COLOREO DE GRAFOS

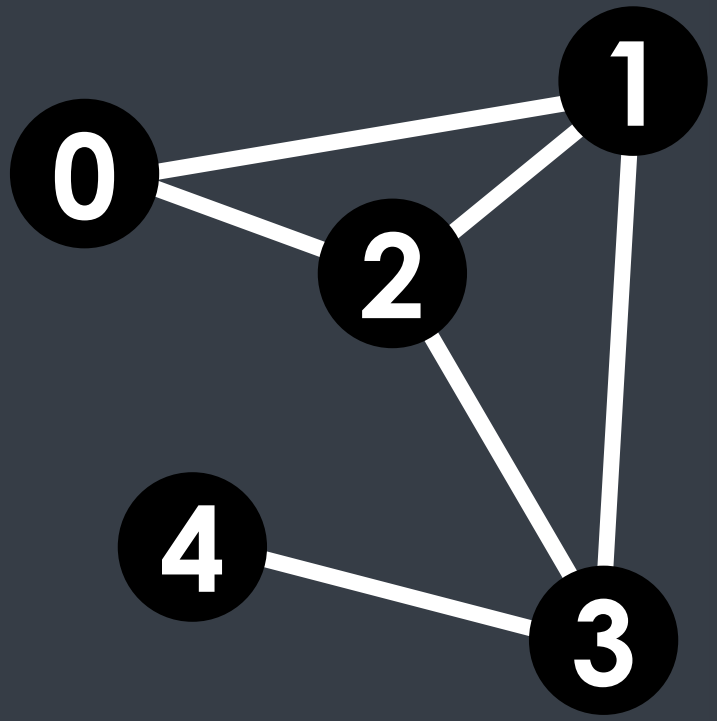
LARREATEGUI CASTRO, ANGEL VIDAL

• Algoritmo Greedy para la coloración de grafos

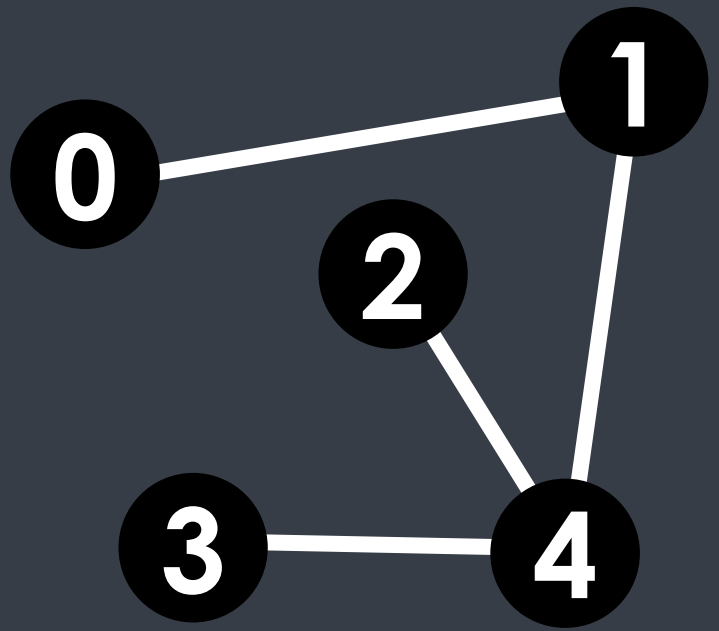
- No existe un algoritmo eficiente para poder colorear un grafo con un número mínimo de colores ya que el problema es un problema **NP-Completo** conocido.
- Existen algoritmos aproximados para poder resolver este problema.
- El algoritmo greedy de coloración para asignar colores no garantiza usar la mínima cantidad de colores pero si garantiza usar un número máximo de colores.
- Este algoritmo jamás usa más de "D+1" colores en donde "D" es el grado máximo de un vértice en el grafo a colorear.

DESCRIPCIÓN DEL PROBLEMA

GRAFO 1



GRAFO 2



• ALGORITMO GREEDY PARA LA COLORACIÓN DE GRAFOS

- Colorea el primer vértice con el primer color.
- Se repite lo mismo para los siguientes “V-1” vértices.
- Consideremos un vértice seleccionado y lo coloreamos con el color cuyo número es el más bajo y que no se ha usado para colorear algún vértice adyacente a este. Si todos los colores usados aparecen en los vértices adyacentes a “V” entonces se le asignará un nuevo color

```cpp
// Programa en C++ que implementa el Algoritmo Greedy en la Coloracion de Grafos
#include <iostream>
#include <list>
using namespace std;

// Clase que representa a un grafo no dirigido
class Grafo
{
    int V;    // Numero de vertices
    list<int> *adj;    // Un array dinamico de listas de adyacencia
public:
    // Constructor y destructor
    Grafo(int V)   { this->V = V; adj = new list<int>[V]; }
    ~Grafo()       { delete [] adj; }

    // Funcion que agrega una arista al grafo
    void agregarArista(int v, int w);

    // Imprime colores Greedy de los vertices
    void greedyColoracion();
};

void Grafo::agregarArista(int v, int w)
{
    adj[v].push_back(w);
    adj[w].push_back(v);  // Nota: El grafo no está dirigido
}
```

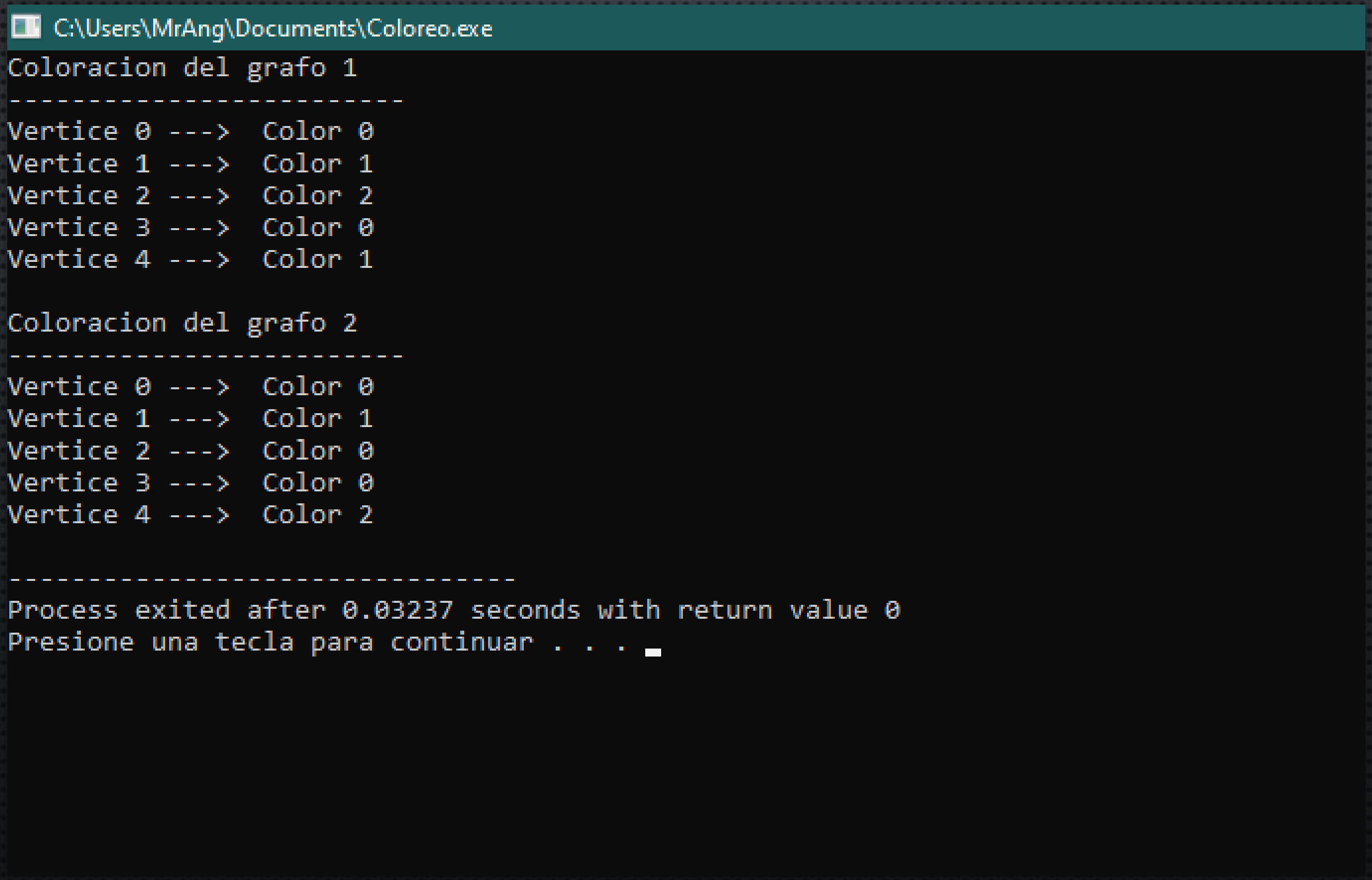
```
29  // Asigna colores (comenzando desde 0) a todos los vertices e imprime la asignacion de colores
30  void Grafo::greedyColoracion()
31  {
32      int resultado[V];
33
34      // Asigna el primer color al primer vertice
35      resultado[0]  = 0;
36
37      // Inicializar los V-1 vertices restantes como no asignados
38      for (int u = 1; u < V; u++)
39          resultado[u] = -1;  // Ningun color esta asignado a V
40
41      // Un arreglo temporal para almacenar los colores disponibles. El verdadero valor
42      // de available[cr] significaria que el color "cr" se asigna a uno de sus vertices adyacentes
43      bool disponible[V];
44      for (int cr = 0; cr < V; cr++)
45          disponible[cr] = false;
46
47      // Asignar colores a los V-1 vertices restantes
48      for (int u = 1; u < V; u++)
49      {
50          // Procesa todos los vertices adyacentes y marca sus colores como no disponibles
51          list<int>::iterator i;
52          for (i = adj[u].begin(); i != adj[u].end(); ++i)
53              if (resultado[*i] != -1)
54                  disponible[resultado[*i]] = true;
```

```
55
56          // Encuentra el primer color disponible
57          int cr;
58          for (cr = 0; cr < V; cr++)
59              if (disponible[cr] == false)
60                  break;
61
62          resultado[u] = cr; // Asigna el color encontrado
63
64          // Resetea los valores a Falso para la siguiente iteracion
65          for (i = adj[u].begin(); i != adj[u].end(); ++i)
66              if (resultado[*i] != -1)
67                  disponible[resultado[*i]] = false;
68      }
69
70      // Imprime el resultado
71      for (int u = 0; u < V; u++)
72          cout << "Vertice " << u << " --->  Color "
73               << resultado[u] << endl;
74  }
```

```
76  int main()
77  {
78      Grafo g1(5);
79      g1.agregarArista(0, 1);
80      g1.agregarArista(0, 2);
81      g1.agregarArista(1, 2);
82      g1.agregarArista(1, 3);
83      g1.agregarArista(2, 3);
84      g1.agregarArista(3, 4);
85      cout << "Coloracion del grafo 1 \n------------------------ \n";
86      g1.greedyColoracion();
87
88      Grafo g2(5);
89      g2.agregarArista(0, 1);
90      g2.agregarArista(1, 4);
91      g2.agregarArista(2, 4);
92      g2.agregarArista(4, 3);
93      cout << "\nColoracion del grafo 2 \n------------------------ \n";
94      g2.greedyColoracion();
95
96      return 0;
97  }
```

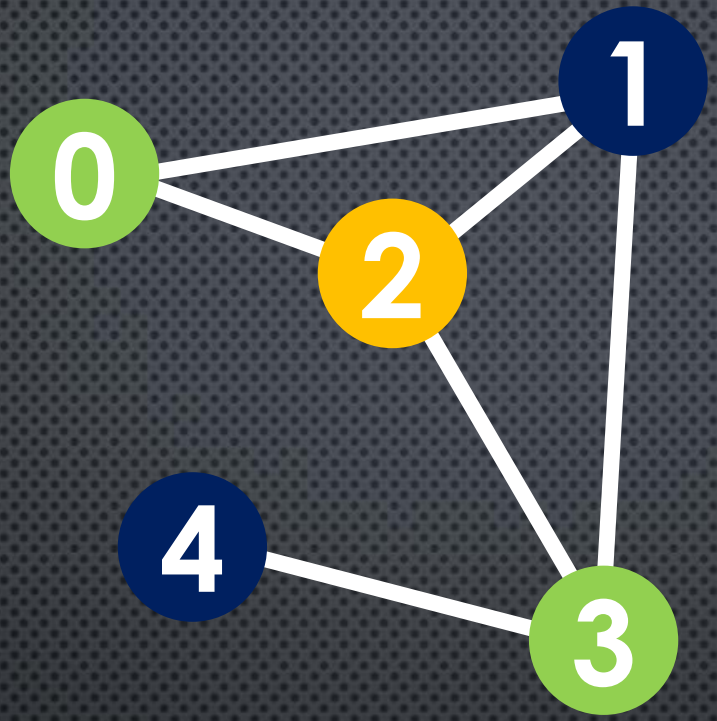
GRAFO 1

GRAFO 2

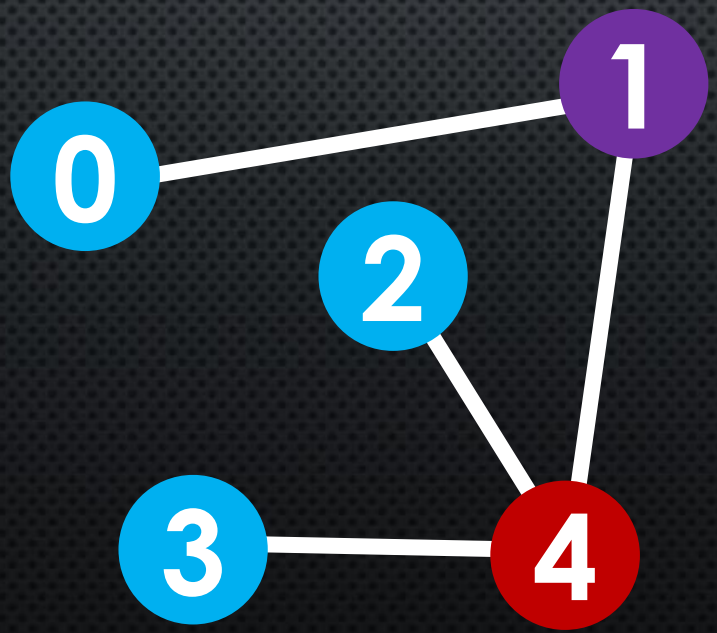

C:\Users\MrAng\Documents\Coloreo.exe
Coloracion del grafo 1
Vertice 0 ---> Color 0
Vertice 1 ---> Color 1
Vertice 2 ---> Color 2
Vertice 3 ---> Color 0
Vertice 4 ---> Color 1
Coloracion del grafo 2
Vertice 0 ---> Color 0
Vertice 1 ---> Color 1
Vertice 2 ---> Color 0
Vertice 3 ---> Color 0
Vertice 4 ---> Color 2
Process exited after 0.03237 seconds with return value 0
Presione una tecla para continuar . . .

COLOREO DE LOS GRAFOS 1 Y 2

GRAFO 1



GRAFO 2



Algoritmo Greedy para la coloración de grafos

- La complejidad del algoritmo es $\mathbf{O(V^2 + E)}$ en el peor caso.
- Como ya lo habíamos mencionado, este algoritmo no siempre usa un número mínimo de colores.
- La cantidad de colores usados a veces depende del orden en el que los vértices son procesados.
- Hay diferentes formas de encontrar un algoritmo para este problema, uno de ellos es el algoritmo de Welsh-Powell, el cual considera los vértices en orden descendente de grados.


MUCHAS
GRACIAS