Data Structure: Introduction to Algorithm Basics

-by DexA1948

Algorithm:

Algorithm is list of steps to carry out a task.

Algorithm 1

Algorithm for adding two numbers:

Step 1: Start

Step 2: Input two numbers, say A and B

Step 3: Calculate sum, Sum = A + B

Step 4: Print the Sum

Step 5: End

Algorithm 2

Algorithm for comparing two numbers:

Step 1: Start

Step 2: Input two numbers, say A and B

Step 3: If A>B, print A is greater than B

else if A<B, print A is less than B

else print A is equal to B

Step 4: End

How to write a Algorithm:

Algorithm can be written in various ways. There can be multiple ways of solving a same problem and reaching the same solution.

Things that should be considered in writing an algorithm is its clarity, unambiguity, independency and finiteness. Also an algorithm shall produce the desired output.

Algorithm may vary but it can never be personal. Algorithm can be used as a tool to break down and list the steps required in a problem solving. In a programming context where collaborative efforts are used for a problem solving algorithm also serves as a way to communicate with others. In other words algorithm must be unambiguous and independent of any programming language or individual.

Algorithm 3

Another Algorithm for comparing two numbers:

Step 1: Start

Step 2: Input two numbers, say A and B

Step 3: If A>B, go to step 4

else if A<B, go to step 5

else go to step 6

Step 4: print A is greater than B, go to step7

Step 5: print B is greater than A, go to step 7

Step 6: print A is equal to B, go to step 7

Step 7: End

Both Algorithm 2 and 3 solve same problem in varied ways and both will produce same desired output. Things that will differ between these algorithms is the performance. An algorithm may be more desirable if it successfully solves a problem with better performance measured in terms of memory and running time known as its complexities, space complexity for memory space and time complexity for run time performance. This complexities will start to play more roles as our algorithms becomes more advanced.

Algorithm 4

Algorithm for finding out whether a number is prime or composite:

Step 1: Start

Step 2: Input Number, say N

Step 3: Set D=1, C=0

Step 4: Divide N by D and find the remainder

Step 5: If remainder = 0, increase C by 1

Step 6: Increase D by 1, i.e; D = D + 1

Step 7: If D < = N go to Step 4

Step 8: If C>2, N is Composite else N is Prime

Step 9: End

Algorithm 5

Algorithm for finding out whether a number is prime or composite:

1. Start
2. Input Number, say N
3. D=2
4. Remainder = N mod D
5. If remainder = 0, N is Composite, exit
6. D = D + 1
7. If D > N/2, N is prime, exit
8. Go to Step 4
9. End

|  |  |
| --- | --- |
| Algorithm 4 | Algorithm 5 |
| 4 variables N, D, C and remainder | 3 variables N, D and remainder |
| Loops for N times from step 4 during each execution | Loops until a factor is found and half times of N if given number is prime |
| Consumes more memory | Consumes less memory |
| More run time | Less Run Time |