**DESIGN AND ANALYSIS OF ALGORITHMS**

**LAB 01: Asymptotic notation of complexity**

# Complexity notation

, – nonnegative, monotonically increasing functions, is the input size of algorithms.

## Big-Theta Notation:

### **Definition:**

### **Example**

### if

, i.e.

### **Big-Theta is a tight bound on the computational complexity function of algorithms**.

### **Application: After we get the complexity function, we drop all insignificant terms in the function and the coefficients to get the efficiency class of the complexity function.**

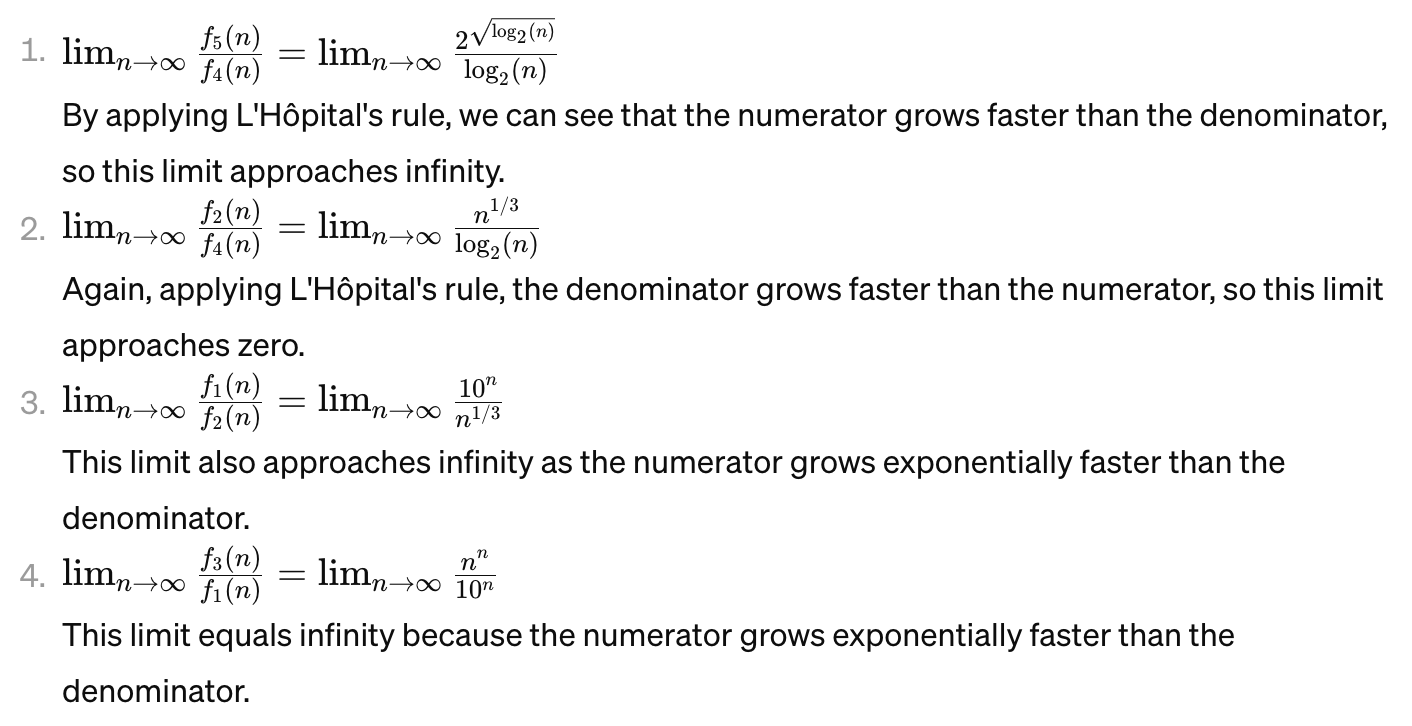
# Sample exercises with solutions on complexity notations

Question: sort the following functions in ascending order of the growth

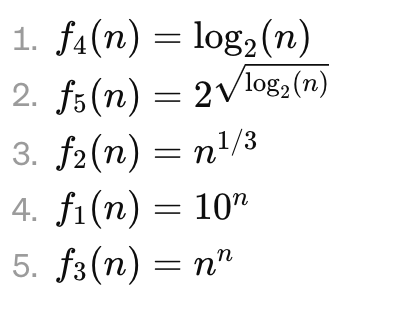
**Solution:**

To compare the growth rates of these functions, let's take the limit of their ratios as n approaches infinity:

Let's analyze each pair:



So, the functions in ascending order of growth are:



# Exercises

## Identify the efficiency class (big-Theta) of the following algorithms [in the worst cases]

### def function()

# input: A – an array of numbers

1. for to do

2. for to do

3. if return false

4. return true

### def f(A):

1. # input A is an array of n number
2. n = len(A)
3. for i in range(0, n-1):
4. max = A[i]
5. imax = i
6. for j in range(i+1, n):
7. if A[j] > max:
8. max = A[j]
9. imax = j
10. A[i], A[imax] = A[imax], A[i]

### Function F(n):

#Input: k integer, positive

#Output: count

1. count ← 0; i = n
2. while (i ≥ 1)
   1. for j ← 1 to n do
      1. count ← count + 1
      2. print(j)
   2. end for
   3. i ← i/3

end while

### ALGORITHMS F()

1. for to of the

2. for to of the

3. if return false

4. return true

### Function f(k):

1. int i, even;
2. i := 1;
3. even := 0;
4. while( i < k ) {
5. even := even + 2;
6. i := i + 1;
7. }
8. return even .

### Algorithm: f(natural number k)

1. int i, power;
2. i := 0;
3. power := 1;
4. while( i < k ) {
5. power := power \* 2;
6. i := i + 1;
7. }
8. return power .

### Algorithm bubbleSort(A : list of sortable items)

1 . n := length(A)

2 . repeat

3 . swapped := false

4 . for i := 1 to n-1 inclusive do

5 . /\* if this pair is out of order \*/

6 . if A[i-1] > A[i] then

7 . swap(A[i-1], A[i])

8 . swapped := true

9. end if

10. end for

11. until not swapped

12.end procedure

### ALGORITHMS func()

1. for to do

2. for to do

3. if return false

4. return true

### function insertionSort(array A)

01. i ← 1

02. while i < length(A)

03. x ← A[i]

04. j ← i - 1

05. while j >= 0 and A[j] > x

06. A[j+1] ← A[j]

07. j ← j - 1

08. end while

09. A[j+1] ← x

10. i ← i + 1

11.end while

## Are the following statements true or false?

* 1. ?
  2. ?
  3. If and then

(Hint: the statement is false; you may prove it with a counterexample)

## Are the following statements true or false, given two functions ?

## Arrange the following functions in ascending order of growth:

## Arrange the following functions in ascending order of growth: