FEBRERO 2014 – SEMANA 1 (MODELO A)

1. Se tiene una tabla hash con n= 23 índices ocupados y de tamaño M= 1000. El factor de carga δ será:

(a) 0,023, es decir n/M
(b) 0,046, es decir 2n/M
(c) -3,7723, es decir Ln (n/M)
(d) Ninguna de las anteriores

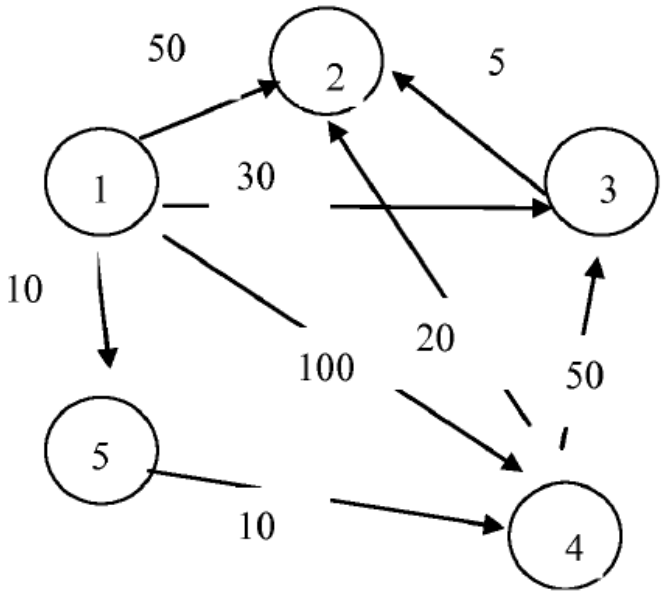
El factor de carga (δ) es un parámetro que determina cuantas posiciones de la tabla se han ocupado y la probabilidad de encontrar una entrada vacía. El factor de carga se define como:

$$\delta = n / M$$

$\delta = 23 / 1000$; ***δ = 0'023***

Respuesta A)

2. Dado el siguiente grafo dirigido:



Indique cuál sería el valor del vector especial[] en el primer y penúltimo paso del algoritmo de Dijkstra:

(a) [50,30,100,10] y [35,30,20,10]

(b) [50,30,100,10] y [40,30,20,10]

(c) [50,30,100,10] y [35,30,100,10]

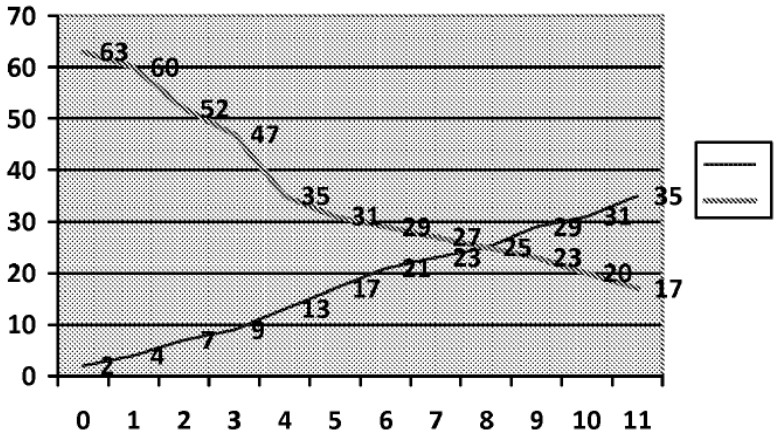
(d) Ninguna de las anteriores

Nodos que han salido	Nodos que no han salido	Vector Distancia especial [ ]				Predecesores			
		2	3	4	5	2	3	4	5
1	2, 3, 4, 5	50	30	100	**10**	1	1	1	1
1, 5	2, 3, 4	50	30	**20**	10	1	1	5	1
1, 5, 4	2, 3	40	**30**	20	10	4	1	5	1
1, 5, 4, 3	2	40	30	20	10	4	1	5	1

Nota: En la última iteración, el valor de ir al nodo 1 al 2, sigue siendo 40, ya que su predecesor era 4, no siendo posible ir de 1 a 3 y de 3 a 2 sin pasar antes por 4.

Respuesta B)

3. Considere dos vectores f y g de n elementos que representan los valores que toman dos funciones en el intervalo [0..n-1]. Los dos vectores están ordenados, pero el primero f es un vector estrictamente creciente (f[O] < f[1] < ... < f[n-1]), mientras g es un vector estrictamente decreciente (g[O] > g[1]> ... > g[n-1]). Las curvas que representan dichos vectores se cruzan en un punto concreto, y lo que se desea saber es si dicho punto está contenido entre las componentes de ambos vectores, es decir, si existe un valor i tal que f[i] = g[i] para 0 <= i <= n-1. La figura muestra un ejemplo en el que las gráficas se cruzan en i=8 que se corresponde con el valor 25 en ambas curvas.



Se busca un algoritmo que compruebe si el punto de cruce está contenido en las componentes de ambos vectores. ¿Cuál de las siguientes opciones es <u>cierta</u>?

(a) Se puede encontrar un algoritmo recursivo de coste logarítmico.

Cierto. Con el algoritmo Divide y Vencerás, cuyo coste logarítmico es O(log n).

(b) El algoritmo más eficiente que se puede encontrar es O(n).

(c) El algoritmo más eficiente que se puede encontrar es $O(n^2)$.

(d) Ninguna de las anteriores.

Respuesta A)

4. ¿Cuál de las siguientes afirmaciones es <u>falsa</u>?

(a) El algoritmo de ordenación por fusión (mergesort) es O(n log n). → Cierto

(b) La eficiencia del algoritmo de ordenación rápida (quicksort) es independiente de que el pivote sea el elemento de menor valor del vector.

> Falso: Si siempre eliges el pivote de menor valor, o si el vector esta ordenado de menor a mayor y siempre coges el primer elemento, obtendrás coste $O(n^2)$.

(c) El algoritmo de ordenación rápida en el caso peor es $O(n^2)$. → Cierto

(d) El algoritmo de ordenación basada en montículos (heapsort) es O(n log n). → Cierto

<u>Respuesta B)</u>

5. ¿Cuál de las siguientes afirmaciones es <u>falsa</u>?

(a) El grado de un vértice de un grafo no dirigido es el número de aristas que salen o entran en él. → Cierto

(b) Un camino en un grafo dirigido es una secuencia finita de arcos entre dos vértices, tal que el vértice del extremo final de cada arco coincide con el del extremo inicial del arco siguiente. → Cierto

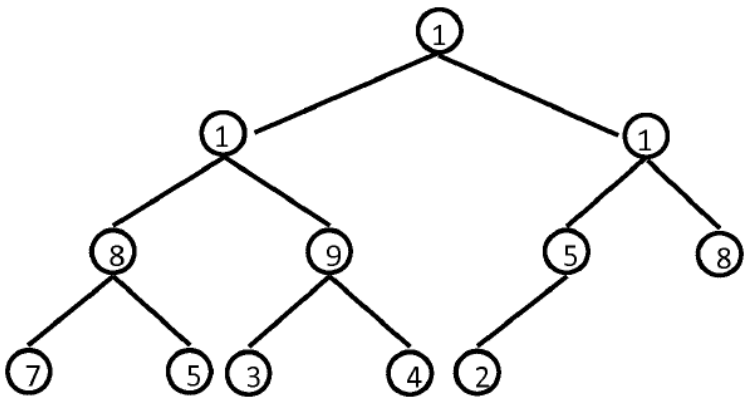
(c) Un grafo nulo es un grafo sin vértices. → Cierto

(d) La longitud de un camino es el número de aristas y nodos que contiene.

> Falso. La longitud de un camino es el número de aristas que contiene, pero no el número de nodos.

<u>Respuesta D)</u>

6. Dado el siguiente montículo:



Indique cuál de las siguientes afirmaciones es <u>falsa</u>:

(a) El montículo propuesto es un montículo de máximos.
(b) El vector que lo representa de forma correcta es [15,13,12,8,9,5,8,7,5,3,4,2].

(c) El orden de complejidad de la operación de inserción de un nuevo elemento en el montículo es O(n).
(d) Los montículos son estructuras de datos de gran utilidad a la hora de implementar colas de prioridad.

ERRATA: Esta pregunta se ha anulado en el examen. Si lo consideramos como indica el apartado B), entonces la única respuesta falsa es C), porque al insertar un elemento hay que restaurar la propiedad montículo, y esto se hace con flotar, coste log(n). En el gráfico deberían haber aparecido los nodos 15, 13 y 12.

PROBLEMA (4 Puntos)

Se tienen 3 palos verticales y n discos agujereados por el centro. Los discos son todos de diferente tamaño y en la posición inicial están insertados en el primer palo ordenados en tamaños en sucesión decreciente desde la base hasta la altura. El problema consiste en pasar los discos del 1er al 3er palo, utilizando el segundo como auxiliar, observando las siguientes reglas:

a) Se mueven los discos de 1 en 1.
b) Nunca un disco puede colocarse encima de uno menor que éste.

La resolución de este problema debe incluir, por este orden:

1. Elección del esquema más apropiado, el esquema general y explicación de su aplicación al problema (0,5 puntos).

El esquema más apropiado es el de Divide y Vencerás. De hecho, es un ejemplo clásico de problema que se resuelve con este tipo de esquema.

El esquema general es el siguiente:

```
fun DyV(problema)
    si trivial(problema) entonces
        dev solución-trivial
    sino hacer
        {p1, p2, ...pk} ← descomponer(problema)
        para i ∈ (1..k) hacer
            si ← DyV(pi)
        fpara
    fsi
    dev combinar(s1, s2, ...sk)
ffun
```

La aplicación de este esquema al problema consiste en trasladar primero los n-1 discos superiores desde la varilla origen hasta la varilla auxiliar; a continuación, trasladar el disco mayor desde el origen al destino, y, por último, volver a trasladar los n-1 discos desde la varilla auxiliar hasta el destino.

2. Algoritmo completo a partir del refinamiento del esquema general (3 puntos solo si el punto 1 es correcto). Si se trata del esquema voraz debe hacerse la demostración de optimalidad.

Un algoritmo recursivo para resolver las Torres de Hanoi con n discos consiste en trasladar primero los n − 1 discos superiores desde el origen hasta la varilla auxiliar, a continuación, se traslada el disco mayor desde el origen al destino, y, por último, se vuelven a trasladar los n−1 discos desde la varilla auxiliar hasta el destino.

El procedimiento mover(A, B : torre) quita el disco superior de la torre A y lo coloca sobre la torre B.

```
proc Hanoi(origen, destino, auxiliar : torre, e n : nat)
    si n = 1 entonces mover(origen, destino)
    si no
        Hanoi(origen, auxiliar, destino, n − 1)
        mover(origen, destino)
        Hanoi(auxiliar, destino, origen, n − 1)
    fsi
fproc
```

3. Estudio del coste del algoritmo desarrollado (0.5 puntos solo si el punto 1 es correcto).

El número de movimientos se obtiene a partir de la siguiente recurrencia:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n-1)+1 & n > 1 \end{cases}$$

Resolviendo la misma tenemos:

$$T(n) = 2^i T(n-i) + \sum_{j=0}^{i-1} 2^j = 2^{n-1} T(1) + \sum_{j=0}^{n-2} 2^j = \sum_{j=0}^{n-1} 2^j = 2^n - 1$$

El problema se soluciona con 2^n-1 movimientos, por tanto, el coste total del algoritmo es $O(2^n)$.