Universidad Cenfotec

Autor(es):

-Isabel Galeano Hernández, Cédula: 1-1888-0968

-Daniel Zúñiga Rojas, Cédula: 1-1811-0097

-Isaías Arce Rodríguez, Cédula: 3-0525.0698,

José Gracía Quirós, Cédula: 1-1782-0076

Tema:
Proyecto de Grafos

Curso:
BISOFT-20 Estructuras de datos 2

Profesor:
Christian Sibaja Fernández

Fecha de entrega:
06/12/2021

Cuatrimestre:
2021-3

Índice

Descripción del Problema

El presente proyecto de ingeniería muestra una posible solución orientada a la determinación de distancias entre posiciones geográficas. Todo esto, utilizando estructuras de datos, tales como: listas simples para el almacenamiento de las playas y su ubicación geográfica, grafos que poseen vértices y sus respectivos arcos, los cuales están representados por las mismas playas, la tabla de hash, para la búsqueda de las playas y por último el algoritmo de Dijkstra para buscar los caminos más cortos de una playa a otra.

El programa permite ubicar una playa determinada o bien calcular la ruta más corta entre dos playas, todo esto mediante una interfaz gráfica.

Especificación de estructuras y algoritmos implementados

I. Grafos

Un grafo está formado por un conjunto de nodos(o vértices) y un conjunto de arcos. Cada arco en un grafo se especifica por un par de nodos.[1] Los grafos presentan términos de definición para su composición, entre ellos se encuentran: arista, vértices adyacentes, factor de peso, longitud y ciclos. Las aristas representan un arco de un grafo no dirigido. Los vértices adyacentes son nodos que se unen por un arco.

El factor de peso es un valor para asociar a un arco y sus conexiones. La longitud es el número de arcos o enlaces en el recorrido entre dos nodos. Un ciclo es un grafo con igual número de vértices y aristas, cuyos vértices se pueden ordenar de tal forma que se forma un círculo con dos vértices que son adyacentes si y sólo si son consecutivos en el círculo.

Para entender estos términos también se necesita de los grafos y sus conexiones los cuales son los grafos no dirigidos (conexo enlazado) y los grafos dirigidos que pueden presentar conexos fuertes y conexos débiles. Los conexos enlazados son los que presentan un camino entre cualquier par de nodos o vértices. Los conexos fuertes presentan un camino entre cualquier nodo y los conexos débiles presentan una cadena o sucesión entre cualquier par de nodos.[2]

Los grafos pueden mantenerse en una computadora de dos formas: mediante una matriz de adyacencia y por listas enlazadas conjuntas. La matriz de adyacencia es básicamente una matriz de nxn, donde n es la cantidad de vértices o nodos presentes en el grafo. Consta de dos únicos valores para rellenarse: 0 y 1. Para la matriz, 0 representa que no hay enlaces y 1 representa que hay enlace. Para determinar si hay enlace, la matriz se conforma de columnas con cada uno de los vértices y de filas con cada uno de los nodos de igual forma.Si hay enlace entre los nodos de la fila y columna, se coloca un 1 y en caso de que no hubiera enlace, se coloca un 0.

Las listas enlazadas o de adyacencia representan por filas y cuadros las relaciones de nodos. Se identifican todos los nodos presentes y se colocan horizontalmente los nodos enlazados.
Por lo tanto, se realizó un grafo de las playas en el país de una lista. De esta forma se pueden saber las relaciones geográficas y distancias de relación y cercanía.

II. Tabla de Hash

Una tabla hash o mapa hash es una estructura de datos que asocia llaves o claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos (teléfono y dirección, por ejemplo) almacenados a partir de una clave generada usando el nombre, número de cuenta o id. Funciona transformando la clave con una función hash en un hash, un número que la tabla hash utiliza para localizar el valor deseado.[3]

La tabla hash tiene operaciones básicas de inserción y búsqueda. La búsqueda permite la devolución de un valor. Dependiendo del uso de las tablas, se puede implementar una función de eliminación de llave en la tabla. Esta se combina con una búsqueda para poder lograr el borrado del elemento deseado. Algunas tablas hash permiten almacenar múltiples valores bajo la misma clave.

Los valores ingresados a las tablas se convierten en claves mediante una función hash que convierte el valor de input en un valor clave de cierta cantidad de dígitos según el algoritmo utilizado. Los algoritmos se pueden basar en división, centro de cuadrados, plegamiento, truncamiento, etc. La eficiencia de los algoritmos depende de los tipos de datos que se manejen y el tamaño. La inserción de los datos en las tablas pueden presentar colisiones que son asignaciones de una misma dirección para varias claves. Esto se puede evitar utilizando una lista para todas las direcciones con colisión y los datos colisionados o colocando los elementos colisionados en otra posición del arreglo de la tabla.

III. Listas

La lista es una colección ordenada de elementos en la que se pueden insertar y eliminar elementos en el lugar que se requiera. Aunque normalmente se crean listas ordenadas.[4] Las listas son estructuras que permiten ingresar datos linealmente, desplazándose de izquierda a derecha en la estructura. En la lista, todos los elementos menos el primero tienen un predecesor directo y todos los elementos menos el último tienen un sucesor.

Con las listas se pueden insertar, buscar y eliminar datos. Las inserciones son lineales y las búsquedas igual. La eliminación de un dato puede presentar dos casos: cuando se necesita borrar el primer elemento y cuando se busca borrar uno en específico. El primer caso solo requiere de mover el nodo inicial al siguiente y borrar su auxiliar. El segundo caso requiere iterar en la lista hasta encontrar el nodo buscado, guardar el anterior, apuntar al siguiente auxiliar y borrar el anterior previamente adelantado.

Las listas enlazadas se utilizan para las tablas hash permitiendo evitar colisiones, por lo que se usa primordialmente el segundo método de eliminación.

IV. Algoritmo de Dijkstra

El algoritmo de Dijkstra calcula las distancias mínimas desde un nodo específico a todos los demás. Para hacerlo, en cada paso se toma el nodo más cercano al inicial que aún no fue visitado. Este nodo tiene calculada la menor distancia al nodo inicial. Luego, se recalculan todos los caminos mínimos, teniendo en cuenta al nodo buscado como camino intermedio. Así, en cada paso se tendrá un subconjunto de nodos que ya tienen calculada su mínima distancia y los demás tienen calculada su mínima distancia si sólo se pueden usar los nodos del conjunto como nodos intermedios.[5]

Con cada iteración se agrega un nodo más al conjunto, hasta resolver el problema de distancia en su totalidad. Por lo tanto, este algoritmo permite relacionar distancias entre los nodos de un grafo para saber la más corta. En este caso se utiliza para conocer la distancia más cercana entre dos playas geográficamente a partir del grafo general de playas utilizado. La implementación del algoritmo puede variar en un una cola de prioridad si este presenta muchas aristas o no. La cola de prioridad se utiliza para reducir la dificultad de búsqueda en caso de muchas aristas en el grafo a analizar.

Detalles de la implementación

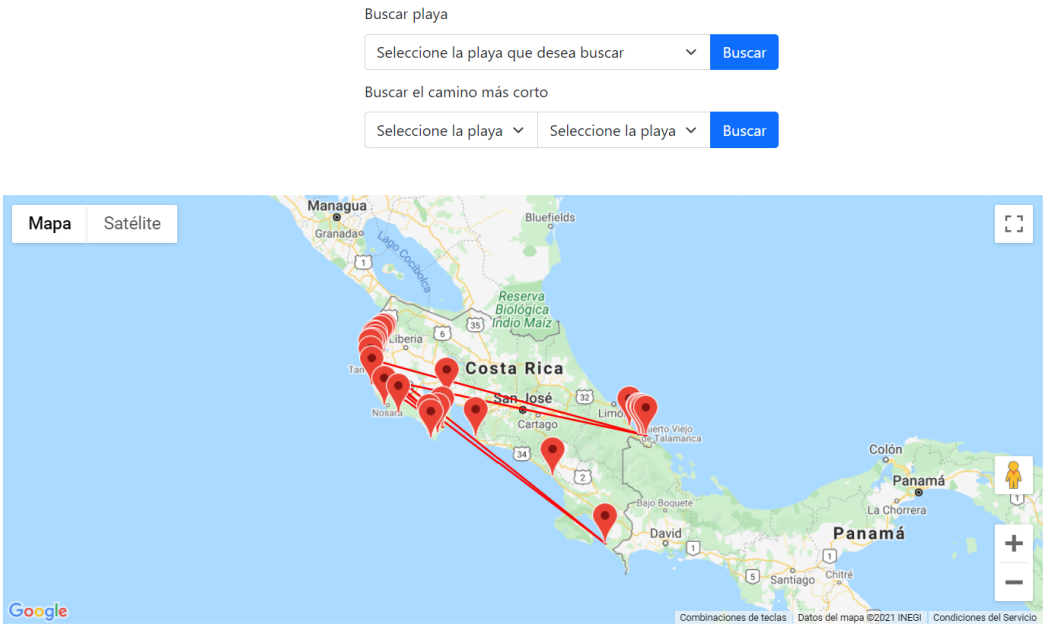
I. **Elementos implementados**: Se hizo uso de algoritmos y estructuras de datos para realizar el proyecto. Lo cual describiremos a continuación; En la parte del backend, se trabajó con el lenguaje de programación Java y el framework Spring Boot para realizar la conexión del backend con el frontend. En el backend, se desarrollaron todos los elementos del proyecto, creación de objetos playas con sus debidas ubicaciones, creación de listas adyacentes, para crear grafos con sus vértices y arcos, asimismo, tabla hash para la búsqueda de playas, y por último el algoritmo de Dijkstra para buscar el camino más corto de una playa a otra.

De acuerdo a lo anterior, procederemos a explicar la implementación del frontend. En el lado del frontend, se trabajó con el lenguaje de programación Java. Se utilizó HTML y CSS. Para la visualización del mapa, con las ubicaciones de las playas, con marcadores y caminos se utilizó el API de Google maps.

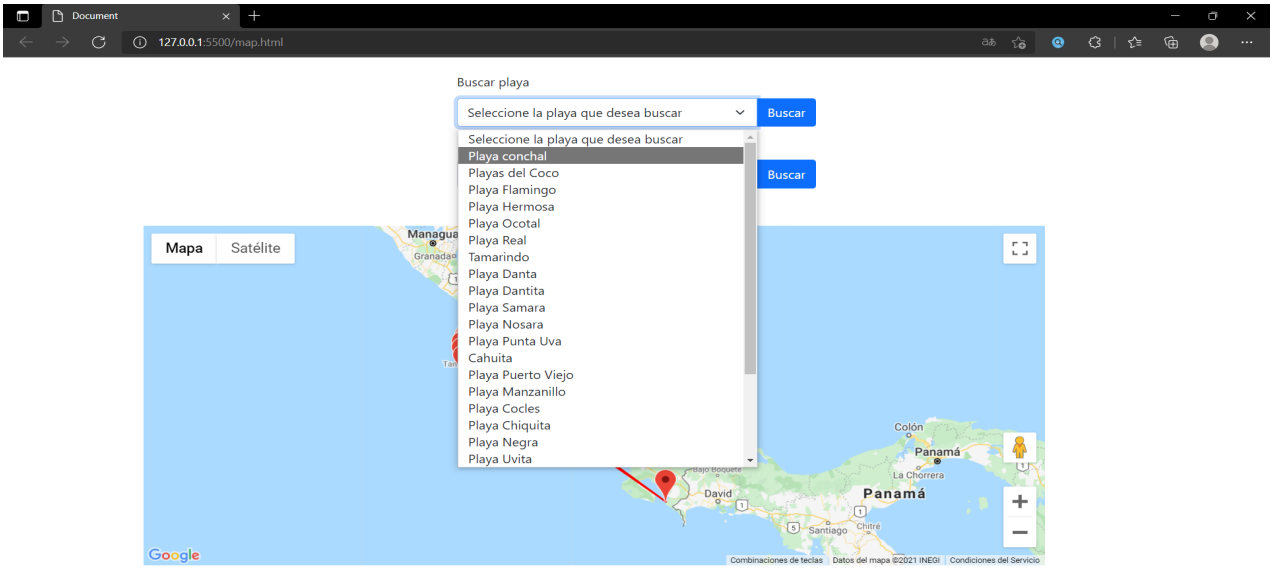
II. **Problemas presentados**: El mayor problema que hubo fue la conexión desde JS con Spring Boot, debido a que fallaba, y a la hora de hacer los request, saltaba el error de CORS Policy, esto denegaba la conexión. Al inicio, se estaba utilizando Node, específicamente axios para poder hacer los request al backend. Luego se cambió a utilizar JS puro y se utilizó fetch para hacer los request.

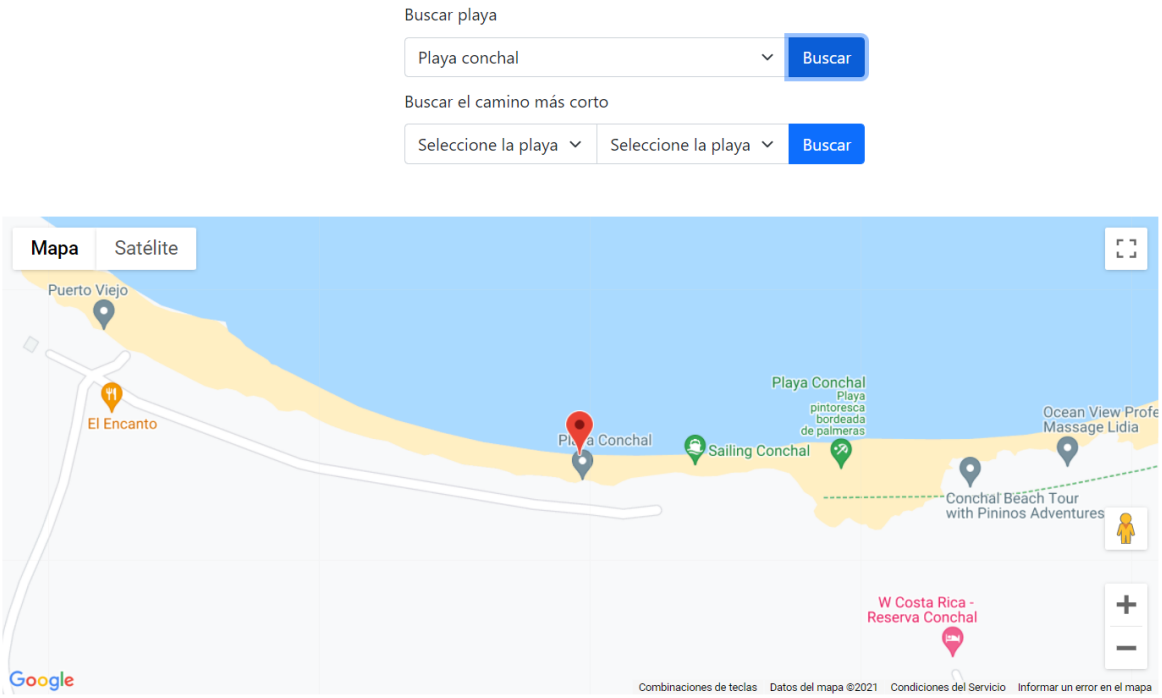
Instrucciones de la operación

1. Pantalla inicial, donde se pueden realizar todas las operaciones. Inicialmente, se pueden visualizar todos los vértices y sus arcos del grafo. En este caso, las ubicaciones de las playas y sus conexiones con otras playas.

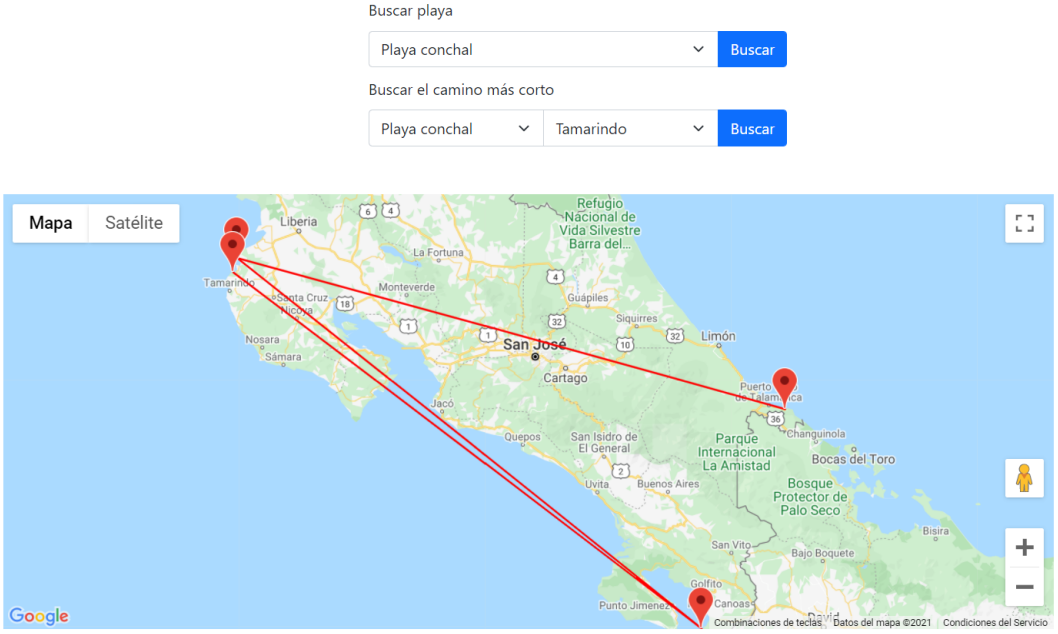


2. Buscar una playa: Para buscar una playa, se debe seleccionar en el dropdown la playa que se desea buscar y luego dar click en el botón que dice buscar.





3. Para buscar el camino más corto de una playa a otra, se deben seleccionar ambas playas en los dropdowns y posteriormente darle click al botón de buscar.



El estado actual de la aplicación es totalmente operativo.

Conclusiones

El proyecto se considera un ejercicio exitoso ya que implementa distintas estructuras y algoritmos vistos en clase para resolver un problema práctico. Queda en evidencia la utilidad del algoritmo de Dijkstra para la determinación de rutas apropiadas entre dos puntos.

También destaca la implementación de grafos mediante listas para la representación de los nodos y espacio para aplicar el algoritmo.

Referencias

1. Castillo, O. (2011). Grafos. Universidad Veracruzana. https://www.uv.mx/personal/ocastillo/files/2011/04/Teoria-de-grafos.pdf

2. Seymour, L. (2018). Grafos. Instituto Tecnológico de Nuevo Laredo. http://fcaenlinea1.unam.mx/anexos/1566/1566_u4_anexo3.pdf

3. Escuela de Computación de la Universidad de Don Bosco. (2020). Tablas Hash. Universidad Don Bosco. https://www.udb.edu.sv/udb_files/recursos_guias/informatica-ingenieria/programacion-con-estructuras-de-datos/2020/i/guia-8.pdf

4. Martínez, D. (2020). Listas. Universidad Tecnológica de la Mixteca. https://www.utm.mx/~dtorres/cursos/estructuradedatos/Tema4-Listas.pdf

5. Sclar, M. (2016). Camino mínimo en grafos. Universidad de Buenos Aires. http://www.oia.unsam.edu.ar/wp-content/uploads/2017/11/dijkstra-prim.pdf