**CHAPTER 03**

Q1. Let f(n) and g(n) be asymptotically nonnegative functions. Using the basic definition of Θ-notation, prove that ***max(f(n),g(n)) = Θ(f(n) + g(n)).***

Ans. By definition of Θ.

We have to show that there exists constants c1 , c2 , n0 > 0 such that

0 ≤ c1 g(n) ≤ f(n) ≤ c2 g(n)

i.e. ***0 < c1 (f(n) + g(n)) ≤ max(f(n),g(n)) ≤ c2 (f(n) + g(n)) for all n> n0***

As the functions are asymptotically non-negative, we can assume that for some n0 > 0 , f(n) > 0 and g(n)> 0.

Let n0 = max(n1,n2) and for n > n0

(f(n) + g(n)) ≥ max(f(n),g(n)) ---(1)

f(n) ≤ max(f(n), g(n)) --(2)

g(n) ≤ max(f(n) , g(n)) -- (3)

From 2 and 3

f(n) + g(n) ≤ 2 max(f(n),g(n))

(f(n) + g(n))/2 ≤ max(f(n),g(n)) ---(3)

From 1 and 3 we get c1 = 0.5 and c2 = 1

Proof:-

0 ≤ c1 g(n) ≤ f(n) ≤ c2 g(n)

Substituting c1  and c2  problem eqn. we get

all n> n0

Since we can find c1 c2  such that Θ-definition holds.

max(f(n) , g(n)) = Θ(f(n) + g(n))

Q2. Show that for any real constant a and b , where b> 0,

Ans. To prove this we need to find c1 , c2 , n0 > 0 s.t.

---(1)

Now

n +a ≤ n + |a|

n +a ≤ 2n, when |a| ≤ n

and

n + a ≤ n - |a| ---(2)

n + a ≤ n/2 , when |a| ≤ n/2 --(3)

Therefore, from 2 and 3 we get

Raising them to the power of b we get

----(4)

Comparing 1 and 4 we get

c1 = 1/2b , c2 = 2b and n0= 2|a|

Q3. Explain why the statement, “The running time of algorithm A is at least O(n2) is meaningless.

Ans.

Given statement :- T(n) is at least Θ(n­2)

Since from the statement we get either the upper bound or lower bound or both.

Let us see what this statement provide us

a. Upper bound :- Since T(n) ≥ Θ(n2) . It doesn’t provide us any information about the upper bound.  
b. Lower bound : - Let us assume that f(n) = Θ(n2), then the statement becomes: T(n) ≥ f(n) , but f(n) can ne any function that is smaller n2 (eg a constant , n …) . Hence it doesn’t give any idea about lower bound too.

Other way to understand

Lower bound without an upper bound isn’t much helpful in real analysis. Since the algorithm doesn’t tell us whether it finishes early or universe does.

As we know O when used properly give us upper and using O as an indicative of lower bound is clearly an abuse of O notation.

Though statement is not incorrect if we see but what matter for us is , that this statement is clearly an abuse of O-notation. This is should be represented using Ω.

See usually O(n2) tell us that the run time will be n2 or faster. But as per statement if it is at least O(n2) that means the algorithm will be O(n2) or slower. Hence it can be anything. (exp. too).

To get an intuition. You can think of it as saying. ***“My age is at least zero”.*** This is not a wrong statement but it is less informative. As you can 0 or 200 or 2000.

Q4. Is 2n+1 = O(2n)? Is 22n = O(2n) ?

Ans.

a. True . Since 2n+1 = 2 x 2n . So for n0 = 0 , we can choose c ≥ 2 , such that

for n > n0 . By definition of O-notation 2n+1 = O(2n).

b. False :- Since 22n = 2n x 2n = 4n. But we can’t find any c such that for n ≥ n0/

Other way

Let us assume that there exist c and n ≥ n0 such that

⇒ \* ≤ c \* 2n

⇒ ≤ c

Which is not true since ***c***  is constant.

Q5. Prove theorem 3.1

Ans.

***Statement :-***  For any two functions f (n) and g(n), we have f(n) = Θ(g(n)) if and only if

f (n) = O(g(n)) and f (n) = Ω(g(n)).

***Proof:-***  To prove this statement we need to show the logic both ways.

* If f(n) ∈ Θ(n) , then for n ≥ n0. As for n ≥ n0, f(n) ∈ O(g(n)). As for n ≥ n0, f(n) = (g(n)).
* If f(n) = , then and if f(n) = O(g(n)), then

For n ≥ n2.

Combining both equations we get for n > max(n1 , n2 ). i.e. f(n) = Θ(g(n)).

Q6. Prove that the running time of an algorithm is Θ(g(n)) if and only if its worst-case running time is O(g(n)) and its best-case running time is (g(n)).

Ans. If *T*w​ is the worst-case running time and *T*b​ is the best-case running time, we know that

for n > nb

for n > n w

Combining them we get

for n > max(nb , n­w).

Since the running time is bound between Tb and Tw​ and the above is the definition of the Θ-notation, ***proved.***

Q7. Prove that o(g(n)) ⋂ ω(g(n)) is the empty set.

Ans. Let us assume that f(n) ∈ o(g(n)) ⋂ ω(g(n)). Then we have

[ By definition]

Which is a contradiction.

Q8. We can extend our notation to the case of two parameters n and m that can go to infinity independently at different rates. For a given function g(n,m) , we denote by O(g(n,m)) the set of functions

O(g(n,m)) = { f(n,m) : there exist positive constants c,n0, and m­­0 such that 0 ≤ f(n,m) ≤ cg(n,m) for all n ≥ n0 or m ≥ m0 }

Give corresponding definitions for Ω(g(n,m)) and Θ(g(n,m)).

Ans.