

Ejercicio Complejidad2

Estructuras de Datos

Tema 1: tipos abstractos de datos y algoritmia

1º Grado en Ingeniería de la Computación

© Profesor Dr. Carlos Grima Izquierdo (www.carlosgrima.com)

URJC (www.urjc.es)

Calcula el $T(n)$ y a continuación su $O(n)$ temporal en los siguientes algoritmos recursivos. Para ello, sigue los siguientes pasos para cada uno de ellos:

1. Calcula su $T(n)$ recurrente, no olvidando las condiciones iniciales.
2. Intenta eliminar la recurrencia intuitivamente, desarrollando la serie
3. Si no te es posible obtener intuitivamente la solución, utiliza WolframAlpha
4. A partir del $T(n)$ no recurrente obtenido, calcula su $O(n)$

Apartado a

Tomamos el exponente como “n”, ya que es el único parámetro que influye en el tiempo que tarda el algoritmo en ejecutarse.

```
float elevarAPotencia (float base, int exponente) {  
    if (exponente == 0) return 1;  
    else return (base*elevarAPotencia(base,exponente-1));  
}
```

Apartado b

```
int sumarHasta (int n) {  
    if (n == 0) return 0;  
    else return (n+sumarHasta(n-1));  
}
```

Apartado c

```
bool buscar(int *vector, int n, int numeroABuscar) {  
    if (n==0) return false;  
    else  
        if (vector[n-1] == numeroABuscar) return true;  
    else return (buscar(vector, n-1, numeroABuscar));  
}
```

Apartado d

Tomamos el operandoA como “n”, ya que es el único parámetro que influye en el tiempo que tarda el algoritmo en ejecutarse.

```

long multiplicarALaRusa(long operandoA, long operandoB) {
    if (operandoA==1) return(operandoB); // Caso trivial
    else {
        long nuevoA = operandoA / 2; // Nuevo primer operando
        long nuevoB = operandoB * 2; // Nuevo segundo operando
        // Distinguimos el caso de que A sea impar o par
        if ((operandoA%2)!=0) // A impar
            return (operandoB+multiplicarALaRusa(nuevoA, nuevoB));
        else // A par
            return (multiplicarALaRusa(nuevoA, nuevoB));
    }
}

```

Apartado e

Como el algoritmo está en pseudocódigo, en vez de contar exactamente el número de operaciones elementales, vamos a sustituir cualquiera de esos números por constantes que denotaremos por c_1 , c_2 , c_3 , c_4 ... Ej: $c_1n^2+c_2\log n+c_3$. Esto lo vamos a hacer así porque, para calcular el $O(n)$, las constantes se desprecian.

```

int calcular(int n) {
    float a = 1;
    if (n == 1) return(0);
    else {
        for (int i = 1; i <= n; i++) a = a + i + n * n;
        return(calcular(n / 2) + 1 / n);
    }
}

```