tp2

Niick Dæmøn

June 2024

1. Estructuras

1.1. Clase Materia

alumne es String
cargo es String

TAD **Materia** {
obs alumnos: $seq\langle alumne \rangle$
obs docentes: $seq\langle cargo \rangle$
}

`proc nuevaMateria` () : Materia
`asegura` $\{res.alumnos = \{\} \ \wedge \ res.docentes = \{\}\}$

`proc inscribirAlumno` (inout m:Materias, in a:alumnes)
`requiere` $\{m = m0\}$
`asegura` $\{m.docentes = m0.docentes\}$
`asegura` $\{|m.alumnos| = |m0.alumnos| + 1\}$
`asegura` $\{a \notin m0.alumnos \ \wedge \ a \in m.alumnos\}$
`asegura` $\{(\forall e : alumne) \ (e \in m.alumnos \longrightarrow_L a \in m0.alumnos)\}$

`proc inscribirDocente` (inout m:Materia, in c:cargo)
`requiere` $\{m = m0\}$
`asegura` $\{m.alumnos = m0.alumnos\}$
`asegura` $\{|m.docentes| = |m0.docentes| + 1\}$
`asegura` $\{c \notin m0.docentes \ \wedge \ c \in m.docentes\}$
`asegura` $\{(\forall c : cargo) \ (c \in m.docentes \longrightarrow_L c \in m0.docentes)\}$

`proc obtenerAlumnos` (in m:Materia) : $seq\langle alumne \rangle$
`asegura` $\{res = m.alumnos\}$

`proc obtenerPlantel` (in m:Materia) : $seq\langle cargo \rangle$
`asegura` $\{res = m.docentes\}$

Modulo MateriaImpl **implementa** Materia

var alumnos: ListaEnlazada< *alumne* >
var docentes: Array< *cargo* >

`proc nuevaMateria` () : MateriaImpl

```
var res: MateriaImpl

res.alumnos = new listaVacia()
res.docentes = new Array<T>(4)

return res
```

- Complejidad O(1)

`proc inscribirAlumno` (inout m:MateriaImpl, in a:alumne)

```
m.alumnos.agregarAtras(a)
```

- Complejidad O(1)

`proc inscribirDocente` (inout m:MateriaImpl, in c:cargo)

```
if (c == 'profesor') do
    m.docentes[0] += 1

else if (c == 'JTP') do
    m.docentes[1] += 1

else if (c == 'AY1') do
    m.docentes[2] += 1

else
    m.docentes[3] += 1
endif
```

- Complejidad O(1).

`proc obtenerAlumnos` (in m:MateriaImpl) : ListaEnlazada()

```
var res: ListaEnlazada()

res := m.alumnos

return res
```

- Complejidad O(1)

`proc obtenerPlantel` (in m:MateriaImpl) : Array¡cargo¿

```
1 var res: Array<cargo>
2
3 res := m.docentes
4
5 return res
```

- Complejidad O(1)

1.2. Clase Estudiantes

LU es string
materias es string

TAD **Estudiantes**
obs datos: Diccionario$< LU, seq\langle materias \rangle >$

`proc nuevosEstudiantes` () : Estudiantes
`asegura` $\{res = \{\}\}$

`proc agregarMateria` (inout e:Estudiantes, in lu:LU, in m:materias)
`requiere` $\{e = e0\}$
`requiere` $\{m \notin e[lu]\}$
`asegura` $\{|e.datos| = |e0.datos|\}$
`asegura` $\{(\forall alumno : LU)\ (alumno \neq lu \ \wedge \ alumno \in e.datos \longrightarrow_L alumno \in e0.datos \ \wedge e0.datos[alumno] = e.datos[alumno])\}$
`asegura` $\{|e.datos[lu]| = |e0.datos[lu]| + 1\}$
`asegura` $\{m \notin e0.datos[lu] \ \wedge \ m \in e.datos[lu]\}$

Modulo EstudiantesImpl **Implementa** Estudiantes
var datos: Trie$< LU, ListaEnlazada < materias \gg$

`proc nuevosEstudiantes` () : EstudiantesImpl

```
1 var res: EstudiantesImpl
2
3 res := new Trie()
4
5 return res
```

- Complejidad O(1)

`proc agregarMateria` (inout e:EstudiantesImpl, in lu:LU, in m:materias)

```
1 e.datos.obtener(lu).agregarAtras(m)
```

- Complejidad O(1)

2. SIU

Modulo SIU **Implementa** SistemaSiu
var carreras: Trie$<'$ $carrera', Trie <'$ $materia', MateriaImpl \gg$
var estudiantes: EstudiantesImpl

2.1. Metodo inscribir

inscribir(inout sistema:SIU, in materia:String, in carrera: String, in estudiante: String)

Trie materias = sistema.carreras.obtener(carrera)

- accedo a las materias disponible para esa carrera en tiempo O($|c|$)

MateriaImpl mat = materias.obtener(materia)

- accedo a la materia a la cual quiero agregar al estudiante en tiempo O($|m|$)

mat.inscribirAlumno(estudiante)

- agrego el alumno a esa materia en tiempo O(1)

sistema.estudiantes.agregarMateria(estudiante,materia)

- agrego esa materia al conjunto de materias de ese estudiante en tiempo O(1)

La complejidad queda $O(|c| + |m| + 1 + 1) = O(|c| + |m|)$

2.2. Metodo inscriptos

inscriptos(in sistema:SIU, in carrera: string, in nombreMateria: string) : $\mathbb{N}$

Trie Materias = sistema.carreras.obtener(carrera)

- accedo a las materias disponible para esa carrera en tiempo O($|c|$)

MateriaImpl mat = Materias.obtener(nombreMateria)

- accedo a la materia de la cual quiero saber su cantidad de alumnos en tiempo O($|m|$)

mat.obtenerAlumnos(nombreMateria).longitud()

- chequeo cuantos alumnos tiene en tiempo O(1)

- la complejidad queda $O(|c| + |m|)$

2.3. Metodo agregarDocente

agregarDocente(inout sistema:SIU, in cargo:CargoDocente, in carrera:string, in nombreMateria:string)

Trie Materias = sistema.carreras.obtener(carrera)

- accedo a las materias disponible para esa carrera en tiempo O($|c|$)

MateriaImpl mat = Materias.obtener(nombreMateria)

- accedo a la materia de la cual quiero saber su cantidad de alumnos en tiempo O($|m|$)

mat.inscribirDocente(cargo)

- agrego el docente a la materia correspondiente en tiempo O(1)

- Complejidad total $O(|c| + |m|)$

2.4. Metodo PlantelDocente

plantelDocente(in sistema:SIU, in carrera:string , in nombreMateria:string): $seq\langle\mathbb{N}\rangle$

Trie Materias = sistema.carreras.obtener(carrera)

- accedo a las materias disponible para esa carrera en tiempo O($|c|$)

MateriaImpl mat = Materias.obtener(nombreMateria)

- accedo a la materia de la cual quiero saber su cantidad de alumnos en tiempo O($|m|$)

var res: $seq\langle\mathbb{N}\rangle$
res = mat.obtenerPlantel(materia)
return res

- me devuelve la lista de docentes en tiempo O(1)

- Complejidad total $O(|c| + |m|)$

2.5. Metodo excedeCupo?

exedeCupo(in sistemas:SIU, in carrera:string, in string nombreMateria) : bool
Trie Materias = sistema.carreras.obtener(carrera)

- accedo a las materias disponible para esa carrera en tiempo O($|c|$)

MateriaImpl mat = Materias.obtener(nombreMateria)

- accedo a la materia de la cual quiero saber su cantidad de alumnos en tiempo O($|m|$)

```
numEstudiantes = mat.obtenerAlumnos().longitud()
cupoTotal = 0

for(CargoDocente : docentes)
   if (docente.cargo = 'profesor')
     cupoTotal += 250
   else if (docente.cargo = 'JTP')
     cupoTotal += 100
   else if (docente.cargo = ÁY1')
     cupoTotal += 20
    else if (docente.cargo = ÁY2')
     cupoTotal += 30
return numEstudiantes > cupoTotal
```

2.6. Metodo Carreras

aca usando el iterador del trie nos da complejidad en tiempo O($\sum_{c \in C} |c|$)

2.7. Metodo materias

aca idem al anterior solo que recorremos el trie Materias y nos da la complejidad del enunciado.

2.8. Metodo materiasInscriptas

aca preguntamos en sistema.estudiantes la longitud y se hace en tiempo O(1).

2.9. Metodo cerrarMateria

la complejidad $O(|c|)$ se logra al recorrer la carrera de la materia que queremos borrar.
la complejidad O($|m|$) se logra recorriendo que materia queremos borrar.
la complejidad O($\sum_{n \in Nm} |n|$) se logra recorriendo las materias que son la misma pero con distinto nombre y la complejidad O(Em) se llega sacando esa materia de cada alumno que la tenga.