Algorithmically Distinguishing Irreducible Characters of $\mathfrak{S}_n$

Timothy Y. Chow and Jennifer Paulhus

December 10, 2020

Representation Theory of Finite Groups

A **representation** of a finite group G is a homomorphism $\rho : G \to GL(\mathbb{C}^d)$, where d is the **degree** or **dimension** of ρ.

Representation Theory of Finite Groups

A **representation** of a finite group G is a homomorphism $\rho : G \to GL(\mathbb{C}^d)$, where d is the **degree** or **dimension** of ρ.

A representation ρ is **reducible** if there is a nontrivial subspace $0 \subsetneq V \subsetneq \mathbb{C}^d$ that is invariant; i.e., $\rho(g)V \subseteq V$ for all $g \in G$. If ρ is not reducible then we say it is **irreducible**.

Representation Theory of Finite Groups

A **representation** of a finite group G is a homomorphism $\rho : G \to GL(\mathbb{C}^d)$, where d is the **degree** or **dimension** of ρ.

A representation ρ is **reducible** if there is a nontrivial subspace $0 \subsetneq V \subsetneq \mathbb{C}^d$ that is invariant; i.e., $\rho(g)V \subseteq V$ for all $g \in G$. If ρ is not reducible then we say it is **irreducible**.

The number of non-isomorphic irreducible representations of a finite group is finite, and is equal to the number of conjugacy classes.

Classifying the irreducible representations of a finite group G gives us a lot of information about G.

Irreducible Characters

If ρ is a representation, then its **character** is the trace of $\rho(g)$.
It is a complex-valued function.

The trace is invariant under conjugation, so a character is a **class function**; i.e., it is constant on conjugacy classes.

Irreducible Characters

If ρ is a representation, then its **character** is the trace of $\rho(g)$.
It is a complex-valued function.

The trace is invariant under conjugation, so a character is a **class function**; i.e., it is constant on conjugacy classes.

It is a remarkable fact that if two representations have the same character then they are isomorphic.

The trace of an irreducible representation is called an **irreducible character**.

Irreducible Characters of $\mathfrak{S}_n$

Let $\mathfrak{S}_n$ denote the **symmetric group** of permutations on n elements.

Irreducible Characters of $\mathfrak{S}_n$

Let $\mathfrak{S}_n$ denote the **symmetric group** of permutations on n elements.

A **partition** of n is a weakly decreasing sequence $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ of positive integers that sum to n. We write $\lambda \vdash n$.

Note that the number of conjugacy classes of $\mathfrak{S}_n$ is equal to the number of partitions of n.

Irreducible Characters of $\mathfrak{S}_n$

Let $\mathfrak{S}_n$ denote the **symmetric group** of permutations on n elements.

A **partition** of n is a weakly decreasing sequence $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ of positive integers that sum to n. We write $\lambda \vdash n$.

Note that the number of conjugacy classes of $\mathfrak{S}_n$ is equal to the number of partitions of n.

Theorem. Given any $\lambda \vdash n$, one can construct an irreducible representation, called a **Specht module** and written S^λ, corresponding to λ. The Specht modules are pairwise non-isomorphic, and every irreducible representation of $\mathfrak{S}_n$ is isomorphic to some S^λ.

Irreducible Characters of $\mathfrak{S}_n$

Let $\mathfrak{S}_n$ denote the **symmetric group** of permutations on n elements.

A **partition** of n is a weakly decreasing sequence $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ of positive integers that sum to n. We write $\lambda \vdash n$.

Note that the number of conjugacy classes of $\mathfrak{S}_n$ is equal to the number of partitions of n.

Theorem. Given any $\lambda \vdash n$, one can construct an irreducible representation, called a **Specht module** and written S^λ, corresponding to λ. The Specht modules are pairwise non-isomorphic, and every irreducible representation of $\mathfrak{S}_n$ is isomorphic to some S^λ.

If ρ is (isomorphic to) a Specht module S^λ, then we write χ_λ for the corresponding irreducible character.

A Basic Question About Irreducible Characters

Suppose that χ_λ and χ_μ are two distinct irreducible representations of $\mathfrak{S}_n$. How hard is it to find $\pi \in \mathfrak{S}_n$ such that $\chi_\lambda(\pi) \neq \chi_\mu(\pi)$?

A Basic Question About Irreducible Characters

Suppose that χ_λ and χ_μ are two distinct irreducible representations of $\mathfrak{S}_n$. How hard is it to find $\pi \in \mathfrak{S}_n$ such that $\chi_\lambda(\pi) \neq \chi_\mu(\pi)$?

Surprisingly, this question does not seem to have been asked before anywhere in the literature.

A Basic Question About Irreducible Characters

Suppose that χ_λ and χ_μ are two distinct irreducible representations of $\mathfrak{S}_n$. How hard is it to find $\pi \in \mathfrak{S}_n$ such that $\chi_\lambda(\pi) \neq \chi_\mu(\pi)$?

Surprisingly, this question does not seem to have been asked before anywhere in the literature.

In one sense, the answer is, "Not hard." Empirically, if one simply tries various permutations with lots of fixed points, then it seems to take at most a few tries to find some π such that $\chi_\lambda(\pi) \neq \chi_\mu(\pi)$.

However, giving a *provable* upper bound on the number of tries seems difficult.

A Basic Question About Irreducible Characters

Suppose that χ_λ and χ_μ are two distinct irreducible representations of $\mathfrak{S}_n$. How hard is it to find $\pi \in \mathfrak{S}_n$ such that $\chi_\lambda(\pi) \neq \chi_\mu(\pi)$?

Surprisingly, this question does not seem to have been asked before anywhere in the literature.

In one sense, the answer is, "Not hard." Empirically, if one simply tries various permutations with lots of fixed points, then it seems to take at most a few tries to find some π such that $\chi_\lambda(\pi) \neq \chi_\mu(\pi)$.

However, giving a *provable* upper bound on the number of tries seems difficult.

Moreover, in general, even determining whether $\chi_\lambda(\pi) \neq 0$ is already NP-hard (Pak–Panova, 2017).

The Main Result

To have **oracle access** to a function f on $\mathfrak{S}_n$ means that the only way we can obtain information about f is to submit a *query* (i.e., an input value that we are free to choose) $\pi \in \mathfrak{S}_n$ to an oracle, which then truthfully tells us the value of $f(\pi)$.

The Main Result

To have **oracle access** to a function f on $\mathfrak{S}_n$ means that the only way we can obtain information about f is to submit a *query* (i.e., an input value that we are free to choose) $\pi \in \mathfrak{S}_n$ to an oracle, which then truthfully tells us the value of $f(\pi)$.

Theorem. There is a deterministic algorithm that, given oracle access to a function f that is promised to be an irreducible character of S_n, determines which irreducible character it is using $O(n^{3/2})$ queries (actually $O(n)$ according to a more careful analysis by Jiasheng Hu).

The Main Result

To have **oracle access** to a function f on $\mathfrak{S}_n$ means that the only way we can obtain information about f is to submit a *query* (i.e., an input value that we are free to choose) $\pi \in \mathfrak{S}_n$ to an oracle, which then truthfully tells us the value of $f(\pi)$.

Theorem. There is a deterministic algorithm that, given oracle access to a function f that is promised to be an irreducible character of S_n, determines which irreducible character it is using $O(n^{3/2})$ queries (actually $O(n)$ according to a more careful analysis by Jiasheng Hu).

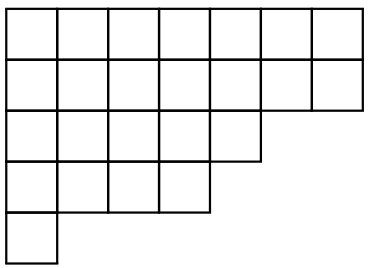
We can solve the original problem by running this algorithm on (say) χ_λ until we reach a query π for which $\chi_\lambda(\pi) \neq \chi_\mu(\pi)$.

Note: It turns out that the queries we need are easy to compute if we know the partition.

Young Diagrams

If $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ is a partition, then its **Young diagram** is a left-justified grid of boxes having λ_i boxes in row i.

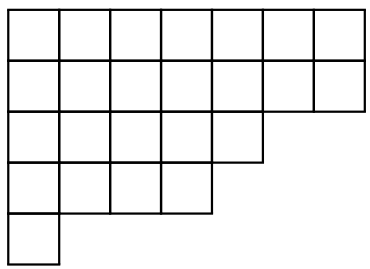
Below is the Young diagram of $(7, 7, 5, 4, 1)$.



Young Diagrams

If $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ is a partition, then its **Young diagram** is a left-justified grid of boxes having λ_i boxes in row i.

Below is the Young diagram of $(7, 7, 5, 4, 1)$.



The **conjugate** λ' of λ is the sequence of column lengths of λ.
The conjugate of $(7, 7, 5, 4, 1)$ is $(5, 4, 4, 4, 3, 2, 2)$.

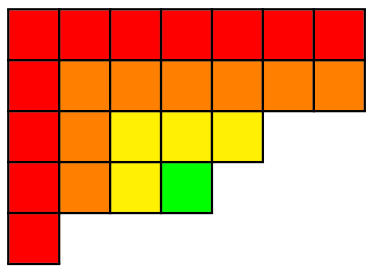
Principal Hooks

The i**th principal hook** of a Young diagram D is the set

$$H_i := \{(i,j) \in D : j \geq i\} \cup \{(j,i) \in D : j \geq i\}.$$

The ith principal hook length h_i is the area of H_i.

Example.



$$h_1 = 11, \quad h_2 = 8, \quad h_3 = 4, \quad h_5 = 1.$$

Outline of Algorithm

The algorithm has two phases.

During the **forward pass**, the principal hook *lengths* $h_1, h_2, h_3, \ldots$ are determined one at a time.

Outline of Algorithm

The algorithm has two phases.

During the **forward pass**, the principal hook *lengths* $h_1, h_2, h_3, \ldots$ are determined one at a time.

During the **backward pass**, the principal hook *shapes* are determined one at a time, in *reverse order*, starting with the innermost principal hook and working backward.

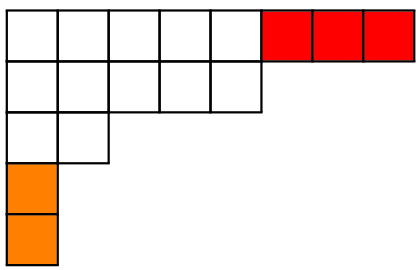
Outline of Algorithm

The algorithm has two phases.

During the **forward pass**, the principal hook *lengths* $h_1, h_2, h_3, \ldots$ are determined one at a time.

During the **backward pass**, the principal hook *shapes* are determined one at a time, in *reverse order*, starting with the innermost principal hook and working backward.

Each step of the backward pass determines the amount by which the principal hook **overhangs** the arm and the leg of the principal hook just inside of it. (Below, the arm overhang is 3 and the leg overhang is 2.)



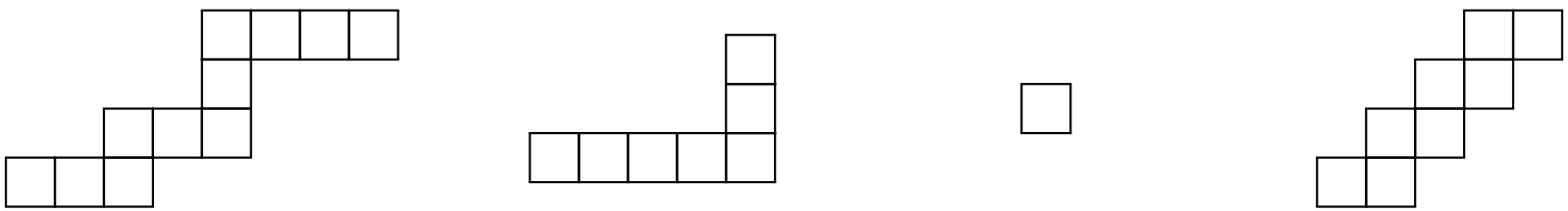
Border Strips

The algorithm relies heavily on the **Murnaghan–Nakayama rule,** a combinatorial recipe for computing irreducible characters of $\mathfrak{S}_n$.

Border Strips

The algorithm relies heavily on the **Murnaghan–Nakayama rule,** a combinatorial recipe for computing irreducible characters of $\mathfrak{S}_n$.

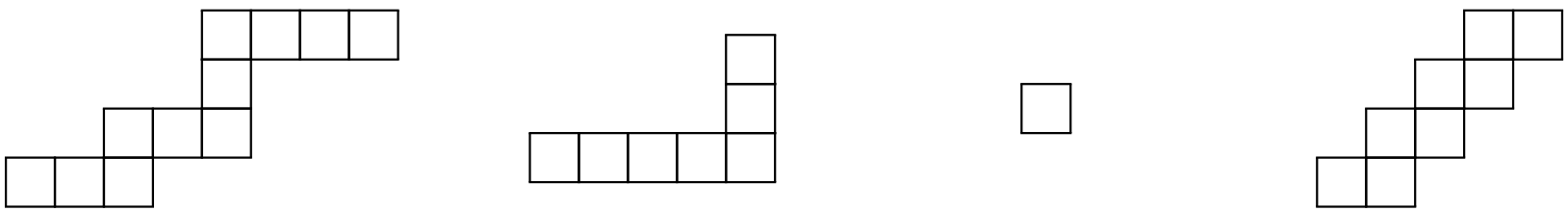
A **border strip** is a finite set of boxes such that in each row, the boxes in that row are contiguous, and except for the top row, the *rightmost* box in each row lies directly underneath the *leftmost* box in the row above it.



Border Strips

The algorithm relies heavily on the **Murnaghan–Nakayama rule,** a combinatorial recipe for computing irreducible characters of $\mathfrak{S}_n$.

A **border strip** is a finite set of boxes such that in each row, the boxes in that row are contiguous, and except for the top row, the *rightmost* box in each row lies directly underneath the *leftmost* box in the row above it.



The **area** of a border strip is the total number of boxes.
If B is a border strip, its **height** $h(B)$ is the number of rows of B minus 1. The above four border strips have areas 11, 7, 1, 8 and heights 3, 2, 0, 3 respectively.

Border Strip Tableaux

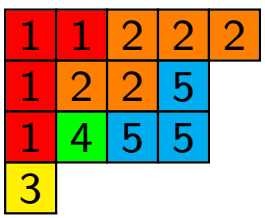
A **composition** is like a partition except that the sequence is not necessarily weakly decreasing.

Border Strip Tableaux

A **composition** is like a partition except that the sequence is not necessarily weakly decreasing.

Let λ be a partition of n and let α be a composition of n.
A **border-strip tableau (BST) of shape λ and type α** is a tiling of the Young diagram of λ with border strips such that

1. the area of the ith border strip is α_i, and
2. if the number i is written in each box of the ith border strip, then the numbers weakly increase across every row and down every column.



$$\lambda = (5, 4, 4, 1) \qquad \text{and} \qquad \alpha = (4, 5, 1, 1, 3)$$

The Murnaghan–Nakayama Rule

Let λ be a partition of n, and let χ_λ be the irreducible character of $\mathfrak{S}_n$ indexed by λ. If $\pi \in \mathfrak{S}_n$ and (α_i) is the sequence of cycle lengths of π, then

$$\chi_\lambda(\pi) = \sum_T \prod_{B \in T} (-1)^{h(B)},$$

where the sum is over all BSTs T of shape λ and type α, and the product is over the border strips B that tile T.

The Murnaghan–Nakayama Rule

Let λ be a partition of n, and let χ_λ be the irreducible character of $\mathfrak{S}_n$ indexed by λ. If $\pi \in \mathfrak{S}_n$ and (α_i) is the sequence of cycle lengths of π, then

$$\chi_\lambda(\pi) = \sum_T \prod_{B \in T} (-1)^{h(B)},$$

where the sum is over all BSTs T of shape λ and type α, and the product is over the border strips B that tile T.

Example. Let $\lambda = (5, 4, 2)$ and $\alpha = (6, 3, 2)$.

$$\chi_\lambda(\pi) = (-1)^1(-1)^0(-1)^0 + (-1)^1(-1)^1(-1)^0 = 0.$$

The Murnaghan–Nakayama Rule (continued)

Example. Let $\lambda = (5, 4, 2)$ and $\alpha = (6, 3, 2)$.

$$\chi_\lambda(\pi) = (-1)^1(-1)^0(-1)^0 + (-1)^1(-1)^1(-1)^0 = 0.$$

The Murnaghan–Nakayama Rule (continued)

Example. Let $\lambda = (5, 4, 2)$ and $\alpha = (6, 3, 2)$.

$$\chi_\lambda(\pi) = (-1)^1(-1)^0(-1)^0 + (-1)^1(-1)^1(-1)^0 = 0.$$

Note that if we were to take $\lambda = (5, 4, 2)$ and $\alpha = (6, 2, 3)$, then there are **no** BSTs of shape λ and type α. Again we conclude $\chi_\lambda(\pi) = 0$, but for a "different reason."

The Murnaghan–Nakayama Rule (continued)

Example. Let $\lambda = (5, 4, 2)$ and $\alpha = (6, 3, 2)$.

$$\chi_\lambda(\pi) = (-1)^1(-1)^0(-1)^0 + (-1)^1(-1)^1(-1)^0 = 0.$$

Note that if we were to take $\lambda = (5, 4, 2)$ and $\alpha = (6, 2, 3)$, then there are **no** BSTs of shape λ and type α. Again we conclude $\chi_\lambda(\pi) = 0$, but for a "different reason."

In what follows, we take queries to be *compositions* α rather than *permutations*.

The Forward Pass

Key Observation: There cannot exist a BST of shape λ and type α if $\alpha_1 > h_1$, because the 1st border strip would be simply too large to fit inside the Young diagram.

The Forward Pass

Key Observation: There cannot exist a BST of shape λ and type α if $\alpha_1 > h_1$, because the 1st border strip would be simply too large to fit inside the Young diagram.

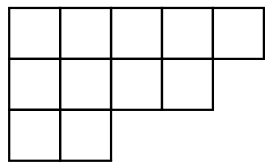
Suppose for example that $n = 10$. Then we make the following successive queries:

$$\alpha = (10)$$
$$\alpha = (9, 1)$$
$$\alpha = (8, 1, 1)$$
$$\alpha = (7, 1, 1, 1)$$
$$\dots$$

Then $f(\alpha) = 0$ if $\alpha_1 > h_1$. But when $\alpha_1 = h_1$, there will be a BST whose first border strip covers the 1st principal hook, with everything else covered by singletons. All BSTs will look like this and will have the same sign.

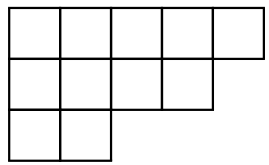
Example

Suppose $n = 11$ and $\lambda = (5, 4, 2)$. Then $h_1 = 7$.



Example

Suppose $n = 11$ and $\lambda = (5, 4, 2)$. Then $h_1 = 7$.



When $\alpha_1 > 7$, we have $\chi_\lambda(\alpha) = 0$. But when we reach $\alpha_1 = 7$, we obtain the following 3 BSTs, which all have the same sign.

1	1	1	1	1
1	2	3	4	
1	5			

1	1	1	1	1
1	2	3	5	
1	4			

1	1	1	1	1
1	2	4	5	
1	3			

The Forward Pass (continued)

To recover h_2, we "freeze" the largest part of α to be h_1, and then try decreasing values for the next largest part of α.

For example, if $n = 20$ and $h_1 = 10$ then we try the queries

$$\begin{aligned} \alpha &= (10, 10) \\ \alpha &= (10, 9, 1) \\ \alpha &= (10, 8, 1, 1) \\ \alpha &= (10, 7, 1, 1, 1) \\ &\dots \end{aligned}$$

until we find α such that $f(\alpha) \neq 0$. (In fact, the first two queries are redundant because necessarily $h_2 \leq h_1 - 2$.)

The Backward Pass

During the backward pass, we recover the shapes of the principal hooks (not just their sizes, which we know from the forward pass) one at a time, starting with the innermost principal hook.

The Backward Pass

During the backward pass, we recover the shapes of the principal hooks (not just their sizes, which we know from the forward pass) one at a time, starting with the innermost principal hook.

It turns out that one can reduce to the case where we know the entire shape except for the first principal hook.

Two Useful Facts

Lemma 1. A border strip of a BST cannot contain more than one box on the principal diagonal.

Two Useful Facts

Lemma 1. A border strip of a BST cannot contain more than one box on the principal diagonal.

Proof. If the border strip contains (i, i) then by the definition of a border strip, it cannot contain (j, j) with $j > i$.

Two Useful Facts

Lemma 1. A border strip of a BST cannot contain more than one box on the principal diagonal.

Proof. If the border strip contains (i, i) then by the definition of a border strip, it cannot contain (j, j) with $j > i$.

Lemma 2. For all k, the first k border strips must be entirely contained in the first k principal hooks.

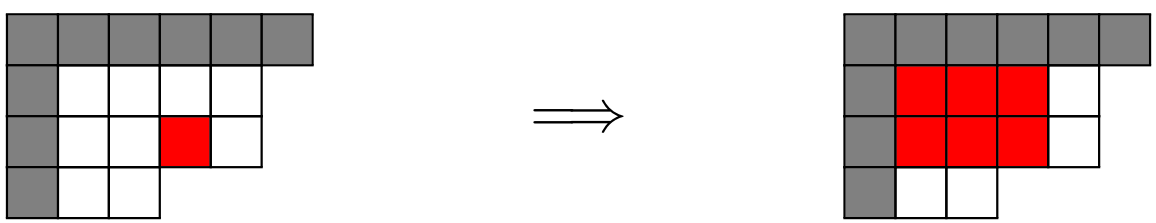
Two Useful Facts

Lemma 1. A border strip of a BST cannot contain more than one box on the principal diagonal.

Proof. If the border strip contains (i, i) then by the definition of a border strip, it cannot contain (j, j) with $j > i$.

Lemma 2. For all k, the first k border strips must be entirely contained in the first k principal hooks.

Proof. Induction on k. If the $(k+1)$st border strip has a box beyond the $(k+1)$st principal hook then it has to contain $(k+2, k+2)$ and $(k+1, k+1)$, contradicting Lemma 1.



Determining Overhangs

Assume we know the principal hook lengths h_i and we want to determine the arm and leg overhang lengths of the first principal hook.

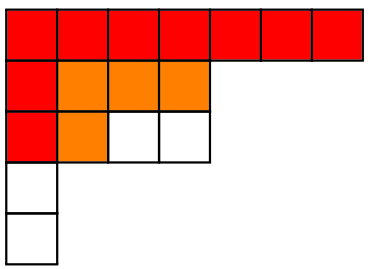
We apply the following sequence of queries until we encounter a nonzero value of $f(\alpha)$:

$$\alpha = (h_1 - 1, h_2 + 1, h_3, h_4, \ldots, h_k)$$
$$\alpha = (h_1 - 2, h_2 + 2, h_3, h_4, \ldots, h_k)$$
$$\alpha = (h_1 - 3, h_2 + 3, h_3, h_4, \ldots, h_k)$$
$$\alpha = (h_1 - 4, h_2 + 4, h_3, h_4, \ldots, h_k)$$
$$\ldots$$

Why does this work?

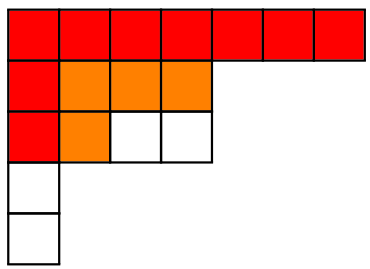
Determining Overhangs (continued)

By Lemma 2, for the 2nd border strip to contain more than h_2 boxes, it must "steal" some boxes from the 1st principal hook. This cannot happen if the 1st border strip is too large, because then it will "block" the 2nd border strip from reaching boxes in the 1st principal hook.



Determining Overhangs (continued)

By Lemma 2, for the 2nd border strip to contain more than h_2 boxes, it must "steal" some boxes from the 1st principal hook. This cannot happen if the 1st border strip is too large, because then it will "block" the 2nd border strip from reaching boxes in the 1st principal hook.



The smallest i that permits border strips of $h_1 - i$ and $h_2 + i$ tells us the length of the shorter overhang. If we know the length of the shorter overhang then we can deduce the length of the longer overhang.

Which Overhang is Which?

Suppose we know that one overhang has length 3 and the other overhang has length 4. There are only two possible shapes, depending on which one is the arm overhang and which one is the leg overhang. We call these two shapes λ and $\hat{\lambda}$, and say that $\hat{\lambda}$ is the **doppelgänger** of λ.

Which Overhang is Which?

Suppose we know that one overhang has length 3 and the other overhang has length 4. There are only two possible shapes, depending on which one is the arm overhang and which one is the leg overhang. We call these two shapes λ and $\hat{\lambda}$, and say that $\hat{\lambda}$ is the **doppelgänger** of λ.

Distinguishing λ from $\hat{\lambda}$ surprisingly subtle question and more than half our paper is devoted to solving it on a case-by-case basis.

The general strategy is to devise α that "barely work" in the sense that there are very few BSTs of type α. We can then analyze exactly what the BSTs must look like and deduce what λ must be.

Concluding Remarks

Can a more efficient algorithm be found?

Empirically, permutations that consist mostly of fixed points are good distinguishers. Enumerating the corresponding BSTs leads naturally to enumerating skew tableaux.

Concluding Remarks

Can a more efficient algorithm be found?

Empirically, permutations that consist mostly of fixed points are good distinguishers. Enumerating the corresponding BSTs leads naturally to enumerating skew tableaux.

Skew tableaux are enumerated by the **Naruse hook-length formula** which generalizes the Frame–Robinson–Thrall hook-length formula. The trouble is that it seems hard to prove that various alternating sums of hook-length formulae cannot "accidentally" result in the same value.

If this could be done, then the number of queries might be drastically reduced from $O(n)$, perhaps even to $O(1)$.