Integrantes:

- Juan Felipe Castillo
- Daniel Guzmán
- Mayumi Tamura

Método de la Ingeniería

Contexto problemático

Los misterios del cosmos y su inconmensurable tamaño han captado por siglos la atención de la humanidad. Como forma de difundir el conocimiento astronómico, la empresa NASITA ha solicitado la elaboración de un programa que permita visualizar las estrellas más cercanas al sistema solar, así como información sobre las estrellas.

Milky Way Explorer es el nombre del programa cuya finalidad será la de permitir al usuario explorar el vecindario estelar a través de las estrellas cercanas al sol, dando información sobre cada estrella que el usuario desee. Ejemplos de las funcionalidades del programa serán la de mostrar información como tamaño de la estrella, edad, color, planetas orbitantes (si existen) y por supuesto, su nombre.

Otra funcionalidad del programa será la de mostrar las distancias entre estrellas, el camino más corto y la distancia más corta del Sol a cualquier estrella, como también dar la posibilidad al usuario de agregar y quitar estrellas del vecindario estelar. Por último, se desea conocer una lista de las estrellas más cercanas al sol en orden de menor distancia a mayor.

Paso 1. Identificación del Problema

Identificación de necesidades

- Visualizar el vecindario de estrellas, su información y sus posibles conexiones.
- Poder agregar o eliminar estrellas que sean descubiertas o que hayan muerto en el vecindario.
- Conocer las distancias y caminos más cortos entre estrellas.

Definición del Problema

La compañía NASITA requiere de la implementación del software Milky Way Explorer para el manejo de información de las estrellas más cercanas al sistema solar, que cumpla con las solicitudes de visualización de estas y sus conexiones para permitir demás funcionalidades.

Requerimientos Funcionales:

Nombre	**RF1. Obtener información sobre estrellas.**
Descripción	Permite conocer las características de una estrella tal como edad, color, tamaño, nombre y el número de planetas orbitantes.
Entrada	Nombre de la estrella.
Salida	Características del planeta.

Nombre	**RF2. Obtener distancias entre estrellas.**
Descripción	Permite mostrar la distancia entre estrellas en años luz.
Entrada	Par de estrellas solicitadas.
Salida	Distancia entre las estrellas.

Nombre	**RF3. Obtener el camino más corto entre el Sol y una estrella.**
Descripción	Permite mostrar el camino más corto del sol a una estrella.
Entrada	Nombre de la estrella.
Salida	Camino más corto hasta la estrella.

Nombre	**RF4. Obtener la distancia más corta entre el Sol y una estrella.**
Descripción	Permite mostrar la distancia más corta entre el sol y una estrella.
Entrada	Nombre de la estrella.
Salida	Distancia más corta hasta la estrella.

Nombre	**RF5. Agregar estrella al vecindario.**
Descripción	Permite agregar una estrella al vecindario estelar.
Entrada	Nombre de la estrella.
Salida	

Nombre	**RF6. Eliminar estrella del vecindario.**
Descripción	Permite eliminar una estrella al vecindario estelar.
Entrada	Nombre de la estrella.
Salida	

Nombre	**RF7. Obtener la lista de estrellas de menor distancia.**
Descripción	Permite conocer las estrellas más cercanas al sol ordenadas en una lista ordenada de menor a mayor.
Entrada	
Salida	Lista de estrellas.

Paso 2. Recopilación de Información

Fuentes:

https://www.istr.unican.es/asignaturas/eda/cap4-grafos-2en1.pdf

https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra

https://es.wikipedia.org/wiki/Algoritmo_de_Floyd-Warshall

https://es.wikipedia.org/wiki/Algoritmo_de_Prim

https://es.wikipedia.org/wiki/Algoritmo_de_Kruskal

https://es.wikipedia.org/wiki/B%C3%BAsqueda_en_anchura

https://es.wikipedia.org/wiki/B%C3%BAsqueda_en_profundidad

https://es.wikipedia.org/wiki/V%C3%ADa_L%C3%A1ctea

https://es.wikipedia.org/wiki/Estrella

Definiciones

Grafo

Los grafos, constituyen estructuras de datos en las que se pueden expresar relaciones de conexión entre diversos elementos denominados vértices. Cada conexión se representa por un dato llamado arista Los grafos tienen gran cantidad de aplicaciones; por ejemplo:

- Representación de circuitos electrónicos analógicos y digitales
- Representación de caminos o rutas de transporte entre localidades
- Representación de redes de computadores.

Uno de los problemas más importantes en los grafos es el de encontrar el camino de coste mínimo.

Dijkstra

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista. Su nombre alude a Edsger Dijkstra, científico de la computación de los Países Bajos que lo describió por primera vez en 1959.

La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen hasta el resto de los vértices que componen el grafo, el algoritmo se detiene. Se trata de una especialización de la búsqueda de costo uniforme y, como tal, no funciona en grafos con aristas de coste negativo (al elegir siempre el nodo con distancia menor, pueden quedar excluidos de la búsqueda nodos que en próximas iteraciones bajarían el costo general del camino al pasar por una arista con costo negativo).

Floyd-Warshall

En informática, el algoritmo de Floyd-Warshall, descrito en 1959 por Bernard Roy, es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados. El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución. El algoritmo de Floyd-Warshall es un ejemplo de programación dinámica.

Muchos problemas de la vida cotidiana se pueden expresar e incluso resolver en forma de grafo. Existen algoritmos que encuentran distintos tipos de soluciones, tanto booleanas como de eficiencia. El grafo se representa en una tabla (matriz) que se conoce como "matriz de adyacencia" y representa si existe una unión entre dos nodos (boolean).

Prim

El algoritmo de Prim es un algoritmo perteneciente a la teoría de los grafos para encontrar un árbol recubridor mínimo en un grafo conexo, no dirigido y cuyas aristas están etiquetadas.

En otras palabras, el algoritmo encuentra un subconjunto de aristas que forman un árbol con todos los vértices, donde el peso total de todas las aristas en el árbol es el mínimo posible. Si el grafo no es conexo, entonces el algoritmo encontrará el árbol recubridor mínimo para uno de los componentes conexos que forman dicho grafo no conexo.

El algoritmo fue diseñado en 1930 por el matemático Vojtech Jarnik y luego de manera independiente por el científico computacional Robert C. Prim en 1957 y redescubierto por Dijkstra en 1959. Por esta razón, el algoritmo es también conocido como algoritmo DJP o algoritmo de Jarnik.

Kruskal

El algoritmo de Kruskal es un algoritmo de la teoría de grafos para encontrar un árbol recubridor mínimo en un grafo conexo y ponderado. Es decir, busca un subconjunto de aristas que, formando un árbol, incluyen todos los vértices y donde el valor de la suma de todas las aristas del árbol es el mínimo. Si el grafo no es conexo, entonces busca un bosque expandido mínimo (un árbol expandido mínimo para cada componente conexa). Este algoritmo toma su nombre de Joseph Kruskal, quien lo publicó por primera vez en 1956. Otros algoritmos que sirven para hallar el árbol de expansión mínima o árbol recubridor mínimo es el algoritmo de Prim, el algoritmo del borrador inverso y el algoritmo de Boruvka.

BFS

En Ciencias de la Computación, Búsqueda en anchura (en inglés BFS - Breadth First Search) es un algoritmo de búsqueda no informada utilizado para recorrer o buscar elementos en un grafo (usado frecuentemente sobre árboles). Intuitivamente, se comienza en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo) y se exploran todos los vecinos de este nodo. A continuación para cada uno de los vecinos se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el árbol.

Formalmente, BFS es un algoritmo de búsqueda sin información, que expande y examina todos los nodos de un árbol sistemáticamente para buscar una solución. El algoritmo no usa ninguna estrategia heurística.

Si las aristas tienen pesos negativos aplicaremos el algoritmo de Bellman-Ford en alguna de sus dos versiones.

DFS

Una Búsqueda en profundidad (en inglés DFS o Depth First Search) es un algoritmo de búsqueda no informada utilizado para recorrer todos los nodos de un grafo o árbol (teoría de grafos) de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa

(Backtracking), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

Milky Way

La galaxia de la Vía Láctea es una galaxia espiral donde se encuentra el sistema solar y a su vez se encuentra la Tierra. Según las observaciones, posee una masa de 1012 masas solares y es una espiral barrada. Su diámetro medio se estima en unos 100.000 años luz, equivalentes a casi un trillón y medio (1,42×1018) de kilómetros o 9480 millones de unidades astronómicas. Se calcula que contiene entre 200.000 y 400.000 millones de estrellas. La distancia desde el Sol hasta el centro de la galaxia es de alrededor de 25 766 años luz (7900 pc), es decir, el 52 % del radio total galáctico. La Vía Láctea forma parte de un conjunto de unas cuarenta galaxias llamado Grupo Local, y es la segunda más grande y brillante tras la galaxia de Andrómeda.

Estrella

Una estrella es una esfera luminosa de plasma que mantiene su forma gracias a su propia gravedad. La estrella más cercana a la Tierra es el Sol. Otras estrellas son visibles a simple vista desde la Tierra durante la noche, apareciendo como una diversidad de puntos luminosos fijos en el cielo debido a su inmensa distancia de la misma. Históricamente, las estrellas más prominentes fueron agrupadas en constelaciones y asterismos, y las estrellas más brillantes pasaron a denominarse con nombres propios. Los astrónomos han recopilado un extenso catálogo, proporcionándole a las estrellas designaciones estandarizadas. Sin embargo, la mayoría de las estrellas en el Universo, incluyendo todas las que están fuera de nuestra galaxia, la Vía Láctea, son invisibles a simple vista desde la Tierra. De hecho, la mayoría son invisibles desde la dicha Tierra incluso a través de los telescopios de gran potencia.

Paso 3. Búsqueda de Soluciones Creativas

Se realiza una lluvia de ideas donde se plantean soluciones sobre cómo podría funcionar la aplicación, la interfaz y que representaciones de grafos y demás estructuras de datos, se podrían usar para cumplir con las diferentes funcionalidades requeridas.

Clases:

- IGraph
- GraphAL
- GraphAM
- Node
- Edge

Ideas planteadas:

Representación de los Grafos:

Se plantea la implementación de dos grafos, usando dos representaciones diferentes.

Alternativa 1: Grafo Lista de Adyacencia:

- Esta alternativa consiste en representar el grafo de estrellas como una lista de adyacencia.

Alternativa 2: Grafo Matriz de Adyacencia:

- Esta alternativa consiste en representar el grafo de estrellas como una matriz de adyacencia.

Alternativa 3: Grafo Matriz de Incidencias:

- Esta alternativa consiste en representar el grafo de estrellas como una matriz de incidencias.

Alternativa 4: Clase Grafo y representaciones como herencias:

- Esta alternativa consiste en implementar el grafo como una clase padre genérica de la cual heredan sus principales funcionalidades las clases de grafos con dos representaciones diferentes.

Alternativa 5: Interfaz IGrafo y representaciones como clases separadas:

- Esta alternativa consiste en implementar las dos representaciones de grafos distintas como clases particulares que implementan una interfaz IGraph, con lsa principales funciones de un grafo.

Estructuras de datos para las funciones de los Grafos:

Se plantean las estructuras de datos a usar para las representaciones de los nodos de los grafos, así mismo como las estructuras necesarias para los principales métodos de los grafos.

Alternativa 1: LinkedList de arrays:

- Esta alternativa consiste en implementar los vértices del grafo como una LinkedList con arrays para la lista de nodos adyacentes a cada uno.

Alternativa 2: Hash Map:

- Esta alternativa consiste en implementar los vértices del grafo como un HashMAp.

Diseño de la interfaz:

Se plantea la implementación de dos grafos, usando dos representaciones diferentes.

Alternativa 1: Swing y awt:

- Esta alternativa consiste en diseñar la interfaz de la aplicación utilizando swing y awt.

Alternativa 2: Java fx:

- Esta alternativa consiste en diseñar la interfaz de la aplicación utilizando Java fx.

Paso 4. Transición de las Ideas a los Diseños Preliminares

(alternativas preseleccionadas)

Representación de los Grafos:

Alternativa 1: Grafo Lista de Adyacencia:

- Esta alternativa ofrecería un fácil acceso a los nodos del grafo, lo cual sería una ventaja para implementar los algoritmos principales de un grafo.

Alternativa 2: Grafo Matriz de Adyacencia:

- Esta alternativa ofrecería una buena representación de las conexiones entre los nodos.

Alternativa 4: Clase padre Grafo:

- Esta alternativa sería útil para implementar los diferentes grafos, sin embargo, no sería muy coherente para el contexto del problema una herencia.

Alternativa 5: Interfaz IGrafo:

- Esta alternativa sería más coherente para la implementación de las representaciones de grafos.

Estructuras de datos para las funciones de los Grafos:

Alternativa 1. Linked.

Esta alternativa ofrecería una sencilla implementación, sin embargo, su complejidad temporal de acceso a los elementos sería poco óptima.

Alternativa 2. Hash Map para los nodos.

Esta alternativa ofrecería una complejidad temporal más óptima que una lista, dado su rápido acceso a los elementos.

Diseño de la interfaz:

Alternativa 1. Swing y awt.

Esta alternativa sería óptima debido al buen manejo y amplio dominio de los desarrolladores.

Alternativa 2. Java fx.

Esta alternativa ofrecería gran facilidad para representar la interfaz de la aplicación, sin embargo, su variedad de características no son totalmente dominadas por el equipo.

Paso 5. Evaluación y Selección de la Mejor Solución

Criterios

Criterio A. Dominio de la solución por los desarrolladores.

- [1] Las alternativas de estructuras de datos o implementaciones planteadas deben ser bien conocidas por todo el equipo de desarrolladores.

Criterio B. Complejidad óptima del algoritmo en el contexto o representación de grafo.

- [2] Las alternativas planteadas deben tener una óptima complejidad según sean las características de las representaciones del grafo.

Criterio C. Trazabilidad con la solución del problema.

- [2] La implementación de la alternativa planteada debe ser coherente con la solución del problema, es decir, con la representación del grafo más óptima.

Evaluación

	Criterio A	**Criterio B**	**Criterio C**	**Total**
Grafo lista de Adyacencia	1	2	2	5
Grafo matriz de Adyacencia	1	2	2	5
Clase padre Graph	1	1	0	2
Interfaz IGraph	1	2	2	5
Diseño de la Interfaz con swing y awt	1	2	2	5

Diseño de la Interfaz con Java fx	0	2	2	4
Dijkstra	1	2	2	5
Floyd Warshall	1	2	2	5
BFS	1	0	0	1
DFS	1	0	0	1
Prim	1	2	2	5
Kruskal	0	1	1	2
Nodos en LinkedList	1	0	2	3
Nodos en Hash Map	0	2	2	4

Selección

Se escogen las alternativas:

- Implementación de un grafo por Lista de Adyacencia.
- Implementación de un grafo por Matriz de Adyacencia.
- Implementación de una interfaz IGraph que implementarán las dos representaciones de grafo.
- Diseño de interfaz en swing y awt.
- Implementación de Dijkstra, Prim y Floyd Warshall.
- Implementación de los nodos del grafo como Hash Map.

La elección de implementar estas estructuras de datos para la solución del problema se da como una elección en grupo, teniendo en cuenta los criterios planteados y la evaluación de las alternativas dadas.

Paso 6. Preparación de Informes y Especificaciones

Especificación del Problema (en términos de entrada/salida)

Definición de TADS

Documentación: TADs, diseño de pruebas, diagramas de clases y de pruebas, se encuentran en la carpeta Docs del repositorio.

Paso 7. Implementación del Diseño

Implementación en Java.

https://github.com/TakumiMay/GalaxyGraph