THE SPERNER PROPERTY FOR INTEGER PARTITION POSETS

BRIAN BOWERS, KERRY GANNON, KATIE JONES, ANNA KIRKPATRICK

Abstract

We examine the integer partition poset, $\mathcal{P}_n$, ordered by a covering relation to determine whether or not it has the Sperner property. To show that $\mathcal{P}_n$ is Sperner we must prove it satisfies unimodality and the bipartite matching property. It has been shown computationally by Canfield for $n \leq 2000$ and asymptotically by Szekeres for n sufficiently large that $\mathcal{P}_n$ is unimodal. Computationally, we have increased the known bounds for $\mathcal{P}_n$ being unimodal to 25,000. It was previously shown by Canfield that the bipartite matching property holds for $n \leq 45$. We have computationally worked toward improving this bound. We prove that the bipartite matching property holds between each pair of consecutive level sets from level $k = \left\lceil \frac{n}{2} \right\rceil$ to $k = n$. We also prove that the bipartite matching property holds between other specific levels of the poset $\mathcal{P}_n$.

Acknowledgement

We would like to thank our mentor Neil J. Calkin and our graduate student advisor Janine E. Janoski. We would also like to thank the other organizers of the REU, Jim Brown and Kevin James. Additionally, we would like to thank Rodney Canfield, Teena Carroll, Dana Randall, and Matthew Saltzman for their indispensable assistance and input.

Bowers, Calkin, Gannon, Janoski, and Jones were partially supported by the NSF grant DMS 0552799.

Kirkpatrick was partially supported by the South Carolina Governor's School for Science and Mathematics Summer Program for Research Interns.

1. Introduction

Definition. *Let n be a nonnegative integer. A rank k partition of n is a k-tuple of positive integers $(\lambda_1, \ldots, \lambda_k)$ satisfying*

$$n = \sum_{i}^{k} \lambda_i, \qquad 1 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_k.$$

We write

$$(1^{a_1} 2^{a_2} \cdots) \text{ to represent } \left(\underbrace{1, ..., 1}_{a_1}, \underbrace{2, ..., 2}_{a_2}, \ldots \right),$$

where

$$\sum_{i \geq 1} i a_i = n$$

Example. *The following table shows all of the partitions of 4 in both notations and their respective ranks.*

First Representation	Second Representation	Rank
$(1,1,1,1)$	1^4	4
$(1,1,2)$	$1^2 2$	3
$(1,3)$	13	2
$(2,2)$	2^2	2
(4)	4	1

Let $p(n,k)$ denote the number of partitions of n into k parts, and $P(n,k)$ denote the number of partitions of n into at most k parts.

Definition. *If λ and λ' are two partitions of the same number, then we say that λ is covered by λ', written $\lambda \lessdot \lambda'$, if two summands of λ can be added to form λ'.*

Example. $(1,1,1,2,3) \lessdot (1,1,1,5)$

Definition. *We define $\mathcal{P}_n$ to be the set of all partitions of the nonnegative integer n with the ordering induced by the reflexive and transitive closure of the covering relation. We will denote this transitive closure by $\leqslant$. $\mathcal{P}_n$ is a poset (partially ordered set) under $\leqslant$.*

Definition. *We say that λ and λ' are comparable if $\lambda \leqslant \lambda'$ or $\lambda' \leqslant \lambda$. Two partitions which are not comparable are called incomparable.*

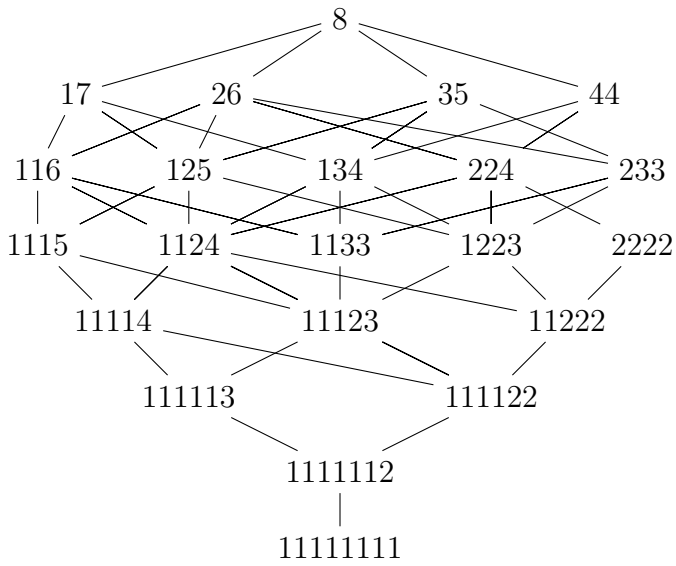
We can organize $\mathcal{P}_n$ using a graphical representation called a Hasse Diagram in the following way: We create the graph with vertices

$$V = \{\lambda : \lambda \in \mathcal{P}_n\}$$

arranged in rows, called "level sets", according to rank. We then add edges

$$E = \{\{\lambda, \lambda'\} : \lambda, \lambda' \in \mathcal{P}_n \text{ and either } \lambda \lessdot \lambda' \text{ or } \lambda' \lessdot \lambda\}.$$

Example. *The Hasse diagram for $\mathcal{P}_8$:*



Definition. *A chain is a set of partitions $\{\lambda', \lambda'', \ldots, \lambda^{(r)}\}$ such that $\lambda' \lessdot \lambda'' \lessdot \ldots \lessdot \lambda^{(r)}$.*

Definition. *We define a chain partition of $\mathcal{P}_n$ as a set of non-intersecting chains whose union is $\mathcal{P}_n$.*

Definition. *An antichain is a set of partitions which are pairwise incomparable.*

The elements within a level set form an antichain. For example, 111113 and 111122 in $\mathcal{P}_8$ form an anitchain.

We are interested in determining if $\mathcal{P}_n$ is Sperner for all nonnegative integers n.

Definition. *A ranked poset is said to be Sperner if the size of its largest antichain is equal to the size of its largest level set.*

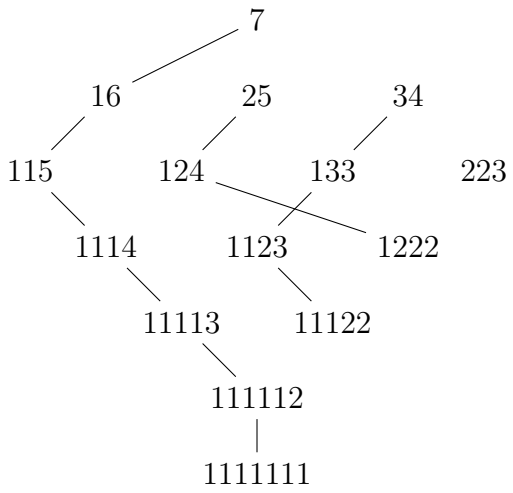
The following definitions are required to state an equivalent formulation of the Sperner property:

Definition. *We say that $\mathcal{P}_n$ is unimodal if there exists a number $k_{1,n}$, called the mode, such that $p(n,k) < p(n,k+1)$ for $k < k_{1,n}$, and $p(n,k) > p(n,k+1)$ for $k > k_{1,n}$.*

Definition. *$\mathcal{P}_n$ is said to have the bipartite matching property if, for any two consecutive level sets, $\Lambda_{n,a}$ and $\Lambda_{n,b}$ with $p(n,a) < p(n,b)$, there is an injective matching using the edges of the Hasse Diagram of $\mathcal{P}_n$ from $\Lambda_{n,a}$ onto $\Lambda_{n,b}$. We call such a matching a maximum matching.*

With these definitions, we can state our equivalent formulation of the Sperner property. If a ranked poset has the bipartite matching property and it is unimodal, then it is Sperner.

For example, $\mathcal{P}_7$ below satisfies the Sperner property because it is both unimodal and has the bipartite matching property.



Some progress has been made toward proving that $\mathcal{P}_n$ is Sperner for all n using this equivalent definition.

In 1952, Szekeres proved that, for "sufficiently large n," $\mathcal{P}_n$ is unimodal [8]. It has been previously computationally verified that $\mathcal{P}_n$ is unimodal for $n \leq 2000$. Canfield proved in 2003 that $\mathcal{P}_n$ is Sperner for all $n \leq 45$ [2]. Canfield proved this by using the Ford-Fulkerson Algorithm, to show that for $1 \leq n \leq 45$, $\mathcal{P}_n$ has the bipartite matching. Because Canfield proved that these integer posets had the bipartite matching property, he was able to conclude that $\mathcal{P}_n$ is Sperner in each of these cases since it had already been computationally verified that $\mathcal{P}_n$ is unimodal [2].

The remainder of the paper will be broken into four sections. In section 2, we will examine various computational and proof-based approaches to determining the unimodality of $\mathcal{P}_n$. Next, in section 3, we will study the Bipartite Matching Property of $\mathcal{P}_n$, again using both computational and proof-based methods. We also include an appendix of the Sage methods that we created.

2. Unimodality

We will explore the unimodality of $\mathcal{P}_n$ in two ways. First, we will use computational approaches to prove unimodality for finitely many n. Then, we will examine a 1952 paper by Szekeres [8] to determine the range of n values for which his proof of unimodality holds.

2.1. Computational Approaches for Unimodality.

Using several computational approaches, we have determined that $\mathcal{P}_n$ is unimodal for all $n \leq 25,000$.

We first expanded the values of n for which $\mathcal{P}_n$ is known to be unimodal from $n \leq 2,000$ to $n \leq 10,000$. We created a Sage algorithm that generates a matrix for which the (n, k) entry represents $p(n, k)$. This algorithm defines all entries for which $k = 1$ or $n = k$ to be 1 and then calculates the remaining entries according to the recursive formula $p(n, k) = p(n-1, k-1) + p(n-k, k)$. We then used another algorithm to ensure that each row of the matrix was unimodal. Combining these two algorithms, we concluded that $\mathcal{P}_n$ is unimodal up to $n = 10,000$.

We also created a function that finds the column number of the maximum element of each row of the $p(n, k)$ matrix, effectively calculating the rank of the mode of $\mathcal{P}_n$, where n is the row of the matrix. Using this algorithm, we were able to calculate the rank of the mode of $\mathcal{P}_n$ up to $n = 10,000$.

Additionally, we used an improved algorithm that encompasses the utilities of the previous algorithms. First, it finds the values of p(n,k) for a specified range of n values. Then, it determines if each $\mathcal{P}_n$ is unimodal and returns the k-value of the column of the first appearance of the mode if applicable. This function is an improvement over the previous functions in that, at each step, it deletes all p(n,k) values which become unnecessary for future computations. This decreased the necessary data storage by over 50% and allowed us to test the unimodality of larger values of n. Using this improved algorithm, we found that $\mathcal{P}_n$ is unimodal for all $n \leq 25,000$.

We also created another algorithm that generates the matrix of $p(n, k)$ values modulo a specified prime. In this manner, we can compute the same matrix modulo many relatively small primes and later recover specific values of the original matrix using the Chinese Remainder Theorem. The advantage of this approach is that it decreases the file sizes for each individual stored matrix, thereby allowing us to check unimodality for still larger values of n.

2.2. Szekeres' Proof of Unimodality. [1]

[1] In this section we will use Szekeres' notation from [7]

We would like to prove that the integer partition poset is unimodal for all n, and fortunately a great deal of the theoretical work required has already been done by Szekeres in [7]. He proves the following theorem regarding unimodality:

Theorem. *If n is sufficiently large, then there exists a number $k_1 = k_1(n)$ such that $p(n,k) < p(n,k+1)$ for $k \leq k_1, p(n,k) > p(n,k+1)$ for $k > k_1$.* [7]

To do so, he uses the recursion relation $p(n,k) = P(n-k,k)$. He first determines the size of $P(n,k)$ asymptotically and uses this result in his proof of the theorem above. To determine the size of $P(n,k)$, we let

$$F(w) = \prod_{\nu=1}^{k} (1-w^{\nu})^{-1} = \sum_{n=0}^{\infty} P(n,k)w^n.$$

By Cauchy's theorem we can use a contour integral to estimate this series, given by

$$P(n,k) = \frac{1}{2\pi i} \int F(w) w^{-n-1} dw,$$

centered at 0 with a radius $\rho < 1$, such that ρ is a real number taken as the positive root of $\dfrac{wF'}{F(w)} = n$. If we recognize the left side of this equation as the derivative of $log(n)$ we can rewrite this as the sum

$$\sum_{\nu=1}^{k} \frac{\nu\rho^{\nu}}{1-p^{\nu}} = n.$$

If we set $\alpha = -log\rho$, then we can make a change of variables to see

$$\sum_{\nu=1}^{k} \frac{\nu}{e^{\nu\alpha}-1} = n.$$

We would like to find bounds on ρ so that we may better estimate Szekeres' integral, and thus find explicit bounds on the size on n. To do this, we would first like to find the relative size of α. So far, we have considered three cases for α.

Case 1: Let α be very small, such that $\alpha \ll \frac{1}{k}$. We have concluded that in this case we would like α to be much smaller than 1, so that $\dfrac{k^2}{n} = o(1)$, or $k = o(n^{\frac{1}{2}})$. In this case, the integral equation for $P(n,k)$ has a contour with radius about $\rho = 1 - \dfrac{k}{n}$.

Case 2: Let α be very large, so that $e^{-\alpha} = o(1)$. Then, upon doing some analysis, we see that $n \cong e^{-\alpha}$, which would imply that $\alpha \cong -logn$, but this cannot happen because we have restricted $\alpha > 0$.

Case 3: We are also interested in the case where $\alpha = \frac{u}{k}$ where $u > 0$ is a constant. This case has not yet been resolved.

Szekeres then lets $f(x) = \dfrac{x}{e^{\alpha x} - 1}$ and $\phi(t) = \dfrac{t}{e^t - 1}$ so that he may use the well-known Euler Maclaurin summation formula

$$f(1)+...+f(k) = \int_0^k f(x)dx+\frac{1}{2}[f(k)-f(0)]+\sum_{\nu=0}^{m} \frac{1}{(\nu+1)!}B_{\nu+1}[f^{\nu}(k)-f^{\nu}(0)]+$$

$$\int_0^k P_{m+1}(x)^{m+1}(x)dx$$

where B_ν is the νth Bernoulli number and P_{m+1} is the mth Bernoulli polynomial to estimate the size of $\displaystyle\sum_{\nu=1}^{k} \frac{\nu}{e^{\alpha\nu} - 1}$, which is equal to n.

Using the substitution $t = \alpha x$ and writing $u = \alpha k$, Szekeres is able to write

$$\int_0^k \frac{x}{e^{\alpha x} - 1}dx$$

as

$$\alpha^{-2} \int_0^u \frac{t}{e^t - 1}dt.$$

Using this change of variables, Szekeres is able to write

$$n = \sum_{\nu=1}^{k} \frac{\nu}{e^{\alpha}\nu} - 1 = \alpha^{-2} \int_0^u \frac{t}{e^t - 1}dt + \frac{1}{2}\alpha^{-1}\left(\frac{u}{e^u - 1} - 1\right) +$$

$$+\frac{1}{12}\left(\frac{1}{e^u - 1} - \frac{ue^u}{(e^u - 1)^2} + \frac{1}{2}\right)+\alpha^2 A(u)+...+\alpha^{m-2}A(u)+O\left(\alpha^{m-1}\right),$$

where he computes out the first term in the sum and writes the rest as a power of α times A(u), where A(u) is some polynomial bounded for $u \geq 0$. The final integral term is treated as the error bound.

We are then interested in bounding the ϕ and its derivatives. In [7], Szekeres finds an upper bound for $|\phi^{(m)}(t)|$, however it appears that it will be sufficient to only bound up $|\phi^{(4)}(t)|$. We have so far determined bounds for $|\phi(t)|$ and $|\phi'(t)|$.

Theorem. $|\phi(z)| < \dfrac{e}{e-1} z e^{-z}$

Proof. Suppose $z > 1$. Now suppose, for contradiction, that

$$\left| \frac{e^z}{e^z - 1} \right| > \frac{e}{e-1}.$$

The term is positive, so we can remove the absolute values to see

$$e^z > \frac{e}{e-1} e^z - \frac{e}{e-1},$$

so that

$$e^z \left(1 - \frac{e}{e-1} \right) > -\frac{e}{e-1},$$

which means

$$\frac{1}{e-1} e^z < \frac{e}{e-1}.$$

Thus,

$$e^z < e,$$

and

$$z < 1.$$

This is a contradiction of our initial supposition, so it must be the case that

$$\left| \frac{e^z}{e^z - 1} \right| < \frac{e}{e-1},$$

which means that

$$|\phi(z)| = \left| \frac{z}{e^z - 1} \right| < \frac{e}{e-1} z e^{-z}$$

□

Theorem. *With ϕ as given above,* $|\phi'(z)| < \dfrac{e}{e-1} z e^{-z}$

Proof. Suppose $z > 1$. Let $f(z) = \dfrac{1}{z} - \dfrac{1}{e^z - 1}$. Then

$$f(z) = \frac{1}{z} \left(1 - \frac{z}{e^z - 1} \right) = \frac{1}{z} \left(\frac{e^z - z - 1}{e^z - 1} \right)$$

Since $0 < \frac{1}{z} < 1$ and $0 < \frac{e^z - z - 1}{e^z - 1} < 1$, we find that $0 < f(z) < 1$. Thus,

$$\begin{aligned}
\left|\frac{\phi'(z)}{\phi(z)}\right| &= \left|\frac{\frac{1}{e^z-1} - \frac{ze^z}{(e^z-1)^2}}{\frac{z}{e^z-1}}\right| \\
&= \left|\frac{1}{z} - \frac{e^z}{e^z-1}\right| \\
&= \left|-1 + \frac{1}{z} - \frac{1}{e^z-1}\right| \\
&= |-1 + f(z)|\,.
\end{aligned}$$

Now, applying our bounds for $f(z)$, we see that

$$\left|\frac{\phi'(z)}{\phi(z)}\right| < 1.$$

Therefore,

$$|\phi'(z)| < |\phi(z)| < \frac{e}{e-1} z e^{-z}.$$

□

Ultimately, we would like to find the specific value of n for which Szekeres' theorem on unimodality holds or at least to find sufficient bounds on the size of n so that we may computationally verify unimodality up to this point. Szekeres claims that we can discard the condition in his theorem that n be large if we replace the O-notations in his proofs by explicit constants and check the conjecture for smaller values of n.

Before we continue, we must first first make a brief digression into our usage "big O notation." We use this notation to describe the error term in the approximation of a mathematical function. So, for example, $O(n^2)$ indicates that the error term is smaller in absolute value than some constant times n^2 for n sufficiently close to 0. We arrive at this type of approximation by truncating infinite sums and condensing the value of the remaining terms within a reasonable error bound.

Using techniques in analysis, we found explicit constants for several of the error terms in Szekeres 1. We have determined the value of $O(1)$ in [8] in the following way: we are given

$$\int_0^\infty |\phi^{(m+1)}(t)|dt = O(1)$$

with the condition that $\phi^{\nu}(t)$ is bounded for $t \geq 0$ and tends to zero monotonically for $t \geq t_{\nu}$, where $\phi(t) = t(e^t - 1)^{-1}$. So we have $|\phi^{\nu}(t)| \leq M_{\nu}$ for all t. If we fix $\nu = m$ we can split our integral and see that

$$\int_0^{\infty} |\phi^{(m+1)}(t)|dt = \int_0^{t_{m+1}} |\phi^{\nu}(t)|dt + \int_{t_{m+1}}^{\infty} |\phi^{(m+1)}(t)|dt$$

where,

$$\int_0^{t_{m+1}} |\phi^{\nu}(t)|dt < M_{m+1}t_{m+1}$$

and

$$\int_{t_{m+1}}^{\infty} |\phi^{(m+1)}(t)|dt = [\phi^{(m)}(t)]_{t_{m+1}}^{\infty} = 0 - \phi^{(m)}(t_{m+1}) < M_m$$

and thus our bound becomes $c_1 = M_{m+1}t_{m+1} + M_m$.

It appears that it may be difficult to provide explicit constants and find an effective bound for n theoretically, and it may be more feasible to find an sufficiently accurate estimate. In [7], Szekeres determined that, for sufficiently large n, the rank of the mode of $\mathcal{P}_n$ is equal to

$$c\sqrt{n}L + c^2\left(\frac{3}{2} + \frac{3L}{2} - \frac{L^2}{4}\right) - \frac{1}{2} + O\left(\frac{log^4 n}{\sqrt{n}}\right)$$

where

$$c = \frac{\sqrt{6}}{\pi}$$

and

$$L = log\big(c\sqrt{n}\big)$$

[8].

We have created a spreadsheet which compares the actual rank of the mode to Szekeres' estimate as well as the actual error of the estimate to the big-O expression from Szekeres' formula. We observed that up to 10,000, Szekeres' prediction is always within ± 2 of the actual mode. It is possible that such comparisons can lead to estimates of the minimum n value for which Szekeres' formula is correct.

By using this equation, if we fix n, we can figure out the rank of the mode, k_1.

Theorem 1.

$$k = \nu\beta$$

$$n = \beta^2 \int_0^{\nu} \frac{t}{e^t - 1}\, dt + \frac{1}{2}\beta\left(\frac{\nu}{e^{\nu} - 1} - 1\right) + \frac{1}{12}\left(\frac{1}{2} + \frac{1}{e^{\nu} - 1} - \frac{\nu e^{\nu}}{(e^{\nu} - 1)^2}\right)$$

and

$$P(n,k) = \frac{1}{2\pi} B_0^{-\frac{1}{2}} \beta^{-2} exp\{2\beta \int_0^\nu \frac{t}{e^t - 1}\, dt - (\nu\beta + \frac{1}{2}) log(1 - e^{-\nu})$$

$$+ \frac{1}{2}\left(\frac{\nu}{e^\nu - 1} - 1\right)\}[1 + B_1(\nu)\beta^{-1} + \cdots + B_{m-1}(\nu)\beta^{-m+1} + O(\beta^{-m})]$$

for any given $m > 0$*, where*

$$B_0 = \int_0^\nu \frac{t^2 e^t}{(e^t - 1)^2}\, dt,$$

and

$$B_1(\nu) = -[\frac{1}{12}\nu e^\nu (e^\nu - 1)^{-2} + B_0^{-1}\left\{\frac{1}{8} + \frac{1}{4}\nu^2 e^\nu (e^\nu - 1)^{-2}\right\}$$

$$+ B_0^{-2}\left\{\frac{1}{8}\nu^4 (e^\nu + e^{2\nu})(e^\nu - 1)^{-3} - \frac{3}{4}\nu^3 e^\nu (e^\nu - 1)^{-2}\right\}$$

$$+ \frac{5}{24} B_0^{-3} \nu^6 e^{2\nu} (e^\nu - 1)^{-4}].$$

If we solve for β in terms of ν and k, and plug into the equation of n in the theorem above, we can solve for ν and, in turn, β. Using these values and the equation for n, an approximation of $P(n,k)$ can be made explicit.

One way in which we can possibly find a "large enough n" that satisfies Szekeres' theorem, is to compute the exact value of $P(n,k)$ and compare it to the approximation produced by Szekeres' formulas. We can find the precise number of partitions of n into at most k parts, by a recursion formula,

$$P(n,k) = \sum_{j=1}^{k} p(n,j)$$

where

$$p(n,j) = p(n-1, j-1) + p(n-j, j).$$

If the difference between the two values, or error, is sufficiently small, then the fixed n may be an adequately "large enough n". If it is not sufficiently small, fix new values of n until error is minimized.

Denote $P_a(n,k)$ as the actual value of $P(n,k)$ produced by the recursion formula, and $P_e(n,k)$ as the estimated value of $P(n,k)$ by Szekeres' formula. Also, let the actual mode of an integer n be denoted as $k_{a,n}$ and the estimated mode be denoted $k_{e,n}$. The actual and estimated modes are listed below:

$k_{a,100} = 18$	$k_{a,5,000} = 223$	$k_{a,10,000} = 341$
$k_{e,100} = 17.65654513$	$k_{e,5,000} = 222.6936757$	$k_{e,10,000} = 341.1610199$

Because we are only concerned with integers n for our integer partition posets, we must use the ceiling and floor of $k_{n'}$ to find $P_e(n,k)$. The values of $P_a(n,k)$ and $P_e(n,k)$ are listed below for $n = 100, 5000, 10000$:

$P_a(100,17) = 6.4684584 \cdot 10^7$
$P_e(100,17) = 6.491125405671335711264330798147086224386088978643 5 \cdot 10^7$

$P_a(100,18) = 7.5772412 \cdot 10^7$
$P_e(100,18) = 7.605340252560065165335807055065464072473169141292 1 \cdot 10^7$

$P_a(5000,222) = 6.045371862293365908731689686213203333070178866520 6 \cdot 10^{73}$
$P_e(5000,222) = 6.050036379143173802792905286250823779205053467294 4 \cdot 10^{73}$

$P_a(5000,223) = 6.166780273576910888230593099560808255039074899083 3 \cdot 10^{73}$
$P_e(5000,223) = 6.171552682508774701382497872490204710029297623210 0 \cdot 10^{73}$

$P_a(10000,341) = 1.304086530657895811808089283143747471526080536475 7 \cdot 10^{106}$
$P_e(10000,341) = 1.304829691943197361815697818718597239556428714319 7 \cdot 10^{106}$

$P_a(10000,342) = 1.322120199963872711501617498730903598288371817584 8 \cdot 10^{106}$
$P_e(10000,342 = 1.322874779544476074492957281737983869306662382589 6 \cdot 10^{106}$

With this information, $\Delta P(n,k)$ can be determined by

$$\Delta P(n,k) = P(n,k_e) - P(n,k_a),$$

and the error percentage can be determined by

$$\%\text{error} = \left|\frac{\Delta P(n,k)}{P(n,k_a)}\right|.$$

$$\Delta P(100,17) = 2.266700567133571126433079814708622438608897864 35 \cdot 10^5$$
$$\%\text{error } P(100,17) = 0.003504236136\%$$

$$\Delta P(100,18) = 2.809905256006516533580705506546407247316914129 21 \cdot 10^5$$
$$\%\text{error } P(100,17) = 0.003708348701\%$$

$$\Delta P(5000,222) = 4.664516849807894061215600037620446134874600773 7 \cdot 10^{70}$$
$$\%\text{error } P(5000,222) = 0.0007715847687\%$$

$$\Delta P(5000,223) = 4.772408931863813151904772929396454990222724126 7 \cdot 10^{70}$$
$$\%\text{error } P(5000,222) = 0.0007738898939\%$$

$$\Delta P(10,000,341) = 7.431612853015500076085355748497680303481778439 \cdot 10^{102}$$
$$\%\text{error } P(10,000,341) = 0.0005698711455\%$$

$$\Delta P(10,000,342) = 7.545795806033629913397830070802710182905650047 \cdot 10^{102}$$
$$\%\text{error } P(10,000,342) = 0.0005707344768\%$$

The error percentage for $n = 10,000$ is smaller than for $n = 5,000$ and $n = 100$, which may imply that $n = 10,000$ is a sufficiently large enough n to satisfy Szekeres' Theorem that $\mathcal{P}_n$ is unimodal.

Szekeres also examines $\Delta log(p(n,k))$ in depth, where

$$\Delta log(p(n,k)) = log((p,k+1)) - log(p(n,k))$$
$$= -log(1-e^{-u}) - \alpha[1 + \frac{1}{e^u-1} + \frac{1}{2}A_0^{-1}\frac{u^2(e^u+1)}{(e^u-1)^2} - \frac{3}{2}A_0^{-1}\frac{u}{e^u-1} + \frac{1}{2}A_0^{-2}\frac{u^4e^u}{(e^u-1)^3}]$$
$$+\frac{3}{2}A_0^{-1}\alpha^2 + O(u^2\Delta\alpha),$$

where $u = \frac{k\pi}{\sqrt{6}\sqrt{n}}$, $A_0 = \frac{\pi^2}{3} - \int_u^\infty x^2e^{-x}$, and $\alpha = \frac{1}{\frac{6}{\pi^2}u} \cdot \frac{k}{n}$, and $O(u^2\Delta\alpha)$ is our error term.

We note the following remarks:

Remark. *If $\Delta log(p(n,k))$ is positive, then $p(n,k+1) > p(n,k)$.*

Remark. *If $\Delta log(p(n,k))$ is negative, then $p(n,k) > p(n,k+1)$.*

Remark. *If $\Delta log(p(n,k))$ is zero, then $p(n,k) = p(n,k+1)$.*

Remark. *If $\Delta log(p(n,k))$ is positive for $k < k_1$, where k_1 is the mode, and $\Delta log(p(n,k))$ is negative thereafter, then $\mathcal{P}_n$ is unimodal.*

Another approach to finding a "sufficiently large enough n" is to test this approximation for $\Delta log(p(n,k))$ for k near and around the mode for several n and to compare it to the actual, computed value of $\Delta log(p(n,k))$.

Let $\Delta_a log(p(n,k))$ denote the actual value of $\Delta log(p(n,k))$, and $\Delta_e log(p(n,k))$ denote Szekeres' approximation for $\Delta log(p(n,k))$.

$$\Delta_a log(p(100,17)) = 0.005905118538$$
$$\Delta_a log(p(100,18)) = -0.01837046188$$
$$\Delta_a log(p(5000,222)) = 0.0001173838973$$
$$\Delta_a log(p(5000,223)) = -0.0002627018839$$
$$\Delta_a log(p(10000,340)) = 0.0001501608640$$
$$\Delta_a log(p(10000,341)) = -0.00003598257899$$
$$\Delta_a log(p(10000,342)) = -0.0002194270492$$

$$\Delta_e log(p(100,17)) = -0.03014042401$$
$$\Delta_e log(p(100,18)) = -0.04184253798$$
$$\Delta_e log(p(5000,222)) = -0.0009911772790$$
$$\Delta_e log(p(5000,223)) = -0.001303067991$$
$$\Delta_e log(p(100,341)) = -0.0006048430625$$
$$\Delta_e log(p(100,342)) = -0.0007621053514$$

With this information, $\Delta\Delta log(p(n,k))$ can be determined by

$$\Delta\Delta log(p(n,k)) = \Delta_e log(p(n,k)) - \Delta_a log(p(n,k)),$$

and the error percentage can be deteremined by

$$\%\text{error} = \left| \frac{\Delta\Delta log(p(n,k))}{\Delta_a log(p(n,k))} \right|.$$

$$\Delta(\Delta log(p(100,17))) = -0.03604554255$$
$$\%\text{error } \Delta log(p(100,17)) = 6.104118371\%$$

$$\Delta(\Delta log(p(100,18))) = -0.02347207610$$
$$\%\text{error } \Delta log(p(100,18)) = 1.277707455\%$$

$$\Delta(\Delta log(p(5000,222))) = -0.001108561176$$
$$\%\text{error } \Delta log(p(5000,222)) = 9.443894789\%$$

$$\Delta(\Delta log(p(5000,223))) = -0.001040366107$$
$$\%\text{error } \Delta log(p(5000,223)) = 3.960253697\%$$

$$\Delta(\Delta log(p(100000,341))) = 0.0005688604835$$
$$\%\text{error } \Delta log(p(100000,341)) = .9405092309\%$$

$$\Delta(\Delta log(p(100000,342))) = 0.0005426783022$$
$$\%\text{error } \Delta log(p(100000,342)) = .7120778003\%$$

Similar to the error percentage for $P(n,k)$, the error percentage for $\Delta log(p(n,k))$ for $n = 10,000$ is smaller than for $n = 5,000$ and $n = 100$. This reinforces that $n = 10,000$ is a good estimate as a sufficiently large enough n.

However, the $\Delta_e log(p(n,k))$ listed above, do not indicate that the k's selected above are the mode, because the sign of $\Delta_e log(p(n,k))$ does not change from positive to negative. The k values for which $\Delta_e log(p(n,k))$ does change sign, which would signify the mode, are listed below, in comparison with Szekeres' predicted mode produced by

$$k_1 = c\sqrt{n}L + c^2\left(\frac{3}{2} + \frac{3L}{2} - \frac{L^2}{4}\right) - \frac{1}{2} + O\left(\frac{log^4 n}{\sqrt{n}}\right).$$

k_n	Szekeres' Estimated Mode	Δ Sign for $\Delta(\Delta log(p(n,k)))$
k_{100}	17.65654513	14
$k_{5,000}$	222.6936757	218
$k_{10,000}$	341.1610199	337
$k_{15,000}$	436.7677659	432
$k_{20,000}$	519.9072324	515
$k_{100,000}$	1358.790795	1354
$k_{200,000}$	2041.843714	2037
$k_{1,000,000}$	5191.672085	5186

The difference in the modes produced by these formulations may indicate an error which is worth investigating further.

2.3. **Log-Concavity.** We are also investigating the log-concavity of $p(n,k)$. We are interested in this because it would imply the unimodality of $\mathcal{P}_n$. We say that $p(n,k)$ is log-concave if, for all $1 < k < n$,

$$p(n,k)^2 \geq p(n,k-1)p(n,k+1). \tag{1}$$

Trivially, we can see that $p(n,k)$ is not log-concave for any $n > 4$ because $p(n,n-1) = p(n,n) = 1$ and $p(n,n-2) \geq 2$. Thus,

$$p(n,n-1)^2 = 1 < 2 \leq p(n,n-2) = p(n,n-2)p(n,n).$$

However, we have computationally shown that, for all $51 \leq n \leq 10,000$, $p(n,k)$ is log-concave for all $1 < k < n - 25$. We conjecture that this phenomenon will continue for all $n > 10,000$, thus proving that it will be true for all $n \geq 51$. At first glance, this seems highly improbable; however, a close inspection of the bottom half of $\mathcal{P}_n$ for any $n = n_1$ reveals a bijection between $p(n,k)$ for $\lceil \frac{n_1}{2} \rceil \leq k \leq n_1$ and the partition function, $p(n)$, for $n \leq \lfloor \frac{n_1}{2} \rfloor$. Using this bijection, we can study the log-concavity of $p(n)$ to help us determine the log-concavity of $p(n,k)$. Rademacher's formula implies

$$p(n)^2 - p(n-1)p(n+1) = \frac{C_1}{n^{7/2}} e^{C_2\sqrt{n}}(1 + O(n^{-1/2})). \tag{2}$$

This equation will show that $p(n)$ is log-concave for n sufficiently large. Therefore, if we can prove that equation (2) holds and then find the sufficiently large n, we will have a much clearer picture of the log-concavity of $p(n,k)$. In near future work, we will show that equation (2) holds for all $n > 25$.

3. Bipartite Matching Property

We examine the Bipartite Matching Property of $\mathcal{P}_n$ in three ways:

(1) We use various algorithms to generate data which we will study to find patterns and matching schemes.
(2) We introduce several matching schemes and explore their utilites using computational verification and proofs.
(3) We use the Ford-Fulkerson algorithm to find maximum matchings between level sets and, in turn, determine values of n for which $\mathcal{P}_n$ has the Bipartite Matching Property.

3.1. **Matching Data and Conclusions.** We would like to prove that $\mathcal{P}_n$ has the bipartite matching property for all values of n. In order to do so, we need to look for possible patterns in the perfect matchings of the integer partition poset. We used several computational approaches to characterize the matchings between two given level sets.

The first approach precisely determines the number of perfect matchings between two consecutive level sets of equal size by finding the permanent of the adjacency matrix formed by the two level sets. First, a function creates the adjacency matrix in which the (i, j) entry equals 1 if the ith element of the first level set covers the jth entry of the second or 0 otherwise. Next, we find the permanent of this matrix. Sage has a built-in permanent function called *permanent* that uses Ryser's algorithm [6]. However, in the interest of computing efficiency, we also created a function which implements the algorithm by Mittal and Al-Kurdi [5] to more quickly determine the permanent of a sparse matrix whose entries are either 0 or 1. Next, for each edge of the Hasse diagram between the two levels, the two vertices which comprise the edge are removed from the graph and the number of perfect matchings of the resulting graph is determined as above. This number tells us how many perfect matchings between the two level sets use the edge determined by the two removed vertices. In this manner, we determine the relative frequency with which each edge appears and display the resulting data.

Another approach uses a random walk to find maximum matchings. The algorithm begins with a graph containing no edges. Alternately, certain fixed edges can also be specified *a priori*. At each step, an edge is selected at random. The edge may already be present in the graph, or it may be an edge that could be added based upon the covering relation. If that edge is not already in the graph and can be added, it is added. If the edge is already in the graph, it is removed with probability 1/B. After a specified number of maximum matchings are acquired in this manner, the algorithm returns the frequency with which each edge has appeared in the generated maximum matchings.

We also created a function which implements our random walk algorithm to search for a maximum matching between two consecutive level sets. Essentially, the function repeats the random walk process until a matching is found. By using this new algorithm for each pair of consecutive levels in a particular $\mathcal{P}_n$, we can determine that $\mathcal{P}_n$ has the Bipartite Matching Property.

Additionally, we used a function which uses the previously described random walk until it finds a specified number of every recorded maximum matching that it has found. Each time a matching is found, its corresponding counter is increased. However, a maximum matching which occurs within five steps of its previous appearance is not recorded; this allows for faster mixing of matchings. Using this function, we can approximate the number of unique matchings between two

levels by repeating the process until each matching obtained reaches a certain quota of occurences.

We used the above algorithm to successfully determine approximately how often a given edge appears in a maximum matching. These results are organized as a list of the possible edges with the percentage of how often it appears in a random walk. We may be able to use these results to show that there exists a bipartite matching for every n. One way in which we can do so is by using a greedy algorithm to find a maximum matching. First, select the edge between two given level sets with the highest probability of appearing and add it to the matching. Eliminate all other possible edges containing either of the vertices incident to the edge previously selected. Iterate until no edges remain. This creates a bipartite matching in which the smaller level set is completely saturated. We have computationally verified that this algorithm finds a maximum matching for cases for $n \leq 10$.

Our data also allows us to determine the total probability of any given vertex appearing in a maximum matching by adding up the probabilities of all the edges in which that vertex appears. The probabilities of vertices of a smaller level set will always be 1, because in order to be a bipartite matching, each vertex of the smaller level set must be matched. On the other hand, the probabilities of the vertices of the larger level set are less than 1 because not all of the vertices appear in each maximum matching, minus a few exceptions.

The data produced by these algorithms is critical in finding possible matchings schemes which will possibly produce a maximum matching.

3.2. **Matching Schemes.** By examining many Hasse Diagrams, we have determined several matching schemes that can be used to help create maximum matchings for all n.

Remark. *For even positive integers,* $2n$*, the partition of* (2^n) *always matches to* $(2^{n-2}4)$.

Remark. *For positive integers,* $3n$*, the partition of* (3^n) *always matches to* $(3^{n-2}6)$.

Remark. *In general, for positive integers,* bn*, where* $b = 1, 2, 3, 4...$*, the partition of* (b^n) *always matches to* $(b^{n-2}2b)$.

So, if $(b^{n-2}2b)$ is in the larger level set, it always has probability of 1 of being included in a maximum matching.

We looked for patterns between many levels of our Hasse Diagrams. We believe that it is feasible to find four sets of rules: one for $\Lambda_{n,1}$ to $\Lambda_{n,m}$ (where $\Lambda_{n,m}$ designates the mode), one for the levels adjacent to

the mode, one for $\Lambda_{n,m}$ to $\Lambda_{n,\lceil \frac{n}{2} \rceil}$, and one from $\Lambda_{n,\lceil \frac{n}{2} \rceil}$ to $\Lambda_{n,n}$. We will sort our results into these sections.

3.2.1. $\Lambda_{n,1}$ *to* $\Lambda_{n,m}$. The proof of the existence of a maximum matching from $\Lambda_{n,1}$ to $\Lambda_{n,2}$ is trivial. We have proven the following theorems regarding maximum matchings:

Theorem. *For all $n > 6$ there exists a maximum matching from $\Lambda_{n,2}$ to $\Lambda_{n,3}$.*

Proof. Let $\lambda \in \Lambda_{n,2}$. Then $\lambda = (x, y)$ where $x \leq y$. If n is odd, then it follows that $x < y$, and we can assign a mapping from $\Lambda_{n,2}$ to $\Lambda_{n,3}$ such that $(1, x, y-1) \lessdot (x, y)$ for all $(x, y) \in \Lambda_{n,2}$. Since each pair (x, y) will be unique, it follows that this is an injective mapping and thus will form a maximum matching.

When n is even, we must consider the exception where $x = y = \frac{n}{2}$. This case will occur only once for each n, and we use the mapping such that $\left(2, \frac{n}{2} - 2, \frac{n}{2}\right) \lessdot \left(\frac{n}{2}, \frac{n}{2}\right)$. We know that $\left(2, \frac{n}{2} - 2, \frac{n}{2}\right)$ has not previously been covered by an element of $\Lambda_{n,2}$ because each of our partitions in the level set above contains at least one 1, whereas $\left(2, \frac{n}{2} - 2, \frac{n}{2}\right)$ contains none when $n > 6$. Thus we create a maximum matching between the two level sets.

□

Theorem. *For all $n \geq 4$ there exists a maximum matching from $\Lambda_{n,3}$ to $\Lambda_{n,4}$.*

Proof. It has been proven computationally that $\mathcal{P}_n$ has the bipartite matching property for all $n \leq 45$. The following scheme proves that there is maximum matching for $n > 9$.

Let $\lambda \in \Lambda_{n,3}$ such that $\lambda = (x, y, z)$ where $x \leq y \leq z$.

Let $\phi : \Lambda_{n,3} \to \Lambda_{n,4}$ be a function defined as follows:

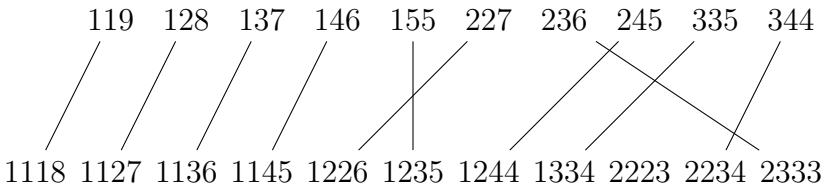
(1) For odd n, $\phi\left(\left(2, \frac{n-5}{2}, \frac{n+1}{2}\right)\right) = \left(2, 3, \frac{n-5}{2}, \frac{n-5}{2}\right)$
(2) If $y < z$, then $\phi\left((x, y, z)\right) = (1, x, y, z-1)$, unless it has already been matched under (1)
(3) If $y = z$, then $\phi\left((x, y, y)\right) = (2, x, y-2, y)$,

where $\phi(\lambda) = \mu$ implies $\mu \lessdot \lambda$. It is clear that (2) and (3) will form an injective mapping. Note that we must take extra care in (1). Let $\lambda' = \phi\left(\left(2, \frac{n-5}{2}, \frac{n+1}{2}\right)\right) = \left(2, 3, \frac{n-5}{2}, \frac{n-5}{2}\right)$. Since $n > 9$, it follows that $\frac{n+1}{2} > 4$, thus $\frac{n+1}{2} - 3 > 1$. Therefore λ' does not contain any 1's, thus will not form a matching of type (2). If λ' came from our third class of matchings, then it would have come from $\left(3, \frac{n-5}{2}, \frac{n-5}{2} + 2\right)$, but clearly $\frac{n-5}{2} \neq \frac{n-5}{2} + 2$ and so it would not be matched under (3). Therefore it is guaranteed that $\left(2, \frac{n-5}{2}, \frac{n+1}{2}\right)$ will map to a unique vertex in $\Lambda_{n,4}$.

Thus ϕ is an injective map and we can form a maximum matching from $\Lambda_{n,3}$ to $\Lambda_{n,4}$.

□

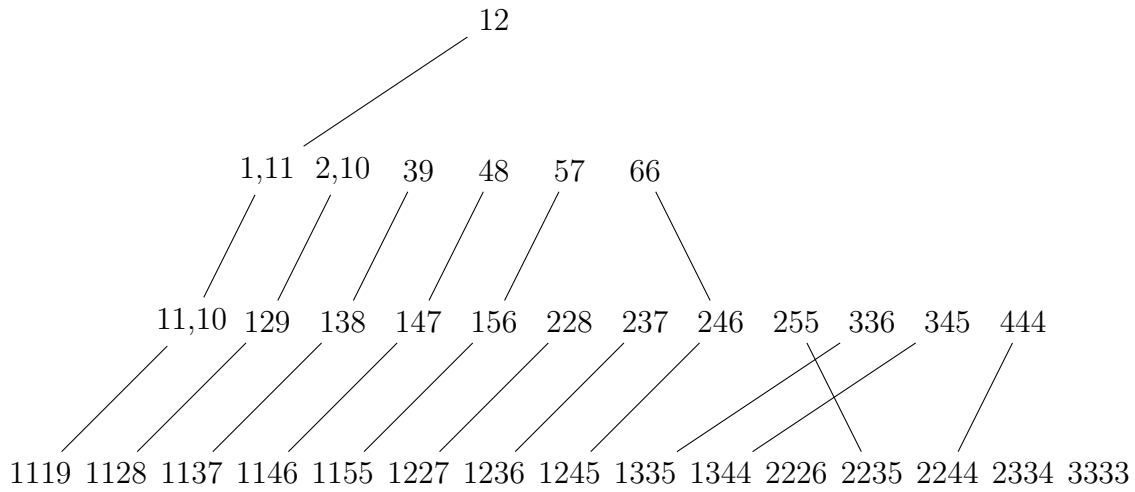
Example. *A mapping from* $\Lambda_{11,3}$ *to* $\Lambda_{11,4}$.



Using schemes similiar to the ones listed in the proofs above, we hope that we may be able to categorize the way that mappings are formed from Λ_k to Λ_{k+1} where $k, k+1 < m$ and hopefully we will be able to write a proof based on generalizing the pattern of exceptions as we look for maximum matchings between level sets of increasingly larger ranks.

We are also pursuing an alternate idea for finding maximum matchings from $\Lambda_{n,1}$ to the mode. Let $\Lambda_{n,m}$ designate the level set at the mode. Then we have produced a matching scheme which works from $\Lambda_{n,1}$ to $\Lambda_{n,m-1}$ up to $n = 32$; we will call this scheme *Leftmost to the Mode (LTM)*. To produce such a matching, look at two consecutive levels, Λ_k and Λ_{k+1}, with lexicographic ordering, where $1 \leq k, k+1 < m$. First, take $\lambda \in \Lambda_{n,k}$ and check if it covers the leftmost element of $\Lambda_{n,k+1}$, call this μ. If this is so, we draw an edge between λ and μ. If not, look at the next element in $\Lambda_{n,k+1}$ and apply the same process. We continue this process until λ has been matched with some $\mu \in \Lambda_{n,k+1}$. Then we repeat the process with the next element in $\Lambda_{n,k}$, call this λ'. We repeat this process until we have matched all of the elements in $\Lambda_{n,k}$ with elements of $\Lambda_{n,k+1}$. We have found that LTM fails at $n = 35$, but this scheme may still be useful for improving algorithm efficiency.

Example. *A matching produced by this scheme for* $\mathcal{P}_{12}$*. Note that this is an exceptional case because our matching works to the mode at* $k = 4$*.*



3.2.2. *Levels adjacent to the mode.* As mentioned above, our LTM scheme oftentimes works down to the mode. In cases where this rule fails for $\Lambda_{n,m-1}$ to $\Lambda_{n,m}$ we have observed that $p(n, m-1) \approx p(n, m)$.

3.2.3. *From $\Lambda_{n,m}$ to $\Lambda_{n,\lceil \frac{n}{2} \rceil}$:* We tried applying several rules for this section of the graph, but all had multiple exceptions which we expect to become increasingly more numerous and complex as the size of n increases. If we again consider the partition $\lambda = \left(j\bar{\lambda}\right)$ where j is the largest summand, then the rule such that λ maps to $\lambda' = \left(1(j-1)\bar{\lambda}\right)$ will produce and non-crossing matching across at least half of each level. In this instance, our rule breaks down when our partition λ contains no ones, but instead ends in repeated 2's or 3's. We suspect that this trend will continue for partitions than end in repeated summands i such that $i \geq 2$. In this instance, we were unable to find any rule for exceptions that worked uniformly, but did consider the following three ideas:

1. Whenever a vertex has a double matching, redirect the first matching by adding the smallest two summands together.
2. Break the second-largest summand, i, into $i-2$ and 2.
3. Match the vertex to any other vertex that will work.

The third rule worked in all examples we studied but is not very helpful in terms of a proof. We have explored some other ideas for creating maximum matchings between two consecutive level sets more easily. Denote the partitions of two consecutive level sets as $\Lambda_{n,k} = u_1, u_2, ..., u_{i_1}$ and $\Lambda_{n,k-1} = t_1, t_2, ..., t_{i_2}$, where $i_1 = p(n,k)$ and $i_2 = p(n,k-1)$. For $9 \leq n \leq 13$, the first four vertices in two adjacent

level sets with $p(n,k) \geq 4$, match consecutively with our lexographical ordering. In other words, u_i matches to t_i for $i \leq 4$. Also, the last vertex in the larger level set can be removed and a maximum matching still exists.

Similarly, for any two adjacent level sets in $\mathcal{P}_{12}$ and $\mathcal{P}_{13}$, no matter their rank, if the leftmost vertices of both level sets can be, and are consecutively matched, i.e. u_i is matched to t_i, then there still exists a maximum matching. In $\mathcal{P}_{13}$ the last vertex of the larger level set can be left off as well and the above statement still holds. These matching properties can make our matching algorithms, such as Ford-Fulkerson, more efficient because these edges and vertices can be disregarded.

In order to make proving the bipartite matching property more feasible, we hope to find a way to consecutively match all vertices within any two adjacent level sets. In order to do so, we must find another way to order the vertices in the level sets. Lexographically, producing a non-crossing matching is not possible. Ordering the vertices in order of descending probability of appearing, and ordering the partitions by their increasing number of summands have failed to yield a non-crossing matching as well.

3.2.4. *$\Lambda_{n,\lfloor \frac{n}{2} \rfloor}$ to $\Lambda_{n,n}$.*

Lemma 3.1. *Suppose n is a non-negative integer and $1 \leq k \leq n$. If $\lambda \in \Lambda_{n,k}$, then λ has at least max$(\{(2k-n), 0\})$ summands equal to 1.*

Proof. Suppose, for contradiction, $\lambda \in \Lambda_{n,k}$ has $(2k-n-\delta)$ summands equal to 1 (where $\delta \in \mathbb{N}$). Then it must have $k-(2k-n-\delta) = n-k+\delta$ summands which are greater than or equal to 2. Thus, the sum of these $(n-k+\delta)$ summands is at least $2(n-k+\delta) = 2n-2k+2\delta$. Therefore the total of all of the summands is at least $(2n-2k+2\delta)+(2k-n-\delta) = (n+\delta)$. This is a contradiction, since $\delta > 0$ and λ is a partition of n. Thus, it must be the case that λ has at least $(2k-n)$ summands equal to 1. □

Theorem. *For any positive integer n, for all integers $\lceil \frac{n}{2} \rceil \leq k < n$, there exists a maximum matching from $\Lambda_{n,k+1}$ to $\Lambda_{n,k}$.*

Proof. Suppose $\lambda \in \Lambda_{n,k}$where $\lceil \frac{n}{2} \rceil + 1 \leq k \leq n$. By our previous lemma, we can conclude that λ contains at least $2k-n \geq 2(\lceil \frac{n}{2} \rceil + 1) - n \geq 2$ summands equal to 1. So, we can write λ as $1^2\bar{\lambda}$, where $\bar{\lambda}$ represents the $(k-2)$ other parts of the partition.

We define $\phi : \Lambda_{n,k+1} \to \Lambda_{n,k}$ by $\phi(1^2\bar{\lambda}) = 2\bar{\lambda})$.

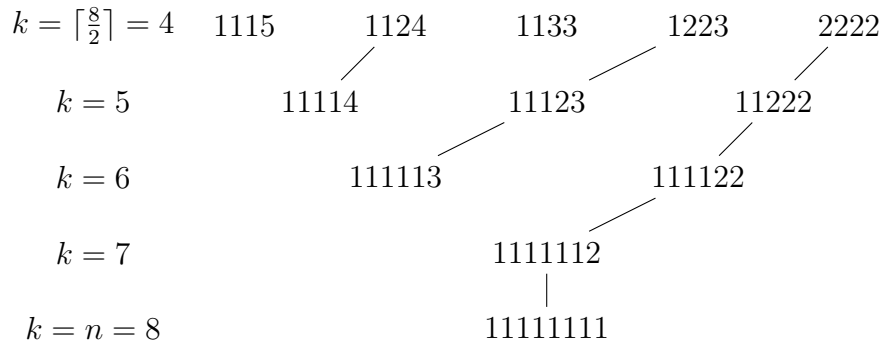
Now, suppose there exists another partition $\lambda' \in \Lambda_{n,k}$ such that $\phi(\lambda) = \phi(\lambda')$. We know that λ and λ' can be expressed as $1^2\bar{\lambda}$ and

$1^2\bar{\lambda}'$, respectively. Thus,

$$\phi(1^2\bar{\lambda}) = \phi(1^2\bar{\lambda}')$$
$$2, \bar{\lambda} = 2, \bar{\lambda}'$$
$$\bar{\lambda} = \bar{\lambda}'$$
$$\lambda = 1^2\bar{\lambda} = 1^2\bar{\lambda}' = \lambda'.$$

Therefore, ϕ is 1-1. Therefore, using ϕ to choose our edges, we have $\lambda \lessdot \phi(\lambda)$, and we create a maximum matching between $\Lambda_{n,k+1}$ and $\Lambda_{n,k}$. □

Example. *The partial Hasse diagram below shows our matching scheme ϕ applied to the bottom half of $\mathcal{P}_8$.*



Theorem. *For all positive odd integers n, for $k = \lceil \frac{n}{2} \rceil$, there exists a maximum matching between the k^{th} and the $(k-1)^{th}$, or $\lfloor \frac{n}{2} \rfloor$, level.*

Proof. Suppose $\lambda \in \Lambda_{n,\lceil \frac{n}{2} \rceil}$. By our previous lemma, each partition of n into k parts has at least $(2k-n)$ summands equal to 1. For odd n, when $k = \lceil \frac{n}{2} \rceil = \frac{n+1}{2}$, we have at least

$$2\left(\frac{n+1}{2}\right) - n = n + 1 - n = 1$$

summands equal to 1. So λ has at least one summand equal to 1.

Let $\lambda \in \Lambda_{n,\lceil \frac{n}{2} \rceil}$ be denoted as $(1\bar{\lambda}l_\lambda)$, where $\bar{\lambda}$ is the $(\lceil \frac{n}{2} \rceil - 2)$ other summands of the partition, and l_λ is the largest summand of that partition.

Define $\phi : \lambda_k \to \lambda_{k-1}$ by $\phi((1\bar{\lambda}l)) = (\bar{\lambda}(1+l))$.

Suppose there exists a partition, $\mu \in \Lambda_{n,\lceil \frac{n}{2} \rceil}$, such that $\phi(\lambda) = \phi(\mu)$.

Then

$$\phi((1\bar{\lambda}l_\lambda)) \;=\; \phi((1\bar{\mu}l_\mu)).$$

That is,

$$(\bar{\lambda}(1+l_\lambda)) \;=\; (\bar{\mu}(1+l_\mu)).$$

Since l_λ is our largest summand in λ, then $1+l_\lambda$ is our largest summand in $\phi(\lambda)$.

So

$$l_\lambda = l_\mu.$$

Thus

$$(\bar{\lambda}(1+l_\lambda)) = (\bar{\mu}(1+l_\lambda)),$$

so that

$$\bar{\lambda} = \bar{\mu}.$$

Thus,

$$\lambda = \mu.$$

Hence, ϕ is injective. Since $\lambda \lessdot \phi(\lambda)$, we have a maximum matching. □

3.3. **Ford-Fulkerson Algorithm.** As previously stated, Rodney Canfield utilized the Ford Fulkerson Algorithm to prove that $\mathcal{P}_n$ has the bipartite matching property for $n \leq 45$. We implement this algorithm as well in hopes of producing more maximum matchings. The Ford Fulkerson algorithm, in general form, is a systematic way to determine the maximum flow through a graph in which each edge has a specified capacity. We used a specialized form of this algorithm. The algorithm generates a maximum matching or verifies that no such matching exists.

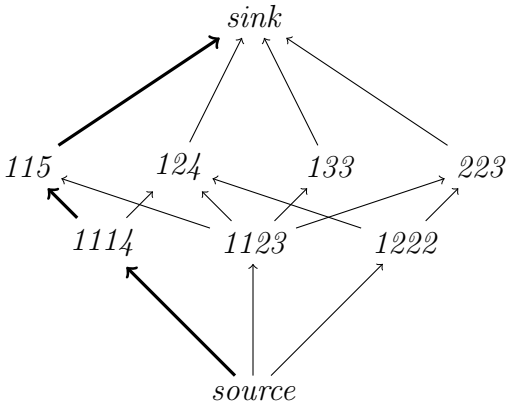
Our algorithm first sets up a graph. This graph consists of two adjacent level sets. Each partition becomes a vertex in the graph, and covering relations correspond to the edges. Additionally, we create a source vertex and a sink vertex. The source vertex is connected by edges to each vertex in one level set, and the sink vertex is connected in a similar fashion to all the vertices in the other level set. All of the edges in this graph are assigned an initial capacity of one. That is, these are forward edges that can transmit flow in a forword direction. These edges may later be converted to backwards edges, which can transmit flow in the opposite direction.

The algorithm then performs a search to find a path through the graph from the source vertex to the sink vertex. In searching for this path, the algorithm can push flow forword through forword edges and backwards through backwards edges. Our algorithm uses a depth-first search to find a path, but the Ford-Fulkerson algorithm does not specify which type of search should be used, and other types of searches could be employed. After a path is found, the algorithm pushes flow

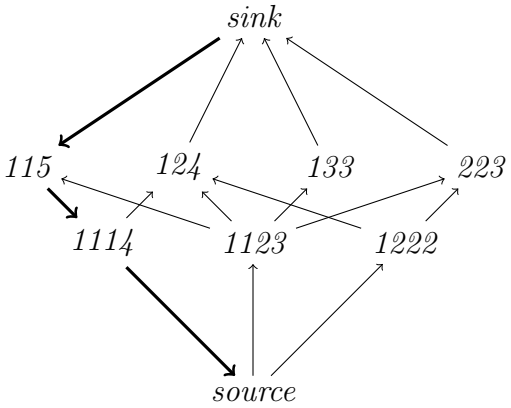
along this path, modifying the graph. All edges which are traveled in a forward direction have their capacities reduced to zero. This converts the edges to backwards edges. All edges which are traveled in a backwards direction have their capacities increased to one, converting them to forword edges. After the graph has been appropriately modified, the algorithm searches for another path and repeats the above steps.

Once no more complete paths can be found, the algorithm computes the total flow through the graph by counting the number of backwards edges that are adjacent to the source node. This number is then compared to the number of vertices in the smaller level set. If these two numbers are equal, there is a maximum matching. Otherwise, no such matching exists. With rational flow values, the algorithm is guaranteed to terminate.

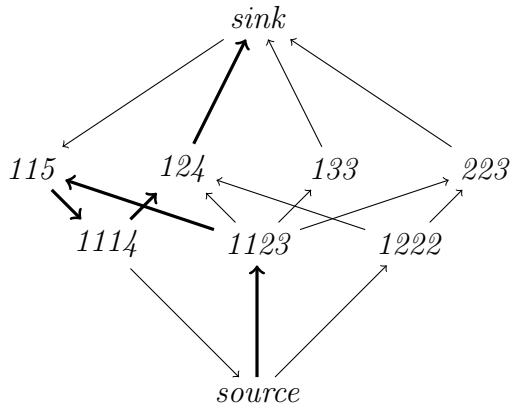
Example. *To create a maximum matching between* $\Lambda_{7,3}$ *and* $\Lambda_{7,4}$*, the algorithm would first find the bold path.*



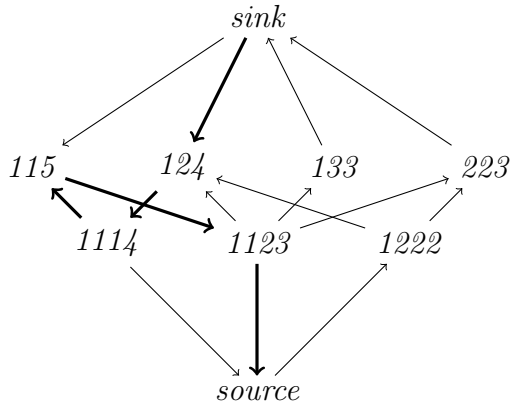
The graph would then be altered by flipping the arrows (altering the capacities) along the path to give:



Next, this path would be found:



The graph would be altered to form:

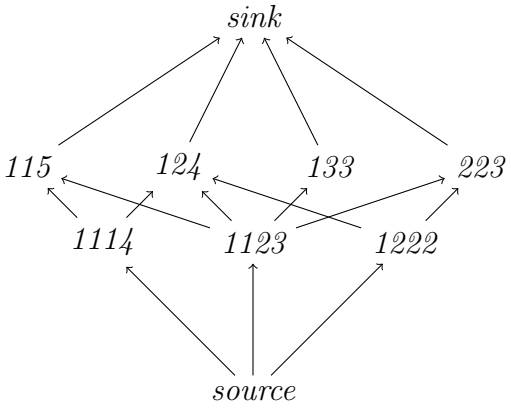


The process is continued until no more paths can be found.

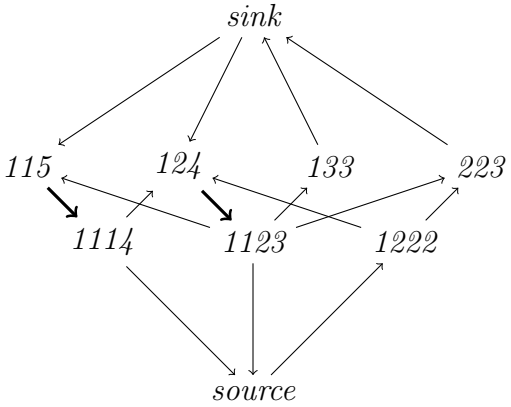
This algorithm must be run on every pair of adjacent level sets in a given poset to verify the bipartite matching property. In order to reduce the running time, we have parallelized the algorithm, running each pair of level sets on a different computer.

In addition to running the algorithm as described above, we have tried seeding Ford Fulkerson with a partial matching. Instead of starting with a graph in which all of the edges have capacity one, we set the capacity of several of these edges to zero. Specifically, we matched a partition of smaller rank to a partition of larger rank which was identical except that the largest summand had been split into one and itself minus one. For most pairs of level sets, it is not possible to connect all partitions in this manner. The Ford-Fulkerson algorithm can fill in the remaining edges.

Example. *The Ford Fulkerson algorithm without partial matchings would attempt to create a maximum matching between* $\Lambda_{7,3}$ *and* $\Lambda_{7,4}$ *beginning with the graph:*



The algorithm which uses partial matchings, however, would begin with the follow graph. The bold edges are those whose directions have been reversed. They create a partial matching.



Due to the long running time of these algorithms when applied to large level sets, we have not yet been able to verify the bipartite matching property for any values of n beyond those that Canfield verified.

4. Conclusion

We have found that $\mathcal{P}_n$ is unimodal for all $n \leq 25,000$. We have also explored potential values for the "sufficiently large n" for which Szekeres' proof of unimodality holds.

We have also proven several matching schemes which apply for all n. These steps represent fairly substantial progress toward proving the Sperner property for $\mathcal{P}_n$ for all n. However, several obvious avenues for improvement still remain open:

If the bounds suggested by our data for Szekeres' proof can be rigorously proven, then the question of unimodality will be proven for all n (assuming the computationally-proven values of n surpass the "sufficiently large n" found from the paper).

If $\mathcal{P}_n$'s unimodality must be proven for larger values of n than 25,000, an algorithm similar to the one which generates the $p(n,k)$ matrix could be made with the following modifications: the new algorithm would store the values in overlapping two-dimensional arrays by column. In this way, no individual file will become too large. Each of these values can be stored modulo several medium-sized primes to further reduce the necessary capacity of each file. In order to implement this algorithm, one should first calculate the left-most columns for a specified range of k values and then pass the last calculated column to the function that calculates the next set of columns. Proceed in this manner until no columns remain for the specified range of k values. At this point, repeat the entire process with an incrementally larger range of k values until all user-specified $p(n,k)$ values are calculated.

The Ford-Fulkerson algorithm can be modified so that it starts with a partial matching. In order to do this, a breadth-first search should be implemented instead of a depth-first, search. Additionally, by seeding the Ford-Fulkerson algorithm with "good" edges, we believe that the runtime of the algorithm can be significantly reduced.

References

[1] Ivona Bezáková, Daniel Stefankovic, Vijay V. Vazirani, and Eric Vigoda. Accelerating simulated annealing for the permanent and combinatorial counting problems. In *In Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA*, pages 900–907. ACM Press, 2006.

[2] E. Rodney Canfield. Integer partitions and the sperner property. *Theoretical Computer Science*, 307(3):515–529, 2003.

[3] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4), 2004.

[4] L. Lovász and M. D. Plummer. *Matching theory*, volume 121 of *North-Holland Mathematics Studies*. North-Holland Publishing Co., Amsterdam, 1986. Annals of Discrete Mathematics, 29.

[5] R. C. Mittal and Ahmad Al-Kurdi. Efficient computation of the permanent of a sparse matrix. *Intern. J. Computer Math.*, 77:189–199, 2001.

[6] H.J. Ryser. *Combinatorial Mathematics*. Mathematical Association of America, 1963.

[7] G. Szekeres. Some asymptotic formulae in the theory of partitions. *Quarterly Journal of Mathematics*, 2(2):85–108, 1951.

[8] G. Szekeres. Some asymptotic formulae in the theory of partitions(ii). *Quarterly Journal of Mathematics*, 2(4):96–111, 1953.