DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Recuperatorio

Ejercicio X2

27 de julio de 2020 — Algoritmos y Estructuras de Datos II

Integrante	LU	Correo electrónico
Rodriguez, Miguel	57/19	mmiguerodriguez@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3300
https://exactas.uba.ar

Ejercicio

usuario **es** string
sala **es** string
infoClase **es** tupla(usuario, conj(usuario), conj(usuario))

TAD SistemaConsultasVirtuales

géneros scv

igualdad observacional

$$(\forall s, s' : \text{scv}) \left(s =_{\text{obs}} s' \iff \begin{pmatrix} \text{docentes}(s) =_{\text{obs}} \text{docentes}(s') \wedge \\ \text{salas}(s) =_{\text{obs}} \text{salas}(s') \wedge \\ \text{alumnos}(s) =_{\text{obs}} \text{alumnos}(s') \wedge \\ \text{clases}(s) =_{\text{obs}} \text{clases}(s') \wedge_{\text{L}} \\ (\forall u : \text{usuario})((u \in \text{docentes}(s) \vee \\ u \in \text{alumnos}(s)) \Rightarrow_{\text{L}} \text{asistencias}(s, u) = \\ \text{asistencias}(s', u)) \end{pmatrix} \right)$$

observadores básicos

alumnos	: scv	$\longrightarrow$	conj(usuario)	
docentes	: scv	$\longrightarrow$	conj(usuario)	
salas	: scv	$\longrightarrow$	conj(sala)	
clases	: scv	$\longrightarrow$	dicc(sala, infoClase)	
asistencias	: scv s $\times$ usuario u	$\longrightarrow$	nat	$\{u \in \text{docentes}(s) \vee u \in \text{alumnos}(s)\}$

generadores

generar	: conj(usuario) $\times$ conj(sala)	$\longrightarrow$	scv
abrir	: scv s' $\times$ sala s $\times$ usuario u	$\longrightarrow$	scv $\left\{ \begin{matrix} \neg\text{enClase}(s', u) \wedge s \in \text{salas}(s') \wedge_{\text{L}} (\neg\text{ocupada(s', s)} \wedge u \in \text{docentes}(s') \vee \\ \text{ocupada}(s', s)) \end{matrix} \right\}$
salir	: scv s $\times$ usuario u	$\longrightarrow$	scv $\{u \in \text{alumnos}(s) \vee u \in \text{docentes}(s) \wedge \text{enClase}(s, u)\}$

otras operaciones

enClase	: scv s $\times$ usuario u	$\longrightarrow$	bool	
claseDe	: scv s $\times$ usuario u	$\longrightarrow$	sala $\{u \in \text{alumnos}(s) \vee u \in \text{docentes}(s) \wedge \text{enClase}(s, u)\}$
ocupada	: scv s' $\times$ sala s	$\longrightarrow$	bool	$\{s \in \text{salas}(s')\}$
esAdmin	: scv s' $\times$ usuario u $\times$ sala s	$\longrightarrow$	bool $\{u \in \text{docentes}(s') \wedge s \in \text{salas}(s')\}$	
masAsistidores	: scv	$\longrightarrow$	conj(usuario)	

axiomas

alumnos(generar(ds, ss))	$\equiv$	$\emptyset$
alumnos(abrir(s', s, u))	$\equiv$	**if** $u \notin \text{docentes}(s') \wedge u \notin \text{alumnos}(s')$ **then** Ag(u, alumnos(s')) **else** alumnos(s') **fi**

$$
\begin{aligned}
&\text{alumnos}(\text{salir}(s,\, u)) && \equiv \text{alumnos}(s)\\
&\text{docentes}(\text{generar}(ds,\, ss)) && \equiv ds\\
&\text{docentes}(\text{abrir}(s',\, s,\, u)) && \equiv \text{docentes}(s')\\
&\text{docentes}(\text{salir}(s,\, u)) && \equiv \text{docentes}(s)\\
&\text{salas}(\text{generar}(ds,\, ss)) && \equiv ss\\
&\text{salas}(\text{abrir}(s',\, s,\, u)) && \equiv \text{salas}(s')\\
&\text{salas}(\text{salir}(s,\, u)) && \equiv \text{salas}(s)\\
&\text{clases}(\text{generar}(ds,\, ss)) && \equiv \text{vacío}\\
&\text{clases}(\text{abrir}(s',\, s,\, u)) && \equiv \textbf{if } \neg\text{ocupada}(s',\, s) \textbf{ then}\\
&&& \quad \text{definir}(s,\, \langle u,\, \{u\},\, \{u\}\rangle,\, \text{clases}(s'))\\
&&& \textbf{else}\\
&&& \quad \text{definir}(s,\, \langle \text{admin}(s',\, s),\, \text{Ag}(u,\, \text{conectados}(s',\, s)),\, \text{Ag}(u,\, \text{historial}(s',\, s))\rangle,\, \text{clases}(s'))\\
&&& \textbf{fi}\\
&\text{clases}(\text{salir}(s,\, u)) && \equiv \textbf{if } \text{esAdmin}(u,\, \text{claseDe}(s,\, u)) \textbf{ then}\\
&&& \quad \text{borrar}(\text{claseDe}(s,\, u),\, \text{clases}(s))\\
&&& \textbf{else}\\
&&& \quad \text{definir}(\text{claseDe}(s,\, u),\, \langle \text{admin}(s,\, \text{claseDe}(s,\, u)),\, \text{conectados}(s,\, \text{claseDe}(s,\, u)) - \{u\},\, \text{historial}(s,\, \text{claseDe}(s,\, u))\rangle,\, \text{clases}(s))\\
&&& \textbf{fi}\\
&\text{asistencias}(\text{generar}(ds,\, ss),\, u) && \equiv 0\\
&\text{asistencias}(\text{abrir}(s',\, s,\, u),\, u') && \equiv \textbf{if } u = u' \textbf{ then}\\
&&& \quad \textbf{if } \neg\text{ocupada}(s',\, s) \textbf{ then}\\
&&& \quad\quad 1 + \text{asistencias}(s',\, u')\\
&&& \quad \textbf{else}\\
&&& \quad\quad \textbf{if } u' \notin \text{historial}(s',\, s) \textbf{ then}\\
&&& \quad\quad\quad 1 + \text{asistencias}(s',\, u')\\
&&& \quad\quad \textbf{else}\\
&&& \quad\quad\quad \text{asistencias}(s',\, u')\\
&&& \quad\quad \textbf{fi}\\
&&& \quad \textbf{fi}\\
&&& \textbf{else}\\
&&& \quad \text{asistencias}(s',\, u')\\
&&& \textbf{fi}\\
&\text{asistencias}(\text{salir}(s,\, u),\, u') && \equiv \text{asistencias}(s,\, u')\\
&\text{enClase}(s,\, u) && \equiv \text{enClaseAux}(s,\, \text{claves}(\text{clases}(s)),\, u)\\
&\text{enClaseAux}(s,\, claves,\, u) && \equiv \textbf{if } \text{vacío?}(claves) \textbf{ then}\\
&&& \quad \text{false}\\
&&& \textbf{else}\\
&&& \quad u \in \text{conectados}(s,\, \text{dameUno}(claves)) \vee \text{enClaseAux}(s,\, \text{sinUno}(claves),\, u)\\
&&& \textbf{fi}\\
&\text{claseDe}(s,\, u) && \equiv \text{claseDeAux}(s,\, \text{claves}(\text{clases}(s)),\, u)\\
&\text{claseDeAux}(s,\, claves,\, u) && \equiv \textbf{if } u \in \text{conectados}(s,\, \text{dameUno}(claves)) \textbf{ then}\\
&&& \quad \text{dameUno}(claves)\\
&&& \textbf{else}\\
&&& \quad \text{claseDeAux}(s,\, \text{sinUno}(claves),\, u)\\
&&& \textbf{fi}
\end{aligned}
$$

ocupada(s', s) $\equiv$ def?(s, clases(s'))

esAdmin(s', s, u) $\equiv$ u = admin(s', s)

masAsistidores(s) $\equiv$ masAsistidoresAux(s, alumnos(s) $\cup$ docentes(s))

masAsistidoresAux(s, us) $\equiv$ **if** vacío?(us) **then**
$\emptyset$
else
if asistencias(s, dameUno(us)) = maxAsistencias(s) **then**
Ag(dameUno(us), masAsistidoresAux(s, sinUno(us)))
else
masAsistidoresAux(s, sinUno(us))
fi
fi

maxAsistencias(s) $\equiv$ maxAsistenciasAux(s, alumnos(s) $\cup$ docentes(s), 0)

maxAsistenciasAux(s, us, max) $\equiv$ **if** vacío?(us) **then**
max
else
if asistencias(s, dameUno(us)) > max **then**
maxAsistenciasAux(s, sinUno(us), asistencias(s, dameUno(us)))
else
maxAsistenciasAux(s, sinUno(us), max)
fi
fi

auxiliares

admin(s', s) $\equiv$ π_1(obtener(s, clases(s')))

conectados(s', s) $\equiv$ π_2(obtener(s, clases(s')))

historial(s', s) $\equiv$ π_3(obtener(s, clases(s')))

Fin TAD

Justificaciones

Enunciado

1. Una clase no es lo mismo que una sala. Las clases ocurren en salas. No puede haber más de una clase por sala. Una vez que el administrador finaliza una clase, esta es terminada, y al abrir la misma sala, cuenta como una clase nueva.

2. Para que dos sistemas sean observacionalmente iguales, no nos importará cuáles fueron las clases en las que ingresaron las personas, si no que la cantidad de clases distintas a las que asistieron sean las mismas en ambos sistemas. Lo que si nos importará son las clases que están siendo dictadas en el momento (diccionario clases), que deben ser iguales. En caso de querer implementar un sistema en el cuál se distinga además por las clases a las que ingresaron los usuarios, en vez de guardar en asistencias la cantidad de asistencias de un usuario a clases distintas, guardaría una lista de salas a las que ingresó en clases distintas y usar la longitud de esta lista para calcular la cantidad de asistencias.

3. El enunciado no dice que el sistema tenga un conjunto de alumnos predeterminado que pueden unirse a las clases, por lo tanto, lo propuesto fue agregar a los alumnos al observador

de tipo `conj(usuario)` cuando un usuario ingresa a una clase y no está en el conjunto de docentes ni alumnos del sistema.

Resolución

1. Cuando se genera el sistema le pasamos los docentes y las salas disponibles.
2. Cuando ingresamos a una clase, debe ocurrir que la sala esté en las salas disponibles. Si no está ocupada entonces debemos ser un docente (que va a terminar siendo el administrador), mientras que si está ocupada puede ingresar tanto un docente como un alumno.
3. En caso de ser docente y ser el que está abriendo la sala por primera vez, definimos esa sala en el diccionario de clases con el docente como administrador. Además, lo agregamos al conjunto de conectados e historial de la clase
4. En caso de que la sala esté ocupada, agregamos al docente o alumno que quiera ingresar a la clase en el diccionario de la sala actual. Lo agregamos en el historial de la clase y en los que estan conectados actualmente a la clase.
5. Al ingresar a una clase, si el usuario se encuentra en el historial de usuarios que entraron a esa clase, no se le suma a sus asistencias, caso contrario, se le suma 1 a la cantidad de asistencias a clases distintas que tiene. Una clase es distinta para un usuario cuando este **no está** en el historial para esa clase. Ejemplo: si existe una clase en la sala 1, el usuario entra, el administrador la cierra y se vuelve a crear otra clase en la sala 1 y el mismo usuario entra, entonces se cuenta como que tiene 2 asistencias (ya que para mi interpretación estas serían clases distintas).
6. Cuando una persona se retira de una sala, si es el administrador, se borra del diccionario de clases la sala en la que está, esto a su vez borra el administrador de la clase, los usuarios conectados y el historial. En caso de no ser el administrador, se elimina al usuario del conjunto de conectados para esa clase, pero se mantiene al usuario en el historial de usuarios que se conectaron a la clase (esto nos permitirá no sumarle asistencias cuando el usuario ingrese a una clase más una vez).
7. Para calcular los usuarios con más asistencias a clases primero utilizamos `maxAsistencias` para tener el número máximo de asistencias entre todos los usuarios. Luego, comparamos las asistencias de cada usuario en `masAsistidores`, y todos los que tengan cantidad de asistencias = máximo de asistencias, se agregan al conjunto de resultado.