Topics in Graph Compositions

Todd Tichenor

Topics in Graph Compositions

Todd Tichenor

Dissertation submitted to the
Eberly College of Arts and Sciences
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Mathematics

Department of Mathematics

Morgantown, WV
2015

Keywords: Graph Composition, Complete Graph,
Bell number, Aitken's array

ABSTRACT

For any discrete undirected graph G with vertex set $V(G)$ and edge set $E(G)$ (respectively), a *graph composition* of G is defined to be a partition of $V(G)$ where every element of the partition yields a connected, induced subgraph of G. This dissertation is comprised of 5 chapters. The first is a general introduction to the concept of graph compositions and a survey of previously researched work; the second focuses on the composition number of deletions of specific graphs from complete graphs; the third focuses on establishing bounds for the composition number of general graphs and the Bell number coefficients of general graphs; the fourth focuses on the connection between graph compositions and Aitken's array, a well researched array; finally, the fifth focuses on the number of compositions of graphs where the number of components is restricted.

Contents

Chapter 1

A Guide to Graph Compositions

1.1 Introduction

The idea of graph compositions was introduced by Knopfmacher and Mays in [4]. The work develops formulae for the composition number of a few families of graphs (e.g. trees, cycles, complete graphs, etc.) and defines the concept of a graph composition. A study on the composition number of unions of graphs was conducted by Ridley and Mays [7]. Graph compositions were used in [3] to gain insight into series-parallel graphs (which are relevant to the study of electrical networks) and were connected to flats of matroid cycles by Mphako-Banda [5]. Graph compositions are also related to partitions of positive integers and partitions of finite sets; specifically, the number of non-isomorphic compositions of K_n (the complete graph on n vertices) is equal to the number of partitions of the positive integer n, and the number of compositions of K_n is equal to the number of set partitions of S, where S is any set of cardinality n, for $n \in \mathbb{Z}^+$.

Many theorems in Chapter 2 were found out of a necessity for data needed for a larger problem. The problem is as follows: for any graph G, is there a method for locating an edge e_1 such that $C(G^{-e_1}) \leq C(G^{-e})$ for all $e \in E(G)$ (other than calculating $C(G^{-e})$ for every $e \in E(G)$)? Originally, this problem was not thought of as one with much substance to offer. However, repeated (failed) attempts to answer it seem to suggest otherwise. Its stubbornness and will to remain unresolved have led to insights about graph compositions that perhaps would not have been otherwise considered.

1.2 Definitions, Concepts, and Notation

Let G be a graph and $E(G)$ and $V(G)$ represent the edge and vertex sets of G (respectively). A *composition* of G is defined as a partition of $V(G)$ into vertex sets of connected induced subgraphs of G. Hence, a composition of G yields a set of connected subgraphs of G, $\{G_1, G_2, ..., G_m\}$, where $\overset{m}{\underset{i=1}{\cup}} V(G_i) = V(G)$ and $V(G_i) \cap V(G_j) = \emptyset$ for $i \neq j$ [4].

The concept of graph compositions will be illustrated below by explicitly listing all graph compositions of $K_{2,3}$.

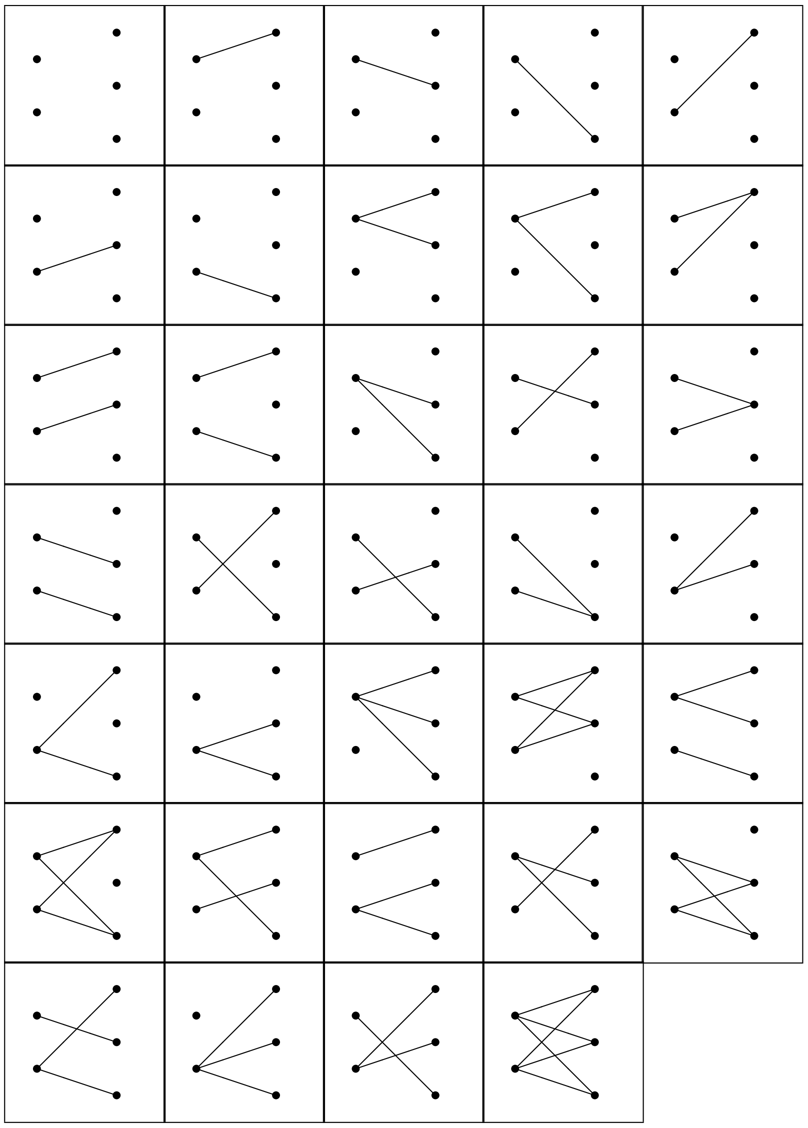


Table 1.1: All compositions of $K_{2,3}$ [4]

As seen in table 1.1, every composition of G determines a unique subgraph of G. However, the converse is not necessarily true (i.e. two distinct subgraphs of G can determine the same composition of G). An example is given below.

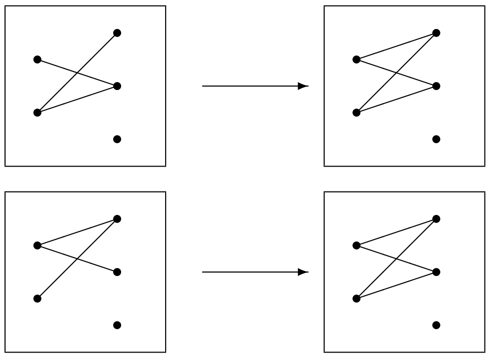


Table 1.2: Two subgraphs of $K_{2,3}$ which induce the same composition

As you can see, both subgraphs of $K_{2,3}$ listed above determine the same composition of G.

Before discussing results, some concepts, notation, and previous results used throughout the paper are discussed. For all that follows, let G be a subgraph of K_N on n vertices, $\mathcal{C}$ be a composition of G, G_i be a component of $\mathcal{C}$, and H be a subgraph of G. It follows that $\mathcal{C} = \{G_1, G_2, ..., G_m\}$ such that $\bigcup\limits_{i=1}^{m} V(G_i) = V(G)$ and $V(G_i) \cap V(G_j) = \emptyset$ for $i \neq j$. Every $V(G_i) \in \mathcal{C}$ will be referred to as a *component* of $\mathcal{C}$ and will be denoted as G_i when there is no chance of confusion (given in [4]). Furthermore, G^{-H} denotes the graph with vertex set $V(G)$ and edge set $E(G)\backslash E(H)$. We refer to this graph as "the deletion of H from G" and refer to the process of obtaining G^{-H} as "deleting H from G". If $V(G_i) \subseteq V(H)$ and the complement of $H|_{G_i}$ is disconnected, then G_i will be referred to as a *bad component* from G with respect to G^{-H}; otherwise, G_i will be referred to as a *good component* of G with respect to G^{-H}. If $\mathcal{C}$ contains a bad component from G with respect to G^{-H}, then it will be referred to as a *bad composition* of G with respect to G^{-H}; otherwise, it will be referred to as a *good composition* of G with respect to G^{-H}. An example of bad compositions and bad components for a specific graph is provided below. Any $e \in E(G)$ is said to be "contained" in a composition $\mathcal{C}$ if there exists a component of G which contains both vertices of e. Additionally, if $G^{-e_1} \cong G^{-e_2}$ for every $e_1, e_2 \in E(G)$, then we denote G^{-e} as G^{-} for all $e \in E(G)$. The *composition number* of G is the number of distinct compositions of G and is denoted by $C(G)$ [4]. $C(K_N)$ is clearly $B(N)$ [4], where $B(N)$ is the N^{th} term in the *Bell number sequence*, which counts the number of partitions of a set with cardinality n [1]. The set of all compositions of a graph G will be denoted by $\mathcal{C}(G)$ and the set of all subgraphs of G will be denoted by $\mathcal{P}(G)$. Finally, if $A \subseteq V(G)$, then $G[A]$ will denote the induced graph of G on A.

Next, we illustrate the concept of bad components and compositions. Consider the graph $K_4^{-P_4}$ seen below.

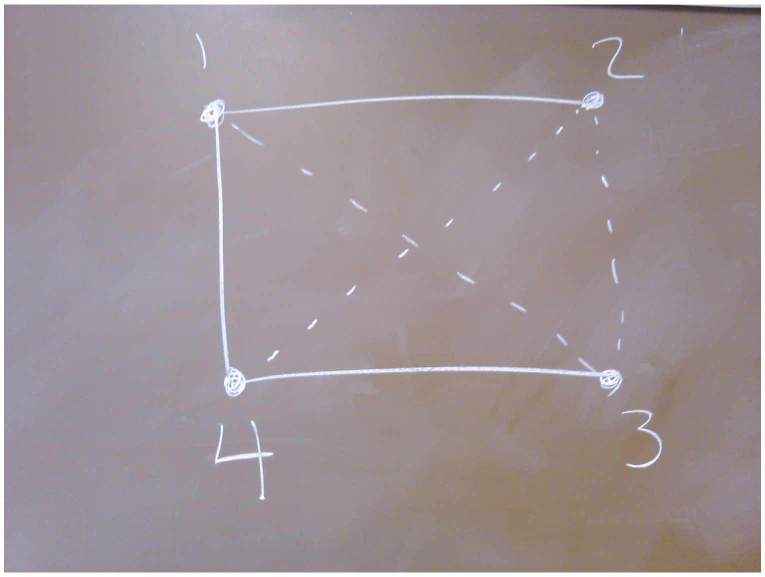


Figure 1.1: Graph of $K_4^{-P_4}$ with labelled vertices.

A component of any composition of $K_4^{-P_4}$ which is comprised solely of edges from the edge set $\{\{1,3\},\{2,3\},\{2,4\}\}$ will be a bad component by definition. Next, illustrations are provided for all compositions of K_4; they will be grouped into bad compositions (i.e. compositions of $K_4 \backslash K_4^{-P_4}$) and good compositions (i.e. compositions of $K_4 \cap K_4^{-P_4}$).

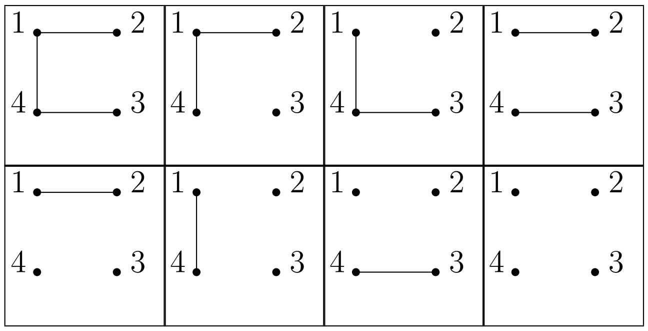


Figure 1.2: Good compositions of K_4 with respect to $K_4^{-P_4}$

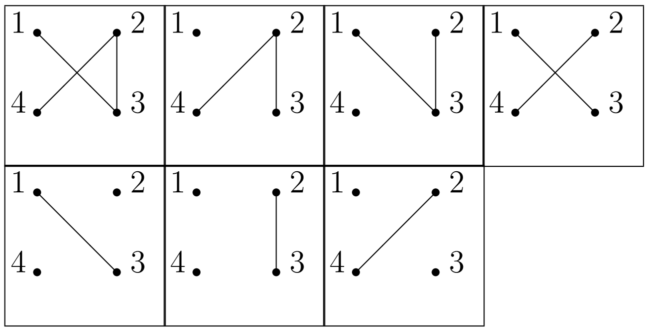


Figure 1.3: Bad compositions of K_4 with respect to $K_4^{-P_4}$

1.3 Previous Work

Theorem 1. *(Knopfmacher and Mays, [4]) If $G = G_1 \cup G_2$ and there are no edges incident to a vertex from G_1 and G_2 (i.e. G is disconnected), then $C(G) = C(G_1) \cdot C(G_2)$. The same result holds if G_1 and G_2 have exactly one vertex in common.*

Theorem 2. *(Knopfmacher and Mays, [4]) $C(K_N) = B(N)$*

Theorem 3. *(Knopfmacher and Mays, [4]) $C(K_N^-) = B(N) - B(N-2)$*

For the following theorem, some background information from [2] is necessary. Define a matrix M where all entries are from the set $\{0, 1\}$ as a $(0, 1)$ matrix. Consider all $2 \times 2(0, 1)$ matrices. Let P and Q be 2×2 $(0, 1)$ matrices and define an equivalence relation $\sim$ by $P \sim Q$ if and only if Q can be obtained by performing a finite number of row and column switches on P. There will be 7 equivalence classes formed by this relation. Each equivalence class will be represented by some upper case letter. The label for each equivalence class along with a representative for the class is listed below using notation from [2]:

$$I : \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\Gamma : \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \; C : \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$T : \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \; L : \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

$J : \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ $0 : \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

For a matrix P and a set of matrices α, we say that P *avoids* α if and only if P has no submatrices in α.

Finally, $S(n, k)$ will be used to denote the number of partitions of a set of n elements into k nonempty parts (i.e. $S(n, k)$ is a Stirling number of the 2^{nd} kind). A portion of the array of Stirling numbers may be found on page 41 of this dissertation.

Theorem 4. *(Ju and Seo, [2]) The number of $k \times n$ matrices which avoid Γ is given by $\phi(k, n; \Gamma) = \sum_{m=0}^{min\{n,k\}} m! \cdot S(n + 1, m + 1)S(k + 1, m + 1)$.*

Remark 1. *Note that setting $k = n$ in Theorem 4 yields the number of $n \times n$ matrices which avoids Γ is $\sum_{m=0}^{n} m! \cdot [S(n + 1, m + 1)]^2$.*

Chapter 2

Involving the Composition Numbers of Specific Graphs

The focus of this section is on the composition number of "dense graphs" (i.e. graphs whose complements have small edge sets); more specifically, we concern ourselves with the composition number of deletions of certain families of graphs (e.g. paths, cycles, stars, etc.) from complete graphs. The motivation for this study came from a result in [4] which states $C(K_N^-) = B(N) - B(N-2)$ for $N \geq 2$. The article goes on to state that adjacency of the edges must be taken into account when deleting more than a single edge from K_n, but does not discuss the topic any further. We conclude this introduction by stating that Theorem 5 was developed by examining the patterns of Theorems 6, 8, 10, and 12.

2.1 The Deletion of General Graphs from Complete Graphs

Theorem 5. *Let G be a subgraph of K_N for $|V(G)| = n$ and $b_{j,k,n}$ represent the number of ways of choosing k disjoint bad components of K_N^{-G} such that the cardinality of the union of vertices of all aforementioned bad components is j. If $b_{j,n}$ is defined by $b_{j,n} = \sum_{k=0}^{n} (-1)^k \cdot b_{j,k,n}$, then $C(K_N^{-G}) = \sum_{j=0}^{n} b_{j,n} B(N-j)$.*

Proof: We begin by denoting the set of all compositions of K_N by Γ and the number of disjoint bad components contained in γ by $B(\gamma)$ for all $\gamma \in \Gamma$.

$$C(K_N^{-G}) = \sum_{\substack{\gamma \in \Gamma \\ B(\gamma)=0}} 1 = \sum_{\substack{\gamma \in \Gamma \\ B(\gamma)=0}} 1 + \sum_{\substack{\gamma \in \Gamma \\ B(\gamma)\neq 0}} 0 = \sum_{\gamma \in \Gamma} \sum_{k=0}^{B(\gamma)} (-1)^k \binom{B(\gamma)}{k}$$

since the alternating sum of the k^{th} row of Pascal's Triangle is 0 for $k > 0$ and 1 for $k = 0$. Note that $\binom{B(\gamma)}{k}$ will count the number of times γ is counted as a composition of K_N with at least k disjoint bad components for a fixed γ and k.

$$\sum_{\gamma \in \Gamma} \sum_{k=0}^{B(\gamma)} \binom{B(\gamma)}{k} = \sum_{j=0}^{n} \sum_{k=0}^{n} b_{j,k,n} B(N-j),$$

so

$$\begin{aligned}
C(K_N^{-G}) &= \sum_{\gamma\in\Gamma}\sum_{k=0}^{B(\gamma)}(-1)^k\binom{B(\gamma)}{k} \\
&= \sum_{j=0}^{n}\sum_{k=0}^{n}(-1)^k b_{j,k,n}B(N-j) \\
&= \sum_{j=0}^{n} b_{j,n}B(N-j).
\end{aligned}$$

Remark 2. *It is trivial (but none-the-less important) to note that* $b_{0,n} = 1$ *(since you can choose 0 vertices exactly 1 way) and* $b_{1,n} = 0$ *(since there is no way to choose 1 vertex to be a single component) for all* $n \in \mathbb{Z}^+$*. Also,* $b_{j,n}$ *is undefined for* $j > n$ *(since you cannot choose more than* n *vertices from a set of* n *vertices).*

Remark 3. *It is also of merit to note that Theorem 3 is the special case of Theorem 5 when* $G = K_2$*.*

2.2 Deletions of Paths From Complete Graphs

Theorem 6. *Let* P_n *denote the path with* $n \leq N$ *vertices. If* $p_{j,k,n}$ *denotes the number of ways of choosing* k *disjoint bad components from* $K_N^{-P_n}$ *such that the number of vertices of all aforementioned bad components is* j *and* $p_{j,n} = \sum_{k=0}^{n}(-1)^k \cdot p_{j,k,n}$*, then* $C(K_N^{-P_n}) = \sum_{j=0}^{n} p_{j,n}B(N-j)$ *and* $p_{j,n} = p_{j,n-1} - p_{j-2,n-2} - p_{j-3,n-3}$ *for* j *and* $n \geq 3$.

Proof: $C(K_N^{-P_n}) = \sum_{j=0}^{n} p_{j,n}B(N-j)$ is a result of application of Theorem 5 on $K_N^{-P_n}$. If $\mathcal{C}$ is a composition of K_N, then $\mathcal{C}$ will **not** be a composition of $K_N^{-P_n}$ if and only if $\mathcal{C}$ contains a bad component of $K_N^{-P_n}$.

Note if we delete a subpath of length $t \geq 3$ of P_n from K_N, then the complement is necessarily connected. The endpoints of P_{t+1} will be adjacent and every interior point of the path will be adjacent to both endpoints. Hence, the only bad components that exist when deleting P_n from K_N are subpaths of P_n of length 1 or 2.

Next, define $S_{j,n} = \sum_{\text{k is even}} p_{j,k,n}$ and $L_{j,n} = \sum_{\text{k is odd}} p_{j,k,n}$. $S_{j,n}$ represents the number of ways of choosing an even number of disjoint bad components from $V(P_n)$ where the cardinality of the union of bad vertices is j and $L_{j,n}$ analogously represents the number of ways of choosing an odd number of disjoint bad components from $V(P_n)$ where the cardinality of the union of bad vertices is j. Note that $p_{j,n} = S_{j,n} - L_{j,n}$.

If u represents one of the terminal vertices of the deleted path and C is a component which contains u, then one of three cases must occur:

1. C is not a bad component. There are $S_{j,n-1}$ [$L_{j,n-1}$] ways of choosing an even [odd] number of disjoint bad components from $V(P_N)$ such that the cardinality of union of vertices of components which do not include C is j.

2. C is a 2-element bad component. There are $L_{j-2,n-2}$ [$S_{j-2,n-2}$] ways of choosing an even [odd] number of disjoint bad components from $V(P_n)$ such that the cardinality of union of vertices of components which include C is j.

3. C is a 3-element bad component. There are $L_{j-3,n-3}$ [$S_{j-3,n-3}$] ways of choosing an even [odd] number of disjoint bad components from

$V(P_n)$ such that the cardinality of union of vertices of components which include C is j.

The above "world encompassing" cases give us

$$S_{j,n} = S_{j,n-1} + L_{j-2,n-2} + L_{j-3,n-3}$$
$$L_{j,n} = L_{j,n-1} + S_{j-2,n-2} + S_{j-3,n-3}$$

which yields

$$p_{j,n} = p_{j,n-1} - p_{j-2,n-2} - p_{j-3,n-3}.$$

A table of values for $p_{j,n}$ is given below:

n \ j	0	1	2	3	4	5	6	7	...	
0	1	-	-	-	-	-	-	-	...	
1	1	0	-	-	-	-	-	-	...	
2	1	0	-1	-	-	-	-	-	...	
3	1	0	-2	-1	-	-	-	-	...	
4	1	0	-3	-2	1	-	-	-	...	
5	1	0	-4	-3	3	2	-	-	...	
6	1	0	-5	-4	6	6	0	-	...	
7	1	0	-6	-5	10	12	-1	-3	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table 2.1: Table of Values of $p_{j,n}$

Remark 4. *$p_{j,n} = -(n-1)$ for $j = 2$ since this implies the bad component being chosen from $V(P_n)$ is an edge and there are $n-1$ ways of choosing an edge from P_n.*

Theorem 7. *If $F(x,y) = \sum\limits_{j=0}^{\infty}\sum\limits_{n=0}^{\infty} p_{j,n}x^n y^j$, then $F(x,y) = \frac{1}{1-x+x^2y^2+x^3y^3}$.*

Proof: Using our recurrence relation for $p_{j,n}$ we get

$$p_{j,n}x^n y^j = x \cdot p_{j,n-1}x^{n-1}y^j - x^2y^2 p_{j-2,n-2}x^{n-2}y^{j-2} - x^3y^3 \cdot p_{j-3,n-3}x^{n-3}y^{j-3}$$

for j and $n \geq 3$. This in turn yields

$$\sum_{j=3}^{\infty}\sum_{n=3}^{\infty} p_{j,n}x^n y^j = x\sum_{j=3}^{\infty}\sum_{n=2}^{\infty} p_{j,n}x^n y^j - x^2y^2\sum_{j=1}^{\infty}\sum_{n=1}^{\infty} p_{j,n}x^n y^j - x^3y^3\sum_{j=0}^{\infty}\sum_{n=0}^{\infty} p_{j,n}x^n y^j.$$

If we consider $p_{j,n} = 0$ for $j > n$, then

$$\sum_{j=3}^{\infty}\sum_{n=0}^{\infty} p_{j,n}x^n y^j = x\sum_{j=3}^{\infty}\sum_{n=0}^{\infty} p_{j,n}x^n y^j - x^2y^2\sum_{j=1}^{\infty}\sum_{n=0}^{\infty} p_{j,n}x^n y^j - x^3y^3\sum_{j=0}^{\infty}\sum_{n=0}^{\infty} p_{j,n}x^n y^j.$$

Rewriting the entire equation in terms of $F(x,y)$ yields

$$F(x,y) - \sum_{n=0}^{\infty} p_{0,n}x^n - \sum_{n=0}^{\infty} p_{1,n}x^n y - \sum_{n=0}^{\infty} p_{2,n}x^n y^2$$
$$= x(F(x,y) - \sum_{n=0}^{\infty} p_{0,n}x^n - \sum_{n=0}^{\infty} p_{1,n}x^n y - \sum_{n=0}^{\infty} p_{2,n}x^n y^2) - x^2y^2(F(x,y) - \sum_{n=0}^{\infty} p_{0,n}x^n) - x^3y^3F(x,y).$$

Observation of Remark 2 and manipulation of the equation yields

$$F(x,y)(1 - x + x^2y^2 + x^3y^3) = 1,$$

which leads directly to the desired result.

2.3 Deletions of Cycles From Complete Graphs

After examination of $C(K_N^{-P_n})$, it becomes a natural extension to examine $C(K_N^{-C_n})$ where C_n denotes the cycle with n vertices ($n \geq 3$). Hence, this section is concerned with the study of $C(K_N^{-C_n})$.

Theorem 8. *Let $n \leq N$ and $p_{j,n}$ denote the coefficients defined in Theorem 6. If $c_{j,k,n}$ denotes the number of ways of choosing k disjoint bad components of $K_N^{-C_n}$, where the number of vertices of the bad components is j and $c_{j,n} = \sum_{k=0}^{n}(-1)^k \cdot c_{j,k,n}$, then $C(K_N^{-C_n}) = \sum_{j=0}^{n} c_{j,n}B(N-j)$ and $c_{j,n} = p_{j,n-1} - 2 \cdot p_{j-2,n-2} - 3 \cdot p_{j-3,n-3}$ for j and $n \geq 3$, and $n \neq j$ for $n \in \{3,4\}$.*

Proof: The result $C(K_N^{-C_n}) = \sum_{j=0}^{n} c_{j,n}B(N-j)$ follows from application of Theorem 5 on $K_N^{-C_n}$. If $\mathcal{C}$ is a composition of K_N, then $\mathcal{C}$ will **not** be a composition of $K_N^{-C_n}$ if and only if $\mathcal{C}$ contains a bad component of $K_N^{-C_n}$.

Next we describe the bad components of $K_N^{-C_n}$. The graph $K_N^{-C_n}$ will inherit subpaths of length 1 and 2 of C_n as bad components since you can always find some $P_n \subseteq C_n$.

Consider a composition of K_N for which all of C_n has been deleted. It is easily verified that if $n > 4$, then the composition is good. Hence, the only remaining bad components of $K_N^{-C_n}$ are 3 or 4 element cyclic components.

If we define $S_{j,n} = \sum_{\text{k is even}} c_{j,k,n}$ and $L_{j,n} = \sum_{\text{k is odd}} c_{j,k,n}$, then $S_{j,n}$ represents the number of ways of choosing an even number of disjoint *bad* components from $V(C_n)$ where the cardinality of the union of vertices is j. $L_{j,n}$ analogously represents the number of ways of choosing an odd number of disjoint *bad* components from $V(C_n)$ where the cardinality of the union of vertices is j. Note $c_{j,n} = S_{j,n} - L_{j,n}$.

If we let $w \in V(C_n)$ and fix the component, $C_w \in \mathcal{C}$, which contains w, then one of four cases occur:

1. C_w is not a bad component. There are $S_{j,n-1}$ $[L_{j,n-1}]$ ways of choosing an even [odd] number of disjoint bad components from $V(C_N)$ (with cardinality of the union of vertices j) which do not include C_w.

2. C_w is a 2-element bad component. There are $2{\cdot}L_{j-2,n-2}$ $[2{\cdot}S_{j-2,n-2}]$ ways of choosing an even [odd] number of disjoint bad components from $V(C_n)$ (with cardinality of the union of vertices j) which include C_w.

3. C_w is a 3-element bad component. If C_w is comprised of just a path ($n > 3$), then there are $3{\cdot}L_{j-3,n-3}$ $[3{\cdot}S_{j-3,n-3}]$ ways of choosing an even [odd] number of disjoint bad components from $V(C_n)$ (with cardinality of the union of vertices j) which include C_w. If C_w is a cycle, then there is exactly one way of choosing an odd number of bad components and no way of choosing an even number of bad components.

4. C_w is a 4-element *bad* component. There is exactly one way of choosing an odd number of bad components and no way of choosing an even number of bad components.

The above "world encompassing" cases give us

$$S_{j,n} = S_{j,n-1} + 2 \cdot L_{j-2,n-2} + 3 \cdot L_{j-3,n-3}$$
$$L_{j,n} = L_{j,n-1} + 2 \cdot S_{j-2,n-2} + 3 \cdot S_{j-3,n-3}$$

which yields

$$c_{j,n} = p_{j,n-1} - 2 \cdot p_{j-2,n-2} - 3 \cdot p_{j-3,n-3}.$$

The first few values of $c_{j,n}$ are shown in the table below:

n \ j	0	1	2	3	4	5	6	7	...	
0	1	-	-	-	-	-	-	-	...	
1	1	0	-	-	-	-	-	-	...	
2	1	0	-1	-	-	-	-	-	...	
3	1	0	-3	-1	-	-	-	-	...	
4	1	0	-4	-4	1	-	-	-	...	
5	1	0	-5	-5	5	5	-	-	...	
6	1	0	-6	-6	9	12	1	-	...	
7	1	0	-7	-7	14	21	0	-7	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table 2.2: Table of Values for $c_{j,n}$

Remark 5. *$c_{2,n} = -n$ for $n > 2$ since $c_{2,n}$ is the number of ways of choosing a single edge from C_n.*

Theorem 9. *If $G(x,y) = \sum_{j=0}^{\infty}\sum_{n=0}^{\infty} c_{j,n}x^n y^j$, then $G(x,y) = 1 + x^2y^2 + 2x^3y^3 - x^4y^4 + \frac{x-2x^2y^2-3x^3y^3}{1-x+x^2y^2+x^3y^3}$.*

Proof: Given that $c_{j,n} = 0$ for $j > n$ and $c_{1,n} = 0$, it is easily established that

$$\sum_{j=3}^{\infty}\sum_{n=3}^{\infty} c_{j,n}x^n y^j = G(x,y) - \sum_{n=0}^{\infty} c_{0,n}x^n - y^2\sum_{n=2}^{\infty} c_{2,n}x^n$$

$$= \sum_{j=3}^{\infty} c_{j,3}x^3 y^j + \sum_{j=3}^{\infty}\sum_{n=4}^{\infty} c_{j,n}x^n y^j$$

$$= c_{3,3}x^3y^3 + c_{4,4}x^4y^4 + y^3\sum_{n=4}^{\infty} c_{3,n}x^n + \sum_{j=4}^{\infty}\sum_{n=5}^{\infty} c_{j,n}x^n y^j.$$

Substitution of known values and sums yields

$$-x^3y^3 + x^4y^4 + \frac{y^3x^4(3x-4)}{(1-x)^2} + \sum_{j=4}^{\infty}\sum_{n=5}^{\infty} c_{j,n}x^n y^j.$$

Application of Theorem 8 yields

$$-x^3y^3 + x^4y^4 + \frac{y^3x^4(3x-4)}{(1-x)^2} + x\sum_{j=4}^{\infty}\sum_{n=4}^{\infty} p_{j,n}x^n y^j - 2x^2y^2\sum_{j=2}^{\infty}\sum_{n=3}^{\infty} p_{j,n}x^n y^j - 3x^3y^3\sum_{j=2}^{\infty}\sum_{n=2}^{\infty} p_{j,n}x^n y^j$$

$$= -x^3y^3 + x^4y^4 + \frac{y^3x^4(3x-4)}{(1-x)^2} + x\left[F(x,y) - \sum_{n=0}^{\infty} p_{0,n}x^n - y^2\sum_{n=2}^{\infty} p_{2,n}x^n - y^3\sum_{n=3}^{\infty} p_{3,n}x^n\right]$$

$$= -2x^2y^2\left[y^2\sum_{n=3}^{\infty} p_{2,n}x^n + F(x,y) - \sum_{n=0}^{\infty} p_{0,n}x^n] - 3x^3y^3[F(x,y) - \sum_{n=0}^{\infty} p_{0,n}x^n\right].$$

The above equation along with substitution and algebraic manipulation yields the result

$$\begin{aligned} G(x,y) &= 1 - x^2y^2 - x^3y^3 + x^4y^4 + \frac{(1-x^2)(2x^2y^2+3x^3y^3-2x^4y^4)}{(1-x)^2} + F(x,y)[x - 2x^2y^2 - 3x^3y^3] \\ &= 1 + x^2y^2 + 2x^3y^3 - x^4y^4 + \frac{x - 2x^2y^2 - 3x^3y^3}{1 - x + x^2y^2 + x^3y^3}. \end{aligned}$$

2.4 Deletions of Star Graphs From Complete Graphs

Recall that a star graph on $n \geq 2$ vertices (denoted S_n) is the tree of n vertices and $n-1$ edges where one vertex (which we call the central vertex) is adjacent to all other vertices.

Theorem 10. *If* $1 \leq n \leq N-1$, *then* $C(K_N^{-S_{n+1}}) = B(N) - \sum\limits_{k=1}^{n} \binom{n}{k} \cdot B(N-(k+1))$.

Proof: If $\mathcal{C}$ is a composition of K_N, then $\mathcal{C}$ is *not* a composition of $K_N^{-S_{n+1}}$ if and only if $\mathcal{C}$ contains a bad component of $K_N^{-S_{n+1}}$. Every bad component is isomorphic to some S_{k+1} for $k \leq n$, and the vertex sets of all distinct bad components will intersect exactly at the central vertex. So it is impossible to simultaneously choose more than 1 unique disjoint bad component. The number of compositions which contain exactly one bad component G_i, where $|V(G_i)| = k+1$, is $\binom{n}{k} \cdot B(N-(k+1))$. Hence, $C(K_N^{-S_{n+1}}) = B(N) - \sum\limits_{k=1}^{n} \binom{n}{k} \cdot B(N-(k+1))$.

Remark 6. *Note* $C(K_N^{-S_{n+1}})$ *can be written as* $\sum\limits_{j=0}^{n} s_{j,n} B(N-j)$, *where* $s_{j,n}$ *is undefined for* $j > n$, $s_{0,n} = 1$ *for all* n, $s_{1,n} = 0$ *for all* $n > 0$, *and* $s_{j,n} = -\binom{n-1}{j-1}$ *for* j *and* $n \geq 2$ *and* $j \leq n$. *Having an explicit formula for*

$s_{j,n}$ makes it trivial to establish that $s_{j,n} = s_{j-1,n-1} + s_{j,n-1}$ for j and $n \geq 3$. This is important for no other reason than to recover a generating function in order to keep with the structure of the paper thus far.

Remark 7. *Note also $C(K_N^{-S_N}) = B(N-1)$ since $K_N^{-S_N} \cong K_{N-1}$. Setting $N = N+1$ yields $C(K_{N+1}^{-S_{N+1}}) = B(N)$ and applying Theorem 10 to $C(K_{N+1}^{-S_{N+1}})$ yields $C(K_{N+1}^{-S_{N+1}}) = B(N+1) - \sum_{k=1}^{N} \binom{N}{k} B(N-k)$. Setting the two equations equal yields $B(N+1) = B(N) + \sum_{k=1}^{N} \binom{N}{k} B(N-k) = \sum_{k=0}^{N} \binom{N}{k} B(N-k)$, a well-known recursion of the Bell numbers.*

Theorem 11. *If $S(x,y) = \sum_{j=0}^{\infty}\sum_{n=0}^{\infty} s_{j,n} x^n y^j$, then $S(x,y) = \frac{1-x-xy-x^2y^2}{1-2x+x^2-xy+x^2y}$.*

Proof: Using Remark 6, algebraic manipulation, and known values for $s_{j,n}$, we get

$$S(x,y) - \sum_{n=0}^{\infty} s_{0,n} x^n - y^2 \sum_{n=0}^{\infty} s_{2,n} x^n$$

$$= \sum_{j=3}^{\infty}\sum_{n=3}^{\infty} s_{j,n} x^n y^j$$

$$= xy \sum_{j=2}^{\infty}\sum_{n=2}^{\infty} s_{j,n} x^n y^j + x \sum_{j=3}^{\infty}\sum_{n=3}^{\infty} s_{j,n} x^n y^j$$

$$= xy \cdot (S(x,y) - \sum_{n=0}^{\infty} s_{0,n} x^n) + x \cdot (S(x,y) - \sum_{n=0}^{\infty} s_{0,n} x^n - y^2 \cdot \sum_{n=2}^{\infty} s_{2,n} x^n).$$

This yields

$$S(x,y)[1-x-xy]$$

$$= (1-x-xy)\sum_{n=0}^{\infty} s_{0,n}x^n + y^2(1-x)\sum_{n=2}^{\infty} s_{2,n}x^n$$

$$= \frac{1-x-xy-x^2y^2}{1-x}.$$

Hence, $S(x,y) = \frac{1-x-xy-x^2y^2}{(1-x)(1-x-xy)} = \frac{1-x-xy-x^2y^2}{1-2x+x^2-xy+x^2y}$.

2.5 Deletions of "Disjoint" Graphs From Complete Graphs

In graph theory, a k-matching is a graph which consists of k vertex disjoint edges and is denoted by M_k. We will use the notation D_k when referring to a k-matching.

Theorem 12. *If $2n \leq N$, then $C(K_N^{-D_n}) = \sum_{k=0}^{n} (-1)^k \binom{n}{k} \cdot B(N-2k)$.*

Proof: If $\mathcal{C}$ is a composition of K_N, then C is *not* a composition of $K_N^{-D_n}$ if and only if $\mathcal{C}$ contains a bad component of $K_N^{-D_n}$.

If we let Γ represent the set of all compositions of K_N, $B(\gamma)$ represent the number of bad components of γ for all $\gamma \in \Gamma$, and $A_{k,n}$ represent the set of compositions of K_N with at least k *bad* components, then, as in Theorem 6,

$$C(K_N^{-D_n}) = \sum_{\gamma\in\Gamma}\sum_{k=0}^{B(\gamma)}(-1)^k\binom{B(\gamma)}{k}.$$

Using

$$|A_{k,n}| = \binom{n}{k}B(N-2k),$$

we get the result

$$C(K_N^{-D_n}) = \sum_{\gamma\in\Gamma}\sum_{k=0}^{B(\gamma)}(-1)^k\binom{B(\gamma)}{k} = \sum_{k=0}^{n}(-1)^k\cdot|A_{k,n}| = \sum_{k=0}^{n}(-1)^k\cdot\binom{n}{k}B(N-2k).$$

Remark 8. *Setting $d_{j,n} = (-1)^j\binom{n}{j}$ yields $C(K_N^{-D_n}) = \sum_{j=0}^{n} d_{j,n}\cdot B(N-j)$ for all j and $n\in\mathbb{Z}^+$. As with Remark 6, having the explicit formula for $d_{j,n}$ allows us to establish $d_{j,n} = d_{j,n-1} - d_{j-1,n-1}$ for j and $n\geq 1$.*

Theorem 13. *If $D(x,y) = \sum_{j=0}^{\infty}\sum_{n=0}^{\infty} d_{j,n}x^ny^j$, then $D(x,y) = \frac{1}{1-x+xy}$.*

Proof: Using Remark 8, we obtain the result

$$D(x,y) - \sum_{n=0}^{\infty} d_{0,n}x^n\sum_{j=1}^{\infty}\sum_{n=1}^{\infty} d_{j,n}x^ny^j$$

$$= x\sum_{j=1}^{\infty}\sum_{n=1}^{\infty} d_{j,n}x^ny^j - xy\sum_{j=0}^{\infty}\sum_{n=0}^{\infty} d_{j,n}x^ny^j$$

$$= x(D(x,y) - \sum_{n=0}^{\infty} d_{0,n}x^n) - xy\cdot D(x,y).$$

This implies

$$D(x,y)[1-x+xy] = (1-x)\sum_{n=0}^{\infty} d_{0,n}x^n$$

$$= \frac{1-x}{1-x} = 1.$$

Therefore, $D(x,y) = \frac{1}{1-x+xy}$.

2.6 Miscellaneous Results

Results in this section were found during the course of research of the problem stated in the introduction of Chapter 1 , but seemed inappropriate to include in the main body of this chapter.

Theorem 14. *Let $K_{a,b}$ denote the complete bipartite graph with vertex bipartition sets of sizes a and b. If $0 \leq m \leq N$, then $C(K_N^{-K_{m,N-m}}) = B(m) \cdot B(N-m)$.*

Proof: The graph $K_N^{-K_{m,N-m}}$ consists of exactly 2 disjoint connected components K_m and K_{N-m}. Application of Theorems 1 and 2 yields $C(K_N^{-K_{m,N-m}}) = C(K_m) \cdot C(K_{N-m}) = B(m) \cdot B(N-m)$.

Theorem 15. *If $n \in \mathbb{Z}^+$, then $C(K_{n,n}) = \sum\limits_{k=0}^{n} k! \cdot [S(n+1,k+1)]^2$.*

Proof: Let A and B be 2 disjoint sets of vertices such that $|A| = |B| = n$. The graph $K_{n,n}$ satisfies the criteria $V(G) = A \cup B$ and $E(G) = \{\{u,v\}|u \in A, v \in B\}$ by definition. Label all vertices of A using each element of the set $\{1,...,n\}$ exactly once. Label all vertices of B in a similar manner. Let $i \in A$

and $j \in B$. Define an adjacency matrix M_H for $H \subseteq K_{n,n}$ as follows: M_H is an $n \times n$ matrix where $m_{i,j}$ denotes the $(i,j)^{th}$ entry of M_H and

$$m_{i,j} = \begin{cases} 1, & \text{i is adjacent to j} \\ 0, & \text{otherwise} \end{cases}$$

Let $\mathcal{C}$ be a composition of $K_{n,n}$ and consider $M_{\mathcal{C}}$, the adjacency matrix of the subgraph of $K_{n,n}$ determined by $\mathcal{C}$. The resulting matrix cannot have any 2×2 submatrices which contain three 1s and one 0 since it must represent an induced subgraph of $K_{n,n}$. This yields that the adjacency matrix for every composition of $K_{n,n}$ is an $n \times n$ $(0,1)$ matrix where every 2×2 submatrix avoids Γ. It is trivial to note that $C(K_{n,n})$ can be calculated simply by counting all matrices of the form described above. So by Remark 1, $C(K_{n,n}) = \sum_{k=0}^{n} k! \cdot [S(n+1,k+1)]^2$.

Chapter 3

Compositions of General Graphs

The results in the following sections involve the composition number of general graphs. In particular, we examine the composition number of subgraphs of any general graph, bounds on K_N^{-G} for any graph G which is a subgraph of K_N, and bounds on the Bell number coefficients which can be used to determine $C(K_N^{-G})$. Results on the bounds of the Bell number coefficients of K_N^{-G} came about by examining the Bell number coefficients of K_N^{-G} for certain G; results on the bounds of K_N^{-G} were found by examining illustrations of all graphs on N vertices for certain values of N that were arranged by composition number in ascending order.

3.1 Elementary Results

Theorem 16. *If G is a graph and H is a subgraph of G such that $|E(H)| < |E(G)|$, then $C(H) < C(G)$.*

Proof: Since H is a subgraph of G, then $E(H) \subset E(G)$. This implies that every composition of H will also be a composition of G (i.e. $C(H) \leq C(G)$). Also, there exists an $e \in E(G) \backslash E(H)$. Let $\mathcal{C}$ be a composition of G which has e as a component. $\mathcal{C}$ cannot be a composition of H since $e \notin E(H)$. Hence, $C(H) < C(G)$.

Theorem 17. *If G is a graph and H is a proper subgraph of G such that $|E(H)| + k = |E(G)|$, then $C(H) \geq \frac{1}{2^k} C(G)$.*

Proof: We prove this theorem via induction. If G is a graph and H is a proper subgraph of G such that $|E(H)| + 1 = |E(G)|$, then there exists an $e \in E(G) \backslash E(H)$. For every composition $\mathcal{C}$ of G that contains e, either $\mathcal{C}$ is or is not a composition of H. Assume there are exactly s compositions which contain e that are compositions of H and t compositions which contain e that are not compositions of H. If $\mathcal{C}$ is not a composition of H which contains e, then deleting e will yield a composition of H which obviously does not contain e. This one-to-one correspondence yields $C(H) = s + t$ and $C(G) = 2t + s \leq 2t + 2s = 2C(H)$, which leads directly to the result of Theorem 17 for $k = 1$.

Assume that the theorem holds for $k \geq 1$. If G is a graph and H is a subgraph of G such that $|E(H)| + (k + 1) = |E(G)|$, then there exists $e \in E(G) \backslash E(H)$. $H \subset G^{-e}$ and $|E(G^{-e})| = |E(H)| + k$. The result $C(H) \geq \frac{1}{2^k} C(G^{-e})$ follows from our induction hypothesis. Additionally, $G^{-e} \subset G$ and

$|E(G^{-e})|+1 = |E(G)|$ yields $C(G^{-e}) \geq \frac{1}{2}C(G)$. Hence, $C(H) \geq \frac{1}{2^k}C(G^{-e}) \geq \frac{1}{2^{k+1}}C(G)$ and Theorem 17 is true.

3.2 The Bell Number Coefficients For Graph Compositions

For any subgraph G of K_N, let $b_{j,k,n}$ denote the number of ways of choosing k disjoint bad components from K_N^{-G} such that the cardinality of the union of vertices of all bad components is j. Define $b_{j,n} = \sum\limits_{k=0}^{n}(-1)^k b_{j,k,n}$. Theorem 5 states that $C(K_N^{-G}) = \sum\limits_{j=0}^{n} b_{j,n}B(N-j)$. Hence, a solution for $C(K_N^{-G})$ of this form can always be found. We will refer to these $b_{j,n}$ as the *Bell number coefficients* of K_N^{-G}. The following theorems involve the Bell number coefficients of K_N^{-G}.

Theorem 18. *If G is a subgraph of K_N such that $|E(G)| = k$ and $|V(G)| = n$ and $b_{j,n}$ represents the j^{th} Bell number coefficient of K_N^{-G}, then $|b_{j,n}| \leq \binom{k}{j-1}$.*

Proof: If we define $S_{j,n} = \sum\limits_{\text{k is even}} b_{j,k,n}$ and $L_{j,n} = \sum\limits_{\text{k is odd}} b_{j,k,n}$, then $b_{j,n} = S_{j,n} - L_{j,n}$.

Next, define an equivalence relation $\sim$ on $\mathcal{P}(G)$ as follows: for any $A, B \in \mathcal{P}(G)$, $A \sim B$ if and only if $G[V(A)] = G[V(B)]$. Every equivalence class will contain exactly 1 member that is a composition of G and at least 1 member that is a union of trees. It is trivial to notice that $b_{j,n}$ is maximized if every subgraph of G with exactly j vertices and an even number of components is a tree and an induced subgraph of G (i.e. $S_{j,n} = \binom{k}{j-1}$ and $L_{j,n} = 0$) and will be minimized if every subgraph of G with exactly j vertices and an odd number of components is a tree and an induced subgraph of G (i.e. $L_{j,n} = \binom{k}{j-1}$ and $S_{j,n} = 0$). Hence, $|b_{j,n}| \leq \binom{k}{j-1}$.

Theorem 19. *If G_1 and G_2 are subgraphs of K_N such that $|V(G_1)| = n_1$, $|V(G_2)| = n_2$, and $V(G_1) \cap V(G_2) = \emptyset$ and b_{j,n_1} denotes the Bell number coefficients of $C(K_N^{-G_1})$, b_{j,n_2} the Bell number coefficients of $K_N^{-G_2}$, and $b_{j,n_1 \oplus n_2}$ the Bell number coefficients of $K_N^{-(G_1 \cup G_2)}$, then $b_{j,n_1 \oplus n_2} = \sum\limits_{k=0}^{j} b_{k,n_1} \cdot b_{j-k,n_2}$.*

Proof: Define b_{j,k,n_1} to be the number of compositions of G_1 with k disjoint bad components, where the cardinality of the union of vertex sets of said disjoint bad components is j. Define b_{j,k,n_2} and $b_{j,k,n_1 \oplus n_2}$ analogously for G_2 and $G_1 \cup G_2$ (respectively). Define

$$S(b_{j,n_1}) = \sum_{\text{k is even}} b_{j,k,n_1}$$

and

$$L(b_{j,n_1}) = \sum_{\text{k is odd}} b_{j,k,n_1}.$$

If $S(b_{j,n_2})$, $L(b_{j,n_2})$, $S(b_{j,n_1 \oplus n_2})$, and $L(b_{j,n_1 \oplus n_2})$ are defined analogously, then $b_{j,i} = S_{j,i} - L_{j,i}$ for $i \in \{n_1, n_2, n_1 \oplus n_2\}$.

$$\begin{aligned} S_{j,n_1 \oplus n_2} &= \sum_{k=0}^{j} S_{k,n_1} \cdot S_{j-k,n_2} + \sum_{k=0}^{j} L_{k,n_1} \cdot L_{j-k,n_2} \\ L_{j,n_1 \oplus n_2} &= \sum_{k=0}^{j} S_{k,n_1} \cdot L_{j-k,n_2} + \sum_{k=0}^{j} L_{k,n_1} \cdot S_{j-k,n_2}. \end{aligned}$$

Since $V(G_1) \cap V(G_2) = \emptyset$,

$$\begin{aligned}
b_{j,n_1+n_2} &= S_{j,n_1 \oplus n_2} - L_{j,n_1 \oplus n_2} \\
&= \sum_{k=0}^{j} (S_{k,n_1} \cdot S_{j-k,n_2} + L_{k,n_1} \cdot L_{j-k,n_2} - (S_{k,n_1} \cdot L_{j-k,n_2} + L_{k,n_1} \cdot S_{j-k,n_2})) \\
&= \sum_{k=0}^{j} S_{k,n_1}(S_{j-k,n_2} - L_{j-k,n_2}) - L_{k,n_1}(S_{j-k,n_2} - L_{j-k,n_2}) \\
&= \sum_{k=0}^{j} (S_{k,n_1} - L_{k,n_1})(S_{j-k,n_2} - L_{j-k,n_2}) \\
&= \sum_{k=0}^{j} b_{k,n_1} \cdot b_{j-k,n_2}.
\end{aligned}$$

3.3 Bounds on the Composition Number of General Graphs

Lemma 1. *If G is a graph such that $|E(G)| = k$, then $C(G) \leq 2^k$.*

Proof: Every composition of G yields a unique subgraph of G (however, it is possible for 2 subgraphs of G to determine the same composition of G). Hence we can define an injective function $f : \mathcal{C}(G) \rightarrow \mathcal{P}(G)$. Therefore, $|\mathcal{P}(G)| = 2^k$ implies $C(G) = |\mathcal{C}(G)| \leq 2^k$.

Theorem 20. *If $N \in \mathbb{Z}^+$ and $k \leq N - 1$ and G is a graph such that $|E(G)| = k$, then $C(K_N^{-S_{k+1}}) \leq C(K_N^{-G})$.*

Proof: Let $b_{j,n}$ represent the j^{th} Bell number coefficient of K_N^{-G} and $s_{j,n}$ represent the j^{th} Bell number coefficient of $K_N^{-S_{k+1}}$. Remark 2 yields $s_{0,n} = b_{0,n} = 1$ and $s_{1,n} = b_{1,n} = 0$ and Remark 6 yields the result $s_{j,n} = -\binom{k}{j-1}$ for

j and $n \geq 2$ and $j \leq n$. Theorem 18 yields $s_{j,n} \leq b_{j,n}$ for $0 \leq j \leq n$. Hence, Theorem 20 is true.

Theorem 21. *If G is a subgraph of K_N such that $|V(G)| \leq N-2$, then $C(K_N^{-(G \cup \{e\})}) \leq C(K_N^{-(G \cup \{e'\})})$ for any e' and $e \notin E(K_N^{-G})$, where e' is not incident to any vertex of $V(G)$.*

Proof: Let e and e' be of the form outlined above. Let $C_1 = \mathcal{C}(K_N) \backslash \mathcal{C}\left(K_N^{-(G \cup \{e'\})}\right)$, $C_2 \subseteq \mathcal{C}(K_N) \backslash \mathcal{C}\left(K_N^{-(G \cup \{e\})}\right)$ where $C \in C_2$ if and only if $G^c[C]$ contains e as a single edge component, and $C_3 = \mathcal{C}(K_N) \backslash C_2$. Then $C\left(K_N^{-(G \cup \{e'\})}\right) = B(N) - |C_1|$ and $C\left(K_N^{-(G \cup \{e\})}\right) = B(N) - (|C_2| + |C_3|)$. It is also trivial to notice that $|C_1| = |C_2|$. Also, if e is not incident to some vertex in $V(G)$, then $|C_3| = 0$; otherwise, $|C_3| > 0$. Hence, $C\left(K_N^{-(G \cup \{e\})}\right) \leq C\left(K_N^{-(G \cup \{e'\})}\right)$.

Theorem 22. *If G is a graph such that $|E(G)| = k$ and $2k \leq N \in \mathbb{Z}^+$, then $C(K_N^{-G}) \leq C(K_N^{-D_k})$.*

Proof: Let C_1 be the set of compositions of K_N which are not compositions of $K_N^{-D_k}$, C_2 be the set of compositions of K_N which are not compositions of K_N^{-G} and whose complements are comprised solely of single edge components of G, and C_3 be the set of compositions of K_N which are not compositions of K_N^{-G} that are not present in C_2. $C(K_N^{-D_k}) = B(N) - |C_1|$ and $C(K_N^{-G}) = B(N) - (|C_2| + |C_3|)$. It is trivial to notice that $|C_1| = |C_2|$ and $|C_3| \geq 0$. Hence, $C(K_N^{-G}) \leq C(K_N^{-D_k})$.

Corollary 1. *If G is a graph such that $E(G) = k$ and $2k \leq N \in \mathbb{Z}^+$, then $C(K_N^{-S_{k+1}}) \leq C(K_N^{-G}) \leq C(K_N^{-D_k})$.*

Proof: The observation of $k+1 \leq 2k$ for all $k \in \mathbb{Z}^+$, where $1 \leq k$ allows us to apply Theorems 22 and Theorem 20 simultaneously to $C(K_N^{-G})$. Hence, Corollary 1 is verified.

Chapter 4

Connection to Aitken's Array

4.1 Introduction

Aitken's array is an array with a long history, appearing as early as 1880 in [6]. The entry in the n^{th} row and k^{th} column is denoted by $A(n,k)$, and is defined as the number of partitions of the set $\{1, 2, ..., n+1\}$ which contain $\{k+1\}$ as the largest singleton component. A portion of the array appears below.

1						
1	2					
2	3	5				
5	7	10	15			
15	20	27	37	52		
52	67	87	114	151	203	
203	255	322	409	523	674	877

Table 4.1: Portion of Aitken's array

Using the interpretation, it is trivial to notice that $A(n,n) = B(n)$. Additionally, it can be shown that $A(n,n) = A(n+1,1)$. The theorems that follow will use the interpretation of Aitken's array provided by [8] to find 2 recurrences of Aitken's array in terms of the Bell numbers. We then use Theorem 10 to tie Aitken's array to graph compositions. Theorem 24 and Corollary 3 were originally found independently from the results of Sun and Wu with Corollary 3 illustrating a very detailed connection between graph compositions and Aitken's array; access to the work of Sun and Wu, however, allowed the connection to be shown in a more efficient way.

4.2 Previous Work

Previous work in Aitken's array was performed by Sun and Wu in [8]. The authors denoted $A(n,k)$ by $A_{n,k}$ and $B(n)$ by B_n. The results from [8] listed below are used to establish other connections between Aitken's array and the Bell number sequence.

Theorem 23. *(Sun and Wu, [8]) For any integers $n, m,$ and $k \geq 0$,*

1. $A_{n+m,m} = \sum_{j=0}^{n} (-1)^{n-j} \binom{n}{j} B_{m+j}$.

2. $A_{n+m+k,m+k} = \sum_{j=0}^{m} \binom{m}{j} A_{n+k+j,k}$.

Corollary 2. *(Sun and Wu, [8]) For any integers n and $m \geq 0$,*

$$A_{n+m+1,m+1} = \sum_{j=0}^{m} \binom{m}{j} B_{n+j}.$$

The results that follow were proved using the survey of results above.

Theorem 24. *For n and $k \in \mathbb{Z}^+$ and $k \leq n$, $A(n,k) = \sum_{j=0}^{n-k} (-1)^j \binom{n-k}{j} B(n-j)$.*

Proof: By setting $m = k$ and $n = n - m$ in Theorem 23.1, then

$$A(n,k) = \sum_{j=0}^{n-k} (-1)^{n-k-j} \binom{n-k}{j} B(k+j).$$

By setting $j = n - k - j$, we get

$$A(n,k) = \sum_{j=0}^{n-k} (-1)^{n-k-(n-k-j)} \binom{n-k}{n-k-j} B(k+n-k-j)$$

$$= \sum_{j=0}^{n-k} (-1)^{j} \binom{n-k}{j} B(n-j).$$

Corollary 3. $A(n,k) = \sum\limits_{j=1}^{k} \binom{k-1}{j-1} B(n-j)$.

Proof: If we set $k = m+1$ in Corollary 2, then we get

$$A(n+k,k) = \sum_{j=0}^{k-1} \binom{k-1}{j} B(n+j)$$

$$= \sum_{j=0}^{k-1} \binom{k-1}{j} B(n+(k-1)-j)$$

$$= \sum_{j=1}^{k} \binom{k-1}{j-1} B(n+k-j).$$

Setting $n = n - k$ in the above yields
$A(n,k) = \sum\limits_{j=1}^{k} \binom{k-1}{j-1} B(n-j)$.

4.3 Connection to Graph Compositions

Corollary 4. $C(K_n^{-S_k}) - C(K_n^{-S_{k+1}}) = A(n,k)$.

Proof: Application of Theorem 10 yields

$$C(K_{n+1}^{-S_k}) = B(n+1) - \sum_{j=1}^{k-1} \binom{k-1}{j} B(n-j)$$

$$C(K_{n+1}^{-S_{k+1}}) = B(n+1) - \sum_{j=1}^{k} \binom{k}{j} B(n-j).$$

Hence

$$C(K_{n+1}^{-S_k}) - C(K_{n+1}^{-S_{k+1}}) = \sum_{j=1}^{k} \binom{k}{j} B(n-j) - \sum_{j=1}^{k-1} \binom{k-1}{j} B(n-j)$$

$$= \sum_{j=1}^{k-1} \left[\left(\binom{k}{j} - \binom{k-1}{j} \right) B(n-j) \right] + B(n-k)$$

$$= \sum_{j=1}^{k-1} \left[\binom{k-1}{j-1} B(n-j) \right] + B(n-k)$$

$$= \sum_{j=1}^{k} \binom{k-1}{j-1} B(n-j).$$

Therefore, by Corollary 3, $C(K_n^{-S_k}) - C(K_n^{-S_{k+1}}) = A(n,k)$.

Remark 9. *The proof of Corollary 4 establishes the connection between Aitken's array and graph compositions.* $K_{n+1}^{-S_k} \backslash K_{n+1}^{-S_{k+1}}$ *is isomorphic to a graph* K_n *and some isolated vertex* v *where exactly* $n-k+1$ *edges span* $V(K_n)$ *and* $\{v\}$*.* $A(n,k)$ *counts the number of compositions of this graph that exist as a result of the addition of the* $(n-k+1)^{st}$ *edge between* $V(K_n)$ *and* $\{v\}$*.*

Chapter 5

Compositions of Graphs Limiting the Number of Components

5.1 Introduction

The previous chapters of this dissertation have dealt with finding the composition number of general graphs. The introduction of Chapter 1 discusses the connection between graph compositions and partitions of positive integers. In a traditional number theory setting, it is common to discuss general partitions of positive integers, and then move on to partitions which have extra criteria added (e.g. compositions of positive integers with exactly k components). This frame of mind is the motivation for Chapter 5, which takes results from [4] and adds the extra criterion that each composition counted in the composition number must have exactly k components. Not surprisingly, Stirling numbers of the second kind play the role that the Bell numbers do for compositions of a graph without restrictions on the number of components. The results of Chapter 5 are interesting as they help us to more clearly understand the results of Knopfmacher and Mays [4]. The results for the composition number of 2 graphs which are disjoint or share exactly one vertex are of particular interest since the results are identical in [4], but differ slightly in this chapter, which was an unexpected surprise.

Recall that Stirling numbers of the second kind is the number of partitions of a set S such that $|S| = n$, where every element of the partition has exactly k non-empty subsets of S. This means that $S(n, k)$ also counts the number of partitions of the positive integer n which contain exactly k positive summands. A portion of the array appears below.

1						
1	1					
1	3	1				
1	7	6	1			
1	15	25	10	1		
1	31	90	65	15	1	
1	63	301	350	140	21	1

Table 5.1: Portion of Stirling Numbers of the 2^{nd} Kind

For a graph G, $C^k(G)$ will be used to denote the number of compositions of G with exactly k components where $|V(G)| = n$. For the theorems that follow, unless otherwise specified, it will be understood that $0 < k \leq n$.

Lemma 2. *$C(G) = \sum_{k=1}^{n} C^k(G)$ for any graph G such that $|V(G)| = n$.*

Remark 10. *The above lemma may seem trivial, but it is none-the-less useful for verifying $C^k(G)$ once it is found. It should also be noted that $C^1(G) = 1$ and $C^n(G) = 1$. Furthermore, $C^k(G) = 0$ for $k > n$.*

5.2 Common Graphs

Theorem 25. *If T_n is a tree with n vertices, then $C^k(T_n) = \binom{n-1}{k-1}$.*

Proof: It is well known that T_n has exactly $n-1$ edges. Furthermore, it is trivial to notice that the deletion of a single edge from any subgraph of T_n will result in a distinct subgraph which has exactly 1 more component than the original subgraph. Since T_n has 1 component to begin with, we must delete $k-1$ edges to produce a subgraph of T_n which will contain exactly k components. Hence $C^k(T_n) = \binom{n-1}{k-1}$.

Lemma 3. *$C^k(K_n) = S(n,k)$.*

Remark 11. *Theorem 25 and Lemma 3 are extreme cases for any connected graph G such that $|V(G)| = n$. Hence, it is easily observed that $\binom{n-1}{k-1} \leq C^k(G) \leq S(n,k)$ for any connected G such that $|V(G)| = n$.*

Theorem 26. *If $G = G_1 \cup G_2$ where $V(G_1) \cap V(G_2) = \emptyset$ and $E(G_1) \cap E(G_2) = \emptyset$, then $C^k(G) = \sum_{j=1}^{k-1} \left[C^j(G_1) \cdot C^{k-j}(G_2)\right]$.*

Proof: Let $k \geq j \in \mathbb{Z}^+$ and C be a composition of G_1 which has j components. Then there are exactly $C^{k-j}(G_2)$ components of G which contain C and have exactly k components. Hence, $C^k(G) = \sum_{j=1}^{k-1} \left[C^j(G_1) \cdot C^{k-j}(G_2)\right]$.

Corollary 5. *If $G = G_1 \cup G_2$ where $|V(G_1) \cap V(G_2)| = 1$ and $E(G_1) \cap E(G_2) = \emptyset$, then $C^k(G) = \sum_{j=1}^{k} \left[C^j(G_1) \cdot C^{k+1-j}(G_2)\right]$.*

Proof: Let v be the vertex which $V(G_1)$ and $V(G_2)$ share and $C \in \mathcal{C}^k(G)$. If we separate C into 2 disjoint graphs by disconnecting v from all vertices in $V(G_2)$ and adding a new vertex which is adjacent to all of the original neighbors of v in $V(G_2)$, then the result is a composition of $k+1$ components. Hence by Theorem 26, $C^k(G) = \sum_{j=1}^{k} \left[C^j(G_1) \cdot C^{k+1-j}(G_2)\right]$.

Theorem 27. *If $G = G_1 \cup G_2$ where $V(G_1) \cap V(G_2) = \emptyset$ and G has exactly one edge incident to vertices from G_1 and G_2, then $C^k(G) = \sum_{j=1}^{k-1} C^j(G_1) \cdot \left[C^{k+1-j}(G_2) + C^{k-j}(G_2)\right] + C^k(G_1)$.*

Proof: Let e be the edge incident to vertices in G_1 and G_2 and $C \in \mathcal{C}^k(G)$. If the vertices of e in C are connected, then $C^{-\{e\}}$ will have exactly $k+1$ components. Hence, there are exactly $C^{k+1}(G^{-\{e\}})$ compositions of this form. Also, there are $C^k(G^{-\{e\}})$ compositions of $G^{-\{e\}}$ in which the vertices of e are not connected. Theorem 26 yields the results

$$C^{k+1}(G^{-\{e\}}) = \sum_{j=1}^{k} \left[C^j(G_1) \cdot C^{k+1-j}(G_2)\right]$$

$$C^{k}(G^{-\{e\}}) = \sum_{j=1}^{k-1} \left[C^j(G_1) \cdot C^{k-j}(G_2)\right].$$

$$\begin{aligned} \text{Hence, } C^k(G) &= C^{k+1}(G^{-\{e\}}) + C^k(G^{-\{e\}}) \\ &= \sum_{j=1}^{k} \left[C^j(G_1) \cdot C^{k+1-j}(G_2)\right] + \sum_{j=1}^{k-1} \left[C^j(G_1) \cdot C^{k-j}(G_2)\right] \\ &= \sum_{j=1}^{k-1} C^j(G_1) \cdot \left[C^{k+1-j}(G_2) + C^{k-j}(G_2)\right] + C^k(G_1) \cdot C^1(G_2) \\ &= \sum_{j=1}^{k-1} C^j(G_1) \cdot \left[C^{k+1-j}(G_2) + C^{k-j}(G_2)\right] + C^k(G_1). \end{aligned}$$

Theorem 28. *$C^k(K_n^-) = S(n,k) - S(n-2,k-1)$.*

Proof: If C is a composition of K_n that has exactly k components, then C will not be a composition of $K_n^{-\{e\}}$ if and only if e is a singleton component of C. There are exactly $S(n-2,k-1)$ of these types of compositions. Hence, Theorem 28 is true.

Theorem 29. $C^k(C_n) = \begin{cases} 1, & k=1 \\ \binom{n}{k}, & 1 < k \le n \end{cases}$

Proof: $k=1$ is the trivial case of our theorem and is discussed in Remark 10. Assume that $k>1$. Deleting a single edge from C_n yields P_n. Theorem 25 yields $C^k(P_n) = \binom{n-1}{k-1}$. Hence, $C^k(C_n) = \binom{n}{k}$.

5.3 Deletions of General Graphs from Complete Graphs

Theorem 30. *Let G be a graph with $n \le N$ vertices. If $b_{j,m,n}$ represents the number of ways of choosing m disjoint bad components of K_N^{-G} such that the cardinality of the union of vertices of all* bad *components is j, then $C^k(K_N^{-G}) = \sum_{j=0}^{n}\sum_{m=0}^{j}(-1)^m b_{j,m,n} \cdot S(N-j,k-m)$.*

Proof: We begin by denoting the set of all compositions of K_N with exactly k components by Γ. If $B(\gamma)$ denotes the number of disjoint bad components contained in γ for all $\gamma \in \Gamma$, then

$$C^k(K_N^{-G}) = \sum_{\substack{\gamma\in\Gamma \\ B(\gamma)=0}} 1 = \sum_{\substack{\gamma\in\Gamma \\ B(\gamma)=0}} 1 + \sum_{\substack{\gamma\in\Gamma \\ B(\gamma)\neq 0}} 0 = \sum_{\gamma\in\Gamma}\sum_{m=0}^{B(\gamma)}(-1)^m\binom{B(\gamma)}{m}$$

since the alternating sum of the m^{th} row of Pascal's Triangle is 0 for $m > 0$ and 1 for $m = 0$. Note that $\binom{B(\gamma)}{m}$ will count the number of times γ is counted as a composition of K_N with at least m disjoint *bad* components for a fixed γ and m.

$$\sum_{\gamma \in \Gamma} \sum_{m=0}^{B(\gamma)} \binom{B(\gamma)}{m} = \sum_{j=0}^{n} \sum_{m=0}^{n} b_{j,m,n} S(N-j, k-m).$$

So

$$\begin{aligned} C^k(K_N^{-G}) &= \sum_{\gamma \in \Gamma} \sum_{m=0}^{B(\gamma)} (-1)^m \binom{B(\gamma)}{m} \\ &= \sum_{j=0}^{n} \sum_{m=0}^{n} (-1)^m b_{j,m,n} S(N-j, k-m). \end{aligned}$$

Remark 12. *It is important for later theorems to note that*

- $b_{j,m,n} = 0$ *if* $j > n$, $m > j$, *or* $m > n$.
- $b_{0,0,n} = 1$.
- $b_{1,m,n} = 0$.

5.4 Deletions of Specific Graphs From Complete Graphs

Theorem 31. *Let n and N be positive integers with $n \leq N$. If $p_{j,m,n}$ represents the number of ways of choosing m disjoint bad components of $K_N^{-P_n}$ such that the cardinality of the union of vertices of all components* *is j, then* $C^k(K_N^{-P_n}) = \sum_{j=0}^{n}\sum_{m=0}^{j}(-1)^m p_{j,m,n} \cdot S(N-j, k-m)$ *and* $p_{j,m,n} = p_{j,m,n-1} + p_{j-2,m-1,n-2} + p_{j-3,m-1,n-3}$.

Proof: $C^k(K_N^{-P_n}) = \sum_{j=0}^{n}\sum_{m=0}^{j}(-1)^m p_{j,m,n} \cdot S(N-j, k-m)$ is a result of application of Theorem 30 to $K_N^{-P_n}$. If $\mathcal{C}$ is a composition of K_N, then $\mathcal{C}$ will **not** be a composition of $K_N^{-P_n}$ if and only if $\mathcal{C}$ contains a bad component of $K_N^{-P_n}$. Theorem 6 yields that all *bad* components of $K_N^{-P_n}$ will be either single edge components or subpaths of P_n of length 2.

If u represents one of the terminal vertices of P_n and C is a component which contains u, then one of three cases must occur:

1. C is not a bad component. There are $p_{j,m,n-1}$ ways of choosing m disjoint bad components from $V(P_n)$ such that the cardinality of the union of vertices of all components which include C is j.

2. C is a single edge bad component. There are $p_{j-2,m-1,n-2}$ ways of choosing m disjoint bad components from $V(P_n)$ such that the cardinality of union of vertices of components which include C is j.

3. C is a 3-element bad component. There are $p_{j-3,m-1,n-3}$ ways of choosing m disjoint bad components from $V(P_n)$ such that the cardinality of union of vertices of components which include C is j.

The above "world encompassing" cases yield the result

$$p_{j,m,n} = p_{j,m,n-1} + p_{j-2,m-1,n-2} + p_{j-3,m-1,n-3}.$$

Theorem 32. *If* $F^k(x,y,z) = \sum_{j=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} p_{j,m,n}x^ny^mz^j$*, then* $F^k(x,y,z) = \frac{1}{1-x-x^2yz^2-x^3yz^3}$.

Proof: As stated in Theorem 31, $p_{j,m,n}$ is the number of ways of choosing m disjoint bad components of $K_N^{-P_n}$ where the cardinality of bad vertices is j. Every vertex in P_n is either:

- A *good* vertex - represented by x.
- Contained in a 2 element bad component- represented by x^2yz^2.
- Contained in a 3 element (non-cyclic) bad component- represented by x^3yz^3.

$(x + x^2yz^2 + x^3yz^3)^m$ represents the number of ways one can choose at most m disjoint bad components of $K_N^{-P_n}$. $F^k(x,y,z)$ contains coefficients for all m, so

$$\begin{aligned} F^k(x,y,z) &= \sum_{m=0}^{\infty}(x + x^2yz^2 + x^3yz^3)^m \\ &= \frac{1}{1-(x + x^2yz^2 + x^3yz^3)} \\ &= \frac{1}{1-x-x^2yz^2-x^3yz^3}. \end{aligned}$$

Theorem 33. *Let n and N be positive integers with $n \leq N$. If $p_{j,m,n}$ is defined as in Theorem 31 and $c_{j,m,n}$ represents the number of ways of choosing m disjoint bad components of $K_N^{-C_n}$ where the number of vertices of the bad components is j, then $C^k(K_N^{-C_n}) = \sum_{j=0}^{n}\sum_{m=0}^{j}(-1)^m c_{j,m,n} \cdot S(N-j, k-m)$ and $c_{j,m,n} = p_{j,m,n-1} + 2 \cdot p_{j-2,m-1,n-2} + 3 \cdot p_{j-3,m-1,n-3}$ for j and $n \geq 3$, and $n \neq j$ for $n \in \{3,4\}$.*

Proof: $C^k(K_N^{-C_n}) = \sum_{j=0}^{n}\sum_{m=0}^{j}(-1)^m c_{j,m,n} \cdot S(N-j, k-m)$ follows from the application of Theorem 30 to $K_N^{-C_n}$. If $\mathcal{C}$ is a composition of K_N, then $\mathcal{C}$ will **not** be a composition of $K_N^{-C_n}$ if and only if $\mathcal{C}$ contains a bad component of $K_N^{-C_n}$.

We next describe the bad components of $K_N^{-C_n}$. Theorem 8 tells us that bad components of $K_N^{-C_n}$ are limited to subpaths of length 1 and 2 of C_n and 3 and 4 element cyclic components of C_n.

If $w \in V(C_n)$ and $C_w \in \mathcal{C}$ contains w, then one of four cases occurs:

1. C_w is not a bad component. There are $p_{j,m,n-1}$ ways of choosing m disjoint bad components from $V(C_n)$ (with cardinality of the union of vertices j) which do not include C_w.

2. C_w is a 2-element bad component. There are $2{\cdot}p_{j-2,m-1,n-2}$ ways of choosing m disjoint bad components from $V(C_n)$ (with cardinality of the union of vertices j) which include C_w.

3. C_w is a 3-element bad component. If C_w is a path ($n > 3$), then there are $3{\cdot}p_{j-3,m-1,n-3}$ ways of choosing m disjoint bad components from $V(C_n)$ (with cardinality of the union of vertices j) which include C_w. If C_w is a cycle ($n = 3$), then there is exactly one way of choosing 1 bad component.

4. C_w is a 4-element bad component. There is exactly one way of choosing 1 bad component.

The above "world encompassing" cases give us

$$c_{j,m,n} = p_{j,m,n-1} + 2 \cdot p_{j-2,m-1,n-2} + 3 \cdot p_{j-3,m-1,n-3}.$$

Theorem 34. *If* $G^k(x,y,z) = \sum_{j=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} c_{j,m,n}x^n y^m z^j$, *then* $G^k(x,y,z) = 1 - 2x^3yz^3 + x^4yz^4 + \frac{x+2x^2yz^2+3x^3yz^3}{1-x-x^2yz^2-x^3yz^3}$.

Proof: The proof of Theorem 34 is similar to that of Theorem 32. The formula must be adjusted since there are 2 ways of choosing a bad edge for every vertex in $V(C_n)$ and 3 ways of choosing a bad subpath of length 2 for every vertex in $V(C_n)$. The coefficients for x^3yz^3 and x^4yz^4 do not follow from the recurrence listed in Theorem 32, prompting the need for *error* terms which will correct the coefficients of the preceding terms.

Theorem 35. *If* $n+1 \leq N$, *then* $C^k(K_N^{-S_{n+1}}) = S(N,k) - \sum_{j=1}^{n}\binom{n}{j}S(N-(j+1),k-1)$.

Proof: Let $s_{j,m,n}$ represent the number of ways of choosing exactly m disjoint components from S_n such that the cardinality of the set of vertices of the components chosen is j. As stated in Theorem 10, all bad components of $K_N^{-S_{n+1}}$ will be $S_{j+1} \subseteq S_{n+1}$ for $1 \leq j \leq n$. This further implies $m \in \{0,1\}$. It is easily verified that

$$s_{j,1,n} = \binom{n-1}{j-1}.$$

Hence, by Theorem 30, $C^k(K_N^{-S_{n+1}}) = S(N,k) - \sum_{j=1}^{n}\binom{n}{j} \cdot S(N-(j+1),k-1)$.

Corollary 6. *If* $S^k(x,y) = \sum_{j=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} s_{j,m,n}x^n y^j$, *then* $S^k(x,y) = \frac{1}{1-x+xy}$.

Proof: The coefficients of formulae in Theorems 10 and 35 match. This means that the generating functions for both sets of coefficients must also match. Hence, the result is true by Theorem 11.

Theorem 36. *If $2n \leq N$, then $C^k(K_N^{-D_n}) = S(N,k) - \sum_{j=1}^{n}(-1)^j\binom{n}{j}S(N - 2j, k-j)$.*

Proof: Let $d_{j,m,2n}$ represent the number of ways of choosing exactly m disjoint components from D_n such that the cardinality of the set of vertices of the components chosen is j. Note that $d_{j,m,2n} = 0$ if j is odd or $j \neq 2m$. Also $d_{2j,j,2n} = \binom{n}{j}$. Hence,

$$\begin{aligned}
C(K_N^{-D_n}) &= \sum_{j=0}^{2n}\sum_{m=0}^{j}(-1)^m d_{j,m,2n}S(N-j,k-m) \\
&= \sum_{j=0}^{2n}\sum_{m=\frac{j}{2}}(-1)^m d_{j,m,2n}S(N-j,k-m) \\
&= \sum_{j=0}^{n}(-1)^j d_{2j,j,2n}S(N-2j,k-j) \\
&= \sum_{j=0}^{n}(-1)^j\binom{n}{j}S(N-2j,k-j).
\end{aligned}$$

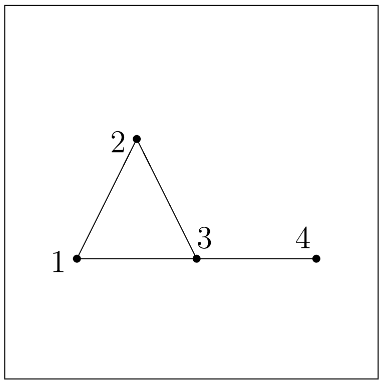
Corollary 7. *If $D^k(x,y) = \sum_{j=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} d_{j,m,n}x^n y^j$, then $D^k(x,y) = \frac{1}{1-x+xy}$.*

Proof: The coefficients of formulae in Theorems 12 and 36 match. This means that the generating functions for both sets of coefficients must also match. Hence, the result is true by Theorem 13.

Appendix A

CompositionCounter

The following is the source code for a program affectionately dubbed "CompositionCounter". During the early stages of research, data on the composition numbers of certain graphs was needed in order to gain some intuition on the subject. Analysis of this data led to the main results of Chapter 2. The program was written using Java. It uses a GUI to receive the edges of a graph (using 2 positive integers separated by a comma as input) and the number of components that each composition is to have (also using a positive integer). The output of the program is the composition number of the specified graph with the specified number of components. If the component number field is left blank, then the composition number is displayed by default. In the interest of saving time and memory, the number of compositions where the number of components is the same as the number of vertices of the graph is never considered in the count. There is always one such composition when this event occurs, so there is no need to compute it. If the entire composition number of a graph is sought, then 1 is added to the count to compute the composition number. An unfortunate side effect of the utilization of this method is the program yields an incorrect answer of 0 if the number of vertices of the specified graph is listed as the component number. An example graph along with the input and output for the graph is given below.



input:

edge: 1,2
edge: 2,3
edge: 1,3
edge: 3,4
number of components: ⟨blank⟩

output: 10

CompositionCounter.java

```
import java.awt.*;
import java.util.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;


public class CompositionCounter extends JFrame
implements ActionListener, DocumentListener,
ListSelectionListener
{
JLabel inputLabel;
JLabel compLabel;
JLabel outputLabel;

JTextField inputTF;
JTextField compTF;
JTextField outputTF;

JPanel ioPanel;
JPanel layoutPanel;
JPanel inputButtonPanel;
JPanel componentsPanel;
JPanel inputPanel;
JPanel inputEdgePanel;
JPanel inputCompPanel;
JPanel outputPanel;

DefaultListModel inputList;
JList inputListBox;
JScrollPane inputScrollPane;

JButton genButton;
JButton inputButton;
JButton removeButton;
JButton clearButton;

EdgeSet edgeSet;
PowerSet powerSet;
CompositionSet compSet;


public static void main(String[] args)
{
new CompositionCounter();
}

CompositionCounter()
{
inputLabel           = new JLabel("Edge");
compLabel            = new JLabel("Number of Components: ");
outputLabel          = new JLabel("Number of Compositions: ");

inputTF              = new JTextField(10);
compTF               = new JTextField(5);
outputTF             = new JTextField(5);

inputTF.getDocument().addDocumentListener(this);
compTF.getDocument().addDocumentListener(this);

outputTF.setEditable(false);

ioPanel              = new JPanel(new GridLayout(2,1));
layoutPanel          = new JPanel(new BorderLayout());
inputButtonPanel     = new JPanel(new FlowLayout());
```

```
componentsPanel      = new JPanel(new GridLayout(2,1));
inputPanel           = new JPanel(new GridLayout(2,1));
inputEdgePanel       = new JPanel(new FlowLayout());
inputCompPanel       = new JPanel(new FlowLayout());
outputPanel          = new JPanel(new FlowLayout());

inputList            = new DefaultListModel();
inputListBox         = new JList(inputList);

genButton            = new JButton("Count Compositions");
inputButton          = new JButton("Enter Edge");
removeButton         = new JButton("Remove Edge");
clearButton          = new JButton("Clear List");

edgeSet = new EdgeSet();
compSet = new CompositionSet();
inputButton.addActionListener(this);
inputButton.setActionCommand("INPUT");

genButton.addActionListener(this);
genButton.setActionCommand("GENERATE");

removeButton.addActionListener(this);
removeButton.setActionCommand("REMOVE");

clearButton.addActionListener(this);
clearButton.setActionCommand("CLEAR");

inputButton.setEnabled(false);
genButton.setEnabled(false);
removeButton.setEnabled(false);

inputListBox.addListSelectionListener(this);
inputScrollPane = new JScrollPane(inputListBox);

setupComponents();
setupCompositionCounter();
}

public void setupCompositionCounter()
{
Toolkit tk;
Dimension d;

tk = Toolkit.getDefaultToolkit();

d = tk.getScreenSize();

setLocation(d.width/4, d.height/4);
setSize(600, 700);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setTitle("Composition Counter");
```

```
setVisible(true);
}

public void setupComponents()
{
inputEdgePanel.add(inputLabel);
inputEdgePanel.add(inputTF);

inputCompPanel.add(compLabel);
inputCompPanel.add(compTF);

inputPanel.add(inputEdgePanel);
inputPanel.add(inputCompPanel);

outputPanel.add(outputLabel);
outputPanel.add(outputTF);

inputButtonPanel.add(inputButton);
inputButtonPanel.add(removeButton);
inputButtonPanel.add(genButton);
inputButtonPanel.add(clearButton);

componentsPanel.add(inputPanel);
componentsPanel.add(inputButtonPanel);

ioPanel.add(componentsPanel);
ioPanel.add(outputPanel);

layoutPanel.add(ioPanel, BorderLayout.CENTER);
layoutPanel.add(inputScrollPane, BorderLayout.EAST);

getRootPane().setDefaultButton(inputButton);
getContentPane().add(layoutPanel);

inputTF.requestFocus();
}


public void valueChanged(ListSelectionEvent e)
{
if (!inputListBox.isSelectionEmpty())
    {
    genButton.setEnabled(true);
    removeButton.setEnabled(true);
    }

else
    {
    genButton.setEnabled(false);
    removeButton.setEnabled(false);
    }
}
```

```
public void actionPerformed(ActionEvent e)
{
if (e.getActionCommand().equals("INPUT"))
    {
    StringTokenizer strTok;
    String input;

    int x;
    int y;

    input = inputTF.getText().trim();
    strTok = new StringTokenizer(input, ",");

    x = Integer.parseInt(strTok.nextToken().trim());
    y = Integer.parseInt(strTok.nextToken().trim());

    if (x > y)
        {
        x += y;
        y = x-y;
        x = x-y;
        }

    if (!inputList.contains(x + "," + y))
        inputList.addElement(x + "," + y);



    inputButton.setEnabled(false);
    inputTF.setText("");
    inputTF.requestFocus();

    inputListEmpty();
    outputTF.setText("");
    }

else if (e.getActionCommand().equals("GENERATE"))
    {
    int k;
    Edge e0;
    String input;
    int numCompositions;
    int numComponents;

    compSet.clear();
    compSet.setMaxVertex(0);

    for(k=0; k < inputList.size(); k++)
        {
        e0 = new Edge((String)(inputList.getElementAt(k)));
        edgeSet.add(e0);

        if (compSet.getMaxVertex() <= e0.e1)
```

```
                    compSet.setMaxVertex(e0.e1);

              if (compSet.getMaxVertex() <= e0.e2)
                    compSet.setMaxVertex(e0.e2);
              }

           edgeSet.setCardinality(inputList.size());
           powerSet = new PowerSet(edgeSet);

           compSet.getPowerSet(powerSet);
           compSet.buildCompositions();

           input = compTF.getText().trim();

           numCompositions = 0;


              if (input.isEmpty())
                    numCompositions = compSet.getCardinality() + 1;

              else
              {
              numComponents  = Integer.parseInt(input);

              for (k=0; k < compSet.getCardinality(); k++)
                    {
                    if (compSet.get(k).getCardinality() == numComponents)
                          numCompositions++;
                    }
              }

           outputTF.setText(new Integer(numCompositions).toString());

           for (k=0; k < compSet.getCardinality(); k++)
                 {
                 compSet.get(k).print();
                 System.out.println("\n");
                 }

           edgeSet.clear();
           genButton.setEnabled(false);
           }

        else if (e.getActionCommand().equals("REMOVE"))
           {
           int k;

           k = inputListBox.getSelectedIndex();

           while (k != -1)
              {
              inputList.remove(k);
              k = inputListBox.getSelectedIndex();
              }
```

```
    removeButton.setEnabled(false);
    inputTF.requestFocus();

    inputListEmpty();
    outputTF.setText("");
    }

else if (e.getActionCommand().equals("CLEAR"))
    {
    inputList.clear();
    outputTF.setText("");
    }
}


public void insertUpdate(DocumentEvent e)
{
inputButton.setEnabled((validateInput()));
genButton.setEnabled(validateCompTF());
}


public void removeUpdate(DocumentEvent e)
{
inputButton.setEnabled((validateInput()));
genButton.setEnabled(validateCompTF());
}


public void changedUpdate(DocumentEvent e)
{

}

public void inputListEmpty()
{
genButton.setEnabled(!inputList.isEmpty());
}


public boolean validateCompTF()
{
String input;
boolean myReturn;
int numInput;

input = compTF.getText().trim();

myReturn = false;
if (input.isEmpty())
        myReturn = true;

else
```

```
    {
    try
        {
        numInput = Integer.parseInt(input);

        if (numInput > 0)
                myReturn  = true;
        }

    catch(Exception e)
        {

        }
    }

return myReturn;
}

public boolean validateInput()
{
boolean myReturn;
String input;
StringTokenizer strTok;

int x;
int y;

myReturn = false;

input = inputTF.getText().trim();

strTok = new StringTokenizer(input, ",");

if (strTok.countTokens() == 2)
    {
    try
        {
        x = Integer.parseInt(strTok.nextToken().trim());
        y = Integer.parseInt(strTok.nextToken().trim());

        if (x > 0 && y > 0)
        myReturn = true;
        }

    catch (Exception e)
        {

        }
    }

return myReturn;
}
}

class Edge extends Object
```

```
{
int e1;
int e2;

Edge(String s)
{
StringTokenizer str;

str = new StringTokenizer(s, ",");

e1 = Integer.valueOf(str.nextToken().trim()).intValue();
e2 = Integer.valueOf(str.nextToken().trim()).intValue();
}

public void print()
{
System.out.print("{" + e1 + ", " + e2 + "}");
}
}

class EdgeSet extends Vector<Edge>
{
private int cardinality;

EdgeSet()
{
cardinality = 0;
}
public void print()
{
int k;
System.out.print("{");

for(k=0; k < size(); k++)
{
super.get(k).print();

if (k < size() - 1)
    System.out.print(", ");
}

System.out.print("}");
}

public void setCardinality(int x)
{
cardinality = x;
}

public int getCardinality()
{
return cardinality;
}
}
```

```
class PowerSet extends Vector<EdgeSet>
{
private int cardinality;

PowerSet(EdgeSet e)
{
EdgeSet e2;

int exp;
int exp2;

int k;
int k2;

int j;

boolean positionFound;

exp = e.getCardinality();

cardinality = 1;

for (k=0; k < exp; k++)
    cardinality *= 2;

cardinality--;

for (k=0; k < cardinality; k++)
{
e2 = new EdgeSet();
k2 = k+1;
exp2 = exp;

while ((int)Math.pow(2, exp2) > k2)
exp2--;

while (k2 > 0)
{
if ((int)Math.pow(2,exp2) <= k2)
{
if (e2.isEmpty())
e2.add(e.get(exp2));

else
{
positionFound = false;

for (j=0; j < e2.size() && !positionFound; j++)
    {
    if (e2.get(j).e1 > e.get(exp2).e1)
        {
        positionFound = true;
        e2.add(j, e.get(exp2));
        }
    else if (e2.get(j).e1 == e.get(exp2).e1)
        {
        if (e2.get(j).e2 > e.get(exp2).e2)
            {
            positionFound = true;
            e2.add(j, e.get(exp2));
            }
        }
    }

    if (!positionFound)
    e2.add(e.get(exp2));
    }

    e2.setCardinality(e2.size());
    k2 -= (int)Math.pow(2,exp2);
    }
    exp2--;
    }
    add(e2);
    }
}

public void print()
{
int k;

System.out.print("{");
for(k=0; k < size(); k++)
    {
    super.get(k).print();

    if (k < size() - 1)
    System.out.print(", ");
    }
System.out.print("}");
}


public int getCardinality()
{
return cardinality;
}
}

class Component extends Vector<Integer>
{
private int cardinality;
private int maxVertex;
Component()
{
cardinality = 0;
maxVertex = 0;
}
```

```
Component(int x)
{
add(new Integer(x));
cardinality = 1;
maxVertex = x;
}

Component (Component c)
{
int k;

cardinality = c.getCardinality();
maxVertex = c.getMaxVertex();



for (k=0; k < cardinality; k++)
    add(c.get(k));
}

public void print()
{
int k;

System.out.print("{");

for (k=0; k < size(); k++)
    {
    System.out.print(get(k).intValue());
    if (k < size()-1)
    System.out.print(", ");
    }

System.out.print("}");
}

public void setCardinality(int x)
{
cardinality = x;
}

public int getCardinality()
{
return cardinality;
}

public boolean intersects(Component c)
{
boolean intersects = false;

int j;
int k;
```

```
for (j=0; j < getCardinality() && !intersects; j++)
    {
    for (k=0; k < c.getCardinality() && !intersects; k++)
        {
        if (get(j).intValue() == c.get(k).intValue())
        intersects = true;
        }
    }
return intersects;
}

public void merge(Component c)
{
int k;
int j;

if (getMaxVertex() < c.getMaxVertex())
setMaxVertex(c.getMaxVertex());

for(k=0; k < c.getCardinality(); k++)
    {
    j=0;
    while (j < getCardinality() && (get(j).intValue() < c.get(k).intValue()))
        j++;


    if (j == getCardinality() && (get(j-1).intValue() < c.get(k).intValue()))
    add(c.get(k));

    else if (get(j).intValue() > c.get(k).intValue())
    add(j, c.get(k));

    setCardinality(size());
    }
c.clear();
}

public void setMaxVertex(int x)
{
maxVertex = x;
}
public int getMaxVertex()
{
return maxVertex;
}

public boolean contains (int x)
{
int k;
boolean contains;
contains = false;
for (k=0; (k < cardinality) && !contains && (get(k).intValue() <= x); k++)
    {
    if (get(k).intValue() == x)
    contains = true;
    }

return contains;
}

public boolean equals (Component c)
{
int k;
boolean notEqual;

notEqual = false;

if (cardinality != c.getCardinality())
    notEqual = true;

if (!notEqual)
    {
    for (k=0; k < cardinality && !notEqual; k++)
        {
        if (get(k).intValue() != c.get(k).intValue())
        notEqual = true;
        }
    }

return !notEqual;
}
}

class Composition extends Vector<Component>
{
private int cardinality;
private int maxVertex;

Composition()
{
cardinality = 0;
maxVertex  = 0;
}

public void print()
{
int k;
System.out.print("{");
for(k=0; k < size(); k++)
    {
    super.get(k).print();

    if (k < size() - 1)
    System.out.print(", ");
    }

System.out.print("}");
}
```

```
public void setCardinality(int x)
{
cardinality = x;
}

public int getCardinality()
{
return cardinality;
}

public void sweep()
{
int k;

for (k=0; k < getCardinality(); k++)
    {
    if (get(k).isEmpty())
        {
        remove(k);
        setCardinality(size());
        }
    }
}

public void addMissingComponents(int m)
{
int j;
int k;

Vector<Component> singleComponents;

singleComponents = new Vector<Component>();

boolean componentContainsJ;

        for (j=m; j > 0; j--)
        {
        componentContainsJ = false;
        for (k=0; k < cardinality && !componentContainsJ;  k++)
        {
        if(get(k).contains(j))
        componentContainsJ = true;
        }
    if (!componentContainsJ)
    singleComponents.add(new Component(j));
    }

for (k=0; k < singleComponents.size(); k++)
    add(singleComponents.get(k));

setCardinality(size());
}

public void sort()
```

```
{
int j;
int k;

int n;

boolean swapped;

Component c;

for (j=0; j < cardinality; j++)
    {
    for (k=j+1; k < cardinality; k++)
        {
        if (get(j).getCardinality() > get(k).getCardinality())
            {
            c = new Component(get(j));
            set(j, get(k));
            set(k, c);
            }

        else if (get(j).getCardinality() == get(k).getCardinality())
            {
            swapped = false;
            for (n = 0; n < get(j).getCardinality() && !swapped; n++)
                {
                if (get(j).get(n).intValue() > get(k).get(n).intValue())
                    {
                    c = new Component(get(j));
                    set(j, get(k));
                    set(k, c);
                    swapped = true;
                    }
                else if (get(j).get(n).intValue() < get(k).get(n).intValue())
                swapped = true;
                }
            }
        }
    }
}

public int getMaxVertex()
{
return maxVertex;
}

public void setMaxVertex(int x)
{
maxVertex = x;
}

public boolean equals(Composition c)
{
int k;
```

```
boolean notEqual;

notEqual = false;

if (cardinality != c.getCardinality())
    notEqual = true;

if (!notEqual)
    {
    for (k=0; k < cardinality && !notEqual; k++)
        {
        if (!get(k).equals(c.get(k)))
            notEqual = true;
        }
    }


return !notEqual;
}
}

class CompositionSet extends Vector<Composition>
{
private int cardinality;
private int maxVertex;

CompositionSet()
{

}

public int getMaxVertex()
{
return maxVertex;
}

public void setMaxVertex(int x)
{
maxVertex = x;
}

public void getPowerSet(PowerSet powerSet)
{
Composition comp;
Component c;

int k;
int j;

cardinality = powerSet.getCardinality();

for (k=0; k < cardinality; k++)
    {
    comp = new Composition();
    for (j=0; j < powerSet.get(k).getCardinality(); j++)
```

```
            {
            c = new Component();
            c.add(new Integer(powerSet.get(k).get(j).e1));
            c.add(new Integer(powerSet.get(k).get(j).e2));
            c.setMaxVertex(powerSet.get(k).get(j).e2);
            c.setCardinality(c.size());
            comp.add(c);

            if (comp.getMaxVertex() < c.getMaxVertex())
                comp.setMaxVertex(c.getMaxVertex());
            }
        comp.setCardinality(comp.size());
        add(comp);
        }
}

public void buildCompositions()
{
int k;

k = 0;
while (k < cardinality)
    {
    while (intersection(get(k)))
    merge(get(k));

    get(k).sort();
    get(k).addMissingComponents(getMaxVertex());

    k = trim(get(k), k);
    }
}

public boolean intersection(Composition c)
{
int j;
int k;

boolean intersection;

intersection = false;
for (j=0; j < c.getCardinality() && !intersection; j++)
    {
    for (k=j+1; k < c.getCardinality() && !intersection; k++)
        {
        if (c.get(j).intersects(c.get(k)))
            intersection = true;
        }
    }

return intersection;
}

public void print()
```

```
{
int k;

System.out.print("{");
for(k=0; k < size(); k++)
    {
    get(k).print();

    if (k < size() - 1)
        System.out.print(", ");
    }

System.out.print("}");
}

public void merge(Composition c)
{
int j;
int k;
boolean mergeComplete;

mergeComplete = false;

for(j=0; j < c.getCardinality() && !mergeComplete; j++)
    {
    for(k=j+1; k < c.getCardinality() && !mergeComplete; k++)
        {
        if (!(c.get(j).isEmpty() || c.get(k).isEmpty()))
            {
            if (c.get(j).intersects(c.get(k)))
                {
                c.get(j).merge(c.get(k));
                mergeComplete =  true;
                }
            }
        c.get(j).setCardinality(c.get(j).size());
        }
    c.sweep();
    }
}

public int trim(Composition c, int position)
{
int k;
boolean found;

found = false;
for (k=0; k < position && !found; k++)
    {
    if (c.equals(get(k)))
        {
        remove(c);
        cardinality--;
        found = true;
```

```
          }

     }

if (!found)
     position++;

return position;
}

public void sweep()
{
int k;

Vector<Composition> removeCompositions;

removeCompositions = new Vector<Composition>();

k = 0;
while (k < cardinality)
     {
     if (get(k).isEmpty())
          {
          removeCompositions.add(get(k));
          cardinality = size();
          }

     else
     k++;
     }
}

public int getCardinality()
{
return cardinality;
}
}

class TFunction
{
TFunction()
{

}

public static void pause()
{
System.out.println("Press any key to continue...");
new Scanner(System.in).nextLine();
}
}
```

Bibliography

[1] H.W. Gould, *Research bibliography of two special sequences*, Sixth Edition, 1985.

[2] Hyeong-Kwan Ju and Seunghyun Seo, *Enumeration of (0,1)-matrices avoiding some 2 x 2 matrices*, Discrete Math **312** (2012), 2473–2481.

[3] Brian Kell, *Compositions of series-parallel graphs*, Unpublished, 3 2006.

[4] A. Knopfmacher and M.E. Mays, *Graph compositions 1: Basic enumeration*, Integers **1** (2001), 1–11.

[5] Eunice Mphako-Banda, *Graph compositions and flats of cycle matroids*, Quaestiones Mathematicae **32** (2009), 523–527.

[6] C.S. Peirce, *On the algebra of logic*, Amer. J. Math **3** (1880), 15–57.

[7] J.N. Ridley and M.E. Mays, *Compositions of unions of graphs*, Fibonacci Quart. **42** (2004), 222–230.

[8] Yidong Sun and Xiaojuan Wu, *The largest singletons of set partitions*, European J. Combin **32** (2011), 369–382.