Esta primera práctica está orientada al desarrollo de técnicas de búsqueda tanto local como global. La práctica se divide en dos partes: uso de la técnica *Hill Climbing*; y uso de *Simulated Annealing*. El alumno deberá realizar un análisis teórico y experimenta de los apartados que se detallan en este documento, contestando a las preguntas de manera clara y usando los resultados/tablas/gráficas que sean necesarias para dar respuesta a las cuestiones planteadas. Tanto el **código nuevo generado como el informe detallado deberán entregarse por el grupo de prácticas antes de la fecha establecida**, teniendo en cuenta que esta primera práctica tiene una duración de 2 sesiones.

1. *Hill Climbing*

En este primer apartado vamos a trabajar con una búsqueda local utilizando el código Python *HillClimbing.py* que se proporciona. El código corresponde a un algoritmo genérico de escalada por la máxima pendiente para resolver el problema del TSP. En este problema, se considerará un vecino de una solución dada como aquel que intercambia dos ciudades en su representación. Por ejemplo, el recorrido [0, 2, 1, 3] será vecino del [0, 1, 2, 3], pero no será vecino del [1, 3, 2, 0]. Analiza el código proporcionado y comprueba que es correcto y cumple con los requisitos. Toda experimentación deberá llevarse a cabo con los conjuntos de datos que se proporcionan en la **Sección 3**. En primer lugar, prueba el código con los conjuntos de datos proporcionados con el fin de familiarizarte con tanto con el algoritmo como con los conjuntos de datos que se proporcionan. Una vez preparado, realiza las siguientes tareas:

- Analiza en detalle cómo se comporta el algoritmo a medida que aumentamos la complejidad del problema (espacio de búsqueda). Considera el número de ciudades para ello. Para ello, analiza el espacio de búsqueda.
- Realiza la experimentación que consideres oportuna para contestar a las siguientes cuestiones. Utiliza las pruebas y resultados experimentales que necesites: ¿El algoritmo obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?
- Realiza las modificaciones que consideres oportunas sobre el código proporcionado para obtener un Iterated Local Search. Una vez hecho esto, realiza la experimentación que consideres oportuna para contestar a las siguientes cuestiones. Utiliza las pruebas y resultados experimentales que necesites: ¿El algoritmo obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?
- Realiza una comparativa entre los dos algoritmos para comprobar cual se comporta mejor. Analiza tiempos, caminos obtenidos, y cualquier otra cosa que se te ocurra para comparar.

2. *Simulated Annealing*

En este segundo apartado vamos a trabajar con una búsqueda global utilizando el código Python *SimAnnealing.py*. El código proporcionado corresponde a un algoritmo genérico de recocido simulado para resolver el problema del TSP. El algoritmo trae por defecto una temperatura inicial de 10 y una función de enfriamiento en el que la temperatura desciende un 1 % en cada iteración. El criterio de parada es una temperatura igual o inferior a 0.05. Al igual que antes, un vecino será aquel que intercambia dos ciudades en su representación. Por ejemplo, el recorrido [0, 2, 1, 3] será vecino del [0, 1, 2, 3], pero no será vecino del [1, 3, 2, 0]. Analiza el código y comprueba que es correcto y cumple con los requisitos. En este caso, también se deberán de utilizar los conjuntos de datos expuestos en la **Sección 3**. En primer lugar, prueba el código con los conjuntos de datos proporcionados con el fin de familiarizarte con tanto con el algoritmo como con los conjuntos de datos que se proporcionan. Una vez preparado, realiza las siguientes tareas:

- Analiza en detalle cómo se comporta el algoritmo a medida que aumentamos la complejidad del problema (espacio de búsqueda). Considera el número de ciudades para ello. Para ello, analiza el espacio de búsqueda.
- Realiza la experimentación que consideres oportuna para contestar a las siguientes cuestiones. Utiliza las pruebas y resultados experimentales que necesites: ¿El algoritmo obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?

- Realiza las modificaciones que consideres oportunas sobre el código proporcionado para considerar diferentes criterios de parada y temperatura inicial. Una vez hecho esto, realiza la experimentación que consideres oportuna para contestar a las siguientes cuestiones. Utiliza las pruebas y resultados experimentales que necesites: ¿Ha variado el comportamiento? ¿De qué manera? ¿Por qué?.
- Modifica el código para utilizar diferentes funciones de enfriamiento:
  - Logarítmico (ver Ecuacion 1). T_o es la temperatura inicial; α es la velocidad de enfriamiento (valores menores de 1 aceleran el enfriamiento); k es la iteración en la que te encuentras.

$$T_k = \frac{\alpha * T_o}{ln(1+k)} \tag{1}$$

  - Geométrico (ver Ecuación 2). T_o es la temperatura inicial; α es la velocidad de enfriamiento (valores menores de 1); k es la iteración en la que te encuentras.

$$T_k = \alpha^k * T_o \tag{2}$$

  Realiza la experimentación que consideres oportuna para contestar las siguientes cuestiones usando los experimentos: ¿Cómo afectan estas funciones a los resultados finales? ¿Por qué? Si lo deseas, puedes buscar nuevas funciones y compararlas a las anteriores.
- ¿Cómo mejorarías el algoritmo? Modifica el código con la idea que se te ocurra y analiza los resultados.
- Utiliza todo el conocimiento que has adquirido hasta este punto, así como los resultados experimentales que tienes, para comparar Hill Climbing (y cualquier modificación) con Simulated Annealing (y cualquier modificación). ¿A qué conclusiones llegas?

3. Conjuntos de datos

Para llevar a cabo la experimentación, se colocará en *moodle* un fichero comprimido con los diferentes conjuntos de datos para el problema del TSP. De estos conjuntos de datos se conoce la solución óptima, permitiendo de esta forma comprobar, rápidamente, si los resultados obtenidos por los algoritmos de búsqueda han conseguido proporcionar la solución óptima. En la **Tabla 1**, se pueden encontrar los diferentes conjuntos de datos que se deberán de utilizar. Estos serán utilizados tanto para responder a las preguntas del *Hill Climbing* como para *Simulated Annealing*.

Conjunto de datos	Nº Ciudades	Distancia Mínima
FIVE	5	19
P01	15	291
GR17	17	2085
FRI26	26	937
DANTZIG42	42	699
ATT48	48	33523

Cuadro 1: Listado de conjuntos de datos con sus soluciones óptimas