Basic Mathematics for Algorithms

Definitions and Notations

Notation

N set of natural numbers {1,2, 3, …}

R set of real numbers

$R \geq 0$ set of real nonnegative real numbers

If X is a finite set, |X| is number of elements in X

Definitions and Notations

Polynomials

A polynomial of degree n is a function of the form

$$p(x) = c_n x^n + c_{n-1} x^{n-1} + \ldots + c_1 x + c_0,$$

with $c_n \neq 0$, c_i coefficient

Example; The function

$$p(x) = 3x^5 - 12x^3 + 9x^2 - 200x + 4$$

is a polynomial of degree 5. The coefficients are

$$c_5 = 3, \quad c_4 = 0, \quad c_3 = -12, \quad c_2 = 9, \quad c_1 = -200, \quad c_0 = 4.$$

Definitions an Notations

Upper bounds

A number a is said to be an upper bound for X if $x \leq a$ for all $x \in X$

Lower bounds

A number a is said to be an lower bound for X if $x \geq a$ for all $x \in X$

Mathematical Induction

Mathematical induction can be used to prove a sequence of statements indexed by the positive integers.

To prove a sequence of statements S(1),S(2),S(3),…
We must
Basis Step. Prove S(1) is true.
Inductive Step. Assume that S(n) is true (inductive hypothesis), and prove that S(n+1) is true, for all $n \geq 1$.

Mathematical Induction

Example 1. Prove that

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \text{ for all } n \geq 1.$$

Basis Step. We must show that the equation is true for n $= 1$; that is, we Must show that

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

The truth is immediate since both sides are equal to 1.

Mathematical Induction

Inductive Step. We must assume that the equation is true for n and prove That it is true for $n + 1$. Thus, we are assuming that

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}. \qquad \text{(Inductive hypothesis)}$$

We must prove that

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

The key to mathematical induction is to find case n "within" case n + 1. Here, the sum for n+1 is obtained from the sum for n by adding the (n+1)st term. In mathematical notation,

$$\sum_{i=1}^{n+1} i = \left(\sum_{i=1}^{n} i\right) + (n+1).$$

Since we are assuming that

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

We obtain

$$\sum_{i=1}^{n+1} i = \left(\sum_{i=1}^{n} i\right) + (n+1) = \frac{n(n+1)}{2} + (n+1).$$

Mathematical Induction

A little algebra shows that

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

Therefore,

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

And the proof is complete.

Mathematical Induction

Example 2.

Prove that

$$2n+1 \le 2^n \text{ for all } n \ge 3.$$

Basis Step. Since n=3 is the first statement, the Basis Step becomes

$$2 \cdot 3+1 \le 2^3.$$

Since $2 \cdot 3+1=7$ and $2^3=8$, the inequality is true for $n = 3$.

Inductive Step. We must assume that the inequality is true for n and prove that it is true for n+1. Thus, we are assuming that

$$2n+1 \le 2^n. \quad \text{(Inductive hypothesis)}$$

We must prove that

$$2(n+1)+1 \le 2^{n+1}.$$

Here, case n is "within" case n+1, in the sense that

$$2(n+1)+1=(2n+1)+2.$$

Noting that $2 \le 2^n$, for $n \ge 1$, we obtain

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

We have completed the Inductive Step.

Analysis of Algorithms

Analysis of algorithm

The process of deriving estimates for the time and space needed to execute the algorithm.

The time needed to execute an algorithm is a function of the input.

Analysis of Algorithms

For input of size **n**, we can ask for the maximum time and average time needed to execute an algorithm.

Worst-case time

Average-case time

- Difficult to obtain an explicit formula for the time complexity function.

- Primarily concerned with **estimating the time** of an algorithm rather than computing its exact time.

- Interested in how the **time grows** as the size of the input increases

Analysis of Algorithms

n	$T(n)=60n^2+5n+1$	$60n^2$
10	6,501	6,000
100	600,501	600,000
1,000	60,005,001	60,000,000
10,000	6,000,050,001	6,000,000,000

Comparing the growth of $t(n)$ with $60n^2$

To describe how the time grows as the size of the input ***n*** increases, we seek for the dominant term and ignore constant coefficients.

Since the dominant term is $60n^2$, $t(n)$ is of order n^2

$$t(n) = \Theta(n^2).$$

i.e., $t(n)$ grows like n^2 as n increase

Analysis of Algorithms

Definition 2.3.2 Let f and g be nonnegative functions on the positive integers. We write

$$f(n) = O(g(n))$$

and say that $f(n)$ is of order at most $g(n)$ or $f(n)$ is big oh of $g(n)$ if there exist constants $C_1 > 0$ and N_1 such that

$$f(n) \leq C_1 g(n) \text{ for all } n \geq N_1.$$

We write

$$f(n) = \Omega(g(n))$$

and say that $f(n)$ is of order at least $g(n)$ or $f(n)$ is omega of $g(n)$ if there exist constants $C_2 > 0$ and N_2 such that

$$f(n) \geq C_2 g(n) \text{ for all } n \geq N_2.$$

We write

$$f(n) = \Theta(g(n))$$

and say that **f(n) is of order g(n)** or $f(n)$ is theta of $g(n)$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Analysis of Algorithms

Example 3.

Since $\quad 60n^2 + 5n + 1 \leq 60n^2 + 5n^2 + n^2 = 66n^2$ for all $n \geq 1$,

we may take $C_1 = 66$ and $N_1 = 1$ in the definition and conclude that

$$60n^2 + 5n + 1 = O(n^2).$$

Since $\quad 60n^2 + 5n + 1 \geq 60n^2$ for all $n \geq 1$,

we may take $C_2 = 60$ and $N_2 = 1$ in the definition and conclude that

$$60n^2 + 5n + 1 = \Omega(n^2).$$

Since $60n^2 + 5n + 1 = O(n^2)$ and $60n^2 + 5n + 1 = \Omega(n^2)$,

$$60n^2 + 5n + 1 = \Theta(n^2).$$

Analysis of Algorithms

Example 4.

Since $2n + 3\lg n < 2n + 3n = 5n$ for all $n \geq 1$,

Thus, $2n + 3\lg n = O(n)$.

Also, $2n + 3\lg n \geq 2n$ for all $n \geq 1$.

Thus, $2n + 3\lg n = \Omega(n)$.

Therefore, $2n + 3\lg n = \Theta(n)$.

Analysis of Algorithms

Theta Form	Name
$\Theta(1)$	Constant
$\Theta(\lg \lg n)$	Log log
$\Theta(\lg n)$	Log
$\Theta(n^c)$, $0 < c < 1$	Sublinear
$\Theta(n)$	Linear
$\Theta(n \lg n)$	*n* log *n*
$\Theta(n^2)$	Quadratic
$\Theta(n^3)$	Cubic
$\Theta(n^k)$, $k \geq 1$	Polynomial
$\Theta(c^n)$, $c > 1$	Exponential
$\Theta(n!)$	Factorial

Common growth functions.