ALGORITHMS

Algorithm: definition

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

ALGORITHM 1 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}

Algorithm: properties

- *Input*. An algorithm has input values from a specified set.
- *Output*. From each set of input values an algorithm produces output values from a specified set. The output values are the solution to the problem.
- *Definiteness*. The steps of an algorithm must be defined precisely.
- *Correctness*. An algorithm should produce the correct output values for each set of input values.
- *Finiteness*. An algorithm should produce the desired output after a finite (but perhaps large) number of steps for any input in the set.
- *Effectiveness*. It must be possible to perform each step of an algorithm exactly and in a finite amount of time.
- *Generality*. The procedure should be applicable for all problems of the desired form, not just for a particular set of input values.

Algorithm: examples

- Linear search algorithm

ALGORITHM 2 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*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

Algorithm: examples

- Binary search algorithm

ALGORITHM 3 The Binary 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}

Algorithm: examples

- The bubble sort algorithm

ALGORITHM 4 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$\}$

Algorithm: examples

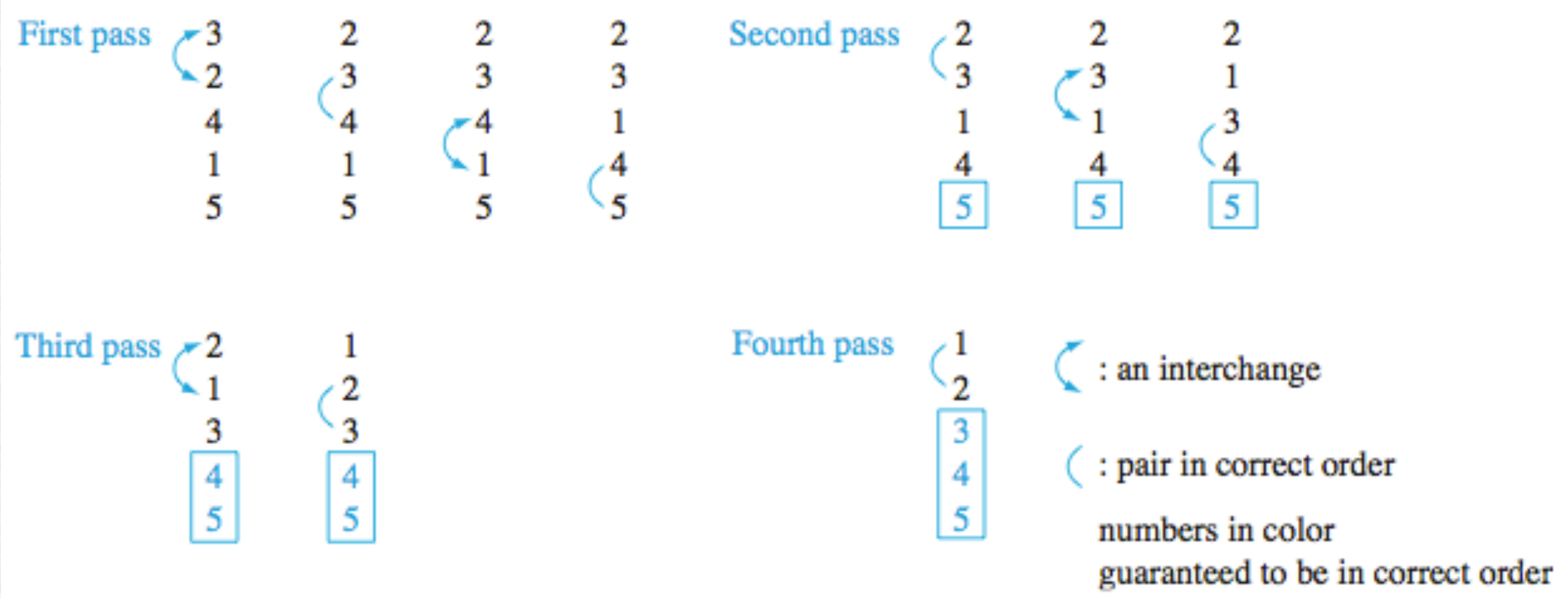


FIGURE 1 **The Steps of a Bubble Sort.**

Algorithm: examples

- The insertion sort algorithm

ALGORITHM 5 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$\}$

Algorithm: examples

- Greedy algorithms
  - ✓ Optimization problems
  - ✓ Selects the **best choice at each step**, instead of considering all sequences of steps that may lead to an optimal solution

Algorithm: examples

- The problem of making n cents change
  - ✓ quarters, dimes, nickels, and pennies
  - ✓ using the least total number of coins

ALGORITHM 6 Greedy Change-Making Algorithm.

procedure *change*($c_1, c_2, \ldots, c_r$: values of denominations of coins, where
    $c_1 > c_2 > \cdots > c_r$; n: a positive integer)
for $i := 1$ **to** r
    $d_i := 0$ {d_i counts the coins of denomination c_i used}
    while $n \geq c_i$
        $d_i := d_i + 1$ {add a coin of denomination c_i}
        $n := n - c_i$
{d_i is the number of coins of denomination c_i in the change for $i = 1, 2, \ldots, r$}

Algorithm: examples

- On-class discussion
  - ✓ LEMMA 1 - p.199

THE GROWTH OF FUNCTIONS

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 $g(x)$."]

- **Remark**: Intuitively, f (x) grows slower that some fixed multiple of g(x) as x grows without bound
  - ✓ C and k are called witnesses

Big-O Notation

Remark: The fact that $f(x)$ is $O(g(x))$ is sometimes written $f(x) = O(g(x))$. However, the equals sign in this notation does *not* represent a genuine equality. Rather, this notation tells us that an inequality holds relating the values of the functions f and g for sufficiently large numbers in the domains of these functions. However, it is acceptable to write $f(x) \in O(g(x))$ because $O(g(x))$ represents the set of functions that are $O(g(x))$.

When $f(x)$ is $O(g(x))$, and $h(x)$ is a function that has larger absolute values than $g(x)$ does for sufficiently large values of x, it follows that $f(x)$ is $O(h(x))$. In other words, the function $g(x)$ in the relationship $f(x)$ is $O(g(x))$ can be replaced by a function with larger absolute values. To see this, note that if

Big-O Notation:example

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

Solution: We observe that we can readily estimate the size of $f(x)$ when $x > 1$ because $x < x^2$ and $1 < x^2$ when $x > 1$. It follows that

$$0 \le x^2 + 2x + 1 \le x^2 + 2x^2 + x^2 = 4x^2$$

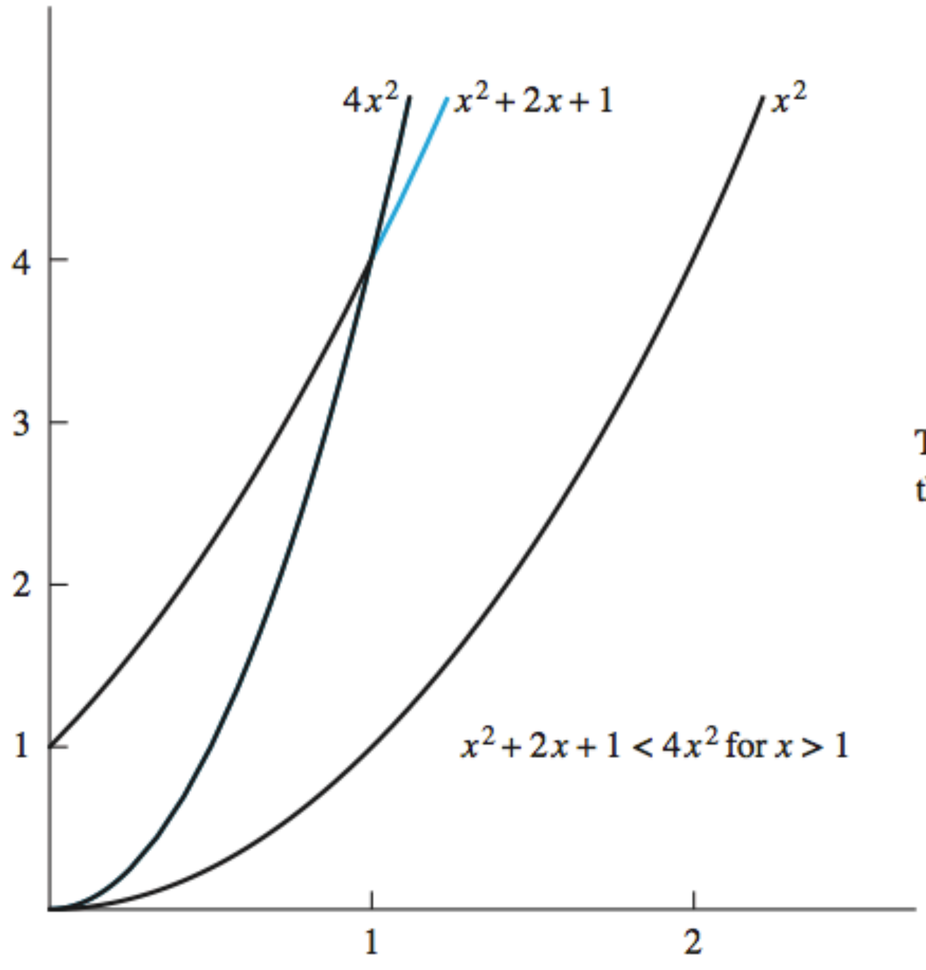
whenever $x > 1$, as shown in Figure 1. Consequently, we can take $C = 4$ and $k = 1$ as witnesses to show that $f(x)$ is $O(x^2)$. That is, $f(x) = x^2 + 2x + 1 < 4x^2$ whenever $x > 1$. (Note that it is not necessary to use absolute values here because all functions in these equalities are positive when x is positive.)

Alternatively, we can estimate the size of $f(x)$ when $x > 2$. When $x > 2$, we have $2x \le x^2$ and $1 \le x^2$. Consequently, if $x > 2$, we have

$$0 \le x^2 + 2x + 1 \le x^2 + x^2 + x^2 = 3x^2.$$

It follows that $C = 3$ and $k = 2$ are also witnesses to the relation $f(x)$ is $O(x^2)$.

Big-O Notation:example



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

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

Big-O Notation

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 $O(x^n)$.

Proof: Using the triangle inequality (see Exercise 7 in Section 1.8), if $x > 1$ we have

$$\begin{aligned} |f(x)| &= |a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0| \\ &\leq |a_n|x^n + |a_{n-1}|x^{n-1} + \cdots + |a_1|x + |a_0| \\ &= x^n \left(|a_n| + |a_{n-1}|/x + \cdots + |a_1|/x^{n-1} + |a_0|/x^n\right) \\ &\leq x^n \left(|a_n| + |a_{n-1}| + \cdots + |a_1| + |a_0|\right). \end{aligned}$$

This shows that

$$|f(x)| \leq Cx^n,$$

where $C = |a_n| + |a_{n-1}| + \cdots + |a_0|$ whenever $x > 1$. Hence, the witnesses $C = |a_n| + |a_{n-1}| + \cdots + |a_0|$ and $k = 1$ show that $f(x)$ is $O(x^n)$.

Big-O Notation

- Theorem

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 $O(\max(|g_1(x)|, |g_2(x)|))$.

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

- Theorem

Suppose that $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$. Then $(f_1 f_2)(x)$ is $O(g_1(x)g_2(x))$.