Algorithms

Formally :

A tool for solving a well-specified computational problem.

Algorithms: Is a finite sequence of unambiguous instructions for solving a well-specified computational problem.

Important Features :

1-Finiteness 2- Definiteness 3- input

4- Output 5- Effectiveness

Turing Machine : used to measure the efficiency of the algorithm.

Algorithm analysis :

- Determining performance characteristics
  - Time
  - Memory
  - Communication bandwidth
- Why analyze Algorithms ?
  - Choose the most efficient of several possible Algorithm for the same problem.
  - Choose the Algorithm with best running time

Running Time :

Run time expression should be machine independent

Use a model of computation or “hypothetical” computer

- Model should be :
  - Simple
  - Applicable

RAM Model :

- Generic single processor model
- Supports simple constant-time instructions
- Run time (cost) is uniform (I time unit)
- Memory is unlimited
- Flat memory model → no hierarchy
- Access to a word of memory takes I time unit

Running time : is the number of steps executed by the Algorithm on that input often referred to as the Complexity.

Complexity of an Algorithm depends on :

- Size of input
- Other characteristics of the input data
  - Data is Sorted
  - Data cycles in the graph

Complexity

Worst case complexity :

Maximum numbers of steps the Algorithm takes for any possible input.

Average case complexity :

Average of the running times of all possible input.

Best case complexity :

Minimum numbers of steps the Algorithm takes for any possible input.

Pseudo code conventions :

- Indentation (for block structure)
- Value of loop counter variable upon loop termination
- Call by value not reference
- Local variable
- Error handling are omitted

LinearSearch(A, *key*)	*cost*	*times*
$i \leftarrow 1$	c_1	1
while $i \leq n$ **and** A[i] != *key*	c_2	x
do i++	c_3	x-1
if $i \leq n$	c_4	1
then return *true*	c_5	1
else return *false*	c_6	1

x ranges between 1 and n+1.
So, the running time ranges between

$c_1 + c_2 + c_4 + c_5$ – **best case**

and

$c_1 + c_2(n+1) + c_3 n + c_4 + c_6$ – **worst case**

Worst Case :

1+n+1+n+1+1 = 2n + 4

O(n)

Average case :

1 +n/2 + n/2 +1+1 = n+3

Θ(n)

Best Case :

1+1+1+1 = 4

Ω(1)

Complexity of an algorithm : is denoted by the highest order term in the expression for running time

Order of growth : how running time grows with input size

Asymptotic complexity : The running time for large inputs