Ball State Undergraduate Mathematics Exchange, 18(1), Fall 2024, 26-50
https://openjournals.bsu.edu/mathexchange
DOI: 10.33043/92ra679d

Pages 26 – 50

Finding the Cheapest Way to Build a Graph

*Shu Qian, Samiha Rao**

Shu Qian was a senior at George Washington University majoring in Pure Mathematics when working on this paper. She is currently a master student in the Mathematics program at New York University. She is interested in Number Theory and Combinatorics but is still exploring other interesting fields.

Samiha Rao worked on this paper while a senior at George Washington University where she pursued a double major in Applied Mathematics and International Affairs. She is currently working at Bristol Myers Squibb in Information Technology, specializing in analytics. She is passionate about working in spaces where math and analytics can help bring social change.

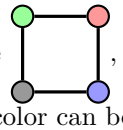
Abstract

The Concept Reinforcement Problem is a graph optimization problem introduced by Novikoff. One seeks to build a graph G vertex by vertex in the cheapest way possible for that graph. The cost function for each vertex is a positive, decreasing, convex function where the input is determined by the number of neighbors already built. We solve this problem for a variety of different graphs such as simple connected small-sized graphs, trees, cycles, wheels, grid graphs, ladder graphs, and complete bipartite graphs.

1 Introduction

1.1 Background Consider building a network G vertex by vertex. The cost of building this graph is the sum of the costs of vertices, and the cost of building each vertex v is determined by the number of neighbors of v that have already been built. For an arbitrary graph $G = (V, E)$, there are different

***Corresponding author**:samiharao1002@gmail.com

ways to build it. For example, for the four-vertex cycle  , if the green vertex is installed first, any other vertex in a different color can be built next. Installing a vertex v_i costs $f(k)$ where k is the number of vertices built before v_i and adjacent to v_i. So in the case of the four-vertex square, if we build the vertices with permutation $\{\text{green}, \text{red}, \text{blue}, \text{black}\}$, the total cost of this graph is $f(0)+f(1)+f(1)+f(2)$; but if the permutation is $\{\text{green}, \text{blue}, \text{red}, \text{black}\}$, the cost is changed to $f(0)+f(0)+f(2)+f(2)$. Which permutation gives a cheaper cost? How about other graphs? Is there a patterned way to build an arbitrary graph to obtain the cheapest installing cost?

In this paper, we consider a graph-theoretic optimization problem introduced by Novikoff in his study of the Concept Reinforcement Problem. We will use standard terminology and notations from graph theory as found in Bonin's *Introduction to Graph Theory* [2].

Novikoff's thesis gives an introduction to our problem discussed in this paper. He showed that for arbitrary cost functions, the problem is computationally intractable in general [1, Section 5.3.1]. However, for a linear cost function $f(k) = ak + b$, the problem is simple: as described in [1, Section 5.3.2], the optimal cost is $am + bn$ where m is the number of edges and n is the number of vertices in G, and it is achieved for any building order. This led Novikoff to mention the Concept Reinforcement Problem, where he discusses what happens when f is convex and decreasing, which is the case we consider in this paper. One consequence of convexity that will be of great use to us is the following formula: if $b \leq a-2$ then

$$f(a)+f(b) > f(a-1)+f(b+1). \tag{1. A}$$

Novikoff applies this formula in the case of two adjacent vertices that are built consecutively, where the first vertex gives a cost of $f(a)$ and the second vertex gives a cost of $f(b)$ such that $b \leq a-2$; in this case, it will be cheaper to switch the order of these vertices in this permutation. Once these vertices are switched, Novikoff remarks that the costs become $f(a-1)$ and $f(b+1)$, since the earlier vertex would have lost an already built neighbor. Then, the cost of the two vertices combined would be cheaper than before due to convexity, while the cost of all other vertices would be unchanged.

Our problem is to find the cheapest permutation of a graph, using inequality 1. A to analyze different families of graphs. Some families of graphs we will explore include graphs on less than five vertices, trees, cycles, wheels, and some bipartite graphs.

1.1 Our Results

Since there are not many simple connected graphs with one, two, three, or four vertices, we list all graphs and all possible cost expressions for each graph in Section 2, and we compare their costs to find the optimal cost. In this paper, we also check and prove the optimal cost for paths, stars (which are two kinds of trees), general trees, cycles, wheels, and some bipartite graphs including ladder graphs, grid graphs, and complete bipartite graphs.

In Section 3, we consider trees with n vertices, including paths (Section 3.1), stars Section 3.2, and general trees Section 3.3. We show that the cheapest cost for a tree is $f(0)+(n-1)f(1)$, which we get when we first build an arbitrary vertex and then keep building vertices that have a neighbor already built.

For cycles with n vertices, we build an arbitrary vertex first and then keep building vertices that have a neighbor already built. Then we achieve the cheapest cost $f(0)+(n-2)f(1)+f(2)$, as shown in Section 4.

For wheels with n vertices, we show in Section 5 that the cheapest cost is $f(0)+f(1)+(n-3)f(2)+f(3)$. We can get this cost when we start by building two vertices, where one of them must be the central vertex and the other is arbitrary. Then we keep building the other vertices such that each of them have at least two pre-existing neighbors.

In Section 6, we look at bipartite graphs, focusing on ladder graphs, grid graphs, and complete bipartite graphs. The ladder graph L_n is formed by taking two paths of n vertices and connecting corresponding vertices by an edge, as illustrated in Figure 1. In Section 6.1, we prove that the optimal cost for L_n is $f(0)+nf(1)+(n-1)f(2)$, which can be achieved when we first build the vertices of one path in order, then build the vertices of the other path in order. Then in Section 6.2, we generalize this for the cheapest cost of a grid graph $G_{m\times n}$. Similar to the construction of the ladder graph, we conclude that the cheapest cost for the graph $G_{m\times n}$ with mn vertices is $f(0)+(m+n-2)f(1)+(mn-m-n+1)f(2)$. In Section 6.3, we show that for a complete bipartite graph $K_{m,n}$ where $n\leq m$, the optimal cost is $f(0)+2[f(1)+f(2)+\cdots+f(n-1)]+(m-n+1)f(n)$. To accomplish this, we alternate between the two vertex sets until all vertices in one set are built, and then build any remaining vertices.

We mainly use the inequality 1. A to prove these results. In some proofs, we also use the fact that the sum of all inputs of the cost function for a graph is equal to the number of edges of the whole graph. Take the four-vertex cycle above as an example. Its two possible cost expressions are $f(0)+f(1)+f(1)+f(2)$ and $f(0)+f(0)+f(2)+f(2)$. The sum of the inputs $0+1+1+2=0+0+2+2=4$ is the number of edges in this four-vertex cycle.

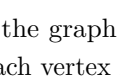
Lemma 0.1. *For any graph G, for any order on the vertices, the sum of the inputs to the cost function when building G is the number of edges of G.*

Proof. Each input of the cost function of a vertex v is the number of its neighbors built previously, and each of these neighbors is connected by exactly one edge to the vertex v. Thus, when we sum the inputs, each edge is counted exactly

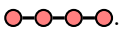
once, for the second of its endpoints that we build. Therefore, the sum of all inputs is the total number of edges. □

2 Simple connected graphs with fewer than five vertices

Obviously, there is only one possible cost for $n = 1$ and $n = 2$. When $n = 1$, the cost is $f(0)$, and when $n = 2$, the cost is $f(0) + f(1)$.

When $n = 3$, the graph can be the path  or the cycle  . For the cycle, since each vertex is adjacent to all of other vertices, the cost can only be $f(0) + f(1) + f(2)$. For the path, when we pick one of these three vertices as our first vertex, it is either (1) one of the leaves, or (2) the central vertex, and the cost of this vertex is $f(0)$ because no vertex has been built before. If the first vertex we build is one of the leaves (case (1)), the second vertex can either be (a) the central vertex or (b) the other leaf. If the central vertex is the second vertex, its cost is $f(1)$, and the cost for the path is $f(0) + f(1) + f(1)$ for case (1a). If the second vertex is another leaf, its cost is $f(0)$ because it is not adjacent to the first built leaf, then the cost of the central vertex is $f(2)$. So the cost in case (1b) is $f(0) + f(0) + f(2)$. In the case (2), when the central vertex is our first vertex, the cost is $f(0) + f(1) + f(1)$ because it is the same no matter which leaf is our second vertex. Therefore, there are two possible costs for the path: $\text{Cost}_1 = f(0) + f(1) + f(1)$ and $\text{Cost}_2 = f(0) + f(0) + f(2)$. By the convexity of f, since we have $0 \leq 2 - 2$, we get the inequality $f(1) + f(1) < f(0) + f(2)$, so $\text{cost}_1 < \text{cost}_2$.

When $n = 4$, there are six different simple connected graphs [3]. We will discuss them separately below. Since we can pick any of four vertices to be the first vertex, any of the remaining three to be the second, and either of the remaining two to be the third one, for each graph with four vertices, there are $4! = 24$ orders to build it. However, depending on the different properties of the graph, such as its symmetries or its greatest degree, each graph does not give 24 distinct cases.

The first graph discuss is the path $G_1 =$  . The cost of the first vertex in any permutation is $f(0)$, so the first term in any expression is $f(0)$. Since the second vertex can either be a vertex that is not adjacent to the first or be a vertex adjacent, the cost of the second term can be $f(0)$ or $f(1)$. If the first two are not adjacent, the third vertex has to be adjacent to at least one of them, so the cost is either $f(1)$ or $f(2)$. After knowing the first three vertices, the cost of the fourth vertex is also decided, and it is the other of $f(1)$ and $f(2)$. If the first two vertices are adjacent, the cost of the third vertex can be $f(0)$, when the first two are not the central ones and the third is the degree-one vertex that is none of the first two (and in this case the fourth vertex costs $f(2)$), or $f(1)$ (and in this case the fourth vertex costs $f(1)$). So there are only four possible

cost expressions for G_1:

$$f(0)+\begin{cases} f(0)+\begin{cases} f(1)+f(2) & \text{(2. B)} \\ f(2)+f(1) & \text{(2. C)} \end{cases} \\ f(1)+\begin{cases} f(0)+f(2) & \text{(2. D)} \\ f(1)+f(1) & \text{(2. E)} \end{cases} \end{cases}$$

Since the value of the first three are equal, we only need to compare one of these expressions and the expression 2. E. By the inequality 1. A, we have the inequality $f(a)+f(b) > f(a-1)+f(b+1)$ for all $b \le a-2$. Since $0 \le 2-2$, we have that $f(0)+f(2) > f(1)+f(1)$. Therefore, the term 2. E is the cheapest cost for the graph G_1.

By a similar case-analysis, the possible costs of the graph $G_2 =$  (discussed in the introduction) are

$$f(0)+\begin{cases} f(0)+f(2)+f(2) & \text{(2. F)} \\ f(1)+f(1)+f(2) & \text{(2. G)} \end{cases}$$

Using the inequality $f(a)+f(b) \ge f(a-1)+f(b+1)$ for all $b \le a-2$, we have $f(0)+f(2) > f(1)+f(1)$. Therefore, the term 2. G is cheaper than the term 2. F.

There are three fundamentally different orders in which to build the graph $G_3 =$  . Their costs are

$$f(0)+\begin{cases} f(0)+\quad f(2)+f(3) & \text{(2. H)} \\ f(1)+\begin{cases} f(1)+f(3) & \text{(2. I)} \\ f(2)+f(2) & \text{(2. J)} \end{cases} \end{cases}$$

Invoking inequality 1. A, and that $0 \le 3-2$ and $1 \le 3-2$, we get

$$f(0)+f(3) > f(1)+f(2) \qquad \text{(2. K)}$$

and

$$f(1)+f(3) > f(2)+f(2). \qquad \text{(2. L)}$$

Adding $f(0)+f(2)$ to the both sides of the inequality 2. K, we have that the term 2. H is greater than the term 2. J. Similarly, adding $f(0)+f(1)$ to both sides of the inequality 2. L, we get that term 2. I is greater than term 2. J. Therefore, the term 2. J is the cheapest one for the graph G_3.

Let G_4 be the graph  . The cost can be

$$f(0)+\begin{cases} f(0)+\begin{cases} f(1)+f(3) & \text{(2. M)} \\ f(2)+f(2) & \text{(2. N)} \end{cases} \\ f(1)+\begin{cases} f(0)+f(3) & \text{(2. O)} \\ f(1)+f(2) & \text{(2. P)} \\ f(2)+f(1) & \text{(2. Q)} \end{cases} \end{cases}$$

The term 2. P is equal to the term 2. Q, and the term 2. M is equal to the term 2. O. Compare the term 2. M and the term 2. N, using the inequality 1. A, since $f(1)+f(3) > f(2)+f(2)$, we get that the term 2. N is less than the term 2. M. Similarly, we have $f(0)+f(2) > f(1)+f(1)$ by the inequality 1. A. Adding $f(0)+f(2)$ to both sides, we can get that the terms 2. P and 2. Q are

the cheapest cost for G_4.

Let G_5 denote the star  . The possible costs are

$$f(0)+\begin{cases} f(0)+\begin{cases} f(0)+f(3) & \text{(2. R)} \\ f(2)+f(1) & \text{(2. S)} \end{cases} \\ f(1)+\quad f(1)+f(1) & \text{(2. T)} \end{cases}$$

As above, we have $f(0)+f(2) > f(1)+(1)$. So the term 2. S is greater than the term 2. T. Since $f(0)+f(3) > f(2)+f(1)$, we get the term 2. R is greater than the term 2. S. Thus, the term 2. R is greater than the term 2. T. In conclusion, the term 2. T is the cheapest one for G_5.

Let G_6 be the graph  . The cost for G_6 is $f(0)+f(1)+f(2)+f(3)$ in any order.

In this section for graphs on 4 vertices or fewer, we use the brute-force approach to find the cheapest order to build the graphs. Our reasoning for this is that once we start to look at graphs on vertices of 5 or more, the number of connected graphs increase drastically. For example, the number of connected graphs on 5 vertices is 21, and the number of connected graphs on 6 vertices is 112. If we jump to 7 vertices, there are 853 possible connected graphs [4]. Although these graphs would be interesting to study, this section only analyzes graphs up to 4 vertices for simplicity's sake. However, we will look at graphs on 5 and 6 vertices later in this paper, in Section 7, using a computer to do a cost analysis. In later sections, we also discuss cheapest orders for infinite families of graphs.

3 Trees

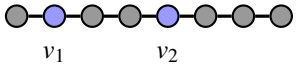
Trees are graphs where any two vertices are connected by exactly one path, wherein there are no cycles. In this section, we start with simplest form of trees such as paths and stars. Then we move on to the case of general trees.

3.1 Paths

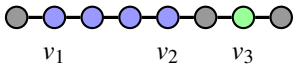
A path is an alternating sequence of vertices and edges with no repeating vertices. Thus, all vertices in a path will have degree two except the first and last vertex, which will have degree one.

Theorem 1. *The cost* $f(0)+f(1)+f(1)+\cdots+f(1)$ *is the cheapest cost for a path.*

Proof. If we start to build a path from one leaf and every vertex we build next is the vertex next to the vertex we built before, then we end by building the other leaf. This proves that the cost $f(0)+f(1)+\cdots+f(1)$ is achievable.
In general, the degree of any vertex in a connected path is either 1 or 2, so the input of a cost function for a vertex is at most 2. The first vertex v_1 that is built will always cost $f(0)$. After the first vertex, if we build a new isolated vertex v_2, that would add an $f(0)$ to the cost expression. Then, there has to be another vertex, built later, between these two vertices and adjacent to two vertices built before in order to create a path from v_1 to v_2. So this vertex costs $f(2)$.



After forming this path, we assign another isolated vertex, v_3. Again, there must be a vertex, in the later construction, between v_2 and v_3 that connects both to make a path.



The first vertex, v_1 has cost $f(0)$, but whenever there is another $f(0)$, there also must be an $f(2)$ in the later part of the expression. Thus, the general expression is equal to

$$f(0)+k[f(0)+f(2)]+mf(1)$$

where $0 \le k \le (n-1)/2$ when n is odd or $0 \le k \le (n-2)/2$ when n is even, and $m = n-(2k+1)$. Now, we can use the inequality 1. A to compare 0 and 2 with the inputs 1 and 1. Since $f(0)+f(2) > f(1)+f(1)$, we see that $f(0)+f(1)+f(1)+\cdots+f(1)$ is the cheapest cost for a path. □

3.2 Stars

A star is a tree that has one vertex with degree $n-1$ and all other vertices have degree one.

Theorem 2. *The cost* $f(0)+f(1)+\cdots+f(1)$ *is a cheapest cost for a star.*

Proof. We can first pick either the central vertex or a non-central vertex, and the cost of this vertex is $f(0)$. If the central vertex is our first vertex, we get our expression $f(0)+f(1)+\cdots+f(1)$. In the case when a non-central vertex is the first vertex, the cost of a non-central vertex before the central vertex is $f(0)$, the

cost of the central vertex is $f(m)$ where m is the number of vertices built before the central vertex, and the cost of any vertex after the central vertex is $f(1)$. For example, if a non-central vertex is the first vertex, and it is followed by the central vertex, then the expression will be $f(0)+f(1)+\cdots+f(1)$. Therefore, the general expression for a star with n vertices in this case is

$$\text{Cost}_{n\text{-star}}(m) = \underbrace{f(0)+\cdots+f(0)}_{m}+f(m)+\underbrace{f(1)+\cdots+f(1)}_{n-(m+1)}$$

Thus, we have Thus, we have

$$\begin{aligned}\text{Cost}_{n\text{-star}}(i) &= \underbrace{f(0)+\cdots+f(0)}_{i}+f(i)+\underbrace{f(1)+\cdots+f(1)}_{n-(i+1)}\\ &= \underbrace{f(0)+\cdots+f(0)}_{i}+f(i)+f(1)+\underbrace{f(1)+\cdots+f(1)}_{n-(i+2)}.\end{aligned}$$

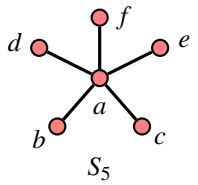
and

$$\begin{aligned}\text{Cost}_{n\text{-star}}(i+1) &= \underbrace{f(0)+\cdots+f(0)}_{i+1}+f(i+1)+\underbrace{f(1)+\cdots+f(1)}_{n-(i+2)}\\ &= \underbrace{f(0)+\cdots+f(0)}_{i}+f(0)+f(i+1)+\underbrace{f(1)+\cdots+f(1)}_{n-(i+2)}.\end{aligned}$$

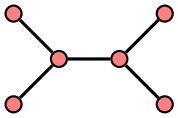
where $1 \le i \le n-2$. By inequality 1. A, we get that that $f(1)+f(i) < f(0)+f(i+1)$ when $0 \le (i+1)-2$, i.e., when $i \ge 1$. So $_{n\text{-star}}(i) <_{n\text{-star}} (i+1)$ for all $1 \le i \le n-2$. Thus, the expression $_{n\text{-star}}(1) = f(0)+f(1)+\cdots+f(1)$ is the cheapest cost for a star. □

3.3 General Trees

Since trees are quite complicated, we start by finding a general cost expression for building trees in any order. Then, we minimize this cost using inequality 1. A repeatedly to find the order for the cheapest cost.



S_5



Lemma 2.1. *For any tree T and any order of its vertices, the cost is*

$$f(0)+\sum_{i=1}^{d} a_i[(i-1)f(0)+f(i)]$$

for some nonnegative integers a_i, where d is the maximum vertex degree of T.

Example 2.1. *Take the graph S_5 shown above as an example. If we build the star in the order $\{a,b,c,d,e,f\}$, the cost will be $f(0)+5f(1)$. In this case, $a_1 = 5$ and $a_i = 0$ for $i = 2,\cdots,5$. If we build the star in the order $\{b,c,e,a,d,f\}$, the cost will be $3f(0)+2f(1)+f(3) = f(0)+2[0\cdot f(0)+f(1)]+2f(0)+f(3)$ where $a_1 = 2$, $a_3 = 1$ and $a_2 = a_4 = a_5 = 0$.*

Proof of Lemma 2.1. Fix an n-vertex tree T and some permutation of its ver-

tices, and let a_i be the number of vertices that cost $f(i)$. Thus, we have $\sum_{i=0}^{d} a_i = n$, and the cost for the tree is $a_0 f(0) + a_1 f(1) + \cdots + a_d f(d)$, where d is the greatest degree of the tree. By Lemma 0.1, we have $a_0 \cdot 0 + \sum_{i=1}^{d} a_i \cdot i = |E|$. Since $|E| = |V| - 1 = n - 1$ in a tree, by the Handshaking Lemma, we have $a_0 \cdot 0 + \sum_{i=1}^{d} a_i \cdot i = \sum_{i=1}^{d} a_i \cdot i = n - 1$, so $n = 1 + \sum_{i=1}^{d} a_i \cdot i$. Since $\sum_{i=0}^{d} a_i = n$, we have

$$\begin{aligned} a_0 &= n - \sum_{i=1}^{d} a_i \\ &= 1 + \sum_{i=1}^{d} a_i \cdot i - \sum_{i=1}^{d} a_i \\ &= 1 + \sum_{i=1}^{d} a_i \cdot (i-1). \end{aligned}$$

Therefore, the cost of the tree is

$$\begin{aligned} a_0 f(0) + a_1 f(1) + \cdots + a_d f(d) &= \left[1 + \sum_{i=1}^{d} a_i \cdot (i-1)\right] f(0) + \sum_{i=1}^{d} a_i f(i) \\ &= f(0) + \sum_{i=1}^{d} a_i \cdot (i-1) f(0) + \sum_{i=1}^{d} a_i f(i) \\ &= f(0) + \sum_{i=1}^{d} a_i \left[(i-1) f(0) + f(i)\right]. \square \end{aligned}$$

Lemma 2.2. *A graph G has an order with cost $f(0) + f(1) + f(1) + \cdots + f(1)$ if and only if G is a tree.*

Proof. First assume that G is a tree. We pick an arbitrary vertex, costing $f(0)$, to be our first vertex. After this, if we continue choosing to build vertices that already have a neighbor built, then the cost of each of these vertex is $f(1)$. So the total cost is $f(0) + f(1) + f(1) + \cdots + f(1)$. Next, assume that the graph G has a permutation with cost $f(0) + f(1) + f(1) + \cdots + f(1)$. To prove G is a tree, we need to prove (i) G is connected, and (ii) G has no cycles. Since there is only one $f(0)$, the graph G has to be connected, as the number of $f(0)$ is at least the number of components. If G has a cycle, then the last vertex of the cycle to be build will cost $f(k)$ with $k \geq 2$. But there is no $f(2)$ in the cost expression, so G is acyclic. Since G is a connected acyclic graph, we have that G is a tree. $\square$

Theorem 3. *The cost $f(0) + f(1) + \cdots + f(1)$ is a cheapest cost for any tree.*

Proof. By Lemma 2.2, for any tree we can achieve the cost expression $f(0) + f(1) + \cdots + f(1)$. From Lemma 2.1, we know that the cost for an arbitrary order of a tree is $f(0) + \sum_{i=1}^{d} a_i[(i-1)f(0) + f(i)]$. Using inequality 1. A, our strategy is to minimize the expression by repeatedly replacing expressions of the form $f(0) + f(a)$ with $f(1) + f(a-1)$. We continue taking away $f(0)$s and implementing this substitution until all $f(0)$s are exhausted. Indeed, for each

term $(i-1)f(0)+f(i)$ where $i \geq 2$, we get

$$\begin{aligned}
\underbrace{f(0)+f(0)+\cdots+f(0)}_{i-1}+f(i) &= \underbrace{f(0)+f(0)+\cdots+f(0)}_{i-2}+f(0)+f(i) \\
&> \underbrace{f(0)+f(0)+\cdots+f(0)}_{i-2}+f(1)+f(i-1) \\
&= \underbrace{f(0)+f(0)+\cdots+f(0)}_{i-3}+f(0)+f(i-1)+f(1) \\
&> \underbrace{f(0)+f(0)+\cdots+f(0)}_{i-3}+f(1)+f(i-2)+f(1) \\
&\vdots \\
&> f(0)+f(1)+f(2)+\underbrace{f(1)+\cdots+f(1)}_{i-3} \\
&> f(1)+f(1)+f(1)+\underbrace{f(1)+\cdots+f(1)}_{i-3} \\
&= f(1)+\cdots+f(1).
\end{aligned}$$

Therefore, for each $a_i[(i-1)f(0)+f(i)]$ where $1 \leq i \leq d$, we have $a_i[(i-1)f(0)+f(i)] > a_i \cdot i \cdot f(1)$. Since $\sum_{i=1}^{d} a_i \cdot i = n-1$, we conclude that the cost $f(0)+f(1)+\cdots+f(1)$ is a cheapest cost for a tree. □

4 Cycles

In Section 2, we computed costs for all 3-vertex and 4-vertex graphs. For the 3-vertex cycle  , every permutation gives the cost expression $f(0)+f(1)+f(2)$, while for the four-cycle  , the two possible cost expressions are

$$f(0)+\begin{cases} f(0)+f(2)+f(2) \\ f(1)+f(1)+f(2). \end{cases}$$

Using inequality 1. A, we were able to compare different build orders to find that the cheapest cost is $f(0)+f(1)+f(1)+f(2)$. We can use this framework to find the cheapest cost expression for cycles of any length.

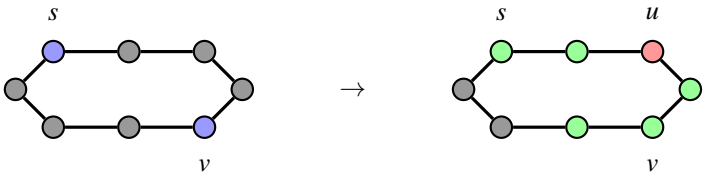
Theorem 4. *The cost $f(0)+f(1)+f(1)+\cdots+f(1)+f(2)$ is the cheapest for a cycle.*

Proof. We can achieve this if we build a cycle clockwise or counterclockwise and each vertex built next is a neighbor of the vertex built before. Now we show that it is minimal.

As in each example above, the cost expression of a cycle starts with the first isolated vertex resulting in $f(0)$. Also, each expression of a cycle has at least one $f(2)$: by definition, every vertex in a cycle has degree 2, and therefore the last vertex will always have two neighbors already built, resulting in the last $f(2)$ in the expression.

After building the first vertex s, we can either build an adjacent vertex costing

$f(1)$ or build an isolated vertex. But whenever we build another isolated vertex, v, costing $f(0)$, this isolated vertex will be in a part that is not connected to the first vertex. Thus, when we build other vertices later, there is a "connecting" vertex, u, connecting the part where the first vertex s is to the isolated part where the vertex v is. So in a later term of the expression, there is an $f(2)$ for the vertex u. An example is below. The vertex u connects the two green parts where s and v separately are.



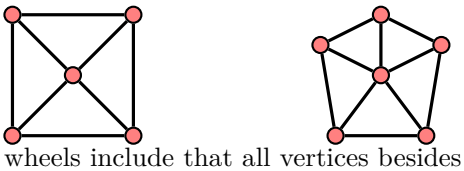
This continues similarly when we build other new isolated vertices: each $f(0)$ corresponds to creating a new component, and each $f(2)$ (other than the last vertex) corresponds to merging two components. Thus, the general cost expression of a cycle with n vertices is

$$m[f(0)+f(2)]+(n-2m)f(1)$$

where $1 \leq m \leq \left\lfloor \frac{n}{2} \right\rfloor$. This general formula allows us to use inequality 1. A in the form $f(0)+f(2) > f(1)+f(1)$ to reduce m by 1 while making the cost smaller. Therefore, the cost $f(0)+f(1)+f(1)+\cdots+f(1)+f(2)$ is the cheapest cost for a cycle. □

5 Wheels

As described in [2], a *wheel* with n vertices is formed from a cycle C_{n-1} (called the *rim*) by adding a vertex (called the *hub*) and, for each vertex in the rim, an edge that is incident with that vertex and the hub. Some examples of wheels are as follows:



Some properties of wheels include that all vertices besides the central vertex have degree 3 and the central vertex has degree $n-1$.

Lemma 4.1. *The general cost expression of a wheel is*

$$af(0)+bf(1)+cf(2)+f(a+b+c)+(a-c)f(3)+k[f(1)+f(3)]+\ell f(2)$$

where a, b, c, k, l are nonnegative integers such that $\ell = n-(2a+b+1+2k)$ *and either* $a=b=c=0$, *or* $a=c$ *and* $a+b+c=n-1$ *and* $k=0$, *or* $c \leq a-1$.

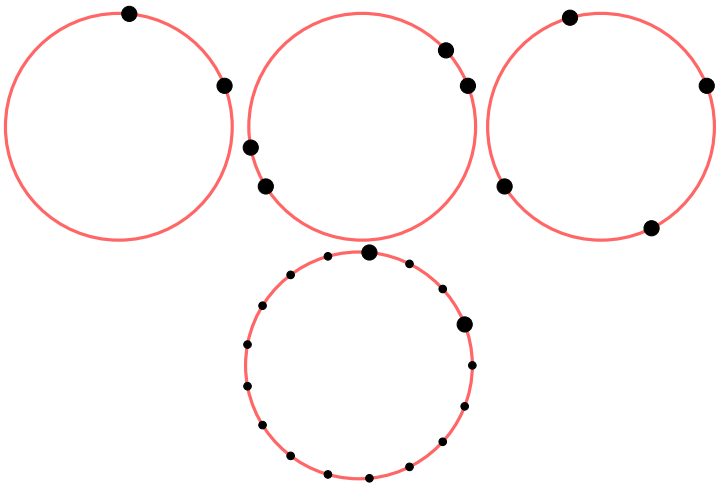
Proof. In the case where we build the hub first, we see by the proof of Theorem 4 that the resulting cost will be $f(0)+k[f(1)+f(3)]+(n-1-2k)f(2)$ where k and $n-1-2k$ are nonnegative integers, and this is exactly the case $a=b=c=0$ of the claim.

Similarly, in the case where we build the hub last, we see from the proof of Theorem 4 that the resulting cost will be $a[f(0)+f(2)]+(n-1-2a)f(1)+f(n-$

1), which is exactly the case $a = c$ and $a+b+c = n-1$ and $k = 0$ of the claim. Now consider the case that we do not build the hub first or last.

Let us divide the target expression into two parts: (i) $af(0)+bf(1)+cf(2)$ which will represent the cost before we build the hub, and (ii) $f(a+b+c)+(a-c)f(3)+k[f(1)+f(3)]+\ell f(2)$ which will represent the cost of the part of the graph that we build after the hub (including the cost of the hub itself).

For the first part, let $i = a+b+c$ be the number of vertices that we build on the rim before we build the hub. The cost of each of these vertices is either $f(0)$, $f(1)$, or $f(2)$. Let a be the number of these vertices that cost $f(0)$, let b be the number that cost $f(1)$, and let c be the number that cost $f(2)$. At this stage, a vertex costing $f(2)$ is only built when it joins two components, each of which was started by a vertex that cost $f(0)$, so $c \le a-1$.



Then build the hub, which has i neighbors on the rim and so costs $f(i)$. As we complete the rim, there must be $m = a-c$ vertices that join rim components that were built before the hub, and these all cost $f(3)$; moreover, each time we create a new rim component (at a cost $f(1)$), we are forced later to merge two rim components (at a cost $f(3)$). Let the number of times we do that be k, accounting for $2k$ vertices. Each of the other $\ell = n-(i+1+m+2k)$ vertices not already accounted for is built on the rim with one rim neighbor and the hub as a neighbor, and so costs $f(2)$. Every vertex falls into one of these cases; substituting $m = a-c$ and $i = a+b+c$ where needed completes the proof. □

Theorem 5. *The cheapest cost for the wheel with n vertices is $f(0)+f(1)+(n-3)f(2)+f(3)$.*

Proof. First we show that this value is achievable. We pick the central vertex first, with cost $f(0)$. Then we build vertices consecutively around the cycle. The cost of the first vertex on the cycle in this order is $f(1)$, and the cost of the last vertex (which closes the wheel) is $f(3)$. Each other vertex is adjacent to the central vertex and the one built immediately before it, so has cost $f(2)$. Therefore the cost expression under this construction is $f(0)+f(1)+(n-3)f(2)+f(3)$. We can also get this cost expression if one vertex on the wheel is built first and

the central vertex is built second. We claim this cost expression is the cheapest cost for the wheel.

Now consider an arbitrary order, with cost given by Lemma 4.1. By the inequality 1. A, we have $f(1)+f(3)>2f(2)$ and $f(0)+f(2)>2f(1)$. By applying the first of these inequalities $k+(a-c-1)$ times and the second one $a-1$ times, we get

$$\begin{aligned} &af(0)+bf(1)+cf(2)+f(a+b+c)+(a-c)f(3)+\\ &\quad+(n-2a-b-1-2k)f(2)+k[f(1)+f(3)]\\ &>f(0)+(a+b+c-1)f(1)+(n-a-b-c-2)f(2)+f(a+b+c)+f(3). \end{aligned} \tag{5. U}$$

If $a+b+c=1$ or 2, this is exactly the inequality we claimed. Suppose instead that $a+b+c\geq 3$, and let us deal first with the sum of the second and the fourth terms $(a+b+c-1)f(1)+f(a+b+c)$. By the inequality 1. A, we get $f(1)+f(i)>f(2)+f(i-1)$ for all $i\geq 3$. We keep pairing one copy of $f(1)$ with the extra term, applying the inequality $f(1)+f(a+b+c-k)>f(2)+f(a+b+c-k-1)$ for $k=0,1,\cdots,a+b+c-3$. Thus, for $a+b+c\geq 3$, we have the inequalities

$$\begin{aligned} &(a+b+c-1)f(1)+f(a+b+c)\\ &\qquad>f(1)+(a+b+c-2)f(2)\\ &\qquad+f(a+b+c-(a+b+c-2))\\ &\qquad=f(1)+(a+b+c-1)f(2). \end{aligned} \tag{5. V}$$

Combining the inequalities (5. V) and (5. U), we have that, for $a+b+c\geq 3$, the general cost expression from Lemma 4.1 is larger than

$$\begin{aligned} &[f(0)+(n-a-b-c-2)f(2)+f(3)]+[f(1)+(a+b+c-1)f(2)]\\ &\qquad=f(0)+f(1)+(n-3)f(2)+f(3). \end{aligned}$$

Therefore, since we also have the same inequality for $a+b+c<3$, we conclude that the cheapest cost for the wheel is $f(0)+f(1)+(n-3)f(2)+f(3)$, as claimed. □

6 Ladder Graphs, Grid Graphs, and Complete Bipartite Graphs

In this section, we consider three families of bipartite graphs: the ladder graphs, the general grid graphs, and the complete bipartite graphs.

6.1 Ladder Graphs

A ladder graph is formed by taking two paths of the same length and connecting corresponding vertices by an edge. Thus, when drawn in the plane, the ladder graph L_n looks like $n-1$ "boxes" stacked on top of each other, and has $2n$ vertices and $3n-2$ edges. Some examples of ladders are shown in Figure 1.

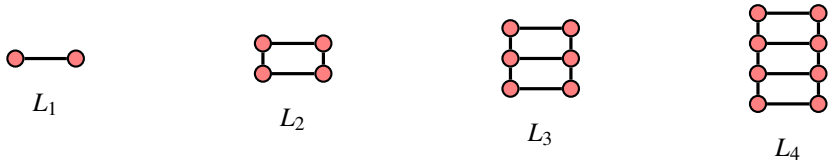


Figure 1: Small ladder graphs

Lemma 5.1. *The general cost expression for the ladder graph L_n is*

$$af(0)+bf(1)+c(2)+df(3), \qquad (6.\ \mathrm{W})$$

where a,b,c,d are nonnegative integers such that $a+b+c+d=2n$, $b+2c+3d=3n-2$, and $1\le a\le n$.

Proof. Obviously, we have $a+b+c+d=2n$ since there are $2n$ vertices. By Lemma 0.1, we get $a\cdot 0+b\cdot 1+c\cdot 2+d\cdot 3=3n-2$, i.e., $b+2c+3d=3n-2$. The cost of the first vertex built is always $f(0)$ so a is at least 1. For any graph and any order of the vertices, the vertices of cost $f(0)$ form an independent set in the graph. Since the largest independent set in L_n has size n, we get that $1\le a\le n$. □

Theorem 6. *The cheapest cost for the ladder graph L_n is $f(0)+nf(1)+(n-1)f(2)$.*

Proof. We start to build either the left or the right vertex of the top row of the ladder L_n, and its cost is $f(0)$. Then keep building a vertex that is one row lower and adjacent to the last vertex until the last row, and then build the vertex that is adjacent to the last vertex, and the cost of each is $f(1)$. We build the remaining vertices consecutively all the way up, and each costs $f(2)$. So the cost expression is $f(0)+nf(1)+(n-1)f(2)$. We claim that this expression is a cheapest cost for the ladder graph L_n. Since we have $a+b+c+d=2n$, we get $b+c+d=2n-a$. We can rewrite $\begin{cases} b+c+d=2n-a \\ b+2c+3d=3n-2 \end{cases}$ as $\begin{cases} c=n-2d+a-2 \\ b=n+d-2a+2 \end{cases}$. Plugging these in the general cost expression (6. W) for L_n, we get that the cost of any order is of the form $af(0)+(n+d-2a+2)f(1)+(n-2d+a-2)f(2)+df(3)$. By the inequality 1. A, we have $f(0)+f(3)>f(1)+f(2)$ and $f(1)+f(3)>2f(2)$.

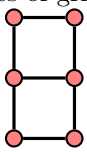
Applying these to the general cost expression, we get inequalities

$$\begin{aligned} & af(0)+(n+d-2a+2)f(1)+(n-2d+a-2)f(2)+df(3) \\ & > f(0)+(n+d-2a+2)f(1)+(n-2d+a-2)f(2)+(d-a+1)f(3) \\ & \quad +(a-1)f(1)+(a-1)f(2) \\ & > f(0)+nf(1)+(n-2d+2a-3)f(2)+2(d-a+1)f(2) \\ & = f(0)+nf(1)+(n-1)f(2). \end{aligned}$$

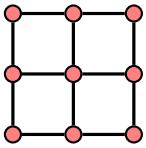
Therefore, $f(0)+nf(1)+(n-1)f(2)$ gives us the cheapest cost. □

6.2 Grid Graphs

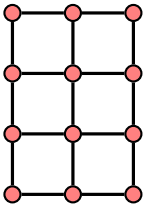
For the grid graph $G_{m\times n}$, m is the number of rows and n is the number of columns, so there are mn vertices and $(m-1)n+m(n-1)=2mn-m-n$ edges. Since the highest degree for a grid graph is 4, the largest possible input for the function f is 4. Thus, the cost expression for the $G_{m\times n}$ is $af(0)+b(1)+cf(2)+df(3)+ef(4)$, where we have $a+b+c+d+e=mn$ and $b+2c+3d+4e=2mn-m-n$. Here are some examples of grid graphs:



$G_{3\times 2}$



$G_{3\times 3}$



$G_{4\times 3}$

Theorem 7. *The cheapest cost for the grid graph $G_{m\times n}$ is $f(0)+(m+n-2)f(1)+(mn-m-n+1)f(2)$.*

Proof. First we show that this cost is achievable. If we start from the leftmost vertex of top row, and then build down consecutively until the bottom row and then build right consecutively until the last column; and we build second column from the bottom to top, and keep doing that consecutively for the remaining columns, we will get the cost expression $f(0)+(m+n-2)f(1)+(mn-m-n+1)f(2)$. It remains to show that this cost is optimal.

Solving the equations $\begin{cases} a+b+c+d+e=mn \\ b+2c+3d+4e=2mn-m-n \end{cases}$ for c and d gives

$$\begin{cases} d=2a+b-2e-m-n \\ c=mn-3a-2b+e+m+n \end{cases}. \qquad (6.\ \text{X})$$

An arbitrary cost expression for a grid graph $G_{m\times n}$ is $af(0)+bf(1)+cf(2)+df(3)+ef(4)$. Plugging 6. X into this expression, it becomes

$$af(0)+bf(1)+(mn-3a-2b+e+m+n)f(2)$$
$$+(2a+b-2e-m-n)f(3)+ef(4). \quad (6.\ \mathrm{Y})$$

We split our analysis into two cases: (i) $a+b-m-n+1\geq 0$, and (ii) $a+b-m-n+1<0$. The inequality 1. A gives $f(2)+f(4)>2f(3)$, $f(0)+f(3)>f(1)+f(2)$, and $f(1)+f(3)>2f(2)$. Thus, for the case (i), replacing $ef(2)+ef(4)$ with the cheaper $2ef(3)$ and $(a-1)f(0)+(a-1)f(3)$ with the cheaper $(a-1)f(1)+(a-1)f(2)$ in the cost expression 6. Y gives the relation

$$\begin{aligned}
&af(0)+bf(1)+(mn-3a-2b+e+m+n)f(2)\\
&\quad+(2a+b-2e-m-n)f(3)+ef(4)\\
>&f(0)+(a+b-1)f(1)+(mn-2a-2b+m+n-1)f(2)\\
&\quad+(a+b-m-n+1)f(3).
\end{aligned}$$

Since $a+b-m-n+1\geq 0$ in this case, replacing $(a+b-m-n+1)f(1)+(a+b-m-n+1)f(3)$ with the cheaper $2(a+b-m-n+1)f(2)$ in this last expression gives

$$\begin{aligned}
&f(0)+(a+b-1)f(1)+(mn-2a-2b+m+n-1)f(2)\\
&\quad+(a+b-m-n+1)f(3)\\
>&\ f(0)+(m+n-2)f(1)+(mn-2a-2b+m+n-1)f(2)\\
&\quad+2(a+b-m-n+1)f(2)\\
=&\ f(0)+(m+n-2)f(1)+(mn-m-n+1)f(2).
\end{aligned}$$

For case (ii), when $a+b-m-n+1<0$, we have $m+n-a-b-1>0$. In this case, replace $ef(2)+ef(4)$ with the cheaper $2ef(3)$ and then replace $(2a+b-m-n)f(0)+(2a+b-m-n)f(3)$ with the cheaper $(2a+b-m-n)f(1)+(2a+b-m-n)f(2)$; after that, replace $(m+n-a-b-1)f(0)+(m+n-a-b-1)f(2)$ with the cheaper $2(m+n-a-b-1)f(1)$. This gives

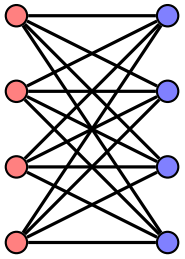
$$\begin{aligned}
&af(0)+bf(1)+(mn-3a-2b+e+m+n)f(2)+(2a+b-2e-m-n)f(3)+ef(4)\\
>&af(0)+bf(1)+(mn-3a-2b+m+n)f(2)+(2a+b-m-n)f(3)\\
>&(m+n-a-b)f(0)+(2a+2b-m-n)f(1)+(mn-a-b)f(2)\\
>&f(0)+(m+n-2)f(1)+(mn-m-n+1)f(2).
\end{aligned}$$

Since the two cases are exhaustive and the inequality holds in both cases, our claimed cost expression is optimal. □

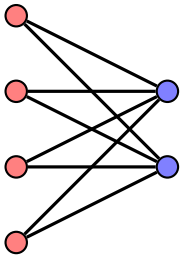
6.3 Complete Bipartite Graphs

A bipartite graph is a graph with two vertex sets, M and N, such that no vertex in M is connected to another vertex in M, and similarly no vertex in N is connected to another vertex in N. The complete bipartite graph $K_{m,n}$ is characterized by the fact that M has size m, N has size n, and every vertex in set M is connected to every vertex in set N. Thus, the degree of each vertex in M is equal to n, and similarly, the degree for each vertex in N is equal to m. Some complete bipartite graphs, denoted $K_{n,n}$, have the same number of vertices in each set. Below are a few examples.

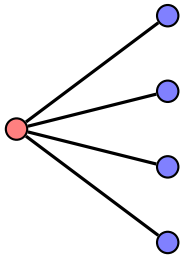
Example 7.1.



$K_{4,4}$

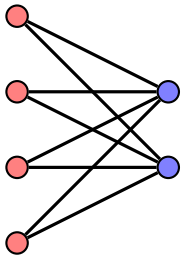


$K_{4,2}$

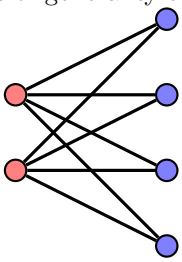


$K_{1,4}$, *or a claw*

It is important to note that $K_{n,m}$ and $K_{m,n}$ are isomorphic, as illustrated in the figure below, so that we can assume without loss of generality that $m \geq n$.



$K_{4,2}$



$K_{2,4}$

Theorem 8. *The cheapest cost of a complete bipartite graph $K_{m,n}$, where $m \geq n$, is $f(0)+2[f(1)+f(2)+f(3)+\cdots+f(n-1)]+(m-n+1)f(n)$, where m is the number of vertices in the larger set and n is the number of vertices in the smaller set.*

Proof. Given a complete bipartite graph $K_{m,n}$, consider the following special kinds of permutations of the vertices: start by building a vertex in one of the vertex sets, then the second vertex built must be from the other set. The third vertex should be from the same set as the first vertex built. Continue building this complete bipartite graph by alternating between vertex sets. Once the smaller set has been completed and all vertices in that set are built, build the remaining vertices in the larger set in any order. We will refer to this as "the zigzag way," and we represent these orders by the permutations $\{M,N,M,N,\cdots,M,N,M,M,\cdots,M\}$ and $\{N,M,N,M,\cdots,N,M,M,\cdots,M\}$, depending on which side we start on.

Our claim is that these two are the cheapest ways to build $K_{m,n}$. We observe that they both have cost $f(0)+2[f(1)+f(2)+\cdots+f(n-1)]+(m-n+1)f(n)$. We will prove that this is optimal by contradiction, starting with the assumption that there is a cheaper non-zigzag way to build $K_{m,n}$. Here are some non-zigzag examples: $\{M,M,M,N,N,S\}$, $\{N,N,M,N,M,N,M,S\}$, and $\{M,N,M,N,\cdots,M,M,M,N,M,N,S\}$, where S represents an arbitrary order of whatever vertices remain. No matter the permutation, every non-zigzag permutation belongs in one of these two cases: (i) there are two or more Ms built consecutively before all Ns have been built, or (ii) there are two or more Ns built consecutively.

We first prove that if two vertices from M are built consecutively before all vertices from N have been built, then the permutation is not optimal. For instance, take the permutation $Y_1=\{M,N,M,N,\cdots,M,N,M,M,N,S\}$, where there are two Ms built together, followed by an N. Here, S is the rest of the permutation after the change in pattern, and is arbitrary. Define a new permutation $Z_1=\{M,N,M,N,\cdots,M,N,M,N,M,S\}$ by switching the second of the two consecutive Ms with the N after it. Note here that S remains the same, thus the rest of the permutation is identical for both Y_1 and Z_1. Then, we have that the cost expression for Y_1 and Z_1 respectively are

$$\begin{aligned}\text{Cost}_{Y_1} &= f(0)+2f(1)+2f(2)+\cdots+2f(b)\\ &\quad +f(b)+f(b+2)+\text{Cost}_S.\end{aligned}$$

and

$$\begin{aligned}\text{Cost}_{Z_1} &= f(0)+2f(1)+2f(2)+\cdots+2f(b)\\ &\quad +2f(b+1)+\text{Cost}_S.\end{aligned}$$

where b is the number of Ns before the two consecutive Ms in Y_1. Since $f(b)+f(b+2)>2f(b+1)$ by the inequality 1. A, we have $\text{Cost}_{Y_1}>\text{Cost}_{Z_1}$.

More generally, suppose we have a permutation Y_k that (after some zigzag alternation) has a group of $k+1$ Ms built consecutively before the last N where $k\geq 1$. If Y_k starts with M, then we can write it as

$$Y_k=\{M,N,M,N,\cdots,M,N,M,\underbrace{M,\cdots,M}_{k},N,S\}.$$

Then define

$$Z_k=\{M,N,M,N,\cdots,M,N,M,\underbrace{M,\cdots,M}_{k-1},N,M,S\}.$$

Their cost expressions are

$$\begin{aligned}\text{Cost}_{Y_k} &= f(0)+2f(1)+2f(2)+\cdots+2f(b)\\ &\quad +[kf(b)+f(b+k+1)]+\text{Cost}_S.\end{aligned}$$

and

$$\begin{aligned}\text{Cost}_{Y_k} &= f(0)+2f(1)+2f(2)+\cdots+2f(b)\\ &\quad +[kf(b)+f(b+k+1)]+\text{Cost}_S.\end{aligned}$$

Since $f(b)+f(b+k+1)>f(b+1)+f(b+k)$ by inequality 1. A, we have that $kf(b)+f(b+k+1)>(k-1)f(b)+f(b+1)+f(b+k)$, so that $\text{Cost}_{Y_k}>\text{Cost}_{Z_k}$, and therefore Y_k is not optimal. The case that Y_k begins with N is very similar,

and we omit the details.
Now we prove that if two or more vertices from N are built consecutively, then the permutation is not optimal. We choose some such permutation

$$\overline{Y}_k = \{M,N,M,N,\cdots,M,N,\underbrace{N,\cdots,N}_{k},M,S\},$$

and its cost expression is

$$\begin{aligned}\text{Cost}_{\overline{Y}_k} &= f(0)+2f(1)+2f(2)+\cdots+2f(b)\\ &\quad +f(b+1)+kf(b+1)+f(b+k+1)+\text{Cost}_S.\end{aligned}$$

Then define the permutation

$$\overline{Z}_k = \{M,N,M,N,\cdots,M,N,\underbrace{N,\cdots,N}_{k-1},M,N,S\},$$

whose cost expression is

$$\begin{aligned}\text{Cost}_{\overline{Z}_k} &= f(0)+2f(1)+2f(2)+\cdots+2f(b)\\ &\quad +f(b+1)+(k-1)f(b+1)+f(b+k)+f(b+2)+\text{Cost}_S.\end{aligned}$$

Since $f(b+1)+f(b+k+1) > f(b+2)+f(b+k)$, we get $\text{Cost}_{\overline{Y}_k} > \text{Cost}_{\overline{Z}_k}$. The case that $\overline{Y}_k$ begins with N is again similar.
Now let Y be an optimal permutation. If Y is not a zigzag way, then we have shown that there is another permutation Z which costs less than Y, a contradiction. Therefore, the zigzag way is the cheapest permutation to build the complete bipartite graph $K_{m,n}$. □

7 Cost Analysis for Graphs on Five Vertices and Six Vertices

Now that we have analyzed various families of graphs to find the cheapest cost, we can extend this paper to graphs that are not necessarily categorized into a family, such as bipartite graphs or trees. Instead, we can look at all possible connected graphs on five and six vertices, as well as the Petersen graph. As discussed in Section 2, we know that there are 21 connected graphs on five vertices and 112 connected graphs on six vertices. To explore these graphs, we used an algorithm to compute the optimal degree sequence with the cheapest cost of each graph on five and six vertices. We chose two positive, decreasing, convex functions to analyze: $f(x) = (1+x)^{-1}$ and $g(x) = 2^{-x}$.
Using the programming language Python and the package NetworkX, we first found all possible permutations of the vertices for each graph on five or six vertices. Then, we recorded the degree of every vertex in each permutation at the time is was built. Finally, we collected this data and were able to compute the costs for each permutation, outputting an array with the cheapest cost and its optimal degree sequence. The code can be found below in Appendix A.
In this experiment, we observed that for all connected graphs on five and six vertices, the cheapest cost was achieved with the same orders across both cost functions. We see this by following the degreeswhenbuilt variable, which records what orders the vertices must be built to attain the cheapest cost. Each graph

had the same orders for degreeswhenbuilt, illustrating that for 5-vertex and 6-vertex graphs, the orders for their respective cheapest costs are the same. The optimal degree sequences for finding the cheapest cost for each cost function on five-vertex graphs are given in Table 1. Finally, we tested the Petersen graph which has ten vertices, to see if it would present us with the same conclusion. We found that for the Petersen graph, it was again true that the orders for the cheapest costs using functions f and g were the same. This code can be used in principle to see if the same conclusion holds for higher order graphs. Our limitations included computer processing time, since it was able to quickly give back data on five and six vertices, but not on higher order graphs.

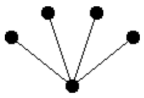
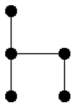
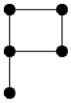
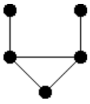
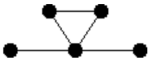
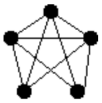
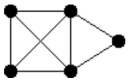
8 Can we go further?/Future expectation

In every case we studied above, the optimal cost is achieved for a permutation in which we never build a vertex that creates a new component. Thus, we conjecture that, for any graph, to get the minimum cost, the optimal permutation must be a connected order, i.e., we never start a new component. There are a lot of families of simple connected graphs other than those considered in this paper, and there are also graphs that are not simple connected. Thus, there are many ways to extend this project and further discover more about cost effectively building graphs. Here are some examples that seem particularly interesting.

Question 8.1. *What are the optimal orders for building hypercube graphs?*

Question 8.2. *How does the question change for graphs that are not simple (i.e., with loops or multiple edges allowed)? What about directed graphs (where the cost to build v is $f(k)$ where k is the number of vertices w such that $w \to v$ is a directed edge in the graph)?*

It might also be interesting to study whether finding optimal costs is easier in special families of graphs, like Eulerian or Hamiltonian graphs.

Graph	Optimal degrees
	0, 1, 1, 1, 1
	0, 1, 1, 1, 1
	0, 1, 1, 1, 1
	0, 1, 1, 1, 2
	0, 1, 1, 1, 2
	0, 1, 1, 1, 2
	0, 1, 1, 1, 2
	0, 1, 2, 3, 4
	0, 1, 2, 3, 3
	0, 1, 2, 2, 3
	0, 1, 2, 2, 3

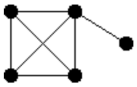
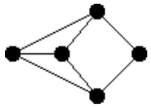
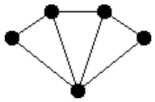
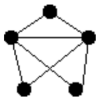
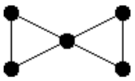
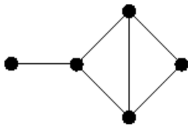
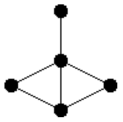
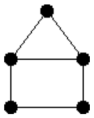
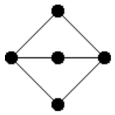
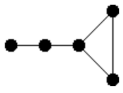
Graph	Optimal degrees
	0, 1, 1, 2, 3
	0, 1, 2, 2, 2
	0, 1, 2, 2, 2
	0, 1, 2, 2, 2
	0, 1, 1, 2, 2
	0, 1, 1, 2, 2
	0, 1, 1, 2, 2
	0, 1, 1, 2, 2
	0, 1, 1, 2, 2
	0, 1, 1, 1, 2

Table 1: This table shows every isomorphism class of connected graphs on 5 vertices and the cheapest order to build them.

9 Acknowledgements

This project was carried out as an undegraduate research project at —George Washington University, under the mentorship of Professor —Joel Lewis . We are grateful to an anonymous referee for detailed comments that improved the paper.

Bibliography

[1] Novikoff, Timothy. *Algorithmic Education Theory.* Ph.D. thesis, Cornell University, 2013. https://ecommons.cornell.edu/handle/1813/33956

[2] Bonin, Joseph. *Notes for Math 3632 Introduction to Graph Theory.* April 12 2021.

[3] Maclagan, Diane, *Connected simple graphs on four vertices*, Lecture notes, University of Warwick, 2005. https://homepages.warwick.ac.uk/~masgar/Teach/2005_428/2005_09_12lecture_paths.pdf

[4] Sloane, N. J. A. *Number of connected graphs with n nodes.* The On-Line Encyclopedia of Integer Sequences. August 15 2023https://oeis.org/A001349

0 APPENDIX A. Python code

Below includes code specifically for star graphs, random trees, and complete bipartite graphs. Each of them need to be commented out individually to see the code executed. All connected graphs on five and six vertices need to be built according to NetworkX commands found here https://networkx.org/documentation/stable/tutorial.html.

```
import networkx as nx
import pandas as pd
from math import *
import numpy as np

# code for stars
#v = 4
#G = nx.star_graph(v)

# code for random trees
#v = 4
#G = nx.random_tree(v)

# code for complete bipartite graphs
#n_1 = 4
#n_2 = 2
#nodes = list(range(0,n_1+n_2))
#G = nx.complete_bipartite_graph(n_1,n_2,create_using=None)
```

```
nodes = list(range(0,v))
nx.draw(G, with_labels=True)


# Finds all possible orders of building a graph with v
vertices.
Source: https://www.geeksforgeeks.org/generate-all-the-
permutation-of-a-list-in-python/

def permutation(lst):
    if len(lst) == 0:
        return []
    if len(lst) == 1:
        return [lst]
    l = []
    for i in range(len(lst)):
        m = lst[i]
        remLst = lst[:i] + lst[i+1:]
        for p in permutation(remLst):
            l.append([m] + p)
    return l

# Takes each permutation and computes the cost using
two different functions
(costreciprocal and costexponential). Records the
degree of each vertex
(degreeswhenbuilt) at the time it is built and puts
both the costs and the
degreeswhenbuilt into their respective arrays.

data = list(G.nodes)
float_list_Rec = []
float_list_Ex = []
array_recip=[]
array_exp=[]
for p in permutation(data):
    costreciprocal = 0
    costexponential = 0
    degreeswhenbuilt = []
    dictRec = {"A":[],"B":[]}
    for x in nodes:
        H = G.subgraph([p[i] for i in range(0, x+1)])
        d = list(H.degree([p[x]]))[0][1]
        degreeswhenbuilt.append(d)
        costreciprocal += 1/(d+1)
        costexponential += 2**(-d)
```

```
        if x==v-1:
            if [costreciprocal, str(degreeswhenbuilt)]

            not in array_recip:
                array_recip.append([costreciprocal,

                str(degreeswhenbuilt)])
            if [costexponential, str(degreeswhenbuilt)]
            not in array_exp:
                array_exp.append([costexponential,

                str(degreeswhenbuilt)])
            float_list_Rec.append(costreciprocal)
            float_list_Ex.append(costexponential)

# Sorts vales in ascending order.

array_recip.sort()
array_exp.sort()

print('The minimum cost using the reciprocal cost function is ',
      min(float_list_Rec), ', and here is the list of degree
      sequences and costs:')
print(array_recip)
print()
print('The minimum cost using the exponential cost function is ',
      min(float_list_Ex), ', and here is the list of degree
      sequences and costs:')
print(array_exp)
```