Algoritmos y Estructuras de Datos
Segundo Parcial – Viernes 22 de Noviembre de 2024

#Orden	Libreta	Apellido y Nombre	E1	E2	E3	E4	Nota Final

- Es posible tener una hoja (2 carillas), escrita a mano, con los anotaciones que se deseen, más los dos apuntes de la materia
- El parcial se aprueba con 50 puntos y hay que tener 60 puntos de promedio entre los dos parciales

E1. Complejidad (20 pts)

Sean $f, g, h : \mathbb{Z} \to \mathbb{Z}$. Indique si las siguientes afirmaciones son verdaderas o falsas. En caso de ser **verdaderas**, **justifique** formalmente con las definiciones o propiedades vistas en clase. En caso de ser **falsas**, dé un **contraejemplo**.

a) Si $f \times g \in \Theta(h)$, entonces $h \in O(g)$
b) $f \in O(g) \Rightarrow O(g) \cap \Omega(f) \neq \emptyset$
c) Si el mejor caso de un algoritmo es $\Theta(n)$, el peor no puede ser $\Omega(log(n))$

E2. Rep&Abs (20 pts)

Cientifico, *Paper* es $\mathbb{Z}$

```
TAD CitasCientificas {
obs publicaciones: dict⟨Paper, seq⟨Cientifico⟩⟩
obs citas: dict⟨Paper, conj⟨Paper⟩⟩
proc iniciarSistema() : CitasCientificas
    asegura(El sistema comienza con ambos diccionarios vacíos)
proc registrarPaper(inout cc : CitasCientificas, in p : Paper, in cs : seq⟨Cientifico⟩)
    requiere(El paper no se encuentra en el sistema previamente)
    asegura(Registra el paper en publicaciones con su lista de autores y en citas con vacío)
proc citarPaper(inout cc : CitasCientificas, in p : Paper, in c : Paper)
    requiere(Ambos papers existen y el paper c no había sido citado previamente por el paper p)
    asegura(Se agrega c al conjunto de citas de p en citas)
}
```

```
Módulo CitasCientíficasImpl implementa CitasCientíficas <
  var publicaciones: Diccionario<Científico, Conjunto<Paper>>
   // Todos los científicos que han publicado algún paper asociados a todos sus papers
  var autores: Diccionario<Paper, Vector<Científico>>
   // Para cada paper, la lista de científicos en el orden que aparecen en la publicación
  var citas: Conjunto<Tupla<Paper, Paper>>
   // Tuplas donde la primer componente es el paper que cita y la segunda el paper citado
  var masCitados: ColaDePrioridadMax<Tupla<int, Paper>>
   // La primera componente de las tuplas es la cantidad de veces que el paper de la segunda fue citado.
   // La prioridad está determinada solamente por la primera componente
>
```

a) Indique si cada una de las sentencias propuestas pertenecen al invariante de representación para la estructura propuesta
   I) Dados dos científicos distintos en **publicaciones**, sus significados son disjuntos
   II) Para todo paper de **autores** no hay repetidos en el vector de su significado
   III) Para todo par de tuplas del conjunto de **citas** si existe (a,b) no existe (b,a)
   IV) No hay tuplas repetidas en **citas**
   V) Hay la misma cantidad de claves definidas en **autores** que elementos en **citas**
   VI) Para cada tupla de **citas**, ambos papers son claves de **autores**
b) Proponga dos sentencias más para el invariante de representación (en castellano preciso)
c) Escribir la función de abstracción en lógica de primer orden

E3. Diseño (30 pts)

Un instituto de cocina quiere organizar las inscripciones a su Escuela Culinaria de Invierno (ECI) donde ofrecen varias materias con capacidad limitada y nombres únicos de largo acotado. Los alumnos pueden intentar inscribirse a las materias y el sistema les informará si han quedado inscriptos o no (en caso que los inscriptos a la materia ya alcanzaran su capacidad). Si no pudieran inscribirse, igualmente quedan registrados en el sistema y serán agregados a la materia si algún inscripto se da de baja, por estricto orden de inscripción.

El sistema deberá implementar las siguientes operaciones en la complejidad pedida, donde a_m es la cantidad de alumnos inscriptos a la materia m. En todos los casos se puede asumir que la materia está registrada en el sistema. Materia es `string`. Alumno y Capacidad son `int`

- `proc abrirMateria`(inout i : *Inscripciones*, in m : *Materia*, in c : *Capacidad*)
  - **Descripción**: Abre la inscripción a una materia
  - **Complejidad**: $O(1)$
- `proc IntentarInscripcion`(inout i : *Inscripciones*, in m : *Materia*, in a : *Alumno*) : Bool
  - **Descripción**: Intenta inscribir a un alumno a una materia a la que no está inscripto. Devuelve si el alumno quedó inscripto o no
  - **Complejidad**: $O(log(a_m))$
- `proc DarseDeBaja`(inout i : *Inscripciones*, in m : *Materia*, in a : *Alumno*)
  - **Descripción**: Saca al alumno de los inscriptos (donde ya debería estar) y agrega al siguiente de la lista de espera
  - **Complejidad**: $O(log(a_m))$
- `proc InscriptosMateria`(in i : *Inscripciones*, in m : *Materia*) : *Conjunto*(*Alumno*)
  - **Descripción**: Devuelve el conjunto de alumnos inscriptos en la materia
  - **Complejidad**: $O(1)$

Se pide:

a) Plantear la estructura de representación del módulo `ECIInscripcionesImpl`, que provea las operaciones mencionadas cumpliendo las complejidades requeridas

b) Escribir el algoritmo de `DarseDeBaja`. Justifique **detalladamente** que se cumple la complejidad requerida

c) **Teórica**. ¿Qué características de un `Heap` permiten representarlo con un arreglo? ¿Qué permite saber cuáles son los nodos hijos y cuál es el padre? Desarrolle en no más de 15 renglones

E4. Sorting (30 pts)

a) **Teórica**. Dado un arreglo de tamaño n ¿Por qué todos los algoritmos de ordenamiento tienen complejidad $\Omega(n)$? ¿Por qué todos los algoritmos de ordenamiento tienen complejidad $\Omega(n.log(n))$ cuando no se tiene información previa sobre sus elementos? Desarrolle en no más de 15 renglones

b) Se está por realizar un gran censo de perros en la Ciudad de Buenos Aires. La información recaudada llegará en forma de un arreglo de tuplas `<Nombre, Raza, Edad, Peso>`. Las razas son `strings` de largo acotado L, pero no sabemos cuántas razas pueden existir, aunque sí sabemos que entre ellas está la mejor raza de todas: raza Perro. Las edades serán enteros positivos entre 1 y 20 y los pesos serán de tipo `float`.

Proponga un algoritmo que organice la información en dos arreglos:

- Uno con los perros raza Perro ordenados por edad y, en caso de empate, por peso, ambos en forma ascendente
- Otro arreglo con la información de las razas y la cantidad de perros de cada una en forma de tuplas `<Raza, int>`, ordenadas en forma decreciente por la cantidad y, en caso de empate, alfabéticamente por la raza

La complejidad temporal del algoritmo debe ser $O(n + p.log(p))$, donde n es el total de perros y p el total de perros raza Perro.

Ejemplo

Si recibimos:

```
[ <Nulus, Perro, 4, 15.3>, <Joe, Cocker, 14, 8.0>, <Lady, Caniche, 10, 6.1>, <Dobby, Perro, 7, 18.3>
<Res, Dogo, 3, 50.0>, <Haru, Perro, 7, 30.0>, <Clementina, Perro, 14, 10.0>, <Choqui, Caniche, 16, 5.0>]
```

Deberíamos devolver:

```
P: [ <Nulus, 4, 15.3>, <Dobby, 7, 18.3>, <Haru, 7, 30.0>, <Clementina, 14, 10.0>]

R: [ <Caniche, 2>, <Cocker, 1>, <Dogo, 1>]
```