



Introduction To Algorithms

Lecture 1 – Algorithm and Flowcharts

2nd Year, 2nd Semester – Musa Hodman

06 Sep, 2022

Contents

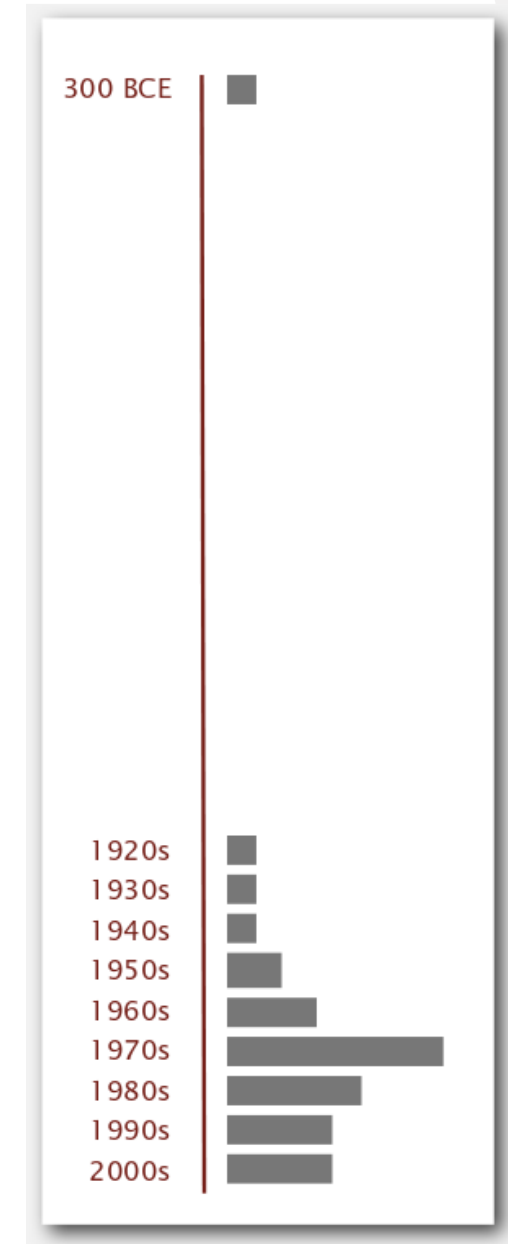
- Algorithm definition
 - Algorithms specification
 - What kinds of problems are solved by algorithms?
- Algorithms as a technology
 - Efficiency
 - Algorithm and other technologies
- Pseudo-code definition
 - Pseudo-code examples
 - Categories of Algorithms operations
 - Rules for Pseudo-code
- Programming design
- Flowchart

Why Study Algorithms?

- Their impact is broad and far-reaching
 - Internet. Web search, packet routing, distributed and sharing, ...
 - Biology. Human genome project, protein folding, ...
 - Computers. Circuit layout, compilers, ...
 - Computer Graphics. Movies, video games, virtual reality, ...
 - Security. Cell phones, e-commerce, voting machines, ...
 - Multimedia. MP3, JPG, DivX, HDTV, face recognition, ...
 - Social Network. Recommendations, news feeds, advertisements,...
 - Physics. N-body simulation, particle collision simulation, ...

Why Study Algorithms?

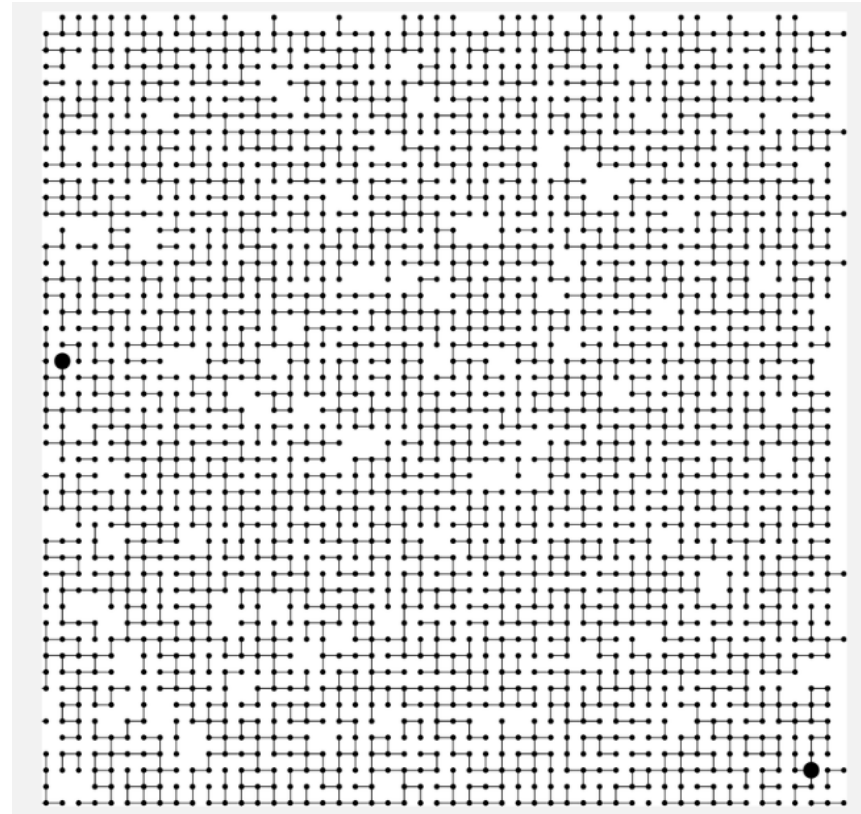
- Study of algorithms dates at least to Euclid.
- Formalized by Church and Turing in 1930s.
- Some important algorithms were discovered by undergraduates in a course like this!



Why Study Algorithms?

- To solve problems that could not otherwise be addressed.

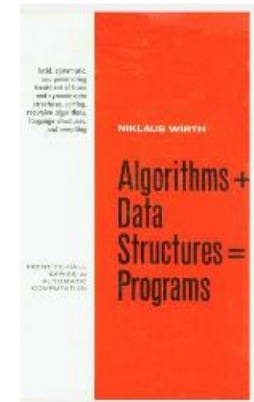
Ex: Network connectivity



Why Study Algorithms?

- To become a proficient programmer.

I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships. Linus Torvalds (creator of Linux)



Algorithms + Data Structures = Programs. Niklaus Wirth

Why Study Algorithms?

- They may unlock the secrets of life and of the universe
- Computational models are replacing math models in scientific inquiry.

$$\begin{aligned} E &= mc^2 \\ F &= ma \qquad F = \frac{Gm_1m_2}{r^2} \\ \left[-\frac{\hbar^2}{2m} \nabla^2 + V(r) \right] \Psi(r) &= E \Psi(r) \end{aligned}$$

20th century science
(formula based)

```
for (double t = 0.0; true; t = t + dt)
  for (int i = 0; i < N; i++)
  {
    bodies[i].resetForce();
    for (int j = 0; j < N; j++)
      if (i != j)
        bodies[i].addForce(bodies[j]);
  }
```

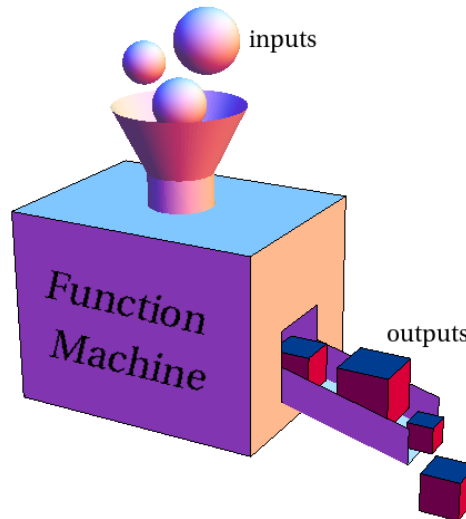
21st century science
(algorithm based)

Algorithms: a common language for nature, human, and computer.
Avi Wigderson

Algorithms Definition

Informally, an algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm is thus a sequence of computational steps that transform the input into the output.



Algorithm Definition in Computer Science

The term algorithm is used in computer science to describe a problem-solving method suitable for implementation as a computer program.

Example

- We might need to sort a sequence of numbers into no decreasing order.

Here is how we formally define the sorting problem.

1. **Input:** A sequence of n numbers ($a_1; a_2; \dots a_n$).
2. **Output:** A permutation (reordering) $(a'_1, a'_2, \dots, a'_n)$ of the input sequence such that $a'_1 \leq a'_2 \leq a'_3$

Given the input sequence (31; 41; 59; 26; 41; 58), a sorting algorithm returns as output the sequence (26; 31; 41; 41; 58; 59). Such an input sequence is called an instance of the sorting problem.

Algorithm Specification

- Correctness:

- 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.

- Running Time:

- It is not enough for a computation to eventually get the correct answer. It must also do so using a reasonable amount of time and memory space.

What Kinds of Problems are Solved by Algorithms?

1. The Human Genome Project has made great progress toward the goals of identifying all the 100,000 genes in human DNA (Needs Database).
2. The Internet enables people all around the world to quickly access and retrieve large amounts of information.
3. Electronic commerce enables goods and services to be negotiated and exchanged electronically.
4. Manufacturing and other commercial enterprises often need to allocate scarce resources in the most beneficial way.

Algorithm as a Technology

Suppose computers were infinitely fast and computer memory was free. Would you have any reason to study algorithms?

Algorithms as a Technology

- Of course, computer may be fast, but they are not infinitely fast. And computer memory may be inexpensive, but it is not free.
- Computing time is therefor a bounded resource, and so is space in memory.
- These resources should be used wisely, and algorithms that are efficient in terms of time or space will help you do so.

Efficiency

- Different algorithms devised to solve the same problem often differ dramatically in their efficiency.
- These differences can be much more significant than differences due to hardware and software.

Problem

- Efficiency of insertion sort($2n^2$) vs merge sort ($50n\log n$)
 - Computer A running insertion sort.
 - Computer B running merge sort.
 - They each must sort an array of 10 million numbers.
- Suppose that
 - Computer A executes 10 billion instructions per second.
 - Computer B executes 10 million instructions per second.

Solution

- Computer A: Insertion Sort ($2n^2$)

- $\frac{2 \cdot (10^7)^2 \text{ Instructions}}{10^{10} \text{ Instructions/second}} = 20000 \text{ seconds}$ (More than 5.5 hours)

- Computer B: Merge Sort ($50n \log n$)

- $\frac{50 \cdot 10^7 \cdot \log 10^7 \text{ instructions}}{10^7 \text{ instructions/second}} \approx 1163 \text{ seconds}$ (less than 20 min)

Algorithms and Other Technologies

- The example above shows that we should consider algorithms, like computer hardware, as a technology.
- Total system performance depends on choosing efficient algorithms as much as on choosing fast hardware.
 1. Advanced computer architecture and fabrication technologies.
 2. Easy-to-use, intuitive, graphical user interfaces (GUIs)
 3. Object-oriented systems.
 4. Integrated web technologies.
 5. Fast networking, both wired and wireless.

Pseudo-code Definition

Pseudocode is an artificial and informal language that helps programmers develop algorithms.

Pseudocode is a text-based detail(algorithmic) design tool.

Pseudo-code

1. Pseudo-code is similar to **C**, **Python** or **Java**
2. What separates pseudo-code from real code is that in pseudo-code, we employ whatever expressive method is most clear and concise to specify a given algorithm.
3. Sometimes, the clearest method is English.
4. Pseudocode is not typically concerned with issues of software engineering.

Pseudo-code Example 1

- Start
- If student grade is greater than or equal to 60
- Print passed
- Else
- Print failed
- Halt

Pseudo-code Example 2

1. Pseudo-code the task of computing final price of an item after figuring in sales tax. Note these types of instruction: **start**, input(**get**), process/calculate(=), output(**display**) and **halt**.

- **START**
- **GET** price of an item
- **GET** sales tax rate
- sales tax = price of item times sales tax rate
- final price = price of item plus sales tax
- **DISPLAY** final price
- **Halt**

Pseudo-code Example 3

Gross pay depends on the pay rate and the number of hours worked per week. However, if you work more than 40 hours, you get paid time-and-a-half for all hours worked over 40.

Pseudo-code Example 2

1. **START**
2. **GET** hours worked
3. **GET** pay rate
4. **IF** hours worked 40 then
 - gross pay = pay rate times hours worked.
5. **Else**
 - gross pay = pay rate times 40 plus 1.5 times rate times (hours worked minus 40)
6. **Display** gross pay
7. **Halt**

Calculate Your Quizzes Average

1. START
2. GET number of quizzes
3. $\text{sum} = 0$
4. $\text{count} = 0$
5. WHILE count number of quizzes
 - GET quiz grade
 - $\text{sum} = \text{sum} + \text{quiz grade}$
 - $\text{count} = \text{count} + 1$
6. $\text{average} = \text{sum} / \text{number of quizzes}$
7. DISPLAY average
8. HALT

Variables: number of quizzes, sum, count, quiz grade, average

Categories of Algorithmic Operations

1. **Sequential Operations:** instructions are executed in order.
2. **Conditional(Question ask) Operations:** a control structure that ask a true/false question and then executes the next instruction based on the answer.
3. **Iterative operations(loops):** a control structure that repeats the execution of a block of instructions.

Rules for Pseudo-code

1. Write only one statement per line
2. Capitalize initial keywords
3. Indent to show hierarchy
4. Completeness

Contents

- Programming design
 - Steps in problem solving
- Flowchart
 - Flowchart Specifications
 - Advantages of Flowchart
 - Flowcharting Symbols
 - Flowcharting Rules
 - Flowcharting Limitations
 - Example of Algorithm and Flowchart

Programming Design

1. Algorithms for large scale processes are very complex.
2. They have to be divided into pieces.
3. People cannot normally understand the actual machine code that the computer needs to run a program.
4. You have to choose the right algorithm.
5. These algorithms can be designed through the use of flowcharts or **pseudo-code**.

Programming Design

- A typical programming task can be divided into two phases:
 1. Problem solving phase:
 - Produce an ordered sequence of steps that describe solution of problem.
 - This sequence of steps is called an algorithm.
 2. Implementation phase:
 - Implement the program in some programming language.

Steps in problem solving

- First produce a general algorithm(one can use pseudo code)
- Refine the algorithm successively to get step by step detailed algorithm that is very close to computer language.
- Pseudo code is an artificial and informal language that helps programmers develop algorithms. Pseudo code is very similar to everyday English.

Flowchart

Dictionary:

A schematic representation of a sequence of operations, as in a manufacturing process or computer program.

Technical:

A graphical representation of the sequence of operations in an information system or program. Information system flowcharts show how data flows from source documents through the computer to final distribution to users.

Flowchart Specifications

1. Flowcharting is a tool developed in the computer industry, for showing the steps involved in a process.
2. The flowchart is a means of visually presenting the flow of data through an information processing system.
3. Flowcharts facilitate communication between programmers and business people.
4. It is quite helpful in understanding the logic of complicated and lengthy problems.

Advantages of Flowcharts

1. **Communication:** Flowcharts are better way of communicating the logic of a system to all concerned.
2. **Effective analysis:** With the help of flowchart, problem can be analyzed in more effective way.
3. **Efficient Coding:** The flowcharts acts as a guide or blueprint during the systems analysis and program development phase.
4. **Proper Debugging:** The flowchart helps in debugging process.

Flowcharting Symbols

A flowchart is a diagram made up boxes, diamonds and other shapes, connected by arrows each shape represents a step in the process, and the arrow show the order in which they occur. It is used to show figuratively the operation of an algorithm.




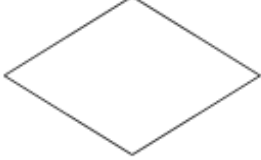


Flowcharting Symbols

There are 6 basic symbols commonly used in flowcharting of programs: Terminal, Process, Input/Output, Decision, Connector and Predefined Process. This is not a complete list of all the possible flowcharting symbols, it is the ones used most often.

Flowchart Rules

1. All boxes of the flowchart are connected with arrows. (Not lines)
2. Flowchart symbols have an entry point on the top of the symbol with no other entry points. The exit point for all flowchart symbols is on the bottom except for the Decision symbol.
3. The Decision symbol has two exit points; these can be on the sides or the bottom and one side.
4. All flowcharts start with a Terminal and all flowcharts end with a terminal or contentious loop.

Flowcharting Symbols

Name	Symbol	Use in Flowchart
Oval		Denotes the beginning or end of the program
Parallelogram		Denotes an input operation
Rectangle		Denotes a process to be carried out e.g. addition, subtraction, division etc.
Diamond		Denotes a decision(or branch) to be made. The program should continue along one of two routes (e.g. IF/THEN/ELSE)
Hybrid		Denotes an output operation
Flow line		Denotes the direction of logic flow in the program

Flowchart Limitations

1. **Complex logic:** Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy.
2. **Alterations and Modifications:** if alterations are required the flowchart may require drawing completely.
3. **Reproduction:** As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem.

Example 1 of Algorithm and Flowchart

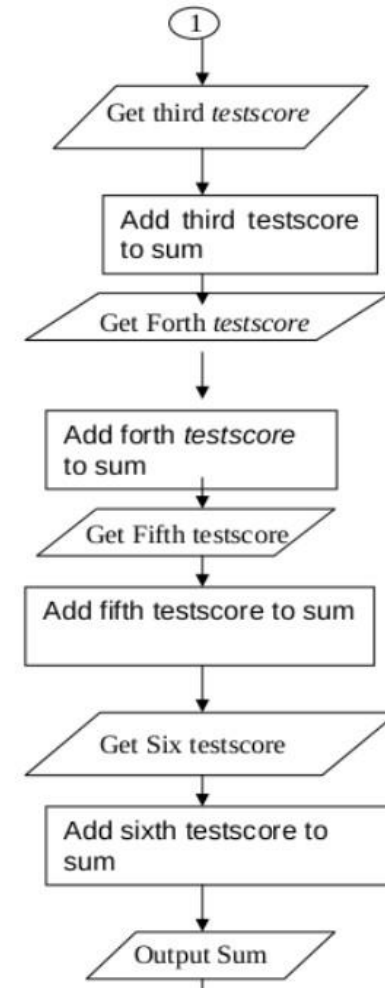
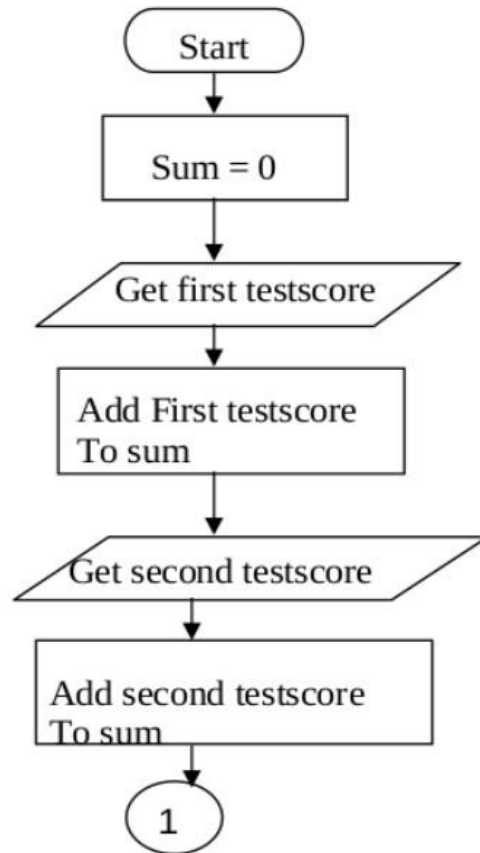
Design an algorithm and the corresponding flowchart for adding the test scores as given below:

26, 49, 98, 87, 62, 75

Algorithm for Example 1

1. START
2. $\text{sum} = 0$
3. GET the first test score
4. ADD first test score to sum
5. GET the second test score
6. ADD second test score to sum
7. GET the third test score
8. ADD third test score to sum
9. GET the forth test score
10. ADD forth test score to sum
11. GET the fifth test score
12. ADD fifth test score to sum
13. GET the sixth test score
14. ADD sixth test score to sum
15. DISPLAY sum
16. Halt

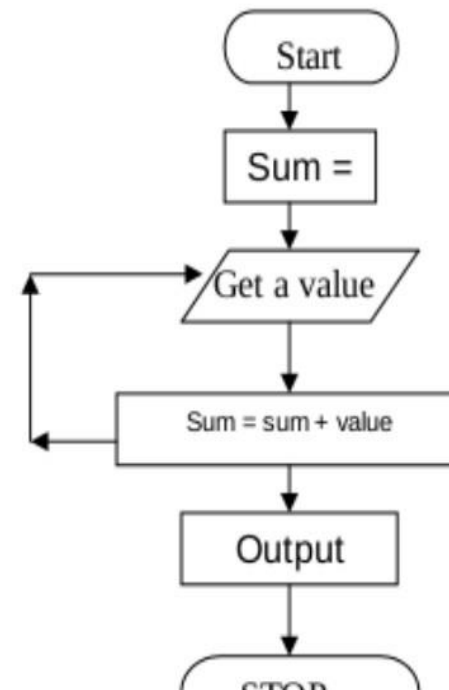
Flowcharting Symbols



Enhanced Flowchart for Example 1

The problem with this program is that, some of the steps appear more than once i.e. step 5 get second number, step 7, get third number, etc. One could shorten the algorithm or flowchart as below.

1. START
2. $\text{sum} = 0$
3. GET a value
4. $\text{sum} = \text{sum} + \text{value}$
5. GO TO step 3 to get next value
6. OUTPUT sum
7. STOP

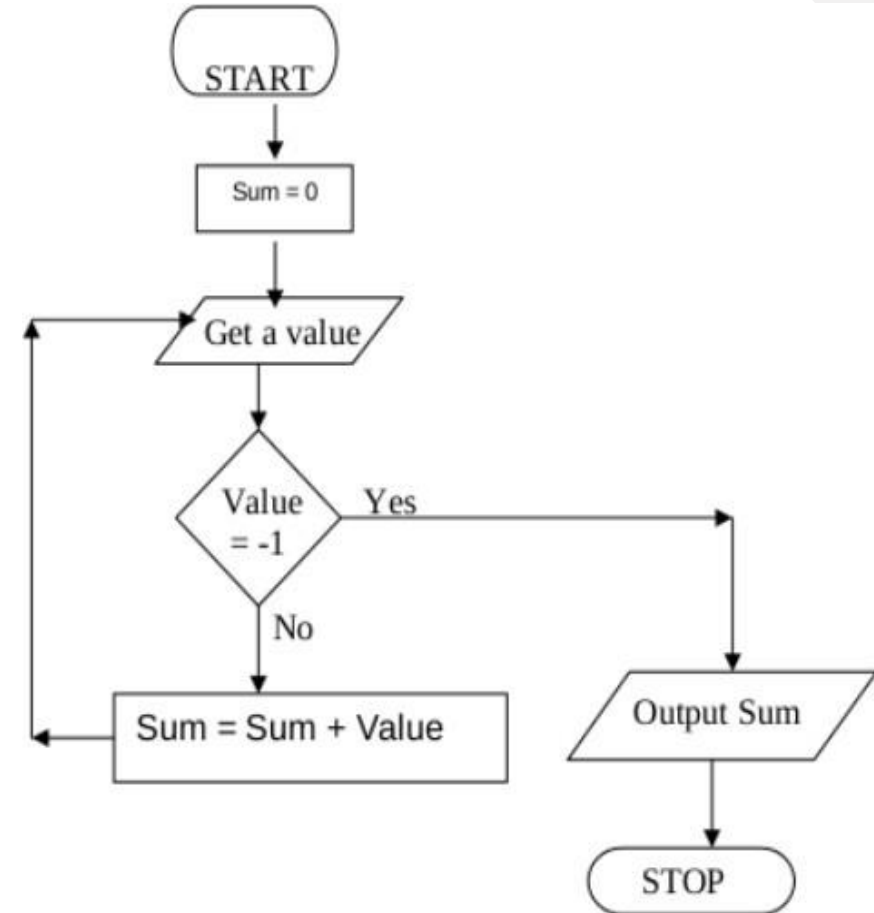


Your Turn!!!

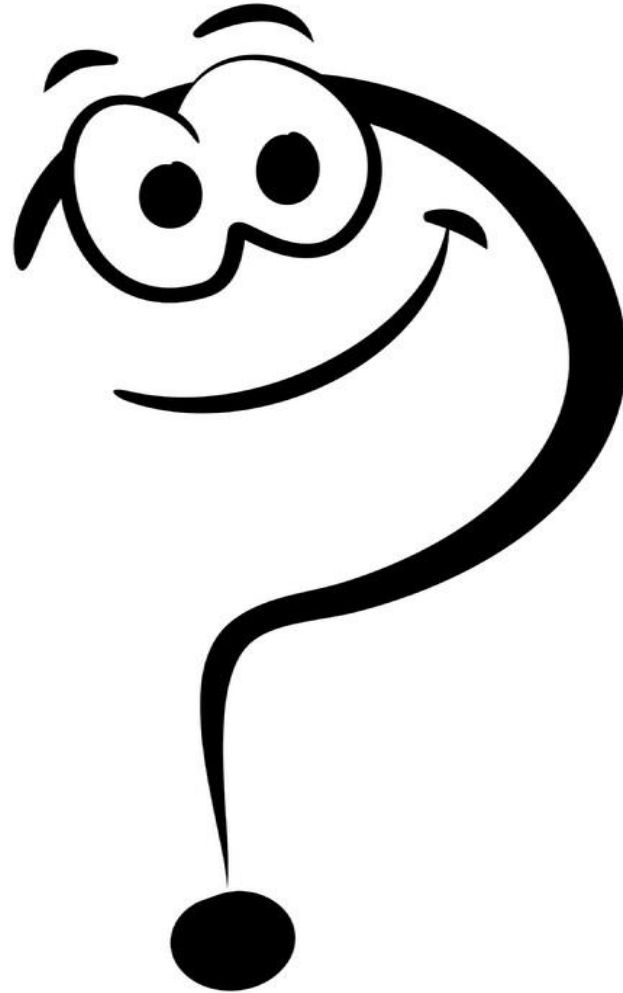
1. What is the problem for Flowchart and Algorithm in previous example?
2. How can you make it more efficient?
3. Do you know any other Algorithm for the solution?

Solution

1. START
2. Sum = 0
3. GET a value
4. IF the value is equal to -1, GO TO step 7
5. ADD to sum (sum = sum + value)
6. GO TO step 3 to get next value
7. OUTPUT the sum
8. STOP



Questions





THANK YOU



HODMAN.M2015@GMAIL.COM