DISTRIBUCIÓ DE PRODUCTES A UN SUPERMERCAT

Versió 3.0

Grup 41.4

Sara Canaleta i Garcia - sara.canaleta.i@estudiantat.upc.edu
Ferran Cots Villegas - ferran.cots@estudiantat.upc.edu
Iker Díaz Tellez - iker.diaz.tellez@estudiantat.upc.edu
Pol Hernández Vigo - pol.hernandez.vigo@estudiantat.upc.edu

ÍNDEX

PRESENTACIÓ

CtrlPresentació

La classe CtrlPresentació s'encarrega de la gestió de la capa de presentació i coordinar les operacions entre diferents classes de l'aplicació. Els seus atributs són CtrlDominio cd per poder obtenir les dades de la capa de domini i transferir-les a la capa de presentació, i PantallaInici pi per poder inicialitzar la pantalla d'inici.

PantallaInici

La classe PantallaInici és una finestra principal d'una aplicació gràfica que serveix com a punt d'entrada per a la gestió de diferents aspectes d'un sistema. Aquesta finestra conté tres botons que permeten accedir a diferents pantalles de gestió: Productes, Prestatgeries i Distribució. Cada botó obre una nova finestra corresponent a la gestió d'aquests elements i tanca la finestra actual.

PantallaGestioProductes

La classe PantallaGestioProductes és una interfície gràfica que permet gestionar els productes del sistema. Des d'aquesta pantalla els usuaris poden veure el llistat de productes disponibles actualment i decidir si volen afegir-ne de nous, eliminar algun ja existent o modificar-lo. Aquesta classe es connecta amb el controlador de presentació per poder realitzar les operacions necessàries sobre les dades.

PantallaProducte

La classe PantallaProducte representa una interfície gràfica que permet crear un nou producte dins l'aplicació, especificant el seu nom i les similituds amb els altres productes ja existents. A més, carrega automàticament els productes existents per introduir de manera fàcil i pràctica les similituds necessàries. Aquesta interfície es comunica amb el controlador de presentació per validar i emmagatzemar la informació.

PantallaEliminar

La classe PantallaEliminar és una interfície gràfica que permet seleccionar i eliminar productes de l'aplicació. La pantalla mostra una llista dels productes existents i permet que l'usuari esculli quin vol eliminar. Aquestes accions es comuniquen amb el controlador de presentació per gestionar els productes.

PantallaSeleccionProducto

La classe PantallaSeleccionProducto és una interfície gràfica que permet a l'usuari seleccionar un producte d'una llista mostrada en una taula. Aquesta selecció es passa a una nova pantalla per modificar les similituds del producte seleccionat.

PantallaModificarSimilitudes

PantallaModificarSimilitudes és una classe de la capa de presentació que permet a l'usuari modificar els valors de similitud d'un producte seleccionat prèviament respecte als altres productes. Mostra una interfície gràfica basada en JFrame i conté una taula editable per ajustar els valors de les similituds.

PantallaGestióPrestatgeria

La classe PantallaGestióPrestatgeria és la interfície que s'implementa com el seu nom bé indica, per poder gestionar les prestatgeries guardades. Amb aquesta pantalla podrem escollir si volem visualitzar, modificar o eliminar la prestatgeria seleccionada.

PantallaGestióDistribució

PantallaGestióDistribució és una pantalla de la capa de Presentació que permet a l'usuari crear la distribució de la prestatgeria amb els productes que seleccioni, amb les files i columnes que es seleccionin i amb l'algorisme que es seleccioni.

PantallaSeleccionar

La classe PantallaSeleccionar és una interfície gràfica que permet a l'usuari seleccionar i deseleccionar productes d'una llista desplegable i visualitzar els productes seleccionats en una llista. Els productes seleccionats es poden confirmar per passar-los a una altra pantalla.

PantallaPrestatgeria

La classe PantallaPrestatgeria és una finestra gràfica que permet gestionar una prestatgeria, ja sigui per modificar-la, distribuir-la o visualitzar-la. Depenent del mode seleccionat (modificar, distribuir o visualitzar), es mostren diferents components i funcionalitats. Aquesta classe gestiona la creació i la visualització de la prestatgeria, així com les opcions per modificar la seva mida, guardar els canvis o intercanviar les files i columnes.

Prestatge

La classe Prestatge és un panell personalitzat que crea una representació visual d'una prestatgeria amb camps de text disposats en una matriu de files i columnes. Els usuaris poden interactuar amb els camps de text, seleccionant-ne fins a dos, i veure'n el canvi de color quan estan seleccionats. Aquesta classe permet gestionar visualment la disposició de productes o elements en una estructura de prestatgeria, amb la possibilitat de modificar l'aspecte dels camps seleccionats.

PantallaNouIdPrestatgeria

La classe PantallaNouIdPrestatgeria és una finestra gràfica que permet a l'usuari introduir un nou ID per a una prestatgeria. Depenent del mode (modificar o distribuir), la finestra es comporta de manera diferent. En el mode de modificació, es permet canviar l'ID de la prestatgeria, mentre que en el mode de distribució es permet establir un nou ID per a una prestatgeria en una distribució. Quan l'usuari introdueix l'ID i prem el botó "Guardar", el nou ID es guarda i la finestra es tanca.

DOMINI

Algoritmo Estanteria

Descripció general

La classe Algoritmo Estanteria és la interfície dels algoritmes que s'utilitzen, és a dir, dels algorismes d'aproximació i de força bruta. En aquesta classe es defineix el comportament esperat de l'algoritme encarregat d'organitzar els productes de la prestatgeria.

Mètodes Principals

- ejecutarAlgoritmo(ArrayList<Producto> productos, int filas, int columnas): Vector<Producto>
    - Executa l'algoritme encarregat d'organitzar productes i torna un vector amb els productes ordenats
- getCostoMaximo(): int
    - Retorna el cost de la prestatgeria

Aproximacion

Descripció general

La classe Aproximacion serveix per a trobar la disposició òptima del conjunt dels productes en una prestatgeria maximitzant la similitud acumulada entre els productes adjacents, buscant una solució aproximada al problema TSP. Aquesta solució consisteix en la creació d'un MST, la generació d'un cicle eulerià, la construcció d'un cicle hamiltonià (aconseguit mitjançant l'eliminació dels nodes repetits) i, finalment, l'ús de la millora 2-OPT.

Implementa la classe interna Arista, la qual serveix per encapsular les arestes del graf.

Mètodes Principals

- ejecutarAlgoritmo(ArrayList<Producto> productos, int filas, int columnas): Vector<Producto>
    - Executa l'algoritme i retorna el resultat però transformat en un vector de productes.
- getCostoMaximo(): Integer
    - Retorna el cost màxim de la distribució.
- encontrarRaiz(int vertice, Integer[] padres) : int

    - Retorna l'enter que representa l'arrel del conjunt al qual pertany el vèrtex d'entrada.
- conectar(int v1, int v2, Integer[] padres) : boolean
    - Retorna true si és possible realitzar amb èxit la unió de dos conjunts, i false si no ha estat possible unir els dos conjunts.
- aplicarKruskal(List<Arista> grafo, List<Arista> arbolMaximo, Integer[] padres)
    - Implementa l'algorisme de Kruskal per a generar el Minimum Spanning Tree i la guarda a arbolMaximo.
- realizarDFS(Map<Integer, List<Integer>> grafo, int vertice, List<Integer> ciclo)
    - Realitza una Depth First Search per a trobar un cicle eulerià de grafo i guarda el resultat a ciclo.
- eliminarRepetidos(List<Integer> listaFinal, List<Integer> ciclo)
    - Elimina els vèrtexs repetits de ciclo i els guarda a listaFinal.
- calcularCosto(List<Integer> listaVertices, List<Arista> aristas) : Integer
    - Retorna el cost total del cicle donat.
- hillClimbing(List<Integer> solucionini, List<Arista> grafo) : List<Integer>
    - Retorna una solució inicial mitjançant l'algorisme 2-OPT.
- a2opt(List<Integer> cycle, int i, int j) : List<Integer>
    - Retorna una el cycle modificat per a aplicar l'algorisme 2-OPT.
- ArbolMaximo(productos, nproductos): List<Producto>
    - Construeix una solució basada en un algorisme d'aproximació, retornant una llista de productes ordenats.

CtrlDominio

Descripció general

La classe CtrlDominio s'encarrega de la gestió de la capa de domini i coordinar les operacions entre diferents classes de l'aplicació. Aquesta classe proporciona una interfície tant per a gestionar les operacions dels productes i els algorismes d'optimització del prestatge.

Els seus atributs són GA ga, per a gestionar els algorismes d'optimització, i GestioProductes gp per a la creació i manipulació dels productes i les seves similituds.

Mètodes Principals

- getNumProd()

    - Retorna el nombre de productes que hi ha introduïts al sistema.
- afegirProducte(String nombreProducto, HashMap<String, Integer> similitudes)
    - Afegeix un nou producte amb les seves similituds.
- eliminarProducte(String nombreProducto)
    - Elimina un producte.
- getProductosControl(): ArrayList<Producto>
    - Retorna la llista de tots els productes.
- getAllProductosNombres(): ArrayList<String>
    - Retorna la llista amb els noms de tots els productes.
- getProductoControl(String name): Producto
    - Retorna el producte amb el nom especificat.
- setProductos (List<Map.Entry<String, HashMap<String, Integer>>> nombresConSimilitudes)
- modificarSimilitudEntreProductosControl(String n1, String n2, Integer valor)
    - Modifica la similitud entre dos productes amb nom n1 i n2.
- calcularAlgoritmo(String algoritm, int filas, int columnas)
    - Executa l'algorisme especificat (aproximació o força bruta) sobre els productes, verificant que el prestatge tingui prou espai pels productes i emmagatzemant la distribució del prestatge calculada a GA.
- getIDPrestatgeries(): Set<String>
    - Retorna el conjunt de Ids de totes les prestatgeries existents.
- getSimilitud(String id): int
    - Retorna l'índex de similitud de la prestatgeria amb l'id especificat.
- getColumnes(String id): int
    - Retorna el nombre de columnes de la prestatgeria amb l'id especificat.
- getFiles(String id): int
    - Retorna el nombre de files de la prestatgeria amb l'id especificat.
- crearPrestatgeria(String id, Vector<String> distribucio, int files, int columnes, int indexSimilitut)
    - Crea una prestatgeria amb els atributs especificats.
- eliminarPrestatgeria(String id)
    - Elimina la prestatgeria amb l'id indicat.
- modificarMidaPrestatgeria(String id, int files, int columnes)
    - Modifica les files i columnes de la prestatgeria amb l'id indicat.
- modificarDistribucioPrestatgeria(String id, Vector<String> distribucio)
    - Modifica la distribució de la prestatgeria amb l'id indicat.
- calculaIndexSimilitud(Vector<String> distribucio): int

    - Calcula i retorna l'índex de similitud de la distribució passada com a paràmetre.
- guardarProductops(String… customPath)
    - Guarda la informació dels productes existents a la ruta especificada.
- cargarProductos(String path)
    - Carrega els productes des d'un arxiu situat a la ruta especificada.
- guardarPrestatgeries()
    - Guarda la informació de les prestatgeries a la capa de persistència.
- cargarPrestatgeries(String path)
    - Carrega les prestatgeries des d'un arxiu situat a la ruta especificada.

FuerzaBruta

Descripció general

La classe FuerzaBruta implementa una solució per a trobar la disposició òptima del conjunt dels productes en una prestatgeria maximitzant la similitud acumulada entre els productes adjacents, passant per totes les distribucions possibles i retornant la que compleixi millor la condició de similitud.

Mètodes Principals

- ejecutarAlgoritmo(ArrayList<Producto> productos, int filas, int columnas): Vector<Producto>
    - Executa l'algoritme i retorna el resultat però transformat en un vector de productes.
- getCostoMaximo(): int
    - Retorna el cost màxim de la distribució.
- fuerza_bruta(int n_productos, ArrayList<Producto> productos, int tam1, int tam2) : String[][]
    - Retorna una matriu amb els noms dels productes disposats de manera òptima, és a dir, amb major o igual similitud a la resta de distribucions possibles.
- busqueda(String[][] s, ArrayList<Producto> productos, Boolean[] visitado, int similitud, int columna, int fila, int n_productos, Producto prod_ant) : int
    - Mètode recursiu que prova totes les combinacions possibles de productes i retorna la millor similitud trobada durant el càlcul.

- copiarMatriz(String[][] matriz) : String[][]
    - Retorna una còpia de la matriu d'entrada matriz.

GA

Descripció general

La classe GA (gestió d'algorismes) és un controlador que selecciona i executa diferents algorismes per a generar la distribució d'una prestatgeria de productes.

Mètodes Principals

- getCostMax(): int
    - Retorna el cost màxim de la distribució.
- getListaProductos(): Vector<Producto>
    - Retorna un vector amb els productes ordenats.
- GestionAlgoritmo(String algoritmoNombre, ArrayList<Producto> productos, int filas, int columnas)
    - Executa l'algorisme especificat (aproximació o força bruta) i emmagatzema el resultat a Prestatge. Si l'algorisme és vàlid, emmagatzema la distribució resultant i calcula el cost màxim i el temps d'execució de l'algorisme.

GestioPrestatgeries

Descripció general

La classe GestioPrestatgeries s'encarregade gestionar el conjunt de prestatgeries d'un supermercat. Proporciona funcionalitats per afegir, modificar, consultar i eliminar prestatgeries, assegurant que les seves característiques, com la mida, la distribució de productes i la similitud, es mantinguin consistents. També permet obtenir informació detallada de les prestatgeries, com el seu identificador, distribució i índex de similitud.

Mètodes Principals

- getAllIdPrestatgeria(): Set<String>
    - Retorna tots els identificadors de les prestatgeries.
- getPrestatgeria(String id): Vector<String>
    - Retorna la prestatgeria associada a un id donat.

- getSimilitudPrestatgeria(String id): int
    - Retorna l'índex de similitud de la prestatgeria amb un id indicat.
- getSimilitud(String id): int
    - Retorna l'índex de similitud de la prestatgeria amb l'id especificat.
- getColumnes(String id): int
    - Retorna el nombre de columnes de la prestatgeria amb l'id especificat.
- getDistribucio(String id): Vector<String>
    - Retorna la distribució de productes (els seus noms) d'una prestatgeria especificada per id.
- getDistribucio(String id): Vector<Producto>
    - Retorna la distribució de productes (els seus noms) d'una prestatgeria especificada per id.
- eliminarPrestatgeria(String id)
    - Elimina la prestatgeria amb un id donat del mapa.
- afegirPrestatgeria(String id, Vector<Producto> productes, int files, int columnes, int similitut): boolean
    - Afegeix una nova prestatgeria al mapa si l'id no està en ús, retornant si ha tingut èxit.
- canviarIdPrestatgeria(String idAntic, String idNou)
    - Canvia l'id d'una prestatgeria existent a un nou id.
- modificarMidaPrestatgeria(String id, int files, int columnes)
    - Modifica el nombre de files i columnes d'una prestatgeria especificada per id.
- actualitzarSimilitud(String n1, String n2)
    - Actualitza l'índex de similitud de les prestatgeries que contenen els productes amb noms n1 i n2.

GestionProductos

Descripció general

La classe GestionProductos s'encarrega de gestionar el catàleg de productes del supermercat, com per exemple afegir, modificar, consultar i eliminar productes. Proporciona funcionalitats per afegir, eliminar i consultar productes, assegurant-se que les similituds entre els productes són coherents, simètriques i adequades.

Mètodes Principals

- getNumProd(): int
    - Retorna el nombre total de productes.
- addProducto(Producto p)
    - Afegeix un nou producte al catàleg, actualitzant les similituds amb els altres productes.
- verificarSimilitudesValidas(Producto p) (privat)
    - Comprova que les similituds del producte fan referència només a productes existents.
- verificarSimilitudesConProductosExistentes(Producto p) (privat)
    - Comprova si el producte té similitud amb el producte p.
- actualizarSimilitudesDeProductosExistentes(Producto nuevoProducto) (privat)
    - Actualitza la similitud de tots els productes existents, afegint-hi la similitud amb el nou producte.
- deleteProducto(Producto p)
    - Elimina un producte al catàleg actualitzant les similituds amb els altres productes.
- eliminarSimilitudesConProducto(Producto p) (privat)
    - Elimina les similituds dels altres productes amb el producte p.
- getProductos(): ArrayList<Producto>
    - Retorna la llista de tots els productes disponibles.
- getProducto(nombre): Producto
    - Retorna el producte amb el nom indicat.
- setProducto(nuevosProductos)
    - Actualitza la llista de productes després de verificar que tots els atributs són correctes.
- verificarNoSimilitudConsigoMismo(Producto p) (privat)
    - Comprova si el producte té una similitud amb si mateix.
- verificarNoSeRepitenProductos(ArrayList<Producto> nuevosProductos) (privat)
    - Comprova si el producte té similituds repetides.
- verificarSimilitudesEntreProductos(ArrayList<Producto> productos) (privat)
    - Assegura que totes les similituds entre els productes són simètriques.
- productoYaExiste(Producto p) : boolean (privat)
    - Retorna true si un producte amb el mateix nom ja existeix.
- modificarSimilitudEntreProductos(String nombreProducto1, String nombreProducto2, Integer valorSimilitud)
    - Modifica la similitud entre dos productes existents.

Prestatgeria

Descripció general

La classe Prestatgeria representa un conjunt de productes organitzats en una matriu d'unes determinades dimensions, i també pot calcular un índex de similitud entre els productes disposats.

Mètodes Principals

- modificarMida(Integer files, Integer columnes)
    - Modifica les dimensions de la prestatgeria (files i columnes).
- intercanvia(int pos1, int pos2): Integer
    - Intercanvia els productes de dues posicions dins de la distribució i calcula, actualitza i retorna l'índex de similitud.
- conteProdt(String nom): boolean
    - Retorna true si la prestatgeria conté el producte amb el nom passat com a paràmetre, i false altrament.
- calculaIndexSimilitut(): Integer
    - Calcula, actualitza i retorna l'índex de similitud de la prestatgeria basant-se en la distribució actual de productes.

Producto

Descripció general

La classe Producto representa un element amb un nom i un conjunt de similituds amb altres productes. Aquestes similituds són les que s'utilitzen per a optimitzar la disposició de la prestatgeria.

Mètodes Principals

- getSimilitud(String p): Integer
    - Retorna la similitud entre aquest producte i un altre producte especificat pel seu nom, o 0 si no existeix.
- getSimilitud(Producto p): Integer
    - Retorna la similitud entre aquest producte i un altre Producto, o 0 si no existeix.
- addSimilitudes(String p, Integer sim)

    - Afegeix una nova similitud entre aquest producte i un altre producte (indicat pel seu nom) amb el valor de similitud especificat.
- modificarSimilitud(String p, Integer sim)
    - Modifica la similitud d'aquest producte amb un altre producte (indicat pel seu nom) amb el valor de similitud especificat.
- tieneSimilitudCon(Producto otroProducto): boolean
    - Retorna true si aquest producte té una similitud amb un altre producte, false altrament.
- eliminarSimilitudCon(Producto p)
    - Elimina la similitud entre aquest producte i un altre producte especificat.

PERSISTÈNCIA

CtrlPersistencia

La classe CtrlPersistencia s'encarrega de la gestió de la capa de persistencia i d'utilitzar les operacions de les altres classes per gestionar les dades del programa. Aquesta classe proporciona una interfície tant per a guardar les dades dels productes y les prestatgeries generades per el programa i per carregar dades d'aquests al programa.

Els seus atributs són persistenciaPrestatgeries per a la gestió de les dades de GestioPrestatgeries i persitenciaProductos per a la gestió de les dades de GestionProductos.

PersistenciaGestionProductos

La classe PersistenciaGestionProductos s'encarrega de la gestió de les dades dels productes del programa i permet la importacio de productes desde un archiu json.

Els seu atribut es path que es un string al archiu json on es guarden tots els productes del programa.

PersistenciaGestionPrestatgeries

La classe PersistenciaGestionPrestatgeries s'encarrega de la gestió de les dades de les prestatgeries del programa i permet la importacio de prestatgeries desde un archiu json.

Els seu atribut es path que es un string al archiu json on es guarden totes les prestatgeries del programa.

DESCRIPCIÓ D'ESTRUCTURES I ALGORISMES

GestionProductos

Per a implementar la classe GestionProductos inicialment vam considerar utilitzar un HashSet per guardar el conjunt de productes. Això ens permetia assegurar-nos que no hi hagués productes duplicats, però teniem el problema de que els elements d'un HashSet no estan ordenats i ens complicava la realització dels jocs de prova.

Finalment, vam decidir utilitzar un ArrayList. Aquesta estructura ens permet mantenir l'ordre dels productes, cosa que ens facilitava la realització de proves. Per evitar duplicats hem implementat funcions que comproven que no es puguin afegir productes que ja estan a la llista, assegurant així la consistència de les dades sense perdre la simplicitat i flexibilitat que ofereix un ArrayList.

Algorisme Força Bruta

Aquest algorisme ens serveix per a trobar la distribució òptima dels productes a una prestatgeria segons el grau de similitud amb els productes adjacents. Per a trobar la millor solució al problema, es calcula el total de les similituds de totes les distribucions possibles i es retorna la que tingui aquest grau de similitud més elevat.

El primer que fa aquest algorisme és, de manera recursiva, anar creant distribucions possibles i sumant les similituds entre productes adjacents, per tal de calcular la similitud total de la distribució, i en cas de trobar una distribució millor a les trobades anteriorment, establir aquesta com a disposició òptima.

Les principals estructures de dades usades són:

- String[][] disposicio_optima

  Aquesta matriu emmagatzema la configuració òptima trobada durant l'execució de l'algorisme. Cada element de la matriu representa un producte en una posició concreta de la prestatgeria.

- String[][] actual

  És una matriu temporal que s'utilitza per emmagatzemar la distribució de la prestatgeria en el procés de cerca. A mesura que es van generant noves

distribucions, aquesta estructura es va actualitzant per calcular les similituds acumulades.

- Boolean[] visitado

  Aquesta estructura es fa servir per a portar un registre de quins productes ja han estat col·locats a la matriu durant la generació de les distribucions, evitant la repetició de productes.

L'algorisme de Força Bruta ens assegura l'obtenció de la distribució òptima dels productes, però el seu cost és elevat, ja que, com que es generen totes les distribucions possibles dels productes, el cost d'aquest algorisme és exponencial, és a dir, el cost és O(n!), on n representa el nombre de productes.

Algorisme TSP: 2-Aproximació

L'algorisme de 2-aproximació ens serveix per a trobar la distribució sub-òptima dels productes a una prestatgeria, aplicant un algorisme d'aproximació basat en el TSP (Traveling Salesman Problem). Aquest algorisme ens retorna una solució sub-òptima perquè la solució trobada és, com a molt, dues vegades pitjor respecte a la solució òptima.

L'algorisme comença construint un graf ponderat on els nodes representen els productes i les arestes la similitud entre ells. Després s'utilitza l'algorisme de Kruskal per a construir un arbre d'expansió màxima. Seguidament, es dupliquen les arestes de l'arbre per a garantir que tots els nodes tinguin grau parell, ja que és un requisit per als cicles eulerians. Mitjançant un DFS recursiu, es genera el cicle eulerià, i un cop generat, s'eliminen les repeticions del cicle per a crear un cicle hamiltonià. Finalment, s'usa l'optimització Hill Climbing per a intentar maximitzar el cost del cicle.

Les principals estructures de dades usades són:

- Arista

  Classe interna que guarda els dos extrems d'una aresta del graf i el pes d'aquesta aresta.

- List<Arista> grafo

  Representa el graf complet dels productes, amb cada aresta contenint els vèrtexs d'inici, final i el pes associat a la similitud.

- Map<Integer, Producto> productosmapped

  Mapa que indexa els productes per a assignar els identificadors als vèrtexs del graf.

- Map<Integer, List<Integer>> grafoDuplicado

  Mapa que emmagatzema el graf resultant del MST, amb les arestes duplicades per a la construcció del cicle eulerià.

- List<Integer> cicloEuleriano

  Seqüència de vèrtexs que formen el cicle eulerià, un cop eliminats els vèrtexs repetits.

- List<Integer> cicloSinRepetidos

  Cicle hamiltonià derivat del cicle eulerià, amb els vèrtexs repetits eliminats.

- List<Integer> resultadoOPT

  Cicle millorat després de l'aparició de l'estratègia de Hill Climbing.

Podem dividir la complexitat de l'algorisme per parts. Primer, el cost de la construcció del graf té un cost de $O(n^2)$. L'algorisme de Kruskal per a construir el MST té un cost de O(ElogE). El DFS té un cost lineal de O(V+A) i, finalment, el Hill Climbing té una complexitat, en el pitjor cas, de O(n^4).

Aquest algorisme ofereix una solució amb millor rendiment que la força bruta, però la qualitat de la solució no és tan bona com la de l'altre algorisme.

Algorisme Kruskal

L'algorisme de Kriskal s'utilitza per a construir l'arbre d'expansió màxim (MST). El que fa aquest algorisme és recórrer la llista d'arestes i, per a cada aresta, es comprova si connecta dos no estan connectats encara. Si es poden connectar els dos nodes prèviament disjunts, l'aresta es guarda a la llista de l'arbre d'expansió màxim. El cost d'aquest algorisme és O(ElogE), on E representa el nombre d'arestes del graf.

Algorisme DFS

L'algorisme de Depth-First Search és un algorisme recursiu que, a partir de l'arbre d'expansió màxim amb les arestes duplicades, recorre el graf i construeix el cicle eulerià seguint totes les arestes. Com ja s'ha mencionat anteriorment, el cost d'aquest algorisme és O(V+A), on V representa el nombre de vèrtexs i A el nombre d'arestes.

Optimització Hill Climbing

El Hill Climbing és un algorisme d'optimització aplicat per a millorar la solució inicial del cicle hamiltonià obtingut. Partint del cicle hamiltonià obtingut, s'intercanvien cada parell d'índexs i i j on i < j. Si el cost del nou cicle és més alt que l'actual, el nou cicle trobat es converteix en el cicle òptim, i el procés continua fins que ja no es puguin trobar millores. Com ja s'ha mencionat anteriorment, el cost d'aquest algorisme és, en el pitjor cas, de O(n^4).

TAULA DE RESULTATS DELS ALGORISMES

Utilitzem els datasets proporcionats per obtenir els resultats sobre quin algorisme és millor en cada cas.

Datasets	**Aproximació (cost/temps)**	**Força bruta (cost/temps)**
4 productes	95/8.68ms	95/0.75ms
5 productes	29/8.41ms	29/1.51ms
9 productes	548/5.09ms	548/96.97ms
15 productes	910/9.57ms	??/massa temps executant-se(+10min)
48 productes	222334/244.28ms	??/massa temps executant-se(+10min)

En conclusió, veiem que per pocs productes l'algorisme de força bruta és bastant millor que l'algorisme d'aproximació, però a mesura que va augmentant la quantitat de productes (partir dels 9 productes a l'exemple) veiem com l'algorisme d'aproximació comença a ser molt millor que la força bruta i podem veure com amb 15 productes i 48 productes, l'algorisme no arriba a 1s mentres que la força bruta amb 15 productes arriba a sobrepassar els 10 minuts i amb 48 productes segueix passant el mateix. No tenim resultats d'aquests datasets degut a la lentitud de l'algorisme de força bruta.