THE HIDDEN STRUCTURE OF LONGEST INCREASING SUBSEQUENCES

RYAN BANSAL

Abstract. In this exposition, we explore the rich combinatorial and probabilistic structure underlying the longest increasing subsequences (LIS). Starting with algorithms foundations such as patience sorting, we build toward deeper results through the Robinson–Schensted correspondence, Young tableaux, and Greene's theorem. We then analyze the hook-length formula and Plancharel measure to study asymptotic behavior of LIS lengths. Our journey culminates in the Ulam-Hammersley problem, showing that the expected LIS length in a random permutation of size n is asymptotically $2\sqrt{n}$, and the remarkable limit shape theorem, which shows the asymptotics of Young tableaux. Along the way, we highlight key ideas from a wide range of areas in an accessible and structured way.

1. Introduction

The longest increasing subsequence (LIS) reveals intrinsic order within seemingly chaotic data. Understanding this fundamental structure has profound implications across diverse fields and provides a window into the hidden patterns that govern complex systems.

The study of longest increasing subsequences extends far beyond pure mathematics, serving as a powerful tool for extracting hidden order from random or noisy sequences across many fields. In financial market analysis, LIS algorithms identify sustained periods of growth within volatile price movements, helping detect trends while filtering out short term fluctuations, presented in [LZZZ16]. In bioinformatics and gene sequencing, LIS algorithms find ordered patterns crucial for understanding evolutionary relationships. The LIS length also serves as a "thermometer for disorder" where it can quantitatively measure how organized or chaotic a sequence is. A short LIS indicates a highly jumbled sequence, while a long LIS suggests almost sorted data. This property makes LIS valuable for anomaly detection and data preprocessing in machine learning. More broadly, LIS provides a robust feature for summarizing the essential structure of complex sequential data, helping reduce noise while preserving critical ordering information in applications ranging from speech recognition to time series analysis.

This practical need to quantify order within disorder gained urgency in the 1960s as computers began processing increasingly large datasets. Questions about measuring how "sorted" or "random" a sequence really is became crucial for algorithm design and data analysis. It was in this context that Stanisław Ulam posed his famous question about the expected behavior of the longest increasing subsequence in a random permutation. His question was motivated by practical concerns about sorting efficiency and understanding data structure, but it opened up connections to many deeper areas of mathematics. The subsequent work revealed that this simple measure of order connects to the geometry of Young tableaux, representation theory, and even the physics of random matrices.

Guided by this perspective, we begin by establishing more precise definitions for our study.

Date: July 13, 2025.

Definition 1.1 (Subsequence)**.** Let S_n denote the group of permutations with length n. If $\sigma \in S_n$ is a permutation, a *subsequence* of σ is a sequence $\sigma(i_1), \sigma(i_2), \ldots, \sigma(i_k)$, where $1 \leq i_1 < i_2 < \cdots < i_k \leq n$. A subsequence is called *increasing* if $\sigma(i_1) < \sigma(i_2) < \cdots < \sigma(i_k)$, and *decreasing* if $\sigma(i_1) > \sigma(i_2) > \cdots > \sigma(i_k)$.

Definition 1.2 (Longest Monotone Subsequence)**.** We define $L(\sigma)$ to be the maximal length of an increasing subsequence of the permutation σ. Similarly, we define $D(\sigma)$ to be the maximal length of a decreasing subsequence of the permutation σ.

To illustrate these concepts, consider the permutation $\sigma = (3, 1, 6, 7, 2, 5, 4)$. We can verify that $L(\sigma) = 3$ by identifying the increasing subsequence $(3, 6, 7)$ shown in red in Figure 1a, while confirming that no increasing subsequence of length 4 exists. Similarly, we can verify that $D(\sigma) = 3$ by examining the decreasing subsequence $(7, 5, 4)$ shown in red in Figure 1b.

$\{3, 1, 6, 7, 2, 5, 4\}$ $\{3, 1, 6, 7, 2, 5, 4\}$

(a) Longest Increasing Subsequence **(b)** A Longest Decreasing Subsequence

Figure 1. Examples of Longest Increasing and Decreasing Subsequences

The true depth of studying longest increasing subsequences emerges from their connections with other areas of mathematics. The field was transformed by the discovery of the Robinson-Schensted correspondence, which creates a bijection between any permutation and a pair of Young tableaux of equal shape. This correspondence reveals that the seemingly simple question of finding the longest increasing subsequence is intimately connected to the rich theory of symmetric functions, representation theory, and algebraic combinatorics.

The Robinson-Schensted correspondence not only provides efficient algorithms for computing longest increasing subsequences but also opens the door to understanding the probabilistic behavior of these structures. Through this lens, we can analyze the expected length of the longest increasing subsequence in random permutations and reveal beautiful limit shapes governing the asymptotic behavior of even deeper structures.

In this paper, we embark on a comprehensive journey through the theory of longest increasing subsequences, always guided by the question of how to measure and understand order in chaos. Throughout this exploration, we will use concrete examples, visualizations, and algorithms that make the LIS tangible and intuitive before building up to the abstract structures and asymptotic results. By tracing this journey, we will see how a simple combinatorial question leads to profound connections across mathematics.

2. Patience sorting

We first develop an algorithm called *patience sorting* to determine $L(\sigma_n)$ for a given permutation $\sigma_n \in S_n$. We start with an empty row of spaces, each of which can hold a "stack" of numbers. When we add a number to the array, it will be either added to the top of a stack in the array, or create a new stack at the end of the array. The addition of an element x is governed by two rules

Definition 2.1 (Patience sorting)**.** Patience sorting builds piles using the following rules:

Rule 1 ("New Pile"): If the number is larger than all top cards, start a new pile.

Rule 2 ("Bumping Down"): Otherwise, place the number on the *leftmost* pile whose top card is larger, "bumping down" the rest of the cards in the pile.

The motivation for these rules is to only allow a new pile when the longest increasing subsequence can be extended. We will illustrate this algorithm by taking the following example in Figure 2.

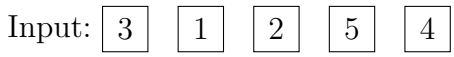


Figure 2. Input permutation: (3, 1, 2, 5, 4)

We add the elements in the sequence in order following Rule 1 because there are no elements. So, 3 will be the first element added to a new pile as seen in Figure 3.

Figure 3. Step 1: Start a new pile with 3

Now, referring to Rule 2, 1 will be added to the top of 3 as it is the leftmost pile with a value greater than 1, bumping it down as seen in Figure 4.

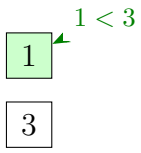


Figure 4. Step 2: Place 1 on pile with 3 (by Rule 2)

When adding 2, we see that it is bigger than the value at the top of all piles (namely 2 is greater than 1), so we follow Rule 1 in Figure 5.

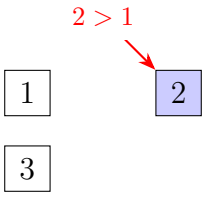


Figure 5. Step 3: 2 starts a new pile since $2 > 1$ (Rule 1)

Again, when adding 5 it is greater than the values at the top of all the piles (1 and 2), so we follow Rule 1 in Figure 6.

When adding 4, it is not larger than the top values of all the piles, so we must refer to Rule 2. We see that 4 is greater than 1 and 2, but less than 5, thus the leftmost pile with a value larger than 4 is the pile with 5. So, we add 4 on that pile as seen in Figure 6.

So, in this example, we end up with 3 stacks after applying patience sorting. Now, if we go back and analyze the original sequence $\sigma = (3, 1, 2, 5, 4)$, we can see that $L(\sigma) = 3$. In fact, the value of $L(\sigma)$ being equal to the number of stacks after patience sorting holds for every permutation.

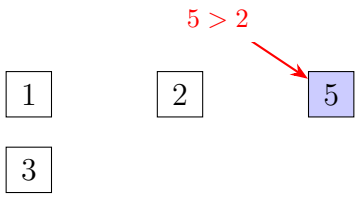


Figure 6. Step 4: 5 starts a new pile since $5 > 2$ (Rule 1)

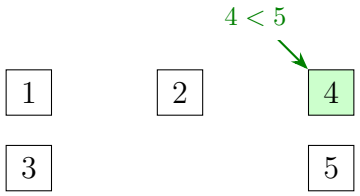


Figure 7. Step 5: Place 4 on top of 5 (by Rule 2)

Theorem 2.2. *For a permutation σ, The amount of stacks left after applying patience sorting is $L(\sigma)$.*

Proof. First, due to the insertion rules in Definition 2.1, the values at the top of the stacks are in increasing order. Additionally, stacks are increasing from top to bottom because of Rule 2, which states that only a value smaller than the current top of a stack can be inserted, pushing the rest of the stack down.

The main claim is that for an increasing subsequence $(\sigma(i_1), \sigma(i_2), \ldots, \sigma(i_k))$, every element must be in a different stack. This is because each successive value $\sigma(i_{j+1})$ is greater than the top values for all the stacks which contain $\sigma(i_1), \ldots, \sigma(i_j)$ due to the stacks being increasing, so $\sigma(i_{j+1})$ will be placed in a stack to the right of the stacks containing all the values in the subsequence before it. This bounds the length of an increasing subsequence to at most the amount of stacks.

To obtain an increasing subsequence with exactly the same length as the amount of stacks, start with the top number in the rightmost stack $x = x_s$, and repeatedly go to the value in the stack to the left that was at the top of the stack at the time when x was added to get $x_{s-1}, x_{s-2}, \ldots, x_1$. Due to the order we chose the values in, we also have that $x_1 < x_2 < \cdots < x_s$, so we are done. ∎

3. A Fast LIS Algorithm

Patience sorting forms stacks $S_1, S_2, \ldots$ by placing each card a_k on the *leftmost* stack whose top card is $\geq a_k$ (creating a new stack if none qualifies). Using this, we see that the index of the stack that receives a_k equals the length of the longest increasing subsequence (LIS) ending at k because all values in later stacks are larger than a_k and cannot be part of the sequence.

Only the top element of each stack influences future placements, so we may compress the stack tops into a single increasing array L and search it with binary search instead of scanning linearly to improve time complexity.

After processing the prefix $a_1, \ldots, a_k$ we maintain:

I $L[1] < L[2] < \cdots < L[|L|]$.

Algorithm 1 Binary-Search LIS

```
L ← ⟨ ⟩                                          ▷ array of current stack tops
for k ← 1 to n do
    if L = ⟨ ⟩ or a_k > L[|L|] then
        append a_k to L                          ▷ new rightmost stack
    else
        i ← min{j : L[j] ≥ a_k}                  ▷ binary search
        L[i] ← a_k                               ▷ place a_k atop stack i
    end if
end for
return |L|                                       ▷ number of stacks = LIS length
```

II $|L| = \ell_k$, the LIS length of the prefix.

III For each $1 \le r \le |L|$, $L[r]$ is the *minimal possible tail* of any increasing subsequence of length r inside the prefix.

Initialization is trivial. We can focus on maintaining the invariants by splitting the insertion into two cases.

Case 1: $a_k > L[|L|]$ (Append).

- **I:** Appending a larger element preserves strict increase.
- **II:** A new stack appears, so $|L|$ increases by one. The sequence of stack tops is itself an increasing subsequence of the new length, showing $|L| \ge \ell_k$. Conversely, any subsequence uses at most one element per stack, so $|L| \le \ell_k$ and equality holds.
- **III:** Tails for lengths $r < |L|$ are unchanged and remain minimal. For $r = |L|$ the unique length-$|L|$ subsequence ends with a_k, so $L[|L|] = a_k$ is minimal by definition.

Case 2: $a_k \le L[|L|]$ (Replace).
Binary search returns the first index i with $L[i] > a_k$ and $L[i-1] < a_k < L[i]$.

- **I:** Replacing $L[i]$ by the smaller value a_k preserves $L[i-1] < L[i]$ and $L[i] < L[i+1]$ (if they exist), so the array stays strictly increasing.
- **II:** No new stack is added, so $|L|$ is unchanged. Because we have not created any longer subsequence, $|L| \ge \ell_k$ still holds, while the "one-element-per-stack" argument gives $|L| \le \ell_k$. Hence $|L| = \ell_k$.
- **III:**
  (a) Lengths $r < i$: tails unchanged, hence minimal.
  (b) Length $r = i$: the new value a_k is *smaller* than the old tail, and every length-i subsequence ending at or before k has tail $\ge a_k$; thus the minimal tail is now $a_k = L[i]$.
  (c) Lengths $r > i$: no subsequence of length r uses $L[i]$ as its tail (that slot belonged to shorter subsequences), so their minimal tails are unchanged.

When $k = n$, II yields the answer L_n. Each of n iterations performs one binary search in an array of size $\le \log n$, so the running time is $O(n \log n)$ with space complexity $O(n)$. The "virtual stacks" grow exactly as in patience sorting, just flattened for speed.

We have shown that we can make Patience Sorting into a fast algorithm to quickly identify the longest increasing subsequence and distill some of the structure in a random permutation.

However, we can take it even further by making a few tweaks to the algorithm to bring out even more information.

4. Robinson-Schensted algorithm

The idea behind the *Robinson-Schensted algorithm* is to recursively apply patience sorting to each "layer" or "level" of the stacks. Whenever an element is added to a stack, we "bump down" the current top value of the stack to the next layer and apply patience sorting recursively to insert the bumped value.

We will illustrate this by looking at an example of recursive patience sorting on the permutation shown in Figure 8.

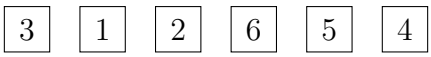


Figure 8. Input permutation: (3, 1, 2, 6, 5, 4)

We first simply add 3 to the first layer as shown in Figure 9.

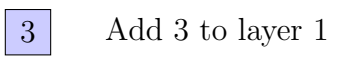


Figure 9. Step 1: Start with 3

Now, we see that 3 is the leftmost number in layer 1 which is greater than 1, so adding 1 will bump down 3 into layer 2. Due to there being no numbers in layer 2, 3 is the largest number and is simply added to the end of the layer as seen in Figure 10.

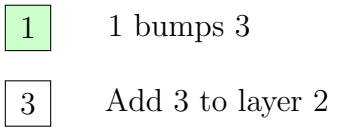


Figure 10. Step 2: Add 1 which bumps 3 to layer 2

When adding 2, it is larger than all numbers in layer 1 (namely 1), so it is simply added to the end of the layer (Figure 11).

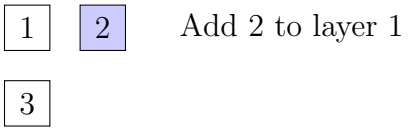


Figure 11. Step 3: 2 is added to layer 1, 3 remains in layer 2

Similarly, 6 is larger than all the current numbers in layer 1, so it is added to the end of the layer (Figure 12)

Because 6 is the leftmost number in layer 1 greater than 5, it is bumped down and replaced by 5. Now, shifting our attention to adding 6 in layer 2, we see that it is the largest number so it is simply added at the end (Figure 13).

Finally, adding 4 leads to a cascade of bumping operations as seen in Figure 14. In layer 1, 5 is the leftmost number greater than 4, so it is bumped down. In layer 2, to add 5 we

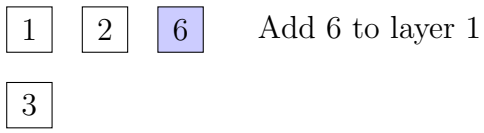


Figure 12. Step 4: 6 is added to layer 1

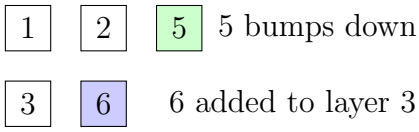


Figure 13. Step 5: 5 bumps 6 which is placed in layer 2

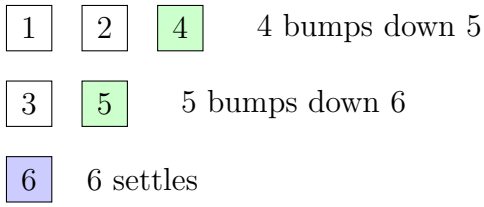


Figure 14. Step 6: Adding 4 leads to recursive bumping in deeper layers

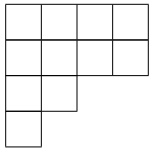
see that the leftmost number greater than it in the layer is 6. So, 5 bumps 6 down to layer 3 where it settles.

This leads to many monotonic sequences in the resulting structure, and the creation of a "*Young tableau*" as described in 5.2.

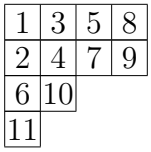
5. Young tableaux

Definition 5.1 (Young diagram)**.** A *Young diagram* is defined as a series of non-increasing numbers $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)$ which represent the length of each row, and the size $n = |\lambda| = \sum \lambda_i$. We also create notation that $\lambda \vdash n$.

Definition 5.2 (Young tableau)**.** A Young diagram from 5.1 filled with the numbers $1, \ldots, n$ such that each row is increasing from left to right, and each column is increasing top to bottom is called a *Young tableau*.



(a) A Young diagram



(b) A Young tableau

Figure 15. Examples of Young diagram and tableau

We call the figure depicted in Figure 14 as P, created from any general permutation by recursively using the patience sorting method. To add an element x, we look at the leftmost position in layer 1 containing an element y greater than x, then replace y with x and repeat the algorithm for the next layer. If x is greater than the entire row, we simply place it at the end of the row.

Lemma 5.3. *P is a Young tableau.*

Proof. We will use induction for this proof. The base case where only 1 number is in P is trivial. Next, we will prove that if we have a Young tableau, adding a number using recursive patience sorting will preserve that fact.

First, we see that if two adjacent layers a and b have equal length, and an element is popped from layer a at index i, it must be smaller than the element in layer b and index i, so it will be inserted into some index at most i in layer b as seen in Figure 16. Thus, a row cannot be made longer than the row above it and P always maintains a valid shape of a Young diagram. We now only have to prove that the numbers in each row and column form increasing sequences.

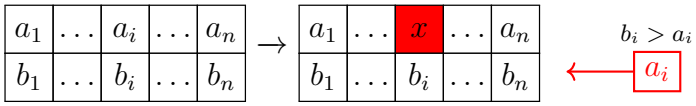


Figure 16. Diagram Preservation

A number is always inserted in a row such that a number to its left is smaller and the number to its right is larger. So, the increasing nature of the rows is always maintained due to the insertion method.

When a number a is bumped from one row to the next, it always ends up in a position in the row below which is either directly below its original position, or further to the left. So, a is either directly under the number b which displaced it, which is smaller than a, or a number to the left of b, which is less than b and consequently less than a. Hence, the insertion method also maintains the increasing nature of the rows.

So, we have shown that the insertion method for recursive patience sorting maintains the increasing nature of the rows and columns. Thus, combined with the base case we have proven that P is always a Young tableau. ■

To complete the bijection, we have to construct another Young tableau Q by recording the order of when spots in the tableau were added. For example, in Figure 13, 6 is pushed down into the second index of layer 2 as the 5th operation. Thus, the $(2, 2)$ cell of Q has a value of 5, signifying it was the 5th cell to obtain a number. In a sense, we can think of Q as the "recording" tableau, which keeps track of when moves took place.

Lemma 5.4. *Q is a Young tableau.*

Proof. To insert a value into P, we start at the first layer and either replace a value and pop it to the next layer, which does not add any new cells to the layer, or we add the value to the end of the layer, which adds a new cell and also terminates the process. So, there is always exactly one new cell created with each insertion, and it is at the end of a layer.

Going back to Q, we see that it has the same shape as P because it is simply recording the order of when each cell in P was filled. Additionally, each digit added to Q is always larger than all the previous digits, and in particular larger than all the digits above it and to its left. So, rows and columns will always contain strictly increasing sequences. ■

The motivation behind creating P and Q for a bijection is because we can use them to recreate the original permutation by "deletion steps", where the information in Q is used in conjunction with the information in P to remove one element at a time from P, reconstructing the sequence.

Lemma 5.5. *A sequence is uniquely determined by any pair of Young tableaux of the same shape.*

Proof. The position of the largest element in Q tells us which number was added on to a row of P. This number, call it x, must have been displaced from the previous row by the largest number which is smaller than it. There will always be at least one number smaller than x in the previous row because the one directly above x is smaller. The number in the previous row must have also been displaced from the next row up and so on. In the end, we get to the first row and find what number was inserted into it. We also know what P and Q were before the digit was inserted. This "deletion" step has in turn reversed the process of inserting a digit. We can repeat this procedure to find every digit of the sequence, sometimes called the *inverse Robinson-Schensted algorithm.* ■

Lemma 5.3 and Lemma 5.4 show that a sequence uniquely determines two Young tableaux named P and Q, and Lemma 5.5 shows the other direction where any two Young tableaux of the same shape uniquely determine a sequence. Thus, we have completed the proof for the Robinson-Schensted correspondence.

Theorem 5.6 ([Sch61])**.** *The Robinson-Schensted algorithm creates a mapping that takes a permutation $\sigma \in S_n$ and returns a triple (λ, P, Q) where $\lambda \vdash n$, and P and Q are two Young tableaux of shape λ. This creates a bijection between S_n and ordered pairs of Young tableaux of size n and the same shape.*

In accordance with Theorem 2.2, we also obtain the following theorem relating LIS to tableau shape.

Theorem 5.7. *$L(\sigma)$ is the length of the first row of P since it is equivalent to the number of stacks in patience sorting.*

6. Greene's Theorem

Take a random permutation

$$\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$$

Running Theorem 5.6 on σ produces (λ, P, Q) with $\lambda = (\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_m)$. Suppose we are allowed to pick k increasing subsequences that do not share an index. We aim to determine the maximum number of indices that such a set can cover. Formally we define

$$L_k(\sigma) = \max\Big\{ |\, Y_1 \cup \cdots \cup Y_k| : Y_i \text{ increasing},\ Y_i \cap Y_j = \varnothing \Big\},$$

where Y_i represent the disjoint increasing subsequences. Now, using the result from [Gre74] which extends the thoughts of Schensted for finding $L(\sigma)$ yields the following theorem.

Theorem 6.1 (Greene's Theorem)**.**

$$L_k(\sigma) \ = \ \lambda_1 + \lambda_2 + \cdots + \lambda_k.$$

Proof. The proof requires establishing inequalities in both directions.

For the upper bound, we use a pigeonhole argument on the columns of the insertion tableau P. A key property of the Robinson-Schensted correspondence, is that any two elements in the same column of P must form a increasing subsequence in σ because the index of x in σ is greater than the index of y if x appears above y. Therefore, an increasing subsequence

(where indices and values both increase) can contain at most one element from any given column.

A set of k disjoint increasing subsequences can therefore select at most k elements from any given column. The total number of elements in their union is thus bounded by the sum, over all columns, of the minimum of k and the column's height:

$$L_k(\sigma) \leq \sum_{\text{columns } j} \min(k, \lambda'_j)$$

where λ'_j is the height of column j. This sum is a standard combinatorial identity for the number of cells in the first k rows, $\sum_{i=1}^{k} \lambda_i$, establishing the upper bound.

For the lower bound, we must construct k disjoint increasing subsequences with a total of $\sum_{i=1}^{k} \lambda_i$ elements. While the elements in a row of P are sorted by value, their original indices are not necessarily sorted. However, a deeper result from Greene's analysis shows that the elements corresponding to each row can be arranged to form a true increasing subsequence of σ.

The constructive proof involves using the reverse Robinson-Schensted algorithm on the first k rows of the (P, Q) tableaux. This procedure reconstructs the parts of the original permutation corresponding to these cells, and in doing so, naturally partitions them into k disjoint increasing subsequences. The total number of elements recovered is the number of cells in the first k rows, $\lambda_1 + \cdots + \lambda_k$, which gives the lower bound and completes the proof. ■

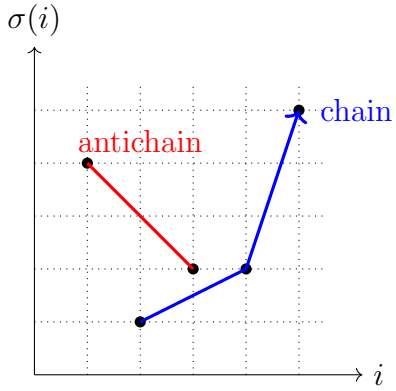
The statement can be reformulated in the language of partially ordered sets. We plot the permutation as dots at integer points

$$\big(i,\ \sigma(i)\big), \qquad 1 \leq i \leq n,$$

so index grows to the right and value grows upward. Reading "above-and-to-the-right" as the relation

$$(i, \sigma(i)) \ \prec\ (j, \sigma(j)) \quad \Longleftrightarrow \quad i < j \text{ and } \sigma(i) < \sigma(j),$$

an *increasing subsequence* is a path that steps north-east through the dot, which is also called a *chain*. A set of dots no two of which lie in that order is an *antichain*, forming a south-east staircase. These represent *decreasing subsequences*.



To see Greene's theorem in action, take

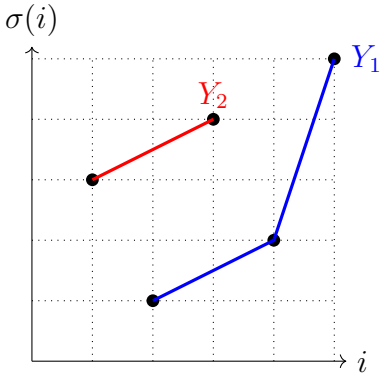
$$\sigma = (3, 1, 4, 2, 5)$$

Inserting the values using the Robinson-Schensted algorithm gives an insertion tableaux shown in Figure 17a with a shape of $\lambda = (3, 2)$. Greene's theorem predicts that two disjoint increasing subsequences can cover 5 indices because $3 + 2 = 5$, and indeed the sequences $(1, 2, 5)$ and $(3, 4)$ satisfy that. The dot diagram in Figure 17b highlights those chosen positions as chains.

$P =$

1	2	5
3	4	

(a) Insertion tableau of σ



(b) Dot diagram of σ

Figure 17. A concrete example of Greene's theorem

7. The Hook-Length

Going back to Theorem 5.6, we can get a very interesting sum. For all $\lambda \vdash n$, say that d_λ is the number of Young tableau of the shape λ. We get that

$$\sum_{\lambda \vdash n} d_\lambda^2 = n!$$

.

This also implies that the probability of a random permutation having insertion tableau shape λ after Robinson-Schensted is the following distribution.

$$\mathbb{P}(\lambda^{(n)} = \lambda) = \frac{d_\lambda^2}{n!}$$

Definition 7.1 (Plancherel measure)**.** The probability measure on partitions of n that assigns probability $\frac{d_\lambda^2}{n!}$ to any partition $\lambda \vdash n$, where d_λ is the number of standard Young tableaux of shape λ.

The clear next question, now that we have a probability measure on partitions of n, is to calculate the actual distributions, i.e., find the values of d_λ for any shape λ.

Definition 7.2 (hook-length)**.** The hook-length of a position $h_\lambda(i, j)$ is the amount of cells in the hook $H_\lambda(i, j)$, which consists of cells (i', j') where $i' = i$ and $j' \geq j$, or vice versa as shown in Figure 18.

We will now derive a formula for d_λ, the amount of Young tableaux of a specific shape λ, using a probabilistic proof outlined in [GNW79].

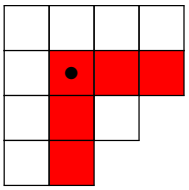


Figure 18. $H_\lambda(2,2)$ depicted in red for the Young diagram $\lambda = (4,4,3,2)$

Theorem 7.3 (hook-length formula)**.**

$$d_\lambda = \frac{n!}{\prod_{(i,j)} h_\lambda(i,j)}$$

First, define the "corners" of the Young tableau to be cells which are the last in their row and their column as can be seen in Figure 19

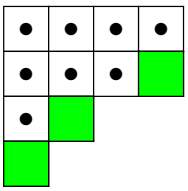


Figure 19. Corners of Young tableau shown in green

We develop the notation $\mu \nearrow \lambda$ indicating that μ can be obtained by removing a corner cell of λ. Due to the fact that the maximum element n always appears in a corner cell of λ, we can make a clear recurrence of

$$d_\lambda = \sum_{\mu \nearrow \lambda} d_\mu$$

and $d_\emptyset = 1$. So, our goal shifts now to show the RHS of 7.3, which we will denote as e_λ, satisfies the same recurrence.

We can make this recurrence a more probabilistic form by trying to prove

$$\sum_{\mu \nearrow \lambda} \frac{e_\mu}{e_\lambda} = 1$$

To construct this probability measure on the Young diagrams, we will create a process called the hook walk depicted in Figure 20.

Definition 7.4 (hook walk)**.** We aim to create a natural distribution of Young diagram corners.

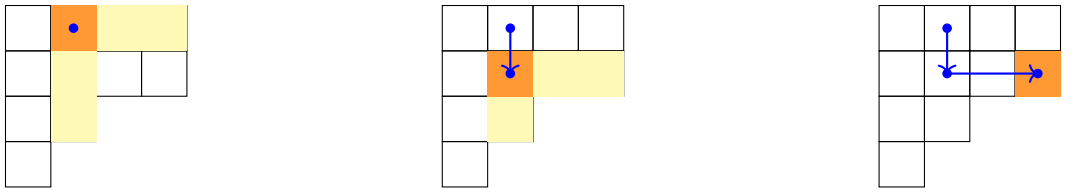
Step 1: choose a cell (i,j) in λ uniformly at random
Step 2: repeatedly choose successor cell to the current cell in the hook $H_\lambda(i,j)\backslash\{(i,j)\}$.

Denote the terminating corner of the hook walk as $\mathbf{c}$.

Lemma 7.5. *For any corner cell (a,b) of λ, we have that*

$$\mathbb{P}(\mathbf{c} = (a,b)) = \frac{e_\mu}{e_\lambda}$$

where $\mu = \lambda\backslash\{(a,b)\}$



(a) Start $(1,2)$ **(b)** Step to $(2,2)$ **(c)** End at $(2,4)$

Figure 20. Hook walk in the Young tableau $(4,4,2,1)$ ending at $\mathbf{c} = (2,4)$

Proof. First, we can simplify the RHS by simply canceling common terms with e_μ and e_λ

$$\frac{e_\mu}{e_\lambda} = \frac{1}{n} \prod_{1 \le i < a} \frac{h_\lambda(i,b)}{h_\lambda(i,b)-1} \prod_{1 \le j < b} \frac{h_\lambda(a,j)}{h_\lambda(a,j)-1} \tag{7.1}$$

$$= \frac{1}{n} \prod_{1 \le i < a} \left(1 + \frac{1}{h_\lambda(i,b)-1}\right) \prod_{1 \le j < b} \left(1 + \frac{1}{h_\lambda(a,j)-1}\right) \tag{7.2}$$

Now, for a hook walk $(x,y) = (x_1,y_1) \to \cdots \to (x_k,y_k) = (a,b)$ we define the vertical and horizontal projections to be the sets $X = \{x_1, x_2, \ldots, x_k\}$ and $Y = \{y_1, y_2, \ldots, y_k\}$. Now, we can analyze the probability $p(X, Y \mid x, y)$ that a hook walk starting at (x,y) has the vertical and horizontal projections of X and Y. We will prove the following using induction.

Lemma 7.6.

$$p(X, Y \mid x, y) = \prod_{i \in A \setminus \{a\}} \frac{1}{h_\lambda(i,b)-1} \prod_{j \in B \setminus \{b\}} \frac{1}{h_\lambda(a,j)-1}$$

The base case where $A = \emptyset$ or $B = \emptyset$ is trivial because you only have one way to move.

At each cell, we can either move horizontally to the next x value, or the next y value, so we have the recursion

$$p(X, Y \mid x, y) = \frac{1}{h_\lambda(x,y)-1}(p(X \setminus \{x_1\}, Y \mid x_2, y_1) + p(X, Y \setminus \{y_1\} \mid x_1, y_2)).$$

Writing the RHS of the lemma we are trying to prove as $\prod$, we see that the recursion simplifies to

$$p(X, Y \mid x, y) = \frac{1}{h_\lambda(x,y)}((h_\lambda(x_1,b) - 1 + h_\lambda(a,y_1) - 1).$$

But $h_\lambda(x,y) - 1 = (h_\lambda(x_1,b) - 1) + (h_\lambda(a,y_1) - 1)$ as illustrated by the following diagram, proving the lemma.

Continuing on, the probability for the end cell can be done by summing the conditional probabilities with respect to the first cell, and then, for all possible vertical and horizontal projections. This gives

$$p(a,b) = \frac{1}{n} \sum_{\substack{X \subseteq \{1,\ldots,a\} \\ Y \subseteq \{1,\ldots,b\}}} p(X, Y \mid \min X, \min Y) \tag{7.3}$$

$$= \frac{1}{n} \prod_{1 \le i < a} \left(1 + \frac{1}{h_\lambda(i,b) - 1}\right) \prod_{1 \le j < b} \left(1 + \frac{1}{h_\lambda(a,j) - 1}\right). \tag{7.4}$$

■

The hook walk also gives us a way to create a uniformly random Young tableau on a diagram of shape λ. We simply repeatedly run the hook walk with a random cell each time, and the corner cell it finishes on is filled with numbers starting from n and going down, then removed from the diagram to repeat the algorithm.

8. Asymptotics

So far, we have focused on understanding the structure of individual permutations through their longest increasing subsequences and corresponding Young tableaux. Now we turn to a fundamental question about the collective behavior of all permutations of a given size. In particular, what is the expected length of the longest increasing subsequence in a random permutation?

This question was first posed in [Ula61] by Stanisław Ulam, launching what would become known as the Ulam-Hammersley problem. To understand the nature of this problem, let us begin with a concrete example.

Example. For $n = 3$, there are $3! = 6$ permutations. Let us compute $L(\sigma)$ for each:

Permutation	**Longest Increasing Subsequence**	**Length**
$(1,2,3)$	$(1,2,3)$	3
$(1,3,2)$	$(1,3)$ or $(1,2)$	2
$(2,1,3)$	$(1,3)$ or $(2,3)$	2
$(2,3,1)$	$(2,3)$	2
$(3,1,2)$	$(1,2)$	2
$(3,2,1)$	any single element	1

The sum of all LIS lengths is $3 + 2 + 2 + 2 + 2 + 1 = 12$, giving us an expected value of $\mathbb{E}[L_3] = 12/6 = 2$.

This example illustrates the computational challenge: even for small n, determining the expected LIS length requires examining all $n!$ permutations, a task that quickly becomes intractable.

Definition 8.1 (The Ulam-Hammersley Problem)**.** For a uniformly random permutation $\sigma \in S_n$, we define

$$\ell_n = \mathbb{E}[L(\sigma)] = \frac{1}{n!} \sum_{\sigma \in S_n} L(\sigma)$$

as the expected length of the longest increasing subsequence.

John Hammersley's 1972 paper [Ham72] provided the first major insight into the asymptotic behavior of ℓ_n by finding bounds. He was also able to rigorously prove that $\ell_n \sim c\sqrt{n}$ for some constant c.

The Robinson-Schensted correspondence reveals that the length of the longest increasing subsequence in a permutation equals the length of the first row of its corresponding Young tableau. This connection allows us to reformulate the Ulam-Hammersley problem in terms of Young tableaux.

By Theorem 5.6 and Theorem 5.7, we can express the expected LIS length using the Plancherel measure from Definition 7.1:

Lemma 8.2. *The expected length of the longest increasing subsequence equals*

$$\ell_n = \sum_{\lambda \vdash n} \lambda_1 \cdot \mathbb{P}(\lambda)$$

where $\mathbb{P}(\lambda) = \frac{d_\lambda^2}{n!}$ *is the Plancherel probability.*

To understand ℓ_n, we need to identify which shapes are most probable. Using Theorem 7.3 and working in logs, we can express:

$$\log \mathbb{P}(\lambda) = \log n! - \log \left(\prod_{(i,j)\in\lambda} h_\lambda(i,j)^2 \right) \tag{8.1}$$

$$= \log n! - \sum_{(i,j)\in\lambda} \log(h_\lambda(i,j)^2) \tag{8.2}$$

$$= \log n! - 2 \sum_{(i,j)\in\lambda} \log h_\lambda(i,j) \tag{8.3}$$

Since $\log n!$ is constant for fixed n, maximizing $\mathbb{P}(\lambda)$ is equivalent to minimizing

$$S(\lambda) = \sum_{(i,j)\in\lambda} \log h_\lambda(i,j)$$

This sum can be interpreted as an "energy" of the shape, and shapes with lower energy are more probable.

For large n, we expect the shape of a typical Young tableau to stabilize in some sense. To make this precise, we need to view tableaux shapes as continuous objects. We rescale the tableau so that it has a fixed size as $n \to \infty$.

Definition 8.3 (Scaled Tableau)**.** Given a tableau of shape λ and size n, we scale both axes by $\frac{1}{\sqrt{n}}$, mapping cell (i,j) to $\left(\frac{i}{\sqrt{n}}, \frac{j}{\sqrt{n}}\right)$.

Under this scaling, each of the n cells has area $\frac{1}{n}$, so the total area remains at 1. To simplify analysis, we also rotate by 45 degrees using:

$$u = \frac{x+y}{\sqrt{2}}, \quad v = \frac{y-x}{\sqrt{2}}$$

This transforms the staircase boundary into a more symmetric curve. In the continuous limit, the discrete energy $S(\lambda)$ can be expressed as an integral.

Lemma 8.4 (Continuous Energy Functional)**.** *As $n \to \infty$, the energy converges to*

$$J[\omega] = \int_{-2}^{2} \int_{-2}^{2} \log \left| \frac{\omega(s) - \omega(t)}{s - t} \right| \, ds \, dt$$

where $\omega(t)$ describes the rotated boundary.

The term $\frac{\omega(s)-\omega(t)}{s-t}$ represents the slope between boundary points. However, we must be careful because not every ω corresponds to a valid tableau, so we need a constraint from the Young diagram properties.

Definition 8.5 (Lipschitz Constraint)**.** A function $\omega : [-2, 2] \to \mathbb{R}$ represents a valid tableau boundary if

$$|\omega'(t)| \leq 1 \quad \text{for all } t$$

This constraint ensures rows decrease in length. If the boundary rises at a slope greater than 1, it would violate the diagram property.

We can now ask among all functions ω satisfying the Lipschitz constraint, which one maximizes $J[\omega]$? Using calculus of variations we can find the functions that optimize integral expressions.

Lemma 8.6 (Optimal Shape)**.** *The maximizing function has derivative*

$$\omega'(t) = \frac{2}{\pi} \arcsin\left(\frac{t}{2}\right)$$

Proof. To find the optimal ω, we use Lagrange multipliers, leading to an equation involving the "functional derivative" of J.

The calculation shows that the optimal ω satisfies a certain integral equation. Remarkably, this equation has an explicit solution: $\omega'(t) = \frac{2}{\pi} \arcsin(t/2)$.

We can verify the Lipschitz constraint. Since $|\arcsin(t/2)| \leq \pi/2$ for $|t| \leq 2$, we have $|\omega'(t)| \leq \frac{2}{\pi} \cdot \frac{\pi}{2} = 1$. ∎

Now we can find the explicit form of the limit shape by integrating:

$$\omega(t) = \int_0^t \frac{2}{\pi} \arcsin\left(\frac{s}{2}\right) \, ds \tag{8.4}$$

$$= \frac{2t}{\pi} \arcsin\left(\frac{t}{2}\right) + \frac{2}{\pi}\sqrt{4-t^2} - \frac{4}{\pi} \tag{8.5}$$

Where the last term serves only as a vertical shift and does not affect the shape. Thus, we finally get an exact form for the limiting shape of a random Young tableau, which was proven in the pair of papers [LS77] and [VK77].

Theorem 8.7 (Limit shape theorem)**.** *Young tableaux under the Plancherel measure concentrate around the limit shape as depicted in Figure 21*

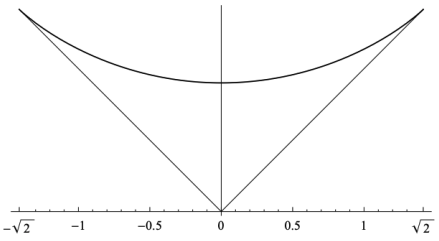
$$\boxed{f_*(x) = \frac{2}{\pi}\left(x \arcsin\left(\frac{x}{2}\right) + \sqrt{4-x^2}\right)}$$

for $|x| \leq 2$*, and* $f_*(x) = 0$ *for* $|x| > 2$*.*

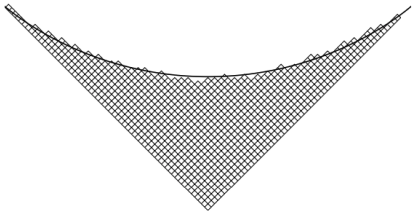
The concentration phenomenon means that for large n, almost all Young tableaux have shapes very close to this limit curve.

Theorem 8.8 (Solution to Ulam-Hammersley)**.** *For a uniformly random permutation* σ *of length* n*,*

$$\mathbb{E}[L(\sigma)] = 2\sqrt{n} + o(\sqrt{n}) \quad \text{as } n \to \infty$$



(a) The limit shape curve.



(b) A large random Young diagram follows the limit shape.

Figure 21. Asymptotics of the Young diagram

Proof. The first row length λ_1 corresponds to the distance along the diagonal to the curve. In the simplified coordinates, this occurs at $(\sqrt{2}, \sqrt{2})$, which corresponds to a length of $2\sqrt{n}$ after scaling back to normal coordinates. ■

This remarkable result shows that random permutations contain a predictable amount of structure. The coefficient 2 is naturally embedded in the side length of the expected Young tableau, and gives us the solution to the variational problem. As a byproduct, we are also left with the fascinating limit shape, embedding so much more information about random permutations.

9. Conclusion

Our journey through the theory of longest increasing subsequences has revealed remarkable connections spanning diverse areas of combinatorics. What began as a simple question about the length of the longest increasing subsequence has led us to deep results about the asymptotic behavior of random structures and the beautiful geometry of limit shapes.

The Robinson-Schensted correspondence stands as the central pillar of this theory, providing not only efficient algorithms for computing longest increasing subsequences but also a profound connection between the discrete world of permutations and the rich algebraic structure of Young tableaux. Through this correspondence, we have seen how the seemingly chaotic behavior of random permutations gives rise to predictable patterns governed by elegant laws.

The theory naturally extends in several fascinating directions. For example, the study of alternating subsequences, patterns that alternate between increasing and decreasing steps, connects longest increasing subsequences to pattern avoidance in permutations and leads to surprising connections with Catalan numbers and Dyck paths, described deeply in [CK23].

Another rich direction involves the longest common subsequence (LCS) problem, which seeks the longest subsequence shared by multiple sequences. While computing LIS takes $O(n \log n)$ time, finding the LCS for m sequences of length n requires exponential time and becomes NP-hard for multiple sequences. With applications ranging from DNA sequence analysis to plagiarism detection in text, studying this problem is a natural topic for future research.

Additionally, the fluctuations of LIS lengths around their expected value can also be modeled following the Tracy-Widom distribution, originally discovered in random matrix theory. This connection reveals that the same mathematical structures governing eigenvalue distributions in random matrices also control the behavior of combinatorial objects

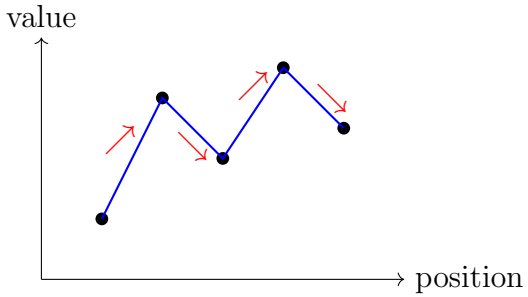


Figure 22. An alternating sequence showing the up-down pattern

like permutations. The limit shape theorem itself connects to integrable partial differential equations and variational principles, showing how continuous mathematics emerges from discrete combinatorial problems.

These connections suggest that longest increasing subsequences occupy a special position in the mathematical landscape. They are simple enough to be understood by many people concretely, yet rich enough to reveal deep mathematical truths for those who study them. The story of longest increasing subsequences is ultimately a story about a simple combinatorial question which can lead to profound insights that illuminate the deep structures underlying seemingly distant topics.

10. Acknowledgements

The author would like to thank Dean Menezes for his guidance and mentorship throughout the process of writing this paper. The author would also like to thank Simon Rubinstein-Salzedo for his lectures on paper writing, as well as creating an environment that inspired the author to delve into mathematical paper writing.

References

[CK23] Johann Cigler and Christian Krattenthaler. Bounded dyck paths, bounded alternating sequences, orthogonal polynomials, and reciprocity, 2023.

[GNW79] Curtis Greene, Albert Nijenhuis, and Herbert S Wilf. A probabilistic proof of a formula for the number of young tableaux of a given shape. *Advances in Mathematics*, 31(1):104–109, 1979.

[Gre74] Curtis Greene. An extension of schensted's theorem. *Advances in Mathematics*, 14(2):254–265, 1974.

[Ham72] J. M. Hammersley. A few seedlings of research. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics*, pages 345–394. University of California Press, 1972.

[LS77] B.F Logan and L.A Shepp. A variational problem for random young tableaux. *Advances in Mathematics*, 26(2):206–222, 1977.

[LZZZ16] Youhuan Li, Lei Zou, Huaming Zhang, and Dongyan Zhao. Computing longest increasing subsequence over sequential data streams, 2016.

[Sch61] C. Schensted. Longest increasing and decreasing subsequences. *Canadian Journal of Mathematics*, 13:179–191, 1961.

[Ula61] Stanislaw M. Ulam. Monte carlo calculations in problems of mathematical physics. In *Modern Mathematics for the Engineers*, pages 261–281. McGraw-Hill Book Co., Inc., New York-Toronto-London, 1961.

[VK77] A. M. Vershik and S. V. Kerov. Asymptotics of the plancherel measure of the symmetric group and the limiting form of young tableaux. *Dokl. Akad. Nauk SSSR*, 233(6):1024–1027, 1977. English translation in *Soviet Math. Dokl.* **18** (1977), 527–531.