Spring 2021

Exploring Plane Partitions

Nicholas Heim
nwh8@zips.uakron.edu

Exploring Plane Partitions

Nicholas W. Heim

The University of Akron

Mathematics Honors Research Project

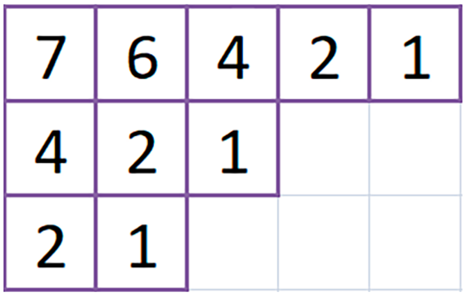
Dr. James P. Cossey

Abstract

The combinatorial theory of partitions has a number of applications including the representation theory of the symmetric group. A particularly important result counts the number of standard Young tableau of a given partition in terms of the hook lengths of the partition. In this paper we explore the analog of the hook length formula for plane partitions, the three-dimensional analog of ordinary partitions. We show that equality does not always hold but we conjecture that a certain inequality holds. Using a computer program, we verify this conjectured inequality for all 1982 plane partitions up to n = 11.

1. Introduction

Let n be a positive integer. A partition λ of n is a sequence $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_{k-1}, \lambda_k)$ of positive integers such that $\lambda_1 + \lambda_2 + \cdots + \lambda_{k-1} + \lambda_k = n$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{k-1} \geq \lambda_k$. Each λ_i is referred to as a part of λ. A Young diagram, drawn in the English notation, is a collection of left-justified rows of boxes with lengths corresponding to parts of λ in non-increasing order. To get the conjugate of λ, denoted λ', we merely switch the rows and columns of the diagram and record the new lengths. Further, the hook length of a box is defined as the count of the box itself and all boxes directly east and south. The following equation describes the hook length of each box where λ is the partition and (i, j) is the row and column of the box and figure 1 offers an example where the hook length has been filled in.



7	6	4	2	1
4	2	1		
2	1			

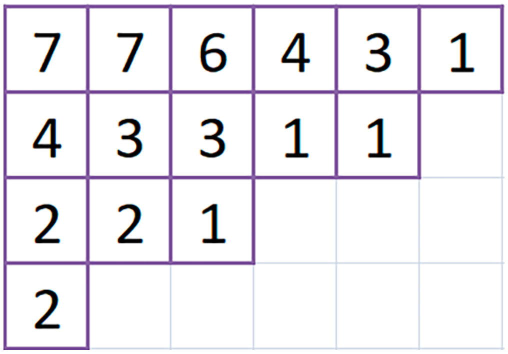
Figure 1: A partition of shape λ = (5, 3, 2) with hook lengths filled in.

$$h_\lambda(i,j) = (\lambda_i - j) + (\lambda'_j - i) + 1$$

A standard Young tableau (SYT) of shape λ is created by inserting all whole numbers in [1, *n*] into the diagram such that the numbers are increasing across each row and down each column. In fact, the following non-trivial theorem first discovered by Frame, Robinson, and Thrall [3] shows that the number of arrangements can be counted quite simply. A proof of this theorem will not be provided as it is beyond the scope of this paper. Using the following theorem, we can count the total number of SYT of shape λ:

$$|SYT(\lambda)| = H_\lambda = \frac{n!}{\prod_{(i,j)\in\lambda} h_\lambda(i,j)}$$

To assist with clarity, from here on out the previously mentioned partitions will be referred to as 2D partitions. Denoted Λ, plane partitions are a 3D analog of 2D partitions. One can be constructed by first creating a 2D partition to act as the "base" of the plane partition. From here, each square of the base is filled with a number representing the number of "cubes" stacked here, inclusive of the base. These numbers must be weakly decreasing across rows and down columns. If separated by levels, we have a stack of 2D partitions. We then reference a value of a 2D partition on level ℓ via $\lambda_\ell = (\lambda_{\ell,1}, \lambda_{\ell,2}, \ldots, \lambda_{\ell,k-1}, \lambda_{\ell,k})$. Let λ'_ℓ hold the same definition as before: the swapping of rows and columns. An example is provided here:



7	7	6	4	3	1
4	3	3	1	1	
2	2	1			
2					

Figure 2: A plane partition Λ.

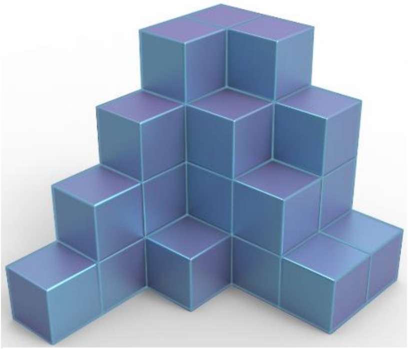


Figure 3: A 3D visualization of a different plane partition Λ = ((4, 4, 3, 1), (4, 3, 2, 1), (3, 1), (2), (1)).

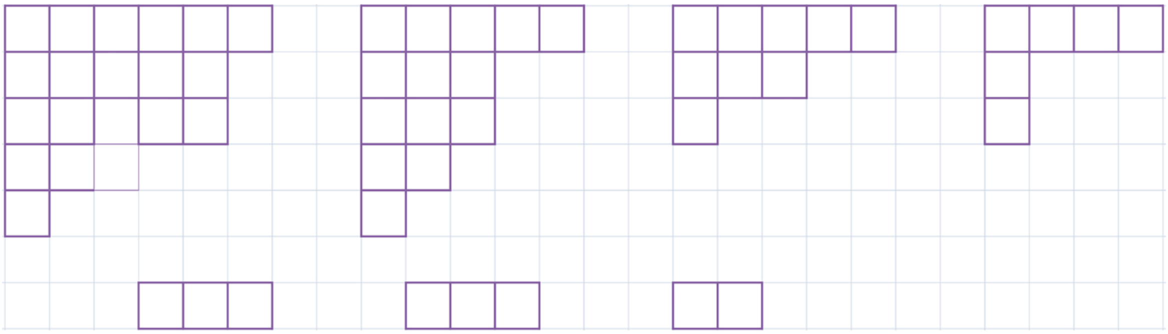


Figure 2: The individually sliced levels of the plane partition Λ in figure 2.

Then let $\Lambda_{i,j}$ denote the number representing the height at row *i* and column *j* in Λ. We can now define the hook length of the box of row *i*, column *j*, and level ℓ plane partition as follows:

$$h_\Lambda(\mathrm{i},\mathrm{j},\ell) = \left(\lambda_{\ell,i} - j\right) + \left(\lambda'_{\ell,j} - i\right) + \left(\Lambda_{\mathrm{i},\mathrm{j}} - \ell\right) + 1$$

Each term in the above equation, from left to right, represents the count of boxes to the east, south, and above respectively and finally the + 1 to count the box itself. We now state the conjecture that will be the focus of the remainder of the paper.

Conjecture: For a plane partition Λ, we have the following:

$$H_\Lambda = \frac{n!}{\prod_{(i,j,\ell)\in\Lambda} h_\Lambda(i,j,\ell)} \leq |SYT(\Lambda)|.$$

While an attempt to prove the whole conjecture will not be provided, a proof of a subcase for equality will be provided in section 3. Interestingly enough, the smallest counterexample of strict equality is the partition $\Lambda = ((2,1),(1,1))$. In this case, we have that $|SYT(\Lambda)| = 7.5$ while the actual count of $H_\Lambda \; is \; 8$.

2. Methods

For many of the results a Python3 program was used to assist in finding patterns and results for longer computations. Much of this section will reference the code which can be found in *partitionCreator.py*.

The algorithm contained within countPSYT (count plane SYT) works around the passed partition instance. To generate all of the Young Tableau of a fixed shape Λ we take the numbers in [1, n] and generate all possible permutations. Once a permutation has been generated and placed into position it must then be checked to determine if it is a PSYT. This is done in verifyPSYT by checking the values east, south, and above the block at position (i, j, k). By far, this is the longest part of the program with a O(*n*!).

The next large algorithm is that of genPartitions. This takes in a single variable, *size*, and generates all partitions of that size. The algorithm has three main levels. The first level is shaped around managing sets of partitions. Here, a set container is used to save space by only taking in unique partitions. Due to this some compromises are to be made. Namely, some set operations are used here to avoid recomputing results.

In the outermost level of the algorithm, we loop through all values in uniqueParts (as it updates) which stores all the unique partitions. To avoid rerunning partitions in the second loop we only loop through, mathematically, uniqueParts – usedPartitions. Then, as each item in the resulting set is selected, it is added to usedPartitions. The expected returns from putting a partition through the remainder of the algorithm is another set of partitions, as described in the next paragraph. This set is then added into uniqueParts and the algorithm begins again. This will repeat until we have gone through every element of uniqueParts. Once the computations have finished the resulting partitions are saved to a file.

In the second level findLegalBlock is passed a single partition from the first part. Using this partition, it loops through each value in each row searching for a movable block. A movable block is defined as a block that can be removed from the top of the current position without the resulting partition violating any rules of a plane partition. Once a block is found the partition and the position of the movable block are passed to findLegalPositions. The expected return from findLegalPositions is a set of all partitions that result from the removal and replacement of the block in a legal manner.

Finally, in the third level of the algorithm, the block is first removed from the partition. Afterwards, the removal conditions are checked to see if it can be placed at another position. For simplicity, all positions are checked. If the block can be placed in a certain position, then the partition is added to a set with the block in place. Once all positions have been checked, the resulting set is returned.

2.1 Evidence for the Main Conjecture

Table 1: Algorithm Results

n	T. Count	A. Count	Largest PSYT
1	1	1	1
2	3	3	1
3	6	6	2
4	13	13	6
5	24	24	12
6	48	48	30
7	86	86	96
8	160	160	336
9	282	282	1540
10	500	500	8640
11	859	859	33372

Results for the above algorithms have been collected for values [1, 11]. Running the algorithm on this range took close to 4 hours. All partition shapes have been generated for values [1, 20]. All of the results for the values [1, 11] were then checked by a quick loop in the main function. There are no counter examples in what has been calculated: 1982 cases. In table 1, T. and A. Count represent Theoretical and Actual counts, respectively. The theoretical count was calculated using python code written by John Burkardt [2].

3. Analysis of a Subcase

Let Λ be a plane partition of size *n*. Let there be one row east of length *k*, one column south of length ℓ, and one stack of height *m*. Then we have the following theorem:

$$\frac{n!}{\prod_{(i,j,\ell)\in\Lambda} h_\Lambda(i,j,\ell)} = |SYT(\Lambda)|$$

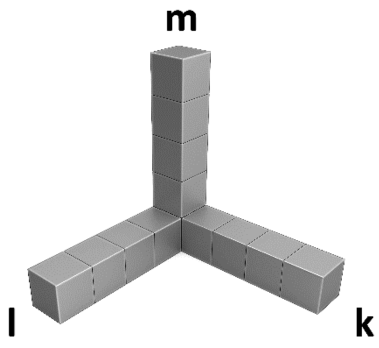


Figure 3: Reference image for the theorem.

Proof. To construct an SYT, 1 must be in the bottom north west block. Due to the shape of this plane partition we now need to place the whole numbers [2, *n*]. With 1 being placed the previously mentioned row, stack, and column are nearly independent problems. Thus, we are able to pick numbers, with any arrangement of *k*, ℓ, and *m* in the following way:

$$\binom{n\ -\ 1}{k\ -\ 1}\binom{n\ -\ k}{\ell\ -\ 1}\binom{n\ -\ k\ -\ \ell+1}{m\ -\ 1}=\binom{n\ -\ 1}{k\ -\ 1}\binom{n\ -\ k}{\ell\ -\ 1}$$

$$=\frac{(n-1)!\,(n-k)!}{(k-1)!\,(n-k)!\,(\ell-1)!\,(m-1)!}$$

$$=\frac{(n-1)!}{(k-1)!\,(\ell-1)!\,(m-1)!}=\frac{n!}{\prod_{(i,j,\ell)\in\Lambda}h_\Lambda(i,j,\ell)}=|SYT(\Lambda)|$$

■

4. Future Work

In order to use the program for larger values of *n* parallelization of a few functions will be necessary. Currently, the longest computation time is taken by countPSYT due to the O(n!) time complexity. However, a close second is genPartitions. Likely the simplest method of parallelization would be to parallelize genPartitions and have multiple threads working with individually passed partition shapes.

In terms of continuing the research on counting $|SYT(\Lambda)|$ the next step is to find more countable subsets of plane partitions and generalize the results as much as possible.

We are also interested in proving this conjecture. In Sagan's book (p. 164), there is a proof by Novelli, Pak, and Stoyanovskii using an algorithmically defined bijective proof of the 2D equality [4]. This could perhaps generalize to the 3D inequality by proving injectivity. There is also an inductive proof by Bandlow which could potentially generalize to the 3D inequality [1].

References

[1] Bandlow, J. (2008). An Elementary Proof of the Hook Formula. *The Electronic Journal of Combinatorics*, 15(1). https://doi.org/10.37236/769

[2] Bukardt, J. (2014) plane_partition_num.py (Version 1.0) [Source code]. https://people.sc.fsu.edu/~jburkardt/py_src/polpak/plane_partition_num.py

[3] Frame, J., Robinson, G., & Thrall, R. (1954). *The Hook Graphs of the Symmetric Group*. Canadian Journal of Mathematics, 6, 316-324.

[4] Sagan, B. E. (1991). In *The symmetric group: representations, combinatorial algorithms, and symmetric functions* (p. 164). essay, Wadsworth &amp; Brooks/Cole Advanced Books &amp; Software.