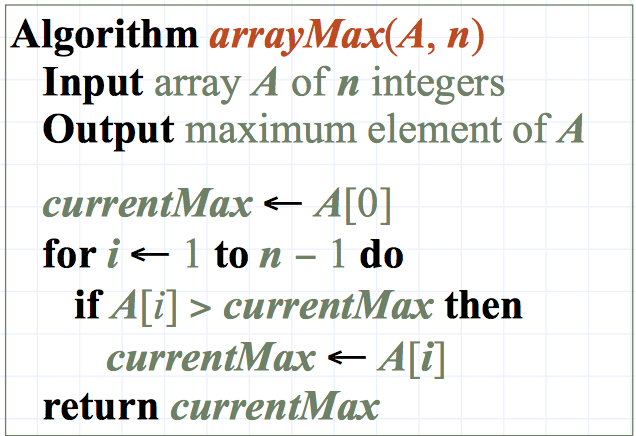
# Ch 4 Analysis of Algorithms

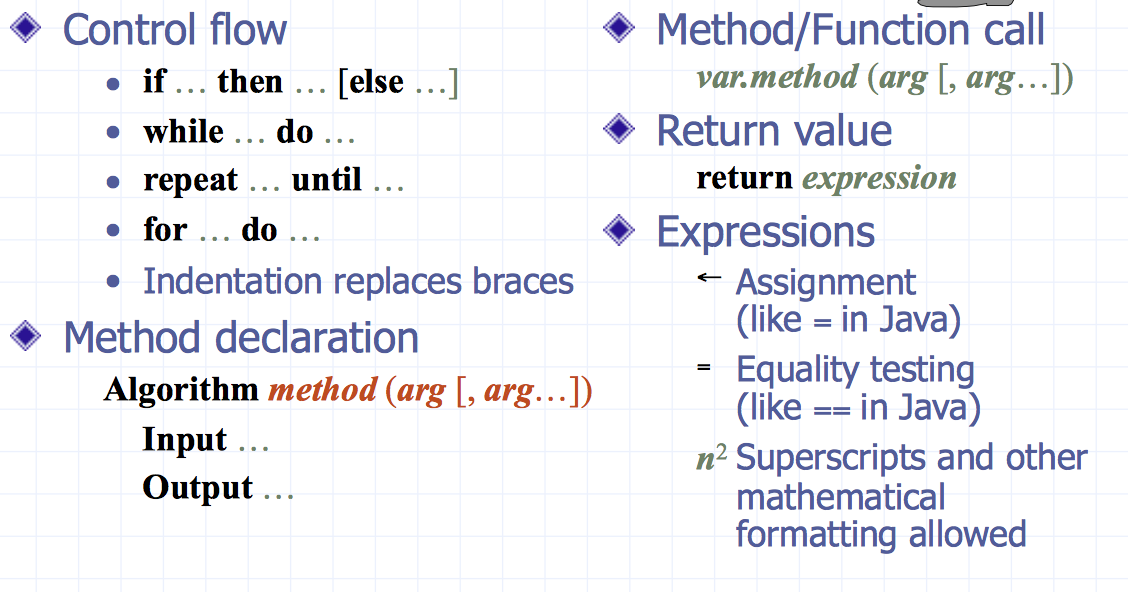
Algorithm: step-by-step procedure for solving a problem in a finite amount of time

Running Time: grows with the input size; focus on worst case; System.currentTimeMillis() to of the actual running time

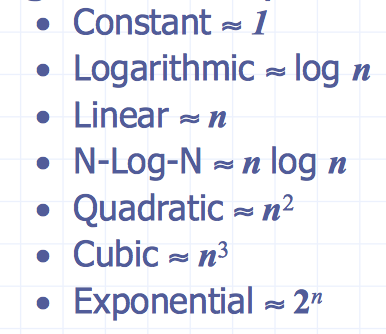
In order to compare two algorithms, the same hardware and software environments must be used

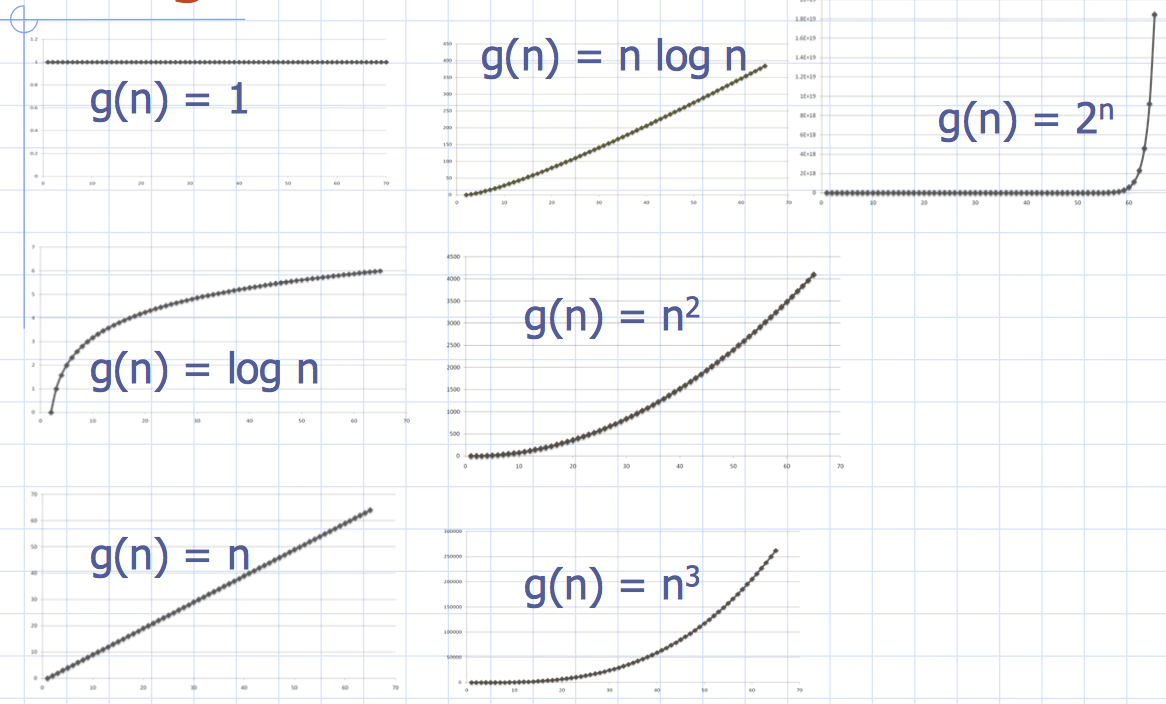
Pseudocode: High-level description of an algorithm; Less detailed than a program





## Seven functions:

 **ORDER: 1, logn, n, nlogn, n2, n3, 2n, n!, nn**

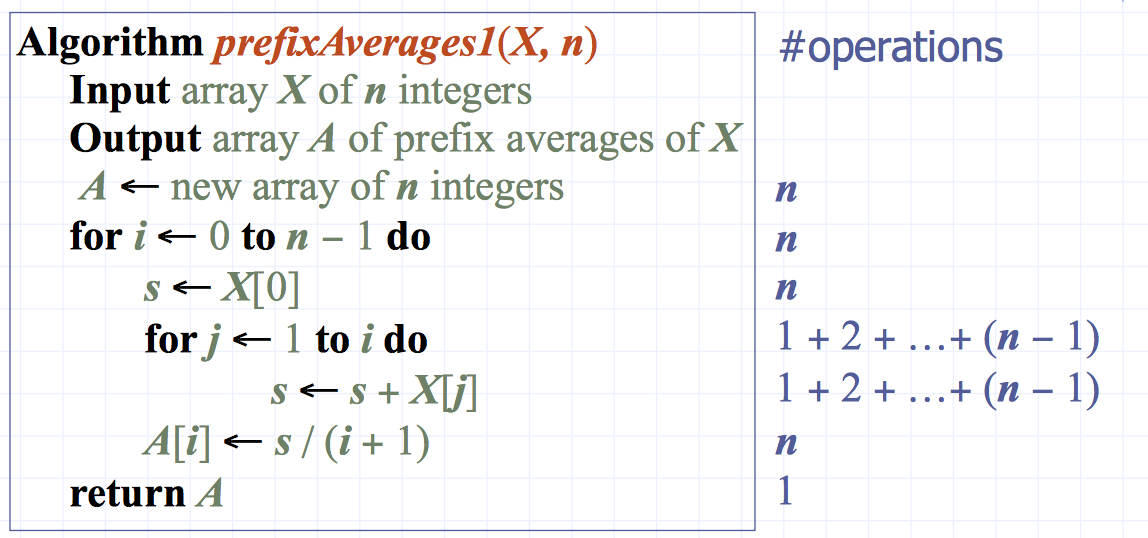


Estimating Running Time : check every line of pseudocode and estimate there time

Big-Oh Notation: f(n)≤cg(n) for n≥n0 Ex 2n + 10 is O(n)

It gives an ***upper bound*** on the growth rate of a function.

***Rules***: If f(n) is a polynomial of degree d, then f(n) is O(nd) (drop lower-order terms; drop constant factors). Use smallest possible class functions; use simplest expression of the class.



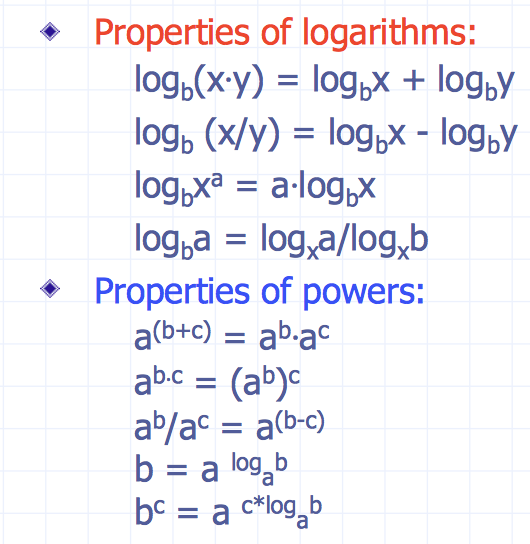
*Ex:*

*Running time is: )(1+2+…+n)*

*Sum of first n is n(n+1)/2*

*It runs in O(n2) time*

NOT FORGET



Big Omega: f(n) ≥ cg(n) for n≥n0 ex: 3nlogn – 2n is Ω(nlogn)

Big Theta : c1g(n) ≤ f(n) ≤ c2g(n), for n≥n0

**Element uniqueness**: when an array of n elements & asked whether all elements of collection are distinct

Iterative algorithm O(n2)

Using sorting O(nlogn)