

An algorithm is a sequence of ~~an~~ unambiguous steps (instruction) for solving a problem to obtain a required output for any legitimate input in a finite amount of time.

Important points illustrating motion of algorithm

- (i) The non-ambiguity required for each step of an algorithm cannot be compromised.
- (ii) The range of inputs for which an algorithm works, has to be specified carefully.
- (iii) The same algorithm can be represented several ways.
- (iv) Several algorithms for solving same problem can exist.
- (v) Algorithm for same problems can be based on various different ideas and can solve the problem with dramatically different speeds.

- 1 Buying the ticket for a p
- 2 Going to the cinema hall
- 3 Searching for the seat.
- 4 Watching the movie

1 Read m and n

2 while ( $n \neq 0$ )

    1  $r = m \bmod n$

    2  $m = n$

    3  $n = r$

3 ~~print~~ Display m

1 Read m and n

2 Till n is not equal to zero

    1 assign  $m \bmod n$  to  $r$

    2 assign n to m

    3 assign r to n

3 Display m

Algorithm (Pseudo Code) Euclid (m, n)

# Computes gcd by Euclid's algorithm.

# Input : Two non-negative integers, both non-zero

# Output : gcd of two integer i.e m & n.

while ( $n \neq 0$ )

$r \leftarrow m \bmod n$

$m \leftarrow n$

$n \leftarrow r$

return m

## FUNDAMENTALS OF ALGORITHMIC PROBLEM SOLVING

Understanding the Problem

Decide on:

- Computational needs

- exact vs approximate solving

- Data Structure

- Algorithm design technique

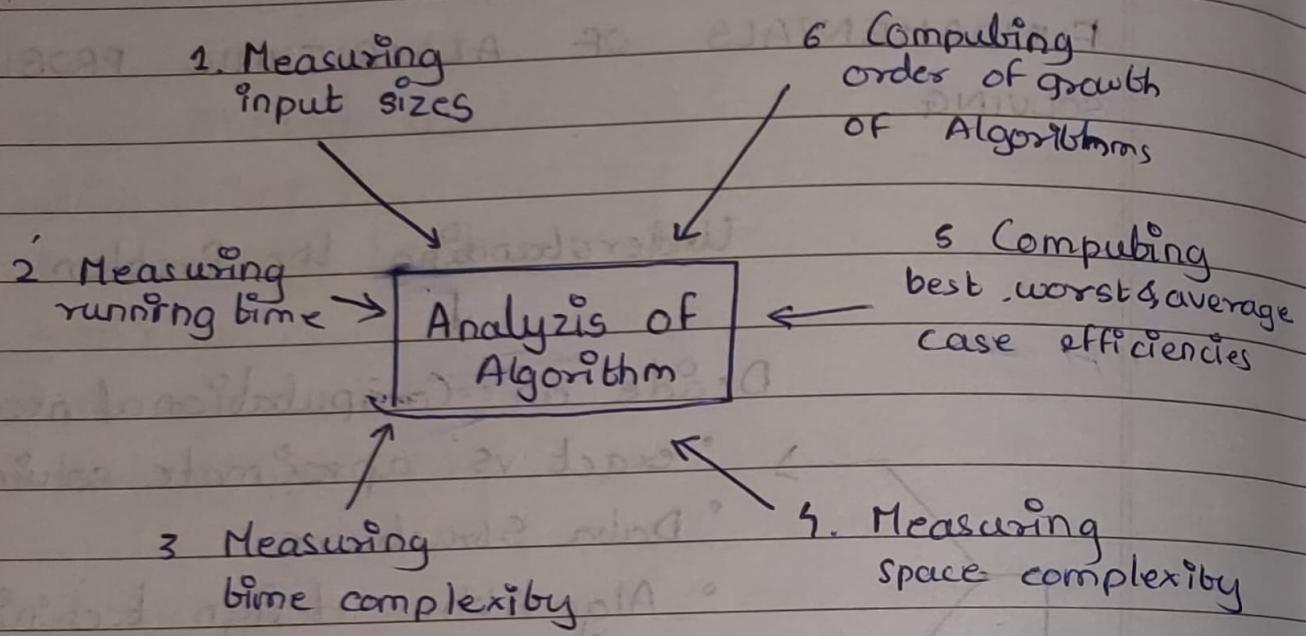
Design of algorithm

Prove Correctness

Analyze the algorithm

Code the algorithm

ANALYSIS FRAMEWORK



### Measuring Space Complexity

Algorithm add ( $x, n$ )

$sum \leftarrow 0$

$sum = sum + x[i]$

return sum

$$S(P) = c + S_p$$

where,

$S(P)$  → space complexity of program P

$c$  → constants for inputs and outputs

$S_p$  → instance characteristic of program  
(number of element in a list or array)

$$\text{sum} = 1$$

$$i = 1$$

$$n = 1$$

$$3 = s_p$$

$$s(p) = c + s_p$$

$$= n + 3$$

Measuring Time complexity:

for ( $i=0$  :  $i \leq n$ ;  $i++$ )

- $\text{sum} = \text{sum} + a[i];$

Executable Statement

$i \leftarrow 0$

1

$i < n$

$n+1$

$i++$

$n$

$\text{sum} += a[i]$

$n$

$$3n + 2 \approx 3n$$

Frequency Count

$$\text{Complexity} = n [O(n)]$$

Measuring running Time

$$T(n) = C_o p * C(n)$$

$T(n) \rightarrow$  Running time of an algorithm.

$C_o p \rightarrow$  Time taken by basic operation to execute

$C(n) \rightarrow$  number of times the operation needs to be executed.

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

$$\frac{T(2n)}{T(n)} = \frac{\text{Cop} * ((2n)^2 / 2)}{\text{Cop} * (n^2 / 2)}$$

$\text{Cop} \leftarrow C(n)$

$$\frac{T(2n)}{T(n)} = \text{Algorithm's unit performance}$$

Computing order of growth

Measuring the performance of an algorithm in relation with input size

$n$	$\log n$	$n$	$n \log n$	$n^2$	$n^3$	$2^n$	$n!$
1	0	$2^0$	0	$1^2$	$1^3$	2	1
2	1	$2^1$	2	$2^2$	$2^3$	4	2
4	2	$2^2$	8	$2^4$	$2^6$	16	24
:	:	(0.101)	(0.101)	(0.101)	(0.101)	(0.101)	(0.101)
:	:	(0.101)	(0.101)	(0.101)	(0.101)	(0.101)	(0.101)
:	:	(0.101)	(0.101)	(0.101)	(0.101)	(0.101)	(0.101)

Computing best, worst & average efficiency.  
best  $\rightarrow \Omega(n)$

avg  $\rightarrow \Theta(n)$

worst  $\rightarrow O(n)$

~~Then~~  $a = [10, 20, 30]$

1 Probability of successive search.  $\rightarrow p$

2 Probability of unsuccessful search.  $\rightarrow 1-p$

$$\begin{aligned}
 \text{Avg}(n) &= \left[ 1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + \dots + i \cdot \frac{p}{n} \right] + n(1-p) \\
 &= \frac{p}{n} [1 + 2 + \dots + i < n] + n(1-p) \\
 &= \frac{p}{n} \cdot \frac{n(n+1)}{2} + n(1-p) \\
 &= \frac{p(n+1)}{2} + n(1-p)
 \end{aligned}$$

If  $p = 1$  then it is successive search.

If  $p = 0$  then it is unsuccessful search.

### ASYMPTOTIC NOTATIONS

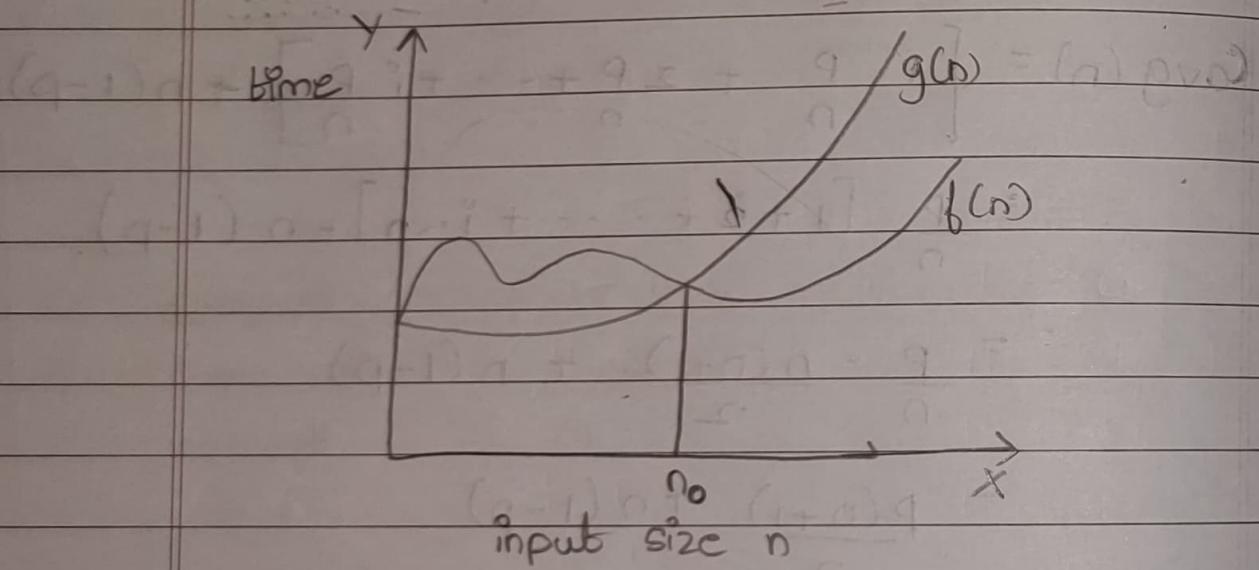
Upper Bond  $\Theta$

lower Bond  $\Omega$

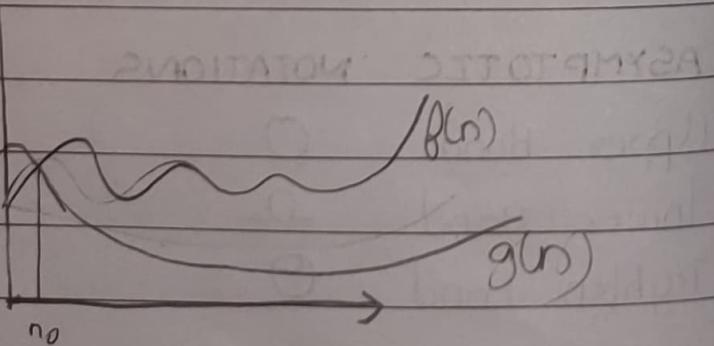
Tighten Bond  $\Theta$

Upper Bound  $O$ 

If  $f(n)$  is a function said to be in  $O(g(n))$  denoted by,  $f(n) \in O(g(n))$ , if  $f(n)$  is bounded above by some constant multiple of  $g(n)$  for all large  $n$ , i.e., if there exist some positive constant  $c$  and some non-negative integer  $n_0$  such that,  $f(n) \leq c \cdot g(n)$  for all  $n \geq n_0$ .

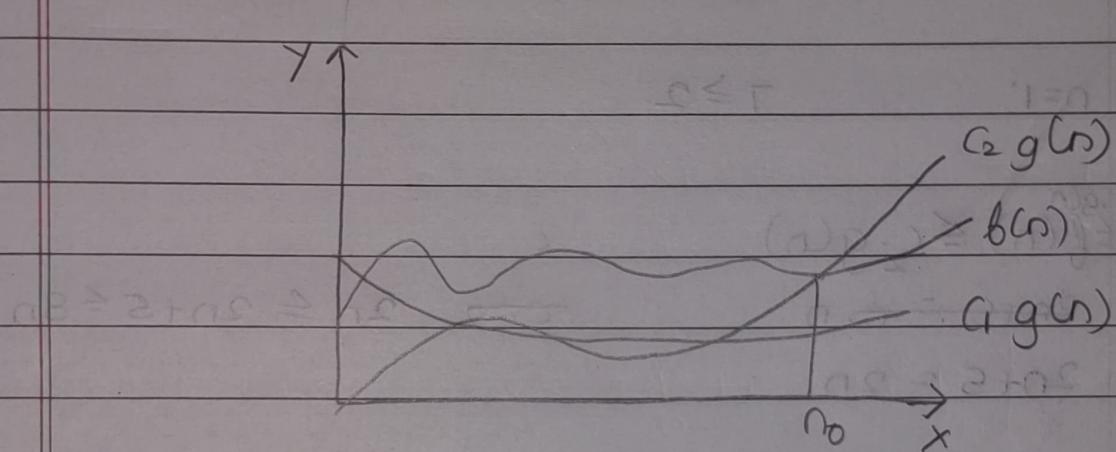
Lower Bound  $\Omega$ 

$$f(n) \geq c \cdot g(n) \quad f(n) \in \Omega(g(n))$$



## Tighten Bound Θ

For a Tighten Bound formulae we define  
 $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$



$$1 \quad f(n) = 2n + 5$$

Find Upper Bound, lower bound and Tighten bound

$$\rightarrow g(n) = n.$$

$$\text{ii)} \quad f(n) \leq c \cdot g(n)$$

$$2n + 5 \leq c \cdot n$$

$$2n + 5 \leq 3n$$

UB	LB	TB
$3n$	$2n$	$2n$

A:

$$n=1$$

$$7 \leq 3 \quad \times$$

$$n=2$$

$$9 \leq 6 \quad \times$$

$$n=3$$

$$11 \leq 9 \quad \times$$

$$n=4$$

$$13 \leq 12 \quad \times$$

$$n=5$$

$$15 \leq 15 \quad \checkmark$$

$$(ii) f(n) \geq^c g(n)$$

$$2n+5 \geq c \cdot n$$

$$c = 2$$

$$2n+5 \geq 2n$$

$$n=1$$

$$7 \geq 2$$

$$(iii) f(n) \leq^c g(n)$$

$$2n+5 \leq c \cdot n$$

$$c$$

$$c=2 \quad 2n \leq 2n+5 \leq 3n$$

$$2n+5 = 2n$$

$$n=1$$

$$7 \geq 2 \quad \times \quad 2 \leq 7 \leq 3 \quad \times$$

$$n=2$$

$$9 \geq 4 \quad \times \quad 4 \leq 9 \leq 6 \quad \times$$

$$n=3$$

$$11 \geq 6 \quad \times \quad 6 \leq 11 \leq 9 \quad \times$$

$$n=4$$

$$13 \geq 8 \quad \times \quad 8 \leq 13 \leq 12 \quad \times$$

$$n=5$$

$$15 \geq 10 \quad \times \quad 10 \leq 15 \leq 15 \quad \checkmark$$

$$n=6$$

$$17 \geq 12$$

$$n=7$$

$$19 \geq 15$$

$$n=8$$

$$21 \geq 16$$

$$n=9$$

$$= 18$$

$$n=10$$

$$25 \geq 20$$

## MATHEMATICAL ANALYSIS OF NON-RECURSIVE ALGMS

(i) check

Maximum element in an array.

input : 2, 7, 8, 9, 10, 3

output : 10

Algorithm .

maxval  $\leftarrow A[0]$

for ( $i=0$  ;  $i < n$  ;  $i++$ )

if ( $maxval < A[i]$ )

maxval  $\leftarrow A[i]$

return maxval.

FORMULAE

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

$l = \text{lower}$     $u = \text{upper}$

Summation Manipulation

$$\sum_{i=l}^u c \cdot a^i = c \sum_{i=l}^u a^i$$

$$\sum_{i=l}^u a^i + b^i = \sum_{i=l}^u a^i + \sum_{i=l}^u b^i$$

Input :

n

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

$$u = n - 1, l = 0$$

## (ii) Matrix multiplication

Algorithm

```
for (i=0 ; i<n ; i++)
    for (j=0
```

 $c[i][j] = 0$ 
 $\text{for } (k=0$ 

$$c[i][j] = c[i][j] + A[i][k] * B[k][j]$$

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1(n-1) = n^3$$

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

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

$$= n^3$$

## (iii) ELEMENT UNIQNESS

Algorithm

```
for (i=0 . . . n-2)
```

 $\text{for } (j=i+1 . . . n-1)$ 
 $\text{if } (A[i] = A[j])$ 
 $\text{return false}$ 
 $\text{return true}$

$$\begin{aligned}
 &= \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \cdot n-1 - (i+1) + 1 \\
 &= \sum_{i=0}^{n-2} (n-2) - (0) \times n - (0+1) \\
 &= (n-2) \cdot (n-1)
 \end{aligned}$$

$$\begin{aligned}
 &= \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 \\
 &= \sum_{i=0}^{n-2} n-1 - (i+1) + 1
 \end{aligned}$$

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

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

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

$$= (n-1) [(n-2)+1] - \frac{(n-2)(n-1)}{2}$$

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

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

$$= n^2 + 1 - 2n - \frac{1}{2} n^2$$

$\approx n^2$  [higher order] degree to bell worst case]

(iv)

Count number of digits in an binary number.

Algorithm

count  $\leftarrow 1$

while  $n > 1$

    count  $\leftarrow$  count + 1

$n = n / 2$

return count

Complexity  $\log_2 n$

$1 + (1+2) + \dots + (n-1)$  for  $n=6$

Complexity 3

## MATHEMATICAL ANALYSIS OF RECURSIVE RELATION

### (i) Factorial

Algorithm fact(n)

if  $n=0$

    return 1

else

    return fact( $n-1$ )  $\times n$

$$M(n) = M(n-1) + 1 \quad \forall n > 1$$

$$M(0) = 1 \quad \text{for } n=0$$

$$\begin{aligned}
 M(n) &= M(n-1) + 1 \\
 &= M(n-2) + 1 + 1 \\
 &= M(n-3) + 3 \\
 &= M(n-4) + 4 \\
 &\vdots
 \end{aligned}$$

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

$$\begin{aligned}
 &= M(n-i) + i \quad [\text{By Generalising}] \quad [iii] \\
 &= M(n-n) + n \quad \text{for } n-i=n \\
 &= M(0) + n \\
 &= 1 + n \\
 &= \Theta(n) \rightarrow O(n)
 \end{aligned}$$

(ii) To answers of

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

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

$$= 2[2M(n-2) + 1] + 1 \quad M(n-1) = 2M(n-2) + 1$$

$$= 4M(n-2) + 2 + 1 \quad M(n-2) = 2M(n-3) + 1$$

$$= 4[2M(n-3) + 1] + 3$$

$$= 8M(n-3) + 4 + 3$$

$$= 8M(n-3) + 7$$

$$\vdots$$

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

$$= 2^{n-1} M(n-n+1) + 2^{n-1} - 1 \quad \text{for } i=n-1$$

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

$$= 2^{n-1}(1) + 2^{n-1} - 1$$

$$= \frac{2^n}{2} + \frac{2^n}{2} - 1 = 2^n - 1$$

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

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

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

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

(iii)

Algorithm Bin(n)

if  $n=1$ 

return 1

else

return  $\cdot \text{Bin}(\lfloor n/2 \rfloor + 1)$ 

$$A(n) = A(n/2) + 1 + \dots \quad n=2^k$$

$$A(1) = 1 \quad k = \log_2 n$$

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

~~$$A(n) = A(n/2) + 1$$~~

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

~~$$A(n) = A(2^{k-1}) + 1$$~~

$$A(n) = A(2^{k-1}) + 1 + 1$$

~~$$A(2^k) = A(2^{k-1}) + 1$$~~

$$= A(2^{k-2}) + 2$$

~~$$A(2^k) = A(2^{k-2}) + 1$$~~

$$= A(2^{k-3}) + 3$$

$$= A(2^{k-1}) + 9$$

$$= \Theta(2^{k-k}) + k$$

$$= 1 + k.$$

$$= 1 + \log_2 n$$

$$\in O(\log_2 n)$$

## BRUTE FORCE

SELECTION SORT

Algorithm Selection Sort ( $A[0], \dots, A[n]$ )

for ( $i = 0 \dots n-1$ )

$\min \leftarrow A[i]$

for ( $j = i+1 \dots n-1$ )

if [ $A[j] < A[\min]$ ]

$\min \leftarrow j$

swap ( $A[i]; A[\min]$ )

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

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

$$= n(n+1) \in O(n^2)$$

$$1 \quad x(n) = x(n-1) + 5 \quad \forall n > 1, \quad x(1) = 0$$

$$\begin{aligned} \rightarrow x(n) &= x(n-1) + 5 & x(n) &= x(n-1) + 5 \\ &= x(n-2) + 5 + 5 & x(n-1) &= x(n-2) + 5 \\ &= x(n-2) + 10 & x(n-2) &= x(n-3) + 5 \\ &= x(n-3) + 5 + 10 & & \\ &= x(n-3) + 15 & & \\ &\vdots & & \\ &= x(n-i) + i \times 3 & & \\ &= x(0) + (n-1) \times 3 & \text{for } i = n-1 & \\ &= 0 + (n-1) \times 3 & & \\ &\in O(n) & & \end{aligned}$$

$$2 \quad x(n) = 3x(n-1) \quad \forall n > 1, \quad x(1) = 4$$

$$\begin{aligned} x(n) &= 3x(n-1) & x(n) &= 3x(n-1) \\ x(n) &= 3 \cdot (3x(n-2)) & x(n-1) &= 3x(n-2) \\ &= 3^2 \cdot 3x(n-2) & x(n-2) &= 3x(n-3) \\ &= 3^2 [3x(n-3)] & & \\ &= 3^3 x(n-3) & & \\ &\vdots & & \\ &= 3^i x(n-i) & & \end{aligned}$$

$$= 3^{n-1} x(1)$$

$$= \frac{4}{3} 3^n$$

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

$$xc(n) = xc(n/3) + 1 \quad + n > 1, xc(1) = 1$$

CLASSMATE  
Date \_\_\_\_\_  
Page \_\_\_\_\_

Merge sort  $T(n) = 2(T(n/2)) + n$

$$\begin{aligned}
 3 \quad xc(n) &= xc(n-1) + n & xc(0) &= 0 \\
 \rightarrow xc(n) &= xc(n-1) + n & xc(n) &= xc(n-1) + n \\
 &= xc(n-2) + n + n - 1 & xc(n-1) &= xc(n-2) + n - 1 \\
 &= xc(n-2) + 2n - 1 & xc(n-2) &= xc(n-3) + n - 2 - 1 \\
 &= xc(n-3) + n - 3 + 2n - 1 & \\
 &= xc(n-3) + 3n - 4 & \\
 &\vdots & \\
 &= xc(n-i) + i * n - (i-1) & \\
 &= xc(0) + n^2 - (n-1) & \text{for } i=n \\
 &= n^2 - n + 1 & \\
 &\in \Theta(n^2)
 \end{aligned}$$

~~Merge sort~~

$$4 \quad xc(n) = xc(n/2) + n \quad + n > 2, xc(1) = 1 \text{ solve for } n=2^K$$

~~xc~~ Consider  $n=2^K$

$$\begin{aligned}
 xc(n) &= xc(n/2) + n & xc(2^K) &= xc(2^K/2) + 2^K \\
 xc(2^K) &= xc(2^{K-1}) + 2^K & xc(2^K) &= xc(2^{K-1}) + 2^K \\
 &= xc[2^{K-2}] + 2^{K-1} + 2^K & xc(2^{K-1}) &= xc[2^{(K-2)}] + 2^{K-1} \\
 &= xc[2^{K-3}] + 2^{K-2} + 2^{K-1} + 2^K & xc(2^{K-2}) &= xc[2^{(K-3)}] + 2^{K-2} \\
 &= xc[2^{K-3}] + \frac{2^K}{2^2} + \frac{2^K}{2^1} + \frac{2^K}{2^0} & \\
 &= xc[2^{K-3}] + 2^K \left[ \frac{1}{2^2} + \frac{1}{2^1} + \frac{1}{2^0} \right]
 \end{aligned}$$

$$\begin{aligned}
 &= x[2^{k-i}] + 2^k + 1 + (c-a)x \\
 &= x[2^{k-k}] + 2^k + 1 + (c-a)x \quad i=0 \\
 &= x[1] + 2^k + 1 + (c-a)x \\
 &= 1 + n + (c-a)x \\
 &= \Theta(n)
 \end{aligned}$$

5  $\alpha(n) = \alpha(n/3) + 1$   $\forall n > 1$   $\alpha(1) = 1$  (solve  $n=3^k$ )

$$\begin{aligned}
 \rightarrow n &= 3^k \\
 k &= \log_3 n
 \end{aligned}$$

$$\begin{aligned}
 \alpha(n) &= \alpha(n/3) + 1 \\
 \alpha(3^k) &= \alpha(3^{k-1}/3) + 1 \\
 \alpha(3^k) &= \alpha(3^{k-1}) + 1 & \alpha(3^k) &= \alpha(3^{k-1}) + 1 \\
 &= \alpha(3^{k-2}) + 1 + 1 & \alpha(3^{k-1}) &= \alpha(3^{k-2}) + 1 \\
 &= \alpha(3^{k-3}) + 1 + 1 + 1 & \alpha(3^{k-2}) &= \alpha(3^{k-3}) + 1 \\
 &= \dots & & \\
 &= \alpha(3^{k-i}) + i & & \\
 &= \alpha(3^{k-k}) + k & \text{for } i=k \\
 &= 1 + k \\
 &= 1 + \log_3 n \\
 &= \Theta(\log n)
 \end{aligned}$$

$$6 \quad x(n) = x(n/3) + n \quad \text{if } n = 3^k$$

$$7 \quad x(n) = 2x[n/2] + 1 \quad x(1) = 1$$

$$\rightarrow x(n) = 2x[n/2] + 1$$

Consider  $n = 2^k$

$$k = \log_2 n$$

$$x(2^k) = 2x(2^k/2) + 1$$

$$x(2^k) = 2x[2^{k-1}] + 1$$

$$= 2[2x(2^{k-2}) + 1] + 1$$

$$= 2^2 x(2^{k-2}) + 3$$

$$= 2^2 [2x(2^{k-3}) + 1] + 3$$

$$= 2^3 x(2^{k-3}) + 7$$

$$x(2^k) = 2x[2^{k-1}] + 1$$

$$x(2^{k-1}) = 2x[2^{k-2}] + 1$$

$$x(2^{k-2}) = 2x(2^{k-3}) + 1$$

$$= 2^i x(2^{k-i}) + 2^i - 1$$

$$= 2^k x(1) + 2^k - 1 \quad \text{for } i=k$$

$$= 2^k + 2^k - 1$$

$$= 2 \cdot 2^k - 1$$

$$= 2 \cdot n - 1$$

$\in \Theta(n)$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 - f(n) \text{ has smaller order growth than } g(n) \\ c > 0 - f(n) \text{ has same order as } g(n) \\ \infty - f(n) \text{ has a larger order growth than } g(n) \end{cases}$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 - f(n) = O(g(n)) \\ c > 0 - f(n) \in \Theta(g(n)) \\ \infty - f(n) \in \Omega(g(n)) \end{cases}$$

### STIRLING'S FORMULA

$$n! = \sqrt{2\pi n} (n/e)^n$$

L' Hopital's Rule

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

▷  $\frac{1}{2}n(n-1)$  and  $n^2$

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

$$\begin{aligned}
 &= \frac{1}{2} \lim_{n \rightarrow \infty} \frac{n^2 - \frac{n}{n^2}}{n^2} \\
 &= \frac{1}{2} \lim_{n \rightarrow \infty} 1 - \frac{1}{n} \\
 &\stackrel{2}{=} \frac{1}{2} \left[ 1 - \frac{1}{\infty} \right] \\
 &= \frac{1}{2} > 0
 \end{aligned}$$

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

2)  $\log n$  and  $\sqrt{n}$

$$\frac{\log n}{\sqrt{n}} = \lim_{n \rightarrow \infty} \frac{\log n}{\sqrt{n}}$$

$$\begin{aligned}
 &= \lim_{n \rightarrow \infty} \frac{1/n}{2\sqrt{n}} \\
 &= 0
 \end{aligned}$$

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

3)  $n!$  and  $2^n$

$$\frac{n!}{2^n} = \lim_{n \rightarrow \infty} \frac{\sqrt{2\pi n}}{2^n} \left(\frac{n}{e}\right)^n$$

$$= \lim_{n \rightarrow \infty} \frac{\sqrt{2\pi n}}{2^n} \frac{n^n}{e^n} \times \frac{1}{e^n}$$

$$= \frac{1}{4} \cdot \sqrt{2\pi} \lim_{n \rightarrow \infty} \frac{n^{n+\frac{1}{2}}}{e^{2n}}$$

$$= \sqrt{2\pi} \cdot \infty$$

$$n! \in \Omega(2^n)$$

↳  $3n^2 + 4n$  and  $n^2$

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

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

$$= \lim_{n \rightarrow \infty} 3 + \frac{4}{n}$$

$$= 3$$

$$3n^2 + 4n = \Theta(n^2)$$

5  $\frac{n^2 - 3n}{3}$  and  $n^2$

$$\rightarrow \frac{\frac{n^2 - 3n}{3}}{n^2} = \lim_{n \rightarrow \infty} \frac{n^2 - 3n}{n^2}$$

$$= \lim_{n \rightarrow \infty} \frac{n^2 - 3n}{3n^2}$$

$$= \frac{1}{3}$$

$$\frac{n^2 - 3n}{3} \in \Theta(n^2)$$

6 log n and n

$$\rightarrow \frac{\log n}{n} = \lim_{n \rightarrow \infty} \frac{\log n}{n}$$

$$= \lim_{n \rightarrow \infty} \frac{1/n}{1}$$

$$= 0$$

$$\log n \in O(n)$$