Sums

One of the most common types of combinatorial formulae are summations, using Σ notation

$$\sum_{i=1}^{N} a_i = \sum_{1 \le i \le N} a_i = a_1 + a_2 + \cdots + a_{N-1} + a_N$$

The purpose of a notation is to make it easy to understand and manipulate expressions. Cumbersome notation is difficult to use, and can lead to errors:

$$\sum_{\substack{1 \le k < 100 \\ k \text{ odd}}} k^2 = \sum_{k=0}^{49} (2k+1)^2$$

} messy and dangerous, even if standard.

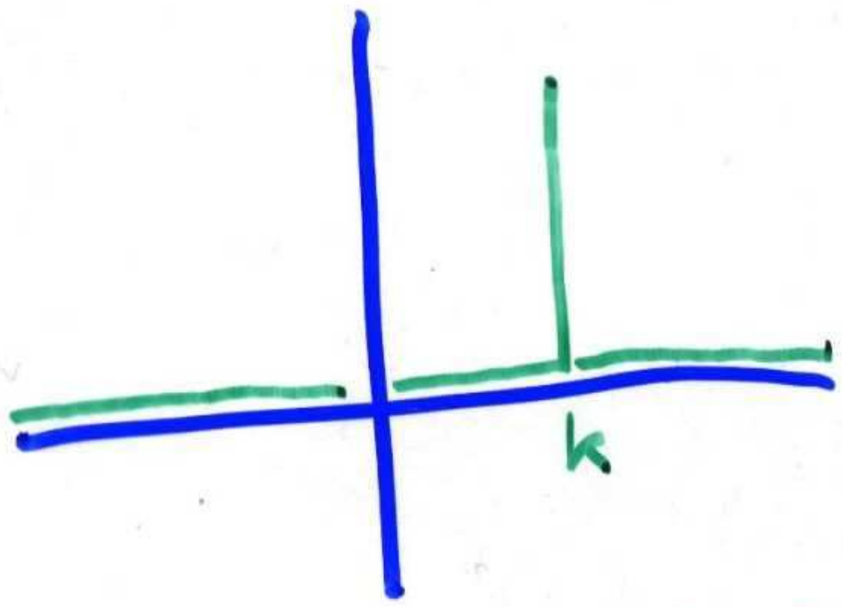
Introducing a new notation can be an important contribution (see the history of Calculus)

Knuth is powerful enough to introduce his own notation!

As a tool for working with sums we will use Iverson's Boolean notation from APL.

$$(p \text{ prime}) = \begin{cases} 1 & \text{if } p \text{ is prime} \\ 0 & \text{if } p \text{ is not prime.} \end{cases}$$

Thus the kronecker delta function can be written as $(n = k)$



We can stick such logical terms into expressions and manipulate them as anything else, taking into account that such terms are "more strongly zero" than anything else.

$$\sum_{\substack{p \le n \\ p \text{ prime}}} \frac{1}{p} = \sum_{p} (p \text{ prime})(p \le n)/p$$

1. Thus $p = 0$ is no problem $0 \cdot \frac{1}{0} = 0$
2. We can manipulate the logical terms as anything else.

Sums and Recurrences

Since last class we became "expert" in solving recurrences, we can use these skills to compute sums.

$$S_N = \sum_{k=0}^{N} a_k \longrightarrow \begin{aligned} S_0 &= a_0 \\ S_N &= S_{N-1} + a_N, \quad N > 0 \end{aligned}$$

The recurrence gives all prefix sums in the series.

Suppose $a_N = \beta + \gamma N$, this defines R:

$$\begin{aligned} R_0 &= \alpha \\ R_N &= R_{N-1} + \beta + \gamma N, \quad N > 0 \end{aligned}$$

We can use the repertoire method to solve it.

☆ The key idea: if we can convince ourselves that the solution is $R_N = A(N)\alpha + B(N)\beta + C(N)\gamma$, with independent solutions for three different sets of α, β, γ we define A, B, C!

Linear independence of the α, β, γ values is necessary but not sufficient. We must be able to solve the resulting special cases!

Thus we work backwards - identifying special cases we can solve + hoping they are independent.

Ex: $R_0 = \alpha$

$$R_N = R_{N-1} + \beta + \gamma_N$$

$$R_N = \sum_{k=1}^{N} (\gamma_N + \beta) + \alpha$$

Guess $R_N = A(N)\alpha + B(N)\beta + C(N)\gamma$

Case 1: $R_N = 1 \longrightarrow \alpha = 1$

$\beta = \gamma = 0$

trying $R_N = 2$ as a second case wouldn't be good since $2(1,0,0) = (2,0,0)$

$$R_0 = \alpha$$
$$R_N = R_{N-1} + \beta + \gamma_N$$

Case 2: $R_N = N \rightarrow \alpha = 0$

$$1 = \beta + \gamma$$
$$2 = 1 + \beta + 2\gamma \rightarrow \beta = 1,\ \gamma = 0$$

$R_N = R_{N-1} + 1$ (makes sense)

Case 3: $R_N = N^2 \rightarrow \alpha = 0$

$$1 = \beta + \gamma$$
$$4 = 1 + \beta + 2\gamma \rightarrow \gamma = 2,\ \beta = -1$$

$R_N = R_{N-1} + 2N - 1$ (proof by induction: $(N-1)^2 + 2N - 1 = N^2$)

Thus: $A(N) = 1$

$B(N) = N$

$C(N) = \frac{N^2 - B(N)\beta}{\gamma} = \frac{N^2 + N}{2}$

$$R_N = \alpha + \beta N + \gamma\left(\frac{N^2+N}{2}\right)$$

$\sum_{i=1}^{n} i \Rightarrow \gamma = 1,\ \beta = \alpha = 0 \Rightarrow \frac{N(N+1)}{2}$

$$R_0 = \alpha$$

$$R_N = R_{N-1} + \beta + \gamma_N \quad N > 0$$

N	R_N
0	α
1	$\alpha + \beta + \gamma$
2	$\alpha + 2\beta + 3\gamma$
3	$\alpha + 3\beta + 6\gamma$
4	$\alpha + 4\beta + 10\gamma$
5	$\alpha + 5\beta + 15\gamma$

Thus $R_N = A(N)\alpha + B(N)\beta + C(N)\gamma$

Guessing $A(N) = 1$ & $B(N) = N$ is easy, but we are stuck trying to guess $C(N)$ → hence the <u>repertoire</u> method

Converting History 1 Recurrences to Sums

The history of a recurrence is the number of terms it refers back to. In general, history 1 recurrences can be written as sums:

$$T_0 = 0 \qquad \rightarrow \qquad T_0/2^0 = 0$$

$$T_N = 2T_{N-1} + 1 \qquad T_N/2^N = \frac{T_{N-1}}{2^{N-1}} + \frac{1}{2^N}$$

By substituting $S_N = T_N/2^N$,

$$S_0 = 0$$

$$S_N = S_{N-1} + \frac{1}{2^N} \rightarrow S_N = \sum_{i=1}^{N} 2^{-i} = 1-\left(\tfrac{1}{2}\right)^N$$

Therefore $T_N = 2^N S_N = \underline{2^N - 1}$

This involved cleverness to see the substitution, but there is a general method with recurrences of the form

$$a_N T_N = b_N T_{N-1} + c_N$$

Suppose we multiply both sides of our recurrence by a function S_N, where $S_N b_N = S_{N-1} a_{N-1}$

$$a_N T_N = b_N T_{N-1} + c_N$$

$$S_N a_N T_N = S_N b_N T_{N-1} + c_N S_N$$

$$S_N a_N T_N = S_{N-1} a_{N-1} T_{N-1} + c_N S_N$$

so the substitution $\mathcal{S}_N = S_N a_N T_N$, gives

$$\mathcal{S}_N = \mathcal{S}_{N-1} + S_N c_N$$

But how do we find S_N?

$$S_N = \frac{S_{N-1} a_{N-1}}{b_N} = \frac{a_{N-1}}{b_N} \frac{S_{N-2} a_{N-2}}{b_{N-1}}$$

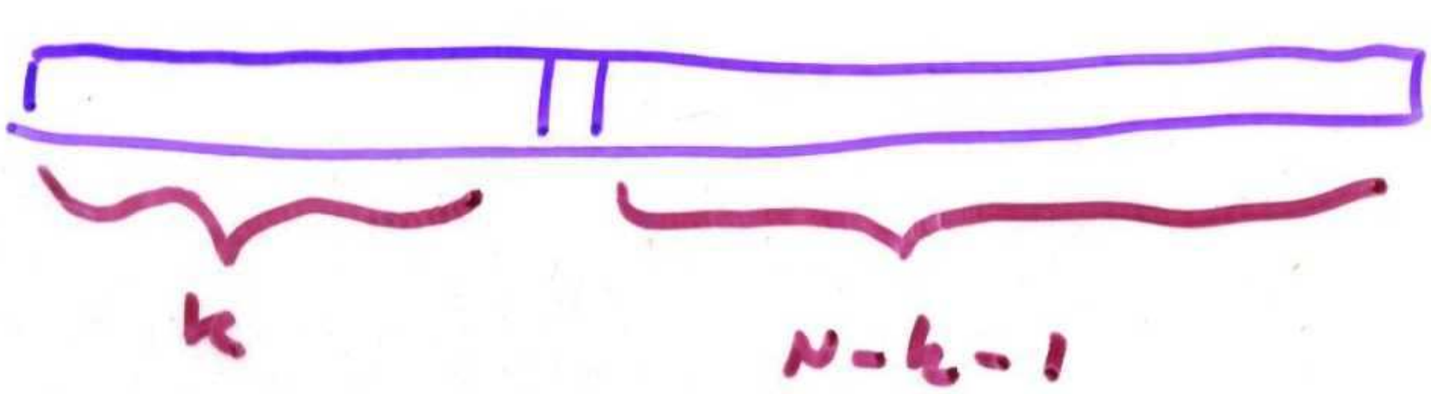
$$= \cdots\cdots = \boxed{\frac{a_{N-1} a_{N-2} \cdots a_1}{b_N \cdots\cdots b_2} S_0}$$

S_0 can be any constant >0, say $S_0 = 1$

In $T_N = 2T_N + 1$, $a_i = 1$
$b_i = 2$, $\rightarrow$ $S_N = 1/2^{N-1}$

The Quicksort Recurrence

Quicksort is the fastest general purpose sorting algorithm, which sorts recursively by randomly picking a pivot element, partitioning the data into what is $> p$ + $< p$, the recurrence



Let C_N be the average number of comparisons in sorting N items.

$$C_0 = 0$$

$$C_N = N+1 + \frac{1}{N}\sum_{k=0}^{N-1} C_k + C_{N-k-1}$$

$$= \underbrace{N+1}_{\text{partitioning}} + \frac{2}{N}\sum_{k=0}^{N-1} C_k$$

To solve the recurrence, the first step is to multiply both sides by N.

$$N C_N = N^2 + N + 2\sum_{k=0}^{N-1} C_k$$

To get rid of the sum, replace N by N-1 and subtract

$$-\ (N-1) C_{N-1} = (N-1)^2 + (N-1) + 2\sum_{k=0}^{N-2} C_k$$

$$N C_N - (N-1) C_{N-1} = \left(N^2 - (N-1)^2\right) + 1 + 2 C_{N-1}$$

$$\boxed{\begin{aligned} N C_N &= (N+1) C_{N-1} + 2N \\ C_0 &= 0 \end{aligned}}$$

} much nicer - history 1 recurrence

Multiplying both sides by $\frac{2}{N(N+1)}$ gives:

$$\frac{2 C_N}{N+1} = \frac{2 C_{N-1}}{N} + \frac{4}{N+1}$$

substitution $\rightarrow S_N = C_N/N+1$

$$\boxed{S_N = S_{N-1} + \frac{2}{N+1}}$$

} this is $2\sum_{k=1}^{N} \frac{1}{k+1}$

The N^{th} Harmonic number, $H_N = \sum_{k=1}^{N} \frac{1}{k}$

is approximately $\ln N$, so this proves quicksort is $O(N \log N)$

□

Manipulation of Sums

We have three facts about addition to help us.

$$\sum_{k \in K} c \cdot a_k = c \sum_{k \in K} a_k \qquad \text{(distributive law)}$$

$$\sum_{k \in K} (a_k + b_k) = \sum_{k \in K} a_k + \sum_{k \in K} b_k \qquad \text{(associative law)}$$

$$\sum_{k \in K} a_k = \sum_{p(k) \in K} a_{p(k)} \qquad \text{(commutative law)}$$

p is an arbitrary permutation

The Art of solving summations is the Art of applying these in the right order. Since we only know the right order when we are done, this means we must be willing to make mistakes and try again.

Arithmetic Progressions

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

$$= \sum_{0 \le N-k \le N} (a + b(N-k))$$

Commutative law in action see how this notation prevents mistakes.

$$\therefore 2S = \sum_{0 \le k \le N} ((a+bk) + (a+bN-bk)) = \sum_{0 \le k \le N} (2a + bN)$$

associative law in action!

$$2S = \sum_{0 \le k \le N} (2a + bN) = (2a + bN) \sum_{0 \le k \le N} 1 = (2a+bN)(N+1)$$

distributive law in action!

$$S = \left(a + \frac{b}{2}N\right)(N+1)$$

This idea of writing the same thing two different ways is called the perturbation method or name-and-conquer

Geometric Progressions

$$S_N = \sum_{0 \le k \le N} a x^k$$

perturb by taking the first and last terms of S_{N+1}

$$S_N + a x^{N+1} = a x^0 + \sum_{0 \le k \le N} a x^{k+1}$$

} sum is $x S_N$!

$$S_N + a x^{N+1} = a + x S_N$$

$$S_N = \frac{a - a x^{N+1}}{1 - x} \qquad x \neq 1$$

} very useful expression!

Good Ideas for Summations

1. Pull out what is independent of the sum index
2. Don't cancel zero terms too quickly
3. Be on the lookout for a direct substitution for the messy stuff, so we can treat it as a variable.
4. Remember what special cases we know how to solve!