ReporteComplejidad

Víctor Federico Torres Trejo
Alejandro Maldonado Vázquez

29 de febrero de 2024

1 Descripcion del problema de optimizacion

Ejemplar: Una grafica $G = (V, E)$ donde V es el conjunto de vertices y E el de aristas no dirigidas y un entero k
Pregunta: Encontrar una funcion de coloracion $f : V \to \{1, \ldots, k\}$ tal que para $(u, v) \in E$, donde u y v son vertices adyacentes, $f(u) \neq f(v)$, minimizar el valor de k, es decir usar el menor numero de colores posibles.

2 Descripcion del problema de decision

Ejemplar: Una grafica $G = (V, E)$ donde V es el conjunto de vertices y E el de aristas no dirigidas y un entero k
Pregunta: ¿Existe una función de coloración $f : V \to \{1, \ldots, k\}$ tal que para cada $(u, v) \in E$, donde u y v son vertices adyacentes, $f(u) \neq f(v)$?

3 Esquema de Codificacion

Para la codificacion, solicitaremos que la entrada sea de la forma:
k
n
$(r, v), (v, u), (u, t), \ldots$
Donde $k >= 2, n >= 1$ y donde $r, v, u, t \in V$ lo cual quiere decir que el numero que los denota tiene que ser menor a n, escrito con tuplas delimitadas por parentesis, cada tupla se separa por una coma.

4 Codigo:

4.1 Leer entrada

Para leer la entrada vamos a definir una clase auxiliar:

```
class Grafica:
    def __init__(self, n:int, edges:list) -> None:
        self.n = n
        self.lista_adyacencias = [[] for _ in range(n + 1)]
        for edge in edges:
            v1, v2 = edge
            self.lista_adyacencias[v1].append(v2)
            self.lista_adyacencias[v2].append(v1)
        return
    def get_neighbors(self, v)->list:
        return self.lista_adyacencias[v] if v <= self.n else []
```

Donde al inicializar definimos la lista de adyacencia para cada vertice y lo metemos, es decir sean los vertices adyacentes u, v, entonces $v \in L[u], u \in L[v]$. Ademas definimos una funcion que nos devuelve los vecinos de un vertice valido

4.2 k-coloracion

Primero definimos que k-coloracion con $k > 2$, es un problema np

```
if k > 2:
        print(f"El problema de la {k}-coloracion es NP")
        sys.exit()
```

Si $k = 2$ entonces procedemos a hacer la 2 coloracion, vamos a analizar el algoritmo por parte : Inicializamos la cola con un vertice al azar y establecemos un vector de colores, todos inicalizados en 0, usaremos el 0 para denotar 'sin color'

```
queue = [random.randint(1, g.n)]
colors = [0] * (g.n + 1)
```

Entramos a un ciclo con la condicion que nuestra cola no sea vacia, entonces extraemos el primer elemento y obtenemos a todos sus vecinos, asignamos el color contrario dependiendo en cual estamos, usaremos 1 para determinar un color y 2 para el otro color.
Recorremos para cada vecino, si no tiene color asignado le asignamos uno, si queremos asignar un color distinto al color ya asignado, entonces estamos ante una grafica que no es 2-coloreable.

```
while(queue.__len__() > 0):
        v = queue.pop(0)
        neighbors = g.get_neighbors(v)
        color = 1 if colors[v] == 2 else 2
        for neighbor in neighbors:
            if colors[neighbor] == 0:
                colors[neighbor] = color
                queue.append(neighbor)
```

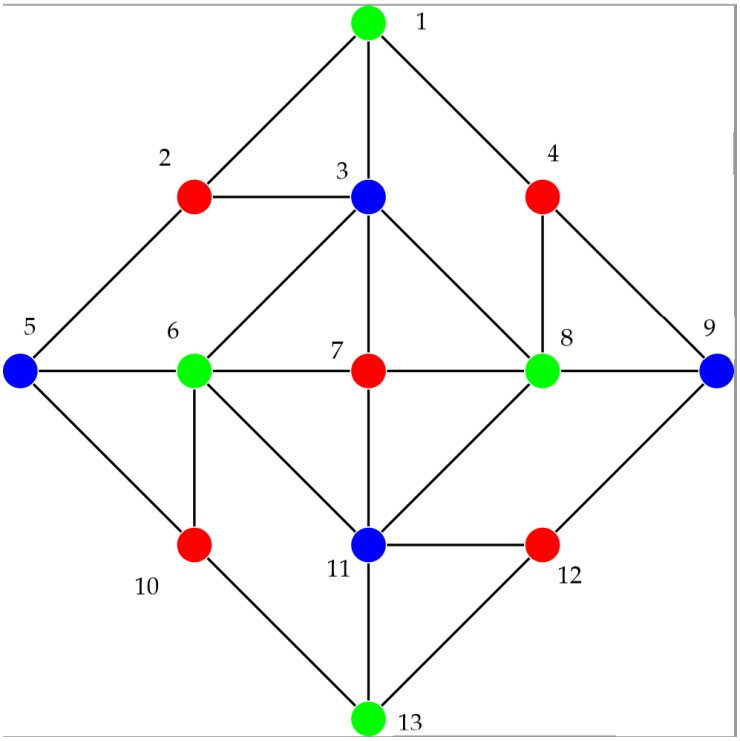
```
            elif colors[neighbor] != color:
                print("Grafica no 2-colorable")
                sys.exit()
```

Por ultimo imprimimos el color correspondiente a cada vertice para mostrar que si es 2 colorable y acabamos

```
print("Grafica 2 coloreable")
for i in range(1, colors.__len__()):
    print(f"Para v={i}, su color es:{colors[i]}")
return
```

Ejemplos de ejecución

- ejemplar 3 - coloreable



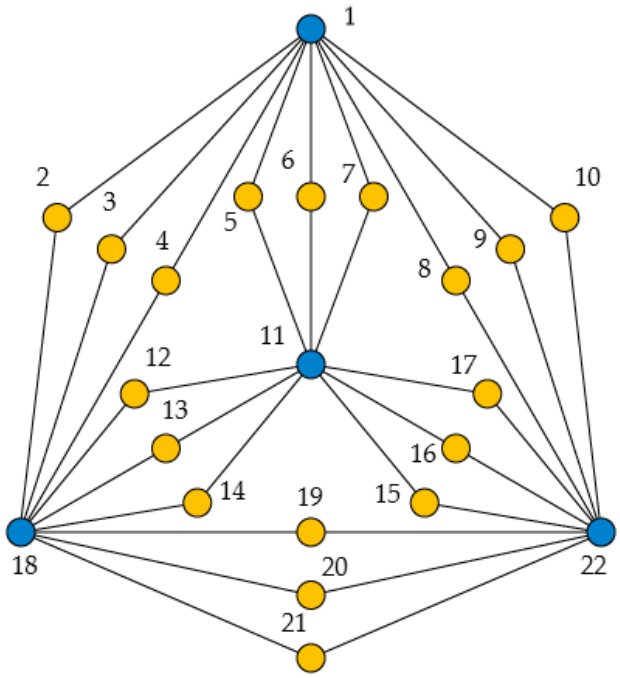
codificación:

```
3
13
(1,2), (1,3), (1,4), (2,3), (2,5), (3,6), (3,7), (3,8), (4,8), (4,9), (5,6), (5,10), (6,10),
(6,11), (6,7), (7,11), (7,8), (8,11), (8,9), (9,12), (10,13), (11,12), (11,13), (12,13)
```

Ejecución:

```
C:\Users\alexa\OneDrive\Desktop\practicasCumplejidad\Complejidad\Practic
a1\src>python script_coloracion.py
Lista de tuplas: [(1, 2), (1, 3), (1, 4), (2, 3), (2, 5), (3, 6), (3, 7)
, (3, 8), (4, 8), (4, 9), (5, 6), (5, 10), (6, 10), (6, 11), (6, 7), (7,
 11), (7, 8), (8, 11), (8, 9), (9, 12), (10, 13), (11, 12), (11, 13), (1
2, 13)]
El problema de la 3-coloracion es NP
```

- ejemplar 2 - coloreable



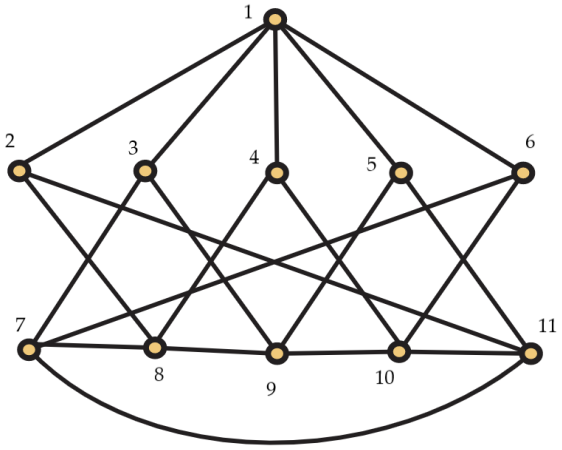
Codificación:

```
2
22
(1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (1,8), (1,9), (1,10), (2,18), (3,18), (4,18),
(5,11), (6,11), (7,11), (8,22), (9,22), (10,22), (11,12), (11,13), (11,14), (11,15), (11,16),
(11,17), (12,18), (13,18), (14,18), (15,22), (16,22), (17,22), (18,19), (18,20), (18,21),
(19,22), (20,22), (21,22)
```

Ejecución:

```
C:\Users\alexa\OneDrive\Desktop\practicasCumplejidad\Complejidad\Practic
a1\src>python script_coloracion.py
Lista de tuplas: [(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8)
, (1, 9), (1, 10), (2, 18), (3, 18), (4, 18), (5, 11), (6, 11), (7, 11),
 (8, 22), (9, 22), (10, 22), (11, 12), (11, 13), (11, 14), (11, 15), (11
, 16), (11, 17), (12, 18), (13, 18), (14, 18), (15, 22), (16, 22), (17,
22), (18, 19), (18, 20), (18, 21), (19, 22), (20, 22), (21, 22)]
Grafica 2 coloreable
Para v=1, su color es:2
Para v=2, su color es:1
Para v=3, su color es:1
Para v=4, su color es:1
Para v=5, su color es:1
Para v=6, su color es:1
Para v=7, su color es:1
Para v=8, su color es:1
Para v=9, su color es:1
Para v=10, su color es:1
Para v=11, su color es:2
Para v=12, su color es:1
Para v=13, su color es:1
Para v=14, su color es:1
Para v=15, su color es:1
Para v=16, su color es:1
Para v=17, su color es:1
Para v=18, su color es:2
Para v=19, su color es:1
Para v=20, su color es:1
Para v=21, su color es:1
Para v=22, su color es:2
```

- ejemplar que NO sea 2 - coloreable



Codificación:

```
2
11
(1,2), (1,3), (1,4), (1,5), (1,6), (2,8), (2,11), (3,7), (3,9), (4,8), (4,10), (5,9), (5,11),
(6,7), (6,10), (7,8), (7,11), (8,9), (9,10), (10,11)
```

Ejecución:

```
C:\Users\alexa\OneDrive\Desktop\practicasCumplejidad\Complejidad\Practic
a1\src>python script_coloracion.py
Lista de tuplas: [(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 8), (2, 11
), (3, 7), (3, 9), (4, 8), (4, 10), (5, 9), (5, 11), (6, 7), (6, 10), (7
, 8), (7, 11), (8, 9), (9, 10), (10, 11)]
Grafica no 2-colorable
```