PAC1 – Primera prova d'avaluació continuada

Presentació

A continuació us presentem l'enunciat de la primera prova d'avaluació continuada del curs. Tingueu en compte que la PAC s'ha de resoldre individualment.

Competències

Les competències que treballareu a la PAC son les següents:

Específiques

- Capacitat per analitzar un problema en el nivell d'abstracció adequat a cada situació i aplicar les habilitats i coneixements adquirits per a resoldre'l.
- Capacitat per dissenyar i construir aplicacions informàtiques mitjançant tècniques de desenvolupament, integració i reutilització.
- Capacitat per proposar i avaluar diferents alternatives tecnològiques per resoldre un problema concret.

Transversals

- Ús i aplicació de les TIC en l'àmbit acadèmic i professional.
- Capacitat per a innovar i generar noves idees.

Objectius

Els objectius que es persegueixen en el desenvolupament de la PAC són els següents:

- Entendre el concepte de TAD i saber-ne fer l'especificació.
- Conèixer la biblioteca de TADs de l'assignatura i saber utilitzar-los per dissenyar i implementar noves estructures de dades.
- Saber calcular l'eficiència espacial i temporal d'una estructura de dades i dels algorismes associats per tal de comparar diferents alternatives i poder triar-ne la millor en termes d'eficiència (temporal o espacial)
- Ser capaç d'identificar l'estructura de dades utilitzada en un programa i entendre'n el funcionament.
- Entendre el funcionament dels contenidors seqüencials i els arbres presentats a l'assignatura. Saber quan i com utilitzar aquests contenidors.

Descripció de la PAC a realitzar

La PAC consta de 4 exercicis, alguns més teòrics i altres més pràctics, en els que posareu en pràctica els coneixements adquirits en l'estudi dels tres primers mòduls de l'assignatura.

Recursos

Els recursos necessaris per a desenvolupar la PAC son els següents:

Bàsics (material didàctic de l'assignatura)

- Mòdul 1
- Mòdul 2
- Mòdul 3

Complementaris

Llibreria de TADs de l'assignatura

Criteris de valoració

La puntuació global de la PAC és la suma de les puntuacions individuals obtingudes a cada un dels exercicis que la formen. La puntuació individual de cada exercici s'especifica a cadascun d'ells.

Format i data de lliurament

Data de publicació

3 d'octubre de 2013

Data de lliurament

24 d'octubre de 2013

Format de lliurament

El lliurament cal fer-lo a través de l'espai de Lliurament i registre d'AC amb un únic fitxer preferentment en format PDF o, si no és possible, Word o OpenOffice. Aquest fitxer contindrà la solució (incloent-ne les classes Java). Si us plau, no hi copieu l'enunciat, feu-hi constar el vostre nom a cada pàgina (per exemple, amb un peu de pàgina), i numereu les pàgines.

Enunciat

Es vol dissenyar una estructura de dades per gestionar un sistema de control de versions SVN. En aquesta PAC farem una prova pilot d'una part del sistema (un conjunt reduït de funcionalitats) i treballarem amb volums d'informació petits per tal de que us familiaritzeu amb el problema a resoldre i "practiqueu" una mica el disseny i la composició d'estructures de dades per donar forma i solució al problema plantejat. Concretament, per a la realització de l'exercici considereu:

- El nombre d'usuaris U serà força gran i conegut.
- El nombre de grups G serà força petit però en constant augment. Són els mateixos usuaris que creen diferents grups per treballar en diferents projectes.
- El nombre de repositoris R, serà estable i el limitarem a 100.
- El nombre de grups d'un usuari GU serà petit i variable.
- El nombre d'usuaris d'un grup UG serà relativament petit però molt variable (hi haurà grups d'un sol usuari i d'altres amb centenars d'usuaris).
- El nombre de grups que treballen en un repositori GR serà petit però variable.
- El nombre de fitxers d'un repositori FR serà molt variable, però el limitarem a 100.
- El nombre de revisions d'un fitxer FC serà molt variable i il·limitat.

Exercici 1 [2'5 punts]

Especifiqueu un TAD *Svn* que permeti les operacions:

- Afegir un nou usuari al sistema. De cada usuari en sabrem el seu codi identificador (natural), el seu *email* i *password*. Si ja existeix un usuari amb aquest codi, actualitzem les seves dades.
- Afegir un nou grup al sistema. De cada grup en sabrem el seu codi identificador i el nom del grup. Cal que el codi del grup sigui diferent a tots els grups donats d'alta prèviament.
- Afegir un usuari a un grup a partir dels seus codis. Si l'usuari o el grup no existeixen retorna un error.
- Afegir un nou repositori al sistema. De cada repositori en sabrem el seu codi identificador (un número de 1..100), el seu *path* (ho podeu entendre com un *string*), i una descripció. Cal que no existeixi cap repositori amb el mateix codi.
- Afegir un grup a un repositori a partir dels seus codis. Cal que existeixi el repositori i l'usuari i, per tant, no retorna mai error.
- Actualitzar el codi font d'un fitxer al sistema. Sabrem el codi del repositori, i de l'usuari que fa l'actualització, la data, el *path* del fitxer (*string* per exemple 'src/main/hello.java'), el codi font (per simplificar, un *string*), i el número de revisió. Cal comprovar que el repositori i l'usuari existeixin, i que l'usuari pertany a algun grup del repositori, sinó retorna error. Si el repositori no conté un fitxer amb el *path* indicat, cal afegir-lo. Si el repositori conté el fitxer amb el *path* i número de revisió indicats, retorna un error.

- Consultar els fitxers d'un repositori, d'una revisió concreta. Cal que el repositori existeixi. Tothom podrà consultar el codi font dels repositoris.
- Consultar el codi font d'un fitxer, d'un repositori i d'una revisió concreta. Cal que el repositori existeixi i que el fitxer ja s'hagués afegit en aquella revisió. Tothom podrà consultar el codi font dels repositoris.
- Consultar quin és el repositori més actiu (el que més actualitzacions ha fet).
- Consultar quin és el grup més actiu (el que més actualitzacions ha fet).

Apartat a) [1 punt]

Doneu la signatura del TAD *Svn*. És a dir, indiqueu el nom que donaríeu a les operacions encarregades de cada funcionalitat requerida. Indiqueu també quins caldria que fossin els paràmetres d'entrada i quina la sortida en cas que es necessités.

Solució

- addUser(id, email, password)

- addGroup(id, name)

- groupAddUser(idGroup, idUser)

- addRepository(idRepository, path, description)

- repositoryAddGroup(idRepository, idGroup)

- commit(idRepository, idRevision, idUser, datetime, filePath, newSourceCode)

- getFiles(idRespository, idRevision):Iterator

- getFile(idRepository, idRevision, filePath):string

- mostActiveRepository(): Repository

- mostActiveGroup(): Group

Apartat b) [1,5 punts]

Feu l'especificació contractual de les operacions del TAD *Svn*. En la redacció de l'especificació podeu fer servir, si et cal, qualsevol de les operacions del TAD. preneu com a model l'especificació de l'apartat 1.2.3 del Mòdul 1 dels materials docents. Es valorarà especialment la concisió (absència d'elements redundants o innecessaris), precisió (definició correcta del resultat de les operacions), completesa (consideració de tots els casos possibles en què es pot executar cada operació) i manca d'ambigüitats (coneixement exacte de com es comporta cada operació en tots els casos possibles) de la solució. És important respondre aquest apartat usant una descripció condicional i no procedimental. L'experiència ens demostra que no sempre resulta fàcil distingir entre les dues descripcions, és per això que fem especial èmfasi insistint que poseu molta atenció a les vostres definicions.

A títol d'exemple indicarem que la descripció condicional (la correcta a utilitzar en el contracte) d'omplir un got buit amb aigua seria:

> @pre el got es troba buit.
> @post el got és ple d'aigua.

En canvi una descripció procedimental (incorrecta per utilitzar en el contracte) tindria una forma semblant a:

> @pre el got hauria de trobar-se buit, si no es trobés buit s'hauria de buidar.
> @post s'acosta el got a l'aixeta i s'hi tira aigua fins que estigui ple.

Cal també tenir en compte que un contracte hauria de disposar d'invariant sempre que aquesta fos necessària per descriure el TAD.

Solució

@pre cert

@post si el codi d'usuari és nou, els usuaris seran els mateixos més un nou usuari amb les dades indicades. Sinó les dades de l'usuari s'hauran actualitzat amb les noves

- addUser(id, email, password)

@pre no existeix cap grup amb el codi id

@post els grups seran els mateixos més un nou grup amb les dades indicades

- addGroup(id, name)

@pre cert

@post si grup i usuari existeixen, els usuaris del grup seran els mateixos més el nou usuari. Altrament retorna error

- groupAddUser(idGroup, idUser)

@pre no existeix cap repository amb el codi idRepository

@post els repositoris seran els mateixos més un nou repositori amb les dades indicades

- addRepository(idRepository, path, description)

@pre existeix un repositori amb codi idRepository, i un grup amb codi idGrup

@post els grups del repositori seran els mateixos més el nou grup

- repositoryAddGroup(idRepository, idGroup)

@pre cert

@post si el repositori o l'usuari no existeixen, o si el repositori ja tenia una actualització del fitxer amb el mateix número de revisió, retorna error. Si l'usuari no pertany a cap del grups del repositori retorna error. Si el repositori no tenia un fitxer amb el path indicat, el repositori tindrà els mateixos fitxers més el nou, amb les dades de la primera revisió. Altrament, el fitxer tindrà les mateixes revisions més una de nova, amb les dades indicades.

- commit(idRepository, idRevision, idUser, datetime, filePath, newSourceCode)

@pre existeix un repositori amb idRepository

@post retorna un iterador per recórrer els fitxers que formaven part del repositori en aquella revisió

- getFiles(idRespository, idRevision):Iterator

@pre existeix un repositori amb idRepository i, en aquella revisió, ja conté un fitxer amb filePath

@post retorna el codi font del fitxer en aquella revisió

- getFile(idRepository, idRevision, filePath): string

@pre cert

@post retorna el repositori més actiu, o un d'ells en cas d'empat

- mostActiveRepository(): Repository

@pre cert

@post retorna el grup més actiu, o un d'ells en cas d'empat.

- mostActiveGroup(): Group

Exercici 2 [3,5 punts]

A l'exercici 1 heu definit l'especificació d'un nou TAD, el TAD *Svn*. Ara us demanem que feu el disseny de les estructures de dades que formaran aquest TAD. Dissenyeu, doncs, el sistema per tal que sigui el màxim d'eficient possible, tant a nivell d'eficiència espaial com temporal, tenint en compte els volums d'informació i les restriccions especificades a l'enunciat.

Per a la realització de la PAC1 us haureu de limitar als TAD's dels mòdul 3 dels apunts. Queden descartats, per tant, els arbres i les taules de dispersió.

Tingueu en compte només les operacions que es demanen a l'enunciat a l'hora de fer aquest disseny.

Apartat a) [0,5 punts]

Dubtem entre utilitzar un vector, un vector ordenat o una llista encadenada ordenada per a emmagatzemar els repositoris. Justifiqueu quina creieu que és la millor opció.

Solució

L'enunciat ens diu que el nombre de repositoris és estable i limitat a 100. Això ens desaconsella utilitzar una estructura dinàmica com la llista encadenada ordenada ja que malgastem espai.

A més, també ens diu que el codi d'un repositori serà un número d'1..100. Això ens permet guardar cada repositori directament a la posició del vector que indica el seu codi. Descartem els vectors ordenats ja que ens portarien a costos logarítmics, mentre que un simple vector tindria costos constants en totes les operacions.

Apartat b) [0,5 punts]

Dubtem entre utilitzar un vector, un vector ordenat, o una llista encadenada ordenada per a emmagatzemar els usuaris. Justifiqueu quina creieu que és la millor opció.

Solució

L'enunciat ens diu que el nombre d'usuaris serà gran i conegut. A més, sabem que en diferents operacions caldrà buscar els usuaris a partir del seu codi. Per tant, descartem les llistes encadenades ordenades ja que ocupen més espai i tindrien costos lineals.

La utilització d'un vector implicaria costos lineals a l'hora de fer les cerques, i constants a les insercions. En el nostre TAD, l'operació més habitual i important serà la consulta d'usuaris a partir del codi i, per tant, els vectors serien ineficients en aquest aspecte.

Per contra, els vectors ordenats tindrien costos lineals a les insercions i logarítmics a les consultes. Per tant, ens quedem amb els vectors ordenats.

Apartat c) [0,5 punts]

Dubtem entre utilitzar un vector, un vector ordenat, o una llista encadenada per a emmagatzemar els fitxers d'un repositori. Justifiqueu quina creieu que és la millor opció.

Solució

L'enunciat ens diu que el nombre de fitxers d'un repositori serà molt variable però limitat a 100. Aquest límit ens podria fer pensar en una estructura de dades afitada, com els vectors o vectors ordenats ja que, en principi, les llistes encadenades ocupen més espai. Però no tenim cap garantia que aquests vectors estiguin sempre plens de fitxers. Tot el contrari. L'enunciat diu que el nombre de fitxers d'un repositori serà molt variable. Per tant, si optem per estructures estàtiques estarem malgastant molt d'espai.

En aquest cas, optarem per una llista encadenada. Tampoc no és una solució perfecte, ja que les operacions de cerca seran lineals, mentre que en un vector ordenat serien logarítmiques. Però el nombre màxim de fitxers serà de 100 i, per aquests volums tan petits, els algoritmes lineals són totalment acceptables.

Apartat d) [1 punt]

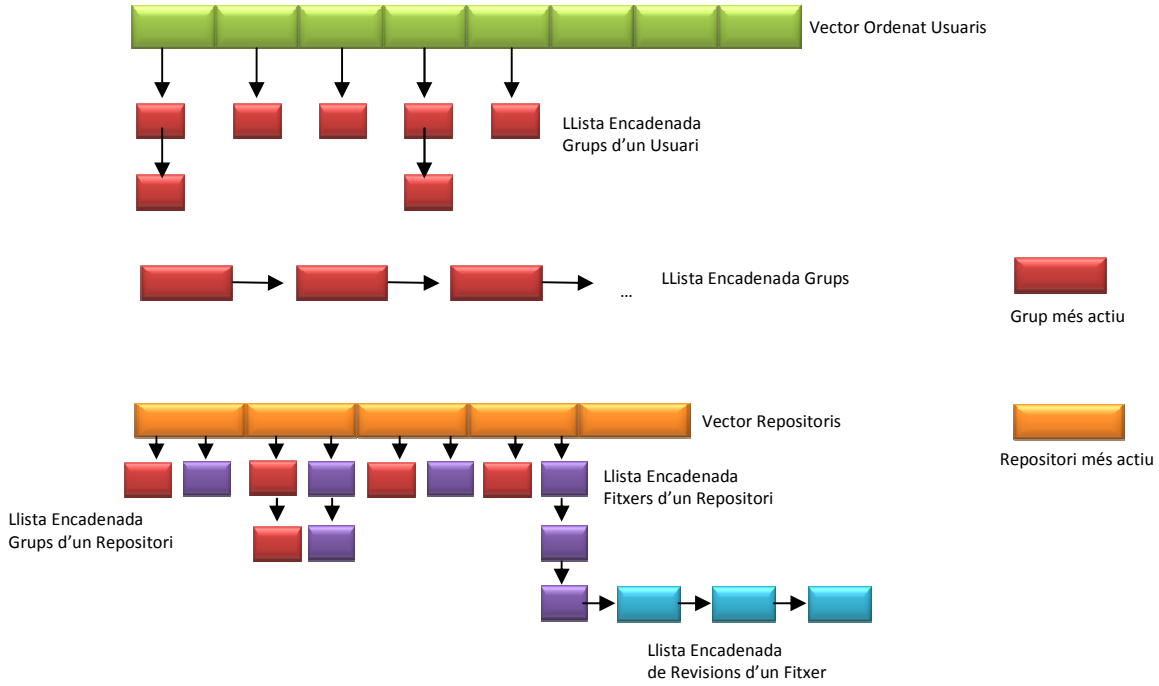
Justifiqueu la resta d'estructures de dades per fer el disseny del TAD. La justificació de cada una de les estructures de dades ha de ser de l'estil:
"Per guardar XXX triem una llista encadenada ordenada ja que el número d'elements no és gaire gran, ens fa falta accés directe i ens calen recorreguts ordenats."

Solució

- Per guardar els grups d'un usuari, utilitzarem una llista encadenada, ja que el nombre de grups d'un usuari és petit però variable.
- Per guardar els grups utilitzarem una llista encadenada, ja que el nombre de grups va en constant augment i no sabem el límit.
- Per guardar els usuaris d'un grup no utilitzarem res, ja que no s'utilitza a cap operació.
- Per guardar els grups d'un repositori utilitzarem una llista encadenada, ja que serà variable i no en sabem el límit.
- Per guardar les diferents revisions d'un fitxer utilitzarem una llista encadenada, ja que serà variable i il·limitat.
- Per saber el repositori més actiu, guardarem el nombre d'actualitzacions de cada repositori, i un màxim global.
- Per saber el grup més actiu, guardarem el nombre d'actualitzacions de cada grup i un màxim global.

Apartat e) [1 punt]

Feu un dibuix de l'estructura de dades global pel TAD *Svn* on es vegin clarament les estructures de dades que trieu per representar cada una de les parts i les relacions entre elles. Cal que feu el dibuix de l'estructura complerta, amb totes els estructures que us permetin implementar les operacions definides a l'especificació.



Exercici 3 [3 punts]

A l'exercici 1 heu definit l'especificació del TAD *Svn* amb les seves operacions i a l'exercici 2 heu triat les estructures de dades per cada part del TAD. En aquest exercici us demanem que us fixeu en els algorismes que us serviran per implementar algunes de les operacions especificades i en l'estudi d'eficiència de les mateixes. Tingueu en compte que la implementació de les operacions va estretament lligada a l'elecció de les estructures de dades que hagueu fet.

Apartat a) [1,5 punts]

Descriviu i feu l'estudi d'eficiència de l'operació que hagueu definit per actualitzar una nova revisió d'un fitxer en un repositori. Per fer-ho, heu de descriure breument el seu comportament indicant els passos que la componen (amb frases com ara: "inserir en l'arbre AVL / esborrar de la taula de dispersió / consulta del piló / ordenar el vector..."), dient l'eficiència asimptòtica de cada pas i donant l'eficiència total de l'operació.

Solució

- Cercar el repositori al vector de repositoris => O(1)
- Cercar l'usuari al vector ordenat d'usuaris => O(logU)
- Accedir als grups de l'usuari => O(1)
- Accedir als grups del repositori =>O(1)
- Comprovar que hi ha algun grup de l'usuari als grups del repositori =>O(GU*GR)
- Cercar el fitxer a la llista de fitxers del respositori =>O(FR)
- Si existeix:
    - Cercar la revisió a la llista de revisions del fitxer =>O(FC)
    - Si existeix retorna error O(1)
- Si no existeix:
    - Afegim un nou fitxer a la llista de fitxers del repositori =>O(1)
- Afegim una nova revisió a la llista de revisions del fitxer =>O(1)
- Incrementem el comptador d'actualitzacions del repositori =>O(1)
- Actualitzem el repositori més actiu, si s'escau =>O(1)
- Per cada grup de l'usuari: GU*
    - Incrementem el comptador d'actualitzacions del grup =>O(1)
    - Actualitzem el grup més actiu si s'escau =>O(1)

Per tant, el cost total de la operació seria de **O(logU+GU*GR+FR+FC)**

Apartat b) [1 punt]

Descriviu i feu l'estudi d'eficiència de l'operació que hagueu definit per obtenir el codi font d'un fitxer, d'un repositori, d'una revisió concreta. Com a l'exercici anterior, heu de descriure breument el seu comportament indicant els passos que la componen (amb frases com ara: "inserir en l'arbre AVL / esborrar de la taula de dispersió / consulta del piló / ordenar el vector..."), dient l'eficiència asimptòtica de cada pas i donant l'eficiència total de l'operació.

Solució

- Cercar el repositori al vector de repositoris => O(1)
- Cercar el fitxer a la llista de fitxers del repositori =>O(FR)
- Cercar la darrera revisió del fitxer anterior o igual a la revisió que ens demanen =>O(RF)
- Retornar el codi font de la revisió =>O(1)

Per tant, el cost total de la operació seria de **O(FR+RF)**
Nota: la precondició d'aquest mètode ja garanteix que el repositori, i el fitxer existeixen. Per això no hi ha condicions, a diferència de l'apartat anterior.

Apartat c) [0,5 punt]

Suposeu ara que volem oferir una funcionalitat que permeti que un usuari pugui eliminar un fitxer d'un repositori, a partir d'una revisió. Expliqueu quins canvis faríeu a l'estructura de dades i l'algorisme que utilitzaríeu.

Solució

Afegir un camp a cada fitxer per definir la revisió final, a partir de la qual el fitxer no existeix.

Exercici 4 [1 punt]

Indiqueu quins dels TADs de la biblioteca de TADs de l'assignatura et semblen més adients per utilitzar-los en la implementació de cada una de les estructures de dades definides pel TAD *Svn*.

Solució

- Per guardar els grups, grups d'un usuari, grups d'un repositori, fitxers d'un repositori i revisions d'un fitxer utilitzarem una *LlistaEncadenada*.
- Per guardar els repositoris utilitzarem un *array* de Java ja que només necessitem accés directe per posició.
- Per guardar els usuaris, implementarem una nova classe que implementi un vector ordenat amb operacions per afegir ordenadament i fer consultes mitjançant cerca dicotòmica. Per integrar aquesta classe a la biblioteca de classes d'EI, implementarem les interfícies de *ContenidorAfitat* i de *Diccionari*.