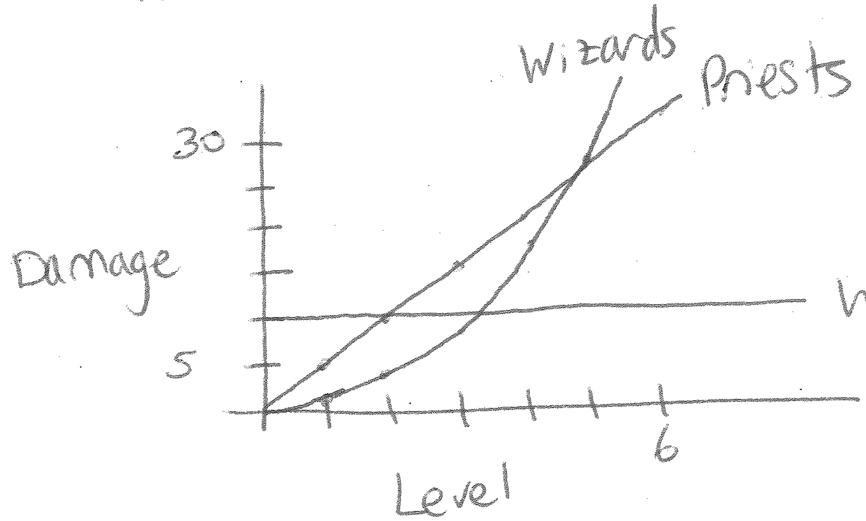
4.1 Rates of Growth and Big O Notation

Ex: In a computer game,

Warriors do 10 points of damage (regardless of level)
Priests do 5 points of damage per level
Wizards do $(\text{level})^2$ points of damage



a) Who does most damage at Level 1?

Warriors

b) Who does most damage at Level 6?

Wizards

c) Who does most damage at level 100?

Wizards

d) What level is the breakeven point between Priests and Warriors?

Priests and Warriors do the same damage at Level 2.

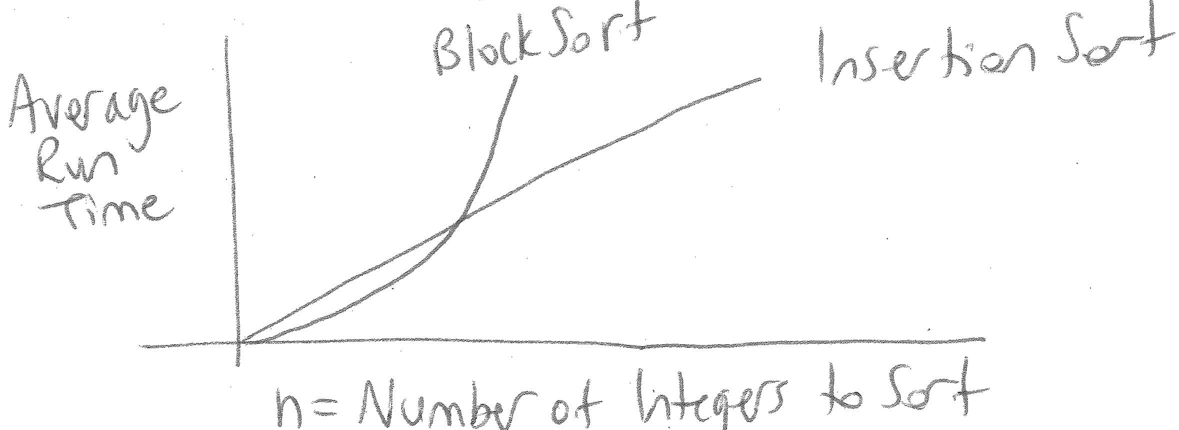
Consider a program that sorts integers.

e.g. Input is 19, 2, 10, 7

Output is 2, 7, 10, 19

Various algorithms exist.

<u>Ex</u>: Consider the graph



a) Which algorithm performs best for very small values of n?

Block Sort

(Best means smallest average run time.)

b) which algorithm performs best for very large values of n?

Insertion Sort

In practice, we only care about very large n.

The <u>dominant term</u> of $3n+5$ is n.

- Select highest power of n
- Ignore coefficients

We say that $3n+5$ is $O(n)$ or "of order n."

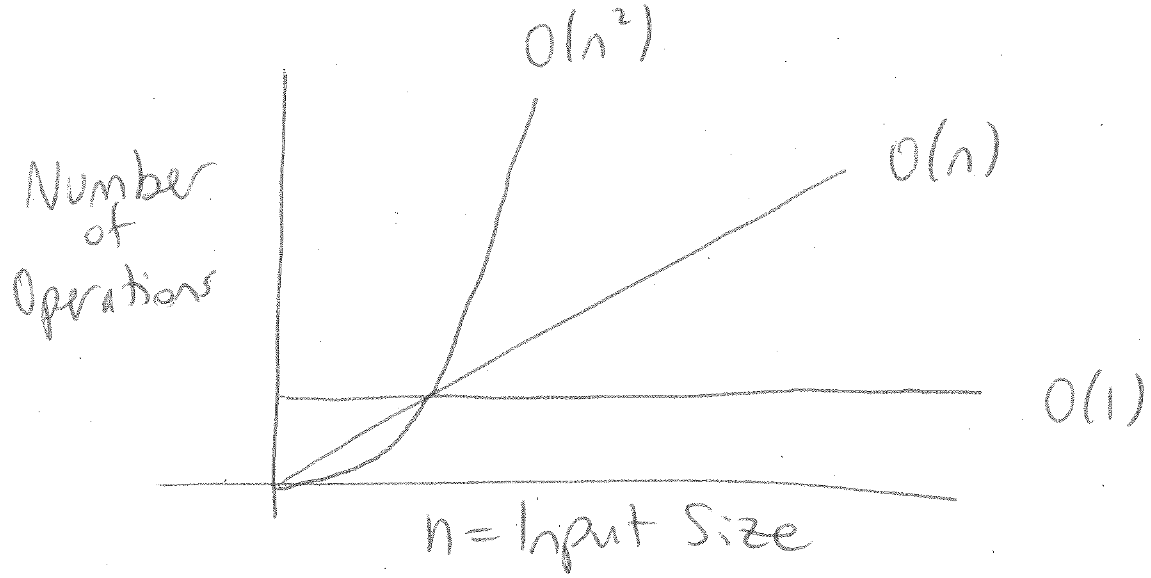
Expression	Dominant Term	Order
$4n+6$	n	$O(n)$
$4n$	n	$O(n)$
n	n	$O(n)$
$3n^2$	n^2	$O(n^2)$
n^2	n^2	$O(n^2)$
$4n^2+6n+2$	n^2	$O(n^2)$
3	1	$O(1)$

Let n = input size of an algorithm (also called the number of elements).

Programmers can calculate the number of operations and find the order of the algorithm.

<u>Quick Ex</u>: An algorithm has $6n^2+2$ operations, where n is the input size. What is the order of the algorithm?

$O(n^2)$



An algorithm that is $O(1)$ performs faster than $O(n)$, $O(n^2)$ for large n.

An algorithm that is $O(n^2)$ performs slower than $O(1)$, $O(n)$ for large n.