

COSC201 ~ 9/28/2011

↳ How to determine the time complexity of an algorithm

↳ look for the dominant term ~ what takes the most time

- Generally, the dominant term will be loops

- Look for the most deeply nested ~~if~~ loops

$O(1)$  → no loops

$O(n)$  → a stand-alone loop

$O(n^2)$  → 2 nested loops

$O(n^3)$  → 3 nested loops

$O(\log n)$  → recursion (usually), parallel processing

\* Note that we care most about the dominant term -

that's why we only report the dominant term in our Big-Oh.

↳ Total Time Complexity

↳ includes every loop + line of code; not just dominant term

$$\text{Program 1: } n + n^3 + c \rightarrow O(n^3)$$

↑      ↑      ↑  
single loop   triple nested loop   other lines of code (non-loops)

$$\text{Program 2: } n + n + c \rightarrow O(n)$$

choose one to report

Total Complexity

The Complexity we report