

Analysis of Algorithm

Instructor: Yonas Birku
Bule Hora University

Chapter one

Introduction and Elementary Data Structure

- ≡ A data structure is a way of storing data in a computer so that it can be used efficiently.
- ≡ It is the specification of the elements of the structure, the relationships between them, and the operation that may be performed upon them.
- ≡ Consider as an example books in a library. It can be stored in a shelf arbitrarily or using some defined orders such as sorted by Title, Author and so on.
- ≡ It can be said that the study of data structures involves the storage, retrieval and manipulation of information.
- ≡ In other words, the possible logical and physical arrangement of data items to provide an efficient solution for a problem is called Data Structure.

...cont

≡ A program is written in order to solve a problem. A solution to a problem actually consists of two things:

- A way to organize the data
- Sequence of steps to solve the problem

≡ The way data are organized in a computers memory is said to be Data Structure and

≡ the sequence of computational steps to solve a problem is said to be an algorithm.

Therefore,

Program=Algorithm + Data structure

Operations on data Structures

≡ The various operations that can be performed over Data Structures are

- Create
- Destroy
- Access
- Update

Classification of Data structure

- ☰ **Data structures are broadly divided in two:-**
 - ❖ **Primitive data structures.**
 - ❖ **Non primitive data structures.**

Primitive Data Structures

- ≡ These are the basic data structure and are directly operated up on the machine instruction, which is in primitive level. They are integers, floating point numbers, characters, string constants, pointers etc.

Non-Primitive Data Structure

- ≡ It is more sophisticated data structure emphasizing on structuring of a group of homogenous (same type) or heterogeneous (different type) data items.
- ≡ For example : Array, list, files, linked list, trees and graph .

Algorithm Analysis

-  **An algorithm is a well-defined computational procedure that takes some value or a set of values as input and produces some value or a set of values as output.**
-  **Algorithm is a step by step procedure to solve a problem. E.g. industrial activities, Student Registration, etc, all need an algorithm to follow.**
-  **More than one algorithm is possible for the same task.**

Properties of algorithms

- ≡ **Finiteness:** any algorithm should have finite number of steps to be followed.
- ≡ **Language Independence:** it must not depend on any one programming language.
- ≡ **Sequential:** the first step (start step) and last step (halt step) must be clearly noted.
- ≡ **Correctness:** it must compute correct answer all possible legal inputs.
- ≡ **Completeness:** it must solve the problem completely.
- ≡ **Input/output:** zero or more input, one or more output.
- ≡ **And so on.**

Algorithm Analysis Concepts

- ≡ Complexity analysis is concerned with determining the efficiency of algorithms.
- ≡ What is Efficiency depends on?
 - Execution speed (most important)
 - Amount of memory used
- ≡ Efficiency differences may not be noticeable for small data, but become very important for large amounts of data

Algorithm's Performance

- Two important ways to characterize the effectiveness of an algorithm are its space complexity and time complexity.
 - Time complexity of an algorithm concerns determining an expression of the number of steps needed as a function of the problem size.
 - The space complexity of an algorithm is the amount of memory it needs to run to completion.
 - It will be difficult for us to determine the run time of a program exactly. Because the run time may depend on
 - Processor speed
 - Current processor load
 - Input size of the given algorithm
 - Available memory, etc
- So, it will be difficult for us to obtain exact running time of the program.

How do we estimate the complexity of algorithms?

- ≡ 1. Algorithm analysis - Find $f(n)$ - helps to determine the complexity of an algorithm
- ≡ 2. Order of Magnitude - $g(n)$ belongs to $O(f(n))$ - helps to determine the category of the complexity to which it belongs.

Analysis Rule:

- 1. Assume an arbitrary time unit
- 2. Execution of one of the following operations takes time unit 1
 - Assignment statement
 - Single I/O statement;
 - Single Boolean statement.
 - Single arithmetic.
 - Function return.
- 3. selection statement
 - Time for condition evaluation + the maximum time of its clauses
- 4. Loop statement
 - $\Sigma(\text{no of iteration of Body}) + 1 + n + 1 + n$ (initialization time + checking + update)
- 5. For function call
 - $1 + \text{time}(\text{parameters}) + \text{body time}$

Estimation of Complexity of Algorithm

Examples:

```
1. int count(){  
    int k=0;  
    cout<< "Enter an integer";  
    cin>>n;  
    for (i=0;i<n;i++)  
        k=k+2;  
    return 0;}
```

Time Units to Compute

- 1 for the assignment statement: `int k=0`
- 1 for the output statement.
- 1 for the input statement.

In the for loop:

- 1 assignment, $n+1$ tests, and n increments.
- n loops of 2 units for an assignment, and an addition.
- 1 for the return statement.

$$T(n) = 1 + 1 + 1 + (1 + n + 1 + n) + 2n + 1 = 4n + 6 = O(n)$$

...

- Calculate $T(n)$ for the following

- 1. $k=0$;

$\text{Cout} << \text{"enter an integer"};$

$\text{Cin} >> n;$

For ($i=0; i < n; i++$)

$K++$

- $T(n) = 1 + 1 + 1 + (1 + n + 1 + n + n)$
 $= 5 + 3n$

- **2. $i=0$;**
 While ($i < n$)
 {
 $x++$;
 $i++$;
 }
 $J=1$;
 While($j \leq 10$)
 {
 $x++$;
 $j++$
 }
- **$T(n) = 1 + n + 1 + n + n + 1 + 11 + 10 + 10$**
 $= 3n + 34$

...

3. **for(i=1;i<=n;i++)**
 for(j=1;j<=n; j++)
 k++;
T(n)=1+n+1+n+n(1+n+1+n+n)=3n²+4n+2

4. *Sum=0;*

if(test==1)

{

for (i=1;i<=n;i++)

sum=sum+i

}

else

{

cout<<sum;

}

• **$T(n) = 1 + 1 + \text{Max}(1 + n + 1 + n + 2n, 1) = 4n + 4$**

Finding Asymptotic Complexity: Examples

≡ Rules to find Big-O from a given $T(n)$

- Take highest order
- Ignore the coefficients

Finding Big O of given Algorithm

≡ 1. for (i=1;i<=n;i++)

● Cout<<i;

● $T(n) = 1+n+1+n+n$

$$= 3n+2$$

● $T(n) = O(n)$