ANALISIS ASINTOTICOS DE ALGORITMOS RECURSIVOS

private static double product (double a, double b) {

if(b==1){ ……………………………………………………. max(1,max(1,1+b-1)) = b

return a; ………………………………………………… 1

}else{

return a + producto(a,b-1); ……………………. 1 + (b -1) = b

}

}

Al desarrollar nos sale que la función tiempo es b por lo tanto la notación big O es O(b) lineal

TASA DE CRECIMIENTO

|  |  |
| --- | --- |
| b | F(b) |
| 1 | 1 |
| 10 | 10 |
| 1000 | 1000 |

private static double potencia (double base, double exponente) {

if (exponente == 0) { ………………………………………………………….. max(1,1) = 1

return 1; ………………………………………………………………………… 1

} else if (exponente % 2 == 0) {…………………………………………… max(2,1+n/2) = 1+ T(n/2)

double temp = potencia(base, exponente / 2);………………. 1+n/2

return temp \* temp; ……………………………………………………… 2

} else {

double temp = potencia(base, (exponente-1) / 2); ………… 1+T((n-1)/2)

return base \* temp \* temp; …………………………………………… 3

}

}

• El tiempo para la primera llamada recursiva t(N/2) = 1 + t(N/4)

• El tiempo para la segunda llamada recursiva t(N/4) = 1 + t(N/8)

• El tiempo para la tercera llamada recursiva t(N/8) = 1+ t(N/16)

Así sucesivamente:

• El tiempo para la penúltima llamada recursiva t(2) = 1 + t(N/2k)

Por lo tanto:

t(N/2) = 1 + 1 + 1 + ... + 1 + t(N/2k)

Y como t(N/2k) = t(1) N = 2k log2N = log22k k = log2N

por lo tanto la notación big O es O(log2N) logaritmica

TASA DE CRECIMIENTO

|  |  |
| --- | --- |
| n | F(n) |
| 1 | 1 |
| 8 | 3 |
| 256 | 8 |
| 1024 | 10 |

private static double factorial (int n) {

if(n==0){ ……………………………………………………. max(1,max(1,1+(n-1))) = n

return 1; ………………………………………………… 1

}else{

return n\*factorial(n-1); ………………………….. 1 + (n -1) = n

}

}

Al desarrollar nos sale que la función tiempo es n por lo tanto la notación big O es O(n) lineal

TASA DE CRECIMIENTO

|  |  |
| --- | --- |
| n | F(n) |
| 1 | 1 |
| 10 | 10 |
| 1000 | 1000 |