Algorithm analysis & design

Algorithm Analysis

Presented By:

T.A. **Asmaa Hamad El-saied**

E-mail: **eng.asmaa134@gmail.com**

Agenda

- **Algorithm Analysis**
  - Running Time
  - Example
- **Guiding Principles**
- **Running Time vs. Order**
- **How to calculate Order?**
- **Examples**

Algorithm Analysis

- The main goal is to determine the cost of running an algorithm and how to reduce that cost. Cost is expressed as Complexity

- Time Complexity
- Space Complexity

Space & Time Complexities

- Space complexity
  - How much space is required
- Time complexity
  - How much time does it take to run the algorithm

- Often, we deal with estimates!

Time Complexity

- **Running Time** : # steps that the algorithm takes to solve the problem.
  - # steps expressed as a function of input size $T(n)$.

- (n) is the Problem Size:

  n could be the number of specific operations, or the size of data (e.g. an array) or both.

Time Complexity

- Number of operations (# steps)
  - Each "simple" operation (+, -, =, <, >=) is one operation.
  - Loops and function calls are not simple operations, but depend upon the size of the data and the contents of a function.
  - Each memory access is one operation.

We measure T(n) of an algorithm by counting the number of steps (operations).

Algorithm Analysis: Example

- Design an Algorithm that get the max element from an array, and then analyze this algorithm.

Init 1
Increment n
Cond (comp) n+1

- max ← -∞ 1
- For i ← 0 to n
  - If A[i] > max 2
    - max ← A[i] 2
  - End if
- End for
- Return max 1

4n

- Running Time = T(n) = 6n + 4.

Guiding Principles

- Worst Case
- Asymptotic analysis (Large input Size)
- Constants are not important
  - What is order?
  - Running time vs Order.

Running Time vs. Order

- **Order :**

  is the dominant factor in running time without constants.

- **Example :**
  - $T(n) = 6n + 4 \longrightarrow O(n)$
  - $T(n) = 10\ n^2 + 7n + 10^6 \longrightarrow O(n^2)$

- **Example:**
  - $Algo_1 \longrightarrow T(n) = 10^6 \log n \longrightarrow O(\log n)$
  - $Algo_2 \longrightarrow T(n) = 20\ n \longrightarrow O(n)$

How to calculate Order?

- **Analysis**
  1. Statements
  2. Conditions
     - If-statement
     - Switch case
     - Nested if
  3. Loops
  4. Recursion

How to calculate Order?

- **Analysis**

1. Statements → O(1)

   Exception:-

   M=GetMax(A,N) → O(Fn)

2. Conditions
   - If-statement
     - For condition → O(1)
     - For body → O(body)
     - O(IF)=O(1)+O(Body)= O(Body)

How to calculate Order?

- **Analysis**

  2. Conditions

     If Condition then

       B1

     Else if condition Then

       B2

       ……

     Else

       Bk

- Order → O(Max(B1,B2…..,BK))

How to calculate Order?

- **Analysis**
  1. Loops

     Loop

         body

     End loop

Order=O(body) x # iteration

How to calculate Order?

- **Example 1:-**

O(n)

O(1)

- max ← -∞ O(1)
- for i ← 1 to n
  - If A[i] > max O(1)
    - max ← A[i] O(1)

  End if
- End for
- Return max O(1)

- **O(n)**

How to calculate Order?

- **Example 2:**

```
Switch (choice)
    Case 1:
        print N                         B1= O(1)
    Case 2:
        Max:= GetMax(A,N)               B2= O(N)
        Write Max
    Case 3:
        Sum:=0                          O(1)
        For I:=1 to N
            If sum< GetMax(A,N)         O(N)
O(N) X N        O(N)    Sum+=A[I]       O(1)
B3=O(N^2)
            End If
        End for
        Print sum                       O(1)
End switch
```

- **O(N^2)**

How to calculate Order?

- **Example 3:**

O(1) X #Iteration
O(1) X Log N= O(log N)
	While N>1
	O(1)**N=N/10** **// int division**
	End While

Step no	0	1	2	3	....	K-1	K
Value of N	N	$\frac{N}{10}$	$\frac{N}{10^2}$	$\frac{N}{10^3}$	....	$\frac{N}{10^{K-1}}$	$\frac{N}{10^K}$

- **Termination occur:**
  - $\frac{N}{10^K} = \mathbf{1}$ → 10^K=N
  - $log_{10} 10^K = log_{10} N$ → $K\ log_{10} 10 = log_{10} N$
  - **K=**$log_{10} N$
- **O(log N)**

Thanks