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 l'alumne posarà 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 cada un d'ells.

Format i data de lliurament

Data de publicació

7 d'octubre de 2013

Data de lliurament

21 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 servei de "bicing". 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'estacions de bicicletes S inicialment és petit i conegut, d'unes desenes.
- El nombre de places d'una estació P és petit i conegut en el moment de donar d'alta l'estació.
- El nombre de bicicletes B és conegut i relativament petit, d'uns centenars.
- El nombre de bicicletes aparcades en una estació BS és petit però molt variable.
- El nombre de places de pàrquing de totes les estacions és força més gran que B, per garantir poder aparcar les bicicletes amb certa facilitat.
- El nombre d'usuaris U del servei serà relativament petit, però anirà augmentant lentament.
- El nombre de serveis realitzats per una bicicleta BU serà gran i anirà en constant augment.
- El nombre de serveis realitzats per un usuari UU serà petit però anirà en augment.

Exercici 1 [2'5 punts]

Especifiqueu un TAD *Bicing* que permeti les operacions:

- Afegir una nova estació al sistema a partir del seu codi identificador (natural), coordenades geogràfiques (latitud i longitud), i el nombre de pàrquings de bicicletes que tindrà.
- Afegir una nova bicicleta. Caldrà especificar el codi identificador de la bicicleta (string amb la matrícula), el codi de l'estació i el número de plaça de pàrquing on s'aparcarà inicialment.
- Afegir un nou usuari al sistema. De cada usuari en sabrem el seu codi identificador (natural) i el seu nom.
- Obtenir una bicicleta lliure d'una estació per a un usuari. El sistema buscarà la bicicleta menys utilitzada, de les que estan aparcades a l'estació en aquell moment.
- Registrar que una bicicleta ha estat aparcada (i, per tant, retornada) a una estació determinada.
- Consultar els serveis realitzats per una bicicleta determinada en ordre cronològic.
- Consultar quina és l'estació més utilitzada.
- Consultar quin és l'usuari que ha agafat més cops la bicicleta.

Apartat a) [1 punt]

Dóna la signatura del TAD *Bicing*. És a dir, indica el nom que donaries a les operacions encarregades de cada funcionalitat requerida. Indica, també, quins caldria que fossin els paràmetres d'entrada i quina la sortida en cas que es necessités.

Solució

addStation(stationID, latitude, longitude, nParkings)

addBicycle(bicycleID, stationID, nParking)

addUser(userID, name)

Bicycle getBicycle(userId, stationID)

returnBicycle(bicycleID, stationID)

Iterador servicesByBicycle(bicyclesID)

Station mostActiveStation()

User mostActiveUser();

Apartat b) [1,5 punts]

Fes l'especificació contractual de les operacions del TAD *Bicing*. En la redacció de l'especificació pots fer servir, si et cal, qualsevol de les operacions del TAD. Pren 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 no existeix cap estació amb l'identificador especificat

@post les estacions són les mateixes més una nova estació amb les dades indicades

addStation(stationID, latitude, longitude, nParkings)

@pre no existeix cap bicicleta amb l'identificador especificat, existeix una estació amb el codi especificat i té el número de pàrquing lliure.

@post l'estació identificada stationID té les mateixes bicicletes més una de nova amb les dades inidicades.

addBicycle(bicycleID, stationID, nParking)

@pre no existeix cap usuari amb l'identificador especificat

@post els usuaris són els mateixos més un nou usuari amb les dades especificades

addUser(userID, name)

@pre existeix un usuari i una estació amb els identificadors especificats i, com a mínim, una bicicleta lliure

@post l'estació stationID té les mateixes bicicletes excepte la menys utilitzada que anomenem X. L'usuari té assignada la bicicleta X. Retorna la bicicleta X.

Bicycle getBicycle(userId, stationID)

@pre existeix una bicicleta i estació amb els identificadors especificats. L'estació té, com a mínim, una plaça lliure. La bicicleta, en aquest moment, està assignada a un usuari (és a dir, no està aparcada a cap estació).

@post l'estació stationID té les mateixes bicicletes més la que retorna l'usuari. L'usuari no té cap bicicleta assignada.

returnBicycle(bicycleID, stationID)

@pre existeix una bicicleta amb l'identificador indicat

@post retorna un iterador per recórrer els serveis fets per la bicicleta.

Iterador servicesByBicycle(bicyclesID)

@pre cert

@post retorna l'estació amb més serveis fets.

Station mostActiveStation()

@pre cert

@post retorna l'usuari amb més serveis fets.

User mostActiveUser();

Nota: A tot l'exercici hem especificat com a precondicions algunes comprovacions que es podrien fer dins les operacions. També seria correcte considerar que les comprovacions les fa la mateixa operació, en aquest cas, les precondicions no haurien d'aparèixer i les potscondicions haurien d'ampliarse (o afeblir-se) per definir l'estat quan les condicions fallen.

Exercici 2 [3,5 punts]

A l'Exercici 1 heu definit l'especificació d'un nou TAD, el TAD *Bicing*. 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.

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, una llista encadenada o una llista encadenada ordenada per a emmagatzemar les estacions. Justifiqueu quina creieu que és la millor opció.

Solució

L'enunciat ens diu que el nombre d'estacions S és conegut i petit. Per volums petits de dades l'estructura de dades no és gaire important, ja que totes seran ràpides. En tot cas, l'opció més simple (i també la més eficient) és utilitzar un vector on cada estació ocupa la posició que defineix el seu identificador. Així, la cerca d'una estació pel seu identificador tindrà cost constant O(1).

Apartat b) [0,5 punts]

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

Solució

L'enunciat ens diu que el nombre de bicicletes B és relativament petit i conegut. Amb volums de dades al voltant del centenar l'elecció de l'estructura de dades no és crítica. Així doncs, qualsevol estructura de dades seria acceptable ja que no representaria una pèrdua d'eficiència important. Tot i així, l'estructura mes adequada seria un vector ordenat, ja que ocupa l'espai just i necessari i permet fer cerques dicotòmiques O(logB) per localitzar bicicletes a partir de la seva matrícula. En un vector sense ordenar hauríem de fer recorreguts lineals fins a trobar la bicicleta O(B). Les llistes encadenades, tot i ser ordenades, no permeten fer cerques dicotòmiques i, per tant, també hauríem de fer recorreguts lineals O(B).

Apartat c) [0,5 punts]

Dubtem entre utilitzar un vector, un vector ordenat, o una llista encadenada per a emmagatzemar les bicicletes aparcades en una estació. Justifiqueu quina creieu que és la millor opció.

Solució

L'enunciat ens diu que el nombre de bicicletes aparcades en una estació BS és petit (ja que P és petit) però variable. Com que el volum de dades és petit, no hi hauria grans diferències entre les estructures proposades, però seria la llista encadenada la que millor s'adaptaria a la variabilitat de BS. Utilizar un vector, ordenat o no ordenat, permetria guardar tantes bicicletes com places de pàrquing, però l'enunciat ens diu que B és força més petit i, a més, durant el dia no totes les bicicletes estan aparcades.

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."

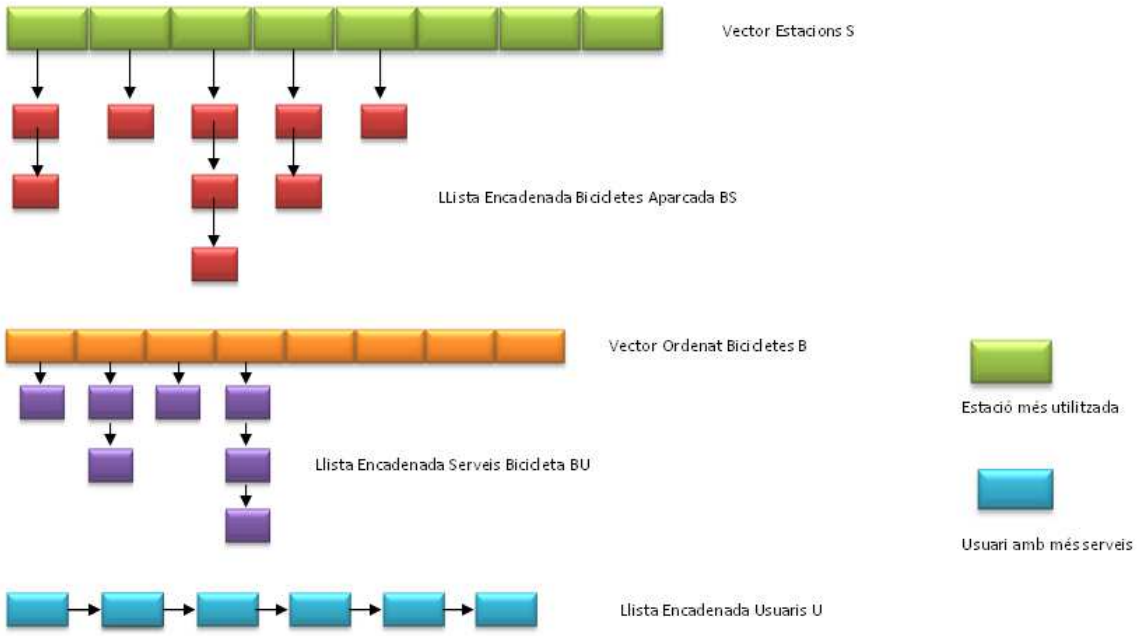
Solució

- Per guardar les estacions utilitzarem un vector ja que el nombre d'estacions és petit i conegut.
- Per guardar les bicicletes utilitzarem un vector ordenat, ja que el nombre de bicicletes és conegut i podrem fer cerques dicotòmiques per matrícula.
- Per guardar les bicicletes aparcades en una estació utilitzarem una llista encadenada ja que, tot i petit, és molt variable.
- Per guardar els usuaris utilitzarem una llista encadenada ja que el nombre d'usuaris és variable. Els vectors permetrien cerques més eficients, però el nombre d'usuaris anirà creixent indefinidament i això n'impedeix el seu ús.
- Per guardar els serveis d'una bicicleta utilitzarem una llista encadenada ja que aniran en constant augment i no es poden afitar.
- Per saber l'estació més activa, guardarem el nombre de serveis de cada estació, i un màxim global.
- Per saber l'usuari més actiu, guardarem el nombre de serveis de cada usuari, i un màxim global.

Apartat e) [1 punt]

Feu un dibuix de l'estructura de dades global pel TAD *Bicing* 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.

Solució



Exercici 3 [3 punts]

A l'Exercici 1 heu definit l'especificació del TAD *Bicing* 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 lligat a l'elecció de les estructures de dades que hagueu fet.

Apartat a) [1 punts]

Descriviu i feu l'estudi d'eficiència de l'operació que hagueu definit per obtenir una bicicleta per un usuari. 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 l'usuari a la llista d'usuaris => O(U)
- Cercar l'estació al vector d'estacions => O(1)
- Cercar la bicicleta menys utilitzada a l'estació => O(BS)
- Eliminar la bicicleta de l'estació => O(1) (combinant-ho amb l'operació de cerca)
- Associar l'usuari i la bicicleta => O(1)
- Crear un nou servei definint l'hora i estació d'inici =>O(1)
- Afegir el servei a la llista de serveis de la bicicleta =>O(1)
- Augmentar el nombre de serveis de la bicicleta => O(1)
- Augmentar el nombre de serveis de l'estació =>O(1)
- Actualitzar l'estació més activa si s'escau => O(1)
- Augmentar el nombre de serveis de l'usuari => O(1)
- Actualitzar l'usuari més actiu si s'escau =>O(1)

Per tant, el cost total de la operació seria de **O(U+BS)**

Apartat b) [1 punt]

Descriviu i feu l'estudi d'eficiència de l'operació que hagueu definit per retornar una bicicleta. 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 la bicicleta a la llista de bicicletes => O(logB)
- Cercar l'estació al vector d'estacions =>O(1)
- Desassociar usuari i bicicleta => O(1)
- Cercar el darrer servei afegit a la bicicleta=> O(1)
- Definir l'hora d'arribada i estació d'arribada del servei => O(1)
- Actualitzar el temps d'utilització de la bicicleta => O(1)
- Afegir la bicicleta a l'estació => O(1)

Per tant, el cost total de la operació seria de **O(logB)**

Apartat c) [1 punt]

Suposeu ara que volem oferir una funcionalitat que permeti que un usuari pugui reservar una bicicleta aparcada durant mitja hora. Si al cap de mitja hora l'usuari no ha agafat la bicicleta, aquesta torna a quedar lliure. Expliqueu quins canvis faríeu a l'estructura de dades i l'algorisme que utilitzaríeu.

Solució

Per permetre reserves, caldria diferenciar les bicicletes aparcades lliures de les reservades. Així, en comptes d'una llista encadenada de bicicletes aparcades, podríem afegir una segona llista amb les bicicletes reservades. I, per cada bicicleta reservada, hauríem de saber l'hora de caducitat de la reserva.

Quan un usuari sense reserva sol·licita una bicicleta, haurem de buscar la bicicleta lliure menys utilitzada. Com que no tenim cap garantia que aquesta bicicleta és una de les lliures, o de les reservades caducades, haurem de recórrer les dues llistes.

Si l'usuari havia fet una reserva, començarem la cerca per la llista de bicicletes reservades. Si encara hi és (si la reserva no ha caducat o ha caducat però ningú l'ha agafat), li assignem, sinó continuem com si no hagués fet la reserva.

Exercici 4 [1 punt]

Indica 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 *Bicing*.

Solució

- Per guardar les bicicletes d'una estació, els serveis d'una bicicleta i els usuaris, utilitzarem una *LlistaEncadenada*.
- Per guardar les estacions utilitzarem un *array* de Java ja que només necessitem accés directe per posició.
- Per guardar les bicicletes, 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, implementaria les interfícies de *ContenidorAfitat* i de *Diccionari*.