



Estructuras de Datos II

(I.T. Informática de Gestión)
(I.T. Informática de Sistemas)

Convocatoria de Junio de 2005

La Rábida, 17 de junio de 2005

Cuestión 1

1 punto

(a)
elementosMenores (e, Δ) = ∅
elementosMenores (e1, e<iz, de>) =
 si e < e1 **entonces**
 poner (e, elementosMenores (e1, iz) ∪ elementosMenores (e1, de))
 sino
 elementosMenores (e1, iz) ∪ elementosMenores (e1, de)
 fsi

(b)
esEquilibrado (Δ) = verdad
esEquilibrado (e<iz, de>) =
 en caso de
 vacío? (iz) ∧ vacío? (de): verdad
 ¬vacío? (iz) ∧ vacío? (de): altura (iz) == 0
 vacío? (iz) ∧ ¬vacío? (de): altura (de) == 0
 ¬vacío? (iz) ∧ ¬vacío? (de):
 abs (altura (iz) - altura (de)) < 2 ∧ esEquilibrado (iz) ∧ esEquilibrado (de)
 fcaso

Cuestión 2

1,75 puntos

```
template <typename T>
void MatrizDispersa::asignar (int i, int j, const T& e) throw (DiemisionErroneaExcepcion)
var
    Lista<Fila<T>> ::Iterador itf;
    Columna nc(j, e);
    Lista<Columna<T>> > nlc, lc;
    Lista<Columna<T>> ::Iterador itc;
fvar;
inicio
si i < 1 ∨ i > numFil ∨ j < 1 ∨ j > numCol entonces
    lanzar DiemisionErroneaExcepcion()
sino
    // Busco la fila i en la lista de filas
    itf = matriz.principio();
    mientras itf ≠ matriz.final() ∧
        itf.observar(matriz).getFila() < i hacer
        itf.avanzar(matriz);
    fmientras;
```

```
si itf == matriz.final() ∨
    itf.observar(matriz).getFila() > i entonces
    // La fila i no está en la lista
si e ≠ 0 entonces
    // Los 0 no se almacenan
    // Creo la lista de columnas e inserto una nueva fila
    nlc.anadirIzq(nc);
    matriz.insertar(itf, Fila(i, nlc));
fsi;
sino
    // La fila i está en la lista
    // Recorro la lista de columnas
    lc = itf.observar(matriz).getCols();
    itc = lc.principio();
    mientras itc ≠ lc.final() ∧
        itc.observar(lc).getColumna() < j hacer
        itc.avanzar(lc);
    fmientras;
si itc == lc.final() ∨
    itc.observar(lc).getColumna() > j entonces
    // La columna no está en la lista
    si e ≠ 0 entonces
        // Los 0 no se almacenan
        // Inserto la nueva columna
        lc.insertar(itc, nc);
    fsi;
sino
    // La columna está en la lista
    si e == 0 entonces
        // Elimino la columna
        lc.eliminar(itc);
    si lc.esVacia() entonces
        // No quedan otras columnas => elimino la fila
        matriz.eliminar(itf)
    fsi
sino
    // Modifico la columna con el nuevo valor
    lc.modificar(itc, nc)
fsi
fsi
fsi
fsi
fsi
fsi
fin;
```

Cuestión 3

1 punto

```
template <typename T>
bool esInferior (const Arbin<T>& a1, const Arbin<T>& a2)

inicio

si (a1.esVacio() ∨ a2.esVacio()) entonces
    devolver verdadero
fsi
devolver ( a1.getRaiz() < a2.getRaiz() ∧ esInferior (a1.subIzq(), a2.subIzq()) ∧
    esInferior (a1.subDer(), a2.subDer()) )

fin;
```

Cuestión 4**1,25 puntos****float** longMedia(Grafo g, Vértice v1, Vértice v2)

```
var
    Lsta<vértice> camino;
    int lt=0, nc=0;
    int n = g.numVertices();
    bool visitados [n+1];
fvar;
para (int i=1; i <= n; i++)
    visitados[i] = false;
fpara
mediaCaminos (g, v1, v2, visitados, lt, nc);
devolver (lt / nc);
fin;
```

void mediaCaminos(Grafo g, Vértice v1, Vértice v2, **bool*** visitados, **int &** total, **int &** nCam)

```
var
    Conjunto<Vértice> adys;
    Vértice w;
    int suma;
fvar;

si v1 = v2 entonces
    nCam = nCam+1;
    suma = 0;
    para (int i=1; i <= n; i++)
        si visitados[i] entonces suma++;
    fpara
        total = total+ suma;
sino
    visitados[v1] = verdadero;
    adys = g.adyacentes(v1);
    mientras (no adys.esVacio() ) hacer
        w=adys.quitar();
        si (no visitados[w]) entonces
            mediaCaminos (g, w, v2, visitados, total, nCam)
        fsi
    fmientras
        visitados [v1] = falso;
fsi;
fin;
```