Ejercicios página 21

Ejercicios 5

1. Si T(N) es O(F(N)) y S(N) es O(F(N)) ¿Cuáles de las siguientes afirmaciones son ciertas?
2. T(N)+S(N) es O(F(N)) }
3. T(N)-S(N) es O(F(N))
4. T(N)/S(N) es O(1)
5. T(N) es O(S(N))

La c

1. Para la tabla siguiente que resume los tiempos tomados para diferentes entradas de un algoritmo cualquiera, ¿cómo se pueden analizar los valores de las columnas de la derecha para determinar, en notación O, el orden de crecimiento del tiempo de ejecución?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | T | T/N | T/N^2 | T/ (N log N) |
| 10000 | 100 | 0.01 | 0.000001 | 0.0025 |
| 20000 | 200 | 0.01 | 0.0000005 | 0.00232502 |
| 40000 | 400 | 0.01 | 0.00000025 | 0.00217294 |
| 80000 | 800 | 0.01 | 1.25E-07 | 0.00203953 |
| 160000 | 1600 | 0.01 | 6.25E-08 | 0.00192155 |
| 320000 | 3200 | 0.01 | 3.125E-08 | 0.00181648 |
| 640000 | 6400 | 0.01 | 1.5625E-08 | 0.0017223 |

Como podemos observar mientras aumenta N en T/N nos damos cuenta que existe un valor determinado por lo cual podríamos decir que el orden de crecimiento del algoritmo evaluado es O(N), también podríamos decir que O(NlogN) puede ser el orden de crecimiento del tiempo de algoritmo ya que sus valores son semejantes, pero sabemos que la notación más acertada para este caso es O(N)

Ejercicio 6

1. Mediante la notación asintótica, obténgase los tiempos de ejecución del peor caso supuesto para cada uno de los procedimientos Siguientes como una función de N.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datos | prod\_mat | misterio | muy\_impar | recursiva |
| 1000 | 2.653 | 0.002 | 0.16 | indefinido |
| 2000 | 47.36 | 0.007 | 0.985 | indefinido |
| 4000 | 317.026 | 0.02 | 10.834 | indefinido |
| 8000 | desborda | 0.055 | 75.37 | indefinido |
| 16000 | desborda | 0.186 | desborda | indefinido |

El código recursivo no se pudo definir el tiempo con esa cantidad de datos así que vamos a calcularlo con una menor cantidad de datos

|  |  |
| --- | --- |
| Datos | recursiva |
| 10 | 0.001 |
| 20 | 0.005 |
| 30 | 4.554 |
| 40 | indefinido |
| 50 | indefinido |

1. Un algoritmo tarda 0,05 milisegundos para una entrada de tamaño10. ¿Cuánto tardará con una entrada de tamaño 20 si el tiempo de ejecución es a) cúbico b) exponencial 2n? c) Explique cual de los dos tiempos de ejecución es menos recomendable.

Si el tiemo de ejecución es cubico tardaría 0,4 milisegundos

Si el tiempo de ejecución es 2^n entonces tardaría 51,2 milisegundos

Por lo tanto podemos determinar que el tiempo de ejecución menos recomendable es el 2^n ya que es mucho mayor al tiempo de ejecución de n^3

1. Ciertos seres alienígenas invadieron la tierra y se reproducen en una proporción por minuto equivalente a la potencia cuarta de alienígenas que había el minuto anterior. Por ejemplo, al primer minuto había 4 cuatro seres, al segundo minuto existían 4\*4\*4\*4=216 seres, al tercero 216\*216\*216\*216 seres, etc. Cada vez que nace un nuevo ser alienígena se apodera misteriosamente de la mente (una sola por alienígena) de algún ser humano y la destruye. ¿Cuál sería la cantidad de alienígenas en el minuto k? b) ¿Cuántos minutos harían falta para que estos seres se apoderen de las mentes de los 6.000 millones de personas que habitan en el planeta?

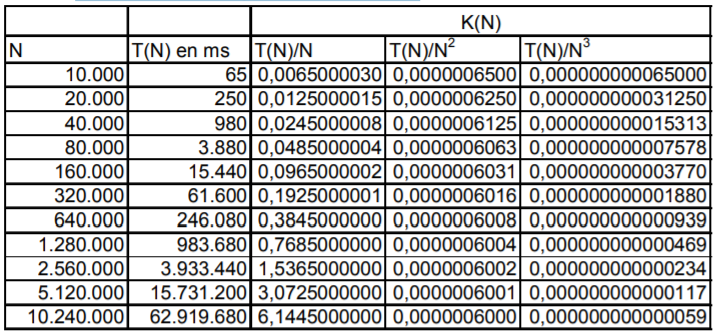
En el minuto K la cantidad de alienígenas

Cuantos minutos harían falta para que estos seres se apoderen de los 6000 millones de personas.

1. Explique cuál es el orden de ejecución del método llamado Misterio1(). b) Como se muestra en la llamada a Misterio1(2) y de acuerdo al orden de ejecución O encontrado, si la entrada fuera N=2 de un cálculo aproximado del tiempo de ejecución c) ¿Cuál es el valor de la variable cuenta al final de la ejecución de la llamada a Misterio1(2)? ¿Esta relacionado este valor de cuenta con el tiempo de ejecución? Explique.

El orden de proceso es n^5 para lo cual el valor cuenta cundo n=2 es igual a 8 podríamos hacer una relación de de que cuenta al final es igual a n^3 ya que probando para dos valores como n=2 y n=5 tenemos la misma relación donde 2^3=8 y 5^3= 125

1. Suponga que un algoritmo tiene un tiempo de ejecución cuya fórmula se ignora y para el cual se han realizado las siguientes mediciones:



1. ¿Cuál debería ser el valor de N para un tiempo T(N)=2.000.000.000? Escriba una fórmula aproximada para T(N) y explique como la obtiene.

Existe una relación en el tiempo de ejecución y el valor N como vemos el valor N siempre es el doble del anterior y el valor de T(N) siempre es la multiplicación por 3.8 del valor del T(N) anterior así que podemos decir que en el tiempo en el que nos encontremos habrá un N inicial \* 2^x que tendrá un T(N)inicial \*3.8^x donde el x es el mismo para ambos casos. Por lo tanto, hemos definido que para T(N)=2,000,000,000 N=62,083,800

1. Observando el cuadro anterior, mediante una expresión algebraica explique porque la columna para T(N)/N3 tiende a disminuir en la medida que N aumenta.

Como podemos observar ningun valor T(N) es mayor que un N^3 por los tanto mientras mas grande sea el valor de N la operación va a disminuir en este caso la expresion algebraica seria de la forma y/x^3 -> x^3>Y

1. Observando el cuadro anterior, mediante una expresión algebraica explique porque la columna para T(N)/N2 tiende a ser constante en la medida que N aumenta.

En primer lugar eso determinaría el orden de la operación por lo que el orden seria N^2 y también por que existe una relación de N^2 con el T(N) y es por eso que cada dato nos da una constante

1. ¿F(N)=150N acota superiormente a T(N)? ¿F(N)=150N2 acota superiormente a T(N)? ¿F(N)=150N3 acota superiormente a T(N)? Según sus respuestas explique cual es el orden de ejecución del algoritmo analizado.

En todos los casos propuestos de F(N) seran una cota superios de T(N) pero el valor K(N) = T(N)/F(N) esconstante unicamente cuando N^2 por lo que podeos definir que el orden para este caso será un O(N^2)

1. Escriba una fórmula para calcular el tiempo de ejecución T(N) en función de N para el siguiente fragmento de código:

void fun\_Ex(int N){

int cuenta=0;

int i=3

while (i<N)

{ cuenta=cuenta+10;

int j=2;

while (j<N)

{ cuenta=cuenta+100;

j=j+1;

}

i=i+1;

}

}

int main(){

clock\_t tmI=0, tmF=0;

int n=40;

double tiempo;

tmI = clock();

recursiva(n);

cout << n << endl;

tmF = clock();

tiempo = (double)((double)(tmF - tmI) / CLOCKS\_PER\_SEC);

cout << tiempo << " segundos" << endl;

cout << endl;

return 0;

}