USING FINITE AND INFINITE SUMS TO DEVELOP ALGORITHMIC THINKING

John M. Morrison

March 10, 2020

1 The sum $\sum_{k=1}^{n} k^r$

All of us have had first-year calculus courses in which this sum arises from the Riemann sum for the integral $\int_0^b x^r\,dx$.

Typically we would show our students the case of and invoke the formula

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

Promptly we get asked, "Where did you get that formula?" Verifying a few simple cases palliates the incurious. Proving it by induction leaves the more interested student still asking how we got the formula. We shall begin by answering this question using an algorithm that requires surprisingly little machinery.

We begin by writing down the equation

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

the students easily believe this because they see that the sum of n ones is n. We then look at the telescoping sum

$$\sum_{k=1}^{n} (a_k - a_{k-1}) = a_n - a_0.$$

The middle terms all collapse out, much like the middle pieces of an inexpensive pocket telescope. We have assembled all the necessary machinery.

To see this in its full generality we must re-index and cancel as follows.

$$\sum_{k=1}^{n} (a_k - a_{k-1}) = \sum_{k=1}^{n} a_k - \sum_{k=1}^{n} a_{k-1} = a_n + \sum_{k=1}^{n=1} a_k - \left(a_0 + \sum_{k=2}^{n} a_{k-1} \right).$$

Now re-index the second sum like so

$$\sum_{k=2}^{n} a_{k-1} = \sum_{k=1}^{n-1} a_k.$$

The telescoping sum theorem is now proved. It is a simple exercise to produce a second proof by doing an induction on n.

Now apply the telescoping sum theorem to the case of $a_k = k^2$; we get

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

Next, unpack the summand and separate to see that

$$n^2 = 2\sum_{k=1}^{n} k - \sum_{k=1}^{n} 1 = \sum_{k=2}^{n} a_{k-1} - n.$$

We rearrange to see that

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

The reader can now turn the crank for himself. Using the fact that

$$n^3 = \sum_{k=1}^{n} k^3 - (k-1)^3,$$

you can expand to see that

$$n^3 = 3\sum_{k=1}^{n} k^2 - 3\sum_{k=1}^{n} k + n.$$

Solve for

$$\sum_{k=1}^{n} k^2$$

as if it were a giant X. You can repeat this procedure and obtain sums for higher powers.

We will now get a recurrence relation for the general case. For non-negative integers r and n, we define

$$P_r(n) = \sum_{k=1}^{n} k^r.$$

In the telescoping sum theorem, let $a_k = k^{r+1}$, $k \geq 0$. Applying the telescoping sum theorem, we have

$$n^{r+1} = \sum_{k=1}^{n} k^{r+1} - (k-1)^{r+1}.$$

Next, we invoke the Binomial theorem on the power of $r+1$ inside the sum to obtain

$$n^{r+1} = \sum_{k=1}^{n} k^{r+1} - \left(\sum_{j=0}^{r+1} \binom{r+1}{j} k^j (-1)^{r+1-j} \right).$$

In the parenthesized sum here we see that the $j = r+1$ term yields a

$$\binom{r+1}{r+1} k^{r+1} (-1)^0 = k^{r+1}$$

This causes a cancellation which leaves us with

$$\begin{aligned} n^{r+1} &= -\sum_{k=1}^{n} \sum_{j=0}^{r} \binom{r+1}{j} k^j (-1)^{r+1-j} \\ &= \sum_{j=0}^{r} \sum_{k=1}^{n} \binom{r+1}{j} k^j (-1)^{r-j} \end{aligned}$$

Next, we interchange the sums to obtain

$$\begin{aligned} n^{r+1} &= -\sum_{j=0}^{r} \binom{r+1}{j} (-1)^{r+1-j} \sum_{k=1}^{n} k^j \\ &= \sum_{j=0}^{r} \binom{r+}{j} (-1)^{r-j} P_j(n) \\ &= (r+1) P_r(n) + \sum_{j=0}^{r-1} \binom{r+1}{j} (-1)^{r-j} P_j(n) \end{aligned}$$

We now isolate the $P_r(n)$ and divide by $r+1$ to obtain

$$P_r(n) = \frac{n^{r+1}}{r+1} + \frac{1}{r+1} \sum_{j=0}^{r-1} \binom{r+1}{j} (-1)^{r-j} P_j(n).$$

This formula shows us several things. First it computes each polynomial P_r in terms of its predecessors P_0, P_1, $\cdots P_r$.

We know the degree of P_0 is 1 since $P_0(n) = n$ for all positive integers a nice example of a poor procedure, the Bozo sort. Imagine you have a list of names recorded on index cards. Shuffle the cards; check their order and stop if the order is right. Otherwise, repeat the procedure until the order is correct.

Suppose you have n index cards; the probability that any given shuffling places them in the correct order is . This is a sequence of independent trials with the same probability of success. The average time to resolution is ; there is no upper bound on the amount of time the procedure can take. The worst-case scenario is very bad, indeed. n. By way of induction, assume that $\deg(P_j) = j + 4$. for all $j < r$. Our recursive formula allows us to conclude that $\deg(P_r) \leq r + 1$. But the only n^{r+1} term present is

$$\frac{n^{r+1}}{r+1},$$

so we have $\deg(P_r) = r + 1$. In fact, we know

$$P_r(n) = \frac{n^{r+1}}{r+1} + \text{lower degree terms}$$

so the lead term for P_r is

$$\frac{n^{r+1}}{r+1}.$$

A nice project for a clever programming student is to use this formula to compute the P_r polynomials up to a specified degree. This is an excellent little test for a Python class that handles polynomials with rational coefficients. Note the Python3 has a new Fraction type; you can create a class for Polynomials using a list implementation.

In this way, students can use polynomials as if they are a built-in data type. Developing these classes hugely solidifies the student's understanding of algebraic manipulation. As an alternative, students can also program this in a symbolic super-high-level language such as Maple, Mathematica, Sage or Macsyma. Note that Sage is freely available.

2 An Analysis of Some Sorting Procedures

Suppose we are confronted with a list or vector of orderable items, such as numbers, characters, or words. What is an efficient procedure for sorting such a list or vector? We shall discuss several procedures here and use a bit of the last section to perform an analysis of the efficiency of these procedures.

Here is a nice example of a poor procedure, the Bozo sort. Imagine you have a list of names recorded on index cards. Shuffle the cards; check their order and stop if the order is right. Otherwise, repeat the procedure until the order is correct. Suppose you have n index cards; the probability that any given shuffling places them in the correct order is $n!$.

This is a sequence of independent trials with the same probability of success. The average time to resolution is $n!$; there is no upper bound on the amount of time the procedure can take. The worst-case scenario is very bad, indeed.

Next, we shall look at the bubble sort. Suppose we have n items and that they are shuffled in random order. We shall sort them from smallest to largest. Begin by examining the first pair of elements. In the manner of a computer scientist, we shall begin counting at zero; we compare the zeroth and first elements. These are either in the right or the wrong order; if the order is wrong fix it, else do nothing. Now go to the next pair, the first and second element. Check their order; if it is correct, do nothing, else switch them. Continue in this fashion until you have compared the $n-1$th and nth elements. On this first pass, we have done comparisons. We should also notice that the largest element has "bubbled up" to the top. Therefore, when we make the second pass, we have one fewer pairs to check. The second pass takes n - 2 comparisons. Continuing in this fashion, we make

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

comparisons. The amount of work it takes to sort a list of size n is roughly $n^2/2$. Observe that there is no luck involved here; you have to do all the comparisons to do the bubble sort.

Now let us look at the insertion sort. This sort is familiar to anyone who plays card games such as hearts or bridge. Players often order their cards by suit (spades, hearts, clubs, diamonds) then by rank, deuce through Ace. As the player receives each new card, he inserts it in his hand in the proper order. This is how we proceed in an insertion sort. We start at one end of the list, picking off the elements in succession, until the list is depleted. We maintain a separate list, in which we file each element in its proper order.

Here, there is a luck factor involved. The worst case scenario is that when our sorted list has n items in it. we will have to compare our new element with all n items. However, this seldom happens. More realistically we have to look at about $n/2$ elements.

The worst-case scenario yields

$$w = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$$

comparisons. The average case yields

$$a = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{4}$$

comparisons, about half that of the bubble sort.

For small sorts, the insertion sort is a reasonable choice. Both the insertion and bubble sort are called *quadratic sorts*, because the cost of running them rises quadratically with the size of the job. Quadratic sorts are OK for sorting

small collections of sortable objects. But if you want to sort something such as the Manhattan telephone directory, they become glacially slow.

We will take a look at a sorting procedure called *quicksort*. This is a recursive method that, typically, is quite speedy. In the worst case scenario, it becomes quadratic.

Suppose we have a list of sortable items. Take the list of items and move all of the items smaller than the zeroth item to the left of the zeroth item and all larger ones to the right. This procedure is called a *pivot*. Now call quicksort on the two sub-lists on either side of the pivot. Do this recursively. The base case occurs when a sub-list has one element; in this event it is sorted.

Let us define by $T(n)$ the time it takes to sort a list of n elements. The pivot procedure requires roughly n checks. Let us make the seemingly bold assumption that the pivot winds up roughly near the middle of the list each time. In this event, we have

$$T(n) = 2T(n/2) + cn,$$

where c is the unit cost of making a check. Now let us use this iteratively a few times.

$$\begin{aligned} T(n) &= 2T(n/2) + cn \\ &= 2(2T(n/4) + cn/2) + cn \\ &= 4T(n/4) + 2cn \\ &= 4(2T(n/8) + cn/4) + cn \\ &= 8T(n/8) + 3cn \\ &= 2^k R(n/2^k) + k)cn \end{aligned}$$

Continuing in this fashion, we get Now put $k = 2^n$ to get

$$T(n) = nT(1) + cn\log(n),$$

so that $T(n) = O(n\log(n))$, or that $T(n)$ is at worst proportional to $n\log(n)$.

In a worst case scenario with a ton of bad luck, this can become quadratic. We can by and large prevent that by using a preliminary step of swapping the first element with some other randomly chosen element and proceeding with the pivot process.

3 Summation of Rational Functions

Consider the familiar sum

$$\sum_{n-1}^{\infty} \frac{1}{n(n+1)}.$$

We have all seen the familiar demonstration that this is actually a telescoping sum; here it is

$$\sum_{n-1}^{N} \frac{1}{n(n+1)} = \sum_{n-1}^{N} \frac{1}{n} - \frac{1}{n+1} = 1 - \frac{1}{N+1}.$$

Now let $N \to \infty$ to see that

$$\sum_{n-1}^{\infty} \frac{1}{n(n+1)} = 1.$$

Can this method be expanded to handle other situations? Here we show a modest extension. Suppose we wish to find

$$\sum_{n=1}^{\infty} \frac{n}{(n+1)(n+2)(n+3)}.$$

We know that this converges by limit comparison with

$$\sum_{n=1}^{\infty} \frac{1}{n^2}.$$

Begin by performing the partial fractions expansion

$$\frac{n}{(n+1)(n+2)(n+3)} = \frac{A}{n+1} + \frac{B}{n+2} + \frac{C}{n+3}.$$

Clearing the denominators we obtain

$$n = A(n+2)(n+3) + B(n+1)(n+3) + C(n+1)(n+2).$$

Taking $n = -1$ we get

$$-1 = A(1)(2) = 2A$$

so we have

$$A = -1/2.$$

Taking $n = -2$ we get

$$-2 = B(-1)(1) = -B,$$

so $B = 2$. Finally, taking $n = -3$ gives

$$-3 = C(-1)(-2) = 2C,$$

so $C = -3/2$. Hence, we have for any positive integer N,

$$\sum_{n=1}^{N} \frac{n}{(n+1)(n+2)(n+3)} = -\frac{1}{2}\sum_{n=1}^{N} \frac{1}{n+1} + 2\sum_{n=1}^{N} \frac{1}{n+2} - \frac{3}{2}\sum_{n=1}^{N} \frac{1}{n+3}.$$

Now we re-index subtract, and clean up to see that

$$\begin{aligned}\sum_{n=1}^{N} \frac{n}{(n+1)(n+2)(n+3)} &= -\frac{1}{2}\sum_{n=2}^{N+1}\frac{1}{n} + 2\sum_{n=3}^{N+2}\frac{1}{n} - \frac{3}{2}\sum_{n=4}^{N+3}\frac{1}{n}.\\ &= -\frac{1}{2}\left(\frac{1}{2}+\frac{1}{3}+\sum_{n=4}^{N+1}\frac{1}{n}\right) + 2\left(\frac{1}{3}+\frac{1}{N+2}+\sum_{n=4}^{N+1}\frac{1}{n}\right)\\ &\quad -\frac{3}{2}\left(\frac{1}{N+2}+\frac{1}{N+3}+\sum_{n=4}^{N+1}\frac{1}{n}\right)\\ &= -\frac{1}{2}\left(\frac{1}{2}+\frac{1}{3}\right) + 2\left(\frac{1}{3}+\frac{1}{N+2}\right) - \frac{3}{2}\left(\frac{1}{N+2}+\frac{1}{N+3}\right)\end{aligned}$$

Let $N \to \infty$ to conclude that

$$\sum_{n=1}^{\infty} \frac{n}{(n+1)(n+2)(n+3)} = -\frac{5}{12} + \frac{2}{3} = \frac{1}{4}.$$

To convince the timid, we provide this little Python program to play with. Note: Use Python 3 or cast `n` in the function to a float.

```
def a(n):
    return n/((n+1)*(n+2)*(n+3))

total = 0
for k in range(1, 101):
    total += a(k)
print("total = %s" % total)
```

running it produces a result of .2404. Go to the 1000th partial sum and you get .2490.

Recall the definition of the *Riemann Zeta Function*,

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}, \qquad s > 1.$$

This is a differentiable (in fact, complex analytic) function defined on $(1, \infty)$. This function can be evaluated analytically for positive even integers values of n. For example, $\zeta(2) = \pi^2/6$ and $\zeta(4) = \pi^4/90$, and $\zeta(6) = \pi^6/945$. You can see some very hairy calculations carried out at [1]. These evaluations can be accomplished by appealing to the Parseval Theorem for Fourier Series; a good reference is [2]. This book is very accessible to a good Calculus student and quite inexpensive.

We will perform a calculation that uses the Zeta function. Consider the series

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

We begin by performing the partial fractions expansion

$$\frac{1}{n^2(n+1)} = \frac{A}{n} + \frac{B}{n^2} + \frac{C}{n+1)}.$$

Clearing denominators gives

$$1 = An(n+1) + B(n+1) + Cn^2.$$

Put $n = 0$ to obtain $1 = B$. Next, put $n = -1$ to get $1 = C$. By picking coefficients, we have $A + C = 0$, so $C = -1$.

Hence, we have

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

Summing, we get

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

Now trim the sums and re-index as we did before to get

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

Re-indexing, we have

$$-\sum_{n=1}^{N} \frac{1}{n} + \sum_{n=1}^{N} \frac{1}{n+1} = -\sum_{n=1}^{N} \frac{1}{n} + \sum_{n=2}^{N+1} \frac{1}{n} = -1 + \frac{1}{N+1}.$$

Reassembling the pieces we have

$$\sum_{n=1}^{N} \frac{1}{n^2(n+1)} = \sum_{n=1}^{N} \frac{2}{n^2} - 1 + \frac{1}{N+1}$$

Now let $N \to \infty$ to get

$$\sum_{n=1}^{\infty} \frac{1}{n^2(n+1)} = \zeta(2) - 1 = \frac{\pi^2}{6} - 1$$

This procedure lets the student compute many sums that we now are content to say are convergent by the integral or limit comparison test.

4 References

[1] Tolstov, G, *Fourier Series*

[2] Wolfram Math World article on the Riemann Zeta Function, `http://mathworld.wolfram.com/RiemannZetaFunction.html`