# GRAFOS Y OBJETOS JSON

Andrés Rodríguez Prada [1]
Alejandra Pedraza Cárdenas [2]
Valentina Cortés Castellar [3]

04 de Marzo de 2020

Computación Gráfica

## Introducción

Una de las teorías que tiene más aplicaciones en la actualidad es la teoría de grafos, creada en 1730 por Leonhard Euler. La eficacia de este conjunto de vértices y aristas se debe a la clara representación que da de cualquier relación; en una aplicación de transito administrar las diferentes rutas que se tienen a un destino en específico y determinar cuál es la que tomará menos tiempo, cómo está construida una página web, hasta determinar cuál es el protagonista de una serie mediante su relación con otros personajes. Mediante los grafos se facilita tanto el planteamiento como la resolución del problema.

JSON (JavaScript Object Notation) es un formato para intercambio de datos constituido por una lista donde se almacenan objetos pertenecientes a una misma categoría, la información que se encuentra en los objetos se convierte en cadenas de caracteres para transmitir datos entre un servidor y una página web .

Palabras Claves: JSON, grafo, vértices, aristas, objeto, arreglo, carácter.

## 1. Marco teórico

**Grafos:** Es un conjunto de objetos no vacío, llamados vértices y pares de vértices llamados aristas. Este se considera uno de los primeros resultados geométricos que no dependen de ninguna medida.[1]

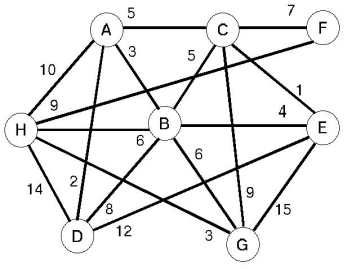



***Figura 1**. Ejemplo de grafo*

**Aristas dirigidas:** Son aristas que tienen

[1]6000363@unimilitar.edu.co

[2]6000370@unimilitar.edu.co

[3]6000360@unimilitar.edu.co

asignado un sentido, haciendo que el grafo obtenga una orientación.

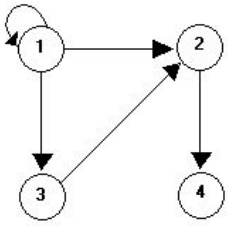



***Figura 2.*** *Aristas dirigidas.*

**Aristas no dirigidas:** Son aristas cuya dirección no está asignada, en consecuencia se consideran bidireccionales. Cuando un grafo presenta este tipo de aristas es conveniente decir que existen dos aristas entre vértices, donde cada una se orienta hacia un sentido opuesto.

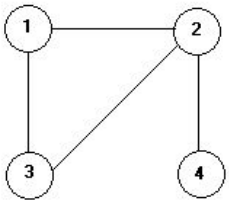



***Figura 3.*** *Aristas no dirigidas*

**Subgrafo:** Es un grafo cuyos conjuntos de vértices y aristas son subconjuntos del grafo original (G).

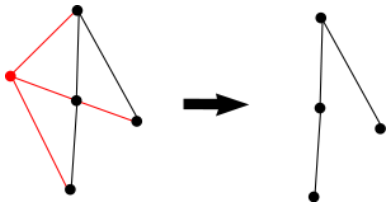


***Figura 4.*** *Subgrafo.*

**Estructuras de datos en representación de grafos:** Debido a la existencia de diversas formas de almacenamiento de datos existen diferentes tipos de grafos que dependen del algoritmo usado para manipularlos. Las dos estructuras más comunes son las listas y matrices.

**Matriz de incidencia:** En esta estructura el grafo se representa por una matriz de n aristas por i vértices, donde cada par (n-i) contiene la información de la arista representado como 1 indica conectado y 0 indica no conectado.

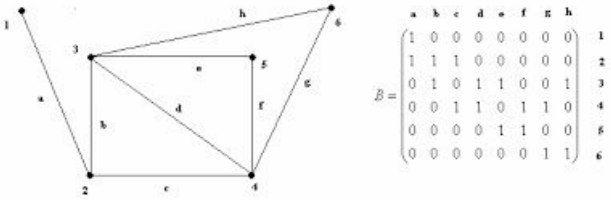


***Figura 5.*** *Matriz de incidencia*

**Lista de incidencia:** Es una estructura donde las aristas se representan con un vector de pares ordenados (Si el grafo está dirigido), donde cada par representa una de las aristas.

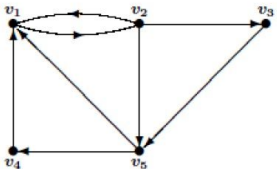
**Matriz de adyacencia:** Esta estructura

[1]6000363@unimilitar.edu.co

[2]6000370@unimilitar.edu.co

[3]6000360@unimilitar.edu.co

representa al grafo a través de una matriz cuadrada M de tamaño $n^2$ , donde n es el número de vértices.




| | V1 | V2 | V3 | V4 | V5 | |
|---|---|---|---|---|---|---|
| V1 | 0 | 1 | 0 | 0 | 0 | 1 |
| V2 | 1 | 0 | 1 | 0 | 1 | 3 |
| V3 | 0 | 0 | 0 | 0 | 1 | 1 |
| V4 | 1 | 0 | 0 | 0 | 0 | 1 |
| V5 | 1 | 0 | 0 | 1 | 0 | 2 |

***Figura 6.*** *Matriz de adyacencia.*

**Lista de adyacencia:** Este tipo de estructura combina las matrices de adyacencia con listas de aristas. Cada vértice, se tiene un arreglo de vértices adyacentes a él, es decir, una lista de adyacencia por vértice. Tiene como objetivo asociar cada vértice del grafo a una lista de vértices adyacentes a él. Este tipo de grafos se representan a través de un vector de n componentes, donde cada una será una lista de adyacencia correspondiente a cada vértice del grafo.

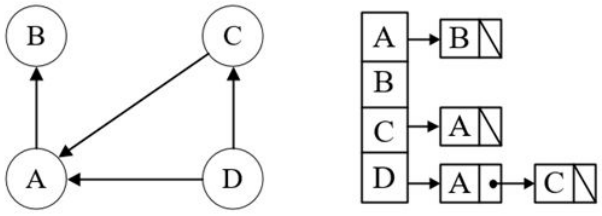



***Figura 7.*** *Lista de adyacencia.*

**javaScript Object Notation (JSON)**

Notación de Objetos de JavaScript es un formato ligero de intercambio de datos. JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros.[3]

En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval() (Eval en varios lenguajes de programación, es una función que evalúa el contenido pasado como parámetro como si fuera una expresión.), sin embargo, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo!, Google, Mozilla, etc, que atienden a millones de usuarios) cuando la fuente de datos es explícitamente de fiar y donde no es importante el hecho de no disponer de procesamiento XSLT para manipular los datos en el cliente.[2]

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Los tipos de datos disponibles con JSON son:

- Números: Se permiten números negativos y opcionalmente pueden

[1]6000363@unimilitar.edu.co

[2]6000370@unimilitar.edu.co

[3]6000360@unimilitar.edu.co

contener parte fraccional separada por puntos. Ejemplo: 123.456

- Cadenas: Representan secuencias de cero o más caracteres. Se ponen entre doble comilla y se permiten cadenas de escape. Ejemplo: "Hola"
- Booleanos: Representan valores booleanos y pueden tener dos valores: true y false
- null: Representan el valor nulo.
- Array: Representa una lista ordenada de cero o más valores los cuales pueden ser de cualquier tipo. Los valores se separan por comas y el vector se mete entre corchetes. Ejemplo ["juan","pedro","jacinto"]
- Objetos: Son colecciones no ordenadas de pares de la forma <nombre>:<valor> separados por comas y puestas entre llaves. El nombre tiene que ser una cadena y entre ellas. El valor puede ser de cualquier tipo. Ejemplo: {"departamento":8,"nombredepto": "Ventas","director": "juan rodriguez","empleados":[{"nombre":"Pedro","apellido":"Fernandez"}, {"nombre":"Jacinto","apellido":"Benavente"} ]}

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    },
    {
      "name": "Madame Uppercut",
      "age": 39,
      "secretIdentity": "Jane Wilson",
      "powers": [
        "Million tonne punch",
        "Damage resistance",
        "Superhuman reflexes"
      ]
    },
    {
      "name": "Eternal Flame",
      "age": 1000000,
      "secretIdentity": "Unknown",
      "powers": [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
        "Interdimensional travel"
      ]
    }
  ]
}
```


***Figura 8.*** *Es posible incluir los mismos tipos de datos básicos dentro de un JSON que en un objeto estándar de JavaScript - cadenas, números, arreglos, booleanos, y otros literales de objeto. Esto permite construir una jerarquía de datos.*

Si se carga este objeto en un programa de JavaScript, traducido (parsed) en una

[1]6000363@unimilitar.edu.co

[2]6000370@unimilitar.edu.co

[3]6000360@unimilitar.edu.co

variable llamada superHeroes por ejemplo, se podría acceder a los datos que contiene utilizando la misma notación de punto/corchete.

```
superHeroes.homeTown
superHeroes['active']
```

***Figura 9.*** *Forma de cargar el objeto JSON en JavaScript.*

Para acceder a los datos que se encuentran más abajo en la jerarquía, simplemente se debe concatenar los nombres de las propiedades y los índices de arreglo requeridos. Por ejemplo, para acceder al tercer superpoder del segundo héroe registrado en la lista de miembros, se debería hacer esto:

```
superHeroes['members'][1]['powers'][2]
```

***Figura 10.*** *Forma de obtener información de un objeto JSON.*

1. Primero el nombre de la variable superHeroes.
2. Dentro de esta variable para acceder a la propiedad members utilizamos ["members"].
3. members contiene un arreglo poblado por objetos. Para acceder al segundo objeto dentro de este arreglo se utiliza [1].
4. Dentro de este objeto, para acceder a la propiedad powers utilizamos ["powers"].
5. Dentro de la propiedad powers existe un arreglo que contiene los superpoderes del héroe seleccionado. Para acceder al tercer superpoder se utiliza [2].

**Notas sobre JSON:**

- JSON es sólo un formato de datos contiene sólo propiedades, no métodos.
- JSON requiere usar comillas dobles para las cadenas y los nombres de propiedades. Las comillas simples no son válidas
- Una coma o dos puntos mal ubicados pueden producir que un archivo JSON no funcione. Se debe ser cuidadoso para validar cualquier dato que se utilizar (aunque los JSON generados por computador tienen menos probabilidades de tener errores, mientras el programa generador trabaje adecuadamente). Es posible validar JSON utilizando una aplicación como JSONLint.
- JSON Puede tomar la forma de cualquier tipo de datos que sea válido para ser incluido en un JSON, no sólo arreglos u objetos. Así, por ejemplo, una cadena o un número único podrían ser objetos JSON válidos.
- A diferencia del código JavaScript en que las propiedades del objeto pueden no estar entre comillas, en JSON, sólo las cadenas entre comillas pueden ser utilizadas como propiedades.

## 2. Aplicación

El proyecto que se realizará consiste en investigar y reconstruir los árboles genealógicos de una serie de televisión, evidenciando así los diferentes tipos de relaciones que se dan entre los personajes.

Se investigará acerca de la implementación de recursos gráficos como imágenes para que así la estética de la aplicación sea llamativa.

Se planea utilizar diferentes herramientas como eventos de mouse, botones y más

[1]6000363@unimilitar.edu.co

[2]6000370@unimilitar.edu.co

[3]6000360@unimilitar.edu.co

recursos que ofrece el lenguaje de programación HTML para lograr que la aplicación final tenga una interactividad adecuada. Lo anterior mencionado se planea lograr permitiendo que las diferentes acciones que realice el usuario sobre la aplicación tengan un efecto sobre los elementos gráficos del entorno, alternandolos para así hacer el proyecto final más dinámico e interesante. Este proceso se implementará al momento de mostrar los datos de cada miembro de la familia, las conexiones de cada personaje y de dar un vistazo general por el programa.

Para esto se utilizara la teoría de grafos, los cuales se construyen mediante objetos de tipo JSON, los cuales ayudan a construir la estructura del grafo definiendo la dirección y asociación de cada nodo; otra de las implementaciones de los objetos tipo JSON que se usarán en este proyecto es la facilidad de intercambio de datos, ya que estos se pueden convertir en tipo “String”, de este modo, al utilizar objetos para almacenar la información de cada personaje esta se puede manejar de una forma más eficiente dando como resultado una aplicación más óptima.

## 3. Bibliografía


[1]Universidad de Pamplona. (s.f.). Recuperado 2 marzo, 2020, de http://www.unipamplona.edu.co/unipamplona/portalIG/home_23/recursos/general/11072012/grafo3.pdf

Figura 1. Wikipedia, la enciclopedia libre. Recuperado 3 marzo, 2020, de https://es.wikipedia.org/wiki/Problema_del_camino_m%C3%A1s_corto

Figura 2. GRAFOS. (s.f.). Recuperado 3 marzo, 2020, de http://decsai.ugr.es/%7Ejfv/ed1/tedi/cdrom/docs/grafos.htm

Figura 3. GRAFOS. (s.f.). Recuperado 3 marzo, 2020, de http://decsai.ugr.es/%7Ejfv/ed1/tedi/cdrom/docs/grafos.htm

Figura 4. Contribuidores da Wikipédia. (2020, 16 febrero). um subgrafo de um grafo G é um grafo cujo conjunto de vértices é um subconjunto do conjunto de vértices G e o conjunto de arestas é um subconjunto do conjunto de arestas de G. Recuperado 3 marzo, 2020, de https://pt.wikipedia.org/wiki/Subgrafo

Figura 5. Udea. (s.f.). Representación de grafos. Matriz de incidencia. Matriz de adyacencia.. Recuperado 2 marzo, 2020, de http://docencia.udea.edu.co/regionalizacion/teoriaderedes/informaci%F3n/C1_RepresentacionMatrices.pdf

Figura 6. Jhony Castro, J. C. (2018, 15 mayo). Matrices de Adyacencia para representación de Grafos Dirigidos. Recuperado 3 marzo, 2020, de http://jhonycastromoran.blogspot.com/2010/05/matrices-de-adyacencia-para.html

Figura 7. Bianco, S. B. (2016, 1 junio). Research Gate. Recuperado 3 marzo, 2020, de https://www.researchgate.net/figure/Grafo-



[1]6000363@unimilitar.edu.co

[2]6000370@unimilitar.edu.co

[3]6000360@unimilitar.edu.co

dirigido-con-su-representacion-como-lista-de-adyacencia_fig14_309278789

[2]Trabajando con JSON. (2019, 2 septiembre). Recuperado 3 marzo, 2020, de https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON

Figura 8.Trabajando con JSON. (2019, 2 septiembre). Recuperado 3 marzo, 2020, de https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON

Figura 9.Trabajando con JSON. (2019, 2 septiembre). Recuperado 3 marzo, 2020, de https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON

Figura 10.Trabajando con JSON. (2019, 2 septiembre). Recuperado 3 marzo, 2020, de https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON

[3]JSON. (s.f.). Recuperado 3 marzo, 2020, de https://www.json.org/json-es.html

[4]JSON. (s.f.). Recuperado 3 marzo, 2020, https://www.w3schools.com/js/js_json_syntax.asp

Analítica y visualización de datos con D3.js (Parte I). (2017, 6 febrero). Recuperado 4 marzo, 2020, de https://impactotecno.wordpress.com/2017/02/06/analitica-y-visualizacion-de-datos-con-javascript/

Neo4J: trabajando con grafos. (2019, 19 noviembre). Recuperado 4 marzo, 2020, de https://www.paradigmadigital.com/dev/neo4j-trabajando-grafos/

[1]6000363@unimilitar.edu.co

[2]6000370@unimilitar.edu.co

[3]6000360@unimilitar.edu.co