An algorithm is an sequence of ~~set~~ un-ambigious steps (instruction) for solving a problem to obtain a required output for any legimate input in a finite amount of time.

Important points illustrating motion of algorithm

[i] The non-ambigity required for each step of an algorithm cannot be compromized.

[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 algorithm 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

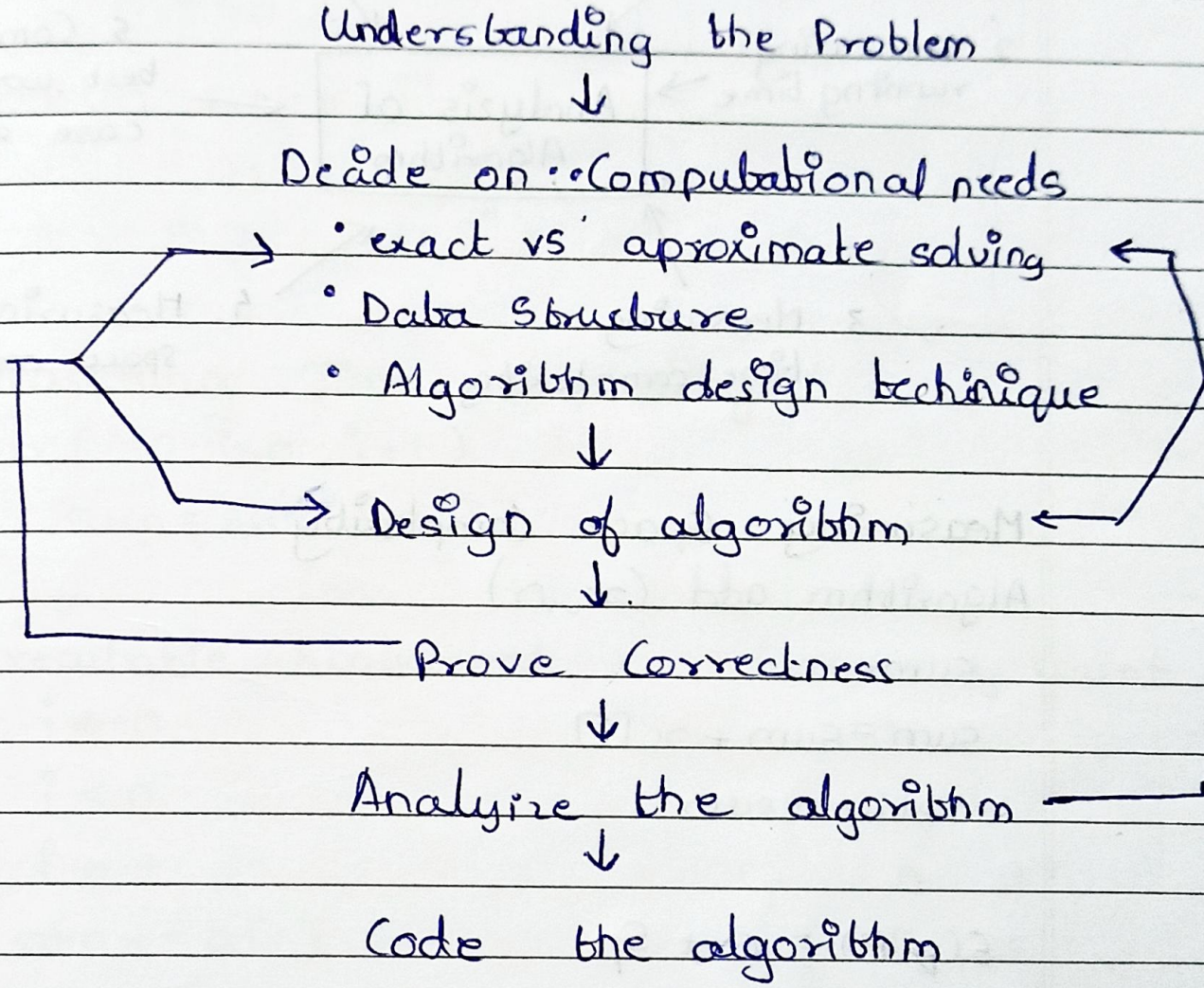
1 Buying the ticket for a p
2 Going to the Cinema hall
3 Searching for the seat.
4 Watching the movie

1 Read m and n
2 while (n ≠ 0)
  1 r = m mod n
  2 m = n
  3 n = r
3 ~~print~~ Display m.

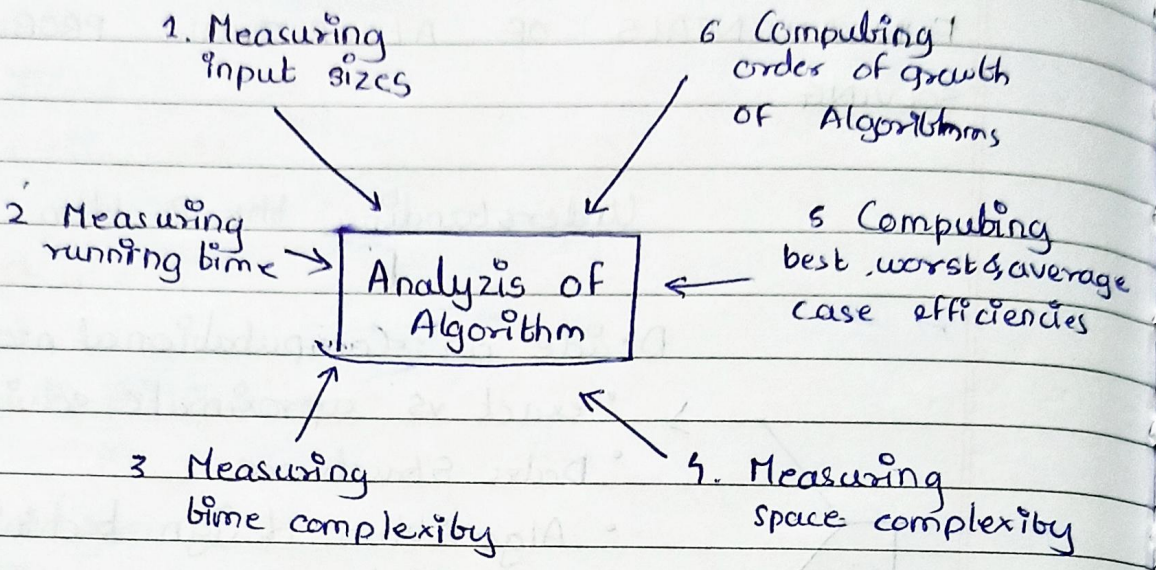
1 Read m and n
2 Till n is not equal to zero.
  1 assign m mod n to r
  2 assign n to m
  3 assign r to n
3 Display m

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 ≠ 0)
r ← m mod n
m ← n
n ← r
return m

FUNDAMENTALS OF ALGORITHMIC PROBLEM SOLVING



ANALIZES FRAMEWORK



Measuring Space Complexity

```
Algorithm add (x, n)
  sum ← 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)

sum = 1

i = 1

n = 1

3 = Sp

S(p) = C + Sp

= n + 3

≃ n

Measuring Time complexity

```
for (i=0 ; i<n ; i++)
    sum = sum + a[i];
```

Executable Statement	Frequency Count
i ← 0	1
i < n	n + 1
i++	n
sum += a[i]	n
	3n + 2 ≃ 3n

Complexity = n [O(n)]

Measuring running Time

T(n) = Cop * C(n)

T(n) ⟶ Running time of an algorithm.

Cop ⟶ Time taken by basic operation to execute

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

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

$$\frac{T(2n)}{T(n)} = \frac{C_{op} * ((2n)^2/2)}{C_{op} * (n^2/2)} \qquad \begin{matrix} C_{op} \approx C(2n) \\ C_{op} \approx C(n) \end{matrix}$$

$$\frac{T(2n)}{T(n)} = 4$$

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
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Computing best, worst & average efficiency.

best → $\Omega(n)$

avg → $O(n)$
worst → $O(n)$

~~Their~~ a = [10, 20, 30]

1. Probability of successive search → p
2. Probability of unsuccessive search → 1 − p

$$C_{avg}(n) = \left[1 \cdot \frac{p}{n} + 2\frac{p}{n} + \cdots + i\frac{p}{n}\right] + n(1-p)$$

$$= \frac{p}{n}\left[1 + 2 + \cdots + i \cdots + n\right] + n(1-p)$$

$$= \frac{p}{n} \cdot \frac{n(n+1)}{2} + n(1-p)$$

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

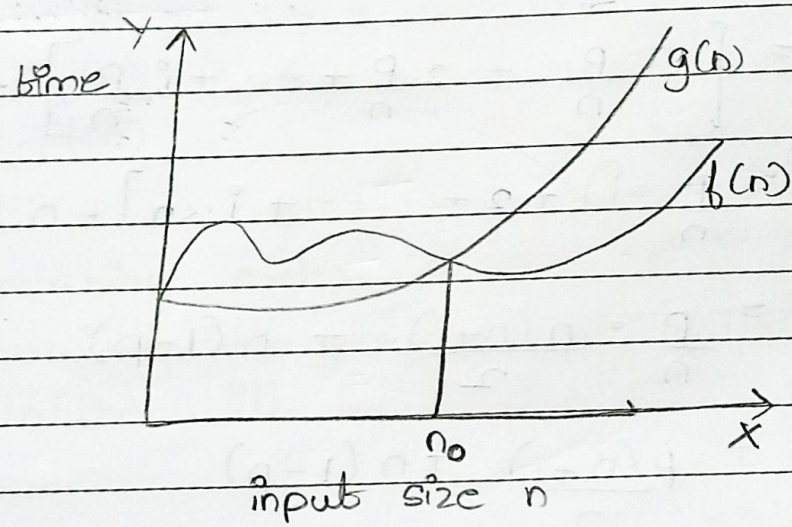
If $p \equiv 1$ then it is successive search.
If $p = 0$ then it is unsuccesive search.

ASYMPTOTIC NOTATIONS

Upper Bond	O
lower Bond	Ω
Tighten Bond	Θ

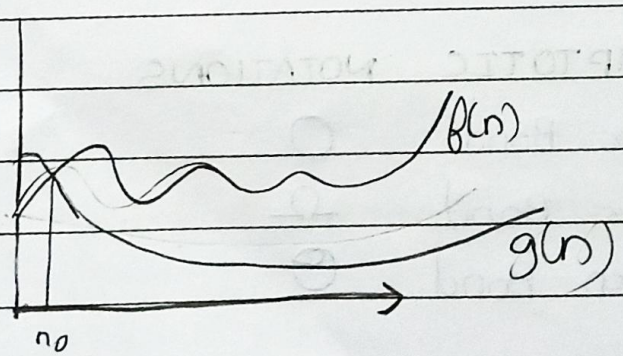
Upper Bound O

If $f(n)$ is a function said to be is $O(g(n))$ 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 their 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 Ω

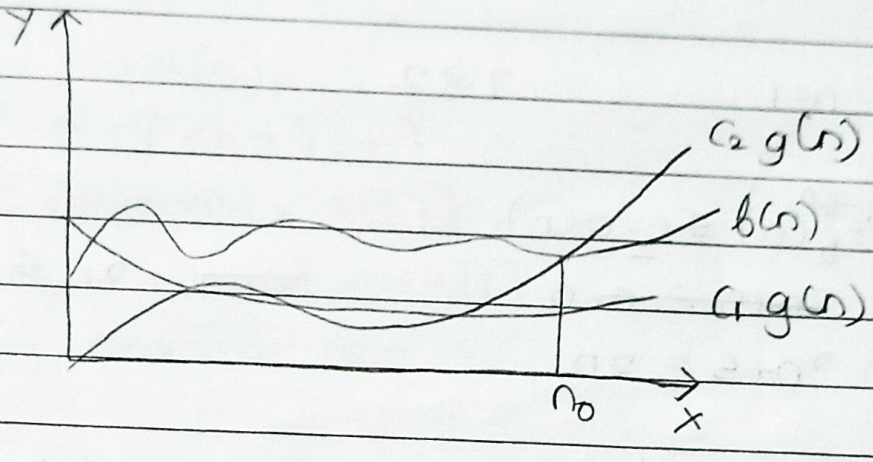
$f(n) \geq c \cdot g(n)$ $\quad\quad$ $f(n) \in O(g(n))$



Tighten Bound Θ

For a tighten Bound formulae we define

$c_1 \cdot g(n) \leq f(n) \leq c_2 \, g(n)$



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

Find Upper Bound, lower Bound and tighten bound

→ $g(n) = n$

ii) $f(n) \leq c \cdot g(n)$

$2n + 5 \leq c \cdot n \qquad c = 3$

$2n + 5 \leq 3n$

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

$n = 1$	$7 \leq 3$	×
$n = 2$	$9 \leq 6$	×
$n = 3$	$11 \leq 9$	×
$n = 4$	$13 \leq 12$	×
$n = 5$	$15 \leq 15$	✓

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

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

$2n+5 \geq 2n$

$n=1$ $\quad 7 \geq 2$

(iii) $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

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

$2n+5 = 2n$

$n=1$	$7 \geq 2$ ✓	$2 \leq 7 \leq 3$ ✗
$n=2$	$9 \geq 4$ ✓	$4 \leq 9 \leq 6$ ✗
$n=3$	$11 = 6$	$6 \leq 11 \leq 9$ ✗
$n=4$	$13 = 8$	$8 \leq 13 \leq 12$ ✗
$n=5$	$15 = 10$	$10 \leq 15 \leq 15$ ✓
$n=6$	$17 = 12$	
$n=7$	$19 = 14$	
$n=8$	$21 = 16$	
$n=9$	$= 18$	
$n=10$	$25 = 20$	

MATHEMATICAL ANALYSIS OF NON-RECURSIVE ALGMs

(i) Check Maximum element in an array.

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

Algorithm.

```
maxval ← A[0]
for (i=0 ; i<n ; i++)
    if (maxval < A[i])
        maxval ← A[i]
return maxval.
```

FORMULAE

$$\sum_{i=l}^{u} 1 = u - l + 1 \qquad l = \text{lower} \quad u = \text{upper}$$

Summation Manipulation

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

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

$$\sum_{i=l}^{u} a_i + b_i = \sum_{i=l}^{u} a_i + \sum_{i=l}^{u} b_i$$

~~Input~~ :

n

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

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

(ii) Matrix multiplication

Algorithm

```
for (i=0 ; i<n ; i++)
    for (j=0
        c[i][j] = 0
        for (k=0
            c[i][j] = c[i][j] + A[i][k] * B[k][j]
```

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

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

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

$$= n^3$$

(iii) ELEMENT UNIQNESS

Algorithm

```
for (i=0 . . . . . n-2)
    for (j=i+1 . . . . n-1)
        if (A[i] = A[j])
            return false
return true
```

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

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

$$= (n-2)\cdot(n-1)$$

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

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

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

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

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

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

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

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

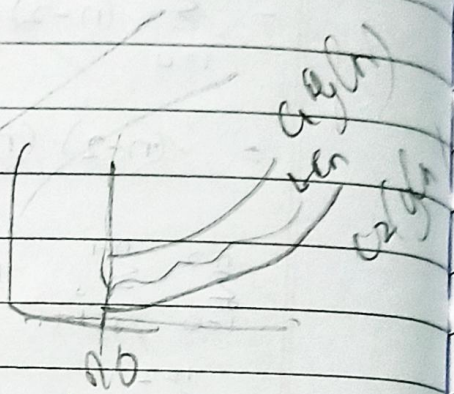
$$= n^2 + 1 - 2n - \frac{1}{2} n^2$$

$\simeq n^2$ [higher order degree to tell worst case]

[iv] Count number of digits in an binary number

Algorithm

```
count ← 1
while 'n > 1
    count ← count + 1
    n = n/2
return count
```



Complexity $\log_2 n$

for n = 6

Complexity 3

MATHEMATICAL ANALYSIS OF RECURSIVE RELATION

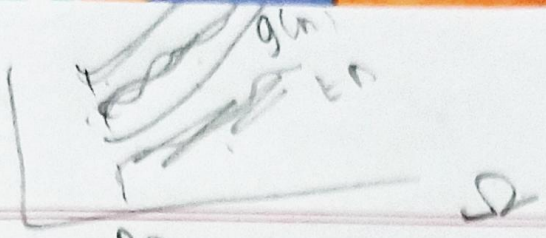
(i) Factorial

Algorithm fact(n)

```
if n==0
    return 1
else
    return fact(n-1)*n
```

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

$M(0) = 1$ for $n = 0$



$M(n) = M(n-1) + 1$
$= M(n-2) + 1 + 1$
$= M(n-3) + 3$
$= M(n-4) + 4$
$\vdots$
$= M(n-i) + i$ [By Generalising]
$= M(n-n) + n$ for $n-i=n$
$= M(0) + n$
$= 1 + n$
$= \Theta(n) \rightarrow O(n)$

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

(ii) To answers of

$M(n) = 2M(n-1) + 1$
$= 2[2M(n-2)+1] + 1$
$= 4M(n-2) + 2 + 1$
$= 4[2M(n-3)+1] + 3$
$= 8M(n-3) + 4 + 3$
$= 8M(n-3) + 7$
$\vdots$
$= 2^i M(n-i) + (2^i - 1)$
$= 2^{n-1} M(n-n+1) + 2^{n-1} - 1$ for $i = n-1$
$= 2^{n-1} M(1) + 2^{n-1} - 1$
$= 2^{n-1}(1) + 2^{n-1} - 1$
$= \frac{2^n}{2} + \frac{2^n}{2} - 1 = 2^n - 1$

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

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

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

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

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

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

[iii]

Algorithm Bin(n)

if n=1

return 1

else

return Bin($\lfloor n/2 \rfloor + 1$)

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

$A(1) = 1$

$n = 2^k$

$k = \log_2 n$

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

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

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

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

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

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

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

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

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

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

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

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

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

$= 1 + K$

$= 1 + \log_2 n$

$\in \Theta(\log_2 n)$

BRUTE FORCE

SELECTION SORT

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

for $(i = 0 \ldots$ ~~to~~ $\ldots n-2)$

min $\leftarrow$ ~~A[i]~~ i

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

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

min $\leftarrow j$

swap $[A[i]; A[min]]$

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

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

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

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

→ $x(n) = x(n-1) + 5$

$= x(n-2) + 5 + 5$

$= x(n-2) + 10$

$= x(n-3) + 5 + 10$

$= x(n-3) + 15$

$\vdots$

$= x(n-i) + i * 3$

$= x(1) + (n-1) * 3$ for $i = n-1$

$= 0 + (n-1) * 3$

$\in O(n)$

$x(n) = 3x(n-1)$

$x(n-1) = 3x(n-2)$

$x(n-2) = 3x(n-3)$

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

$x(n) = 3x(n-1)$

$x(n) = 3(3x(n-2))$

$= 3^2 x(n-2)$

$= 3^2 [3x(n-3)]$

$= 3^3 x(n-3)$

$\vdots$

$= 3^i x(n-i)$

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

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

$\in \Theta(3^n)$

$x(n) = x(n/3) + 1 \quad \forall\ n>1 \quad x(1)=1$ (solve for $n=3^k$)

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

3 $x(n) = x(n-1) + n \qquad x(0) = 0$

→ $x(n) = x(n-1) + n$

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

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

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

$= x(n-3) + 3n - 4$

$\vdots$

$= x(n-i) + i*n - (i-1)$

$= x(0) + n^2 - (n-1)$ for $i = n$

$= n^2 - n + 1$

$\in \Theta(n^2)$

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

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

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

~~Merge sort~~

4 $x(n) = x(n/2) + n \qquad \forall\ n>1,\ x(1)=1$ solve for $n=2^k$

Conside $n = 2^k$

$k = \log_2 n$

$x(n) = x(n/2) + n$

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

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

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

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

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

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

$= x[2^{k-3}] + \frac{2^k}{2^2} + \frac{2^k}{2^1} + \frac{2^k}{2^0}$

$= x[2^{k-3}] + 2^k \left[\frac{1}{2^2} + \frac{1}{2^1} + \frac{1}{2^0} \right]$

$= x[2^{k-i}] + 2^k * 1$

$= x[2^{k-k}] + 2^k \qquad i = k$

$= 1 + 2^k$

$= 1 + n$

$= \Theta(n)$

5 $x(n) = x(n/3) + 1 \quad \forall\ n > 1 \quad x(1) = 1$ (Solve $n = 3^k$)

→ $n = 3^k$

$k = \log_3 n$

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

$x(3^k) = x(3^k/3) + 1$

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

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

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

$= \vdots$

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

$= x(3^{k-k}) + k \qquad$ for $i = k$

$= 1 + k$

$= 1 + \log_3 n$

$= \Theta(\log_3 n)$

6 $x(n) = x(n/3) + n \quad \forall n = 3^k$

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

→ $x(n) = 2x[n/2] + 1$

Conside $n = 2^k$

$k = \log_2 n$

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

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

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

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

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

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

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

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

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

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

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

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

$= 2 \cdot 2^k - 1$

$= 2 \cdot n - 1$

$\in \Theta(n)$

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

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

STIRLINGE FORMULAE

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

L' Hopital's Rule

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

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

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

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

$$= \frac{1}{2} \lim_{n \to \infty} 1 - \frac{1}{n}$$

$$= \frac{1}{2} \left[1 - \frac{1}{\infty} \right]$$

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

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

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

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

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

$$= 0$$

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

3) $n!$ and 2^n

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

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

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

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

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

4) $3n^2 + 4n$ and n^2

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

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

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

$$= 3$$

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

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

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

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

$= \frac{1}{3}$

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

6 $\log n$ and n

→ $\frac{\log n}{n} = \lim_{n \to \infty} \frac{\log n}{n}$

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

$= 0$

$\log n \in O(n)$