Big-O notation

Lecture 10

Running time of selection sort

- We showed that running selection sort on an array of n elements takes in the worst case $T(n) = 1 + 15\,n + 5\,n^2$ primitive operations
- When n is large, $T(n) \approx 5\,n^2$
- When n is large,

  $T(2n) / T(n) \approx 5\,(2n)^2 / 5\,n^2$

  ≈ 4

  Doubling n quadruples T(n)

  N.B. That is true for any coefficient of n^2 (not just 5)

n	T(n)
10	661
20	2301
30	4951
40	8601
...	...
1000	5015001
2000	20030001

Big - O notation

- Goals:
  - Simplify the discussion of algorithm running times
  - Describe how the running time of an algorithm increases as a function of n (the size of the problem), when n is LARGE
  - Get rid of terms that become insignificant when n is large
- We will say things like:

  The worst-case running time of selectionSort on an array of n elements is $O(n^2)$

  The worst-case running time of mergeSort on an array of n elements is $O(n \log(n))$

Big-O definition

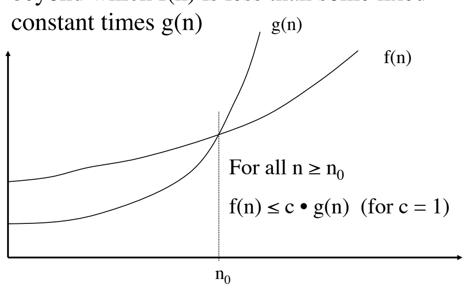
- Let f(n) and g(n) be two non-negative functions defined on the natural numbers N
- We say that f(n) is O(g(n)) if and only if:
  - There exists an integer n_0 and a real number c such that: for all $n \geq n_0$, $f(n) \leq c \bullet g(n)$

  More mathematically, we would write

  – $\exists\, n_0 \in N, \exists\, c \in R : \forall\, n \geq n_0, f(n) \leq c \bullet g(n)$
- N.B. The constant c must *not* depend on n

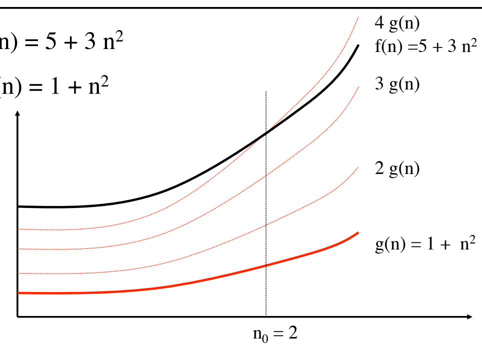
Intuition and visualization

- "f(n) is O(g(n))" iff there exists a point n_0 beyond which f(n) is less than some fixed constant times g(n)



$f(n) = 5 + 3\,n^2$

$g(n) = 1 + n^2$



f(n) is O(g(n)), because there exists $n_0 = 2$ and $c = 4$ such that for all $n \geq n_0$, $f(n) \leq c \bullet g(n)$

Proving big-O relations

- To prove that f(n) is O(g(n)), we must find n_0 and c such that f(n) ≤ c • g (n)
- Example: Prove that $5 + 3n^2$ is $O(1 + n^2)$

We need to pick c greater 3. Let's pick c = 5.

If we choose $n_0 = 1$, we get that if $n \geq n_0$, then

$5 + 3n^2 \leq 5 + 5n^2$ (since $n \geq n_0$)

$= 5(1 + n^2)$

$= c(1 + n^2)$

Examples

- Prove that 2n + 3 is O(n)

We must find values c, n_0 s.t. $2n + 3 \leq c \cdot n$ for all $n \geq n_0$

Option 1:

$$2n + 3 \overset{\text{if } n \geq 1}{\leq} 2n + 3 \cdot n = 5 \cdot n = \underbrace{c}_{\text{where } c = 5} \cdot n$$

So, pick $n_0 = 1$ and $c = 5$, then $2n + 3 \leq c \cdot n$ for all $n \geq n_0 \implies 2n + 3$ is $O(n)$

Option 2:

$$2n + 3 \overset{\text{if } n \geq 3}{\leq} 2n + n = 3 \cdot n = \underbrace{c}_{c=3} \cdot n$$

$n_0 = 3$ and $c = 3$.

Examples

- Prove that $f(n) = 10^{100}$ is O(1)

Prove that $f(n) = 10^{100}$ is $O(\underbrace{1}_{g(n)})$

We need to find n_0 s.t. $f(n) \leq c \cdot g(n) \; \forall n \geq n_0$

$10^{100} \leq 10^{100} \cdot 1 \leq \underbrace{c}_{10^{100}} \cdot 1$

So, if $c = 10^{100}$, $n_0 = 0$, then $f(n) \leq c \cdot g(n) \; \forall n \geq n_0 \implies f(n)$ is $O(g(n))$

$O(1) \implies$ runtime is constant.

Examples

- Prove that n (sin(n) + 1) is O(n)

We need to show that there exist c, n_0 s.t. $n \cdot (sin(n) + 1) \leq c \cdot n \forall n \geq n_0$

$$n(sin(n) + 1) \leq n(1 + 1) = 2 \cdot n$$
$$\implies n(sin(n) + 1) \leq c \cdot n \forall n \geq n_0$$
$$\text{where } c = 2 \;\; n_0 = 0$$
$$\implies n(sin(n) + 1) \text{ is } O(n)$$

Proving that f(n) is *not* O(g(n))

- To prove that f(n) is *not* O(g(n)), one must show that for any n_0 and c, there exists an $n \geq n_0$ such that f(n) > c g(n)
- Procedure: Assume n_0 and c are given, and find a value of n such that f(n) > c g(n). The value of n will usually depend on n_0 and c

Examples

- Prove that n^2 is *not* O(n)

Prove that n^2 is not $O(100 \cdot n)$

Suppose someone proposes n_0 and c and claims that $f(n) = n^2 \leq c \cdot g(n) = c \cdot 100 \cdot n \forall n \geq n_0$

We need to find $\underbrace{\text{a value for } n}_{\text{dependent on } c}$ s.t. $f(n) \nleq c \cdot g(n) \iff f(n) > c \cdot g(n)$

We want $n^2 > c \cdot 100 \cdot n \iff n > c \cdot 100$

Choose $n = c \cdot 100 + 1$, then $f(n) > c \cdot g(n) \implies f(n)$ is not $O(g(n))$.

Did not mention n_0. To be more precise: $n = \max(n_0, c \cdot 100 + 1)$

Examples

- Prove that n (sin(n) + 1) is O(n)

Examples

- Prove that 2^n is *not* $O(n^3)$