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, …

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}

The term *algorithm* is a corruption of the name *al-Khowarizmi*.

Describe an algorithm for finding the maximum (largest) 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.

Pseudocode is **an artificial and informal language that helps programmers develop algorithms**. Pseudocode is a "text-based" detail (algorithmic) design tool.

PROPERTIES OF ALGORITHMS There are several properties that algorithms generally share. They are useful to keep in mind when algorithms are described. These properties are:

- *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.

The first algorithm that we will present is called the **linear search**, or **sequential search**.

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}

Worst-case complexity:
2n + 2

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}

Worst-case complexity:
3 + 3log(n) + 2

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 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}

Algorithms that make what seems to be the "best" choice at each step are called **greedy algorithms**.

Algorithms

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

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

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

Number of steps:

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

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:

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

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

algorithm: a finite sequence of precise instructions for performing a computation or solving a problem

searching algorithm: the problem of locating an element in a list

linear search algorithm: a procedure for searching a list element by element

binary search algorithm: a procedure for searching an ordered list by successively splitting the list in half

sorting: the reordering of the elements of a list into prescribed order

$f(x)$ **is** $O(g(x))$**:** the fact that $|f(x)| \leq C|g(x)|$ for all $x > k$ for some constants C and k

witness to the relationship $f(x)$ **is** $O(g(x))$**:** a pair C and k such that $|f(x)| \leq C|g(x)|$ whenever $x > k$

$f(x)$ **is** $\Omega(g(x))$**:** the fact that $|f(x)| \geq C|g(x)|$ for all $x > k$ for some positive constants C and k

$f(x)$ **is** $\Theta(g(x))$**:** the fact that $f(x)$ is both $O(g(x))$ and $\Omega(g(x))$

time complexity: the amount of time required for an algorithm to solve a problem

space complexity: the amount of space in computer memory required for an algorithm to solve a problem

worst-case time complexity: the greatest amount of time required for an algorithm to solve a problem of a given size

average-case time complexity: the average amount of time required for an algorithm to solve a problem of a given size

algorithmic paradigm: a general approach for constructing algorithms based on a particular concept

brute force: the algorithmic paradigm based on constructing algorithms for solving problems in a naive manner from the statement of the problem and definitions

greedy algorithm: an algorithm that makes the best choice at each step according to some specified condition

tractable problem: a problem for which there is a worst-case polynomial-time algorithm that solves it

intractable problem: a problem for which no worst-case polynomial-time algorithm exists for solving it

solvable problem: a problem that can be solved by an algorithm

unsolvable problem: a problem that cannot be solved by an algorithm

Algorithm	Data structure	Time complexity:Best	Time complexity:Average	Time complexity:Worst	Space complexity:Worst
Bubble sort	Array	O(n)	O(n^2)	O(n^2)	O(1)
Insertion sort	Array	O(n)	O(n^2)	O(n^2)	O(1)
Selection sort	Array	O(n^2)	O(n^2)	O(n^2)	O(1)

ALGORITHM	BEST CASE	AVERAGE CASE	WORST CASE
LINEAR SEARCH	O(1)	O(N)	O(N)
BINARY SEARCH	O(1)	O(LOG2N)	O(LOG2N)

What are the different ways to describe algorithms?

Ans. in English, in a computer language, in pseudocode

What is the difference between an algorithm for solving a problem and a computer program that solves this problem?

An algorithm is a step-by-step procedure for solving the problem while programming is a set of instructions for a computer to follow to perform a task.