Champion Spiders in the Game of Graph Nim

Allison Nelson, Sydney Ryan, Chao Xu

July 13, 2012

Abstract

In the game of Graph Nim, players take turns removing one or more edges incident to a chosen vertex in a graph. The player that removes the last edge in the graph wins. A spider graph is a champion if it has a Sprague-Grundy number equal to the number of edges in the graph. We investigate the the Sprague-Grundy numbers of various spider graphs when the number of paths or length of paths increase.

1 Introduction

Nim is an impartial two-player game traditionally played with multiple piles of sticks or stones.

Definition An *impartial game* is a game in which at any given state state of the game, each player has the same set of possible moves [5].

On a player's turn, one or more stones are removed from a pile, and the player that removes the last stone (or stones) is the winner [5]. We want to study a variation of this game, called Graph Nim. In Graph Nim, players remove edges incident to a vertex and the player to remove the last edge(s) is the winner [1].

Definition A *move* in a game of Graph Nim is the removal of one or more edges adjacent to a single vertex.

In particular, we are interested in patterns and periodic behavior of the Sprague-Grundy number, or nimber, as we vary the graph that we are playing on.

Definition The *minimal excluded value*, or *mex*, of a set of integers is the smallest non-negative integer not included in the set.

Definition The *followers* of a graph G are the set of all graphs that can be reached in one move on G. We denote the followers of G as $F(G)$.

Using the previous definitions, we can define the Sprague-Grundy number as follows:

Definition Let G be a graph. We define a function $g(G) = \text{mex}(\{g(H)|H \in F(G)\})$. We call $g(G)$ the *Sprague-Grundy number*, or *nimber* of G. We denote $\{g(H)|H \in F(G)\} = gF(G)$.

The Sprague-Grundy Theorem allows us to assume that games of Graph Nim actually have a Sprague-Grundy number.

Theorem 1.1. *(Sprague-Grundy Theorem) Every impartial game under the normal play convention is equivalent to a nimber.*

We have the following theorem for disjoint games.

Theorem 1.2. *The Sprague-Grundy number of a game consisting of disjoint components is the nim-sum of the Sprague-Grundy numbers of those components [3].*

Definition We define the *nim-sum* of two non-negative integers as binary addition with no carries. We denote it as $\oplus$ [2].

We have shown that for certain classes of graphs the nimbers become periodic as we make the graphs larger.

Definition A *spider graph* is a graph with one vertex of degree greater than 2 and all other vertices with degree at most 2. We will extend the class of spider graphs to include paths for our purposes.

For example, the graph in Figure 1 is a spider graph. By studying spider graphs, we describe

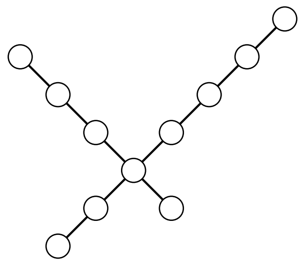

Figure 1: A spider graph

methods for determining the nimbers of several different types of spiders used in a game of Graph Nim. We are interested in the periodicity of the nimbers as we increase the length and number of paths attached to the spider. The Sprague-Grundy number of a game analyzes whether or not the “first player” has a winning strategy. A non-zero nimber ensures that the first player will always win if he plays the correct strategy. A nimber of zero means that the first player does not have a winning strategy and will always lose if his opponent plays the correct strategy [2].

To find the Sprague-Grundy number of a graph G, we want to find the minimal excluded value out of the Sprague-Grundy numbers of all graphs that can be obtained in one move on G. For example, the graph in Figure 2 has the followers in Figure 3.

The nimber for the graph in Figure 2 is $g(G) = \text{mex}(\{3, 2, 1, 0\}) = 4$.
The following result bounds the Sprague-Grundy number for any graph.

Theorem 1.3. *$g(G) \leq e(G)$ where G is a graph and $e(G)$ is the number of edges in G.*

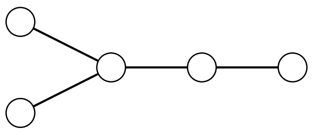
Figure 2: The graph G

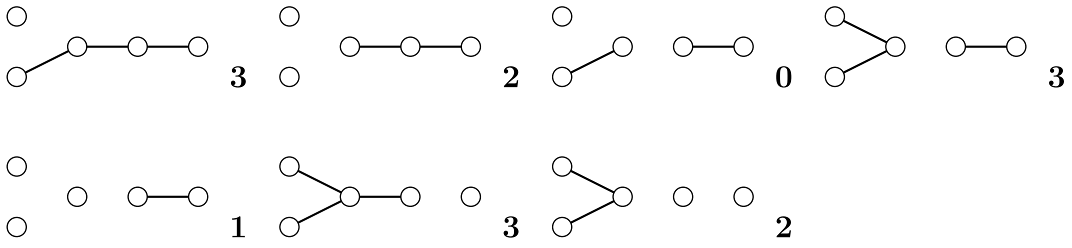

Figure 3: Followers of G and their S-G Numbers

Proof. Proof by contradiction.
Let G be a counterexample with a minimal number of edges. A graph with no edges has a nimber of 0, thus it can't be a counterexample, and thus $e(G) > 0$. This means $g(G) > e(G)$, so there must be a follower $G' \in F(G)$ whose nimber is $e(G)$. $e(G) > e(G')$, since G' is a follower of G. This means that $g(G') > e(G')$, which makes G' a counterexample as well. We assumed that G was the minimal counterexample, so this is a contradiction. Thus $g(G) \leq e(G)$. □

From now on we will use $e(G)$ to denote the number of edges in G.
We can think of a spider as consisting of a central vertex, or a body, and a set of paths adjacent to it, the legs.

Definition The *body* of a spider graph is the vertex of degree > 2.

Definition The *legs* of a spider graph are the paths connected to the body.

There are two distinct types of moves in Graph Nim on spider graphs, we call them stomps and cuts.

Definition A *stomp* removes one or more edges adjacent to the body.

Definition A *cut* removes edges not adjacent to the body.

We want to study spider graphs that are somehow related to see if there is any pattern to their Sprague-Grundy numbers. To do this, we consider a mapping between spider graphs and integer partitions. If a spider graph consists of a_1 legs of length 1, a_2 legs of length 2, …, a_n legs of length n, we represent this graph by the partition

$$(\underbrace{n,\dots,n}_{a_n \text{ times}}, \underbrace{n-1,\dots,n-1}_{a_{n-1} \text{ times}}, \dots, \underbrace{1,\dots,1}_{a_1 \text{ times}}).$$

We have created programs in both Java and Sage to calculate the nimbers of various graphs represented as partitions and output them in lists that we can sort and analyze (See data in Appendix A and programs in Appendix B). After looking through the data that was gathered, a few simple patterns were observed.

2 Sprague-Grundy Numbers of Spider Graphs

First, we consider spider graphs with k legs of length 1.

Definition A *k-star* is a graph consisting of a single vertex and k paths of length 1 adjacent to the vertex. We denote a k-star as S_k.

Theorem 2.1. *For all $k \geq 1$, $g(S_k) = k$ where S_k is a star graph with k edges [4].*

Let P_k denote a path of length k. Denote $\underbrace{P_k \cup \ldots \cup P_k}_{n \text{ times}}$ by $(n)P_k$. The Sprague-Grundy numbers for paths are known [1]. The Sprague-Grundy numbers become periodic as the length of the path grows larger than 72. We can use the path nimbers to help analyze simple spider graphs.

Theorem 2.2. *For all $a_1 \geq 2$, $g(3^1 1^{a_1}) = 3 + a_1$.*

Proof. The proof is by induction on a_1.

Base Case: We verify these first few cases computationally.

$$a_1 = 2,\ g(3^1 1^2) = 5$$
$$a_1 = 3,\ g(3^1 1^3) = 6$$
$$a_1 = 4,\ g(3^1 1^4) = 7$$
$$a_1 = 5,\ g(3^1 1^5) = 8$$

Inductive Step: Assume that $g(3^1 1^k) = 3 + k$ for all $k < a_1$. Consider a_1. We generate these followers and nimbers for $3^1 1^{a_1}$ by applying a cut:

$$\begin{aligned} g(2^1 1^{a_1}) &= 2 + a_1 \\ g(P_1) \oplus g(S_{a_1+1}) &= 1 \oplus (1 + a_1) \\ g(S_{a_1+1}) &= 1 + a_1 \\ g(P_2) \oplus g(S_{a_1}) &= 2 \oplus a_1 \\ g(P_1) \oplus g(S_{a_1}) &= 1 \oplus a_1 \end{aligned}$$

In addition, we generate these followers by applying a stomp:

$$\begin{aligned} g(2^1) \oplus g(S_i) = 2 \oplus i &\text{ for } 0 \leq i \leq a_1 \\ g(3^1 1^i) &\text{ for } 0 \leq i < a_1 \end{aligned}$$

We need to show that for each possibility of a_1, the simplified value does not equal $a_1 + 3$ but generates every value below it. We know that each follower is bounded by its number of edges, so we simply need to show that there is followers that generate every nimber in $\{0, \ldots, a_1 + 2\}$.

Case 1 $a_1 \equiv 0 \mod 4$. The followers of the form $P_2 \cup S_i$ for $0 \leq i \leq a_1$ will produce nimbers $\{0, \ldots, a_1 - 1, a_1 + 2\}$. The follower $P_1 \cup S_{a_1+1}$ will give the nimber a_1 and according to Theorem 2.1 S_{a_1+1} produces the nimber $a_1 + 1$. Since $gF(3^1 1^{a_1}) = \{0, \ldots, a_1 + 2\}$, $g(3^1 1^{a_1}) = a_1 + 3$.

Case 2 $a_1 \equiv 1 \mod 4$. The followers of the form $P_2 \cup S_i$ for $0 \leq i \leq a_1$ will produce nimbers $\{0, \ldots, a_1 - 2, a_1 + 1, a_1 + 2\}$. The follower $P_1 \cup S_{a_1}$ will give the nimber $a_1 - 1$. By induction, $3^1 1^{a_1 - 3}$ gives nimber a_1. Since $gF(3^1 1^{a_1}) = \{0, \ldots, a_1 + 2\}$, $g(3^1 1^{a_1}) = a_1 + 3$.

Case 3 $a_1 \equiv 2 \mod 4$. The followers of the form $P_2 \cup S_i$ for $0 \leq i \leq a_1$ will produce nimbers $\{0, \ldots, a_1 - 2, a_1, a_1 + 1\}$. The follower $P_1 \cup S_{a_1}$ will give the nimber $a_1 - 1$. By induction, $g(3^1 1^{a_1 - 1}) = a_1 + 2$ and $g(3^1 1^{a_1 - 4}) = a_1 - 1$. Since $gF(3^1 1^{a_1}) = \{0, \ldots, a_1 + 2\}$, $g(3^1 1^{a_1}) = a_1 + 3$.

Case 4 $a_1 \equiv 3 \mod 4$. The followers of the form $P_2 \cup S_i$ for $0 \leq i \leq a_1$ will produce nimbers $\{0, \ldots, a_1\}$. By induction, $g(3^1 1^{a_1 - 1}) = a_1 + 2$ and $g(3^1 1^{a_1 - 2}) = a_1 + 1$. Since $gF(3^1 1^{a_1}) = \{0, \ldots, a_1 + 2\}$, $g(3^1 1^{a_1}) = a_1 + 3$.

□

Since we now know something about graphs of the form $3^1 1^{a_1}$, let's look at $4^1 1^{a_1}$. Eventually, we hope to generalize this to graphs of the form $n^{a_n} \ldots 2^{a_2} 1^{a_1}$.

Theorem 2.3. *For all $a_1 \geq 4$, $g(4^1 1^{a_1}) = 4 + a_1$.*

Proof. The proof is by induction on a_1.

Base Case: We computationally verify the first few cases.

$$\begin{aligned} a_1 = 4,\ g(4^1 1^4) &= 8 \\ a_1 = 5,\ g(4^1 1^5) &= 9 \\ a_1 = 6,\ g(4^1 1^6) &= 10 \\ a_1 = 7,\ g(4^1 1^7) &= 11 \end{aligned}$$

Inductive Step: Assume that $g(4^1 1^k) = 4 + k$ for all $k < a_1$. Consider a_1. The cuts for $4^1 1^{a_1}$ are:

$$\begin{aligned} g(3^1 1^{a_1}) &= 3 + a_1 \\ g(2^1 1^{a_1}) &= 2 + a_1 \\ g(P_1) \oplus g(2^1 1^{a_1}) &= 1 \oplus (2 + a_1) \\ g(P_1) \oplus g(S_{a_1+1}) &= 1 \oplus (1 + a_1) \\ g(P_2) \oplus g(S_{a_1+1}) &= 2 \oplus (1 + a_1) \\ g(P_2) \oplus g(S_{a_1}) &= 2 \oplus a_1 \\ g(P_3) \oplus g(S_{a_1}) &= 3 \oplus a_1 \end{aligned}$$

The stomp moves for $4^1 1^{a_1}$ are:

$$\begin{aligned} g(P_3) \oplus g(S_i) = 3 \oplus i &\text{ for } 0 \leq i \leq a_1 \\ g(4^1 1^i) &\text{ for } 0 \leq i < a_1 \end{aligned}$$

We need to show that for each possibility of a_1, the simplified value does not equal a_1+4 but generates every value below it. We know that each follower is bounded by its number of edges, so we simply need to show that there is followers that generate every nimber in $\{0,\dots,a_1+3\}$.

Case 1 $a_1 \equiv 0 \mod 4$. The followers of the form $P_3 \cup S_i$ for $0 \leq i \leq a_1$ will produce nimbers of $\{0,\dots,a_1-1,a_1+3\}$. The follower $P_1 \cup S_{a_1+1}$ will give the nimber a_1 and the follower $P_2 \cup S_{a_1}$ produces the nimber a_1+2. By induction, we have that $g(4^1 1^{a_1-2})$ gives the nimber a_1+2. Since $gF(4^1 1^{a_1}) = \{0,\dots,a_1+3\}$, $g(4^1 1^{a_1}) = a_1+4$.

Case 2 $a_1 \equiv 1 \mod 4$. The followers of the form $P_3 \cup S_i$ for $0 \leq i \leq a_1$ will produce nimbers of $\{0,\dots,a_1-2,a_1+1,a_1+2\}$. The follower $P_3 \cup S_{a_1}$ will give the nimber a_1-1. By induction, $g(4^1 1^{a_1-4}) = a_1$ and $g(4^1 1^{a_1-1}) = a_1+3$. Since $gF(4^1 1^{a_1}) = \{0,\dots,a_1+3\}$, $g(4^1 1^{a_1}) = a_1+4$.

Case 3 $a_1 \equiv 2 \mod 4$. The followers of the form $P_3 \cup S_i$ for $0 \leq i \leq a_1$ will produce nimbers of $\{0,\dots,a_1\}$. The follower $2^1 1^{a_1}$ will give the nimber a_1+2. By induction, $g(4^1 1^{a_1-1}) = a_1+3$ and $g(4^1 1^{a_1-3}) = a_1+1$. Since $gF(4^1 1^{a_1}) = \{0,\dots,a_1+2\}$, $g(4^1 1^{a_1}) = a_1+4$.

Case 4 $a_1 \equiv 3 \mod 4$. The followers of the form $P_3 \cup S_i$ for $0 \leq i \leq a_1$ will produce nimbers of $\{0,\dots,a_1-3,a_1-1,a_1,a_1+1\}$. The follower $P_2 \cup S_{a_1}$ generates the nimber a_1-2. By induction, $g(4^1 1^{a_1-2}) = a_1+2$ and $g(4^1 1^{a_1-1}) = a_1+3$. Since $gF(4^1 1^{a_1}) = \{0,\dots,a_1+3\}$, $g(4^1 1^{a_1}) = a_1+4$.

Since for all cases of a_1 we have generated nimbers from $\{0,\dots,a_1+3\}$ we can conclude that the nimber for $4^1 1^{a_1}$ is a_1+4. □

Now we examine graphs of the form $5^1 1^{a_1}$. We will find that we get slightly different behavor than we did for $3^1 1^{a_1}$ and $4^1 1^{a_1}$.

Theorem 2.4. *For all $a_1 \geq 8$,*

$$g(5^1 1^{a_1}) = \begin{cases} a_1+5 & \text{if } a_1 \equiv 0 \text{ or } 3 \mod 4, \\ a_1+2 & \text{if } a_1 \equiv 1 \mod 4, \\ a_1 & \text{if } a_1 \equiv 2 \mod 4. \end{cases}$$

Proof. The proof is by induction on a_1.

Base Case:

$$\begin{aligned} a_1 = 8,\ g(5^1 1^8) &= 13 \\ a_1 = 9,\ g(5^1 1^9) &= 11 \\ a_1 = 10,\ g(5^1 1^{10}) &= 10 \\ a_1 = 11,\ g(5^1 1^{11}) &= 16 \\ a_1 = 12,\ g(5^1 1^{12}) &= 17 \end{aligned}$$

Inductive Step: Assume that our hypothesis holds for all values less than a_1. Now consider the graph $5^1 1^{a_1}$.

We generate the Sprague-Grundy numbers for each of the follower graphs of $5^1 1^{a_1}$. The follower graphs formed from cuts on $5^1 1^{a_1}$ are:

$$\begin{aligned}
g(4^1 1^{a_1}) &= 4 + a_1 \\
g(3^1 1^{a_1}) &= 3 + a_1 \\
g(P_1) \oplus g(3^1 1^{a_1}) &= 1 \oplus (3 + a_1) \\
g(P_1) \oplus g(2^1 1^{a_1}) &= 1 \oplus (2 + a_1) \\
g(P_2) \oplus g(2^1 1^{a_1}) &= 2 \oplus (2 + a_1) \\
g(P_2) \oplus g(S_{a_1+1}) &= 2 \oplus (1 + a_1) \\
g(P_3) \oplus g(S_{a_1+1}) &= 3 \oplus (1 + a_1) \\
g(P_3) \oplus g(S_{a_1}) &= 3 \oplus a_1 \\
g(P_4) \oplus g(S_{a_1}) &= 1 \oplus a_1.
\end{aligned}$$

In addition, the followers created by stomps are:

$$\begin{aligned}
g(P_4) \oplus g(S_i) &= 1 \oplus i \text{ for } 0 \le i \le a_1 \\
g(5^1 1^i) &\text{ for } 0 \le i < a_1.
\end{aligned}$$

Case 1 $a_1 \equiv 0 \mod 4$.

We have:

$$\begin{aligned}
g(4^1 1^{a_1}) &= 4 + a_1 \\
g(3^1 1^{a_1}) &= 3 + a_1 \\
g(P_1) \oplus g(3^1 1^{a_1}) &= 1 \oplus (3 + a_1) = 2 + a_1 \\
g(P_2) \oplus g(2^1 1^{a_1}) &= 2 \oplus (2 + a_1) = a_1 \\
g(P_4) \oplus g(S_{a_1}) &= 1 \oplus a_1 = a_1 + 1.
\end{aligned}$$

Next we consider $P_4 \cup S_i$ for $0 \le i \le a_1$. If $i \equiv 1 \mod 2$, then the expression $i \oplus 1 = i - 1$. If $i \equiv 0 \mod 2$, then the expression $i \oplus 1 = i + 1$.

Since a_1 is even, then the values $\{0 \ldots a_1 - 1, a_1 + 1\}$ are generated. Since all followers have at most $a_1 + 4$ edges we have $gF(G) = \{0, \ldots, a_1 + 4\}$. Thus, $g(5^1 1^{a_1}) = a_1 + 5$ for $a_1 \equiv 0 \mod 4$.

Case 2 $a_1 \equiv 1 \mod 4$.

We have:

$$\begin{aligned}
g(4^1 1^{a_1}) &= 4 + a_1 \\
g(3^1 1^{a_1}) &= 3 + a_1 \\
g(P_1) \oplus g(3^1 1^{a_1}) &= 1 \oplus (3 + a_1) = 4 + a_1 \\
g(P_1) \oplus g(2^1 1^{a_1}) &= 1 \oplus (2 + a_1) = 1 + a_1 \\
g(P_2) \oplus g(2^1 1^{a_1}) &= 2 \oplus (2 + a_1) = a_1 \\
g(P_2) \oplus g(S_{a_1+1}) &= 2 \oplus (1 + a_1) = a_1 - 1 \\
g(P_3) \oplus g(S_{a_1+1}) &= 3 \oplus (1 + a_1) = a_1 - 1 \\
g(P_3) \oplus g(S_{a_1}) &= 3 \oplus a_1 = a_1 + 1 \\
g(P_4) \oplus g(S_{a_1}) &= 1 \oplus a_1 = a_1 - 1
\end{aligned}$$

Next we consider $P_4 \cup S_i$ for $0 \leq i \leq a_1$. If $i \equiv 1 \mod 2$, then the expression $i \oplus 1 = i - 1$. If $i \equiv 0 \mod 2$, then the expression $i \oplus 1 = i + 1$.

Since a_1 is odd, then the values $\{0 \dots a_1\}$ are generated. For $5^1 1^{a_i - j}$, where $j \in \{1, 2, 3\}$, we get these values by the inductive hypothesis:

$$\begin{aligned}
g(5^1 1^{a_1-1}) &= a_1 + 4 \text{ since } a_1 - 1 \equiv 0 \mod 4 \\
g(5^1 1^{a_1-2}) &= a_1 + 3 \text{ since } a_1 - 2 \equiv 3 \mod 4 \\
g(5^1 1^{a_1-3}) &= a_1 - 3 \text{ since } a_1 - 3 \equiv 2 \mod 4
\end{aligned}$$

Since all other followers have at most $a_1 + 1$ edges, we have $gF(G) = \{0, \dots, a_1 + 1, a_1 + 3, a_1 + 4\}$. Thus, $g(5^1 1^{a_1}) = a_1 + 2$ for $a_1 \equiv 1 \mod 4$.

Case 3 $a_1 \equiv 2 \mod 4$.

We have:

$$\begin{aligned}
g(4^1 1^{a_1}) &= 4 + a_1 \\
g(3^1 1^{a_1}) &= 3 + a_1 \\
g(P_1) \oplus g(3^1 1^{a_1}) &= 1 \oplus (3 + a_1) = 2 + a_1 \\
g(P_1) \oplus g(2^1 1^{a_1}) &= 1 \oplus (2 + a_1) = 3 + a_1 \\
g(P_2) \oplus g(2^1 1^{a_1}) &= 2 \oplus (2 + a_1) = a_1 + 4 \\
g(P_2) \oplus g(S_{a_1+1}) &= 2 \oplus (1 + a_1) = a_1 - 1 \\
g(P_3) \oplus g(S_{a_1+1}) &= 3 \oplus (1 + a_1) = a_1 - 2 \\
g(P_3) \oplus g(S_{a_1}) &= 3 \oplus a_1 = a_1 - 2 \\
g(P_4) \oplus g(S_{a_1}) &= 1 \oplus a_1 = a_1 + 1
\end{aligned}$$

Next we consider $P_4 \cup S_i$ for $0 \leq i \leq a_1$. If $i \equiv 1 \mod 2$, then the expression $i \oplus 1 = i - 1$. If $i \equiv 0 \mod 2$, then the expression $i \oplus 1 = i + 1$.

Since a_1 is even, then the values $\{0 \dots a_1 - 1, a_1 + 1\}$ are generated. For $5^1 1^{a_i - j}$, where $j \in \{1, 2, \dots, 5\}$, we get these values by the inductive hypothesis:

$$\begin{aligned}
g(5^1 1^{a_1-1}) &= a_1 + 1 \text{ since } a_1 - 1 \equiv 1 \mod 4\\
g(5^1 1^{a_1-2}) &= a_1 + 3 \text{ since } a_1 - 2 \equiv 0 \mod 4\\
g(5^1 1^{a_1-3}) &= a_1 + 2 \text{ since } a_1 - 3 \equiv 3 \mod 4\\
g(5^1 1^{a_1-4}) &= a_1 - 4 \text{ since } a_1 - 4 \equiv 2 \mod 4\\
g(5^1 1^{a_1-5}) &= a_1 - 3 \text{ since } a_1 - 5 \equiv 1 \mod 4
\end{aligned}$$

Since all other followers have at most $a_1 - 1$ edges we have $gF(G) = \{0, \ldots, a_1 - 1, a_1 + 1, a_1 + 2, a_1 + 3, a_1 + 4\}$. Thus, $g(5^1 1^{a_1}) = a_1$ for $a_1 \equiv 2 \mod 4$.

Case 4 $a_1 \equiv 3 \mod 4$.

We have:

$$\begin{aligned}
g(4^1 1^{a_1}) &= 4 + a_1\\
g(3^1 1^{a_1}) &= 3 + a_1\\
g(P_1) \oplus g(2^1 1^{a_1}) &= 1 \oplus (2 + a_1) = 1 + a_1\\
g(P_3) \oplus g(S_{a_1}) &= 3 \oplus a_1 = a_1 - 3\\
g(P_4) \oplus g(S_{a_1}) &= 1 \oplus a_1 = a_1 - 1
\end{aligned}$$

Consider $P_4 \cup S_i$ for $0 \leq i \leq a_1$. If $i \equiv 1 \mod 2$, then the expression $i \oplus 1 = i - 1$. If $i \equiv 0 \mod 2$, then the expression $i \oplus 1 = i + 1$.

Since a_1 is odd, then the values $\{0 \ldots a_1\}$ are generated. By induction, we have $g(5^1 1^{a_1-3}) = a_1 + 2$. Since all followers have at most $a_1 + 4$ edges we have $gF(G) = \{0, \ldots, a_1 + 4\}$. Thus, $g(5^1 1^{a_1}) = a_1 + 5$ for $a_1 \equiv 3 \mod 4$.

□

3 Champion Spiders

Definition A *champion spider* is a spider graph whose Sprague-Grundy number is equal to its number of edges. In other words, $g(n^{a_n}(n-1)^{a_{n-1}} \ldots 1^{a_1}) = n(a_n) + (n-1)(a_{n-1}) + \ldots + a_1$ where n is the maximum leg length.

Notice that Theorem 2.2 and Theorem 2.3 tell us that spiders of the form $3^1 1^{a_1}$ and $4^1 1^{a_1}$ are champion.

Theorem 3.1. *$g(2^{a_2} 1^{a_1}) = 2a_2 + a_1$ if $a_1 \geq 2a_2 - 2$ for a_1, $a_2 \in \mathbb{N}$.*

Proof. Proof by induction on a_2.

Base case: $a_2 = 0$. $g(2^0 1^{a_1}) = a_1$ for all a_1 by Theorem 2.1.

Inductive Step:

Assume the hypothesis is true for all integers less than a_2. Consider $2^{a_2} 1^{a_1}$, where $a_1 \geq 2a_2 - 2$. The followers of $2^{a_2} 1^{a_1}$ are as follows:

1. $2^{a_2-1}1^{a_1+1}$
2. $2^{a_2-1}1^{a_1}$
3. $2^j1^k \cup (a_2-j)P_1$ for all $j \le a_2, k \le a_1$ and $j+k < a_2+a_1$. Note:
$$g(2^j1^k \cup (a_2-j)P_1) = \begin{cases} g(2^j1^k \cup P_1) & \text{if } a_2 - j \equiv 1 \mod 2 \\ g(2^j1^k) & \text{if } a_2 - j \equiv 0 \mod 2. \end{cases}$$

Claim: $\{0, \dots, 2a_2+a_1-1\} \subseteq gF(2^{a_2}1^{a_1})$. For any $h \in \{0, \dots, 2a_2+a_1-1\}$, there are three cases:

Case 1 $h = 2a_2+a_1-1 = g(2^{a_2-1}1^{a_1+1})$ by induction.

Case 2 $h = 2a_2+a_1-2 = g(2^{a_2-1}1^{a_1})$ by induction.

Case 3 $h < 2a_2+a_1-2$. Note that we can write $h = 2j+k$ where $0 \le j \le a_2-1$ and $2j-2 \le k < a_1$. We will show that there exists a follower graph with nimber equal to h for all h in this range.

1. If a_2-j is even, then $g(2^j1^k) = 2j+k = h$ by hypothesis.
2. If a_2-j is odd and h is odd, then k is odd. Therefore $k > 2j-2$ and $k-1 \ge 2j-2$, so
$$g(2^j1^{k-1} \cup P_1) = (h-1) \oplus 1 = h.$$
3. If a_2-j is odd and h is even. $k < a_1$ thus $k+1 \le a_1$, so
$$g(2^j1^{k+1} \cup P_1) = (h+1) \oplus 1 = h.$$

This shows $\{0, \dots, 2n+a_1-1\} \subseteq gF(2^{a_2}1^{a_1})$, and because $g(2^{a_2}1^{a_1}) \le 2a_2+a_1$, we have $g(2^{a_2}1^{a_1}) = 2a_2+a_1$. □

Definition The *integer interval* between a and b is the set of consecutive integers between and including a and b, denoted $[a..b]$.

Definition The numbers $[a..k]$, $a \ge 0$, are *generated* by a graph G if there exists a follower of G whose Sprague-Grundy number equals c for all $c \in [a..k]$.

We denote $k \oplus [a..b] = \{k \oplus x | x \in [a..b]\}$. Similarly, we denote $k+[a..b] = \{k+x | x \in [a..b]\}$.

Lemma 3.2. *$x - y \le x \oplus y \le x + y$ for $x, y \in \mathbb{Z}$.*

Proof. Since $\oplus$ is binary addition without carries, $x \oplus y \le x + y$. Assume $x \oplus y < x - y$ for some x and y. Then
$$x = x \oplus (y \oplus y) = (x \oplus y) \oplus y < (x-y) \oplus y \le (x-y)+y = x.$$
This is a contradiction. Thus, $x - y \le x \oplus y$. □

Lemma 3.3. *For $a, b, k \in \mathbb{N}$, $[a+k..b-k] \subseteq k \oplus [a..b]$.*

Proof. Let $x \in [a+k..b-k]$. We can write $t \oplus k = x$. Then $x \oplus k = t$ and $x-k \leq t \leq x+k$ by Lemma 3.2. Thus $t \in [(a+k)-k..(b-k)+k] = [a..b]$ so $x \in k \oplus [a..b]$. □

Lemma 3.4. *For $b, k \in \mathbb{N}$, $[0..b-k] \subseteq k \oplus [0..b]$.*

Proof. Let $x \in [0..b-k]$. We can write $t \oplus k = x$, so $x \oplus k = t$. This implies $t \in [0-k..(b-k)+k]$. Since $t \geq 0$, we must have that $t \in [0..b]$. Thus $x \in k \oplus [0..b]$. □

When working with a union of intervals, it is useful to know whether they all overlap and can be expressed instead as a single interval. We now prove a Lemma allowing us to check whether this is the case, and if so, what the final interval is.

Lemma 3.5. *Suppose $b_{i-1} \leq b_i$, and $c \geq b_i - b_{i-1} \geq 0$ for all $i \leq n$. If $b_i - a_i \geq c$ for all i, then*

$$\bigcup_{i=1}^{n}[a_i..b_i] = [\min_{1\leq i\leq n}(a_i)..b_n].$$

Example 3.6. *Let $a_i = 0, 2, 1, 7$ and $b_i = 3, 5, 8, 10$. Then,*

$$\bigcup_{i=1}^{n}[a_i..b_i] = [0..3] \cup [2..5] \cup [1..8] \cup [7..10].$$

Notice that we have that $b_{i-1} \geq b_i$ and $3 \geq b_i - b_{i-1} \geq 0$. For all $i \leq 4$, $b_i - a_i \geq 3$. Thus, we can apply the Lemma to express this as a single interval:

$$[0..3] \cup [2..5] \cup [1..8] \cup [7..10] = [0..10].$$

Proof. We have that

$$\bigcup_{i=1}^{n}[a_i..b_i] \subseteq [\min_{1\leq i\leq n}(a_i)..\max_{1\leq i\leq n}(b_i)] = [\min_{1\leq i\leq n}(a_i)..b_n].$$

Assume $x \in [\min_{1\leq i\leq n}(a_i)..b_n]$. For the sake of obtaining a contradiction, suppose $x \notin [a_i..b_i]$ for all i. Then it is in $[b_j..b_{j+1}]$ for some $j < n$. Since $a_{j+1} \leq b_{j+1} - c \leq b_j$, we know that $x \in [a_{j+1}..b_{j+1}]$. This is a contradiction. Thus, $\bigcup_{i=1}^{n}[a_i..b_i] = [\min_{i=1}^{n}(a_i)..b_n]$. □

Lemma 3.7. *For all $a_1, \ldots, a_n \in \mathbb{N}$, $a_n \neq 0$.*

$$gF(n^{a_n} \ldots 1^{a_1}) \supseteq \begin{cases} [0..a_1-1] & \text{if } n = 1, 2 \\ [0..a_1-3] & \text{if } n \in [3..5] \\ [0..a_1-7] & \text{if } n \in [6..27] \\ [0..a_1-15] & \text{if } n \in [28..\infty] \end{cases}$$

In particular, for all n, $gF(n^{a_n} \ldots 1^{a_1}) \supseteq [0..a_1-15]$.

Proof. The $n = 1$ case is implies by Theorem 2.1. For $n > 1$, consider all followers of the form $1^k \cup G$, $0 \le k \le a_1$, where $G = \bigcup_{i=2}^{n} (a_i) P_{i-1}$. Notice that $g(1^k \cup G) = g(1^k) \oplus g(G) = k \oplus g(G)$.

We know inequalities about the nimber of paths, [1].

$$g(P_n) \le \begin{cases} 3 & \text{if } 2 \le n \le 4 \\ 7 & \text{if } 5 \le n \le 26 \\ 8 & \text{if } 27 \le n \end{cases}$$

Then

$$g(G) \le \begin{cases} 3 & \text{if } 2 \le n \le 4 \\ 7 & \text{if } 5 \le n \le 26 \\ 15 & \text{if } 27 \le n. \end{cases}$$

So we have

$$\begin{aligned} gF(n^{a_n} \dots 1^{a_1}) &\supseteq \{g(1^k \cup G) | 0 \le k \le a_1\} \\ &= g(G) \oplus [0..a_1] \supseteq \begin{cases} [0..a_1 - 1] & \text{if } n = 2 \\ [0..a_1 - 3] & \text{if } n \in [3..5] \\ [0..a_1 - 7] & \text{if } n \in [6..27] \\ [0..a_1 - 15] & \text{if } n \in [28..\infty] \end{cases} \end{aligned}$$

□

4 Stability

Definition Define the operation $+$ between two spiders to be

$$n^{a_n} \dots 1^{a_1} + n^{b_n} \dots 1^{b_1} = n^{a_n + b_n} \dots 1^{a_1 + b_1}$$

Note that this is a vertex contraction on the two root vertices and that it is is associative and commutative.

Definition The spider G is said to be *stable* if all spiders of the form $G + 1^k$ where $k \ge 0$ are champion spiders.

In other words, if a spider is stable then we can add any number of legs of length 1 to the spider and the resulting spider is champion.

Definition Let $f(G)$ be the smallest natural number such that $G + 1^{f(G)}$ is stable. If no such number exists, it is defined as ∞.

Example 4.1. *We have $f(3^1) = 2$. That is, when $a_1 \ge 2$, $3^1 1^{a_1}$ is champion, and $3^1 1^{2-1}$ is not a champion. Additionally, $f(3^1 1^1) = 1$ since $3^1 1^{1+1}$ is champion but $3^1 1^1$ is not.*

4.1 Algorithm for finding $f(G)$

In order to find an algorithm to calculate $f(G)$, we need an algorithm to decide if $G + 1^{a_1}$ is stable. The following theorem provides a method for determining a graph's stability by checking only a finite number of cases. This implies that an algorithm to find $f(G)$ exists.

Theorem 4.2. *Let G be a spider. If $G+1^{a_1}$ is champion for $0 \leq a_1 \leq d$ where $d = 13+e(G)$, then G is stable.*

Proof. To show that $G + 1^{a_1}$ is stable, we must show that $G + 1^{a_1}$ is champion for all $a_1 \geq 0$. We induct on a_1.

Base case We have that $G + 1^{a_1}$ is champion for $0 \leq a_1 \leq d$ by the inductive hypothesis.

Inductive Step Assume the hypothesis is true for all $d < a_1 < m$. Consider the case when $m = a_1$. In order to show $G + 1^{a_1}$ is a champion, we have to show $gF(G + 1^{a_1}) = [0..e(G + 1^{a_1}) - 1]$.

We have

$$\{g(G + 1^k)|0 \leq k < a_1\} = [e(G)..e(G + 1^{a_1 - 1})] = [e(G)..e(G + 1^{a_1}) - 1]$$

and by Lemma 3.7, we know

$$gF(G + 1^{a_1}) \supseteq [0..a_1 - 15] \supseteq [0..d + 1 - 15] = [0..14 + e(G) - 15] = [0..e(G) - 1].$$

Therefore, we can see that $G + 1^{a_1}$ is champion. □

Remark In the above proof, there is a sharper bound for d if we know more information about the length of longest leg in G. In particular, if we let the longest leg in G have length n, we get:

$$d = \begin{cases} e(G) - 1 & \text{if } n \in [1..2] \\ 1 + e(G) & \text{if } n \in [3..5] \\ 5 + e(G) & \text{if } n \in [6..27] \\ 13 + e(G) & \text{if } n \in [28..\infty] \end{cases}$$

This comes from applying Lemma 3.7 slightly differently. For example, suppose that we know that $n \in [3..5]$. Then, by Lemma 3.7, we know that $gF(G+1^{a_1}) \supseteq [0..a_1 - 3] \supseteq [0..d+1-3]$. In order to get this equal to $[0..e(G) - 1]$, we need $d = e(G) + 1$.

If we know that $n \in [1..2]$, then by Lemma 3.7, we know that $gF(G+1^{a_1}) \supseteq [0..a_1 - 1] \supseteq [0..d + 1 - 1]$. In order to get this equal to $[0..e(G) - 1]$, we need $d = e(G) - 1$.

If we know that $n \in [6..27]$, then by Lemma 3.7, we know that $gF(G+1^{a_1}) \supseteq [0..a_1 - 7] \supseteq [0..d + 1 - 7]$. In order to get this equal to $[0..e(G) - 1]$, we need $d = e(G) + 5$.

If we know that $n \in [28..\infty]$, then by Lemma 3.7, we know that $gF(G + 1^{a_1}) \supseteq [0..a_1 - 15] \supseteq [0..d + 1 - 15]$. In order to get this equal to $[0..e(G) - 1]$, we need $d = e(G) + 13$.

Theorem 2.2 and Theorem 2.3 can be proven by showing $f(3^1) = 2$ and $f(4^1) = 4$. These values can be verified by observing $3^1 1^k$ is a champion for $k \in [2..2+3+1] = [2..6]$, and $4^1 1^k$ is a champion for $k \in [4..4 + 4 + 1] = [4..9]$.

Using Theorem 4.2, we have an algorithm, Stable(G), to check if G is stable.

An algorithm for computing $f(G)$ follows by checking if $G + 1^k$ is stable for every k. If the algorithm terminates, then we found $f(G)$. Otherwise $f(G) = \infty$. Note that we have no bounds on $f(G)$, and thus there is no way to determine that $f(G) = \infty$ without running the program for an infinite amount of time.

```
procedure STABLE($G$)
    $bound \leftarrow 13 + e(G)$
    for $k$ from $0 \to bound$ do
        if $e(G+1^k) \neq g(G+1^k)$ then
            return $false$
        end if
    end for
    return $true$
end procedure
```

```
procedure F($G$)
    for $k$ from $0 \to \infty$ do
        if stable($G+1^k$) then
            return $k$
        end if
    end for
end procedure
```

4.2 A result about champions, $f(G)$ and $f(G+2^1)$

Theorem 4.3. *For all spiders G,*

$$f(G+2^1) \leq f(G) + e(G) + 15$$

Proof. For convenience, let $N = e(G) + a_1 + 2$.

If $f(G) = \infty$ then the theorem is trivially true.

Suppose $f(G)$ is finite, and let $c' = f(G) + (N - 2 - a_1) + 15$. We want to show for any $a_1 \geq c'$, $G + 2^1 + 1^{a_1}$ is a champion. If $gF(G + 2^1 + 1^{a_1}) = [0..N-1]$, then $G + 2^1 + 1^{a_1}$ is a champion. Consider the following three cases that will generate all the nimbers in the range.

Case 1 The follower graphs $G + 1^{a_1+1}$ and $G + 1^{a_1}$ have the nimbers $N-1$ and $N-2$ respectively.

Case 2 Consider the set of graphs $X = \{G + 1^{f(G)} \cup P_1, G + 1^{f(G)+1} \cup P_1, \ldots, G + 1^{a_1} \cup P_1\}$. Each one is a follower of $G + 2^1 + 1^{a_1}$. Note that all of these graphs are champion spiders union a path of length 1. Thus, we get

$$\{g(x)|x \in X\} = 1 \oplus [f(G) + e(G)..e(G) + 2 + a_i] = 1 \oplus [c' - 15..N-2] \supseteq [c' - 14..N-3]$$

by Lemma 3.3.

Case 3 Using Lemma 3.7, we know

$$gF(G + 2^1 + 1^{a_1}) \supseteq [0..a_1 - 15] \supseteq [0..c' - 15]$$

The three sets cover the entire interval $[0..N-1]$, therefore $G + 2^1 + 1^{a_1}$ is a champion for all $a_1 \geq c'$.

$$c' = f(G) + (N - 2 - a_1) + 15 = f(G) + e(G) + 15 \geq f(G + 2^1)$$

. ☐

Thus, we have a bound on $f(G + 2^1)$ in terms of $f(G)$. It would be useful to have a similar theorem for length 3 legs.

Theorem 4.4. *Let G be a spider. If*

$$\begin{aligned} a_1 &\geq \max(f(G), f(G + 2^1)) + e(G) + 16 \\ \textit{and} \quad e(G + 3^1 + 1^{a_1}) &\equiv 2 \textit{ or } 3 \pmod 4, \end{aligned}$$

then $G + 3^1 + 1^{a_1}$ is a champion spider.

Proof. Let $c = \max(f(G), f(G + 2^1))$.

If $c = \infty$, then the theorem is trivially true.

Suppose c is finite. For convenience, let $N = e(G + 3^1 + 1^{a_1})$. Let

$$a_1 \geq \max(f(G), f(G + 2^1)) + 16 + e(G)$$

and let $c' = c + (N - a_1 - 3) + 16$. If $g(G + 3^1 + 1^{a_1}) = N$, then the graph is a champion. We want to show for any $a_1 \geq c'$ and $N \equiv 2$ or $3 \pmod 4$, $G + 3^1 + 1^{a_1}$ is a champion.

Let $a_1 \geq c'$. We want to show that $gF(G + 3^1 + 1^{a_1}) = [0..N-1]$. Consider 4 possible cases that will generate all the nimbers in the range.

Case 1 The follower graphs $G + 2^1 + 1^{a_1}$ and $G + 1^{a_1+1}$ generate $[N-2..N-1]$.

Case 2 Consider the set of graphs $X = \{G + 1^c \cup P_2, G + 1^{c+1} \cup P_2, \ldots, G + 1^{a_1} \cup P_2\}$. Each one is a follower of $G + 3^1 + 1^{a_1}$. Note that these graphs are champion spiders union a path of length 2. Thus, we get

$$\{g(x) | x \in X\} = 2 \oplus [c + N - 3 - a_1..N-3] = 2 \oplus [c' - 16..N-3] \supseteq [c' - 14..N-5]$$

by Lemma 3.3.

Case 3 $gF(G + 3^1 + 1^{a_1}) \supseteq [0..a_1 - 15] \supseteq [0..c' - 15]$

Case 4 Notice that $[N-4..N-3]$ has not yet been generated. Consider the following two cases:

If $N \equiv 2 \mod 4$,

$$\begin{cases} g(G + 1^{a_1-2} \cup P_2) & \text{generates } N - 3 \\ g(G + 1^{a_1-3} \cup P_2) & \text{generates } N - 4. \end{cases}$$

If $N \equiv 3 \mod 4$,

$$\begin{cases} g(G + 1^{a_1+1} \cup P_1) & \text{generates } N - 3 \\ g(G + 1^{a_1-3} \cup P_2) & \text{generates } N - 4. \end{cases}$$

In either case, this shows that $[N-4..N-3]$ is generated by some followers. Thus, we know that $g(G + 3^1 + 1^{a_1})$ are champions for all $a_1 \geq c'$ and $N \equiv 2$ or $3 \pmod 4$. ☐

5 Periodicity in the Discrepancy

Definition The discrepancy of a graph G, $d(G)$, is defined as

$$d(G) = e(G) - g(G)$$

Notice that champion spiders are precisely the spiders with discrepancy 0.

We get the following property for the discrepancy. Let G, G', H and H' be graphs. If $d(G) = d(G')$ and $d(H) = d(H')$, then $d(G \cup H) = d(G' \cup H')$.

Definition Let G be a spider. Let $c(G)$ be the smallest natural number such that for all $x \geq c(G)$,

$$d(G + 1^x) = d(G + 1^{x+p(G)})$$

where p(G) is the smallest period of the sequence $d(G + 1^x)$, $x \geq c(G)$.

We can think of $c(G)$ as the number of new length 1 legs that a spider needs to begin showing periodic behavior in the discrepancy. Notice that by definition,

$$f(G) = k \text{ if and only if } c(G) = k, p(G) = 1, \text{ and } d(G + 1^k) = 0.$$

We can use the concept of discrepancy to restate the notion in Theorem 2.4.

Theorem 5.1. *Consider the graph 5^1. The following are true:*

- $c(5^1) = 8$
- $p(5^1) = 4$
- $(d(5^1 1^8), d(5^1 1^9), d(5^1 1^{10}), d(5^1 1^{11})) = (0, 3, 5, 0)$.

Can we find and prove values for $c(G)$ and $p(G)$? If so, then Theorem 2.4 can be proven by verifying that $c(5^1) = 8$ and $p(5^1) = 4$. In this section, we hope to develop a theorem that gives us an effective way to compute $c(G)$ and $p(G)$.

Definition Let L be the set of spiders without any legs of length 1.

Theorem 5.2. *Let $G \in L$ and fix $c, p \in \mathbb{N}$. Let $m = \max(\{d(G + 1^k) | c \leq k < c + p\} \cup \{e(G) + 13\})$. If the following conditions hold, then $c(G) \leq c$ and $p(G) \leq p$.*

1. *For all $c \leq x \leq c + p + m$,*
$$d(G + 1^x) = d(G + 1^{x+p}).$$
2. $\max(\{c(G') | G' \subset G, G' \in L\}) \leq c$.
3. $\mathrm{lcm}(\{p(G') | G' \subset G, G' \in L\}) | p$.

Notice that we choose m in this theorem so that $d(G + 1^k) \leq m$ for all $c \leq k < c + p$. Thus, $e(G + 1^k) - g(G + 1^k) \leq m$, so we have $g(G + 1^k) \geq e(G + 1^k) - m$.

This theorem allows us to find $c(5^1)$ and $p(5^1)$ mechanically, and thus prove theorems similar to Theorem 5.1.

Proof of Theorem 5.1. It is known that $c(4^1) = 4, p(4^1) = 1$ by Theorem 2.3, $c(3^1) = 2, p(3^1) = 1$ by Theorem 2.2, and $c(2^1) = 0, p(2^1) = 1$ by Theorem 3.1. Let $c = 8$ and $p = 4$.

$$\begin{aligned}
m &= \max(\{d(G+1^k)|c \leq k < c+p\} \cup \{e(G)+13\}) \\
&= \max(\{d(G+1^c), d(G+1^{c+1}), \ldots, d(G+1^{c+p-1})\} \cup \{18\}) \\
&= \max(\{d(G+1^8), d(G+1^9), d(G+1^{10}), d(G+1^{11})\} \cup \{18\}) \\
&= \max(\{13 - g(G+1^8), 14 - g(G+1^9), 15 - g(G+1^{10}), 16 - g(G+1^{11}), 18\} \\
&= \max(\{0, 3, 5, 18\}) \\
&= 18
\end{aligned}$$

We need to check if $d(5^1 1^k)$ has a period of 4 starting at $k = 8$. In other words, we need to show that $c(G) \leq c$ and $p(G)|p$.

1. By the data in Appendix A.2, $d(5^1 1^k) = d(5^1 1^{k+4})$ for all $8 \leq k \leq 8 + 4 + 18 = 30$.
2. $8 \geq \max(\{c(4^1), c(3^1), c(2^1)\}) = 4$
3. 4 is a multiple of $\text{lcm}(\{p(4^1), p(3^1), p(2^1)\}) = 1$.

By Theorem 5.2, $c(5^1) \leq 8$ and $p(5^1) \leq 4$. Since $d(5^1 1^7) \neq d(5^1 1^{11})$, we know that $c(G) > 7$, so $c(G) = 8$. Since $d(5^1 1^8) \neq d(5^1 1^{10})$, we know $p(G) \neq 2$ and since $d(5^1 1^8) \neq d(5^1 1^9)$ we know $p(G) \neq 1$ and since $p(G)|4$, we get that $p(G) = 4$. □

Proof of Theorem 5.2. . To show that $c(G) \leq c$ and $p(G) \leq p$, we must show that for all $x \geq c$,

$$d(G+1^x) = d(G+1^{x+p})$$

The desired relation between the discrepancy can be expressed as a relation between the nimbers.

$$d(G+1^x) = d(G+1^{x-p}) \text{ if and only if } g(G+1^x) = g(G+1^{x-p}) + p$$

We will use induction to show that $g(G+1^x) = g(G+1^{x-p}) + p$.

Proof by induction on x.

Base Case: The theorem holds for $c \leq x \leq c+p+m$ by hypothesis.
Inductive Step: Assume the hypothesis is true for all $c+p+m < x < t$. Consider $x = t$. This shows $x - m > c + p$. Note: in this proof, we will use G' to be a subgraph of a spider G with $G' \in L$ and we will use P to denote some finite set of disjoint paths.

Let

$$X = \{G' + 1^k \cup P | c+p \leq k \leq x,\ G' + 1^k \cup P \in F(G+1^x)\}.$$

and let

$$Y = \{G' + 1^{k-p} \cup P | G' + 1^k \cup P \in X\}.$$

By this definition, we see that X contains graphs that consist of spider graphs that are covered by either the inductive hypothesis or the conditions in the theorem. Y contains the set of spiders for which if you add p legs of length 1 to each, you get an element of X. We also define $\overline{X} = F(G+1^x) \backslash X$ and $\overline{Y} = F(G+1^{x-p}) \backslash Y$. Notice that by definition, $X \subseteq F(G+1^x)$ and $Y \subseteq F(G+1^{x-p})$.

We need to show the following:

1. $[0..g(G+1^{x-k})+p-1] \subseteq gF(G+1^x)$
2. $g(G+1^{x-p})+p \notin gF(G+1^x)$

We will show 1 by showing that Y must generate nimbers in $[e(G)+c..g(G+1^{x-p})-1]$, and thus X must generate nimbers in $[e(G)+c+p..g(G+1^{x-p})+p-1]$. We will show 2 by showing that nothing in X can have the nimber $g(G+1^{x-p})+p$ by contradiction and that the nimber of everything in $\overline{X}$ is less than $g(G+1^{x-p})+p$.

By the property discussed at the beginning of Section 5, we have for every element $G'+1^k \cup P \in X$, $e(G')+e(P) \leq e(G)$ and

$$d(G'+1^k \cup P) = d(G'+1^{k-p} \cup P)$$

or equivalently

$$g(G'+1^k \cup P) = g(G'+1^{k-p} \cup P)+p. \quad (*)$$

Proof that $[0..g(G+1^{x-p})+p-1] \subseteq gF(G+1^x)$ Using $(*)$, we know if $G'+1^{k-p} \cup P \in Y$, then we get $g(G'+1^{k-p} \cup P)+p \in gF(G+1^x)$.

The nimbers generated by Y give us information about the nimbers generated by X. In fact, if $\{g(H)|H \in Y\} \supseteq [a..b]$, then $\{g(H')|H' \in X\} \supseteq [a+p..b+p]$.

We know the nimbers generated by $Y \cup \overline{Y} = F(G+1^{x-p})$ covers $[0..g(G+1^{x-p})-1]$. Consider what can be generated by only $\overline{Y}$. Since for every $H \in \overline{Y}$, $e(H) < e(G+1^c) = e(G)+c$, $\overline{Y}$ can only generate elements in $[0..e(G)+c-1]$. Then Y must generate $[e(G)+c..g(G+1^{x-p})-1]$. Thus X must generate $[e(G)+c+p..g(G+1^{x-p})+p-1]$.

Lemma 3.7 shows

$$gF(G+1^x) \supseteq [0..x-15] \supseteq [0..(e(G)+c+p+13+1)-15] = [0..e(G)+c+p-1]$$

Thus $[0..g(G+1^{x-p})+p-1] \subseteq gF(G+1^x)$.

Proof that $g(G+1^{x-p})+p \notin gF(G+1^x)$ We must show that there is no graph H in X or $\overline{X}$ such that $g(H) = g(G+1^{x-p})$. First, consider the followers in X. Let $G'+1^k \cup P \in X$. First, we will show that no member of X has the nimber $g(G+1^{x-p})+p$. In other words,

$$g(G'+1^k \cup P) \neq g(G+1^{x-p})+p.$$

To find a contradiction, assume this is false. Then, there exist $G'+1^k \cup P \in X$, such that

$$\begin{aligned} g(G+1^{x-p}) &= g(G'+1^k \cup P)-p & (1)\\ &= g(G'+1^{k-p} \cup P) & (2)\\ &\in gF(G+1^{x-p}) & (3) \end{aligned}$$

where (2) is true by the inductive hypothesis if $G' = G$ or by the conditions required by the theorem if $G' \subset G$. This is a contradiction, since for any graph H, $g(H) \notin gF(H)$. Thus $g(G+1^{x-p})$ is not generated by a member of X.

Second, we will show that no member of $\overline{X}$ can have the nimber $g(G+1^{x-p})+p$ by showing that if $H \in \overline{X}$, then $g(H) < g(G+1^{x-p})+p$.

We show this bound by showing $e(H) < g(G+1^{x-p}) + p$. Note that by the definition of $\overline{X}$, H is of the form $G' + 1^k \cup P$, where $k < c+p < x-m$. Since $e(G'+1^k \cup P) \leq e(G+1^k)$, we know

$$\begin{aligned} e(H) &\leq e(G+1^{c+p}) & (4)\\ &= e(G) + c + p & (5)\\ &< e(G) + x - m. & (6) \end{aligned}$$

Recall that $g(G+1^k) \geq e(G+1^k) - m$. Thus, $g(G+1^{x-p}) \geq e(G+1^{x-p}) - m = e(G) + x - p - m$. Combining (3) and (6), we get

$$g(G+1^{x-p}) + p \geq e(G) + x - m > e(H) \geq g(H)$$

So, we get $g(G+1^{x-p}) + p \notin X \cup \overline{X} = gF(G+1^x)$.

Thus, we have $g(G+1^{x-p}) + p \notin gF(G+1^x)$ and $[0..g(G+1^{x-p}) + p - 1] \subseteq gF(G+1^x)$, which imply that

$$g(G+1^x) = g(G+1^{x-p}) + p$$

which means

$$d(G+1^x) = d(G+1^{x+p})$$

so finally, we know $c(G) \leq c$ and $p(G) \leq p$.

□

Notice that although this theorem only considered spiders with no legs of length 1, it can be extended to spiders by noting $c(G+1^k) = \max(c(G)-k, 0)$ and $p(G+1^k) = p(G)$ where $G \in L$. Thus we can easily apply this theorem to spiders with any legs.

6 Conjectures on Champion Spiders

6.1 $3^{a_3}2^{a_2}1^{a_1}$

Conjecture 6.1. *$3^{a_3}2^{a_2}1^{a_1}$ is a champion spider if $a_2 > 0$ and for all $3^i 2^j \subseteq 3^{a_3}2^{a_2}$, $a_1 > f(3^i 2^j) + 15$ where $f(3^i 2^j)$ is defined in Section 4.*

Our attempted proof was by induction on a_3 and a_2. By Theorem 3.1, we know that $2^{a_2}1^{a_1}$ is champion for all $a_1 \geq 2a_2 - 2$. Thus, $f(3^0 2^{a_2}) = 2a_2 - 2$ which exists for all a_2. We can finitely compute $f(3^a 2^b)$, so our base case consisted of $f(3^1 2^0) = 2$ and $f(3^1 2^1) = 4$ and assumed that we could build up from there in our inductive step. This turns out to be inadequate as a base case, since we don't have any bounds on $f(3^{a+1}2^b)$ in terms of $f(3^a 2^b)$. Thus in order for our proof to be sufficient, we need to know the value of $f(3^a 2^b)$ for all $a \in \mathbb{N}$ in our base case, which is impossible without a nice, evaluatable expression for $f(3^a 2^b)$. Theorem 4.4 works toward a bound on $f(3^{a+1}2^b)$.

Our inductive step was very similar to the proof of Theorem 3.1. We broke the interval $[0..3a_3 + 2a_2 + 1a_1 - 1]$ into the subintervals

$$\begin{gathered} [0 \ldots 5 + f(3^1 2^1)], \\ [6 + f(3^1 2^1) \ldots 3a_3 + 2a_2 + a_1 - 3], \\ [3a_3 + 2a_2 + a_1 - 2..3a_3 + 2a_2 + a_1 - 1] \end{gathered}$$

```
procedure C($G + 1^k$)
    $c, p \leftarrow$ findcp(G)                          ▷ $G \in L$ in all the functions.
    return $\max(0, c - k)$
end procedure
procedure P($G + 1^k$)
    $c, p \leftarrow$ findcp(G)
    return p
end procedure
procedure FINDCP(G)
    $c_0 \leftarrow 0$
    $p_0 \leftarrow 1$
    for all $G' < G$ and $G' \in L$ do
        $c_0 \leftarrow \max(c(G'), c_0)$
        $p_0 \leftarrow \mathrm{lcm}(p(G'), p_0)$
    end for
    for all $c \geq c_0, p_0|p$ do
        $m \leftarrow 0$
        for all $c \leq k < c + p$ do
            $m \leftarrow \max(d(G + 1^k), m)$
        end for
        $m \leftarrow \max(m, e(G) + 15)$
        $success \leftarrow true$
        for all $c \leq x \leq c + p + m$ do
            if $d(G + 1^x) \neq d(G + 1^{x+p})$ then
                $success \leftarrow false$
            end if
        end for
        if success then
            return findboundedcp$(G, c, p)$
        end if
    end for
end procedure
procedure FINDBOUNDEDCP(G,c',p')
    for $c$ from 0 to $c'$ do
        If $d(G + 1^x) = d(G + 1^{x+p'})$ for all $c \leq x \leq c' + p'$, stop.
    end for
    for all $p$ from 0 to $p'$ do
        If $d(G + 1^x) = d(G + 1^{x+p})$ for all $c \leq x \leq c + p'$, stop.
    end for
    return $c, p$
end procedure
```

and specified followers that achieve all of the numbers in each set. However, we realized that our proof assumed that $a_2 > 1$ for all followers, which is not the case. When we noticed this error, we began working on proving the following conjecture.

Conjecture 6.2. *Let $a_3 \geq 10$. For*

$$a_1 \geq \begin{cases} 3a_3 - 7 & \text{if } a_3 \text{ even} \\ 3a_3 - 6 & \text{if } a_3 \text{ odd,} \end{cases}$$

$3^{a_3}1^{a_1}$ is champion.

We formed this conjecture by looking at values of $d(3^{a_3}1^{a_1})$, which are contained in Appendix A.1. After a certain number of 1s, we noticed that the discrepancy became zero and stayed there for larger values of a_1, indicating that these graphs become stable after a point.

Conjecture 6.2 can be shown true for a given a_3 by applying a refined version of Theorem 4.2. This theorem states that we only need to verify if $d(3^{a_3}1^k) = 0$ for all $3a_3 - 6 \leq k \leq 6a_3 - 5$ if a_3 is odd, and if $d(3^{a_3}1^k) = 0$ for all $3a_3 - 7 \leq k \leq 6a_3 - 6$ if a_3 is even. We have done this for all $a_3 \leq 30$.

In fact, if we call m the number of ones for which the graphs become stable, we found predictable behavior of $d(3^{a_3}1^{m-1}), d(3^{a_3}1^{m-2}), \ldots, d(3^{a_3}1^{m-5})$ for $a_3 \geq 12$. In particular, we have the following conjecture.

Conjecture 6.3. *Let $a_3 \geq 12$. For*

$$a_1 = \begin{cases} 3a_3 - 7 & \text{if } a_3 \text{ even} \\ 3a_3 - 6 & \text{if } a_3 \text{ odd,} \end{cases}$$

$3^{a_3}1^{a_1}$ is stable if and only if we get the following:

$$\text{For } a_3 \text{ even, } \begin{cases} d(3^{a_3}1^{3a_3-8}) = 3 \\ d(3^{a_3}1^{3a_3-9}) = 0 \\ d(3^{a_3}1^{3a_3-10}) = 4 \\ d(3^{a_3}1^{3a_3-11}) = 4 \\ d(3^{a_3}1^{3a_3-12}) = 8 \end{cases} \qquad \text{For } a_3 \text{ odd, } \begin{cases} d(3^{a_3}1^{3a_3-7}) = 5 \\ d(3^{a_3}1^{3a_3-8}) = 0 \\ d(3^{a_3}1^{3a_3-9}) = 0 \\ d(3^{a_3}1^{3a_3-10}) = 3 \\ d(3^{a_3}1^{3a_3-11}) = 5 \end{cases}$$

Example 6.4. *Let $a_3 = 15$. Then, let $a_1 = 3a_3 - 6 = 39$. If we calculate $d(3^{a_3}1^{a_3-6-i})$ for $i \in [1, 2, 3, 4, 5]$, we get the desired values 5,0,0,3,5. This means that $3^{15}1^{39}$ is champion. Alternatively, if we examine a spider of the form 3^i1^j that we know to be champion that satisfies the requirements on i and j, we can look it up in the table and if the conjecture is true, we will see the forementioned pattern in the discrepancies of smaller graphs.*

It seems like the larger we force a_3 to be, we can predict $d(3^{a_3}1^{m-i})$ for larger i.

6.2 $4^{a_4}3^{a_3}2^{a_2}1^{a_1}$

Conjecture 6.5. *$f(4^{a_4}3^{a_3}2^{a_2})$ is finite.*

We attempted to prove this in the same manner as Conjecture 6.2, but with induction on a_4, a_3, and a_2. We discovered similar problems: we would need base cases for graphs of the form $4^i3^j1^{f(3^i2^j)}$ for all $i, j \in \mathbb{N}$. The proof also incorrectly assumes that a_4, a_3, a_2 are always greater than 1 for all followers.

7 Conclusion

We have developed methods for computing the Sprague-Grundy numbers of certain types of spider graphs. Ultimately, our results discuss only a small subset of these graphs. We hope in the future to be able to extend our results using the new methods presented in Section 5. In addition, we plan to further work on the conjectures in Section 6 and show that the graphs $4^{a_4}3^{a_3}2^{a_2}1^{a_1}$ and $3^{a_3}2^{a_2}1^{a_1}$ become stable. Additionally, we would like to be able to say something about the length of the predictable pattern in values of the discrepancy as we force a_3 to be larger, which we mention after Conjecture 6.3. It is possible that this pattern before acheiving stability could also be generalized to other spider graphs.

Acknowledgements

We would like to thank Neil J. Calkin and Janine E. Janoski for their invaluable guidance and support this summer. Without their knowledge we would not have made the progress that we did. In addition, we would also like to thank the National Science Foundation for providing funding this summer through grants DMS-1156761 and DMS-1156734 and to Clemson University for hosting us. Lastly, we would like to thank Jim Brown, Kevin James, and Mohammed Tessema for leading the program this summer.

References

[1] Neil J. Calkin, Kevin James, Janine E. Janoski, Sarah Leggett, Bryce Richards, Nathan Sitaraman, and Stephanie Thomas. Computing strategies for graphical nim. 2009.

[2] John H. Conway. *On numbers and games.* A.K. Peters, Natick, Mass, 2001.

[3] John H. Conway. *Winning ways for your mathematical plays.* A.K. Peters, 2001.

[4] Mikio Kano. Edge-removing games of star type. *Discrete Mathematics*, 151(13):113 – 119, 1996.

[5] Dierk Schleicher and Michael Stoll. An introduction to Conway's games and numbers. *Moscow Math Journal*, 6(2):359–388, 2006.

A Computational Data for $g(4^{a_4}3^{a_3}2^{a_2}1^{a_1})$

List of all $g(4^{a_4}3^{a_3}2^{a_2}1^{a_1})$ where $a_4 = 0$, $a_3 \leq 2$, $a_2 \leq 3$, $a_1 \leq 100$.
It is in the format $[a_4, a_3, a_2, a_1]$ $g(4^{a_4}3^{a_3}2^{a_2}1^{a_1})$.

```
[0, 0, 0, 0] 0
[0, 0, 0, 1] 1
[0, 0, 0, 2] 2
[0, 0, 0, 3] 3
[0, 0, 0, 4] 4
[0, 0, 0, 5] 5
[0, 0, 0, 6] 6
[0, 0, 0, 7] 7
[0, 0, 0, 8] 8
[0, 0, 0, 9] 9
[0, 0, 0, 10] 10
[0, 0, 0, 11] 11
[0, 0, 0, 12] 12
[0, 0, 0, 13] 13
[0, 0, 0, 14] 14
[0, 0, 0, 15] 15
[0, 0, 0, 16] 16
[0, 0, 0, 17] 17
[0, 0, 0, 18] 18
[0, 0, 0, 19] 19
[0, 0, 0, 20] 20
[0, 0, 0, 21] 21
[0, 0, 0, 22] 22
[0, 0, 0, 23] 23
[0, 0, 0, 24] 24
[0, 0, 0, 25] 25
[0, 0, 0, 26] 26
[0, 0, 0, 27] 27
[0, 0, 0, 28] 28
[0, 0, 0, 29] 29
[0, 0, 0, 30] 30
[0, 0, 0, 31] 31
[0, 0, 0, 32] 32
[0, 0, 0, 33] 33
[0, 0, 0, 34] 34
[0, 0, 0, 35] 35
[0, 0, 0, 36] 36
[0, 0, 0, 37] 37
[0, 0, 0, 38] 38
[0, 0, 0, 39] 39
[0, 0, 0, 40] 40
[0, 0, 0, 41] 41
[0, 0, 0, 42] 42
[0, 0, 0, 43] 43
[0, 0, 0, 44] 44
[0, 0, 0, 45] 45
[0, 0, 0, 46] 46
[0, 0, 0, 47] 47
[0, 0, 0, 48] 48
[0, 0, 0, 49] 49
[0, 0, 0, 50] 50
[0, 0, 0, 51] 51
[0, 0, 0, 52] 52
[0, 0, 0, 53] 53
[0, 0, 0, 54] 54
[0, 0, 0, 55] 55
[0, 0, 0, 56] 56
[0, 0, 0, 57] 57
[0, 0, 0, 58] 58
[0, 0, 0, 59] 59
[0, 0, 0, 60] 60
[0, 0, 0, 61] 61
[0, 0, 0, 62] 62
[0, 0, 0, 63] 63
[0, 0, 0, 64] 64
[0, 0, 0, 65] 65
[0, 0, 0, 66] 66
[0, 0, 0, 67] 67
[0, 0, 0, 68] 68
[0, 0, 0, 69] 69
[0, 0, 0, 70] 70
[0, 0, 0, 71] 71
[0, 0, 0, 72] 72
[0, 0, 0, 73] 73
[0, 0, 0, 74] 74
[0, 0, 0, 75] 75
[0, 0, 0, 76] 76
[0, 0, 0, 77] 77
[0, 0, 0, 78] 78
[0, 0, 0, 79] 79
[0, 0, 0, 80] 80
[0, 0, 0, 81] 81
[0, 0, 0, 82] 82
[0, 0, 0, 83] 83
[0, 0, 0, 84] 84
[0, 0, 0, 85] 85
[0, 0, 0, 86] 86
[0, 0, 0, 87] 87
[0, 0, 0, 88] 88
[0, 0, 0, 89] 89
[0, 0, 0, 90] 90
[0, 0, 0, 91] 91
[0, 0, 0, 92] 92
[0, 0, 0, 93] 93
[0, 0, 0, 94] 94
[0, 0, 0, 95] 95
[0, 0, 0, 96] 96
[0, 0, 0, 97] 97
[0, 0, 0, 98] 98
[0, 0, 0, 99] 99
[0, 0, 0, 100] 100
[0, 0, 1, 0] 2
[0, 0, 1, 1] 3
[0, 0, 1, 2] 4
[0, 0, 1, 3] 5
[0, 0, 1, 4] 6
[0, 0, 1, 5] 7
[0, 0, 1, 6] 8
[0, 0, 1, 7] 9
[0, 0, 1, 8] 10
[0, 0, 1, 9] 11
[0, 0, 1, 10] 12
[0, 0, 1, 11] 13
[0, 0, 1, 12] 14
[0, 0, 1, 13] 15
[0, 0, 1, 14] 16
[0, 0, 1, 15] 17
[0, 0, 1, 16] 18
[0, 0, 1, 17] 19
[0, 0, 1, 18] 20
[0, 0, 1, 19] 21
[0, 0, 1, 20] 22
[0, 0, 1, 21] 23
[0, 0, 1, 22] 24
[0, 0, 1, 23] 25
[0, 0, 1, 24] 26
[0, 0, 1, 25] 27
[0, 0, 1, 26] 28
[0, 0, 1, 27] 29
[0, 0, 1, 28] 30
[0, 0, 1, 29] 31
[0, 0, 1, 30] 32
[0, 0, 1, 31] 33
[0, 0, 1, 32] 34
[0, 0, 1, 33] 35
[0, 0, 1, 34] 36
[0, 0, 1, 35] 37
[0, 0, 1, 36] 38
[0, 0, 1, 37] 39
[0, 0, 1, 38] 40
[0, 0, 1, 39] 41
[0, 0, 1, 40] 42
[0, 0, 1, 41] 43
[0, 0, 1, 42] 44
[0, 0, 1, 43] 45
[0, 0, 1, 44] 46
[0, 0, 1, 45] 47
[0, 0, 1, 46] 48
[0, 0, 1, 47] 49
[0, 0, 1, 48] 50
[0, 0, 1, 49] 51
[0, 0, 1, 50] 52
[0, 0, 1, 51] 53
[0, 0, 1, 52] 54
[0, 0, 1, 53] 55
[0, 0, 1, 54] 56
[0, 0, 1, 55] 57
[0, 0, 1, 56] 58
[0, 0, 1, 57] 59
[0, 0, 1, 58] 60
```

```
[0, 0, 1, 59] 61    [0, 0, 2, 3] 7      [0, 0, 2, 48] 52    [0, 0, 2, 93] 97
[0, 0, 1, 60] 62    [0, 0, 2, 4] 8      [0, 0, 2, 49] 53    [0, 0, 2, 94] 98
[0, 0, 1, 61] 63    [0, 0, 2, 5] 9      [0, 0, 2, 50] 54    [0, 0, 2, 95] 99
[0, 0, 1, 62] 64    [0, 0, 2, 6] 10     [0, 0, 2, 51] 55    [0, 0, 2, 96] 100
[0, 0, 1, 63] 65    [0, 0, 2, 7] 11     [0, 0, 2, 52] 56    [0, 0, 2, 97] 101
[0, 0, 1, 64] 66    [0, 0, 2, 8] 12     [0, 0, 2, 53] 57    [0, 0, 2, 98] 102
[0, 0, 1, 65] 67    [0, 0, 2, 9] 13     [0, 0, 2, 54] 58    [0, 0, 2, 99] 103
[0, 0, 1, 66] 68    [0, 0, 2, 10] 14    [0, 0, 2, 55] 59    [0, 0, 2, 100] 104
[0, 0, 1, 67] 69    [0, 0, 2, 11] 15    [0, 0, 2, 56] 60    [0, 0, 3, 0] 3
[0, 0, 1, 68] 70    [0, 0, 2, 12] 16    [0, 0, 2, 57] 61    [0, 0, 3, 1] 7
[0, 0, 1, 69] 71    [0, 0, 2, 13] 17    [0, 0, 2, 58] 62    [0, 0, 3, 2] 5
[0, 0, 1, 70] 72    [0, 0, 2, 14] 18    [0, 0, 2, 59] 63    [0, 0, 3, 3] 9
[0, 0, 1, 71] 73    [0, 0, 2, 15] 19    [0, 0, 2, 60] 64    [0, 0, 3, 4] 10
[0, 0, 1, 72] 74    [0, 0, 2, 16] 20    [0, 0, 2, 61] 65    [0, 0, 3, 5] 11
[0, 0, 1, 73] 75    [0, 0, 2, 17] 21    [0, 0, 2, 62] 66    [0, 0, 3, 6] 12
[0, 0, 1, 74] 76    [0, 0, 2, 18] 22    [0, 0, 2, 63] 67    [0, 0, 3, 7] 13
[0, 0, 1, 75] 77    [0, 0, 2, 19] 23    [0, 0, 2, 64] 68    [0, 0, 3, 8] 14
[0, 0, 1, 76] 78    [0, 0, 2, 20] 24    [0, 0, 2, 65] 69    [0, 0, 3, 9] 15
[0, 0, 1, 77] 79    [0, 0, 2, 21] 25    [0, 0, 2, 66] 70    [0, 0, 3, 10] 16
[0, 0, 1, 78] 80    [0, 0, 2, 22] 26    [0, 0, 2, 67] 71    [0, 0, 3, 11] 17
[0, 0, 1, 79] 81    [0, 0, 2, 23] 27    [0, 0, 2, 68] 72    [0, 0, 3, 12] 18
[0, 0, 1, 80] 82    [0, 0, 2, 24] 28    [0, 0, 2, 69] 73    [0, 0, 3, 13] 19
[0, 0, 1, 81] 83    [0, 0, 2, 25] 29    [0, 0, 2, 70] 74    [0, 0, 3, 14] 20
[0, 0, 1, 82] 84    [0, 0, 2, 26] 30    [0, 0, 2, 71] 75    [0, 0, 3, 15] 21
[0, 0, 1, 83] 85    [0, 0, 2, 27] 31    [0, 0, 2, 72] 76    [0, 0, 3, 16] 22
[0, 0, 1, 84] 86    [0, 0, 2, 28] 32    [0, 0, 2, 73] 77    [0, 0, 3, 17] 23
[0, 0, 1, 85] 87    [0, 0, 2, 29] 33    [0, 0, 2, 74] 78    [0, 0, 3, 18] 24
[0, 0, 1, 86] 88    [0, 0, 2, 30] 34    [0, 0, 2, 75] 79    [0, 0, 3, 19] 25
[0, 0, 1, 87] 89    [0, 0, 2, 31] 35    [0, 0, 2, 76] 80    [0, 0, 3, 20] 26
[0, 0, 1, 88] 90    [0, 0, 2, 32] 36    [0, 0, 2, 77] 81    [0, 0, 3, 21] 27
[0, 0, 1, 89] 91    [0, 0, 2, 33] 37    [0, 0, 2, 78] 82    [0, 0, 3, 22] 28
[0, 0, 1, 90] 92    [0, 0, 2, 34] 38    [0, 0, 2, 79] 83    [0, 0, 3, 23] 29
[0, 0, 1, 91] 93    [0, 0, 2, 35] 39    [0, 0, 2, 80] 84    [0, 0, 3, 24] 30
[0, 0, 1, 92] 94    [0, 0, 2, 36] 40    [0, 0, 2, 81] 85    [0, 0, 3, 25] 31
[0, 0, 1, 93] 95    [0, 0, 2, 37] 41    [0, 0, 2, 82] 86    [0, 0, 3, 26] 32
[0, 0, 1, 94] 96    [0, 0, 2, 38] 42    [0, 0, 2, 83] 87    [0, 0, 3, 27] 33
[0, 0, 1, 95] 97    [0, 0, 2, 39] 43    [0, 0, 2, 84] 88    [0, 0, 3, 28] 34
[0, 0, 1, 96] 98    [0, 0, 2, 40] 44    [0, 0, 2, 85] 89    [0, 0, 3, 29] 35
[0, 0, 1, 97] 99    [0, 0, 2, 41] 45    [0, 0, 2, 86] 90    [0, 0, 3, 30] 36
[0, 0, 1, 98] 100   [0, 0, 2, 42] 46    [0, 0, 2, 87] 91    [0, 0, 3, 31] 37
[0, 0, 1, 99] 101   [0, 0, 2, 43] 47    [0, 0, 2, 88] 92    [0, 0, 3, 32] 38
[0, 0, 1, 100] 102  [0, 0, 2, 44] 48    [0, 0, 2, 89] 93    [0, 0, 3, 33] 39
[0, 0, 2, 0] 1      [0, 0, 2, 45] 49    [0, 0, 2, 90] 94    [0, 0, 3, 34] 40
[0, 0, 2, 1] 5      [0, 0, 2, 46] 50    [0, 0, 2, 91] 95    [0, 0, 3, 35] 41
[0, 0, 2, 2] 6      [0, 0, 2, 47] 51    [0, 0, 2, 92] 96    [0, 0, 3, 36] 42
```

```
[0, 0, 3, 37] 43   [0, 0, 3, 82] 88    [0, 1, 0, 26] 29   [0, 1, 0, 71] 74
[0, 0, 3, 38] 44   [0, 0, 3, 83] 89    [0, 1, 0, 27] 30   [0, 1, 0, 72] 75
[0, 0, 3, 39] 45   [0, 0, 3, 84] 90    [0, 1, 0, 28] 31   [0, 1, 0, 73] 76
[0, 0, 3, 40] 46   [0, 0, 3, 85] 91    [0, 1, 0, 29] 32   [0, 1, 0, 74] 77
[0, 0, 3, 41] 47   [0, 0, 3, 86] 92    [0, 1, 0, 30] 33   [0, 1, 0, 75] 78
[0, 0, 3, 42] 48   [0, 0, 3, 87] 93    [0, 1, 0, 31] 34   [0, 1, 0, 76] 79
[0, 0, 3, 43] 49   [0, 0, 3, 88] 94    [0, 1, 0, 32] 35   [0, 1, 0, 77] 80
[0, 0, 3, 44] 50   [0, 0, 3, 89] 95    [0, 1, 0, 33] 36   [0, 1, 0, 78] 81
[0, 0, 3, 45] 51   [0, 0, 3, 90] 96    [0, 1, 0, 34] 37   [0, 1, 0, 79] 82
[0, 0, 3, 46] 52   [0, 0, 3, 91] 97    [0, 1, 0, 35] 38   [0, 1, 0, 80] 83
[0, 0, 3, 47] 53   [0, 0, 3, 92] 98    [0, 1, 0, 36] 39   [0, 1, 0, 81] 84
[0, 0, 3, 48] 54   [0, 0, 3, 93] 99    [0, 1, 0, 37] 40   [0, 1, 0, 82] 85
[0, 0, 3, 49] 55   [0, 0, 3, 94] 100   [0, 1, 0, 38] 41   [0, 1, 0, 83] 86
[0, 0, 3, 50] 56   [0, 0, 3, 95] 101   [0, 1, 0, 39] 42   [0, 1, 0, 84] 87
[0, 0, 3, 51] 57   [0, 0, 3, 96] 102   [0, 1, 0, 40] 43   [0, 1, 0, 85] 88
[0, 0, 3, 52] 58   [0, 0, 3, 97] 103   [0, 1, 0, 41] 44   [0, 1, 0, 86] 89
[0, 0, 3, 53] 59   [0, 0, 3, 98] 104   [0, 1, 0, 42] 45   [0, 1, 0, 87] 90
[0, 0, 3, 54] 60   [0, 0, 3, 99] 105   [0, 1, 0, 43] 46   [0, 1, 0, 88] 91
[0, 0, 3, 55] 61   [0, 0, 3, 100] 106  [0, 1, 0, 44] 47   [0, 1, 0, 89] 92
[0, 0, 3, 56] 62   [0, 1, 0, 0] 3      [0, 1, 0, 45] 48   [0, 1, 0, 90] 93
[0, 0, 3, 57] 63   [0, 1, 0, 1] 1      [0, 1, 0, 46] 49   [0, 1, 0, 91] 94
[0, 0, 3, 58] 64   [0, 1, 0, 2] 5      [0, 1, 0, 47] 50   [0, 1, 0, 92] 95
[0, 0, 3, 59] 65   [0, 1, 0, 3] 6      [0, 1, 0, 48] 51   [0, 1, 0, 93] 96
[0, 0, 3, 60] 66   [0, 1, 0, 4] 7      [0, 1, 0, 49] 52   [0, 1, 0, 94] 97
[0, 0, 3, 61] 67   [0, 1, 0, 5] 8      [0, 1, 0, 50] 53   [0, 1, 0, 95] 98
[0, 0, 3, 62] 68   [0, 1, 0, 6] 9      [0, 1, 0, 51] 54   [0, 1, 0, 96] 99
[0, 0, 3, 63] 69   [0, 1, 0, 7] 10     [0, 1, 0, 52] 55   [0, 1, 0, 97] 100
[0, 0, 3, 64] 70   [0, 1, 0, 8] 11     [0, 1, 0, 53] 56   [0, 1, 0, 98] 101
[0, 0, 3, 65] 71   [0, 1, 0, 9] 12     [0, 1, 0, 54] 57   [0, 1, 0, 99] 102
[0, 0, 3, 66] 72   [0, 1, 0, 10] 13    [0, 1, 0, 55] 58   [0, 1, 0, 100] 103
[0, 0, 3, 67] 73   [0, 1, 0, 11] 14    [0, 1, 0, 56] 59   [0, 1, 1, 0] 4
[0, 0, 3, 68] 74   [0, 1, 0, 12] 15    [0, 1, 0, 57] 60   [0, 1, 1, 1] 6
[0, 0, 3, 69] 75   [0, 1, 0, 13] 16    [0, 1, 0, 58] 61   [0, 1, 1, 2] 7
[0, 0, 3, 70] 76   [0, 1, 0, 14] 17    [0, 1, 0, 59] 62   [0, 1, 1, 3] 5
[0, 0, 3, 71] 77   [0, 1, 0, 15] 18    [0, 1, 0, 60] 63   [0, 1, 1, 4] 9
[0, 0, 3, 72] 78   [0, 1, 0, 16] 19    [0, 1, 0, 61] 64   [0, 1, 1, 5] 10
[0, 0, 3, 73] 79   [0, 1, 0, 17] 20    [0, 1, 0, 62] 65   [0, 1, 1, 6] 11
[0, 0, 3, 74] 80   [0, 1, 0, 18] 21    [0, 1, 0, 63] 66   [0, 1, 1, 7] 12
[0, 0, 3, 75] 81   [0, 1, 0, 19] 22    [0, 1, 0, 64] 67   [0, 1, 1, 8] 13
[0, 0, 3, 76] 82   [0, 1, 0, 20] 23    [0, 1, 0, 65] 68   [0, 1, 1, 9] 14
[0, 0, 3, 77] 83   [0, 1, 0, 21] 24    [0, 1, 0, 66] 69   [0, 1, 1, 10] 15
[0, 0, 3, 78] 84   [0, 1, 0, 22] 25    [0, 1, 0, 67] 70   [0, 1, 1, 11] 16
[0, 0, 3, 79] 85   [0, 1, 0, 23] 26    [0, 1, 0, 68] 71   [0, 1, 1, 12] 17
[0, 0, 3, 80] 86   [0, 1, 0, 24] 27    [0, 1, 0, 69] 72   [0, 1, 1, 13] 18
[0, 0, 3, 81] 87   [0, 1, 0, 25] 28    [0, 1, 0, 70] 73   [0, 1, 1, 14] 19
```

```
[0, 1, 1, 15] 20   [0, 1, 1, 60] 65    [0, 1, 2, 4] 11    [0, 1, 2, 49] 56
[0, 1, 1, 16] 21   [0, 1, 1, 61] 66    [0, 1, 2, 5] 12    [0, 1, 2, 50] 57
[0, 1, 1, 17] 22   [0, 1, 1, 62] 67    [0, 1, 2, 6] 13    [0, 1, 2, 51] 58
[0, 1, 1, 18] 23   [0, 1, 1, 63] 68    [0, 1, 2, 7] 14    [0, 1, 2, 52] 59
[0, 1, 1, 19] 24   [0, 1, 1, 64] 69    [0, 1, 2, 8] 15    [0, 1, 2, 53] 60
[0, 1, 1, 20] 25   [0, 1, 1, 65] 70    [0, 1, 2, 9] 16    [0, 1, 2, 54] 61
[0, 1, 1, 21] 26   [0, 1, 1, 66] 71    [0, 1, 2, 10] 17   [0, 1, 2, 55] 62
[0, 1, 1, 22] 27   [0, 1, 1, 67] 72    [0, 1, 2, 11] 18   [0, 1, 2, 56] 63
[0, 1, 1, 23] 28   [0, 1, 1, 68] 73    [0, 1, 2, 12] 19   [0, 1, 2, 57] 64
[0, 1, 1, 24] 29   [0, 1, 1, 69] 74    [0, 1, 2, 13] 20   [0, 1, 2, 58] 65
[0, 1, 1, 25] 30   [0, 1, 1, 70] 75    [0, 1, 2, 14] 21   [0, 1, 2, 59] 66
[0, 1, 1, 26] 31   [0, 1, 1, 71] 76    [0, 1, 2, 15] 22   [0, 1, 2, 60] 67
[0, 1, 1, 27] 32   [0, 1, 1, 72] 77    [0, 1, 2, 16] 23   [0, 1, 2, 61] 68
[0, 1, 1, 28] 33   [0, 1, 1, 73] 78    [0, 1, 2, 17] 24   [0, 1, 2, 62] 69
[0, 1, 1, 29] 34   [0, 1, 1, 74] 79    [0, 1, 2, 18] 25   [0, 1, 2, 63] 70
[0, 1, 1, 30] 35   [0, 1, 1, 75] 80    [0, 1, 2, 19] 26   [0, 1, 2, 64] 71
[0, 1, 1, 31] 36   [0, 1, 1, 76] 81    [0, 1, 2, 20] 27   [0, 1, 2, 65] 72
[0, 1, 1, 32] 37   [0, 1, 1, 77] 82    [0, 1, 2, 21] 28   [0, 1, 2, 66] 73
[0, 1, 1, 33] 38   [0, 1, 1, 78] 83    [0, 1, 2, 22] 29   [0, 1, 2, 67] 74
[0, 1, 1, 34] 39   [0, 1, 1, 79] 84    [0, 1, 2, 23] 30   [0, 1, 2, 68] 75
[0, 1, 1, 35] 40   [0, 1, 1, 80] 85    [0, 1, 2, 24] 31   [0, 1, 2, 69] 76
[0, 1, 1, 36] 41   [0, 1, 1, 81] 86    [0, 1, 2, 25] 32   [0, 1, 2, 70] 77
[0, 1, 1, 37] 42   [0, 1, 1, 82] 87    [0, 1, 2, 26] 33   [0, 1, 2, 71] 78
[0, 1, 1, 38] 43   [0, 1, 1, 83] 88    [0, 1, 2, 27] 34   [0, 1, 2, 72] 79
[0, 1, 1, 39] 44   [0, 1, 1, 84] 89    [0, 1, 2, 28] 35   [0, 1, 2, 73] 80
[0, 1, 1, 40] 45   [0, 1, 1, 85] 90    [0, 1, 2, 29] 36   [0, 1, 2, 74] 81
[0, 1, 1, 41] 46   [0, 1, 1, 86] 91    [0, 1, 2, 30] 37   [0, 1, 2, 75] 82
[0, 1, 1, 42] 47   [0, 1, 1, 87] 92    [0, 1, 2, 31] 38   [0, 1, 2, 76] 83
[0, 1, 1, 43] 48   [0, 1, 1, 88] 93    [0, 1, 2, 32] 39   [0, 1, 2, 77] 84
[0, 1, 1, 44] 49   [0, 1, 1, 89] 94    [0, 1, 2, 33] 40   [0, 1, 2, 78] 85
[0, 1, 1, 45] 50   [0, 1, 1, 90] 95    [0, 1, 2, 34] 41   [0, 1, 2, 79] 86
[0, 1, 1, 46] 51   [0, 1, 1, 91] 96    [0, 1, 2, 35] 42   [0, 1, 2, 80] 87
[0, 1, 1, 47] 52   [0, 1, 1, 92] 97    [0, 1, 2, 36] 43   [0, 1, 2, 81] 88
[0, 1, 1, 48] 53   [0, 1, 1, 93] 98    [0, 1, 2, 37] 44   [0, 1, 2, 82] 89
[0, 1, 1, 49] 54   [0, 1, 1, 94] 99    [0, 1, 2, 38] 45   [0, 1, 2, 83] 90
[0, 1, 1, 50] 55   [0, 1, 1, 95] 100   [0, 1, 2, 39] 46   [0, 1, 2, 84] 91
[0, 1, 1, 51] 56   [0, 1, 1, 96] 101   [0, 1, 2, 40] 47   [0, 1, 2, 85] 92
[0, 1, 1, 52] 57   [0, 1, 1, 97] 102   [0, 1, 2, 41] 48   [0, 1, 2, 86] 93
[0, 1, 1, 53] 58   [0, 1, 1, 98] 103   [0, 1, 2, 42] 49   [0, 1, 2, 87] 94
[0, 1, 1, 54] 59   [0, 1, 1, 99] 104   [0, 1, 2, 43] 50   [0, 1, 2, 88] 95
[0, 1, 1, 55] 60   [0, 1, 1, 100] 105  [0, 1, 2, 44] 51   [0, 1, 2, 89] 96
[0, 1, 1, 56] 61   [0, 1, 2, 0] 7      [0, 1, 2, 45] 52   [0, 1, 2, 90] 97
[0, 1, 1, 57] 62   [0, 1, 2, 1] 8      [0, 1, 2, 46] 53   [0, 1, 2, 91] 98
[0, 1, 1, 58] 63   [0, 1, 2, 2] 9      [0, 1, 2, 47] 54   [0, 1, 2, 92] 99
[0, 1, 1, 59] 64   [0, 1, 2, 3] 10     [0, 1, 2, 48] 55   [0, 1, 2, 93] 100
```

```
[0, 1, 2, 94] 101  [0, 1, 3, 38] 47   [0, 1, 3, 83] 92   [0, 2, 0, 27] 33
[0, 1, 2, 95] 102  [0, 1, 3, 39] 48   [0, 1, 3, 84] 93   [0, 2, 0, 28] 34
[0, 1, 2, 96] 103  [0, 1, 3, 40] 49   [0, 1, 3, 85] 94   [0, 2, 0, 29] 35
[0, 1, 2, 97] 104  [0, 1, 3, 41] 50   [0, 1, 3, 86] 95   [0, 2, 0, 30] 36
[0, 1, 2, 98] 105  [0, 1, 3, 42] 51   [0, 1, 3, 87] 96   [0, 2, 0, 31] 37
[0, 1, 2, 99] 106  [0, 1, 3, 43] 52   [0, 1, 3, 88] 97   [0, 2, 0, 32] 38
[0, 1, 2, 100] 107 [0, 1, 3, 44] 53   [0, 1, 3, 89] 98   [0, 2, 0, 33] 39
[0, 1, 3, 0] 5     [0, 1, 3, 45] 54   [0, 1, 3, 90] 99   [0, 2, 0, 34] 40
[0, 1, 3, 1] 7     [0, 1, 3, 46] 55   [0, 1, 3, 91] 100  [0, 2, 0, 35] 41
[0, 1, 3, 2] 11    [0, 1, 3, 47] 56   [0, 1, 3, 92] 101  [0, 2, 0, 36] 42
[0, 1, 3, 3] 12    [0, 1, 3, 48] 57   [0, 1, 3, 93] 102  [0, 2, 0, 37] 43
[0, 1, 3, 4] 13    [0, 1, 3, 49] 58   [0, 1, 3, 94] 103  [0, 2, 0, 38] 44
[0, 1, 3, 5] 14    [0, 1, 3, 50] 59   [0, 1, 3, 95] 104  [0, 2, 0, 39] 45
[0, 1, 3, 6] 15    [0, 1, 3, 51] 60   [0, 1, 3, 96] 105  [0, 2, 0, 40] 46
[0, 1, 3, 7] 16    [0, 1, 3, 52] 61   [0, 1, 3, 97] 106  [0, 2, 0, 41] 47
[0, 1, 3, 8] 17    [0, 1, 3, 53] 62   [0, 1, 3, 98] 107  [0, 2, 0, 42] 48
[0, 1, 3, 9] 18    [0, 1, 3, 54] 63   [0, 1, 3, 99] 108  [0, 2, 0, 43] 49
[0, 1, 3, 10] 19   [0, 1, 3, 55] 64   [0, 1, 3, 100] 109 [0, 2, 0, 44] 50
[0, 1, 3, 11] 20   [0, 1, 3, 56] 65   [0, 2, 0, 0] 3     [0, 2, 0, 45] 51
[0, 1, 3, 12] 21   [0, 1, 3, 57] 66   [0, 2, 0, 1] 2     [0, 2, 0, 46] 52
[0, 1, 3, 13] 22   [0, 1, 3, 58] 67   [0, 2, 0, 2] 5     [0, 2, 0, 47] 53
[0, 1, 3, 14] 23   [0, 1, 3, 59] 68   [0, 2, 0, 3] 8     [0, 2, 0, 48] 54
[0, 1, 3, 15] 24   [0, 1, 3, 60] 69   [0, 2, 0, 4] 10    [0, 2, 0, 49] 55
[0, 1, 3, 16] 25   [0, 1, 3, 61] 70   [0, 2, 0, 5] 6     [0, 2, 0, 50] 56
[0, 1, 3, 17] 26   [0, 1, 3, 62] 71   [0, 2, 0, 6] 9     [0, 2, 0, 51] 57
[0, 1, 3, 18] 27   [0, 1, 3, 63] 72   [0, 2, 0, 7] 13    [0, 2, 0, 52] 58
[0, 1, 3, 19] 28   [0, 1, 3, 64] 73   [0, 2, 0, 8] 14    [0, 2, 0, 53] 59
[0, 1, 3, 20] 29   [0, 1, 3, 65] 74   [0, 2, 0, 9] 15    [0, 2, 0, 54] 60
[0, 1, 3, 21] 30   [0, 1, 3, 66] 75   [0, 2, 0, 10] 16   [0, 2, 0, 55] 61
[0, 1, 3, 22] 31   [0, 1, 3, 67] 76   [0, 2, 0, 11] 17   [0, 2, 0, 56] 62
[0, 1, 3, 23] 32   [0, 1, 3, 68] 77   [0, 2, 0, 12] 18   [0, 2, 0, 57] 63
[0, 1, 3, 24] 33   [0, 1, 3, 69] 78   [0, 2, 0, 13] 19   [0, 2, 0, 58] 64
[0, 1, 3, 25] 34   [0, 1, 3, 70] 79   [0, 2, 0, 14] 20   [0, 2, 0, 59] 65
[0, 1, 3, 26] 35   [0, 1, 3, 71] 80   [0, 2, 0, 15] 21   [0, 2, 0, 60] 66
[0, 1, 3, 27] 36   [0, 1, 3, 72] 81   [0, 2, 0, 16] 22   [0, 2, 0, 61] 67
[0, 1, 3, 28] 37   [0, 1, 3, 73] 82   [0, 2, 0, 17] 23   [0, 2, 0, 62] 68
[0, 1, 3, 29] 38   [0, 1, 3, 74] 83   [0, 2, 0, 18] 24   [0, 2, 0, 63] 69
[0, 1, 3, 30] 39   [0, 1, 3, 75] 84   [0, 2, 0, 19] 25   [0, 2, 0, 64] 70
[0, 1, 3, 31] 40   [0, 1, 3, 76] 85   [0, 2, 0, 20] 26   [0, 2, 0, 65] 71
[0, 1, 3, 32] 41   [0, 1, 3, 77] 86   [0, 2, 0, 21] 27   [0, 2, 0, 66] 72
[0, 1, 3, 33] 42   [0, 1, 3, 78] 87   [0, 2, 0, 22] 28   [0, 2, 0, 67] 73
[0, 1, 3, 34] 43   [0, 1, 3, 79] 88   [0, 2, 0, 23] 29   [0, 2, 0, 68] 74
[0, 1, 3, 35] 44   [0, 1, 3, 80] 89   [0, 2, 0, 24] 30   [0, 2, 0, 69] 75
[0, 1, 3, 36] 45   [0, 1, 3, 81] 90   [0, 2, 0, 25] 31   [0, 2, 0, 70] 76
[0, 1, 3, 37] 46   [0, 1, 3, 82] 91   [0, 2, 0, 26] 32   [0, 2, 0, 71] 77
```

```
[0, 2, 0, 72] 78    [0, 2, 1, 16] 24    [0, 2, 1, 61] 69    [0, 2, 2, 5] 15
[0, 2, 0, 73] 79    [0, 2, 1, 17] 25    [0, 2, 1, 62] 70    [0, 2, 2, 6] 16
[0, 2, 0, 74] 80    [0, 2, 1, 18] 26    [0, 2, 1, 63] 71    [0, 2, 2, 7] 17
[0, 2, 0, 75] 81    [0, 2, 1, 19] 27    [0, 2, 1, 64] 72    [0, 2, 2, 8] 18
[0, 2, 0, 76] 82    [0, 2, 1, 20] 28    [0, 2, 1, 65] 73    [0, 2, 2, 9] 19
[0, 2, 0, 77] 83    [0, 2, 1, 21] 29    [0, 2, 1, 66] 74    [0, 2, 2, 10] 20
[0, 2, 0, 78] 84    [0, 2, 1, 22] 30    [0, 2, 1, 67] 75    [0, 2, 2, 11] 21
[0, 2, 0, 79] 85    [0, 2, 1, 23] 31    [0, 2, 1, 68] 76    [0, 2, 2, 12] 22
[0, 2, 0, 80] 86    [0, 2, 1, 24] 32    [0, 2, 1, 69] 77    [0, 2, 2, 13] 23
[0, 2, 0, 81] 87    [0, 2, 1, 25] 33    [0, 2, 1, 70] 78    [0, 2, 2, 14] 24
[0, 2, 0, 82] 88    [0, 2, 1, 26] 34    [0, 2, 1, 71] 79    [0, 2, 2, 15] 25
[0, 2, 0, 83] 89    [0, 2, 1, 27] 35    [0, 2, 1, 72] 80    [0, 2, 2, 16] 26
[0, 2, 0, 84] 90    [0, 2, 1, 28] 36    [0, 2, 1, 73] 81    [0, 2, 2, 17] 27
[0, 2, 0, 85] 91    [0, 2, 1, 29] 37    [0, 2, 1, 74] 82    [0, 2, 2, 18] 28
[0, 2, 0, 86] 92    [0, 2, 1, 30] 38    [0, 2, 1, 75] 83    [0, 2, 2, 19] 29
[0, 2, 0, 87] 93    [0, 2, 1, 31] 39    [0, 2, 1, 76] 84    [0, 2, 2, 20] 30
[0, 2, 0, 88] 94    [0, 2, 1, 32] 40    [0, 2, 1, 77] 85    [0, 2, 2, 21] 31
[0, 2, 0, 89] 95    [0, 2, 1, 33] 41    [0, 2, 1, 78] 86    [0, 2, 2, 22] 32
[0, 2, 0, 90] 96    [0, 2, 1, 34] 42    [0, 2, 1, 79] 87    [0, 2, 2, 23] 33
[0, 2, 0, 91] 97    [0, 2, 1, 35] 43    [0, 2, 1, 80] 88    [0, 2, 2, 24] 34
[0, 2, 0, 92] 98    [0, 2, 1, 36] 44    [0, 2, 1, 81] 89    [0, 2, 2, 25] 35
[0, 2, 0, 93] 99    [0, 2, 1, 37] 45    [0, 2, 1, 82] 90    [0, 2, 2, 26] 36
[0, 2, 0, 94] 100   [0, 2, 1, 38] 46    [0, 2, 1, 83] 91    [0, 2, 2, 27] 37
[0, 2, 0, 95] 101   [0, 2, 1, 39] 47    [0, 2, 1, 84] 92    [0, 2, 2, 28] 38
[0, 2, 0, 96] 102   [0, 2, 1, 40] 48    [0, 2, 1, 85] 93    [0, 2, 2, 29] 39
[0, 2, 0, 97] 103   [0, 2, 1, 41] 49    [0, 2, 1, 86] 94    [0, 2, 2, 30] 40
[0, 2, 0, 98] 104   [0, 2, 1, 42] 50    [0, 2, 1, 87] 95    [0, 2, 2, 31] 41
[0, 2, 0, 99] 105   [0, 2, 1, 43] 51    [0, 2, 1, 88] 96    [0, 2, 2, 32] 42
[0, 2, 0, 100] 106  [0, 2, 1, 44] 52    [0, 2, 1, 89] 97    [0, 2, 2, 33] 43
[0, 2, 1, 0] 4      [0, 2, 1, 45] 53    [0, 2, 1, 90] 98    [0, 2, 2, 34] 44
[0, 2, 1, 1] 9      [0, 2, 1, 46] 54    [0, 2, 1, 91] 99    [0, 2, 2, 35] 45
[0, 2, 1, 2] 7      [0, 2, 1, 47] 55    [0, 2, 1, 92] 100   [0, 2, 2, 36] 46
[0, 2, 1, 3] 11     [0, 2, 1, 48] 56    [0, 2, 1, 93] 101   [0, 2, 2, 37] 47
[0, 2, 1, 4] 12     [0, 2, 1, 49] 57    [0, 2, 1, 94] 102   [0, 2, 2, 38] 48
[0, 2, 1, 5] 13     [0, 2, 1, 50] 58    [0, 2, 1, 95] 103   [0, 2, 2, 39] 49
[0, 2, 1, 6] 14     [0, 2, 1, 51] 59    [0, 2, 1, 96] 104   [0, 2, 2, 40] 50
[0, 2, 1, 7] 15     [0, 2, 1, 52] 60    [0, 2, 1, 97] 105   [0, 2, 2, 41] 51
[0, 2, 1, 8] 16     [0, 2, 1, 53] 61    [0, 2, 1, 98] 106   [0, 2, 2, 42] 52
[0, 2, 1, 9] 17     [0, 2, 1, 54] 62    [0, 2, 1, 99] 107   [0, 2, 2, 43] 53
[0, 2, 1, 10] 18    [0, 2, 1, 55] 63    [0, 2, 1, 100] 108  [0, 2, 2, 44] 54
[0, 2, 1, 11] 19    [0, 2, 1, 56] 64    [0, 2, 2, 0] 2      [0, 2, 2, 45] 55
[0, 2, 1, 12] 20    [0, 2, 1, 57] 65    [0, 2, 2, 1] 4      [0, 2, 2, 46] 56
[0, 2, 1, 13] 21    [0, 2, 1, 58] 66    [0, 2, 2, 2] 9      [0, 2, 2, 47] 57
[0, 2, 1, 14] 22    [0, 2, 1, 59] 67    [0, 2, 2, 3] 13     [0, 2, 2, 48] 58
[0, 2, 1, 15] 23    [0, 2, 1, 60] 68    [0, 2, 2, 4] 14     [0, 2, 2, 49] 59
```

```
[0, 2, 2, 50] 60    [0, 2, 2, 89] 99    [0, 2, 3, 27] 39    [0, 2, 3, 66] 78
[0, 2, 2, 51] 61    [0, 2, 2, 90] 100   [0, 2, 3, 28] 40    [0, 2, 3, 67] 79
[0, 2, 2, 52] 62    [0, 2, 2, 91] 101   [0, 2, 3, 29] 41    [0, 2, 3, 68] 80
[0, 2, 2, 53] 63    [0, 2, 2, 92] 102   [0, 2, 3, 30] 42    [0, 2, 3, 69] 81
[0, 2, 2, 54] 64    [0, 2, 2, 93] 103   [0, 2, 3, 31] 43    [0, 2, 3, 70] 82
[0, 2, 2, 55] 65    [0, 2, 2, 94] 104   [0, 2, 3, 32] 44    [0, 2, 3, 71] 83
[0, 2, 2, 56] 66    [0, 2, 2, 95] 105   [0, 2, 3, 33] 45    [0, 2, 3, 72] 84
[0, 2, 2, 57] 67    [0, 2, 2, 96] 106   [0, 2, 3, 34] 46    [0, 2, 3, 73] 85
[0, 2, 2, 58] 68    [0, 2, 2, 97] 107   [0, 2, 3, 35] 47    [0, 2, 3, 74] 86
[0, 2, 2, 59] 69    [0, 2, 2, 98] 108   [0, 2, 3, 36] 48    [0, 2, 3, 75] 87
[0, 2, 2, 60] 70    [0, 2, 2, 99] 109   [0, 2, 3, 37] 49    [0, 2, 3, 76] 88
[0, 2, 2, 61] 71    [0, 2, 2, 100] 110  [0, 2, 3, 38] 50    [0, 2, 3, 77] 89
[0, 2, 2, 62] 72    [0, 2, 3, 0] 5      [0, 2, 3, 39] 51    [0, 2, 3, 78] 90
[0, 2, 2, 63] 73    [0, 2, 3, 1] 8      [0, 2, 3, 40] 52    [0, 2, 3, 79] 91
[0, 2, 2, 64] 74    [0, 2, 3, 2] 14     [0, 2, 3, 41] 53    [0, 2, 3, 80] 92
[0, 2, 2, 65] 75    [0, 2, 3, 3] 15     [0, 2, 3, 42] 54    [0, 2, 3, 81] 93
[0, 2, 2, 66] 76    [0, 2, 3, 4] 13     [0, 2, 3, 43] 55    [0, 2, 3, 82] 94
[0, 2, 2, 67] 77    [0, 2, 3, 5] 17     [0, 2, 3, 44] 56    [0, 2, 3, 83] 95
[0, 2, 2, 68] 78    [0, 2, 3, 6] 18     [0, 2, 3, 45] 57    [0, 2, 3, 84] 96
[0, 2, 2, 69] 79    [0, 2, 3, 7] 19     [0, 2, 3, 46] 58    [0, 2, 3, 85] 97
[0, 2, 2, 70] 80    [0, 2, 3, 8] 20     [0, 2, 3, 47] 59    [0, 2, 3, 86] 98
[0, 2, 2, 71] 81    [0, 2, 3, 9] 21     [0, 2, 3, 48] 60    [0, 2, 3, 87] 99
[0, 2, 2, 72] 82    [0, 2, 3, 10] 22    [0, 2, 3, 49] 61    [0, 2, 3, 88] 100
[0, 2, 2, 73] 83    [0, 2, 3, 11] 23    [0, 2, 3, 50] 62    [0, 2, 3, 89] 101
[0, 2, 2, 74] 84    [0, 2, 3, 12] 24    [0, 2, 3, 51] 63    [0, 2, 3, 90] 102
[0, 2, 2, 75] 85    [0, 2, 3, 13] 25    [0, 2, 3, 52] 64    [0, 2, 3, 91] 103
[0, 2, 2, 76] 86    [0, 2, 3, 14] 26    [0, 2, 3, 53] 65    [0, 2, 3, 92] 104
[0, 2, 2, 77] 87    [0, 2, 3, 15] 27    [0, 2, 3, 54] 66    [0, 2, 3, 93] 105
[0, 2, 2, 78] 88    [0, 2, 3, 16] 28    [0, 2, 3, 55] 67    [0, 2, 3, 94] 106
[0, 2, 2, 79] 89    [0, 2, 3, 17] 29    [0, 2, 3, 56] 68    [0, 2, 3, 95] 107
[0, 2, 2, 80] 90    [0, 2, 3, 18] 30    [0, 2, 3, 57] 69    [0, 2, 3, 96] 108
[0, 2, 2, 81] 91    [0, 2, 3, 19] 31    [0, 2, 3, 58] 70    [0, 2, 3, 97] 109
[0, 2, 2, 82] 92    [0, 2, 3, 20] 32    [0, 2, 3, 59] 71    [0, 2, 3, 98] 110
[0, 2, 2, 83] 93    [0, 2, 3, 21] 33    [0, 2, 3, 60] 72    [0, 2, 3, 99] 111
[0, 2, 2, 84] 94    [0, 2, 3, 22] 34    [0, 2, 3, 61] 73    [0, 2, 3, 100] 112
[0, 2, 2, 85] 95    [0, 2, 3, 23] 35    [0, 2, 3, 62] 74
[0, 2, 2, 86] 96    [0, 2, 3, 24] 36    [0, 2, 3, 63] 75
[0, 2, 2, 87] 97    [0, 2, 3, 25] 37    [0, 2, 3, 64] 76
[0, 2, 2, 88] 98    [0, 2, 3, 26] 38    [0, 2, 3, 65] 77
```

A.1 $d(3^{a_3}1^{a_1})$ Data

The nth column and mth row represents $d(3^{n+9}1^{m-1})$.

Note the table have to go up to $a_1 = 6a_3 - 5$ for each a_3 in order to show we actually have the stable behavior.

	3^{10}	3^{11}	3^{12}	3^{13}	3^{14}	3^{15}	3^{16}	3^{17}	3^{18}	3^{19}	3^{20}	3^{21}	3^{22}	3^{23}	3^{24}	3^{25}	3^{26}	3^{27}	3^{28}	3^{29}	3^{30}
1^{0}	22	25	24	24	27	33	30	33	37	40	40	40	43	49	48	48	51	57	56	56	59
1^{1}	21	20	24	27	29	30	33	33	36	36	40	43	45	44	48	51	53	52	56	59	61
1^{2}	19	17	21	24	26	27	27	30	36	35	35	38	44	43	43	46	52	49	53	56	58
1^{3}	17	17	20	20	23	27	30	28	31	35	38	36	39	43	46	46	49	49	52	54	57
1^{4}	12	15	19	22	20	23	27	30	28	31	35	38	38	38	41	43	46	46	49	49	52
1^{5}	15	15	18	20	23	23	26	28	31	31	34	34	38	41	43	42	46	49	49	52	52
1^{6}	12	12	15	17	20	20	23	23	26	30	33	33	33	36	42	41	41	44	46	49	49
1^{7}	12	15	17	16	20	23	23	26	26	29	29	33	36	38	37	41	44	46	45	46	52
1^{8}	11	17	16	15	19	19	20	26	29	29	29	32	34	37	37	40	40	44	47	45	46
1^{9}	13	11	14	14	18	21	17	21	24	26	26	29	31	31	34	40	37	40	44	47	46
1^{10}	11	14	14	14	17	17	21	24	26	26	29	27	31	34	36	37	40	40	43	43	46
1^{11}	10	10	14	17	17	17	20	26	23	24	27	26	29	29	32	32	35	39	42	40	44
1^{12}	10	10	13	13	17	20	22	23	23	26	25	28	29	32	32	35	35	38	40	40	43
1^{13}	10	13	13	12	16	16	17	23	26	25	28	26	27	27	30	32	31	36	40	43	42
1^{14}	8	8	11	11	15	18	17	20	22	21	25	28	27	30	32	32	35	35	35	38	40
1^{15}	3	7	10	10	13	13	16	18	21	21	24	24	27	28	28	31	37	35	38	36	40
1^{16}	3	6	12	9	13	16	14	17	21	24	24	27	27	28	31	33	32	33	36	35	38
1^{17}	6	8	9	9	12	14	17	17	20	20	23	27	25	28	29	32	32	35	34	37	38
1^{18}	8	5	9	9	8	13	17	20	19	22	20	24	27	25	28	32	35	34	37	35	33
1^{19}	5	5	5	9	12	12	12	15	17	20	20	23	25	28	28	31	31	34	35	34	38
1^{20}	5	5	10	9	8	12	15	13	16	20	23	21	24	28	31	31	34	34	34	38	38
1^{21}	5	4	8	8	8	11	13	16	16	19	21	24	24	27	27	30	34	34	33	37	35
1^{22}	3	0	3	5	8	8	11	13	16	16	19	19	22	26	29	29	29	32	33	36	34
1^{23}	0	3	5	8	8	11	9	12	16	19	19	22	22	25	25	29	32	34	31	34	34
1^{24}	0	0	8	5	4	8	11	11	11	14	16	16	19	25	22	26	29	31	30	34	37
1^{25}	0	0	4	4	4	7	13	11	14	16	16	19	21	22	22	25	31	30	30	33	35
1^{26}	0	5	4	4	7	9	8	8	16	13	13	21	18	22	22	27	26	27	33	31	35
1^{27}	0	0	0	3	9	8	8	11	13	13	13	17	17	22	21	25	28	27	30	28	32
1^{28}	0	0	3	5	4	8	11	13	13	13	18	17	17	20	20	23	23	26	28	28	31
1^{29}	0	0	0	3	0	8	9	9	9	18	17	14	15	22	19	23	26	24	28	31	33
1^{30}	0	0	0	0	8	5	9	9	14	13	14	14	17	19	19	22	24	24	27	33	32
1^{31}	0	0	0	0	4	4	9	8	12	15	14	17	19	19	22	20	24	27	29	28	32
1^{32}	0	0	0	5	4	4	7	7	7	10	12	12	15	17	20	20	23	25	28	28	31
1^{33}	0	0	0	0	0	3	9	7	10	12	12	15	17	16	20	23	21	24	28	31	29
1^{34}	0	0	0	0	3	5	4	7	5	8	8	17	16	16	19	21	24	24	27	29	32
1^{35}	0	0	0	0	0	3	0	6	9	8	8	11	13	16	16	19	21	21	24	26	29
1^{36}	0	0	0	0	0	0	8	5	9	8	11	13	12	16	19	17	21	24	26	25	29
1^{37}	0	0	0	0	0	0	4	4	4	7	13	12	12	15	17	17	20	26	25	25	28
1^{38}	0	0	0	0	0	5	4	4	7	9	8	12	15	13	17	20	22	21	25	28	26
1^{39}	0	0	0	0	0	0	0	3	9	8	8	11	13	13	16	22	21	21	24	26	26
1^{40}	0	0	0	0	0	0	3	5	4	8	11	13	13	16	18	17	21	24	26	26	29
1^{41}	0	0	0	0	0	0	0	3	0	8	9	9	9	18	17	17	20	22	22	25	31
1^{42}	0	0	0	0	0	0	0	0	8	5	9	9	14	13	14	20	18	22	25	25	28
1^{43}	0	0	0	0	0	0	0	0	4	4	9	8	12	15	14	17	17	20	20	23	25
1^{44}	0	0	0	0	0	0	0	5	4	4	7	7	7	10	12	12	15	17	17	20	22
1^{45}	0	0	0	0	0	0	0	0	0	3	9	7	10	12	12	15	17	17	20	22	21
1^{46}	0	0	0	0	0	0	0	0	3	5	4	7	6	9	10	13	13	16	16	20	23
1^{47}	0	0	0	0	0	0	0	0	0	3	0	6	9	9	9	13	16	16	16	19	21
1^{48}	0	0	0	0	0	0	0	0	0	0	8	5	9	9	8	12	12	16	19	17	20
1^{49}	0	0	0	0	0	0	0	0	0	0	4	4	4	7	8	11	9	12	16	19	17
1^{50}	0	0	0	0	0	0	0	0	0	5	4	4	7	9	8	9	12	12	15	17	20
1^{51}	0	0	0	0	0	0	0	0	0	0	0	3	9	8	8	8	12	15	13	16	20
1^{52}	0	0	0	0	0	0	0	0	0	0	3	5	4	8	8	8	11	13	16	16	19
1^{53}	0	0	0	0	0	0	0	0	0	0	0	3	0	8	8	8	8	11	13	16	16
1^{54}	0	0	0	0	0	0	0	0	0	0	0	0	8	5	8	8	11	13	12	16	19
1^{55}	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	7	13	12	12	15	17
1^{56}	0	0	0	0	0	0	0	0	0	0	0	5	4	4	7	9	8	12	15	13	17
1^{57}	0	0	0	0	0	0	0	0	0	0	0	0	0	3	9	8	8	11	13	13	16
1^{58}	0	0	0	0	0	0	0	0	0	0	0	0	3	5	4	8	11	13	13	16	18
1^{59}	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	8	9	9	9	18	17
1^{60}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	5	9	9	14	13	14
1^{61}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	9	8	12	15	14
1^{62}	0	0	0	0	0	0	0	0	0	0	0	0	0	5	4	4	7	7	7	10	12
1^{63}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	9	7	10	12	12
1^{64}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	5	4	7	6	9	10
1^{65}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	6	9	9	9

1^{66}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	5	9	9	8
1^{67}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	7	8
1^{68}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	4	4	7	9	8
1^{69}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	9	8	8
1^{70}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	5	4	8	8
1^{71}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	8	8
1^{72}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	5	8
1^{73}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4
1^{74}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	4	4	7
1^{75}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	9
1^{76}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	5	4
1^{77}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0
1^{78}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
1^{79}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
1^{80}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	4
1^{81}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1^{82}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
1^{83}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1^{84}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1^{85}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1^{86}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1^{87}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1^{88}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1^{89}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1^{90}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1^{175}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A.2 $d(n^1 1^k)$

	2^1	3^1	4^1	5^1	6^1
1^0	0	0	3	1	3
1^1	0	3	1	3	5
1^2	0	0	0	5	7
1^3	0	0	0	3	9
1^4	0	0	3	1	1
1^5	0	0	0	3	0
1^6	0	0	0	5	0
1^7	0	0	0	3	0
1^8	0	0	0	0	0
1^9	0	0	0	3	5
1^{10}	0	0	0	5	0
1^{11}	0	0	0	0	9
1^{12}	0	0	0	0	0
1^{13}	0	0	0	3	0
1^{14}	0	0	0	5	3
1^{15}	0	0	0	0	0
1^{16}	0	0	0	0	0
1^{17}	0	0	0	3	0
1^{18}	0	0	0	5	0
1^{19}	0	0	0	0	0
1^{20}	0	0	0	0	0
1^{21}	0	0	0	3	0

1^{22}	0	0	0	5	0
1^{23}	0	0	0	0	0
1^{24}	0	0	0	0	0
1^{25}	0	0	0	3	0
1^{26}	0	0	0	5	0
1^{27}	0	0	0	0	0
1^{28}	0	0	0	0	0
1^{29}	0	0	0	3	0
1^{30}	0	0	0	5	0
1^{31}	0	0	0	0	0
1^{32}	0	0	0	0	0
1^{33}	0	0	0	3	0
1^{34}	0	0	0	5	0
1^{35}	0	0	0	0	0
1^{36}	0	0	0	0	0
1^{37}	0	0	0	3	0
1^{38}	0	0	0	5	0
1^{39}	0	0	0	0	0
1^{40}	0	0	0	0	0
1^{41}	0	0	0	3	0
1^{42}	0	0	0	5	0
1^{43}	0	0	0	0	0
1^{44}	0	0	0	0	0
1^{45}	0	0	0	3	0
1^{46}	0	0	0	5	0
1^{47}	0	0	0	0	0
1^{48}	0	0	0	0	0
1^{49}	0	0	0	3	0
1^{50}	0	0	0	5	0

B Program Listing

```
/**
 * star.java
 * ----------
 *
 * Chao Xu
 *
 * Purpose: This program computes and outputs the Sprague-Grundy
 * numbers for a given spider graph. It saves all previous results to
 * speed up future computations.
 *
 * Usage: First compile the program using 'javac star.java'
 * Then run the program using 'java star'.
 * Then you may type in the spider graph that you wish to test, formatted
 * like so:
 *      5 5 2 2 1 1 1 1
 * tests a spider with 2 legs of length 5, 2 legs of length 2, and 4 legs
 * length 1.
 *
 * When you are finished computing, type -1 at the prompt, and it will save
 * your data. Wait for it to say "Done!", then you may exit.
 *
**/
import java.util.*;

public class star {
  public static HashMap<Firework, Integer> h =
    new HashMap<Firework, Integer>();
  public static int count=0;
  public static void main(String[] args){

    //int n = 3;
    ArrayList<Integer> z = new ArrayList<Integer>();
    for(int i=0;i<args.length;i++){
      z.add(Integer.parseInt(args[i]));
    }
    h.put(new Firework(0), 0);
    Firework f = new Firework(z);
    //All the computation is done in this following line
    sg(f); //<------this line, takes 99.999% of the running time
    //all the rest is just process the results, which is stored in h

    //System.out.println(h);
    //System.out.println(sg(f));
    Firework[] F = new Firework[h.size()];
    h.keySet().toArray(F);
    Arrays.sort(F);
    for(int i=0;i<F.length;i++){
      System.out.println(F[i]+" "+h.get(F[i]));
    }
    //System.out.println(count);
    //System.out.println(F2);
  }
  //Find the sg number of a graph with many components
  public static int sg(ArrayList<Firework> f){
    int s = 0;
    for(int i=0;i<f.size();i++){
      s^=sg(f.get(i));
    }
    return s;
  }
  //find the sg number of a graph with 1 component.
  public static int sg(Firework f){
    count++;
    if(h.containsKey(f)){
      return h.get(f);
    }
    //The idea is to do every possible move
    HashSet<Integer> s = new HashSet<Integer>();
    //Now start playing with delete 1 or 2 edges...
    //I call it snap...
    for(int i=0;i<f.a.size();i++){
      for(int j=1;j<=f.a.get(i);j++){
        s.add(sg(f.snap(i, j, 0)));
        s.add(sg(f.snap(i, j, 1)));
                //System.out.println(f+" "+i+" "+ j+" "+s+" s");
      }
    }

    //Then try with doing stuff in the center...
    //I call it blow
    //BY TEST EVERY COMBINATION OF COURSE (O(2^n) time T_T)
    //Maybe it can be improved
    blows(new ArrayList<Integer>(), f.a.size()-1, f, s);
    int k = mex(s);
    h.put(f,k);
    return k;
  }
  //blows recursively find the sg number of all possible
  //subgraphs built from remove edges from the center
  public static void blows(ArrayList<Integer> t, int n, Firework f, HashSet<Integer> s){
    if(n==-1){
      if(t.size()>0){
        s.add(sg(f.blow(t)));
      }
      return;
    }
    t.add(n);
    blows(t,n-1,f,s);
    t.remove(t.size()-1);
    blows(t,n-1,f,s);
  }
  public static int mex(HashSet<Integer> s){
    for(int i=0;i<s.size();i++){
      if(!s.contains(i)){
        return i;
      }
    }
    return s.size();
  }
}




class Firework implements Comparable<Firework> {
  public ArrayList<Integer> a;
  public int hash;
  Firework(ArrayList<Integer> b){
    a = new ArrayList<Integer>();
    for(int i=0;i<b.size();i++){
      if(b.get(i)>0){
        a.add(b.get(i));
      }
    }
    if(a.size()==2){
      Firework F = new Firework(a.get(0)+a.get(1));
      a = F.a;
      hash = F.hash;
    }else{
      Collections.sort(a);
      hash = Arrays.deepHashCode(a.toArray());
    }
  }
  Firework(int t){
    a = new ArrayList<Integer>();
    if(t>0){
      a.add(t);
    }
    hash = Arrays.deepHashCode(a.toArray());
  }
  //r=1 if remove 2, =1 if remove 1
  public ArrayList<Firework> snap(int k, int j, int r){
    int l = a.get(k);
    Firework p = new Firework(l-j-r);
    ArrayList<Integer> c = new ArrayList<Integer>(a);
    c.set(k, j-1);
    if(j-1==0){
      c.remove(k);
    }
    Firework n = new Firework(c);
    ArrayList<Firework> result = new ArrayList<Firework>();
    result.add(p);
    result.add(n);
    return result;
  }

  public ArrayList<Firework> blow(ArrayList<Integer> r){
    int c = r.size()-1;
    ArrayList<Firework> f = new ArrayList<Firework>();
    ArrayList<Integer> n = new ArrayList<Integer>();
    for(int i=0;i<a.size();i++){
      if(c>-1 && r.get(c)==i){
        c--;
        f.add(new Firework(a.get(i)-1));
      }else{
        n.add(a.get(i));
      }
    }
    f.add(new Firework(n));
    return f;
  }

  public int hashCode() {
    return hash;
  }
  public String toString(){
    return a.toString();
  }
  public boolean equals( Object obj )
  {
    boolean flag = false;
    Firework emp = ( Firework )obj;
    if( this.a.equals(emp.a))
      flag = true;
    return flag;
  }
  @Override
  public int compareTo(Firework f) {
    if(a.size()>f.a.size()){
      return 1;
    }
    if(a.size()<f.a.size()){
      return -1;
    }
    for(int i=0;i<a.size();i++){
      if(a.get(i)<f.a.get(i)){
        return -1;
      }
      if(a.get(i)>f.a.get(i)){
        return 1;
      }
    }
    return 0;
  }
}

import java.util.*;
//The path length is at most 4.
public class fours {
  //number of 1,2,3,4 paths's
  public static int[][][][] h;
  public static void main(String[] args){

    int n1=Integer.parseInt(args[0]);
    int n2=Integer.parseInt(args[1]);
    int n3=Integer.parseInt(args[2]);
    int n4=Integer.parseInt(args[3]);
    h = new int[n1+n2+n3+n4+1][n2+n3+n4+1][n3+n4+1][n4+1];
    for(int a4=0;a4<n4+1;a4++){
      for(int a3=0;a3<n3+n4+1;a3++){
        for(int a2=0;a2<n2+n3+n4+1;a2++){
          for(int a1=0;a1<n1+n2+n3+n4+1;a1++){
            h[a1][a2][a3][a4] = -1;
          }
        }
      }
    }
    h[0][0][0][0] = 0;
    sg(n1,n2,n3,n4);
    //output k,n2,n3,n4, 0<=k<=n1
    /*for(int i1=0;i1<=n1;i1++){
      System.out.println(h[i1][n2][n3][n4]);
    }*/
    for(int i4=0;i4<=n4;i4++){
      for(int i3=0;i3<=n3;i3++){
        for(int i2=0;i2<=n2;i2++){
          for(int i1=0;i1<=n1;i1++){
            System.out.println("["+i4+", "+i3+", "+i2+", "+i1+"] "+h[i1][i2][i3][i4]);
          }
        }
      }
    }


    /*for(int i1=0;i1<=n1;i1++){
        for(int i3=10;i3<n3;i3++){
        System.out.print(3*i3+i1 - h[i1][0][i3][0]);
                System.out.print(" & ");
        }
        System.out.print(3*n3+i1 - h[i1][0][n3][0]);
            System.out.println("\\\\");
    }*/
  }

  public static int sg(int n1, int n2, int n3, int n4){
    //System.out.println(n1+" "+n2+" "+n3+" "+n4);
    if(n4<0||n3<0||n2<0||n1<0){
      return -1;
    }
    if(h[n1][n2][n3][n4]!=-1){

      //System.out.println("orz");
      return h[n1][n2][n3][n4];
    }
    HashSet<Integer> s = new HashSet<Integer>();
    //snap 4
    if(n4>0){
      s.add(sg(n1,n2,n3+1,n4-1)^p(0));
      s.add(sg(n1,n2+1,n3,n4-1)^p(0));

      s.add(sg(n1,n2+1,n3,n4-1)^p(1));
      s.add(sg(n1+1,n2,n3,n4-1)^p(1));

      s.add(sg(n1+1,n2,n3,n4-1)^p(2));
      s.add(sg(n1,n2,n3,n4-1)^p(2));
    }
    //snap 3
    if(n3>0){
      s.add(sg(n1,n2+1,n3-1,n4)^p(0));
      s.add(sg(n1+1,n2,n3-1,n4)^p(0));

      s.add(sg(n1+1,n2,n3-1,n4)^p(1));
      s.add(sg(n1,n2,n3-1,n4)^p(1));
    }
    //snap 2
    if(n2>0){
      s.add(sg(n1+1,n2-1,n3,n4)^p(0));
      s.add(sg(n1,n2-1,n3,n4)^p(0));
    }
    //blow
    for(int i4=0;i4<=n4;i4++){
      for(int i3=0;i3<=n3;i3++){
        for(int i2=0;i2<=n2;i2++){
          for(int i1=0;i1<=n1;i1++){
            if(!(i1==0&&i2==0&&i3==0&&i4==0)){
              //blow it up yo
              //System.out.println((n1-i1)+" "+(n2-i2)+" "+(n3-i3)+" "+(n4-i4));
              s.add(sg(n1-i1,n2-i2,n3-i3,n4-i4)^
                (((i2)%2)*p(1))^(((i3)%2)*p(2))^(((i4)%2)*p(3)));
            }
          }
        }
      }
    }
    //System.out.println(s);
    int k = mex(s);
    h[n1][n2][n3][n4] = k;
    return k;
  }

  public static int p(int i){
    if(i<0){
      return 0;
    }
    int[] z = {0, 1, 2, 3, 1, 4, 3, 2, 1, 4, 2, 6, 4, 1, 2, 7, 1,
        4, 3, 2, 1, 4, 6, 7, 4, 1, 2, 8, 5, 4, 7, 2, 1, 8, 6, 7,
        4, 1, 2, 3, 1, 4, 7, 2, 1, 8, 2, 7, 4, 1, 2, 8, 1, 4, 7,
        2, 1, 4, 2, 7, 4, 1, 2, 8, 1, 4, 7, 2, 1, 8, 6, 7};
    int[] y = {4,1,2,8,1,4,7,2,1,8,2,7};
    if(i>=72){
      return y[i%12];
    }
    return z[i];
  }

  public static int mex(HashSet<Integer> s){
    for(int i=0;i<s.size();i++){
      if(!s.contains(i)){
        return i;
      }
    }
    return s.size();
  }
}
```