



Tecnológico de Monterrey

Campus Querétaro

Programación de estructuras de datos y algoritmos fundamentales | Gpo 601

### **Act 4.3 - Actividad Integral de Grafos (Evidencia Competencia)**

Maestro

Francisco Javier Navarro Barrón

Presenta:

Karla Alejandra Padilla González A01705331

Alejandra Cabrera Ruiz A01704463

26 de octubre de 2022

## **Grafos**

Un grafo es un par compuesto por dos conjuntos  $V$  y  $A$ . Al conjunto  $V$  se le llama conjunto de vértices o nodos del grafo, por otro lado, el conjunto  $A$  se les conoce como arcos o ejes del grafo. Y existen dos clases de grafos: dirigidos y no dirigidos, en el caso del grafo dirigido cada arco está representado por un par ordenado de vértices y en un grafo no dirigido el par de vértices representa un arco que no está ordenado, se pueden representar de manera gráfica, permiten representar conjuntos de objetos arbitrariamente relacionados.

Existen múltiples formas de representación de grafos entre ellas matriz de adyacencia, listas de adyacencia y matriz de incidencia. En el caso del desarrollo de la actividad 4.3 se utilizó una matriz sin peso para representar el grafo. La matriz de adyacencia es una matriz booleana que representa las conexiones entre pares de vértices. Este tipo de matriz de un grafo tiene múltiples características entre las cuales destaca su simetría, el hecho de que si el grafo es simple la matriz estará compuesta por ceros y unos, y su diagonal por ceros.

Existen múltiples tipos de recorridos, el utilizado para esta actividad es DFS o Depth First Search donde a partir de un nodo dado se localizan sus vértices adyacentes y para cada uno de ellos se inicia nuevamente el recorrido dfs.

## **Reflexión Karla Padilla**

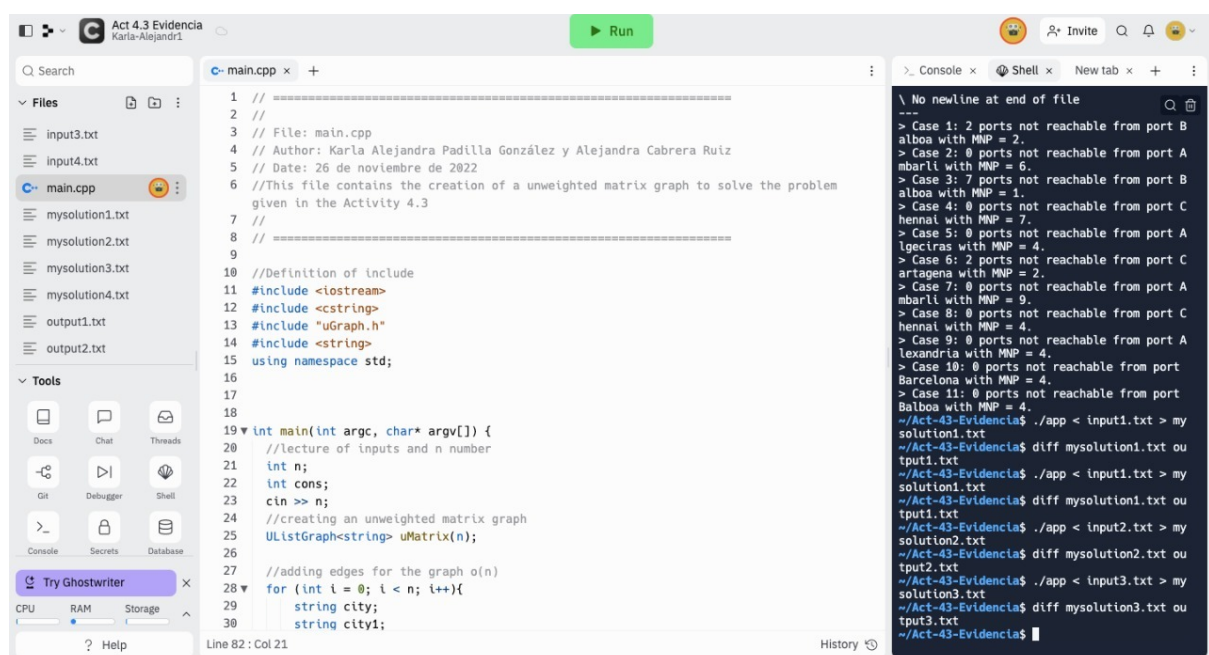
Son importantes y los vemos relacionados en problemas de la vida real, como lo son en modelos matemáticos de numerosas situaciones reales como: un mapa de carreteras, el plano de un circuito eléctrico, el esquema de la red telefónica de una compañía, etc. Su representación es de tres maneras básicas, mediante matrices, listas y matrices dispersas, cada una de las representaciones tienen ciertas ventajas y desventajas. Los grafos son importantes porque es una representación natural de redes y que además permiten expresar de forma visualmente sencilla las relaciones que se dan entre los elementos de un estudio, esto quiere decir que facilita la resolución de problemas de una manera práctica, que además representan una eficiencia algorítmica en aplicación cuando se utilizan. Por último, considero que el poder aplicar este método para relacionar un conjunto de datos y de manera gráfica facilitan el uso de alguna búsqueda o de obtener información de manera más sencilla, y disminuyendo cualquier tipo de error en su búsqueda, ya que existen diferentes tipos de representación de estos grafos que se han mencionado anteriormente.

## **Reflexión Alejandra Cabrera**

A lo largo del desarrollo de esta actividad nos dimos cuenta de la utilidad que tiene una estructura de datos como es un grafo, y la aplicación que podría tener en la vida real. En el caso de esta situación problema hablábamos de una compañía naviera y de la cantidad de puertos que puede visitar un carguero, el resultado de la implementación de nuestro algoritmo podría ayudar a optimizar los viajes y crear perspectivas reales. Facilitando y automatizando el análisis de los puertos a visitar de acuerdo con su MNP.

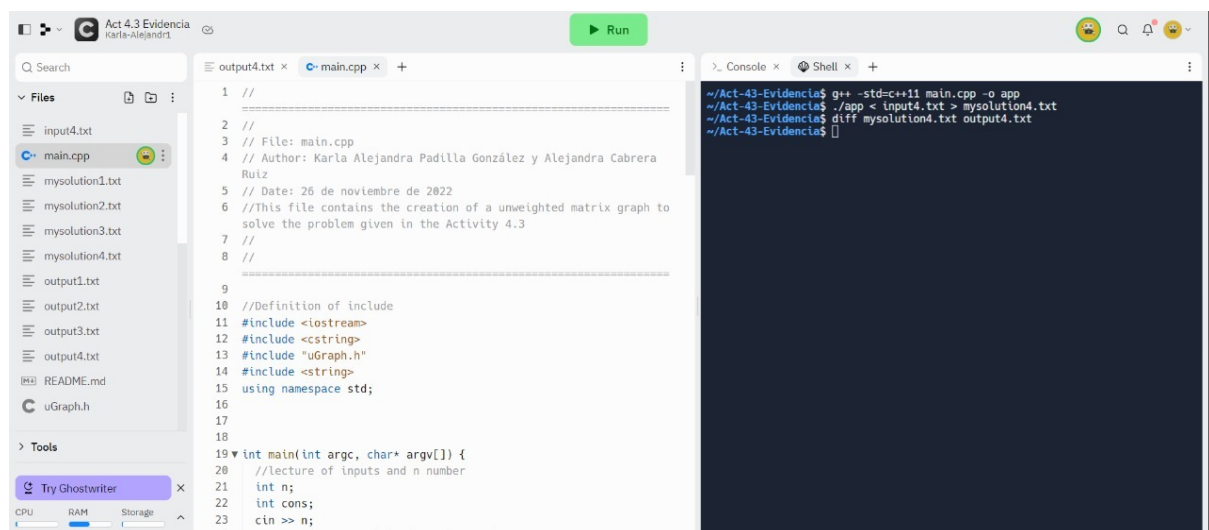
Al final, los grafos tienen muchas aplicaciones comunes como los circuitos eléctricos, el análisis de redes e incluso en el área de los negocios para diseñar nuevas estrategias de mercado. Algo que nos fue útil en el desarrollo de la evidencia fue el ver implementadas las funciones de un Unweighted Matrix Graph como el dfs, ya que al final este recorrido tiene una implementación parecida o igual para todos los casos. Creo que una de las cosas que más me cuesta en lo personal es detectar los pequeños errores al correr el código en shell porque muchas veces el diff sale diferente y es cuestión de un espacio o incluso un punto, como en este caso. Pero logramos detectarlos y concluir la evidencia exitosamente con los cuatro casos de prueba.

## Demostración de los casos de prueba



```
1 // =====
2 //
3 // File: main.cpp
4 // Author: Karla Alejandra Padilla González y Alejandra Cabrera Ruiz
5 // Date: 26 de noviembre de 2022
6 //This file contains the creation of a unweighted matrix graph to solve the problem
  given in the Activity 4.3
7 //
8 // =====
9
10 //Definition of include
11 #include <iostream>
12 #include <cstring>
13 #include "uGraph.h"
14 #include <string>
15 using namespace std;
16
17
18
19 int main(int argc, char* argv[]) {
20     //Lecture of inputs and n number
21     int n;
22     int cons;
23     cin >> n;
24     //creating an unweighted matrix graph
25     UListGraph<string> uMatrix(n);
26
27     //adding edges for the graph o(n)
28     for (int i = 0; i < n; i++){
29         string city;
30         string city1;
```

```
\ No newline at end of file
---
> Case 1: 2 ports not reachable from port B
alboa with MNP = 2.
> Case 2: 0 ports not reachable from port A
mbarli with MNP = 6.
> Case 3: 7 ports not reachable from port B
alboa with MNP = 1.
> Case 4: 0 ports not reachable from port C
hennal with MNP = 7.
> Case 5: 0 ports not reachable from port A
lgeciras with MNP = 4.
> Case 6: 2 ports not reachable from port C
artagena with MNP = 2.
> Case 7: 0 ports not reachable from port A
mbarli with MNP = 9.
> Case 8: 0 ports not reachable from port C
hennal with MNP = 4.
> Case 9: 0 ports not reachable from port A
lexandria with MNP = 4.
> Case 10: 0 ports not reachable from port
Barcelona with MNP = 4.
> Case 11: 0 ports not reachable from port
Balboa with MNP = 4.
~/Act-43-Evidencia$ ./app < input1.txt > my
solution1.txt
~/Act-43-Evidencia$ diff mysolution1.txt ou
tput1.txt
~/Act-43-Evidencia$ ./app < input1.txt > my
solution1.txt
~/Act-43-Evidencia$ diff mysolution1.txt ou
tput1.txt
~/Act-43-Evidencia$ ./app < input2.txt > my
solution2.txt
~/Act-43-Evidencia$ diff mysolution2.txt ou
tput2.txt
~/Act-43-Evidencia$ ./app < input3.txt > my
solution3.txt
~/Act-43-Evidencia$ diff mysolution3.txt ou
tput3.txt
~/Act-43-Evidencia$
```



```
1 //
2 //
3 // File: main.cpp
4 // Author: Karla Alejandra Padilla González y Alejandra Cabrera
  Ruiz
5 // Date: 26 de noviembre de 2022
6 //This file contains the creation of a unweighted matrix graph to
  solve the problem given in the Activity 4.3
7 //
8 //
9
10 //Definition of include
11 #include <iostream>
12 #include <cstring>
13 #include "uGraph.h"
14 #include <string>
15 using namespace std;
16
17
18
19 int main(int argc, char* argv[]) {
20     //Lecture of inputs and n number
21     int n;
22     int cons;
23     cin >> n;
```

```
~/Act-43-Evidencia$ g++ -std=c++11 main.cpp -o app
~/Act-43-Evidencia$ ./app < input4.txt > mysolution4.txt
~/Act-43-Evidencia$ diff mysolution4.txt output4.txt
~/Act-43-Evidencia$
```

## Referencias

García, M. (n.d.). *Estructura de Datos* Página 1 de 13.

<https://www.utm.mx/~mgarcia/ED4%28Grafos%29.pdf>

UV. (2022). Grafos. Estructura de datos.

[http://informatica.uv.es/iiguia/AED/oldwww/2001\\_02/Teoria/Tema\\_15alfa.pdf](http://informatica.uv.es/iiguia/AED/oldwww/2001_02/Teoria/Tema_15alfa.pdf)