5-2012

A Collection of Problems in Combinatorics

Janine Janoski
Clemson University, janinejanoski@gmail.com

A Collection of Problems in Combinatorics

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mathematics

---

by
Janine E. Janoski
May 2012

Accepted by:
Dr. Neil Calkin, Committee Chair
Dr. Kevin James
Dr. Gretchen Matthews
Dr. Colin Gallagher

Abstract

We present several problems in combinatorics including the partition function, Graph Nim, and the evolution of strings.

Let $p(n)$ be the number of partitions of n. We say a sequence a_n is log-concave if for every n, $a_n^2 \geq a_{n+1}a_{n-1}$. We will show that $p(n)$ is log-concave for $n \geq 26$. We will also show that for $n < 26$, $p(n)$ alternatively satisfies and does not satisfy the log-concave property. We include results for the Sperner property of the partition function.

The second problem we present is the game of Graph Nim. We use the Sprague-Grundy theorem to analyze modified versions of Nim played on various graphs. We include progress made towards proving that all G-paths are periodic.

The third topic we present is on the evolution of strings. Consider a string of length l over an alphabet of size k. At each stage of the evolution, with a probability of q, we randomly select a new letter to replace a correct letter. Using the transition matrix we will study the absorption rate to the correct string.

Acknowledgments

I owe my deepest gratitude to my committee chair Neil Calkin, for his advice, guidance, and wisdom on this endeavor. I would like to thank the members of my committee, Colin Gallagher, Gretchen Matthews, and Kevin James. A special thanks to thank Sarah Leggett, Bryce Richards, Nathan Sitaraman, and Stephanie Thomas for their contribution to the Graph Nim problem and to Brian Bowers, Kerry Gannon, Katie Jones, and Anna Kirkpatrick for their contribution to the Sperner property for the partition function. I would like to express my love and gratitude to my family and friends; for their understanding and support throughout this process.

Table of Contents

List of Figures

Chapter 1

The Partition Function

1.1 Introduction

We define a partition of a positive integer n to be a sequence of non-increasing positive integers that sum to n. Partitions occur in many fields of math such as Rogers-Ramanujan identities, symmetric polynomials and group representation theory. The topic has been studied by many mathematicians such as Euler, Legendre, Ramanujan, Hardy, Rademacher, Selber, and Andrews, to name just a few. We will focus our research on properties concerning the number of partitions for a given n.

Definition. *Let $p(n)$ be the number of partitions of a nonnegative integer n.*

For example, $p(4) = 5$, since

$$\begin{aligned} 4 &= 4 \\ &= 3+1 = 2+2 \\ &= 2+1+1 = 1+1+1+1. \end{aligned}$$

Definition. *A sequence a_n is log-concave if for every n*

$$a_n^2 \geq a_{n+1}a_{n-1}.$$

In 1918, Hardy and Ramanujan proved that

$$p(n) \sim \frac{1}{4n\sqrt{3}} e^{\pi\sqrt{2n/3}}, \text{ as } n \to \infty.$$

In 1937, Rademacher improved this result by expressing $p(n)$ as a convergent series. In Section 1.2 we will show that $p(n)$ is log-concave for $n \geq 26$ [5]. We will also discuss the alternating behavior for $p(n)$ satisfying log-concavity and not satisfying this property for $n < 26$. To prove log-concavity, we will use Rademacher's formula for the partition function.

In Section 1.3 and 1.4 we will discuss the Sperner property for partitions. This work was joint work with Neil Calkin while advising Brian Bowers, Kerry Gannon, Katie Jones, and Anna Kirkpatrick begun during the 2010 Clemson REU.

Definition. *Let n be a non-negative 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

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

For example the partitions of 4 are

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

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

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

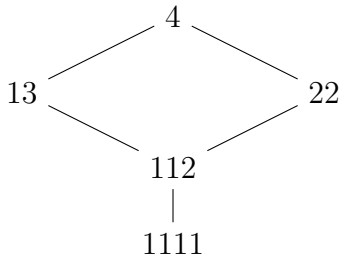
We can organize $\mathcal{P}_n$ using a graphical representation called a Hasse diagram. We create the graph with vertices

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

arranged in rows, called "level sets", according to rank. The edge set is defined by

$$E = \{\{\lambda, \lambda'\} : \lambda, \lambda' \in \mathcal{P}_n, \lambda \text{ and } \lambda' \text{ are comparable}\}.$$

Example. *The Hasse diagram for $\mathcal{P}_4$:*



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 decomposition 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.*

Since elements within a level set are incomparable, every level set is an antichain. For $\mathcal{P}_4$ the only antichain is $\{(1,3),(2,2)\}$.

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.*

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

Example. *In $\mathcal{P}_4$, since the size of the largest antichain is two and the size of the largest level set is two, we can conclude that $\mathcal{P}_4$ is Sperner.*

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

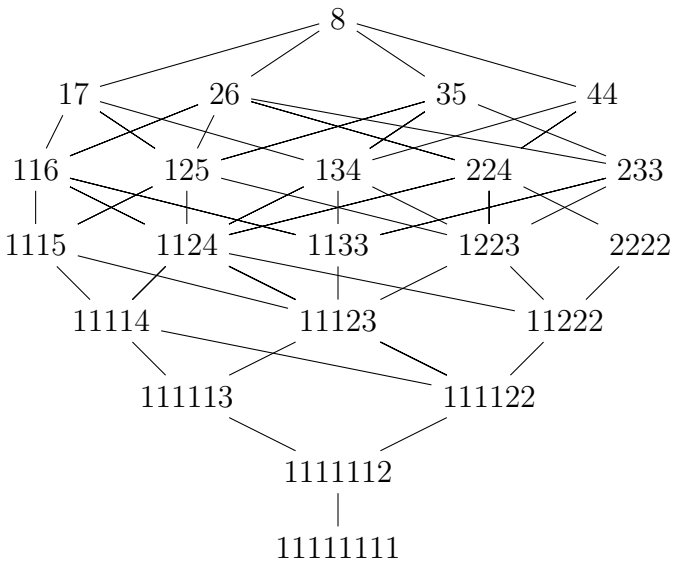
Definition. *We say that $\mathcal{P}_n$ is unimodal if there exists a number $k_{1,n}$, called the mode, such that $p(n,k) \leq p(n,k+1)$ for $k < k_{1,n}$, and $p(n,k) \geq p(n,k+1)$ for $k > k_{1,n}$.*

Definition. *We say that $\mathcal{P}_n$ has 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.

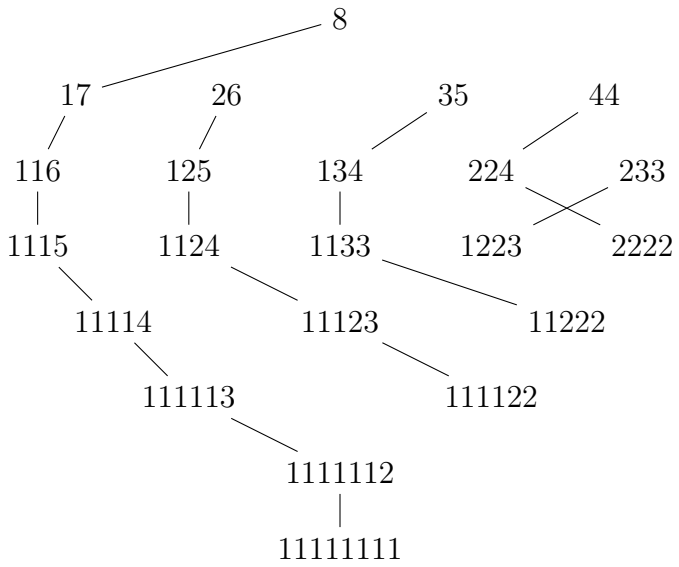
Example. *Below we show that $\mathcal{P}_8$ is Sperner.*

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



We see that the largest anti-chain has size 5 which is the size of the largest level set. Alternatively we can use the unimodal and bipartite matching properties. It is easy to see that $\mathcal{P}_n$ is unimodal, since the first level set is size one and then the level sets increase to a level set of size five and then decrease to a level set of size

one. Below we show that the bipartite matching property can be satisfied for each consecutive level set.



In 1952, Szekeres proved that, for "sufficiently large n," $\mathcal{P}_n$ is unimodal [18]. 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 $n \leq 45$ [7]. 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 [7].

In Section 1.3, we will discuss computational approaches we have used to push the bounds of the unimodality of $\mathcal{P}_n$. In Section 1.4, we will study the Bipartite Matching Property of $\mathcal{P}_n$.

1.2 Log-Concavity of $p(n)$

We first began to study the property of log-concavity for $p(n,k)$ and discovered that things did not appear to be log-concave. In fact, it appeared that $p(n,k)$ satisfied the log-concave inequality and then alternately satisfied and did not. It appeared that $p(n,k)$ is log-concave except when $n-k$ is odd and at most 25.

The second half of the sequence $p(n,k)$ is given by $p(n-k)$, that is, the reverse of the first $\frac{n}{2}$ terms of the partition function $p(n)$. We then looked closely at the log-concavity of $p(n)$. We have proven the following theorem.

Theorem. *For $n \geq 26$, $p(n)^2 \geq p(n+1)p(n-1)$.*

By Rademacher's formula we have

$$\begin{aligned} p(n) &= \frac{1}{\pi\sqrt{2}} \sum_{k=1}^{\infty} A_k(n)\sqrt{k}\frac{d}{dn}\left(\frac{\sinh\left(\frac{\pi}{k}\sqrt{\frac{2}{3}(n-\frac{1}{24})}\right)}{\sqrt{n-\frac{1}{24}}}\right) \\ &= \frac{1}{\pi\sqrt{2}} \sum_{k=1}^{\infty} A_k(n)\sqrt{k}d(n,k). \end{aligned}$$

The function $A_k(n)$ is periodic in n with period k and is given by

$$A_k(n) = \sum_{0\leq m<k;\gcd(m,k)=1} e^{\pi i(s(m,k)-2nm/k)},$$

where $s(m,k)$ is a Dedekind sum. That is,

$$s(m,k) = \sum_{n \mod k} \left(\left(\frac{n}{k}\right)\right)\left(\left(\frac{mn}{k}\right)\right),$$

where

$$((x)) = \begin{cases} x - \lfloor x \rfloor - \frac{1}{2} & x \in \mathbb{Q} \\ 0 & x \in \mathbb{Z} \end{cases}.$$

Letting $t = n - \frac{1}{24}$ and $c = \pi\sqrt{\frac{2}{3}}$, we can write

$$\begin{aligned} d(n,k) &= \frac{d}{dt}\left(\frac{\sinh\left(\frac{c\sqrt{t}}{k}\right)}{\sqrt{t}}\right) = \frac{c\sqrt{t}}{2kt^{3/2}}\cosh\left(\frac{c\sqrt{t}}{k}\right) - \frac{1}{2t^{3/2}}\sinh\left(\frac{c\sqrt{t}}{k}\right) \\ &= \frac{1}{2t^{3/2}}\left(\frac{c\sqrt{t}}{k}\cosh\left(\frac{c\sqrt{t}}{k}\right) - \sinh\left(\frac{c\sqrt{t}}{k}\right)\right) \\ &= \frac{1}{4t^{3/2}}\left(\frac{c\sqrt{t}}{k} - 1\right)\exp\left(\frac{c\sqrt{t}}{k}\right)\left(1 + \frac{c\sqrt{t}+k}{c\sqrt{t}-k}\exp\left(\frac{-2c\sqrt{t}}{k}\right)\right). \end{aligned}$$

Note that, we will use the notation t and n interchangeably where appropriate.

We have that $A_1(n) = 1$. Thus the first term in Rademacher's formula is

$$\frac{1}{\pi\sqrt{2}}d(n,1) = \frac{1}{4\pi\sqrt{2}t^{3/2}}(c\sqrt{t}-1)\exp(c\sqrt{t})\left(1 + \frac{c\sqrt{t}+1}{c\sqrt{t}-1}e^{-2c\sqrt{t}}\right).$$

We have that $A_2(n) = (-1)^n$. Thus the second term in Rademacher's formula is

$$\frac{(-1)^n}{\pi}d(n,2) = \frac{(-1)^n}{4\pi t^{3/2}}\left(\frac{c\sqrt{t}}{2} - 1\right)\exp\left(\frac{c\sqrt{t}}{2}\right)\left(1 + \frac{c\sqrt{t}+2}{c\sqrt{t}-2}\exp\left(-c\sqrt{t}\right)\right).$$

We have that

$$A_3(n) = \exp\left(\pi i\left(\frac{1}{18} - \frac{2n}{3}\right)\right) + \exp\left(\pi i\left(-\frac{1}{18} - \frac{4n}{3}\right)\right).$$

Thus the third term in Rademacher's formula is

$$\frac{1}{4\pi\sqrt{6}t^{3/2}}\left(\exp\left(\pi i\left(\frac{1}{18}-\frac{2n}{3}\right)\right)+\exp\left(\pi i\left(-\frac{1}{18}-\frac{4n}{3}\right)\right)\right)\times$$

$$\left(c\sqrt(t)-3\right)\exp\left(\frac{c\sqrt{t}}{3}\right)\left(1+\frac{c\sqrt{t}+3}{c\sqrt{t}-3}\exp\left(-\frac{2c\sqrt{t}}{3}\right)\right).$$

We want to consider

$$\frac{1}{\pi\sqrt{2}}\sum_{k=3}^{c\sqrt{t}}A_k(n)\sqrt{k}d(n,k) > c\sqrt{t}\frac{\sqrt{3}}{\pi\sqrt{2}}A_3(n)d(n,3)$$

$$=\frac{c\sqrt{t}}{4\pi\sqrt{6}t^{3/2}}\left(\exp\left(\pi i\left(\frac{1}{18}-\frac{2n}{3}\right)\right)+\exp\left(\pi i\left(-\frac{1}{18}-\frac{4n}{3}\right)\right)\right)\times$$

$$\left(c\sqrt(t)-3\right)\exp\left(\frac{c\sqrt{t}}{3}\right)\left(1+\frac{c\sqrt{t}+3}{c\sqrt{t}-3}\exp\left(-\frac{2c\sqrt{t}}{3}\right)\right).$$

For k large, let $\theta=\frac{c\sqrt{t}}{k}$ and we have

$$\frac{1}{2t^{3/2}}(\theta\cosh(\theta)-\sinh(\theta))=\frac{1}{2t^{3/2}}\sum_{j=1}^{\infty}\theta^{2j+1}\left(\frac{1}{(2j)!}-\frac{1}{(2j+1)!}\right)$$

$$=\frac{1}{2t^{3/2}}\sum_{j=1}^{\infty}\theta^{2j+1}\left(\frac{1}{(2j+1)(2j-1)!}\right)=\frac{1}{2t^{3/2}}\frac{\theta^3}{3}(1+o(1))$$

when θ is $o(1)$ as $k\to\infty$.

So

$$\left|\frac{1}{\sqrt{2}\pi}\sum_{k=T}^{\infty}A_k(t)\sqrt{k}d(t,k)\right|=\frac{1}{\sqrt{2}\pi}\sum_{k=T}^{\infty}\left|A_k(t)\sqrt{k}\frac{c^3}{2k^3}\right|(1+o(1))$$

$$=\frac{\pi^2}{3\sqrt{3}}\sum_{k=T}^{\infty}\left|A_k(t)\sqrt{k}\frac{1}{k^3}\right|(1+o(1))<\frac{2\pi^2}{3\sqrt{3}\sqrt{T}}(1+o(1)).$$

Thus

$$\sum_{k=T}^{\infty}\left|\sqrt{k}\frac{1}{k^2}\right| < \frac{2}{\sqrt{T}}.$$

If we increase the constant slightly, we can let $T = 1$ and get a contribution of $O(1)$ independent of n.

Thus, using the first three terms of Rademacher's formula, we have the following estimate for $p(n)$:

$$\begin{aligned}
p(n) &= \left(\frac{1}{4\pi\sqrt{2}t^{3/2}}(c\sqrt{t}-1)\exp(c\sqrt{t})\left(1+\frac{c\sqrt{t}+1}{c\sqrt{t}-1}\exp\left(-2c\sqrt{t}\right)\right)\right)\times \\
&\quad \left(\frac{(-1)^n}{4\pi t^{3/2}}\left(\frac{c\sqrt{t}}{2}-1\right)\exp\left(\frac{c\sqrt{t}}{2}\right)\left(1+\frac{c\sqrt{t}+2}{c\sqrt{t}-2}\exp\left(-c\sqrt{t}\right)\right)\right)\times \\
&\quad \left(\frac{c\sqrt{t}\left(c\sqrt{t}-3\right)}{4\pi\sqrt{6}t^{3/2}}\left(\exp\left(\pi i\left(\frac{1}{18}-\frac{2n}{3}\right)\right)+\exp\left(\pi i\left(-\frac{1}{18}-\frac{4n}{3}\right)\right)\right)\times \right. \\
&\quad \left.\exp\left(\frac{c\sqrt{t}}{3}\right)\left(1+\frac{c\sqrt{t}+3}{c\sqrt{t}-3}\exp\left(-\frac{2c\sqrt{t}}{3}\right)\right)\right) \\
&> \frac{1}{4\pi\sqrt{2}t^{3/2}}(c\sqrt{t}-1)\exp(c\sqrt{t})\left(1+\frac{(-1)^n(c\sqrt{t}-2)}{\sqrt{2}(c\sqrt{t}-1)}\exp\left(-\frac{c\sqrt{t}}{2}\right)\right)\times \\
&\quad \left(1+\frac{c\sqrt{t}}{\sqrt{3}}\left(\frac{c\sqrt{t}-3}{c\sqrt{t}-1}\right)\exp\left(-\frac{2c\sqrt{t}}{3}\right)\times \right. \\
&\quad \left.\left(\exp\left(\pi i\left(\frac{1}{18}-\frac{2n}{3}\right)\right)+\exp\left(\pi i\left(-\frac{1}{18}-\frac{4n}{3}\right)\right)\right)\right).
\end{aligned}$$

If we consider just the real part of

$$\left(\exp\left(\pi i\left(\frac{1}{18}-\frac{2n}{3}\right)\right)+\exp\left(\pi i\left(-\frac{1}{18}-\frac{4n}{3}\right)\right)\right),$$

then we have

$$p(n) > \frac{1}{4\pi\sqrt{2}t^{3/2}}(c\sqrt{t}-1)\exp(c\sqrt{t})\left(1+\frac{(-1)^n\sqrt{2}(c\sqrt{t}-2)}{(c\sqrt{t}-1)}\exp\left(-\frac{c\sqrt{t}}{2}\right)\right)\times$$

$$\left(1-\frac{2c\sqrt{t}}{\sqrt{3}}\left(\frac{c\sqrt{t}-3}{c\sqrt{t}-1}\right)\exp\left(-\frac{2c\sqrt{t}}{3}\right)\right).$$

We want to consider

$$\frac{p(n)^2}{p(n+1)p(n-1)}.$$

We will first analyze

$$\frac{p_1(n)^2}{p_1(n-1)p_1(n+1)},\ \text{where } p_1(n)=\frac{1}{4\pi\sqrt{2}t^{3/2}}(c\sqrt{t}-1)\exp(c\sqrt{t}).$$

We will first consider the ratio of the exponential parts. We have

$$2c\sqrt{t}-c\sqrt{t+1}-c\sqrt{t-1} \;=\; \frac{c}{4t^{\frac{3}{2}}}+\frac{5c}{64t^{\frac{7}{2}}}+\frac{21c}{512t^{\frac{11}{2}}}-2\sum_{i=1}^{\infty}\binom{\frac{1}{2}}{2i+6}t^{\frac{-(4i+11)}{2}}.$$

Since $-2\binom{\frac{1}{2}}{2i+6}>0$, we have that

$$2\sqrt{t}-\sqrt{t+1}-\sqrt{t-1}>\frac{c}{4t^{\frac{3}{2}}}+\frac{5c}{64t^{\frac{7}{2}}}.$$

Thus

$$\exp(2c\sqrt{t}-c\sqrt{t+1}-c\sqrt{t-1})>\exp\left(\frac{c}{4t^{\frac{3}{2}}}+\frac{5c}{64t^{\frac{7}{2}}}\right).$$

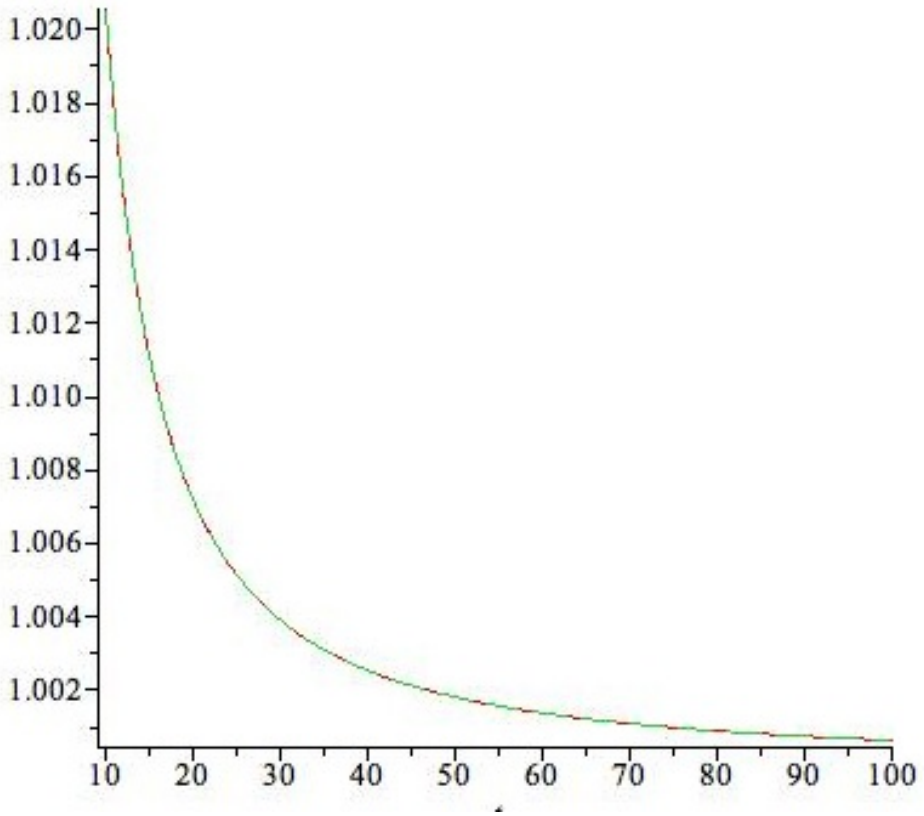
Figure 1.1 plots this approximation.

Next we consider the rational function $\sqrt{t}$. We have that

$$\frac{(t^{\frac{3}{2}})^2}{(t+1)^{\frac{3}{2}}(t-1)^{\frac{3}{2}}}=1-\frac{3}{2t^2}+\frac{3}{8t^4}+\cdots>1-\frac{3}{2t^2}.$$

Figure 1.2 plots this approximation.

Figure 1.1: A plot of $\exp(2c\sqrt{t} - c\sqrt{t+1} - c\sqrt{t-1})$ and $\exp\left(\frac{c}{4t^{\frac{3}{2}}} + \frac{5c}{64t^{\frac{7}{2}}}\right)$



Last we want to consider

$$\frac{(c\sqrt{t}-1)^2}{(c\sqrt{(t-1)}-1)(c\sqrt{(t+1)}-1)}.$$

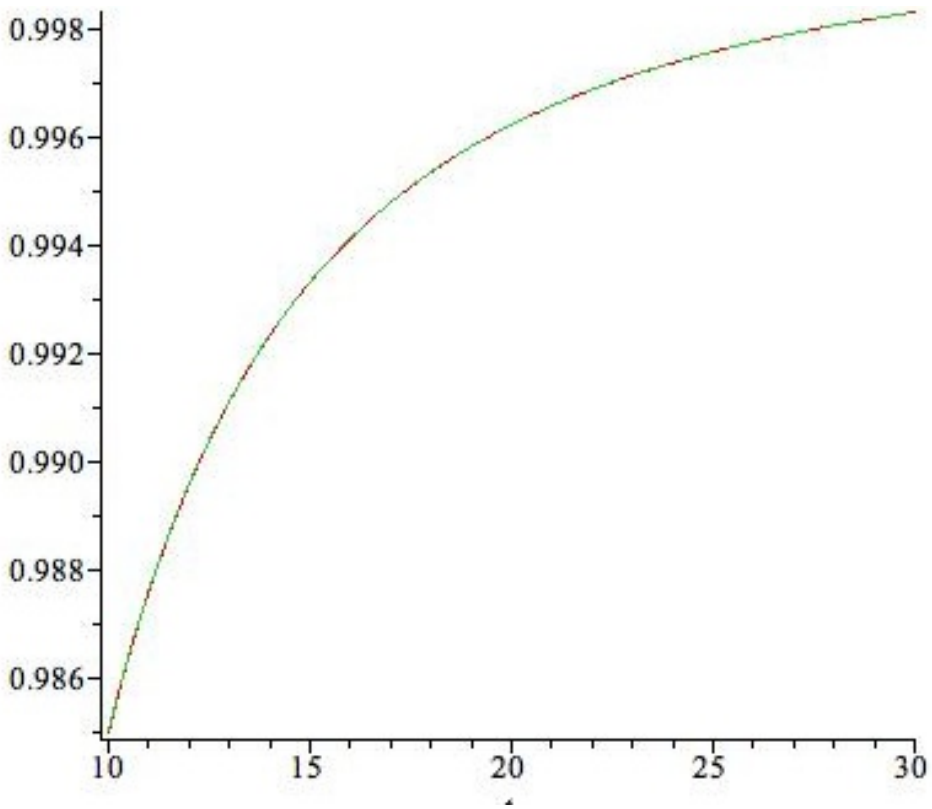
We have that

$$\frac{(c\sqrt{t}-1)^2}{(c\sqrt{(t-1)}-1)(c\sqrt{(t+1)}-1)} = 1 + \frac{1}{2t^2} + \frac{3}{4t^{5/2}c} + \cdots > 1 + \frac{1}{2t^2} + \frac{3}{4t^{5/2}c}.$$

Figure 1.3 plots this approximation.

Now we want to be able to show this term actually works. Let's consider only

Figure 1.2: A plot of $1 - \frac{3}{2t^2}$ and $\frac{(t^{\frac{3}{2}})^2}{(t+1)^{\frac{3}{2}}(t-1)^{\frac{3}{2}}}$



the denominator

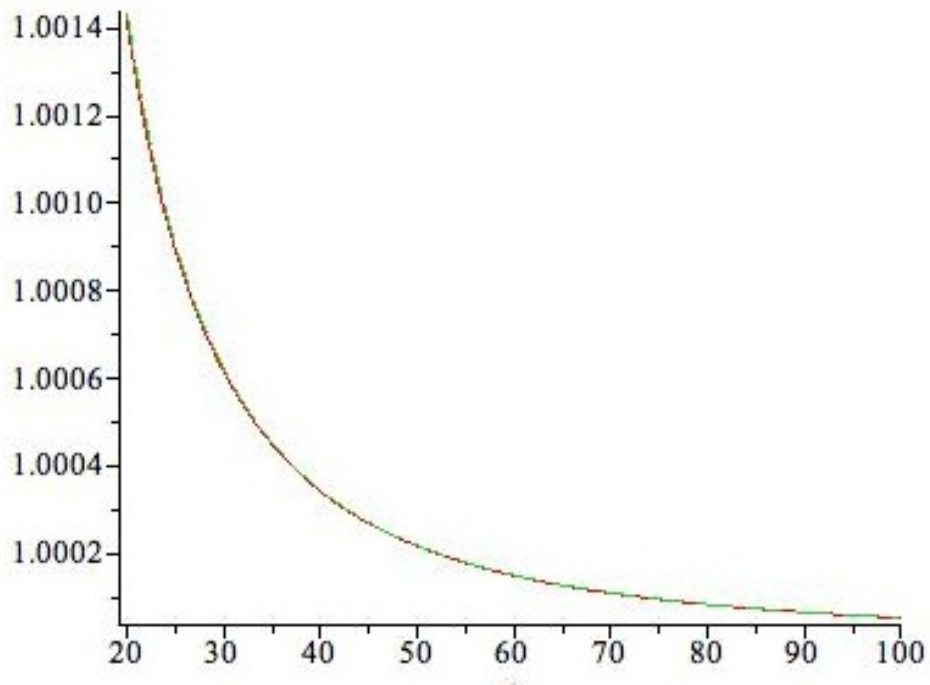
$$(c\sqrt{t-1}-1)(c\sqrt{t+1}-1) = c^2\sqrt{t-1}\sqrt{t+1} - c\sqrt{t+1} - c\sqrt{t-1} + 1.$$

Now

$$c^2\sqrt{1/x-1}\sqrt{1/x+1} = c^2\left(\frac{1}{x}\sqrt{1+x}\sqrt{1-x}\right) = \frac{c^2}{x} - c^2\sum_{i=1}^{\infty}\left|\binom{\frac{1}{2}}{i}\right| x^{2i-1}$$

$$= \frac{c^2}{x} - \frac{c^2}{2}x - \frac{c^2}{8}x^3 - \ldots,$$

Figure 1.3: A plot of $1+\frac{1}{2t^2}+\frac{3}{4t^{5/2}c}$ and $\frac{(c\sqrt{t}-1)^2}{(c\sqrt{(t-1)}-1)(c\sqrt{(t+1)}-1)}$



$$c\sqrt{1/x+1}=c\sum_{i=0}^{\infty}\binom{\frac{1}{2}}{i}x^{(2i-1)/2},$$

and

$$c\sqrt{1/x-1}=\frac{x}{\sqrt{x}}-c\sum_{i=01}^{\infty}\left|\binom{\frac{1}{2}}{i}\right|x^{(2i-1)/2}.$$

So we have

$$\frac{1}{4\pi\sqrt{2}t^{3/2}}(c\sqrt{t}-1)\exp(c\sqrt{t})>\exp\left(\frac{c}{4t^{\frac{3}{2}}}+\frac{5c}{64t^{\frac{7}{2}}}\right)\left(1-\frac{3}{2t^2}\right)\left(1+\frac{1}{2t^2}+\frac{3}{4t^{5/2}c}\right)$$

$$>\exp\left(\frac{c}{4t^{\frac{3}{2}}}+\frac{5c}{64t^{\frac{7}{2}}}\right)\left(1-\frac{1}{t^2}+\frac{3}{4t^{5/2}c}\right).$$

Next we want to consider

$$\frac{p_2(n)^2}{p_2(n-1)p_2(n+1)}, \text{ where } p_2(n) = \left(1 + \frac{(-1)^n(c\sqrt{t}-2)}{\sqrt{2}(c\sqrt{t}-1)} \exp\left(-\frac{c\sqrt{t}}{2}\right)\right).$$

We know that for n even this will contribute a positive amount, so we are only interested when n is odd. Let

$$\epsilon_{i+2} = \frac{(-1)^n(c\sqrt{t+i}-2)}{\sqrt{2}(c\sqrt{t+i}-1)} \exp\left(-\frac{c\sqrt{t+i}}{2}\right).$$

So, for n odd, we can write

$$\begin{aligned}
\frac{p_2(n)^2}{p_2(n-1)p_2(n+1)} &= \frac{(1-\epsilon_2)^2}{(1+\epsilon_1)(1+\epsilon_3)} \\
&= (1-2\epsilon_2+\epsilon_2^2)\left(1-\epsilon_1+\frac{\epsilon_1^2}{1+\epsilon_1}\right)\left(1-\epsilon_3+\frac{\epsilon_3^2}{1+\epsilon_3}\right) \\
&> (1-2\epsilon_2)(1-\epsilon_1)(1-\epsilon_3) \\
&= 1-2\epsilon_2-\epsilon_1-\epsilon_3+2\epsilon_1\epsilon_2+2\epsilon_2\epsilon_3+\epsilon_1\epsilon_3-2\epsilon_1\epsilon 2\epsilon 3 \\
&> 1-2\epsilon_2-\epsilon_1-\epsilon_3,
\end{aligned}$$

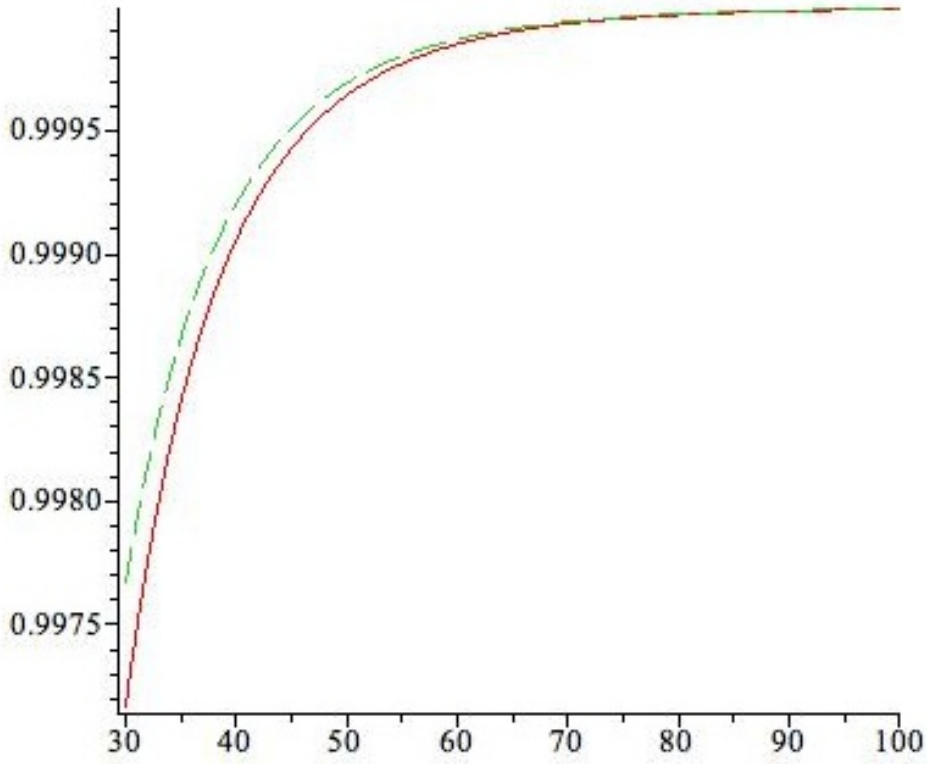
as $2\epsilon_1\epsilon_2 + 2\epsilon_2\epsilon_3 + \epsilon_1\epsilon_2 > 2\epsilon_1\epsilon_2\epsilon_3$.

Thus

$$\begin{aligned}
\frac{p_2(n)^2}{p_2(n-1)p_2(n+1)} &> 1 - \frac{1}{\sqrt{2}}\left(\exp\left(-\frac{c\sqrt{t-1}}{2}\right) + 2\exp\left(-\frac{c\sqrt{t}}{2}\right) + \exp\left(-\frac{c\sqrt{t+1}}{2}\right)\right) \\
&> 1 - \frac{4}{\sqrt{2}}\exp\left(-\frac{c\sqrt{t-1}}{2}\right).
\end{aligned}$$

Figure 1.4 plots this approximation.

Figure 1.4: A plot of $1 - \frac{4}{\sqrt{2}} \exp\left(-\frac{c\sqrt{t-1}}{2}\right)$ and $\frac{p_2(n)^2}{p_2(n-1)p_2(n+1)}$



Next we want to consider

$$\frac{p_3(n)^2}{p_3(n-1)p_3(n+1)}, \text{ where } p_3(n) = \left(1 - \frac{2c\sqrt{t}}{\sqrt{3}}\left(\frac{c\sqrt{t}-3}{c\sqrt{t}-1}\right)\exp\left(-\frac{2c\sqrt{t}}{3}\right)\right).$$

Let

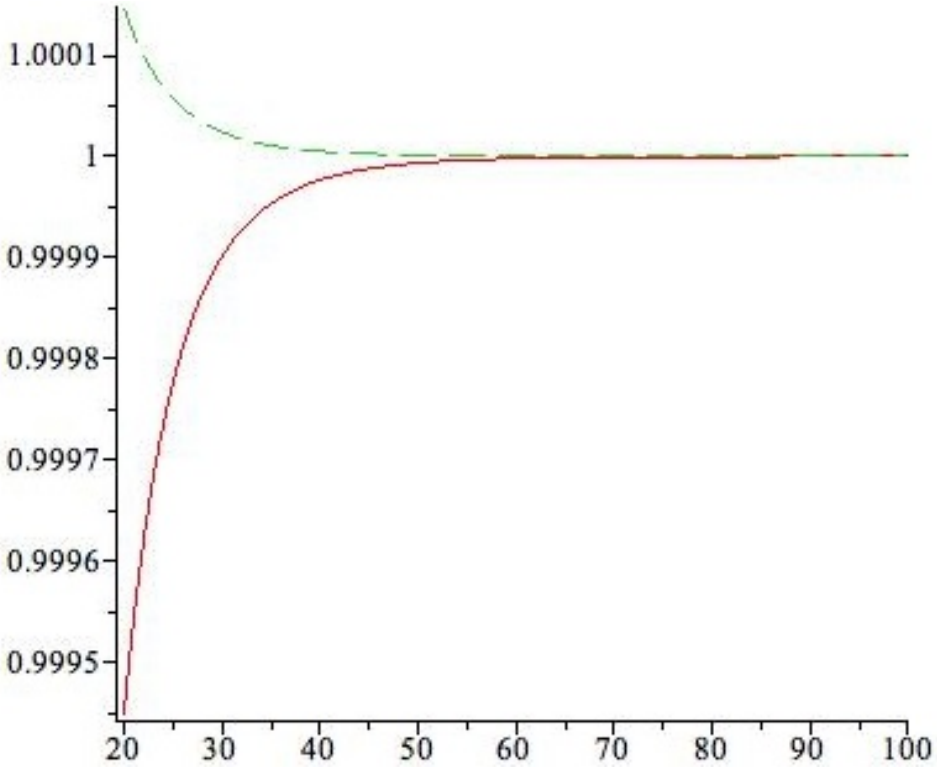
$$\epsilon'_{i+2} = \frac{2c\sqrt{t}}{\sqrt{3}}\left(\frac{c\sqrt{t}-3}{c\sqrt{t}-1}\right)\exp\left(-\frac{2c\sqrt{t}}{3}\right).$$

Then

$$\begin{aligned}
\frac{p_3(n)^2}{p_3(n-1)p_3(n+1)} &= \frac{(1-\epsilon_2')^2}{(1-\epsilon_1')(1-\epsilon_3')} \\
&= (1-2\epsilon_2'+(\epsilon_2')^2\left(1+\epsilon_1'+\frac{(\epsilon_1')^2}{1-\epsilon_1'}\right)\left(1+\epsilon_3'+\frac{(\epsilon_3')^2}{1-\epsilon_3'}\right) \\
&> 1-2\epsilon_2' \\
&= 1-2\frac{2c\sqrt{t}}{\sqrt{3}}\left(\frac{c\sqrt{t}-3}{c\sqrt{t}-1}\right)\exp\left(-\frac{2c\sqrt{t}}{3}\right) \\
&> 1-\frac{2}{\sqrt{3}}\exp\left(-\frac{2c\sqrt{t}}{3}\right).
\end{aligned}$$

Figure 1.5 plots this approximation.

Figure 1.5: A plot of $1-\frac{2}{\sqrt{3}}\exp\left(-\frac{2c\sqrt{t}}{3}\right)$ and $\frac{p_3(n)^2}{p_3(n-1)p_3(n+1)}$



So we have that, for n odd,

$$\begin{aligned}\frac{p(n)^2}{p(n+1)p(n-1)} &> \exp\left(\frac{c}{4t^{\frac{3}{2}}}+\frac{5c}{64t^{\frac{7}{2}}}\right)\left(1-\frac{1}{t^2}+\frac{3}{4t^{5/2}c}\right)\times \\ &\quad \left(1-\frac{4}{\sqrt{2}}\exp\left(-\frac{c\sqrt{t-1}}{2}\right)\right)\left(1-\frac{2}{\sqrt{3}}\exp\left(-\frac{2c\sqrt{t}}{3}\right)\right) \\ &> \exp\left(\frac{c}{4t^{\frac{3}{2}}}\right)\left(1-\frac{1}{t^2}\right)\left(1-\frac{4}{\sqrt{2}}\exp\left(-\frac{c\sqrt{t-1}}{2}\right)\right)\times \\ &\quad \left(1-\frac{2}{\sqrt{3}}\exp\left(-\frac{2c\sqrt{t}}{3}\right)\right).\end{aligned}$$

Figure 1.6 shows a plot of the exact value $\frac{p(n)^2}{p(n+1)p(n-1)}$ and the estimate given above.

We also have that

$$\exp\left(\frac{c}{4t^{\frac{3}{2}}}\right)\left(1-\frac{1}{t^2}\right) > 1+\frac{c}{4t^{3/2}}.$$

Figure 1.7 shows this approximation.

So we have

$$\frac{p(n)^2}{p(n+1)p(n-1)} > \left(1+\frac{c}{4t^{3/2}}\right)\left(1-\frac{4}{\sqrt{2}}\exp\left(-\frac{c\sqrt{t-1}}{2}\right)\right)\left(1-\frac{2}{\sqrt{3}}\exp\left(-\frac{2c\sqrt{t}}{3}\right)\right).$$

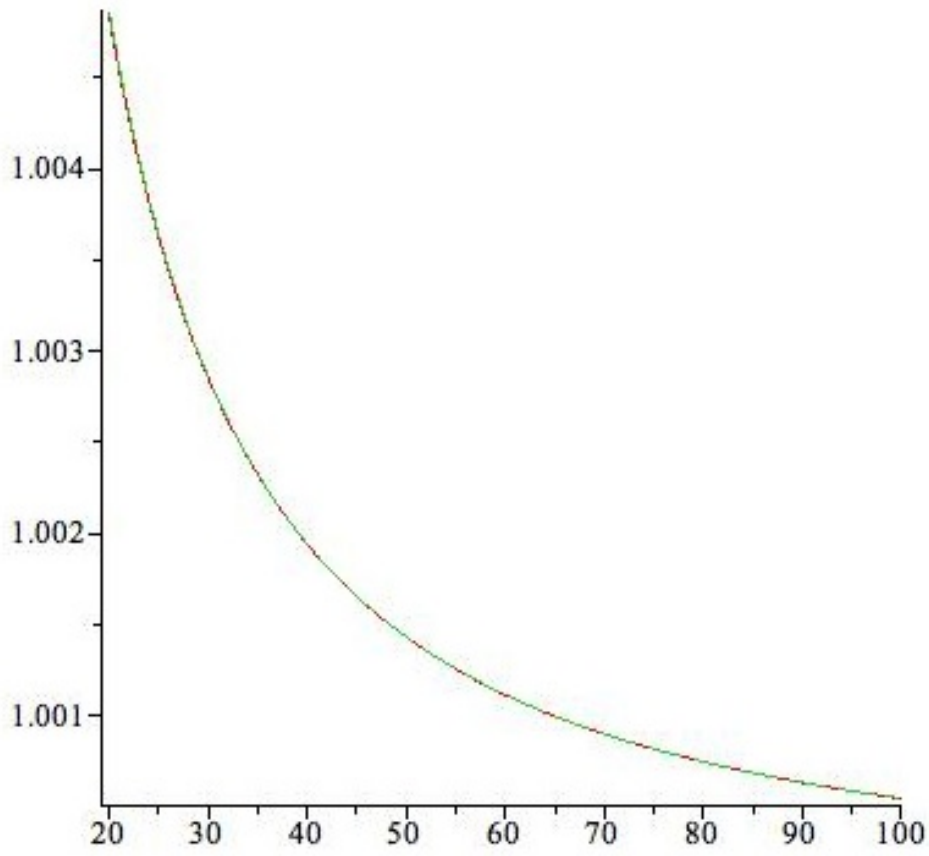
Let

$$a=\frac{c}{4t^{3/2}} \quad , \quad b=\frac{4}{\sqrt{2}}\exp\left(-\frac{c\sqrt{t-1}}{2}\right) \quad , \quad d=\frac{2}{\sqrt{3}}\exp\left(-\frac{2c\sqrt{t}}{3}\right).$$

We want to show that

$$(1+a)(1-b)(1-d)>1$$

Figure 1.6: A plot of the exact value of $\frac{p(n)^2}{p(n+1)p(n-1)}$ and the approximation based on estimates of Rademacher's formula.



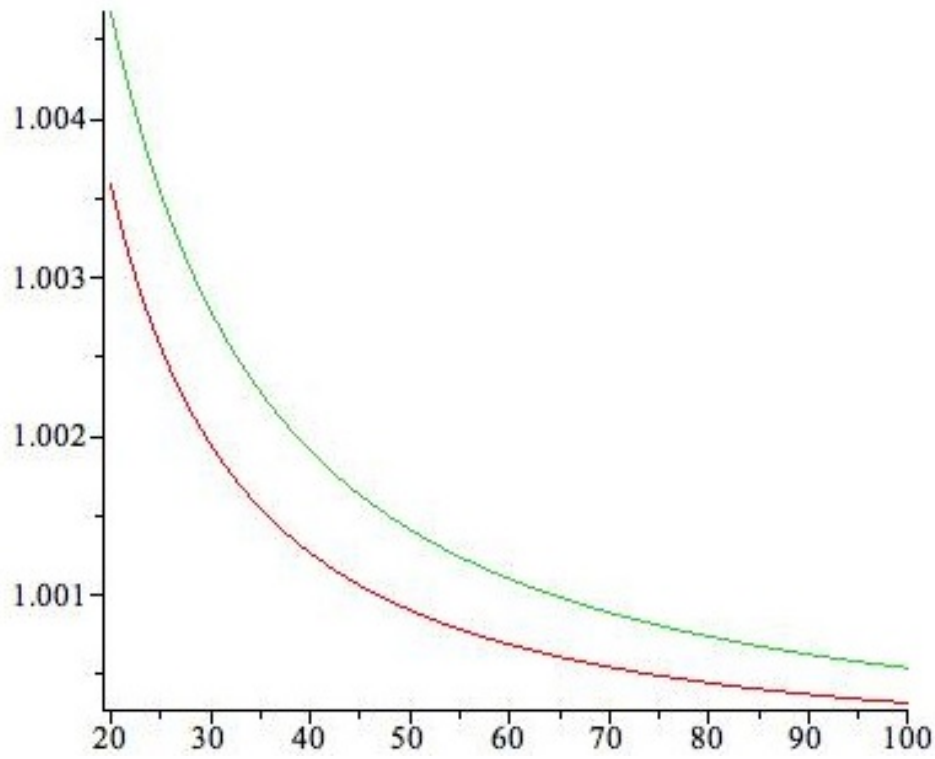
for all $t \geq T$. We need

$$1 - d - b - ad - ab + bd + a + abd > 1.$$

That is, we need to show that

$$a + bd + abd > a > (d + b) + a(d + b) = (d + b)(a + 1),$$

Figure 1.7: A plot of $\exp\left(\frac{c}{4t^{\frac{3}{2}}}\right)\left(1-\frac{1}{t^2}\right)$ and $1+\frac{c}{4t^{3/2}}$



i.e.

$$\frac{a}{1+a} > \frac{a}{2} > d + b.$$

Now we have that

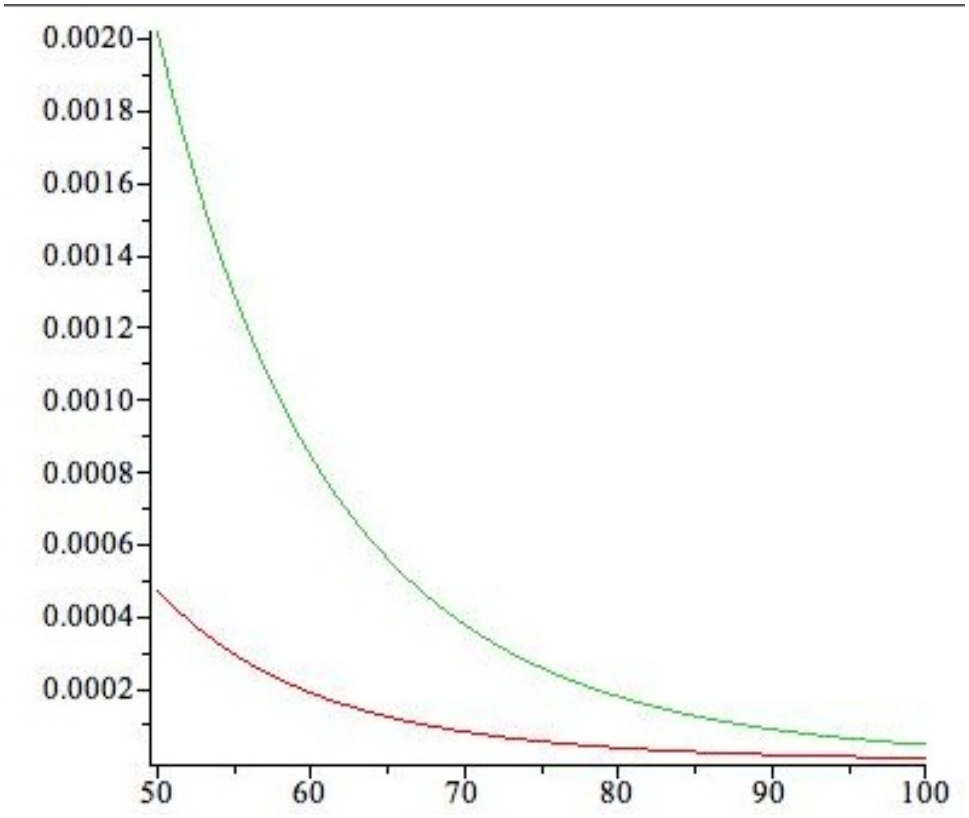
$$2\exp\left(-\frac{c\sqrt{t-1}}{2}\right) > \frac{4}{\sqrt{2}}\exp\left(-\frac{c\sqrt{t-1}}{2}\right) + \frac{2}{\sqrt{3}}\exp\left(-\frac{2c\sqrt{t}}{3}\right) = b + d.$$

Figure 1.8 shows this approximation.

So we want to show that

$$\frac{c}{16t^{3/2}} > 2\exp\left(-\frac{c\sqrt{t-1}}{2}\right) > \frac{4}{\sqrt{2}}\exp\left(-\frac{c\sqrt{t-1}}{2}\right) + \frac{2}{\sqrt{3}}\exp\left(-\frac{2c\sqrt{t}}{3}\right).$$

Figure 1.8: A plot of $b + d$ and $2\exp\left(-\frac{c\sqrt{t-1}}{2}\right)$



That is, we want to show

$$2.5651 \approx c > 32t^{3/2}\exp\left(-\frac{c\sqrt{t-1}}{2}\right).$$

Note that

$$32(41)^{3/2}\exp\left(-\frac{c\sqrt{41-1}}{2}\right) \approx 2.5207.$$

Now

$$\frac{d}{dt}32t^{3/2}\exp\left(-\frac{c\sqrt{t-1}}{2}\right) = 48t^{1/2}\exp\left(-\frac{c\sqrt{t-1}}{2}\right) - \frac{8t^{3/2}c}{\sqrt{t-1}}\exp\left(-\frac{c\sqrt{t-1}}{2}\right) < 0,$$

for $t \geq 41$.

Since the derivative is negative, we know that

$$32t^{3/2} \exp\left(-\frac{c\sqrt{t-1}}{2}\right)$$

is decreasing for $t \geq 41$. Thus we have that

$$c > 32t^{3/2} \exp\left(-\frac{c\sqrt{t-1}}{2}\right)$$

for $t \geq 41$.

So we have

$$\frac{p(n)^2}{p(n+1)p(n-1)} > \left(1+\frac{c}{4t^{3/2}}\right)\left(1-\frac{4}{\sqrt{2}}\exp\left(-\frac{c\sqrt{t-1}}{2}\right)\right)\left(1-\frac{2}{\sqrt{3}}\exp\left(-\frac{2c\sqrt{t}}{3}\right)\right) > 1$$

for $t \geq 41$.

It is easy to computationally show that for $26 \leq t \leq 40$ unimodality will hold. When $t < 26$ we computationally notice that $p(n)$ alternately satisfies and does not the property of unimodality. Using our above estimates we can see that before 41 our inequalities only hold for t even. Recall our estimate for $p(n)$ is

$$p(n) > \frac{1}{4\pi\sqrt{2}t^{3/2}}(c\sqrt{t}-1)\exp(c\sqrt{t})\left(1+\frac{(-1)^n\sqrt{2}(c\sqrt{t}-2)}{(c\sqrt{t}-1)}\exp\left(-\frac{c\sqrt{t}}{2}\right)\right)\times$$

$$\left(1-\frac{2c\sqrt{t}}{\sqrt{3}}\left(\frac{c\sqrt{t}-3}{c\sqrt{t}-1}\right)\exp\left(-\frac{2c\sqrt{t}}{3}\right)\right).$$

When $n = 2m$, the sign on the second term is computed by $(-1)^{2m}$; this term will contribute a positive amount. When $n = 2m + t$ the second term will be negative. This negative term will contribute a large enough amount so that the condition of

log-concavity will not be satisfied.

1.3 Unimodality

Recall the definition for unimodality.

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

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

We have expanded the values of n for which $\mathcal{P}_n$ is known to be unimodal from $n \leq 2,000$ to $n \leq 10,000$. To do accomplish this, we have created a Sage algorithm that generates a matrix for which the (n,k) entry represents $p(n,k)$ according to the recursive formula $p(n,k) = p(n-1,k-1) + p(n-k,k)$. We then checked that each row of the matrix was unimodal. We also calculated the rank of the mode of $\mathcal{P}_n$ up to $n = 10,000$.

Additionally, we used an improved algorithm that 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 hoped to be able to expand on Szekeres' paper on unimodality [18] to find n large enough so that for all $N > n$, $\mathcal{P}_n$ is unimodal. We have not made enough progress on this area of research to include it in this paper. We suspect that "large enough" will not exceed 25,000, and so a proof of unimodality for all n would be complete.

1.4 Bipartite Matching

Recall the definition of a bipartite matching.

Definition. *We say that $\mathcal{P}_n$ has 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.*

Recall we write

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

where

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

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 consider 3 different types of level sets. Level sets from $\Lambda_{n,1}$ to $\Lambda_{n,m}$, level sets from $\Lambda_{n,m}$ to $\Lambda_{n,\lceil\frac{n}{2}\rceil}$, and level sets from $\Lambda_{n,\lfloor\frac{n}{2}\rfloor}$ to $\Lambda_{n,n}$.

1.4.1 $\Lambda_{n,1}$ 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:

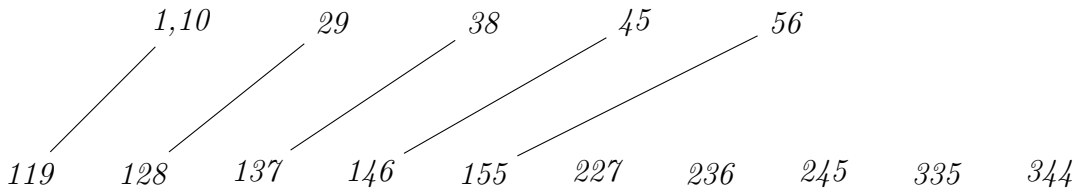
Theorem. *For all $n > 6$ there exists a maximum matching from $\Lambda_{n,2}$ to $\Lambda_{n,3}$.*

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.

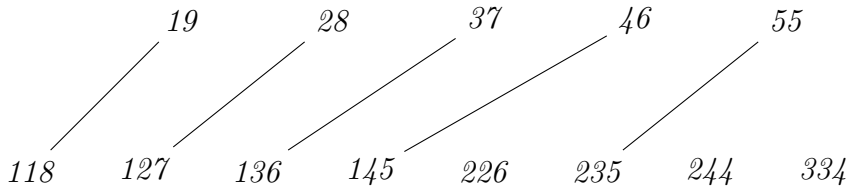
When n is even, we must consider the exception where $x = y = \frac{n}{2}$. This case will occur only once for each n, and we use the mapping such that $\left(2, \frac{n}{2} - 2, \frac{n}{2}\right) \lessdot \left(\frac{n}{2}, \frac{n}{2}\right)$. We know that $\left(2, \frac{n}{2} - 2, \frac{n}{2}\right)$ 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.

□

Example. *A mapping from $\Lambda_{11,2}$ to $\Lambda_{11,3}$.*



Example. *A mapping from $\Lambda_{10,2}$ to $\Lambda_{10,3}$.*



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 $x \leq y \leq z$.

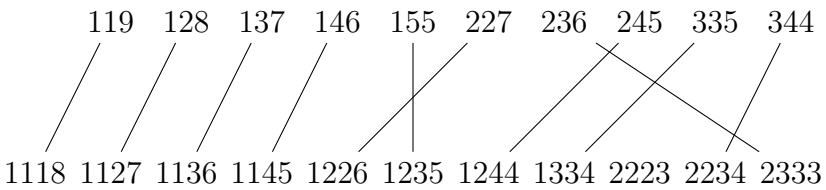
Let $\phi : \Lambda_{n,3} \to \Lambda_{n,4}$ be a function defined as follows:

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 $\phi((x, y, z)) = (1, x, y, z-1)$, unless it has already been matched by (1)
3. If $y = z$, then $\phi((x, y, y)) = (2, x, y-2, y)$,

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 $\Lambda_{11,4}$.*

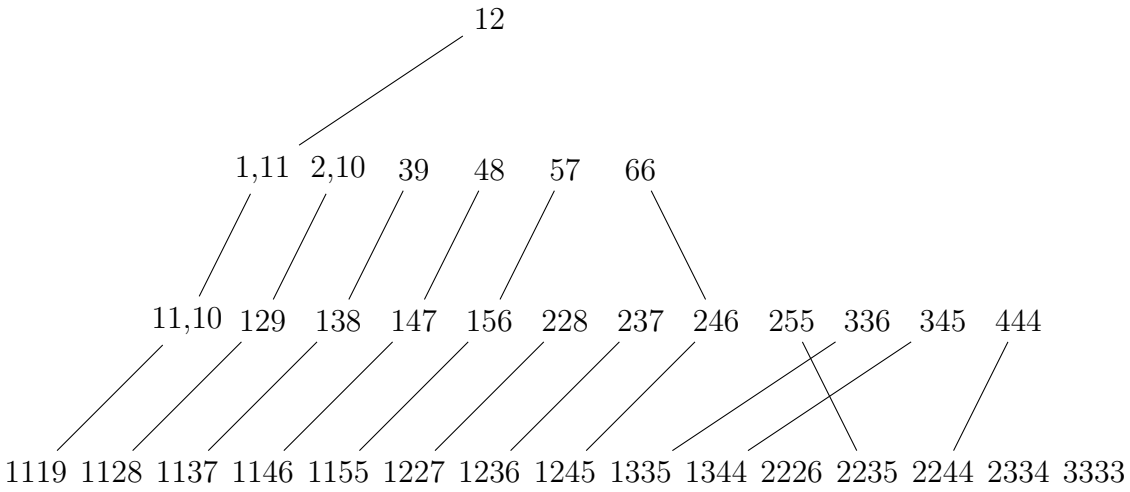


Using schemes similar to the ones listed in the proofs above, we hope that we may be able to categorize the way that mappings are formed from Λ_k to Λ_{k+1} where $k+1 < m$.

Next we describe an alternate idea for finding maximum matchings from $\Lambda_{n,1}$ to $\Lambda_{n,m}$. We have produced a matching scheme which works from $\Lambda_{n,1}$ to $\Lambda_{n,m-1}$ for $n \leq 34$; 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. In cases where this rule fails for $\Lambda_{n,m-1}$ to $\Lambda_{n,m}$ we have observed that $p(n, m-1) \approx p(n, m)$.

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 $k = 4$.*



1.4.2 $\Lambda_{n,m}$ 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 $\lambda' = \left(1(j-1)\bar{\lambda}\right)$. 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 that 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. There is much work remaining for these types of level sets. Com-

putationally we can use the ideas above as partial matchings to seed a matching algorithm, such as Ford-Fulkerson.

1.4.3 $\Lambda_{n,\lfloor \frac{n}{2} \rfloor}$ to $\Lambda_{n,n}$

Lemma 1.4.1. *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 the sake of contradiction, $\lambda \in \Lambda_{n,k}$ has $(2k-n-\delta)$ summands equal to 1, for some $\delta \in \mathbb{N}$. Then there are $k-(2k-n-\delta) = n-k+\delta$ summands which are greater than or equal to 2. 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,

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

$$2, \bar{\lambda} = 2, \bar{\lambda'}$$

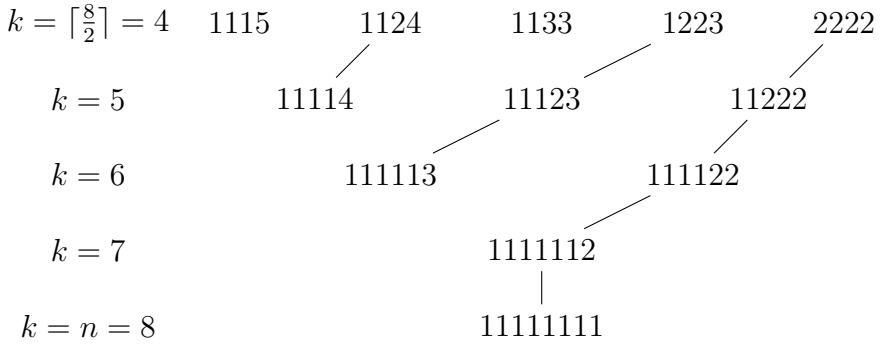
$$\bar{\lambda} = \bar{\lambda}'$$

$$\lambda = 1^2\bar{\lambda} = 1^2\bar{\lambda}' = \lambda'.$$

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.

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

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

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.

□

1.5 Conclusion

We have shown that $p(n)$ is log-concave for $n \geq 26$. We have also seen that for $n < 26$, $p(n)$ alternately does and does not satisfy the property of log-concavity. We have found that $\mathcal{P}_n$ is unimodal for all $n \leq 25,000$.

We have also proven several matching schemes which apply for all n. There are still many times of level sets which need matching schemes in order to prove the Sperner property for $\mathcal{P}_n$ for all n.

There are several areas to continue working on this problem. The first is to return to trying to find a large enough n so that $\mathcal{P}_N$ is unimodal for all $N > n$. This work would be based on tightening the asymptotics done by Szekeres.

We also hope to be able to show that the bipartite matching holds for all level sets. We can also try a computational approach to pushing this bound. Using an algorithm such as Ford-Fulkerson we can easily increase Canfield's bound of 45. Another idea would be to run Ford-Fulkerson with a partial matching. There are

certain edges that will be included with a high probability, so having the algorithm run assuming these edges will help to speed up run time.

Chapter 2

Graph Nim

2.1 Introduction

Impartial two player games are games where players alternate turns with the same rules on the same set of finite set moves. Since there are a finite set of positions available in the game, the game must end when a player no longer can make a move. Impartial games include nim, sprouts, tic-tac-toe, and chomp. Games such as chess and go are not impartial.

Nim is a simple impartial two player game in which players take turns removing rocks from disjoint piles until there are no rocks remaining. The player to pick up the last rock or group of rocks is the winner [3].

The traditional game of nim has been studied on graphs with weighted edges [9, 10]. The game begins by selecting a starting vertex where a game piece is placed. Players then choose an edge incident to this vertex, and move the game piece along that edge while decreasing the weight on the edge to any non-negative integer smaller than it's original weight. The game continues until a player cannot move because all edges adjacent to the piece's vertex have weight zero.

In our version of Graph Nim, the players take turns removing edges that are incident to a given vertex. The maximum number of edges that can be removed in a player's turn is equal to the degree of a specific vertex. For example, if a vertex has degree 4, a player can remove 1, 2, 3, or all 4 edges from that vertex. The object of Graph Nim is to be the person to remove the last set of edges from a given vertex.

We are interested in studying several different types of graphs using the Sprague-Grundy function. The analysis of paths, caterpillars, and G-paths is joint work with Neil Calkin and Kevin James while advising Sarah Leggett, Bryce Richards, Nathan Sitaraman, and Stephanie Thomas during the summer 2009 Clemson REU [6].

2.2 Sprague-Grundy

Definition. *A follower is a position a player can obtain in one move in a game.*

Definition. *Given a finite set of integers S, x is the minimum excluded value, mex, if it is the smallest non-negative integer such that $x \notin S$.*

Let $F(x)$ denote the followers of a given position x. The Sprague-Grundy function, $g(x)$, is defined as

$$g(x) = \text{ mex } \{g(y) : y \in F(x)\}.$$

Positions in impartial games, such as Nim, are either N-positions or P-positions. A P-position is winning for the previous player, while the N-position is winning for the next player. For the traditional game of Nim, the winning strategy is to finish every move leaving the game's Sprague-Grundy value at zero due to the following theorem.

Theorem. *A position in Nim is a P-position if and only if the nim-sum of its components is zero. [8]*

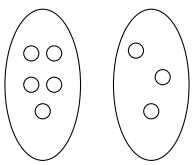
It follows from the definition of the Sprague-Grundy function that once a player is given a position in a game with a Sprague-Grundy number equal to zero, then any move that player makes will change the value of the game to some non-zero Sprague-Grundy number. It is also known that a player can force a Sprague-Grundy value of zero onto the next player only if the player was not handed a Sprague-Grundy value of zero at the beginning of their turn.

Sprague-Grundy function values are helpful when analyzing Nim played on graphs. The Sprague-Grundy theorem explains the reason why we take interest in computing Sprague-Grundy numbers.

Theorem. *(Sprague-Grundy Theorem) The Sprague-Grundy, S-G, value of a game consisting of many disjoint games is the nim-sum of the Sprague-Grundy values of those components.*

In the case of traditional Nim, the S-G number of the entire game is the addition of each pile's S-G number. In other words, we can consider each pile as a distinct game with its own S-G number. To calculate a nim-sum with traditional Nim, we note that the S-G number for a pile of rocks is simply the number of rocks in the pile. We then take the S-G numbers from all piles and convert them to binary. The nim-sum is then found by the addition of all converted S-G numbers mod 2.

For example, in a game with two piles below,



we take the number of rocks in each pile, 5 and 3, and convert to binary. We then nim-sum to find a value of 6. Since the S-G number for this game is greater than zero, player one has a winning strategy.

With the nim-sum calculated for a given game, it is possible for a player to determine whether a game win is achievable. We know that a game with a nim-sum of zero is in P-position. Otherwise, if the nim-sum is non-zero, the game is in N-position. So in order for a player to win a game of Nim from an N-position, he should remove enough rocks to force the nim-sum to zero.

The strategy for forcing a nim-sum to zero is as follows:

1. Find the left most 1 in the calculated sum, say column k.
2. From that 1, trace up the column until you find a 1. Note that the row you found this 1 in will be the only row to be modified, say row r.
3. Change the 1 is row r column k to a 0.
4. Manipulate the 1's and 0's in the same row we changed in the last step so that each column's nim-sum is zero.
5. Subtract the value of the altered form of row r from the original value of row r to determine the winning move. The difference in the rows will be the number of rocks you need to remove from the pile represented by row r.

Let's demonstrate how this strategy works. Take into consideration the game played on two piles mentioned earlier. The nim-sum we calculated was:

$$\begin{array}{rccc} & 1 & 0 & 1 \\ + & & 1 & 1 \\ \hline & 1 & 1 & 0 \end{array}$$

Step 1: We see that the first 1 is in column one.

$$\begin{array}{rccc} & 1 & 0 & 1 \\ + & & 1 & 1 \\ \hline & \mathbf{1} & 1 & 0 \end{array}$$

Step 2: We will modify row one.

$$\begin{array}{rccc} & \mathbf{1} & 0 & 1 \\ + & & 1 & 1 \\ \hline & 1 & 1 & 0 \end{array}$$

Step 3: Change the 1 in row one column one to a 0.

$$\begin{array}{rccc} & 0 & 0 & 1 \\ + & & 1 & 1 \\ \hline & 0 & 1 & 0 \end{array}$$

Step 4: Change the 0 in row one column two to a 1, forcing column two to have a nim-sum of 0.

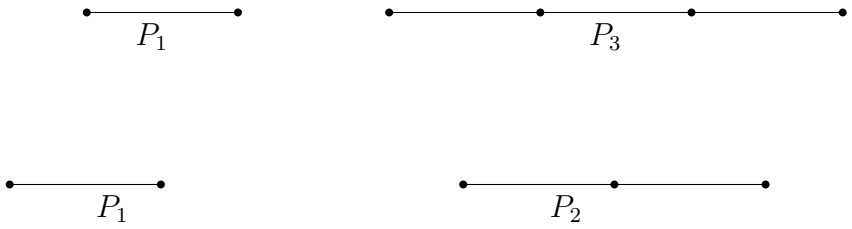
$$\begin{array}{rccc} & 0 & 1 & 1 \\ + & & 1 & 1 \\ \hline & 0 & 0 & 0 \end{array}$$

Since $5 - 3 = 2$, we remove 2 rocks from the pile represented in the first row. Since the next player will have no choice but to change the nim-sum to a non-zero value, we know that we are able to force the nim-sum to zero again. We continue this pattern until the game is won.

2.3 Graph Nim on Paths

Definition. *A path with n edges, denoted P_n, is a tree with two vertices of degree one and all other vertices of degree two.*

When playing Path Nim, we consider disjoint paths instead of disjoint piles of rocks. A move is made by by removing either one edge, or two edges connected to the same vertex. For example, from P_5, we can take away one edge to create two disjoint paths P_1 and P_3. Similarly, we can take away two edges to create disjoint paths P_1 and P_2 as shown below.



We can also take away one edge, therefore creating two paths of equal length. Creating two paths of equal length turns out to be the winning strategy for Player 1.

By creating two paths of equal length, we force each path to have the same Sprauge-Grundy number. It is easy to see that when we nim-sum two equal numbers, we get zero. Therefore, if we are faced with P_{2c}, where $c \in \mathbb{Z}^+$, we remove the two inner edges incident with the center vertex. Similarly, if we are faced with P_{2c+1}, we only need to remove the center edge. Therefore, since we know that Player 1 can always force the nim-sum to zero at the end of their turn, paths can always be won by Player 1. Figure 2.1 shows the Sprague-Grundy numbers for paths. The column labels $0-11$ represent the least residues of the congruence $n \mod 12$. The row labels represent the length of path in intervals of 12.

We notice that, a periodic behavior occurs, so that a path of length greater than 72 has an easily computable S-G number. Say that we have a path of length

Figure 2.1: A table of S-G numbers for P_n. The column labels $0-11$ represent the least residues of the congruence $n \mod 12$. The row labels represent the length of path in intervals of 12. We see a periodic behavior starting at length 72.

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	2	3	1	4	3	2	1	4	2	6
12	4	1	2	7	1	4	3	2	1	4	6	7
24	4	1	2	8	5	4	7	2	1	8	6	7
36	4	1	2	3	1	4	7	2	1	8	2	7
48	4	1	2	8	1	4	7	2	1	4	2	7
60	4	1	2	8	1	4	7	2	1	8	6	7
72	4	1	2	8	1	4	7	2	1	8	2	7

n, where $n \geq 72$, we can use modular arithmetic to calculate its S-G number. For example, for $n = 87$, we have

$$87 \equiv 3 \mod 12 \Rightarrow g(87) = 8.$$

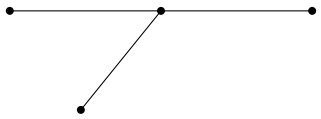
Note that when we compute S-G numbers for paths of length less than 72 there are multiple exceptions before the S-G numbers become become periodic.

2.4 Graph Nim on Caterpillars

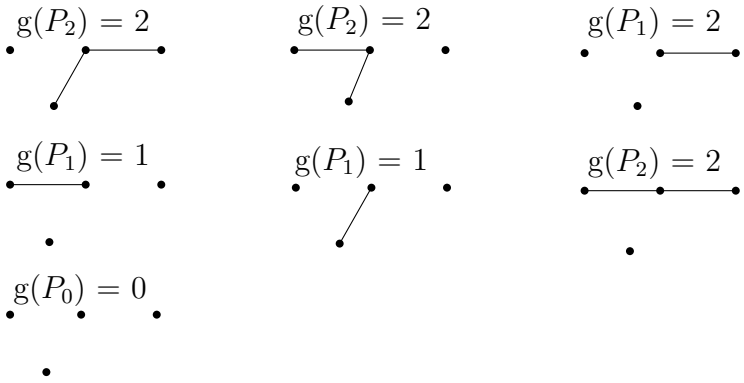
Definition. *A caterpillar, C_n, is defined as a path consisting of n edges with one or more edges appended to t vertices of the path, $t \geq 1$. Any vertex not in the path has degree one and distance one from the main path.*

Definition. *A caterpillar, $C_{n,k}$ is defined as a caterpillar of length n, where n is the number of edges, and consisting of one extra edge (or a leg) on index k, where the leftmost vertex is index zero.*

The gameplay of nim on caterpillars is similar to that on paths. Since there are legs attached to the main path, there are more possible moves available to a player. For example, consider $C_{2,1}$ shown below:

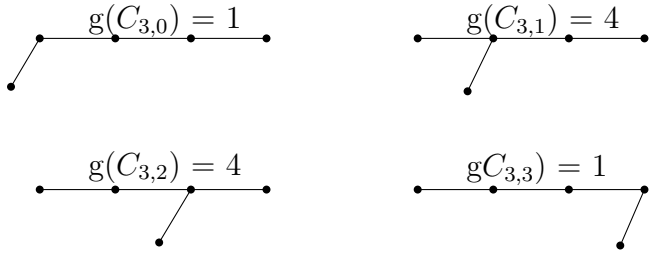


We can obtain the following graphs and corresponding S-G values in one move:



We see that the set of S-G values for followers of $C_{2,1}$ is $\{0, 1, 2\}$. Taking the mex of this set, we conclude that the S-G number of $C_{2,1}$ is 3.

We can continue to analyze caterpillars in this way, obtaining the following S-G numbers for caterpillars of length three:



When we fix the index of the extra edge and increase the length of the main path, the S-G numbers of caterpillars become periodic. These S-G values are similar to that of paths in that we have discovered periods of 12 in both.

Figure 2.2 shows the S-G values of caterpillars of the form $C_{n,1}$ and figure 2.3 shows the S-G values of caterpillars of the form $C_{n,16}$. Note that the periodic behavior of caterpillars does not always begin at the same length. The periodic behavior of $C_{n,1}$ caterpillars begins at length 156, while the periodic behavior of $C_{n,16}$ caterpillars does not begin until length 204.

Figure 2.2: A table of S-G numbers for $C_{n,1}$. The column labels $0-11$ represent the least residues of the congruence $n \mod 12$. The row labels represent the length of path in intervals of 12. We see a periodic behavior starting at length 156.

	0	1	2	3	4	5	6	7	8	9	10	11
0	$*$	2	3	4	5	6	2	1	0	8	6	0
12	1	2	3	8	5	12	7	1	0	8	9	14
24	1	2	3	11	4	7	12	14	0	16	2	4
36	12	2	3	10	4	7	15	1	16	9	18	16
48	12	2	3	10	16	7	12	1	16	18	11	16
60	12	2	22	11	16	7	12	1	20	24	16	26
72	12	13	22	11	16	24	15	14	16	22	19	16
84	12	13	19	11	16	24	15	14	16	25	11	16
96	12	13	22	11	16	7	15	1	20	25	19	11
108	12	13	22	11	32	19	22	14	20	22	19	11
120	12	13	22	11	25	19	22	14	16	22	19	11
132	21	13	22	11	25	19	22	14	21	25	19	11
144	21	13	22	11	25	19	22	14	20	22	19	11
156	21	13	22	11	25	19	22	14	21	22	19	11

2.5 Firework Graphs

Definition. *The star graph S_n is a tree on n vertices with one vertex having degree $n-1$ and $n-1$ vertices with degree one.*

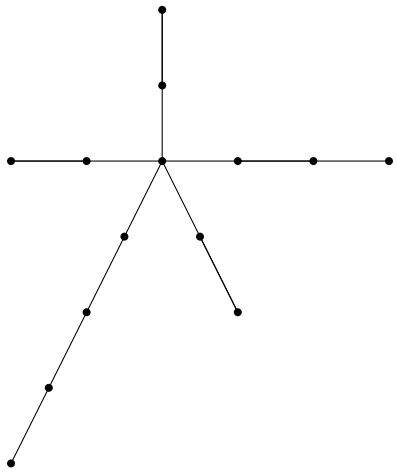
Figure 2.3: A table of S-G numbers for $C_{n,16}$. The column labels $0-11$ represent the least residues of the congruence $n \mod 12$. The row labels represent the length of path in intervals of 12. We see a periodic behavior starting at length 204.

	0	1	2	3	4	5	6	7	8	9	10	11
0	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
12	$*$	$*$	$*$	$*$	4	12	13	0	16	10	7	9
24	0	10	7	9	4	7	16	1	15	10	7	9
36	16	2	10	5	4	12	2	1	0	6	7	12
48	1	16	24	5	19	7	16	0	10	21	7	4
60	24	3	16	27	19	7	27	0	22	10	7	4
72	29	16	26	27	16	7	12	0	21	9	7	32
84	29	12	21	5	36	7	12	0	32	10	7	4
96	28	3	16	9	29	7	12	0	21	9	7	31
108	32	3	21	9	38	7	12	0	21	9	7	40
120	28	3	16	9	29	7	12	0	16	10	7	29
132	22	3	32	9	29	7	12	0	32	10	7	22
144	28	3	35	9	29	7	12	0	16	10	7	22
156	28	3	32	9	37	7	12	0	16	10	7	37
168	28	3	32	9	29	7	12	0	16	10	7	22
180	28	3	32	9	37	7	12	0	16	10	7	22
192	28	3	32	9	37	7	12	0	16	10	7	22
204	28	3	32	9	29	7	12	0	16	10	7	22

We are interested in graphs that are a generalization of a star graph.

Definition. *A firework graph $S_{a_1,a_2,a_3,\ldots a_n}$ is a tree on $a_1+a_2+\cdots+a_n+1$ vertices with a center vertex of degree $a_1+a_2+\cdots+a_n$, n leaves, and $a_1+a_2+\cdots+a_n-(n+1)$ vertices of degree two. That is, we append the paths $P_{a_1}, P_{a_2}, \ldots P_{a_n}$ at a center vertex.*

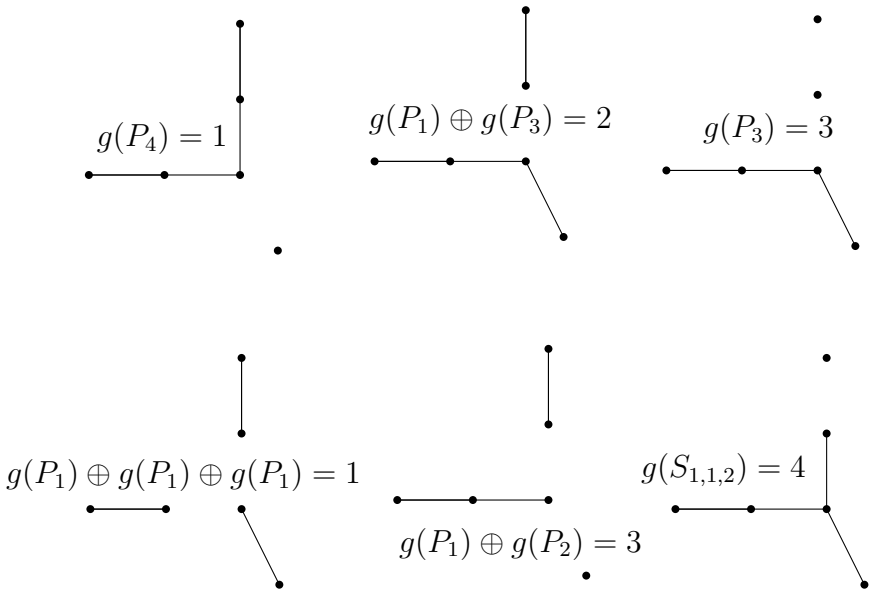
For example, $S_{2,2,2,3,4}$.



If we consider a firework graph consisting of two paths P_{a_1} and P_{a_2} then this graph is just a path of length $a_1 + a_2$. We will begin our analysis of fireworks with 3 paths.

We have two different types of moves we can make. We can remove edges that are adjacent to the center vertex. These moves will result in followers that consist only of paths. If we remove any other edges, the follower will be a path and a shorter 3-firework graph.

The followers for $S_{1,2,2}$.



If we consider fixing two of the paths and letting the third path grow we notice a periodic behavior of the S-G numbers. We see that for $S_{1,1,m}$ we have the same S-G numbers as $C_{n,1}$. The data for $S_{1,1,m}$ is shifted to the left by one position because, unlike for caterpillars, we can define $S_{1,1,0}$.

Next we look at 4-firework graphs, which consisting of four paths. We have two different types of moves we can make. We can remove edges that are adjacent to the center vertex. These moves will result in followers that are smaller firework graphs and paths. If we remove any other edges, the follower will be a path and a shorter 4-firework graph. Figure 2.6 shows the periodic behavior for $S_{1,1,1,m}$ as m grows.

Below are the followers for $S_{1,1,1,2}$.

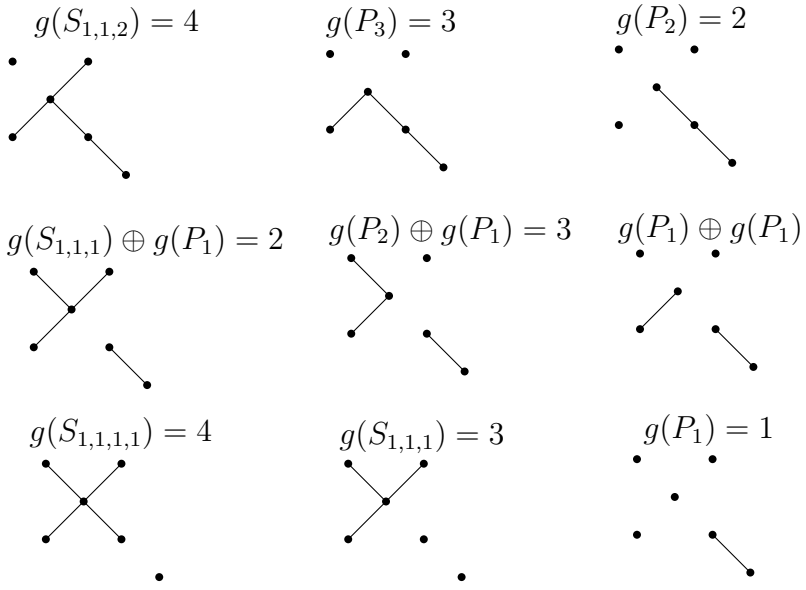


Figure 2.4: A table of S-G numbers for $S_{1,1,m}$. The column labels $0 - 11$ represent the least residues of the congruence $m \mod 12$. The row labels represent the length of path in intervals of 12.

	0	1	2	3	4	5	6	7	8	9	10	11
0	2	3	4	5	6	2	1	0	8	6	0	1
12	2	3	8	5	12	7	1	0	8	9	14	1
24	2	3	11	4	7	12	14	0	16	2	4	12
36	2	3	10	4	7	15	1	16	9	18	16	12
48	2	3	10	16	7	12	1	16	18	11	16	12
60	2	22	11	16	7	12	1	20	24	16	26	12
72	13	22	11	16	24	15	14	16	22	19	16	12
84	13	19	11	16	24	15	14	16	25	11	16	12
96	13	22	11	16	7	15	1	20	25	19	11	12
108	13	22	11	32	19	22	14	20	22	19	11	12
120	13	22	11	25	19	22	14	16	22	19	11	21
132	13	22	11	25	19	22	14	21	25	19	11	21
144	13	22	11	25	19	22	14	20	22	19	11	21
156	13	22	11	25	19	22	14	21	22	19	11	21

It is easy to generalize to an n-firework. Removing edges adjacent to the center vertex will give a follower consisting of paths and a k-firework graph where $0 \leq k < n$.

Figure 2.5: A table of S-G numbers for $S_{1,2,m}$. The column labels $0-11$ represent the least residues of the congruence $m \mod 12$. The row labels represent the length of path in intervals of 12.

	0	1	2	3	4	5	6	7	8	9	10	11
0	3	4	5	6	7	8	0	10	6	8	1	6
12	3	8	5	7	13	8	0	10	13	4	0	2
24	3	9	13	7	16	1	0	10	17	8	18	10
36	3	8	18	6	13	18	19	20	17	13	18	17
48	3	20	18	6	17	13	0	10	17	4	13	17
60	3	9	22	6	13	18	23	10	17	4	13	17
72	3	9	18	6	13	17	19	10	17	8	13	17
84	3	24	18	6	13	18	11	10	17	4	13	17
96	3	9	18	6	13	18	11	10	17	4	13	17
108	3	9	18	6	13	17	11	10	17	4	13	17
120	3	9	18	6	13	18	11	10	17	4	13	17

Figure 2.6: A table of S-G numbers for $S_{1,1,1,m}$. The column labels $0-11$ represent the least residues of the congruence $m \mod 12$. The row labels represent the length of path in intervals of 12.

	0	1	2	3	4	5	6	7	8	9	10	11
0	3	4	5	6	7	5	0	7	6	5	1	8
12	3	10	5	6	3	10	0	8	6	10	0	2
24	3	10	5	9	12	1	0	8	6	10	15	2
36	3	10	5	6	16	14	0	8	6	5	18	13
48	3	10	5	6	15	14	18	8	6	5	12	9
60	3	10	5	6	15	14	18	8	6	4	18	9
72	3	10	18	20	15	14	18	8	16	18	12	9
84	3	10	18	12	15	14	18	8	6	18	12	9
96	3	10	5	12	15	14	18	8	6	5	12	9
108	3	10	22	12	15	14	18	8	16	25	12	9
120	3	10	18	12	15	14	18	8	17	25	12	9
132	3	10	18	12	15	14	18	8	6	25	12	9
144	3	10	22	12	15	14	18	8	25	28	12	9
156	3	10	18	12	15	14	18	8	25	28	12	9
168	3	10	18	12	15	14	18	8	20	25	12	9
180	3	10	18	12	15	14	18	8	25	28	12	9

Removing edges along one of the paths will result in a smaller n-firework graph and a path. Appendix A gives the periodic behavior for several firework graphs.

We note that, our firework graphs can be viewed as partitions. In particular, a k-firework gives all the partitions with k parts. We hope to be able to prove that all k-fireworks are periodic. If we can prove this, we can define equivalence classes for the partitions. After running data to look at all partitions of n, we will be able to determine if these equivalence classes are interesting.

2.6 Cycle-Paths

Next we want to explore what happens to the nimbers of cycles.

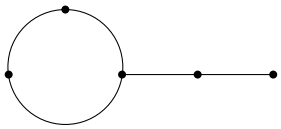
Definition. *A cycle of length, C_n, with $n \geq 3$ is a connected closed chain.*

For a cycle C_n there are only two possible follower graphs. Removing one edge will result in P_{n-1} and removing two edges will result in P_{n-2}. Since $g(P_m) > 0$ for all $m > 0$, $g(C_n) = 0$ for all n.

Next we want to study what will happen when we attach a path to a cycle.

Definition. *A cycle-path C_nP_m is a cycle of length n with a path of length m attached to one of the vertices of the cycle.*

For example C_3P_2 is shown below.



Any move made on C_3P_m will have followers which are combinations of paths and $S_{1,1,m}$. We see that the possible followers for C_3P_2 are

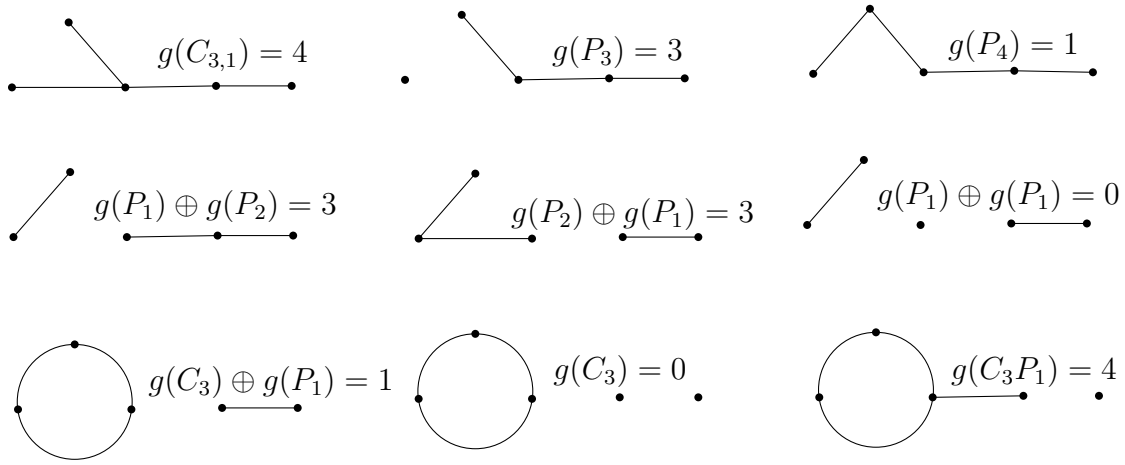


Figure 2.7: A table of S-G numbers for C_3P_m. The column labels $0 - 11$ represent the least residues of the congruence $n \mod 12$. The row labels represent the length of path in intervals of 12. We see a periodic behavior starting at length 84.

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	4	2	6	7	4	3	7	1	8	8	6
12	8	1	12	7	8	4	8	8	14	4	8	8
24	15	1	13	7	16	1	8	2	1	4	13	2
36	16	1	2	7	16	4	8	2	1	16	18	7
48	11	1	2	7	16	4	8	2	1	4	8	8
60	14	1	13	7	16	4	8	2	1	4	13	16
72	11	1	2	7	16	4	8	2	1	4	13	7
84	11	1	2	7	16	4	8	2	1	4	13	16

Next we want to consider the S-G numbers for C_4P_m. We see that all the followers are a combination of paths and stars-paths of the form $S_{1,2,m}$. Figure 2.8 shows the periodic nimbers for C_4P_m.

We note that computing the S-G numbers for cycle-paths C_nP_m will require 3-star paths. For example C_5P_m will require $S_{2,2,m}$ and C_6P_m will require $S_{2,2,m}$ and $S_{2,3,m}$. In general, $C_{2k-1}P_m$ will require a $S_{k,k,m}$ and $C_{2k}P_m$ will require $S_{k,k,m}$ and $S_{k,k+1,m}$. Appendix B gives the periodic behavior for several additional cycle-paths.

Figure 2.8: A table of S-G numbers for C_4P_m. The column labels $0-11$ represent the least residues of the congruence $n \mod 12$. The row labels represent the length of path in intervals of 12. We see a periodic behavior starting at length 84.

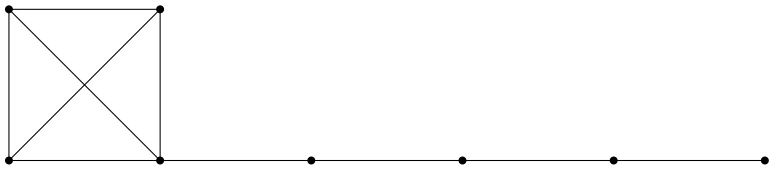
	0	1	2	3	4	5	6	7	8	9	10	11
0	0	5	6	7	5	0	2	6	5	9	2	8
12	0	5	6	9	0	5	3	9	5	10	2	9
24	0	10	6	9	5	0	15	9	5	11	6	9
36	0	10	6	9	5	0	3	12	5	11	2	9
48	0	16	6	9	5	15	3	9	5	15	2	9
60	0	10	6	9	5	10	3	9	5	15	2	9
72	0	10	6	9	5	0	3	9	5	11	2	9
84	0	10	6	9	5	16	3	9	5	15	6	9
96	0	10	6	9	5	16	3	9	5	15	2	9
108	0	10	6	9	5	15	3	9	5	15	6	9

2.7 G-Paths

After noticing eventual periodicity of the S-G numbers for caterpillars, firework graphs, and cycle-paths, a natural question to ask is: Will the S-G numbers of any similarly "shaped" graph eventually become periodic? In this section, we present the progress we have made in this area.

Definition. *We call path appended to one vertex of a graph G, a G-path. We say that a G-path has length L if the appended path has L edges, and write G_L*

We have a G-path consisting of K_4 and a path of length 4, i.e. G_4.



If we make a move on a G-path that deletes any of the edges of the original graph G, we say that we have made a move on G. Conversely, if we make a move that

deletes only edges from the appended path P_L, we say that we have made a move on P_L.

Note that caterpillars, firework graphs, and cycle-paths are all unified under the definition of a G-path. We now provide sufficient conditions for the S-G numbers of a G-path of growing length to stay periodic once an initial period has developed.

Theorem. *Suppose we have a G-path whose S-G numbers have started exhibiting a period of p some time before the G-path has reached length $L-1$. Suppose also that all G'-paths have become periodic for every follower graph G' by the time they reach length $L-1$. Let $d = lcm\{p' : p' is\ the\ period\ of\ the\ S\text{-}G\ numbers\ of\ a\ G'\text{-}path\}$, and let $E = lcm\{p, d\}$. Then if the G-path's S-G numbers stay periodic past length $L+72+E$, they will remain periodic.*

Proof. We want to show that $g(G_{L+72+k}) = g(G_{L+72+k+nE})$ for all $n \in \mathbb{N}, 0 \leq k < E$. We will show that any S-G number obtainable by a move on G_{L+72+k} is obtainable by a move on $G_{L+72+k+nE}$, and vice versa. Consider three positions of where the moves can be made.

Case 1: Moves made on G yield identical S-G numbers for G_{L+72+k} and $G_{L+72+k+nE}$, since $L+72+k = L+72+k+nE \mod d$, and all G'-paths have already become periodic by the time they reach length $L-1$.

Case 2: Suppose we make a move on P_{L+72+k}, leaving G_s and P_t where $s+t = L+72+k-1$ or $s+t = L+72+k-2$ and $0 \leq s < L-1$. On $P_{L+72+k+nE}$ make a move that leaves G_s and P_{t+nE}, where $s+t+nE = L+72+k+nE-1$ or $s+t+nE = L+72+k+nE-2$. Since $t \geq 72$ and $g(P_t) = g(P_{t+nE})$, the two disjoint components resulting from each move nim-sum to the same number.

Conversely, suppose we make a move on $P_{L+72+k+nE}$, leaving G_s and P_t, where $s+t = L+72+k+nE-1$ or $s+t = L+72+k+nE-2$ and $0 \leq s < L-1$. Make

a move on P_{L+72+k} that leaves G_s and P_{t-nE}, where $s+t-nE = L+72+k-1$ or $s+t-nE = L+72+k-2$. Since $t-nE \geq 72$ and $g(P_t) = g(P_{t-nE})$, the two disjoint components resulting from each move nim-sum to the same number.

Case 3: Suppose we make a move on P_{L+72+k}, leaving G_s and P_t where $s+t = L+72+k-1$ or $s+t = L+72+k-2$ and $L-1 \leq s$. Make a move on $P_{L+72+k+nE}$ that leaves G_{s+nE} and P_t where $s+nE+t = L+72+k+nE-1$ or $s+nE+t = L+72+nE-2$. Since $L-1 \leq s$, s is periodic past this point, so $g(G_s) = g(G_{s+nE})$. Thus the two disjoint components resulting from each move nim-sum to the same number.

Conversely, if we make a move on $P_{L+72+k+nE}$, leaving G_s and P_t, where $s+t = L+72+k+nE-1$ or $s+t = L+72+k+nE-2$ and $L-1 \leq s-nE$, make a move on P_{L+72+k} that leaves G_{s-nE} and P_t, where $s-nE+t = L+72+k-1$ or $s-nE+t = L+72+k-2$. Since G is periodic at $s-nE \geq L-1$, $g(G_s) = g(G_{s-nE})$. Thus the two disjoint components resulting from each move nim-sum to the same number.

Since these three cases cover all the possible moves we can make on the two G-paths, we conclude that $g(G_{L+72+k}) = g(G_{L+72+k+nE})$ for all $n \in \mathbb{N}, 0 \leq k < E$. Since $p|E$, the G-path retains its period p.

□

Theorem. *The S-G numbers of a G-path become periodic if and only if they are bounded.*

Proof. ($\Rightarrow$) Immediate.

($\Leftarrow$) Suppose the S-G numbers for a G-path are bounded above by N. Suppose also that $e(G) = m$, and that all G'-paths with $G' \leq G$ and $e(G') < m$, are eventually periodic. Since we know that the empty graph G_0 (with $e(G_0) = 0$) is a subgraph of G, and that a G_0-path is simply a path, we have a valid base case for our inductive

assumption on $e(G)$. Let $d = \text{lcm}\{p : p \text{ is the period of a } G'\text{-path}\}$. Note that since $G_0 \leq G$, and a G_0-path has eventual period 12, $12|d$. And so $g(P_m) = g(P_{m+dt})$ for all $t \in \mathbb{N}$.

Consider any 72 consecutive G-paths, $G_q, G_{q+1}, \ldots, G_{q+71}$; call these G-paths a 72-block. Since there are at most N^{72} ways for the 72 S-G numbers to be distributed over the 72-block, and since there are infinitely many disjoint 72-blocks, there must exist a 72-long sequence of consecutive S-G numbers that is repeated infinitely many times. Call this sequence $a_1, a_2, \ldots, a_{72}$.

Now consider G-paths (both on the same graph G) of length r and s,with $r + 72 \leq s$ and $s = r \mod d$,such that $g(G_r) = a_{72} = g(G_s), g(G_{r-1}) = a_{71} = g(G_{s-1}), \ldots, g(G_{r-71}) = a_1 = g(G_{s-71})$. In other words, we have two sequences of S-G numbers, one of length r and one of length s, that both terminate with "tails" of 72 identical numbers. Choose these G-paths such that all G'-paths have already become periodic by the time they reach length r. Also choose these two G-paths such that for every G-path G_k, with $r < k < s - 71$ there exists G_j with $0 \leq j < r - 71$ and $j = k \mod d$ such that $g(G_k) = g(G_j)$. We are guaranteed to have two G-paths that meet these requirements.

We argue that the S-G number computed for the G-paths of length $r + 1$ and $s + 1$ must be identical to each other, by showing that any S-G number obtained by a move on G_{r+1} can be obtained by a move on G_{s+1}, and vice versa. Consider three cases of where the moves can be made.

Case 1: Suppose we make a move on G. We have $r + 1 = s + 1 \mod d$. Since all G' paths are periodic by length r, we have G_{r+1} and G_{s+1}.

Case 2: Suppose we make a move on P_{r+1}, leaving G_k and P_v where $k+v = r-1$ or $k + v = r$ and $0 \leq k < r - 71$. Make a move on P_{s+1} that leaves G_k and P_{v+dt} for some $t \in \mathbb{N}$. Since $g(P_v) = g(P_{v+dt})$, the two disjoint components resulting from each

move nim-sum to the same number.

Conversely, if we make a move on P_{s+1}, leaving G_k and P_v, where $k+v = s-1$ or $k + v = s$ and $r < k < s - 71$, by our choice of G_{r+1} andG_{s+1} there exists G_j with $0 \leq j \leq r$ and $j \equiv k \mod p$, such that $g(G_j) = g(G_k)$. So, consider the move on G_{r+1} that leaves G_j and P_{v+dt} for some $t \in \mathbb{N}$. Since $g(P_v) = g(P_{v+dt})$, the two disjoint components resulting from each move nim-sum to the same number.

Case 3: Suppose we make a move on P_{r+1}, leaving G_{r-i} and P_v with $r-i+v = r$ or $r - i + v = r - 1$ and $0 \leq i \leq 71$. Make a move on P_{s+1} that leaves G_{s-i} and P_v. Since $g(G_{s-i}) = g(G_{r-i})$, the two disjoint components resulting from each move nim-sum to the same number.

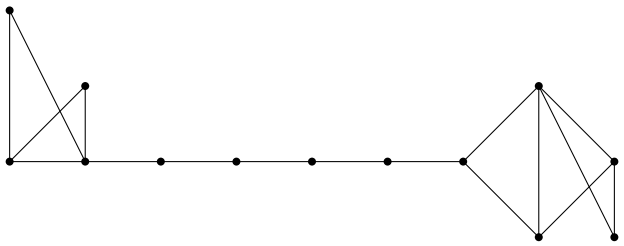
Conversely, if we make a move on P_{s+1}, leaving G_{s-i} and P_v with $0 \leq i \leq 71$, make a move on G_{r+1} that leaves G_{r-i} and P_v. Since $g(G_{r-i}) = g(G_{s-i})$, the two disjoint components resulting from each move nim-sum to the same number.

Since these three cases cover all of the available moves that we can make on G_{r+1} and G_{s+1}, we conclude that $g(G_{r+1}) = g(G_{s+1})$. By inductively extending this argument, we see that $g(G_{r+t}) = g(G_{s+t})$ for all $t \in \mathbb{N}$. And so the G-path's S-G numbers become periodic. □

We now present a result about the periodicity of S-G numbers of what we call a GH-path.

Definition. *Let a GH-path be two graphs, G and H, joined together by a path that meets each graph at exactly one vertex. We write GH_L to be a GH-path where the path has length L.*

For example, below we show a GH path of length 5.



Theorem. *If the S-G numbers of a G-path and an H-path become periodic, then the S-G numbers of the GH-path formed by appending the path to the same vertices as the G and H paths will also become periodic.*

Proof. We proceed by induction on the total number of edges in G and H, $e(G)$ and $e(H)$. For $e(G)+e(H) = 0$, the statement is trivial, since the GH-path is just a path.

So assume for all G-paths and H-paths with periodic S-G numbers and with $e(G) + e(H) < M$, the S-G numbers of the GH-path become periodic, and consider a GH-path with $e(G) + e(H) = M$.

Let $d = \text{lcm}\{p' : p' \text{ is the period of the S-G numbers of a } HG'\text{-path or a } GH'\text{-path}\}$. Let $E = d \cdot p_H \cdot p_G$, where p_H is the periodicity of the S-G numbers of the H-path and p_G is the periodicity of the S-G numbers of the G-path. Let H_s and G_r be the shortest H-path and G-path whose S-G numbers have become periodic. Without loss of generality assume $s > r$.

Consider GH_{2s+E+k} and $GH_{2s+2E+k}$ for $k \in \mathbb{N}$. By a similar argument as was made in proving the preceding two theorems, we can show that any S-G number obtained by a move on GH_{2s+E+k} can be obtained by a move on $GH_{2s+2E+k}$, and vice versa. Consider three cases of where the moves can be made:

Case 1: Moves made on G yield identical S-G numbers for GH_{2s+E+k} and $GH_{2s+2E+k}$ since $2s + E + k \equiv 2s + 2E + k \mod d$ and all the HG'-paths have already become periodic by the time they reach $2s$.

Similarly moves made on H yield identical S-G numbers for GH_{2s+E+k} and $GH_{2s+2E+k}$ since $2s+E+k \equiv 2s+2E+k \mod d$ and all the HG'-paths have already become periodic by the time they reach $2s$.

Case 2: Consider a move made on P_{2s+E+k} leaving G_t and H_u where $t+u = 2s+E+k-1$ or $t+u = 2s+E+k-2$ where $0 \leq t \leq 2s$. A move made on $P_{2s+2E+k}$ leaves G_t and H_{u+r}. Since $g(H_u) = g(H_{u+r})$ the move on P_{2s+E+k} will give the same nim-sum as the move on $P_{2s+2E+k}$

Case 3: Consider a move made on P_{2s+E+k} leaving G_t and H_u where $t+u = 2s+E+k-1$ or $t+u = 2s+E+k-2$ where $0 \leq u \leq 2s$. A move made on $P_{2s+2E+k}$ leaves G_{t+r} and H_u. Since $g(G_t) = g(H_{t+r})$ the move on P_{2s+E+k} will give the same nim-sum as the move on $P_{2s+2E+k}$

Thus, $g(GH_{2s+E+k}) = g(GH_{2s+2E+k})$ for all $k \in \mathbb{N}$. Thus the GH-path's S-G numbers become periodic. □

2.7.1 General G-paths

Next we discuss some observations about the proofs in the pervious and the implications for a general proof for all G-paths. We note that the work in this section is not a proof that all G-paths are periodic. The techniques in this section may lead to a complete proof after further work.

The proofs in the previous section are based on the fact that there are three possible sets of moves we can make: moves on G, moves that give a shorter G-path and a path of length greater than 72, and moves that result in a shorter G-path and a path which is shorter than length 72.

In case one, we are considering moves made on the graph G. To prove this case for a general G-path we would have to know that all G'-paths are periodic. We

can inductively assume this by inducting on the number of edges. Since the empty graph will form a path as a G-path we will have a valid inductive step. Thus a proof for a general G-path will follow as the proofs above.

In case two, we assume we make a move which results in a shorter G-path and a path of length greater than 72. Since the paths are periodic, we need no assumptions for our graph G. So a proof for a general G-path will follow as the above proofs.

Case three is where a general proof fails to follow as the above proofs. In this case we have a shorter G path and a path of length 72. We can use extra assumptions, such as the G-path displaying a period past some length to work around the fact that the paths are not periodic before length 72. The question is, can we get around this without extra assumptions on the shorter G-paths? We begin by noting, as in figure **??**, there are only 14 cases before 72 where the path fails to be periodic.

Figure 2.9: A table of exceptions for the path nimbers

Path Length	Path Length mod12	Nim Value	Periodic Nim Value
0	0	0	4
3	3	3	8
6	6	3	7
9	9	4	8
11	11	6	7
15	3	7	8
18	6	3	7
21	9	4	8
22	10	6	2
28	4	5	1
34	10	6	2
39	3	3	8
57	9	4	8
70	10	6	2

Suppose we have G_s and G_{s+nE}, where E is a multiple of 12, we claim E is the period of G, and $n \in \mathbb{N}$. We want to show if we make a move on G_s that results in

G_t, P_r with $t+r = s-1$ or $t+r = s-2$ with $r < 72$ then we have a move on G_{s+nE} which will result in the same nimber. If P_r is any value other than the exceptions listed, then we are done. We can make the move on G_{s+nE} which results in G_t and P_{r+nE}, since $g(P_r) = g(P_{r+nE})$. If r is one of the exceptions we need to find a move on G_{s+nE} that will result in the same nimber as $g(G_t) + g(P_r)$. We have been unable to find a follower for all G_{s+nE} that will result in this nimber.

Another approach to this case would be to find a contradiction. Assume that $g(G_s) \neq g(G_{s+nE}$ for any n. Assume that $x = g(G_s) < g(G_{s+nE}$, where $x = g(G_t)+g(P_r)$ with $t+r = s-1$ or $t+r = s-2$, $r < 72$ and r is one of our exceptions. Since $g(G_s) < g(G_{s+nE})$ there must be a follower of G_{s+nE} with nimber x. Since there are only 14 exceptions, we can look directly at the different exceptions to find a contradiction. It is unclear if each exception will lead to a contradiction.

Both of these directions seem promising. We are currently gathering data from G-paths that we have studied to determine if there is any patterns to possible moves from G_s and G_{s+nE} that will result in the same nimber. An analysis of this data should give insight to if either of the above methods can prove case 3.

A third approach to a proof is by using the fact that the G-path is periodic if and only if the nimbers are bounded. So we wish to show that no G-paths have unbounded nimbers. We will try to prove this by contradiction: assume that there are unbounded G-paths. Of all these, let us consider one with the smallest (in terms of number of edges) base graph, G. Call this minimal unbounded G-path GP. Since GP has the fewest edges, any move made on GP, i.e. deleting edges from GP will produce a G-path with bounded nimbers. Thus, we can pick a number B such that for any move made on the base graph of GP, the resulting graph has a nimber $\leq B$. We will use this fact in trying to prove that GP cannot in fact have unbounded nimbers.

If GP where unbounded, then it has some useful properties. Assume that

GP is unbounded when we consider the nimbers mod 12. That is, for all $k \in \{0, 1, 2 \ldots, 11\}$ and any $N > 0$, these exists $n \equiv k \mod 12$ such that $g(G_n) > N$. Define a 16-block $\{n_0, n_1, \ldots, n_{15}\}$ to be a list of 16 nimbers. Since GP has unbounded nimbers, after some length L, every GP path has a move that will result in a follower with a nimber in the 16-block. Further we can assume that these moves are made so that the follower is a shorter G-path and a path of length greater than 72. This ensures that the path nimbers are non-exceptions.

The 16 block will give us information about which nimbers must appear in shorter G-paths. Suppose we have a G-path of length $l_0 \equiv 0 \mod 12$, $l_0 > L$. Since $l_0 > L$ there is a move on G_{l_0} that will result in every number in the 16-block. Consider all the moves that result in one of these nimbers, say n_1. The follower that will give us this nimber will be of the form G_k and P_m where $k + m - 1 = l_0$ or $k + m - 2 = l_0$. That is, $g(G_k) \oplus g(P_m) = n_1$. Note, we have $k = l_0 - m - 1 \equiv -m - 1$ mod 12 or $k = l_0 - m - 2 \equiv -m - 2 \mod 12$.

Write $n_1 = 16t + 1, t \in \mathbb{N}$. Then $n_1 \oplus g(P_m) = (16t + 1) \oplus g(P_m) = 16t \oplus (1 + g)P_m))$, since $g(P_m) < 9$ for all $m > 72$. So, we can write $g(G_k) = n_1 \oplus g(P_m) = n_{1 \oplus g(P_m)}$. This gives us information about the possible nimbers that can appear in a G-path of length k. Working mod 12, k depends only on the length of the path, m mod 12, and the number of edges, $e = \{1, 2\}$, that we deleted from GP. We can list these possibilities as a tuple (m, e). There are 24 possible tuples that will result in a nimber of $n_{1 \oplus g(P_m)}$. One of these 24 possibilities must be met.

We can generalize this idea for each nimber in the 16-block; that is we want to consider how we can obtain n_t from G_{l_s}, where $t \in \{n_0, n_1, \ldots, n_{15}\}$, $l_s > L$, $l_s \equiv s$ mod 12. We would have a move that would give us a G-path of length $s - m - e$ mod 12, such that, $g(G_{s-m-e}) = n_{(t \oplus g(P_m))}$. For each t we will have a list of 24 possibilities that must be met. For each t one of these conditions must be met. There

are 24^{16} possible combinations. We need be able to show that none of these conditions can be met at the same time. To do this we will discuss restrictions as to which pairs can appear at the same time.

Consider two G-paths, G_n and G_N, where we consider n and N mod 12. Suppose we know the value of $g(G_n)$. If $N > n$, we can make a move on G_N which has G_n as part of a follower. So we have $g(G_N) \neq g(G_n) \oplus g(P_{N-n-e})$, where $e = \{1, 2\}$ is the number of edges we removed from G_N. If $n > N$, we can make a move on G_n which will give G_N as part of a follower. So we have $g(G_n) \neq g(G_N) \oplus g(P_{n-N-e})$, i.e. $g(G_N) \neq g(G_n) \oplus g(P_{n-N-e})$.

If $g(P_{n-N-e}) = g(P_{N-n-e})$ then we have $g(G_N) \neq g(G_n) \oplus g(P_{n-N-e})$. We can consider a list of restrictions based on all the possible arrangements of n, N, e. For example, consider $n \equiv 2 \mod 12, N \equiv 6 \mod 12$ and $g(G_n) = 3$. If $N > n$ then we have $g(G_N) \neq g(G_n) \oplus g(P_{N-n-2}) \equiv 3 \oplus g(P_{N-n-e}) \mod 12$. Although we do not know $g(P_{N-n-e})$, we have that $N - n - e \equiv 2 \mod 12$ or $N - n - e \equiv 3 \mod 12$, where $e = \{1, 2\}$.

If $n > N$, then we have $g(G_n) \neq g(G_n) \oplus g(P_{n-N-e})$. Rearranging we have that $g(G_N) \neq g(G_n) \oplus g(P_{n-N-e})$. Note that, $n-N-e \equiv 6$ or $n-N-e \equiv 7 \mod 12$. We notice that $g(P_2) = g(P_6) = 2$. Thus it does not matter if $n > N$ or $N > n$, in either case we have, $g(G_N) \neq g(G_n) \oplus g(P_{n-N-e}) = 3 \oplus 2 = 1$. This is the restriction we can place on G_N.

Suppose we know $g(G_n)$. Let $N = n + k \mod 12$. As in the example above we can examine possible restrictions by using the path nimbers. We have, $g(G_N) \neq g(G_n) \oplus x$ for the following pairs

$$(k, x) : (2, 1), (3, 1), (3, 2), (4, 2), (6, 4), (7, 4), (8, 2), (9, 2), (9, 1), (10, 1).$$

Recall for each t value in our 16 block there are 24 possible conditions, one of which must be met. We can use this restriction list to rule out possible configurations from our list of conditions. After running the 24^{16} possible combinations verse the list of restrictions above, we notice that there are still too many combinations to rule out. We would need to consider extra restrictions based on how the nimbers can appear.

2.8 Graph Nim on Graphs

2.8.1 Winning and Losing Complete Graphs

Definition. *A complete graph on n vertices, denoted K_n, is the simple graph in which every pair of distinct vertices is connected by an edge.*

At the outset of this project, it was known that K_1 (trivially), K_3, and K_5 were losing graphs and K_2, K_4, and K_6 were winning. We want to determine if this pattern continues. We first state a proposition on the number of winning complete graphs.

Proposition. *There are infinitely many winning complete graphs.*

Proof. If K_n is losing then K_{n+1} will be winning, since on K_{n+1} Player 1 can delete the n edges incident to one vertex, leaving the Player 2 with K_n. Thus at least half of the complete graphs must be winning. □

Since $K_{2(n+1)}$ is winning if K_{2n+1} is losing, in order to see if the pattern above continues we only must prove K_{2n+1} is losing.

We give the following definitions to allow us to talk about the relationship between two graphs.

Definition. *We call an addition of edges that are all incident to the same vertex an anti-move.*

Definition. *We say that a graph G' is a child of the graph G if G' may be obtained from G in a single move. Equivalently, if G may be obtained from G' in a single anti-move. We call G a parent of G'.*

In order to compute whether K_7 is a win or loss, we need to know the status of all 1043 subgraphs of K_7. Below we describe the program we wrote to find all n-vertex losing graphs, which we were able to run for $1 \leq n \leq 11$.

Algorithm 1. *We use the fact that every parent of a losing graph is a winning graph. We start by creating a list of losing graphs; initially this list consists only of the empty graph on n vertices, letting n = 2k+1 where $k \in Z$. We then iterate through the possible edge-numbers of n-vertex graphs, generating lists of all non-isomorphic graphs with $1, 2, 3, ..., \binom{n}{2}$ edges, which we will call $L_1, L_2, L_3, ..., L_{\binom{n}{2}}$.*

After L_m has been generated, we make anti-moves on each of the losing graphs, which have fewer than m edges, in every way possible that leaves an m-edge graph. Since all m-edge graphs obtained in this manner are the parents of losing graphs, they must be winning; we delete them from L_m. After performing all possible anti-moves on the losing graphs and deleting the resulting graphs from L_m, we will have eliminated all of the winning graphs from L_m. Thus, we add any graphs remaining in L_m to our list of losing graphs, generate L_{m+1}, and repeat the above process. In this fashion, we determine whether every graph on n vertices is winning or losing.

This program computed that K_7, K_9, and K_{11} are losing graphs (and, by implication, that K_8, K_{10}, and K_{12} are winning graphs). Thus, we have extended the pattern of complete graphs alternating between winning and losing from $n = 6$ to $n = 12$. We hope to extend this to find a winning strategy for player 2.

2.8.2 The Sprague-Grundy Approach

Rather than directly computing a list of the losing graphs on n vertices, we now discuss the approach of computing the S-G numbers of all n-vertex graphs. Since those graphs with S-G number 0 are the losing graphs, this approach encompasses the approach of the previous section. However, what is gained in information is lost in efficiency; computing the S-G numbers of every n-vertex graph is a more demanding computational task than determining which graphs are losing. The program that finds the S-G numbers of n-vertex graphs operates similarly to the program that finds the S-G numbers for trees, and so we will provide just a brief description of how it works.

Algorithm 2. *Assuming we have calculated the S-G numbers for all graphs with fewer than m edges, we describe how we will calculate the S-G numbers of the m-edge graphs. For each graph G with m edges, we generate all the children of G. We then find the minimal excluded number of the set of the S-G numbers of these child graphs; this will be the S-G number of G. After computing the S-G numbers of all m-edge graphs in this manner, we move up to $(m+1)$-edge graphs, and continue likewise until the S-G number of every graph on n vertices has been computed.*

Figure 2.10 shows the distribution of the S-G numbers for 7-vertex graphs. A few observations regarding the S-G number distributions can be made immediately. For instance, graphs have the maximum possible S-G number (i.e., $e(G) = g(G)$) roughly half of the time. Also, for graphs of size m, the percentage of graphs with a particular S-G number, s, tends to peak at $s \approx 0.6m$, before bottoming out to nearly zero for $0.6m < s < m$. In order to understand these and other patterns in the S-G number distribution, we conducted a heuristic analysis of the data.

2.8.3 Heuristic Analysis of S-G Number Distribution

Given the distribution of the S-G numbers for n-vertex graphs of size $0, 1, \ldots, m-1$, we want to predict what the distribution should be for graphs of size m. Here we present a heuristic method for making this prediction.

We will try to predict the percentage distribution of the S-G numbers for labeled graphs of order n and size m, given that we know the distributions for graphs of order n and size $< m$. Working with labeled graphs has the advantage of allowing us to consider the deletion of any two distinct subsets of edges to be distinct moves, regardless of graph isomorphism.

Definition. *We define a typical labeled graph on n vertices with m edges, $G_{n,m}$, to be a graph whose degree sequence is the average of the degree sequences of all n-vertex, m-edge labeled graphs when the degrees are ordered from least to greatest. We write the degree sequence of $G_{n,m}$ as $(d_1, d_2, \ldots, d_n)$, where $d_1 \leq d_2 \leq \ldots \leq d_n$.*

In order to predict the S-G number distribution for labeled graphs, we ask ourselves the question: For the typical n-vertex, m-edge labeled graph, $G_{n,m}$, what is the probability that the S-G number will be $0, 1, \ldots, m$?

Once we have calculated the degree sequence of $G_{n,m}$, we will know how many children of size $(m-1), (m-2), \ldots, (m-d_n)$ $G_{n,m}$ has. We make the assumption that each of these children of size $(m-k)$ for $1 \leq k \leq d_n$ will be a random labeled graph of size $(m-k)$. That is, a child graph of size $(m-k)$ will have the S-G number $0, 1, \ldots, (m-k)$ with probability equal to the percentage of labeled $(m-k)$-edge graphs whose S-G numbers are $0, 1, \ldots, m-k$. Since we know the number of children, we can can calculate the probability that $G_{n,m}$ will have S-G number $0, 1, \ldots, m$.

In order to run this heuristic, we need the distribution of S-G numbers for

labeled graphs of size $m-k$. We find this by counting each graph's S-G number N times, where N is the number of unique relabelings of the graph. Figure 2.11 shows the heuristic for labeled 7-vertex graphs:

We need the following definitions to describe our heuristic method.

Definition. *Let $P_e(s)$ to be the probability that a random labeled graph with e edges will have S-G number s. Similarly, let $P_{G_{n,m}}(s)$ to be the probability that our typical graph $G_{n,m}$ has S-G number s.*

Definition. *Let c_k to be the number of children of $G_{n,m}$ with $m-k$ edges.*

Definition. *Let $N_e(s,t_e)$ to be the probability that given t_e random labeled graphs with e edges, none of them will have S-G number s. Note that $N_e(s,t_e) = (1-P_e(s))^t$. Also note that $N_e(s,t_e) = 0$ if $s > e$, since $g(H) \leq e(H)$ for all graphs H.*

Definition. *Lastly, let $N_{G_{n,m}}(s)$ to be the probability that $G_{n,m}$ has no children with S-G number s.*

With this notation, we can describe the exact heuristic method used. Assume that the probability that a child of $G_{n,m}$ with $(m-k)$ edges has S-G number s, is equal to $P_{(m-k)}(s)$. Take $1 \leq j \leq n$ to be the maximum index s.t. $m - d_j \geq s$. Then

$$N_{G_{n,m}}(s) = N_{m-1}(s, c_1) \times \ldots \times N_{m-d_j}(s, c_{d_j}).$$

We know that $P_{G_{n,m}}(s)$ is equal to the probability that $G_{n,m}$ has children with S-G numbers $0, 1, \ldots, (s-1)$ and no children with S-G number s. Thus,

$$P_{G_{n,m}}(s) = \{1 - N_{G_{n,m}}(0)\} \times \ldots \times \{1 - N_{G_{n,m}}(s-1)\} \times N_{G_{n,m}}(s).$$

From the above equations, we see that if we can find the number of children of

$G_{n,m}$, $c_1, \ldots, c_{d_n}$, we will know the probability that $G_{n,m}$ has S-G number 1 through m. Since $G_{n,m}$ is labeled, computing $c_1, \ldots, c_{d_n}$ is a simple task.

This method yields reasonably accurate predictions of the S-G number distributions. Figure 2.12 shows a table of our heuristic predictions of the S-G number distributions compared to the actual distributions for $n = 7$ and $m = 4, 5, \ldots, 18$.

Figure 2.10: The number in column m and row s indicates the percentage of 7-vertex graphs of size m whose S-G number is s. For instance, 50% of size-2 graphs have S-G number 2, 0% have S-G number 1, and 50% have S-G number 0.

$*$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	100	0	50.0	20.0	20.0	19.0	12.2	12.3	8.2	9.2	4.1	5.4	8.4	6.2	0	4.9	4.8	10.0	20.0	0	0	100
1	$*$	100	0	20.0	20.0	23.8	9.8	4.6	17.5	6.1	2.0	3.4	12.2	4.1	0	7.3	0	20.0	0	0	0	0
2	$*$	$*$	50.0	0	30.0	9.5	4.9	13.8	5.2	5.3	4.1	8.1	3.1	5.2	6.2	2.4	0	0	0	0	0	0
3	$*$	$*$	$*$	60.0	0	0	12.2	16.9	4.1	4.6	7.4	4.1	3.1	12.4	6.2	4.9	0	0	0	0	0	0
4	$*$	$*$	$*$	$*$	30.0	4.8	4.9	9.2	15.5	5.3	2.7	2.0	6.9	4.1	6.2	0	23.8	10.0	0	0	0	0
5	$*$	$*$	$*$	$*$	$*$	42.9	0	0	5.2	18.3	8.1	1.4	1.5	2.1	9.2	12.2	4.8	0	0	50.0	0	0
6	$*$	$*$	$*$	$*$	$*$	$*$	56.1	0	1.0	0.8	8.1	19.6	3.8	0	0	9.8	0	0	0	50.0	0	0
7	$*$	$*$	$*$	$*$	$*$	$*$	$*$	43.1	0	0.8	0.7	9.5	9.9	5.2	0	0	0	10.0	40.0	0	0	0
8	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	43.3	0.8	0	0	3.1	12.4	6.2	2.4	4.8	0	0	0	100	0
9	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	48.9	0	0	0	3.1	10.8	17.1	4.8	0	0	0	0	0
10	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	62.8	0	0	0	0	7.3	9.5	10.0	0	0	0	0
11	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	46.6	0	0	0	0	14.3	0	0	0	0	0
12	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	48.1	1.0	0	0	0	20.0	0	0	0	0
13	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	44.3	1.5	0	0	0	0	0	0	0
14	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	53.8	0	0	0	0	0	0	0
15	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	31.7	0	0	0	0	0	0
16	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	33.3	0	0	0	0	0
17	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	20.0	0	0	0	0
18	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	40.0	0	0	0
19	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	0	0	0
20	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	0	0
21	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	0

Figure 2.11: The number in column m and row s indicates the predicted percentage of 7-vertex graphs of size m whose S-G number is s. For example, our heursitic predicts that 6.1% of 7-vertex graphs of size 10 will have S-G number 3.

*	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	100	0	50.0	2.6	12.3	14.1	9.0	6.7	6.9	4.2	2.1	4.2	6.1	3.9	0	5.2	6.2	10.5	2.6	0	0	100
1	*	100	0	7.9	24.6	13.9	4.7	4.3	16.8	3.8	0.3	0.8	9.4	3.0	0	2.0	0	10.5	0	0	0	0
2	*	*	50.0	0	33.3	8.3	0.6	18.4	3.1	1.7	2.2	7.3	3.6	3.0	2.3	1.2	0	0	0	0	0	0
3	*	*	*	89.5	0	0	9.5	15.1	2.8	3.7	6.1	3.1	1.0	9.1	3.5	0.8	0	0	0	0	0	0
4	*	*	*	*	29.8	12.4	3.5	6.7	12.2	2.6	2.0	1.5	6.6	1.0	1.7	0	11.2	1.8	0	0	0	0
5	*	*	*	*	*	51.3	0	0	3.4	22.4	5.1	0.5	0.2	1.9	5.1	14.5	0.5	0	0	50.0	0	0
6	*	*	*	*	*	*	72.6	0	0.6	0.0	5.3	16.0	0.9	0	0	9.4	0	0	0	50.0	0	0
7	*	*	*	*	*	*	*	48.8	0	0.1	0.7	12.7	10.1	1.8	0	0	0	1.8	39.5	0	0	0
8	*	*	*	*	*	*	*	*	54.1	0.9	0	0	2.3	15.1	3.3	0.1	6.2	0	0	0	100	0
9	*	*	*	*	*	*	*	*	*	60.7	0	0	0	4.3	9.6	14.3	1.5	0	0	0	0	0
10	*	*	*	*	*	*	*	*	*	*	76.1	0	0	0	0	3.4	7.2	21.1	0	0	0	0
11	*	*	*	*	*	*	*	*	*	*	*	53.9	0	0	0	0	10.3	0	0	0	0	0
12	*	*	*	*	*	*	*	*	*	*	*	*	59.9	0.2	0	0	0	12.3	0	0	0	0
13	*	*	*	*	*	*	*	*	*	*	*	*	*	56.8	2.2	0	0	0	0	0	0	0
14	*	*	*	*	*	*	*	*	*	*	*	*	*	*	72.4	0	0	0	0	0	0	0
15	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	49.1	0	0	0	0	0	0
16	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	56.8	0	0	0	0	0
17	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	42.1	0	0	0	0
18	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	57.9	0	0	0
19	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0
20	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0
21	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0

Figure 2.12: The number in row m indicates the number of edges of a 7-vertex graph and the column s indicates the S-G number. Here we compare the heuristic and computed results, as indicated by type.

edges	*type*	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
4	*heuristic*	22.5	55.8	5.4	0	16.3	*	*	*	*	*	*	*	*	*	*	*	*	*	*
4	*computed*	12.3	24.6	33.3	0	29.8	*	*	*	*	*	*	*	*	*	*	*	*	*	*
5	*heuristic*	22.7	12.5	4.3	0	10.3	50.2	*	*	*	*	*	*	*	*	*	*	*	*	*
5	*computed*	14.1	13.9	8.3	0	12.4	51.3	*	*	*	*	*	*	*	*	*	*	*	*	*
6	*heuristic*	17.8	5.7	4.0	7.6	3.5	0.8	60.6	*	*	*	*	*	*	*	*	*	*	*	*
6	*computed*	9.0	4.7	0.6	9.5	3.5	0	72.6	*	*	*	*	*	*	*	*	*	*	*	*
7	*heuristic*	10.1	9.5	15.7	32.2	3.8	0	0	28.7	*	*	*	*	*	*	*	*	*	*	*
7	*computed*	6.7	4.3	18.4	15.1	6.7	0	0	48.8	*	*	*	*	*	*	*	*	*	*	*
8	*heuristic*	5.4	10.4	6.0	5.2	8.1	0.9	0	0.3	63.7	*	*	*	*	*	*	*	*	*	*
8	*computed*	6.9	16.8	3.1	2.8	12.2	3.4	0.6	0	54.1	*	*	*	*	*	*	*	*	*	*
9	*heuristic*	7.2	5.2	1.8	2.0	5.5	27.9	0	0	0	50.3	*	*	*	*	*	*	*	*	*
9	*computed*	4.2	3.8	1.7	3.7	2.6	22.4	0.0	0.1	0.9	60.7	*	*	*	*	*	*	*	*	*
10	*heuristic*	6.4	0.9	5.4	5.4	2.2	3.2	5.1	0.1	0	0	71.3	*	*	*	*	*	*	*	*
10	*computed*	2.1	0.3	2.2	6.1	2.0	5.1	5.3	0.7	0	0	76.1	*	*	*	*	*	*	*	*
11	*heuristic*	2.9	0.5	5.8	2.4	1.4	0	11.6	0.6	0	0	0	74.7	*	*	*	*	*	*	*
11	*computed*	4.2	0.8	7.3	3.1	1.5	0.5	16.0	12.7	0	0	0	53.9	*	*	*	*	*	*	*
12	*heuristic*	7.0	9.0	7.6	2.2	7.0	0	1.4	5.2	0.2	0	0	0	60.4	*	*	*	*	*	*
12	*computed*	6.1	9.4	3.6	1.0	6.6	0.2	0.9	10.1	2.3	0	0	0	59.9	*	*	*	*	*	*
13	*heuristic*	2.3	7.9	1.1	2.1	6.1	0.6	0	0.1	11.2	0	0	0	0	68.7	*	*	*	*	*
13	*computed*	3.9	3.0	3.0	9.1	1.0	1.9	0	1.8	15.1	4.3	0	0	0.2	56.8	*	*	*	*	*
14	*heuristic*	0.6	0.6	0.7	2.3	1.8	16.9	0.1	0	2.7	6.2	0	0	0	0	68.1	*	*	*	*
14	*computed*	0	0	2.3	3.5	1.7	5.1	0	0	3.3	9.6	0	0	0	2.2	72.4	*	*	*	*
15	*heuristic*	0.1	0.1	0.2	0	0.6	6.2	0.3	0	0	0.7	0	0	0	0	0	91.9	*	*	*
15	*computed*	5.2	2.0	1.2	0.8	0	14.5	9.4	0	0.1	14.3	3.4	0	0	0	0	49.1	*	*	*
16	*heuristic*	0.4	0.5	0.5	0	1.9	0.1	2.9	0.3	0	0	12.8	0.1	0	0	0	0	80.5	*	*
16	*computed*	6.2	0	0	0	11.2	0.5	0	0	6.2	1.5	7.2	10.3	0	0	0	0	56.8	*	*
17	*heuristic*	0	1.2	0.8	0	0.7	0	0	6.4	0	0	2.1	3.0	0	0	0	0	0	85.6	*
17	*computed*	10.5	10.5	0	0	1.8	0	0	1.8	0	0	21.1	0	12.3	0	0	0	0	42.1	*
18	*heuristic*	0	1.3	6.2	1.6	0	0	0	39.6	0	0	0	0	0.8	0	0	0	0	0	50.5
18	*computed*	2.6	0	0	0	0	0	0	39.5	0	0	0	0	0	0	0	0	0	0	57.9

2.9 Conclusion

We have shown a periodic behavior for certain paths, caterpillars, star-paths, and cycle-paths. We want to expand this research to all graphs where we fix a main graph, G, and attach a path of increasing length to one vertex. We have also shown that K_{2n+1} is a losing graph for $0 \leq$ n ≤ 5. We want to determine if K_{13} will be winning or losing. As future work we will explore the connection between star-paths and equivalence classes of partitions.

Chapter 3

The Evolution of Strings

3.1 Introduction

In 2010, Ewens and Wilf considered the question of if there was enough time for evolution to take place [20]. In their model they considered a string of length l over an alphabet of size k. Assuming there exists a correct word, we want to know the expected time it would take to converge from a string of random letters to the correct word. The typical model would assume at every stage of evolution each of the the L genes get replaced by a random letter from the alphabet, leading to an order of k^l. Ewens and Wilf considered a model where at a given stage in evolution, if a letter was correct then it was not reselected. Using this model they found the order for evolution to be $k \log l$. In particular, the mean number of rounds necessary to guess the correct word is $\frac{\log(l)}{\log\left(\frac{k}{k-1}\right)}$.

We have studied a modification of this model. Consider a string of length l and an alphabet of size k. We want to study the convergence to the correct string with the following rules. If a letter in the string is incorrect, we replace that letter with a random letter from our alphabet with probability 1. If a letter in the string is correct,

we replace that letter with a random letter from our alphabet with probability q and we do not replace that letter with a random letter with a probability $p = 1 - q$. For our model, we assume that the probability p is a function of the number of correct letters currently in the string, say c.

Let d the number of correct letters we have after one step in time. Let i be the number of correct letters that become incorrect. Let j be the number of incorrect letters that become correct at a given step. Then we have that $d = c - i + j$, $\max\{0, c - d\} \leq i \leq \min\{c, l - c\}$.

We want to examine the transition matrix for this markov chain. The entries in the transition matrix can be computed as follows. For a given entry we have c correct letters and want to have d correct letters at the next time step. Consider making i correct letters incorrect. We can choose these letters in $\binom{c}{i}$ ways. These letters must get reselected at random, but cannot be the correct letters; we can do this in $\left(q\left(1 - \frac{1}{k}\right)\right)^i$ ways. There are $c - i$ remaining correct letters with $\left(p + q\left(\frac{1}{k}\right)\right)^{c-i}$ possible arrangements.

From the $l - c$ incorrect letters we need to choose j that become correct, $\binom{l-c}{j}$. There are $\left(\frac{1}{k}\right)^j$ ways to make these correct. The remaining elements stay incorrect, $\left(1 - \frac{1}{k}\right)^{l-c-j}$. So an entry in our transition matrix will be

$$A_{c,d} = \sum_{i=\max\{0,c-d\}}^{\min\{c,l-d\}} \binom{c}{i} \left(q\left(1 - \frac{1}{k}\right)\right)^i \left(p + q\left(\frac{1}{k}\right)\right)^{c-i} \binom{l-c}{j} \left(\frac{1}{k}\right)^j \left(1 - \frac{1}{k}\right)^{l-c-j}.$$

We will focus on the transition matrix for different choices for p. For our first model, let $p = \frac{c}{l}$. Letting $k = \frac{1}{x}$, we have

$$A_{c,d} = \sum_{i=\max\{0,c-d\}}^{\min\{c,l-d\}} \binom{c}{i} \left(\left(1 - \frac{c}{l}\right)(x-1)\right)^i \left(1 + \left(1 - \frac{c}{l}\right)(x-1)\right)^{c-i} \binom{l-c}{j} x^j (1-x)^{l-c-j}.$$

The complete transition matrix, A, is

$$A = \begin{bmatrix} 0.3164 & 0.2373 & 0.0791 & 0.0049 & 0 \\ 0.4219 & 0.4219 & 0.3164 & 0.0659 & 0 \\ 0.2109 & 0.2637 & 0.4043 & 0.2999 & 0 \\ 0.0469 & 0.0703 & 0.1758 & 0.4951 & 0 \\ 0.0039 & 0.0068 & 0.0244 & 0.1341 & 1.0000 \end{bmatrix},$$

with eigenvalues

$$\begin{bmatrix} 1 \\ 0.9645 \\ 0.4974 \\ 0.0198 \\ 0.156 \end{bmatrix}$$

and associated eigenvectors

$$\begin{bmatrix} 0.0000 & 0.1626 & 0.5457 & -0.7132 & -0.8277 \\ 0.0000 & 0.3343 & 0.5652 & 1.0000 & 0.2356 \\ 0.0000 & 0.3178 & -0.3845 & -0.3289 & 1.0000 \\ 0.0000 & 0.1853 & -1.0000 & 0.0440 & -0.4528 \\ 1.0000 & -1.0000 & 0.2735 & -0.0020 & 0.0449 \end{bmatrix}.$$

3.2 Computing Eigenvalues

In this section, we are concerned with how to compute the eigenvalues for our transition matrix. As l grows, it is not realistic to compute eigenvalues using typical methods such as the QZ algorithm, which is a generalization of the QR algorithm for dense matrices. This is the algorithm that software packages such as matlab use.

The QZ algorithm requires $O(l^3)$ operations with $O(l^2)$ memory locations. Instead we will use other numerical approaches to compute the eigenvalues.

3.2.1 Power Method

The first numerical tool we will use to find the eigenvalue of a matrix A is the power method. This method begins with a guess for the initial eigenvector, v_0. We then iterate through the following sequence

$$\begin{aligned} v_1 &= Av_0 \\ v_2 = Av_1 &= A^2v_0 \\ v_3 = Av_2 &= A^3v_0 \\ &\dots \\ v_i = Av_{i-1} &= A^iv_0 \end{aligned}$$

This method will find the largest eigenvalue with a convergence rate of $\left|\frac{\lambda_2}{\lambda_1}\right|$, where λ_1 is the dominant eigenvalue and λ_2 is the second largest eigenvalue. We can perform a slight modification of this algorithm which takes the vector v_i and normalizes according to the Euclidean norm before the next iterative step. That is $v_i := \frac{v_i}{||v_i||}$.

Note that v_i represents the probability of being in a given state after n steps. We notice that the columns of A will sum to 1. It is easy to see that one of the eigenvalues of A is 1. We will use the following parts of the Perron-Frobenius Theorem to conclude that 1 is the largest eigenvalue.

Theorem. *(Perron-Frobenius) Let $A = a_{ij}$ be an $n \times n$ non-negative matrix. Then the following statements hold.*

1. *There is a positive real number r, called the Perron root or the Perron-Frobenius*

eigenvalue, such that r is an eigenvalue of A and any other eigenvalue λ is strictly smaller than r in absolute value, $|\lambda| < r$.

2. *There exists an eigenvector $v = (v_1, \ldots, v_n)$ of A with eigenvalue r such that all components of v are positive.*

3. *There are no other positive (moreover non-negative) eigenvectors except positive multiples of v.*

Since A contains only non-negative entries and the eigenvector associate with the eigenvalue of 1 is $[0, 0, \ldots, 0, 1]^T$, we can conclude that 1 must be the largest eigenvalue of A.

We now turn our attention to finding the second largest eigenvalue.

3.2.2 Numeric Methods

The deflation method can be used in conjunction with the power method to find subsequent eigenvalues of a matrix. Below we will discuss two different implementations of the deflation method. Before using any version of the deflation method, we first use the power method to find the largest eigenvalue, λ_1 and it's associated eigenvector, v_1. For A, we have $\lambda_1 = 1$ and $v_1 = [0, 0, \ldots, 0, 1]^T$.

The first version of the deflation method is quite simple [11]. We start by forming the matrix defined by $A_1 = A - \lambda_1 * v_1 * v_1^T$. We then perform the power method on this new matrix, which will give λ_2 and v_2', where λ_2 is the second largest eigenvalue of A. The associated eigenvector, v_2, for A is constructed in the following way. $v_2(i) = \frac{v_2(i)'}{\lambda_2}$ for $1 \leq i \leq n-1$ and $v_2(i) = -1, i = n$.

Recall for a string of length 5 over an alphabet with 4 letters. We have

$$A = \begin{bmatrix} 0.3164 & 0.2373 & 0.0791 & 0.0049 & 0 \\ 0.4219 & 0.4219 & 0.3164 & 0.0659 & 0 \\ 0.2109 & 0.2637 & 0.4043 & 0.2999 & 0 \\ 0.0469 & 0.0703 & 0.1758 & 0.4951 & 0 \\ 0.0039 & 0.0068 & 0.0244 & 0.1341 & 1.0000 \end{bmatrix}.$$

We have $\lambda_1 = 1$ and $v_1 = [0, 0, 0, 0, 1]^T$.

The new matrix we form is

$$A_1 = A - \lambda_1 * v_1 * v_1^T = \begin{bmatrix} 0.3164 & 0.2373 & 0.0791 & 0.0049 & 0 \\ 0.4219 & 0.4219 & 0.3164 & 0.0659 & 0 \\ 0.2109 & 0.2637 & 0.4043 & 0.2999 & 0 \\ 0.0469 & 0.0703 & 0.1758 & 0.4951 & 0 \\ 0.0039 & 0.0068 & 0.0244 & 0.1341 & 0 \end{bmatrix}.$$

Applying the power method to A_1, we find an eigenvalue of 0.9645 with associated eigenvector of

$$v_2' = \begin{bmatrix} 0.1568 \\ 0.3224 \\ 0.3065 \\ 0.1788 \\ 0.0355 \end{bmatrix}.$$

We then construct v_2 for A, giving us the eigenvector

$$v_2 = \begin{bmatrix} 0.1626 \\ 0.3343 \\ 0.3178 \\ 0.1853 \\ -1 \end{bmatrix}.$$

The next version of the deflation method, [19] uses a Householder matrix H_1 such that $H_1 x_1 = k_1 e_1$, where $k_1 \neq 0, e_1 = [1, 0, \ldots, 0]$. We define H_1 using the following algorithm.

$$\begin{aligned} k_1 &= -\text{sgn}(e_1^T v_1)||_1||_2 \\ \beta &= \left[||v_1||_2(||v_1||_2 + ||e_1^T v_1||)\right]^{-1} \\ u &= v_1 - k_1 e_1 \\ H_1 &= I - \beta u u^T \end{aligned}$$

Then we define

$$A_2 = H_1 A H_1^{-1} = \begin{bmatrix} \lambda_1 & b_1^T \\ 0 & B_2 \end{bmatrix},$$

where B_2 is an $(n-1) \times (n-1)$ matrix with the same eigenvalues as A. We can find the largest eigenvalue of B_2, say y_2, where $B_2 y_2 = \lambda_2 y_2$. Suppose we want to find the corresponding eigenvector to P, say v_2. We have

$$v_2 = \begin{bmatrix} \alpha \\ y \end{bmatrix},$$

where

$$\begin{aligned} \lambda_1\alpha + b_1^T y &= \lambda_2\alpha \\ B_2 y &= \lambda_2 y. \end{aligned}$$

Let $y = y_2$ and $\alpha = \frac{b_1^T y}{\lambda_2 - \lambda_1}$. Note, this method makes it easy to compute the eigenvector for A.

Using the length 5, alphabet size 4 as above we have

$$k_1 = -2.2361, \beta = 0.1382$$

and

$$u = \begin{bmatrix} 3.2361 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \end{bmatrix}, H_1 = \begin{bmatrix} -0.4472 & -0.4472 & -0.4472 & -0.4472 & -0.4472 \\ -0.4472 & 0.8618 & -0.1382 & -0.1382 & -0.1382 \\ -0.4472 & -0.1382 & 0.8618 & -0.1382 & -0.1382 \\ -0.4472 & -0.1382 & -0.1382 & 0.8618 & -0.1382 \\ -0.4472 & -0.1382 & -0.1382 & -0.1382 & 0.8618 \end{bmatrix}.$$

Then we have

$$B_2 = \begin{bmatrix} 0.1062 & 0.0496 & -0.1780 & -0.2424 \\ 0.0197 & 0.2092 & 0.1278 & -0.1706 \\ -0.0690 & 0.0854 & 0.4276 & -0.0659 \\ -0.1718 & -0.1054 & 0.0272 & 0.8947 \end{bmatrix}.$$

Performing the power method on this we get the eigenvalue of 0.9645, eigenvector of $[-0.7811, -0.7359, -0.2715, 2.8885]^T$ and $\alpha = 3.3929 \times 10^{-15}$. After normalizing

we have that

$$v_2 = \begin{bmatrix} 0.1626 \\ 0.3343 \\ 0.3178 \\ 0.1853 \\ -1 \end{bmatrix}.$$

Finally, we can exploit the structure of A to find the second largest eigenvalue. In particular we have

$$A = \begin{bmatrix} Q & 0 \\ R & 1 \end{bmatrix}.$$

We see that the eigenvalues of Q will be eigenvalues of A. Using the power method on Q, we will have the correct eigenvalue but an $n-1$ eigenvector, say v_2'. To get the corresponding eigenvector for A, we normalize v_2' and append -1 as the entry for the last row.

For the above example we have

$$Q = \begin{bmatrix} 0.3164 & 0.2373 & 0.0791 & 0.0049 \\ 0.4219 & 0.4219 & 0.3164 & 0.0659 \\ 0.2109 & 0.2637 & 0.4043 & 0.2999 \\ 0.0469 & 0.0703 & 0.1758 & 0.4951 \\ 0.0039 & 0.0068 & 0.0244 & 0.1341 \end{bmatrix}.$$

Using the power method on Q we find an eigenvalue of 0.9645 and eigenvector $v_2' = [0.1626, 0.3343, 0.3178, 0.1853]^T$. After normalizing this vector and appending a

-1 we have

$$v_2 = \begin{bmatrix} 0.1626 \\ 0.3343 \\ 0.3178 \\ 0.1853 \\ -1 \end{bmatrix}.$$

3.2.3 Second Eigenvalue Computations

Using the Q matrix we can study the second eigenvalue. In Figure 3.1 we see the second largest eigenvalues for $l = 10$ to $l = 100$. Using our numeric methods at $l = 71$ we find the second largest eigenvalue is slightly larger than 1. By Perron-Frobenius we know that that $\lambda_2 < 1$. This happens due to roundoff errors in our numeric methods.

Recall we have $Qv_2' = \lambda_2 v_2'$. To find the second eigenvalue of A, we have

$$A \begin{pmatrix} v_2' \\ -m \end{pmatrix} = \lambda_2 \begin{pmatrix} v_2' \\ -m \end{pmatrix},$$

where $\sum v_2'(i) = -m$. Considering the last row, we have

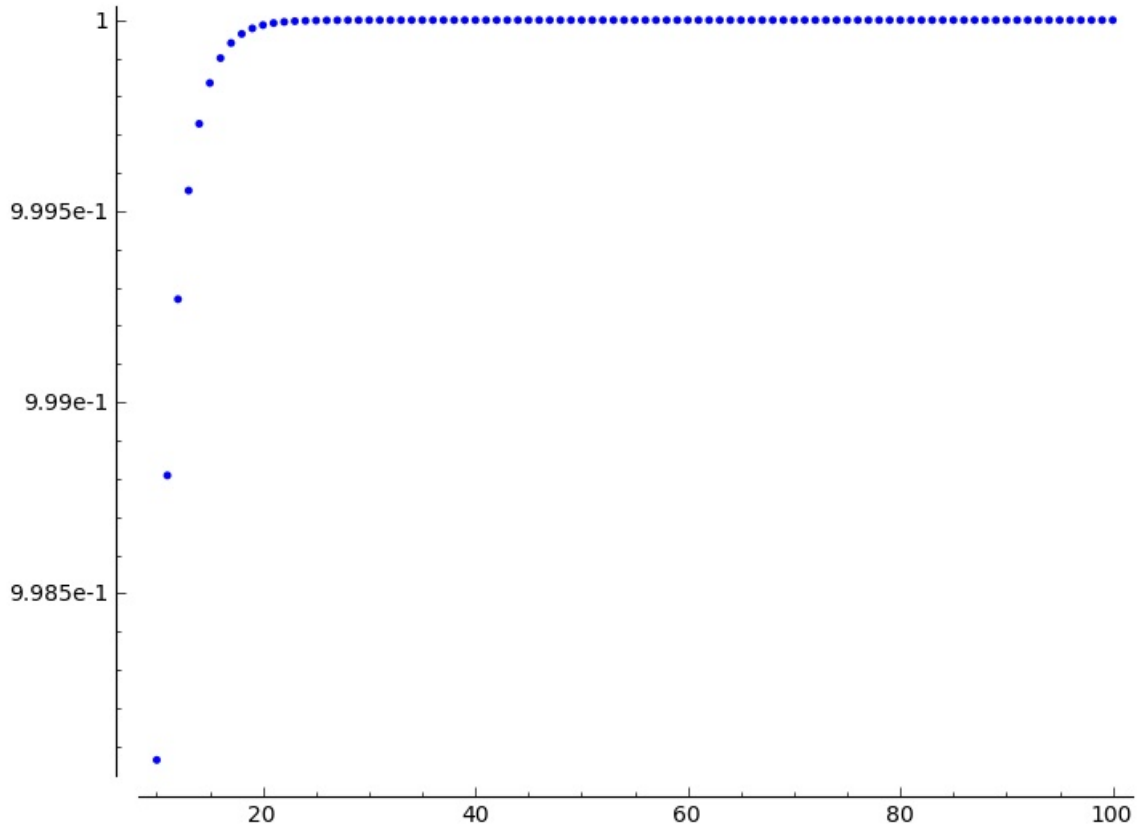
$$\sum A_{nj} v_2'(j) - m = \lambda_2(-m).$$

Thus

$$(1 - \lambda_2)m = \sum A_{nj} v_2'(j).$$

To gain more precision we can compute $\epsilon = (1 - \lambda_2)$. Figure 3.2 shows a plot of the ϵ values for $l = 5$ to $l = 20$. Figure 3.3 shows a plot of $\log(\epsilon)$ for $l = 5$ to $l = 100$.

Figure 3.1: A plot of the length of the string vs. the second largest eigenvalue for our model for an alphabet of size 4 when $p = \frac{c}{l}$.



3.2.4 Absorption

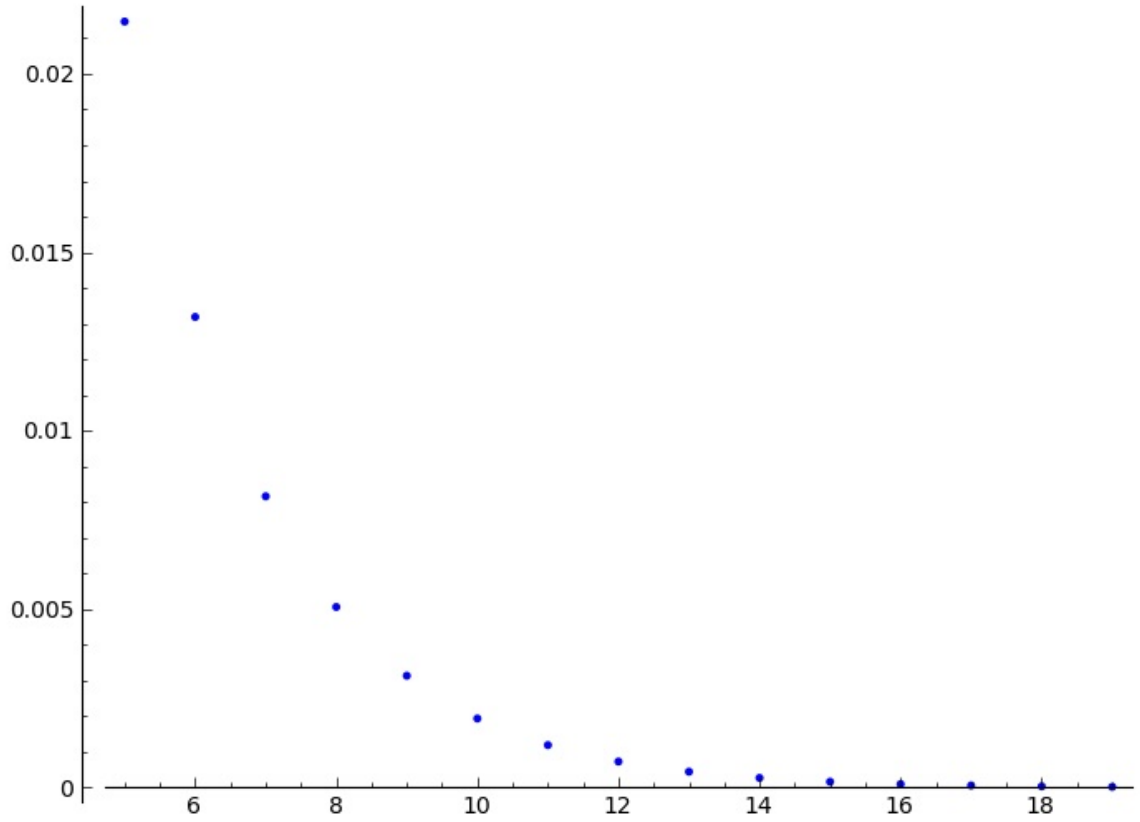
Given c correct letters, we are interested in the time it takes to get to l correct letters. Recall we have

$$A = \begin{bmatrix} Q & 0 \\ R & 1 \end{bmatrix}.$$

The matrix Q represents the transient states of the transition matrix. Consider the matrix

$$A^n = \begin{bmatrix} Q^n & 0 \\ R' & 1 \end{bmatrix}.$$

Figure 3.2: A plot of the length of the string vs. $\epsilon = (1 - \lambda_2)$ for our model when $p = \frac{c}{l}$ and $k = 4$.



Q^n tells us the probability of being in that transient state after n steps and R' tells us the probability that a state is absorbed by time n. Note that $Q^n \to 0$ as $n \to \infty$. Then $I - Q$ has an inverse, say N. Then $n_{ij} \in N$ tells us the expected number of times the markov chain is in state s_j given we started in s_i. Let t_i be the number of steps before the chain is absorbed. Let t be the column vector whose i^{th} entry is t_i. Then $t = cN$, where $c = [1, 1, \ldots, 1]$ [12].

Example. *For example when* $l = 10, k = 4$ *we have*

$$t = \begin{bmatrix} 522.2640 \\ 522.1904 \\ 521.9491 \\ 521.4529 \\ 520.4831 \\ 518.5066 \\ 514.1180 \\ 503.2075 \\ 472.3329 \\ 372.2371 \end{bmatrix}.$$

Figure 3.4 shows the number of correct letters we start with vs. the absorption rate for our model with $l = 100, k = 4, p = \frac{c}{100}$.

This tells us that if we begin with a string with no correct letter the expected number of steps to the correct string is 522. Figure 3.5 shows the number of steps it will take to arrive in the absorbing state having started with no letters correct for an alphabet of size 4 for lengths ranging from 5 to 20.

We can compare our estimate for the number of steps to absorption vs. the estimate using Wilf's model. Recall the mean number of rounds necessary to guess the correct word is $\frac{\log(l)}{\log\left(\frac{k}{k-1}\right)}$. In figure 3.6 we see that Wilf's model shows a much lower mean time than the absorption rate of our model. Recall that Wilf's model assume that once a letter is correct it cannot become incorrect letter.

Although the model when $p = \frac{c}{l}$ has a better absorption rate than the traditional model of k^l, we see it has a much slower absorption rate of the Wilf model. For

example considering $l = 20$ and $k = 4$. The traditional model has a absorption rate of $4^{20} = 1.099 \times 10^{12}$. Our model with $p = \frac{c}{l}$ has a absorption rate of ≈ 77106.46. The Wilf model has a absorption rate of 10.4.

Our model depends on the probability that a correct letter is reselected or not. Using $p = \frac{c}{l}$ considers a model where the probability of reselected a correct letter decreases as the number of correct letters increases. To get a quicker absorption rate we want to consider changing p to a function of c that increases the probability of keep a correct letter. In the next section we discuss models using other probability functions.

3.2.5 Other Models

3.2.5.1 $p = \left(\frac{c}{l}\right)^{\left(\frac{1}{\alpha}\right)}$

We first consider $p = \sqrt[3]{\frac{c}{l}}$. For this model we see that as c increases the probability of reselecting a correct letter is smaller than using $p = \frac{c}{l}$.

Figure 3.7 shows a plot of the ϵ values for $l = 5$ to $l = 20$. Figure 3.5 shows the number of steps it will take to arrive in the absorbing state having started with no letters correct for an alphabet of size 4 for lengths ranging from 5 to 20. In figure 3.9 we see that Wilf's model shows a much lower mean time than the absorption rate of our model.

Our model with $p = \sqrt[3]{\frac{c}{l}}$ has a absorption rate of approximately 35.249, when $l = 20, k = 4$ and our starting string has no correct letters.

We now consider the absorption rate for $p = \left(\frac{c}{l}\right)^{\left(\frac{1}{\alpha}\right)}$ for varying α. In Figure 3.10 we see that as α increases the absorption rate decreases.

3.3 Conclusion

We have begun to analyze our model for various probabilities. It appears that although our model does not behave as well as the Wilf model, it behaves much better than reselecting new letters at each step in the Markov chain. Further analysis needs to be done to study how this behavior changes for larger strings. We note that this analysis is not easy since our matrix is ill-conditioned as l increases. We hope to be able to use our model to study a biological model such as protein mutations.

Figure 3.3: A plot of the length of the string vs. $\log(\epsilon)$ for our model when $p = \frac{c}{l}$ and $k = 4$.

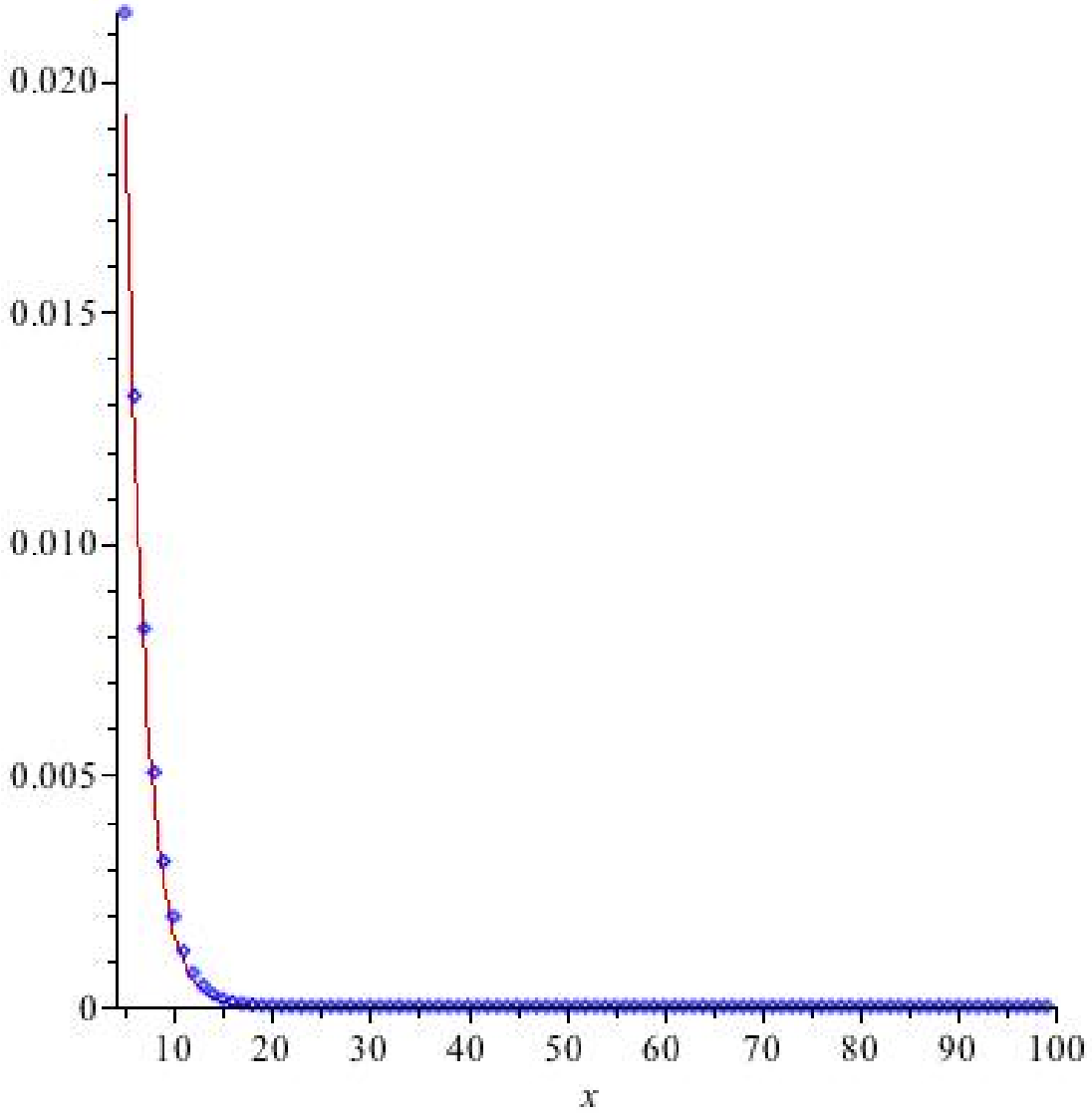


Figure 3.4: A plot of the number of correct letters vs. the absorption rate for our model with $l = 100$, an alphabet of size 4, and $p = \frac{c}{l}$.

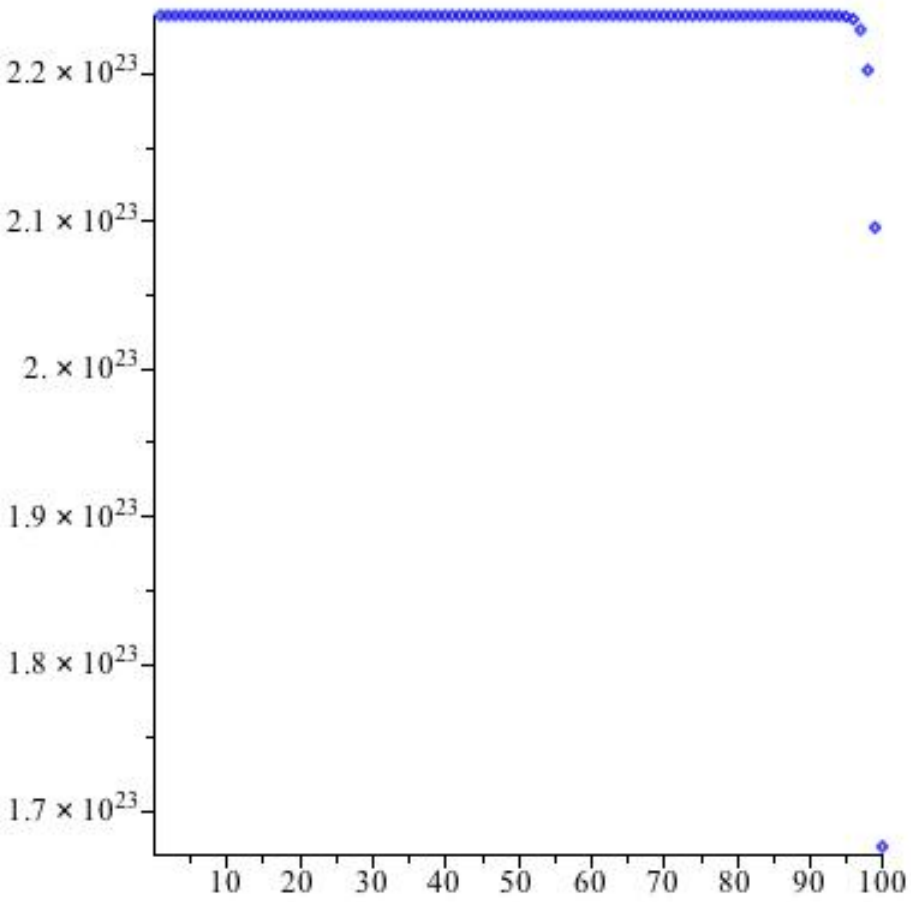


Figure 3.5: A plot of the length of the string vs. the number of steps it will take for a string with zero correct letters to reach a string with l correct letters for an alphabet of size 4 for our model with $p = \frac{c}{l}$.

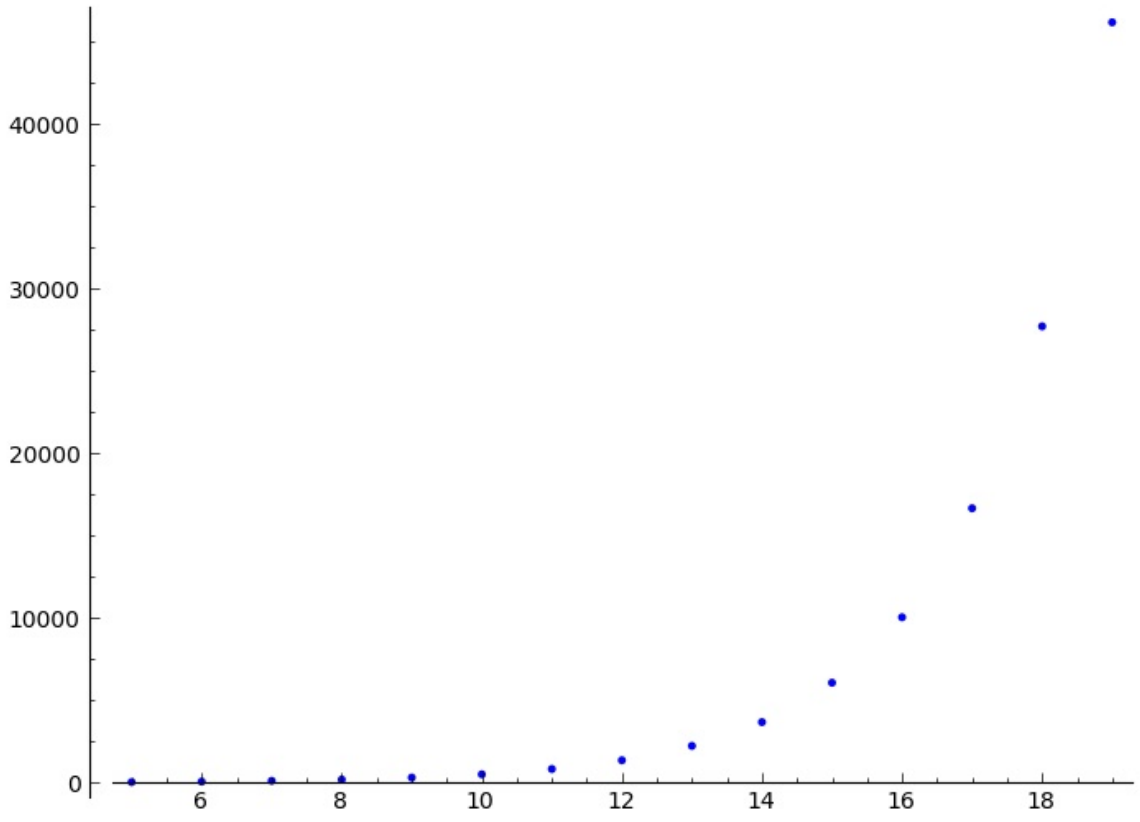


Figure 3.6: A plot of the length of the string vs. the number of steps it will take for a string with zero correct letters to reach a string with l correct letters for an alphabet of size 4. our model with $p = \frac{c}{l}$ is in red and the Wilf model is in blue.

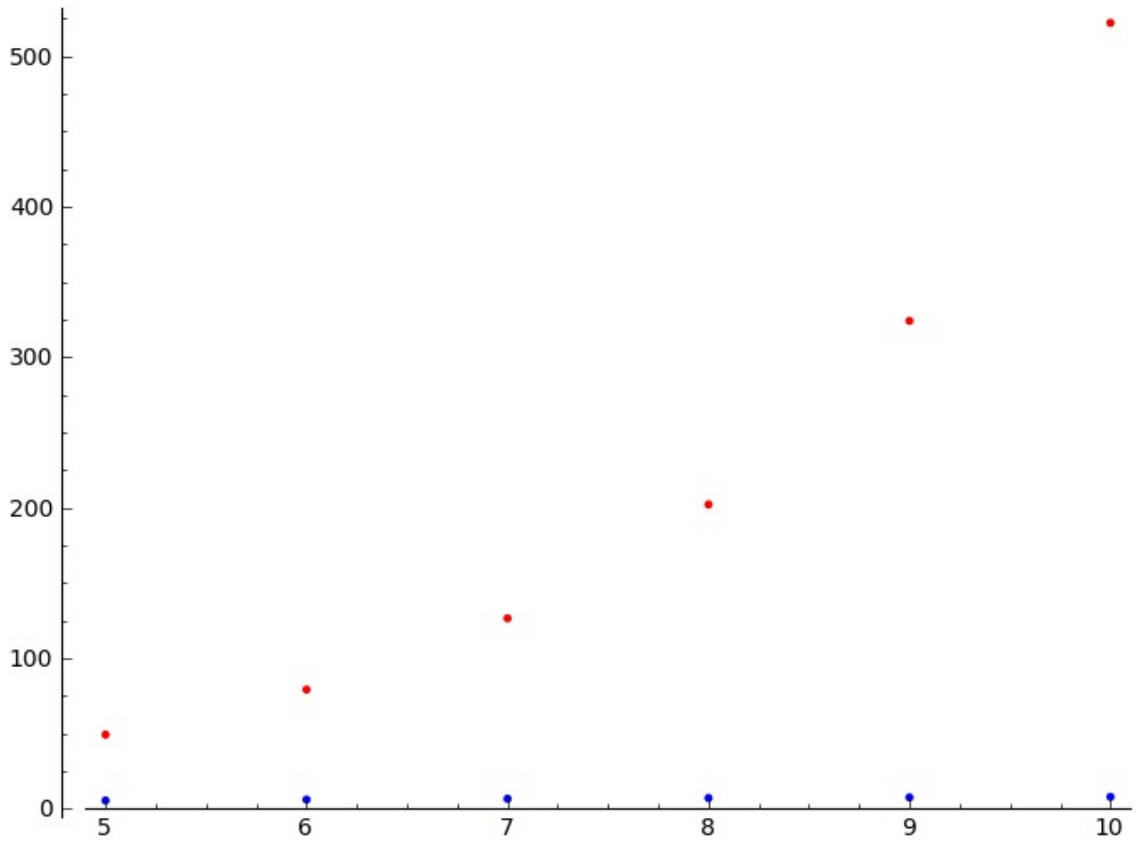


Figure 3.7: A plot of the length of the string vs. $\epsilon = (1 - \lambda_2)$ for our model when $p = \sqrt[3]{\frac{c}{l}}$ and $k = 4$.

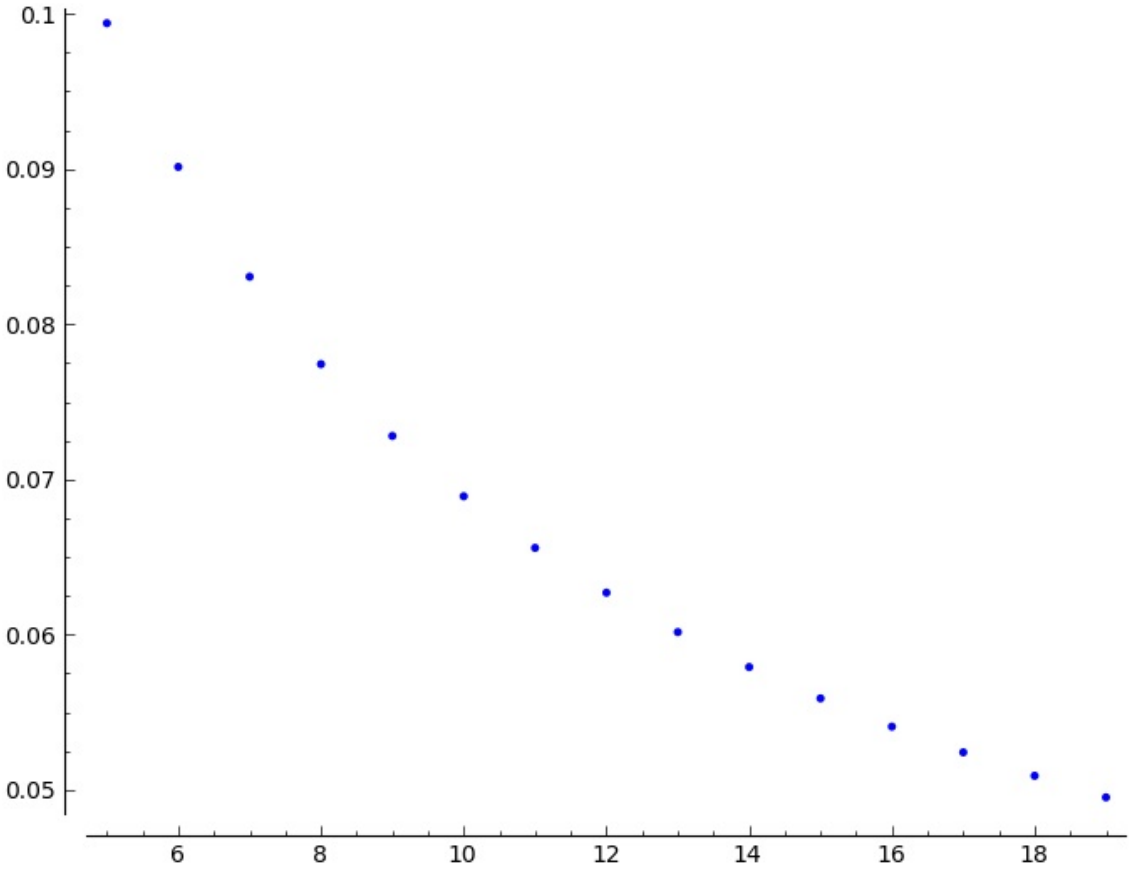


Figure 3.8: A plot of the length of the string vs. the number of steps it will take for a string with zero correct letters to reach a string with l correct letters for an alphabet of size 4 for our model with $p = \sqrt[3]{\frac{c}{l}}$.

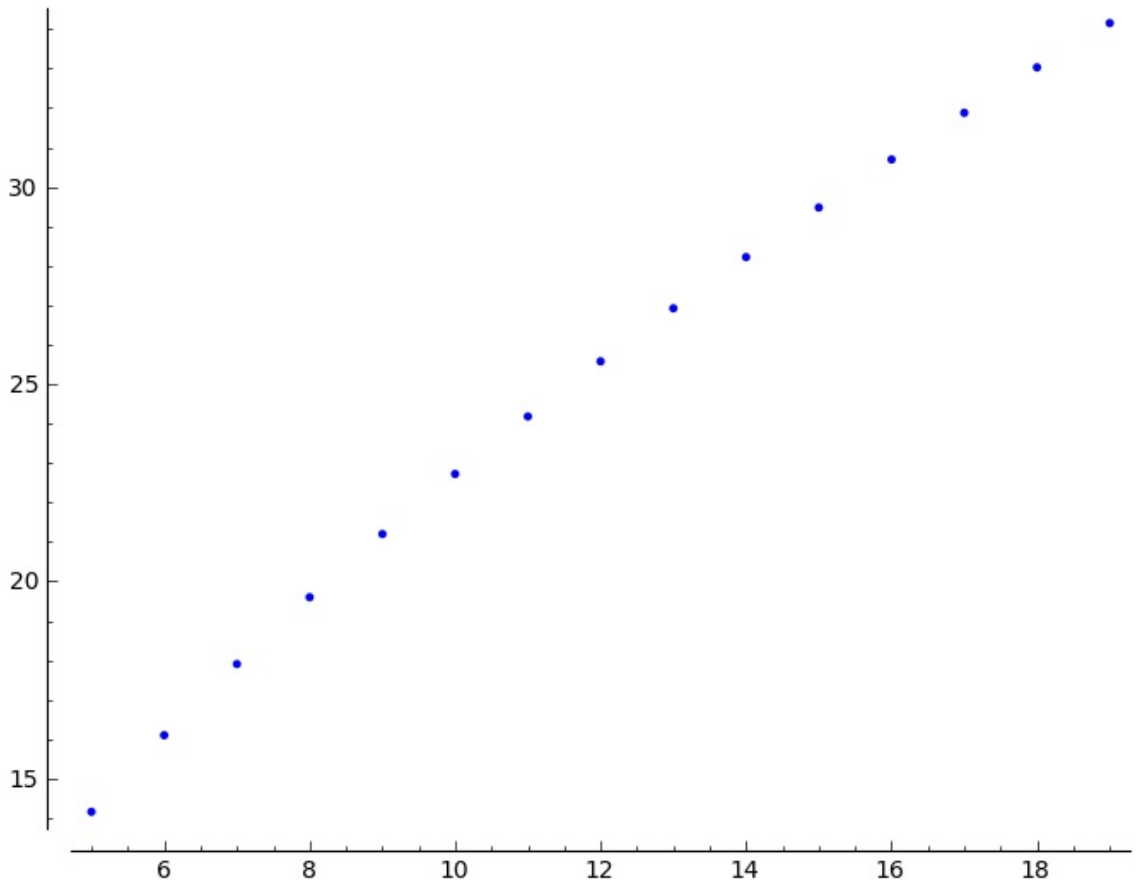


Figure 3.9: A plot of the length of the string vs. the number of steps it will take for a string with zero correct letters to reach a string with l correct letters for an alphabet of size 4. Our model with $p = \sqrt[3]{\frac{c}{l}}$ is in red and the Wilf model is in blue.

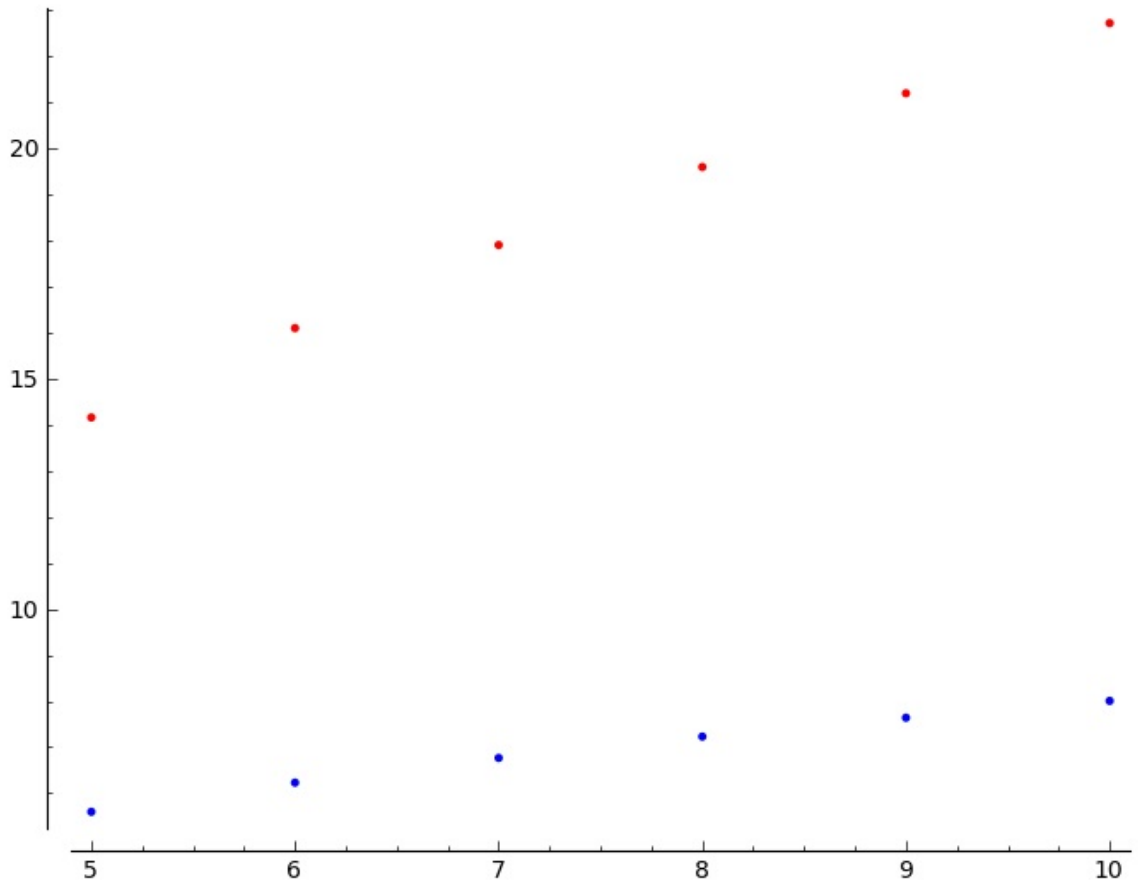
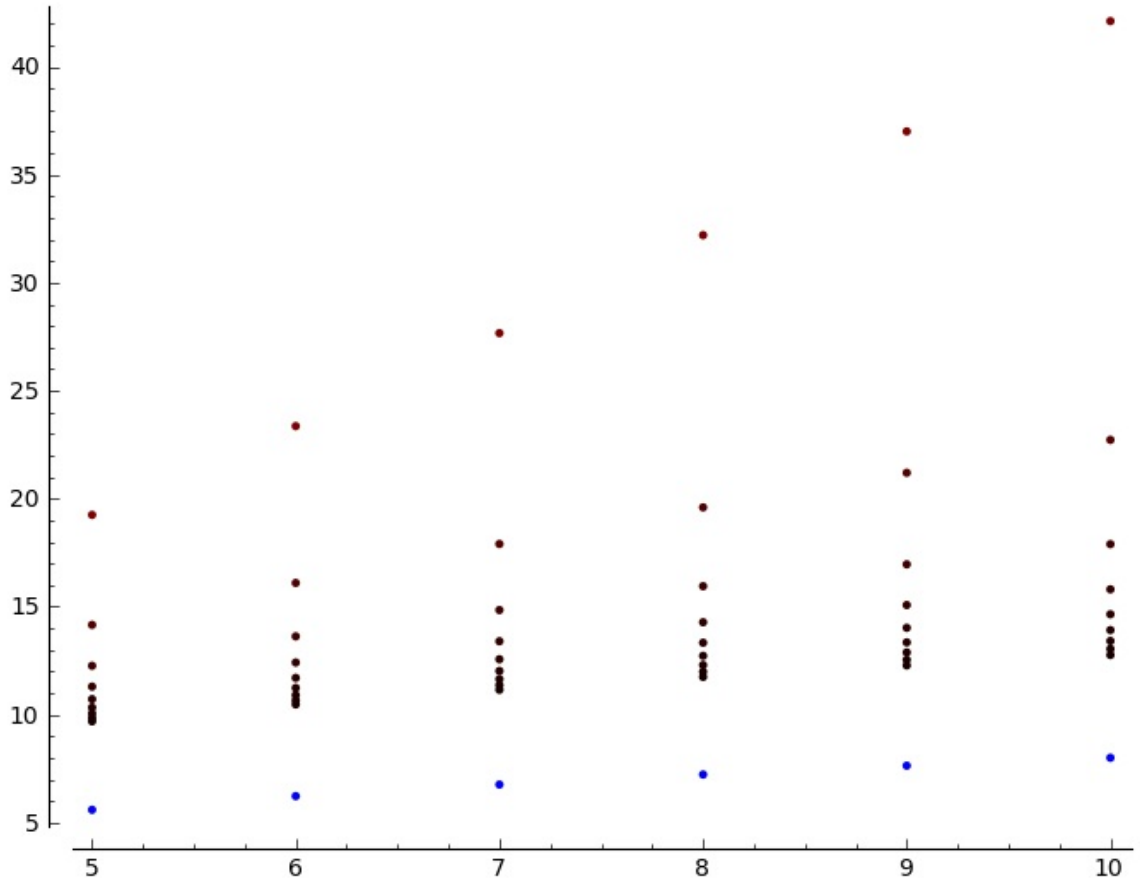


Figure 3.10: A plot of the length of the string vs. the number of steps it will take for a string with zero correct letters to reach a string with l correct letters for an alphabet of size 4. The plot shows our model with $p = \sqrt[\alpha]{\frac{c}{l}}$ for $2 \leq \alpha \leq 10$ and the Wilf model.



Appendices

Appendix A Firework Graphs

Below we give the periods associated with the nimbers of firework graphs. Each star is viewed as $S_{a_1,a_2,a_3,\ldots,a_n}$ where $a_1, a_2, \ldots a_{n-1}$ are fixed lengths a_n varies.

A.1 3-Firework Graphs

$S_{1,1,m}$ (Period beginning at 156):
$13, 22, 11, 25, 19, 22, 14, 21, 22, 19, 11, 21$

$S_{1,2,m}$ (Period beginning at 120):
$3, 9, 18, 6, 13, 18, 11, 10, 17, 4, 13, 17$

$S_{1,3,m}$ (Period beginning at 156):
$16, 11, 19, 28, 12, 30, 19, 8, 5, 14, 15, 19$

$S_{1,4,m}$ (Period beginning at 108):
$11, 14, 4, 13, 14, 1, 8, 13, 7, 19, 22, 26$

$S_{1,5,m}$ (Period beginning at 228):
$27, 17, 20, 23, 6, 7, 24, 1, 10, 20, 5, 30$

$S_{1,6,m}$ (Period beginning at 360):
$26, 4, 19, 2, 31, 19, 11, 21, 16, 31, 4, 21$

$S_{1,7,m}$ (Period beginning at 276):
$23, 29, 18, 19, 8, 36, 14, 30, 17, 16, 7, 38$

$S_{1,8,m}$ (Period beginning at 192):
$29, 20, 27, 4, 1, 14, 31, 13, 24, 7, 2, 37$

$S_{1,9,m}$ (Period beginning at 240):
$11, 16, 12, 22, 11, 28, 25, 16, 15, 21, 0, 31$

$S_{1,10,m}$ (Period beginning at 144):
$7, 4, 18, 3, 16, 1, 4, 7, 17, 1, 19, 10$

$S_{1,11,m}$ (Period beginning at 276):
$16, 1, 22, 13, 1, 34, 19, 2, 21, 14, 2, 32$

$S_{1,12,m}$ (Period beginning at 168):
$21, 14, 30, 1, 4, 11, 21, 12, 19, 32, 7, 8$

$S_{1,13,m}$ (Period beginning at 120):
$14, 18, 25, 11, 7, 2, 13, 4, 11, 7, 21, 1$

$S_{1,14,m}$ (Period beginning at 360):
$36, 24, 11, 20, 30, 13, 14, 11, 33, 34, 22, 14$

$S_{1,15,m}$ (Period beginning at 240):

$1, 4, 7, 6, 8, 21, 2, 7, 4, 5, 8, 18$

$S_{1,16,m}$ (Period beginning at 348):

$29, 7, 12, 0, 16, 10, 7, 22, 28, 3, 32, 9$

$S_{1,17,m}$ (Period beginning at 276):

$16, 32, 4, 35, 21, 36, 1, 35, 10, 19, 25, 35$

$S_{1,18,m}$ (Period beginning at 228):

$22, 4, 32, 2, 37, 13, 22, 15, 32, 1, 19, 14$

$S_{1,19,m}$ (Period beginning at 312):

$0, 29, 30, 37, 24, 20, 3, 18, 9, 24, 27, 23$

$S_{2,2,m}$ (Period beginning at 168):

$9, 0, 3, 12, 5, 15, 10, 3, 0, 5, 6, 12$

$S_{2,3,m}$ (Period beginning at 96):

$23, 18, 17, 4, 24, 11, 20, 17, 18, 7, 24, 8$

$S_{2,4,m}$ (Period beginning at 156):

$3, 2, 32, 16, 7, 18, 0, 9, 25, 14, 4, 17$

$S_{2,5,m}$ (Period beginning at 252):

$19, 12, 5, 10, 19, 5, 16, 15, 6, 9, 16, 25$

$S_{2,6,m}$ (Period beginning at 384):

$31, 13, 2, 19, 0, 21, 7, 14, 1, 16, 19, 22$

$S_{2,7,m}$ (Period beginning at 252):

$22, 2, 20, 29, 23, 15, 20, 9, 18, 14, 32, 29$

$S_{2,8,m}$ (Period beginning at 144):

$2, 3, 21, 30, 6, 25, 1, 0, 3, 15, 5, 26$

$S_{2,9,m}$ (Period beginning at 252):

$20, 0, 18, 11, 21, 5, 22, 29, 16, 0, 22, 6$

$S_{2,10,m}$ (Period beginning at 312):

$5, 30, 40, 24, 5, 31, 6, 34, 0, 20, 2, 23$

$S_{2,11,m}$ (Period beginning at 252):

$1, 19, 34, 4, 16, 11, 2, 21, 33, 15, 19, 0$

$S_{2,12,m}$ (Period beginning at 312):

$6, 22, 28, 12, 3, 18, 4, 13, 28, 18, 0, 17$

$S_{2,13,m}$ (Period beginning at 432):

$27, 13, 29, 3, 32, 9, 27, 15, 29, 0, 32, 10$

$S_{2,14,m}$ (Period beginning at 132):

$9, 0, 10, 19, 0, 16, 10, 3, 0, 12, 3, 19$

$S_{2,15,m}$ (Period (60) beginning at 240):

$28, 44, 26, 22, 25, 14, 16, 44, 25, 15, 31, 13$

$28, 41, 26, 22, 25, 14, 16, 44, 25, 15, 31, 13$

$28, 41, 26, 22, 25, 14, 16, 44, 25, 15, 31, 13$

$28, 44, 26, 22, 25, 14, 16, 47, 25, 15, 31, 13$

$28, 44, 26, 22, 25, 14, 16, 47, 25, 15, 31, 13$

$S_{2,16,m}$ (Period beginning at 204):

$6, 32, 0, 27, 3, 12, 13, 27, 3, 32, 0, 15$

$S_{2,17,m}$ (Period beginning at 432):

$42, 0, 5, 39, 16, 9, 19, 3, 6, 28, 19, 33$

$S_{2,18,m}$ (Period beginning at 240):

$12, 21, 2, 36, 40, 34, 7, 21, 5, 39, 40, 37$

$S_{2,19,m}$ (Period beginning at 372):

$17, 2, 37, 32, 35, 7, 32, 24, 38, 18, 30, 4$

$S_{3,3,m}$ (Period beginning at 180):

$18, 14, 4, 11, 18, 7, 24, 14, 7, 30, 17, 24$

$S_{3,4,m}$ (Period beginning at 132):

$24, 21, 10, 15, 19, 12, 15, 21, 9, 8, 21, 15$

$S_{3,5,m}$ (Period beginning at 528):

$29, 20, 11, 7, 20, 17, 14, 20, 24, 32, 23, 18$

$S_{3,6,m}$ (Period beginning at 360):

$32, 6, 22, 1, 33, 11, 34, 5, 30, 2, 13, 8$

$S_{3,7,m}$ (Period beginning at 348):

$2, 7, 4, 26, 14, 19, 8, 4, 7, 19, 1, 16$

$S_{3,8,m}$ (Period beginning at 552):

$42, 32, 25, 16, 2, 8, 28, 32, 26, 22, 37, 11$

$S_{3,9,m}$ (Period beginning at 144):

$13, 1, 27, 8, 1, 20, 29, 2, 23, 8, 2, 7$

$S_{3,10,m}$ (Period beginning at 168):

$19, 15, 16, 1, 19, 15, 16, 12, 1, 10, 16, 12$

$S_{3,11,m}$ (Period beginning at 228):

$15, 2, 8, 16, 14, 13, 31, 1, 11, 2, 13, 14$

$S_{3,12,m}$ (Period beginning at 156):

$7, 8, 20, 15, 6, 19, 24, 11, 23, 22, 5, 10$

$S_{3,13,m}$ (Period beginning at 540):

$8, 4, 23, 8, 13, 1, 11, 8, 17, 15, 24, 2$

$S_{3,14,m}$ (Period beginning at 180):

$23, 32, 13, 4, 1, 10, 38, 26, 14, 7, 29, 8$

$S_{3,15,m}$ (Period beginning at 252):

$11, 27, 30, 34, 24, 6, 8, 5, 29, 30, 18, 4$

$S_{3,16,m}$ (Period beginning at 528):

$38, 12, 29, 2, 7, 20, 5, 15, 27, 9, 24, 20$

$S_{3,17,m}$ (Period beginning at 180):

$8, 1, 10, 7, 12, 5, 40, 20, 8, 25, 37, 6$

$S_{3,18,m}$ (Period beginning at 312):

$18, 32, 35, 12, 1, 10, 16, 35, 23, 26, 6, 8$

$S_{3,19,m}$ (Period beginning at 372):

$2, 38, 21, 22, 7, 19, 1, 37, 32, 13, 4, 16$

$S_{4,4,m}$ (Period beginning at 192):

$22, 8, 16, 29, 23, 13, 21, 11, 19, 32, 7, 14$

$S_{4,5,m}$ (Period beginning at 312):

$20, 11, 19, 8, 1, 14, 22, 12, 16, 11, 2, 1$

$S_{4,6,m}$ (Period beginning at 192):

$14, 7, 13, 21, 3, 19, 16, 4, 14, 19, 0, 16$

$S_{4,7,m}$ (Period beginning at 144):

$24, 13, 12, 24, 25, 8, 27, 14, 15, 27, 30, 23$

$S_{4,8,m}$ (Period beginning at 204):

$0, 16, 15, 22, 5, 16, 2, 19, 22, 21, 6, 19$

$S_{4,9,m}$ (Period beginning at 180):

$26, 32, 24, 4, 26, 19, 25, 4, 30, 11, 25, 1$

$S_{4,10,m}$ (Period (60) beginning at 204):

$7, 37, 1, 27, 30, 32, 29, 37, 2, 7, 27, 32$

$7, 37, 1, 27, 30, 32, 29, 37, 2, 7, 32, 35$

$7, 37, 1, 27, 30, 32, 29, 37, 2, 7, 32, 35$

$7, 37, 1, 27, 35, 32, 29, 37, 2, 7, 32, 35$

$7, 37, 1, 27, 30, 32, 29, 37, 2, 7, 32, 35$

$S_{4,11,m}$ (Period beginning at 180):

$19, 8, 21, 11, 8, 1, 4, 14, 21, 8, 16, 2$

$S_{4,12,m}$ (Period beginning at 156):

$5, 17, 2, 18, 0, 12, 6, 18, 1, 2, 3, 15$

$S_{4,13,m}$ (Period (60) beginning at 228):

$31, 7, 35, 16, 27, 10, 7, 12, 16, 40, 35, 8$

$31, 7, 35, 16, 27, 10, 7, 12, 16, 40, 35, 8$

$31, 7, 35, 16, 27, 10, 7, 12, 16, 40, 35, 8$

$31, 7, 35, 16, 27, 10, 7, 12, 16, 40, 34, 8$

$31, 7, 35, 16, 27, 10, 7, 12, 16, 40, 35, 8$

$S_{4,14,m}$ (Period beginning at 132):

$2, 19, 4, 8, 7, 21, 1, 8, 7, 11, 1, 21$

$S_{4,15,m}$ (Period beginning at 168):

$1, 8, 13, 14, 4, 24, 2, 11, 14, 13, 4, 33$

$1, 8, 13, 14, 4, 24, 2, 11, 14, 13, 4, 33$

$1, 8, 13, 14, 4, 24, 2, 11, 14, 13, 4, 33$

$1, 8, 13, 14, 4, 24, 2, 11, 14, 13, 4, 32$

$1, 8, 13, 14, 4, 24, 2, 11, 14, 13, 4, 33$

$S_{4,16,m}$ (Period (60) beginning at 312):

$23, 25, 2, 33, 1, 28, 15, 32, 1, 28, 2, 36$

$23, 25, 2, 33, 1, 28, 15, 32, 1, 28, 2, 44$

$23, 25, 2, 33, 1, 28, 15, 32, 1, 28, 2, 36$

$23, 25, 2, 33, 1, 28, 15, 32, 1, 28, 2, 36$

$23, 25, 2, 33, 1, 28, 15, 32, 1, 28, 2, 36$

$S_{4,17,m}$ (Period beginning at 312):

$19, 7, 16, 1, 22, 2, 21, 26, 25, 2, 16, 1$

$S_{4,18,m}$ (Period beginning at 216):

$40, 14, 13, 21, 3, 33, 8, 4, 14, 19, 1, 16$

$S_{4,19,m}$ (Period beginning at 312):

$25, 13, 20, 7, 32, 9, 16, 32, 19, 4, 21, 10$

A.2 4-Firework Graphs

$S_{1,1,1,m}$ (Period beginning at 168):

$3, 10, 18, 12, 15, 14, 18, 8, 20, 25, 12, 9$

$S_{1,1,2,m}$ (Period beginning at 144):

$10, 1, 8, 7, 14, 4, 19, 18, 7, 16, 21, 7$

$S_{1,1,3,m}$ (Period beginning at 144):

$1, 2, 16, 5, 19, 7, 2, 1, 11, 6, 16, 4$

$S_{1,1,4,m}$ (Period beginning at 204):

$9, 3, 32, 9, 20, 6, 10, 32, 15, 10, 18, 5$

$S_{1,1,5,m}$ (Period beginning at 156):

$18, 10, 21, 27, 14, 15, 20, 2, 7, 25, 13, 7$

$S_{1,2,6,m}$ (Period beginning at 168):

$15, 23, 37, 20, 5, 19, 12, 35, 38, 22, 3, 17$

$S_{1,1,6,m}$ (Period beginning at 204):

$10, 15, 35, 18, 15, 24, 9, 12, 29, 17, 7, 24$

$S_{1,2,7,m}$ (Period beginning at 228):

$29, 8, 36, 14, 13, 8, 3, 29, 23, 6, 33, 24$

$S_{1,1,7,m}$ (Period beginning at 168):

$20, 9, 33, 15, 24, 9, 23, 10, 30, 23, 27, 17$

$S_{1,2,8,m}$ (Period beginning at 120):

$8, 18, 10, 24, 3, 6, 27, 21, 32, 24, 21, 5$

$S_{1,1,8,m}$ (Period beginning at 96):

$9, 2, 7, 14, 13, 7, 16, 17, 4, 19, 14, 4$

$S_{1,2,9,m}$ (Period beginning at 252):

$0, 10, 5, 6, 32, 10, 21, 12, 6, 15, 30, 9$

$S_{1,1,9,m}$ (Period beginning at 132):

$2, 1, 4, 13, 19, 4, 1, 2, 18, 25, 16, 7$

$S_{1,3,3,m}$ (Period beginning at 204):

$34, 19, 10, 33, 21, 5, 19, 3, 9, 5, 19, 6$

$S_{1,2,2,m}$ (Period beginning at 132):

$19, 15, 5, 6, 15, 5, 9, 12, 21, 22, 25, 6$

$S_{1,3,4,m}$ (Period beginning at 232):

$27, 1, 8, 6, 22, 4, 24, 2, 11, 16, 32, 7$

$S_{1,2,3,m}$ (Period beginning at 180):

$41, 3, 21, 32, 28, 6, 10, 32, 37, 21, 27, 32$

$S_{1,3,5,m}$ (Period beginning at 144):

$0, 3, 6, 5, 12, 6, 3, 0, 5, 6, 15, 5$

$S_{1,2,4,m}$ (Period beginning at 132):

$24, 18, 11, 17, 29, 7, 27, 21, 12, 32, 30, 20$

$S_{1,3,6,m}$ (Period beginning at 264):

$21, 12, 32, 22, 1, 32, 27, 15, 32, 21, 38, 10$

$S_{1,2,5,m}$ (Period beginning at 204):

$22, 11, 24, 35, 15, 14, 17, 8, 18, 24, 17, 6$

$S_{1,3,7,m}$ (Period beginning at 240):

$13, 1, 11, 32, 31, 4, 21, 2, 8, 16, 21, 7$

$S_{1,3,8,m}$ (Period beginning at 204):

$14, 27, 21, 28, 7, 22, 13, 28, 11, 27, 40, 21$

$S_{1,3,9,m}$ (Period beginning at 228):

$18, 24, 6, 27, 38, 6, 17, 0, 38, 6, 37, 23$

$S_{2,2,2,m}$ (Period beginning at 108):

$18, 9, 4, 7, 13, 4, 16, 10, 7, 13, 14, 7$

$S_{2,2,3,m}$ (Period beginning at 228):

$12, 22, 11, 35, 19, 18, 37, 38, 8, 19, 24, 17$

$S_{2,2,4,m}$ (Period beginning at 132):

$28, 11, 6, 27, 22, 6, 28, 18, 19, 24, 21, 5$

$S_{2,2,5,m}$ (Period beginning at 156):

$21, 10, 9, 16, 14, 15, 19, 9, 10, 16, 13, 7$

$S_{2,2,6,m}$ (Period beginning at 156):

$32, 29, 19, 10, 20, 8, 13, 17, 16, 9, 22, 11$

$S_{2,2,7,m}$ (Period beginning at 168):

$12, 17, 6, 18, 0, 20, 24, 18, 9, 20, 3, 13$

$S_{2,2,8,m}$ (Period beginning at 108):

$26, 10, 21, 22, 19, 7, 21, 9, 26, 21, 28, 4$

$S_{2,2,9,m}$ Period beginning at 144:

$14, 7, 16, 1, 26, 16, 7, 22, 11, 26, 32, 19$

$S_{2,3,3,m}$ (Period beginning at 180):

$8, 1, 13, 7, 31, 22, 7, 2, 13, 4, 11, 7$

$S_{2,3,4,m}$ (Period beginning at 204):

$21, 10, 5, 22, 14, 5, 18, 25, 10, 21, 36, 28$

$S_{2,3,5,m}$ (Period beginning at 204):

$32, 2, 17, 4, 24, 21, 11, 10, 22, 32, 14, 4$

$S_{2,3,6,m}$ (Period beginning at 360):

$37, 48, 3, 35, 17, 13, 23, 37, 17, 35, 18, 27$

$S_{2,3,7,m}$ (Period beginning at 168):

$24, 36, 9, 12, 23, 30, 8, 3, 10, 34, 12, 6$

$S_{2,3,8,m}$ (Period beginning at 216):

$18, 37, 4, 29, 37, 18, 35, 38, 15, 30, 1, 15$

$S_{2,3,9,m}$ (Period beginning at 204):

$33, 8, 11, 36, 14, 17, 34, 1, 2, 23, 14, 18$

Appendix B Cycle Nim

Below we give the periods associated with the nimbers of cycle-paths.

C_5P_m (Period beginning at 144):

$12, 3, 10, 5, 17, 3, 3, 0, 9, 6, 12, 15$

C_6P_m (Period beginning at 144):

$14, 3, 9, 16, 10, 7, 13, 16, 10, 6, 13, 4$

C_7P_m (Period (60) beginning at 156):

$16, 16, 3, 25, 16, 10, 16, 16, 0, 16, 16, 13$

$21, 21, 3, 25, 16, 10, 21, 21, 0, 16, 16, 13$

$16, 16, 3, 25, 16, 10, 16, 16, 0, 16, 16, 13$

$16, 16, 3, 25, 16, 10, 21, 21, 0, 16, 16, 13$

$21, 21, 3, 25, 16, 10, 16, 16, 0, 16, 16, 13$

C_8P_m (Period beginning at 228):

$0, 5, 16, 7, 9, 21, 3, 6, 5, 5, 31, 21$

C_9P_m (Period beginning at 132):

$11, 2, 24, 5, 5, 16, 8, 1, 11, 20, 5, 18$

Appendix C Sage and Matlab Programs

C.1 Unimodal Code

Below is the code used for testing $\mathcal{P}_n$ is unimodal.

```
def P(howfar):
    #This function returns a matrix where each entry (n,k)
    # = #partitions of n into k summands
    M = matrix(ZZ,howfar,howfar)
    for n in range(howfar):
        for k in range(n+1):
            if (k==0 or n==k):
                M[n,k]=1
            else:
                M[n,k]=(M[n-1,k-1]+M[n-k-1,k])
    return M

def mode(M):
    #This function takes a matrix as input and returns a list of
    # the positions of each row's maximal element. mode(P(a))
    #effectively calculates the rank of the mode of P_n for n<=a
    List = []
    for n in range(M.nrows()):
        maxval = 0
        for k in range(n+1):
            if (M[n,k] > maxval):
                max = k+1
                maxval = M[n,k]
        List.append(max)
    return List

def unimodal(M):
    #This function ensures that each row of the inputted matrix is
    #unimodal.
    for n in range(M.nrows()):
        increasing = 1;
        #This means that values should start out increasing.
        value = 0
        for k in range(n+1):
            if (increasing == 1):
                if (M[n,k] < value):
```

```
                    increasing = 0
                    #This means all values should decrease.
                value = M[n,k]
            else:
                if (M[n,k] > value):
                    #If a row is not unimodal, the function terminates
                    #and displays the n-value which is not unimodal.
                    print 'n=',n+1,'is not unimodal.'
                    return 0
                value = M[n,k]
    return 1

def improvedunimodal(input,start,finish):
    #This function encompasses the utilities of P, unimodal,
    #and mode to find the values of P(n,k) from
    #n=start to n=finish.  It determines if each 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 P
    #It deletes all P(n,k) values which become unnecessary for
    #future computations.
    L = input
    modes = []
    for row in range(start-1,finish):
        #First, delete unnecessary entries
        for i in range(floor((row)/2)):
            del(L[i][0])
        #print L
        #Then, we create the next row of entries
        if (row!=0):
            #The first column is always a 1
            L[0].append(1)
            #The 2 through floor((row-1)/2)+1 columns are calculated
            #using both summands
            for column in range(1,floor((row-1)/2)+1):
                L[column].append(L[column-1][column-1]+L[column][0])
            #The floor((row-1)/2)+2 through n-1 columns are calculated
            #with P(n-1,k-1) only
            for column in range(floor((row-1)/2)+1,row):
                L[column].append(L[column-1][row-column])
            #The last column is always a 1
            if (mod(row,2)==0):
                L[floor((row-1)/2)+1][len(L[floor((row-1)/2)+1])-1]=
```

```
                L[floor((row-1)/2)][row-floor((row-1)/2)-2]
        L.append([1])
        #Then we find our mode (if it is unimodal).
        value = 0
        mode = 0
        increasing = 1
        center = (row-1)/2
        maximum = ceil(row/2)+1
        for column in range(row+1):
            index = maximum-floor(abs(center-column))-1
            if (increasing == 1):
                if (L[column][index] < value):
                    increasing = 0
                    #This means all values should decrease.
                if L[column][index]>value:
                    mode = column
                value = L[column][index]
            else:
                if (L[column][index] > value):
                    #If a row is not unimodal, the function immediately terminates
                    #and displays the n-value which is not unimodal.
                    print 'n=',n+1,'is not unimodal.'
                    return 0
                value = L[column][index]
        modes.append(mode+1)
    print L
    return [L,modes]

#import pickle
file=open('logConcave.txt','w')
num=10
pnk=P(num)
print "hi"
for n in range(num):
    for k in range(1,n-1):
        logConc=2*RDF(log(pnk[n][k]))-RDF(log(pnk[n][k-1]))
        -RDF(log(pnk[n][k+1]))
        if logConc<0:
            string='n=', n, 'k=', k, 'concavity is ', logConc, '-'
            s=str(string)+'\n'
            file.write(s)
        else:
```

```
            string='n=', n, 'k=', k, 'concavity is ', logConc
            s=str(string)+'\n'
            file.write(s)
#pickle.dump(unimodalList, open("unimodal.txt","w"))
#filename2='logConcave.txt'
#save(unimodalList, filename2)
```

C.2 Nim Code

Below are a list of sage functions used for Graph Nim computations

```
import pickle

def binConvert(x):
    "Converts a given number to a list of 1's and 0's,
    its binary form. e.g. 6"
    "gets returned as [1,1,0]"
    i=0
    rem=x
    t=2
    temp=[]
    "change while loop to while rem>0"
    while i==0:
        if rem%t==0:
            temp.append(0)
        else:
            temp.append(1)
        rem=(x-x%t)
        t=t*2
        if rem==0:
            i=1
    temp.reverse()
    return temp


def sum(list1, list2):
    "you get two binary numbers in the form of lists, nimsum them,
    and  return the resulting list. e.g. sum([1,1,0], [1,1])
    returns [1,0,1]"
    if len(list1)>len(list2):
        x=len(list1)-len(list2)
        while x>0:
```

```
            list2.insert(0,0)
            x=x-1
    else:
        x=len(list2)-len(list1)
        while x>0:
            list1.insert(0,0)
            x=x-1
    temp=[]
    x=0
    while(x)<len(list1):
        temp.append((list1[x]+list2[x])%2)
        x+=1
    return temp


def numConvert(temp):
    "converts a given binary list back into a normal number"
    x=0
    y=0
    while x<len(temp):
        y=y+temp[x]*(2**(len(temp)-1-x))
        x+=1
    return y

#function to nim-sum two values
def nimSum(x,y):
    xBin=binConvert(x)
    yBin=binConvert(y)
    sumBin=sum(xBin,yBin)
    nimSumValue=numConvert(sumBin)
    return nimSumValue

#function for path nim
def f(x):
    grund=[0,1]
    "grund is the ordered list of all the grundy numbers for paths"
    "of lengths shorter than x. so 0 is f(0), 1 is f(1), and the
     program starts computing grundy numbers for paths of length
      2 and higher"
    prev=[0]
    "prev is the list of grundy numbers for all positions attainable"
     "in one move from a path of length x"
```

```
    while x<100:
        prev.append(grund[x-1])
        prev.append(grund[x-2])
        "clearly the path of length x-1 and length x-2 are attainable
        "in one  move from path of length x, so right away we add
        "the grundy numbers of these two shorter paths into prev"
        y=1
        while x-1-y>=0:
            "Adds the nimsum of the f-g values after removing 1
             vertex to prev"
            bin1=binConvert(grund[y])
            bin2=binConvert(grund[x-1-y])
            binsum=sum(bin1, bin2)
            s=numConvert(binsum)
            prev.append(s)
            y+=1
        y=1
        while x-2-y>=0:
            "Adds the nimsum of the f-g values after removing 2
             vertices to prev"
            bin1=binConvert(grund[y])
            bin2=binConvert(grund[x-2-y])
            binsum=sum(bin1,bin2)
            s=numConvert(binsum)
            prev.append(s)
            y=y+2
        y=0
        while y<len(prev):
            "find the minimal excluded element of prev"

            if prev.count(y)>0:
                y+=1
            else:
                grund.append(y)
                #print "The number ", x, " has grundy number ", y
                y=y+len(prev)
        prev=[0]
        x+=1
    return grund

#function to find the mex of a list of numbers
def findMex(mexList):
```

```
    y=0
    while y<len(mexList):
        if mexList.count(y)>0:
            y+=1
        else:
            mex=y
            y=y+len(mexList)
    return mex

#function to look up the path nimbler from the path nimbler list
def findPathNum(length):
    if length < 72:
        pathNimber=pathNum[length]
    else:
        pathNimber=pathNum[72+length% 12]
    return pathNimber

#function of the caterpillar data at index 1
def catN():
    catNum=[0,2,3,4,5,6,2,1,0,8,6,0,
    1,2,3,8,5,12,7,1,0,8,9,14,
    1,2,3,11,4,7,12,14,0,16,2,4,
    12,2,3,10,4,7,15,1,16,9,18,16,
    12,2,3,10,16,7,12,1,16,18,11,16,
    12,2,22,11,16,7,12,1,20,24,16,26,
    12,13,22,11,16,24,15,14,16,22,19,16,
    12,13,19,11,16,24,15,14,16,25,11,16,
    12,13,22,11,16,7,15,1,20,25,19,11,
    12,13,22,11,32,19,22,14,20,22,19,11,
    12,13,22,11,25,19,22,14,16,22,19,11,
    21,13,22,11,25,19,22,14,21,25,19,11,
    21,13,22,11,25,19,22,14,20,22,19,11,
    21,13,22,11,25,19,22,14,21,22,19,11]
    return catNum

#function to look up the nimbler of a C_{n,1} caterpillar
def findCatNum(length):
    if length < 156:
        catNimber=catNum[length]
    else:
        catNimber=catNum[156+length% 12]
    return catNimber
```

```
#function for the C_{n,2} data
def catN2():
    catNum2=[0,0,3,4,5,6,7,8,0,10,6,8,
    1,6,3,8,5,7,3,8,0,10,13,4,
    0,2,3,8,13,10,16,1,0,10,17,8,
    18,10,3,8,18,6,13,9,18,20,6,5,
    18,10,3,20,5,6,16,13,18,10,17,4,
    18,10,3,8,19,6,16,13,22,10,17,4,
    18,14,3,8,18,6,16,13,19,10,17,8,
    4,14,3,8,18,6,16,13,18,10,17,4,
    18,14,3,8,18,7,16,13,18,10,17,4,
    18,14,3,8,18,7,16,13,18,10,17,4,
    18,14,3,8,18,27,16,13,18,10,17,4,
    18,14,3,8,18,7,16,13,18,10,17,4]
    return catNum2

#function to look up the C_{n,2} nimber
def findCatNum2(length):
    if length < 132:
        catNimber=catNum2[length]
    else:
        catNimber=catNum2[132+length% 12]
    return catNimber

#function to find the numbers of a cycle path. m is the length of
#the cycle, and n is the length of the path, from 1-n
def cyclePath(m,n):
    S = pickle.load(open("StarData.txt"))
    cycleNum=[0]
    for i in range(1,n+1):
        mexList=[]

        #below are moves made on C
mexList.append(findPathNum(i+(m-1)))
mexList.append(findPathNum(i+(m-2)))
mexList.append(nimSum(findPathNum(m-2),findPathNum(i)))
mexList.append(nimSum(findPathNum(m-1),findPathNum(i-1)))
mexList.append(nimSum(findPathNum(m-2),findPathNum(i-1)))
mexList.append(findPathNum(i-1))
        if m%2==1:
            for j in range(1,(m-1)/2):
```

```
        mexList.append(S[m-(j+1)][j][i])
         mexList.append(S[m-(j+2)][j][i])
    mexList.append(S[(m-1)/2][(m-1)/2][i])
else:
            for j in range(1,(m-2)/2+1):
        mexList.append(S[m-(j+1)][j][i])
         mexList.append(S[m-(j+2)][j][i])


        #We now give moves made on the path
#remove the two edges on the path closest to G
mexList.append(findPathNum(i-2))

#Now we can run a for loop to check the rest of the moves
# made on the path
for k in range(1,i-2):
    mexList.append(nimSum(cycleNum[k],findPathNum(i-k-1)))
    mexList.append(nimSum(cycleNum[k],findPathNum(i-k-2)))
cycleNum.append(findMex(mexList))
    return cycleNum

#function to run and store several cycle-paths.
#Change the range of i to change the cycle paths we want.
#It will print the C_mP_n in a latex format.
def findCyclePath():
    for i in range (5,10):
        print '-----', i, '------'
        cycleNum=cyclePath(i,400)
        for i in range(len(cycleNum)):
            if i%12==11:
        print cycleNum[i], "\\"
        print i+1, "&",
            else:
        print cycleNum[i], "&",
    return cycleNum

#3-star paths
def Slkm(l,k,m):
    Stemp=[]
    mexList=[]

    #Moves made from the central vertex
```

```
    mexList.append(nimSum(findPathNum(k+m),findPathNum(l-1)))
    mexList.append(nimSum(findPathNum(l+m),findPathNum(k-1)))
    mexList.append(nimSum(findPathNum(l+k),findPathNum(m-1)))
    mexList.append(nimSum(findPathNum(k-1),
     nimSum(findPathNum(m),findPathNum(l-1))))
    mexList.append(nimSum(findPathNum(l),
     nimSum(findPathNum(k-1),findPathNum(m-1))))
    mexList.append(nimSum(findPathNum(k),
     nimSum(findPathNum(m-1),findPathNum(l-1))))
    mexList.append(nimSum(findPathNum(k-1),
     nimSum(findPathNum(m-1),findPathNum(l-1))))
    #print mexList

    #Moves made on l branch
    if l>1:
        mexList.append(nimSum(findPathNum(m+k),findPathNum(l-2)))
        mexList.append(nimSum(S[1][k][m],findPathNum(l-2)))
        for j in range(1,l-1):
    mexList.append(nimSum(S[j][k][m],findPathNum(l-j-2)))
    mexList.append(nimSum(S[j+1][k][m],findPathNum(l-j-2)))

    #Moves made on k branch
    if k>1:
        mexList.append(nimSum(findPathNum(l+m),findPathNum(k-2)))
        mexList.append(nimSum(S[l][1][m],findPathNum(k-2)))
        for j in range(1,k-1):
    mexList.append(nimSum(S[l][j][m],findPathNum(k-j-2)))
    mexList.append(nimSum(S[l][j+1][m],findPathNum(k-j-2)))


    #Moves made on m branch
    if m>1:
#print S, S[l][k][1]
        mexList.append(nimSum(findPathNum(l+k),findPathNum(m-2)))
        mexList.append(nimSum(S[l][k][1],findPathNum(m-2)))
        for j in range(1,m-1):
    #print S
    #print l, k, m, j
    #print S[l][k][j]
            #print S[l][k][j+1]
    mexList.append(nimSum(S[l][k][j],findPathNum(m-j-2)))
    mexList.append(nimSum(S[l][k][j+1],findPathNum(m-j-2)))
```

```
    Stemp=findMex(mexList)
    return Stemp

#function to run and store several 3-star paths.
#Change the range of l,k,m to change the cycle paths we want.
#It will print the out in a periodic latex format.
def find3Star():
    S=[]
    for l in range (0,20):
        S.append([])
        for k in range(0,20):
      S[l].append([])
            print "----------------"
            print "l=", l, "k=", k
    for m in range(0,1000):
                #print S
        S[l][k].append([])
        if l==0 and k==0 and m==0:
    S[l][k][m]=0
        elif l==0 and k==0:
    S[l][k][m]=findPathNum(m)
        elif l==0 and m==0:
    S[l][k][m]=(findPathNum(k))
        elif k==0 and m==0:
    S[l][k][m]=(findPathNum(l))
        elif l==0:
    S[l][k][m]=(findPathNum(k+m))
        elif k==0:
    S[l][k][m]=(findPathNum(l+m))
        elif m==0:
    S[l][k][m]=(findPathNum(k+l))
        else:
               S[l][k][m]=(Slkm(l,k,m))
                if m%12==11:
                    print S[l][k][m], "\\"
                    print m+1, "&",
                else:
                    print S[l][k][m], "&",
    return S

#function to compute 4-star paths
```

```
def Slkmn(l,k,m,n):
    #print 'l=', l, 'k=', k, 'm=', m
    Stemp=[]
    #for i in range(1,m+1):
    mexList=[]

    #Moves made from the central vertex
    mexList.append(nimSum(T[k][m][n],findPathNum(l-1)))
    mexList.append(nimSum(T[l][m][n],findPathNum(k-1)))
    mexList.append(nimSum(T[l][k][n],findPathNum(m-1)))
    mexList.append(nimSum(T[l][k][m],findPathNum(n-1)))
    mexList.append(nimSum(findPathNum(l+k),
     nimSum(findPathNum(m-1),findPathNum(n-1))))
    mexList.append(nimSum(findPathNum(l+m),
     nimSum(findPathNum(k-1),findPathNum(n-1))))
    mexList.append(nimSum(findPathNum(l+n),
     nimSum(findPathNum(m-1),findPathNum(k-1))))
    mexList.append(nimSum(findPathNum(k+n),
     nimSum(findPathNum(m-1),findPathNum(l-1))))
    mexList.append(nimSum(findPathNum(m+k),
     nimSum(findPathNum(l-1),findPathNum(n-1))))
    mexList.append(nimSum(findPathNum(m+n),
     nimSum(findPathNum(l-1),findPathNum(k-1))))
    mexList.append(nimSum(findPathNum(l),
     nimSum(findPathNum(k-1),nimSum(findPathNum(m-1),
findPathNum(n-1)))))
    mexList.append(nimSum(findPathNum(k),nimSum(findPathNum(l-1),
     nimSum(findPathNum(m-1),findPathNum(n-1)))))
    mexList.append(nimSum(findPathNum(m),nimSum(findPathNum(k-1),
     nimSum(findPathNum(l-1),findPathNum(n-1)))))
    mexList.append(nimSum(findPathNum(n),nimSum(findPathNum(k-1),
     nimSum(findPathNum(m-1),findPathNum(l-1)))))
    mexList.append(nimSum(findPathNum(l-1),nimSum(findPathNum(k-1),
     nimSum(findPathNum(m-1),findPathNum(n-1)))))
    #print mexList

    #Moves made on l branch
    if l>1:
        mexList.append(nimSum(findPathNum(m+k),findPathNum(l-2)))
        mexList.append(nimSum(S[1][k][m][n],findPathNum(l-2)))
        for j in range(1,l-1):
    mexList.append(nimSum(S[j][k][m][n],findPathNum(l-j-2)))
```

```
    mexList.append(nimSum(S[j+1][k][m][n],findPathNum(l-j-2)))

    #Moves made on k branch
    if k>1:
        mexList.append(nimSum(findPathNum(l+m),findPathNum(k-2)))
        mexList.append(nimSum(S[l][1][m][n],findPathNum(k-2)))
        for j in range(1,k-1):
    mexList.append(nimSum(S[l][j][m][n],findPathNum(k-j-2)))
    mexList.append(nimSum(S[l][j+1][m][n],findPathNum(k-j-2)))


    #Moves made on m branch
    if m>1:
#print S, S[l][k][1]
        mexList.append(nimSum(findPathNum(l+k),findPathNum(m-2)))
        mexList.append(nimSum(S[l][k][1][n],findPathNum(m-2)))
        for j in range(1,m-1):
    #print S
    #print l, k, m, j
    #print S[l][k][j]
            #print S[l][k][j+1]
    mexList.append(nimSum(S[l][k][j][n],findPathNum(m-j-2)))
    mexList.append(nimSum(S[l][k][j+1][n],findPathNum(m-j-2)))

    #Moves made on n branch
    if n>1:
        mexList.append(nimSum(findPathNum(l+k),findPathNum(n-2)))
        mexList.append(nimSum(S[l][k][m][1],findPathNum(n-2)))
        for j in range(1,n-1):
    #print S
    #print l, k, m, j
    #print S[l][k][j]
            #print S[l][k][j+1]
    mexList.append(nimSum(S[l][k][m][j],findPathNum(n-j-2)))
    mexList.append(nimSum(S[l][k][m][j+1],findPathNum(n-j-2)))

    Stemp=findMex(mexList)
    return Stemp

#function to run and store several 4-star paths.
#Change the range of l,k,m,n to change the cycle paths we want.
#It will print the out in a periodic latex format.
```

```
for l in range (0,5):
    S.append([])
    for k in range(0,5):
        #print S
S[l].append([])
for m in range(0,10):
            print "----------------"
            print "l=", l, "k=", k, "m=", m
            S[l][k].append([])
    for n in range(0,500):
                #print S
            S[l][k][m].append([])
        if l==0 and k==0 and m==0 and n==0:
    S[l][k][m][n]=0
        elif l==0 and k==0 and m==0:
     S[l][k][m][n]=findPathNum(n)
        elif l==0 and m==0 and n==0:
     S[l][k][m][n]=(findPathNum(k))
        elif k==0 and m==0 and n==0:
     S[l][k][m][n]=(findPathNum(l))
        elif k==0 and l==0 and n==0:
     S[l][k][m][n]=findPathNum(m)
        elif l==0 and n==0:
     S[l][k][m][n]=(findPathNum(k+m))
        elif k==0 and n==0:
     S[l][k][m][n]=(findPathNum(l+m))
        elif m==0 and n==0:
     S[l][k][m][n]=(findPathNum(k+l))
elif l==0 and k==0:
     S[l][k][m][n]=findPathNum(n+m)
elif l==0 and m==0:
     S[l][k][m][n]=findPathNum(n+k)
elif k==0 and m==0:
     S[l][k][m][n]=findPathNum(l+n)
elif l==0:
     S[l][k][m][n]=T[k][m][n]
elif k==0:
     S[l][k][m][n]=T[l][m][n]
elif m==0:
     S[l][k][m][n]=T[l][k][n]
elif n==0:
     S[l][k][m][n]=T[l][k][m]
```

```
            else:
                    S[l][k][m][n]=(Slkmn(l,k,m,n))
                    if n%12==11:
                         print S[l][k][m][n], "\\"
                         print n+1, "&",
                    else:
                         print S[l][k][m][n], "&",
    return S
```

C.3 Evolution of Strings Code

Below are some useful sage functions.

```
import random
from sage.rings.all import Integer
from sage.plot.scatter_plot import ScatterPlot

#create the transition matrix
def createMatrix(length):
    #print 'length', length
    N=length+1
    #mat2={}
    mat2=matrix(QQ,N,N)#RealField(1000),N,N)
    binList=listOfBinCoeff(length)
    for tempd in range(0, length+1):
        s2=0
        for tempi in range(0 ,1):
            s2=s2+myfun2(length, 0, tempd, tempi,1/4,binList)
        mat2[(0,tempd)]=s2 #float(s2)
    for tempc in range(1,length):
        prob= 1-Integer(tempc)/Integer(length)
        for tempd in range(0,length+1):
            s=0
            s2=0
            for tempi in range(max(0,tempc-tempd) ,tempc+1):
                s2=s2+myfun(length, tempc, tempd, tempi, prob,
                1/4,binList)
            mat2[(tempc,tempd)]=s2 #float(s2)
            #print latex(s2), tempc, tempd, tempi
    for tempd in range(0,length):
        mat2[(length,tempd)]=0
    mat2[(length,length)]=1
```

```
    #print mat2
    return mat2.transpose()

#function that computes the matrix entry for all rows except the
# top and bottom rows
def myfun(l,c1,d,i,p,x,binList):
    if c1<i:
        fun=0
    elif (l-c1)<(d-c1+i):
        fun=0
    else:
        fun=binList[c1][i]*(p*(1/x-1)*x)^i*(1-p+p*x)^(c1-i)*
        binList[l-c1][d-c1+i]
        *(x)^(d-c1+i)*((1/x-1)*x)^(l-d-i)
    return fun

#function that computes matrix entry for the top column of the matrix
def myfun2(l,c1,d,i,x,binList):
    fun2=binList[(l-c1)][(d-c1+i)]*(x)^(d-c1+i)*((1/x-1)*x)^(l-d-i)
    return fun2

#List all Binary coefficients so we only compute them once
def listOfBinCoeff(length):
    var('x,y')
    binList=[]
    for x in range(0,length+1):
binList.append([])
for y in range(0,x+1):
    binList[x].append(binomial(x,y))
    return binList

#Power method to find the largest eigenvalue
def powerMethod(mat, N, tol):
    u=matrix(RealField(1000),N,1)
    count=0
    err=1
    for i in range(0,N):
       u[i]=1
    while(tol<err.abs() and count<10000):
err=u.norm(Infinity)
u=1/u[0,0]*u
        #u=1/u.norm(Infinity)*u
```

```
        u=mat*u
err -=u.norm(Infinity)
count+=1
    #print u
    return u

#Test to tell the probability we are in the absorbing state at time t
def convergenceTime(N, mat, num):
    u=matrix(RealField(1000),N,1)
    u[0]=1
    for t in range(num):
u=mat*u
        #print 't=', t, 'u'
        #print u
    return u

#The definition of absorption. t tells the number of steps to get
#to the absorbing state.
def absorption(len, mat):
    Q=matrix(QQ,len,len);
    for i in range(0,len):
        for j in range(0,len):
            Q[i,j]=mat[i,j]
    N=identity_matrix(QQ,len)-Q
    N=N.inverse()
    c=matrix(RealField(25),1,len)
    for i in range(0,len):
        c[0,i]=1
    t=c*N
    print t
    return t

#Wilf model estimate for time to absorption
def Wilf(len, alphabet):
    wilfConvergence=log(len)/(log(alphabet/(alphabet-1)))
    return wilfConvergence

#Plot for our absorbing state vs. Wilf's model
def absorbVsWilf():
    wilfList=[]
    absList=[]
    count=0
```

```
    for len in range(5,11):
        wilfList.append([])
absList.append([])
        mat=createMatrix(len)
        t=absorption(len,mat)
absList[count].append(len)
        absList[count].append(t[0,0])
w=Wilf(len,4)
wilfList[count].append(len)
wilfList[count].append(w)
        count=count+1
    p1=points(absList, rgbcolor=(1,0,0))
    p2=points(wilfList)
    show(p1+p2)



#Plot for the absorbing states
def plotAbsorbingStates():
    plotListAb=[]
    count=0
    for len in range(5,20):
        plotListAb.append([])
        mat=createMatrix(len)
        #u=convergenceTime(len+1,mat,10)
        #print u
        t=absorption(len,mat)
        plotListAb[count].append(len)
        plotListAb[count].append(t[0,0])
        count=count+1
    s= points(plotListAb)
    show(s)

def findSecondEigWithQ(mat,len):
    Q=matrix(RealField(1000),len,len)
    for i in range(len):
        for j in range(len):
    Q[i,j]=mat[i,j]
    u2=powerMethod(Q,len,10^(-30))
    return u2
```

```
#plot for second eigenvalues
def plotSecondEig():
    plotList=[]
    count=0
    for len in range(10,101):
        mat=createMatrix(len)
u2=findSecondEigWithQ(mat,len)
        print "len=", len
        print u2[0,0]
        #If we want to collect a list of data points on the second
        #largest eigenvalue for plotting purposes
        plotList.append([])
        plotList[count].append(len)
        plotList[count].append(u2[0,0])
        count=count+1
        #print u2
        #print u2[0]
        #if u2[0]>1:
    #print 'len', len
    #print plotList
    s= points(plotList)
    show(s)

def OneMinusLambda(len):
    mat=createMatrix(len)
    u2=findSecondEigWithQ(mat,len)
    vecSum=0
    for i in range(len):
vecSum=vecSum+u2[i,0]
    u2=u2*1/vecSum
    sum=0
    for i in range(len):
sum=sum+mat[len,i]*u2[i,0]
    return sum

def plotOneMinusLambda():
    plotList=[]
    count=0
    for len in range(5,20):
        sum=OneMinusLambda(len)
plotList.append([])
plotList[count].append(len)
```

```
plotList[count].append(sum)
count=count+1
    s=points(plotList)
    show(s)
```

Below are some useful matlab functions

```
    function [] = eigenVal(len, k)

digits=50;

%Function that will create a matrix for a string of length l, over
%an alphabet of size k.

N=len+1;
matrix=zeros(N);
%Top row
for tempd=0:N-1
    matrix(1,tempd+1)=matrix(1,tempd+1)+
    matrixEntry(len,0, tempd, 0, 1/k, 0);
end
%Middle of the matrix
for tempc=1:len-1
    prob=1-tempc/len;
    for tempd=0:len
        for tempi=max(0,tempc-tempd):tempc
            matrix(tempc+1,tempd+1)=matrix(tempc+1,tempd+1)+
            matrixEntry(len,tempc,tempd,tempi,1/k,prob);
        end
    end
end
%Bottom row
for tempd=0:(len-1)
    matrix(len+1,tempd+1)=0;
end
matrix(len+1,len+1)=1;
matrix=matrix.';
[V,D]=eig(matrix,'nobalance');
eig(vpa(matrix));
en=zeros(len+1,1);
en(len+1,1)=1;
Q=zeros(len);
```

```
for i=1:(len)
    for j=1:len
    Q(i,j)=matrix(i,j);
    Q(i,j)=matrix(i,j);
    end
end
[vecQ, valQ]=powerMeth(len,Q)

end

function [matEntry]= matrixEntry(l, c, d, i, x, p)
if l-c<d-c+i
    matEntry=nchoosek(c,i)*(p*(1/x-1)*x)^i*(1-p+p*x)^(c-i)*0*
    (x)^(d-c+i)*((1/x-1)*x)^(l-d-i);
else
    matEntry=nchoosek(c,i)*(p*(1/x-1)*x)^i*(1-p+p*x)^(c-i)*
    nchoosek(l-c,d-c+i)*(x)^(d-c+i)*((1/x-1)*x)^(l-d-i);
end
end

function [B,eigenVal,eigenVector]=deflation(len,matrix,eigOld)
v=ones(len+1,1);
e1=zeros(len+1,1);
e1(1)=eigOld;
k1=-sign(e1'*v)*norm(v,2)
n=norm(v,2);
beta=1/(n*(n+norm(e1'*v)))
u=v-k1*e1
H1=eye(len+1)-beta*u*u'
Anew=H1*matrix*(eye(len+1)-beta*u*u')
B=zeros(len);
b1t=zeros(1,len);
for i=2:len+1
    for j=2:len+1
        B(i-1,j-1)=Anew(i,j);
    end
    b1t(i-1)=Anew(1,i);
end
Anew;
B
[u,eig]=powerMeth(len,B)
alpha=(b1t*u)/(eig-1)
```

```
z=zeros(len+1,1);
z(1)=alpha;
for i=2:len+1
    z(i)=u(i-1);
end
eigVec=H1*z;
eigenVal=eig;
eigenVector=eigVec/eigVec(len);
end

function [vec, val]=deflation2(len,matrix, eig1, eigVec1)
B=matrix-eig1*eigVec1*eigVec1.';
[vec, val]=powerMeth(len+1,B);
vecSum=0;
for i=1:length(vec)
    vecSum=vecSum+vec(i);
end
vec=vec*1/vecSum;
end

function []=deflation3(len,matrix, eig1, eigVec1)
x=eigVec1/(norm(eigVec1,2)^2);
B=matrix-eig1*eigVec1*x.';
[vec, val]=powerMeth(len+1,B);
v=(val-eig1)*vec+eig1*(x.'*vec)*eigVec1;
end

function [t]=absorption(len, matrix)
Q=zeros(len);
for i=1:len
    for j=1:len
        Q(i,j)=matrix(i,j);
    end
end
%[U1,V1]=eig(Q,'nobalance')
N=inv(eye(len)-Q);
c=ones(len,1);
t=N*c;
end

function []=gramSchmidtPower(len,matrix)
v=ones(len,1);
```

```
u=zeros(len,1);
for i=1:(len-1)
u(i,1)=1/(len-1);
end
u(len,1)=-1;
for i=1:100
u=u-dot(u,v)/dot(v,v)*v;
u=matrix*u;
end
u;
end

function[]=timeToAbsorption(len,matrix)
e1=zeros(len);
e1(1)=1;
time=0;
while time<10
e1=matrix*e1
end
end

function [u, eigNew]=powerMeth(len,matrix)
count=0;
u=ones(len,1);
u(len-1)=0;
tol=1;
u1=matrix*u;
u=(1/(u1(1))*u1);
eigOld=u1(1);
%for tempi=0:200
while (tol > 10^(-40) && count<1000)
    u1=matrix*u;
    eigNew=u1(1);
    u=(1/eigNew*u1);
    tol=abs((eigNew-eigOld)/eigNew);
    eigOld=u1(1);
    count=count+1;
end
eigNew;
vecSum=0;
for i=1:length(u)
    vecSum=vecSum+u(i);
```

```
end
u=u*1/vecSum;
end
```

Bibliography

[1] Theory of impartial games. Lecture Notes, MIT, 2009.

[2] Julius G. Baron. The game of nim-a heuristic approach. Mathematics Magazine, 1974.

[3] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. Winning ways for your mathematical plays, second edition. A K Peters Ltd., 2001.

[4] 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.

[5] Brian Bowers, Neil Calkin, Kerry Gannon, Janine E. Janoski, Katie Joes, and Anna Kirkpatrick. The log concavity of the partition function. in preparation.

[6] Neil Calkin, Kevin James, Janine E. Janoski, Sarah Leggett, Bryce Richards, and Stephanie Thomas. Computing strategies for graphical nim. *Congressus Numeratntium*, 202:171–186, 2010.

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

[8] Thomas S. Ferguson. Game theory. Lecture Notes, Math 167, School, 2000.

[9] Masahiko Fukuyama. A nim game played on graphs. Theoretical Computer Science, 2003.

[10] Masahiko Fukuyama. A nim game played on graphs ii. Theoretical Computer Science, 2003.

[11] E. Garcia. Matrix tutorial 3: Eigenvalues and eigenvectors. http://www.miislita.com/information-retrieval-tutorial/matrix-tutorial-3-eigenvalues-eigenvectors.htmlpower-method, 2006.

[12] Charles M. Grinstead and J. Laurie Snell. *Introduction to Probability*. American Mathematical Society, 2006.

[13] 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.

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

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

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

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

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

[19] Xi-Fan Wang, Yonghua Song, and Malcolm Irving. *Modern Power System Analysis*. Springer, 2008.

[20] Herbert S. Wilf and Warren J. Ewens. There's plenty of time for evolution. arXiv:1010.5178v1, 2010.