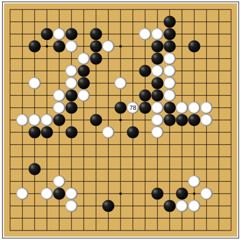
Humans vs Computers in games

In 1994, a team of scientists created Chinook - a **checkers** program that beat the world champion Marion Tinsley (who had 5 loses during 1950–1992)

In 1997, The IBM software Deep Blue defeated the strongest **chess** player at that time, Garry Kasparov, by 3.5 to 2.5 points. This was considered a milestone in the history of AI!

Humans vs Computers in games

The next milestone was the game of **Go**!
In 2016, The Google DeepMind software, *AlphaGo*, beat Lee Sedol (the player ranked second in international titles) by 4-1.



In 2017, DeepMind published a paper on *AlphaGo Zero* - a Go program training itself by self-play (not using human knowledge).

AlphaGo Zero beat AlphaGo 100 games to 0, after 3 days of training.

Humans vs Computers in games

In 2018, DeepMind published a Nature article describing the more general program *AlphaZero*! It achieved super-human performance in Chess, Shogi and Go after a few hours of training.

AlphaZero makes surprising moves and sacrifices. Today's grandmasters improve their game by looking at ideas in games played by AlphaZero!

AlphaZero uses the so-called *Deep RL* (how?).

RL for combinatorial conjectures

In 2021, Wagner proposed an exciting idea:

To disprove a conjecture in combinatorics: define the construction of the corresponding extremal object as a game and use RL to train an agent to play that game (well).

He disproved several conjectures w/ that idea. Here is an example:

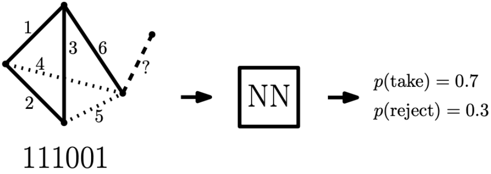
Conjecture (Disproved)

Let G be a connected graph on $n \geq 3$ vertices, with largest eigenvalue λ_1 and matching number μ. Then

$$\lambda_1 + \mu \geq \sqrt{n-1} + 1.$$

The game:

1. Start with the empty graph and a fixed order of the edges.
2. Choose to **take** or **skip** each of the edges, given your previous choices.
3. The neural net tries to learn the optimal $\mathbb{P}(\textit{take})$.
   The reward is $-(\lambda_1 + \mu)$, obtained at the end of each *episode*.



The agent included edges 1, 2, 3, 6, and rejected edges 4, 5 (dotted).

My idea – using KAN with Q-learning

The most popular RL method is Deep Q-learning (DQN).

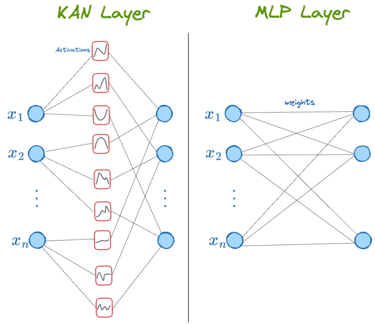
However, Wagner does not use DQN since it requires a lot of time to train.

My idea – using KAN with Q-learning

The most popular RL method is Deep Q-learning (DQN).

However, Wagner does not use DQN since it requires a lot of time to train.

Idea: Use Wagner's approach with DQN and the novel KAN architecture.



The KAN architecture versus the standard MLP architecture.

Why KAN?

Theorem 13 (Kolmogorov-Arnold Representation Theorem)

Any multivariate function can be decomposed into sums and compositions of univariate functions. That is,

$$f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

Advantageous to the standard MLP architechture:

- fewer parameters (thus faster to train).
- better interpretability.

THE END