Comparing Code: Analytically

Basic Operations take "some amount of" Constant Time

- Arithmetic (fixed-width) 10 + 1
- Variable Assignment a = 47
- Access one Java field or array index
- etc.

if(false){
$O(n^3)$
}else{
$O(n)$
}

(This is an approximation of reality: a very useful "lie".)

Complex Operations

Consecutive Statements. Sum of time of each statement
Conditionals. Time of condition + `max(ifBranch, elseBranch)`
Loops. Number of iterations $*$ Time for Loop Body
Function Calls. Time of function's body
Recursive Function Calls. Solve Recurrence

We'd like to be able to compare two functions. Intuitively, we want an operation like "$\leq$" (e.g. $4 \leq 5$), but for functions.

If we have f and $4f$, we should consider them the same:

$f \leq g$ when...

$f \leq cg$ where c is a constant and $c \neq 0$.

We also care about **all values of the function** that are **big enough**:

$f \leq g$ when...

For all n "large enough", $f(n) \leq cg(n)$, where $c \neq 0$

For some $n_0 \geq 0$, for all $n \geq n_0$, $f(n) \leq cg(n)$, where $c \neq 0$

$n \leq n+1$

$n \leq 4n^2+4$

$n \leq 4(n^2+1)$

$n \leq (n+1)^2$

$n \leq \boxed{n}\, n^2$

True or False?

(1) $\cancel{4} + \cancel{3}n \in \mathcal{O}(n)$

(2) $4 + 3n = \mathcal{O}(1)$

(3) $4 + 3n$ is $\mathcal{O}(n^2)$

(4) $n + 2\log n \in \mathcal{O}(\log n)$

(5) $\log n \in \mathcal{O}(n + 2\log n)$

$\cancel{O(1) = 4}$

$=$

n

$f(n) \in O(1)$

Definition (Big-Oh)

We say a function $f : A \to B$ **is dominated by** a function $g : A \to B$ when:

$$\exists (c, n_0 > 0).\ \forall (n \geq n_0).\ f(n) \leq c g(n)$$

Formally, we write this as $f \in \mathcal{O}(g)$.

We want to prove $4 + 3n \in \mathcal{O}(n)$. That is, we want to prove:

$$\exists (c, n_0 > 0).\ \forall (n \geq n_0).\ 4 + 3n \leq cn$$

Proof Strategy

- Choose a c, n_0 that work. $n \geq 4$
- Prove that they work for all $n \geq n_0$. $4 + 3n \leq n + 3n \leq 4n$

Proof