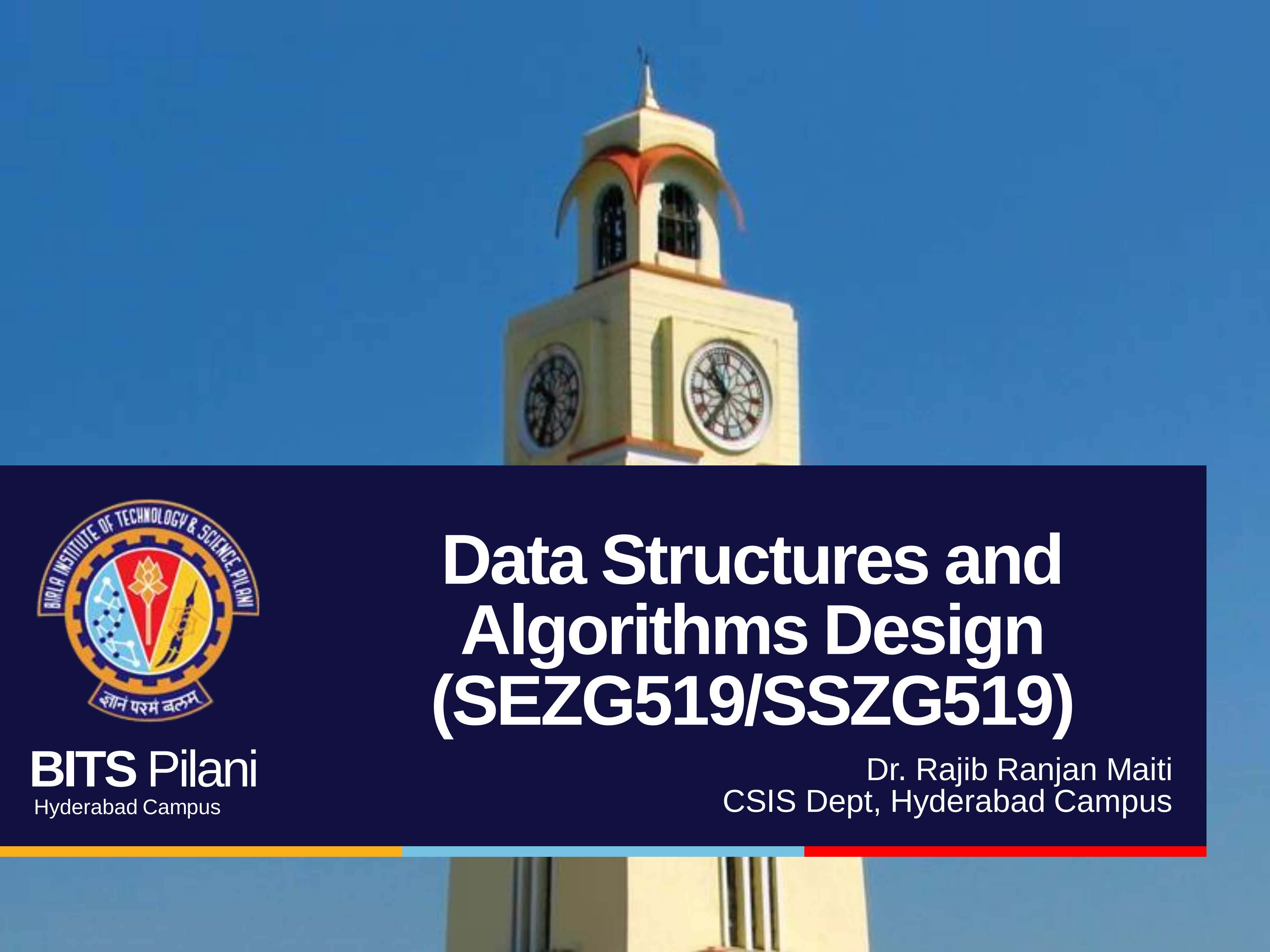


BITS Pilani
Hyderabad Campus

Data Structures and Algorithms Design (SEZG519/SSZG519)

Dr. Rajib Ranjan Maiti
CSIS Dept, Hyderabad Campus

innovate achieve lead

BITS Pilani
Hyderabad Campus

S1 Algorithms, RAM, Time Complexity, Notation of Correctness

Content of S1

1. Algorithms and it’s Specification
2. Random Access Machine Model
3. Notion of best case, average case and worst case
4. Notion of Algorithm Correctness

Content of S1

1. **Algorithms and it’s Specification**
2. Random Access Machine Model
3. Notion of best case, average case and worst case
4. Notion of Algorithm Correctness

Discovery of A Tool

Famous Mathematician Archimédes

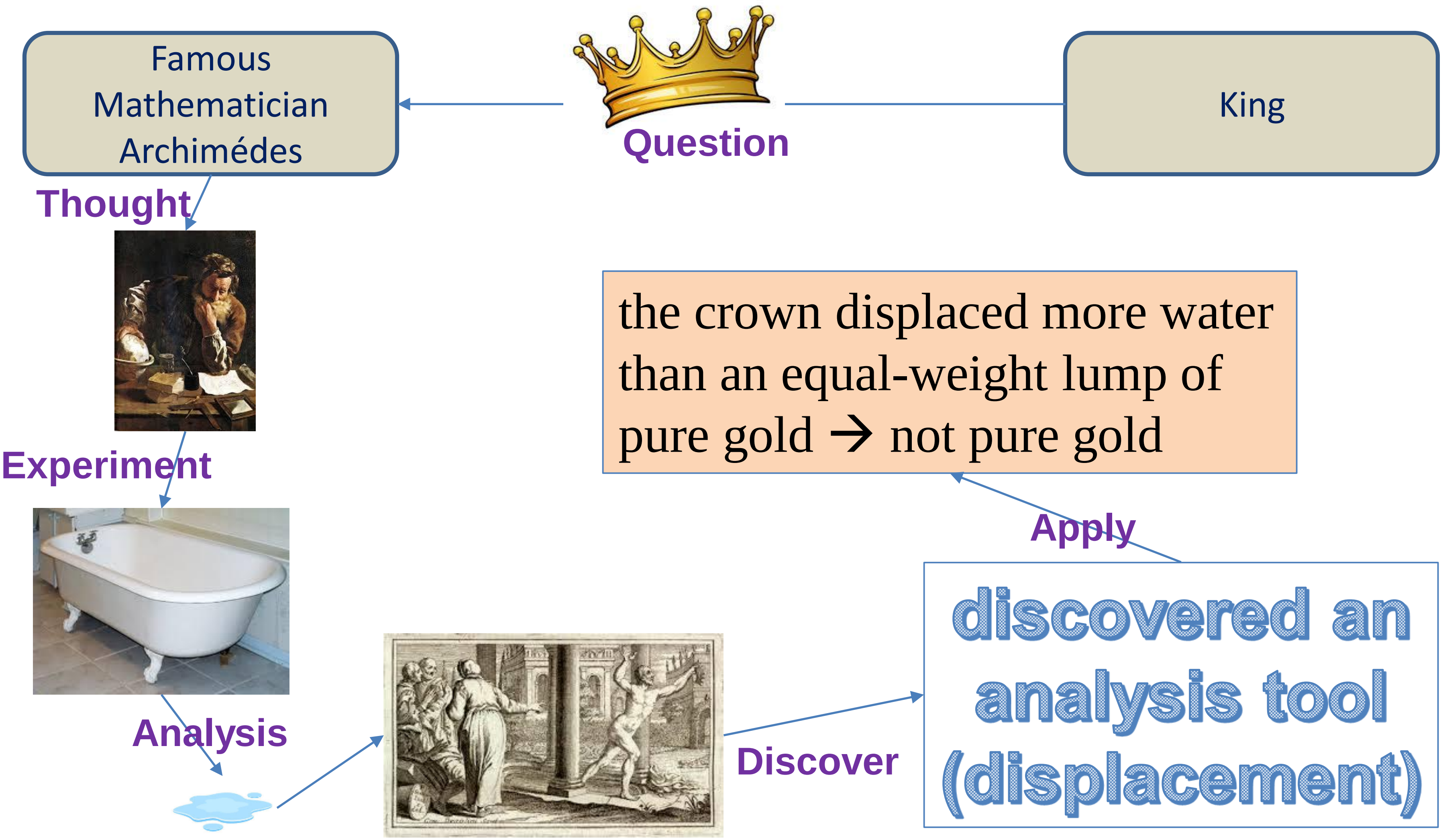
Is it pure Gold? Or, it has silver mixed?

King

the crown displaced more water than an equal-weight lump of pure gold → not pure gold

discovered an analysis tool (displacement)

Discovery of A Tool



Famous Mathematician Archimédes

Question

King

Thought

Experiment

Analysis

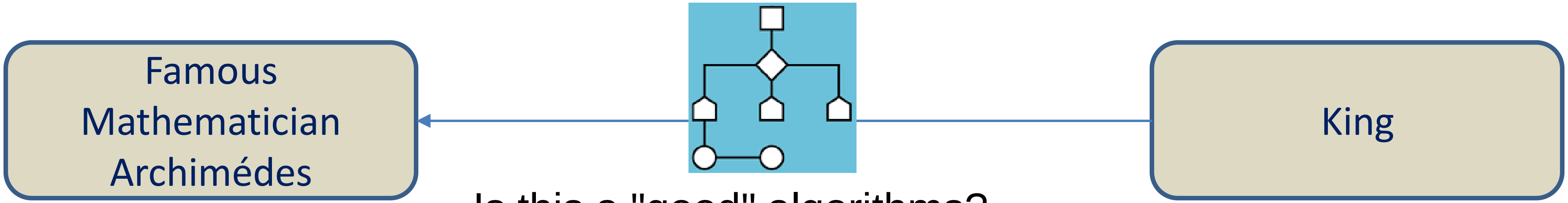
Discover

Apply

the crown displaced more water than an equal-weight lump of pure gold → not pure gold

discovered an analysis tool (displacement)

How analogy fits to Algorithm?

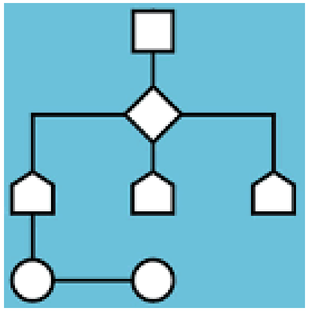


Famous Mathematician Archimédes

King

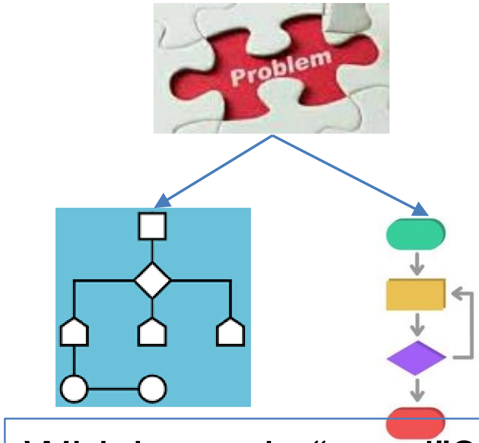
Is this a "good" algorithms?
Or, Is this a “good” data structure?

But, what is an algorithm?



An algorithm: a **step-by-step** procedure for performing some task in a **finite amount of time**

A data structure: a systematic way of **organizing** and **accessing** data.

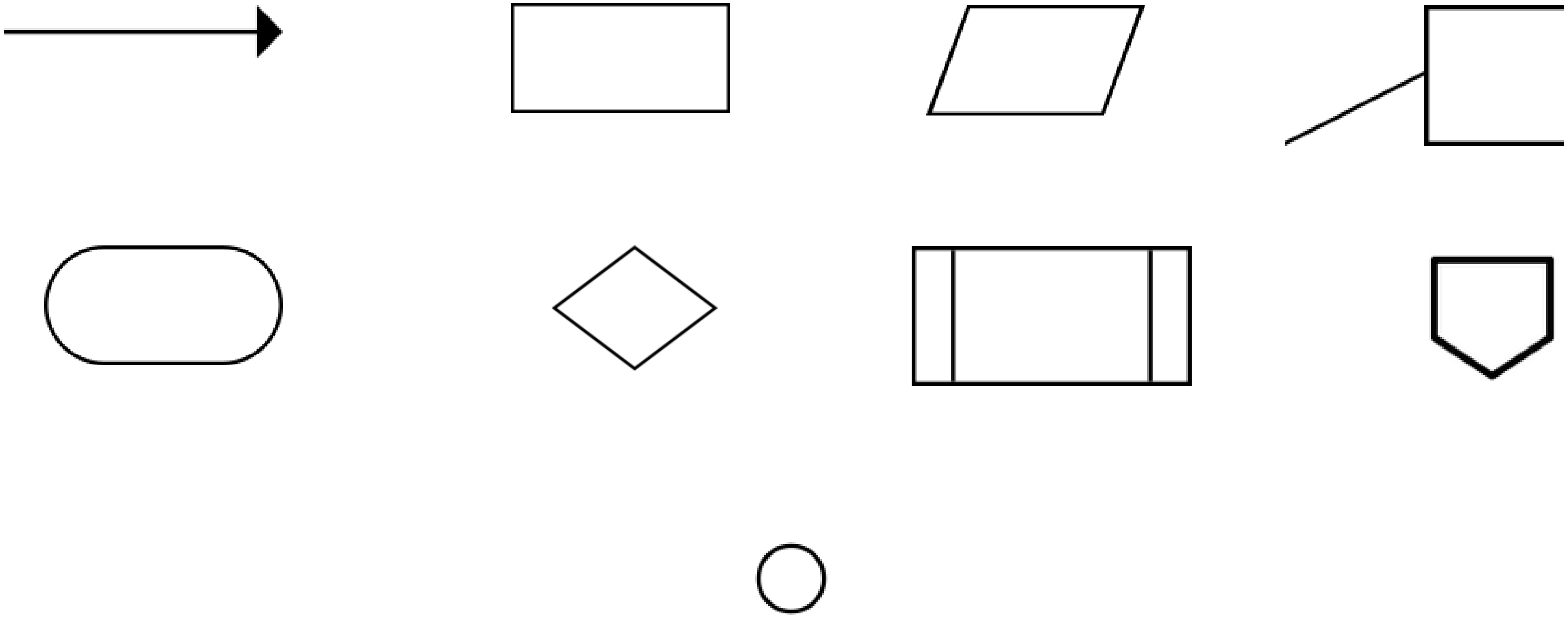


Which one is “good”?
What is the measure of “goodness”?

How to develop an algorithmic thought?

Answer: Flowchart
ANSI set standards for flowcharts and their symbols in the 1960s

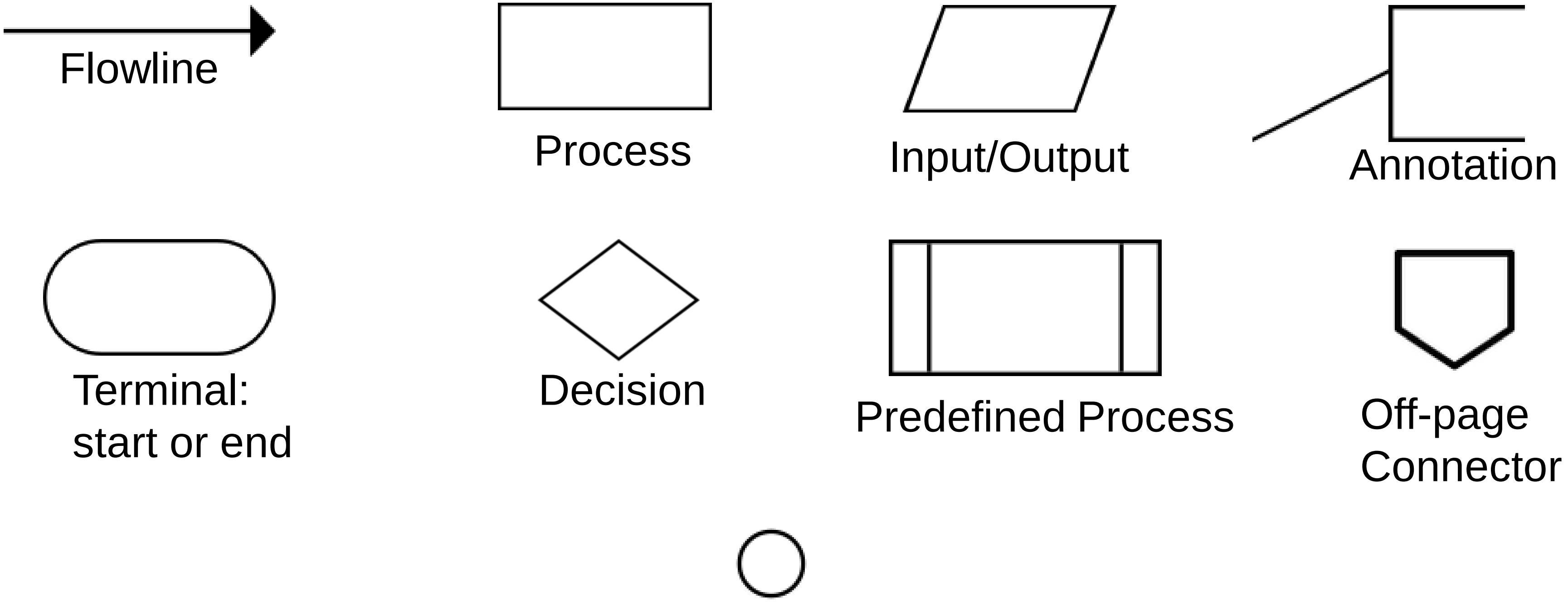
Flowchart
flows from top to bottom and left to right



How to develop an algorithmic thought?

Answer: Flowchart
ANSI set standards for flowcharts and their symbols in the 1960s

Flowchart
flows from top to bottom and left to right



Flowline

Process

Input/Output

Annotation

Terminal:
start or end

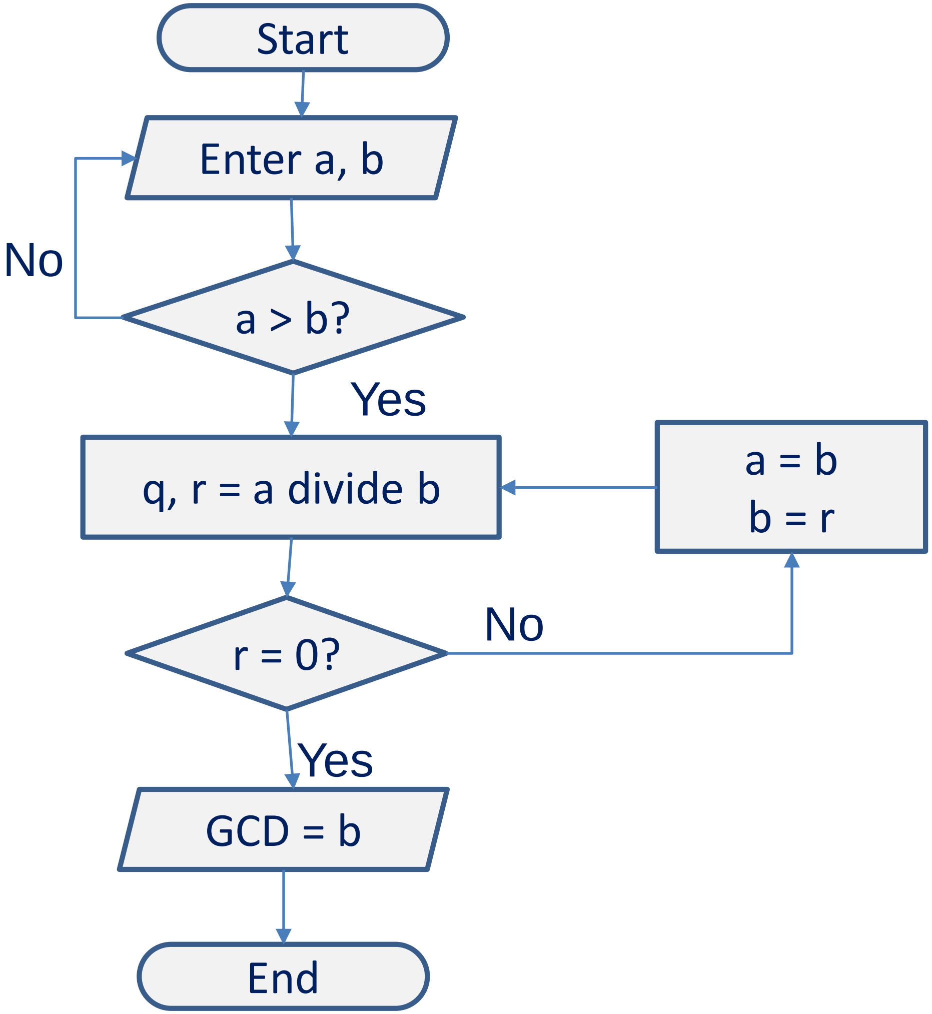
Decision

Predefined Process

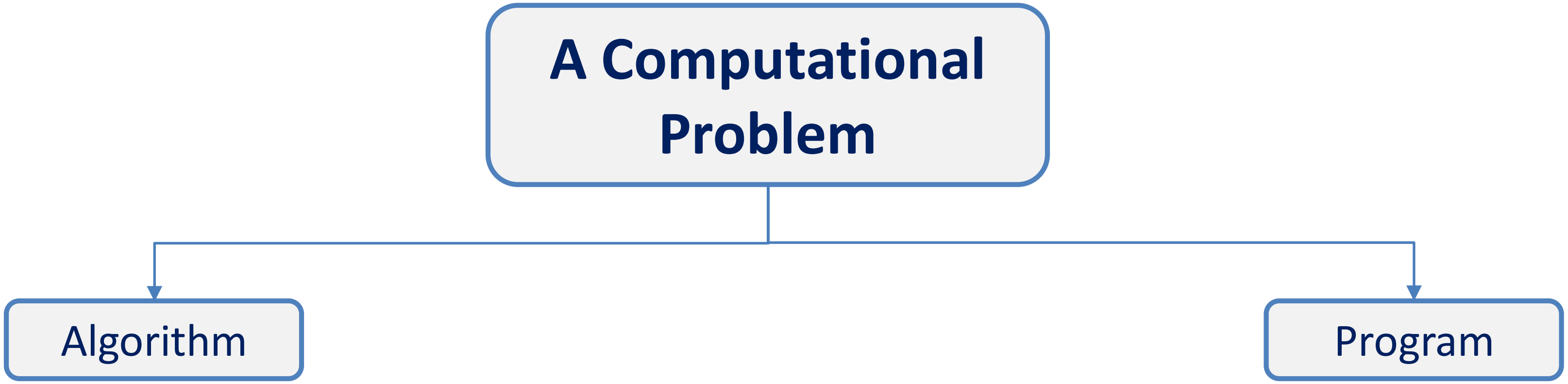
Off-page
Connector

On-page Connector

Example of Flowchart: GCD



How to Write an Algorithm?



A Computational Problem

Algorithm

Program

Well defined Computational steps

Time Complexity: An estimation of execution time based on input size, not based on number of program statements

Based on well defined estimated time of individual steps

An exact sequence of statement written in a particular programming language

Execution time: depends on instruction set architecture, processor speed, operating system, compiler or interpreter

Still get an average run time for the program

Advantage of Run time

When we are interested in an accurate run time measurement, a program would need to be executed and, during execution, exact time for system calls and user functions can be calculated

Obviously, such a time measurement will depend on the language, it library, operating system and so on

However, we can then visualize the results of such experiments by plotting the execution time of each run of the program on y-axis and input size on x-coordinate

An example of run time analysis

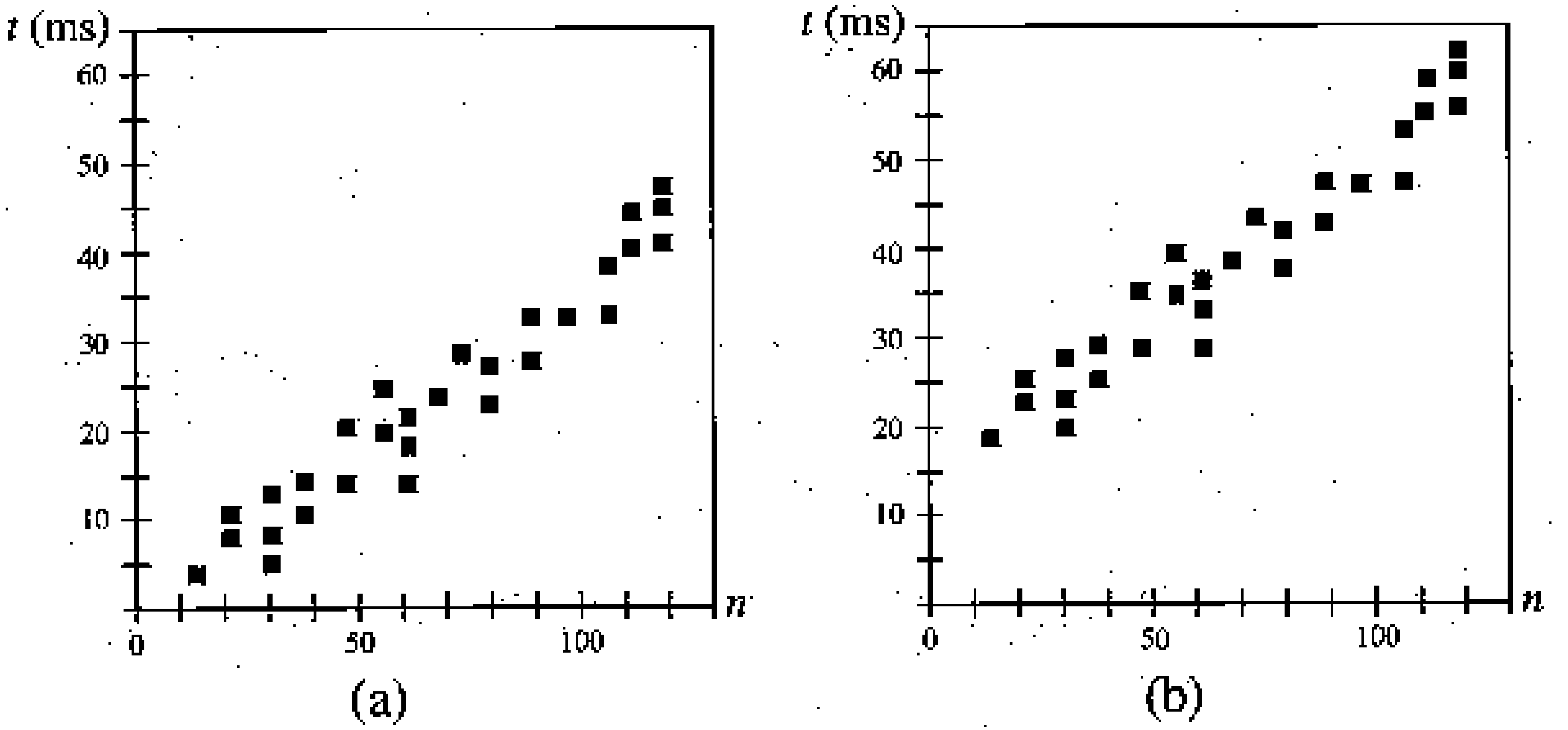


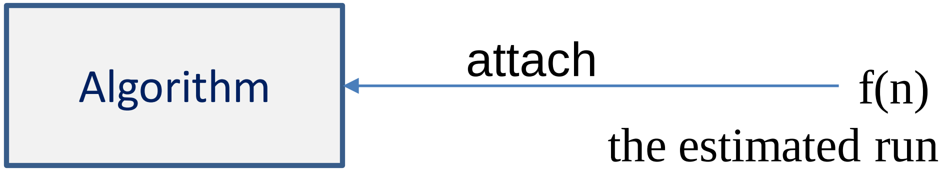
Figure 1.1: Results of an experimental study on the running time of a program.
A dot with coordinates (n, t) indicates that on an input of size n, the running time of the program is t milliseconds (ms)
(a) The algorithm executed on a fast computer;
(b) the algorithm executed on a slow computer.

Limitation of Run time of a program

- Experiments can be done only on a limited set of test inputs, and care must be taken to make sure these are representative.
- It is difficult to compare the efficiency of two algorithms unless experiments on their running times have been performed in the same hardware and software environments.
- It is necessary to implement and execute an algorithm in order to study its running time experimentally.

Advantage of Time Complexity of Algorithm

- Takes into account all possible inputs
- Evaluates the relative efficiency of two algorithms independent of the hardware and software environment
- Performs a study on a high-level description of the algorithm without actually implementing or running it



the estimated run time of the algorithm in terms of the input size n

For example, “Algorithm A runs in time proportional to n” → if we were to perform experiments, we would find that the actual running time of algorithm A on any input of size n never exceeds cn, where c is a constant that depends on the hardware and software environment used in the experiment.

Language of Algorithm

- A language for describing algorithms
- A computational model that algorithms execute
- A metric for measuring algorithm running time
- An approach for characterizing running times, including those for recursive algorithms.

Pseudocode

A mixture of natural language and high-level programming constructs that describe the main ideas behind a generic implementation of a data structure or algorithm.

Expression:
1) standard mathematical symbols to express numeric and Boolean expressions
2) left arrow (←) is the assignment operator
3) equal sign (=) is the equality relation in Boolean expressions

Method declarations:
Algorithm name(paraml , param2 ,...)

Decision structures:
1) if condition, then true-actions [else false-actions].

indentation to indicate a block of expressions

Pseudocode

Loops:

1) While-loops: while condition do actions.
2) Repeat-loops: repeat actions until condition.
3) For-loops for variable-increment-definition do actions

indentation to indicate a block of expressions

Array indexing:

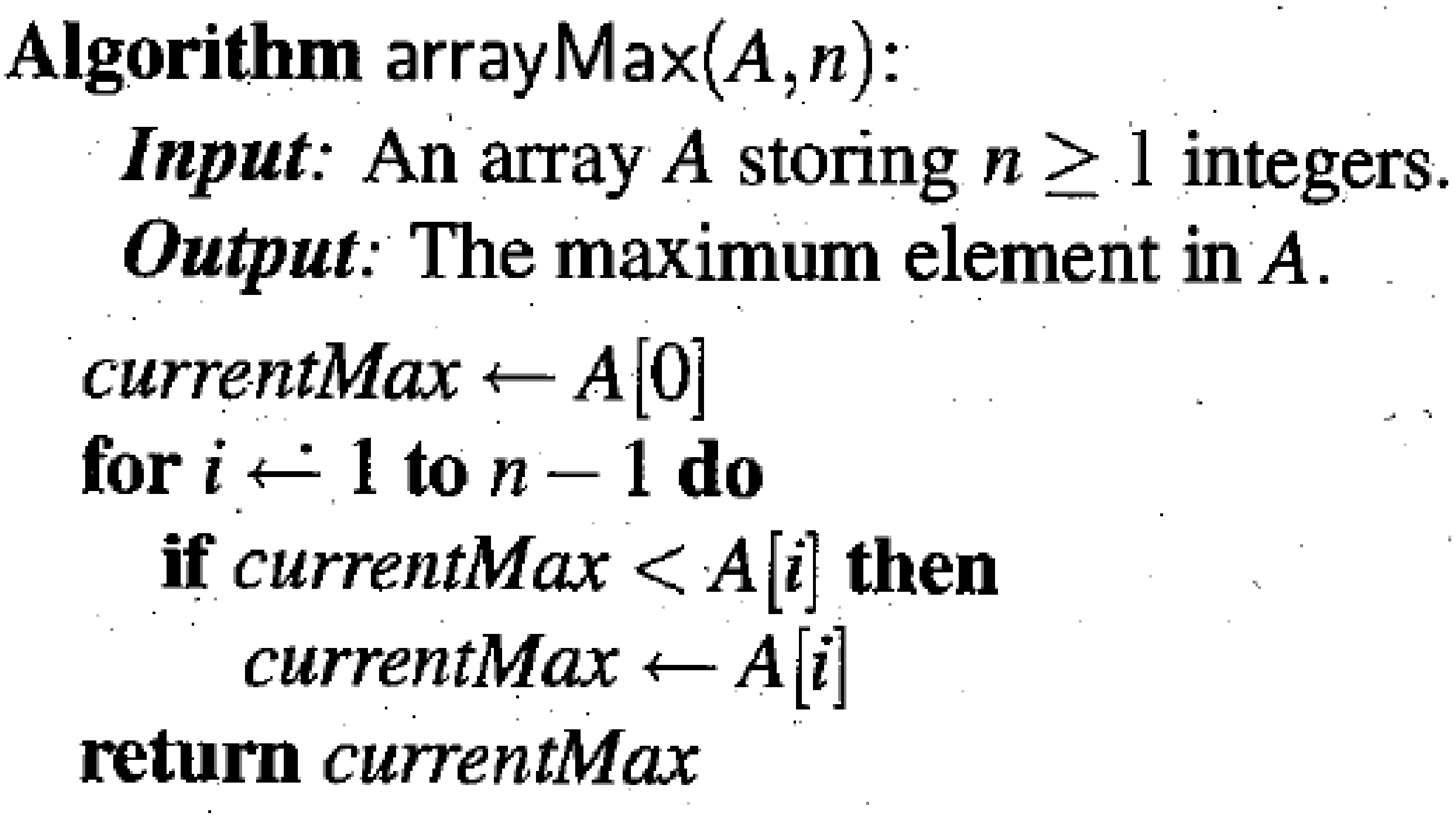
1) A[i] represents the ith cell in the array A
2) The cells' of an n-celled array A are indexed from A[0] to A[n-1]

Method calls: object.method(args) (object is optional)

Method returns: return value

Example Pseudocode

Problem: Find maximum number in and array of n numbers

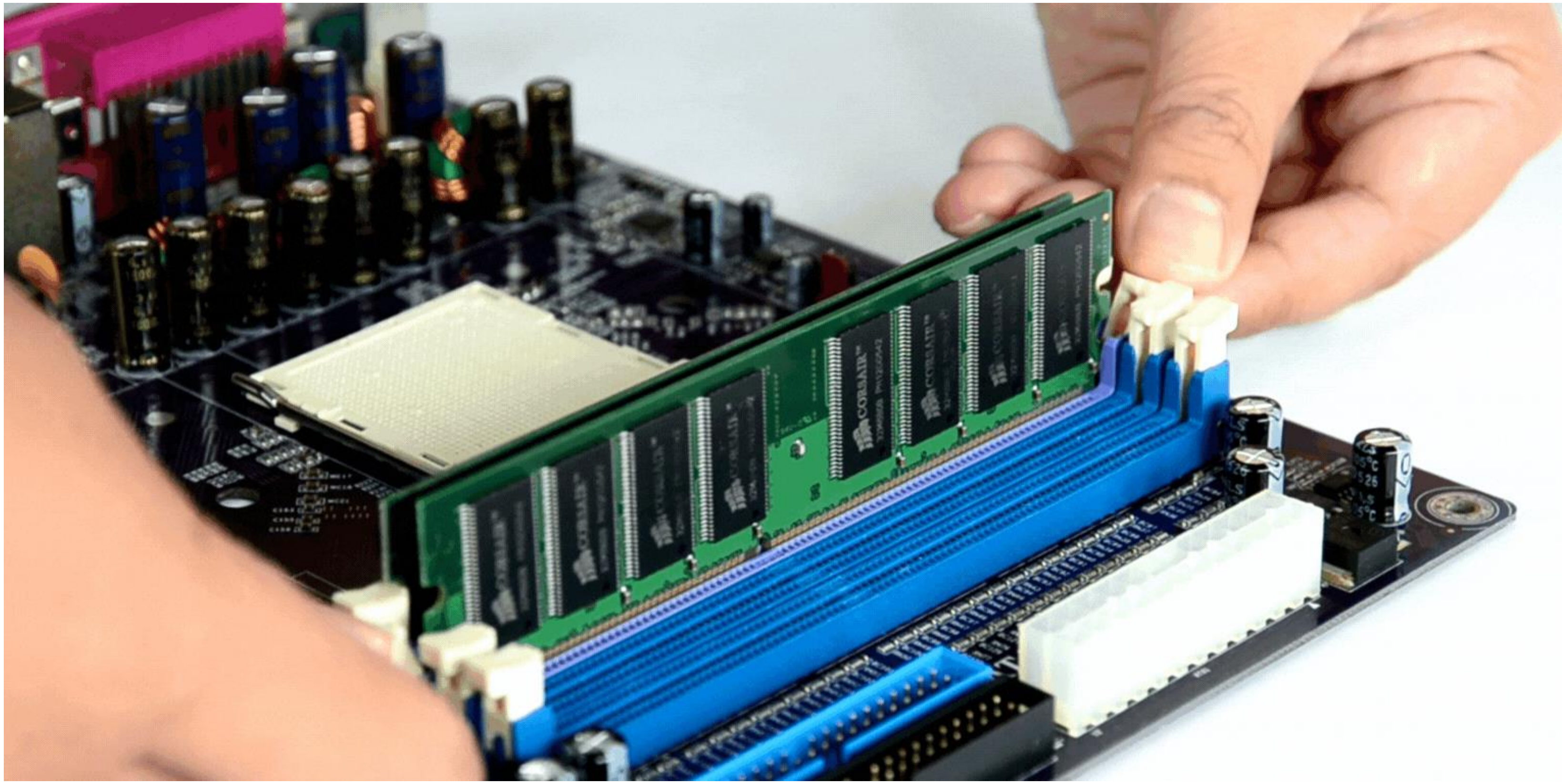


```
Algorithm arrayMax($A, n$):
    Input: An array $A$ storing $n \geq 1$ integers.
    Output: The maximum element in $A$.
  $currentMax \leftarrow A[0]$
  for $i \leftarrow 1$ to $n - 1$ do
      if $currentMax < A[i]$ then
          $currentMax \leftarrow A[i]$
  return $currentMax$
```

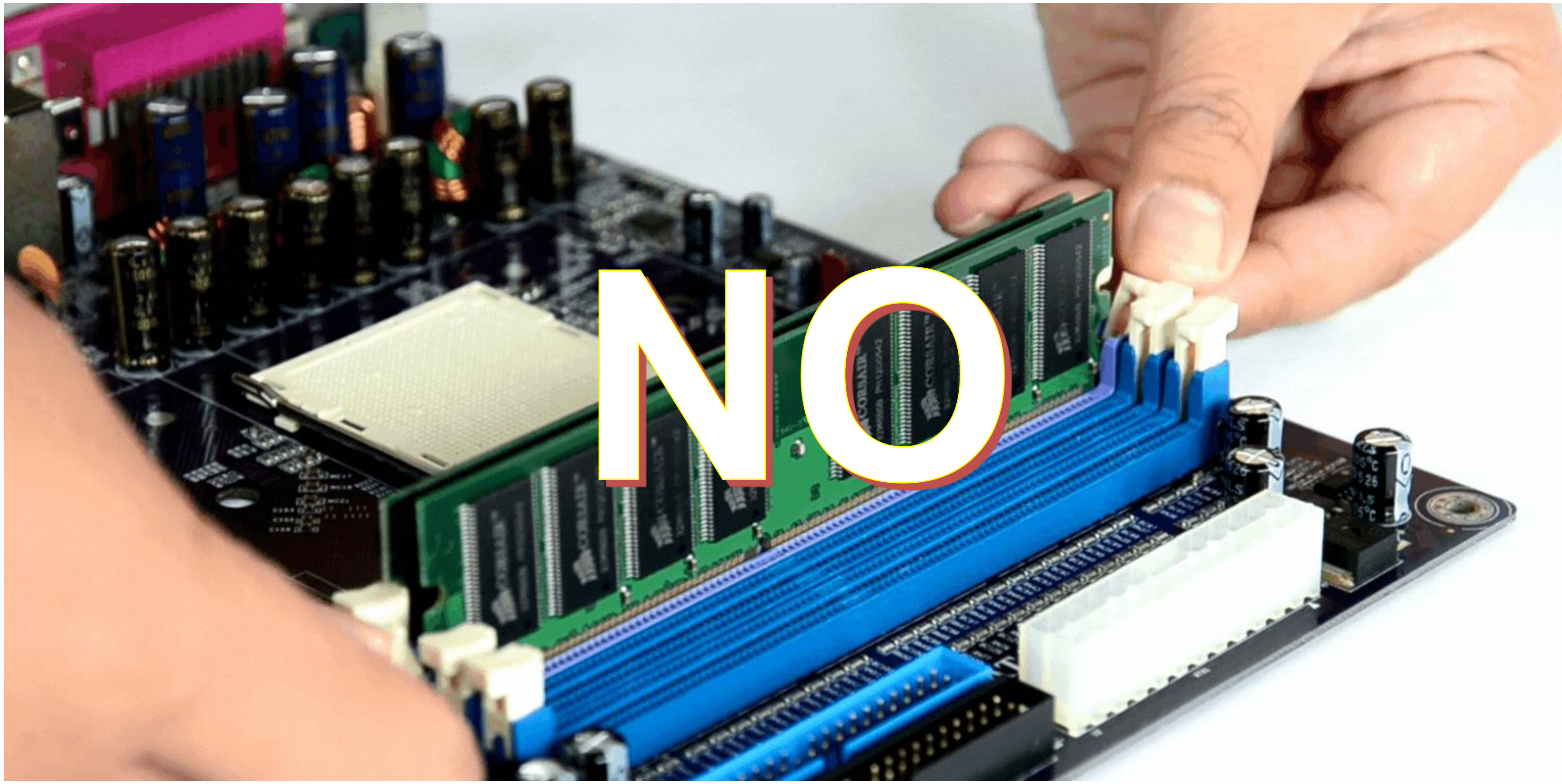
Content of S1

1. Algorithms and it’s Specification
2. **Random Access Machine Model**
3. Notion of best case, average case and worst case
4. Notion of Algorithm Correctness

Random Access Machine (RAM) Model



Random Access Machine (RAM) Model



Analysis on Pseudocode

- Experimental analysis is valuable, but has limitations
- If we wish to analyze a particular algorithm without performing experiments on running time, then we can analyze high-level code or pseudo-code

- For that to happen, we must identify high-level primitive operations
- Primitive operations
  - **independent of the programming language**
  - **available in pseudocode**

Analysis on Pseudocode

- What are primitive operations?

1) Assigning a value to variable	5) Calling a method
2) Performing an arithmetic operation	6) Comparing two basic numbers
3) Indexing into an array	7) Following an object reference
4) Returning from a method	

Instead of trying to determine the specific execution time of each primitive operation, we will simply count how many primitive operations are executed.

We shall use this number t as a high-level estimate of the running time of the algorithm

Analysis on Pseudocode

What shall we do with the primitive operations?

- Instead of finding their execution time, we simply count the number of primitive operations
- This count will be proportional to the actual running time
- The assumption is that primitive operations takes almost similar execution time in any execution environment

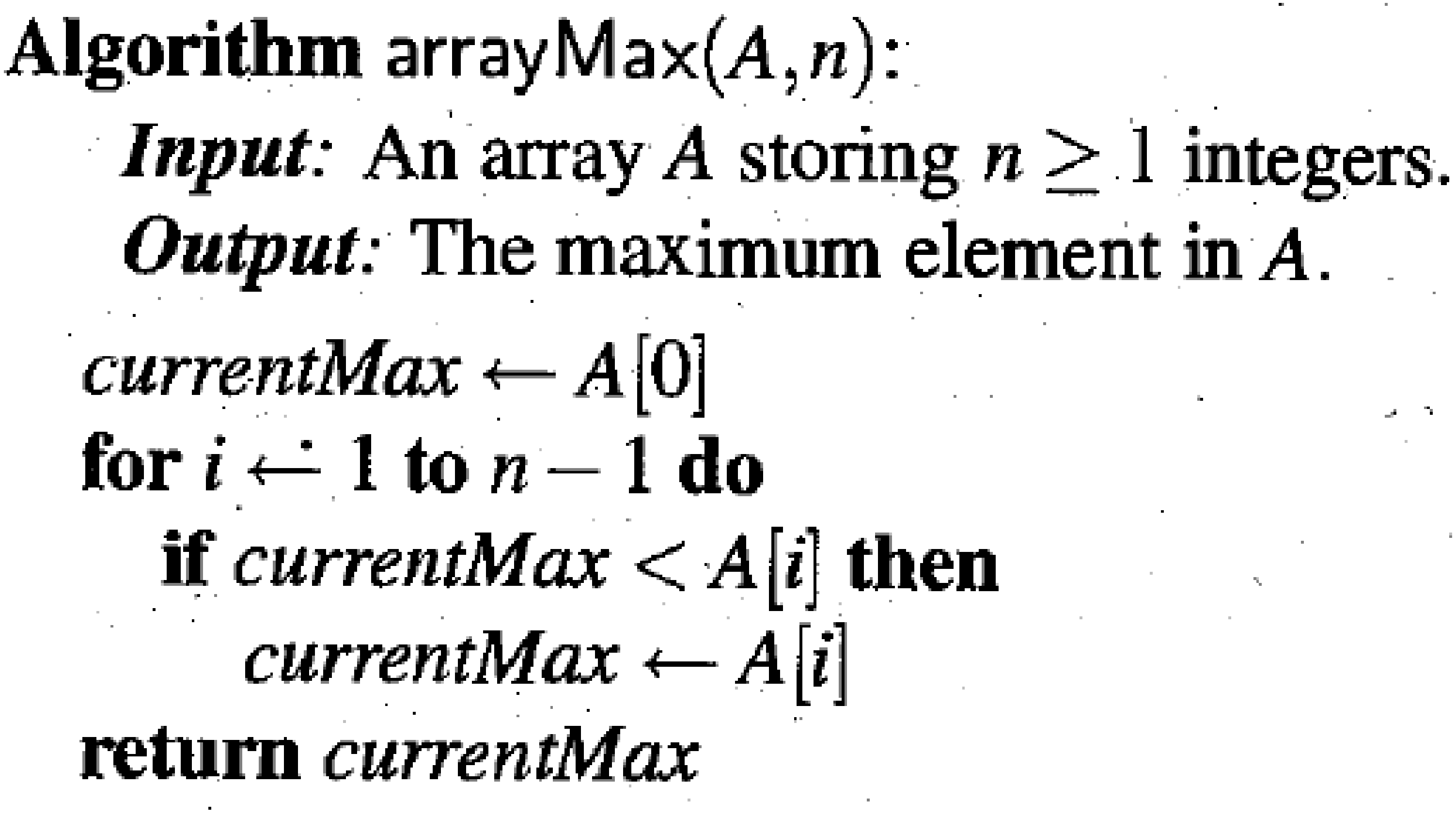
Random Access Machine (RAM) Model

So What is RAM model?

- This approach of simply counting primitive operations gives rise to a computational model called the Random Access Machine (RAM) model
- We assume that the CPU in RAM model can perform any primitive operation in a constant number of steps and this will not depend on the size of input

Example: RAM model

So, how to Count Primitive Operations?



Algorithm arrayMax(A, n):

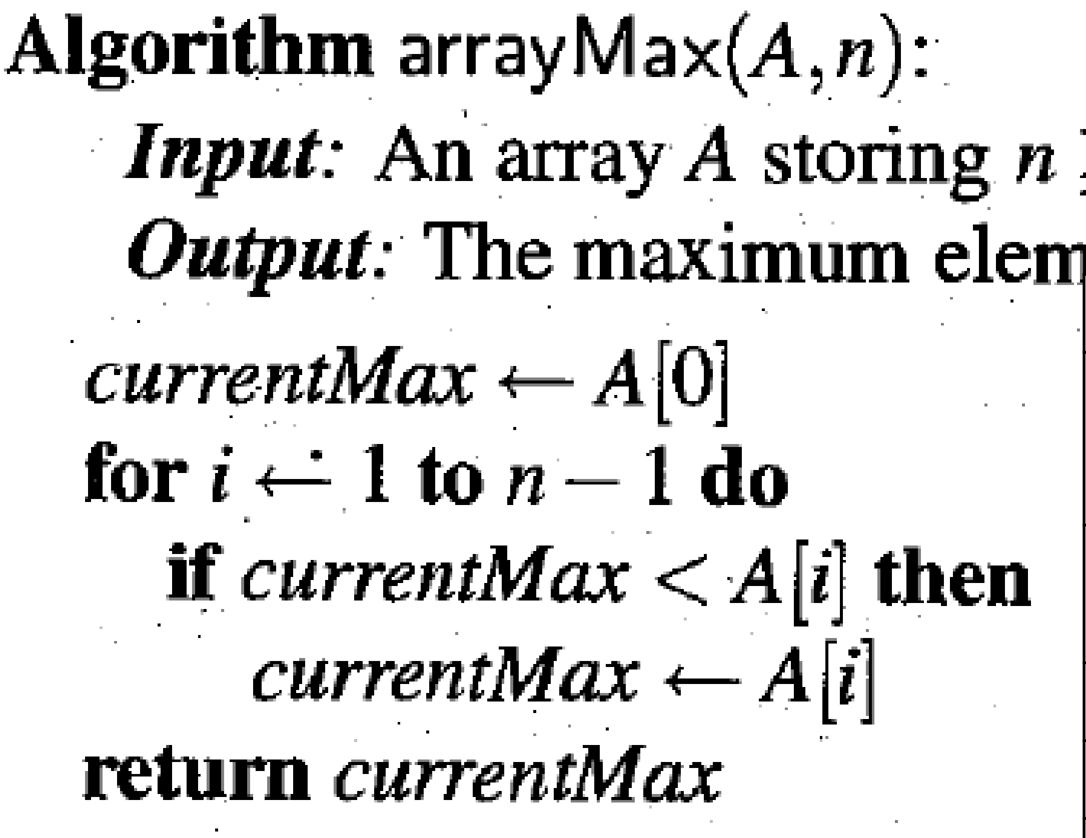
Input: An array A storing $n \geq 1$ integers.

Output: The maximum element in A.

```
$currentMax \leftarrow A[0]$
for $i \leftarrow 1$ to $n - 1$ do
    if $currentMax < A[i]$ then
        $currentMax \leftarrow A[i]$
return $currentMax$
```

Example: RAM model

So, how to Count Primitive Operations?



```
Algorithm arrayMax(A, n):
  Input: An array A storing $n \geq 1$ integers.
  Output: The maximum element in A

  $currentMax \leftarrow A[0]$
  for $i \leftarrow 1$ to $n - 1$ do
      if $currentMax < A[i]$ then
          $currentMax \leftarrow A[i]$
  return $currentMax$
```

1 (access A[0]) +1 (assign)= 2
1 (initialize i) n (check i<n) 2(n-1) (i+1 and assign)
(n-1) (access A[i]) (n-1) (check currentMax < A[i])
(n-1) (access A[i]) + (n-1) (assign)
0 to n-1
1 (return)

Example: RAM model

Thus, the body of the loop contributes between 4(n - 1) and 6(n - 1) units to the count

To summarize, the number of primitive operations t(n) executed by Algorithm arrayMax is at least

2+1+n+4(n-1)+1 = 5n

and at most

2+1+n+6(n-1)+l=7n-2

Content of S1

1. Algorithms and it’s Specification
2. Random Access Machine Model
3. **Notion of best case, average case and worst case**
4. Notion of Algorithm Correctness

Analysis of Time Complexity

- Best case:
  - Minimum number of primitive operations.
  - When? **max** is at the first place in array.
  - t(n) = 5n

Analysis of Time Complexity

- Best case:
  - Minimum number of primitive operations.
  - When? **max** is at the first place in array.
  - t(n) = 5n
- Worst case:
  - maximum number of primitive operations
  - When? max is at the last place in array.
  - t(n) = 7n - 2

Analysis of Time Complexity

- Best case:
  - Minimum number of primitive operations.
  - When? **max** is at the first place in array.
  - t(n) = 5n
- Worst case:
  - maximum number of primitive operations
  - When? max is at the last place in array.
  - t(n) = 7n - 2
- Average case:
  - average number of primitive operations over several runs
  - When? Testcases should cover all possible cases of **max** being in any place
  - typically a difficult task

Graphical representation of time complexity

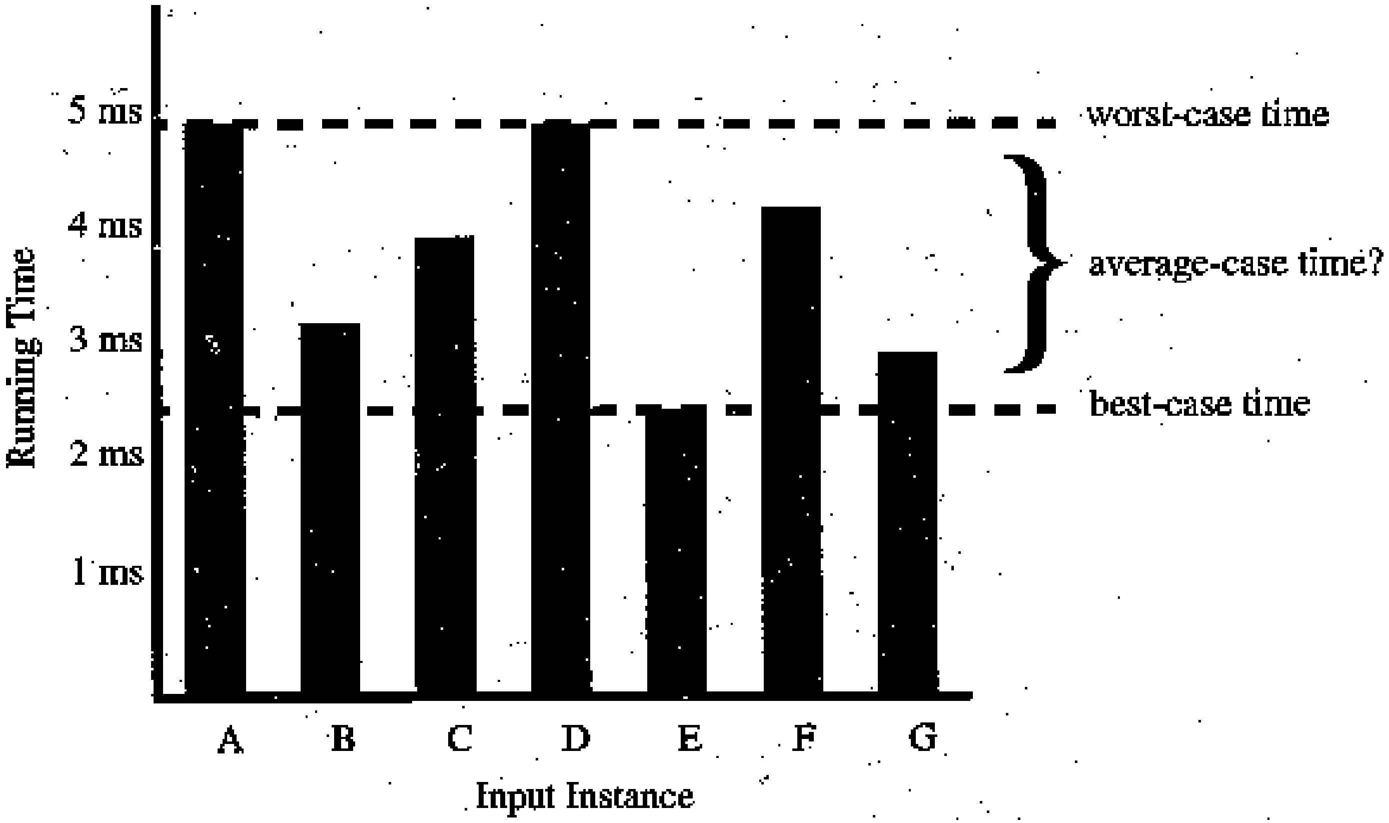


Figure 1.3: The difference between best-case and worst-case time. Each bar represents the running time of an algorithm on a different possible input.

Content of S1

1. Algorithms and it's Specification
2. Random Access Machine Model
3. Notion of best case, average case and worst case
4. **Notion of Algorithm Correctness**

Using Pseudo-Code to Prove Algorithm Correctness

By inspecting the pseudo-code, we can argue about the correctness of algorithm “arrayMax”.

How?

→Variable currçntMax starts out being equal to the first element of A – base case

→We claim that at the beginning of the ith iteration of the loop, currentMax is equal to the maximum of the first i elements in A, i takes values from 0

→ Since we compare currentMax to A [i] in iteration i, if this claim is true before this iteration, it will be true after it for i + 1 (which is the next value of counter i)

→ Thus., after n - 1 iterations, currentMax will equal the maximum element in A

Using Pseudo-Code to Prove Algorithm Correctness

By inspecting the pseudo-code, we can argue about the correctness of algorithm “arrayMax”.

How?

→Variable currçntMax starts out being equal to the first element of A – base case

Using Pseudo-Code to Prove Algorithm Correctness

By inspecting the pseudo-code, we can argue about the correctness of algorithm “arrayMax”.

How?

→Variable currçntMax starts out being equal to the first element of A – base case

→We claim that at the beginning of the ith iteration of the loop, currentMax is equal to the maximum of the first i elements in A, i takes values from 0

Using Pseudo-Code to Prove Algorithm Correctness

By inspecting the pseudo-code, we can argue about the correctness of algorithm “arrayMax”.

How?

→Variable currçntMax starts out being equal to the first element of A – base case

→We claim that at the beginning of the ith iteration of the loop, currentMax is equal to the maximum of the first i elements in A, i takes values from 0

→ Since we compare currentMax to A [i] in iteration i, if this claim is true before this iteration, it will be true after it for i + 1 (which is the next value of counter i)

Using Pseudo-Code to Prove Algorithm Correctness

By inspecting the pseudo-code, we can argue about the correctness of algorithm “arrayMax”.

How?

→Variable currçntMax starts out being equal to the first element of A – base case

→We claim that at the beginning of the ith iteration of the loop, currentMax is equal to the maximum of the first i elements in A, i takes values from 0

→ Since we compare currentMax to A [i] in iteration i, if this claim is true before this iteration, it will be true after it for i + 1 (which is the next value of counter i)

→ Thus., after n - 1 iterations, currentMax will equal the maximum element in A

Correctness of Algorithms

- An algorithm is said to be correct
  - if, for every input instance, it halts with the correct output.
- We say that a correct algorithm
  - solves the given computational problem.
- An incorrect algorithm
  - might not halt at all on some input instances, or
  - it might halt with an incorrect answer.

References

1. Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, Wiley (Students Edition)
2. Data Structures, Algorithms and Applications in C++, Sartaj Sahni, Second Ed, 2005, Universities Press
3. Introduction to Algorithms, TH Cormen, CE Leiserson, RL Rivest, C Stein, Third Ed, 2009, PHI

Theoretical Foundation

Data structure

- Data: value or set of values. E.g. Rohit, 34, {11,33,2,51}, etc.
- Data structure: Logical or mathematical model of particular organization of data is called data structure.
- Array: A list of finite number of similar data elements. A[1], A[2], ..., A[N].
- Linked List: A list linked to one another.
- Stack: A linear list following Last In First Out system (LIFO).
- Queue: A linear list following First In First Out system (FIFO).
- Graph: A non-linear list consisting vertices and edges.
- Tree: A graph without cycles.

Theoretical Foundation

Data structure operations

- Traversing
- Searching
- Inserting
- Deleting
- Sorting
- Merging

innovate achieve lead

BITS Pilani
Hyderabad Campus

Any Question!!

Thank you!!

BITS Pilani
Hyderabad Campus