CIS 606 Analysis of Algorithms

Asymptotic Notations

Rationale

- **The order of growth of the running time of an algorithm gives a simple characterization of the algorithm's efficiency and also allows us to compare the relative performance of alternative algorithms.**

- **When we look at input sizes large enough to make only the order of growth of the running time relevant, we are studying the asymptotic efficiency of algorithms and describe it in Big-O, Big-Ω and Big-Θ notations.**
  - **E.g., $T(n) = n^2 + 100000n + 20 = O(n^2)$**

Objectives

- Understand asymptotic notations
- Learn to use Big-O, Big-Ω and Big-Θ notations to define a given running time function.

Prior knowledge

- Computing the exact running time of an algorithm.
- Understand the concept of the order of growth of a function.

When It Comes to Algorithm

```
bubbleSort(Array A : list of sortable items, n)
{                                                   cost      times
    flag = false                                     1          1
    do
        flag = false                                 1        #ofPass
        for i = 1 to n-1                             1     (n-1) * #ofPass
            if A[i-1] > A[i]                         1     (n-1) * #ofPass
                swap(A[i-1], A[i])                   1     <=(n-1) * #ofPass
                flag = true                          1     <=(n-1) * #ofPass
    while(flag)                                      1        #ofPass
}
```

Compute the asymptotic tight bound of T(n):

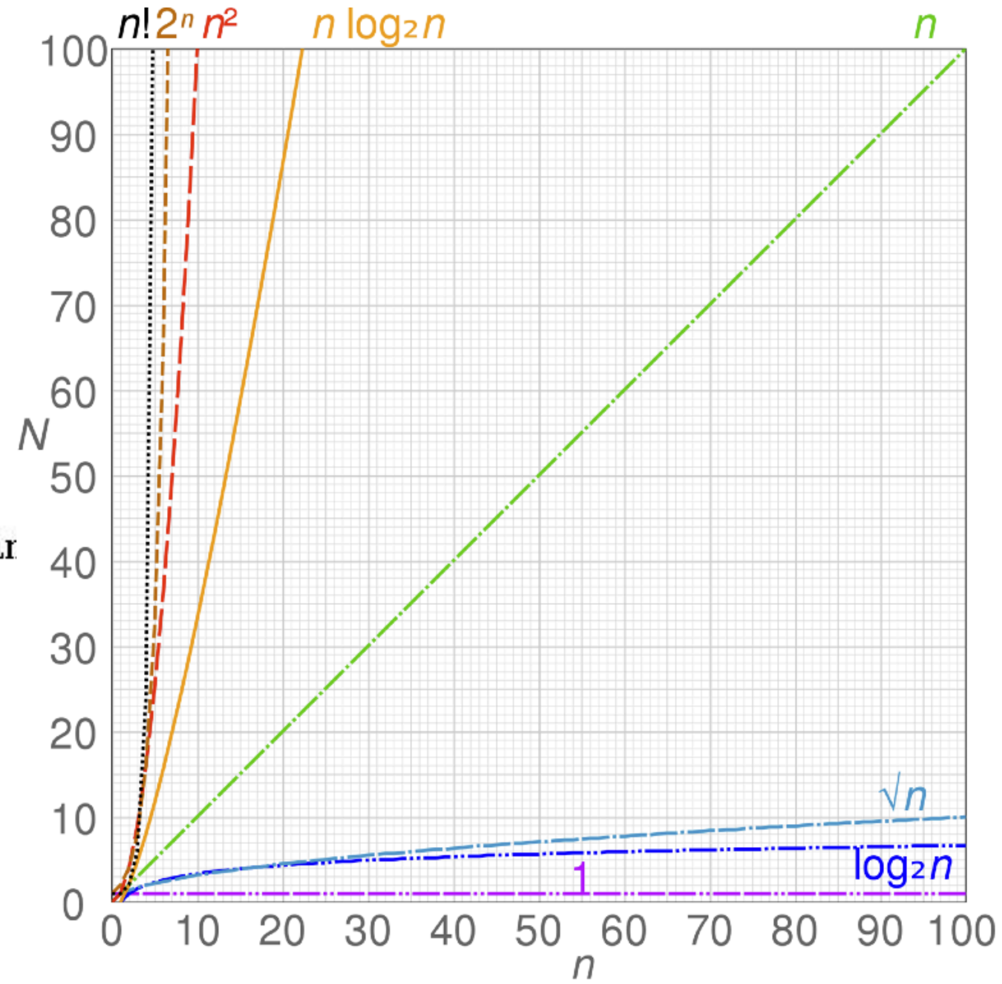
- Ignore the low-order terms
- Drop the leading constant

$$T(n) = 4n^2 - 2n + 1 = O(n^2) \text{ or } \Theta(n^2)$$

Order of Growth

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3 \cdot 10^1$	10^2	10^3	10^3	$3.6 \cdot 10^6$
10^2	6.6	10^2	$6.6 \cdot 10^2$	10^4	10^6	$1.3 \cdot 10^{30}$	$9.3 \cdot 10^{157}$
10^3	10	10^3	$1.0 \cdot 10^4$	10^6	10^9		
10^4	13	10^4	$1.3 \cdot 10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7 \cdot 10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0 \cdot 10^7$	10^{12}	10^{18}		

Table 2.1 Values (some approximate) of several functions importar for analysis of algorithms

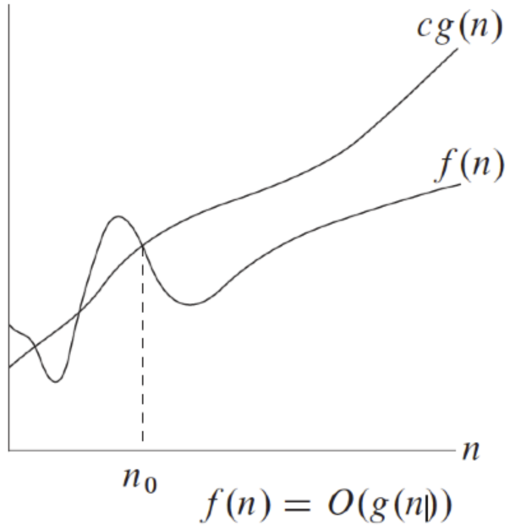


Efficiency Comparison

- Given two algorithms A and B
  - A runs in f(n) in the worst case
  - B runs in g(n) in the worst case
- Comparison between functions:
  - Big-O notation: g(n) is larger than or equal to f(n) asymptotically, i.e., A is faster or as fast as B.
  - Big-Ω notation: g(n) is smaller than or equal to f(n) asymptotically, i.e., A is slower or as slow as B.
  - Big-ϴ notation: g(n) is asymptoticly equal to f(n) asymptotically, i.e., A runs in the same time complexity as B.

Big-O notation — Asymptotic upper bound

- Given two functions f(n) and g(n):
- $f(n) = O(g(n))$ if constants $c>0$ and n_0 exist such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$.
- g(n) is the **asymptotic upper bound** of f(n).
  - $f(n) = n + \log n \quad g(n)= n^2$
  - $f(n) = n^3 + n \log n \quad g(n)= n^3$
  - $f(n) = 1024^2 \quad g(n)= 1$
  - $f(n) = n + \log n \quad g(n)= n$
- Big-O is used to bound the worst-case running time.
- $f(n) = O(g(n))$: $\{f(n)$: c and n_0 exist s.t. $f(n) \le cg(n)$ for all $n > n_0\}$



Example

Given $f(n) = 7n+8$, $g(n) = n$

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

Proof: To prove $f(n) = O(g(n))$, it is necessary to find constants $c>0$ and n_0 such that $f(n) \leq cg(n)$ for all $n \geq n_0$.

Based on the definition, we have

$7n+8 \leq cn \qquad n \geq n_0$

Based on observation, we can make $c=8$ and $n_0 = 10$

$f(n) = 7n+8 \qquad cg(n) = 8n = 7n+n$

$8<=n$

$7n+8 <= 8n$ for any $n>=10$ =====》 $f(n) = O(g(n))$.

Example (cont.)

Given $f(n) = 3n^2+6n+10$, $g(n) = n^3$ $n^2=O(n^3)$

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

Proof: Find $c>0$ and n_0 s.t. $f(n) \leq cg(n)$ for all $n \geq n_0$

$3n^2+6n+10 \leq cn^3$ $n \geq n_0$

$c=19$ $n_0 = 1$

$3n^2+6n+10 \leq 19n^3$

$3n^2+6n+10 \leq 3n^3 +6n^3 +10n^3$

$3n^2 \leq 3n^3$ $6n \leq 6n^3$ $10 \leq 10n^3$

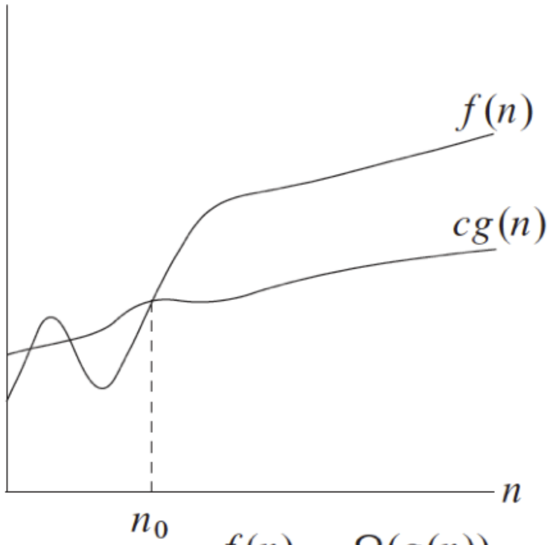
$f(n) \leq 19g(n)$ for all $n \geq 1$

Note n^3 is the asymptotic upper bound of f(n) and n^2 is the asymptotic **tight** bound of f(n).

Big-Ω notation — Asymptotic lower bound

- Given functions f(n) and g(n),
- $f(n) = \Omega(g(n))$ If constants $c>0$ and $n_0>0$ exist such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0$.
- g(n) is the asymptotic lower bound of f(n).
    - $f(n) = 1024\ n^2 + n \quad g(n)= n^2$.
    - $f(n) = n^3 + n \log n \quad g(n)= n^2$. ($c = 1$, $n0 = 1$)
    - $f(n) = 1024^2 \quad g(n)= 1$
    - $f(n) = n + \log n \quad g(n)= \log n$
- Big-Ω is used to bound the at-least time, e.g.,the best case.

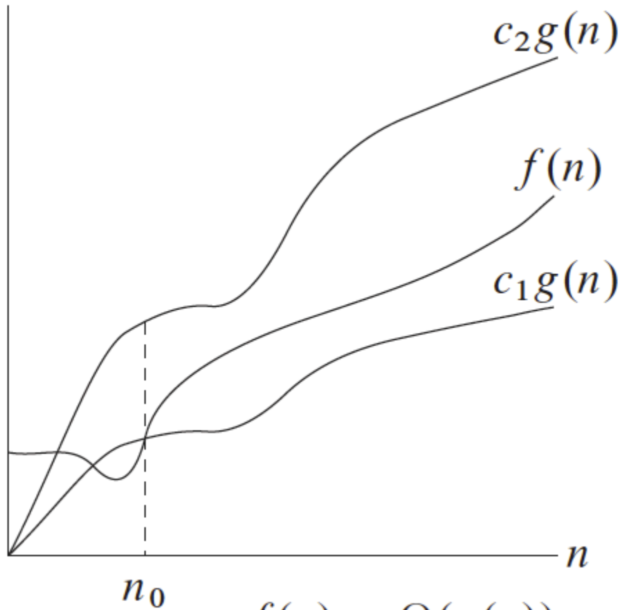
$\Omega(g(n)) = \{f(n):(c, n_0)$ with $f(n)>=cg(n)>0$ when $n>=n_0\}$



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

Big Θ notation — Asymptotic tight bound

- Given functions f(n) and g(n),
- $f(n) = \Theta(g(n))$ iff $f(n) = \Omega(g(n))$ and $f(n) = O(g(n))$ or
- If $c_1>0$, $c_2>0$, $n_0>0$ exist such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$.
- We say g(n) is the **asymptotic tight** bound for f(n)
  - $f(n) = 1024\, n^2 + n \quad g(n)= n^2$
  - $f(n) = n^3 + n \log n \quad g(n)= n^3$
  - $f(n) = 1024^2 \quad g(n)= 1$
  - $f(n) = n + \log n = \Theta(n)\ O(n) \quad g(n)= n$



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

Little o notation

- Given functions f(n) and g(n),
- $f(n) = o(g(n))$ if for all $c > 0$ and $n_0 > 0$ $0 \le f(n) < cg(n)$ for all $n \ge n_0$.
- g(n) is an upper bound of f(n).
  - $f(n) = 2n \quad g(n) = n^2$
  - $f(n) = 1024n^2 \quad g(n) = n^3$
- Little o notation is an upper bound but not asymptotic tight.

Little ω notation

- Given functions f(n) and g(n),
- $f(n) = \omega(g(n))$ if for all $c>0$, $n_0>0$ exists s.t. $0 \le cg(n) < f(n)$ for all $n \ge n_0$.
- g(n) is a lower bound of f(n) but not asymptotic tight.
  - $f(n) = 1024\ n^2 + n \qquad g(n)= n$

SUMMARY

- f<=g: $f(n) = O(g(n))$: f is asymptoticly smaller than or equal to g
- f>=g: $f(n) = \Omega(g(n))$: f is asymptoticly larger than or equal to g
- f=g: $f(n) = \Theta(g(n))$: f is asymptoticly equivalent to g
- f<g: $f(n) = o(g(n))$: f is asymptoticly smaller than g
- f>g: $f(n) = \omega(g(n))$: f is asymptoticly larger than g

Notations and Limits

- **Big-O notation: g(n) is asymptoticly larger than or equal to f(n)**
  - $f(n) = O(g(n)) \Leftrightarrow \lim_{n \to +\infty} \frac{f(n)}{g(n)} \in [0,\infty).$ **E.g., (n+1, n^2), (1024n, n)**
- **Big-Ω notation: g(n) is asymptoticly smaller than or equal to f(n)**
  - $f(n) = \Omega(g(n)) \Leftrightarrow \lim_{n \to +\infty} \frac{f(n)}{g(n)} \in (0,\infty]$ **E.g., (n^2, 1024n+100), (10^{6}n-100, n)**
- **Big-Θ notation: g(n) is asymptoticly equal to f(n)**
  - $f(n) = \Theta(g(n)) \Leftrightarrow \lim_{n \to +\infty} \frac{f(n)}{g(n)} \in (0,\infty)$ **E.g., (4n^2+100n, n^2), (0.1n^3-100n^2, n^3)**
- **Little-o notation: g(n) is asymptoticly larger than f(n)**
  - $f(n) = o(g(n)) \Leftrightarrow \lim_{n \to +\infty} \frac{f(n)}{g(n)} = 0$ **E.g., (1024n+100, n^2)**
- **Little-ω notation: g(n) is asymptoticly smaller than f(n)**
  - $f(n) = \omega(g(n)) \Leftrightarrow \lim_{n \to +\infty} \frac{f(n)}{g(n)} = \infty$ **E.g., (1024n+100, n^2)**

CLASS DISCUSSION

- Is $2^{2n} = O(2^n)$?

$$2^{2n} = (2^2)^n = 2^n \cdot 2^n = 4^n$$

$$lim_{n=\infty} \frac{2^n \cdot 2^n}{2^n} = lim_{n=\infty} 2^n = \infty$$

$$2^{2n} = \Omega(2^n)$$

$$2^n = o(2^{2n}) = o(4^n)$$

$$4^n = o(5^n)$$

$$2^n < 4^n < 5^n$$

The Asymptotic Order of Functions

- Constants<Logarithmic < Polynomial < Exponential<Factorials
- For any constants a>0, b>0, c>1
  - $(\log n)^a < n^b < c^n < n!$

When n is very large,

$1 < \log n < \log^{1024} n < \mathrm{sqrt}(n) < n < n \log n < n^2 < n^2 \log n < n^3 < 2^n < n!$

Computing the increasing asymptotic order

$$10^6,\ 2^n,\ 2^{2n},\ 5^n,\ n!,\ \log^2 n,\ n^2,\ n\log n,\ n^{\frac{1}{3}},\ \sqrt{\log n},\ \log n,\ \log_3 n,\ \log\log n,\ 2^{1000}$$

$\log\log n < \sqrt{\log n} < \log n = \log_3 n < \log^2 n$

$10^6 = 2^{1000}$	<	$\log^2 n$, $\sqrt{\log n}$, $\log n$, $\log_3 n$, $\log\log n$	<	$n^{\frac{1}{3}} < n\log n < n^2$	<	$2^n < 2^{2n} < 5^n$	<	$n!$

$$\lim_{n\to\infty}\frac{\log\log n}{\sqrt{\log n}} = \lim_{n\to\infty}\frac{\frac{1}{\log n}\cdot\frac{1}{n}}{\log^{-0.5} n\cdot\frac{1}{n}} = \lim_{n\to\infty}\frac{1}{\sqrt{\log n}} = 0$$

$$10^6,\ 2^n,\ 2^{2n},\ 5^n,\ n!,\ \log^2 n,\ n^2,\ n\log n,\ n^{\frac{1}{3}},\ \sqrt{\log n},\ \log n,\ \log_3 n,\ \log\log n,\ 2^{1000}$$

$$10^6, 2^{1000} < \log\log n < \sqrt{\log n} < \log_3 n = \log n < \log^2 n < n^{\frac{1}{3}} < n\log n < n^2 < 2^n < 2^{2n} < 5^n < n!$$

$$(\log\log n)' = \frac{1}{\log n}\frac{1}{n} \qquad lim_{n=\infty}\frac{\sqrt{\log n}}{\log\log n} = lim_{n=\infty} \log n(\log n)^{-\frac{1}{2}} = \infty$$

$$\log_3 n = \frac{\log n}{\log 3} < \log n \qquad \log 3 > 1 \qquad \sqrt{\log n} = (\log n)^{\frac{1}{2}}$$

Floors and ceilings

- For any real number x, the floor of x, denoted by $\lfloor x \rfloor$, the greatest integer less than or equal to x.
  - $\lfloor 1.25 \rfloor = 1$; $\lfloor -10 \rfloor = -10$;
- The ceiling of x, denoted by $\lceil x \rceil$, is the least integer greater than or equal to x.
  - $\lceil 1.25 \rceil = 2$; $\lceil -10 \rceil = -10$;
-

Logarithms

- $\log n = \log_2 n$ **(binary logarithms)**
- $\ln n = \log_e n$ **(natural logarithms) e=2.71828…,**
- $\log^k n = (\log n)^k$ **(exponentiation)**
- $\log \log n = \log(\log n)$ **(composition)**
- **For all real a>0, b>0, c>0 and n,**
- $a = b^{\log_b a}$, $\log_c(ab) = \log_c a + \log_c b$, $\log_b a^n = n \log_b a$,

$$\log_b a = \frac{\log_c a}{\log_c b},\ \log_b(1/a) = -\log_b a,\ \log_b a = \frac{1}{\log_a b},\ a^{\log_b c} = c^{\log_b a}$$

$\lg^b n = o(n^a)$ **for any a>0**

Polynomials

- Given a nonnegative integer d, a polynomial in n of degree d is a function p(n) of the form
- $$p(n) = \sum_{i=0}^{d} a_i n^i$$
- where the constants $a_0, a_1, a_2, \cdots, a_d$ are the coefficients of the polynomial and $a_d \neq 0$.
- If $a_d > 0$, we say p(n) is asymptotic positive and have $p(n) = \Theta(n^d)$
- A function f(n) is polynomially bounded if $f(n) = O(n^k)$ for some constant k.

Exponentials

- **For all real a>0, m and n, we have**
- $a^0 = 1$
- $a^{-1} = 1/a$
- $(a^m)^n = a^{mn}$
- $(a^n)^m = (a^m)^n$
- $a^n a^m = a^{m+n}$
- $n^b = o(a^n)$ **for all real constants a>1 and b**

Factorials

- **n factorial is** $n! = 1 * 2 * 3 \cdots n$**.**
- $n! = o(n^n)$
- $n! = \omega(2^n)$
- $\log(n!) = \Theta(n \log n)$

Series

- **Geometric series:** $a^0, a^1, a^2, \cdots, a^n \ for \ a \neq 0$
    - **Summation of finite set:** $\sum_{i=0}^{n} a^i = \frac{a^{n+1} - 1}{a - 1}$
- **If 0<a<1 and n tends to be infinity, then**
- $$\sum_{i=0}^{\infty} a^i = \frac{1}{1 - a}$$

Series (cont.)

- **Arithmetic series:**
- $a_1, a_2, a_3, \cdots, a_n$ **where** $a_{i+1} - a_i = d$ **and** d **is a constant**
- **Summation of finite set:** $\sum_{i=0}^{n} a_i = \frac{n(a_1 + a_n)}{2}$

Series (cont.)

- Harmonic series: $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \cdots$
  - Summation of finite set: $\sum_{i=1}^{n} \frac{1}{i} \approx \log n$
- Others we may use
- $$\sum_{i=1}^{n} i^2 \approx \frac{n^3}{3}$$