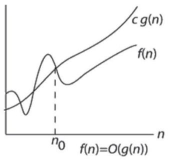
11/30/23

Asymptotic upper bound

Big-O is like $\leq$:

$x = O(x)$
$x = O(x^2)$
$x = O(2^x)$
$\log x = O(x)$
$2^x = O(2^x)$
$2^x = O(3^x)$



Asymptotic lower bound

Big-Ω ("Big-Omega") is **the opposite**, like $\geq$.

$f(x) = O(g(x))$ is the same thing as $g(x) = \Omega(f(x))$

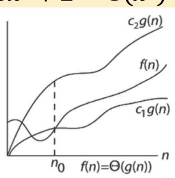
$x = \Omega(x)$
$x = \Omega(\log x)$
$2^x = \Omega(2^x)$
$2^x = \Omega(x)$

Asymptotic tight bound

Big-Θ is like $=$:

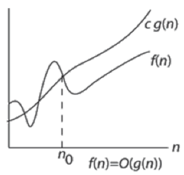
$f(x) = \Theta(g(x))$
means
$f(x) = O(g(x))$
AND
$f(x) = \Omega(g(x))$

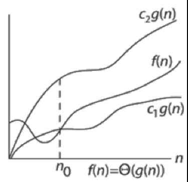
$2^x = \Theta(2^x)$
$3x^2 + 2 = \Theta(x^2)$



L26: Big-O, Big-Omega and Big-Theta

- Let $f, g : \mathbb{R}^+ \to \mathbb{R}^+$ read: f is the big O of g
- **Big-O:** "f is $O(g)$"
  1. Means f grows no faster than as g
     - $\exists k, c$ such that for all $n \geq k$: $f(n) \leq cg(n)$
- **Big-Omega** "f is $\Omega(g)$"
  2. Means f grows as least as fast as g
     - $\exists k, c$ such that for all $n \geq k$: $f(n) \geq cg(n)$
- ***Big-Theta*:** "f is $\Theta(g)$"
  - Means f grows at the same rate as g
  3. f is $\Theta(g)$ iff $f = O(g)$ and $f = \Omega(g)$
     - $\exists k, c_1, c_2$ such that for all $n > k$: $c_1 g(n) \leq f(n) \leq c_2 g(n)$





Exercises

1. $f(n) = 5n^3$. Which are true? DE
   (A) $f = O(\log n)$ (B) $f = O(n)$ (C) $f = O(n^2)$ (D) $f = O((n+1)^3)$ (E) $f = O(n^3)$
2. $f(n) = n^2 + (n+1)^2 + (n+2)^2 - 100n$. CDE
   (A) $f = O(\log n)$ (B) $f = O(n)$ (C) $f = O(n^2)$ (D) $f = O((n+1)^3)$ (E) $f = O(n^3)$
   → tightest bound
3. $f(n) = \log(5n^3)$. So f is $\Theta($ n^3 $)$.
4. $f(n) = (n+1)^3 - n^3 + 5n + 5{,}000$ $= 3n^2 + 8n + 5001$
   So f is $\Theta($ n^2 $)$
5. Which is **not** true for the f in Question 4? E
   (A) $f = \Omega(1)$ (B) $f = \Omega(\log n)$ (C) $f = \Omega(n)$ (D) $f = \Omega(n^2)$ (E) $f = \Omega(n^3)$

$1, \log n, n, n\log n, n^2, n^3, \ldots a^n, n!$

better ← O → worse

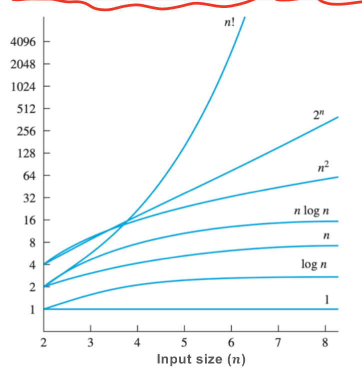
← Ω →

Big-Theta "Cheat Sheet"

Runtime comparison of standard functions

better worse

4. $\Theta(1)$, $\log n$, n, $n\log n$, n^2, n^3, (maybe $n^4, \ldots$), 2^n, $n!$



Scalar multiplication: ignore scalar coefficients
$f(n) = 1000n^3$
is $\Theta($ n^3 $)$

Addition: keep largest term
$f(n) = n^2 + \log n$
is $\Theta($ n^2 $)$

Product: keep all terms
$f(n) = n^2 \log n$
is $\Theta($ $n^2\log n$ $)$

Consider positive-valued functions $f_1(n) = \Theta(g_1(n))$ and $f_2(n) = \Theta(g_2(n))$.

5
- **Addition**
  $(f_1 + f_2)(n) = \Theta(\max(g_1(n), g_2(n)))$
- **Scalar multiplication**
  $af(n) = \Theta(f(n))$
- **Product**
  $(f_1 \cdot f_2)(n) = \Theta(g_1(n) \cdot g_2(n))$

Even More Examples

Consider functions f and g with $f, g \geq 0\ \forall n$ where:

- f is $O(n^3)$ and $\Omega(n)$
- g is $\Theta(\log n)$

Fill in the blanks by finding the largest lower bound and smallest upper bound on h_1 and h_2.

a) If $f + g$ is $\Theta(h_1(n))$, then n $\leq h_1(n) \leq$ n^3

b) If $f \cdot g$ is $\Theta(h_2(n))$, then $n\log n$ $\leq h_2(n) \leq$ $n^3 \log n$

$f_1(n) = \Theta(g_1(n))$ and $f_2(n) = \Theta(g_2(n))$ means there exists constants c_1, c_2, k_1 and c_3, c_4, k_2 such that
$c_1 g_1(n) \leq f_1(n) \leq c_2 g_1(n)$ for all $n \geq k_1$
$c_3 g_2(n) \leq f_2(n) \leq c_4\, g_2(n)$ for all $n \geq k_2$

Adding these inequalities gives
$c_1 g_1(n) + c_3 g_2(n) \leq (f_1 + f_2)(n) \leq c_2 g_1(n) + c_4 g_2(n)$ for all $n \geq \max(k_1, k_2)$

Now $c_2 g_1(n) + c_4 g_2(n) \leq (c_2 + c_4)\max(g_1(n) + g_2(n))$
and $c_1 g_1(n) + c_3 g_2(n) \geq c\, \max(g_1(n), g_2(n))$ for c = min(c_1,c_2)

This gives $(f_1 + f_2)(n) = \Theta(\max(g_1(n), g_2(n))$

Time Complexity of Algorithms

Find the Big-Θ runtime of each algorithm, given its pseudocode:

```
(a) procedure hello_goodbye(n: integer)
        sum := 0
        for i := 1 to 2n
            print "hello world"
            sum := sum + 2

        for i := 1 to n
            for j := 1 to n
                print "goodbye"
                sum := sum + 1
```

$2n$ times $\to \Theta(n)$

$+$

n^2 times $\to \Theta(n^2)$

$\Downarrow$

$\Theta(n^2)$

```
(b) procedure foo(n: integer)
        a := 0
        i := 1
        while i < n
            a := a + i
            i := i * 2
        return a
```

```
procedure square_matrix_mult(A: nxn matrix, B: nxn matrix)
  for i := 1 to n
    for j := 1 to n
      c_ij := 0
      for k := 1 to n
        c_ij := c_ij + a_ik * b_kj
  return C
```

n n n

$\Theta(n^3)$

Time Complexity of Linear Search

```
procedure linear_search(x: integer, a1, a2, …, an: integers)
    i := 1

    while(i ≤ n and x ≠ a_i    )
        i := i + 1

    if i ≤ n then location := i
    else location := 0

    return location
```

at most n times

$\Rightarrow \Theta(n)$

Time Complexity of Binary Search

procedure *binary_search* (x: integer, a_1, a_2, … , a_n: increasing integers)

```
i := 1      {i is left endpoint of search interval}
j := n      {j is right endpoint of search interval}
while i < j
    m := ⌊(i + j)/2⌋
    if x > a_m  then i:=m+1
    else j := m
if x = a_i  then location := i
else location := 0
return location {location is the subscript i of the term a_i
                 equal to x, or 0 if x is not found}
```

find the midpoint
update i to midpoint, or
update j to midpoint

Updates cut remaining list in half each time:
$j - i \approx n$, then $n/2$, then $n/4$, …
So loop iterates $\log n$ times.

$\Rightarrow \Theta(\log n)$

Let $T(n)$ be the number of steps needed.

$\Rightarrow T(n) = T(\frac{n}{2}) + 1 = T(\frac{n}{4}) + 1 + 1$
$= T(\frac{n}{8}) + 1 + 1 + 1 = \ldots = T(\frac{n}{2^k}) + k$

$T(1) = 0 \Rightarrow k = \log_2 n$

$\Rightarrow T(n) = T(1) + \log_2 n = \log_2 n$

```
procedure bar(n: integer)
  a := (n * n - 7) / 2
  for i := 1 to n
    j := n
    while j > 1
      print "hi"
      j := j / 2
  print "bye"

  for i := 1 to 500n
    print "203 is fun!"
```

$\log n$ n $\Rightarrow \Theta(n \log n)$

$500n \Rightarrow \Theta(n)$

Putting it all together, the algorithm is $\boldsymbol{\Theta(n \log n)}$

(who dominates)

Examples: Runtimes Galore!

Determine the Θ estimate for each of the following functions.

a) $f(n) = (n^3 + n^2)(n^2 + 50{,}000)$

n^3 n^2

$= \Theta(n^5)$

b) $f(n) = \left(5n + \frac{n}{2} + 7\right)(3n^4 + 8n!)$

n $n!$

$= \Theta(n!\,n)$

c) $f(n) = (n^5 + 2^n + \log n)(3n + 4n \log n)$

2^n $n \log n$

$= \Theta(2^n \cdot n \log n)$

d) $f(n) = 10n(\log n^2 + \log n^3)(n^2 + 1)$

$2\log n + 3\log n$ n^2

$\log n$

$= \Theta(n^3 \log n)$

b/c $\log n^k = k \log n$

$= \Theta(\log n)$

e) $f(n) = (3^n + n!)(2^n + n^2)$

$n!$ 2^n

$= \Theta(n!\,2^n)$

f) $f(n) = (3^n + n!) + (2^n + n^2)$

$n!$ 2^n

$n!$

$= \Theta(n!)$

g) $f(n) = n^2 \cdot 3^n + \frac{n}{3} \cdot n^3 \log n$

$n^2 3^n$ $n^4 \log n$

$n^2 3^n$

$= \Theta(n^2 3^n)$