



UNIVERSIDAD AUTÓNOMA DE NAYARIT



Unidad Académica de Economía

Carrera

Licenciatura en Sistemas Computacionales

Integrantes

Ricardo Matos Vizcarra

Fernando Ozuna Manso

Erika Alejandra Orozco Vazquez

Obed Moncada Contreras

Tema

Grafos dirigidos y No dirigidos

Reporte Técnico

Materia

Estructura de Datos Avanzada

Sección 1: Diseño del modelo

- **Descripción del mapa:**

¿Qué representa cada vértice?

Un punto de encuentro en el cual le podemos dar nombre de cualquier forma, siempre y cuando lo podamos identificar, si nos basamos en la realización de una ciudad bien puede ser un lugar en específico como lo puede ser un parque.

¿Por qué elegiste esas conexiones?

Se nos hicieron más fáciles de ver gráficamente en el diagrama y la verdad no nos queríamos complicar demasiado por pocos elementos en un grafo.

- **Justificación de aristas:**

¿Porque algunas son bidireccionales y otras no?

Se hizo de esta forma porque buscamos realizar un grafo sencillo y fácil de representar, pero que al mismo tiempo tuviera un poco de complejidad con las indicaciones que se nos dieron.

- **Criterio de pesos:**

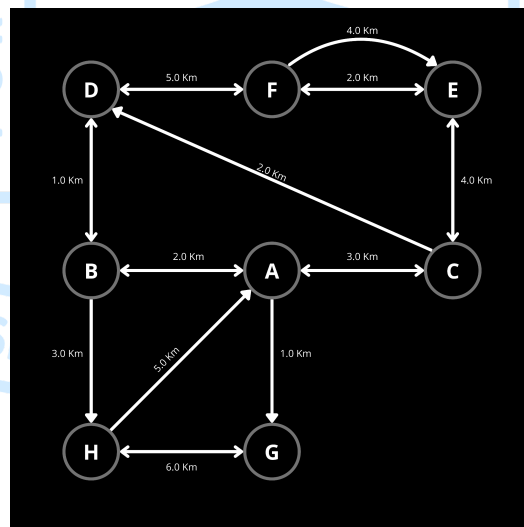
¿Cómo asignaste las distancias?

Se les asignó un poco aleatoriamente, tratando de que este tuviera coherencia y al mismo tiempo que el grafo se viera bien de forma estética y no fuera tan complejo de representar.

¿Son realistas?

No realmente, se les asignó una cierta distancia de una manera que creímos adecuada en base a nuestra percepción, sin tener en cuenta algo tan realista.

- **Diagrama:**



Sección 2: Decisiones de implementación

- **Lista vs Matriz:**

¿Por qué elegiste listas de adyacencia?

($n=8$, $m=12$) La lista de adyacencia es la elección correcta porque tenemos un grafo disperso (pocas conexiones por nodo o vértice) y necesitamos eficiencia en los recorridos.

- **Manejo de direccionalidad:**

¿Como tu código diferencia dirigido vs no dirigido?

Esto se hizo a través de un parametro en la funcion addEdge que marcaba como true cuando tenían direccion

- **Trade-offs identificados:**

¿Qué perdiste y qué ganaste con tu elección?

Es mas lento para identificar aristas, pero a cambio gasta mucho menos memoria y es mas rapido para recorrer vecinos (principalmente)

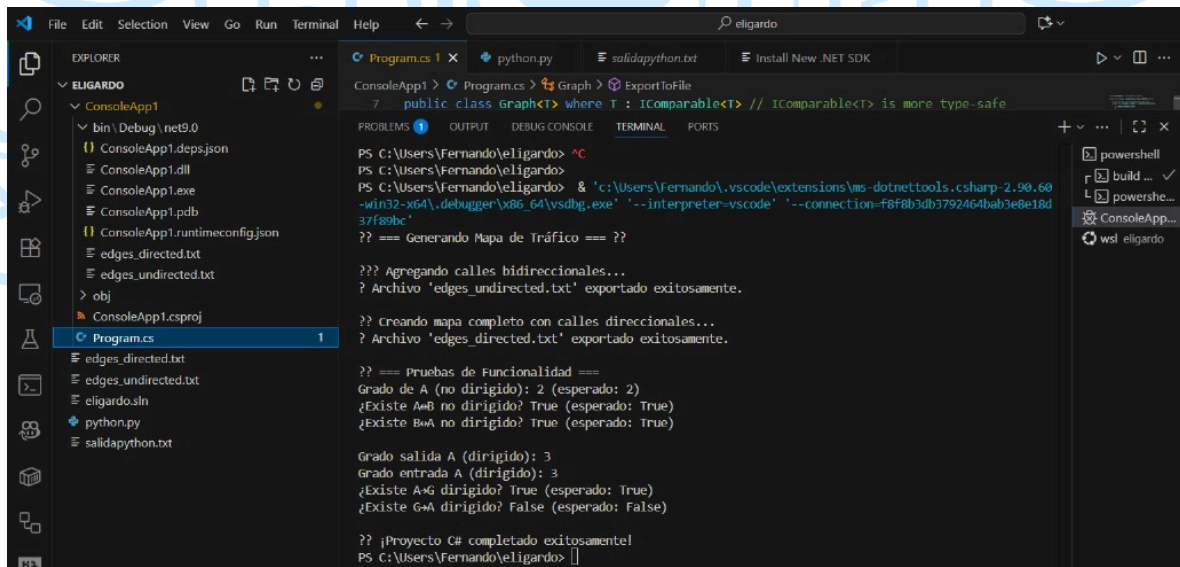
- **Escalabilidad:**

¿Cómo se comportaría tu solución con 100+ vértices?

Se seguiria comportando bien en la mayoría de los aspectos, en la comprobacion de aristas le costaria un poco mas pero a cambio de eso seria ideal en el resto de aspectos

Sección 3: Resultados y validación

- **Screenshots de ejecución:** (Salida de ambos programas)



```
Program.cs 1
public class Graph<T> where T : IComparable<T> // IComparable<T> is more type-safe

PS C:\Users\Fernando\eligardo> ^C
PS C:\Users\Fernando\eligardo>
PS C:\Users\Fernando\eligardo> & 'c:\Users\Fernando\.vscode\extensions\ms-dotnettools.csharp-2.90.60-win32-x64\debugger\x86_64\vsdbg.exe' '--interpreter=vscode' '--connection=f8f8b3db3792464bab3e18d37f89bc'
?? === Generando Mapa de Tráfico === ??

??? Agregando calles bidireccionales...
? Archivo 'edges_undirected.txt' exportado exitosamente.

?? Creando mapa completo con calles direccionales...
? Archivo 'edges_directed.txt' exportado exitosamente.

?? === Pruebas de Funcionalidad ===
Grado de A (no dirigido): 2 (esperado: 2)
¿Existe A↔B no dirigido? True (esperado: True)
¿Existe B↔A no dirigido? True (esperado: True)

Grado salida A (dirigido): 3
Grado entrada A (dirigido): 3
¿Existe A→G dirigido? True (esperado: True)
¿Existe G→A dirigido? False (esperado: False)

?? ¡Proyecto C# completado exitosamente!
PS C:\Users\Fernando\eligardo> 
```

```

=== Análisis de Mapas de Tráfico - Proyecto Semana 3 ===

=====
🔍 Análisis del Grafo No Dirigido
=====

📊 Estadísticas generales:
• Vértices: 8
• Aristas: 14
• Densidad: 0.250

🔍 Detalles por vértice:
A: Out-degree=2, In-degree=2
  ↳ Vecinos: [B(2.0km), C(3.0km)]
B: Out-degree=2, In-degree=2
  ↳ Vecinos: [A(2.0km), D(1.0km)]
C: Out-degree=2, In-degree=2
  ↳ Vecinos: [A(3.0km), E(4.0km)]
D: Out-degree=2, In-degree=2
  ↳ Vecinos: [B(1.0km), F(5.0km)]
E: Out-degree=2, In-degree=2
  ↳ Vecinos: [C(4.0km), F(2.0km)]
F: Out-degree=2, In-degree=2
  ↳ Vecinos: [D(5.0km), E(2.0km)]
G: Out-degree=1, In-degree=1
  ↳ Vecinos: [H(6.0km)]
H: Out-degree=1, In-degree=1
  ↳ Vecinos: [G(6.0km)]

```

```

=====
🔍 Análisis del Grafo Dirigido
=====

📊 Estadísticas generales:
• Vértices: 8
• Aristas: 19
• Densidad: 0.339

🔍 Detalles por vértice:
A: Out-degree=3, In-degree=3
  ↳ Vecinos: [G(1.0km), B(2.0km), C(3.0km)]
B: Out-degree=3, In-degree=2
  ↳ Vecinos: [H(3.0km), A(2.0km), D(1.0km)]
C: Out-degree=3, In-degree=2
  ↳ Vecinos: [D(2.0km), A(3.0km), E(4.0km)]
D: Out-degree=2, In-degree=3
  ↳ Vecinos: [B(1.0km), F(5.0km)]
E: Out-degree=2, In-degree=3
  ↳ Vecinos: [C(4.0km), F(2.0km)]
F: Out-degree=3, In-degree=2
  ↳ Vecinos: [E(4.0km), D(5.0km), E(2.0km)]
G: Out-degree=1, In-degree=2
  ↳ Vecinos: [H(6.0km)]
H: Out-degree=2, In-degree=2
  ↳ Vecinos: [A(5.0km), G(6.0km)]

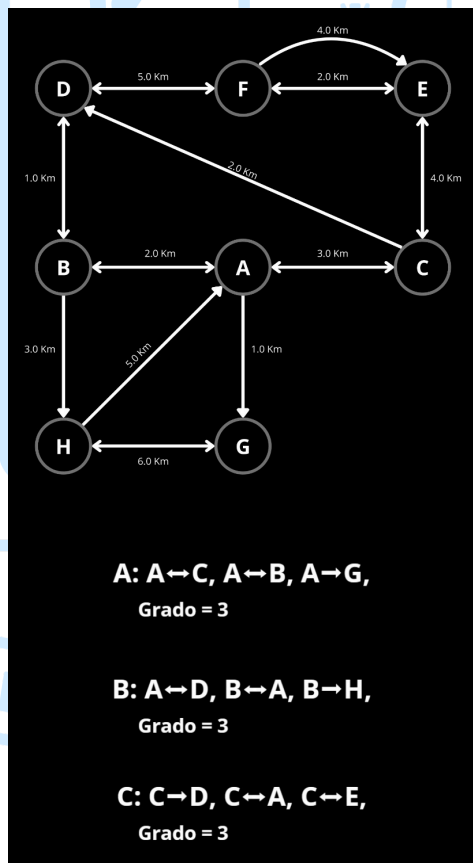
=====
🔗 Pruebas de Conectividad
=====

¿A→G dirigido? True (esperado: True)
¿G→A dirigido? False (esperado: False)
Grado de salida de A: 3
Grado de entrada de A: 3
Vértice más conectado: A

🎉 ¡Análisis completado exitosamente!

```

- Verificación manual: (Comprueba 2-3 cálculos de grado a mano)



- **Casos de prueba:**

¿Qué aristas probaste que existen/no existen?

Comprobamos la arista de $A \rightarrow G$ (Que si existe) y de $G \rightarrow A$ (que no existe)

Sección 4: Reflexión Personal

- **¿Qué fue lo más desafiante del proyecto?**

La realización del código y que funcionara al 100%

- **¿Cómo aplicarías grafos en un problema real de tu interés?**

Nosotros los aplicamos por ejemplo en videojuegos , para realizar estrategias de posicionamiento del equipo y lograr un mejor desempeño.

- **¿Qué cambiarías en tu implementación si empezaras de nuevo?**

Cambiaría el lenguaje de c# a c++ o rust ya que de c# tengo bastante poca experiencia, también haría que creara un solo archivo y no 2 archivos de direccionado y no direccionado, y crear una señal para que el programa de python distinguiera entre direccionado y no direccionado

