Foundations of Computer Science
Algorithmic Foundations

ONE LOVE. ONE FUTURE.

HUST

hust.edu.vn fb.com/dhbkhn

FaMI 1956

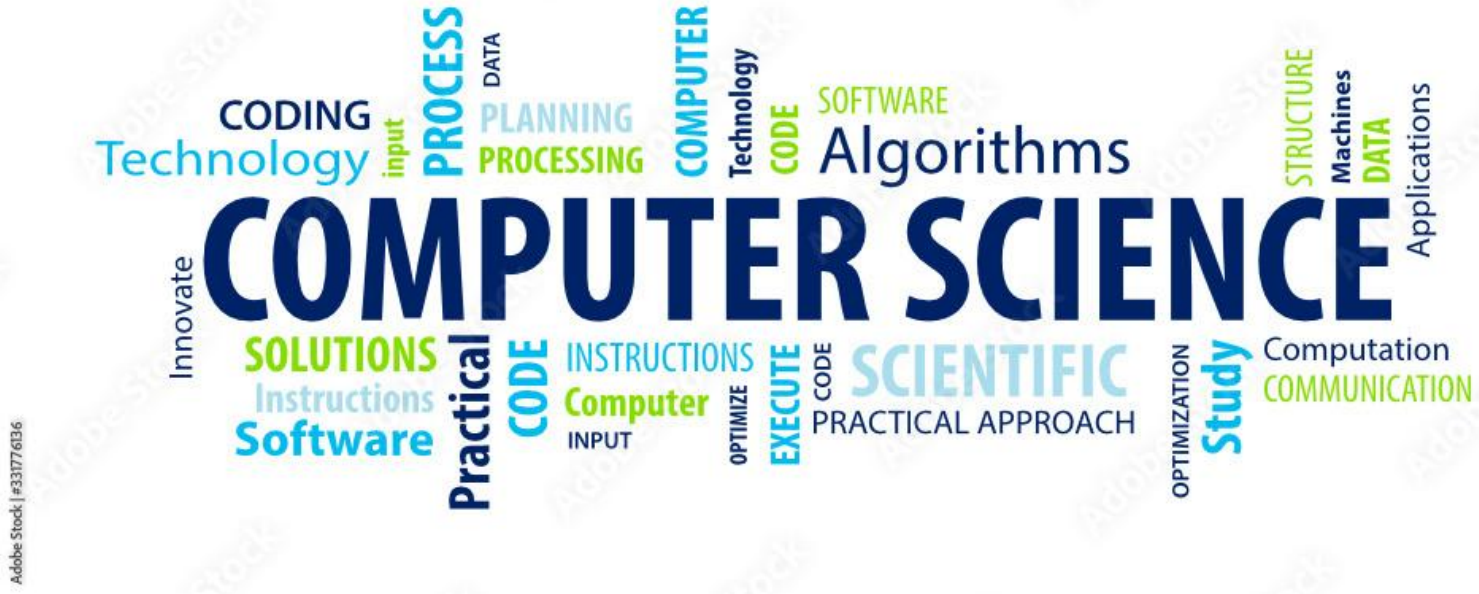
Contents

- Review
- Algorithm Presentation
- Examples of Algorithms
- Algorithm Attributes

Computer Science (in the sense of algorithm)

Computer science: the study of algorithms, including

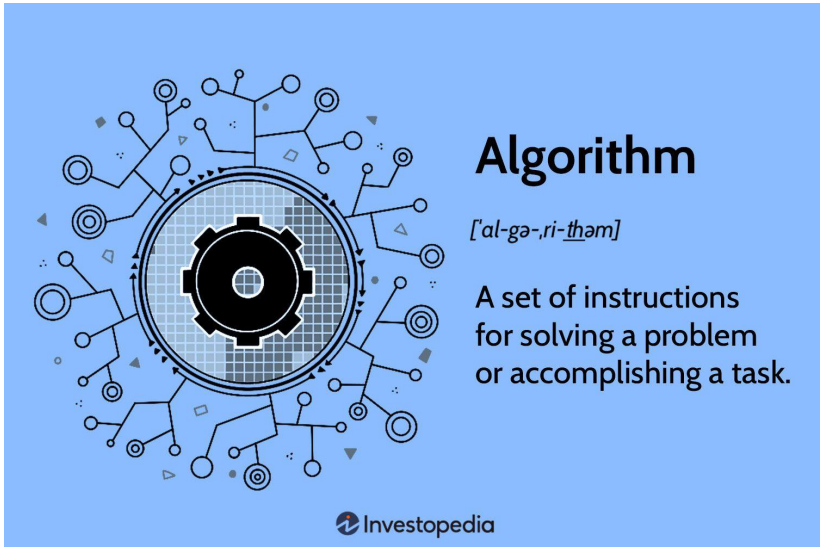
1. Their formal and mathematical properties
2. Their hardware realizations
3. Their linguistic realizations
4. Their applications



Algorithm (Meriam Webster)

al • go • rithm *n. A procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation;*

broadly: a step-by-step method for accomplishing some task.



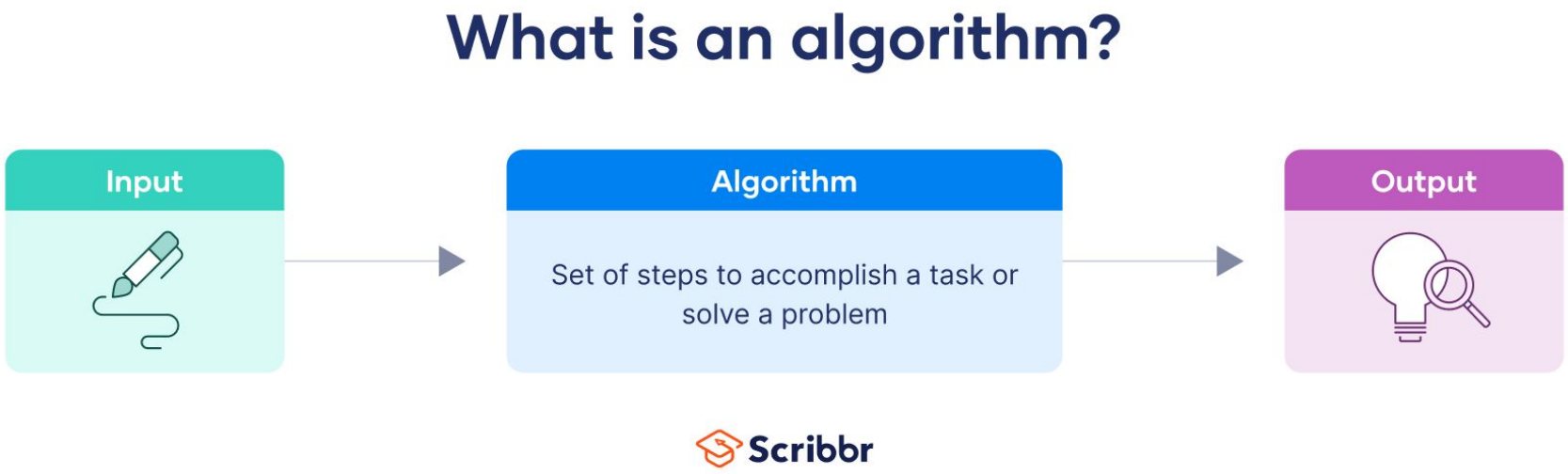
Why algorithm?

If we can specify an algorithm to solve a problem, then we can automate its solution by computing agent.

Computer science: Science of algorithmic problem solving

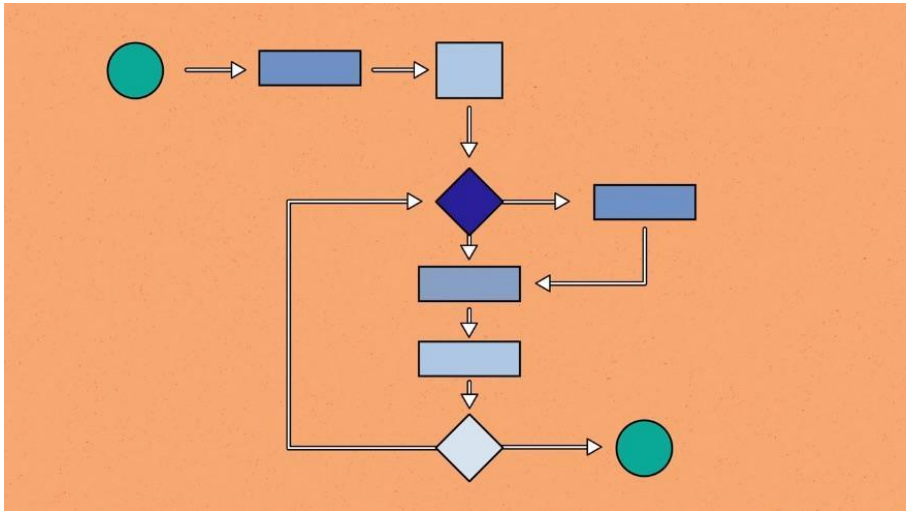
Algorithm

Algorithm: a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.



Operations

Sequential operations carries out single well defined task.
Conditional operations. The next operation is selected based on conditions.
Iterative operations. These are the “looping” instructions of an algorithm.



Algorithmic Foundations
Algorithms presentation

ONE LOVE. ONE FUTURE.

Natural Language or High Level Programming Language?

Initially, set the value of the variable carry to 0 and the value of the variable i to 0.

When these initializations have been completed, begin looping as long as the value of the variable i is less than or equal to (m – 1).

First, add together the values of the two digits a_i and b_i and the current value of the carry digit to get the result called c_i.

Now check the value of c_i to see whether it is greater than or equal to 10.

If c_i is greater than or equal to 10, then reset the value of carry to 1 and reduce the value of c_i by 10;

otherwise, set the value of carry to 0.

When you are finished with that operation, add 1 to i and begin the loop all over again.

When the loop has completed execution, set the left most digit of the result c_m to the value of carry and print out the final result, which consists of the digits $c_m\ c_{m-1} \ldots c_0$.

After printing the result, the algorithm is finished, and it terminates.

```
{
Scanner inp = new Scanner(System.in);
int i, m, carry;
int[] a = new int[100];
int[] b = new int[100];
int[] c = new int[100];
m = inp.nextInt();
for (int j = 0;j <= m-1;j++) {
        a[j] = inp.nextInt();
        b[j] = inp.nextInt();
}
carry = 0;
i = 0;
while (i < m) {
    c[i] = a[i] + b[i] + carry;
    if (c[i] >= 10)
              .
              .
              .
```

Pseudo Code

NOT BAD

```
FOR X = 1 to 10
  FOR Y = 1 to 10
    IF gameBoard[X][Y] = 0
      Do nothing
    ELSE
      CALL theCall(X, Y) (recursively)
      counter += 1
    END IF
  END FOR
END FOR
```

GOOD

```
SET moveCount to 1
FOR each row on the board
  FOR each column on the board
    IF gameBoard position (row, column) is occupied THEN
      CALL findAdjacentTiles with row, column
      INCREMENT moveCount
    END IF
  END FOR
END FOR
```

Pseudocode Example

Good

```
FOR each index of string
    IF string(index) is equal to hash THEN
        RETURN "Hash found"
    END IF
END FOR
```

Bad

```
FOR i = 0 to 20
    IF string[i] = '#'
        RETURN "Hash found"
    END IF
END FOR
```

```
FOR each index of string
    if string[i] = '#'
      return "Hash found"
    END if
END FOR
```

Sequential Operations

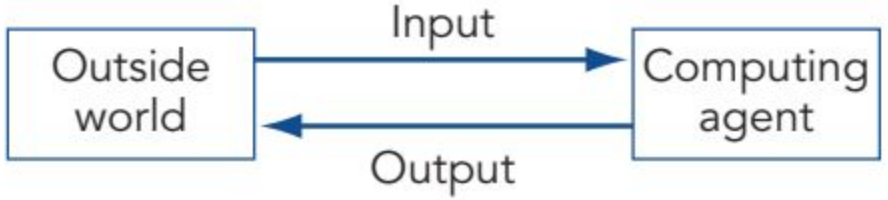
Vocabulary

Variable: storage of value

SET Operations

Variable := Arithmetic Expression

INPUT and OUTPUT Operations



INPUT Variable
OUTPUT Value

Step	Operation
1	Get values for *gallons used*, *starting mileage*, *ending mileage*
2	Set value of *distance driven* to (*ending mileage* – *starting mileage*)
3	Set value of *average miles per gallon* to (*distance driven* ÷ *gallons used*)
4	Print the value of *average miles per gallon*
5	Stop

Examples

Write pseudocode versions of the following:

1. An algorithm that gets three data values x, y, and z as input and outputs the average of those three values.
2. An algorithm that gets the radius r of a circle as input. Its output is both the circumference and the area of a circle of radius r.
3. An algorithm that gets the amount of electricity used in kilowatt-hours and the cost of electricity per kilowatt-hour. Its output is the total amount of the electric bill, including an 8% sales tax.
4. An algorithm that inputs your current credit card balance, the total dollar amount of new purchases, and the total dollar amount of all payments. The algorithm computes the new balance, which includes a 12% interest charge on any unpaid balance.
5. An algorithm that is given the length and width, in feet, of a rectangular carpet and determines its total cost given that the material cost is $23 per square yard.
6. An algorithm that is given three numbers corresponding to the number of times a race car driver has finished first, second, and third. The algorithm computes and displays how many points that driver has earned given 5 points for a first, 3 points for a second, and 1 point for a third place finish.

Conditional Statements and Diagram

IF Condition

Set 1 of Operations

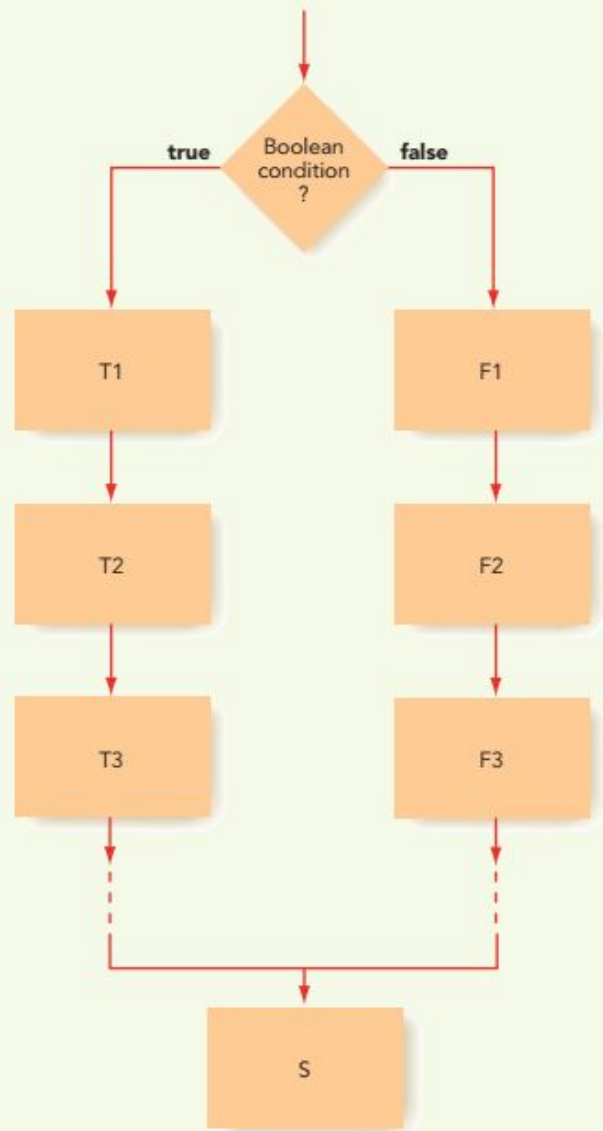
IF Condition

Set 1 of Operations

ELSE

Set 2 of Operations

Step	Operation
1	Get values for *gallons used, starting mileage, ending mileage*
2	Set value of *distance driven* to (*ending mileage – starting mileage*)
3	Set value of *average miles per gallon* to (*distance driven ÷ gallons used*)
4	Print the value of *average miles per gallon*
5	If *average miles per gallon* is >25.0 then
6	Print the message 'You are getting good gas mileage'
	Else
7	Print the message 'You are NOT getting good gas mileage'
8	Stop



More on Conditional Statements

```
IF Condition 1
    Set 1 Of Operations
ELSEIF Condition 2
    Set 2 Of Operations
ELSEIF Condition 3
    Set 3 Of Operations
…
ELSE
    Set n Of Operations
```

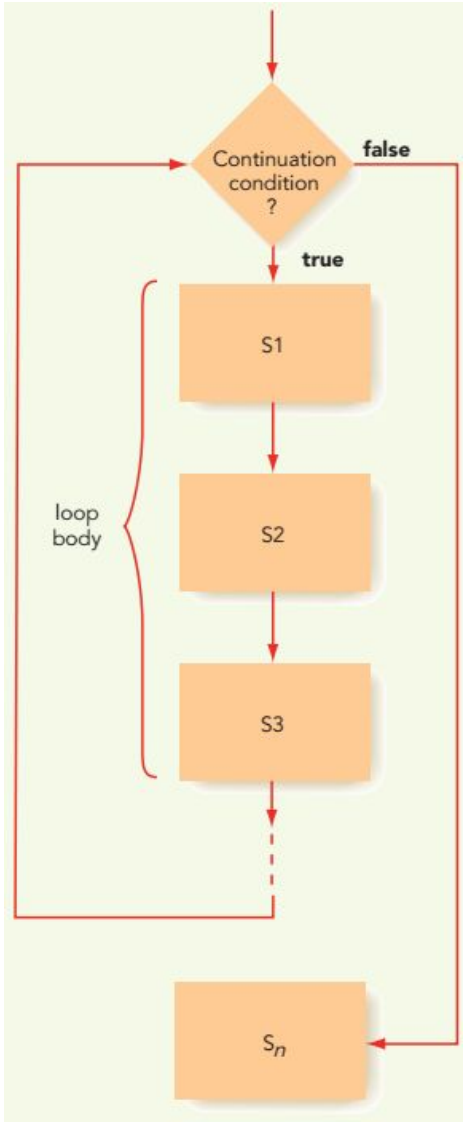
```
CASE Variable
Value 1
    Set 1 Of Operations
Value 2
    Set 2 Of Operations
…
ELSE
    Set n Of Operations
```

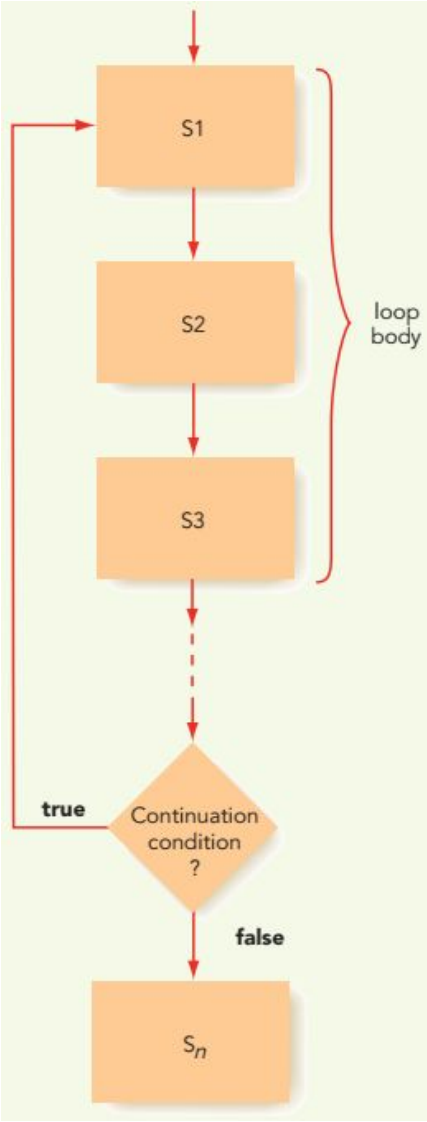
Iteration

FOR Counter = Beginning Value
TO Ending Value
    Set of Conditions

WHILE Condition
    Set Of Operations

DO
    Set Of Operations
WHILE Condition





Example

Step	Operation
1	*response* = Yes
2	While (*response* = Yes) do Steps 3 through 11
3	Get values for *gallons used*, *starting mileage*, *ending mileage*
4	Set value of *distance driven* to (*ending mileage* – *starting mileage*)
5	Set value of *average miles per gallon* to (*distance driven* ÷ *gallons used*)
6	Print the value of *average miles per gallon*
7	If average miles per gallon > 25.0 then
8	Print the message 'You are getting good gas mileage'
	Else
9	Print the message 'You are NOT getting good gas mileage'
10	Print the message 'Do you want to do this again? Enter Yes or No'
11	Get a new value for *response* from the user
12	Stop

Examples

1. Write an if/then/else statement that sets the variable y to the value 1 if x ≥ 0. If x < 0, then the statement should set y to the value 2. (Assume x already has a value.)
2. Write an algorithm that gets as input three data values x, y, and z and outputs the average of these values if the value of x is positive. If the value of x is either 0 or negative, your algorithm should not compute the average but should print the error message 'Bad Data' instead.
3. Write an algorithm that gets as input your current credit card balance, the total dollar amount of new purchases, and the total dollar amount of all payments. The algorithm computes the new balance, which this time includes an 8% interest charge on any unpaid balance below $100, 12% interest on any unpaid balance between $100 and $500, inclusive, and 16% on any unpaid balance above $500.
4. Write an algorithm that gets as input a single nonzero data value x and outputs the three values x 2, sin x, and 1/x. This process is repeated until the input value for x is equal to 999, at which time the algorithm terminates.

Examples

5. Write an algorithm that inputs the length and width, in feet, of a rectangular carpet and the price of the carpet in dollars per square yard. It then determines if you can afford to purchase this carpet, given that your total budget for carpeting is $500.
6. Add the following feature to the algorithm created in the previous Practice Problem: If the cost of the carpet is less than or equal to $250, output a message that this is a particularly good deal.
7. Add the following feature to the algorithm created in Practice Problem 4 above. Assume the input for the data value x can be any value, including 0. Since the value 1/x cannot be computed if x = 0, you will have to check for this condition and output an error message saying that you are unable to compute the value 1/x
8. Given two nonnegative integer values, a ≥ 0, b ≥ 0, compute and output the product (a × b) using the technique of repeated

Assume that we have a list of 10,000 telephone numbers (rather than hundreds of millions) that we represent symbolically as *T*1, *T*2, *T*3, ... , *T*10,000, along with the 10,000 names of the individuals associated with each specific phone number, denoted as *N*1, *N*2, N3, ... , *N*10,000.

Step	Operation
1	Get values for *NUMBER*, T_1, ... , $T_{10,000}$, and N_1, ... , $N_{10,000}$
2	If *NUMBER* = T_1 then print the value of N_1
3	If *NUMBER* = T_2 then print the value of N_2
4	If *NUMBER* = T_3 then print the value of N_3
.	.
.	.
.	.
10,000	If *NUMBER* = $T_{9,999}$ then print the value of $N_{9,999}$
10,001	If *NUMBER* = $T_{10,000}$ then print the value of $N_{10,000}$
10,002	Stop

Step	Operation
1	Get values for *NUMBER*, T_1, ... , $T_{10,000}$, and N_1, ... ,$N_{10,000}$
2	Set the value of *i* to 1 and set the value of *Found* to NO
3	While (*Found* = NO) do Steps 4 through 7
4	If *NUMBER* is equal to the *i*th number on the list, T_i, then
5	Print the name of the corresponding person, N_i
6	Set the value of *Found* to YES
	Else (*NUMBER* is not equal to T_i)
7	Add 1 to the value of *i*
8	Stop

Searching

Step	Operation
1	Get values for *NUMBER*, T_1, … , $T_{10,000}$, and N_1, … , $N_{10,000}$
2	Set the value of *i* to 1 and set the value of *Found* to NO
3	While both (*Found* = NO) and ($i \leq 10{,}000$) do Steps 4 through 7
4	If NUMBER is equal to the ith number on the list, T_i, then
5	Print the name of the corresponding person, N_i
6	Set the value of *Found* to YES
	Else (*NUMBER* is not equal to T_i)
7	Add 1 to the value of *i*
8	If (*Found* = NO) then
9	Print the message 'Sorry, this number is not in the directory'
10	Stop

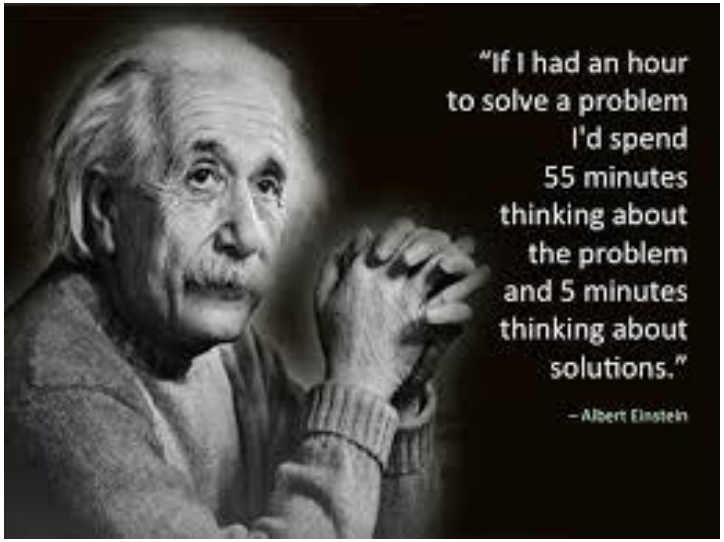
Find the Largest Value in a List

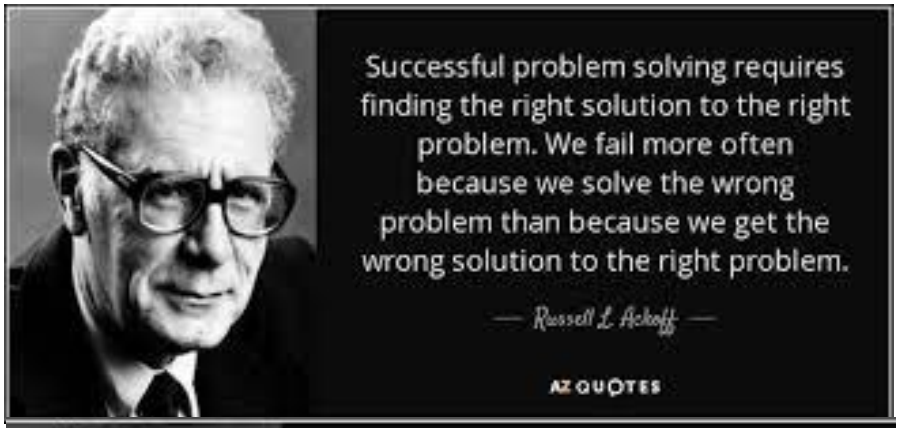
Get a value for n, the size of the list
Get values for $A_1, A_2, \ldots, A_n$, the list to be searched

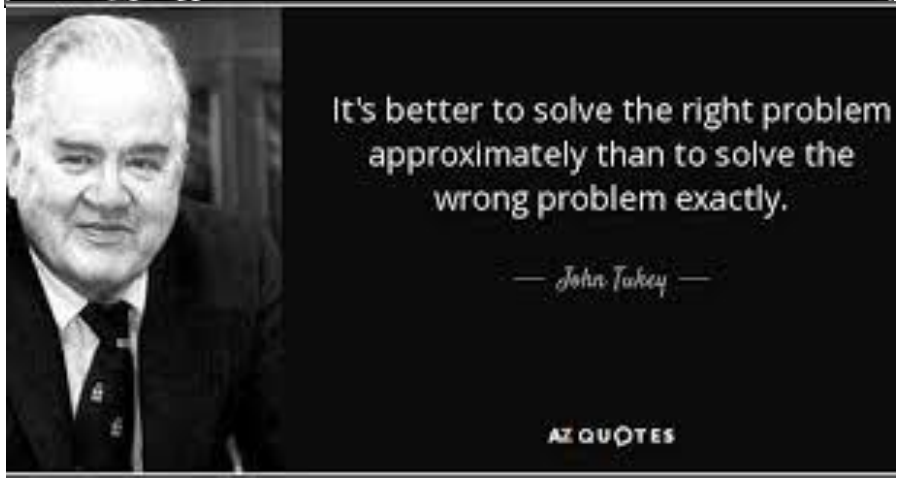
Algorithm Attributes

- Unambiguous
- Finite
- Correct
- Easy to understand
- Elegant
- Efficient

*"Are we solving the right problem?
Are we solving the problem right?"*







Space and Time Efficiency

- Space: Usage of Memory, Storages
- Time: Slow or Fast?

1. On which computer will we run the algorithm? Should we use a smartphone, a modest laptop, or a supercomputer capable of doing many trillions of calculations per second?
2. Which reverse telephone book (list of numbers) will we use: one limited to a relatively small geographic area or one that covers all listed numbers in the North American Numbering Plan?
3. Which number will we search for? What if we pick a number tha st in the list? What if it happens to

A standard against which something is compared

To make things comparative

(Time) Complexity Analysis

Step	Operation
1	Get values for *NUMBER*, T_1, … , $T_{10,000}$, and N_1, … , $N_{10,000}$
2	Set the value of *i* to 1 and set the value of *Found* to NO
3	While both (*Found* = NO) and ($i \leq 10{,}000$) do Steps 4 through 7
4	If NUMBER is equal to the ith number on the list, T_i, then
5	Print the name of the corresponding person, N_i
6	Set the value of *Found* to YES
	Else (*NUMBER* is not equal to T_i)
7	Add 1 to the value of *i*
8	If (*Found* = NO) then
9	Print the message 'Sorry, this number is not in the directory'
10	Stop

Best Case	Worst Case	Average Case
1	*n*	*n*/2

Number of comparisons to find NUMBER in a list of *n* numbers using sequential search

Sorting Problem – Selection Sort

1. Get values for *n* and the *n* list items
2. Set the marker for the unsorted section at the end of the list
3. While the unsorted section of the list is not empty, do Steps 4 through 6
4. Select the largest number in the unsorted section of the list
5. Exchange this number with the last number in the unsorted section of the list
6. Move the marker for the unsorted section left one position
7. Stop

Selection sort algorithm

5, 7, 2, 8, 3 |

Unsorted subsection (entire list) Sorted subsection (empty)

5, 3, 2, | 7, 8

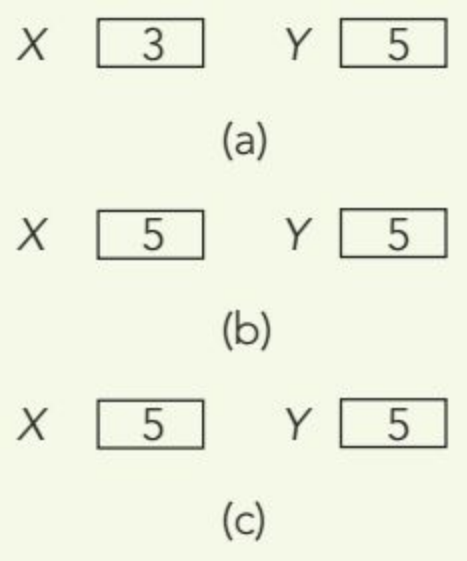
Unsorted subsection Sorted subsection

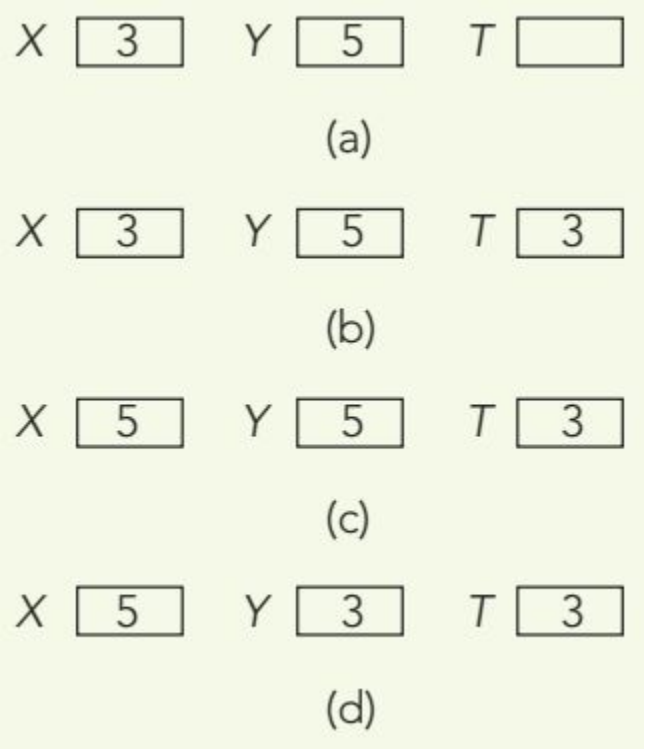
| 2, 3, 5, 7, 8

Unsorted subsection (empty) Sorted subsection (entire list)

Warning: Exchange Values

1. Copy the current value at position Y into position X
2. Copy the current value at position X into position Y





1. Copy the current value at position X into location T
2. Copy the current value at position Y into position X
3. Copy the current value at location T into position Y

HUST