FE DE ERRATAS

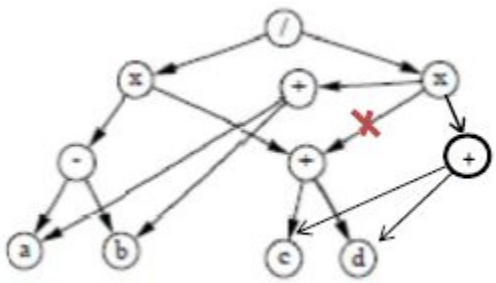
<table>
<tr><td>Pág. 11</td><td>En el primer párrafo, donde dice “En los dos grafos ejemplo, ...” debe decir: “ En el grafo de la figura 2.1 los nodos adyacentes al 6 son el 1 y el 4“

Al final del punto 2.1.1, donde dice “Sin embargo, el grafo de la figura 2.2 no tiene ningún ciclo” debe decir: “Sin embargo, el grafo de la figura 2.2 tiene un bucle”</td></tr>
<tr><td>Pág. 12</td><td>En la definición de subgrafo, donde dice “G’ es un subgrafo de G si N(G’)...” debe decir: “G’ es un subgrafo de G si siendo un grafo, N(G’)...”

En el 2º párrafo donde dice “El grafo dirigido de la figura 2.3a no es conexo ...” debe decir “El grafo de la figura 2.3a es conexo, pero no es fuertemente conexo ya que no hay camino ...”</td></tr>
<tr><td>Pág. 19</td><td>En el tercer párrafo a las operaciones típicas del TAD pila hay que añadir: Vacia.
El algoritmo de recorrido en profundidad iterativo queda:

<pre>
fun RecProfundidadIterativo(v:nodo,visitado: Vector)
var
   u,w:nodo
fvar
P ← PilaVacia
visitado[v]← cierto
Apilar(v,P)
mientras ¬ Vacia(P) hacer
 u = Cima(P)
 Desapilar(P)
 para cada w adyacente a u hacer
  si ¬ visitado[w] entonces
    visitado[w]← cierto
    Apilar(w,P)
  fsi
 fpara
fmientras
ffun
</pre></td></tr>
<tr><td>Pág. 21</td><td>En el párrafo anterior al apartado “Recorrido en amplitud o en anchura” hay que aclarar el orden en el que se apilan los nodos en el recorrido iterativo. El párrafo quedaría:
“En el grafo ejemplo de la figura 2.6a el orden en que se visitan y apilan los nodos debe ser en orden numérico decreciente con el fin de que coincida con el orden en el que son llamados por la función RecProfundidadRecursivo() y que aparecen la figura 2.6b: ...”</td></tr>
</table>

Pág. 22	En el 2º párrafo a las operaciones típicas del TAD cola hay que añadir: Vacia.
Pág. 25	En párrafo anterior al punto 2.1.6 dice: " En el caso de grafos dirigidos …" y debería decir " En el caso de grafos no dirigidos …"
Pág. 26	En el último párrafo, donde dice "El valor de bajo(3) es 3 ya que … que no esté en el grafo " debe decir "El valor de bajo(3) es 3 ya que … que no esté en el árbol "

Pág. 28	El algoritmo de Recorrido en Profundidad en Orden Topológico queda como sigue:
	
	`fun RecProfundidadRecursivoOrdenTopologico(v: nodo, visitado: Vector)`
	`var`
	`  w:nodo`
	`fvar`
	`visitado[v]← cierto`
	`para cada w adyacente a v hacer`
	`  si ¬ visitado[w] OR ¬ recorrida_arista[v,w]`
	`  marcar_visitada(v,w)`
	`  RecProfundidadRecursivoOrdenTopologico(w, visitado)`
	`  fsi`
	`fpara`
	`escribir(v)`
	`ffun`
	
	La figura 2.13 debe ser:
	 

Pág. 29	En el segundo párrafo, cuando describe la salida del algoritmo con el grafo de la figura 2.14 debería poner: "Si lo recorriéramos siguiendo la función anterior empezando por el nodo T1 escribiría: T6 T5 T3 T1 T3 T5 T4 T2". La siguiente frase: "Al invertir la lista anterior tendríamos un orden que garantizaría la ejecución de todas la tareas teniendo en cuenta las precedencias" no debe ser tenida en cuenta.
Pág. 30	En la sección 2.2 en la primera de las propiedades, donde dice "el número de elementos es impar" debe decir "el número de elementos es par"
Pág. 32	La figura 2.19 no es correcta y debe ser igual que la 2.17

Pág. 34	El algoritmo queda como sigue:
	
	`fun Flotar(T: vector, i:natural)`
	
	`// el nodo padre de i es i div 2`
	`mientras (i>1) y (T[i div 2]<T[i]) hacer`
	`  intercambiar(T[i],T[i div 2])`
	`  i ← i div 2`
	`fmientras`
	`ffun`

Pág. 34	En la explicación de la función MonticuloVacio? se debe indicar: "La función comprueba si el montículo tiene elementos y devuelve *falso* si tiene alguno, o *cierto* si está vacío."
Pág. 34	Donde pone: **fun** MonticuloVacio?(m:monticulo) debe poner: **fun** MonticuloVacio?(m:monticulo):bool

Pág. 36	En el algoritmo *hundir* la sentencia **repetir** debe estar a continuación del **fvar** y no a continuación de *p* **<-** *i*
	
	`fun Hundir (T: vector, i:natural )`
	` var`
	` hi,hd,p:natural`
	` fvar`
	` repetir`
	` hi <- 2*i`
	` hd <- 2*i + 1`
	` p <- i`
	` si ( hd <= m.c) && (T[hd] > T[i]) entonces`
	` i <- hd`
	` fsi`
	` si ( hi <= m.c) && (T[hi] > T[i]) entonces`
	` i <- hi`
	` fsi`
	` intercambiar(T[p], T[i])`
	` hasta p=i;`
	`ffun`

Pág. 38	Donde dice "coste =(1)" debe decir "coste O(1)"
Pág. 40	En el algoritmo CreaMonticulo(); entre fpara y dev(m) debe incluirse:

	`m.T ← T` `m.c ← n`
Pág. 40	En el algoritmo CreaMonticuloLineal después de `m.T ← T` debe incluirse: `m.c ← n`
Pág. 43	Donde dice “un árbol binomial de orden *k* contendrá *2k* nodos” debe decir “…contendrá 2^k nodos”
Pág. 43	Donde dice: Los hijos son siempre menores (mayores) o iguales que el padre para un montículo de mínimos (máximos) debe decir: Los hijos son siempre mayores (menores) o iguales que el padre para un montículo de mínimos (máximos)
Pág. 43	En el título de la figura 2.23 donde dice M0, M1 y M2 debe decir M0, M2 y M3
Pág. 49	En la Función Cuadrado, en la primera línea del párrafo donde dice “elimina” debe decir “escoge”.
Pág. 50	En el resultado de la primera operación XOR entre ‘U’ y ‘N’ donde dice 00011111 debe decir 00011011 y el resultado final 00011010
Pág. 52	En el cálculo de doble hashing, la expresión debe ser: $dir = (h(k) + ch'(k)) \bmod m$
Pág. 63	En donde dice “Mientras que la solución óptima sería tres monedas ...” debe decir “Mientras que la solución óptima sería dos monedas de 15 y cinco de 1, que hacen un total de 7 monedas. ” En el párrafo que empieza “En cuanto al coste del algoritmo MonedasCambio …” debería decir "se ejecutan tantas veces como el número de tipos de moneda, n, y ambos contienen instrucciones de coste constante, por lo que el coste global está en O(n)".
Pág. 77	En donde dice “Entonces, habrá al menos dos enteros” debe decir “Entonces, habrá al menos dos naturales”
Pág. 81	En los 2 puntos que explican las 2 situaciones que se pueden dar al tratar de retrasar la tarea a en SI: Texto original:

	“ ● Si hay un hueco en la secuencia *SI* en la unidad de tiempo t_I se pasa la tarea *a* a dicho hueco. ● Si ya hay una tarea *b* planificada en ese tiempo t_I en *SI*, se modifica *SI* intercambiando las tareas *a* y *b* en dicha secuencia” Texto corregido: “ ● Si hay un hueco en la secuencia *SI* en la unidad de tiempo t_J se pasa la tarea *a* a dicho hueco. ● Si ya hay una tarea *b* planificada en ese tiempo t_J en *SI*, se modifica *SI* intercambiando las tareas *a* y *b* en dicha secuencia” es decir se cambia t_I por t_J
Pág. 89 y 90	Problema de la Mochila con Objetos Fraccionables, al final de la página 89 y principio de la 90: Donde dice: “tomaría una fracción calculando el peso que queda disponible en la mochila (20-15) y dividiéndolo por el peso del objeto que es 15. Esto nos da un valor de 0.33 ...” Texto corregido: “tomaría una fracción calculando el peso que queda disponible en la mochila (20-10) y dividiéndolo por el peso del objeto que es 15. Esto nos da un valor de 0.66 ...”
Pág. 90	Algoritmo MochilaObjetosFraccionables, después de fvar: Donde dice: “Ordenar objetos en orden no decreciente de ” Texto corregido: “Ordenar objetos en orden no creciente de ” Donde dice: “peso + p[i] <= W” Texto corregido: “peso + p[i] <= M”
Pág. 106	En el algoritmo de búsqueda binaria, donde dice "si v[m] ≤ x entonces" debe decir "si x ≤ v[m] entonces"
Pág. 106	En la traza del algoritmo de búsqueda binaria donde dice: Bbinaria(1,7,(1,3,8,12,13,32,56),32) con m = 4 Bbinaria(5,7,(-,-,-,-,13,32,56),32) con m = 6 Bbinaria(5,6,(-,-,-,-,13,32,-),32),x) Bbinaria(1,n,v,x) debe decir:

	Bbinaria(1,7,(1,3,8,12,13,32,56),32) con m = 4 Bbinaria(5,7,(-,-,-,-,13,32,56),32) con m = 6 Bbinaria(5,6,(-,-,-,-,13,32,-),32) con m = 5 Bbinaria(6,6,(-,-,-,-,-,32,-),32) caso trivial
Pág 109	Donde dice U[m+1],V[n+1]←∞ debe decir U[n+1],V[m+1]←∞
Pág. 110	Debajo de la ecuación de recurrencia donde pone k=0 debe poner k=1.
Pág. 112	Donde dice “trimino” debe decir “tromino”
Pág. 114	Donde dice: [3, 4, 1, 5, 6, 9] debe decir: [3, 1, 4, 5, 9, 6]
Pág. 114	Donde dice: fun Pivotar(T:vector [i..j] de entero) debe decir fun Pivotar(T:vector [i..j] de entero, pivote: natural)
Pág. 114	Donde dice: intercambiar(T,i,l) ffun debe decir intercambiar(T,i,l) pivote ← l ffun

Pág. 114	Donde dice: fun Quicksort(T[i..j]) debe decir fun Quicksort(T[i..j]) var l:natural fvar
Pág. 114	Donde dice “o que requiere un tiempo lineal” debe decir “lo que requiere un tiempo lineal”
Pág. 115	Donde dice dev Combinar(s_1,s_2) debe decir dev Combinar(s_1,s_2,v)
Pág. 116	En la función ComprobarMayoritario donde dice “si c>n entonces” debe decir “si c > n/2 entonces”
Pág. 116	Penúltima línea debe ser k=1 (en lugar de k=0)
Pág. 118	Donde dice: Combinar(i,j,d+1) debe decir Combinar(i,j,d)
Pág. 120	Donde pone: n ← j-i+1 para s ← 0 hasta n-1 hacer para t ← 0 hasta n-1 hacer a ← i+t b ← (m+s+1) mod (j+1)

	T[a,b] ← d+s fpara fpara ffun debe poner: n ← j-i+1 para s ← 0 hasta n/2-1 hacer para t ← 0 hasta n/2-1 hacer a ← i+t b ← m+1+ ((t+s) % (n/2)) T[a,b] ← s+d+n/2-1 fpara fpara ffun
Pág. 123	En la condicion del primer SI..ENTONCES donde dice menor o igual debe decir menor.
Pág. 123	La función ExtraeOrdenadas() debe llamarse ExtraeAbcisa()
Pág. 129	En la función FibDin t: tabla[0..1] de entero debe ser t: tabla[0..n] de entero El final de la función FibDin debe ser: fpara dev t[n] fsi ffun
Pág. 130	El final de la función FibDin2 debe ser: fpara dev suma fsi ffun
Pag. 131	En el primer árbol los números combinatorios bajo el segundo (2 sobre 1) debe ser (1 sobre 0) y (1 sobre 1). En el segundo árbol, los números combinatorios de la segunda fila debe ser (1 sobre 0) y (1 sobre 1) (en lugar de (0 sobre 0) y (1 sobre 0)).
Pag. 132	- La declaracion de t es:

	t: matriz[0..n,0..k] de entero - El bucle **para** más interno va de 2 a i-1. - Al final de la instrucción **sino** (entre el **fpara** y el **fsi**) falta **dev** t[n,k]
Pag. 133	- penúltimo párrafo, segunda línea para cantidad posible entre 1 y C. debe ser: para cantidad posible entre 0 y C. - penúltimo párrafo, líneas 4 y 6: 0 <= i <= N debe ser 1 <= i <= N
Pag. 134	Los x1 de las filas 2 y 3 de la tablas deben ser x2 y x3 respectivamente
Pag. 135	- tipo Tabla = matriz[1..N,1..C] de entero debe ser: tipoTabla = matriz[1..N,0..C] de entero - Al final de la página debe sustituirse: Si **no** es así no podemos utilizar esa moneda y por tanto, el valor de esa casilla no cambia respecto del de la fila anterior. Si es **menor**, entonces... por: Si es así no podemos utilizar esa moneda y por tanto, el valor de esa casilla no cambia respecto del de la fila anterior. Si es mayor, entonces...
Pag. 139 y 140	T debe aparecer como primer parámetro en la declaración y en las llamadas a MinMultiple y MinMultiple2
Pag. 140	En la función MinMultiple2 sobra la declaración de la variable tmp y la instrucción: tmp ← minimo
Pag, 143	En la tabla, fila 4, columna 8, debe ser 17 en lugar de 15.
Pag. 147	A la función EscribeParentizado le falta el caso base. Debería ser si i = j entonces Imprimir “M”, Imprimir i sino k <- pos[i,j] Imprimir “(” EscribeParentizado(pos,i,k) EscribeParentizado(pos,k+1,j) Imprimir “)” fsi

Pag. 149 y 151	En los dos algoritmos de Floyd y en VerRutas los bucles para empiezan en 1 en lugar de en 0
Pag. 155	En el algoritmo identificarTrans falta la inicialización de las variables i y j: i <- n j <- m
Pag. 161 Pag. 162	**fmientras** **fsi** debe ser **fsi** **fmientras**
Pag 168	En el algoritmo ColoreaGrafo debe usarse v[k] en lugar de v[k+1] y Completable(v) debe ser Completable(g,v,k): v[k] <- 0 exito <- falso mientras v[k] < m /\ ¬ exito hacer v[k] <- v[k] + 1 si Completable(g,v,k) entonces si k = N entonces procesar(v) exito <- cierto sino ColoreaGrafo(g,m,k+1,v,exito) fsi fsi fmientras
Pag. 169	En la última línea del primer párrafo, w[k+1] debe ser v[k].
Pag. 169	En el párrafo anterior a la sección 6.3, donde dice “por lo que una cota al coste del algoritmo es O(m^n)”. debe decir: por lo que una cota al coste del algoritmo es O(n.m^n), el número de nodos del árbol por el coste n de la función Completable que se invoca para cada uno.
Pag. 172	Una cota más ajustada del coste del algoritmo de los ciclos Hamiltonianos es O(n!) (por las mismas razones que en el viajante de comercio).
Pag. 173	En el algoritmo, sobra la variable i y además si k < n entonces debe ser si k<= n entonces

Pag. 176	función Completable debe ser: si sumaParcial + x[k] <= sumaTotal div 2 entonces dev cierto sino dev falso
Pag. 177	La llamada a DividirSociedad debe tener un 0 en lugar de un 1 como argumento k, y en la penúltima posición de la llamada, y v debe estar al final de la llamada: DividirSociedad(x,suma1,suma2, sumaTotal,0,v)
Pag. 178	La declaración de las matrices debe ser: tipo TEdificio = matriz[1..LARGO,1..ANCHO] de caracter tipo TEdificioB = matriz[1..LARGO,1..ANCHO] de booleano
Pag. 179	En la función Caminos, en todos los sitios donde dice hijos <- Añadir(solución, casilla_aux) debe ser hijos <- Añadir(hijos, casilla_aux)
Pag. 186	En el algoritmo los tres signos **<** deben ser <= Además, **si** EstimaciónOpt(hijo) < cota **entonces** Insertar(hijo, monticulo) **fsi** debe ser **si** EstimaciónOpt(hijo) <= cota **entonces** Insertar(hijo, monticulo) **si** EstimacionPes(hijo) < cota **entonces** cota <- EstimacionPes(hijo) **fsi** **fsi**
Pág. 188	Condición bucle **mientras**: mientras ¬Monticulovacio?(monticulo) y EstimacionOpt(Primero(monticulo)) > cota debería ser mientras ¬Monticulovacio?(monticulo) y EstimacionOpt(Primero(monticulo)) **>=** cota
Pag. 189	**si** valor < hijo.valorT **entonces** debe ser

	si valor <= hijo.valorT **entonces**
Pág. 189	Eliminar el **fsi** que hay inmediatamente antes de cada una de las dos líneas "**sino** {la solucion no es completa}"
Pág. 193	EstOpt de N7 debe ser 28.625 (en lugar de 30.875)
Pág. 194	El siguiente nodo que se explora es N10 debería ser N4 que tiene una estimación optimista mejor de 27.625.
Pág. 195	"pasteleros:12435. Esta asignación supone un coste total de 29" debería ser "pasteleros:12435. Esta asignación supone un coste total de 26"
Pag 196 pag 197	Los bucles **para** deben empezar en 1 en lugar de en 0
Pag 197	**fsi** hijo.asignados[i] <- falso {se desmarca} **fpara** debe ser hijo.asignados[i] <- falso {se desmarca} **fsi** **fpara** (la asignación va dentro de la instrucción **si)**
Pág. 197	hijo.estOpt<-- EstimacionOpt(costes,pedido,hijo.k,nodo.costeT) debería ser: hijo.estOpt<-- EstimacionOpt(costes,pedido,hijo.k, hijo.costeT)
Pag. 197	**si** ¬ `hijo.asignado[i]` **`entonces`** `debe ser` **si** ¬ `hijo.asignados[i]` **`entonces`**
Pag. 197	**mientras** ¬Monticulovacio?(monticulo) y EstimacionOpt(Primero(monticulo)) < cota debería ser **mientras** ¬Monticulovacio?(monticulo) y EstimacionOpt(Primero(monticulo)) <= cota y **si** costeT > hijo.costeT **entonces** debe ser **si** costeT >= hijo.costeT **entonces**
Pag. 201	**si** ¬ `hijo.asignado[i] /\ grafo[verticeAnt,i]` `debe ser`

	si ¬ `hijo.asignados[i] /\ grafo[verticeAnt,i]`
Pag. 201	En la tecera línea, donde dice: porque su estimación optimista sea **menor** que una cierta... debe ser porque su estimación optimista sea **mayor** que una cierta
Pág. 202	hijo.estOpt<-- EstimacionOpt(grafo,minArista,hijo.k,nodo.costeT) debería ser: hijo.estOpt<-- EstimacionOpt(grafo,minArista,hijo.k,hijo.costeT)
Pag 202	**fsi** hijo.asignados[i] <- falso {se desmarca} **fpara** debe ser hijo.asignados[i] <- falso {se desmarca} **fsi** **fpara** (la asignación va dentro de la instrucción **si)**
Pag 202	Falta sumar el coste de la arista de vuelta al nodo inicial: **si** hijo.k = n **entonces** **si** grafo[i,1] ≠ ∞ **entonces** **si** … debe ser **si** hijo.k = n **entonces** **si** grafo[i,1] ≠ ∞ **entonces** hijo.costeT <- hijo.costeT + grafo[i,1] **si** …
Pag. 204	**mientras** ¬Monticulovacio?(monticulo) y EstimacionOpt(Primero(monticulo)) > cota debería ser **mientras** ¬Monticulovacio?(monticulo) y EstimacionOpt(Primero(monticulo)) >= cota
Pag 205	**En los dos sitios en que aparece:** **si** beneficioT < hijo.beneficioT **entonces** debe ser **si** beneficioT <= hijo.beneficioT **entonces**
Pag 209	**mientras** ¬Monticulovacio?(monticulo) y EstimacionOpt(Primero(monticulo)) < cota

	debería ser **mientras** ¬Monticulovacio?(monticulo) y EstimacionOpt(Primero(monticulo)) <= cota y **si** costeT > hijo.costeT **entonces** debe ser **si** costeT >= hijo.costeT **entonces**
Pág. 210	hijo.estOpt<-- EstimacionOpt(hijo.long,m, hijo.k, nodo.costeT) debería ser: hijo.estOpt<-- EstimacionOpt(hijo.long,m, hijo.k, hijo.costeT) En el algoritmo Complecciones: si cadenaX[hijo.k] = cadenaY[hijo.k] entonces debe ser si nodo.cadena[hijo.k] = cadenaY[hijo.k] entonces y si cadenaX[hijo.k+1] = cadenaY[hijo.k] entonces debe ser si nodo.cadena[hijo.k+1] = cadenaY[hijo.k] entonces
Pag. 211	Función complecciones, apartados Sustitución e Inserción: en los 4 puntos donde aparece nodo.cadena[hijo.k] debe ser cadenaY[hijo.k] Además después de las líneas {sustitución} e {inserción} hay que recuperar la cadena del hijo tal como estaba antes de la operación anterior: hijo.cadena <- nodo.cadena