

Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática, de Computadores y del Software

Examen Final, Junio 2015

1.[1 pto] Dado un vector de n elementos ($n \geq 0$), se desea obtener la suma de todos los productos de valores situados en parejas de posiciones distintas con una complejidad $\mathcal{O}(n)$. La especificación del algoritmo es la siguiente:

```
{0 ≤ n < longitud(v)}  
fun sumaProductos (int v[], int n) return int p  
{p = ∑ i, j : 0 ≤ i < j < n : v[i] * v[j]}
```

Se han escrito los dos siguientes algoritmos para resolver el problema:

```
(a) k = n; p = 0; s = 0;  
    while (k>0)  
        { p = p + v[k-1] * s;  
          s = s + v[k-1];  
          k = k - 1;  
        };  
    return p;
```

```
(b) k = 0; p = 0; s = 0;  
    while (k<n-1)  
        { p = p + v[k] * s;  
          s = s + v[k];  
          k = k + 1;  
        };  
    return p;
```

Escribir los invariantes de los bucles que permiten demostrar la corrección de cada uno de ellos.

2. [3 ptos] De n actividades conocemos sus instantes de comienzo c_i y finalización f_i ; es decir, el intervalo de realización de la actividad i ($0 \leq i < n$) es $[c_i, f_i]$. Diseñar un algoritmo de vuelta atrás que imprima todos los subconjuntos con r o más actividades que no solapen entre sí (es decir, cuya intersección dos a dos es vacía).

Se puede suponer que los intervalos de las actividades vienen dados en un vector a de parejas de enteros (los instantes de comienzo y finalización) ordenado crecientemente por instante de comienzo.

3. [3 ptos] Podemos utilizar los árboles binarios para representar los caminos en la falda de una montaña. La raíz del árbol representa la cima de la que salen una o dos rutas. Las distintas rutas según se va ensanchando la falda de la montaña se dividen en dos formando caminos que nunca se volverán a conectar. Un escalador está en la cima de la montaña (raíz del árbol) y se da cuenta de que en distintas intersecciones (marcadas en el árbol con 'X') hay amigos que necesitan su ayuda para subir. Tiene que bajar a cada una de las 'X' y ayudarles a subir de uno en uno.

Implementa una función con la cabecera

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

en esa asignatura (o si no ha faltado nunca). Para gestionar esta información diseñan un tipo abstracto de datos *Faltas* con tres operaciones generadoras:

- *anadirAlumno*, que añade un alumno en todas las asignaturas con 0 faltas.
- *anadirFalta*, que incrementa en 1 el número de faltas de un alumno en una asignatura.
- *anadirAsignatura*, que construye una lista con los mismos alumnos de las demás asignaturas, cada uno de ellos con 0 faltas.

A la hora de implementar este TAD han decidido que la lista de faltas de una asignatura viene representada por un diccionario con clave *IdAlumno* (identificador del alumno con un orden definido) y valor asociado el número de faltas del alumno en esa asignatura; y que todas las listas de faltas se hallan almacenadas en un diccionario con clave *IdAsignatura* (identificador de la asignatura) y valor asociado la lista de faltas de esa asignatura. El invariante de la representación incluye el hecho de que las listas de todas las asignaturas contienen exactamente los mismos alumnos:

```
class Faltas
{public :
    void anadirAlumno(const IdAlumno& a);
        //incorpora al alumno en todas las asignaturas que haya con 0 faltas
    void anadirFalta(const IdAlumno& a,const IdAsignatura& s);
        //incrementa en 1 las faltas del alumno en la asignatura
    void anadirAsignatura(const IdAsignatura& s);
        //si es la primera, no tiene aun alumnos,
        //pero si ya existe alguna previa deben de incorporarse todos los alumnos
        //que ya esten presentes en las otras asignaturas

    //INVARIANTE: todas las asignaturas contienen a los mismos alumnos
        ...otros metodos...
private:
    Diccionario<IdAsignatura,Diccionario<IdAlumno,int> > listas_faltas;}
```

En la reunión de fin de curso, ponen en común su información y desean añadir a este TAD las siguientes operaciones:

- *noFaltas*, que por orden alfabético (el dado sobre *IdAlumno*) devuelve una lista con todos los alumnos que no han faltado a ninguna clase en ninguna de las asignaturas.
- *totalFaltas*, que dado un alumno, devuelve el número de faltas que acumula entre todas las asignaturas.

Se pide:

- 1.[0,5 ptos] Elegir justificadamente la implementación de cada uno de los dos diccionarios e indicar, en base a esta decisión, qué le exigis a los tipos *IdAsignatura* e *IdAlumno*, y

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70**

Cartagena99