جامـــــعة المــــستـقـبـل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قــســــــــــم الانظمة الطبية الذكية

Lecture: (1)

Introduction to algorithm

Subject: Computer Programming (I)
Level: First
Lecturer: Dr. Maytham N. Meqdad

Algorithm

In mathematics and computer science, an algorithm is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation. Algorithms are used as specifications for performing calculations and data processing. More advanced algorithms can use conditionals to divert the code execution through various routes (referred to as automated decision-making) and deduce valid inferences (referred to as automated reasoning), achieving automation eventually. Using human characteristics as descriptors of machines in metaphorical ways was already practiced by Alan Turing with terms such as "memory", "search" and "stimulus".

In contrast, a heuristic is an approach to problem solving that may not be fully specified or may not guarantee correct or optimal results, especially in problem domains where there is no well-defined correct or optimal result.

As an effective method, an algorithm can be expressed within a finite amount of space and time and in a well-defined formal language for calculating a function. Starting from an initial state and initial input (perhaps empty), the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output" and terminating at a final ending state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate random input.

Programming Algorithm Defined

A programming algorithm is a procedure or formula used for solving a problem. It is based on conducting a sequence of specified actions in which these actions describe how to do something, and your computer will do it exactly that way every time. An algorithm works by following a procedure, made up of inputs. Once it has followed all the inputs, it will see a result, also known as output.

Characteristics of an algorithm:

1. Precision – the steps are precisely stated.
2. Uniqueness – results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
3. Finiteness – the algorithm stops after a finite number of instructions are executed.
4. Input – the algorithm receives input.
5. Output – the algorithm produces output.
6. Generality – the algorithm applies to a set of inputs.

There are seven different types of programming algorithms:

1. Sort algorithms
2. Search algorithms
3. Hashing
4. Dynamic Programming
5. Exponential by squaring

6. String matching and parsing
7. Primality testing algorithms

The advantages of programming algorithms include:

- A stepwise representation of a solution to a given problem, making it easy to understand.
- Uses a definite procedure.
- Not dependent on a particular programming language.
- Every step in an algorithm has its own logical sequence, making it easy to debug.

Algorithm Examples

Algorithm 1: Add two numbers entered by the user

```
Step 1: Start
Step 2: Declare variables num1, num2 and sum.
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2 and assign the result to sum.
        sum←num1+num2
Step 5: Display sum
Step 6: Stop
```

Algorithm 2: Find the largest number among three numbers

```
Step 1: Start
Step 2: Declare variables a,b and c.
Step 3: Read variables a,b and c.
Step 4: If a > b
           If a > c
              Display a is the largest number.
           Else
              Display c is the largest number.
        Else
           If b > c
              Display b is the largest number.
           Else
              Display c is the greatest number.
Step 5: Stop
```

Algorithm 3: Find Roots of a Quadratic Equation $ax^2 + bx + c = 0$

```
Step 1: Start
Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;
Step 3: Calculate discriminant
         D ← b2-4ac
```

```
Step 4: If D ≥ 0
              r1 ← (-b+√D)/2a
              r2 ← (-b-√D)/2a
              Display r1 and r2 as roots.
        Else
              Calculate real part and imaginary part
              rp ← -b/2a
              ip ← √(-D)/2a
              Display rp+j(ip) and rp-j(ip) as roots
Step 5: Stop
```

Algorithm 4: Find the factorial of a number

```
Step 1: Start
Step 2: Declare variables n, factorial and i.
Step 3: Initialize variables
          factorial ← 1
          i ← 1
Step 4: Read value of n
Step 5: Repeat the steps until i = n
     5.1: factorial ← factorial*i
     5.2: i ← i+1
Step 6: Display factorial
Step 7: Stop
```

Algorithm 5: Check whether a number is prime or not

```
Step 1: Start
Step 2: Declare variables n, i, flag.
Step 3: Initialize variables
        flag ← 1
        i ← 2
Step 4: Read n from the user.
Step 5: Repeat the steps until i=(n/2)
     5.1 If remainder of n÷i equals 0
            flag ← 0
            Go to step 6
     5.2 i ← i+1
Step 6: If flag = 0
           Display n is not prime
        else
           Display n is prime
Step 7: Stop
```

Algorithm 6: Find the Fibonacci series till the term less than 1000

```
Step 1: Start
Step 2: Declare variables first_term,second_term and temp.
Step 3: Initialize variables first_term ← 0 second_term ← 1
Step 4: Display first_term and second_term
Step 5: Repeat the steps until second_term ≤ 1000
     5.1: temp ← second_term
     5.2: second_term ← second_term + first_term
     5.3: first_term ← temp
     5.4: Display second_term
Step 6: Stop
```

Algorithm 7: Find Maximum of Three Numbers

```
1. Start
2. Prompt the user to enter the first number and store it in variable 'num1'
3. Prompt the user to enter the second number and store it in variable 'num2'
4. Prompt the user to enter the third number and store it in variable 'num3'
5. Convert 'num1', 'num2', and 'num3' to numeric values (if necessary)
6. Set 'max_number' to the maximum of 'num1', 'num2', and 'num3' using the
following logic:
   - If 'num1' is greater than or equal to 'num2' and 'num1' is greater than
or equal to 'num3', then 'max_number' is 'num1'.
   - If 'num2' is greater than or equal to 'num1' and 'num2' is greater than
or equal to 'num3', then 'max_number' is 'num2'.
   - Otherwise, 'max_number' is 'num3'.
7. Display 'max_number' as the maximum of the three numbers
8. End
```