



Examen 19 Enero 2019, preguntas

Algoritmos (Universidade da Coruña)

Examen de Algoritmos Enero de 2019

Apellidos:	
Nombre:	DNI:

1. (1'5 puntos) La siguiente tabla contiene los tiempos de ejecución de un algoritmo de ordenación rápida (*quicksort*) con selección del pivote por *mediana de 3* y un *umbral* para detectar vectores pequeños, siendo el algoritmo *ad hoc* la ordenación por inserción.

El algoritmo se ejecuta sobre un vector con 32000 elementos en tres situaciones iniciales distintas: (a) el vector ya está ordenado en orden ascendente, (b) el vector ya está ordenado en orden descendente, y (c) el vector está inicialmente desordenado.

	<i>umbral</i> = 2	<i>umbral</i> = 20	<i>umbral</i> = 200
(a) <i>ascendente</i>	2117	1148	801
(b) <i>descendente</i>	2246	1270	925
(c) <i>aleatorio</i>	5130	4456	8009

¿Cuál es el mejor umbral para cada una de las distintas situaciones iniciales del vector? ¿Por qué? **Razone su respuesta.**

2. (1'5 puntos) A partir de la siguientes definiciones para la implementación de *conjuntos disjuntos*:

tipo

```
Elemento = entero;
Conj = entero;
ConjDisj = vector [1..N] de entero
```

y del siguiente pseudocódigo para la unión de dos conjuntos:

```
procedimiento Unir (C, raíz1, raíz2)
    { supone que raíz1 y raíz2 son raíces que representan a cada conjunto }
    { y C es la colección de conjuntos disjuntos }
    si raíz1 < raíz2 entonces C[raíz2] := raíz1
    sino C[raíz1] := raíz2
fin procedimiento
```

- Escriba el correspondiente pseudocódigo de *Buscar*(C, x) : Conj, que devuelve el nombre del conjunto (es decir, su representante) de un elemento dado.
 - Detalle la complejidad computacional de una secuencia de m búsquedas y $n - 1$ uniones.
 - Rediseñe ahora la operación *Buscar* con la técnica de *compresión de caminos*. ¿Cuál es la mejora aportada por esta técnica?
3. (1 punto) Análisis de la complejidad de la búsqueda binaria: mejor caso y peor caso, aplicando el teorema de resolución de recurrencias divide y vencerás.
4. (1'5 puntos) Presente en una tabla los elementos característicos de los algoritmos voraces que pueda identificar en los algoritmos de Kruskal, Prim y Dijkstra.
5. (2 puntos) Dado el grafo dirigido pesado $G = (N, A)$, con $N = \{a, b, \dots, h\}$ y A determinado por la lista de aristas siguiente (se indica cada arista con su peso, o distancia):

(a,d)	(a,f)	(d,h)	(d,f)	(h,e)	(h,g)	(e,g)	(f,e)	(f,g)	(f,b)	(f,c)	(b,c)	(c,g)
1	2	1	4	2	6	2	1	3	4	5	2	1

Dibuje el grafo G , evitando el cruce de sus aristas, y a continuación dibuje:

- a)* un árbol asociado a un recorrido en profundidad sobre G , partiendo del nodo a
 - b)* un árbol expandido mínimo del grafo no dirigido subyacente, junto con su peso total
 - c)* un árbol con los caminos mínimos entre el nodo a y los demás, indicando en cada nodo la distancia mínima calculada desde el origen
 - d)* de nuevo el grafo G , pero con sus nodos ordenados topológicamente y en una misma línea
6. (2'5 puntos) Considerando un sistema monetario $M = \{v_1, v_2, \dots, v_m\}$, se dispone de una función Monedas que utiliza la técnica de Programación Dinámica para encontrar el número mínimo de monedas para pagar la cantidad n , calculando para ello una tabla T con todos los resultados intermedios (solución óptima para pagar la cantidad j con las monedas $v_1..v_i$). El resultado encontrado puede corresponder a una o varias configuraciones (conjuntos de monedas que suman n).

Se plantea el diseño de una función Composición que, a partir de la tabla T ya construida por la función Monedas, devuelva una configuración posible de la solución, especificando el conjunto de monedas que la componen:

- a)* Proponga un ejemplo del problema, presentando su tabla T y sus posibles soluciones.
- b)* Proponga un tipo de datos adecuado para la salida de la función Composición, que ilustrará con el ejemplo anterior.
- c)* Proponga un pseudocódigo de la función Composición.
- d)* Determine su complejidad.