Chapter 3

Algorithms

Eng. Huda M. Dawoud

November, 2015
Section 1: Algorithms

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

Pseudocode Language

procedure name(argument1: type, argument2: type)

variable := expression

begin statements end

if condition then statement [else statement]

for variable := initial value to final value

    statement

while condition statement

return expression

{comment}

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

Answer:

Input: List of integers with length n.

Output: The sum of all integers in the list.

procedure sum(L: List of integers)

sum := L[1] { indicating the first value }

for i := 2 to n

    sum := sum + L[i]

return sum
4. Describe an algorithm that takes as input a list of n integers and produces as output the largest difference obtained by subtracting an integer in the list from the one following it.

Answer:

Input: List of integers with length n.

Output: The largest difference obtained by subtracting an integer in the list from the one following it.

procedure largest difference(L: List of integers)

diff := L[n] – L[n-1]

for i := 1 to n-2
    if (L[n-i] – L[n-i-1] > diff) then
        diff := L[n-i] – L[n-i-1]

return diff

5. Describe an algorithm that takes as input a list of n integers in nondecreasing order and produces the list of all values that occur more than once. (Recall that a list of integers is nondecreasing if each integer in the list is at least as large as the previous integer in the list.)

Answer:

Input: List of integers with length n in nondecreasing order.

Output: List of all values that occur more than once.

procedure duplicates(L: List of integers in nondecreasing order)

j := 2
k := 0
while $j \leq n$

if $L[j] = L[j-1]$ then

$k := k + 1$

$C[k] := L[j]$

while $j \leq n$ and $C[k] = L[j]$

$j := j + 1$

$j := j + 1$

return $C$ {List of all values that occur more than once}

6. Describe an algorithm that takes as input a list of $n$ integers and finds the number of negative integers in the list.

Answer:

Input: List of integers with length $n$.

Output: The number of negative integers in the list.

procedure negatives($a_1, a_2, \ldots, a_n :$ integers)

$k := 0$

for $i := 1$ to $n$

if $a_i < 0$ then $k := k + 1$

return $k$ {the number of negative integers in the list}
8. Describe an algorithm that takes as input a list of n distinct integers and finds the location of the largest even integer in the list or returns 0 if there are no even integers in the list.

Answer:

Input: List of n distinct integers.

Output: The location of the largest even integer in the list or returns 0 if there are no even integers in the list.

procedure largest even location(a_1, a_2, ..., a_n : integers)

k := 0

largest := −∞

for i := 1 to n

    if (a_i is even and a_i > largest) then

        k := i

        largest := a_i

return k {the desired location (or 0 if there are no evens)}

9. A palindrome is a string that reads the same forward and backward. Describe an algorithm for determining whether a string of n characters is a palindrome.

Answer:

Input: string of n characters.

Output: boolean indicates whether the string is palindrome or not.

procedure palindrome check(a1,a2, ..., an : string)

answer := true
for i:=1 to \([n/2]\)

    if \(a_i \neq a_{n+1-i}\) then answer := false

return answer \{ answer is true if and only if string is a palindrome\}

15. 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.

Answer:

Input: integer \(x\), a list of \(n\) integers in increasing order.

Output: a list of \(n+1\) integers with \(x\) in the appropriate position.

procedure insert(\(x: \text{integer}\), \(L: \text{List of integers}\))

\(i := 1\)

while \(x > L[i]\)

    \(i := i + 1\) \{the loop ends when \(i\) is the index for \(x\)\}

\(C[i] = L[i]\)

\(C[i] := x\)

for \(j := i+1\) to \(n+1\)

    \(C[j] := L[j-1]\)

\(L := C\)

return \(L\)
18. Describe an algorithm that locates the last occurrence of the smallest element in a finite list of integers, where the integers in the list are not necessarily distinct.

Answer:

procedure last smallest\(a_1, a_2, \ldots, a_n : \text{integers}\)

\[
\begin{align*}
\text{min} & := a_1 \\
\text{location} & := 1 \\
\text{for } i & := 2 \text{ to } n \\
& \text{if } \text{min } \leq a_i \text{ then} \\
& \hspace{1em} \text{min} := a_i \\
& \hspace{1em} \text{location} := i \\
\text{return location \{the location of the last occurrence of the smallest element in the list\}}
\end{align*}
\]
Section 2: The growth of functions

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 $g(x)$.”]

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

Suppose that $f_1(x)$ is $O(g_1(x))$ and that $f_2(x)$ is $O(g_2(x))$. Then $(f_1 + f_2)(x)$ is $O(\max(|g_1(x)|, |g_2(x)|))$.

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 $\Omega(g(x))$ if there are positive constants $C$ and $k$ such that

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

whenever $x > k$. [This is read as “$f(x)$ is big-Omega of $g(x)$.”]

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 $\Theta(g(x))$ if $f(x)$ is $O(g(x))$ and $f(x)$ is $\Omega(g(x))$. When $f(x)$ is $\Theta(g(x))$ we say that $f$ is big-Theta of $g(x)$, that $f(x)$ is of order $g(x)$, and that $f(x)$ and $g(x)$ are of the same order.

2. Determine whether each of these functions is $O(x^2)$.

   a) $f(x) = 17x + 11$
   b) $f(x) = x^2 + 1000$
   c) $f(x) = x \log x$
   d) $f(x) = x^4/2$
   e) $f(x) = 2^x$
   f) $f(x) = \lfloor x \rfloor \cdot \lceil x \rceil$

Answer:
Note that the choices of C and k witnesses are not unique.

a) Yes, since $17x + 11 \leq 17x + x = 18x \leq 18x^2$ for all $x > 11$. The witnesses are $C = 18$ and $k = 11$.

b) Yes, since $x^2 + 1000 \leq x^2 + 1000x^2 = 10001x^2$ for all $x > 1$. The witnesses are $C = 2$ and $k = 1$.

c) Yes, since $x \log x \leq x \cdot x = x^2$ for all $x$ in the domain of the function. (The fact that $\log x < x$ for all $x$ follows from the fact that $x < 2^x$ for all $x$, which can be seen by looking at the graphs of these two functions.) The witnesses are $C = 1$ and $k = 0$.

d) No. If there were a constant $C$ such that $x^4/2 \leq Cx^2$ for sufficiently large $x$, then we would have $C \geq x^2/2$. This is clearly impossible for a constant to satisfy.

e) No. If $2^x$ were $O(x^2)$, then the fraction $2^x/x^2$ would have to be bounded above by some constant $C$. It can be shown that in fact $2^x > x^3$ for all $x \leq 10$ (using mathematical induction), so $2^x/x^2 \geq x^3/x^2 = x$ for large $x$, which is certainly not less than or equal to $C$.

f) Yes, since $[x],[x] \leq x(x + 1) \leq x \cdot 2x = 2x^2$ for all $x > 1$. The witnesses are $C = 2$ and $k = 1$.

3. Use the definition of “$f(x)$ is $O(g(x))$” to show that $x^4 + 9x^3 + 4x + 7$ is $O(x^4)$.

Answer:

We need to put some bounds on the lower order terms. If $x > 9$ then we have $x^4 + 9x^3 + 4x + 7 \leq x^4 + x^4 + x^4 + x^4 = 4x^4$

Therefore $x^4 + 9x^3 + 4x + 7$ is $O(x^4)$, taking witnesses $C = 4$ and $k = 9$. 
4. Use the definition of “f (x) is O(g(x))” to show that 

\[ 2^x + 17 \] is \( O(3^x) \).

Answer:

If \( x > 5 \), then \( 2^x + 17 \leq 2^x + 2^x = 2 \cdot 2^x \leq 2 \cdot 3^x \). This shows that \( 2^x + 17 \) is \( O(3^x) \) (the witnesses are \( C = 2 \) and \( k = 5 \)).

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

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

b) \( f (x) = 3x^5 + (\log x)^4 \)

c) \( f (x) = (x^4 + x^2 + 1)/(x^4 + 1) \)

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

Answer:

a) Since \( x^3 \log x \) is not \( O(x^3) \) (because the \( \log x \) factor grows without bound as \( x \) increases), \( n = 3 \) is too small. On the other hand, certainly \( \log x \) grows more slowly than \( x \), so \( 2x^2 + x^3 \log x \leq 2x^4 + x^4 = 3x^4 \). Therefore \( n = 4 \) is the answer, with \( C = 3 \) and \( k = 0 \).

b) The \( (\log x)^4 \) is insignificant compared to the \( x^5 \) term, so the answer is \( n = 5 \). Formally we can take \( C = 4 \) and \( k = 1 \) as witnesses.

c) For large \( x \), this fraction is fairly close to 1. (This can be seen by dividing numerator and denominator by \( x^4 \).) Therefore we can take \( n = 0 \); in other words, this function is \( O(x^0) = O(1) \). Note that \( n = -1 \) will not do, since a number close to 1 is not less than a constant times \( n-1 \) for large \( n \). Formally we can write \( f(x) \leq 3x^4/x^4 = 3 \) for all \( x > 1 \), so witnesses are \( C = 3 \) and \( k = 1 \).
d) This is similar to the previous part, but this time \( n = -1 \) will do, since for large \( x \), \( f(x) \approx 1/x \). Formally we can write \( f(x) \leq 6x^3/x^3 = 6 \) for all \( x > 1 \), so witnesses are \( C = 6 \) and \( k = 1 \).
Section 3: Complexity of Algorithms

**Worst-Case Complexity** The type of complexity analysis done in Example 2 is a worst-case analysis. By the worst-case performance of an algorithm, we mean the largest number of operations needed to solve the given problem using this algorithm on input of specified size. Worst-case analysis tells us how many operations an algorithm requires to guarantee that it will produce a solution.