

Universidad Rey Juan Carlos
Ingeniería Técnica en Informática de Gestión

Asignatura: Estructuras de Datos y de la Información

HOJA DE PROBLEMAS Nº 2: Complejidad de Algoritmos

Estudio de la complejidad (tiempo de ejecución en el peor caso) de algoritmos.

1) Calcular la complejidad de los siguientes trozos de algoritmos en función de n .

- a) $x := 0;$
 PARA i DESDE 1 HASTA n HACER
 PARA j DESDE 1 HASTA n HACER
 PARA k DESDE 1 HASTA n HACER
 $x := x + 1;$
- b) $x := 0;$
 SI $(n \bmod 2 \neq 0)$ ENTONCES
 PARA i DESDE 1 HASTA n HACER
 PARA j DESDE 1 HASTA i HACER
 $x := x + 1;$
 SI NO
 $x := -1;$
- c) $x := 0;$
 PARA i DESDE 1 HASTA n HACER
 PARA j DESDE 1 HASTA n^2 HACER
 PARA k DESDE 1 HASTA n^3 HACER
 $x := x + 1;$
- d) $x := 0;$
 PARA i DESDE 1 HASTA n HACER
 PARA j DESDE 1 HASTA i HACER
 PARA k DESDE j HASTA n HACER
 $x := x + 1;$

2) Algoritmo iterativo para hallar el máximo de un vector v con n elementos.

```
máximo := v(1);  
PARA i DESDE 2 HASTA n HACER  
  SI v(i) > máximo ENTONCES máximo := v(i)
```

3) Algoritmo para ordenar de menor a mayor un vector v con n elementos (ordenación por selección). (se supondrá que la operación *Intercambiar* tiene un tiempo de ejecución constante).

```
PARA i DESDE 1 HASTA n-1 HACER  
  pmin := i;  
  PARA j DESDE i+1 HASTA n HACER  
    SI v(j) < v(pmin) ENTONCES pmin := j;  
  Intercambiar(v(i), v(pmin));
```

- 4) Algoritmo para ordenar de menor a mayor un vector v con n elementos (ordenación por el método de inserción).

```

PARA i DESDE 2 HASTA n HACER
    pos := i;
    x := v(i);
    seguir := CIERTO;
    MIENTRAS pos>1 y seguir HACER
        SI x < v(pos-1) ENTONCES
            v(pos) := v(pos-1)
        SI NO seguir:= FALSO;
        pos := pos - 1;
    v(pos) := x;

```

- 5) Algoritmo para ordenar de menor a mayor un vector v con n elementos (ordenación por el método de la burbuja). (se supondrá que la operación *Intercambiar* tiene un tiempo de ejecución constante).

```

paso := 1;
intercambio := CIERTO;
MIENTRAS (paso <= n-1) Y intercambio HACER
    intercambio := FALSO;
    PARA i DESDE 1 HASTA n-paso HACER
        SI v(i) > v(i+1) ENTONCES
            Intercambiar(v(i), v(i+1));
            Intercambio := CIERTO;
    paso := paso + 1;

```

- 6) Algoritmo para buscar secuencialmente un elemento en un vector v con n elementos.

```

FUNCIÓN Buscar (v:TipoVecor; buscado:TipoElemento) DEVUELVE Booleano
    i := 1;
    MIENTRAS (i<=n) Y (v(i)≠buscado) HACER
        i := i + 1;
    DEVOLVER i<=n;

```

- 7) Algoritmo iterativo para buscar un elemento en un vector ordenado v con n elementos mediante búsqueda binaria.

```

FUNCIÓN Pertenece(v:TipoVector;buscado:TipoElemento)
    DEVUELVE Booleano
    bajo := 1;
    alto := n;
    encontrado := CIERTO;
    REPETIR
        medio := (bajo + alto) DIV 2;
        SI v(medio) = buscado ENTONCES
            encontrado := CIERTO;
        SI NO
            SI v(medio)<buscado ENTONCES
                bajo := medio + 1;
            SI NO
                alto := medio - 1;
    HASTA QUE encontrado O bajo>alto;
    DEVOLVER encontrado;

```

- 8) Algoritmo recursivo para buscar un elemento en un vector ordenado v con n elementos mediante búsqueda binaria.

```
FUNCIÓN Pertenece(v:TipoVector; comienzo, final: TipoPosicion;
    buscado:TipoElemento) DEVUELVE Booleano

    SI comienzo = final ENTONCES
        DEVOLVER v(comienzo)=buscado;
    SI NO
        medio := (comienzo + final) DIV 2;
        SI buscado = v(medio) ENTONCES
            DEVOLVER CIERTO
        SI NO
            SI buscado < v(medio) ENTONCES
                DEVOLVER Pertenece(v, comienzo, medio-1, buscado)
            SI NO
                DEVOLVER Pertenece(v, medio+1, final, buscado)
```

- 9) Algoritmo iterativo para hallar el factorial de un número natural n

```
fact := 1;
PARA i DESDE 1 HASTA n HACER
    fact := fact * i;
```

- 10) Algoritmo recursivo para hallar el factorial de un número natural n

```
FUNCIÓN Factorial (n: Natural) DEVUELVE Natural
    SI n=0 ENTONCES
        DEVOLVER 1;
    SI NO
        DEVOLVER n*Factorial(n-1);
```

- 11) Algoritmo para hallar el n -ésimo elemento de la sucesión de Fibonacci

La sucesión de Fibonacci se define, para números naturales, de la siguiente forma:

$$\text{Fibonacci}(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ \text{Fibonacci}(n-1) + \text{Fibonacci}(n-2) & \text{e.o.c} \end{cases}$$

Un algoritmo para calcular el n -ésimo término de la sucesión es el siguiente:

```
SI n<=1 ENTONCES
    fib_n := 1;
SI NO
    fib_menor := 1;
    fib_mayor := 1;
    PARA i DESDE 2 HASTA n HACER
        aux := fib_menor;
        fib_menor := fib_mayor;
        fib_mayor := aux + fib_mayor;
    fib_n := fib_mayor;
```