Binary trees using the bookshelf and baseball constructions

Kiyoshi Igusa

Email address: igusa@brandeis.edu
Contents

Dyck paths and their bijections
   Bradley Kaplan  
   1

Relationship between binary trees and Young diagrams: a story of bookshelves
   Liang Jinlong and Serra Pelin  
   5

A bijection between binary trees and torsion classes
   Yicheng Tao  
   22

Torsion classes and binary trees
   Max Weinstein  
   26

Young diagrams with gaps and torsion classes
   Ruiyang Hu  
   30

213-avoiding permutations and binary trees
   Zhaonan Li and Michael Richard  
   40
Preface

This volume includes the results of a Fall 2019 undergraduate course at Brandeis University called “Introduction to Mathematical Research”. We studied bijections between various well-known sets having a Catalan number of objects, such as binary trees, Young diagrams, torsion classes and 213-avoiding permutations.

The n-th Catalan number is \(C_n = \frac{1}{n+1} \binom{2n}{n}\). There are a large number of sets having a Catalan number of objects \([1]\). One of these is the set of magma which are patterns of association such as

\[(\bullet \bullet) (\bullet \bullet)\]

There are \(C_n\) magma of size \(n + 1\). For example, there are \(C_2 = 2\) ways to associate three items: \((\bullet \bullet) \bullet\) and \(\bullet (\bullet \bullet)\). These form what is known as the Tamari lattice which is a partial ordering on the collection of magma where the “covering relation” (See Definition A1) is given by shifting one set of parentheses to the right. For example

\[(\bullet \bullet \bullet) (\bullet)\] covers \((\bullet \bullet) (\bullet \bullet)\)

A maximal chain in the Tamari lattice is defined to be a maximal sequence of covering relations for magma of a fixed size. It is a famous open problem to give the formula for the number of such chains.

Recently, Nelson \([2]\) gave a recursive formula for the number of maximal chains in the Tamari lattice. To do this he used the correspondence between certain Young diagrams and Dyck paths and the description of the covering relation given in \([3]\).

The Tamari lattice is also isomorphic to the lattice of torsion classes for the path algebra of the quiver of type \(A_n\). (See the papers by Tao and Weinstein for definitions.) Recently, Barnard, Carroll and Zhu \([4]\) have given a description of the covering relation for torsion classes for arbitrary finite dimensional algebras over any field. These papers grew out of our attempt to understand the relationship between these ideas.

In Chapter 1, Bradley Kaplan describes how certain Young diagrams correspond to Dyck paths. In Chapter 2, Liang Jinjong (Olly) and Serra Pelin use what they call the “stacked bookshelf” construction to gives an explicit bijection between binary trees and the Young diagrams considered in Chapter 1.

Next, we recall the definition of Dyck path and discuss the well-known bijections between Dyck paths and binary trees and Nelson’s bijection between Dyck paths and Young diagrams. Nelson’s construction is simply to “fill in” the space above the Dyck path. Olly and Serra show that these two bijections together give the same bijection as their “bookshelf” construction.

Torsion pairs (torsion classes and torsion-free classes) are discussed next. Yicheng Tao and Max Weinstein show the correspondence between torsion pairs and binary trees using bookshelves without “stacking”. Ruiyang Hu (Sam) gives a direct bijection between torsion pairs and Young diagrams by using Young diagrams with gaps which can be pushed together to give standard Young diagrams without gaps.

In the last chapter, Zhaonan Li (Leo) and Michael Richard discuss pattern avoiding permutations. By a 213-avoiding permutation we mean a permutation \(\sigma\) of \(n\) letters so that there do not exist \(1 \leq i < j < k \leq n\) for which \(\sigma(j) < \sigma(i) < \sigma(k)\). Leo and Mike use the baseball construction to give an explicit bijection between torsion classes and 213-avoiding permutations.

We conclude with several diagrams of representations of the Tamari lattice in which the nodes are the various constructions discussed in the contents of these papers.
This chart shows how these papers are related.

Serra
Young diagrams → Bradley

Serra
Young with gaps → Serra, Olly
	Show this “commutes”

Serra
Dyck paths → Bradley

Sam
Young with gaps → Serra, Olly

Olly
binary trees → Olly

Max, Yicheng
torsion classes → Mike, Leo

Mike, Leo
213 permutations → Mike, Leo
Dyck paths and their bijections

Bradley Kaplan

Abstract. The purpose of this chapter is to define, describe, and construct examples of Dyck paths, to give the history of Dyck paths, to give the bijection between Dyck paths and binary trees, and to give the bijection between Dyck paths and Young diagrams. Following the idea of Nelson.

1.0.1. Introduction. We will review the definition of a Dyck path, give some of the history of Dyck paths, and describe and construct examples of Dyck paths.

In the second section we will show, using the description of a binary tree and the definition of a Dyck path, that there is a bijection between binary trees and Dyck paths. In the third section we will give an explicit bijection between Young diagrams and Dyck paths, which will be used in other parts of this paper (written by others).

1.1. Definition of Dyck path.

Definition 0.1. We recall that a Dyck path is a series of equal length steps, going up and right, that form a staircase walk from (0, 0) to (n, n) that will lie strictly above, or touching, the diagonal x = y.

Now we will discuss the history of Dyck paths. We review: the Catalan number is $C_n$.

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

In 1887, Désiré André found that the Catalan number is the number of Dyck paths of length 2n that can be constructed.

The following are some examples of Dyck paths. Note that the line going from left to right must go up 1 unit (represented here by two blocks) before it goes to the right 1 unit. In later sections we will show that binary trees and Young diagrams can be created from these Dyck paths, and Dyck paths can be created from Young diagrams and binary trees.

![Figure 1. Some examples of Dyck paths](image-url)
1.2. Bijection between Dyck and binary trees. Now we will demonstrate that the above Dyck paths can be translated into binary trees and that binary trees can be translated into Dyck paths, thus giving a bijection between the two.

Remark 0.2. We recall that a binary tree is a diagram that starts with one node at the top (called the root), where each node has two children, except for the leaves which have no children and are located at the bottom of the diagram. Also, lines are drawn between parents and children. Given a parent, one of its children is located diagonally down and to the right of the parent while the other child is located down and to the left of the parent.

As shown by Olivier Bernardi, if we are given any binary tree, we can construct a Dyck path from it. We label any node that has two children (a parent) as a "left" equivalent and any leaf (a child that does not have any children) as a "down" equivalent. We start at the top of the binary tree and start reading the equivalents going strictly diagonally down and to the right. When we cannot read in this direction we read strictly to the left, except when we meet a line segment drawn strictly diagonally from the top left direction to the bottom right direction, in which case we jump to the top of that line segment. We start drawing at the point \((n, n)\) (where the Dyck path would finish if it was normally drawn) and when we read a "left" label we draw a line segment of a single unit’s length going to the left. When we read a "down" label we draw a line segment of a single unit’s length going down the page. We do not read the last node’s label. An example of this labeling of a binary tree and the construction of a Dyck path from it is shown in the rightmost pair of a binary tree and a Dyck path in Figure 2, where "left" is replaced with "L" and "down" is replaced with "D".

Similarly, if we are given any Dyck path, we can construct a binary tree from it. We start tracing the Dyck path at the point \((n, n)\) and follow the line to \((0, 0)\). If the line in the Dyck path goes one unit to the left we, starting at the parent node of the binary tree, draw a line segment of a single unit’s length going down and to the right with a node at the end of it (that can become a parent or an leaf). If the line in the Dyck path goes one unit down the page we create an leaf in the binary tree (with the last node created becoming an leaf). If the leaf cannot be created without drawing an additional line in the binary tree (the leaf would have no parent), a line is drawn between the leaf and the lowest parent node that does not yet have two children. If the line in the Dyck path goes one unit to the left after the first leaf is created, we draw a line segment of a single unit’s length going diagonally down and to the left with a node at the end of it at the lowest node that is not an leaf that does not have two children and if the line in the Dyck path goes multiple units to the left we draw line segments of a single unit’s length going diagonally down and to the right with a node at the end of it starting at the node created by the initial line segment of one unit’s length going to the left in the Dyck path, one for each additional unit the line in the Dyck path goes to the left beyond the initial unit. All nodes created after the first leaf is created must become parents (they cannot become leaves). Next, we add a leftmost leaf. Finally, we need to extend all lines in the binary tree such that all leaves are aligned horizontally. We do this by tracing starting from the root. Of the root’s two children, if one or more of one of the children’s lowest leaves are lower than the other child’s lowest leaves, we extend the segment connecting the parent to the root such that both of the children’s lowest leaves are aligned horizontally. We repeat this process with every parent (with the parent becoming the "root" in this process), starting with the highest parents and ending with the lowest parents.

Thus, since we can construct a binary tree from an equivalent Dyck path and do the converse, a bijection exists between Dyck paths and binary trees.

Below are some examples of Dyck paths and their equivalent binary trees (directly below the corresponding Dyck paths):
1.3. Bijection between Dyck paths and Young diagrams. Now we will go through the easy construction that translates Dyck paths into Young diagrams and the converse translation, thus giving a bijection between the two.

Remark 0.3. Recall that a Young diagram is a picture of several connected squares where, starting from the top left, the length of each successive row or column in the Young diagram (going down and to the right) is strictly non-increasing.

The conversion of Dyck paths to Young diagrams is simple. A Young diagram is constructed by filling in the empty space in the \( n \times n \) square that the Dyck path occupies.

Similarly, given \( n \) and a Young diagram \( \lambda \) with \( \lambda_i + i \leq n \sqrt{i} \) we can construct a Dyck path of size \( 2n \) from it. We start drawing the Dyck path from \((0,0)\). Given the first column of the Young diagram is of length \( x \), we draw a line going up of \( n - x \) units and then draw a line of one unit’s length to the right. Given the second column of the Young diagram is of length \( y \), we draw a line going up of \( x - y \) units and then draw a line of one unit’s length to the right. We repeat this process until we get to the point \((n,n)\).

Thus, since we can construct a Young diagram from an equivalent Dyck path and do the converse, a bijection exists between Dyck paths and Young diagrams.

Below are some examples of Dyck paths and their equivalent Young diagrams (directly below the corresponding Dyck paths):
Figure 3. Dyck paths and their corresponding Young diagrams.
Relationships between Binary Trees and Young Diagrams: A Story of Bookshelves

Liang Jinlong and Serra Pelin

Abstract. In this chapter a new function between Young diagram and binary trees is defined, which we call 'the bookshelf construction'. Both the bookshelf and the inverse bookshelf mappings will be well defined, with important properties, then the bijection will be shown. Further, we will prove that the mapping between Young diagram, Dyck path and binary trees commute.

2.3.1. Introduction. In this paper, we will define a mapping from binary trees to Young diagrams, called the 'bookshelf construction' as well as prove the bijection between the 2 classes by defining the inverse of the 'bookshelf construction'. We will discuss the appropriate Young diagram transformation that allows such a mapping. Further we will prove the commutativity between binary trees, Dyck path and Young diagrams using the 'bookshelf construction' and other well known mappings.

Due to the nature of this topic we will make use of geometric examples and definition as well as algebraic. Our paper builds on top of the recent literature by Nelson (2016) where a relationship between Young diagrams and Tamari lattices are constructed. In turn this implies a relationship between Young diagrams and binary trees, which is the main objective of our paper by using the 'bookshelf construction'.

2.4. Young Diagram.

Definition 1. Young diagram: A Young diagram \( \lambda \) is a geometric object consisting of boxes such that for \( \lambda_i \) representing the number of boxes in row \( i \) of the Young diagram, \( \lambda_j \geq \lambda_i \) for all \( i > j \) and \( \lambda_k+1 = 0 \) for a Young diagram with \( k \) rows. Similarly for each column going from left to right.

Consider the following example of a Young diagram with 5 boxes:

\[
\begin{array}{ccc}
\lambda_1 &=& 3 \\
\lambda_2 &=& 2 \\
\lambda_1' &=& 2 \\
\lambda_2' &=& 2 \\
\lambda_3' &=& 1
\end{array}
\]

Notice that for the rows of the diagram, \( \lambda_1 = 3 > \lambda_2 = 2 \), whereas for the columns of the diagram \( \lambda_1' = 2 = \lambda_2' > \lambda_3' = 1 \), thus satisfying the properties of a Young diagram.

2.5. Binary Tree. In tradition, a binary tree is defined as follow:

Definition 2. Binary tree: A binary tree \( \mathcal{T} \) is a planar graph which is a tree such that each node except leaves has 2 children and one parent (the root doesn’t have a parent).

For the purposes of this paper and the 'bookshelf' construction that will be explained in the upcoming section our definition of a binary tree is amended as follows:
Definition 3. New binary tree: As opposed to the traditional definition, a new binary tree $A'$, which is based on Definition 2, has the amendment such that every child of the binary tree lies on the same, i.e. the last level of the binary tree.

Consider the example below:

the traditional binary tree

In this example notice that the right most and the left most branches are extended (in red) such that all children of the tree are aligned.

Note that by definition any binary tree has 2 subtrees:

where $X$ is the left subtree and $Y$ is the right subtree. Also, $X$ and $Y$ could be the empty set.

2.6. Coordinate system for binary trees. Define a coordinate system on a binary tree with $n$ children, or of size $n - 1$, as defined above as follows:

The root of the tree is centered at $(0,0)$ defined as Level 1. For a point on the binary tree at coordinate $(m,n)$, a row-coordinate $m$ and a column-coordinate $n$, the level of that point is defined as $m + n + 1$.

Example 0.4. Below is a binary tree with 4 leaves. Above each point, its respective coordinate is denoted.

2.7. The 'Bookshelf' Construction.

\footnote{Going forward, any mention of the binary tree refers to the new binary tree as defined in Definition 3.}
2.7.1. Bookshelf Mapping.

Definition 4. Shelf: Given a binary tree that is tilted by 45 degrees counterclockwise, a shelf in this binary tree is any of the horizontal line except the top most one in the binary tree, which we call the ceiling.

Notice that prior to tilting the tree, any shelf is a branch of the binary tree with negative slope, excluding the right most one which is the ceiling.

Consider the binary tree example from before and tilt 45 degrees counter-clockwise:

where the shelf is marked in red and the ceiling marked in blue.

Definition 5. Length of the Shelf: Given any shelf on the binary tree from coordinate \((m, n)\) to \((m, k)\), the length of the shelf is \(k - n\).

Property 1. Each shelf is at most the length of the previous shelf, i.e. the shelf above it.

Proof. To see why is true, notice that (1) if both shelves start at the same column such as \((n, 0)\) and \((m, 0)\), each shelf’s length is the number of points that start with the same row value \(n\) and \(m\) respectively. By our coordinate construction if \(m > n\) then number of points of the form \((n, k)\) is more than that of \((m, k)\) for some \(k\).

(2) if the lower shelf begins at \((m, k)\) and the shelf above \((n, k')\) for \(m > n\) and \(k' > k\) then the shelf at \(n\) is shorter by a similar argument to (1).

(3) Suppose now that the lower shelf starts at a column before the shelf above it, that is the lower shelf begins at \((m, k)\) and the shelf above \((n, k')\) for \(m > n\) but \(k' < k\). Then by the definition of a binary tree the point \((n, k')\) must be a node since otherwise the shelf would start before it. So there are 2 branches from this node, one constructing the shelf along row \(n\) and the other going down along the column \(k'\). Since \(k' < k\) and \(m > n\), there exists the point \((m, k')\) such that there are 2 different branches passing through it. This contradicts the definition of a binary tree.

We have shown that in all possible cases a shelf always has at most the same length as the shelf above it.

Given a binary tree, the ’bookshelf’ construction puts \(n\) boxes above any shelf with length \(n\) recursively up until the ceiling, over all the shelves, i.e. above any branch with negative slope. This construction yields in a diagram that is a Young diagram or Young diagram with gaps.

Consider the following binary tree:

After rotating notice that by the bookshelf construction we have 2 shelves: one from the point \((1, 0)\) to \((1, 3)\) of length 3, second from the point \((2, 1)\) to \((2, 2)\) of length 1.

Putting in boxes (in gray) above each shelf:

---

2By Definition 1, we know that Young Diagram must have \(\lambda_i \geq \lambda_j\) for all \(i > j\). Since the length of the shelf \(l_1\) at coordinate \((i, j)\) is equivalent to the value of \(\lambda_i\), the length of the shelf \(l_2\) at coordinate \((i + 1, k)\) must be smaller than or equal to \(l_1\) by for \(l_1 = \lambda_i \geq \lambda_{i+1} = l_2\).
To get the final Young diagram described by a binary tree, push the boxes row-by-row from right to left such that the left-most box at each row is aligned, thus yielding a Young diagram.

This last step eliminates any gaps that occur in diagram.

2.7.2. Bijection between Binary Trees and Young Diagrams. To show that there’s a bijection between Binary Trees and Young diagram, we need firstly define the inverse book-shelf construction, which maps a given Young Diagram to a binary tree.

2.7.3. Inverse Bookshelf Mapping. Define the ‘Inverse Bookshelf’ construction from a Young diagram to binary tree as follows:

Consider a Young diagram where \( k = \#i’s \) such that \( \lambda'_i = \lambda'_1 \) and \( n = \#j’s \) such that \( \lambda_j = \lambda_1 \).

In order to place gaps appropriately to achieve the corresponding binary tree follow the steps as follows:

- if \( \lambda'_1 + k + 1 > \lambda_1 + n + 1 \): locate first column \( i \) such that \( \lambda'_i > \lambda'_{i+1} + 1 \) and put a column gap between columns \( i \) and \( i + 1 \) of size \( \lambda'_i - \lambda'_{i+1} - 1 \).
  - The gap creates 2 subdiagrams: the subdiagram to the left of the gap forms, \( X \), and the subdiagram to the right of the gap forms, \( Y \).
- if \( \lambda_1 + n + 1 \geq \lambda'_1 + k + 1 \): locate first row \( j \) such that \( \lambda_j > \lambda_{j+1} + 1 \).
  - The boxes below row \( j \) yields a new subdiagram \( Z \).

Repeat the process above for all subdiagrams, \( X,Y,Z \) with a ‘stopping condition’:

- a diagram with single column at \( (i, j) \), i.e. \( \lambda'_{j+1} = 0 \) (or \( \lambda_i = 1 \)) or
- a diagram with a single row at \( (i, j) \), i.e. \( \lambda'_j = 1 \) (or \( \lambda_{i+1} = 0 \)).

Conclude the stopping condition by row shifting to the right:

- any diagram with a single row by \( \lambda_{i-1} - \lambda_i - 1 \)
- any diagram with single column by \( \lambda_{i-1} - \lambda'_i + i \)

Notice that a single box yields the same shift in either case.

This process yields the final corresponding Young diagram with gaps.

To map from the Young diagram with gaps to the binary tree, put shelves below each row. Fill in the rest of the binary tree with positive sloped branches until the definition of a binary tree is satisfied.

---

3The bijection is outlined in detail in the appendix of this paper.
4a box at \( (i, j) \) means the box is at row \( i \) and column \( j \)
Note on the size of the binary tree: Let $k = \#i’s$ such that $\lambda_i’ = \lambda_1’$. Let $n = \#j’s$ such that $\lambda_j = \lambda_1$. Then the minimum size of the binary tree required to fit a Young diagram is:

$$\max\{\lambda_1’ + k + 1, \lambda_1 + n + 1\}$$

Consider the following step-by-step example:

Given a Young Diagram as above.

Tilt it 45 degrees clockwise and set the top-left corner of the diagram to the root of the appropriate sized tree of $\max\{6 + 1 + 1, 5 + 2 + 1\} = 8$ children.

Then since we have $\lambda_1 + 2 + 1 \geq \lambda_1’ + 1 + 1$, we search the rows and we get $\lambda_2 > \lambda_3 + 1$. Then we label the part that is above the row 3 as red.

For the remainder is the subdiagram that is below the red labeled rows, we have $\lambda_1 + 1 + 1 < \lambda_1’ + 1 + 1$. Search and get that $\lambda_1’ > \lambda_2’ + 1$. Then we put a column gap between the columns 1 (marked in yellow) and 2 of size $\lambda_1’ - \lambda_2’ - 1 = 4 - 2 - 1 = 1$.

Here do the same process recursively for the right subdiagram yields a new subdiagram of a single box (stopping condition) below the row marked in green.

Now the diagram is divided into smallest component of a diagram, yielding the final young diagram with gaps.

Put the shelves (negative sloped branches) below each block we have labeled.

For the missing children at bottom that are not connected, we connect them by positive sloped line (as the blue line showed).
2.7.4. Commutative Mapping between Binary Tree and Young Diagram. The bijection between Binary trees and Young diagrams using Dyck paths are well known. The mapping from binary tree to Dyck path is discussed in detail in Section by Bradley. The mapping from Dyck path to Young diagram is also well known. For a given Dyck path just take the shadow to get the Young diagram.

The direct mapping form Binary trees to Young diagrams is given by the 'Bookshelf construction' is described in Section (2).

In this section, it will be proven by induction that the following diagram commutes, i.e the Young Diagram given using the bookshelf construction is equal to the one given by the well known mappings between binary trees and Dyck paths and Young diagrams.

![Diagram](image)

Proof:
Let X denote the left-subtree and Y the right-subtree of a given binary tree. For a given binary tree, there are 4 possible patterns with the subdiagram:

- (A) $X = \emptyset$ and $Y = \emptyset$
- (B) $X = \emptyset$ and $Y \neq \emptyset$
- (C) $X \neq \emptyset$ and $Y = \emptyset$
- (D) $X \neq \emptyset$ and $Y \neq \emptyset$

We prove that the diagram commute in each case:

1. Consider the base case where both subtrees are empty:

   ![Diagram](image)

   It is shown above that in this case the diagram commute.

2. For a binary tree of size $n+1$, i.e. $n$ many children, consider the left subtree, X is non-empty and the right subtree, Y is empty.

   Using the Dyck path mapping, we get:
The size of the first row of the diagram is determined by the size of the binary tree, \( n + 1 \), where \( n \) is the size of the subtree \( X \).

The subdiagram \( D(X) \) is determined from the properties of the subtree \( X \) in a recursive fashion. Using the bookshelf mapping:

The subdiagram \( B(X) \) is determined from the properties of the subtree \( X \).

Notice that in both mappings, since \( Y \) is empty the first row of the Young diagram is dependent on \( X \) but always at least one box longer. Also notice that if \( X \) is the base case, then the final Young diagram is a single box since the corresponding Young diagram of \( X \) is empty.
The diagram commutes in this case if the subdiagrams $B(X) = D(X)$, denote as $Y(X)$.

(3) Now consider the case where left subtree, $X$ is empty and the right subtree, $Y$ is non-empty. Using the Dyck path mapping we get:

$$\begin{array}{c}
(0,0) \\
\uparrow \\
Y \\
\uparrow \\
Y \\
\uparrow \\
D(Y)
\end{array}$$

The Young diagram $D(Y)$ is determined only from the properties of the subtree $Y$.

Using the bookshelf mapping:

$$\begin{array}{c}
(0,0) \\
\uparrow \\
Y \\
\uparrow \\
B(Y)
\end{array}$$

Similarly, the Young diagram $B(Y)$ is determined only from the properties of the subtree $Y$ since there are no shelves outside of $B(Y)$.

$$\begin{array}{c}
(0,0) \\
\uparrow \\
Y \\
\uparrow \\
Y(Y)
\end{array}$$
The diagram above commutes in this case where the Young diagram only consists of the corresponding subtree Y, if the subdiagrams $B(Y) = D(Y)$, denote as $Y(Y)$. Notice that if Y is the base case this results in an empty Young diagram.

(4) Now consider the left subtree, X is non-empty and the right subtree, Y is non-empty. Using the Dyck path mapping we get:

![Diagram](image)

Notice that the rectangle between the 2 subdiagrams is of size $n \times (m - 1)$. The width of the rectangle $(m - 1)$ is the number of 'ups' in the Dyck path construction of X, i.e. the number of 'downs, which is given by the number of children in X not counting the last one. The height of the rectangle $n$ is in a similar fashion is given by Y since Y is a prime path we add one more level thus $n - 1 + 1 = n$.

By the Dyck path it is clear that the subdiagram determined by X can be at most one row shorter than the rectangle and similarly $D(Y)$ can only be at most 2 columns shorter than the rectangle.

Now using the bookshelf mapping:

---

5For a binary tree of length $n + m - 1$, i.e. $n + m$ children notice that if the right end of X ends at child $(n, m - 1)$ then the left end of the right subtree Y must end at child $(n - 1, m)$. Otherwise, there would be a child not connected to any branch. Therefore, it must be that X has its root at $(n, 0)$ and Y at $(0, m)$.
The subdiagram $B(Y)$ is shifted by one column. By the gap elimination operation of the bookshelf construction we get the final Young diagram as follows:

Notice that after tilting the tree the rectangle between $B(X)$ and $B(Y)$ of size $n \times (m - 1)$ is determined above X: height determined by the root node at $n$ and width by the ceiling of $B(X)$ that is $n + m - 1 - n = (m - 1)$. We now that no shelf can be as long as the ceiling of $B(X)$, this explains why the row length of the rectangle will be at least one more than that of $B(X)$. There are 2 row gaps between $B(X)$ and $B(Y)$. Similarly one gap is because of the zig-zag construction of a Young diagram another is because the column gap of the Bookshelf construction.

Each child has one parent and the right most child of X and left most child of Y have to be next to each other, thus creating one row and column gap. The other gap comes from the fact that a box does not fit at the bottom left of Y.
The diagram commutes in this case, if the subdiagrams $B(Y) = D(Y)$ and $B(X) = D(X)$, denote as $Y(X)$. Notice that if $X$ and $Y$ is the base case this results in a rectangle shaped $n \times (m - 1)$ Young diagram.

We have shown that all possible subtrees of any given binary tree maps to the same Young diagram either by the bookshelf construction or by the Dyck path assuming the subdiagrams are equal. Repeating the process for all subtrees using induction concludes that for any binary tree the two maps yield the same Young diagram.

2.7.5. Conclusion. We have shown that using the ‘bookshelf construction’, there exists a bijection between binary trees and Young diagram. Also we have shown the mappings from binary trees with Dyck path and the ‘bookshelf’ mapping yield the same Young diagram. Both proofs have made use of the recursive properties of binary trees and the partitioning of Young diagrams into subdiagrams.

2.8. Appendix. Here we will briefly describe the proof of the bijection between binary trees and Young diagrams using the bookshelf mapping.

Notice that there are 4 possible cases of subtrees that define the recursive behavior of binary trees:

(A) $X = \emptyset$ and $Y = \emptyset$
(B) $X = \emptyset$ and $Y \neq \emptyset$
(C) $X \neq \emptyset$ and $Y = \emptyset$
(D) $X \neq \emptyset$ and $Y \neq \emptyset$

To prove the bijection we will show that indeed the ‘inverse bookshelf’ construction is the inverse of the bookshelf construction.

For a given Young diagram we will take the inverse bookshelf then take the bookshelf construction for each case. If the end result is the same as the given Young diagram, it will show that every Young Diagram is given by the BookShelf construction since the two sets have the same size, so the map is a bijection.

(1) The base case is trivial.

\[ \emptyset \implies \begin{array}{c}
\text{shelf} \\
\text{X}
\end{array} \implies \emptyset \]

(2) Consider the right $Y = \emptyset$:

Notice that this has the following shape of a Young diagram:

Applying the Inverse Bookshelf Mapping algorithm, replace the first row with shelf, which is the only row that has length $\lambda_1$, and the rest of the part that’s below the first row will be the only sub-diagram X. This explicitly shows that the right part cannot have a sub-tree.

Notice that this is a spacial case of the inverse mapping for $\lambda_1 + n + 1 \geq \lambda_1' + k + 1$ where $n = 1$. If $n > 1$ we would have a subtree on the right with no corresponding boxes but rather positive $n - 1$ many sloped lines to satisfy the definition of a binary tree.

(3) Consider the case left $X = \emptyset$:

suppose we have the following transformation:

\[ (0,0) \implies (0,1) \]
In our inverse bookshelf construction such a case will never occur. This is because we have defined the corresponding binary tree to have size such that putting the top-left corner of the diagram fits the appropriate fixed tree exactly. Thus we will never get a tree such as:

\[(0,0)\]

\[\begin{array}{c}
(0,1) \\
Y
\end{array}\]

(4) Consider both subtrees are non-empty:

By the *Inverse-Bookshelf Mapping*, we equivalently take out the maximum rectangle of size \((i,j)\) in a given Young Diagram \(Z\) where \(i\) is the row satisfying \(\lambda_i > \lambda_{i+1}\) and \(j\) is the column satisfying \(\lambda'_j > \lambda_{j+1} + 1\). Thus this determines the rectangle between \(X\) and \(Y\).

All of the part on the right of \(Z\) is \(Y\) and the part below the \(Z\) is \(X\).

This describes any Young diagram that has column shifting property defined as \(\lambda'_1 + k + 1 > \lambda_1 + n + 1\) that plots the diagrams into 2 subdiagrams thus creating the left and the right tree.

Another possibility for this shape is that a diagram with multiple initial rows of maximal length of the same size, i.e. \(\lambda_1 + n + 1 \geq \lambda'_1 + k + 1\) where \(n > 1\) (recall our example for the inverse function with rows marked in red), thus requiring positive sloped subtree on the right to satisfy the definition of a binary tree. This tree will reduce to the base case (1).
A bijection between binary trees and torsion classes

Yicheng Tao

Abstract. We show the definitions and descriptions of torsion pairs, torsion classes, and torsion-free classes. We give a bijection between the set of binary trees with \( n + 1 \) leaves and the set of torsion classes in \( A_{n-1} \) which makes the descending edges in the binary tree correspond to the torsion objects in the category.

There are various sets having a Catalan number of objects, such as Young diagrams, Dyck paths, binary trees, torsion classes, and 213 permutations\(^1\). In this paper, we focus on binary trees and torsion classes. In Section 2, we provide a quick introduction to binary trees, ball structures, and zig-zag diagrams. In Section 3, we show the definitions and detailed descriptions of torsion pairs, torsion classes, and torsion-free classes. In Section 4, we first prove that every binary tree gives a torsion pair in Theorem 4.5 and then give a bijection between the set of binary trees with \( n + 1 \) leaves and the set of torsion classes in \( A_{n-1} \) in Theorem 4.6.

2.9. Binary trees, ball structures, and zig-zag diagrams.

Definition 0.5. A binary tree (sometimes called a "full binary tree") is a rooted tree in which every internal vertex has exactly two children\(^7\).

In this paper, we put balls into the empty spots of binary trees and always consider binary trees with balls without mentioning it. The balls of a binary tree form a ball structure. Binary trees with the same number of leaves have the same ball structure. We also consider the zig-zag diagram of a ball structure, which uses dashed lines to encompass every ball. Figure 1 shows an example of them.

![Figure 4. A binary tree and its ball structure with the zig-zag diagram.](image)

2.10. Torsion pairs, torsion classes, and torsion-free classes.

Definition 0.6. A torsion pair in an abelian category \( A \) is a pair of full subcategories \((\mathcal{G}, \mathcal{F})\) having the following two properties\(^8\).

(A) \( \mathcal{G} \) is the class of all objects \( X \) so that \( \text{Hom}(X, Y) = 0 \) for all \( Y \in \mathcal{F} \);
(B) \( \mathcal{F} \) is the class of all objects \( Y \) so that \( \text{Hom}(X, Y) = 0 \) for all \( X \in \mathcal{G} \).

Definition 0.7. A torsion class is the subcategory \( \mathcal{G} \) in a torsion pair \((\mathcal{G}, \mathcal{F})\)\(^8\).

Definition 0.8. A torsion-free class is the subcategory \( \mathcal{F} \) in a torsion pair \((\mathcal{G}, \mathcal{F})\)\(^8\).

It is immediate from Definition\(^0.6,8\) that \( \mathcal{G}, \mathcal{F} \) determine each other.

As we only consider torsion pairs whose objects are balls in ball structures, there is a really easy way to see what are the Hom’s.
Lemma 0.9. For two balls \( X \) and \( Y \) in a ball structure, \( \text{Hom}(X, Y) \neq 0 \) if and only if the entire rectangle with the left corner \( X \) and the right corner \( Y \) lies in the zig-zag diagram of the ball structure.

This is a well-known lemma. Examples below with Figure 2 show how it works.

(A) \( \text{Hom}(X_1, Z_2) \neq 0 \);
(B) \( \text{Hom}(X, X) \neq 0 \) for any \( X \);
(C) \( \text{Hom}(X_1, Y_3) = 0 \) since the lower end of the rectangle is missing;
(D) \( \text{Hom}(Y_2, Y_1) = 0 \) since \( Y_2 \) and \( Y_1 \) are at the wrong corners of the rectangle;
(E) \( \text{Hom}(X_2, W) = 0 \) since \( X_2 \) and \( W \) are at the wrong corners of the rectangle.

With this lemma, we can determine torsion pairs in a ball structure. For example, in Figure 2 \( \{S_1, S_3, S_5\}, \{S_2, X_2, Y_2, Z_2, S_4, X_4\} \) is a torsion pair, where \( \{S_1, S_3, S_5\} \) is a torsion class and \( \{S_2, X_2, Y_2, Z_2, S_4, X_4\} \) is the corresponding torsion-free class.

2.11. A bijection between binary trees and torsion classes. Having shown what binary trees and torsion classes are, we give a bijection between them.

Definition 0.10. Let \( D_T \) denote the set of all balls on the descending edges of a binary tree \( T \).

Definition 0.11. Let \( A_T \) denote the set of all balls on the ascending edges of a binary tree \( T \).

Remark 0.12. Every ball on the bottom row of a binary tree \( T \) is either in \( D_T \) or in \( A_T \) since there must be an edge to the leaf below it.

Lemma 0.13. Given any binary tree \( T \), \( \text{Hom}(X, Y) = 0 \) for all \( X \in D_T \) and \( Y \in A_T \).

Proof. For any \( X \in D_T \), make the largest rectangle with the left corner \( X \) that lies in the corresponding zig-zag diagram. Then for any ball \( Y_1 \) inside the rectangle, \( \text{Hom}(X, Y_1) \neq 0 \); for any ball \( Y_2 \) outside the rectangle, \( \text{Hom}(X, Y_2) = 0 \). If there is \( Y \in A_T \) inside the rectangle, the ascending edge \( Y \) sits on will cross the descending edge \( X \) sits on, which is not allowed in binary trees. So, any \( Y \in A_T \) is outside the rectangle, which means that \( \text{Hom}(X, Y) = 0 \). Therefore, \( \text{Hom}(X, Y) = 0 \) for all \( X \in D_T \) and \( Y \in A_T \). The lemma holds.

Lemma 0.14. Given any binary tree \( T \), the following statements hold.
A BIJECTION BETWEEN BINARY TREES AND TORSION CLASSES 19

Figure 7. The largest rectangle (blue) with the left corner $X$.

(A) For any $Z \notin \mathcal{A}_T$, there exists $X \in \mathcal{D}_T$ so that $\text{Hom}(X, Z) \neq 0$;

(B) For any $Z \notin \mathcal{D}_T$, there exists $Y \in \mathcal{A}_T$ so that $\text{Hom}(Z, Y) \neq 0$.

Proof. For any $Z \notin \mathcal{A}_T$, consider the lowest ball $Z_1$ in the same ascending row as $Z$. Then $\text{Hom}(Z_1, Z) \neq 0$. By Remark, $Z_1$ is either in $\mathcal{D}_T$ or in $\mathcal{A}_T$. If $Z_1 \in \mathcal{D}_T$, then (1) holds. Otherwise, $Z_1 \in \mathcal{A}_T$. Since $Z \notin \mathcal{A}_T$, there must be a descending edge that blocks the ascending edge $Z_1$ sits on from going to $Z$, on which we can find a ball $Z'_1$ in the same ascending row as $Z$ and $Z_1$. Since $Z'_1 \in \mathcal{D}_T$ and $\text{Hom}(Z'_1, Z) \neq 0$, (1) holds. Therefore, (1) holds.

For any $Z \notin \mathcal{D}_T$, consider the lowest ball $Z_2$ in the same descending row as $Z$. Then $\text{Hom}(Z, Z_2) \neq 0$. By Remark, $Z_2$ is either in $\mathcal{D}_T$ or in $\mathcal{A}_T$. If $Z_2 \in \mathcal{A}_T$, then (2) holds. Otherwise, $Z_2 \in \mathcal{D}_T$. Since $Z \notin \mathcal{D}_T$, there must be an ascending edge that blocks the descending edge $Z_2$ sits on from going to $Z$, on which we can find a ball $Z'_2$ in the same descending row as $Z$ and $Z_2$. Since $Z'_2 \in \mathcal{A}_T$ and $\text{Hom}(Z, Z'_2) \neq 0$, (2) holds. Therefore, (2) holds. \qed

Figure 8. The ascending/descending edge is blocked by another edge (blue).

Theorem 0.15. Given any binary tree $T$, the pair of subsets given by $\mathcal{G} = \mathcal{D}_T$ and $\mathcal{F} = \mathcal{A}_T$ form a torsion pair.

Proof. By Lemma 4.3, we know that for any $X \in \mathcal{D}_T$, $\text{Hom}(X, Y) = 0$ for all $Y \in \mathcal{A}_T$. By Lemma 4.4, we know that for any $X \notin \mathcal{D}_T$, there exists $Y \in \mathcal{A}_T$ so that $\text{Hom}(X, Y) \neq 0$. So, $\mathcal{D}_T$ is the set of all objects $X$ so that $\text{Hom}(X, Y) = 0$ for all $Y \in \mathcal{A}_T$. On the other hand, by Lemma 4.3, we know that for any $Y \in \mathcal{A}_T$, $\text{Hom}(X, Y) = 0$ for all $X \in \mathcal{D}_T$. By Lemma 4.4, we know that for any $Y \notin \mathcal{A}_T$, there exists $X \in \mathcal{D}_T$ so that $\text{Hom}(X, Y) \neq 0$. So, $\mathcal{A}_T$ is the set of all objects $Y$ so that $\text{Hom}(X, Y) = 0$ for all $X \in \mathcal{D}_T$. Therefore, according to Definition 0.16 the pair of subsets given by $\mathcal{G} = \mathcal{D}_T$ and $\mathcal{F} = \mathcal{A}_T$ form a torsion pair. The theorem holds. \qed

Theorem 0.16. There exists a bijection $f$ from the set of binary trees with $n + 1$ leaves to the set of torsion classes in $\mathcal{A}_{n-1}$ so that $f(T) = \mathcal{D}_T$. Furthermore, this bijection is order reversing, i.e., if one binary tree is less than another the torsion class for the smaller tree contains that of the larger tree.
Proof. By theorem 4.5, it is obvious to see that there exists a map \( f \) from the set of binary trees with \( n + 1 \) leaves to the set of torsion classes in \( A_{n-1} \) so that \( f(T) = D_T \). We prove \( f \) is a bijection as follows. For two distinct binary trees \( T_1 \) and \( T_2 \) with \( n + 1 \) leaves, \( T_1 \) must have an edge that \( T_2 \) does not have. If it is a descending edge, then \( f(T_1) = D_{T_1} \neq D_{T_2} = f(T_2) \). If it is an ascending edge, then \( A_{T_1} \neq A_{T_2} \). Assume that \( D_{T_1} = D_{T_2} \). By Theorem 4.5 and the fact that \( T_1 \) and \( T_2 \) both with \( n + 1 \) leaves, \( (D_{T_1}, A_{T_1}) \) and \( (D_{T_2}, A_{T_2}) \) are torsion pairs defined on the same ball structure. By Definition 0.6, \( A_{T_1} \) and \( A_{T_2} \) are both the class of all objects \( Y \) so that \( \text{Hom}(X,Y) = 0 \) for all \( X \in D_{T_1} \), so \( A_{T_1} = A_{T_2} \), which contradicts \( A_{T_1} \neq A_{T_2} \). Thus, the assumption is false and we have \( f(T_1) = D_{T_1} \neq D_{T_2} = f(T_2) \). To sum up, \( f(T_1) \neq f(T_2) \), which gives that \( f \) is an injection.

Moreover, it is well known that there are a Catalan number \( C_n = \frac{1}{n+1} \binom{2n}{n} \) of binary trees with \( n + 1 \) leaves\(^1\) as well as torsion classes in \( A_{n-1} \). Therefore, \( f \) is a bijection from the set of binary trees with \( n + 1 \) leaves to the set of torsion pairs in \( A_{n-1} \). The theorem holds.

Finally, to show that this bijection is order reversing, note that the ordering of binary trees is given by shifting descending edges to the right. In that case the “balls fall off” of this edge and the torsion class becomes strictly smaller. So, the bijection is order reversing. \( \square \)
Torsion Classes and Binary Trees

Max Weinstein

Abstract. This chapter will define and give examples of torsion classes and their respective torsion free classes as they appear in relation to other Catalan objects. Furthermore a clear bijection will be given between torsion classes and binary trees.

2.12. Definitions.

2.12.1. Torsion Classes.

Definition 0.17. A torsion class is collection of objects in a category (for our purposes we will only consider the straight categories $A_n$) which are closed under extension and quotients.

It is useful to think geometrically of the category as being a collection of balls in an equilateral triangle configuration with objects in the torsion class being colored in blue. The property of being closed under extension simply means that if two objects (balls) are in the torsion class, then if a rectangle with vertices at these two dips at most 1 imaginary ball below the bottom of the configuration, the other vertices within the configuration are also in the torsion class.

2.12.2. Torsion-Free Class.

Definition 0.18. The corresponding torsion-free class is the collection of balls in the configuration which have the property that there are no homomorphisms from any balls in the torsion class to them. In this case homomorphisms exist if there is a way to draw a complete rectangle within the configuration going up and right from a torsion class ball to another ball.

We will color objects in the torsion-free class red:

2.12.3. Torsion Pair. The torsion pair is just a natural algebraic way of describing the torsion class and the corresponding torsion free class simultaneously.

Definition 0.19. A torsion pair is a pair $(\mathcal{G}, \mathcal{F})$ of full subcategories such that:

\[
\mathcal{G} = \{ x \mid \text{Hom}(x, y) = 0 \ \forall y \in \mathcal{F} \}
\]

and

\[
\mathcal{F} = \{ y \mid \text{Hom}(x, y) = 0 \ \forall x \in \mathcal{G} \}
\]

Where $\text{Hom} = 0$ means that no rectangle can be drawn from an object $x$ to $y$ where all four vertices are also objects in the category.
2.13. Constructing a Torsion Class. Torsion classes can be generated by the inclusion of an arbitrary number of objects in the category (if no objects are included this is just the empty torsion class) and by considering in advance the Torsion Pair. Call these objects added \( x_n \). Then find the set of all \( Y \) such that \( \text{Hom}(x_n, Y) = 0 \). \( Y \) is the torsion free class. Then find the set of all \( X \) such that \( \text{Hom}(X, Y) = 0 \) which is the full torsion class.

1. Start with an arbitrary selection of objects filled in blue (\( x_n \)).

2. Then locate objects where no rectangle can be drawn up/right or down right from a blue object (\( \text{Hom} = 0 \)) and fill them in red (\( Y \)).

3. Fill in objects blue which have no homomorphism to any red objects.

The total collection of blue objects is the complete torsion class.

2.14. Bijection Between Binary Trees and Torsion Classes.

2.14.1. Geometric Correspondence. For a binary tree with \( n \) leaves, take the torsion classes corresponding with triangular configurations of balls having \( n - 2 \) balls along each edge. (i.e. if a binary tree has 5 leaves, take a triangle configuration having 3 balls on each edge). Each ball should be located directly above an "inside leaf" (just a leaf that’s not on the end).

The bijection is given geometrically like so: Directly above all branches of the binary tree which are descending left to right, color the objects blue. directly above the branches ascending left to right color the objects red.

**Theorem 0.20.** The collection of objects directly above the descending edges (blue) gives a torsion class and the collection of objects directly above the ascending edges (red) form the corresponding torsion-free class.

**Proof** Take our proposed correspondence from binary trees to torsion classes. Above all descending branches should be objects in the torsion class, but why do they form a torsion class? Notice that
along descending branches are objects that if one were in the torsion class, they would all have to be, since by our construction of torsion classes objects in the torsion class will fall diagonally below the already included object, since these objects will share the same torsion free objects in common. This is so because there are always homomorphisms between objects on the same diagonal line (so they will never include both torsion and torsion free objects) and because if an object is in the torsion free class with respect to a given torsion object, then an object diagonally downward cannot have any homomorphisms to any of those torsion free objects. Torsion free objects will always be to the left of the descending line or on the bottom row, all of which will not have a homomorphism to objects on the line. Hence we get the descending branches.

To go the other direction, from an arrangement of balls draw descending branches underneath objects in the torsion class and ascending branches under objects in the torsion free class.

**Theorem 0.21.** The binary tree got from drawing descending branches underneath torsion objects is unique.

**Proof** We know that torsion objects give clear instructions for drawing descending branches of binary trees, but how do we know this tree is unique? The proof follows from the fact that binary trees are entirely determined by their descending (or ascending edges). Suppose you fixed the descending edges of a binary tree. Then for all leaves left unconnected, ascending branches must be linked to them, giving the rest of the tree. Since there are only two options for a branch (it is descending or ascending) giving one gives the other since they are complements. It turns out it is always possible to connect all remaining spaces with ascending edges once descending edges are defined.

**Theorem 0.22.** The correspondence described above is a bijection.

**Proof** This follows from the well known fact that there are a Catalan number $C_n$ of both binary trees with $n+1$ leaves and torsion classes with $n-1$ balls in the bottom row. So since a unique tree is given by every torsion class, and a unique torsion class is given by every binary tree, and there are exactly the same number of each, they are in a bijective relationship and the bijection is given by our rule.

2.14.2. **Induction.** It may still be supposed that our correspondence fails when building larger binary trees but the following induction shows that any new tree formed from smaller trees in fact yields a proper torsion pair.

Connecting two trees gives a new descending branch, a new ascending branch and potentially some objects without branches underneath. Objects along the new descending branch will be filled in blue by our correspondence, but notice that they should also be included to the torsion class by our construction of torsion classes. There will be no homomorphism between objects along the new descending branch and objects down and to the left of them which are in the torsion free class of the previous tree. Therefore the objects along the new descending line MUST be filled blue. Objects along the new ascending branch will be filled red by our correspondence, but notice that the new ascending branch must connect to a leaf at the bottom which is at least one space away from the right-most leaf of the left subtree. This guarantees there is no way to draw a rectangle, and thus find a homomorphism, from any torsion object in the left subtree to the new ascending branch, and obviously any torsion object in the right subtree cannot have a homomorphism going left. The objects along neither new branch are exactly those who are diagonally upward from the objects along the descending branch, meaning that there are existing homomorphism from torsion objects to them, thus meaning they do not lie in either the torsion class or torsion free class.

Thus binary trees with $n+1$ leaves and torsion classes of $A_{n-1}$ are in bijection with exactly a $C_n$ number of each, and our correspondence gives the bijective mapping.
Young diagrams with gaps and torsion classes

Ruiyang Hu

Abstract. The purpose of this short paper is to show an explicit bijection between subdiagrams of a full Young diagram with gaps and torsion classes with the same size $n$.

2.14.3. Introduction. It is well-known that there are $C_{n+1}$ number of Young diagrams that are subdiagrams of a full Young diagram without gaps of size $n$. ($C_{n+1}$ denotes for the $n + 1$th Catalan number.) So is the number of torsion class for $A_n$. Since the cardinality of two sets are the same, if we want to prove a given mapping between Young diagrams with gaps and torsion classes of the same size is a bijection, showing that such mapping is a injection or surjection will be enough.

In the following part of this paper, we will first have to define all concepts we are going to use in this paper. Then we will show some examples of correspondences between Young diagrams with gaps and torsion classes when $n$ is small. Finally, we will use a strong Mathematical induction to construct an explicit bijection between two sets of objects.

2.15. Definitions and Examples. Since this paper tries to prove there is a bijection between Young diagrams with gaps and torsion class that both have the same size $n$, we will first define what is a Young diagram with gaps.

Definition 0.23. A Young diagram with gaps $Y$ is a visual representation of a partition, just like normal Young diagrams without gaps, where:

a) $Y$ is transformed from the corresponding Young diagram without gaps $Y'$ that contains $n$ boxes and $k$ rows. Each row of $Y'$ has various number of boxes in it where number of boxes in $k$ rows sum up to $n$. ($k, n \in \mathbb{Z}^+$)

b) Let $\lambda_i$ denotes number of boxes in the $i$th row of $Y$. If $Y'$ is a valid normal Young diagram, then number of boxes in each row has to satisfies following rule: $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_k > 0$ and $\lambda_{k+1} = 0$.

c) $Y$ is transformed from the corresponding normal Young diagram $Y'$ by first rotating clockwise by $45$ degrees around the upper-left corner of $Y'$. Then the rotated diagram will have ascending rectangles and descending rectangles, which are originally columns and rows of Young diagrams.

d) Depends on number of rows and columns in $Y'$, there are four different ways for four cases to complete the transformation from $Y'$ to $Y$:

1) If $Y'$ contains more rows than columns, then we have to the leftmost rectangle in $Y$ that its lowest box is in the lowest position. Use the lowest corner in that box as start point, draw a horizontal line which is perpendicular to the straight line formed by connecting lowest and highest corners of the box. For all other ascending rectangles on the left and right of this big rectangle, slide them down without breaking the integrity of each ascending rectangles one by one until the lowest corner of the lowest box in each ascending rectangles falls on the horizontal line.

2) If $Y'$ contains more columns than rows, then we need to find the rightmost rectangle in $Y$ that its lowest box is the lowest position. Just like the previous case, we can draw a horizontal line with the lowest corner in the box. Finally, we slide all other ascending
rectangles on the left and right of the big rectangles downward without breaking their integrity until the lowest box in each ascending rectangles falls on the horizontal line.

3) If $Y'$ contains the same number of columns and rows, and the leftmost and rightmost boxes are both the lowest boxes in this diagram. Our next step is to keep the row and the column that contain these two boxes unchanged, and slide all inner ascending rectangles downward so that the lowest box in each inner ascending rectangle falls on the horizontal line.

4) If $Y'$ contains the same number of columns and rows, and the leftmost and rightmost boxes are not in the lowest position. Then we have to find rectangles that contain boxes lay in the lowest position in this diagram. Draw a horizontal line on the lower corner of these boxes, and slide all ascending edges on the left and right of these rectangles that already falls on the horizontal line downwards without breaking integrity of each ascending edge.

e) To transform from Young diagrams with gaps back to Young diagrams without gaps, we first have to sort ascending rectangles by number of boxes in each rectangle with descending order. After that, the ascending rectangle with smallest number of boxes will be put in the rightmost position. Then we will remove gaps between rectangles. Finally, rotate the diagram by 45 degree counterclockwise, and the result is a Young diagram without gaps.

To clarify the definition, here are some examples of valid Young diagrams without gaps and their corresponding Young diagrams with gaps:

For the normal Young diagram $Y_1'$ above, it has 5 boxes so $n = 5$. It has 3 boxes in the first row and 2 boxes in the second one, thus $\lambda_1 = 3$, $\lambda_2 = 2$ and $\lambda_3 = 0$. Since the diagram has two rows, $k = 2$. It is easy to see that all $\lambda$ here statisfy rule b) in definition 2.1. Next step is to transform $Y_1'$ into a Young diagram with gap.

First we rotate $Y_1'$ clockwise by 45 degree, where the result is shown by the graph above. For this example, we have already transformed $Y_1'$ into a Young diagram with gaps because number of rows is larger than number of columns, and it is case 2. By drawing a horizontal line(shown a red line by the diagram below) at the lowest corner of the lowest box in rightmost descending rectangle, we can observe that all lowest corners of the lowest boxes of other descending rectangles already lay on the horizontal line. This means we do not have to slide anything downward.

The Young diagram $Y_2'$ without gaps below belongs to case 1 since it has 3 rows and only 2 columns.

By rotating $Y_2'$, we will get a graph that is shown below. This graph is not a valid Young diagram with gaps because the lowest box of one ascending rectangle does not fall on the horizontal line, which is shown below.
The valid Young diagram $Y^\text{2 with gaps}$ should look like this after we shift all other ascending rectangles downwards:

$Y^\text{3 without gaps}$ without gaps below belongs to case 3) because it has 3 rows and 3 columns.

After we rotate $Y^\text{3 without gaps}$ clockwise by 45 degree, we now have this diagram below, which is currently not a valid Young diagram with gaps:

We can find that although the lowest box of the leftmost ascending rectangle and that of the rightmost descending rectangle falls on the red horizontal line, there is one inner ascending rectangle(which in this case, only contains one box) not laying on the horizontal line. So we will have to slide it down to get a valid Young diagram with gaps, and the result looks like this:

For more examples of transforming between Young diagram without gaps and Young diagrams with gaps, please check Olly Liang and Serra’s paper.

After Young diagrams with gaps are defined, we shall give definitions and examples of full Young diagrams without gaps and full Young diagrams with gaps.

**Definition 0.24.** A full Young diagram without gaps $Y_{fw}$ is a special type of Young diagrams without gaps that not only satisfies all requirements of Young diagram without gaps, but also satisfies following rules:

a) Total number of boxes $N_b$ in the full Young diagrams belongs to the Triangular number sequence, which means that $N_b = \binom{a+1}{2}$ and $a$ is the number of boxes in the longest row of $Y_{fw}$.

b) Numbers of rows and columns in $Y_{fw}$ all equal to $a$.

For example, the diagram below is a full Young diagram without gaps, where $a = 2$ and $N_b = 3$:
Definition 0.25. A full Young diagram with gaps \( Y_f \) is a special type of Young diagrams with gaps that not only satisfies all requirements of Young diagram with gaps, but also satisfies following rules:

a) Total number of boxes \( N_b \) also belongs to the Triangular number sequence, and \( a \) is the number of boxes in the longest ascending rectangle of \( Y_f \).

b) \( Y_f \) has \( a \) ascending rectangles and \( a \) descending rectangles.

As usual, we need an example to make the definition clear:

\[
\begin{array}{ccc}
& & \\
& & \\
& & \\
& & \\
\end{array}
\]

The diagram above is an example of a full Young diagram with gaps, where \( N_b = 6 \), \( a = 3 \) and \( \binom{4}{2} = 6 = N_b \) which satisfies rule \( a \) in definition 2.2. Besides that, the full Young diagram with gaps above corresponds to the full Young diagram without gaps above. By simple counting we can find out that number of boxes in a row decreases by one each time from the highest row to the lowest row. (3 \( \rightarrow \) 2 \( \rightarrow \) 1)

Definition of torsion classes will be defined in the paper written by Max and Yicheng Tao, hence we will skip this part this paper.

In introduction section of this paper we mentioned that there exists a correspondence between subdiagrams of a Young diagram with gaps and torsion classes with the same size, so so it is necessary to define the term size.

Definition 0.26. If a torsion class or a full Young diagram with gaps has a size of \( n \), it means that:

a) The torsion class contains \( \frac{n^2 + n}{2} \) balls, which contains 1 ball in first row (or 0 ball if \( n = 0 \)), then two balls in the second row and so on.

b) The full Young diagram with gaps contains \( \frac{n^2 + n}{2} \) boxes.

Since we already have an example of full Young diagram with gaps that has a size of 3, let us take a look at an example of a torsion class with size of 3:

\[
\begin{array}{ccc}
& & \\
& & \\
& & \\
\end{array}
\]

The diagram above is an example of a torsion class which has a size of 3. As we can see, there are \( \frac{3^2 + 3}{2} = 6 = \binom{3}{2} \) balls.

Because we have already defined all definitions we will need in this paper, we will move to next section, which shows how to transform from a torsion class to the corresponding Young diagram with gaps.
2.16. Method and Examples of Transforming a Torsion Class to Its Correspondence Young Diagram. To construct an explicit bijective correspondence between torsion classes and subdiagrams of a full Young diagram with gaps that both have the same size, we first have to know how the transformation between two kinds of objects works. As we have already mentioned in the introduction part, there are $C_{n+1}$ number of subdiagram of a full Young diagram with gaps of size $n$ and the same number of torsion classes with size $n$. Hence, we will only have to construct an injection or surjection from one set to another and the bijection relationship will be proved. In this paper we will approach from torsion classes to Young diagrams with gaps, so we have to explain how to transform a random torsion class into its corresponding Young diagram with gaps.

Let’s start with how to draw and complete a torsion class:

1) First fill in some balls in the torsion class with color blue.
2) For balls we already filled with blue, check whether there exist a ball that lays in the lower right position relative to a ball that has already been colored with blue. If there is, color that ball with blue too.
3) Keep executing the second step until there is no more balls that is not colored while lays in the lower right position of a colored ball.
4) For each pair of colored balls, check whether they could be put into a rectangle with four corners of four different balls where each of these two balls are neither in the highest nor in the lowest corner of the rectangle. If they could, check whether the ball in the lowest corner of this rectangle is inside the torsion class or could be added at one line below the lowest row of balls in the torsion class. If the answer is yes, color the ball in the highest corner of this rectangle with blue and we call such ball as an “extension”. Then go back to step 2 again.

Here is an example that helps illustrate these rules above:

![Example of a torsion class](image)

The diagram above is currently an incomplete torsion class with size of 3. So let us follow steps of completing a torsion class listed above. There are already two blue balls in the incomplete torsion class, and for each ball there is an uncolored ball in the lower right corner. So according to step 2, we will have to color these two balls with blue color too, and the result will look like this:

![Completed torsion class](image)

However, when we enter step 3, we will have to repeat step 2 again since there still exists a ball uncolored that also lays in lower right position of a colored ball. When we enter step 4 and test every existing pair of blue balls, we can observe that all possible extension are already colored with blue. Here is the complete torsion class:

![Complete torsion class](image)

Let us have another example in order to make torsion classes completion steps clearer:

![Another example of a torsion class](image)

The diagram above is an incomplete torsion class with size of 4 this time. Just like the previous example, we will have to check for empty balls that should be colored. This is what we have before we begin to execute step 4.

![Another complete torsion class](image)
We can find that the pair formed by two blue balls on the third row can form a rectangle where the ball in lowest corner (shown by the yellow ball in the diagram below) is only one line below the lowest row of balls. Here is an invalid rectangle: the pair of balls formed by two balls on the 4th row can not form a valid rectangle because the lowest ball (colored with black) is two lines below the lowest row of this torsion class.

Hence, after we colored the only extension in this example, we have the complete torsion class:

After rules of constructing a complete torsion class is described, we will show how to transform from a torsion class to its corresponding Young diagram with gaps. Similar to the transformation from a binary tree to a Young diagram, which will be introduced by Olly Liang and Serra, we will use to "bookshelf" method too. First we can observe that a torsion class has 'columns' that are formed by balls. For example, these four black balls in the diagram below represents the first column and the diagram has four columns:

Given a torsion class with some balls colored with blue in some columns, we will need to traverse every columns. Starting from the leftmost column, each time we will move to right by one column. And finally we will reach the rightmost column which will always contains only one ball. For each column, traverse it from the lowest ball to the highest ball in the column and find the first ball that is colored with blue. Put boxes on this ball as well as all balls above which are in the same column. Let us use examples to demonstrate this process:

In the diagram above, we can observe that in all three acending edges there exist at least one blue ball, so the corresponding Young diagram with gaps should be:

Let us have one more example, which this time we will use the torsion class above that has a size of 4. Its correspondence Young diagram with gaps should be like this:

Now we have shown how to transform from a torsion class to a Young diagram with gaps, we will move to prove that there exists a bijection between torsion classes and subdiagrams of a full Young diagram with gaps of the same size $n$. 
2.17. Proof of the Bijection between Torsion Classes and Subdiagrams of a Full Young Diagram with Gaps of the Same Size $n$. To construct a bijection, we will use strong induction here. We can define a recursively mapping from a torsion class with size $n$ to its corresponding Young diagram with gaps.

According to the torsion class rules, as long as this torsion class is not empty, there will always be at least one colored ball in the lowest row of the torsion class. Find the leftmost ball on the lowest row of the torsion class such that the corresponding box of this ball and the highest box in the Young diagram with gaps are contained in the same rectangle in the Young diagram with gaps. And it is the rectangle with most number of boxes(balls) in it. Then this big rectangle will divide the torsion class into two parts: we call the part on the left of this rectangle $X$ and the part on the right $Y$ ($X$ and $Y$ could be empty). Then we will do the same operations on $X$ and $Y$, until it reaches the unit torsion class, which has only one ball in it.

Let $P(n)$ denotes that such recursively defined mapping between torsion classes of size $n$ and subdiagrams of a full Young diagram with gaps of size $n$ is a bijection.

First we will have to prove the basis step $P(1)$. There are two torsion class with a size of one. Here are these two torsion classes and two corresponding Young diagram with gaps:

```
  o
  □
```

As we can see, each torsion class corresponds to a unique Young diagram with gaps and all subdiagrams of the full Young diagram with gaps of size 1 are mapped to, and for both cases $X$ and $Y$ equal to $\phi$. So $P(1)$ is true.

Then, we assume that for all integers smaller than $K$, where $K \geq 1$, $P(1), P(2),..., P(K-1)$ are all true. We then have to prove that when size equals to $K$, the mapping defined recursively is still a bijective correspondence. By applying the same method on torsion classes, we will have several cases:

1) The first case is that the leftmost column does not contain any colored balls, this means we can not form a big rectangle that contains the uppermost ball which lays in the first row. However, this also means that we can just omit the uncolored column and focus on those columns that contains colored balls on the right. By omiting the leftmost column, we decrease the size of torsion class by at least one without changing the torsion class and affect its corresponding Young diagram with gaps. Since all $P(C)(1 \leq C < K)$ are true according to assumption, in this case such recursively defined mapping is still a bijection.

2) The second case is that we could find a big rectangle that contains most number of balls where the upper corner is the ball in the first row and the lowest corner is the leftmost ball we could find while keeps number of balls in the rectangle being largest. Then we can divide the torsion class into this rectangle, $X$ and $Y$. Obviously $X$ and $Y$ are all smaller than $K$ so that we could apply the assumption on them and breaks them into even smaller pieces such as $X_1$ and $X_2$ with the recursively defined mapping until they are splitted into the base case.

Thus, as long as $P(C)(1 \leq C < K)$ are true, $P(K)$ is true. Since we already proved that $P(1)$ is true, so $P(K)$ is true for all $K \geq 1$.

Here are some examples of breaking a torsion class into smaller torsion class:

```
  o
  □
```

First we could find that the leftmost ball in the lowest row of this torsion class that forms a rectangle that contains most number of balls and the uppermost ball is the second ball in the lowest row. So we can draw a rectangle to divide this torsion class into smaller pieces:
Here we have $X$ and $Y$, where $X$ belongs to the base case since it has a size of 1. Let’s take a look on $Y$, we can find that there is still a rectangle in $Y$, where the leftmost ball is the second ball in the lowest row:

![Diagram](image)

We could find that this new rectangle divides $Y$ into $Y_1$ and $Y_2$, where both of them belongs to the base case. Hence we are done here.

2.17.1. **Conclusion.** By strong induction, we have shown that such recursively defined mapping is an explicit bijection between torsion classes and subdiagrams of a full Young diagram with gaps of the same size.
213-Avoiding Permutations and Binary Trees

Zhaonan Li, Michael Richard

Abstract. The aim of this paper is to define and describe 213-avoiding permutations, and to prove a bijection with binary trees. Such a bijection was known to have existed as each of these objects are counted by Catalan numbers, however this paper offers one explicitly.

2.17.2. Introduction. It is well known that 213-avoiding permutations and binary trees are in bijection due to the fact that there are a Catalan number of each for a given size. However, this paper seeks to give a new bijective map that allows one to convert 213-avoiding permutations into binary trees and vice versa. The drawing of these binary trees is done so according to another paper and relates to the bookshelf construction (olly & serra reference). This novel method of expressing binary trees also allows for the use of the “baseball” construction, which maps these binary trees to 213-avoiding permutations.

2.18. 213 Avoiding Permutations.

Definition 0.27. A 213-avoiding permutation is a permutation $p = (m_1, m_2, m_3, ..., m_n)$ such that for any $m_i$ and $m_j$ ($j > i$), if $m_i > m_j$ there is no $m_k$ ($k > j$) such that $m_k > m_i$.

Example 0.28. $p = (5, 1, 2, 3, 4)$ is a 213-avoiding permutation of 5.

Example 0.29. $p = (2, 3, 1, 4, 5)$ is not a 213-avoiding permutation of 5, because $m_1 > m_3$, but $m_4 > m_1$.

The following theorem states a well-known fact that we are going to use in this paper:

Theorem 0.30. The number of 213-avoiding permutations with $n$ terms $P_n$ is equal to the $n$-th Catalan number

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

2.19. Converting 213-avoiding Permutations to Binary Trees.

Definition 0.31. Let $|p|$ be defined by the cardinality of, or the number of terms contained within, a permutation $p$.

Example 0.32. For the permutation $p = (4, 1, 2, 3)$, $|p| = 4$

2.19.1. Coordinate system for binary trees.

In this section, we introduce the coordinate system used to describe binary trees in this paper. The two edges extending from the root are regarded as two axes. Every node in the binary tree is represented by two numbers $x$ and $y$, measuring the distance along the two axes. The root of the binary tree is positioned at the origin $(0, 0)$. Every edge extends to the bottom. This coordinate system was used in a paper which can be referred to for examples (Olly and Serra).
2.19.2. Constructing a binary tree given a 213-avoiding permutation.

In this subsection, we define the procedure to convert a 213-avoiding permutation with \( n \) terms to a binary tree with \( n + 1 \) leaves. Given some 213-avoiding permutation \( p = (\phi_1, \phi_2, \phi_3, \ldots, \phi_n) \) with \( n \) terms. Let \( \phi_0 \) be the smallest term in \( p \). Let \( y \) be defined as the sequence of all terms in \( p \), \( \phi_0 \) such that \( b < a \). Let \( x \) be defined as the sequence of all terms in \( p \), \( \phi_0 \) such that \( c > a \). Now, to construct the binary tree which corresponds to \( p \), draw an ascending line from \((n, 0)\) to the root, \((0, 0)\) and a descending line from \((0, 0)\) to \((0, n)\). Then, draw an ascending line from \((|y|, n - |y|)\) to \((0, n - |y|)\) and a descending line from \((n - |x|, 0)\) to \((n - |x|, |x|)\). This second pair of lines creates a new binary tree, a sub-tree with a new root on which this process can be reiterated. If \( x \) or \( y \) contain no terms, then their respective sub-tree is complete. Otherwise, redefine \( x \) and \( y \) as their own permutations \( p \), and repeat the process until both \( x \) and \( y \) contain no terms.

According to the construction defined in section 2.19.2, a lemma immediately follows.

**Lemma 0.33.** Given any 213-avoiding permutation of \( n \), there exists exactly one corresponding binary tree with \( n + 1 \) leaves.

**Example 0.34.** Using the procedure described above, and the coordinate system defined in Section 2.6 the permutation \( p = (5, 1, 2, 3, 4) \) can be converted into the following binary tree.

\[
\begin{array}{c}
(0,0) \\
(2,0) \\
(3,0) \\
(4,0) \\
(5,0) \\
(4,1) \\
(3,2) \\
(2,3) \\
(1,4) \\
(0,4) \\
(0,5)
\end{array}
\]

- **(A)** Given \( p = (5, 1, 2, 3, 4) \), an ascending line from \((5, 0)\) to \((0, 0)\) is formed, and a descending line from \((0, 0)\) to \((0, 5)\) is formed. The smallest term in \( p \) is \( p_2 = 1 \). Thus the sequence of preceding terms \( y = (5) \) contains 1 terms, and the sequence of following terms \( x = (2, 3, 4) \) contains 3 terms. An ascending line from \((1, 4)\) to \((0, 4)\) is formed, and a descending line from \((2, 0)\) to \((2, 3)\) is formed. Reiterate \( x \) and \( y \) at their corresponding origins, which are \((2, 0)\) and \((0, 4)\) respectively.

- **(B)** Given \( p = (5) \), the ascending and descending lines overlap with the existing drawings, and given \( x \) and \( y \) in this case are all empty, the construction for the current \( p \) stops.

- **(C)** Given \( p = (2, 3, 4) \), the ascending and descending lines overlap with the existing drawings. The smallest term is this case is \( p_1 = 2 \). Hence, \( y \) is empty and \( x = (3, 4) \) which contains 2 terms. A descending line from \((1, 0)\) to \((1, 2)\) is formed. This line is adjusted to its origin \((2, 0)\) and a descending line from \((3, 0)\) to \((3, 2)\) in the original drawing is obtained.

- **(D)** The case for \( p = (3, 4) \) follows similarly from step (3), and \( p = (4) \) follows similarly from step (2).

2.20. The "Baseball" Construction: Converting Binary Trees to 213-avoiding Permutations.

Here we describe a construction which can be used to convert binary trees into 213-avoiding permutations. In a binary tree with \( n \) leaves, a triangle of circles with \( n - 2 \) circles at the base is placed within the binary tree. Circles that lie immediately above a descending line are called baseballs, while circles that do not are called cross-balls. The wire diagram is given by connecting lines from right to left through the baseballs and cross-balls.
Example 0.35. Below is the wire-diagram for a “cross-ball”.

Example 0.36. Below is the wire-diagram for a “baseball”.

Example 0.37. Wire diagrams for two simple cases.

The wire diagram on the left corresponds to the permutation (1, 2), and the wire diagram on the right corresponds to the permutation (2, 1). In either case, the permutation is 213-avoiding.

Example 0.38. Below, a binary tree constructed through the procedure described in subsection 2.19.2 from the 213-avoiding permutation $p = (1, 3, 4, 2)$ is shown. Through the binary tree, a wire diagram formed using the baseball construction demonstrates that the correct permutation is obtained.

Theorem 0.39. The number of binary trees with $n + 1$ leaves $B_n$ is equal to the $n$-th Catalan number

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

Theorem 0.40. The procedure provided within the proof of Lemma 0.33 describes a bijective map between 213-avoiding permutations and binary trees.

Proof. We start by presenting a generalization of the baseball construction. The figure below functions as a model for the baseball construction representation of every binary tree. This is because the structure of the area between the $x$ and $y$ sub-trees is the same for all binary trees. This follows from the definition of the baseball construction, as baseballs are placed only above descending lines, and there are no descending lines not contained in either $x$ or $y$, so all balls outside of $x$ and $y$ that do not rest on the descending branch of $x$ must be cross-balls. The figure demonstrates that for a sequence $s = (1, 2, ..., k, ..., n)$ all terms greater than one and less than or equal to $k$ are sent to the $x$ sub-tree, which sends terms to positions after 1. Additionally, all terms greater than $k$ and less than or equal to $n$ are sent from the $y$ sub-tree to positions before 1. Therefore, for $p$ to be a 213-avoiding permutation, it is sufficient to prove that $x$ and $y$ are 213-avoiding permutations. This can be done by recognizing that $x$ and $y$ themselves can be modeled by the generalized baseball construction representation, but will have fewer terms than $p$. So, the process can be reiterated until $x$ and $y$ contain only two terms, which as demonstrated in Example 0.37 are always 213-avoiding.
Therefore, for any binary tree, there exists a 213-avoiding permutation that maps to it, making the map surjective. In addition, based on Theorem 0.30 and Theorem 0.40, there are the same number of 213-avoiding permutations with \( n \) terms as binary trees with \( n + 1 \) leaves, so the map is bijective.

**Example 0.41.** The 213-avoiding permutation \( p = (5, 1, 2, 3, 4) \) corresponds to the binary tree shown in example 0.34. The wire diagram below demonstrates that this binary tree can be converted to obtain the correct permutation. The wire corresponding to 1 which is drawn in green connects to the position directly above \( x \). The wires corresponding to 2 through \( k \), which is in this case equal to 4, drawn in red, connect to the positions within \( x \). And, the wire corresponding to \( k + 1 \) through \( n \), in this case both are equal to 5, connects to a position above the position to which 1 is connected.

**Remark 0.42.** Torsion classes to 213-avoiding permutations: Torsion classes correspond to 213-avoiding permutations in the following way. The placement of torsion classes is identical to the placement of baseballs in the baseball construction. That is, both torsion classes and baseballs are to be placed on the descending shelves of the bookshelf construction. To map directly from torsion classes to 213-avoiding permutations, develop a baseball construction from the torsion class with baseballs only on torsion classes and cross-balls in all other locations. Then, use the baseball construction to determine the permutation. The reverse mapping is provided similarly; generate a baseball construction from the 213-avoiding permutation using the process described in this paper, and place torsion classes on exactly the locations occupied by baseballs.

2.20.1. **Conclusion.** In this paper, we used a coordinate system for binary trees defined in an earlier paper in this volume and used this to derive a rigorous procedure by which 213-avoiding permutations can be converted into binary trees. Then we used a representation called the “Baseball” construction to derive a procedure for converting binary trees into 213-avoiding permutation. These findings all built off the equality between the number of 213-avoiding permutations and the number of binary trees, and hence a bijection between 213-avoiding permutations and the binary trees is
established. An example is given by the permutation $p = (5, 1, 2, 3, 4)$ illustrated in Example 0.34 and Example 0.31. While the existence of the bijection between these objects is well known, the procedures put forth in this paper for the conversions between them are novel.
Figure 9. Above is a Tamari lattice of order 4 in which each node is a binary tree with five leaves. One can see that the covering relation is given by replacing a descending edge with an ascending edge.
Figure 10. Above is the Tamari lattice in which each node indicates a torsion class (in blue) within the category of the representation of the quiver of type $A_3$. The torsion classes occupy the points that are supported by the descending lines of the binary trees in the previous figure.
Figure 11. Above is the Tamari lattice in which each node represents a wire diagram corresponding to a 213-avoiding permutation in which the baseballs are placed on the location of the torsion classes of the previous figure.
Figure 12. The figure above is a Tamari lattice in which the nodes are labeled with the compliment of the torsion classes (indicated in filled and empty red balls). The filled balls denote the torsion-free class. The compliment of the torsion-free class is the Young diagram with gaps (indicated in blue).
Bibliography

[1] R. P. Stanley, “Catalan numbers,” *Cambridge University Press*, 2015.
[2] L. Nelson, “A recursion on maximal chains in the tamari lattices,” *Discrete Mathematics*, vol. 340, no. 4, pp. 661–677, 2017.
[3] O. Bernardi and N. Bonichon, “Intervals in catalan lattices and realizers of triangulations,” *Journal of Combinatorial Theory, Series A*, vol. 116, pp. 55–75, 2009.
[4] E. Barnard, A. Carroll, and S. Zhu, “Minimal inclusions of torsion classes,” *Algebraic Combinatorics*, vol. 2, no. 5, pp. 879–901, 2019.
[5] G. Baracchini, “Dyck paths and up-down walks,” pp. 1–2, 2016.
[6] D. André, “Solution directe du problème résolu par M. Bertrand,” *Comptes Rendus de l’Académie des Sciences*, vol. 105, pp. 436–437, 1887.
[7] K. Igusa and J. Ostroff, “Mixed cobinary trees,” *Journal of Algebra and Its Applications*, vol. 17, no. 09, p. 1850170, 2018.
[8] C.-M. Ringel, “The catalan combinatorics of the hereditary artin algebras,” *Recent Developments in Representation Theory, Contemporary Math*, vol. 673, pp. 51–177, 2016.