ALGORITHM AND FLOW CHART

1.1INTRODUCTION

Intelligence is one of the key characteristics which differentiate a human being from other living creatures on the earth. Basic intelligence covers day to day problem solving and making strategies to handle different situations which keep arising in day to day life. One person goes Bank to withdraw money. After knowing the balance in his account, he/she decides to with draw the entire amount from his account but he/she has to leave minimum balance in his account. Here deciding about how much amount he/she may with draw from the account is one of the examples of the basic intelligence. During the process of solving any problem, one tries to find the necessary steps to be taken in a sequence. In this Unit you will develop your understanding about problem solving and approaches.

1.2 PROBLEM SOLVING

Can you think of a day in your life which goes without problem solving? Answer to this question is of course, No. In our life we are bound to solve problems. In our day to day activity such as purchasing something from a general store and making payments, depositing fee in school, or withdrawing money from bank account. All these activities involve some kind of problem solving. It can be said that whatever activity a human being or machine do for achieving a specified objective comes under problem solving. To make it clearer, let us see some other examples.

Example1: If you are watching a news channel on your TV and you want to change it to a sports channel, you need to do something i.e. move to that channel by pressing that channel number on your remote. This is a kind of problem solving.

Example 2: One Monday morning, a student is ready to go to school but yet he/she has not picked up those books and copies which are required as per timetable. So here picking up books and copies as per timetable is a kind of problem solving.

Example 3: If someone asks to you, what is time now? So seeing time in your watch and telling him is also a kind of problem solving.

Example 4: Some students in a class plan to go on picnic and decide to share the expenses among them. So calculating total expenses and the amount an individual have to give for picnic is also a kind of problem solving.

Now, broadly we can say that problem is a kind of barrier to achieve something and problem solving is a process to get that barrier removed by performing some sequence of activities

Here it is necessary to mention that all the problems in the world can not be solved. There are some problems which have no solution and these problems are called *Open Problems*.

If you can solve a given problem then you can also write an algorithm for it. In next section we will learn what is an *algorithm.*

1.3 ALGORITHM

Algorithm can be defined as: "A sequence of activities to be processed for getting desired output from a given input."

Webopedia defines an algorithm as: "A formula or set of steps for solving a particular problem. To be an algorithm, a set of rules must be unambiguous and have a clear stopping point". There may be more than one way to solve a problem, so there may be more than one algorithm for a problem.

Now, if we take definition of algorithm as: "A sequence of activities to be processed for getting desired output from a given input." Then we can say that:

1. Getting specified output is essential after algorithm is executed.
2. One will get output only if algorithm stops after finite time.
3. Activities in an algorithm to be clearly defined in other words for it to be unambiguous.

Before writing an algorithm for a problem, one should find out what is/are the inputs to the algorithm and what is/are expected output after running the algorithm. Now let us take some exercises to develop an algorithm for some simple problems: While writing algorithms we will use following symbol for different operations:

- **'+'** for Addition
- **'-'** for Subtraction
- **'*'** for Multiplication
- **'/'** for Division and
- ' ← ' for assignment. For example A ← X*3 means A will have a value of X*3.

1.3.1 Example of Algorithm

Problem 1: Find the area of a Circle of radius r.

Inputs to the algorithm:

Radius r of the Circle.

Expected output:

Area of the Circle

Algorithm:

Step1: Read\input the Radius r of the Circle
Step2: Area ⟵ PI*r*r // calculation of area
Step3: Print Area

Problem2: Write an algorithm to read two numbers and find their sum.

Inputs to the algorithm:

First num1.
Second num2.

Expected output:

Sum of the two numbers.

Algorithm:

Step1: Start
Step2: Read\input the first num1.
Step3: Read\input the second num2.
Step4: Sum ⟵ num1+num2 // calculation of sum
Step5: Print Sum
Step6: End

Problem 3: Convert temperature Fahrenheit to Celsius
Inputs to the algorithm:
Temperature in Fahrenheit
Expected output:
Temperature in Celsius

Algorithm:

Step1: Start
Step 2: Read Temperature in Fahrenheit F
Step 3: C ⟵ 5/9*(F32)
Step 4: Print Temperature in Celsius: C
Step5: End

Type of Algorithms

The algorithm and flowchart, classification to the three types of *control structures*. They are:

1. Sequence
2. Branching (Selection)
3. Loop (Repetition)

These three control structures are sufficient for all purposes. The sequence is exemplified by sequence of statements place one after the other – the one above or before another gets executed first. In flowcharts, sequence of statements is usually contained in the rectangular process box.

- The *branch* refers to a binary decision based on some condition. If the condition is true, one of the two branches is explored; if the condition is false, the other alternative is taken. This is usually represented by the 'if-then' construct in pseudo-codes and programs. In flowcharts, this is represented by the diamond-shaped decision box. This structure is also known as the *selection* structure.

Problem1: write algorithm to find the greater number between two numbers

Step1: Start

Step2: Read/input A and B

Step3: If A greater than B then C=A

Step4: if B greater than A then C=B

Step5: Print C

Step6: End

Problem2: write algorithm to find the result of equation: $f(x) = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases}$

Step1: Start

Step2: Read/input x

Step3: If X Less than zero then F=-X

Step4: if X greater than or equal zero then F=X

Step5: Print F

Step6: End

Problem3: A algorithm to find the largest value of any three numbers.

Step1: Start

Step2: Read/input A,B and C

Step3: If (A>=B) and (A>=C) then Max=A

Step4: If (B>=A) and (B>=C) then Max=B

Step5:If (C>=A) and (C>=B) then Max=C

Step6: Print Max

Step7: End

- The *loop* allows a statement or a sequence of statements to be repeatedly executed based on some loop condition. It is represented by the 'while' and 'for' constructs in most programming languages, for unbounded loops and bounded loops respectively. (Unbounded loops refer to those whose number of iterations depends on the eventuality that the termination condition is satisfied; bounded loops refer to those whose number of iterations is known before-hand.) In the flowcharts, a back arrow hints the presence of a loop. A trip around the loop is known as iteration. You must ensure that the condition for the termination of the looping must be satisfied after some finite number of iterations, otherwise it ends up as an infinite loop, a common mistake made by inexperienced programmers. The loop is also known as the *repetition* structure.

Examples:

Problem1: An algorithm to calculate even numbers between 0 and 99

1. Start
2. I ← 0
3. Write I in standard output
4. I ← I+2
5. If (I <=98) then go to line 3
6. End

Problem2: Design an algorithm which gets a natural value, n,as its input and calculates odd numbers equal or less than n. Then write them in the standard output:

1. Start
2. Read n
3. I ← 1
4. Write I
5. I ← I + 2
6. If (I <= n) then go to line 4
7. End

Problem3: Design an algorithm which generates even numbers between 1000 and 2000 and then prints them in the standard output. It should also print total sum:

1. Start
2. I ← 1000 and S ← 0
3. Write I
4. S ← S + I
5. I ← I + 2
6. If (I <= 2000) then go to line 3
else go to line 7
7. Write S
8. End

Problem4: Design an algorithm with a natural number, n, as its input which calculates the following formula and writes the result in the standard output:

S = ½ + ¼ + … +1/n
1. Start
2. Read n
3. I ← 2 and S ← 0
4. S= S + 1/I
5. I ← I + 2
6. If (I <= n) then go to line 4
   else write S in standard output
7. End

Combining the use of these control structures, for example, a loop within a loop (nested loops), a branch within another branch (nested if), a branch within a loop, a loop within a branch, and so forth, is not uncommon. Complex algorithms may have more complicated logic structure and deep

level of nesting, in which case it is best to demarcate parts of the algorithm as separate smaller *modules*. Beginners must train themselves to be proficient in using and combining control structures appropriately, and go through the trouble of tracing through the algorithm before they convert it into code.

1.3.2 Properties of algorithm

Donald Ervin Knuth has given a list of five properties for a,algorithm, these properties are:

1) Finiteness: An algorithm must always terminate after a finite number of steps. It means after every step one reach closer to solution of the problem and after a finite number of steps algorithm reaches to an end point.

2) Definiteness: Each step of an algorithm must be precisely defined. It is done by well thought actions to be performed at each step of the algorithm. Also the actions are defined unambiguously for each activity in the algorithm.

3) Input: Any operation you perform need some beginning value/quantities associated with different activities in the operation. So the value/quantities are given to the algorithm before it begins.

4) Output: One always expects output/result (expected value/quantities) in terms of output from an algorithm. The result may be obtained at different stages of the algorithm. If some result is from the intermediate stage of the operation then it is known as intermediate result and result obtained at the end of algorithm is known as end result. The output is expected value/quantities always have a specified relation to the inputs

5) Effectiveness: Algorithms to be developed/written using basic operations. Actually operations should be basic, so that even they can in principle be done exactly and in a finite amount of time by a person, by using paper and pencil only.

1.4 FLOWCHART

The flowchart is a diagram which visually presents the flow of data through processing systems. This means by seeing a flow chart one can know the operations performed and the sequence of these operations in a system. Algorithms are nothing but sequence of steps for solving problems. So a flow chart can be used for representing an algorithm. A flowchart, will describe the operations (and in what sequence) are required to solve a given problem. You can see a flow chart as a blueprint of a design you have made for solving a problem.

For example suppose you are going for a picnic with your friends then you plan for the activities you will do there. If you have a plan of activities then you know clearly when you will do what activity. Similarly when you have a problem to solve using computer or in other word you need to write a computer program for a problem then it will be good to draw a flowchart prior to writing a computer program. Flowchart is drawn according to defined rules.