Sums and Recurrences

P. Sam Johnson

National Institute of Technology Karnataka (NITK)
Surathkal, Mangalore, India

Overview

In the lecture, we disucss a notation for sum and general techniques that make summation user-friendly.

The following methods are discussed.

- repertoire method
- summation factor method
- perturbation method.

We will be working with sums of the general form

$$a_1 + a_2 + \cdots + a_n$$

where each a_k is a number that has been defined somehow. This "**three-dot**" notation has the advantage that we can "see" the whole sum, almost as if it were written out in full, if we have a good enough imagination. But it can be ambiguous and a bit long-winded.

For example, if $1 + 2 + \cdots + 2^{n-1}$ is supposed to denote a sum of n terms, not of 2^{n-1}, we should write it more explicitly as

$$2^0 + 2^1 + \cdots + 2^{n-1}.$$

The other notation is, notably the **delimited form** (**sigma-notation**)

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

because it uses the Greek letter $\sum$ (upper case sigma).

This notation tells us to include in the sum precisely those terms a_k whose index k is an integer that lies between the lower and upper limits 1 and n, inclusive. In words, we "sum over k, from 1 to n." **Joseph Fourier** introduced this delimited $\sum$-notation in 1820, and it soon took the mathematical world by storm.

Joseph Fourier (1768 - 1830)

Each element a_k of a sum is called a **term**. The quantity after $\sum$ (here a_k) is called the **summand**. The index variable k is said to be **bound** to the $\sum$ sign.

Generalized sigma-notation

A generalized sigma-notation is even more useful than the delimited form. We simply **write one or more conditions under the** $\sum$, to specify the set of indicies over which summation should take place. It can be written as

$$\sum_{1 \le k \le n} a_n.$$

For example, we can express the sum of the squares of all odd positive integers below 100 as follows:

$$\sum_{\substack{1 \le k < 100 \\ k \text{ odd}}} k^2.$$

But the delimited form of this sum

$$\sum_{k=0}^{49}(2k+1)^2$$

is more cumbersome and less clear. Similarly, the sum of reciprocals of all prime numbers between 1 and N is

$$\sum_{\substack{p \le N \\ p \text{ prime}}} \frac{1}{p}.$$

The delimited form would require us to write

$$\sum_{k=1}^{\pi(N)} \frac{1}{p_k},$$

where p_k denotes the kth prime and $\pi(N)$ is the number of primes less than or equal to N.

The biggest advantage of general sigma-notation is that we can manipulate it more easily than the delimited form. For example, suppose we want to change the index variable k to $k+1$. With the general form, we have

$$\sum_{1 \le k \le n} a_k = \sum_{1 \le k+1 \le n} a_{k+1}$$

it is easy to see what is going on, and we can do the substitution almost without thinking. But with the delimited form, we have

$$\sum_{k=1}^{n} a_k = \sum_{k=0}^{n-1} a_{k+1}$$

it is harder to see what has happened, and we are more likely to make a mistake.

On the other hand, the **delimited form is not completely useless**. It is nice and tidy, and we can write it quickly because

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

has seven symbols compared with

$$\sum_{1 \le k \le n} a_k$$

having eight symbols.

Therefore we will often use $\sum$ with upper and lower delimiters when we state a problem or present a result, but we will prefer to work with relations-under-$\sum$ when we are manipulating a sum whose index variables need to be transformed.

Formally, we write

$$\sum_{P(k)} a_k$$

as an abbreviation for the sum of all terms a_k such that k is an integer satisfying a given property $P(k)$. (A “property $P(k)$” is any statement about k that can be either true or false.)

For the time being, we will assume that only finitely many integers k satisfying $P(k)$ have $a_k \neq 0$; otherwise infinitely many nonzero numbers are being added together, and things can get a bit tricky.

At the other extreme, if $P(k)$ is false for all integers k, we have an “empty” sum; the value of an empty sum is defined to be zero.

People are often tempted to write

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

instead of

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

because the terms for $k = 0, 1$, and n in this sum are zero.

We will find it advantageous to keep upper and lower bounds on an index of summation as simple as possible, because sums can be manipulated much more easily when the bounds are simple. Indeed, the form $\sum_{k=2}^{n-1}$ can even be dangerously ambiguous, because its meaning is not at all clear when $n = 0$ or $n = 1$. Zero-valued terms cause no harm, and they often save a lot of trouble.

Kenneth E. Iverson introduced a wonderful idea in his programming language APL and we will see that it greatly simplifies many things. The idea is simply to enclose a true-or-false statement in brackets, and to say that the result is 1 if the statement is true, 0 if the statement is false. For example,

$$[p \text{ prime}] = \begin{cases} 1 & \text{if } p \text{ is a prime number} \\ 0 & \text{if } p \text{ is not a prime number.} \end{cases}$$

Iverson's convention allows us to express sums with no constraints whatever on the index of summation, because we can rewrite

$$\sum_{P(k)} a_k$$

in the form

$$\sum_k a_k[P(k)].$$

If $P(k)$ is false, the term $a_k[P(k)]$ is zero, so we can safely include it among the terms being summed.

This makes it easy to manipulate the index of summation, because we don't have to fuss with boundary conditions.

If we use Iverson's convention to write the sum of reciprocal primes $\leq N$ as

$$\sum_n [p \text{ prime}]\ [p \leq N]/p,$$

there is no problem of division by zero when $p = 0$, because our convention tells us

$$[0 \text{ prime}]\ [0 \leq N]/0 = 0.$$

The **three-dots form** often suggests useful manipulations, particularly the combination of adjacent terms, since we might be able to spot a simplifying pattern if we let the whole sum hang out before our eyes. But too much detail can also be overwhelming.

Sigma-notation is compact, impressive and often suggestive of manipulations that are not obvious in three-dots form. When we work with sigma-notation, zero terms are not generally harmful; in fact, zeros often make $\sum$-manipulation easier.

Sums and Recurrences

There is an intimate relation between sums and recurrences. First we shall see how sum is reduced to recurrence.

The sum $S_n = \sum_{k=0}^{n} a_k$ is equivalent to the recurrence $S_0 = a_0$ $S_n = S_{n-1} + a_n$, for $n > 0$. Therefore we can evaluate sums in closed forms.

Example

Consider the following sum-recurrence

$$\begin{aligned} R_0 &= \alpha \\ R_n &= R_{n-1} + \beta + \gamma n, \quad \text{for } n > 0. \end{aligned}$$

We find $R_1 = \alpha + \beta + \gamma, R_2 = \alpha + 2\beta + 3\gamma$, and so on. In general, the solution can be written in the form $R_n = A(n)\alpha + B(n)\beta + C(n)\gamma$, where $A(n), B(n)$ and $C(n)$ are the coefficients of dependence on the general parameters α, β and γ.

Repertoire method

This method tells us to try plugging in simple funcions on R_n, hoping to find constant parameters α, β and γ, where the solution is especially simple.

$R_n = 1$	$\alpha = 1, \beta = 0 = \gamma$	$A(n) = 1$
$R_n = n$	$\alpha = 0, \beta = 1, \gamma = 0$	$B(n) = n$
$R_n = n^2$	$\alpha = 0, \beta = -1, \gamma = 2$	$2C(n) - B(n) = n^2$ hence $2C(n) = \frac{n^2+n}{2}$

Thus $R_n = \alpha + n\beta + \frac{n^2+n}{2}\gamma$.

Exercise

1. *Evaluate the sum $\sum_{k=0}^{n}(a + bk)$ by repertorie method.*

Conversely, many recurrences can be reduced to sums.

Summation Factor

Consider the tower of Hanoi recurrence

$$\begin{aligned} T_0 &= 0 \\ T_n &= 2T_{n-1}+1, \quad \text{for } n>0. \end{aligned} \tag{1}$$

(1) is rewritten as

$$\begin{aligned} T_0/2^0 &= 0 \\ T_n/2^n &= T_{n-1}/2^{n-1}+1/2^n, \quad \text{for } n>0. \end{aligned}$$

Let $S_n = T_n/2^n$. We get $S_0 = 0, S_n = S_{n-1} + 1/2^n, \quad \text{for } n > 0.$

Hence

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

Thus $T_n = 2^n S_n = 2^n - 1.$

Summation Factor : General Technique

Consider the recurrence of the form $a_n T_n = (b_n T_{n-1} + c_n)$.

We reduce the above recurrence to a sum. The idea is to multiply both sides by a summation factor, s_n (s_n should not be zero).

Choose s_n such that $s_n b_n = s_{n-1} a_{n-1}$.

We get $s_n a_n T_n = s_n b_n T_{n-1} + s_n c_n$. That is,

$$s_n a_n T_n = s_{n-1} a_{n-1} T_{n-1} + s_n c_n.$$

Let $S_n = s_n a_n T_n$. We have a sum-recurrence $S_n = S_{n-1} + s_n c_n$. Hence

$$S_n = s_0 a_0 T_0 + \sum_{k=1}^{n} s_k c_k = s_1 b_1 T_0 + \sum_{k=1}^{n} s_k c_k.$$

Therefore the solution to the original recurrence is

$$T_n = \frac{1}{s_n a_n}\left(s_1 b_1 T_0 + \sum_{k=1}^{n} s_k c_k\right).$$

How to find the summation factor s_n?

The relation $s_n = s_{n-1}\frac{a_{n-1}}{b_n}$ can be **unfolded** to tell us that the fraction

$$s_n = \frac{a_{n-1}a_{n-2}\cdots a_1}{b_n b_{n-1}\cdots b_2}$$

or any convenient constant multiple of this value, will be a suitable **summation factor** for the recurrence

$$a_n T_n = b_n T_{n-1} + c_n.$$

Example

The tower of Hanoi recurrence has $a_n = 1$ and $b_n = 2$. Here $s_n = 2^{-n}$. s_n is to be multiplied both sides of $T_n = 2T_{n-1} + 1 \ \ (n \geq 0)$ if we want to reduce the recurrence to a sum.

Caution! The summation factor method works whenever all the a's and all the b's are nonzero.

A sorting method : Quick-sort

Quick-sort is one of the most important methods for sorting data inside a computer. The average number of **comparison steps** made by quicksort when it is applied to n items in random order satisfies the recurrence

$$\begin{aligned} C_0 &= 0 \\ C_n &= (n+1) + \frac{2}{n}\sum_{k=0}^{n-1} C_k, \quad \text{for} \quad n > 0. \end{aligned} \tag{2}$$

If C_n denotes the average number of moves needed to sort n numbers (in random order initially), it can be shown that C_n satisfies the recursion.

If (2) is rewritten as

$$a_n T_n = b_n T_{n-1} + C_n$$

then the recurrence can be reduced to a sum.

If we multiply both sides of (2) by n, we can get rid of the division.

$$nC_n = n^2 + n + 2\sum_{k=0}^{n-1} C_k, \quad \text{for} \quad n > 0.$$

To get rid of the $\sum$ sign, we have

$$(n-1)C_{n-1} = (n-1)^2 + (n-1) + 2\sum_{k=0}^{n-2} C_k, \quad \text{for} \quad n > 1.$$

Hence

$$nC_n - (n-1)C_{n-1} = 2n + 2C_{n-1}, \quad \text{for} \quad n > 1.$$

Therefore the original recurrence for C_n reduces to a much simpler one :

$$\begin{aligned} C_0 &= 0 \\ nC_n &= (n+1)C_{n-1} + 2n, \quad \text{for} \quad n > 0. \end{aligned}$$

A summation factor s_n with $a_n = n$, $b_n = n + 1$ and $c_n = 2n$ is $s_n = \frac{2}{(n+1)^n}$. Thus

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

The sum

$$\sum_{k=1}^{n} \frac{1}{k+1}$$

is very similar to a quantity that arises frequently in applications. We denote

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k}.$$

The letter H stands for "harmonic"; H_n is a **harmonic number** (because the nth harmonic produced by a violin string is the fundamental tone produced by a string that is $1/k$ times as long.)

C_n in terms of H_n

We can express C_n in terms of H_n as follows:

$$\begin{aligned}
\sum_{k=1}^{n} \frac{1}{k+1} &= \sum_{1\le k\le n} \frac{1}{k+1} \\
&= \sum_{1\le k-1\le n} \frac{1}{k} \\
&= \sum_{2\le k\le n+1} \frac{1}{k} \\
&= \sum_{1\le k\le n} \frac{1}{k} - 1 + \frac{1}{n+1} \\
&= H_n - \frac{n}{n+1}.
\end{aligned}$$

Thus $C_n = 2(n+1)H_n - 2n$.

Connection between calculus and discrete mathematics

We would like to discuss a connection between calculus (infinite calculus) and discrete mathematics (finite calculus).

The infinite series (harmonic series)

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

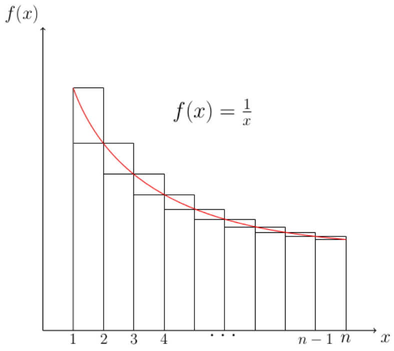
is divergent because a subsequence

$$(s_{2^n})_{n=1}^{\infty}$$

of sequence of partial sums $(s_n)_{n=1}^{\infty}$ is stricly increasing and not bounded above (since $\sum_{n=1}^{\infty} \frac{1}{2^n} > \frac{n}{2}$).

That is, $(H_n)_{n=1}^{\infty}$ diverges. Also $(\log n)_{n=1}^{\infty}$ diverges, by integral test for sequence.

But we shall prove that the sequence $(H_n - \log n)_{n=1}^{\infty}$ converges.



Let $G_n = H_n - \log n$. Consider $y = \frac{1}{x}$.

$$\begin{aligned} \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} &< \int_1^n \frac{1}{x} dx < 1 + \frac{1}{2} + \cdots + \frac{1}{n-1} \\ H_n - 1 &< \log n < H_{n-1} < H_n \end{aligned}$$

Hence $0 < H_n - \log n < 1$.

Euler's constant

$$\begin{aligned} G_{n+1} - G_n &= [H_{n+1} - \log(n+1)] - [H_n - \log n] \\ &= \frac{1}{n+1} + \log(\frac{n}{n+1}) \\ &= \frac{1}{n+1} + \log(1 - \frac{1}{n+1}) \\ &= \frac{1}{n+1} - \frac{1}{n+1} - \frac{1}{2(n+1)^2} - \frac{1}{3(n+1)^3} - \cdots < 0. \end{aligned}$$

Therefore (G_n) is decreasing and bounded above, hence it converges.

Euler showed that $-[\frac{d}{dx}\Gamma(x)]_{x=1} = \gamma = 0.577215664$. The Gamma function is defined by

$$\Gamma(x) = \int_0^\infty e^{-t} t^{x-1} dt \quad (x > 0)$$

The limit of (G_n) is called **Euler's constant** and it is nothing by γ. The numbers e, π, γ are some important constants.

Manipulation of Sums

The key to success with sums is an ability to change one $\sum$ into another that is simpler or closer to some goal.

Let k be any finite set of integers. Sum over the elements of k can be transformed by using 3 single rules.

$\sum_{k\in K} ca_k = c\sum_{k\in K} a_k$	**distributive law** allows us to move constants in and out of a $\sum$.
$\sum_{k\in K}(a_k+b_k) = \sum_{k\in K} a_k + \sum_{k\in K} b_k$	**associative law** allows us to break a $\sum$ into two parts, or to combine two $\sum$'s into one.
$\sum_{k\in K} a_k = \sum_{p(k)\in K} p(k)$	**commutative law** says that we can reorder the terms in any way as we please.

We now compute the general sum of an arithmetic progression

$$S = \sum_{0 \le k \le n} (a + bk).$$

By the commutative law, we can replace k by $n-k$, we get

$$S = \sum_{0 \le n-k \le n} (a + b(n-k)) = \sum_{0 \le k \le n} (a + bn - bk).$$

These two equations can be added by using the associative law:

$$\begin{aligned} 2S &= \sum_{0 \le k \le n} [(a+bk) + (a + bn - bk)] \\ &= \sum_{0 \le k \le n} (2a + bk) = (n+1)(2a+bn). \end{aligned}$$

Hence

$$S = \sum_{0 \le k \le n} (a + bk) = \left(a + \frac{bn}{2}\right)(n+1) = \frac{a + (a+bn)}{2}(n+1),$$

the average of first and last terms times the number of terms.

Splitting off terms

The operation of **splitting off** a term

$$\sum_{0\le k\le n} a_k = a_0 + \sum_{1\le k\le n} a_k \quad \text{for } n \ge 0$$

is the basis of a **perturbation method** that often allows us to evaluate a sum in closed form.

The idea is to start with an unknown sum and call it S_n:

$$S_n = \sum_{0\le k\le n} a_k.$$

We write S_{n+1} in two ways:

1. by splitting off its last term
2. by splitting off its first term.

Splitting off terms

$$S_{n+1} = S_n + a_{n+1} \quad \text{(splitting off last term)}$$

and

$$\begin{aligned} S_{n+1} &= \sum_{0 \le k \le n+1} a_k = a_0 + \sum_{1 \le k \le n+1} a_k \\ S_{n+1} &= \sum_{1 \le k \le n+1} a_k \\ &= \sum_{1 \le k+1 \le n+1} a_{k+1} \end{aligned}$$

$$S_{n+1} = \sum_{0 \le k \le n} a_{k+1} \quad \text{(splitting off first term)}$$

If we express the last sum in terms of S_n, we obtain an equation whose solution is the sum we seek.

Exercises

2. *Find the sum of the following recursions by permutation method.*

   (a) $\sum_{0 \le k \le n} ax^k$ (b) $\sum_{0 \le k \le n} k2^k$ (c) $\sum_{0 \le k \le n} kx^k$.

   By using elementary techniques of differential calculus, find the closed form in a completely different way. Note that the derivative of a sum is the sum of the derivatives of its terms. Start with the equation

$$\sum_{k=0}^{n} x^k = \frac{1 - x^{n+1}}{1 - x}$$

   and take the derivative of both sides with respect to x, we get

$$\sum_{k=0}^{n} kx^{k-1} = \frac{1 - (n+1)x^n + nx^{n+1}}{(1-x)^2}.$$

   This example shows that there is a connection between calculus and

Exercises

Find the following sums.

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

References

1. **Graham, Knuth and Patashnik**, *"Concrete Mathematics – A Foundation for Computer Science"*, Pearson Education.
2. **Marko Petkovsek, Herbert S. Wilf and Doron Zeilberger**, "$A = B$", AK Peters Ltd., Wellesley, Massachusetts.
3. **Herbert S. Wilf**, *"Generatingfunctionology"*, Third Edition, AK Peters Ltd., Wellesley, Massachusetts.