

Sums, Sequences, and Series in The Analysis of Algorithms

by William Shoaff with lots of help

May 23, 2000

Contents

1	The Alice sum	1
1.1	The Alice sequence	3
1.2	The Alice series	3
1.3	Alice summary	3
2	The Gauss sum	3
2.1	The Gauss sequence	4
2.2	The Gauss series	5
2.3	Gauss summary	5
3	The Zeno sum	5
3.1	The Zeno sequence	7
3.2	The Zeno series	7
3.3	Zeno summary	8
4	Other useful sums	8
5	Other useful series	9
6	References	9
7	Problems	9

You can download a postscript version of this file (which is prettier) at

<http://www.cs.fit.edu/%7Ewds/classes/algorithms/Sums/sums.pdf>

1 The Alice sum

Alice, of course, is Lewis Carroll's Alice from "Through the Looking-glass" and "Alice's Adventures in Wonderland." In 'Queen Alice' from "Through the Looking-glass," the Red and White Queen are speaking with Alice about "manners" and "lessons" when

‘Can you do Addition?’ the White Queen asked.

‘What’s one and one and one and one and one and one and one and one and one and one?’

‘I don’t know,’ said Alice. ‘I lost count.’

‘She can’t do Addition,’ the Red Queen interrupted.

‘Can you do Subtraction? Take nine from eight.’

Counting single operations that occur repeatedly is the most fundamental problem in algorithm analysis. Fortunately, we’ve learned to count and add many, many years ago. To count one operation over and over we use summation notation:

$$\sum_{k=0}^{n-1} 1 = n.$$

It may be a constant other than 1 that we need to count:

$$\sum_{k=0}^{n-1} c = cn.$$

Here’s a simple `for` which to analyze we must count a constant number of things.

```
for (int k=0; k < n; k++) { ip += u[k] * v[k]; }
```

The most simple answer is to say that `ip += u[k] * v[k]` is one operation, in which case the time complexity (or running time) is

$$\sum_{k=0}^{n-1} 1 = n.$$

A more detailed analysis would count additions a , multiplications m , assignments s , and array references r , yielding a running time of

$$\sum_{k=0}^{n-1} (a + m + s + 2r) = (a + m + s + 2r)n.$$

In any case, we would summarize this using big- O notation, saying the time complexity $T(n)$ for the inner product loop has *order* $O(n)$, written

$$T(n) = O(n),$$

meaning that never more than (roughly) n operations occur. A more precise answer would be

$$T(n) = \Theta(n),$$

meaning the inner product loop always takes (roughly) n operations. Notice we do not include counts of operations on the `for` loop variable `k`. We *could* if we needed to be really precise, but we usually *don’t*.

1.1 The Alice sequence

Associated with the sum is the sequence of values to be added, in this case

$$\langle 1, 1, 1, \dots \rangle$$

or more generally,

$$\langle c, c, c, \dots \rangle$$

for some constant c . Notice our sequences are infinite.

1.2 The Alice series

We'd like add all the terms in the Alice sequence, but of course we can't because the result is infinite. However, if we multiply each term in the sequence by a power of variable z we can formally evaluate the infinite sum (or series).

$$\sum_{k=0}^{\infty} 1 \cdot z^k = 1 + z + z^2 + z^3 + \dots = \frac{1}{1-z}.$$

The function $G(z) = \frac{1}{1-z}$ is called the *generating function* of the sequence $\langle 1, 1, 1, \dots \rangle$. More on this later.

1.3 Alice summary

A (singly-nested) for loop usually gives rise to a summation of a constant value (which we can assume is normalized to the value 1).

```
for (int k=0; k < n; k++) { do one thing; }
```

$$\sum_{k=0}^{n-1} 1 = n = O(n).$$

You should be able to identify the similarity between the for loop and the summation.

2 The Gauss sum

The Gauss sum is named for a story that's described in E.T. Bell's "Men of Mathematics." Carl Friedrich Gauss (1777-1855) is considered the greatest mathematician of his time and the equal of Archimedes and Isaac Newton.

It seems that Gauss' third grade teacher needed a break so she assigned the class the problem of totaling the sum of the first 100 integers thinking that this would occupy the students for most of the afternoon.

Gauss of course was able to tabulate the sum in a matter of seconds to the chagrin of his teacher. What Gauss needed to compute is:

$$\sum_{k=1}^{100} k = \frac{100 \cdot 101}{2}.$$

More generally, we have

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}.$$

It should come as no surprise that doubly-nested `for` loops give rise to a Gauss sum. Consider the doubly-nested loops:

```
int t1 = 1 - t;
for (j=1; j<= n; j++) {
    for (i=0; i<= n - j; i++) {
c[i]= t1 * c[i] + t * c[i+1] ;
    }
}
```

which are from the de Casteljau algorithm for evaluating points on a Bézier curve. To count the operations, we declare `c[i]= t1 * c[i] + t * c[i+1]` to be one operation, and count how many times it executes inside the inner loop using an Alice sum:

$$\sum_{i=0}^{n-j} 1 = n - j + 1.$$

This is how many operations occur for each value of j . Notice that as j takes on the values from 1 to n , $n - j + 1$ takes on values from n down to 1. Thus counting the operations inside the outer `for` loop gives:

$$\sum_{j=1}^n n - j + 1 = \sum_{j=1}^n j = n(n+1)/2 = O(n^2).$$

Here we've use the *order* or *big-O* notation to indicate that roughly n^2 operations occur in computing a single point on a Bézier curve.

2.1 The Gauss sequence

Associated with the Gauss sum is the sequence of values to be added, in this case

$$\langle 1, 2, 3, \dots \rangle.$$

This is an *arithmetic* sequence. The general arithmetic sequence is

$$\langle c, a + c, 2a + c, \dots \rangle.$$

2.2 The Gauss series

We'd like add all the terms in the Gauss sequence, but of course we can't because the result is infinite. However, if we multiply each term in the sequence by a power of variable z we can formally evaluate the infinite sum (or series).

$$\sum_{k=0}^{\infty} (k+1) \cdot z^k = 1 + 2z + 3z^2 + \dots = \frac{1}{(1-z)^2}.$$

You should note that the above series is the formal derivative of the Alice series

$$\sum_{k=0}^{\infty} (k+1) \cdot z^k = 1 + 2z + 3z^2 + 4z^3 + \dots$$

and the right hand side

$$\frac{1}{(1-z)^2}$$

is the derivative of

$$\frac{1}{1-z}.$$

The function $G(z) = \frac{1}{(1-z)^2}$ is called the *generating function* of the sequence $\langle 1, 2, 3, \dots \rangle$. More on this later.

2.3 Gauss summary

A double-nested for loop usually gives rise to a a summation of natural numbers

```
for (int k=0; k < n; k++) {  
    for (int j=0; j < k; j++) { do one thing; }  
}
```

$$\sum_{k=0}^{n-1} \sum_{j=0}^{k-1} 1 = \sum_{k=0}^{n-1} k = \frac{n(n-1)}{2} = O(n^2).$$

You should be able to identify the similarity between the for loops and the summations. Notice that things are a little more complex since the loop bounds on the inner loop may depend on the loop index of the outer loop.

3 The Zeno sum

The Zeno sum is named for Zeno of Elea, a Greek philosopher, who argued that he could never be killed by an arrow since one would always have to travel halfway from its current position toward his heart, and so could never reach its target. *This is an experiment Zeno should not have tried!*

If the original distance is 2 units from 0, then the halfway point is 1, and the next halfway point is 1/2, the next is 1/4, and so on. Thus Zeno argues that you can only add up a finite number of these term and the result is always less than two, but we know you can sum the series

$$\sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k,$$

and the answer is 2 — Zeno’s dead!

The Zeno sum would truncate the series at some finite upper bound n yielding:

$$\sum_{k=0}^n \frac{1}{2^k} = \frac{1 - \left(\frac{1}{2}\right)^{n+1}}{1 - \frac{1}{2}}.$$

It is a special case of a *geometric sum*

$$\sum_{k=0}^n a^k \frac{1 - a^{n+1}}{1 - a}, \quad a \neq 1.$$

You should be familiar with this type of sum from experience. The most obvious is in the binary representation of the largest number using $n + 1$ bits, that is

$$1111 \dots 1_2 = \sum_{k=0}^n 2^k = 2^{n+1} - 1$$

Zeno-type sums arise naturally from *recursive* algorithms. Consider the inorder binary tree traversal algorithm below.

```
public void inorderTreeWalk(x) {
    if (x != null) {
        inorderTreeWalk(x.left());
        x.print();
        inorderTreeWalk(x.right());
    }
}
```

And assume that the binary tree is balanced so that one-half of the nodes are in the left subtree and the other half are in the right subtree. We’ll find that the time complexity of this algorithm is (roughly) described by the *recurrence relation*

$$T(n) = 2T(n/2) + 1$$

with *initial condition* $T(1) = 1$. The reasoning for the recurrence relation is:

To traverse a binary tree, take two walks: one down the left half and one down the right half, stopping in between to print the information at the current node.

In any event, there are several ways which we will learn later that allow us to write the recurrence relation as:

$$\begin{aligned} T(n) &= 2T(n/2) + 1 \\ &= 2^k + 2^{k-1} + \dots + 2 + 1 \\ &= 2^{k+1} - 1 \end{aligned}$$

where $k = \lg n$, so that $T(n) = 2n - 1$.

3.1 The Zeno sequence

Associated with the Zeno sum is the sequence of values to be added, in this case

$$\langle 1, 1/2, 1/4, \dots \rangle.$$

The general *geometric sequence* is

$$\langle 1, a, a^2, \dots \rangle, \quad a \neq 1.$$

3.2 The Zeno series

We'd like add all the terms in the Zeno series, and we can because the result is finite. However, following the established custom, we multiply each term in the sequence by a power of variable z we formally evaluate the infinite sum (or series).

$$\sum_{k=0}^{\infty} \frac{1}{2^k} \cdot z^k = 1 + z/2 + z^2/4 + \dots = \frac{1}{(1 - z/2)}.$$

You should note that the similarity of the above series with the Alice series

$$\sum_{k=0}^{\infty} z^k = 1 + z + z^2 + z^3 + \dots = \frac{1}{1 - z}$$

The substitution $z \rightarrow z/2$ yields the Zeno series from the Alice series. More generally, we have the geometric series

$$\sum_{k=0}^{\infty} (az)^k = 1 + az + (az)^2 + (az)^3 + \dots = \frac{1}{1 - az}.$$

The function $G(z) = \frac{1}{1 - az}$ is called the *generating function* of the sequence $\langle 1, a, a^2, \dots \rangle$.

3.3 Zeno summary

Geometric sums often arise when analyzing recursive algorithms. This analysis leads to *recursion relations* with *initial conditions* such as:

$$T(n) = aT(n-1) + c, \quad T(0) = d.$$

or

$$T(n) = aT(n/b) + c, \quad T(1) = d.$$

4 Other useful sums

- The sum of squares

$$\sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6}$$

- Harmonic numbers (sum of fractions)

$$\sum_{k=1}^n \frac{1}{k} = H_n = \ln n + \gamma + O(1/2n)$$

where $\gamma \approx 0.5772156649 \dots$ is *Euler's constant*.

- A *factorial* sum

$$\sum_{k=0}^n k(k+1) = \frac{n(n+1)(n+2)}{3}$$

- A Gauss-Zeno sum

$$\sum_{k=0}^n ka^k = \frac{a - (n+1)a^{n+1} + na^{n+2}}{(1-a)^2}$$

- The binomial theorem

$$\sum_{k=0}^n \binom{n}{k} x^k y^{n-k} = (x+y)^n$$

- Parallel summation of binomial coefficients

$$\sum_{k \leq n} \binom{r+k}{k} = \binom{r+n+1}{n}$$

- Summation on the upper index of binomial coefficients

$$\sum_{m \leq k \leq n} \binom{k}{m} = \binom{n+1}{m+1}$$

5 Other useful series

- The exponential function

$$e^z = \sum_{k=0}^{\infty} \frac{z^k}{k!}$$

- The cosine function

$$\cos(z) = \sum_{k=0}^{\infty} (-1)^k \frac{z^{2k}}{(2k)!}$$

- The sine function

$$\sin(z) = \sum_{k=0}^{\infty} (-1)^k \frac{z^{2k+1}}{(2k+1)!}$$

- The harmonic function

$$\ln\left(\frac{1}{1-z}\right) = \sum_{k=1}^{\infty} \frac{z^k}{k}$$

6 References

The best book I know for the topics discussed in these notes is Concrete Mathematics: A Foundation for Computer Science by Ron Graham, Don Knuth, and Oren Patashnik [3]. However, you may be want to start with a more elementary book. Most discrete mathematics books have a section on sums, sequences, and generating functions, for example, see [4]. Finally, most books on the analysis of algorithms have information on sums, sequences, and series, see, for example, [2] or [1].

7 Problems

Problem 1: Evaluate the sums below

- $\sum_{k=0}^n 3^k$
- $\sum_{k=0}^n 5k + 4$
- $\sum_{k=1}^n (2k+1)/k(k+1)$
- $\sum_{k=0}^n (-1)^k k^2$

Problem 2: What are the sequences associated with the following generating functions:

- $G(z) = e^z$
- $G(z) = \cos(z)$

- $G(z) = \sin(z)$
- $G(z) = \ln\left(\frac{1}{1-z}\right)$
- $G(z) = 1/(1+z)^2$

Problem 3: Find simple formulas for the following series

- $\sum_{i=0}^{\infty} \frac{1}{(i+1)(i+2)} z^i$
- $\sum_{k=1}^{\infty} (-1)^k \frac{z^k}{k}$
- $\sum_{k=0}^{\infty} \frac{k(k+1)}{2} z^k$

Problem 4: Use summations to analyze the running time complexity of the code below that computes the Fibonacci number F_n .

```
public int Fibonacci(int n) {
    int last=0;
    int current=1;
    for (int i = 1; i <= n; i++) {
        last += current;
        current = last - current;
    }
    return last;
}
```

Problem 5: Use summations to analyze the running time complexity of the code below that computes the binomial coefficient $\binom{n}{j}$.

```
public int binomialCoefficient(int n, int j) {
    int r=0;
    for (int i = 1; i < n; i++) {
        for (j = i+1; j <= n; j++) {
            for (int k = 1; k <= j; k++) { r = r+1; }
        }
    }
    return r;
}
```

Problem 6: Use summations to analyze the running time of the code below that part of the code that computes the Haar wavelet transform of a linear array $c[1..2^j]$.

```
public void decompose(double c) {
    int h = c.length/2;
    for (int i = 1; i < h; i++) {
```

```
    scratch[i]    = (c[2i-1] + c[2i])/2.0;
    scratch[h+i] = (c[2i-1] - c[2i])/2.0;
}
c = scratch;
```

References

- [1] G. BRASSARD AND P. BRATLEY, *Fundamentals of Algorithmics*, Prentice-Hall, 1996. ISBN 0-13-335068-1.
- [2] T. H. CORMAN, C. E. LEISERSON, AND R. L. RIVEST, *Introduction to Algorithms*, McGraw-Hill, 1990.
- [3] R. L. GRAHAM, D. E. KNUTH, AND O. PATASHNIK, *Concrete Mathematics*, Addison-Wesley, 1989.
- [4] K. H. ROSEN, *Discrete Mathematics and its Application*, McGraw-Hill, 1999. ISBN 0-07-289905-0.