Farey Sequence Sums

Jinxian Lu

Sebastian Dabrowski

Vivienne Zhu

Computer Science Department, Pace University
1 Pace Plaza, New York, NY 10038 USA
sl92982n@pace.edu
sd84470n@pace.edu
xz15682n@pace.edu

Abstract. This article considers the problem of finding the Farey sum of order *N*, which is a positive fraction. Our work is to explore the properties of the Farey series and the Fareys sum. We will compare and contrast the similarities and differences between Farey series and the Stern-Brocot Tree. Finally, we will show an application of the algorithms for evaluating the Farey sum.

1 Introduction

1.1 Farey Series

In the year 1816, John Farey, a geologist and a musicologist, invented the *Farey Series* [1]. The Farey series is an increasing sequence of order *N*. Farey fractions *a/b* are arranged in order of size from 0 to 1 and must be irreducible. The denominator *b* must also not exceed *N*. For example, the Farey series of order $N = 1$ to $N = 6$ are defined as the following:

Table 1. Examples of Farey Series of order *N*

Order	Farey Series	Count
1	$\frac{0}{1}\ \ \frac{1}{1}$	2
2	$\frac{0}{1}\ \ \frac{1}{2}\ \ \frac{1}{1}$	3
3	$\frac{0}{1}\ \ \frac{1}{3}\ \ \frac{1}{2}\ \ \frac{2}{3}\ \ \frac{1}{1}$	5
4	$\frac{0}{1}\ \ \frac{1}{4}\ \frac{1}{3}\ \ \frac{1}{2}\ \ \frac{2}{3}\ \frac{3}{4}\ \ \frac{1}{1}$	7
5	$\frac{0}{1}\ \ \frac{1}{5}\ \frac{1}{4}\ \frac{1}{3}\ \frac{2}{5}\ \frac{1}{2}\ \frac{3}{5}\ \frac{2}{3}\ \frac{3}{4}\ \frac{4}{5}\ \ \frac{1}{1}$	11

6	0 1 1 1 1 2 1 3 2 3 4 5 1 1 6 5 4 3 5 2 5 3 4 5 6 1	13

Theorem 1

In any Farey series, if *a/b* and *a'/b'* are two adjacent terms in which *a/b* < *a'/b'*, then *a'b* - *ab'* = *1*. For instance, in the Farey sequence of order 6, 1/3 and 2/5 are two adjacent terms with 1/3 < 2/5. Therefore 2 * 3 - 1* 5 = 6 - 5 = 1 [2].

Theorem 2

A Farey series contains complementary fractions [3]. If a/b is in a Farey series of order N, then a/b + (b-a)/b = 1. 1/2 is the only fraction in the series without a complement because 1/2's complement is itself. Therefore, most Farey series have an odd number of fractions. For instance, in the Farey sequence of order 6, 0/1 and 1/1, 1/6 and 5/6, 1/5 and 4/5, 1/4 and 3/4, 1/3 and 2/3, 2/5 and 3/5 are complementary fractions respectively. There are thirteen Farey fractions in total. The Farey series of order 1 is an exception. Since 1/2 is not including in the series, the number of fractions in the series is two, which is an even number.

Theorem 3

In a Farey series, between fractions a/b and a'/b' lies the fraction *(a+a') / (b+b')* that is formed by summing the numerators and denominators [1]:

$$a/b < (a+a') / (b+b') < a'/b'$$

For instance, when $N = 3$, the Farey series is 0/1, 1/3, 1/2, 2/3, 1/1. If $a/b = 1/3$ and $a'/b' = 2/3$, then we can generate a new fraction between 1/3 and 2/3 : (1+2) / (3+3) = 3/6 = 1/2 which is larger than 1/3 but smaller than 2/3. Again, if $a/b = 1/2$ and $a'/b' = 1/1$, then we can generate a new fraction between 1/2 and 1/1 : (1+1) / (2+1) = 2/3 which is larger than 1/2 but smaller than 1/1. In short, for any 3 successive fractions, the middle one is the mediant of the other 2 but not always in its simplest form.

1.2 Farey Sum

If the denominators of the Farey series of order N are:

$$d[1], d[2], \ldots , d[K]$$

then the Farey sum of order N is the sum of $d[i] / d[i+1]$ from $i = 1$ to $K - 1$.

In Table 2, the Farey sums of order $N = 1$ to 6 are listed,

Table 2. Examples of the Farey sum of order *N*

Order	Farey Sum
1	1/1
2	1/2 + 2/1 =5/2
3	1/3 + 3/2 + 2/3 + 3/1 = 11/2
4	1/4 + 4/3 +3/2 + 2/3 + 3/4 + 4/1 = 17/2
5	1/5 + 5/4 + 4/3 + 3/5 + 5/2 + 2/5 + 5/3 + 3/4 + 4/5 + 5/1 = 29/2

6	1/6 + 6/5 + 5/4 + 4/3 + 3/5 + 5/2 + 2/5 + 5/3 + 3/4 + 4/5 + 5/6 + 6/1 = 35/2

As stated in Table 2, the number of the terms *d[i] / d[i+1]* in the Farey sum of order *N* is one less than the number of the terms in the Farey series of order N accordingly. Therefore, excluding *N* = 1, there is an even number of the terms *d[i] / d[i+1]* in the Farey sum of order *N*. Moreover, half of such terms are the reciprocal of the other half of the terms. For instance, when *N* = 3, the terms *d[i] / d[i+1]* are 1/3, 3/2, 2/3, and 3/1. The reciprocal of 1/3 is 3/1 and the reciprocal of 3/2 is 2/3.

2 Stern–Brocot tree

The Farey series has close relationship with the Stern-Brocot tree since they both have the same property that is discussed in **Theorem 3** (define the mediant of the two rational numbers).

The operation of forming the mediant of two rational numbers corresponds to vector addition [4].

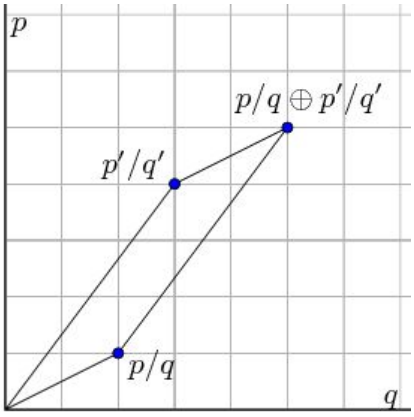
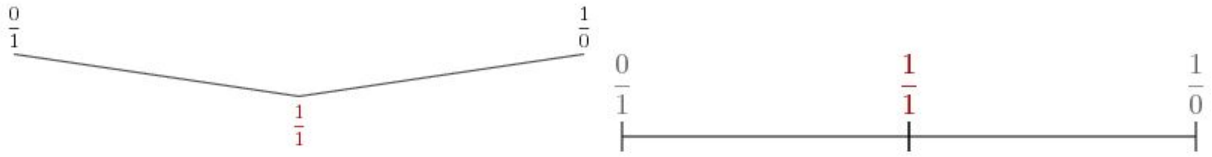
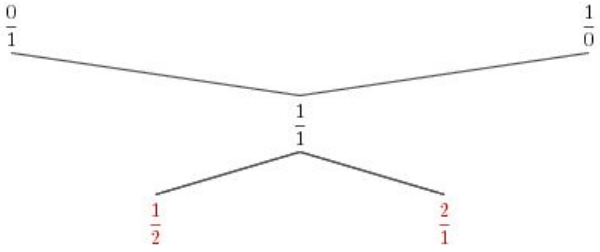


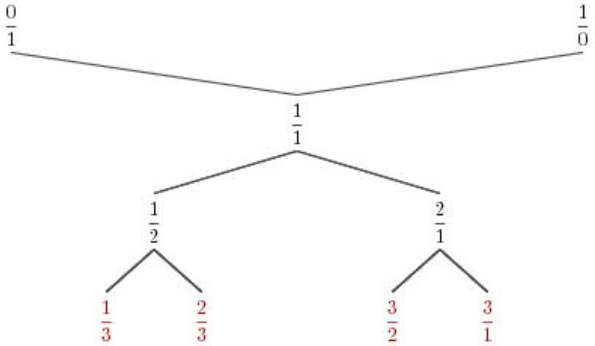
Figure 1. Slope of the mediant

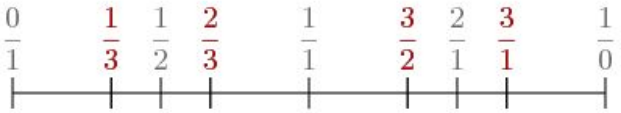
This shows that the slope of the mediant is between the slopes defined by the two original rational numbers p/q and p'/q'. Now, we can construct the Stern-Brocot tree step by step beginning with the two fractions 0/1 and 1/0. To continue the construction, we insert the mediant of any two consecutive rational numbers in the tree.

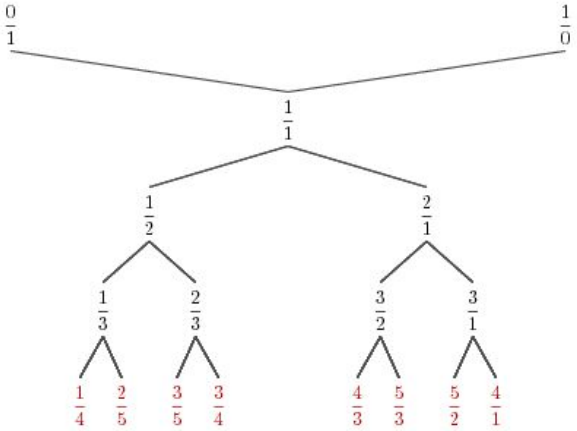



0/1
1/0
1/1
1/2
2/1


0/1
1/2
1/1
2/1
1/0


0/1
1/0
1/1
1/2
2/1
1/3
2/3
3/2
3/1


0/1
1/3
1/2
2/3
1/1
3/2
2/1
3/1
1/0


0/1
1/0
1/1
1/2
2/1
1/3
2/3
3/2
3/1
1/4
2/5
3/5
3/4
4/3
5/3
5/2
4/1


0/1
1/4
1/3
2/5
1/2
3/5
2/3
3/4
1/1
4/3
3/2
5/3
2/1
5/2
3/1
4/1
1/0

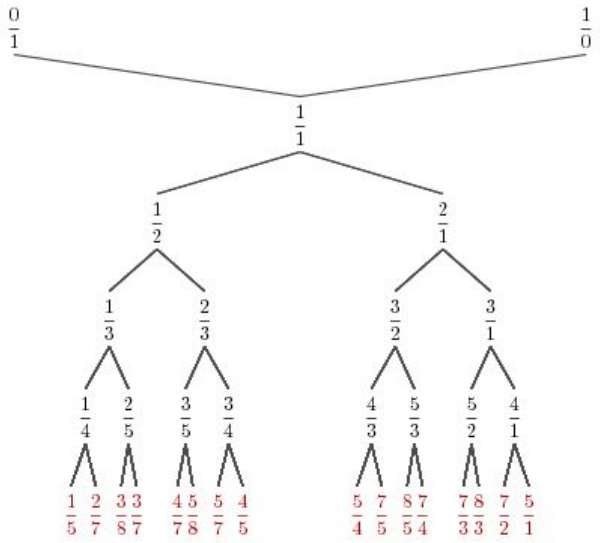


Figure 2. Construction of the Stern-Brocot tree

As shown in Figure 2, the fractions in the Farey series of order *N* from 1 to 5 are included in the Stern-Brocot tree. Since the Farey series only generates fractions from 0 to 1, instead of beginning with 0/1 and 1/0, we begin with 0/1 and 1/1 to construct the tree. So the right half of the Stern-Brocot tree will be cancelled. See Figure 3.

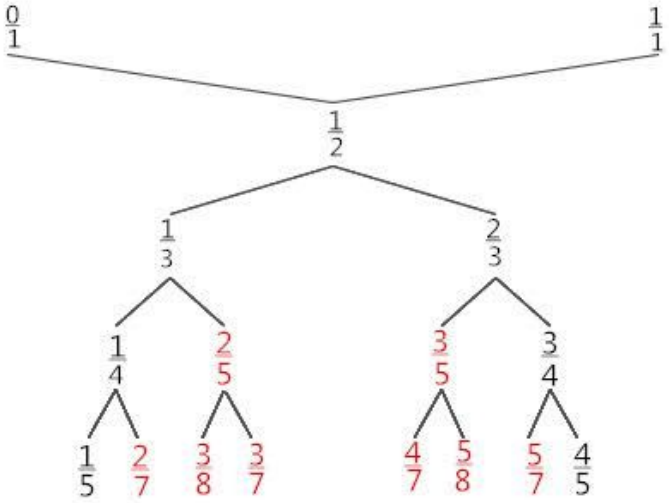


Figure 3. The Farey series in the Stern-Brocot tree

In Figure 3, all the fractions from the 1st to the nth rows are in the Farey series of order *N*. However, because the denominators in the Farey series of order *N* do not exceed *N*, the fractions in red will not be counted in the Farey series. For instance, all the fractions from the 1st to the 4th rows are in the Farey series of order 4 except 2/5 and 3/5 because the denominator 5 is larger than 4. Again, all the fractions from the 1st to the 5th rows are in the Farey series of order 5 except 2/7, 3/8, 3/7, 4/7, 5/8, and 5/7 because the denominators 7 and 8 are larger than 5.

3 Finding the Farey Sum

3.1 1st Algorithm - Farey sequence

In order to find the Farey Sum, one needs to use the Farey sequence. For our first attempt at a solution, our group attempted building the entire Farey sequence using strong induction and theorem 3 of the Farey series, which states that in a Farey series, between fractions a/b and a'/b' lies the fraction *(a+a') / (b+b')* which is formed by adding the numerators and denominators.

Table 3. The 5th and 6th Farey Sequence

5	$\frac{0}{1}\ \frac{1}{5}\ \frac{1}{4}\ \frac{1}{3}\ \frac{2}{5}\ \frac{1}{2}\ \frac{3}{5}\ \frac{2}{3}\ \frac{3}{4}\ \frac{4}{5}\ \frac{1}{1}$
6	$\frac{0}{1}\ \frac{1}{6}\ \frac{1}{5}\ \frac{1}{4}\ \frac{1}{3}\ \frac{2}{5}\ \frac{1}{2}\ \frac{3}{5}\ \frac{2}{3}\ \frac{3}{4}\ \frac{4}{5}\ \frac{5}{6}\ \frac{1}{1}$

In the case of strong induction one must ask oneself the question of how to find the solution for *N*+1 when given *N*. In Table 3, one can see the 5th and 6th Farey sequences. In this example, the 5th Farey sequence would be considered N while the 6th Farey sequence would be *N*+1. In order to create the 6th sequence, *N*+1, one can go through the 5th sequence, *N*, and look for consecutive fractions with denominators which add up to *N*+1. If the sum of the denominators equals *N*+1, one can then add the numerators and create and insert a new fraction between the original two fractions. This new fraction would have the numerator sum as its numerator and the denominator sum as its denominator. Afterwards, the search for consecutive fractions with denominators adding up to *N*+1 can be continued until the end of the sequence is reached.

So taking our example with the 5th and 6th Farey sequence step by step, in order to find the 6th sequence we would start in the 5th sequence at the fraction 0/1. The fraction following 0/1 is 1/5. We add the denominators, 1 and 5, and get 6, which is what we are looking for. We then add the numerators 0 and 1, get 1, and make a new fraction 1/6. We insert this fraction between 0/1 and 1/5 and move on to the next fraction in the sequence, 1/5. The fraction following 1/5 is 1/4. The sum of the two fractions denominators is 9 which is greater than 6. Therefore we do not insert anything and instead we move on to 1/4. In the case of the 5th Farey sequence we will continue skipping fractions in the sequence until the fraction 4/5 is reached. The fraction after 4/5 is 1/1 and again 5 and 1 equal 6 meaning that we have to insert a new fraction. We add up the numerators and create 5/6, which is then inserted between 4/5 and 1/1. When we finally go on to 1/1, we know that we have reached the end of the sequence and are done.

The reason this works is because of the property of the Farey Sequence described in theorem 3 and the fact that the only difference between a Farey sequence of order *N* and a Farey sequence of order *N*+1 is the inclusion of fractions with the denominator *N*+1 in the sequence of order *N*+1. Looking again at the sequences in Figure 4, one can see that only two new fractions are included in the 6th Farey sequence, 1/6 and 5/6. Both have denominators of 6, which is the same as the order of the sequence.

With a guaranteed way of finding *N+1* by using *N*, all that is left in order to do strong induction is a base case and a method of storing the sequence. The base case for a Farey sequence is a sequence of order 1. The only two fractions included in this order are 0/1 and 1/1. As for a method of storing the sequence, our group chose to use a linked list over an array. This is because there is a lot of insertions in the middle of the sequence when using our algorithm. In an array, insertions in the middle take linear time to complete. In a linked list, these same operations are constant. The main disadvantage of a linked list, the fact that it takes linear time to search for an element, also conveniently does not apply to us in this case because we are already going through the entire list in order to build the next sequence. Overall this means that by using a linked list instead of an array, we are being much more time efficient than we would otherwise be.

```
public static FareyNode fareySeq(int n) // Takes in a parameter n which is the order of the Farey Sequence to find.
{
    FareyNode last = new FareyNode(1, 1);
    FareyNode first = new FareyNode(last, 0, 1); //These two nodes form the base case
    FareyNode current = first;
    for (int i = 2; i <= n; i++) //The algorithm is looped through until n is finally reached
    {
        while (current.next != null) //Loops through the entire sequence to check where new fractions should be added
        {
            if (current.getDenom() + current.getNext().getDenom() == i) //Checking if the sum of the
            {                                                            //denominators equals the order number
                int num = current.getNum() + current.getNext().getNum();
                int denom = current.getDenom() + current.getNext().getDenom();
                FareyNode temp = new FareyNode(current.getNext(), num, denom); //Creating and inserting the
                current.setNext(temp);                                         //new fraction into the list
                current = current.getNext();
            }
            current = current.getNext(); // Moving on to the next fraction in the list
        }
        current = first; //resetting the linked list pointer so that we can repeat the process for the next order
    }
    return first; //Returns the beginning of the linked list
}
```

Figure 4. The first algorithm's implementation of finding the Farey sequence. In this implementation, FareyNodes are nodes in a linked list. They store a fraction in the sequence as well the address of the next node in line.

3.2 The 1st Algorithm - Farey Sum

For the first algorithm, once the Farey sequence is found and stored, it can be used to find the fractions in the Farey Sum. This is done by going through the Farey sequence and storing the denominator of the current fraction as well as the denominator of the next fraction in the sequence. We can then use these two denominators to create one of the fractions in the Farey Sum. The first denominator will become the Farey Sum fraction's numerator, while the second denominator will become its denominator. This process is repeated until the end of the sequence is reached at which point all the fractions in the Farey Sum will have been created.

Table 4. The 6th Farey Sequence

6	$\frac{0}{1}\ \frac{1}{6}\ \frac{1}{5}\ \frac{1}{4}\ \frac{1}{3}\ \frac{2}{5}\ \frac{1}{2}\ \frac{3}{5}\ \frac{2}{3}\ \frac{3}{4}\ \frac{4}{5}\ \frac{5}{6}\ \frac{1}{1}$

Looking again at the 6th Farey Sequence in Table 4, we can show the process step by step. The first step will be to start at the beginning at 0/1 and take both the denominator of 0/1 and the denominator of the next fraction in line, 1/6. The 1 in 0/1 becomes the new fraction's numerator while the 6 in 1/6 becomes the new fraction's denominator. From this process we now know that the first fraction in the Farey Sum is 1/6. We can then move on to the next fraction in line, 1/6, taking both it and its neighbor's denominator. So we will take the 6 from 1/6 and the 5 from 1/5 and form the next fraction in the Farey Sum, 6/5. This will continue until we reached the end of the Farey Sequence and have all the fractions in the Farey Sum, shown in Table 5. Once we have all the fractions in the Farey Sum, it is just a matter of adding all of them up in order to find the actual Farey Sum.

Table 5. The fractions included in the 6th Farey Sum.

6	$\frac{1}{6}\ \frac{5}{4}\ \frac{4}{3}\ \frac{3}{5}\ \frac{5}{2}\ \frac{2}{5}\ \frac{5}{3}\ \frac{3}{4}\ \frac{4}{5}\ \frac{5}{6}$

(Note that there is one less fraction in the Farey Sum than in the Farey Sequence.)

In our group's implementation of the algorithm for finding the Farey Sum, we add up all the fractions in the Farey Sum as decimals. We then convert this decimal to its fractional form. This is simple because of the fact that all Farey Sums above order 2 have 2 in the denominator. Therefore we can just double the sum, round it, and put it as the numerator in a fraction with a denominator of 2.

```
public static String fareySum(int n)//The parameter n represents the order
{                                   //of the Farey Sum that is requested
    FareyNode seq = fareySeq(n);
    double sum = 0;
    while(seq.getNext() != null)//Go through entire Farey sequence
    {
        double value = (double)seq.getDenom()/seq.getNext().getDenom();
        sum += value;       // Create Farey sum fraction and add it to the total
        seq = seq.getNext();
    }
    sum = sum * 2; //Converting the sum to it's fraction form
    sum += 0.5;
    int numerator = (int)sum;
    String fraction = numerator + "/2";
    return fraction; //returning the fraction as a string
}
```

Figure 5. The first algorithm's implementation of finding the Farey Sum

3.3 The 1st Algorithm -Time Complexity

With all the above in mind, we can now calculate the time complexity of the first algorithm. In order to find the Farey sequence of order n in this algorithm, we have to start at order 1 and loop through the entire list of fractions to create order 2. We have to repeat this process n-1 times in order to reach order n of the Farey Sequence. Once we have the Farey Sequence, we only have to go through the sequence one more time in order to create the fractions in the Farey Sum and add them to the total. From this we can conclude that the total time complexity of algorithm 1 would be:

$$\left(\sum_{1}^{n} f(x)\right) + f(n)$$

Here f(x) represents the length of the Farey Sequence at any order x while n represents the order of the sequence that is necessary for the answer. Thus, this expression is stating that the total time complexity of algorithm 1 is the sum of all the lengths of the the Farey sequences from 1 to n, plus the length of the final Farey Sequence.

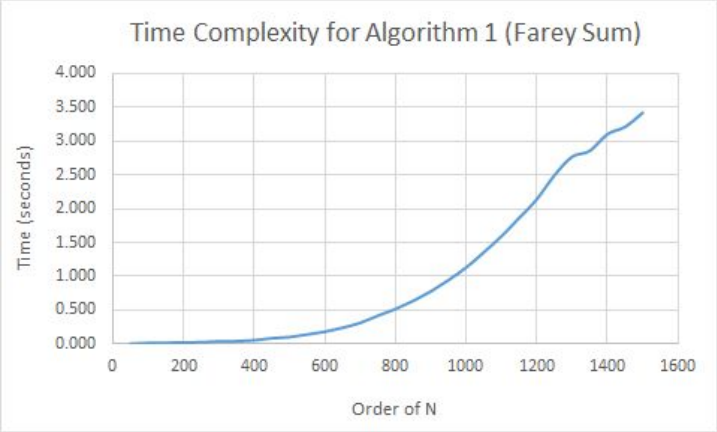


Figure 6. Graph of time taken to find solution of order N in algorithm 1

While this algorithm works and finds the correct Farey Sum, it is not very time efficient as can be seen in Figure 6. At around N = 1000, the time taken to find the solution is over a second and continues increasing exponentially from there. It take a huge amount of time to find the Farey Sum of a large N, and so our second algorithm attempts to improve on this.

3.4 The 2nd Algorithm - Farey Sum

In our second algorithm, our focus was on improving the first algorithm as much as possible while still using the same strong inductive method present in the first algorithm. To do this we attempted to divide and conquer by using a property of the fractions in the Farey Sum.

Table 6. The fractions in the 6th Farey Sum

6	$\frac{1}{6}\ \frac{6}{5}\ \frac{5}{4}\ \frac{4}{3}\ \frac{3}{5}\ \frac{5}{2}\ \frac{2}{5}\ \frac{5}{3}\ \frac{3}{4}\ \frac{4}{5}\ \frac{5}{6}\ \frac{6}{1}$

Looking at Table 6, one can once again see the fractions included in the Farey Sum of order 6. If one cuts this list of fractions in half, the right side contains fractions which are

reciprocals of the fractions on the left side. This property is present in the Farey Sums of every order greater than or equal to 2. Because of this property we only have to find half of the fractions in the Farey Sum in order to know the full list. That in turn means we only have to find half of the Farey Sequence because we only need the denominators of those fractions to find the Farey Sum. This effectively halves the number of elements we need to find and insert into our list.

```
public static FareyNode fareySeq(int n)
{
    FareyNode last = new FareyNode(2);
    FareyNode first = new FareyNode(last, 1);
    FareyNode current = first;
    for (int i = 3; i <= n; i++)
    {
        while (current.next != null)
        {
            if (current.getDenom() + current.getNext().getDenom() == i)
            {
                int denom = current.getDenom() + current.getNext().getDenom();
                FareyNode temp = new FareyNode(current.getNext(), denom);
                current.setNext(temp);
                current = current.getNext();
            }
            current = current.getNext();
        }
        current = first;
    }
    return first;
}
public static String fareySum(int n)
{
    FareyNode seq = fareySeq(n);
    double sum = 0;
    while(seq.getNext() != null)
    {
        int num = seq.getDenom();
        int denom = seq.getNext().getDenom();
        //Add the Farey Sum and it's reciprocal together
        //This compensates for the simpler version of the Farey sequence returned in FareySums2
        double value = (double)num/denom;
        sum += value;
        value = (double)denom/num;
        sum += value;
        seq = seq.getNext();
    }
    //Round the sum and display it as the fraction
    sum = sum * 2;
    sum += 0.5;
    int numerator = (int)sum;
    String fraction = numerator + "/2";
    return fraction;
}
```

Figure 7. Implementation of algorithm 2 to find the Farey Sum of order n.

By analysing the code in Figure 7, one can see that while the code for algorithm 2 is similar to the code for algorithm 1, it nevertheless contains various important changes. The function responsible for computing the Farey sequence of order n now only stores the denominators of the fractions in the linked list. This is because the numerator is unimportant when finding the fractions in the Farey Sum and therefore can be left out. Another important change to the Farey sequence function is the change in the base case used. Instead of a base case of 0/1 and 1/1, the base case is now 0/1 and 1/2. This is because we only need to find the fractions between 0/1 and 1/2 to find the first half of the Farey sequence. The function responsible for finding the Farey Sum compensates for this halving of the Farey Sequence by

immediately finding the reciprocal of any Farey Sum fraction it derives from the Farey Sequence. It then takes the reciprocal and adds it along with the regular fraction to the sum. With the reciprocals accounted for, the second half of the Farey sequence is unnecessary.

3.5 The 2nd Algorithm-Time Complexity

By halving the number of elements we need to find in both the Farey Sequence and the Farey Sum, we effectively cut the time complexity of the whole algorithm in half. This makes the time complexity of algorithm 2 equal to:

$$\frac{(\sum_{1}^{n} f(x)) + f(n)}{2}$$

The reason that the time complexity is cut in half through our divide and conquer method is because our time complexity is almost entirely dependent on the length of the Farey sequence, represented in our time complexity by f(x). Since we can always find the other half of the Farey Sum, we only have to find and store half of both it and the Farey sequence. This leads to only having to loop through half the normal amount of nodes in the linked list, speeding things up significantly.

The change in time complexity can be confirmed by looking at the graph in Figure 8. At all points along the graph the measured runtime of the program is approximately half of what it is in Figure 6. However, the time taken overall is still exponential. This means that while the algorithm is twice as efficient as algorithm 1, it still is not very time efficient when *N* starts to get into the thousands. This means that algorithms using strong induction are not an effective way of computing the Farey Sum. As a result, our group had to find another algorithm.

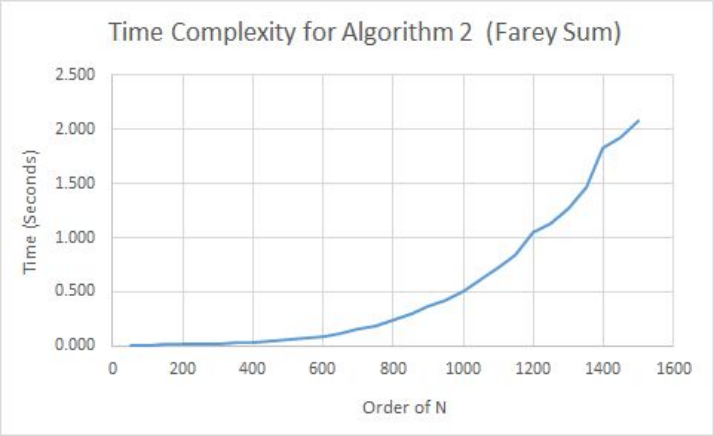


Figure 8. Graph of time taken to find solution of order N in algorithm 2.

3.6 The 3rd Algorithm

With the strong inductive algorithm not working out, our group was stuck for a while. As far as we knew at the time, the only way we knew to find the Farey Sum at the time was to use the Farey sequence denominators to build it. However, algorithm 2 used that method and proved that it was too slow. Eventually we peeked at the judges solution to Farey Sum problem on the

ACM programming contest website and in there found what we were missing. In the comments we found a property of the Farey Sum which made it possible to not have to inefficiently build a giant list of Farey Sum fractions. That property is that the Farey Sum increases by 3/2 for every fraction in the Farey Sequence that is not part of order 1. This property alone allows one to calculate the Farey Sum using a simple equation:

$$FareySum = 1 + (3/2 * (FareySeqenceLength - 2))$$

Using this equation, all we have to do to find the Farey Sum is figure out the Farey sequence length at any order n and we can find the Farey Sum by plugging in the value. In order to find the Farey sequence length, we have to know what exactly makes a sequence the Farey sequence. Well, the Farey sequence is at its core a list of all the irreducible fractions between 0 and 1 which use only numbers less than the order number of the sequence in their numerators and denominators. Because the fractions are irreducible, their numerators and denominators must be relatively prime to each other. Using this definition, we can come up with an algorithm to find the length. This involves looping through every number combination between 1 and the order number n. If a combination of numbers is relatively prime to each other (meaning that their gcd is 1), that combination of numbers would form one fraction in the Farey sequence of order n. Once we have counted the number of unique relatively prime number pairs, that number is our Farey sequence length.

```
public static int findSeqLength(int n) //Finds the length of a Farey Sequence of order n
{
    int length = 2; //Base case Farey Sequence of order 1
    for (int i = 2; i <= n; i++)
    {
        for (int j = 1; j < i; j++) //Finding all the number combinations that
        {                           //are relatively prime between 1 and n
            if (gcd(j, i) == 1)
                length++;
        }
    }
    return length;
}
public static String findFareySum(int n)
{
    int length = findSeqLength(n) - 2;
    double sum = 1 + (1.5 * length);
    int sSum = (int)(sum *2);
    return sSum + "/2";
}
```

Figure 9. Implementation of Algorithm 3 to find the Farey Sum of order n.

3.7 The 3rd Algorithm-Time Complexity

The implementation of the 3rd algorithm that we used is very simple. In order to find the Farey sequence length, we use two loops which end up having the time complexity of:

$$\sum_{i=1}^{n} i$$

This summation simplifies into the equation [n(n+1)]/2 which for time complexity then simplifies down to just Θ(n^2), where n is the number of times that the loop is done. Because the function for finding the Farey Sum just plugs in a number into an equation, we can say that its time complexity is constant or O(1). Because n^2 time is much larger than constant time, we ignore the constant time and just say that algorithm 3 takes Θ(n^2) time. All together this makes algorithm 3 the most time efficient of the algorithms by far.

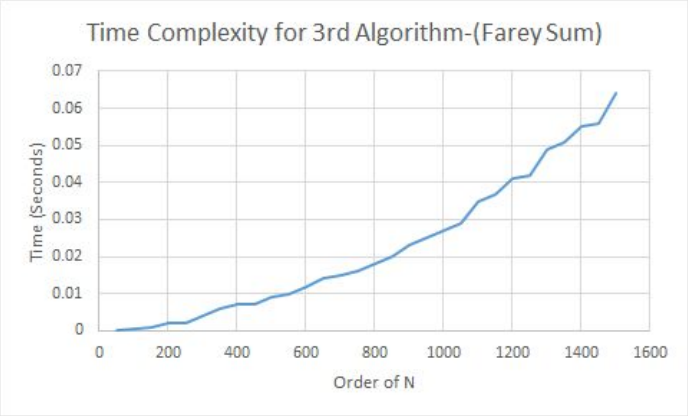


Figure 10. Graph of time taken to find solution of order N in algorithm 3.

3.8 Comparison

In Figure 11, all the algorithms execution times are shown side by side. The first algorithm proved to be the least time efficient of the algorithms. However, as our group discovered more properties of both the Farey sequence and the Farey Sum, we were able to use them to improve the efficiency of our algorithm. The end result was that our final algorithm was both the most efficient and the simplest to code of the three.

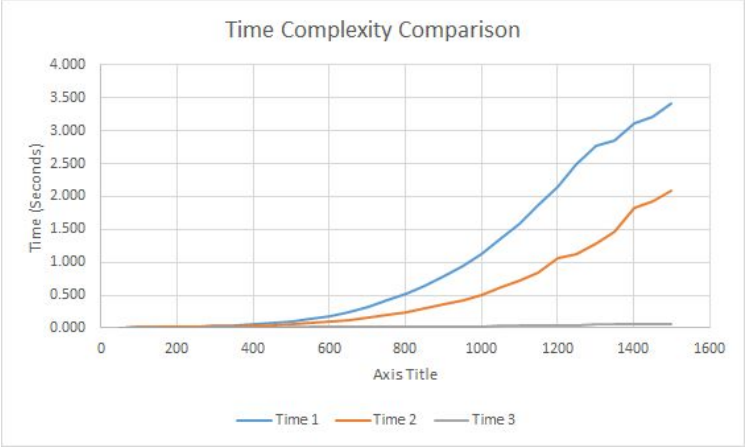


Figure 11. Graph of time taken to find solution of order N in each algorithm.

4 Application of Farey Sequence

Approximation of irrational numbers

4.1 Theorem

For two positive rational numbers b/d and a/c, if $0 < b/d < a/c$, the median is $a+b/c+d$, which means $b/d < a+b/c+d < a/c$. Given an interval [0, n], using the Farey Sequences can help to find a rational approximation of an irrational number under the condition that denominator is less than a given value N. General principle that applies in finding the approximation is to keep finding the mediant in an interval until it reaches to the point when the denominator is closest to N.[5]

4.2 Example of using Farey Sequence to find rational approximation

To find out the closet rational approximation of $1/\pi$ where the denominator is no greater than 100. {0/1, 1/1} is the first interval and the first Farey sequence to start with. The mediant in this interval is 1/2. As it is clear that $\pi > 2$, therefore, $1/\pi < 1/2$, which means $1/\pi$ lies between the interval of {0/1, ½}. The mediant for {0/1, 1/2} now becomes (0+1) / (1+2), which is 1/3. We keep repeating this step until the denominator is approaching 100. The final answer is 29/91 because its denominator is closest to 100. [5]

Drawbacks of the application

When 29/91 is converted to a decimal, it is rounds to 0.3186813186813187 while $1/\pi$ is around 0.31830988618. Despite the fact that 29/91 is the best approximation using this method, it is not guaranteed to be the most accurate approximation. For example, 7/22 =0.31818181818 is a better approximation than 29/91. To find the most suitable approximation, the exact value of an irrational number has to be known. Another restriction is the irrational number to be approximated has to be between 0 and 1 because the mediant search will focus on the left side of the tree {0,1}. [5]

Using Stern-Brocot Tree to find approximation

Back to the Stern-Brocot tree that is discussed in Theorem 3, the implementation of this tree will work effectively for a computer attempting to find the answer. See Figure 12.

Algorithm Design

Input: FindApprox=$1/\pi$
MaxDenominator =100

Output: BestFraction= 29/91
MostAccurateFraction=7/22

Figure 12. Algorithm Design for Rational Approximation

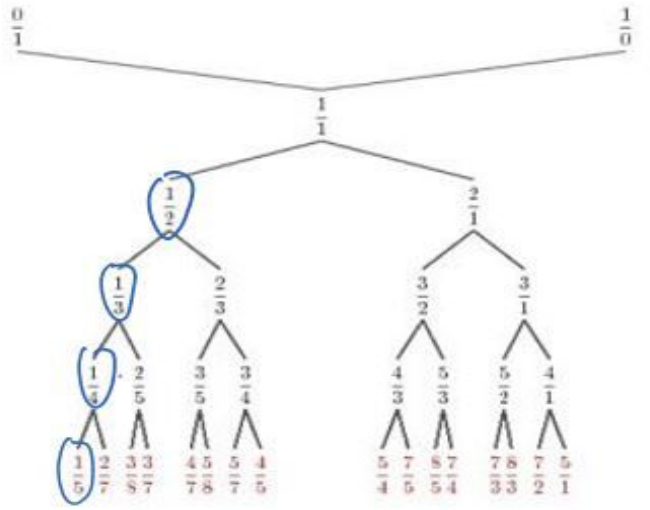


Figure 13. Searching for Mediant in Stern-Brocot Tree

The computer will go through the Stern-Brocot tree and narrow down to one fraction number when it meets the requirement. See Figure 13. For example, it will go the left side of {0/1, 1/1} to further narrow down the rational fraction for 1/π. In this way, the computer will take a very short time to find out the answer by incorporating the Stern-Brocot tree algorithm. This is similar to the divide and conquer method, which narrows the search range into one half of its original and by repeating this step, the answer can be found in a small range. [5]

6. Conclusion

Our team has explored the historical background of the Farey Sum and studied the theorems behind it. The Farey Sequence is displayed in the Stern-Brocot Tree, which allows us to further dig out relationships between fractions. Our team created three algorithms to compute the Farey Sum and continued to improve the algorithms by making them simpler and more efficient with each iteration. As shown above, the algorithms have used various Farey Sum properties and programming methods in order to try and create the most efficient algorithm. In addition to algorithms, our team has also demonstrated how to utilize the properties of Farey Sums to solve practical mathematical problems. The time to find the rational approximation of irrational numbers is almost constant by turning the problem into a Farey Sequence problem. In all, there may be other better algorithms and applications for the Farey Sequence but what remain unchanged are those common theorems in the Farey Sequence that we have studied through this project.

References

1. Guthery, S. B. (2011). *A motif of mathematics*:. Boston: Docent Press.
2. Daintith, J., & Nelson, R. D. (1989). *The Penguin dictionary of mathematics*. London: Penguin Books.
3. Fractions in the Farey Series and the Stern-Brocot Tree. (n.d.). Retrieved April 17, 2016, from http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fractions/fareySB.html#section1.1
4. Austin, D. (n.d.). Feature Column from the AMS. Retrieved April 22, 2016, from http://www.ams.org/samplings/feature-column/fcarc-stern-brocot
5. Dawson, Jonathan Ainsworth Michael, and John Pianta James Warwick. (n.d.). "The Farey Sequence." Retrieve May 2, 2016, from http://www.maths.ed.ac.uk/~aar/fareyproject.pdf