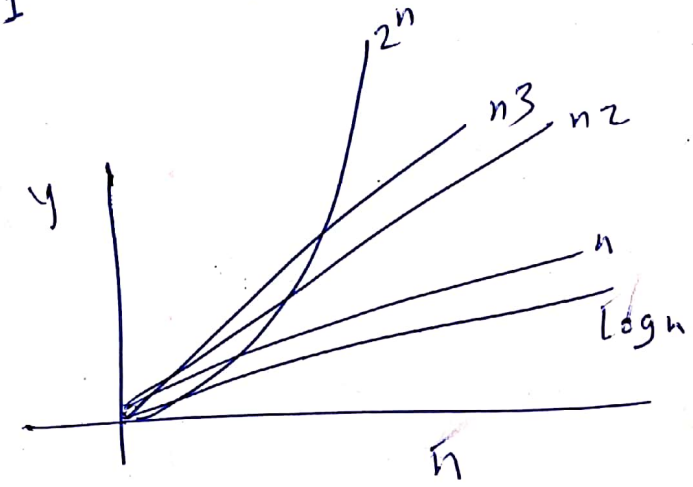
$\log n$	n	n^2	2^n
0	1	1	2
1	2	4	4
2	4	16	16
3	8	64	256
3.1	9	81	512

$\log_2 2 = 1$

النظر الى الفرق



by graphics

Asymptotic notation

This topic come from mathematic function is mathematic
So Asymptotic also mathematic

notations are used for representing the simple form or function or showing the class of function

O — big-oh works upper bound
Ω — big omega lower bound
Θ Theta Average bound of function

theta is one useful

if can't not exact place any function we can upper part of function

Big-oh

The function $f(n) = O(g(n))$ iff $\exists$ +ve constant c and n_0

such that $f(n) \leq c * g(n) \quad \forall \; n \geq n_0$

ex $f(n) = 2n + 3$

$2n+3 \leq$

$2n+3 \leq 10n$

$2(1)+3 \leq 10(1)$

$f(n)$ c $g(n)$

$f(n) = O(n)$

$2n+3 \leq 2n+3 \quad n \geq 1$

$f(n) = O(n)$

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

$c \quad g(n)$

$f(n) = O(n^2)$

true also

$f(n) = O(n)$

$f(n)$ $O(n^2)$ true

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \dots < 2^n < 3^n < \dots$

lower bound — belong — upper bound

$f(n) = 2n + 3$

Aver bound

$f(n) = O(n)$

$f(n) = O(n^2)$

$f(n) \quad O(2^n)$

Wrong ✗ $f(n) = o(\log n)$

log represent lower bound

Try to write closest function ← الأفضل

$f(n) = O(n)$

Omega notation

$f(n) = \Omega(g(n))$ iff $\exists$ +ve constants c, n_0 such that $f(n) \geq c * g(n) \quad \forall n \geq n_0$

$f(n) = 2n + 3$

$2n + 3 \geq 1\, n \quad \forall n \geq 1$

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

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

$2n + 3 \geq 1 \log n$

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

Theta notation

$f(n) = \theta(g(n))$ iff $\exists$ +ve Constant c_1, c_2 and no

such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$

ex $f(n) = 2n + 3$

both side

$1n \leq 2n+3 \leq 5n$

c_1 $g(n)$ $f(n)$ c_2 $g(n)$

$f(n) = \theta(n)$ Average bound

exactly n

I can not write $f(n) = \theta(n^2)$ ✗

do not mix this one with best case or worst case

it is not relate

Divide & Conquer

P.1

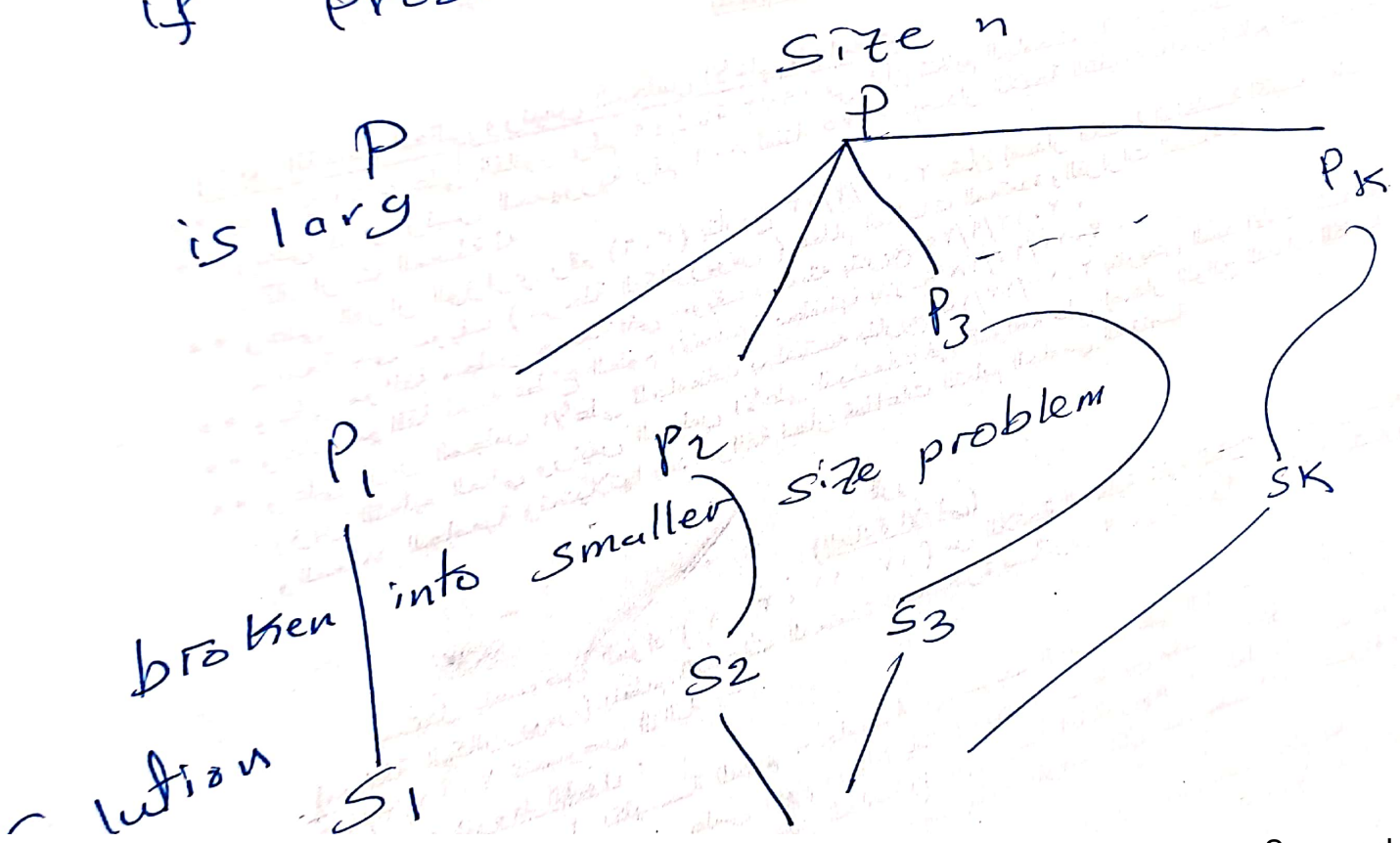
L3

- How analyz algorithm what purpose

- Divide & Conquer is strategy for solving problem
- like dynamic programming, backtracking, 3D method etc

Startegy is a approach & design for solving the proble

if Problem some size n



- whatever the problem is the subproblems will be the same that Problem

ex P is sorting subproblem must be sorting also

- Divid & Conquer when you write or when you take it will b "arecursive" you recursively solve it
- You have a method to Combine subproblem solutions If unable to Combine then you Can not adopt this strategy

General Method for divide & Conquer

```
DAC (P)
{
    if (Small(P))
    {
        S(P)        // S solution
    }
    else
    {
        divide p into P1, P2, P3, ..., Pk
        Apply DAC(P1), DAC(P2) ...
        Combine (DAC(P1), DAC(P2) --
    }
}
```