TEMA 6.- ESQUEMAS ALGORITMICOS: DIVIDE Y VENCERAS

E 1.- Analizar los siguientes algoritmos con bucles

```
ALGORITMO B1
DATOS: n : N
RES: s: N
METODO:
    s ← 0
    Para i ← 1 hasta n hacer
        Para j ← 1 hasta i hacer
            s ← s +1
        fPara
    fPara
fB1
```

```
ALGORITMO B2
DATOS: n : N
RES: s: N
METODO:
    s ← 0
    Para i ← 1 hasta n hacer
        Para j ← 1 hasta i² hacer
            Para k ← 1 hasta j hacer
                s ← s +1
            fPara
        fPara
    fPara
fB2
```

E 2.- (CLASE) Analizar los siguientes algoritmos con bucles

```
ALGORITMO B3
DATOS: n : N
RES: s: N
METODO:
    s ← 0
    Para i ← 1 hasta n hacer
        Para j ← 1 hasta i hacer
            Para k ← j hasta n hacer
                s ← s +1
            fPara
        fPara
    fPara
Fb3
```

```
ALGORITMO B4
DATOS: n : N
RES: s: N
METODO:
    s ← 0
    Para i ← 1 hasta n hacer
        Para j ← 1 hasta i² hacer
            Si j mod i = 0 entonces
                Para k ← 1 hasta j hacer
                    s ← s +1
                fPara
            fSi
        fPara
    fPara
fB4
```

E 3.- (CLASE) Dado un vector que contiene elementos del tipo Color=(Blanco, Azul, Rojo) escribir y analizar un algoritmo que separe los elementos de forma que primero estén los blancos, después los azules y al final los rojos.

E 4.- Una modificación de la búsqueda dicotómica consiste en seleccionar el punto central en cada iteración de forma que el vector quede dividido en dos partes, cuyo tamaño sea dos números consecutivos de la sucesión de Fibonacci (F(n) = F(n-1) + F(n-2)). Más concretamente, si el tamaño del vector cumple que n = F(m) –1 para algún m, el vector se puede dividir en dos partes de tamaño F(m-1)-1 y F(m-2)-1, respectivamente, más el elemento central. El algoritmo continuaría de la misma manera en cualquiera de los dos subvectores.

1.- Escribe un algoritmo recursivo que implemente esta idea.

2.- Analiza el resultado y compáralo con las diferentes versiones de la búsqueda dicotómica.

3.- Escribe un algoritmo iterativo equivalente.

E 5.- (CLASE) Una extensión trivial de la búsqueda binaria consiste en la división del vector en tres partes iguales en lugar de dos. Aunque las comprobaciones serían un poco más costosas, el algoritmo debería terminar en menos iteraciones. Escribe un algoritmo recursivo de "Búsqueda Ternaria" que implementa esta idea y analízalo. Compara los resultados con los obtenidos para la búsqueda binaria.

E 6.- (CLASE) Escribir un algoritmo recursivo que calcule todos los elementos de la base del triangulo de Pascal de orden n (sobre un vector), haciendo servir nada más la definición que se da a continuación. Analizar el algoritmo. ¿ Se puede mejorar el coste de alguna manera si lo que se pide es sólo la suma de los elementos de la base del triangulo ?.

```
                1
            1       1
        1       2       1
    1       3       3       1
1       4       6       4       1
```

El triangulo de pascal se define como una mesa triangular con n líneas centradas en la línea i, que tiene i elementos, que se calculan sumando los dos elementos más próximos de la línea anterior.

E 7.- Otra extensión de la búsqueda binaria consiste en hacer una estimación del lugar del vector donde (probablemente) se encuentre el elemento deseado. A partir del valor de los elementos primero y último de la porción de vector considerada se puede interpolar cual sería la posición del elemento deseado, x, en caso que los elementos en el vector siguieran una distribución uniforme. La figura 1 ilustra esta idea. El algoritmo resultante recibe el nombre de "Búsqueda por interpolación". Escribe el algoritmo y razona sobre su coste en relación a la búsqueda binaria.

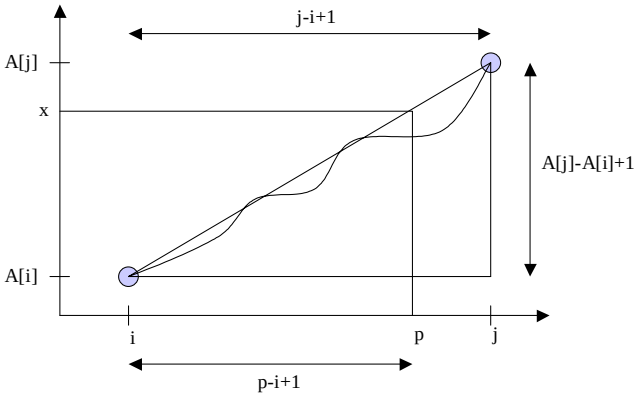

Figura 1.- Obtención de la posición siguiente a comprobar en el algoritmo de búsqueda por interpolación. La posición, p, se obtiene como resultado de suponer una distribución uniforme de los elementos del subvector T[i..j[

E 8.- (CLASE) Escribir una versión recursiva del algoritmo que calcula la potencia n-esima de X (Sugerencia: relacionar los casos n y n/2).

E 9.- Escribir un algoritmo recursivo que calcule la función $\lfloor \log_b n \rfloor$

E 10.- Indica cual es el coste espacial y temporal del siguiente algoritmo.

```
ALGORITMO Multiplicación(x,y:R)R
Metodo:
    Si y = 0 entonces Multiplicación ← 0
    Sino si y mod 2 ≠ 0 entonces
        Multiplicación ← Multiplicación(2x, ⌊y/2⌋) + x
    Sino
        Multiplicación ← Multiplicación(2x, ⌊y/2⌋)
    fSi
fMultiplicación
```

E 11.- (CLASE) Escribe una versión del algoritmo que resuelve el problema de Hanoi, pero en el caso en que los únicos movimientos permitidos sean hacia la izquierda. Analiza el algoritmo resultante.

E 12.- Escribir un algoritmo que resuelva el problema de las torres de Hanoi pero en el caso en que los únicos movimientos permitidos sean los que parten o llegan a la aguja B. Analizar el algoritmo resultante en el peor caso.

E 13.- (CLASE) Resuelve el problema de Hanoi para n discos y cuatro agujas mediante divide y vencerás. Analizar el algoritmo que se obtiene.

E 14.- Suponer que se dispone de un algoritmo A que resuelve un determinado problema P cuyo coste exacto es $T_A(n)$. Si P es susceptible de ser dividido en partes i, y además, es posible encontrar la solución de P dadas las soluciones parciales, entonces P se puede resolver (en principio) mediante la técnica de Divide y Vencerás con un coste que viene dado por la relación de recurrencia:

$$T(n) = \begin{cases} t_0 & si\ n \le 1 \\ 2T(n/2) + f(n) & si\ n > 1 \end{cases}$$

donde normalmente se cumple que $O(T(n)) < O(T_A(n))$

Una práctica habitual al implementar una solución de este tipo consiste en la elección de un límite distinto de 1 para finalizar la recurrencia con objeto de utilizar el algoritmo original en lugar de la solución trivial para un problema de tamaño 1. En este caso, la recurrencia queda de la siguiente forma:

$$T(n) = \begin{cases} T_A(n) & si\ n \le l \\ 2T(n/2) + f(n) & si\ n > l \end{cases}$$

Evidentemente, si l es finito, no se modifican los costes asintóticos, pero se puede rebajar el coste para una determina instancia de una manera considerable. Contestar, ¿Cuándo convendrá aplicar esta técnica? a) ¿En función de la forma de $T_A(n)$? (como se puede encontrar el mejor límite) b) ¿En función de la forma de f(n)?