Kinds of asymptotic analyses

**Worst-case**: (usually)

T(n) = maximum time of algorithm on any input of size n.

• Default case

Average-case: (sometimes)

• T(n) = expected time of algorithm over all inputs of size n.

• Need assumption of statistical distribution of inputs.

**Best-case:** (NEVER)

• Case when an algorithm works fast on some input

•Special case

•Don’t rely on best case

If T(n)= O(1), implies algorithm runs in constant time irrespective of input size, n

If T(n)= O(n), implies performance of algorithm grows linearly and is directly proportional to

input size, n

It is a comparative analysis. Lets A and B are two algorithms.

1.A= Ο(Β), A takes less time then B

2.A= Θ(Β), A and B take similar time

3.A= Ω(Β), A takes more time then B

**Worst case analysis O-notation**

f(n)= Ο(g(n)) if and only if there are positive

constants c and n0 such that f(n)≤ c g(n) for n

**Θ-notation** f(n) is the set of functions with the same order of growth as g(n) f(n)= Θ(g(n)), implies there are positive constants c1,c2 and n0 such that c1g(n)<= f(n)≤ c2 g(n) for n ≥ n0

**Best case Analysis** Ω - notation f(n)= Ω(g(n)) if and only if there are positive constants c and n0 such that c g(n) ≤ f(n) for all n ≥ n0

If f(n)= 2n and g(n)=n3 Then which one is true F(n)= O(g(n)) Or g(n)=O(f(n))

Comparison of growth of various functions

• O(1)< O(log logn)< O(logn)< O(n)< O(n logn)<O(n2)< O(n3)< O(2n)< O(n!)< O(n^n)

**Computing Time complexity of non-recursive Algorithm**

**Assumption:**

Time required for each instruction is constant and is different for different instruction.

Time complexity of a non-recursive program or algorithm is nothing but frequency of the

statement which is executing for the maximum time.

For non-recursive AlgorithmFind the frequency count of each statement and total time is sum of product of time of each instruction and its frequency. We should adapt RAM model.

**Time Complexity of recursive Algorithm**

Computing time complexity of a recursive Algorithm-

• Find the suitable recurrence relation for the algorithm

• Solve the recurrence using methods likeiteration, substitution, recursion-tree, master theorem.

• Solution of the recurrence relation will be the time complexity of the algorithm