Chapter 3
Algorithms

MAD101

Vo Tran Duy

duyvt15@fe.edu.vn

Table of Contents

Example.

1 Algorithms

- Finding the Maximum Element in a Finite Sequence $\{8, 4, 11, 3, 10\}$.
- To search for 19 in the list $1, 2, 3, 5, 6, 7, 8, 10, 12, 13, 15, 16, 18, 19, 20, 22$
- ...

Definition

1 Algorithms

An **algorithm** is a finite sequence of precise instructions (mã lệnh xác định) for performing a computation or for solving a problem.

PROPERTIES OF ALGORITHMS

- **Input (Đầu vào).** An algorithm has input values from a specified set.
- **Output (Đầu ra).** From each set of input values an algorithm produces output values from a specified set (solution).
- **Definiteness (xác định).** The steps of an algorithm must be defined precisely.
- **Correctness (chính xác).** An algorithm should produce the correct output values for each set of input values.
- **Finiteness (hữu hạn).** An algorithm should produce the desired output after a finite (but perhaps large) number of steps for any input in the set.
- **Effectiveness (hiệu quả).** It must be possible to perform each step of an algorithm exactly and in a finite amount of time.
- **Generality (tổng quát).** The procedure should be applicable for all problems of the desired form.

Finding the Maximum Element in a Finite Sequence.

procedure $max(a_1, a_2, \ldots, a_n:$ integers)
$max := a_1$
for $i := 2$ **to** n
  if $max < a_i$ **then** $max := a_i$
return $max\{max$ is the largest element$\}$

Example. Finding the Maximum Element in a Finite Sequence $\{8, 4, 11, 3, 10\}$.

procedure max $(a_1, a_2, a_3, a_4, a_5$: integers)
max:=a_1
for i:=2 **to** 5
if max $< a_i$ **then** max:= a_i
return max {max is the largest element}

Example.

$\{8, 4, 11, 3, 10\}$

Max=8

$i = 2(8 \geq 4) \rightarrow$ Max=8

$i = 3(8 < 11) \rightarrow$ Max=11

$i = 4(11 \geq 3) \rightarrow$ Max=11

$i = 5(11 \geq 10) \rightarrow$ Max=11

Thus, Max=11

The Linear Search Algorithm

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \le n$ and $x \ne a_i$)
    $i := i + 1$
if $i \le n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

Example. List all the steps used to search for 9 in the sequence $2, 3, 4, 5, 6, 8, 9, 11$ using a **linear search**. How many comparisons required to search for 9 in the sequence.

Solution

Below is the linear search algorithm in pseudocode

$2, 3, 4, 5, 6, 8, 9, 11$

procedure linear search (x: integer, $a_1, a_2, ..., a_8$: distinct integers)

$i := 1$

while $(i \leq 8$ and $x \neq a_i$)

$i := i + 1$

if $i \leq 8$ **then** location: $= i$

else location: $= 0$

return location {location is the subscript of the term that equals x, or is 0 if x is not found}

$2, 3, 4, 5, 6, 8, 9, 11$

i = 1

(1 ≤ 8 and 9 ≠ 2) ➔ i:=i+1 = 2

i = 2

(2 ≤ 8 and 9 ≠ 3) ➔ i:= i+1 = 3

i = 3

(3 ≤ 8 and 9 ≠ 4) ➔ i:= i+1 = 4

i = 4

(4 ≤ 8 and 9 ≠ 5) ➔ i:= i+1 = 5

i = 5

(5 ≤ 8 and 9 ≠ 6) ➔ i:= i+1 = 6

i = 6

(6 ≤ 8 and 9 ≠ 8) ➔ i:= i+1 = 7

i = 7

(7 ≤ 8 and 9 ≠ 9) // the condition is false

7 ≤ 9 ➔ location = 7.

Based on the steps above, there are 15 comparisons (≤, ≠) required.

The Binary(nhị phân) Search Algorithm

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    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}

Example. To search for 19 in the list $1, 2, 3, 5, 6, 7, 8, 10, 12, 13, 15, 16, 18, 19, 20, 22$

$1, 2, 3, 5, 6, 7, 8, 10, 12, 13, 15, 16, 18, 19, 20, 22.$ Search 19

procedure binary search (x:integer, $a_1, a_2, \ldots, a_{16}$: increasing integers)

$i := 1$ { i is left endpoint of search interval}

$j := 16$ { j is right endpoint of search interval}

while $i < j$

$m := \lfloor (i + j)/2 \rfloor$
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 of the term that equals x, or is 0 if x is not found}

$i = 1,\ j = 16$
while $i < j$

$$m = \lfloor \frac{1+16}{2} \rfloor = \lfloor 8.5 \rfloor = 8$$

$\color{red}{19} > 10 \to i = 8 + 1 = 9,\ 19 \neq 12 \to$location:=0

$$m = \lfloor \frac{9+16}{2} \rfloor = \lfloor 12.5 \rfloor = 12$$

$\color{red}{19} > 16 \to i = 12 + 1 = 13,\ 19 \neq 18 \to$location:=0

$$m = \lfloor \frac{13+16}{2} \rfloor = \lfloor 14.5 \rfloor = 14$$

$\color{red}{19} \leq 19 \to \color{red}{j = 14},\ 19 \neq 18 \to$location:=0

$$m = \lfloor \frac{13+14}{2} \rfloor = \lfloor 13.5 \rfloor = 13$$

$\color{red}{19} > 18 \to i = 13 + 1 = 14$

$\color{red}{19} = 19 \to$ location $= 14$

Sorting

Sorting is putting the elements into a list in which the elements are in increasing order.
For instance, sorting the list 7, 2, 1, 4, 5, 9 produces the list 1, 2, 4, 5, 7, 9. Sorting the list d, h, c, a, f (using alphabetical order) produces the list a, c, d, f, h.

The Bubble Sort

procedure *bubblesort*($a_1, \ldots, a_n$: real numbers with $n \geq 2$)
for $i := 1$ **to** $n - 1$
    for $j := 1$ **to** $n - i$
        if $a_j > a_{j+1}$ **then** interchange a_j and a_{j+1}
$\{a_1, \ldots, a_n$ is in increasing order$\}$

Example. Use the bubble sort to put $3, 2, 4, 1, 5$ into increasing order.

procedure bubble sort ($a_1, a_2, \ldots, a_5$:real numbers with $5 \geq 2$)
for $i := 1$ **to** 4
for $j := 1$ **to** $5 - i$
if $a_j > a_{j+1}$ **then** interchange a_j and a_{j+1}
$\{a_1, a_2, \ldots, a_5$ is in increasing order$\}$

Use the bubble sort to put $3, 2, 4, 1, 5$ into increasing order

$i = 1$

$j = 1 : 3 > 2 \to 2, 3, 4, 1, 5$

$j = 2 : 3 \leq 4 \to 2, 3, 4, 1, 5$

$j = 3 : 4 > 1 \to 2, 3, 1, 4, 5$

$j = 4 : 4 \leq 5 \to 2, 3, 1, 4, 5$

$i = 2$

$j = 1 : 2 \leq 3 \to 2, 3, 1, 4, 5$

$j = 2 : 3 > 1 \to 2, 1, 3, 4, 5$

$j = 3 : 3 \leq 4 \to 2, 1, 3, 4, 5$

$i = 3$

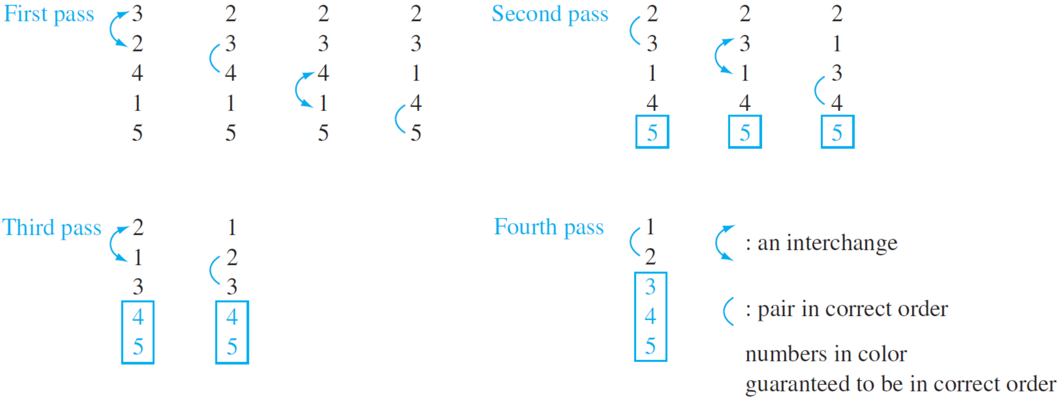
$j = 1 : 2 > 1 \to 1, 2, 3, 4, 5$

$j = 2 : 2 \leq 3 \to 1, 2, 3, 4, 5$

$i = 4$

$j = 1 : 1 \leq 2 \to 1, 2, 3, 4, 5$

Use the bubble sort to put $3, 2, 4, 1, 5$ into increasing order



The Insertion Sort

procedure *insertion sort*($a_1, a_2, \ldots, a_n$: real numbers with $n \geq 2$)
for $j := 2$ **to** n
    $i := 1$
    while $a_j > a_i$
        $i := i + 1$
    $m := a_j$
    for $k := 0$ **to** $j - i - 1$
        $a_{j-k} := a_{j-k-1}$
    $a_i := m$
$\{a_1, \ldots, a_n$ is in increasing order$\}$

($i < j$, while True->compute, False->next)

Example. Use the insertion sort to put the elements of the list 3, 2, 4, 1, 5 in increasing order.

Use the insertion sort to put the elements of the list 3, 2, 4, 1, 5 in increasing order

procedure insertion sort ($a_1, a_2, \ldots, a_5$:real numbers with $5 \geq 2$)
for $j := 2$ **to** 5 *{ j: position of the examined element }*
{ finding out the right position of a_j *}*
    $i := 1$
    while $a_j > a_i$
        $i := i + 1$
    $m := a_j$ *{ save* a_j *}*
    { moving j-i elements backward }
    for $k := 0$ **to** $j - i - 1$
        $a_{j-k} := a_{j-k-1}$
    {move a_j *to the position i}*
    $a_i := m$
$\{a_1, a_2, \ldots, a_5$ is increasing order$\}$

$j = 2$

$i = 1, 2 \leq 3 \rightarrow i = 1, m = 2$

$k = 0 \rightarrow a_2 = 3, a_1 = 2$

$\rightarrow 2, 3, 4, 1, 5$

$j = 3$

$i = 1, 4 > 2$

$i = 2, 4 > 3$

$\rightarrow 2, 3, 4, 1, 5$

$j = 4$

$i = 1, 1 \leq 2 \rightarrow i = 1, m = 1$

$k = 0 \rightarrow a_4 = 4$

$k = 1 \rightarrow a_3 = 3$

$k = 2 \rightarrow a_2 = 2$

$a_1 = 1$

$\rightarrow 1, 2, 3, 4, 5$

$j = 4$

$i = 1, 1 \leq 2 \to i = 1, m = 1$

$k = 0 \to a_4 = 4$

$k = 1 \to a_3 = 3$

$k = 2 \to a_2 = 2$

$a_1 = 1$

$\to 1, 2, 3, 4, 5$

$j = 5$

$i = 1, 5 > 1$

$i = 2, 5 > 2$

$i = 3, 5 > 3$

$i = 4, 5 > 4$

Thus, $\to 1, 2, 3, 4, 5$

Table of Contents

2 The Growth of Functions

Example.

n	constant	log_2n	n	n^2	2^n	n!
1	1	0	1	1	2	1
2	1	1	2	4	4	2
4	1	2	4	16	16	24
8	1	3	8	64	256	977760
16	1	4	16	256	65536	5.073777e+14
32	1	5	32	1024	65536^2	2.6313084e+35

(constant that mean $f(n) = k, \forall n$. In this ex, $k = 1$)

$\rightarrow 1 < log_2n (\equiv logn) < n < n^2 < 2^n < n!$

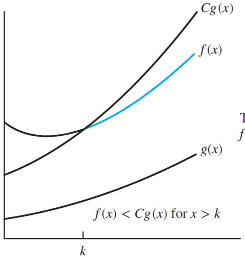
- Growth of $logn <$ growth of $n \rightarrow logn$ is $O(n)$
- In general, growth of $f(n) \leq$ growth of $g(n)$ $\rightarrow f(n)$ is $O(g(n))$

Big-O Notation

Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $O(g(x))$ if there are constants C and k such that

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

whenever $x > k$. [This is read as "$f(x)$ is big-oh of $g(x)$."]



The part of the graph of $f(x)$ that satisfies $f(x) < Cg(x)$ is shown in color.

FIGURE 2 **The function $f(x)$ is $O(g(x))$.**

Example.

2 The Growth of Functions

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

Solution.

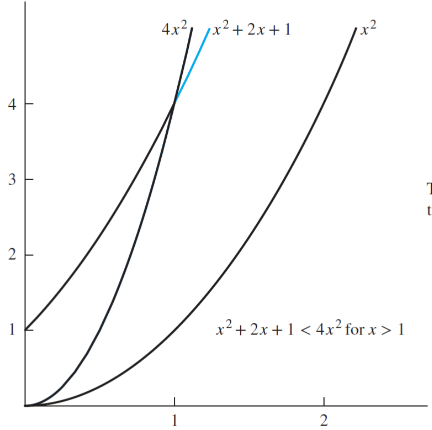
$$\forall x > 1 \implies x^2 > 1 \land x^2 > x$$

$$f(x) = x^2 + 2x + 1 < x^2 + 2x^2 + x^2 = 4x^2$$

Let $g(x) = x^2$

We have $C = 4, k = 1, |f(x)| \leq C|g(x)|$

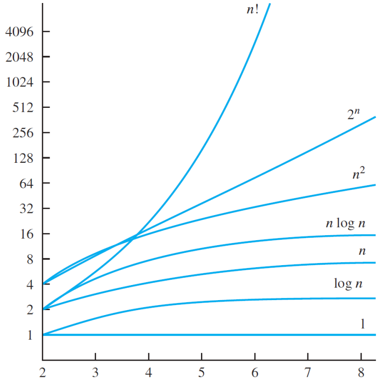
Thus, $f(x)$ is $O(x^2)$



The part of the graph of $f(x) = x^2 + 2x + 1$ that satisfies $f(x) < 4x^2$ is shown in color.

FIGURE 1 **The function $x^2 + 2x + 1$ is $O(x^2)$.**

The growth of functions commonly used in big-O estimates.



$$1 < logn < n < nlogn < n^2 < 2^n < n!$$

Big-O theorem

1. Let $f(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$ where $a_1, a_2, ...a_{n-1}, a_n$ are real numbers. Then, $f(x)$ is $O(x^n)$.
2. $f_1(x) = O(g_1(x)) \wedge f_2(x) = O(g_2(x))$
$$\implies (f_1 + f_2)(x) = O(max(|g_1(x)|, |g_2(x)|))$$
3. $f_1(x) = O(g_1(x)) \wedge f_2(x) = O(g_2(x))$
$$\implies (f_1 f_2)(x) = O(g_1 g_2(x))$$

COROLLARY. Suppose that $f_1(x)$ and $f_2(x)$ are both $O(g(x))$. Then $(f_1 + f_2)(x)$ is $O(g(x))$.

Example.

2 The Growth of Functions

1. Estimate big-oh of functions $100n^2 + 23nlogn + 2019$
2. Give a big-O estimate for $(2n^2 + 17n)(7logn + 15)$
3. Give a big-O estimate for $f(n) = 3nlog(n!) + (n^2 + 3)logn$, where n is a positive integer.
4. Give a big-O estimate for $f(x) = (x + 1)log(x^2 + 1) + 3x^2$.

Solution.

2 The Growth of Functions

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

Solution: First, the product $3n\log(n!)$ will be estimated. From Example 6 we know that $\log(n!)$ is $O(n\log n)$. Using this estimate and the fact that $3n$ is $O(n)$, Theorem 3 gives the estimate that $3n\log(n!)$ is $O(n^2\log n)$.

Next, the product $(n^2+3)\log n$ will be estimated. Because $(n^2+3) < 2n^2$ when $n > 2$, it follows that n^2+3 is $O(n^2)$. Thus, from Theorem 3 it follows that $(n^2+3)\log n$ is $O(n^2\log n)$. Using Theorem 2 to combine the two big-O estimates for the products shows that $f(n) = 3n\log(n!) + (n^2+3)\log n$ is $O(n^2\log n)$. ◀

Give a big-O estimate for $f(x) = (x+1)\log(x^2+1) + 3x^2$.

Solution: First, a big-O estimate for $(x+1)\log(x^2+1)$ will be found. Note that $(x+1)$ is $O(x)$. Furthermore, $x^2+1 \le 2x^2$ when $x > 1$. Hence,

$$\log(x^2+1) \le \log(2x^2) = \log 2 + \log x^2 = \log 2 + 2\log x \le 3\log x,$$

if $x > 2$. This shows that $\log(x^2+1)$ is $O(\log x)$.

From Theorem 3 it follows that $(x+1)\log(x^2+1)$ is $O(x\log x)$. Because $3x^2$ is $O(x^2)$, Theorem 2 tells us that $f(x)$ is $O(\max(x\log x, x^2))$. Because $x\log x \le x^2$, for $x > 1$, it follows that $f(x)$ is $O(x^2)$. ◀

Quizz

2 The Growth of Functions

The function **$f(x) = 4^x + x^5 + 2\log x$** is ...

Select one:

- a. $O(x^5)$
- b. $O(2^x)$
- c. $O(5^x)$
- d. $O(x^4)$

Ans: C

Quizz

2 The Growth of Functions

Which of the following functions is big-oh of n?

Select one or more:

- a. $g(n) = \log n + 2018$
- b. $g(n) = 2018n + 1$
- c. $g(n) = n.\log n + 7$
- d. $g(n) = n^2 - 2n$

Ans: a,b (hàm nào có O(n))($f(x) = O(g(x))$ mean f(x) is big-oh of g(x))

Quizz

Let f(x)=xlogx+ 2018. What is true?

Select one:

- a. f(x) is $O(x^2)$
- b. f(x) is O(1)
- c. f(x) is O(x)
- d. f(x) is O(logx)

Ans: a

Big-Omega and Big-Theta Notation

- $\exists C > 0, k \geq 1 : |f(x)| \geq C|g(x)|$ whenever $x > k \rightarrow f(x) = \Omega(g(x))$
- $f(x) = O(g(x)) \wedge f(x) = \Omega(g(x)) \rightarrow f(x) = \Theta(g(x))$

Example. Show that $f(x) = 1 + 2 + ... + n$ is $\Theta(n^2)$

$$f(x) = 1 + 2 + ... + n = \frac{n(n+1)}{2} \implies \frac{n^2}{2} \leq f(x) \leq n^2$$

$$\implies f(x) = \Omega(n^2) \wedge f(x) = \Theta(n^2) \implies f(x) = \Theta(n^2)$$

Notes.

1. $f(x)$ is $\Omega(g(x))$ **if and only if** $g(x)$ is $O(f(x))$.
2. If $f(x) = \Theta(g(x))$ then f(x) is of order g(x) or f(x) and g(x) are of the same order (cùng độ tăng).
3. Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ where $a_1, a_2, ...a_{n-1}, a_n$ are real numbers. Then, $f(x)$ is $\Theta(x^n)$.

Table of Contents

- Computational complexity = **Time complexity** + space complexity (not be considered).
- Time complexity can be expressed in terms of the number of operations used by the algorithm.
  1. **Worst-case** complexity (tells us how many operations an algorithm requires to guarantee that it will produce a solution.).
  2. **Average-case** complexity (the average number of operations used to solve the problem over all possible inputs of a given size).

$\{8, 4, 11, 3, 10\}$

Max=8

$i = 2(8 \geq 4) \rightarrow$ Max=8

$i = 3(8 < 11) \rightarrow$ Max=11

$i = 4(11 \geq 3) \rightarrow$ Max=11

$i = 5(11 \geq 10) \rightarrow$ Max=11

Thus, Max=11

Example.

Describe the time complexity (Worst-case) of the algorithm for finding the largest element in a set.

procedure $max(a_1, a_2, \ldots, a_n$: integers)
$max := a_1$
for $i := 2$ **to** n
  if $max < a_i$ **then** $max := a_i$
return $max\{max$ is the largest element$\}$

$\to$ Number of comparisons $f(n) = 2(n-1)+1$. Thus, $f(x) = \Theta(n)$.

Example.

```
procedure printsth(n: positive integer)
    for i:=1 to n do
        print "hi"
    for k:=1 to n do
        print "I love you"
```

Estimate big-O of the given algorithm

Ans:f(n)=n+n=2n is O(n)

Example.


```
procedure printsth(n: positive integer)
    for i:=1 to n do
        for k:=1 to n do
            print "I love you"
```

Estimate big-O of the given algorithm

Ans: $f(n) = n.n = n^2$ is $O(n^2)$

Example.


```
procedure printsth(n: positive integer)
    for i:=1 to n do
        print "hi"
    for j:=1 to n do
        for k:=1 to i do
            print "I love you"
```

Estimate big-O of the given algorithm

Ans: $f(n) = n + n^2$ is $O(n^2)$

Quizz

3 Complexity of Algorithms

Consider the algorithm:

procedure thuattoan(a1, a_2, a_3, ..., a_n: integer)

k:=0

for i:=1 to n do

if (a_i > 0) then k:=k+1

print (k)

Determine the complexity of the algorithm.

Select one:

- a. $O(2^n)$
- b. O(n)
- c. O(1)
- d. O(logn)

Ans: b (Number of comparisons $f(n) = 2n + 1$)

Quizz

Consider the algorithm:

procedureGT(n : positive integer)

F:=1

for i:= 1 to n do

    *F: = F * i*

Print(F)

Give the **best big-O complexity** for the algorithm above.

a. O(n)

b. O(logn)

c. O(1)

d. O(n^2)

e. None of these

Ans: a $(f(n) = n)$

Table of Contents

4 Problems

1. List all the steps used by Algorithm "max" to find the maximum of the list 1, 8, 12, 9, 11, 2, 14, 5, 10, 4.

2. Devise an algorithm that finds the sum of all the integers in a list.

3. List all the steps used to search for 9 in the sequence 1, 3, 4, 5, 6, 8, 9, 11 using

a) a linear search b) a binary search.

4. Describe an algorithm that inserts an integer x in the appropriate position into the list $a_1, a_2, \ldots, a_n$ of integers that are in increasing order.

5. Use the bubble sort to sort 3, 1, 5, 7, 4, showing the lists obtained at each step.

6. Consider the Linear search algorithm:

procedure linear search(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)

$i := 1$

while ($i \leq n$ and $x \neq a_i$)

$i := i + 1$

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

else location := 0

return location

Given the sequence a_n: 3, 1, 5, 7, 4, 6. How many comparisons required for searching x = 7?

The Growth of Functions
4 Problems

1. Determine whether each of these functions is O(x).

a) $f(x) = 10$ b) $f(x) = 3x + 7$ c) $f(x) = x2 + x + 1$ d) $f(x) = 5 \log x$

2. Determine whether each of these functions is O(x2).

a) $f(x) = 17x + 11$ b) $f(x) = x^2 + 1000$ c) $f(x) = x \log x$

d) $f(x) = \frac{x^4}{2}$ e) $f(x) = 2^x$ f)) $f(x) = (x^3 + 2x)/(2x + 1)$

3. Find the least integer n such that f (x) is $O(x^n)$ for each of these functions.

a) $f(x) = 2x^3 + x^2 \log x$ b) $f(x) = 3x^3 + (\log x)^4$

c) $f(x) = (x^4 + x^2 + 1)/(x^3 + 1)$ d) $f(x) = (x^4 + 5 \log x)/(x^4 + 1)$

4. Determine whether x^3 is $O(g(x))$ for each of these functions $g(x)$.

a) $g(x) = x^2$ b) $g(x) = x^3$ c) $g(x) = x^2 + x^3$

d) $g(x) = x^2 + x^4$ e) $g(x) = 3^x$ f) $g(x) = x^3/2$

5. Arrange the functions $\sqrt{n}$, $1000 \log n$, $n \log n$, $2n!$, 2^n, 3^n, and $n^2/1{,}000{,}000$ in a list so that each function is big-O of the next function.

6. Give as good a big-O estimate as possible for each of these functions.

a) $(n^2 + 8)(n + 1)$ b) $(n \log n + n^2)(n^3 + 2)$ c) $(n! + 2^n)(n^3 + \log(n^2 + 1))$

1. Consider the algorithm:

procedure *giaithuat*($a_1, a_2, \ldots, a_n$: *integers*)
$count := 0$
for $i := i$ to n do
  if $a_i > 0$ then $count := count + 1$
print($count$)

Give the best big-O complexity for the algorithm above.

2. Consider the algorithm:

```
procedure GT(n : positive integer)
F:=1
for i:= 1 to n do
        F: = F * i
Print(F)
```

Give the best big-O complexity for the algorithm above.

3. Consider the algorithm:

```
procedure max(a ,a ,...,a : reals )
max:=a
for i:=2 to n
if max<a then max:=a
```

Give the best big-O complexity for the algorithm above.

Q&A

Thank you for listening!