The Growth of Functions

Niloufar Shafiei

Algorithms

An **algorithm** is a finite set of precise instructions for performing a computation or for solving a problem.

Algorithms can be described using English language, programming language, pseudo-code, …

Algorithms (example)

Describe an algorithm for finding the maximum value in a finite sequence of integers.

Solution:

- Set the temporary maximum equal to the first integer in the sequence.
- Compare the next integer in the sequence to the temporary maximum, and if it is larger than the temporary maximum, set the temporary maximum equal to this integer.
- Repeat the previous step if there are more integers in the sequence
- Stop when there are no integers left in the sequence.
- The temporary maximum at this point is the largest integer in the sequence.

Algorithms (example)

Describe an algorithm for finding the maximum value in a finite sequence of integers.

Solution:

Procedure max(a_1, a_2, a_3, …, a_n: integers)

 max = a_1

 for i=2 **to** n

  if max $<$ a_i **then** max = a_i

 output max

Number of steps:

$$1 + (n - 1) + (n - 1) + 1 = 2n$$

Algorithms

The **time** required to solve a problem depends on the number of steps it uses.

G**rowth functions** are used to estimate the number of steps an algorithm uses as its input grows.

Worst-case complexity

The largest number of steps needed to solve the given problem using an algorithm on input of specified size is **worst-case complexity**.

Example:

Design an algorithm to determine if finite sequence $a_1,a_2,\ldots,a_n$ has term 5.

Procedure search(a_1, a_2, a_3, …, a_n: integers)

  for i=1 **to** n

    if a_i=5 **then output** True

  output False

Worst-case complexity:

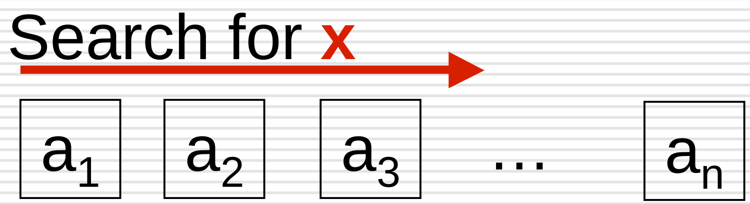
$$n + n + 1 = 2n + 1$$

Algorithm complexity

To describe running time of algorithms, we consider the size of input is very large.

So, we can ignore constants in describing running time.

Linear Search



Procedure linear search(x: integer, a_1, a_2, a_3, …, a_n: integers)

While ($i \leq n$ **and** $x \neq a_i$)

$i = i + 1$

if $i \leq n$ **then** location = i

else location = 0

output location

Worst-case complexity:

2n + 2

Binary Search

Assume $a_1 \leq a_2,\ a_2 \leq a_3,\ \ldots,\ a_{n-1} \leq a_n$.

a_1	a_2	a_3	…	a_n

Search for **18**

1	3	5	12	13	18	20

12<18

1	3	5	**12**	13	18	20

18=18

1	3	5	12	13	**18**	20

Binary Search

Procedure binary search(x: integer, a_1, a_2, a_3, …, a_n: increasing integers)

i = 1

j = n

m = $\lfloor (i+j)/2 \rfloor$

while ($a_m \neq x$ and $i \leq j$)

begin

m = $\lfloor (i+j)/2 \rfloor$

if x > a_m **then** i = m+1 **else** j = m-1

end

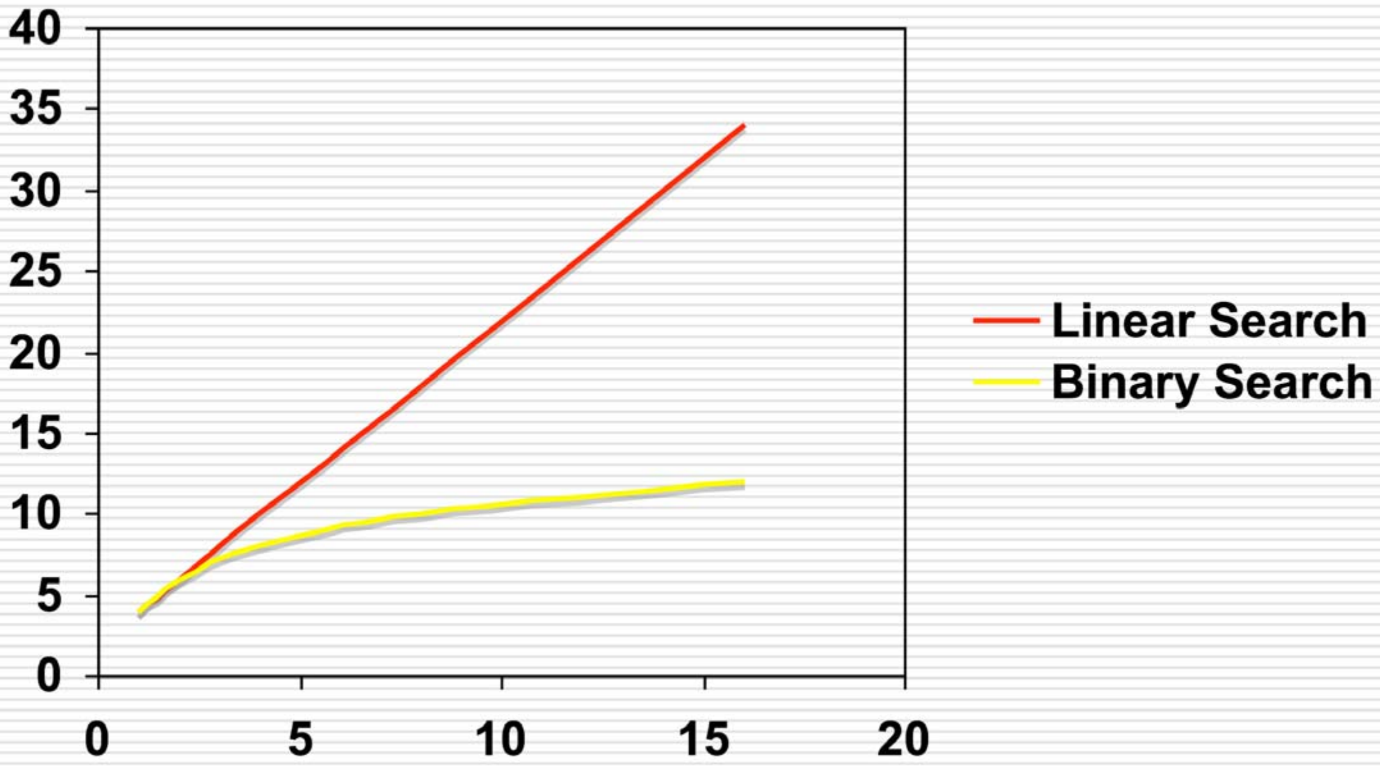
if x = a_m **then** location = m **else** location = 0

output location

Worst-case complexity:

3 + 3log(n) + 2

Linear search vs. binary search



Big-O notation

Assume f:**Z/R**→**R** and g:**Z/R**→**R**.

f(x) is **O(g(x))** if ∃ constants C and k such that

$$|f(x)| \leq C|g(x)|$$

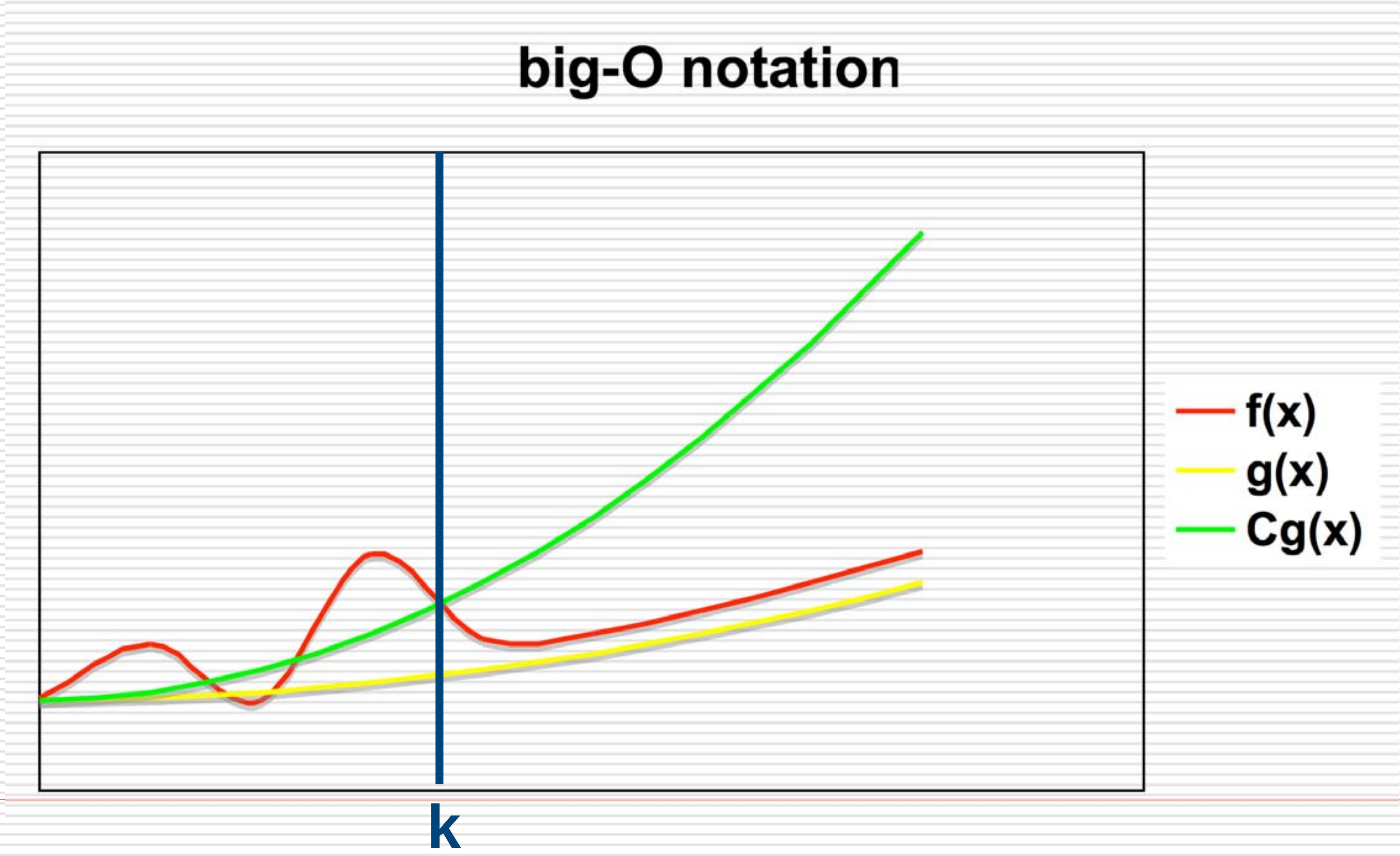
∀ x>k.

Constants C and k are called **witnesses**.

When there is one pair of witnesses, there are infinitely many pairs of witnesses.

Big-O notation

If **|f(x)| ≤ C|g(x)|, ∀x>k** then **f(x) = O(g(x))**



Big-O notation (example)

Show that $f(x)=x^2 + 2x + 1$ is $O(x^2)$.

Solution:

$|f(x)| \leq C|x^2| \quad \forall\ x>k$

$|x^2 + 2x + 1| \leq C(x^2) \quad \forall\ x>k$

$x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 \quad \forall\ x>1$

$x^2 + 2x + 1 \leq 4(x^2) \quad \forall\ x>1$

k = 1 and C = 4

$f(x) = O(x^2)$ or $f(x) \in O(x^2)$

Big-O notation

$x^2 + 2x + 1 \le 4(x^2)$ $\forall$ $x>1$

$x^2 + 2x + 1 = O(x^2)$

$x^2 = O(x^2 + 2x + 1)$

$x^2 \le x^2 + 2x + 1$ $\forall$ $x>1$

So, x^2 and $x^2 + 2x + 1$ are of the **same order**.

Big-O notation

Assume $f(x) = O(g(x))$ and $g(x) < g'(x)$ for large x. Show $f(x) = O(g'(x))$.

Proof:

$|f(x)| \leq C|g(x)| \quad \forall x>k$

$|f(x)| \leq C|g(x)| < C|g'(x)| \quad \forall x>k$

So, $f(x) = O(g'(x))$.

Example:

$x^2 + 2x + 1 = O(x^2) \qquad x^2 + 2x + 1 = O(x^3)$

When $f(x) = O(g(x))$, usually $g(x)$ is chosen to be a simple function that is as small as possible.

Big-O notation (example)

Show that $f(x)=7x^2$ is $O(x^3)$.

Solution:

$|f(x)| \le C|x^3| \quad \forall\ x>k$

$7x^2 \le 7x^3 \quad \forall\ x>1$

k = 1 and C = 7

$f(x) = O(x^3)$ or $f(x) \in O(x^3)$

Big-O notation (example)

Show that n^2 is not $O(n)$.

Solution:

- Show $\neg(\exists\ (C,k)\ \forall\ n>k\ \ (n^2 \leq Cn))$.

  $\forall(C,k)\ \exists n>k\ \neg(n^2 \leq Cn)$

$n^2 \leq Cn \qquad \forall\ n>k$

$n \leq C \qquad \forall\ n>k$

No matter what C and k are, the inequality $n \leq C$ cannot hold for all n with $n>k$.

Big-O notation (example)

Is it true that x^3 is $O(7x^2)$.

Solution:

- Determine whether witnesses exist or not.

$x^3 \le C(7x^2)$ whenever $x>k$

$x \le 7C$ whenever $x>k$

No matter what C and k are, the inequality $x \le 7C$ cannot hold for all n with $n>k$.

So, x^3 is not $O(7x^2)$.

Growth of polynomial functions

The leading term of a polynomial function determines its growth.

Let $f(x)=a_nx^n + a_{n-1}x^{n-1} + \ldots + a_1x + a_0$, where $a_n, a_{n-1}, \ldots, a_1, a_0$ are real numbers.

Then $f(x)$ is **$O(x^n)$.**

Growth of polynomial functions

Let $f(x)=a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$, where $a_n, a_{n-1}, \ldots, a_1, a_0$ are real numbers.

Show $f(x)$ is $O(x^n)$.

Proof:

- Show $\exists(C,k)$ that $|f(x)| \le C|g(x)| \ \forall x>k$.

$|f(x)| = | a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0|$

Assume $x>1$.

$|f(x)| = | a_n| x^n + |a_{n-1}| x^{n-1} + \ldots + |a_1| x + |a_0|$

$= x^n (| a_n| + |a_{n-1}| /x + \ldots + |a_1|/ x^{n-1} + |a_0|/ x^n)$

$\le x^n (| a_n| + |a_{n-1}| + \ldots + |a_1| + |a_0|)$

Let $C = | a_n| + |a_{n-1}| + \ldots + |a_1| + |a_0|$ and $k=1$.

So, $|f(x)| \le C|g(x)| \ \forall x>k$ and $f(x)=O(x^n)$.

Example

Give big-O estimates for the factorial function $f(n)=n!$ and $g(x) = \log(n!)$.

$$(n! = 1 . 2 . 3 . \ldots . (n-1) . n)$$

Proof:

$n! = 1 . 2 . 3 . \ldots . (n-1) . n$

$\leq n . n . n . \ldots . n . n = n^n$

$f(n) = O(n^n)$ taking C=1 and k=1.

$\log n! \leq \log n^n = n \log n$

$g(n) = O(n \log n)$ taking C=1 and k=1.

Example

Show $n = O(2^n)$ and $\log n = O(n)$.

Proof:

$n < 2^n$ where k=1 and C=1

$n = O(2^n)$

$n < 2^n$

$\log n < \log 2^n = n \log 2$

$\log n < n \log 2$ where k=1 and C=log 2

$\log n = O(n)$

Example

Show $\log_b n = O(n)$.

Proof:

$\log_b n = \log n / \log b$

$< n / \log b$

$\log_b n < n / \log b$ where $k=1$ and $C=1 / \log b$

$\log_b n = O(n)$

The growth of combinations of functions

Many algorithms are made up of several procedures.

The number of steps used by the algorithm with input of specified size is the sum of the number of steps used by all procedures.

The growth of combinations of functions

Assume f(x) = O(g(x)) and f'(x) = O(g'(x)). Give big-O estimate of (f + f')(x).

Solution:

f(x) = O(g(x))

$\exists(C,k) \quad |f(x)| \leq C|g(x)| \quad \forall x>k$

f'(x) = O(g'(x))

$\exists(C',k') \quad |f'(x)| \leq C'|g'(x)| \quad \forall x>k'$

$|(f + f')(x)|$

$= |f(x) + f'(x)|$

$\leq |f(x)| + |f'(x)|$

$\leq C|g(x)| + C'|g'(x)| \quad \forall x> \max(k,k')$

The growth of combinations of functions

Assume $f(x) = O(g(x))$ and $f'(x) = O(g'(x))$. Give big-O estimate of $(f + f')(x)$.

Solution:

$|(f + f')(x)| \leq C|g(x)| + C'|g'(x)| \quad \forall x > \max(k,k')$

Assume $h(x) = \max(|g(x)|,|g'(x)|)$

$|(f + f')(x)| \leq C|g(x)| + C'|g'(x)| \quad \forall x > \max(k,k')$

$\leq C|h(x)| + C'|h(x)| \quad \forall x > \max(k,k')$

$\leq (C+C')\,|h(x)| \quad \forall x > \max(k,k')$

Assume $a = C + C'$ and $b = \max(k,k')$.

$|(f + f')(x)| \leq a\,|h(x)| \quad \forall x > b$

$(f + f')(x) = O(\max(|g(x)|,|g'(x)|))$

The growth of combinations of functions

Assume $f(x) = O(g(x))$ and $f'(x) = O(g'(x))$.
Then $(f+f')(x) = O(\max(|g(x)|,|g'(x)|))$.

Assume $f(x) = O(g(x))$ and $f'(x) = O(g(x))$.
Then $(f+f')(x) = O(g(x))$.

The growth of combinations of functions

Assume $f(x) = O(g(x))$ and $f'(x) = O(g'(x))$. Give big-O estimate of $(f \cdot f')(x)$.

Solution:

$f(x) = O(g(x))$

$\exists(C,k) \quad |f(x)| \le C|g(x)| \quad \forall x>k$

$f'(x) = O(g'(x))$

$\exists(C',k') \quad |f'(x)| \le C'|g'(x)| \quad \forall x>k'$

$|(f \cdot f')(x)|$

$= |f(x) \cdot f'(x)|$

$= |f(x)| \cdot |f'(x)|$

$\le C|g(x)| \cdot C'|g'(x)| \quad \forall x> \max(k,k')$

The growth of combinations of functions

Assume $f(x) = O(g(x))$ and $f'(x) = O(g'(x))$. Give big-O estimate of $(f \,.\, f')(x)$.

Solution:

$|(f \,.\, f')(x)| \le C|g(x)| \,.\, C'|g'(x)|$ $\quad \forall x > \max(k,k')$

$\le C \,.\, C' \,|g(x)| \,.|g'(x)|$ $\quad \forall x > \max(k,k')$

$\le C \,.\, C' \,|(g \,.\, g')(x)|$ $\quad \forall x > \max(k,k')$

Assume $h(x) = (g \,.\, g')(x)$, $a = C \,.\, C'$ and $b = \max(k,k')$.

$|(f \,.\, f')(x)| \le a\,|h(x)|$ $\quad \forall x > b$

$(f \,.\, f')(x) = O((g \,.\, g')(x))$

The growth of combinations of functions

Assume $f(x) = O(g(x))$ and $f'(x) = O(g'(x))$.
Then $(f \cdot f')(x) = O((g \cdot g')(x))$.

Example

Give a big-O estimate for $f(n)=3n \log(n!) + (n^2 + 3)$, where n is a positive integer .

Solution:

By previous example, $\log(n!) = O(n \log n)$.

By theorem, $3n = O(n)$.

By theorem, $3n \log(n!) = O(n . n\log n)$.

$3n \log(n!) = O(n^2 \log n)$

$n^2 + 3 \leq 2n^2$ $\forall$ n>2 and C =2

So, $(n^2 + 3) = O(n^2)$

By theorem, $3n \log(n!) + (n^2 + 3) = O(\max(n^2 \log n, n^2))$.

So, $3n \log(n!) + (n^2 + 3) = O(n^2 \log n)$.

Example

Give a big-O estimate for $f(n) = n^2 \log(n^3 + 1) + (n^3 + 4n^2 + 5)$, where n is a positive integer .

Solution:

By theorem, $n^2 = O(n^2)$.

$\log(n^3 + 1)$

$\leq \log(n^3 + n^3)$ $\forall$ n>1

$= \log(2n^3)$ $\forall$ n>1

$= \log 2 + \log n^3 = \log 2 + 3\log n$ $\forall$ n>1

$\leq 3\log(n)$ $\forall$ n>2

Let C be 3 and k be 2. So, $\log(n^3 + 1) = O(\log n)$.

By theorem, $n^2 \log(n^3 + 1) = (n^2 \log n)$

By theorem, $(n^3 + 4n^2 + 5) = O(n^3)$.

By theorem, $f(n) = O(\max(n^2 \log n , n^3)) = O(n^3)$.

Big-Omega

Assume $f:\mathbf{Z}/\mathbf{R}\rightarrow\mathbf{R}$ and $g:\mathbf{Z}/\mathbf{R}\rightarrow\mathbf{R}$.

f(x) is $\Omega(\mathbf{g(x)})$ if $\exists$ positive constants C and k such that

$$|f(x)| \geq C|g(x)| \qquad \forall\, x>k.$$

Big-O and big-Omega

Big-O provides upper bound for functions.

Big-Omega provides lower bound for functions.

Example

Let f(x) be $5x^4+x^2+8$. Show $f(x) = \Omega(x^4)$.

Solution:

$\exists(C,k)$ $5x^4+x^2+8 \geq Cx^4$ $\forall x>k$

$5x^4+x^2+8 \geq x^4$ $\forall x>1$ and $C=1$

So, $f(x) = \Omega(x^4)$.

Example

Give big- Ω for $\log(n^4) + 2^n$.

Solution:

$\log(n^4) + 2^n = 4\log n + 2^n$

$\geq \log n \qquad \forall n>1 \text{ and } C=1$

$\log(n^4) + 2^n = \Omega(\log n)$

Big-O and big-Omega

Show $f(x)=O(g(x))$ if and only if $g(x)=\Omega(f(x))$.

Proof:

- Show if $f(x)=O(g(x))$ then $g(x)=\Omega(f(x))$.

$f(x)=O(g(x))$

$|f(x)| \le C|g(x)| \quad \forall\ x>k$

$(1/C)|f(x)| \le |g(x)| \quad \forall\ x>k$

So, $g(x)=\Omega(f(x))$.

Big-O and big-Omega

Show $f(x)=O(g(x))$ if and only if $g(x)=\Omega(f(x))$.

Proof:

- Show if $g(x)=\Omega(f(x))$ then $f(x)=O(g(x))$.

$g(x)=\Omega(f(x))$

$|g(x)| \geq C|f(x)| \quad \forall\ x>k$

$(1/C)|g(x)| \geq |f(x)| \quad \forall\ x>k$

So, $f(x)=O(g(x))$.

Big-Theta

Assume f:**Z**/**R**→**R** and g:**Z**/**R**→**R**.

f(x) is Θ**(g(x))** if f(x)=O(g(x)) and f(x)=Ω(g(x)), we say f is big-Theta of g(x) and we also say that f(x) is of order g(x).

f(x) = Θ(g(x)), then

$\exists$ (C, k) such that $|f(x)| \le C|g(x)|$ $\forall$ x>k and

$\exists$ (C', k') such that $|f(x)| \ge C'|g(x)|$ $\forall$ x>k'.

Big-Theta provides both upper and lower bounds for functions.

Big-Theta

$f(x) = \Theta(g(x))$ if and only if $g(x) = \Theta(f(x))$.

$f(x) = \Theta(g(x))$ if and only if $f(x) = O(g(x))$ and $g(x) = O(f(x))$.

(Prove these facts as exercises.)

Example

Show 1+2+…+n is $\Theta(n^2)$.

Solution:

1+2+…+n

$\leq n+n+\ldots+n = n^2 \qquad \forall n>1$ and C=1

So, $1+2+\ldots+n = O(n^2)$.

1+2+…+n

$\geq \lceil n/2 \rceil + (\lceil n/2 \rceil+1) + \ldots + n$

$\geq \lceil n/2 \rceil + \lceil n/2 \rceil + \ldots + \lceil n/2 \rceil$

$\geq (n/2)\lceil n/2 \rceil$

$\geq (n/2) . (n/2) = n^2 / 4 \quad \forall n>1$ and C=1/4

So, $1+2+\ldots+n = \Omega(n^2)$.

Thus, $1+2+\ldots+n = \Theta(n^2)$.

Example

Show $n^2 + 5n \log n$ is $\Theta(n^2)$.

Solution:

$5n \log n$

$\leq 5 n^2 \qquad \forall n>1$ and $C=5$

$n^2 + 5n \log n \leq 6n^2 \qquad \forall n>1$ and $C=6$

So, $n^2 + 5n \log n = O(n^2)$

$n^2 + 5n \log n \geq n^2 \qquad \forall n>1$ and $C=1$

So, $n^2 + 5n \log n = \Omega(n^2)$.

Thus, $n^2 + 5n \log n = \Theta(n^2)$.

Big-Theta of polynomial functions

Let $f(x)=a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$, where $a_n, a_{n-1}, \ldots, a_1, a_0$ are real numbers with $a_n \neq 0$.

Then $f(x)$ is of order x^n.

Example

Give big-Theta estimates for the following functions.

- $500\ x^9 + x^2 + 5$

  $= \Theta(x^9)$

- $0.00007\ x^2 + x + 25000$

  $= \Theta(x^2)$

- $-10\ x^5 - 250\ x^2$

  $= \Theta(x^5)$

Recommended exercises

2,4,5,7,9,12,14,15,18,19,21,24,27,29,33,35,
39,41,43,61