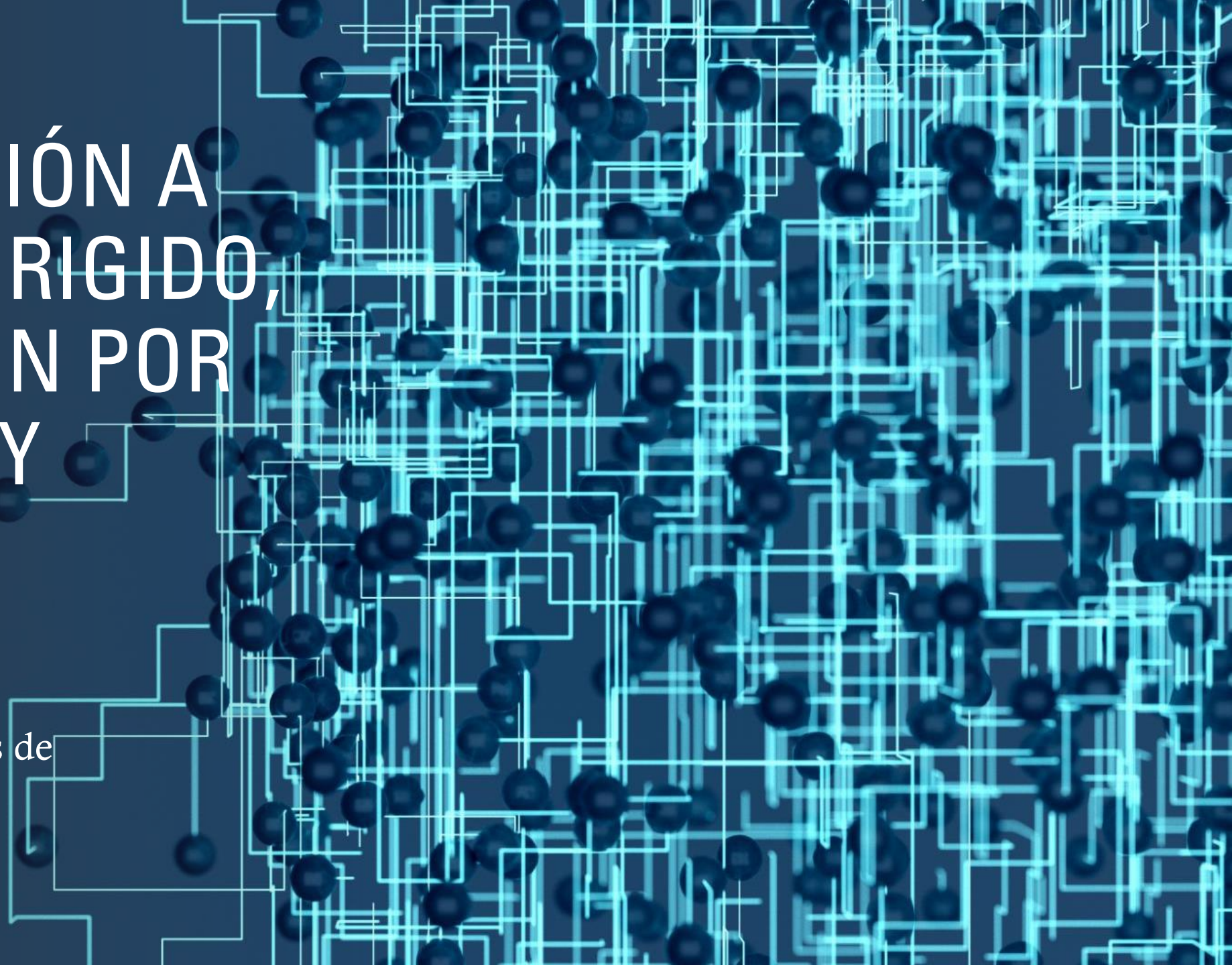
INTRODUCCIÓN A GRAFONODIRIGIDO, ORDENACIÓN POR INSERCIÓN Y VIAJANTE

Conceptos fundamentales de estructuras de datos y algoritmos



PRESENTACIÓN GENERAL

Clase GrafoNoDirigido

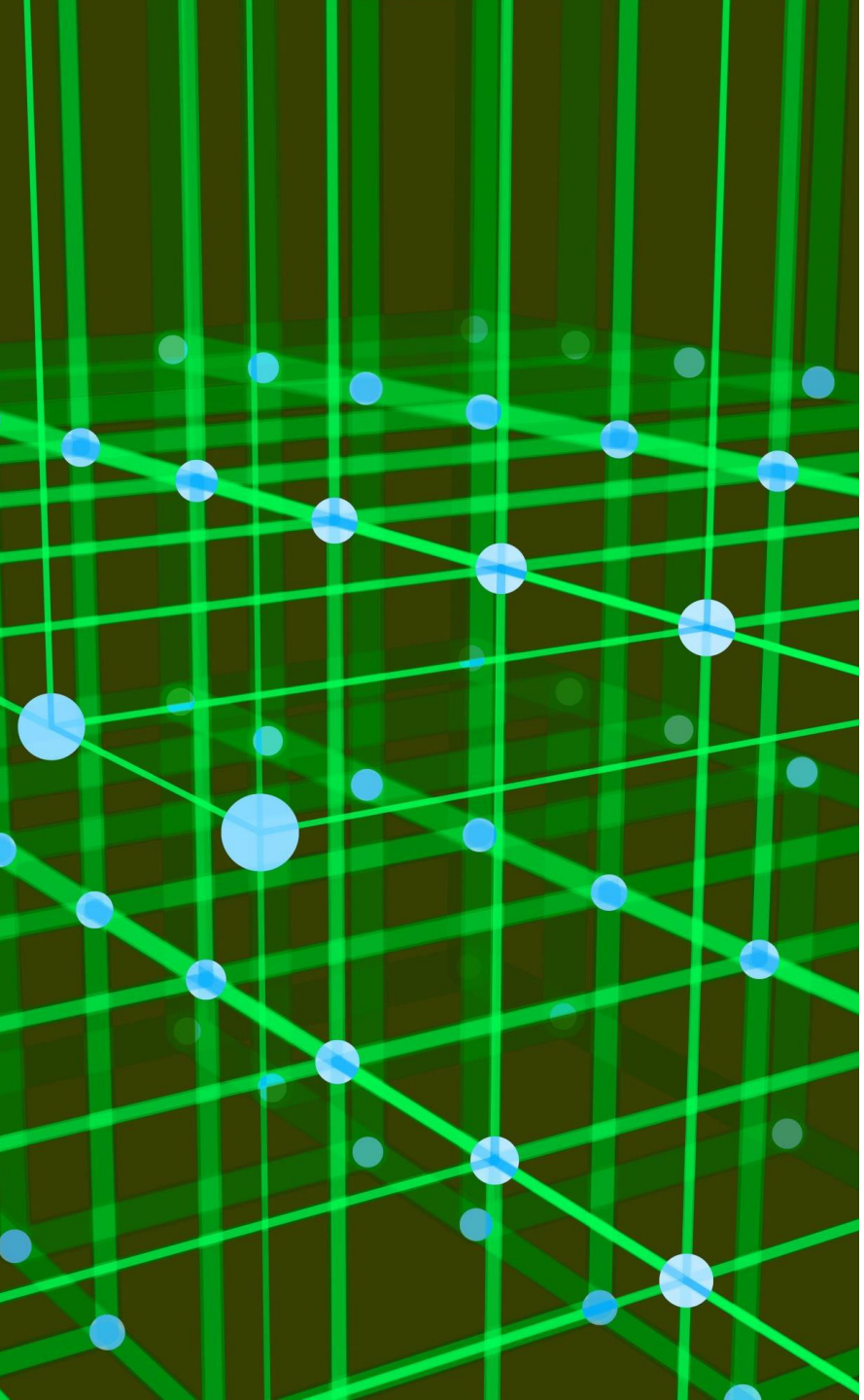
La clase GrafoNoDirigido es una herramienta útil para representar grafos no dirigidos y sus algoritmos. Esto permite modelar conjuntos de objetos y relaciones entre ellos para su posterior análisis y manipulación.

Función de Ordenación por Inserción

La función de Ordenación por Inserción es una técnica simple y eficiente para ordenar un conjunto de elementos. Se basa en comparar cada elemento con los que le preceden en la lista y moverlos a su posición correcta.

Clase Viajante

La clase Viajante permite modelar el problema del viajante de comercio, donde se busca encontrar la ruta más corta que visita un conjunto de ciudades. Este problema se utiliza en muchas áreas, como la logística y la planificación de rutas.



CLASE GRAFONODIRIGIDO Y SUS MÉTODOS

Modelado de Grafo no dirigido

La clase GrafoNoDirigido permite modelar un grafo no dirigido, que es un tipo de grafo en el que las aristas no tienen una dirección fija entre los nodos. Esto resulta útil para modelar relaciones no direccionales, como conexiones entre redes de computadoras.

Métodos para agregar y eliminar nodos y aristas

La clase GrafoNoDirigido tiene varios métodos que permiten agregar y eliminar nodos y aristas en el grafo. Estos métodos incluyen add_node(), add_edge(), remove_node() y remove_edge(). Estos métodos son útiles para manipular la estructura del grafo de acuerdo con los requisitos específicos de la aplicación.

ADD_NODE()

El método add_node() en teoría de grafos es utilizado para agregar un nuevo nodo en el grafo. Este método recibe un parámetro que representa el identificador único del nodo a agregar.

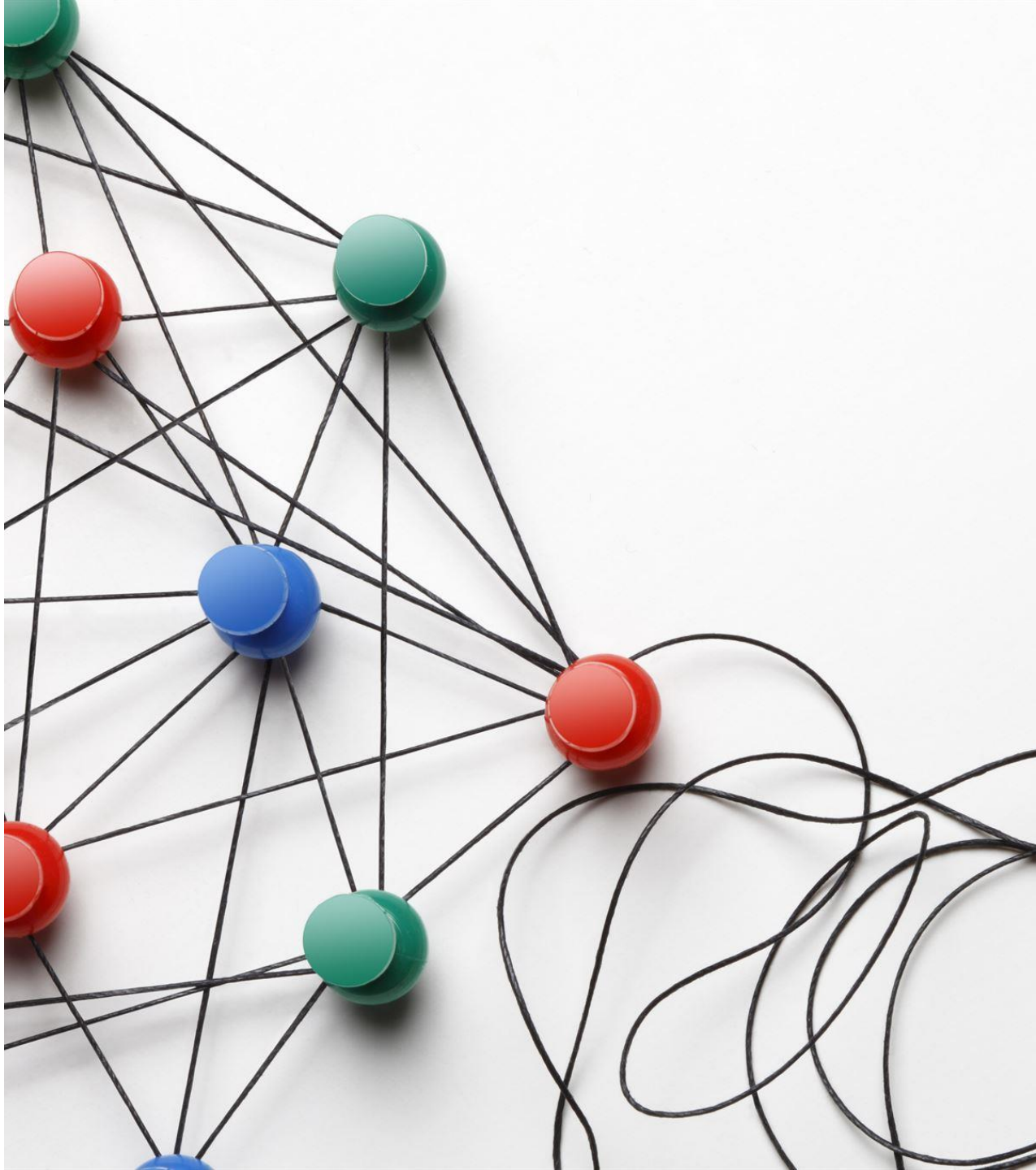
ADD_EDGE()

¿Qué es add_edge()?

El método add_edge() es una función utilizada en teoría de grafos que permite agregar una arista entre dos nodos en el grafo.

¿Cómo funciona add_edge()?

El método add_edge() requiere dos parámetros que representan los identificadores únicos de los nodos que se conectarán. Una vez que se proporcionan estos parámetros, el método agrega una arista entre los dos nodos en el grafo.

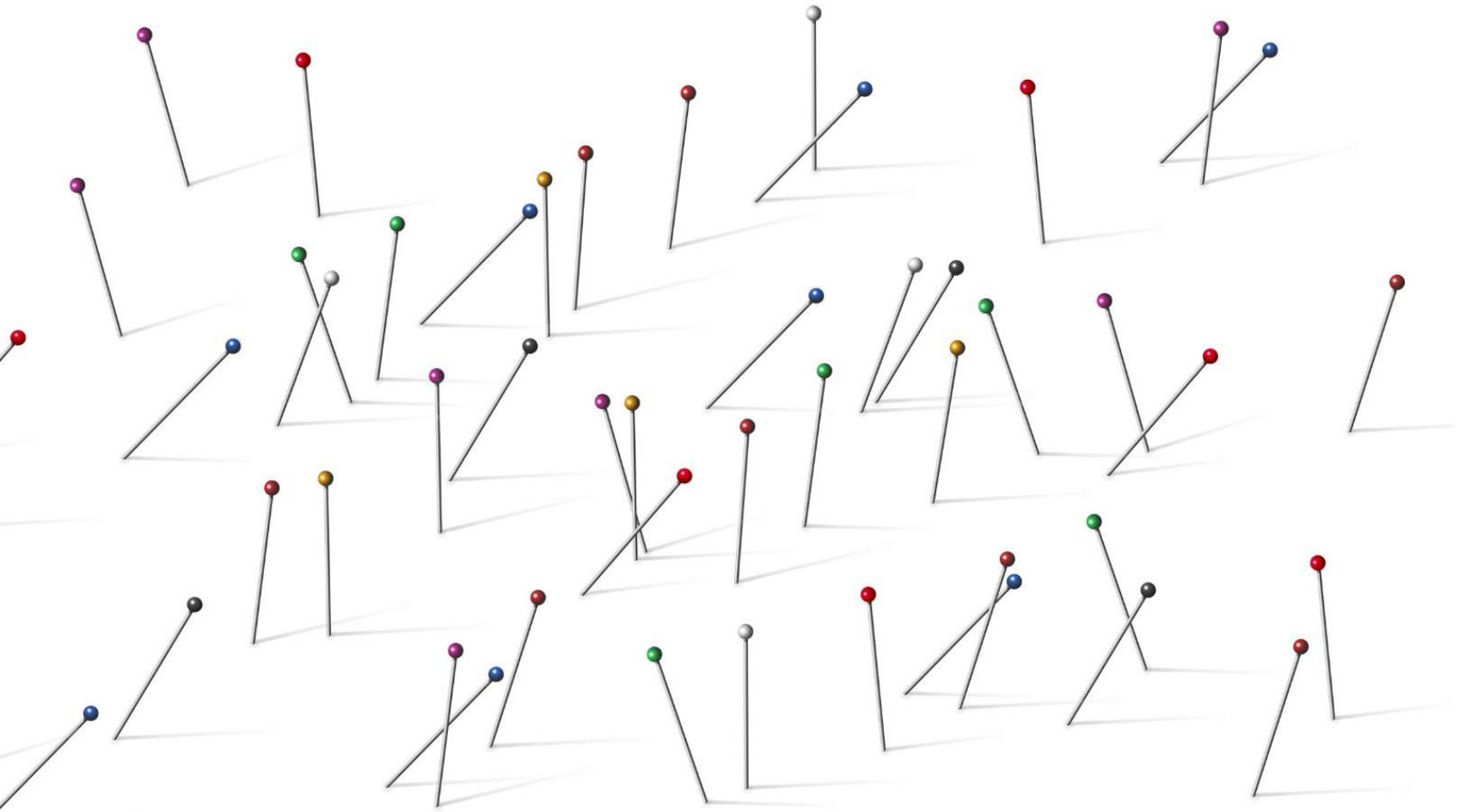


REMOVE_NODE()

El método remove_node() es una función de Python que permite eliminar un nodo del grafo y todas las aristas que se conectan a él. Este método requiere el identificador único del nodo que se desea eliminar como parámetro.

REMOVE_EDGE()

El método remove_edge() es una función en teoría de grafos que permite eliminar una arista entre dos nodos en un grafo. Este método toma dos parámetros que representan los identificadores únicos de los nodos que están conectados por la arista a eliminar.



ORDENACIÓN POR INSERCIÓN Y SU IMPORTANCIA

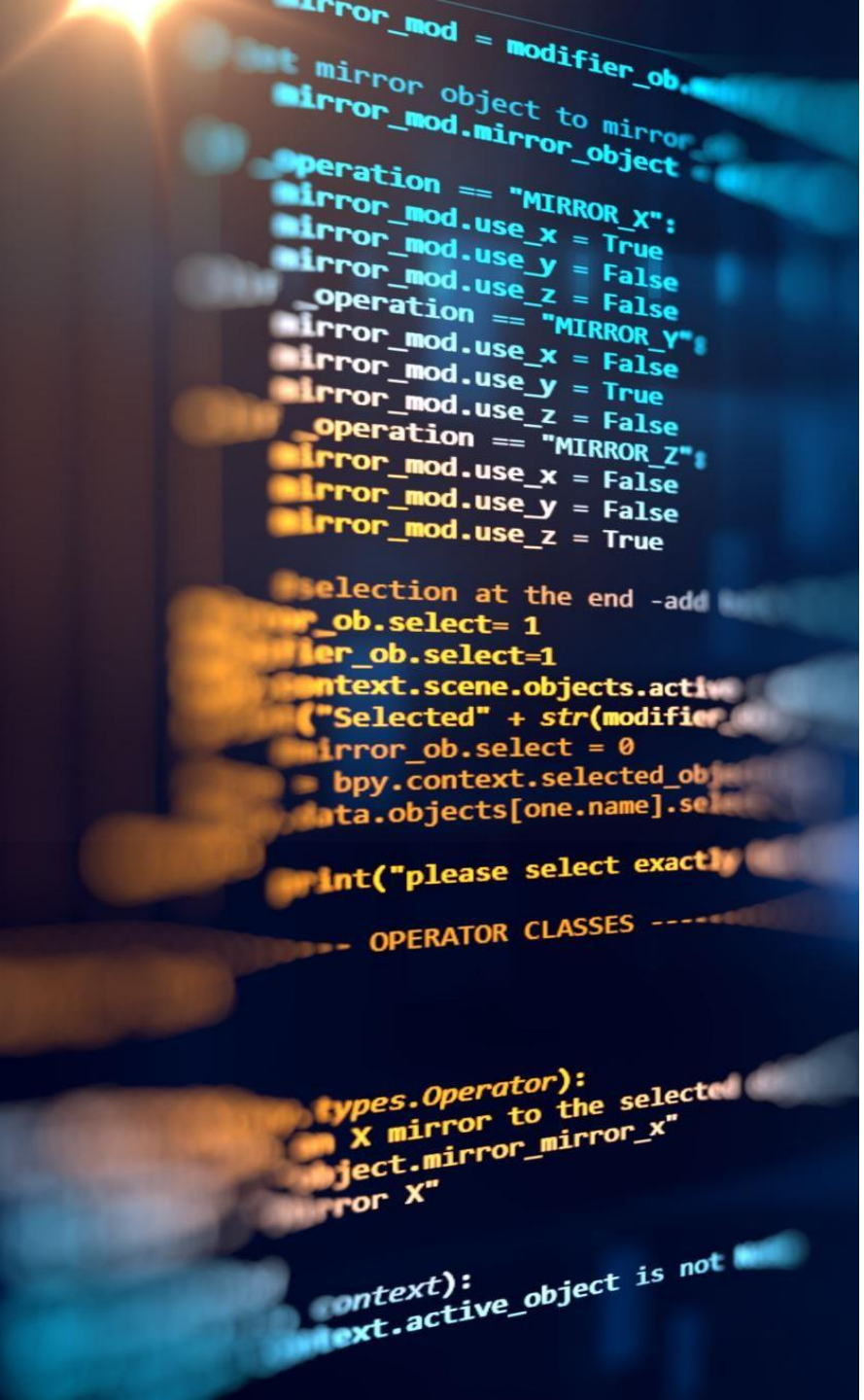
La Ordenación por Inserción es un algoritmo de ordenación que permite ordenar un conjunto de elementos. Además, es sencillo y eficiente para pequeñas cantidades de elementos, lo que la hace muy útil en ciencias de la computación.

¿QUÉ ES LA ORDENACIÓN POR INSERCIÓN?

La Ordenación por Inserción es un algoritmo de ordenación que recorre una lista de elementos y los va ordenando uno por uno, mediante la inserción del elemento en su posición correcta en la lista ya ordenada.

¿CÓMO FUNCIONA LA ORDENACIÓN POR INSERCIÓN?

La Ordenación por Inserción es un algoritmo eficiente para ordenar elementos de una lista por comparación y movimiento de elementos adyacentes hasta que estén en la posición correcta.



¿POR QUÉ ES IMPORTANTE LA ORDENACIÓN POR INSERCIÓN?

Sencillez

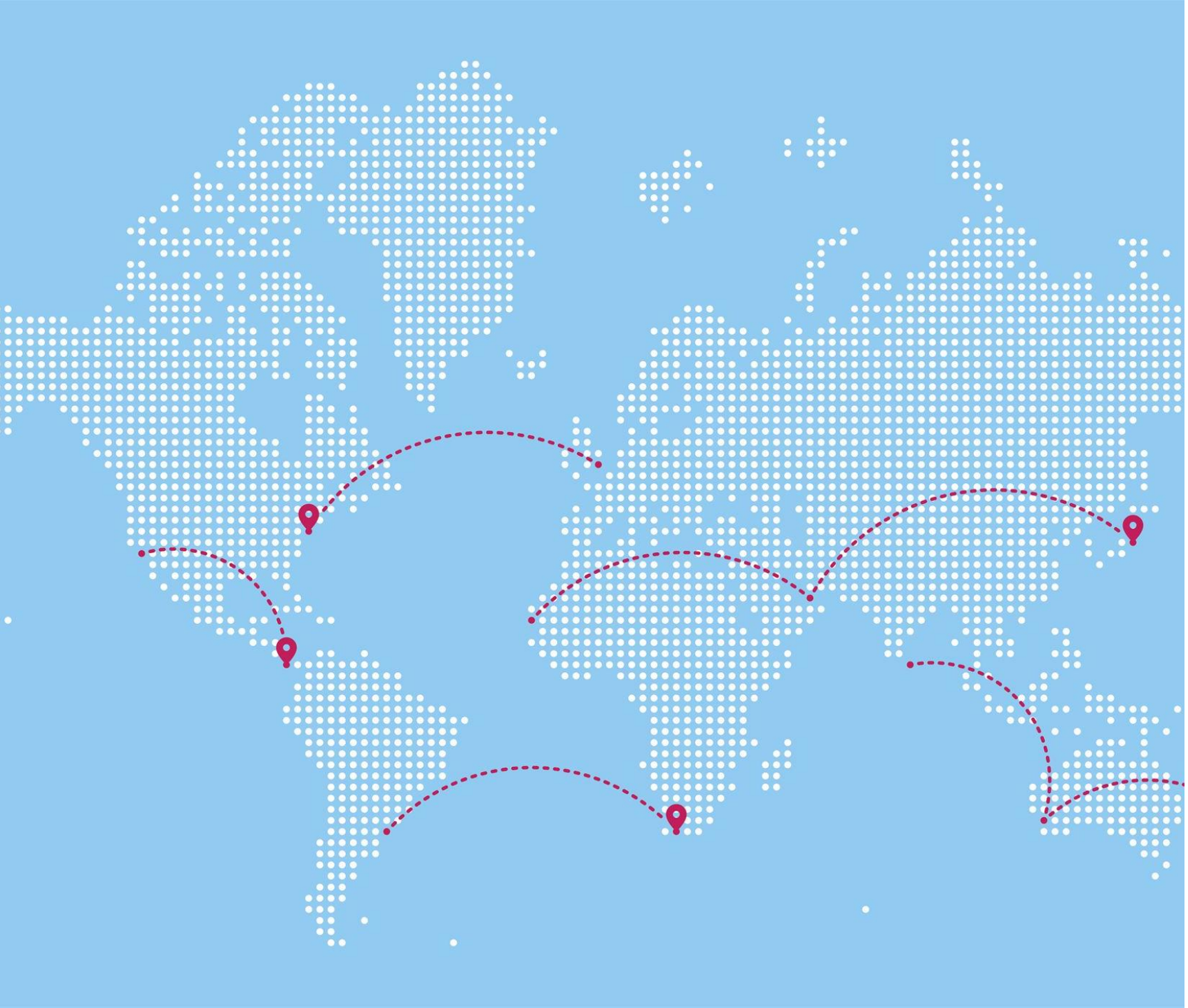
La Ordenación por Inserción es importante en ciencias de la computación debido a su simplicidad, lo que permite una rápida implementación y facilidad de comprensión, especialmente para pequeñas cantidades de elementos.

Eficiencia

La Ordenación por Inserción es un algoritmo eficiente que funciona en tiempo O(n^2) y es ideal para pequeñas cantidades de elementos, lo que lo hace adecuado para entornos con recursos limitados.

Estabilidad

La Ordenación por Inserción es un algoritmo estable que mantiene el orden original de los elementos en caso de que tengan el mismo valor. Esto es importante cuando el orden original debe ser preservado.



CLASE VIAJANTE Y SUS FUNCIONES

La clase Viajante es una herramienta útil para encontrar la ruta más corta que conecta un conjunto de ciudades en el problema del viajante de comercio. Los métodos incluyen get_distance(), get_path(), add_city() y remove_city().

GET_DISTANCE()

El método get_distance() es una función que permite a los usuarios obtener la distancia entre dos ciudades específicas en una ruta de viaje. Este método es útil para calcular la distancia total de un viaje y para planificar rutas de viaje precisas.

GET_PATH()

El método get_path() es una función que permite obtener la ruta del viajante que visita todas las ciudades una sola vez y regresa al punto de partida. Es una solución eficiente para el problema del viajante de comercio.

ADD_CITY()

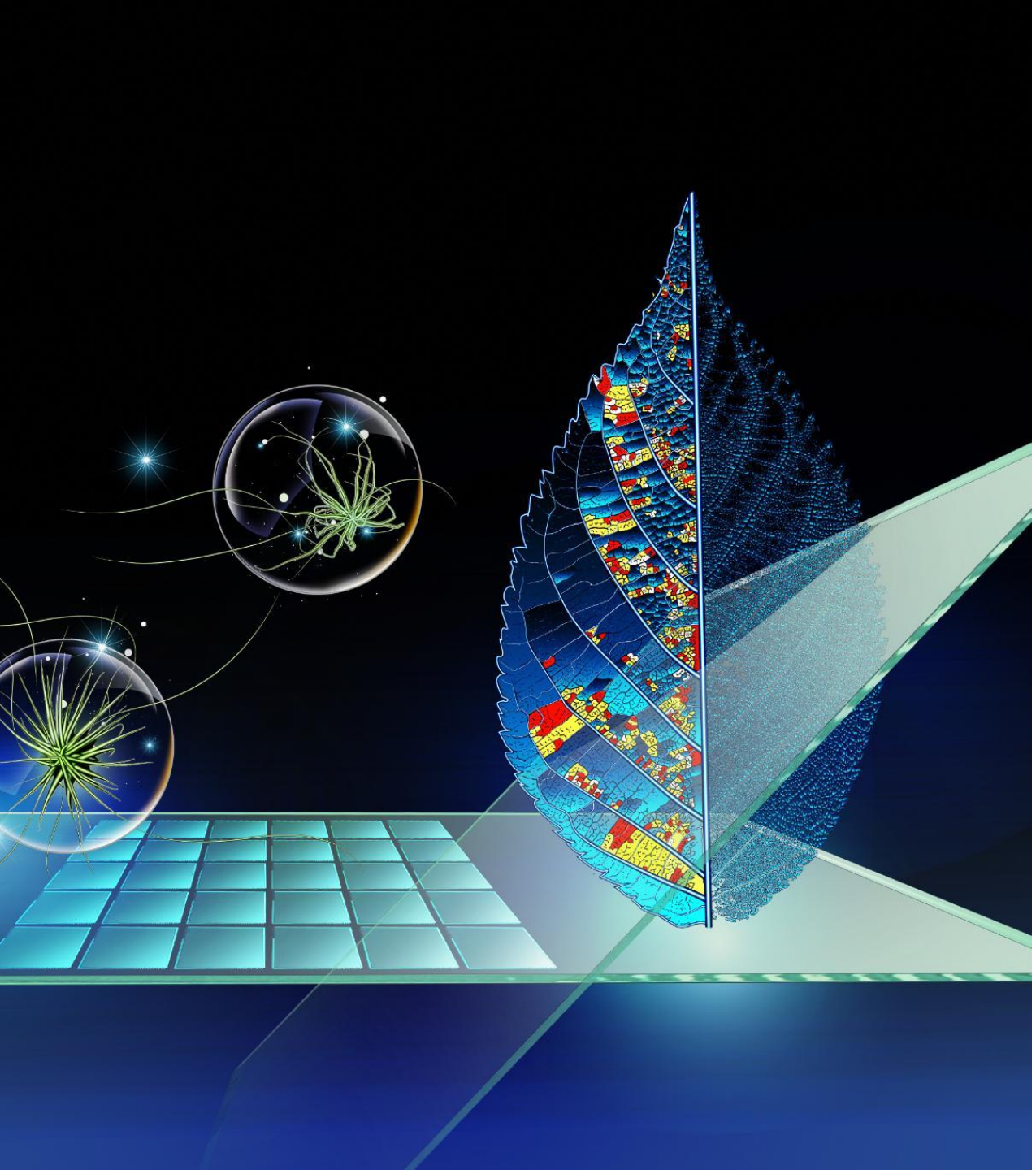
El método add_city() permite agregar una ciudad al conjunto de ciudades que el viajante debe visitar. Este método recibe un parámetro que representa el identificador único de la ciudad a agregar.

REMOVE_CITY()

El método remove_city() es una función que permite eliminar una ciudad específica del conjunto de ciudades que el usuario debe visitar durante un viaje. Este método es útil para ajustar la planificación de un itinerario.

EJECUCIÓN CÓDIGO

- Se define una matriz de distancias entre nodos, se crea una instancia de Viajante con estas distancias.
- Se llama al método heuristica() para calcular el recorrido.
- Se imprimen las aristas seleccionadas, el recorrido y la distancia total recorrida.
- Esto proporciona una solución aproximada al problema del viajante, que busca minimizar la distancia total del recorrido entre un conjunto de ciudades.



CONCLUSIÓN

En esta presentación hemos aprendido acerca de la clase GrafoNoDirigido y sus métodos, la función de Ordenación por Inserción y su importancia, y la clase Viajante y sus funciones. Estos conceptos son importantes para la ciencia de la computación y su aplicación en problemas de optimización y análisis de datos.