Examen Mayo 2011, preguntas y respuestas

Algoritmos (Universidade da Coruña)

EXÁMENES

■ Determine de forma empírica la complej del alg cuya ejecuc produjo la sgte tabla de tiempos:

n	1000	2000	4000	8000	16000	32000	64000	128000
t(n)	7	15	35	81	175	377	815	1747
Compl.	7/1000 = 0'007	15/2000 = 0'00075	35/4000 = 0'0087	81/8000 = 0'0101	175/16000 = 0'0109	377/32000 = 0'0117	815/64000 = 0'012	1747/128000 = 0'0136

■ Con la sgte func hash:

hash ("Luis", 11) = 8 ; hash ("Carmen", 11) = 7 ; hash ("Mª", 11) = 8

hash ("Miguel", 11) = 7 ; hash ("Teresa", 11) = 8 ; hash ("Javier", 11) = 7

muestre el rdo de insertar en este orden las claves: "Luis", "Carmen", "Mª", "Miguel", "Teresa" y "Javier" en:

0	1	2	3	4	5	6	7	8	9	10

cuando se emplea:

→

a) exploración lineal:

0	1	2	3	4	5	6	7	8	9	10
Teresa	Javier						Carmen	Luis	Mª	Miguel

b) exploración cuadrática

0	1	2	3	4	5	6	7	8	9	10
Miguel	Teresa				Javier		Carmen	Luis	Mª	

Mª → 8 (Luis ; colisión) ⇒ $8 + 1^2 = (9)$

Miguel → 7 (Carmen ; colis) ⇒ $7 + 1^2 = 8$ (ocupado)

$7 + 2^2 = (11)$ $(11 \% 11 = 0)$

Teresa → 8 (Luis ; colis) ⇒ $8 + 1^2 = 9$ (ocup)

$8 + 2^2 = (12)$ $(12 \% 11 = 1)$

Javier → 7 (Carmen ; colis) ⇒ $7 + 1^2 = 8$ (ocup)

$7 + 2^2 = 11 = \not{0}$ (ocup)

$7 + 3^2 = (16)$ $(16 \% 11 = 5)$

c) explor. doble usando $5 - (x \bmod 5)$ como 2ª func, dnd

x = rdo de 1ª func de dispers.

0	1	2	3	4	5	6	7	8	9	10
	Teresa	Miguel			Javier		Carmen	Luis		Mª

Mª → 8 (ocup) → $5 - (8 \bmod 5) = 2$

Miguel → 7 (ocup) → $5 - (7 \bmod 5) = 3$

Teresa → 8 (ocup) → $5 - (8 \bmod 5) = 2$

Javier → 7 (ocup) → $5 - (7 \bmod 5) = 3$

9) A partir de la sgte estruct de datos para la implem. de conj disj:

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

y del sgte pseudocód para la unión de 2 conj:

```
procedimiento Unir (C, raíz 1, raíz 2)
    si raíz 1 < raíz 2 entnces C[raíz 2] := raíz 1
    sino C[raíz 1] := raíz 2
fin procedimiento
```

a) Escriba el corresp. pseudocód de Buscar (C, x) : Conj, q devuelve el nom del conj (ed, su repres) de 1 elem dado.

```
funcion Buscar (C, x) : Conj
r := x;
mientras C[r] <> r hacer
    r := C[r]
fin mientras
devolver r
fin funcion
```

b) Determine la complej de 1 sec de m búsq. y

n - 1 uniones.
Unión → O(n)
Búsq → O(m) y n búsq (O(nm))

} ⇒ O (m · n)

■ Explq el modelo computacional q consideramos para el cuál. de los alg indicando sus caract. ppales.

Calc. O para T(n) ≡ contar nº de "pasos" →

→ f(n) ? paso?

Características:

- Ordenación secuencial
- instrucción ↔ paso (no hay instrucc complejas: matrices...)
- entradas: tipo único ("entero") → acc(n)
- mem infinita + "td está en mem"

■ Determ cota exacta para la sol + gral de la relac. de recurrencia sgte:

$t(0) = 0$

$t(n) = 2.t(n-1) + n + 2^n$

← NO HOMOGÉNEA

$$\begin{cases} t_n \begin{cases} 0 & 0 \\ 2t(n-1)+n+2^n & \text{sino} \end{cases} \\ 2t_n - 2t + n + 2^n \end{cases}$$

$t_n - 2t_{n-1} = n + 2^n$ (polin grado 1)

Reducimos al caso homogéneo:

→ (-2), $n-1$ → $-2t_{n-1} + 4t_{n-2} = -2(n-1) - 2 \cdot 2^{n-1}$

→ (4), $n-2$ → $4t_{n-2} - 8t_{n-3} = 4(n-2) + 2^{n-2} \cdot 2^2$

→ Ec ini → $t_n - 2t_{n-1} = n + 2^2$

$t_n - 4t_{n-1} + 8t_{n-2} - 8t_{n-3} = (3n+6) + 2^n$

Enuncie y aplq el th Divide y vencerás para analizar el tmp de ejec. de la orderac. rép. a partir de su pseudocód, identificando el caso para el q es válido este análisis.

Alg. DIVIDE Y VENCERÁS:

- Divide el probl en 2 mitades, q se resuelven recursivam
- Fusiona las mitades ordenadas en 1 vector ordenado.
- Mejor: Orderación x fuserc para vectores peq (n < umbral)

Alg. Orderac Rép

```
procedimiento Qsort (var[i..j])
  si i+UMBRAL <= j entonces
     Mediana3 (T[i..j]);
     pivote := T[j-1] ; k := i ; m := j-1;
     repetir
        repetir k := k+1 hasta T[k] >= pivote;
        repetir m := m-1 hasta T[m] <= pivote;
        intercambiar (T[k], T[m]); {deshace últ cambio}
     hasta m < k;
     intercambiar (T[k], T[j-1]); {pivote en pos k}
     Qsort (T[i..k-1]);
     Qsort (T[k+1..j]);
  fin procedimiento
```

intercambiar (T[k], T[m])

en k está el pivote

→

procedimiento Quicksort (var T[1..n])

Qsort (T[1..n]);

OrdenacionPorInsercion (T[1..n])

fin procedimiento

* <u>Peor caso</u>: p es smpre el menor o el mayor elem.

$\Rightarrow T(n) = T(n-1) + cn \quad , n > 1$

$\Rightarrow \boxed{T(n) = O(n^2)}$

* <u>Mejor caso</u>: p coincide con la mediana

$\Rightarrow T(n) = 2T(n/2) + cn \quad , n > 1$

$\Rightarrow \boxed{T(n) = O(n \log n)}$

* <u>Caso medio</u>:

Sea i: tam de la parte izda;

c/valor posible para i (0..n-1) es equiprobable ($p = \frac{1}{n}$);

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

$\Leftrightarrow T(n) = \frac{2}{n} \left[\sum_{j=0}^{n-1} T(j) \right] + cn \quad , n > 1$

$\Rightarrow \boxed{T(n) = O(n \log n)}$

ALG 14

Se desea configurar la carga + valiosa posible para 1 mochila de capac. limit. en peso, a partir de n objetos fraccionables, caracterizados x su peso y valor, ambos estrict. posit. Diseñe el alg. voraz q resuelva el probl utilizando 1 montículo con estruct aux y determ. su complej. temporal.

función Mochila (w[1..n], v[1..n], W): objetos[1..n]
(w → peso; v → valor; W → capac mochila)

```
para i:= 1 hasta n hacer
    x[i] := 0
peso := 0;
```

} con → mont : CrearMont(X); O(n)

{bucle voraz} ⟶ O(log n) * n (peor caso)

```
mientras peso < W hacer
    i := mejor dem restante;
    si peso + w[i] ≤ W entonces
        x[i] := 1;
        peso := peso + w[i];
    sino
        x[i] := (W - peso) / w[i];
        peso := W
    devolver x
fun función
```

Complej. O(n log n)

■ Tanto el alg de Prim como de Dijkstra manejan en c/iter 1 información provisional con respecto a los nodos q aun no se han integrado al conj S de nodos para los q se ha determ. yá 1 sol. Identifiq en c/1 de estos alg cuál es dicha inform., y cm db actualizarse desp de incorporar 1 nuevo nodo al conj S.

La información provisional se refiere a las distancias mín. en el caso de Dijkstra. Desp de incorporar 1 nuev nodo, lo q db actualizarse es D, 1 vector q contiene las long. de caminos especiales mín. al final de la ejec

El alg de Prim maneja tb dist. mín. Lo q se modifica en c/iter es T, 1 árbol expandido mín del subgrafo (B,A) q contiene los nodos q conforman el recorrido mín.

Algoritmos
1 de diciembre de 2005

Apellidos:	
Nombre:	**Titulación (II / ITIG):**

1. *(2,5 puntos)* A partir de la siguiente estructura de datos para la implementación de *listas doblemente enlazadas*:

```
tipo
   PNodo = puntero a Nodo
   Nodo = registro
     Elemento : Tipo_de_elemento
     Siguiente : PNodo
     Anterior : PNodo
   fin registro
   Posición = PNodo
   Lista = PNodo
```

   a) Diseñe, escribiendo su pseudocódigo, las siguientes operaciones:

   CrearLista(L) que crea una lista vacía.

   Buscar(x, L) que devuelve la posición de la 1ª ocurrencia de x en la lista L, o *nil*.

   Eliminar(x, L) que elimina la 1ª ocurrencia de x de la lista L.

   Insertar(x, L, p) que inserta x en la lista L después de la posición p.

   Si utilizase algún procedimiento auxiliar, refleje también su pseudocódigo.

   b) Indique, justificando su respuesta, la complejidad computacional de cada una de las operaciones anteriores.

2. *(1 punto)* ¿Cómo se determina analíticamente la O de una construcción algorítmica iterativa?

3. *(1 punto)* Dibuje el árbol que represente la ejecución del algoritmo *divide y vencerás* que resuelve el problema de la suma de la subsecuencia máxima con la entrada siguiente:

3	-2	5	-3	-1	-1	7

   Cada nodo del árbol debe indicar:

   a) el vector correspondiente al parámetro de la llamada;

   b) el resultado de la llamada en la forma: $max(valor1, valor2, valor3)$ (o bien, un único valor en las hojas del árbol).

4. *(1 punto)* Enuncie el teorema de resolución de recurrencias divide y vencerás y aplíquelo para determinar la complejidad computacional del algoritmo de la pregunta anterior.

5. *(1,5 puntos)* Presente el pseudocódigo de un algoritmo que calcule las distancias de los caminos mínimos *desde cada uno de los n nodos* de un grafo dirigido ponderado hacia todos los nodos restantes. Su resultado debe ser una tabla *nxn* con las distancias de los caminos mínimos. Determine la complejidad computacional del algoritmo propuesto.

6. *(1 punto)* Construya la tabla con la que podría determinarse en programación dinámica la manera óptima de pagar una cantidad de 16 unidades de valor con un mínimo de monedas, sabiendo que el sistema monetario considerado está constituido por monedas de 1, 2, 8 y 12 unidades de valor. ¿Porqué descartaría el uso de la técnica voraz para resolver este problema?

DICIEMBRE 2005

A partir de la siguiente estructura de datos para la implementación de listas doblemente enlazadas:

```
tipo
    Puntero = puntero a Nodo
    Nodo = registro
              Elemento : TAD-de-elemento
              Siguiente : Puntero
              Anterior : Puntero
           fin registro
    Posición = Puntero
    Lista = Puntero
```

a- Diseñe, escribiendo el pseudocódigo, las siguientes operaciones:

CrearLista (L) que crea una lista vacía

Buscar (x, L) que devuelve la posición de la 1ª ocurrencia de x en la lista L, o nil.

Eliminar (x, L) que elimina la 1ª ocurrencia de x de la lista L.

Insertar (L, x, p) que inserta x en la lista L después de la posición p.

Si utilizase algún procedimiento auxiliar, refleje también su pseudocódigo

b- Indique, justificando su respuesta, la complejidad computacional de cada una de las operaciones anteriores.

* procedimiento Crear_Lista (L)

nuevo (tmp);

si tmp = nil entonces error Memoria agotada

sino

tmp^.Elemento := {nodo cabecera};

tmp^.siguiente := nil;

"tmp^.Anterior := nil";

L := tmp;

fin procedimiento

* función Buscar (x, L): posición de la 1ª ocurrencia o nil

p := L^.siguiente;

mientras "p <> L" y p^.Elemento <> x hacer

p := p^.siguiente;

si p^.Elemento = x entonces devolver p

sino devolver nil;

fin función.

* procedimiento Eliminar (x, L)

p := Buscar (x, L);

si p = nil entonces error No encontrado

sino

* procedimiento Insertar (L, x, p)

nuevo (tmp);

si tmp = NIL entonces error Memoria agotada

sino

tmp^.Elemento := x;

tmp^.Siguiente := p^.Siguiente;

tmp^.Anterior := p;

p^.Siguiente := tmp;

p^.Siguiente^.Anterior := tmp;

fin procedimiento

+ Complejidad computacional:

¿Cómo se determina analíticamente la O de una construcción algorítmica iterativa? ¿Es lo de la página 25? 7.1

Iteración: $T_{B,S}(n)) \wedge n^{\circ}$ iter = $O(f_{B,S}(n))$

⇒ Mientras B valga S' = $O(f_B(n)) * n^{\circ}$ iter(n))

Así el coste de las iteraciones no sería, sino: Σ costes individuales

(caso particular: para i ← x hasta y hacer S = $O(f_S(n) * n^{\circ}$ iter)

B es comparar 2 enteros = $T(1)$, n° iter = $y - x + 1$

Dibuje el árbol que representa la ejecución del algoritmo divide y vencerás que resuelve el problema de la suma de la subsecuencia máxima con la entrada siguiente:

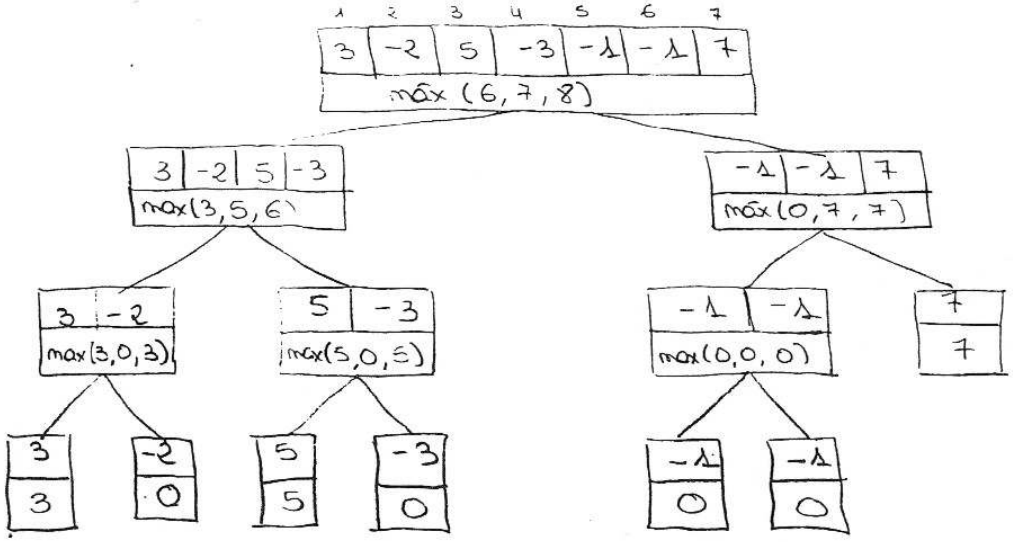
3	-2	5	-3	-1	-1	7

Cada nodo del árbol debe indicar:

a- El vector correspondiente al parámetro de la llamada

b- El resultado de la llamada en la forma: max (valor1, valor2, valor3)

(o bien, un único valor en las hojas del árbol).



Enuncie el teorema de resolución de recurrencias divide y vencerás y aplíquelo para determinar la complejidad computacional del algoritmo de la pregunta anterior.

$$T(n) = \begin{cases} \Theta(n^k) & \text{si } c < b^k \\ \Theta(n^k \log n) & \text{si } c = b^k \\ \Theta(n^{\log_b c}) & \text{si } c > b^k \end{cases}$$

* CM:

$T(n) = cT(n/b) + cn^k$, $n > n_0$,

con $c \geq 1$ $b \geq 2$ $c > 0 \in \mathbb{R}$ $n_0 \geq 0 \in \mathbb{N}$, $n \geq 1 \in \mathbb{N}$

Caso $c = b^k \Rightarrow T(n) = \Theta(n^k \log n)$ $(c=3, b=2), k=1) \Rightarrow T(n) = \Theta(n \log n)$

Presente el pseudocódigo de un algoritmo que calcule las distancias de los caminos mínimos desde cada uno de los n nodos de un grafo dirigido ponderado hacia todos los nodos restantes. Su resultado debe ser una tabla nxn con las distancias de los caminos mínimos. Determine la complejidad computacional del algoritmo propuesto.

NO se usa: se hace que el resultado sea una tabla nxn

Algoritmos
14 de febrero de 2005

Apellidos:		
Nombre:	**ITIG:**	**II:**

1. *(3 puntos)* Dado el algoritmo siguiente:

```
función K ( V[1..n], k: entero ): elemento
    CrearMonticuloMaximos( V, M );
    para i:=1 hasta k hacer
        x:= EliminarMayor( M )
    fin para
    devolver x
fin función
```

   a) Especifique el problema que resuelve este algoritmo.

   b) Diseñe las operaciones `CrearMonticuloMaximos` y `EliminarMayor` definiendo además todos los procedimientos auxiliares que sean necesarios.

   c) Determine la complejidad del algoritmo en función de n y de k.

2. *(1 punto)* Calcule a partir de su pseudocódigo, e indicando las reglas que utiliza, la cota exacta para el tiempo de ejecución de la *ordenación por selección*.

3. *(1 punto)* Determine una cota exacta para la solución más general de la recurrencia siguiente:

$$t_n = 2t_{n-1} + (n+7)3^n$$

4. *(1 punto)* A partir de la definición del concepto de *inversión* entre dos elementos de un vector desordenado, explique (sin incluir los algoritmos en su respuesta):

   a) cómo se determinan los casos en los que la *ordenación por inserción* presenta un comportamiento lineal, y

   b) en qué medida este resultado justifica la utilización que se hace de este algoritmo en la *ordenación rápida*.

5. *(1 punto)* Presente en una tabla las funciones características de los algoritmos voraces que identifique en:

   a) el algoritmo voraz para el *problema de la mochila* con objetos fraccionables,

   b) el algoritmo voraz para la *ordenación topológica*,

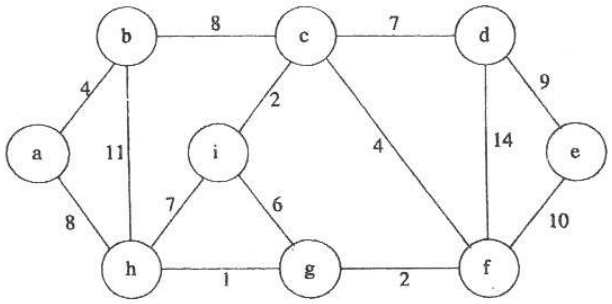
   c) el algoritmo de *Kruskal*.

6. *(1 punto)* Escriba un algoritmo que calcule un coeficiente binomial $C(n,k)$ utilizando la técnica de programación dinámica de tal forma que las necesidades de memoria sean mínimas. Determine la complejidad temporal y espacial del algoritmo propuesto.

Examen de Algoritmos
1 de septiembre de 2006

Apellidos:	
Nombre:	**Titulación (II / ITIG):**

1. *(1 punto)* Dibuje la *matriz de adyacencia* y las *listas de adyacencia* que representan el siguiente grafo no dirigido:



2. *(1 punto)* A partir de la siguiente estructura de datos 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 }
  si raíz1 < raíz2 entonces C[raíz2] := raíz1
  sino C[raíz1] := raíz2
fin procedimiento
```

   a) Escriba el correspondiente pseudocódigo de `Buscar (C, x) : Conj`, que devuelve el nombre del conjunto (es decir, su representante) de un elemento dado.

   b) Indique la complejidad de una secuencia de m búsquedas y $n-1$ uniones.

3. *(1 punto)* Calcule, a partir del pseudocódigo que incluirá en su respuesta, la cota exacta para el tiempo de ejecución de la *ordenación por selección*, indicando las reglas que utiliza.

4. *(1,5 puntos)* Uso del teorema *Divide y Vencerás*. Complete una tabla como la siguiente:

algoritmo	caso	caracterización	relación de recurrencia	resultado
BB	medio	$n_i = n/2$	$T(n/2)+1$	$\log n$
SSM	todos		$2T(n/2)+cn^1$	$n \cdot \log n$
MS	todos		$2T(n/2)+cn^1$	$n \log n$
QS	mejor	cuando $i = \frac{n}{2}$	$2T(n/2)+cn^1$	$n \log n$

Donde las columnas especifican:

- el *caso* en que se utiliza el teorema *Divide y Vencerás* (mejor caso, caso medio, peor, o todos);
- la *caracterización* del caso (¿con qué condiciones se produce, si las hay?);
- la *relación de recurrencia* que corresponde a este caso (sistema de ecuaciones);
- el *resultado* de la aplicación del teorema (complejidad calculada);

y en cada línea se considera un algoritmo diferente:

a) BB: búsqueda binaria;

b) SSM: algoritmo recursivo para el problema de la suma de la subsecuencia máxima;

c) MS: ordenación por fusión;

d) QS: ordenación rápida.

5. *(1,5 puntos)* Presente un pseudocódigo del algoritmo de *Kruskal* donde haga uso de un montículo como estructura auxiliar. Determine su complejidad temporal y explique lo que aporta el montículo para mejorar la eficiencia del algoritmo.

6. *(1 punto)* Construya la tabla con la que podría determinarse en programación dinámica la manera óptima de pagar una cantidad de 17 unidades de valor con un mínimo de monedas, sabiendo que el sistema monetario considerado está constituido por monedas de 1, 2, 8 y 12 unidades de valor. ¿Porqué descartaría el uso de la técnica voraz para resolver este problema?

1. Septiembre. 2006

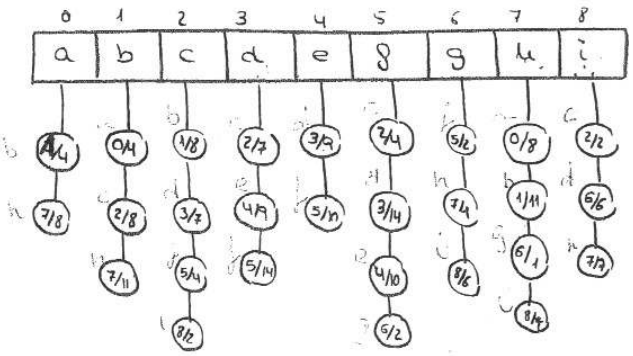
EJERCICIO 1

Matriz de adyacencia

	0	1	2	3	4	5	6	7	8
0	0	4	—	—	—	—	—	8	—
1	4	0	8	—	—	—	—	11	—
2	—	8	0	7	—	4	—	—	2
3	—	—	7	0	9	14	—	—	—
4	—	—	—	9	0	10	—	—	—
5	—	—	4	14	10	0	2	—	—
6	—	—	—	—	—	2	0	1	6
7	8	11	—	—	—	—	1	0	7
8	—	—	2	—	—	—	6	7	0

a=0
b=1
c=2
d=3
e=4
f=5
g=6
h=7
i=8

Lista de adyacencia



EJERCICIO 2

a..

```
funcion Buscar (C, x) : Conj
    r := x
    mientras r <> C[r] hacer
        r := C[r]
    fin mientras
    devuelver r
fin funcion
```

b.. Unir: $O(1)$

Buscar: $O(n)$

m busquedas y $n-1$ uniones $\Rightarrow O(m \cdot n)$

EJERCICIO 3

```
procedure OrdenacionSeleccion (var T[1..n])
    para i := 1 hasta n-1 hacer                                   (n-1-1+1)
    O(1)    minj := i                                             O(n)
    O(1)    minx := T[i]                                          =
                para j := i+1 hasta n hacer                       O(n²)
                    si T[j] < minx entonces                       (n-i+1+1)
                    O(1)    minj := j                             · O(1)
                    O(1)    minx := T[j]                          =
                    fin si                                        O(n)
                fin para
    O(1)    T[minj] := T[i]
    O(1)    T[i] := minx
    fin para
fin procedimiento
```

Examen de Algoritmos

5 de diciembre de 2006

Apellidos:	
Nombre:	**Titulación (II / ITIG):**

1. *(1,5 puntos)* Implementación de pilas a base de listas enlazadas. Partiendo de la declaración de tipos que sigue,

```
tipo
   PNodo = puntero a Nodo
   Nodo = registro
     Elemento : Tipo_de_elemento
     Siguiente : PNodo
   fin registro
   Pila = PNodo
```

   a) Escriba el pseudocódigo de las siguientes operaciones:

   CrearPila(p) que crea una pila vacía.

   EsPilaVacia(p) que comprueba si la pila está vacía.

   Meter(x, p) que inserta x en la pila.

   Cima(p) que devuelve el elemento de la cima de la pila.

   Sacar(p) que elimina la cima de la pila.

   Si utilizase algún procedimiento auxiliar, refleje también su pseudocódigo.

   b) Indique, justificando su respuesta sobre el pseudocódigo, la complejidad computacional de cada una de las operaciones anteriores.

2. *(1 punto)* ¿Cómo se determina analíticamente la O de una construcción algorítmica iterativa?

3. *(1 punto)* Análisis del peor caso de la ordenación de Shell con incrementos de Shell.

4. *(1 punto)* Presente en una tabla las *4 funciones características de los algoritmos voraces* que identifique en cada uno de los algoritmos siguientes:

   a) el algoritmo voraz para el *problema de la mochila* con objetos fraccionables,

   b) el algoritmo de *Kruskal*,

   c) el algoritmo de *Dijkstra*.

5. *(1,5 puntos)* Presente un pseudocódigo del algoritmo de *Prim* donde haga uso de alguna estructura auxiliar sencilla para manejar la información provisional asociada a cada nodo. Determine su complejidad temporal y explique en qué condiciones sería un algoritmo eficiente.

6. *(1 punto)* Describa cómo podría calcular un coeficiente binomial $C(n,k)$ utilizando la técnica de programación dinámica de tal forma que las necesidades de memoria sean mínimas. Determine la complejidad temporal y espacial del método propuesto.

EXAMEN DICIEMBRE 2006

Implementación de pilas a base d listas enlazadas

tipo

PNodo puntero a Nodo

Nodo = registro

Elemento : tipo-de-elemento

Siguiente : PNodo

fin registro

Pila = PNodo

a) Pseudocódigo de

CrearPila (P) q crea una pila vacía

procedimiento CrearPila (P) → O(1)

P^.elemento = {nodo cabecera} → O(1) → new

P^.siguiente = nil → O(1)

fin procedimiento

EsPilaVacia (P) q comprueba si la pila está vacía

función EsPilaVacia (P) : test → O(1)

devolver P^.siguiente = nil → O(1)

fin función

Meter (x, P) q inserta x en la pila O(n)

procedimiento Meter (x, P)

new (nuevo); → O(1)

mientras P^.siguiente <> nil hacer } O(n)

P = P^.siguiente

fin mientras

P^.siguiente = nuevo } O(1)

nuevo^.elemento = x

nuevo^.siguiente = nil

fin procedimiento

Cima (P) q devuelve el elemento de la cima de la pila

```
función Cima (P): tipo_de_elemento   O(n)
    si EsPilaVacia (P) entonces
        error "Pila Vacia"
    sino
        mientras P^.siguiente <> nil hacer   } O(n)
            P = P^.siguiente
        fin mientras
    devolver p^.elemento
fin función
```

Sacar (P) que elimina la cima de la pila O(n)

```
procedimiento Sacar (P)
    si EsPilaVacia (P) entonces
        error Pila Vacia
    sino   pos = BuscarAnterior (Cima(P), P)   -> dispose
           pos^.siguiente = nil
fin procedimiento
```

BuscarAnterior (x, P) q devuelve la posición del elemento anterior o nil

```
función BuscarAnterior (x P): PNodo -> O(n)
    pos = P
    mientras pos^.siguiente <> nil y pos^.siguiente.elemento <> x
        hacer   pos = pos^.siguiente
    fin mientras
    devolver pos
fin función
```

Por lo tanto, el trabajo para colocar los menores sería

$$\sum_{i=1}^{n/2} i - 1 = \Omega(n^2)$$

(2) ¿$O(n^2)$?

Trabajo realizado en la iteración K con el incremento h_K: son h_K ordenaciones por Inserción sobre n/h_K elementos cada una de ellas

Es por tanto

$$h_K \cdot O\left((n/h_K)^2\right) = O\left(h_K \cdot n^2/h_K^2\right) = O\left(n^2/h_K\right)$$

Con el conjunto de todas las iteraciones del algoritmo

$$T(n) = O\left(\sum_{i=1}^{t} n^2/h_i\right) = O\left(n^2 \cdot \sum_{i=1}^{t} 1/h_i\right) = O(n^2)$$

4 Presente en una tabla las 4 funciones características de los algoritmos voraces y identifique a:

a) Mochila b) Kruskal c) Dijkstra

	Mochila	Kruskal	Dijkstra
¿S es solución?		S será solución cuando sólo tengamos una componente conexa	S será solución cuando este conjunto sea igual al de todos los nodos del grafo.
Factible	La solución será factible cuando el peso de la misma sea menor o igual a la capacidad de la mochila	Será factible cuando en nuestro conjunto solución no ∃ ciclos	
Selección	En cada paso iremos seleccionando un objeto. Si su peso más el de todos los elementos q ya están en la mochila es menor o igual q la capacidad de la misma, entonces lo podremos entero; en caso contrario lo partiremos	Luego de ordenar las aristas de modo creciente las elegimos en ese orden y vamos fusionando componentes conexas cuando sea cada (caso en el q los nodos que son unidos por la arista elegida no pertenecan a la misma componente conexa	Partimos de un nodo en concreto. A partir de este, elegiremos aquel cuya distancia del nodo d partida a él sea mínima
Objetivo	Obtener la mochila de manera que se maximice el valor de los objetos transportados con la restricción de la capacidad	Conseguir el árbol expandido mínimo (conjunto de aristas q conectan a todos los nodos y que el peso de ellas sea el mínimo posible)	obtener la longitud del camino mínimo desde un nodo a todos los demás

2 ¿Cómo se determina cualitativamente la O de una construcción algorítmica iterativa?

B; S = $O(t_B, s(n))$ ∧ mientras iter = $O(f_{iter}(n))$

mientras B hacer S = $O(t_B, s(n) * f_{iter}(n))$ si y sólo si el coste de las iteraciones no varía. En caso de que el coste sea diferente será la suma de todos los costes individuales (Σ costes)

3 Análisis del peor caso de la ordenación de Shell con incrementos de Shell.

Los incrementos de Shell consiste en ir dividiendo el incremento a su mitad hasta q este sea 1.

En un principio, incremento es igual al número de elementos, una vez q entramos en el bucle principal, iremos dividiendo incremento entre 2 sucesivamente.

Pero con estos incrementos no conseguimos bajar de un tiempo cuadrático $\Theta(n^2)$. Demostración

(1) ¿$\Omega(n^2)$?

Supongamos q el nº d elementos es una potencia de 2, por lo que todos los incrementos serán pares excepto el último q será 1.

Vamos a ponernos en la peor situación: q los mayores elementos estén en posiciones pares y los menores en impares

Ejemplo (el más favorable)

1	9	2	10	3	11	4	12	5	13	6	14	7	15	8	16

Está 8,4 y 2-ordenado, todo el trabajo queda en 1-ordenar

El i-ésimo menor está en la posición $2i-1$, $i \leq n/2$. Para llevarlo a su lugar correspondiente, tendríamos que hacer $i-1$ desplazamientos; por ejemplo para que el 5 esté en su posición correspondiente tendríamos q hacer 4 desplazamientos

5 Pseudocódigo de Prim2

```
función Prim2 (L[1..n, 1..n]) : árbol
  Distancia Mínima[1] = -1
  T = conjunto vacío
  para i := 2 hasta n hacer
    Más Próximo[i] := 1
    Distancia Mínima[i] = L[i, 1]
  repetir n-1 veces
    min = infinito;
    para j := 2 hasta n hacer
      si 0 <= Distancia Mínima[j] < min entonces
        min = Distancia Mínima[j]
        k := j
    fin para
    T = T ∪ {(Más Próximo[k], k)};
    Distancia Mínima[k] = -1;
    para j := 2 hasta n hacer
      si L[j,k] < Distancia Mínima[j] entonces
        Distancia Mínima[j] := L[j,k];
        Más Próximo[j] := k;
    fin para
  fin repetir
  devolver T
fin función
```

Prim es mejor en el caso en el que tratemos con un grafo denso, ya que obtenemos un tiempo que está en $\Theta(n^2)$ frente a Kruskal $\Theta(n^2 \cdot \log n)$

6 Coeficientes binomiales con programación dinámica de forma q las necesidades de memoria sean mínimas. Det. complejidad temporal y espacial.

El cálculo de un coeficiente binomial es:

$$\binom{n}{k} = \begin{cases} 1 & \text{si } k=0 \text{ o } n=k \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{si } 0<k<n \\ 0 & \text{en otro caso} \end{cases}$$

Supongamos que $0 \leq k \leq n$; podmos calcularlo aplicando la siguiente función:

```
función C(n,k) : número
  si n=0 o n=k entonces devolver 1
  sino devolver (C(n-1,k-1) + C(n-1,k))
fin función
```

pero entonces muchos valores serán calculados varias veces de manera innecesaria.

Si utilizamos una tabla de resultados intermedios (se trataría del triángulo de Pascal), tendríamos un algoritmo más eficiente. Esta tabla la deberíamos ir rellenando línea por línea. Pero incluso no es necesario mantener llena toda la tabla, basta con mantener un vector que nos representa la línea actual e ir rellenando dicho vector de derecha a izquierda. De este modo, para calcular $\binom{n}{k}$ necesitamos un tiempo que está en $O(nk)$ y un espacio que esté en $\Theta(k)$ si suponemos que la suma es una operación elemental

Examen de Algoritmos
febrero de 2008

Apellidos:	
Nombre:	**Titulación (II / ITIG):**

1. *(0,5 puntos)* Determine de forma empírica la complejidad de un algoritmo a partir de la siguiente tabla de tiempos:

n	t(n)
500	0,219
1000	0,526
2000	1,238
4000	2,990
8000	7,021
16000	16,903

2. *(1,5 puntos) Montículos*:

   a) Presente el contenido de un montículo de máximos des
   la siguiente secuencia de claves: 5, 1, 7, 2, 9, 8, 6, 3 (al pr

   b) Escriba el pseudocódigo del procedimiento `Insertar` t
   tos para la implementación de montículos en base a vect
   de todos los procedimientos auxiliares necesarios.

```
tipo Montículo = registro
                   Tam_mont : 0..Tam_max
                   Vec_mont : vector [1..
                 fin registro

procedimiento Inicializar_Montículo ( M )
  M.Tam_mont := 0
fin procedimiento
```

   c) Determine la complejidad del procedimiento `Insertar`. Razone la respuesta.

3. *(1 punto)* Compare la *ordenación por fusión* con la *ordenación rápida* en los siguientes aspectos:

   a) Método para crear las subinstancias del mismo problema (*dividir*).
   b) Número y tamaño de las subinstancias.
   c) Método para calcular la solución a partir de las soluciones parciales (*vencer*).
   d) Estrategia para resolver los casos base de la función recursiva.
   e) Cálculo de la complejidad.

4. *(1 punto)* La *función objetivo*, característica de los algoritmos voraces, está asociada al problema de optimización que se quiere resolver y determina la *función selección* que se utilizará en el bucle voraz. Identifique estas dos funciones en cada uno de los problemas siguientes:

   a) Problema de devolver el cambio.
   b) Problema de la mochila con objetos fraccionables.
   c) Árbol expandido mínimo.

5. *(1 punto)* Dado un recipiente, de capacidad $W = 15$ unidades de peso, y el conjunto siguiente de objetos, caracterizados por su valor (v) y su peso (w):

objeto	1	2	3	4	5
v	4	6	6	5	5
w	3	4	8	5	6

Determine, en cada uno de los casos siguientes, la carga más valiosa posible para este recipiente, a partir de una estructura de datos auxiliar que presentará con todo su contenido inicial, y explicando paso a paso las decisiones tomadas en la selección de los objetos.

a) Los objetos se pueden fraccionar.

b) Los objetos *no* se pueden fraccionar.

6. *(0,5 puntos)* Escriba el pseudocódigo de un algoritmo que permita realizar un *recorrido en anchura* a partir de un nodo cualquiera v de un grafo.

Examen de Algoritmos
septiembre de 2008

Apellidos:	
Nombre:	**Titulación (II / ITIG):**

1. *(1,5 puntos)* Colas: Diseñe, escribiendo su pseudocódigo, los algoritmos `insertar`, `quitarPrimero` y `primero` de modo que todas estas rutinas se ejecuten en tiempo constante. Realice la implementación en base a vectores reflejando en el diseño las estructuras de datos necesarias.

2. *(1 punto)* Calcule, a partir del pseudocódigo que incluirá en su respuesta, la cota exacta para el tiempo de ejecución de la *ordenación por selección*, indicando las reglas que utiliza.

3. *(1 punto)* Compare la *ordenación de Shell* con la *ordenación rápida* desde el punto de vista de su complejidad.

4. *(1,25 puntos)* Identifique los elementos característicos de los algoritmos voraces en el pseudocódigo del algoritmo de Prim, que incluirá en su respuesta.

5. *(1,25 puntos)* Construya la tabla con la que podría determinarse en programación dinámica la manera óptima de pagar una cantidad de 17 unidades de valor con un mínimo de monedas, sabiendo que el sistema monetario considerado está constituido por monedas de 1, 3, 8 y 12 unidades de valor. Indique la solución al problema explicando cómo la obtiene a partir de la tabla anterior. ¿Por qué descartaría el uso de la técnica voraz para resolver este problema?

6. *(1 punto)* Represente mediante un grafo decorado todas las situaciones de juego que podrían alcanzarse a partir de un montón de 5 palillos para la variante del *juego de Nim* vista en clase.

Examen de Algoritmos
Septiembre de 2010

Apellidos:	
Nombre:	**Titulación (II / ITIG):**

1. *(1.5 p) Montículos*:

   a) Presente el contenido de un montículo de máximos después de la llamada a `crear_monticulo` con un vector que contiene la siguiente secuencia de claves: $4, 1, 5, 2, 9, 8, 6, 7$.

   b) Desarrolle el pseudocódigo de la operación `crear_monticulo` partiendo de la siguiente declaración para la implementación de montículos en base a vectores.

```
tipo Monticulo = registro
    Tam : 0..Tam_max
    Vec : vector [1..Tam_max] de Tipo_elem
fin registro

procedimiento Inicializar_Monticulo ( M )
    M.Tam := 0
fin procedimiento
```

   Presente también el pseudocódigo de todos los procedimientos auxiliares necesarios.

   c) Indique, razonando su respuesta sobre el pseudocódigo, la complejidad computacional de la operación anterior.

2. *(1 p)* Explique cómo determinar analíticamente la O de las construcciones algorítmicas iterativas.

3. *(1 p)* Compare la *ordenación por fusión* con la *ordenación rápida* en los siguientes aspectos:

   a) Método para crear las subinstancias del mismo problema (*dividir*).
   b) Número y tamaño de las subinstancias.
   c) Método para calcular la solución a partir de las soluciones parciales (*vencer*).
   d) Estrategia para resolver los casos base de la función recursiva.
   e) Cálculo de la complejidad.

4. *(1 p) Ordenación de Shell con incrementos de Hibbard:*

   a) Complete la siguiente tabla:

	4	3	5	1	2	8	7	3	6	1	5	2	9
h_3													
h_2													
h_1													

   b) Indique la secuencia de incrementos $\{h_3, h_2, h_1\}$ utilizada en la tabla.

   c) Justifique la propuesta de incrementos distintos a los de Shell.

5. *(1 p)* Identifique los elementos característicos de los algoritmos voraces en el pseudocódigo del algoritmo de Kruskal, que incluirá en su respuesta.

6. *(1 p)* Construya la tabla con la que podría determinarse en programación dinámica la manera óptima de pagar una cantidad de 16 unidades de valor con un mínimo de monedas, sabiendo que el sistema monetario considerado está constituido por monedas de 1, 3, 8 y 12 unidades de valor. Indique la solución al problema explicando cómo la obtiene a partir de la tabla anterior. Presente la solución que daría la técnica voraz para resolver este problema y extraiga conclusiones.