Distribució de productes a un supermercat

Descripció de classes

Identificador de l'equip: subgrup-prop41.1
Antonia Laura Apolzan: antonia.laura.apolzan@estudiantat.upc.edu
Hajweria Hussain Shaheen: hajweria.hussain@estudiantat.upc.edu
Ariadna Mantilla Puma: ariadna.mantilla@estudiantat.upc.edu
Guillem Sturlese Ruiz: guillem.sturlese@estudiantat.upc.edu

Índex

Classe Producte

Classe i atributs

Com tots sabem, l'experiència d'anar a un supermercat es basa en un element clau: comprar un producte (o molts!). Els productes són la base de tot i en el nostre supermercat, aquests son molt més que simples objectes estàtics que el client pot comprar.

És aquí on entra en joc la classe Producte, la qual permet que cada article tingui uns atributs propis, com són un codi identificador únic (Integer) i un nom (String), una posició específica dins de la prestatgeria, representada per les coordenades de fila i columna. Fins i tot, cada producte pot establir relacions de similitud amb altres productes a través d'un Map<Integer, Double>, creant una xarxa de connexions que ens permet gestionar-los de manera eficient. Gràcies a aquesta classe, els dotem d'una informació rellevant de cara a establir un sistema organitzat, intel·ligent i òptim que permet afavorir les vendes dels productes.

Cal tenir en compte que en aquesta primera entrega, hem decidit que la disposició dels productes en la prestatgeria del nostre supermercat, es dona en una sola fila i només es té en compte el número de la columna per a referir-nos a la posició d'aquest.

Funcions de la classe

- private void validarId(int id)
- private void validarSimilitud(double similitud)
- Producte(int id, String nom)
- Producte(int id, String nom, Map<Integer, Double> similituds)

Quan es crea un producte, ens assegurem que tot estigui en ordre. Així, hem de validar l'id perquè sigui superior a zero, mentre que el mapa de similituds, si es proporciona, es revisa per garantir que cap valor sigui negatiu. Si no es proporciona, inicialitzem el producte sense cap similitud associada. Per a tots els productes, en un principi, els dotem d'una posició per defecte, que no representa la seva posició final en la prestatgeria (no és una posició vàlida en aquesta encara). Si tot està correcte, el producte ja es pot construir. En cas contrari, llancem una excepció amb un missatge concret que informa a l'usuari d'un error a l'hora de realitzar aquest procés.

- public int getId()
- public String getNom()
- public int getFila()
- public int getColumna()
- public Map<Integer, Double> getSimilituds()

Amb aquestes funcions podem obtenir informació dels atributs d'un producte, ja sigui el seu identificador, el nom, la posició o les similituds amb la resta de productes.

- public void setId(int id)
- public void setNom(String nom)
- public void setFila(int fila)
- public void setColumna(int columna)
- public void setSimilituds(Map<Integer, Double> similituds)

Amb aquestes funcions podem actualitzar o modificar els atributs d'un producte, tenint en compte les restriccions de l'identificador i de les similituds que es poden assignar.

- public void afegirSimilitud(int id, double similitud)
- public void modificarSimilitud(int id, double nova_similitud)
- public double getSimilitud(int id)
- public int getIdProducteMillorSimilitud()

Aquest grup de funcions ens permet gestionar les relacions de similitud d'un producte amb operacions per afegir, modificar, obtenir aquest atribut o determinar amb quin producte té la millor similitud. En el cas que no es compleixi que el valor de la similitud sigui positiu o el producte no existeix llancem una excepció o un missatge informant a l'usuari que no s'ha pogut realitzar aquest procediment.

- public void imprimirSimilituds()
- public String toString()

Com a altres funcionalitats, podem convertir un producte en una representació textual simple, a l'hora de necessitar una descripció general d'aquest, amb l'identificador i el nom, i també podem mostrar totes les similituds d'un producte, enumerant la seva relació amb la resta de productes.

Classe CjtProductes

Classe i atributs

Ja hem establert que en el nostre supermercat, tenim diversos productes amb informació detallada dels seus atributs. Per aquest motiu, és essencial tenir un mecanisme que els organitzi, els relacioni i els gestioni de forma eficient.

Això és el que fa precisament la classe CjtProductes, actuar com un contenidor que agrupa tots els productes en un Map<Integer, Producte>, que els emmagatzema mitjançant la seva identificació (id) com a clau i l'objecte Producte com a valor (amb tota la seva informació), el qual permet un accés ràpid i ordenat als productes. Aquest conjunt de productes és associat a un usuari específic (String) a través d'un nom i funciona com el "propietari" d'aquest conjunt de productes concret. D'aquesta manera, podem garantir una gestió dinàmica i precisa de tots els productes.

Funcions de la classe

- public CjtProductes(String usuari)

Amb aquesta función creadora associem un conjunt de productes, inicialment sense cap producte, a un usuari concret.

- public Map<Integer, Producte> getProductes(String nomUsuari)
- public Producte getProducte(int idProd)
- public Producte[] getVecProductes()

Aquestes funcions ens permeten realitzar consultes de diversos aspectes: ens pot retornar el conjunt de productes associat a un usuari concret, ens pot proporcionar un producte específic segons el seu id o directament, ens retorna un vector amb tots els productes del conjunt.

- public boolean existeixProducte(int idProd)

A través d'aquesta funció, podem determinar si un producte amb un identificador concret es troba dins del conjunt de productes. Una funció senzilla però fonamental per garantir el correcte funcionament de les següents operacions, com afegir, editar o eliminar productes.

- public void afegirProducte(Producte p)

Aquesta funció ens permet afegir un nou producte, sempre que no en tingui ja un amb el mateix id. Si el producte inclou les similituds amb altres productes, també s'actualitzen les relacions bidireccionals de similitud, per tal de mantenir la informació de forma correcta entre tots els productes. Si no té similituds, es genera un error que indica a l'usuari que cal associar-ne almenys una.

- public void editarProducte(Producte p)
- public void editarIdProducte(int idProd, int nou_idProd)
- public void editarNomProducte(int idProd, String nou_nom)
- public void editarPosProducte(int idProd, int nova_pos)

Aquest conjunt de funcions ens permet realitzar diverses operacions d'edició dels atributs d'un producte del conjunt. Tenim opcions diferents segons el que necessitem modificar. D'una banda, podem canviar l'identificador d'un producte i actualitzar totes les relacions de similitud que altres productes tinguin amb ell, per tal de no perdre la resta de la informació del producte original. També podem modificar el nom d'un producte o actualitzar la posició d'un producte (com ja hem mencionat, en aquesta entrega tan sols es modifica el valor de la columna com a representació de la posició d'un producte).

- public void eliminarProducte(int idProd)

Mitjançant aquesta funció, podem treure un producte del conjunt. A més, elimina qualsevol relació de similitud que altres productes tinguessin amb ell, així com en la funció anterior ens assegurem que el sistema mantigui la coherència.

- public void modificarSimilitud(int idProd1, int idProd2, double nova_similitud)

Amb aquesta funció podem actualitzar la similitud entre dos productes. Si no hi ha una relació prèvia, un dels dos o ambdós productes no existeixen, es genera un error.

- public Map<Integer, Double> getSimilituds(int idProd)
- public double[][] getMatriuSimilituds()

Aquestes dues funcions les utilitzem, la primera, per obtenir les similituds d'un producte concret amb la resta, i la segona, per generar una matriu que representa totes les similituds entre els productes del conjunt, una operació molt útil per a realitzar anàlisis posteriorment i que farem servir com a entrada en els algoritmes per tal de determinar la millor disposició dels productes en la prestatgeria.

Classe Usuari

Classe i atributs

La classe Usuari representa els usuaris de l'aplicació, gestionant la seva informació personal i permetent accedir a les seves credencials. Un usuari està definit pels següents atributs. Un username, un atribut de tipus String que emmagatzema el nom d'usuari i una contrasenya, que es un atribut de tipus String que guarda la contrasenya de l'usuari.

Funcions de la classe

- public Usuari (String name, String contra)

Aquesta creadora inicialitza un objecte Usuari amb els atributs username i contrasenya. Rep per paràmetre el nom d'usuari (name) i la contrasenya (contra), i els assigna als atributs corresponents.

- public Usuari getUsuari()
- public String getUsername()
- public String getContrasenya()

Amb aquestes funcions podem obtenir informació dels atributs d'un usuari, ja sigui el seu username, la contrasenya, o l'objecte Usuari.

- public void canviaUsername(String username)
- public void canviaContrasenya(String contra)

La primera funció ens permet canviar el username de l'usuari (tot i què les hem codificat, com a la primera entrega no hi ha capa de dades, no son molt útils encara) i la segona funció permet canviar la contrasenya de l'usuari.

Classe CjtUsuaris

Classe i atributs

Donat un sistema organitzat d'aquesta manera, una classe CjtUsuaris és una peça fonamental per tal de representar el conjunt d'usuaris presents en el sistema. Aquesta ens permet realitzar operacions de gestió i control de la informació dels usuaris associats, el

qual facilita que es mantingui un ordre i coherència en el nostre projecte, ja que dependrà precisament de diversos usuaris (en les pròximes entregues es desenvoluparà que l'usuari pugui guardar, carregar o modificar la informació pertinent sense perdre cap mena de dada).

Mitjançant un identificador únic per al conjunt d'usuaris (String), podem diferenciar aquest conjunt d'altres i a través d'un Map<String, Usuari>, som capaços d'associar els noms d'usuari amb els objectes de tipus Usuari, de forma que actua com una base de dades per accedir als usuaris de manera ràpida i eficient.

Aquesta classe juntament amb la classe Usuari requerirà d'una revisió després d'aquesta entrega, per tal de gestionar els casos d'ús de carregar i guardar informació associada als usuaris.

Funcions de la classe

- public CjtUsuaris(String id)
- public CjtUsuaris(String id, String nomUsuari)
- public CjtUsuaris(String id, String nomUsuari, String contrasenya)
- public CjtUsuaris(String id, Usuari usr)

Amb aquest conjunt de funcions constructores de la classe, podem crear un conjunt d'usuaris buit amb l'identificador especificat, per tal d'inicialitzar-lo abans d'afegir-hi usuaris o ja registrant al conjunt un usuari amb nom, amb contrasenya o sense. Si en té podem associar un objecte Usuari amb el nom i la contrasenya proporcionats. També, tenim una opció directa de relacionar un objecte Usuari (amb la informació pertinent) al conjunt.

- public String getIdCjtUsuaris()
- public Set<String> getUsuaris()
- public Usuari getUsuari(String nomUsuari)
- public void setIdCjtUsuaris(String id)

Utilitzem aquests mètodes per consultar o modificar els atributs de la classe, així ens permet obtenir l'identificador del conjunt, el conjunt en si de noms d'usuaris presents en el conjunt, entre d'altres opcions.

- public void crearUsuari(String nomUsuari, String contrasenya)
- public void crearUsuari(Usuari usr)

Aquestes funcions ens permeten crear un usuari nou a partir del seu nom i contrasenya que s'afegeix al conjunt, així com que ens permet afegir directament un objecte Usuari, prèviament creat, al conjunt.

- public void modificarUsuari(String antic_nomUsuari, String nou_nomUsuari, String nova_contrasenya)
- public void modificarNomUsuari(String antic_nomUsuari, String nou_nomUsuari)
- public void modificarContrasenya(String nomUsuari, String nova_contrasenya)

Per exemple, si un usuari vol actualitzar les seves credencials, a través d'aquestes funcions podem modificar tant el nom d'usuari com la seva contrasenya, actualitzant (si cal) la clau al mapa.

- public void eliminarUsuari(String nomUsuari)
- public void eliminarTotsUsuaris()

Si volem esborrar un usuari específic del conjunt, o el que seria donar de baixa un compte, o simplement esborrar tots els usuaris del conjunt, podem fer servir aquestes funcions de reinicialització.

Classe Prestatgeria

Classe i atributs

La nostra classe Prestatgeria funciona com l'estructura que fem servir en un supermercat, amb l'objectiu d'organitzar els productes en una distribució òptima i funcional. La prestatgeria guarda una identificació única, amb un número id i la informació de la quantitat de productes que conté amb l'atribut numèric numProductes, i disposa d'un array que representa el layout, on cada compartiment d'aquesta guarda un producte en concret.

Funcions de la classe

- public Prestatgeria(int id, int nProd)

Amb aquest constructor creem una nova prestatgeria, que ens permetrà organitzar els productes des de zero. D'entrada, cada compartiment de l'array està totalment buit.

- public Producte[] getLayout()

La disposició dels productes en la prestatgeria es pot consultar amb aquesta funció, on cada compartiment representa una posició. Això ens dona una vista de tots els productes de manera visual.

- public int getId()
- public int getNumProductes()
- public void setId(int id)
- public void setNumProductes(int nProd)
- public void setLayout(Producte[] disposicio)

Amb aquest conjunt de funcions podem consultar el atributs de la classe, així com assignar-ne de noves. També som capaços de redefinir la capacitat de la prestatgeria (tot i que no omple de forma automàtica amb nous productes ni buida els existents) o de reconfigurar tota la prestatgeria d'un sol cop, donant-nos l'opció de definir un nou layout.

- public void intercanviarDosProductes(int posProd1, int posProd2)

Quan cal canviar dos productes de lloc, podem fer servir aquesta funció, la qual s'assegura que el sistema es mantingui sempre coherent amb les ubicacions dels productes.

- public void eliminarPrestatgeria()

Quan una prestatgeria, o millor dit, la disposició dels productes en aquesta ja no ens interessa, fem ús d'aquesta funció que s'encarrega d'esborrar-la completament, fent desaparèixer el seu contingut. És a dir, la prestatgeria queda buida (sense cap producte col·locat) i ens deixa el sistema preparat per altres operacions, per exemple, establir una nova disposició.

Classe CtrlDomini

Classe i atributs

La classe CtrlDomini actua com un controlador centralitzat que connecta usuaris, productes i prestatgeries, i és el responsable de garantir que tot el sistema funcioni correctament i el nostre supermercat sigui el més òptim possible (així podrem vendre més productes!). Aquesta classe implementa el patró Singleton, assegurant que tan sols hi hagi un únic "director" que gestioni totes les operacions a la vegada.

Funcions de la classe

- public CtrlDomini()
- public void inicialitzarCtrlDomini()

Quan es crea un objecte de CtrlDomini, aquest crida al segon mètode per preparar els conjunts d'usuaris i altres elements essencials.

- public static CtrlDomini getInstance()

Fent ús d'aquest mètode podem garantir, com ja hem mencionat anteriorment, que hi hagi una sola instancia d'aquesta classe, de forma que evitem qualsevol tipus de desajust en la gestió del sistema.

- public void iniciarSessio(String username, String pwd)
- public void crearUsuari(String name, String pwd)
- public void canviarUsuari(String nomUsuari, String pwd)

El nostre controlador ens permet gestionar els usuaris d'una forma ben estructurada i senzilla de manejar, on podem controlar l'autenticació d'aquest o la possibilitat d'alternar entre usuaris. En el cas que un usuari no existeixi, el sistema crea un compte nou. També podem registrar nous usuaris amb un nou i una contrasenya corresponents i iniciar sessió en aquell mateix moment.

- public Producte[] llistarProductesUsuari()
- public Producte[] llistarPrestatgeriaUsuari()

Amb aquests mètodes podem veure el conjunt de productes disponibles que té l'usuari actual o el contingut que presenta una prestatgeria.

- public void crearProducte(int id, String nom, Map<Integer, Double> similituds, Boolean bruteForce)
- public void crearPrestatgeria(Boolean bruteForce)
- public void modificarProducte(Integer idProdActual1, Integer nouId, String nouNom)
- public void modificarSimilituds(Integer idProdActual1, Integer idProdActual2, double novaSim, Boolean bruteForce)
- public void modificarPrestatgeria(int pos1, int pos2)
- public void esborrarProducte(int id, Boolean bruteForce)
- public void esborrarPrestatgeria()

Aquest conjunt de funcions ens permet gestionar els casos d'ús dels productes i de les prestatgeries, realitzant operacions de creació, modificació o eliminació corresponents a cada objecte. Donem la possibilitat d'insertar nous productes al sistema amb la seva informació i de generar una prestatgeria per tal de acomodar-los, així com d'editar qualsevol dels atributs d'un producte, ja sigui l'id, el nom, la posició en la que es troba o, fins i tot, podem canviar la similitud que hi ha entre dos productes (aquesta es modificarà correctament si es tracta d'un número vàlid entre 0 i 1). Així mateix, podem eliminar un producte i actualitzar la prestatgeria.

Al crear una prestatgeria, utilitzem un algorisme de generació per disposar els productes segons les seves similituds. Donem l'opció a l'usuari de poder triar entre un enfocament de força bruta amb l'algoritme de BruteForce o un algorisme d'aproximació de DosAproximacio. Addicionalment, es pot reorganitzar manualment la prestatgeria intercanviant la posició de dos productes o buida completament la prestatgeria.

- public String getUsuariActual()

Aquest mètode és un mètode directe per tal d'obtenir el nom d'usuari de l'usuari que està connectat al sistema en un moment donat.

- public void esborrarUsuari()

Si un usuari ja no vol estar en el sistema, hi ha l'opció de poder eliminar-lo completament.

- public void tancarSessio()

Amb aquesta funció podem desconnectar l'usuari actual del sistema i restablir l'objecte UsuariActual, és a dir, restablim l'estat del controlador perquè no hi hagi cap usuari actiu.

Classe GeneradorSolucio

Classe i atributs

La classe GeneradorSolucio és una interfície que defineix els acords que les classes dels algoritmes hauran de seguir. Aquesta interfície proporciona els mètodes necessaris per generar i obtenir els resultats d'un layout de productes, així com per obtenir la millor similitud calculada entre els productes.

Funcions de la classe

La primera funció crida a l'algoritme BruteForce o DosAproximacio segons la configuració desitjada, i retorna la millor disposició dels productes. Les altres dues funcions proporcionen el resultat un cop l'algoritme ha estat executat, permetent obtenir el resultat sense necessitat de cridar de nou a l'algoritme, així com la millor similitud total obtinguda per aquest resultat.

- Producte[] generarLayout();
- Producte[] getResultat();
- double getMillorSimilitud();

Classe BruteForce

Classe i atributs

La classe BruteForce implementa un algorisme de força bruta per resoldre problemes d'optimització en la disposició de productes. Aquesta classe forma part del paquet Domini i implementa la interfície GeneradorSolucio. El seu objectiu principal és determinar la millor configuració de productes que maximitzi la similitud entre productes. Per aconseguir-ho, utilitza una matriu de similituds, matSimilituds, que defineix les relacions entre els productes; un vector de productes, vecProductes, que conté els elements a ordenar; i dos atributs principals per emmagatzemar els resultats: millorSimilitud, que guarda el valor màxim de similitud trobat, i vecResultat, que conté la configuració òptima de productes corresponent a aquesta similitud.

Funcions de la classe

- BigInteger factorial(int n);

Aquesta funció calcula el nombre de permutacions que s'hauran de fer. L'ús de BigInteger garanteix que es puguin tractar números grans sense desbordaments, ja que el nombre de permutacions creix ràpidament amb el nombre de productes.

- double calcularSimilitudTotal(int[] vecActualProd);

Aquesta funció calcula la similitud total d'una configuració específica de productes tenint en compte la geometria circular de la prestatgeria.

- void swap(int[] arr, int i, int j);

- void permutacions(int[] vProd, int L, int R, BigInteger[] numPermutacions);

Amb aquestes funcions calculem totes les permutacions possibles d'un subconjunt de productes i, per cada configuració, cridem a calcularSimilitudTotal i ens guardem la disposició si aquesta és la millor trobada fins al moment. S'ha inclòs un comptador (numPermutacions) per limitar el nombre màxim de permutacions.

- Producte[] generarLayout();

Aquesta funció inicia el procés per trobar la millor configuració de productes. Utilitza la funció de permutacions per explorar totes les opcions i retorna la configuració que maximitza la similitud. El seu disseny garanteix que sigui fàcil d'utilitzar des de fora de la classe, encapsulant tota la complexitat interna.

- Producte[] getResultat();
- double getMillorSimilitud();

Aquestes funcions permeten accedir directament al resultat òptim un cop completat el càlcul.

Classe DosAproximacio

Classe i atributs

La classe DosAproximacio és una implementació de l'algorisme d'aproximació per resoldre un problema de distribució o ordenació de productes, basant-se en les similituds entre ells. A través d'un conjunt d'operacions, la classe genera una configuració òptima de productes per maximitzar la similaritat total. Aquesta classe disposa d'un conjunt d'atributs que defineixen el seu comportament i l'estructura interna.

- **private int n:** Representa el nombre total de productes involucrats en el procés. Es calcula a partir de la longitud de la matriu de similituds,matS, i indica el nombre de nodes (productes) amb què es treballa en els càlculs.
- **private List<Aresta> llista_arestes:** És una llista que emmagatzema les arestes del graf de similituds entre els productes. Cada aresta és representada per un objecte de la classe Aresta que conté la informació sobre els dos productes connectats i la seva similitud corresponent. Aquesta llista es construeix a partir de la matriu de similituds inicial.

- **private int[] pare:** Un array utilitzat per gestionar les unions de conjunts al fer l'algoritme de Kruskal. Cada posició d'aquest array conté el pare d'un node dins d'un conjunt, facilitant les operacions de "find" i "uneix" que es realitzen a l'algorisme.

- **private int[] mida:** Un altre array utilitzat per optimitzar les operacions del Kruskal. En aquest cas, emmagatzema la mida dels conjunts, de manera que l'unió de conjunts es realitzi sempre de la manera més eficient, unint el conjunt més petit amb el més gran.

- **private Producte[] prods:** Un array que conté els productes involucrats en el càlcul. Cada producte es representa amb un objecte de la classe Producte. Aquest array s'usa per associar els productes amb les seves posicions en els càlculs de similitud.

- **private double sumaSimilitud:** Aquesta variable guarda el valor de la millor suma de similituds trobada durant l'execució de l'algorisme. Inicialment es defineix com a 0, i es va actualitzant a mesura que es calculen diferents disposicions possibles dels productes.

- **private Producte[] res_productes**: Un array que emmagatzema la millor configuració de productes trobada segons la suma de similituds. Aquest array es genera com a resultat final de l'algorisme i representa l'ordre òptim dels productes.

- **private double[][] matS:** És la matriu de similituds, on cada element [i][j] representa la similitud entre el producte [i] del vetor *prods* i el producte [j]. Aquesta matriu és essencial per calcular la qualitat de les diferents configuracions de productes.

Funcions de la classe

La funció pública principal generarLayout() busca generar un layout òptim per a una prestatgeria de productes circular, maximitzant la suma de similituds entre productes. Per fer-ho, comença construint un Maximum Spanning Tree utilitzant l'algoritme de Kruskal.

Un cop obtingut aquest MST, es converteix en un cicle eulerià que connecta totes les arestes de l'mst, mantenint una circularitat a la prestatgeria. Aquest cicle es crea recorrent les connexions del graf amb la funció findEuleria.

Finalment, per eliminar els nodes repetits de forma òptima, s'utilitza la funció calculaSuma per provar totes les configuracions possibles eliminant duplicats i calculant la suma de

similituds per a cada opció. Si es troba una configuració millor, es guarda com a resultat final. Finalment, es retorna el vector de productes que maximitza la similitud acumulada.

La funció pública getMillorSimilitud retorna la similitud del vector resultat, mentres que la funció pública getResultat retorna el vector de Productes resultat.