Factorial(numero)

Inicio

Si numero es negativo

    Retornar -1

Si numero es 0 o 1 entonces

    retornar 1

Sino

    retornar numero * Factorial(numero-1)

FinSi

Fin

El caso baso sería el factorial de 0 que va ser el último número que voy a utilizar para generar el resultado, ya que el factorial de 0 es 1 y partir de este puedo generar el resto

Tiene un orden de tiempo de ejecución de O(n) ya que el código base sin contar la recursión es de orden O(1) y la recursión hace llamados a sí misma la según el tamaño de n reduciendo de 1 en 1 hasta llegar a 0.

SumaLineal(A,n)

Inicio

Si A está vacío entonces

    devolver 0

FinSi

Si n es 1 entonces

    Devuelvo A[0]

Sino

    Devolver SumaLineal(A,n-1) + A[n-1]

Finsi

Fin

Los casos bases son tanto el verificar que el array esté vacío, significa que ya no puedo ir sumando elementos y que se pida el último elemento (que puede hacer que la ejecución grande corte gracias a esto). Por ejemplo, tengo 4,6,7,1,2 en un array y pido la suma hasta el 4to número lo que ente caso mi caso base es el índice del último número a sumar y cuando termine de sumar 1 más los anteriores retornaré, que en este caso sería 4+6+7+1 = 18.

Este algoritmo tiene un orden también línea que ya que las líneas que no son recursivas son de orden O(1) y las llamadas recursivas se ejecutan n veces ya que voy a sumar número por número "recorriendo" cada número.

Exponencial(base, exponente)

Inicio

Si exponente es 0 entonces

    Retornar 1

Si (exponente < 0)

    Retornar 1 / Exponencial(base, exponente)

Sino

    Retornar Exponencial(base, exponente -1) * base

FinSi

Fin

Este algoritmo siempre hace que se llegue como caso base al exponente de 0 que es 1.

El algoritmo sería de Orden de tiempo de ejecución O(n) siendo n el exponente dado ya que multiplicaré la base n veces

InvertirArray(array, indiceMin, indiceMax)

Si índiceMin es mayor o igual a indiceMin entonces

    Return array

Else

    Auxiliar ←array[j]

    array[j] ← array[i]

    array[i] ←auxiliar

    return InvertirArray(array, indiceMin +1, índiceMax -1)

El algoritmo invierte tomando los índices más a lo extremos y los intercambia hasta que se encuentre parado sobre el mismo numero o si llega a la mitad del arreglo

El algoritmo es de orden O(N) siendo n el tamaño del array ya que hace en el peor de los casos tiene que invertir todo el array y hace la llamada recursiva n/2 veces pero como 2 es una constantes el algoritmo es de orden O(n)