Mathematical review

Functions

$f : A \to B$, associates for each $a \in A$ exactly one $b \in B$: $b = f(a)$.

- $f : R \to R$, $f(x) = x^2$
- $f : (-\frac{\pi}{2}, \frac{\pi}{2}) \to R, f(x) = tan(x)$
- $f : [0,1]$ x $[0,1] \to [0,2], f(x,y) = x + y$

Exponents

- $x^a x^b = x^{a+b}$
- $(x^a)^b = x^{ab}$
- $\frac{x^a}{x^b} = x^{a-b}$
- $x^n + x^n \neq x^{2n}$

Logarithms

- $log_b(xy) = log_b x + log_b y$
- $log_b(\frac{x}{y}) = log_b x - log_b y$
- $log_b x^\alpha = \alpha log_b x$
- $log_b x = \frac{log_a x}{log_a b}$, $a, b, x > 0, b \neq 1$
- $log 1 = 0$, $log 2 = 1$
- $a = b^{log_b a}$
- $log_b \frac{1}{x} = -log_b x$
- $log_b a = \frac{1}{log_a b}$
- $a^{log_b x} = x^{log_b a}$

Series

$$\sum_{i=1}^{n} S(i) = S(1) + S(2) + \ldots + S(n)$$

Example 1: **arithmetic progression**

$$\sum_{i=1}^{n} i = 1 + 2 + \ldots + n = \frac{n(n+1)}{2}$$

Example 1': $S = 2 + 5 + 8 + \ldots + (3k - 1) = 3(1 + 2 + 3 + \ldots + k) - k$

Example 2: **geometric progression**

$$\sum_{i=0}^{n} a^i = 1 + a + a^2 + \ldots + a^n = \frac{1 - a^{n+1}}{1 - a}$$

where $0 < a \neq 1$

- $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$
- If $0 < a < 1$ then $\sum_{i=0}^{n} a^i \leq \frac{1}{1-a}$ and $\sum_{i=0}^{\infty} a^i = \frac{1}{1-a}$

Example 3: $\sum_{i=1}^{\infty} \frac{i}{2^i} = 2$

Example 4: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Example 4': $\sum_{i=1}^{n} i^k \simeq \frac{n^{k+1}}{|k+1|}$, $k \neq -1$

Example 4": $k = -1$: $\sum_{i=1}^{n} \frac{1}{i}$ (**harmonic** sum)

- $H_n = \sum_{i=1}^{n} \frac{1}{i} \simeq log_e n$ (**harmonic numbers**)

Example 6: $\sum_{i=0}^{\infty} ix^i = \frac{x}{(1-x)^2}, 0 < x < 1$

Example 7: **Telescoping** series: $\sum_{i=1}^{n}(a_i - a_{i-1}) = a_n - a_0$

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

Proof techniques

- **Forward proof** (harder).

  Example 1: **geometric progression**

  $$\sum_{i=0}^{n} a^i = 1 + a + a^2 + \ldots + a^n = \frac{1 - a^{n+1}}{1 - a}$$

  Multiplying by a and then subtracting we get:

  $$(1-a)\sum_{i=0}^{n} a^i = 1 + (a - a) + \ldots + (a^n - a^n) - a^{n+1} = 1 - a^{n+1}$$

  Now divide by $(1 - a)$ to get final result.

  Example 2: $S = \sum_{i=1}^{\infty} \frac{i}{2^i} = \frac{1}{2} + \frac{2}{2^2} + \frac{3}{2^3} + \ldots$
  Multiply by 2 and subtract we obtain that $S = 1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \ldots = 2$

  Example 3: $\sum_{i=0}^{\infty} i x^i = \frac{x}{(1-x)^2}, 0 < x < 1$

  - Proof by differentiation.

- **Counter-example:** to disprove a claim C, give a valid example that contradicts it.

  Example 1: $x^n + x^n = x^{2n}$ is FALSE: x=2, n=2 then $2^2 + 2^2 \neq 2^4$

  Example 2: $F_0 = 1, F_1 = 1, F_{k+1} = F_k + F_{k-1}$: **Fibonacci numbers**. Prove that $F_k \leq k^2$ is false: for $k = 11$ we have $F_{11} = 144 > 11^2$.

- **Contradiction:** suppose claim C is false and prove that this contradicts some known facts (initial assumptions for claim).

  Example: To prove that there is an infinite number of primes assume that there is a largest prime P_k. Take $N = P_1 P_2 \ldots P_k + 1$. Then $N > P_k$ and N is prime (every number is either prime or a product of primes) which contradicts the initial assumption that P_k is the largest prime.

- **Induction:** prove true for all integers $n \geq n_0$

  1. **Basis:** prove C is true for n_0
  2. **Induction hypothesis:** assume C is true for $n_0 \leq i \leq n$. Prove it true for $n + 1$.

Example 1: $S(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
Basis: $S(1) = \frac{1(1+1)}{2} = 1$
Inductive proof: $S(n+1) = (n+1) + \frac{n(n+1)}{2} = \frac{(n+1)(n+2)}{2}$

Example 2: Prove that $F_k < (\frac{5}{3})^k$
Basis: $F_1 = 1 < \frac{5}{3}$
Inductive proof: $F_{k+1} < (\frac{5}{3})^k + (\frac{5}{3})^{k-1} < (\frac{5}{3})^{k-1}\frac{5}{3}\frac{5}{3}$

Example 3: $S_n = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
Basis: $S_1 = 1$
Inductive proof: $S_{n+1} = (n+1)^2 + \frac{n(n+1)(2n+1)}{6} = (n+1)\frac{2n^2+7n+6}{6} = \frac{(n+1)(n+2)(2n+3)}{6}$

Algorithm description: Pseudo-Code

Pseudo-code is a structured description of an algorithm: not as formal as a programming language.

Example: find the maximum element of an array.

Algorithm Max(A, n):

- Input: an array A storing n integers
- Output: maximum element in A.

max = a[1]
for i = 2 to n do
if max < A[i] then max = A[i]

Recurrences

When an algorithm contains a recursive call to itself, its running time can be often described as a recurrence.

- A **recurrence** is an equation or inequality that describes a function in terms of its value on smaller inputs.
- A function defined in terms of itself is called a **recursive function**.

Example 1: $f(x) = 2f(x-1) + x^2$, $f(0) = 0$.

Example 2: $T(n) = T(\frac{n}{2}) + n$, $T(1) = C$, for some constant C.

Note: avoid circular definitions.

Fundamental rules:

1. **base case:** can be solved without recursion.
2. **making progress:** the recursive call must make progress toward base.

Example: **Merge-Sort** (sort n numbers increasingly)

- A **divide-and-conquer** algorithm

1. divide the n numbers sequence in two subsequences of $n/2$ numbers each.
2. sort the two subsequences recursively.
3. merge the two sorted subsequences to produce the final answer.

- $T(n) = 2T(\frac{n}{2}) + cn$, $T(1) = a$, a, c constant: $T(n) = n \log n$.