DESIGN & ANALYSIS of ALGORITHMS

unit - 1

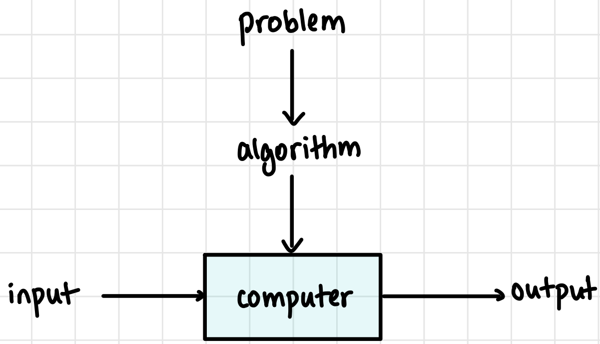
feedback/corrections: vibha@pesu.pes.edu

VIBHA MASTI

Design and Analysis of Algorithms

Algorithms

- sequence of unambiguous instructions for solving a problem, i.e for obtaining an output for a legitimate input in a finite amount of time



COMPUTATIONAL PROBLEMS

Sorting

Statement of Problem

- Input: a sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$
- Output: a reordering of the input sequence $\langle a_1, a_2, \ldots, a_n \rangle$ such that $a_i \leq a_j$ whenever $i < j$

Instance

- Sequence $\langle 5, 3, 2, 8, 3 \rangle$

Algorithms

- Selection sort, insertion sort, bubble sort
- Merge sort
- Quick sort
- Many more

SELECTION SORT

Algorithm

```
for i = 1 to n
    swap a[i] with smallest of a[i] ... a[n]
```

min

pass = 1 5, 3, 2, 8, 3

min

pass = 2 2, 3, 5, 8, 3

min

pass = 3 2, 3, 5, 8, 3

min

pass = 4 2, 3, 3, 8, 5

pass = 5 2, 3, 3, 5, 8

Code in C

```c
void selsort(int *a, int n) {
    int minpos, temp;
    for (int i = 0; i < n; ++i) {
        minpos = i;

        for (int j = i + 1; j < n; ++j) {
            if (a[j] < a[minpos]) {
                minpos = j;
            }
        }

        temp = a[minpos];
        a[minpos] = a[i];
        a[i] = temp;
    }
}
```

GCD

Statement of Problem

- **Input**: a pair of numbers (m, n)
- **Output**: the greatest common divisor of two non-negative integers m and n

EUCLID'S ALGORITHM

$$\boxed{\gcd(m,n) = \gcd(n, m \bmod n)}$$

until the second number becomes 0

eg: $\gcd(60,24) = \gcd(24,12) = \gcd(12,0) = 12$

Algorithm

```
euclid(m, n):
    while n ≠ 0
        r = m mod n
        m = n
        n = r
    return m
```

Code in C

```c
int gcd_euclid(int m, int n) {
    if (m < n) {
        int temp = m;
        m = n;
        n = temp;
    }

    while (n != 0) {
        int r = m % n;
        m = n;
        n = r;
    }
    return m;
}
```

CONSECUTIVE INTEGER CHECKING ALGORITHM

gcd(m,n)

Algorithm

```
t = n

while true
    r = m mod t
    if r = 0
        return t
    else
        t = t - 1

return 1
```

MIDDLE SCHOOL PROCEDURE

gcd (m,n)

Algorithm

- Find prime factorisation of m
- Find prime factorisation of n
- Find common prime factors
- Compute product and return it as gcd (m,n)

Example

gcd (32,24)

$32 = 2 \times 2 \times 2 \times 2 \times 2$
$24 = 2 \times 2 \times 2 \times 3$
$\text{gcd} = 2 \times 2 \times 2 = 8$

SIEVE OF ERATOSTHENES

to find prime numbers from 1 to n

Algorithm

```
for p = 2 to p = n
    A[p] = 1

for p = 2 to p = √n
    if A[p] ≠ 0
        j = p * p
        while j <= n
            A[j] = 0
            j = j + p
```

Example

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

$i=2$

1 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21 ~~22~~ 23 ~~24~~ 25

$i=3$

1 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ 25

$i=4$

1 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ 25

$i=5$

1 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ ~~25~~

More Computational Problems

1. Sorting
2. Searching
3. String processing
4. Graph problems
5. Combinatorial problems
6. Geometric problems
7. Numerical problems

- Data structures are key

issues related to algorithms

- design algorithms
- express algorithms
- proving correctness
- efficiency
  - theoretical analysis (analyse performance w/o implementation)
  - empirical analysis (measure after implementation – profiling)
- optimality

ALGORITHM DESIGN STRATEGIES

- brute force
- divide and conquer
- decrease and conquer
- transform and conquer
- greedy approach
- dynamic programming
- backtracking and branch and bound
- space and time tradeoffs

ANALYSIS OF ALGORITHMS

- lower bounds
- optimality
- correctness
- time efficiency
- space efficiency

APRIORI ANALYSIS

- analysis framework
- parameters: input size (order of matrix, length of array)
  nature of input (sorted ; best / worst case)
- resources: time and space
- identify basic operation and find number of operations
- $T(n) = C_{op} * C(n)$ ← no. of times basic op is performed

  execution time of basic op

  order of growth

average case

- eg: sequential search
- p = prob of finding in i^{th} pos (success)
- $t_{avg} = \left(\frac{1+2+3+\ldots+n}{n} \right) p + n(1-p)$

  avg no. of comparisons

  no of comparisons for failure

  $t_{avg} = (n+1)p + n(1-p) \approx n$

- NOT average of best & worst case
- we usually look at worst case, not average case

Basic Operations

Problem	*Input size measure*	*Basic operation*
Searching for key in a list of *n* items	Number of list's items, i.e. *n*	Key comparison
Multiplication of two matrices	Matrix dimensions or total number of elements	Multiplication of two numbers
Checking primality of a given integer *n*	*n*'size = number of digits (in binary representation)	Division
Typical graph problem	#vertices and/or edges	Visiting a vertex or traversing an edge

Values of Functions as $n \to \infty$

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 important for analysis of algorithms

Order of Growth

1. $O \rightarrow$ big O
2. $\Theta \rightarrow$ theta
3. $\Omega \rightarrow$ omega

notations

n	T1 algorithm 1 1000n	T2 algorithm 2 $10n^2$
1	1000	10
2	2000	40
4	4000	160
10	10^4	10^3
100	10^5	10^5
1000	10^6	10^7
10000	10^7	10^9

grows more rapidly

- coefficients and additive constants can be ignored

Simple Prime Checker

```
for i=2 to n-1
    if n%i ==0
        break
```

```
for i=2 to √n
    if n%i ==0
        break
```

n	$T \propto n$ n-2	$T \propto \sqrt{n}$ $\sqrt{n}$
11	9	2
101	99	9
10^6+3	10^6	10^3
$10^{10}+19$	10^{10}	10^5

primality

Asymptotic Analysis

- time complexity as $n \to \infty$
- succinct function $g(n)$ for fair idea of order of growth
- $O(g(n))$ is the set of all functions with smaller or same order of growth as $c \cdot g(n)$
- eg: $t(n) = 6n+5$

if $t(n) \leq c(g(n))$ then $t(n) \in O(g(n))$

$t(n) \in O(n^2)$ ✓

$t(n) \in O(n)$ ✓ ← most accurate

$t(n) \in O(2^n)$ ✓

upper bound representation

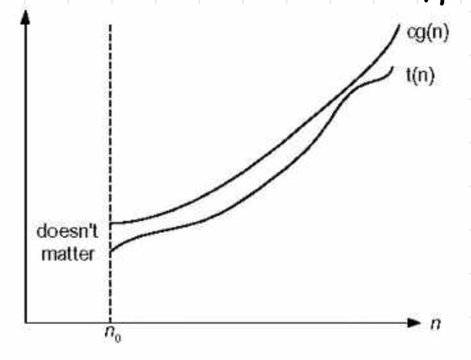


Figure 2.1 Big-oh notation: $t(n) \in O(g(n))$

max. time representation

- $\Omega(g(n))$ is lower bound

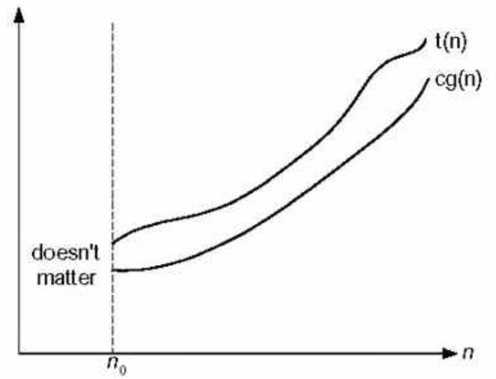


Fig. 2.2 Big-omega notation: $t(n) \in \Omega(g(n))$

- $\Theta(g(n))$ is same bound

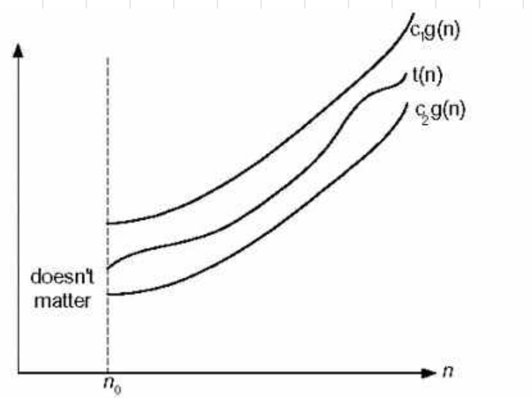


Figure 2.3 Big-theta notation: $t(n) \in \Theta(g(n))$

same asymptotic order
g(n) and t(n)

O-Notation

$f(n)$ is $O(g(n))$, denoted by $f(n) \in O(g(n))$ if the order of growth of $f(n) \leq$ order of growth of a constant multiple of $g(n)$

there exists a positive constant c and a non-negative number n_0 such that

$$f(n) \leq c\, g(n) \text{ for all } n \geq n_0$$

Ω- Notation

$t(n)$ is $\Omega(g(n))$, denoted by $t(n) \in \Omega(g(n))$ if the order of growth of $t(n) \geq$ order of growth of a constant multiple of $g(n)$

there exists a positive constant c and a non-negative number n_0 such that

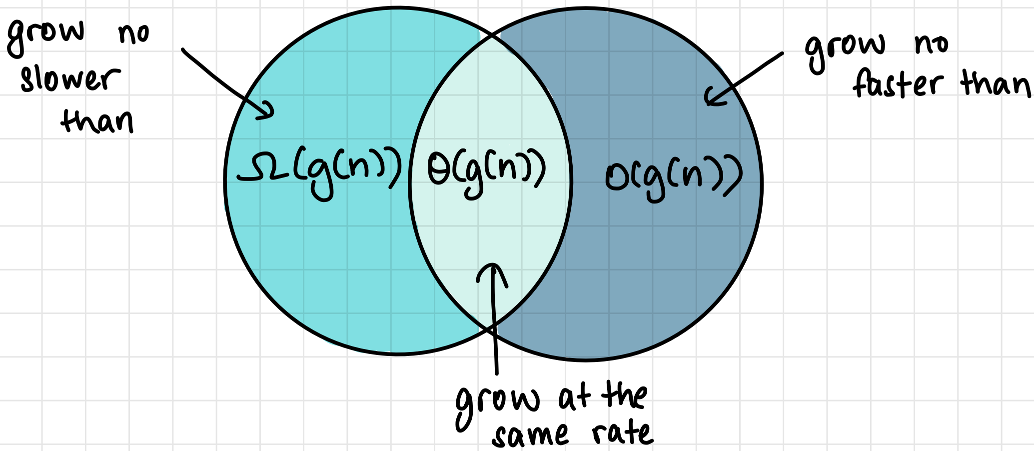
$$t(n) \geq c\, g(n) \text{ for all } n \geq n_0$$

θ- Notation

$t(n)$ is $\Theta(g(n))$, denoted by $t(n) \in \Theta(g(n))$ if $t(n)$ is bounded both above and below by some positive constant multiples of $g(n)$ for all large n

there exist some positive constants c_1 and c_2 and some nonnegative number n_0 such that

$$c_2\, g(n) \leq t(n) \leq c_1\, g(n) \text{ for all } n \geq n_0$$



TRIAL & ERROR

Q: $12n^2+8 \in O(n^2)$

$f(n)$ $\quad$ $g(n)$

$f(n) \le c\,g(n) \ \forall\ n \ge n_0$

$$12n^2+8 \le 12n^2+8n^2$$

not for $n_0=0$

$n_0 = 1, 2, \ldots$

$c=20$

$n_0=1$

$$12n^2+8 \in O(n^2)$$

Q: $100n+5 \in O(n)$

$$100n+5 \le 100n+n$$
$$\le 101n$$

$n_0 = 5, 6, \ldots$

$c=101$

$n_0=5$

Q: $13n^2 + n + 5 \in O(n^2)$

$13n^2 + n + 5 \leq 13n^2 + n^2 + 5n^2$, $n_0 = 1, 2, \ldots$

$13n^2 + n + 5 \leq 19n^2$

$n_0 = 1$
$c = 19$

Q: $n^3 \in \Omega(n^2)$

$f(n) \geq c\,g(n)$

$n^3 \geq n^2$ for $n_0 = 0, 1 \ldots$

$c = 1$
$n_0 = 0$

Q: $n^2 + n \in \Theta(n^2)$

$n^2 \leq n^2 + n$ $n_0 = 0$

$n^2 + n \leq n^2 + n^2$ $n_0 = 0$

$c_2 = 1$ $c_1 = 2$ $n_0 = 0$

Q: $\frac{n}{100} \in \Omega(n)$

$\frac{n}{100} \geq \frac{n}{200}$ $n_0 = 0$
$c = \frac{1}{200}$

Q: $6n^2 - 8n \in \Theta(n^2)$

$6n^2 - 8n \leq 6n^2$ $\quad n_0 = 0$, $c_1 = 6$

$6n^2 - 8n \geq 6n^2 - \frac{8n^2}{8}$ $\quad n_0 = 0$, $c_2 = 5$

$6 - 8 \geq 6 - 1$

theorem

if $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, then

$$t_1(n) + t_2(n) \in O(\max(g_1(n), g_2(n)))$$

the analogous assertions are true for Ω-notation and Θ-notation

- eg: $5n^2 + 3n\log n \in O(n^2)$

Proof

$t_1(n) \leq c_1 g_1(n)$, $n \geq n_1$ $\longrightarrow$ ①

$t_2(n) \leq c_2 g_2(n)$, $n \geq n_2$ $\longrightarrow$ ②

① + ②

$$\begin{aligned} t_1(n) + t_2(n) &\leq c_1 g_1(n) + c_2 g_2(n) \\ &\leq c_3 (g_1(n) + g_2(n)) \\ &\leq 2c_3 \max(g_1(n), g_2(n)) \end{aligned}$$

$c_3 = \max(c_1, c_2)$

$n_3 = \max(n_1, n_2)$

$$\boxed{\begin{aligned} c &= 2c_3 \\ n_0 &= n_3 \end{aligned}}$$

$\therefore t_1(n) + t_2(n) \in O(\max(g_1(n), g_2(n)))$

Binary Search

- prerequisites
  - constant random access time
  - sorted elements in an array

- good sorting algorithm: $n \log n$
- searching: $\log n$

$\therefore$ binary search $\in O(n \log n)$

limits theorem

$$\lim_{n \to \infty} \frac{t(n)}{g(n)} = \begin{cases} 0, & \text{order of growth of } t(n) < g(n) \\ c > 0, & \text{order of growth of } t(n) = g(n) \\ \infty, & \text{order of growth of } t(n) > g(n) \end{cases}$$

- little-o notation

strict inequality

$t(n) < c\, g(n)$, $t(n) \in o(g(n))$

L'hôpital's Rule

if $\lim_{n \to \infty} f(n) = \lim_{n \to \infty} g(n) = \infty$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{f'(n)}{g'(n)}$$

Stirling's Formula

$$n! \approx (2\pi n)^{1/2} \left(\frac{n}{e}\right)^n$$

Q: compare growth rates of $\underset{t(n)}{\log n}$ and $\underset{g(n)}{n}$

$$\lim_{n\to\infty} \frac{\log n}{n} = \frac{1/n}{1} = \frac{1}{n} = 0$$

$$\therefore \quad t(n) < g(n)$$

$$\log n < n$$

$$\log n \in o(n)$$

Q: compare growth rates of $\underset{t(n)}{\frac{1}{2}n(n-1)}$ vs $\underset{g(n)}{n^2}$

$$\lim_{n\to\infty} \frac{\cancel{n}(n-1)/2}{n^{\cancel{2}}} = \lim_{n\to\infty} \frac{(n-1)}{2n} = \lim_{n\to\infty} \frac{1}{2} - \frac{1}{2n}$$

$$= \frac{1}{2} > 0$$

small θ

$$\therefore \quad \frac{1}{2}n(n-1) \in \theta(n^2)$$

Q: $\log n$ vs $\sqrt{n}$
$t(n)$ $g(n)$

$$\lim_{n \to \infty} \frac{\log n}{\sqrt{n}} = \lim_{n \to \infty} \frac{1/n}{1/2\sqrt{n}} = \lim_{n \to \infty} \frac{2\sqrt{n}}{n}$$

$$= \lim_{n \to \infty} \frac{2}{\sqrt{n}} = 0$$

small o

$$\log n \in o(\sqrt{n})$$

Q: $(n^2+1)^{10} = t(n)$, find order of growth

Assume $g(n) = n^{20}$

$$\lim_{n \to \infty} \frac{(n^2+1)^{10}}{n^{20}} = \lim_{n \to \infty} \frac{\cancel{n^{20}}(1 + 1/n^2)^{10}}{\cancel{n^{20}}} = \lim_{n \to \infty} \left(1 + \frac{1}{n^2}\right)^{10}$$

$$= 1 > 0$$

small θ

$$\therefore (n^2+1)^{10} \in \theta(n^{20})$$

Q: Find order of growth of $\sqrt{10n^2 + 7n + 3} = t(n)$

Assume $g(n) = n$

$$\lim_{n\to\infty} \frac{\sqrt{10n^2+7n+3}}{n} = \lim_{n\to\infty} \frac{\cancel{n}\sqrt{10+7/n+3/n^2}}{\cancel{n}}$$

$$= \sqrt{10} > 0$$

small θ

$$\therefore \sqrt{10n^2+7n+3} \in \theta(n)$$

Q: Find order of growth of $2^{n+1} + 3^{n-1} = t(n)$

$$g(n) = 3^n$$

fraction

$$\lim_{n\to\infty} \frac{2^{n+1}+3^{n-1}}{3^n} = \lim_{n\to\infty} 2\times\left(\frac{2}{3}\right)^n + \frac{3^n}{3\times 3^n}$$

$$= 0 + \frac{1}{3} > 0$$

$$\therefore 2^{n+1}+3^{n-1} \in \theta(3^n)$$

Q: $\underset{t(n)}{2^n}$ vs $\underset{g(n)}{n!}$

Stirling's formula: $n! \approx (2\pi n)^{1/2}\left(\frac{n}{e}\right)^n$

$$\lim_{n\to\infty} \frac{2^n}{n!} = \lim_{n\to\infty} \frac{2^n e^n}{(2\pi)^{1/2}\sqrt{n}\; n^n} = \lim_{n\to\infty} \frac{(2e/n)^n}{\sqrt{2\pi}\;\sqrt{n}}$$

$$= \lim_{n\to\infty} \frac{1}{\sqrt{2\pi}}\left(\frac{2e}{n}\right)^n\left(\frac{1}{n}\right)^{1/2} = 0\times 0$$

$\therefore\ 2^n \in o(n!)$

Q: $\underbrace{3n^2 3^n + n\log n}_{t(n)} \in \Theta(\underbrace{n^2 3^n}_{g(n)})$

$$\lim_{n\to\infty} \frac{3n^2 3^n + n\log n}{n^2 3^n} = \lim_{n\to\infty} 3 + \frac{\cancel{n}\log n}{n^{\cancel{2}} 3^n}$$

$$= 3 + \lim_{n\to\infty} \frac{\log n}{n 3^n} = 3 + \lim_{n\to\infty} \frac{1/n}{3^n + n3^n\log 3}$$

$$= 3 + \lim_{n\to\infty} \frac{1}{n3^n + n^2 3^n n\log 3} = 3 > 0$$

$\therefore\ 3n^2 3^n + n\log n \in \Theta(n^2 3^n)$

Q: $n^2+n \in O(n^3)$ using trial-error & limits

$g(n) = n^3$
$t(n) = n^2+n$

trial-error

$n^2+n \le n^2+n\cdot n^2$
$n^2+n \le n^2\cdot n + n^3$
$n^2+n \le 2n^3$

$n_0=0$
$c=2$

limits

$$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \lim_{n\to\infty} \frac{n^2+n}{n^3}$$

$$= \lim_{n\to\infty} \frac{1}{n} + \frac{1}{n^2} = 0$$

$n^2+n \in o(n^3) \Rightarrow n^2+n \in O(n^3)$

Q: $n^3+4n^2 \in \Omega(n^2)$

$\quad t(n) \qquad g(n)$

trial-error

$$n^3+4n^2 \geq 4n^2$$

$n_0 = 0$
$c = 4$

limits

$$\lim_{n\to\infty} \frac{n^3+4n^2}{n^2} = \lim_{n\to\infty} n+4 = \infty$$

$\therefore\ n^3+4n^2 \in \omega(n^2)$

$\Rightarrow n^3+4n^2 \in \Omega(n^2)$

PROPERTIES of ASYMPTOTIC ORDER of GROWTH

1. $f(n) \in O(f(n))$
2. $f(n) \in O(g(n))$ iff $g(n) \in \Omega(f(n))$
3. if $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$, then $f(n) \in O(h(n))$ transitivity
4. if $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then $f_1(n)+f_2(n) \in O(\max(g_1(n), g_2(n)))$
5. $\sum_{i=1}^{n} \Theta(f(i)) = \Theta\left(\sum_{i=1}^{n} f(i)\right)$
6. $f(n) \in \Theta(f(n))$, $\Omega(f(n))$ and $O(f(n))$ reflexivity

ORDER OF GROWTH OF IMPORTANT FUNCTIONS

1. All log functions belong to same class

$$\log_a n = \frac{\log_b n}{\log_b a} = \frac{\log_c n}{\log_c a} = \frac{\log n}{k}$$

$$\Theta(\log n)$$

2. All polynomials of same degree k belong to same class

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_0 \in \Theta(n^k)$$

3. Exponential functions a^n have different orders of growth for different a's

4. $$\underbrace{O(\log n) < O(n^\alpha)\ (\alpha > 0)}_{\substack{\text{polynomial time complexity} \\ \text{(P)} \\ \text{tractable (t)} \\ \text{can be implemented meaningfully}}} < \underbrace{O(a^n) < O(n!) < O(n^n)}_{\substack{\text{non-deterministic polynomial} \\ \text{(NP)} \\ \text{non-tractable (Nt)}}}$$

Efficiency of Non-Recursive Algorithms

Analysis

1. Decide on parameter n - input size
2. Identify basic operation
3. Determine best, average and worst cases for input size n
4. Sum for no. of times basic operation is executed
5. Simplify sum using formulas and rules

Summation Formulas & Rules

1. $$\sum_{i=l}^{u} 1 = 1+1+\dots+1 = u-l+1$$

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

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

3. $$\sum_{i=1}^{n} i^2 = 1^2+2^2+\dots+n^2 = \frac{n(n+1)(2n+1)}{6}$$

4. $$\sum_{i=0}^{n} a^i = 1+a+a^2+\dots+a^n = \frac{a^{n+1}-1}{(a-1)}$$

$$\sum_{i=0}^{n} 2^i = 1 + 2 + 2^2 + \cdots + 2^n = 2^{n+1} - 1 \in \Theta(2^n)$$

5. $$\sum_{i=l}^{u} a_i = \sum_{i=l}^{m} a_i + \sum_{i=m+1}^{u} a_i$$

6. $$\sum (a_i \pm b_i) = \sum a_i \pm \sum b_i$$

7. $$\sum c a_i = c \sum a_i$$

Q: Find efficiency of given program – max element

ALGORITHM *MaxElement*($A[0..n-1]$)

//Determines the value of the largest element in a given array
//Input: An array $A[0..n-1]$ of real numbers
//Output: The value of the largest element in A
maxval ← $A[0]$
for $i \leftarrow 1$ **to** $n-1$ **do** ← no of iterations
  if $A[i] >$ *maxval* ← comparism is basic op
    maxval ← $A[i]$
return *maxval*

$$t(n) = \sum_{i=1}^{n-1} 1$$ ← basic operation

$$t(n) = n-1-1+1 = n-1$$

$$\lim_{n \to \infty} \frac{n-1}{n} = 1 \Rightarrow n-1 \in \Theta(n)$$

Unique elements

Q: **ALGORITHM** *UniqueElements*($A[0..n-1]$)

```
//Determines whether all the elements in a given array are distinct
//Input: An array A[0..n − 1]
//Output: Returns "true" if all the elements in A are distinct
//        and "false" otherwise
for i ← 0 to n − 2 do
    for j ← i + 1 to n − 1 do
        if A[i] = A[j] return false
return true
```

comparism basic operation

$$t(n) = \sum_{i=0}^{n-2} \left(\sum_{j=i+1}^{n-1} 1 \right)$$

$$= \sum_{i=0}^{n-2} n-1-i \cancel{-1} \cancel{+1}$$

$$= \sum_{i=0}^{n-2} n-1-i$$

$$= n-1 + n-2 + \cdots + 1$$

$$= \frac{(n-1)(n)}{2} = \frac{n^2-n}{2}$$

$$\lim_{n \to \infty} \frac{n^2-n}{2n^2} = \lim_{n \to \infty} \frac{1}{2} - \frac{1}{2n} = \frac{1}{2} \Rightarrow \frac{n(n-1)}{2} \in \Theta(n^2)$$

Matrix Multiplication

Q: **ALGORITHM** *MatrixMultiplication*$(A[0..n-1, 0..n-1], B[0..n-1, 0..n-1])$

```
//Multiplies two n-by-n matrices by the definition-based algorithm
//Input: Two n-by-n matrices A and B
//Output: Matrix C = AB
for i ← 0 to n − 1 do
    for j ← 0 to n − 1 do
        C[i, j] ← 0.0
        for k ← 0 to n − 1 do
            C[i, j] ← C[i, j] + A[i, k] * B[k, j]
return C
```

multiply and add: basic op

$$t(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n$$

$$= \sum_{i=0}^{n-1} n^2$$

$$= n^3 \in \Theta(n^3)$$

Gaussian Elimination

Q: **ALGORITHM** *GaussianElimination*$(A[0..n-1, 0..n])$

```
//Implements Gaussian elimination of an n-by-(n+1) matrix A
for i ← 0 to n - 2 do
    for j ← i + 1 to n - 1 do
        for k ← i to n do
            A[j,k] ← A[j,k] - A[i,k] * A[j,i] / A[i,i]
```

basic op

Find the efficiency class and a constant factor improvement.

$$t(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=i}^{n} 1 = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} n-i+1$$

$$= \sum_{i=0}^{n-2} (n-1-i-\cancel{1}+\cancel{1})(n-i+1)$$

$$= \sum_{i=0}^{n-2} (n-i)^2-1 = \sum_{i=0}^{n-2} n^2 -2in + i^2 - 1$$

$$= \sum_{i=0}^{n-2} n^2-1 \; - \; 2n\sum_{i=0}^{n-2} i \; + \; \sum_{i=0}^{n-2} i^2$$

$$= (n-1)(n^2-1) - \frac{\cancel{2}n(n-2)(n-1)}{\cancel{2}} + \frac{(n-2)(n-1)(2n-3)}{6}$$

$$= \underbrace{n^3-n-n^2+1}_{\Theta(n^3)} - \underbrace{n(n^2-3n+2)}_{\Theta(n^3)} + \underbrace{\frac{(n^2-3n+2)(2n-3)}{6}}_{\Theta(n^3)}$$

$$= \Theta(n^3)$$

Q: Counting binary digits required for decimal no.

ALGORITHM *Binary(n)*

//Input: A positive decimal integer n
//Output: The number of binary digits in n's binary representation
$count \leftarrow 1$
while $n > 1$ **do** ← $\log_2 n$ times
  $count \leftarrow count + 1$ → basic op
  $n \leftarrow \lfloor n/2 \rfloor$
return $count$

$16 \to 8 \to 4 \to 2 \to 1$

halving effect

$$t(n) = \Theta(\log n)$$

Efficiency of Recursive Algorithms

Analysis

1. Decide on parameter n - input size
2. Identify basic operation
3. Determine best, average and worst cases for input size n (no. of times basic op is executed varies on different inputs of the same size)
4. Set up recurrence relation
5. Solve recurrence by backward substitution, forward substitution, Master's Theorem

preferred — limited use

Q: Factorial

```
ALGORITHM F(n)
    //Computes n! recursively
    //Input: A nonnegative integer n
    //Output: The value of n!
    if n = 0 return 1
    else return F(n − 1) ∗ n
```

$$f(n) = f(n-1) * n$$

basic op: no. of multiplications

$$M(n) = M(n-1) + 1$$] recurrence relation

Forward Substitution

0

1

2

$\vdots$

n

Backward Substitution

n-1

n-2

$\vdots$

0

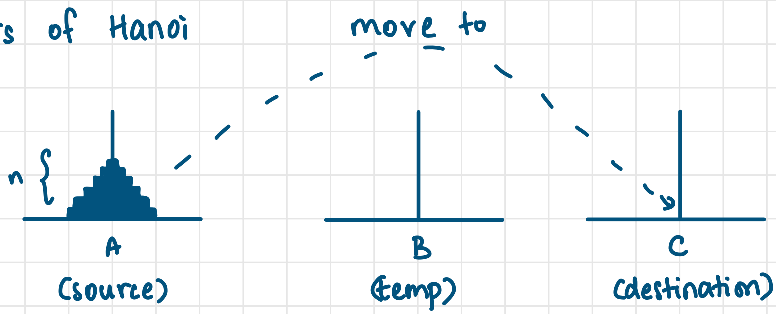
Initial condition

$$M(0) = 0$$

$$\begin{aligned} M(n) &= M(n-1) + 1 \\ &= M(n-2) + 2 \\ &= M(n-3) + 3 \\ &\vdots \\ &= M(0) + n \end{aligned}$$

$$M(n) \in \Theta(n)$$

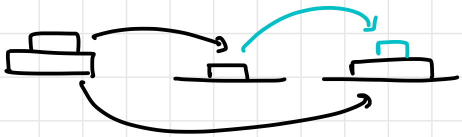
Q: Towers of Hanoi



```
Algorithm: Hanoi(n, s, t, d)

    if n = 1
        move disk from s to d
        return

    Hanoi(n-1, s, d, t)
    move disc from s to d
    Hanoi(n-1, t, s, d)
```



Recurrence relation: no. of moves

$$M(n) = M(n-1) + 1 + M(n-1)$$
$$= 2M(n-1) + 1 \qquad , n > 1$$

Termination condition: $M(1) = 1$

$$M(n) = 2M(n-1) + 1$$
$$= 2(2M(n-2) + 1) + 1$$
$$= 2^2 M(n-2) + 2 + 1$$
$$= 2^2(2M(n-3) + 1) + 2 + 1$$
$$= 2^3 M(n-3) + 2^2 + 2 + 1$$

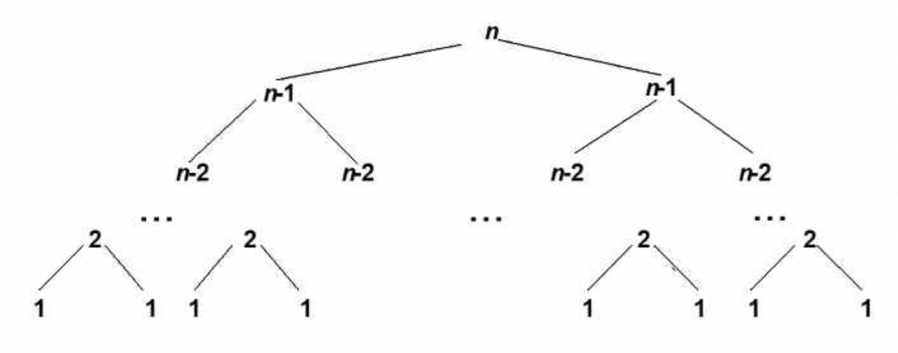
$$= 2^i M(n-i) + 2^{i-1} + 2^{i-2} + \dots + 2 + 1$$

for $i = n-1$

$$= 2^{n-1} M(1) + 2^{n-2} + 2^{n-3} + \dots + 2 + 1$$
$$= 2^{n-1} M(1) + 2^{n-1} - 1$$
$$= 2^n - 1$$

$M(n) \in \Theta(2^n) \leftarrow$ exponential

Recursive tree (2 trees)



Q: Counting binary digits required for decimal no. (recursive)

ALGORITHM *BinRec*(n)

//Input: A positive decimal integer n
//Output: The number of binary digits in n's binary representation
if $n = 1$ **return** 1
else return $BinRec(\lfloor n/2 \rfloor) + 1$

Recurrence relation: number of divisions

$$A(n) = A(\lfloor n/2 \rfloor) + 1$$

Termination condition $\quad n = 2^k$

$A(2^0) = 1 \quad k = \log_2 n$

$$
\begin{aligned}
A(2^k) &= A(2^{k-1}) + 1 \\
&= A(2^{k-2}) + 2 \\
&= A(2^{k-i}) + i \\
i = k \quad &= A(2^0) + k \\
&= 1 + k
\end{aligned}
$$

$$A(2^k) = 1+k$$

$$A(n) = 1 + \log_2 n \in \Theta(\log n)$$

Using smoothness relation, $n =$ any value

$$A(n) = 1 + \log_2 n \in \Theta(\log n) \quad \forall n$$

Q: $x(n) = 3x(n-1)$, $x(1) = 4$

$$\begin{aligned} x(n) &= 3\,x(n-1) \\ &= 3^2\,x(n-2) \\ &= 3^i\,x(n-i) \end{aligned}$$

$$\begin{aligned} i = n-1 \qquad &= 3^n\,x(1) \\ &= 3^n \cdot 4 \end{aligned}$$

$$x(n) \in \Theta(3^n)$$

Q: $x(n) = x(n/2) + n$, $x(1) = 1$, $n = 2^k$

$$x(2^0) = 1$$

$$\begin{aligned} x(2^k) &= x(2^{k-1}) + 2^k \\ &= x(2^{k-2}) + 2^k + 2^{k-1} \\ &= x(2^{k-i}) + 2^k + 2^{k-1} + \dots + 2^{k-i+1} \end{aligned}$$

$$\begin{aligned} i = k \qquad &= x(2^0) + 2^k + 2^{k-1} + \dots + 2 \\ &= x(2^0) + 2(2^k - 1) \end{aligned}$$

$$x(n) = 1 + 2(n-1)$$
$$= 2n - 1$$

$$x(n) \in \Theta(n)$$

Decrease by One Recurrence Relation

$$T(n) = T(n-1) + \text{something}$$

Decrease by Factor Recurrence Relation

$$T(n) = a\,T(n/b) + \text{something}$$

MASTER'S THEOREM

non-decreasing function

$$T(n) = a\,T(n/b) + f(n) \quad , \quad n = b^k \quad , \; k = 1, 2, 3 \ldots$$

$$T(1) = c \quad , \quad a \geq 1 \quad b \geq 2 \quad c > 0$$

if $f(n) \in \Theta(n^d)$ where $d \geq 0$

Solution:

$$T(n) = \begin{cases} \Theta(n^d) & a < b^d \\ \Theta(n^d \log n) & a = b^d \\ \Theta(n^{\log_b a}) & a > b^d \end{cases}$$

Q: $x(n) = x(n/2) + n$, $x(1) = 1$, $n = 2^k$ using Master's Theorem

$$x(n) = 1 \cdot x(n/2) + n \qquad a = 1,\ b = 2,\ c = 1$$
$$x(1) = 1$$

$$n \in \Theta(n) \qquad d = 1$$

$\therefore$ applying Master's Theorem

$$a = 1 \qquad b^d = 2 \qquad \therefore\ a < b^d$$

$$\boxed{x(n) \in \Theta(n)}$$

Q: $x(n) = x(n/3) + 1$
$x(1) = 1$

Master's Theorem

$a = 1 \quad b = 3 \quad d = 0 \quad c = 1$

$a = b^d$
$1 = 1$

$\therefore\ x(n) \in \Theta(n^0 \log n)$

$x(n) \in \Theta(\log n)$

Backward substitution

let $n = 3^k \Rightarrow k = \log_3 n$

$x(3^k) = x(3^{k-1}) + 1$
$x(3^0) = 1$

$$\begin{aligned} x(3^k) &= x(3^{k-1}) + 1 \\ &= x(3^{k-2}) + 1 + 1 \\ &= x(3^{k-3}) + 1 + 1 + 1 \\ &= x(3^{k-i}) + i \end{aligned}$$

for $i = k$, $x(3^k) = x(3^0) + k$
$x(n) = 1 + \log_3 n$

$x(n) \in \Theta(\log n)$

Q: Algorithm S(n)

if n=1 return 1

else return S(n-1) + n*n*n

(i) Recurrence relation?
(ii) Order of growth?

(i) $M(n) = M(n-1) + 2$ ← basic operation: multiplication

(ii) $M(n) = M(n-1) + 2$
$= M(n-2) + 2 + 2$
$= M(n-3) + 2 + 2 + 2$
$= M(n-i) + 2i$

for $i = n-1$
$= M(1) + 2(n-1)$
$= 1 + 2n - 2$
$= 2n - 1$

$\therefore M(n) \in \Theta(n)$

Q: Fibonacci Numbers

$F(n) = F(n-1) + F(n-2)$ for $n > 1$
$F(0) = 0$
$F(1) = 1$

- cannot solve with simple recurrence relation
- Homogeneous second order linear recurrence relation

HOMOGENEOUS SECOND ORDER LINEAR EQUATION

- with constant coefficients

$$a\,x(n) + b\,x(n-1) + c\,x(n-2) = 0$$

- to solve, characteristic equation must be solved:

$$ar^2 + br + c = 0$$

based on roots r

if r are real & distinct: $F(n) = \alpha r_1^n + \beta r_2^n$

- For Fibonacci Numbers

$$F(n) - F(n-1) - F(n-2) = 0$$

$a = 1$
$b = -1$
$c = -1$

- characteristic equation

$$r^2 - r - 1 = 0$$

$$r = \frac{1 \pm \sqrt{1^2 + 4}}{2}$$

$$r = \frac{1 \pm \sqrt{5}}{2} \longrightarrow \text{real \& distinct}$$

$$F(n) = \alpha\left(\frac{1+\sqrt{5}}{2}\right)^n + \beta\left(\frac{1-\sqrt{5}}{2}\right)^n$$

$$F(0) = 0 = \alpha + \beta \Rightarrow \alpha = -\beta$$

$$F(1) = 1 = \alpha\left(\frac{1+\sqrt{5}}{2}\right) + \beta\left(\frac{1-\sqrt{5}}{2}\right)$$

$$1 = -\beta\left(\frac{1+\sqrt{5}}{2}\right) + \beta\left(\frac{1-\sqrt{5}}{2}\right)$$

$$= \beta\left(\frac{-1-\sqrt{5}+1-\sqrt{5}}{2}\right) = \beta\,(-\sqrt{5})$$

$$\beta = \frac{-1}{\sqrt{5}} \qquad \alpha = \frac{1}{\sqrt{5}}$$

$$F(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

$$= \frac{1}{\sqrt{5}}\phi^n - \frac{1}{\sqrt{5}}\hat{\phi}^n$$

$$\boxed{F(n) \in \Theta(\phi^n)} \qquad \phi = \frac{1+\sqrt{5}}{2}$$

Basic Operation: # of Additions

$$A(n) = A(n-1) + A(n-2) + 1$$

$A(n) - A(n-1) - A(n-2) = 1$ ← inhomogeneous

$$[A(n)+1] - [A(n-1)+1] - [A(n-2)+1] = 0$$

$$B(n) = A(n)+1$$

$$B(n) - B(n-1) - B(n-2) = 0 \leftarrow \text{homogeneous}$$

ALGEBRAIC functions

Algebraic Structure

- set S together with zero or more operations, each of which is a function from $S^k \to S$ where k is arity
- groups, rings, fields, vector spaces

Groupoid

- $(S, \oplus)$ if S is closed under $\oplus$
- eg: $(N, +)$, $(Z, -)$, $(Q, *)$ etc.

Group

- $(S, \oplus)$ if following properties hold
  - closure
  - identity
  - associativity
  - inverse

Abelian group

- group $(S, \oplus)$ that also satisfies the commutative property

Ring

- set with 2 binary operations + and x, satisfying the following properties

  1. $(R,+)$ is a commutative group

  2. x is associative

  3. Distributive law holds in R

     $$(a+b)\times c = a\times c + a\times b$$

Eg:

- integer rings
- matrix rings
- polynomial rings

FIELD

- set with 2 binary operations + and x, satisfying the following properties

  1. $(F,+)$ is a commutative group

  2. $(F-\{0\},\times)$ is a commutative group

  3. Distributive law holds in F

     $$(a+b) \quad = a\times c + a\times b$$