Universidad de Magallanes
Facultad de Ingeniería
Departamento de Ingeniería
Estructuras de Datos – Tarea 4

Christian Vásquez Rebolledo
21 de noviembre al 15 de diciembre 2025

0. TRASFONDO DEL PROYECTO

El Problema del Vendedor Viajante (TSP por sus siglas en inglés, *Travelling Salesman Problem*) es un problema clásico que intenta responder a la siguiente interrogante: dada una lista de ciudades y las distancias entre cada par de ellas, ¿cuál es la ruta más corta posible para visitar cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen? Éste es un problema NP-Hard dentro en la optimización combinatoria, muy importante en investigación operativa y en ciencias de la computación. Por lo que, teniendo en cuenta que el problema es computacionalmente complejo, se verán enfrentados a resolver una variante en la que no necesariamente existen todas las conexiones entre cada par de ciudades.

1. OBJETIVOS

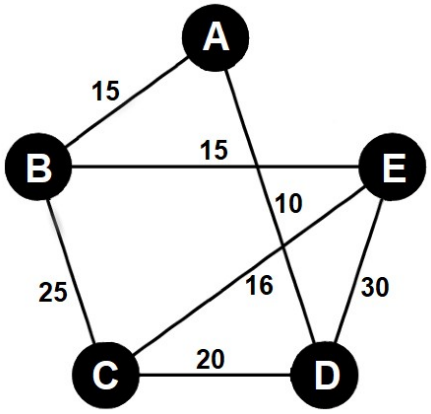
- Implementar y manipular estructuras de datos abstractas como grafos, árboles, listas enlazadas, etc. para almacenar y gestionar los datos manipulados.
- Desarrollar habilidades en programación en lenguaje C, centrándose en el manejo de memoria, punteros y *eficiencia algorítmica (?)*.
- Encontrar una solución práctica para uno de los problemas clásicos de grafos, aplicando técnicas y conocimientos aprendido a lo largo el curso.

2. DESCRIPCIÓN DEL PROYECTO

El problema a abordar consiste en buscar una ruta que recorra todas las ciudades disponibles solamente una vez, además regresando a la ciudad de origen, utilizando la ruta más corta posible. Sin embargo, para esta variante del problema en particular, antes de realizar cálculo alguno, primero se debe verificar la existencia de una ruta que efectivamente permita recorrer todas las ciudades volviendo a la original. Si esto no se cumple, el programa debe indicar que no es posible realizar el circuito esperado en el viaje y terminar inmediatamente; solamente en caso de que exista un camino viable debe continuar con el cálculo de la ruta más corta. Tenga presente que en caso de que exista una conexión entre un par de ciudades, el costo de recorrerla es el mismo independiente al sentido en que se haga.

La metodología para a utilizar para resolver el problema consiste en representarlo como un problema de grafos en que cada ciudad es un nodo y la distancia entre las ciudades son los enlaces entre las ciudades. Tome como ejemplo el siguiente caso, donde se muestran los costos de viajar entre 5 ciudades y el grafo que le corresponde:

	A	B	C	D
B	15			
C		25		
D	10		20	
E		15	16	30



Como se puede apreciar en este ejemplo, la ruta más corta que recorre todas las ciudades volviendo a la inicial corresponde a la ruta ABECDA con un costo total de 76 unidades.

Para obtener este resultado, es necesario dividir el problema en tres partes:
1. Verificar que existe un camino en que se pueda recorrer cada ciudad y regresar a la de origen.
2. Encontrar los posibles caminos que recorren todas las ciudades, volviendo a la inicial.
3. Calcular cuál es la ruta de menor costo.

Para cargar el grafo de las ciudades y las distancias entre cada par de ellas, el programa debe leerlas desde un archivo y luego verificar si es posible recorrer todos los nodos volviendo al original. Como la cantidad de caminos posibles crece exponencialmente, se limitará la cantidad de ciudades a recorrer a una cantidad baja.

Christian Vásquez Rebolledo

3. NORMATIVAS DE CODIFICACIÓN EN C

Consistencia en el estilo del código:

- Usar nombres de variables y funciones descriptivos y seguir un esquema de nomenclatura coherente (snake_case o camelCase).
- Mantener la consistencia en la indentación y espaciado (uso de 4 espacios por nivel de indentación recomendado).

Modularidad:

- Dividir el código en funciones pequeñas y manejables que realicen una única tarea.
- Evitar funciones que superen las 50 líneas de código, a menos que sea estrictamente necesario.

Documentación:

- Incluir comentarios claros y precisos para explicar la funcionalidad de bloques de código complejos.
- Documentar cada función con comentarios que expliquen su propósito, parámetros de entrada, valores de retorno y posibles errores.

Gestión de Errores:

- Implementar mecanismos robustos para la gestión de errores, asegurándose de que todos los errores posibles sean capturados y manejados adecuadamente.

Uso eficiente de memoria:

- Evitar el uso innecesario de variables globales.
- Liberar memoria dinámicamente asignada cuando ya no sea necesaria para evitar fugas de memoria.

Optimización

- Escribir código lo más optimizado posible para este ejercicio, pero sin limitarse por ello, siendo posible utilizar un enfoque de fuerza bruta en caso de ser necesario.

4. EJEMPLO DE USO

Suponga que se necesita calcular el mínimo costo de recorrer 5 ciudades.

Creación del grafo:
Debe inicializar el programa e indicarle que cree la estructura y asigne el tamaño indicado con *>pvv start 5*
Tenga en cuenta que la cantidad de ciudades a recorrer será el equivalente a la cantidad de nodos en su grafo.

Agregar enlaces:
Para leer los enlaces desde el archivo de texto “ruta.txt”, el usuario debe ejecutar el comando *>pvv read ruta.txt*
El archivo debe contener un enlace por línea, con el formato nodo de origen, nodo de destino y costo, separados por un espacio entre cada uno (ej: A B 15). Para cada ruta a verificar se debe tener un archivo de texto distinto.
Luego de leer los enlaces, el programa debe proceder automáticamente a verificar la existencia de una ruta viable.

Ver el grafo completo:
Para imprimir la representación en uso del grafo actual, el usuario puede ejecutar *>pvv graph*

Eliminación de estructuras:
Para eliminar las estructuras y cerrar el programa, el usuario debe ejecutar *>pvv exit*

Flujo de ejecución 1:

```
>pvv start 5
Grafo creado con 5 nodos

>pvv read ruta1.txt
Agregando enlaces desde archivo.
...
Verificando que existe una ruta.
...
No existe un camino que recorra todos las ciudades y regrese a la ciudad de origen.

>pvv exit
Limpiando cache y saliendo ...
```

Christian Vásquez Rebolledo

Flujo de ejecución 2:

```
>pvv start 5
Grafo creado con 5 nodos

>pvv read ruta2.txt
Agregando enlaces desde archivo.
...
Verificando que existe una ruta.
...
Existe un camino que recorre todos las ciudades y regresa a la ciudad de origen.
...
Ruta a seguir: A B E C D A
Costo total mínimo: 76

>pvv exit
Limpiando cache y saliendo ...
```

Para ejecutar pruebas, puede utilizar grafos de 4 y 5 nodos.

Tenga en cuenta que al momento de la revisión se usarán grafos de entre 5 y 9 nodos.

Christian Vásquez Rebolledo

5. ENTREGA Y EVALUACIÓN

- **Código fuente**: el código fuente correspondiente para la ejecución del programa solicitado. Debe estar bien documentado y seguir las normativas de codificación establecidas. *Debe incluir archivo Makefile correspondiente.* Debe poder ser compilado y ejecutado tanto en ambiente windows como linux.
- **Informe escrito**: se debe presentar un informe escrito, destacando las técnicas utilizadas y las decisiones detrás de su elección, así como los desafíos enfrentados y cómo se abordaron, en especial explicando el motivo de la elección de un algoritmo por sobre los demás.

Para un mayor detalle de la forma en que se graduará esta tarea, revisar el documento que contiene la pauta de evaluación.

La tarea deberán realizarla en grupos de 4 integrantes.
Deben enviar por correo el repositorio que usarán para ir subiendo la tarea el día 20 de noviembre, indicando el número de grupo asignado y los integrantes del grupo correspondiente. (Realizado en clases 👍)

La versión del código que se evaluará será la que esté presente en el repositorio designado para la entrega a las 23:59 del día lunes 15 de diciembre.

El informe escrito deberá ser enviarlo por el líder de cada grupo vía correo electrónico, teniendo como plazo la misma hora y fecha antes mencionadas; en ese momento, los líderes de grupo deben agregar una breve reseña del desempeño de sus compañeros, la cual será tomada en cuenta al momento de calificar sus tareas.

El correo electrónico debe tener como título “Informe Tarea 4 Grupo *X*”. Hago especial énfasis en que tenga el **grupo al que corresponde**.

Pista: Hamilton.