Universidad Nacional
Autónoma de México

Facultad de Ciencias

Complejidad Computacional

2024-2

Práctica 2

Linares Gil, Daniel 420490056
Chávez Zamora, Mauro Emiliano 111001079

13 de abril de 2024

Pertenencia a NP

Debemos desmostar que el siguiente problema puede verificarse en tiempo polinomial para demostrar que pertenece a NP.

K-coloración
Ejemplar: Una gráfica finita G y un entero no negativo k.
Pregunta: ¿Existe una coloración de la gráfica con k colores?

Una coloración es una función $c : V \to \mathbb{N}$, que asigna números a cada vértice como si de colores se tratara. Es decir, cada color tiene asignado un número y cada vértice tiene una relación con el número que representa a algún color. Así que hagamos como que c es mi función de coloración y verifiquemos en tiempo polinomial.

Algoritmo:

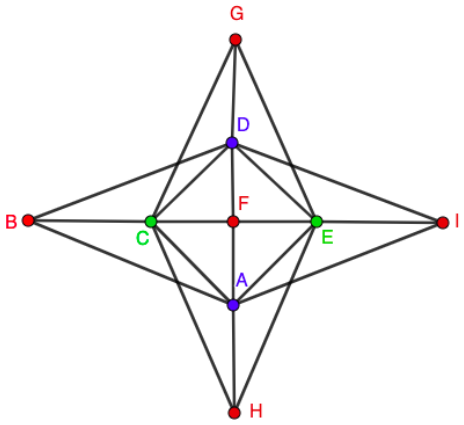
- Tomamos un vértice, ahora revisamos que los vértices adyacentes a él no tengan asignados el mismo color.
    - Si lo tienen asignado entonces rechazamos y c no es una k coloración de G.
    - Si ninguno de ellos tiene el mismo color asignado entonces pasamos al vértice siguiente.
        - Si no hay vértice siguiente entonces aceptamos, y c efectivamente es una k coloración de G.
        - Si hay vértice siguiente entonces repetimos el procedimiento desde el inicio con el nuevo vértice.

Notamos que para una gráfica $G = (V, E)$ vamos a tomar $|V| = n$ pasos para revisar cada vértice, pero además haremos este procedimiento de revisar la coloración a lo más una $n - 1$ cantidad de pasos, es decir $|V - \{v_i\}| = n - 1$ para cada $1 \leq v_i \leq |V|$, ya que no tenemos lazos en la gráfica. Por lo que la complejidad de éste algoritmo es $O(n * (n - 1)) = O(n^2)$.

Debido a que hemos podido dar un algoritmo que verifica en tiempo polinomial una certificado de k-coloración, entonces hemos demostrado que k-coloración pertenece a NP usando la definición de computos deterministas.

Certificado y esquema de codificación a utilizar

En nuestro caso, haremos que los certificados se codifiquen con un vértice asociado a cada línea; el número asignado a esa línea significa que tenemos ese número de color asignado al vértice. Reutilizaremos una gráfica de la práctica pasada, la codificación para los ejemplares permanece igual.



En el caso de ésta gráfica 3-coloreable, tendríamos que $A = 1$, $B = 2$, $C = 3$, $D = 4$, $E = 5$, $F = 6$, $G = 7$, $H = 8$ y $I = 9$. Recordemos que la codificación de la gráfica que utilizamos en la práctica pasada fue:

```
2 3 5 6 8 9 \n 1 3 4 \n 1 2 4 6 7 8 \n 2 3 5 6 7 9 \n 1 4 6 7 8 9 \n
1 3 4 5 \n 3 4 5 \n 1 3 5 \n 1 4 5
```

Por lo que la codificación de su certificado queda de la siguiente manera:

```
1 \n 2 \n 3 \n 1 \n 3 \n 2 \n 2 \n 2 \n 2
```

Con $azul = 1$, $rojo = 2$ y $verde = 3$.

Algoritmo para generar certificado aleatorios

Para nuestro algoritmo pensamos en tomar un vértice cualquiera y asignarle un color al azar, pero el problema es que si llegamos a verificar en algún momento que antes de asignar un color a un vértice éste no sea adyacente a otro vértice del mismo color en general estaríamos generando un procedimiento no aleatorio, estaríamos resolviendo el problema. Por lo que el algoritmo para generar certificados aleatorio es de la siguiente manera:

Algoritmo

- Toma un elemento de los vértices y asígnale un color al azar del conjunto de los posibles colores a elegir.
- Revisa el conjunto restante de vértices:
  - Si es vacío entonces ya terminamos de asignar colores.
  - Si no es vacío repetimos el procedimiento con el conjunto restante.

Este algoritmo es el que hemos implementado en nuestro código anexado a la práctica.

Algoritmo para verificar

- Primero, se recorre el certificado y se verifica que no se hayan utilizado más de k colores.
- Luego, se itera sobre todos los vértices.
  - Para cada vértice, se recorren sus vecinos. Si el certificado asigna el color del vértice a alguno de sus vecinos, se regresa que el ejemplar no es k coloreable con el certificado dado.
- Si se termina de iterar sobre todos los vértices sin que se encuentre dos vértices adyacentes con el mismo color, entonces se regresa que el ejemplar SI es k coloreable con el certificado dado.

Complejidad del algoritmo:
Sea $G = (V, E)$ una gráfica con V el conjunto de vértices y E el conjunto de aristas. Se realizan dos ciclos, en el ciclo externo se itera sobre todos los vértices y para cada vértice se itera sobre sus vecinos. El número total de iteraciones en el peor caso tiene complejidad en tiempo $O(|V| + |E|)$, y como el número de aristas de una gráfica crece a lo más $O(|V|^2)$ respecto al número de vértices, la complejidad en tiempo respecto al número de vértices es $O(|V|^2)$. Por lo tanto la complejidad en tiempo del algoritmo respecto al número de vértices es $O(|V|^2)$.

Resumen pruebas

Ejemplar 1 ($K = 3$):

- Certificado 1 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.1s)
python src/practica.py verificar ejemplares/ej1.txt certificados/ej1_cert1.txt
Número de Vértices: 9
Número de Aristas: 20
Entero K: 3
Número colores utilizados (certificado): 3
¿El ejemplar, con el certificado dado, es K=3 coloreable? NO
```

- Certificado 2 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.084s)
python src/practica.py verificar ejemplares/ej1.txt certificados/ej1_cert2.txt
Número de Vértices: 9
Número de Aristas: 20
Entero K: 3
Número colores utilizados (certificado): 3
¿El ejemplar, con el certificado dado, es K=3 coloreable? NO
```

- Certificado 3 (Si):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.085s)
python src/practica.py verificar ejemplares/ej1.txt certificados/ej1_cert3.txt
Número de Vértices: 9
Número de Aristas: 20
Entero K: 3
Número colores utilizados (certificado): 3
¿El ejemplar, con el certificado dado, es K=3 coloreable? SI
```

- Certificado 4 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.082s)
python src/practica.py verificar ejemplares/ej1.txt certificados/ej1_cert4.txt
Número de Vértices: 9
Número de Aristas: 20
Entero K: 3
Número colores utilizados (certificado): 3
¿El ejemplar, con el certificado dado, es K=3 coloreable? NO
```

- Certificado 5 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.086s)
python src/practica.py verificar ejemplares/ej1.txt certificados/ej1_cert5.txt
Número de Vértices: 9
Número de Aristas: 20
Entero K: 3
Número colores utilizados (certificado): 3
¿El ejemplar, con el certificado dado, es K=3 coloreable? NO
```

Ejemplar 2 ($K = 12$):

- Certificado 1 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.078s)
python src/practica.py verificar ejemplares/ej2.txt certificados/ej2_cert1.txt
Número de Vértices: 12
Número de Aristas: 24
Entero K: 12
Número colores utilizados (certificado): 12
¿El ejemplar, con el certificado dado, es K=12 coloreable? NO
```

- Certificado 2 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.084s)
python src/practica.py verificar ejemplares/ej2.txt certificados/ej2_cert2.txt
Número de Vértices: 12
Número de Aristas: 24
Entero K: 12
Número colores utilizados (certificado): 12
¿El ejemplar, con el certificado dado, es K=12 coloreable? NO
```

- Certificado 3 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.08s)
python src/practica.py verificar ejemplares/ej2.txt certificados/ej2_cert3.txt
Número de Vértices: 12
Número de Aristas: 24
Entero K: 12
Número colores utilizados (certificado): 12
¿El ejemplar, con el certificado dado, es K=12 coloreable? NO
```

- Certificado 4 (Si):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.066s)
python src/practica.py verificar ejemplares/ej2.txt certificados/ej2_cert4.txt
Número de Vértices: 12
Número de Aristas: 24
Entero K: 12
Número colores utilizados (certificado): 12
¿El ejemplar, con el certificado dado, es K=12 coloreable? SI
```

- Certificado 5 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.082s)
python src/practica.py verificar ejemplares/ej2.txt certificados/ej2_cert5.txt
Número de Vértices: 12
Número de Aristas: 24
Entero K: 12
Número colores utilizados (certificado): 12
¿El ejemplar, con el certificado dado, es K=12 coloreable? NO
```

Ejemplar 3 ($K = 6$):

- Certificado 1 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.086s)
python src/practica.py verificar ejemplares/ej3.txt certificados/ej3_cert1.txt
Número de Vértices: 6
Número de Aristas: 15
Entero K: 6
Número colores utilizados (certificado): 6
¿El ejemplar, con el certificado dado, es K=6 coloreable? NO
```

- Certificado 2 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.078s)
python src/practica.py verificar ejemplares/ej3.txt certificados/ej3_cert2.txt
Número de Vértices: 6
Número de Aristas: 15
Entero K: 6
Número colores utilizados (certificado): 6
¿El ejemplar, con el certificado dado, es K=6 coloreable? NO
```

- Certificado 3 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.075s)
python src/practica.py verificar ejemplares/ej3.txt certificados/ej3_cert3.txt
Número de Vértices: 6
Número de Aristas: 15
Entero K: 6
Número colores utilizados (certificado): 5
¿El ejemplar, con el certificado dado, es K=6 coloreable? NO
```

- Certificado 4 (No):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.083s)
python src/practica.py verificar ejemplares/ej3.txt certificados/ej3_cert4.txt
Número de Vértices: 6
Número de Aristas: 15
Entero K: 6
Número colores utilizados (certificado): 6
¿El ejemplar, con el certificado dado, es K=6 coloreable? NO
```

- Certificado 5 (Si):

```
base ~/school/complejidad-computacional/complejidad-computacional/practica2 git:(main)±6 (0.106s)
python src/practica.py verificar ejemplares/ej3.txt certificados/ej3_cert5.txt
Número de Vértices: 6
Número de Aristas: 15
Entero K: 6
Número colores utilizados (certificado): 6
¿El ejemplar, con el certificado dado, es K=6 coloreable? SI
```