Analysis of Algorithm

Muhammad Athar email id: athar@northern.edu.pk WhatsApp# 0333-5077664

(Week 02) Lectures 03 & 04

Objectives: Learning objectives of these lectures are

- Simple Algorithm Examples
  - Finding GCD
  - Adding numbers from 1 to N
    - With loop in N steps
    - With formula in 1 step
  - Finding sum of squares of N numbers
    - With loop in N steps
    - With formula in 1 step
  - Finding the Largest Value among N Values
- Pseudo Code & its Rules

Text Book & Resources:

1. Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, The MIT Press; 3rdEdition (2009). ISBN-10: 0262033844
2. Introduction to the Design and Analysis of Algorithms by Anany Levitin, Addison Wesley; 2ndEdition (2006). ISBN-10: 0321358287
3. Algorithms in C++ by Robert Sedgewick (1999). ASIN: B006UR4BJS
4. Algorithms in Java by Robert Sedgewick, Addison-Wesley Professional; 3rd Edition(2002). ISBN-10: 0201361205

Analysis of Algorithm

Muhammad Athar email id: athar@northern.edu.pk WhatsApp# 0333-5077664

Analysis of Algorithm

In the last week, I gave you a brief introduction about the algorithm & its characteristics. Why to need the algorithm and its basic areas which are design & analysis of algorithm. In this week, we will discuss the analysis of algorithm with some simple algorithm examples. What are pseudo code and its rules, advantage and disadvantages? At the end, we will also discuss the efficiency classes and their notations.

Simple Algorithms Examples

Finding Greatest Common Divisor (GCD)

Algorithm 1: (Euclid's Algorithm)

Input: Two Numbers m, n

Output: largest number that divides both m and n with remainder 0

Procedure (set of instruction):

Step 1 If n = 0, return the value of m as the answer and stop; otherwise, proceed to Step 2.

Step 2 Divide m by n and assign the value of the remainder to r.

Step 3 Assign the value of n to m and the value of r to n. Go to Step 1.

Example:

GCD(80, 24) will be computed as,
GCD(80, 24) = GCD(24, 8) = GCD(8, 0) = 8

GCD(96, 10) will be computed as,
GCD(96, 10) = GCD(10, 6) = GCD(6, 4) = GCD(4, 2) = GCD(2, 0) = 2

Algorithm 2:

Input: Two Numbers n, m
Output: largest number that divides both m and n with remainder 0
Procedure (set of instruction):

Step 1 m%n=0 then return n otherwise goto step 2

Step 2 assign a value of n to d

Step 3 made decrement to d

Step 4 divide m and n by d; if remainder of both is zero return d otherwise goto step 3

Example:

Muhammad Athar email id: athar@northern.edu.pk WhatsApp# 0333-5077664

GCD(80, 24) will be computed as,

- 80 will be divided by 24, remainder will be not zero so 24 will be assigned to d and made decrement d, that will become 23.
- Now 80 and 24 will be divided by 23, again remainder will not zero. Now d will become 22.
- This will continue until d becomes 8 and at this step remainder will be zero hence 8 will be gcd.

Let us made the analysis of both algorithms for GCD, second algorithm will require more operations as compared to first one.

Adding numbers from 1 to N

Algorithm 1:

Input: n

Output: Sum of n numbers

Procedure (set of instruction):

Step 1 Assign zero to sum and 1 to i

Step 2 add i to sum

Step 3 made increment to i

Step 4 repeat steps 2 and 3 n times

No remind mathematical formula as given below

$$\sum_{i=1}^{n} i = 1+2+\cdots+n = \frac{n(n+1)}{2}$$

Example:

Let n=50

Then $\frac{50(50+1)}{2} = 1275$

On the base of this formula we can write algorithm as follow

Algorithm 2:

Input: n

Output: Sum

Procedure (set of instruction):

Step 1 sum=n*(n+1)/2

Algorithm 1 will demand n operations to compute sum while algorithm 2 will do this all in one step.

Adding sum of squares of first N numbers

Algorithm 1:

Muhammad Athar email id: athar@northern.edu.pk WhatsApp# 0333-5077664

Input: n

Output: Sum

Procedure (set of instruction):

Step 1 Assign zero to sum and 1 to i

Step 2 add i*i to sum

Step 3 made increment to i

Step 4 repeat steps 2 and 3 n times

No remind mathematical formula as given below

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Example:

Let n =10 $\frac{10(10+1)(2(10)+1)}{6} = 385$

On the base of this formula we can write algorithm as follow

Algorithm 2:

Input: n

Output: Sum

Procedure (set of instruction):

Step 1 sum=(n*(n+1)*(2*n+1))/6

Algorithm 1 will demand n operations to compute sum of squares of numbers while algorithm 2 will do this all in one step.

Find the value of the largest element in a list of n numbers.

```
MaxElement(A[0..n-1)
    maxVal = A[0];
    for(I = 1; I < n; I++)
        if(A[I] > maxVal)
            maxVal = A[I];
    return maxVal
```

What is Pseudo Code?

Pseudo code is an artificial and informal language that helps programmers to develop algorithms. Pseudo code is a "text-based" detail (algorithmic) design tool. The rules of Pseudo code are

Muhammad Athar email id: athar@northern.edu.pk WhatsApp# 0333-5077664

reasonably straightforward. All statements showing "dependency" are to be indented. These include while, do, for, if, switch. Examples below will illustrate this notion.

Finding Greatest Common Divisor (GCD)

Pseudo Code: (Euclid's Algorithm)
Input: Two nonnegative, not-both-zero integers m and n
Output: Greatest common divisor of m and n
Procedure:

```
while n ≠ 0 do
        r ← m mod n
        m ← n
        n ← r
return m
```

Rules for Pseudo Code

There are some rules that should be followed to write a code

1. Write only one statement per line

Each statement in your pseudo code should express just one action for the computer. If the task list is properly drawn, then in most cases each task will correspond to one line of pseudo code.

EXAMPLE: TASK LIST:

```
Read name, hourly rate, hours worked, deduction rate
Perform calculations
grossPay = hourlyRate * hoursWorked
deduction = grossPay * deductionRate
netpay = grossPay – deduction
Write name, grossPay, deduction, netpay
```

PSEUDO CODE:

```
READ name, hourlyRate, hoursWorked, deductionRate
grossPay = hourlyRate * hoursWorked
deduction = grossPay * deductionRate
netPay = grossPay – deduction
WRITE name, grossPay, deduction, netPay
```

2. Capitalize initial keyword

In the example above, READ and WRITE are in caps. There are just a few keywords we will use:

READ, WRITE, IF, ELSE, ENDIF, WHILE, ENDWHILE, REPEAT, UNTIL

Muhammad Athar email id: athar@northern.edu.pk WhatsApp# 0333-5077664

3. Indent to show hierarchy

We will use a particular indentation pattern in each of the design structures:

SEQUENCE: keep statements that are "stacked" in sequence all starting in the same column.

SELECTION: indent the statements that fall inside the selection structure, but not the keywords that form the selection

```
IF amount < 1000
        interestRate = 0.06 // the "yes" or "true" action
ELSE
        interestRate = 0.10 // the "no" or "false" action
ENDIF
```

EXAMPLE: In the example above, employees whose grossPay is less than 100 do not have any deduction.

TASK LIST:

```
Read name, hourly rate, hours worked, deduction rate
Compute gross, deduction, net pay
Is gross >= 100?
YES: calculate deduction
NO: no deduction
```

Write name, gross, deduction, net pay

PSEUDO CODE:

```
READ name, hourlyRate, hoursWorked
grossPay = hourlyRate * hoursWorked
IF grossPay >= 100
        deduction = grossPay * deductionRate
ELSE
        deduction = 0
END IF
netPay = grossPay – deduction
WRITE name, grossPay, deduction, netPay
```

Nesting IF

```
READ gameNumber
IF gameNumber = 1
        DO ABC
ELSE
        IF gameNumber = 2
                DO CDE
        ELSE
                DO DEF
        END IF
```

Muhammad Athar email id: athar@northern.edu.pk WhatsApp# 0333-5077664

```
    END IF
```

LOOPING: indent the statements that fall inside the loop, but not the keywords that form the loop

```
    count = 0
    WHILE count < 10
        ADD 1 to count
        WRITE count
    END WHILE
    WRITE "The end"
```

4. End multiline structures

See how the IF/ELSE/ENDIF is constructed above. The ENDIF (or END whatever) always is in line with the IF (or whatever starts the structure).

5. Keep statement language independent

Resist the urge to write in whatever language you are most comfortable with. In the long run, you will save time! There may be special features available in the language you plan to eventually write the program in; if you are SURE it will be written in that language, then you can use the features. If not, then avoid using the special features.

Pseudo Code Disadvantages

- It's not visual
- There is no accepted standard, so it varies widely from company to company

Pseudo Code Advantages

- Can be done easily on a word processor
- Easily modified
- Implements structured concepts well

Efficiency Classes for Algorithms

Each algorithm has different efficiency in term of time and space. There are infinite problems in the word and almost each of them has an algorithm to be solved. It is impossible to memorizing the efficiency (execution time) of each algorithm. Existing algorithms have some trend/behavior in the context of efficiency. On the base of those trends; there are standard classes of algorithm's efficiency are defined. In the future we will compute the execution time for any algorithm and decide the efficiency class among the given classes on the base of trend for the specific algorithm. Defined efficiency classes are given below.

Muhammad Athar email id: athar@northern.edu.pk WhatsApp# 0333-5077664

- Constant ≈ ***1***
- Logarithmic ≈ log ***n***
- Linear ≈ ***n***
- Log Linear ≈ ***n*** log ***n***
- Quadratic ≈ $\boldsymbol{n}^2$
- Cubic ≈ $\boldsymbol{n}^3$
- Exponential ≈ $\mathbf{2}^n$

In the above classes; n is actually is data size that will be given to a particular algorithm.

Example:

Adding numbers from 1 to N

Algorithm 1:

Input: n

Output: Sum

Procedure (set of instruction):

Step 1 Assign zero to sum and 1 to i

Step 2 add i to sum

Step 3 made increment to i

Step 4 repeat steps 2 and 3 n times

No remind mathematical formula as given below

$$\sum_{i=1}^{n} i = 1+2+\cdots+n = \frac{n(n+1)}{2}$$

On the base of this formula we can write algorithm as follow

Algorithm 2:

Input: n

Output: Sum

Procedure (set of instruction):

Step 1 sum=n*(n+1)/2

Algorithm 1 will demand n operations to compute sum while algorithm 2 will do this all in one step.

Muhammad Athar email id: athar@northern.edu.pk WhatsApp# 0333-5077664

Here, execution time for algorithm 1 will be increase with increasing n while execution time for algorithm 2 will remain constant whatever the size of the data will be. In this algorithm the efficiency class for algorithm 1 will be linear and for algorithm 2 will be constant. Following table and graph show the difference between the efficiency classes.

	n=1	n=2	n=4	n=8	n=16	n=32
1	1	1	1	1	1	1
logn	0	1	2	3	4	5
n	1	2	4	8	16	32
nlogn	0	2	8	24	64	160
n^2	1	4	16	64	256	1024
n^3	1	8	64	512	4096	32768
2^n	2	4	16	256	65536	4294967296