Section 5.2 Summations and Closed Forms

A *closed form* is an expression that can be computed by applying a fixed number of familiar operations to the arguments. For example, the expression $2 + 4 + \ldots + 2n$ is not a closed form, but the expression $n(n+1)$ is a closed form.

Summation Notation: $$\sum_{k=1}^{n} a_k = a_1 + \cdots + a_n.$$

Summation Facts

(1) $\sum c a_k = c \sum a_k$. (2) $\sum (a_k + b_k) = \sum a_k + \sum b_k$.

(3) $\sum a_k x^{i+k} = x^i \sum a_k x^k$. (4) $\sum_{k=m}^{n} a_{k+i} = \sum_{k=m+i}^{n+i} a_k$.

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

Some Useful Closed Forms

(1) $\sum_{k=m}^{n} c = (n - m + 1)c$. (2) $\sum_{k=1}^{n} k = \dfrac{n(n+1)}{2}$.

(3) $\sum_{k=1}^{n} k^2 = \dfrac{n(n+1)(2n+1)}{6}$. (4) $\sum_{k=0}^{n} a^k = \dfrac{a^{n+1} - 1}{a - 1}$ (where $a \neq 1$).

(5) $\sum_{k=1}^{n} k a^k = \dfrac{a - (n+1)a^{n+1} + n a^{n+2}}{(a-1)^2}$ (where $a \neq 1$).

Example. Find a closed form for the expression $\sum_{k=2}^{n}(k-1)2^{k+1}$.

Solution:
$$\sum_{k=2}^{n}(k-1)2^{k+1} = \sum_{k=1}^{n-1} k2^{k+2} \qquad \text{(Fact 4)}$$
$$= 2^2\sum_{k=1}^{n-1} k2^{k} \qquad \text{(Fact 3)}$$
$$= 2^2(2-n2^n+(n-1)2^{n+1}) \qquad \text{(Form 5)}$$
$$= 2^3-(2-n)2^{n+2}.$$

Example. Find a closed form for $2+2^2\cdot 7+2^3\cdot 14+\cdots+2^n(n-1)\cdot 7$.

Solution: The sum has the form
$$2+\sum_{k=2}^{n}2^k(k-1)\cdot 7$$
$$= 2+7\sum_{k=2}^{n}(k-1)2^{k} \qquad \text{(Fact 1)}$$
$$= 2+7\sum_{k=1}^{n-1}k2^{k+1} \qquad \text{(Fact 4)}$$
$$= 2+14\sum_{k=1}^{n-1}k2^{k} \qquad \text{(Fact 3)}$$
$$= 2+14(2-n2^n+(n-1)2^{n+1}). \qquad \text{(Form 5)}$$

Quiz. Use summation facts and forms to prove that $2+3+\ldots+n=\frac{(n-1)(n+2)}{2}$.

Solution.

$$
\begin{aligned}
2+3+\ldots+n &= \sum_{k=2}^{n} k = \sum_{k=1}^{n-1}(k+1) \\
&= \sum_{k=1}^{n-1} k + \sum_{k=1}^{n-1} 1 \\
&= \frac{(n-1)(n)}{2} + (n-1) \\
&= \frac{(n-1)(n+2)}{2}.
\end{aligned}
$$

Quiz. Use summation facts and forms to find a closed form for $3+7+\cdots+(3+4n)$.

Solution.

$$
\begin{aligned}
\sum_{k=0}^{n}(3+4k) &= \sum_{k=0}^{n} 3 + \sum_{k=0}^{n} 4k \\
&= \sum_{k=0}^{n} 3 + 4\sum_{k=0}^{n} k \\
&= 3(n+1) + \frac{4n(n+1)}{2} \\
&= (3+2n)(n+1).
\end{aligned}
$$

Example. Let count(n) be the number of := statements executed by the following algorithm as a function of n, where $n \in \mathbf{N}$. Find a closed form for count(n).

```
i := 1;                                  (1)
while i < n do
   i := i + 1;                           (n – 1)
   for j := 1 to i do S od               (2 + 3 + … + n)
od
```

The expressions in parentheses indicate the number of times that := is executed.

Therefore, count(n) is the sum:

$$\begin{aligned}\text{count}(n) &= 1 + (n-1) + (2+3+\cdots+n) \\ &= (n-1) + (1+2+3+\cdots+n) \\ &= (n-1) + \frac{n(n+1)}{2}.\end{aligned}$$

Quiz. Let count(n) be the number of executions of S in the preceding algorithm as a function of n. Find a closed form for count(n).

Solution.

$$\begin{aligned}\text{count}(n) &= (2+3+\cdots+n) \\ &= (1+2+3+\cdots+n) - 1 \\ &= \frac{n(n+1)}{2} - 1.\end{aligned}$$

Example. Let count(n) be the number of times S is executed by the following algorithm as a function of n, where $n \in \mathbf{N}$. Find a closed form for count(n).

$i := 1$;
while $i < n$ **do**
  $i := i + 2$;
  for $j := 1$ **to** i **do** S **od**
od

Solution: Each time through the while-loop i is incremented by 2. So the values of i at the start of each for-loop are 3, 5, …, $(2k + 1)$, where $i = 2k + 1 \geq n$ represents the stopping point for the while-loop. So we have

$$\begin{aligned}\text{count}(n) &= 3 + 5 + \cdots + (2k + 1)\\ &= \sum_{i=1}^{k}(2i+1) = 2\sum_{i=1}^{k} i + \sum_{i=1}^{k} 1\\ &= \frac{2k(k+1)}{2} + k = k(k+1) + k = k(k+2).\end{aligned}$$

But we need to write count(n) in terms of n. Since $2k + 1 \geq n$ is the stopping point for the while-loop it follows that $2k - 1 < n$ is the last time the while-condition is true. In other words, we have the inequality $2k - 1 < n \leq 2k + 1$. Solving for k, we have $2k - 2 < n - 1 \leq 2k$, which gives $k - 1 < (n - 1)/2 \leq k$. Therefore $k = \lceil (n - 1)/2 \rceil$. Now we can write count(n) in terms of n as

$$\text{count}(n) = k(k + 2) = \lceil (n - 1)/2 \rceil (\lceil (n - 1)/2 \rceil + 2).$$

Approximating Sums. Suppose we have a sum that doesn't have a closed form or we can't find a closed form. Then we might find an approximation to suit our needs. For example, consider the following sum:

$$H_n = 1 + 1/2 + 1/3 + ... + 1/n.$$

This sum is called the *nth harmonic number* and it has no closed form. It is closely approximated by $\ln n$ because the definite integral of $1/x$ from 1 to n is $\ln n$. A constant, called Euler's constant, with a value close to 0.58, approximates the difference between H_n and $\ln n$ for large n. In other words, we have $H_n > \ln n$ and $H_n - \ln n$ is close to Euler's constant for large n. For example,

$$H_{10} - \ln 10 \approx 2.93 - 2.31 = 0.62$$
$$H_{20} - \ln 20 \approx 3.00 - 2.60 = 0.60$$
$$H_{40} - \ln 40 \approx 4.28 - 3.69 = 0.59$$

Example (Overlapping windows). Suppose we have set of files in the form of windows on a computer screen that are to be displayed as a stack of overlapped windows. How much total space is needed for the display? For example, if A is the area of the top window and each overlapped window has area $A/2$, then a stack of n files will have area

$$A + A(n - 1)/2.$$

This might take too much space for even small values of n, which would force A to be quite small. In this case we could decrease the size of each overlapped window to a smaller fraction of A. But it might be useful to make the overlapped files progressively smaller with sizes such as $A/2$, $A/3$, $A/4$, ... In this case the n files will have total area

$$A + A/2 + A/3 + ... + A/(n - 1) = AH_{n-1},$$

which is approximately equal to $A(\ln(n - 1) + 0.58)$.