Tarea 5 Coloreado de Grafos

Alumno	Fecha
Jiménez Zamudio Vinicio Armando	10/16/2021
Lozano Lozano Alejandro	

Objetivo

Programar el algoritmo de coloreado de vértices para grafos. La implementación descrita en este documento se encuentra en

https://github.com/aguilavajz/iteso-coloreografos

Algortitmo y Requerimientos

Entrada

- Grafo G (Vértices y Aristas)
- Número de colores.
- Nombre de colores asignado el orden prioridad

Salida

- Coloreado Posible/Coloreado no posible.
- En caso de ser posible indicar la asignación de vertices con los colores asignados. En caso de que no sea posible indicar el conflicto.
- Presentar salida gráfica opcional.

Funcionamiento

1. Ordenar los vértices de G por grado.
2. Para los nodos de grado igual, obtener su grado de error.

   ```
   grado de error =  Grado + cantidad de nodos adyacentes de grado igual o superior.  Se ordenan de acuerdo al grado de error.
   ```

3. Se considera el orden de los colores a utilizar, el número de colores y orden se determinal al inicio del programa.
4. Se colorean los nodos considerando el orden de prioridades de color y el acomodo dependiendo el gradro de error del algoritmo.
5. Termina el algoritmo cuando todos los vértices son coloreados o cuando falla para colorear algún vértice debido a la falta de colores.

Implementación

Se optó por empezar el algoritmo en python sin ninguna interfaz gráfica, esto para verificar que la implementación y lógica en el código sea correcta.

Una vez que la primera implementación funcionó, se usaron diferentes herramientas y servicios para generar una interfaz gráfica web. A continuación se enlistan los paquetes de SW, servicios y herramientas usadas:

- UI: Oracle APEX 21.1.3
- Backend: PL/SQL
- Paquetes desarrollados con funciones y procedimientos para el coloreo
- Plugin D3 Force Network Chart (3.1.0) de Oracle Apex
- Always Free Tier Oracle Cloud
- Web Server: Nginx
- Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production Version 19.13.0.1.

Código Python

El siguiente codigo es la parte más importante del algoritmo una vez que los nodos están ordenados.

```
def colorGraph(listOfColoredNodes, listOfColors, listOfNodes):
    for v in listOfNodes:
        for c in listOfColors:
            exists = False

            for n in v.neighbors:
                if c == listOfColoredNodes[n]:
                    exists = True
                    break

            if exists == False:
                listOfColoredNodes[v.value] = c
                break

        if exists == True: return False

    return True
```

Código PL/SQL

El siguiente código es el utilizado en la versión final . De igual manera solo muestra la parte esencial del algoritmo.

```
procedure colorear(
        p_grafo     grafos.id%type,
        p_colores   number,
        p_salida    in out varchar2
    ) is
        l_exists        boolean;
        l_color_vecino  number;
    begin
```

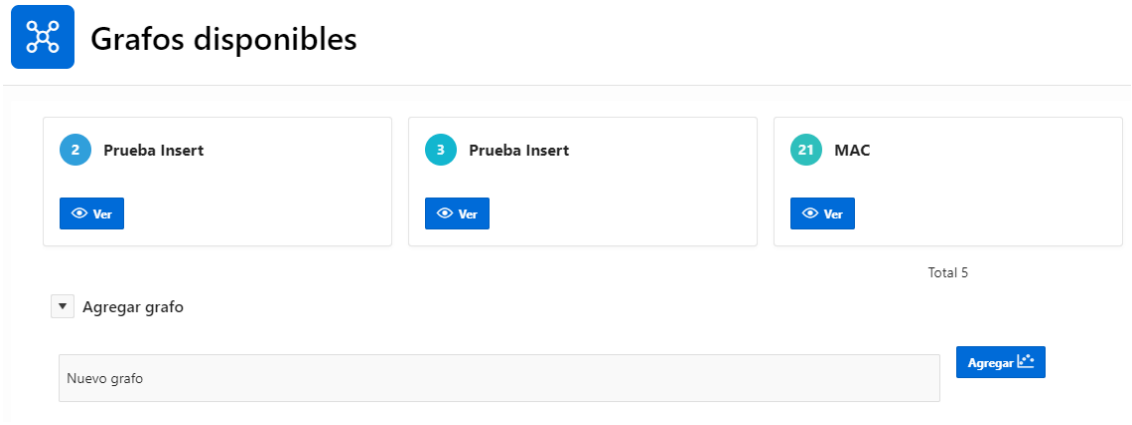
```
        update nodos
        set color = null
        where grafo_id = p_grafo;

        for c in (SELECT * FROM NODOS
        WHERE GRAFO_ID = p_grafo
        order by grado desc, grado_error desc, nombre) loop

            apex_debug.log_message(
                c.nombre,
                true,
                1
            );

            for d in (select * from grafo_colores
                      where rownum <= p_colores
                      order by id) loop

                apex_debug.log_message(
                d.nombre,
                true,
                1
            );
                l_exists    := false;

                for e in (select column_value
                          from
table(apex_string.split(obtener_vecinos(p_grafo,c.id),':')))
                loop
                    apex_debug.log_message(
                        e.column_value,
                        true,
                1
                    );
                    l_color_vecino  := obtener_color_nodo(e.column_value);
                    apex_debug.log_message(
                        obtener_color_name(l_color_vecino),
                        true,
                1
                    );
                    if d.id = l_color_vecino then
                        l_exists := true;
                        exit;
                    end if;

                end loop;

                if not l_exists then
                    update nodos
                    set color = d.id
                    where grafo_id = p_grafo
                    and id = c.id;
                    exit;
                end if;

            end loop;

            if l_exists then
```

```
                p_salida := 'No se pudo colorear con '||p_colores || ' colores.
El conflicto esta en el nodo '|| c.nombre;
                exit;
            end if;

        end loop;

        if not l_exists then
            p_salida := 'Grafo coloreado correctamente.';
        end if;

    end colorear;
```

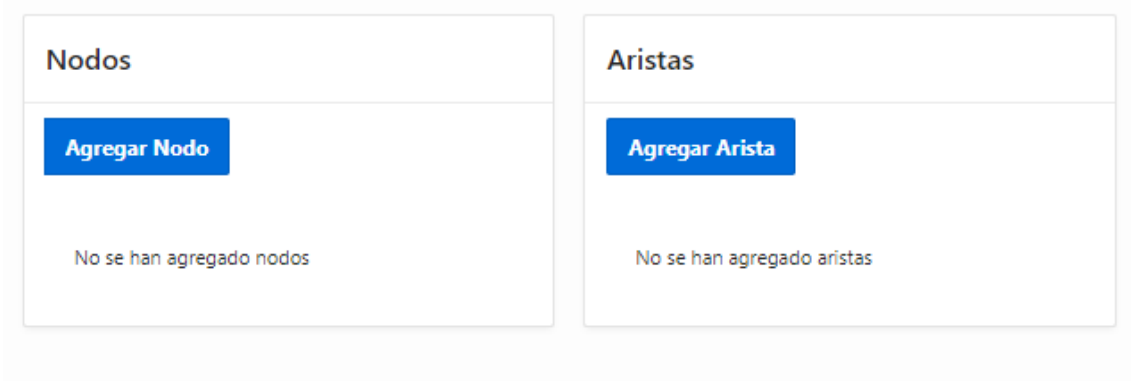
Uso de la aplicacion

Estos son los pasos necesarios para hacer uso de la aplicación

1. Ingresa a la siguiente URL:https://equixoft.ga/ords/f?p=101:LOGIN:9019277304657:::::
2. Usa grafos ya existentes o crea uno dando click en **Agregar grafo** y asígnale un nombre:

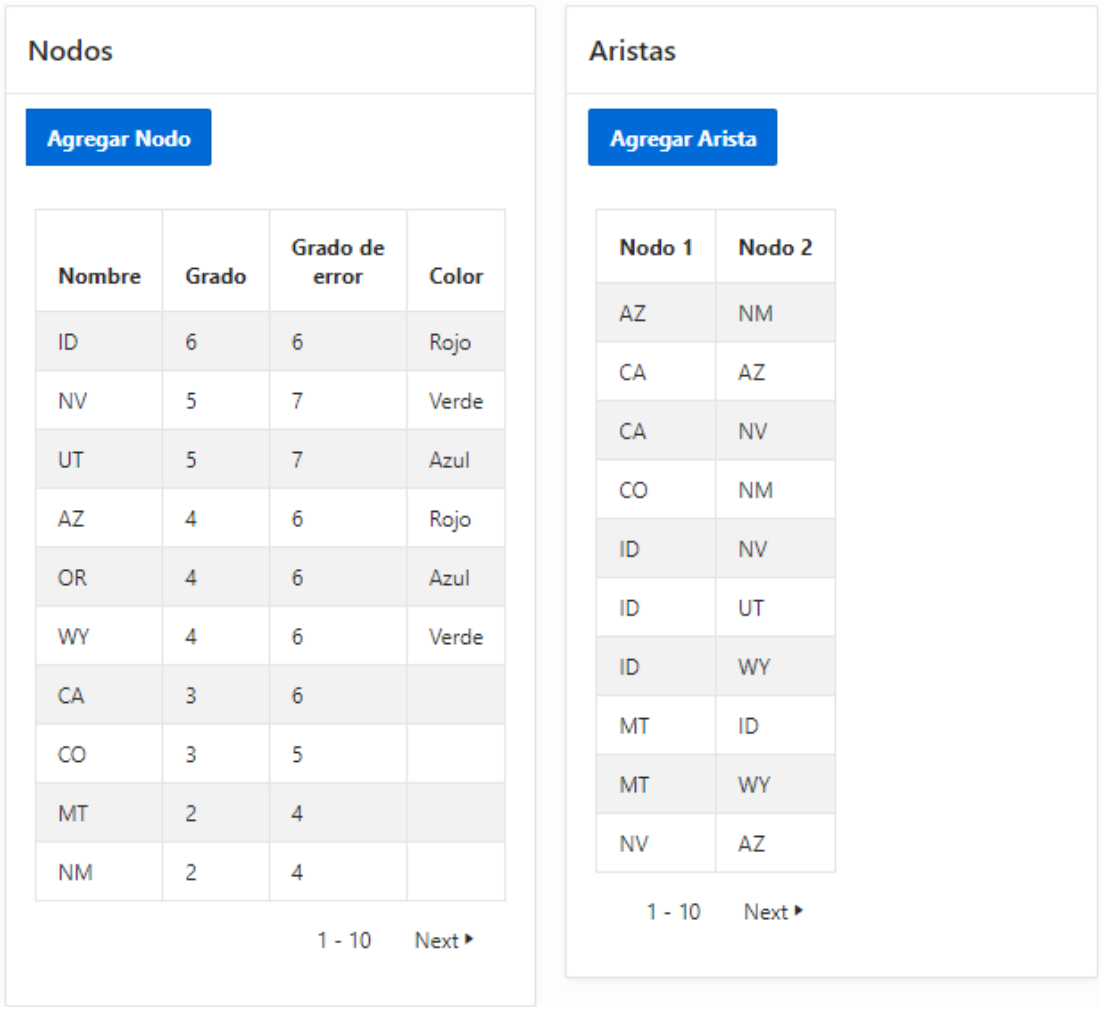
   

   Una vez creado selecciona tu grafo.
3. Agrega Nodos y Aristas para crear tu grafo.

   Conforme se vayan agregando nodos, y aristas, estos se irán mostrando en una tabla

ordenados de forma descendente usando como criterio el grado y el error del nodo.

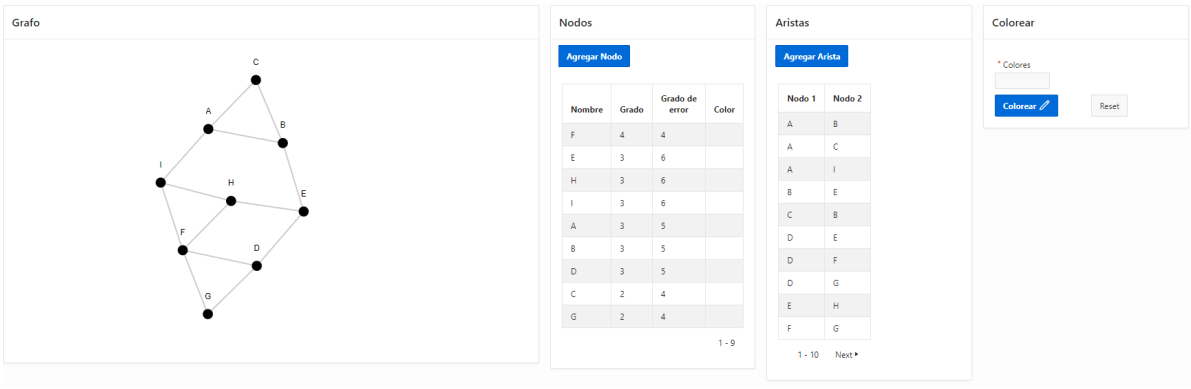


Pruebas

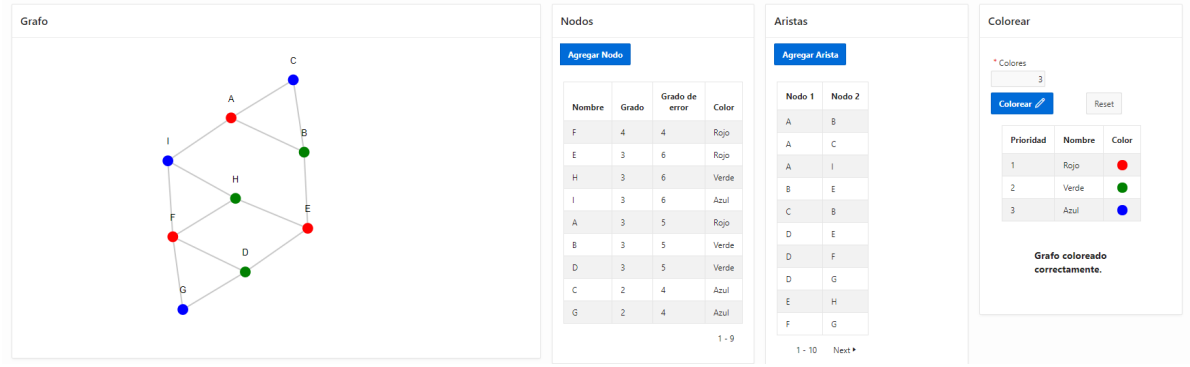
El ususario puede hacer pruebas en la siguiente URL: https://equixoft.ga/ords/f?p=101

Prueba #1

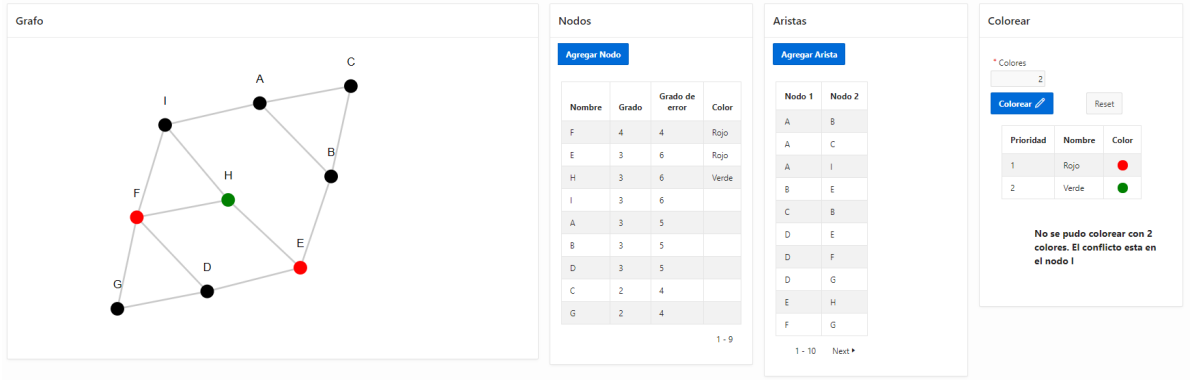
Grafo Inicial



Salida con 3 Colores (Coloreado Exitoso)

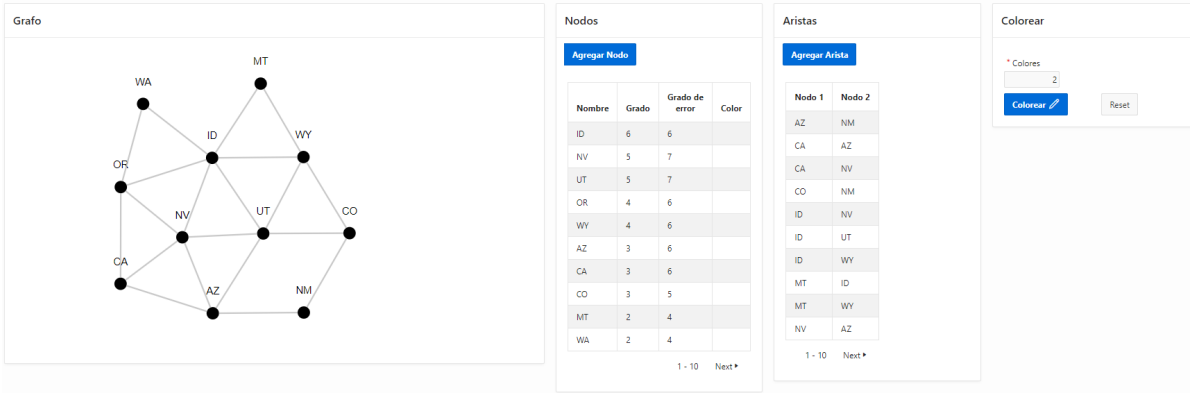


Salida con 2 Colores (Coloreado No Exitoso. Conflicto en I)

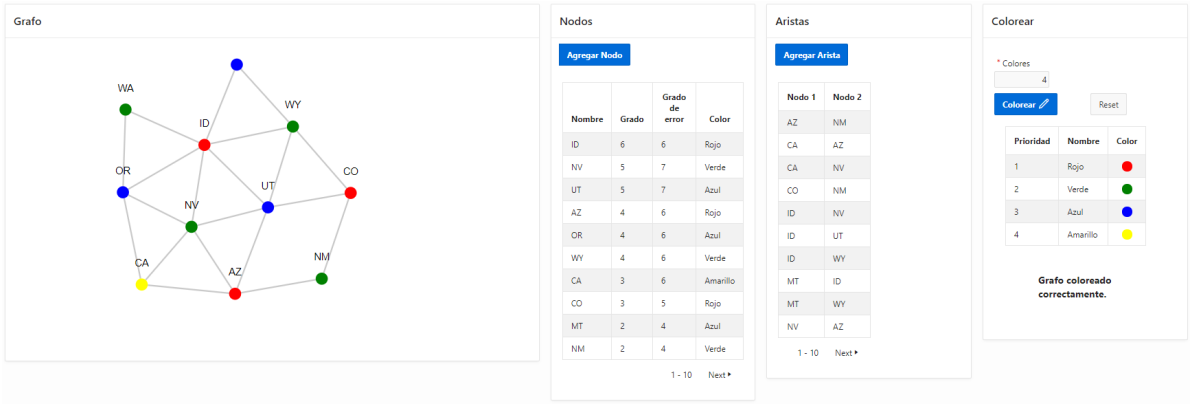


Prueba #2

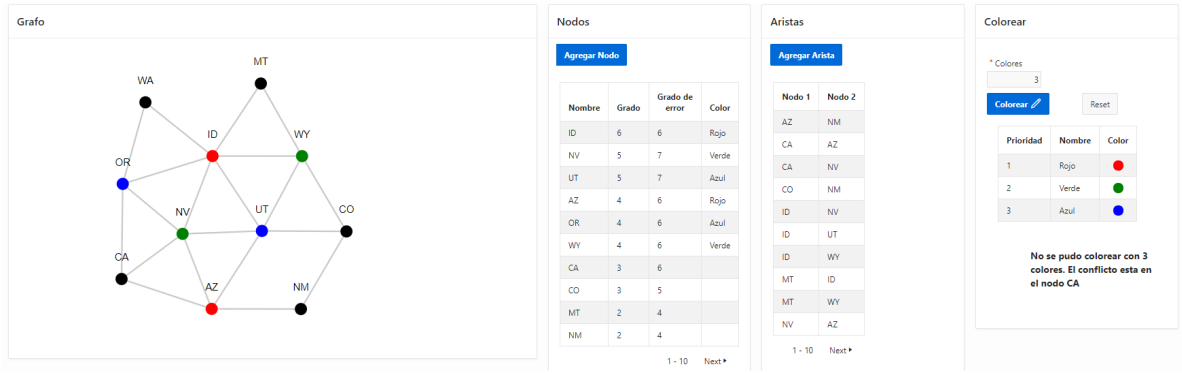
Grafo Inicial



Salida con 4 Colores (Coloreado Exitoso)

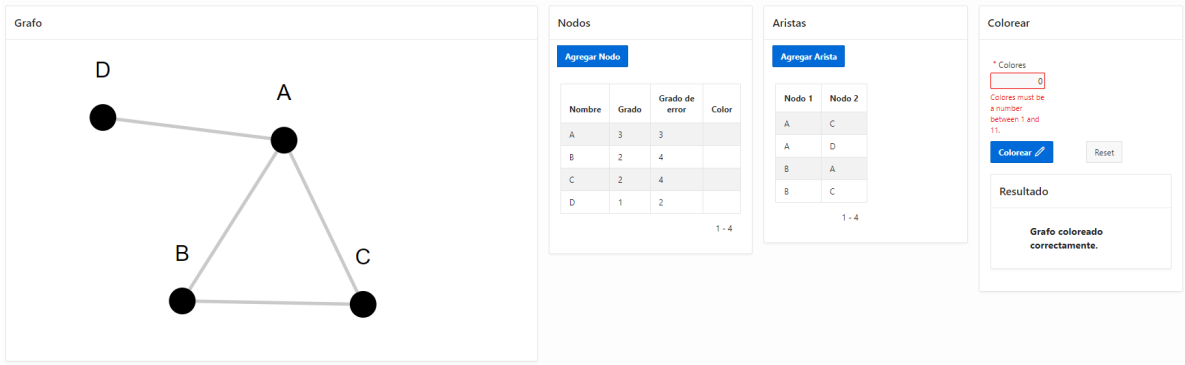


Salida con 3 Colores (Coloreado No Existoso. Conflicto en CA)

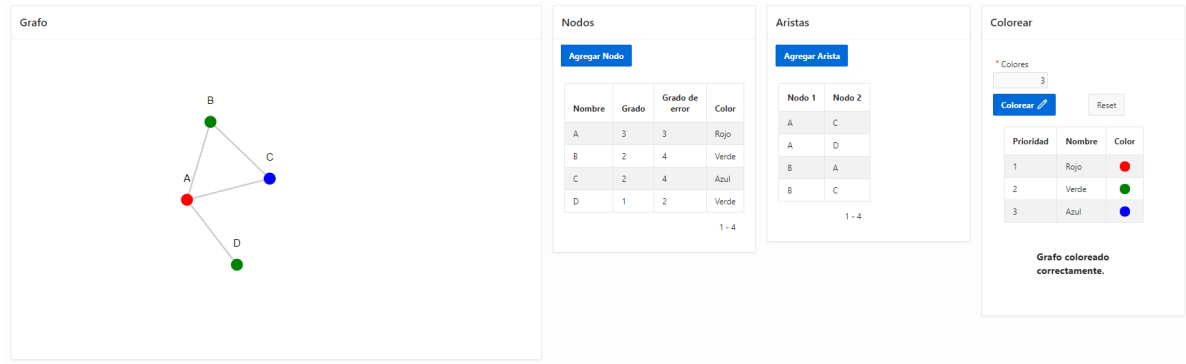


Prueba #3

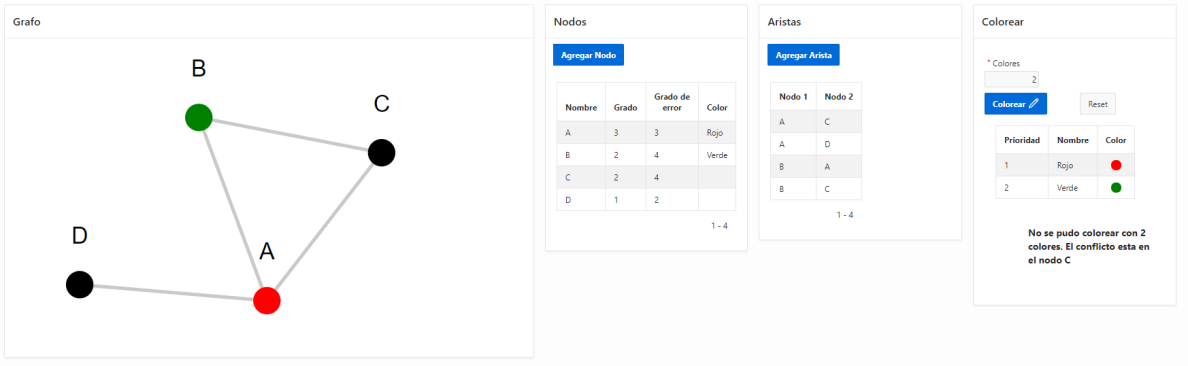
Grafo Inicial



Salida con 3 Colores (Coloreado Exitoso)



Salida con 2 Colores (Coloreado No Exitoso. Conflicto en C)



Conclusiones

Como se demostró. la implementación del este algoritmo es simple y puede ser usado en aplicaciones de escritorio hasta web. Cabe mencionar que el algoritmo es un algoritmo voraz, los cuales se caracterizan por escoger un local óptimo en cada paso con la esperanza de llegar a una solución óptima general. La implementación mostrada en este reporte hace uso de 3 ciclos anidados (nodos, colores y vecinos) por lo que un análisis apriori determinaría una complejidad de O(n^3). Se requiere indagar si es posible reducir dicha complejidad. A pesar de su fácil implementación, investigaciones muestran que dicho algoritmo puede ser útil en aplicaciones como : solución de Sudokus, Planeación de Eventos, Agendar tareas etc.

Hay limitaciones en la aplicación:

- No se pueden agregar nosods con el mismo nombre
- No se puede agregar la arista más de una vez
- El orden/prioridad de los colores está definido con un máximo de 11 colores.

Bibliografía

Algoritmo Voraz https://es.wikipedia.org/wiki/Algoritmo_voraz

Aplications of Graph Colouring https://www.youtube.com/watch?v=y4RAYQjKb5Y&t=9s

D3 Force Network Chart (3.1.0) https://github.com/ogobrecht/d3-force-apex-plugin

APEX: https://www.oracle.com/database/technologies/appdev/apex.html

Oracle Cloud: https://www.oracle.com/cloud/