UADE

Universidad Argentina de la Empresa
Facultad de Ingeniería y Ciencias Exactas

Carrera: Ingeniería Informática

Primer cuatrimestre - Año 2025

Asignatura: Programación II

Profesor: Nicolás Perez

Trabajo Práctico Obligatorio (TPO)

Fecha de entrega: 27/06

Alumnos:

Álvarez, Diego Anibal- Legajo: 1193062

Cembal, Micaela Yael - Legajo: 1189804

Gryner, Agustina Avril - Legajo: 1184295

Kugler, Gael Mateo - Legajo: 1197553

Progano Elmallian, Ariana - Legajo: 1186457

<u>Trabajo Práctico Obligatorio</u>: Mapa de Bicisendas

En muchas ciudades, especialmente en lugares como Buenos Aires, cada vez más personas eligen moverse en bici. Es más accesible económicamente y más rápido en horarios pico y además ayuda al medio ambiente. Sin embargo, existe un problema que se repite: no siempre es fácil encontrar un camino que sea seguro y rápido al mismo tiempo.

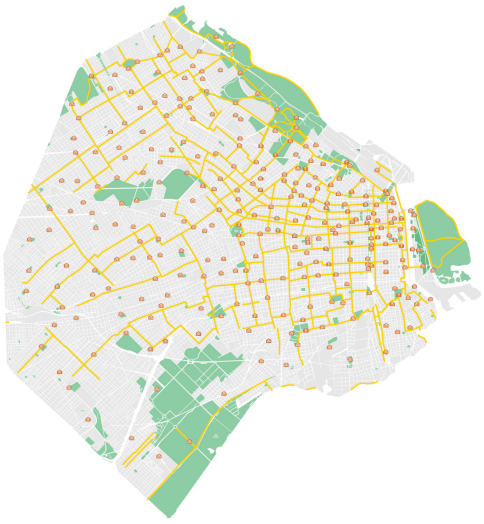
Aunque hay bicisendas y ciclovías por varias partes de la ciudad, muchas veces no están bien conectadas entre sí. Entonces, si alguien quiere ir desde un punto A hasta un punto B en bici, se encuentra con varias dudas:

- ¿Voy por el camino más corto, aunque no tenga bicisenda?
- ¿O me doy toda una vuelta más larga solo para ir por lugares más seguros?
- ¿Qué pasa si no conozco la zona? ¿Y si me meto sin querer en una avenida con mucho tránsito?

Las apps de mapas como Google Maps no siempre proveen la mejor opción para bicicletas. A veces te mandan por calles peligrosas o por avenidas llenas de autos. Además, no todos tienen la información actualizada de dónde hay bicisendas nuevas o de qué calles están en buen estado para pedalear.

Por eso, se vuelve clave contar con una herramienta que propone recorridos pensados para quienes se mueven en bici: que priorice las bicisendas, evite calles peligrosas y no obligue a dar vueltas innecesarias. Una solución que permita llegar de forma ágil y segura, sin tener que resignar comodidad ni tranquilidad.

Para este TPO decidimos representar a la Ciudad de Buenos Aires como un mapa de grafos: cada esquina es un punto (nodo) y cada tramo de bicisenda es una conexión (arista) con un peso según la distancia.



LU: 1193062 Álvarez, Diego Anibal
LU: 1189804 Cembal, Micaela Yael
LU: 1184295 Gryner, Agustina Avril
LU: 1197553 Kugler, Gael Mateo
LU: 1186457 Progano Elmallian, Ariana

Solución propuesta

Utilizamos el algoritmo de Dijkstra para calcular las distancias mínimas desde una esquina origen al resto del mapa y desarrollamos el código bajo el paradigma de Teoría del Dato Abstracto (TDA), lo que nos permitió tener una estructura clara, ordenada y fácil de modificar. Dijkstra fue la mejor opción porque trabaja con grafos ponderados y garantiza encontrar el camino más corto siempre que los pesos sean positivos, como en nuestro caso, donde representan distancias.

Paquetes utilizados:

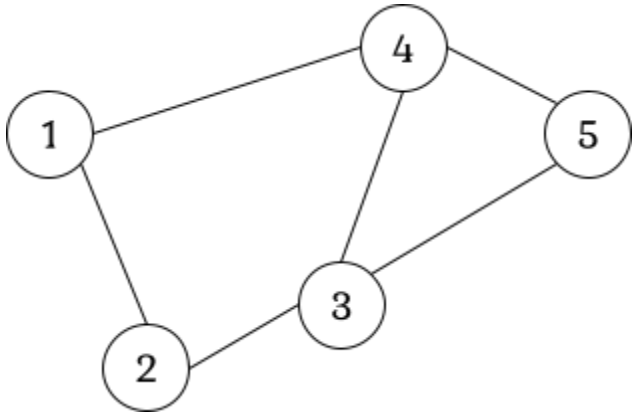
- Interfaces: IGrafo, INodo
- Modelo: Esquina, MapaDeBicisendas
- Servicios: Dijkstra
- Test: TestBicisendasCiudad

Algoritmo de Dijkstra

El algoritmo de Dijkstra permite encontrar el camino más corto desde un nodo origen hacia los demás nodos de un grafo, siempre que los pesos sean positivos. Funciona asignando una distancia inicial de 0 al nodo de origen y una distancia infinita al resto. A medida que avanza, va actualizando las distancias mínimas conocidas hacia cada nodo, eligiendo siempre el nodo no visitado con menor distancia acumulada.

El algoritmo va iterando y siempre elige la esquina que tiene la menor distancia acumulada (de las que todavía no visitó) y revisa si puede llegar más rápido a las esquinas vecinas desde ahí. Si encuentra una mejor opción, actualiza la distancia. Este proceso se repite hasta que ya no quedan esquinas por revisar o se llega al destino que se buscaba. En nuestro caso, esto permite que cualquier persona pueda saber cuál es el mejor camino para ir de una esquina a otra en bici, usando la mayor cantidad posible de bicisendas y recorriendo la menor distancia.

LU: 1193062 Álvarez, Diego Anibal
LU: 1189804 Cembal, Micaela Yael
LU: 1184295 Gryner, Agustina Avril
LU: 1197553 Kugler, Gael Mateo
LU: 1186457 Progano Elmallian, Ariana



El proceso no finaliza hasta haber visitado todos los nodos o hasta encontrar el destino. En nuestro TPO lo utilizamos porque es eficiente, fácil de implementar y se adapta muy bien a la lógica de bicisendas, donde los pesos (distancias) siempre son positivos.

No elegimos el algoritmo de Kruskal ni de Prim para la solución de este problema debido a que Kruskal, por un lado, encuentra un árbol de expansión mínima conectando todos los nodos del grafo con el menor peso total. Por otro lado, Prim también encuentra un árbol de expansión mínima, similar a Kruskal, pero comienza desde un nodo y expande la red paso a paso. Sin embargo, ninguno de los dos está diseñado para encontrar caminos entre dos puntos específicos lo que no permite priorizar bicisendas o características particulares de las rutas. Tampoco optamos por utilizar el algoritmo de A* que a pesar de su rapidez, se guía por una estrategia dirigida que prioriza caminos que "parecen" prometedores, y por eso puede no explorar todas las rutas con la misma profundidad. Dijkstra es más certero porque al calcular todas las variantes posibles no corre el riesgo de dejar fuera alguna ruta más eficiente que una heurística podría pasar por alto.

Diferencias con el algoritmo visto en clase

A la hora de comparar el código de Dijkstra creado por nosotros con el que vimos en clase, encontramos varias diferencias importantes. En el código proporcionado por el docente, los nodos se manejan usando enteros (int), lo que te obligaba a hacer un mapeo adicional si querías trabajar con esquinas reales. Es decir, había que tener una estructura que relacione cada número con una esquina específica del mapa, lo cual hacía el código más complejo, menos intuitivo y más propenso a errores.

En cambio, en nuestro código usamos directamente el objeto Esquina, que implementa la interfaz INodo. De esta forma, el grafo trabaja con entidades que representan exactamente lo que

necesitamos: esquinas con sus vecinos y los pesos de cada conexión (es decir, la distancia entre esquinas). Gracias a esto, Dijkstra no necesita "adivinar" qué representa cada número, porque directamente está trabajando con objetos del tipo Esquina.

Otra diferencia importante es que, en la bibliografía en la que nos basamos, particularmente el material de Melanie Sclar de la Universidad de Buenos Aires, el algoritmo de Dijkstra está pensado principalmente para grafos representados con matrices o listas de adyacencia usando enteros. Nosotros, en cambio, tuvimos que adaptar esa lógica a un enfoque con TDA y clases genéricas. Esto implicó trabajar con nodos que contienen directamente la información que necesitamos (como las esquinas), en vez de tener que traducir números. Además, implementamos nuestra propia cola de prioridad sin utilizar bibliotecas externas, lo que nos llevó a entender más a fondo cómo gestionar los nodos por visitar y optimizar la búsqueda del camino más corto.

```
while (!cola.isEmpty()) {
    Esquina actual = cola.poll();

    if (visitado.contains(actual)) continue;
    visitado.add(actual);
```

En esta parte del recorrido, mientras la cola tenga elementos, vamos sacando la esquina con menor distancia acumulada. Si esa esquina ya fue visitada antes, la salteamos. Si no, la marcamos como visitada y seguimos revisando sus vecinos. Así, el algoritmo va avanzando siempre por el camino que parece más corto, sin repetir nodos ni hacer cálculos innecesarios. Esto nos ayuda a encontrar el mejor recorrido de forma rápida y ordenada.

Bibliografía

- Joshi, V. (2017, 17 de octubre). "Encontrar el camino más corto, con un poco de ayuda de Dijkstra" Basecs – Medium. Este artículo nos ayudó a comprender de manera clara y detallada el funcionamiento del algoritmo de Dijkstra, lo cual fue fundamental para poder aplicarlo correctamente en la representación de nuestro mapa de bicisendas.

LU: 1193062 Álvarez, Diego Anibal
LU: 1189804 Cembal, Micaela Yael
LU: 1184295 Gryner, Agustina Avril
LU: 1197553 Kugler, Gael Mateo
LU: 1186457 Progano Elmallian, Ariana

- Cassingena Navone, E. (2022, 24 de octubre). "Algoritmo de la ruta más corta de Dijkstra - Introducción gráfica y detallada" - freeCodeCamp.org. Esta página nos ayudó a entender paso a paso cómo funciona el algoritmo de Dijkstra aplicado a grafos dirigidos, y gracias a esa explicación clara pudimos implementarlo correctamente en nuestro proyecto para modelar las bicisendas como rutas con dirección y peso.

- Sclar, M. (2016). "Camino mínimo en grafos". Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires. Para implementar el algoritmo de Dijkstra en nuestro proyecto, nos basamos en este material de la Universidad de Buenos Aires, donde se explicaba el funcionamiento del Algoritmo en pseudocódigo. A partir de esa explicación, lo adaptamos y desarrollamos nuestra propia versión en Java, bajo el paradigma de TDA.

Enlace al Github del proyecto

LU: 1193062 Álvarez, Diego Anibal
LU: 1189804 Cembal, Micaela Yael
LU: 1184295 Gryner, Agustina Avril
LU: 1197553 Kugler, Gael Mateo
LU: 1186457 Progano Elmallian, Ariana