Pseudocódigo: Coloreo de Grafos con Algoritmo Greedy

Algoritmo Greedy

24 de septiembre de 2025

1. Descripción del Problema

El problema de coloreo de grafos consiste en asignar colores a los vértices de un grafo de manera que ningún par de vértices adyacentes compartan el mismo color. El objetivo es encontrar una coloración válida usando el menor número posible de colores.

El algoritmo greedy resuelve este problema asignando a cada vértice el menor color disponible que no esté siendo usado por sus vértices adyacentes. Aunque no garantiza usar el número mínimo de colores (número cromático), proporciona una coloración válida de manera eficiente.

2. Pseudocódigo

3. Explicación del Algoritmo

3.1. Estrategia Greedy

El algoritmo greedy para coloreo de grafos utiliza la siguiente estrategia:

- **Orden fijo:** Procesa los vértices en un orden predeterminado (0, 1, 2, ..., V-1)
- **Selección greedy:** Para cada vértice, asigna el menor color disponible que no esté siendo usado por sus vecinos
- **Disponibilidad de colores:** Mantiene un arreglo que indica qué colores están siendo usados por vértices adyacentes

La elección greedy de usar el menor color disponible es una heurística que, aunque no garantiza optimalidad, tiende a producir buenas coloraciones en la práctica.

3.2. Proceso de Asignación

Para cada vértice u (excepto el primero), el algoritmo:

1. **Marca colores no disponibles:** Recorre todos los vértices adyacentes a u y marca sus colores como no disponibles
2. **Busca primer color disponible:** Encuentra el menor color (índice) que no esté marcado como no disponible
3. **Asigna el color:** Asigna el color encontrado al vértice u
4. **Resetea disponibilidad:** Limpia las marcas de disponibilidad para el siguiente vértice

Este proceso garantiza que cada vértice reciba un color válido (diferente al de sus vecinos).

Algorithm 1 Coloreo de Grafos con Algoritmo Greedy

Require: Un grafo G representado como lista de adyacencia con V vértices
Ensure: Una asignación válida de colores para todos los vértices

```
result ← arreglo de tamaño V inicializado en -1
result[0] ← 0 {Asignar primer color al primer vértice}
available ← arreglo booleano de tamaño V inicializado en false
{Asignar colores a los vértices restantes}
for u = 1 to V − 1 do
    {Marcar colores de vértices adyacentes como no disponibles}
    for i = 0 to |adj[u]| − 1 do
        if result[adj[u][i]] ≠ −1 then
            available[result[adj[u][i]]] ← true
        end if
    end for
    {Encontrar el primer color disponible}
    cr ← 0
    while cr < V do
        if available[cr] = false then
            break
        end if
        cr ← cr + 1
    end while
    {Asignar el color encontrado}
    result[u] ← cr
    {Resetear valores para la siguiente iteración}
    for i = 0 to |adj[u]| − 1 do
        if result[adj[u][i]] ≠ −1 then
            available[result[adj[u][i]]] ← false
        end if
    end for
end for
return result
```

3.3. Complejidad

- **Tiempo:** $O(V + E)$ donde V es el número de vértices y E el número de aristas
  - $O(V)$ para procesar cada vértice
  - $O(E)$ para procesar todas las aristas al verificar adyacencias
  - El bucle interno para encontrar el primer color disponible es $O(V)$ en el peor caso
- **Espacio:** $O(V)$ para almacenar el resultado y el arreglo de disponibilidad

La complejidad total es $O(V^2 + E)$ en el peor caso, pero típicamente es mucho mejor en grafos dispersos.

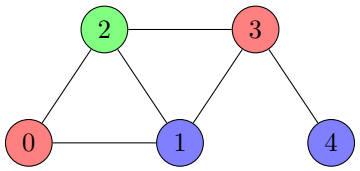
4. Resolución Paso a Paso

4.1. Ejemplo 1: Grafo 1 con 5 vértices

Grafo de entrada:

- Vértice 0: conectado a [1, 2]
- Vértice 1: conectado a [0, 2, 3]
- Vértice 2: conectado a [0, 1, 3]
- Vértice 3: conectado a [1, 2, 4]
- Vértice 4: conectado a [3]

Representación visual del grafo:



Paso 1: Inicialización

- $result = [-1, -1, -1, -1, -1]$
- $result[0] = 0$ (color rojo)
- Vértice 0 coloreado con color 0

Paso 2: Procesar vértice 1

- Vértices adyacentes a 1: [0, 2, 3]
- Vértice 0 tiene color 0 $\Rightarrow$ $available[0] = true$
- Vértices 2 y 3 no tienen color asignado
- Primer color disponible: color 1
- $result[1] = 1$ (color azul)
- Resetear $available[0] = false$

Paso 3: Procesar vértice 2

- Vértices adyacentes a 2: [0, 1, 3]

- Vértice 0 tiene color 0 $\Rightarrow$ $available[0] = true$
- Vértice 1 tiene color 1 $\Rightarrow$ $available[1] = true$
- Vértice 3 no tiene color asignado
- Primer color disponible: color 2
- $result[2] = 2$ (color verde)
- Resetear $available[0] = false$, $available[1] = false$

Paso 4: Procesar vértice 3

- Vértices adyacentes a 3: [1, 2, 4]
- Vértice 1 tiene color 1 $\Rightarrow$ $available[1] = true$
- Vértice 2 tiene color 2 $\Rightarrow$ $available[2] = true$
- Vértice 4 no tiene color asignado
- Primer color disponible: color 0
- $result[3] = 0$ (color rojo)
- Resetear $available[1] = false$, $available[2] = false$

Paso 5: Procesar vértice 4

- Vértices adyacentes a 4: [3]
- Vértice 3 tiene color 0 $\Rightarrow$ $available[0] = true$
- Primer color disponible: color 1
- $result[4] = 1$ (color azul)
- Resetear $available[0] = false$

Resultado Final Grafo 1:

- Vértice 0 $\Rightarrow$ Color 0 (rojo)
- Vértice 1 $\Rightarrow$ Color 1 (azul)
- Vértice 2 $\Rightarrow$ Color 2 (verde)
- Vértice 3 $\Rightarrow$ Color 0 (rojo)
- Vértice 4 $\Rightarrow$ Color 1 (azul)

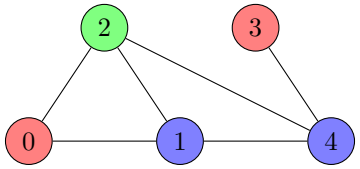
Número de colores usados: 3

4.2. Ejemplo 2: Grafo 2 con 5 vértices

Grafo de entrada:

- Vértice 0: conectado a [1, 2]
- Vértice 1: conectado a [0, 2, 4]
- Vértice 2: conectado a [0, 1, 4]
- Vértice 3: conectado a [4]
- Vértice 4: conectado a [1, 2, 3]

Representación visual del grafo:



Proceso de coloreo:
Paso 1: $result[0] = 0$ (color rojo)
Paso 2: Vértice 1

- Adyacente a 0 (color 0) $\Rightarrow available[0] = true$
- Primer color disponible: color 1
- $result[1] = 1$ (color azul)

Paso 3: Vértice 2

- Adyacente a 0 (color 0) y 1 (color 1) $\Rightarrow available[0] = true$, $available[1] = true$
- Primer color disponible: color 2
- $result[2] = 2$ (color verde)

Paso 4: Vértice 3

- Adyacente a 4 (sin color) $\Rightarrow$ No hay restricciones
- Primer color disponible: color 0
- $result[3] = 0$ (color rojo)

Paso 5: Vértice 4

- Adyacente a 1 (color 1), 2 (color 2), 3 (color 0) $\Rightarrow available[0] = true, available[1] = true, available[2] = true$
- Primer color disponible: color 3
- $result[4] = 3$ (color amarillo)

Resultado Final Grafo 2:

- Vértice 0 $\Rightarrow$ Color 0 (rojo)
- Vértice 1 $\Rightarrow$ Color 1 (azul)
- Vértice 2 $\Rightarrow$ Color 2 (verde)
- Vértice 3 $\Rightarrow$ Color 0 (rojo)
- Vértice 4 $\Rightarrow$ Color 3 (amarillo)

Número de colores usados: 4

5. Análisis del Algoritmo

5.1. Ventajas del Algoritmo Greedy

- **Eficiencia:** Complejidad lineal en el número de vértices y aristas
- **Simplicidad:** Fácil de implementar y entender
- **Determinismo:** Produce siempre el mismo resultado para el mismo grafo
- **Coloración válida:** Garantiza encontrar una coloración válida

5.2. Limitaciones

- **No optimalidad:** No garantiza usar el número mínimo de colores
- **Dependencia del orden:** El resultado depende del orden de procesamiento de vértices
- **Coloración subóptima:** Puede usar más colores de los necesarios

Ejemplo de suboptimalidad: Para un grafo bipartito (que requiere solo 2 colores), el algoritmo greedy podría usar hasta V colores dependiendo del orden de procesamiento.

5.3. Comparación con Fuerza Bruta

- **Complejidad:**
  - Greedy: $O(V^2 + E)$
  - Fuerza Bruta: $O(k^V \cdot V \cdot E)$ donde k es el número máximo de colores
- **Ventajas del Greedy:**
  - Mucho más eficiente para grafos grandes
  - Tiempo polinomial vs. exponencial
  - Práctico para aplicaciones reales

6. Conclusiones

El algoritmo greedy para coloreo de grafos es una solución eficiente y práctica que garantiza encontrar una coloración válida en tiempo polinomial. Aunque no garantiza optimalidad, su simplicidad y eficiencia lo hacen adecuado para muchas aplicaciones prácticas donde se requiere una coloración válida rápidamente.

Para casos donde se necesita el número mínimo de colores, se requieren algoritmos más sofisticados o técnicas de optimización, pero el algoritmo greedy proporciona una excelente base y punto de partida para el análisis de problemas de coloreo de grafos.