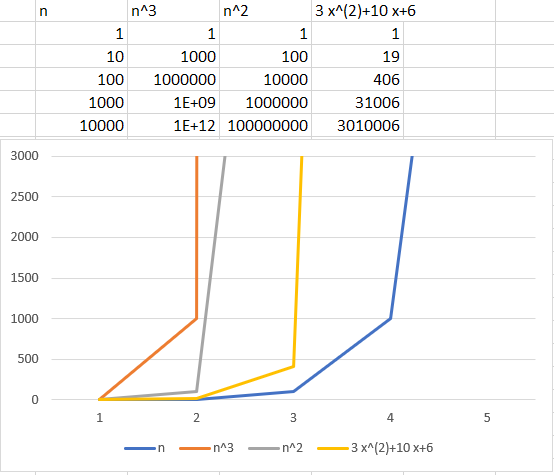


3\*100^2+10\*100+6 = 31006 100^3 = 1.000.000 100^2=10.000 100



1. Verdadero, O(n3) crece mas rápido que 3n2+10n+6 cumple que 3n2+10n+6 <= O(c.n3) para un c=3 acota superiormente
2. Verdadero, O(n2) crece mas rápido que 3n2+10n+6 cumple que 3n2+10n+6 <= O(c.n2) para un c=3 acota superiormente
3. Falso, la función 3n2+10n+6 crece mas rápido que O(n) 3n2+10n+6 > O(c.n) para c=3 No acota superiormente
4. Falso, ya que 3n2+10n+6 <= (n3) no cumple que f(n) es siempre ≥ (c.n) para c=3
5. Verdadero, la función es siempre > (c.n2) para un c=3
6. Verdadero, la función es siempre > para (c.n) para un c=3
7. Falso ya que solo cumple la definición para O(n3)
8. Verdadero ya que cumple para O(n2) y para (n2)
9. Falso ya que solo cumple definición para (n)

2) para la multiplicación de matrices, se deben de recorrer utilizando tres bucles for anidados, lo que implica que para una matriz de n x n, se realizaran n3 operaciones para recorrerla por completo, por lo tanto, la complejidad de tiempo del algoritmo será de O(n3)

*def* multi(matrizA,matrizB,filasA,columnasB):  
 matrizC = [[0 *for* \_ *in* range(columnasB)] *for* \_ *in* range(filasA)] *# crea una matriz llena de 0 para luego poner los resultados donde van  
 for* i *in* range(filasA):  
 *for* j *in* range(columnasB):  
 *for* k *in* range(len(matrizB)):   
 matrizC[i][j] += matrizA[i][k] \* matrizB[k][j]  
 *#orden O(n^3) causa de los 3 fors para recorrer ambas matrices  
 return* matrizC

en mi código también existen 2 fors mas que llenan las matrices A y B con números aleatorios y unos cuantos prints pero como estamos interesados en la complejidad asintótica, se pueden ignorar estos términos constantes o de orden menor, por lo que nos quedaría que el orden es de n3

para matrices cuadradas de 10x10 20x20 y 50x50

para n = 10 🡪 103 = 1000 operaciones para hacer el calculo

para n = 20 🡪 203 = 8000 operaciones para hacer el calculo

para n = 50 🡪 503 = 125000 operaciones para hacer el calculo

3) para el algoritmo que evalúa polinomios:

a)En el peor caso se hacen 2n multiplicaciones(xpower:=x∗xpower y ai∗xpower) y n sumas (p:=p+ai∗xpower)

b)En el caso promedio, el número de multiplicaciones sera aproximadamente de 2n/2 🡪 ósea el numero promedio de multiplicaciones será de n. Se necesitan n/2 polinomios para obtener un resultado y cada termino requiere una multiplicion

c) existe un algoritmo llamada método de horner el cual se reorganiza la expresión lo que hace que se minimice el numero de iteraciones necesarias para evaluar e l polinomio dado. El algoritmo utiliza una variable acumuladora p para almacenar el valor del polinomio evaluado hasta el momento. En cada iteración del bucle, se multiplica p por x y se le suma el siguiente coeficiente ai, de manera que se van evaluando los términos del polinomio de manera secuencial.

Su seudocódigo seria:

p := an

para i := n-1 hasta 0 hacer

p := p \* x + ai

fin para

4)

Graficos y cuentas en el exel

|  |  |
| --- | --- |
| n |  |
| (1/3)^n | exponencial |
| log(log(n)) | Logarítmica |
| log(n) | logarítmica |
| n\*log(n) | Logarítmica |
| n^(1/3)+log(n) | Orden mayor al logaritmo |
| n/(log(n)) | Lineal para n>= 3 |
| log(n)^2 | lineal |
| raiz(n) | Raíz cuadrática |
| 6 | Constante |
| n^2 | cuadratico |
| n^2+log(n) | Cuadratico |
| n^3 | cubico |
| (3/2)^n | exponencial |
| 2^n | exponencial |
| n-n^3+7n^5 | De grado 5 |
| n! | factorial |

5)

a) si, para O(n²) es posible que tenga una complejidad de O(n) solo para algunos datos, O(n²) se refiere a la complejidad máxima del algoritmo

b) no, si el peor caso del algoritmo es O(n²), siempre tendrá esa misma complejidad o peor, no se pueda dar el caso que para todos los datos la complejidad sea cambie a O(n)

c)si, para un algoritmo que sea Θ(n²) se puede dar el caso de que para algunos datos sea O(n), es posible que el algoritmo tenga mejor rendimiento para ciertos datos y posea una complejidad menor

d) no , si se demuestra que el algoritmo es de Θ(n²), entonces su complejidad siempre será al menos cuadrática para todos los tamaños de entrada, no puede tener una complejidad menor que cuadrática para todos los datos, ósea no puede ser O(n) para todos los datos