Alumno:	Erik Rangel Limón	**Complejidad Computacional**
No. Cuenta	318159287	*2024-1*
Correo	erikrangel.014@ciencias.unam.mx	Práctica 1
Alumno:	Axel Ducloux Hurtado	*Oscar Hernández Constantino*
No. Cuenta:	316309132	*18 de Septiembre de 2023*
Correo:	gorymirror95@ciencias.unam.mx	
Alumno:	Alejandro Terrazas Rivera	
No. Cuenta:	421006692	

Esquemas de Codificación

Considera el siguiente problema

Ruta Inducida:

Ejemplar: Una gráfica $G = (V, E)$, un entero positivo $K \leq |V|$

Pregunta: ¿Existe un subconjunto $V' \subseteq V$, con $|V'| \geq K$, tal que la subgráfica inducida por V' es una ruta simple con $|V'|$ vértices?

---

1. Propón una versión de optimización para el problema anterior.

   Ejemplar: Una gráfica $G = (V, E)$

   Pregunta: ¿Cuál sería la máxima K posible para la cual existe un subconjunto $V' \subseteq V$ con $|V'| = K$ tal que la subgráfica inducida por V' es una ruta simple con K vértices? (¿cuál sería un ejemplo de ese subconjunto?)

---

2. Describir e implementar un esquema de codificación razonable para ejemplares del problema Ruta Inducida (en su versión de decisión).

   El esquema de codificación propuesto utilizado para gráficas no puede ser una matriz.

   Utilizaremos un esquema de codificación de gráficas por listas de vecindades.

   El alfabeto que utilizaremos es el siguiente:

   $$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \P, (,)\}$$

   El esquema de codificación que proponemos utiliza ¶, el salto de línea, y la coma.

   Si hay $|V| = n$ vértices en la gráfica, cada vértice será representado por un número del 0 al $n - 1$ en base 10.

   Codificaremos un ejemplar del problema como sigue:

   - El número K (el del ejemplar) en base 10 seguido de un sálto de línea ¶.
   - Para cada vértice v en la gráfica (del 0 al $n - 1$, en ese orden):
     - Si $N(v)$ es el conjunto de vecinos de v, enlistamos todos los vecinos en $N(v)$ por medio de su representación numérica en base 10 separando cada uno de ellos por una coma $(,)$. Después de enlistar $N(v)$ le sigue un salto de línea ¶.

   Por ejemplo, el ejemplar $K = 2$ y la gráfica $G_1 = (V_1, E_1)$, que sería la siguiente:

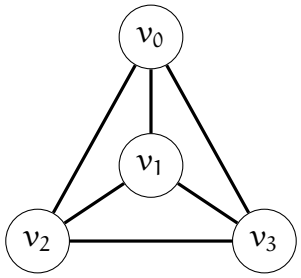


Figura 1: Gráfica G_1

Se codificaría de la siguiente forma:

$$2¶1,2,3¶0,2,3¶0,1,3¶0,1,2¶$$

Y en un archivo de texto se vería así:

```
2
1,2,3
0,2,3
0,1,3
0,1,2
```

---

3. Describir e implementar un algoritmo para transformar el esquema de codificación propuesto, a uno que utilice codificación de gráficas como matrices.

   El programa implementada debe recibir como entrada:

   - Nombre del archivo de entrada con el ejemplar a leer.
     El ejemplar debe seguir el esquema de codificación propuesto.
   - Nombre del archivo de salida con el resultado de la codificación como matriz.

   Como resultado de la ejecución, el programa deberá producir como salida:

   - Imprimir en consola el número de vértices y aristas en la gráfica G.
   - Imprimir en consola el valor de K
   - Escribir en el archivo de texto (indicando en el segundo parámetro del programa) una cadena (como texto plano) que represente los parámetros del ejemplar (aplicando un esquema de codificación como matriz para la gráfica).

   [Extra] implementar la salida en el archivo con un esquema de codificación que haga uso únicamente del alfabeto binario.

   Primero proponemos una función que lee un ejemplar usando la codificación que propusimos.

   function Decode(E *La cadena de texto del ejemplar usando la codificación propuesta*)
   $L \leftarrow$ Split(E, ¶) ▷ Divide la cadena por cada aparicion de ¶
   $k \leftarrow$ ParseInteger(L_0) ▷ La listas son 0-indexadas
   $n \leftarrow$ Length(L) $-$ 1
   $M \leftarrow$ Matrix(n, n) ▷ Inicializa una matriz vacía de tamaño $n \times n$
   $e \leftarrow 0$
   $i \leftarrow 1$

```
    while i ≤ n do
        V ← Split(L_i, ' , ')                          ▷ Separa L_i por comas
        for v ∈ V do
            e ← e + 1
            j ← ParseInteger(v)
            M_{(i−1) j} ← 1                            ▷ La matriz también debe ser 0-indexada (por eso el i − 1)
        end for
        i ← i + 1
    end while
    return (k, M, n, e/2)
end function
```

En resumen, primero separamos por saltos de línea y luego por comas, poniendo un 1 en su respectivo vecino en la matriz.

Nuestra codificación de entrada imprime el número K del ejemplar en decimal, por lo que tiene al menos complejidad $O(\log K)$, y luego cada vecino se representa con un número decimal, por lo que si hay $|V|$ vecinos, cada uno de ellos tiene tamaño a lo más $O(\log |V|)$, y como se enumeran todos los vecinos de cada vértices, hay tantos vecinos nombrados como los grados de cada uno de los vértices y como la suma de grados de los vértices es dos veces el número de aristas, la complejidad que tiene nuestra codificación tiene complejidad $O(2|E| \log |V| + \log K) = O(|E| \log |V| + \log K)$

El algoritmo que mostramos recorre una vez toda la codificación para hacer la primera división por saltos de línea $O(|E| \log |V| + \log |K|)$, luego cada una de las divisiones las dividimos por comas, para obtener los vecinos de cada vértices y luego ponerlo en su posición correspondiente en la matriz, de esta manera en la iteración visitamos de nuevo a todos los vecinos nombrados, por lo que sabemos que nos toma tiempo $O(|E| \log |V|)$, y por tanto, éste primer paso nos toma $O(|E| \log |V| + \log K)$ (vamos a considerar que la creación de una matriz vacía puede hacerse en un sólo paso si ya se tiene previamente)

Luego definimos una función que traduce de decimal a binario:

```
function DecToBin(k un número natural)
    if k = 0 then
        return ''0''
    end if
    n ← k
    r ← ε                                              ▷ La cadena vacía
    while n ≥ 1 do
        if n  mód 2 = 1 then
            r ← ''1'' ++ r                             ▷ El operador ++ es la concatenación de cadenas
        else
            r ← ''0'' ++ r
        end if
        n ← ⌊n/2⌋                                      ▷ La división entera
    end while
    return r
end function
```

Éste procedimiento es el tradicional para tranformar un número decimal a binario, calculamos los módulos y ese será nuestro dígito al final, dividimos entre dos y repetimos el proceso.

Éste procedimiento toma log K pasos, pues en cada paso dividimos entre dos hasta que el resultado sea uno, lo que es equivalente a resolver la siguiente ecuación:

$$\frac{k}{2^x} = 1$$
$$k = 2^x$$
$$\log_2 k = x$$

Donde x sería el número de pasos

Teniendo esto, podemos definir una función que recibe una matriz y devuelve una cadena binaria de la representación de esa matriz.

function GRAPHTOBIN(k *Un número entero positivo*, M *La matriz de adyacencias*, n *El número de vértices*)
  $r \leftarrow \varepsilon$
  $i \leftarrow 0$
  while $i < n$ **do**
    $r \leftarrow r$ ++ ''10''
    $j \leftarrow 0$
    while $j < n$ **do**
      if $M_{ij} = 0$ **then**
        $r \leftarrow r$ ++ ''00''
      else
        $r \leftarrow r$ ++ ''01''
      end if
      $j \leftarrow j + 1$
    end while
    $i \leftarrow i + 1$
  end while
  $r \leftarrow r$ ++ ''11'' ++ DECTOBIN(k)
  return r
end function

Éste procedimiento lo que hace es recorrer la matriz poniendo "00", "01", "10" o "11" según sea el caso.

Al recorrer la matriz éste algoritmo toma $O(|V|^2)$ necesariamente, y para agregar la codificación binaria de la K nos toma $O(\log K)$, por tanto toma $O(|V|^2 + \log K)$

La semántica de la codificación de salida en matriz es la siguiente:

- "10" Indica que inicia una nueva fila en la matriz.
- "00" Indica que no existe una arista entre los vértices ij.
- "01" Indica que hay una arista entre los vértices ij.
- "11" Indica que termina la matriz.
- Lo que sigue después de la cadena "11" es la representación binaria del número K del ejemplar.

De esta manera se puede leer la cadena de dos en dos caracteres interpretando la matriz hasta leer "11", en cuyo caso sabremos que ya habremos terminado de leer la matriz y comenzamos a leer un número en su representación binaria.

Por último, nuestro programa principal sería parecido al siguiente:

function MAIN(input *El primer parámetro del programa*, output *El segundo*)
  $(k, M, n, e) \leftarrow$DECODE(GETCONTENTS(input))
  PRINT("*La gráfica tiene* " ++ n ++ " *vértices*")
  PRINT("*y* " ++ e ++ " *aristas*")

PRINT(*"El valor de K es:"* ++ k)
WRITE(output, GRAPHTOBIN(k, M, n))
end function

Por lo tanto el procedimiento en total toma tiempo $O(|V|^2 + |E| \log |V| + \log K)$

---

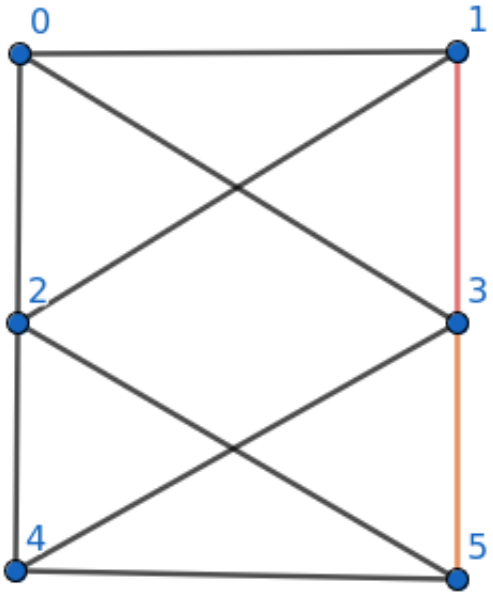
4. Incluir ejemplos de ejecución, así como ejemplos gráficos de los ejemplares con los que prueben su programa.

   - Ejemplar con $|V| \geq 5$, $|E| \geq 10$, $K = 3$ y respuesta **sí**:
     Mostramos un suconjunto de tamaño $K = 3$ que su gráfica inducida es una ruta simple.

     a) **cadena que codifica a cada ejemplar**

```
La entrada fue:
3
1,2,3
0,2,3
0,1,4,5
0,1,4,5
2,3,5
2,3,4
```

     b) **representación visual del ejemplar (sin codificar)**

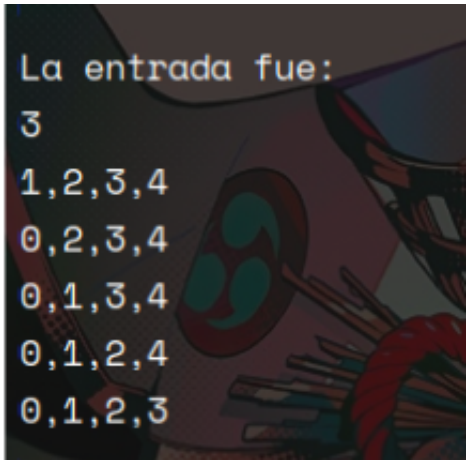


     c) **resultado de la ejecución del programa**

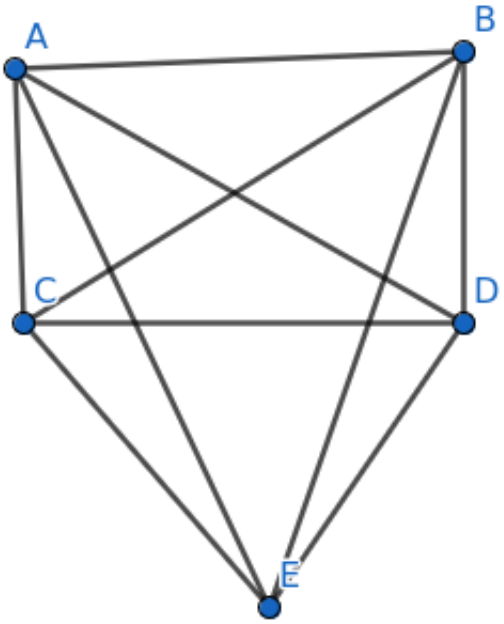
```
axelducloux@axelducloux-Nitro-AN515-55:~/Complejidad/practicas-complejidad/Practica1$ python3
La gráfica tiene 6 vértices y 10 aristas
El valor de K es 3

La entrada fue:
3
1,2,3
0,2,3
0,1,4,5
0,1,4,5
2,3,5
2,3,4

Su codificación binaria es:
1000010101000010010001010000100101000001011001010000010110000001010001100000010101001111
```

- Ejemplar con $|V| \geq 5$, $|E| \geq 10$, $K = 2$ y respuesta **no**:
  Siempre que el conjunto de aristas sea distinto a vacío, para cualquiera arista, a saber, uv, podríamos tomar los vértices u y v para el subconjunto de tamaño dos, y su gráfica inducida sería siempre una ruta simple, por lo que no es posible que el problema de decisión regrese *no* con estas condiciones.
  Mostramos una gráfica con éstas mismas características, pero con $K = 5$, respuesta **no**.

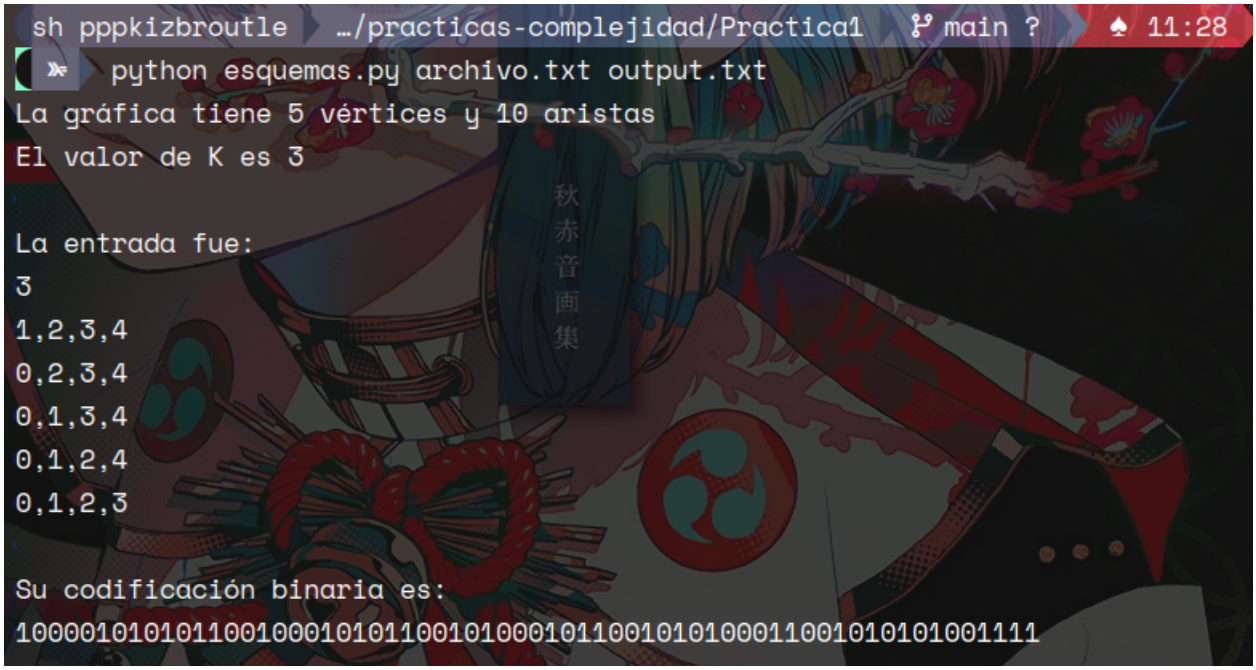
  a) **cadena que codifica a cada ejemplar**



  b) **representación visual del ejemplar (sin codificar)**



  c) **resultado de la ejecución del programa**



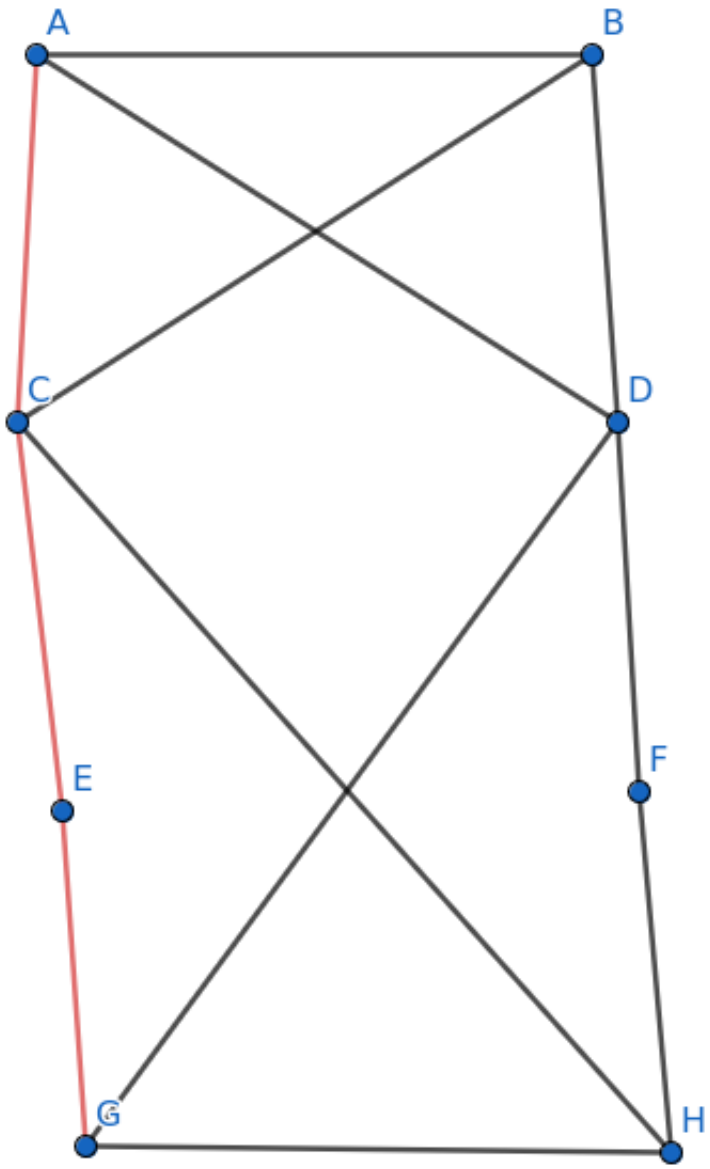
Al ser una gráfica completa, cualesquiera 3 vértices que tomemos van a estar conectados entre sí, por lo que no sería posible tomar un subconjunto de tamaño 3 cuya gráfica inducida sea una ruta simple.

- Ejemplar con $|V| \geq 6$, $|E| \geq 10$, $K = 3$ y respuesta **sí**:
  Mostramos un suconjunto de tamaño $K = 3$ (que es válido para nuestro problema de decisión) que su gráfica inducida es una ruta simple.

  a) **cadena que codifica a cada ejemplar**

```
La entrada fue:
3
1,2,3
0,2,3
0,1,4,7
0,1,5,6
2,6
3,7
3,4,7
2,5,6
```

  b) **representación visual del ejemplar (sin codificar)**



c) **resultado de la ejecución del programa**

```
axelducloux@axelducloux-Nitro-AN515-55:~/Complejidad/practicas-complejidad/Practica1$ python3 esquemas.py ejemplo3.txt ejemplo3.1.txt
La gráfica tiene 8 vértices y 12 aristas
El valor de K es 3

La entrada fue:
3
1,2,3
0,2,3
0,1,4,7
0,1,5,6
2,6
3,7
3,4,7
2,5,6

Su codificación binaria es:
1000010101000000001001000101000000001001010000100000110010100000001010010000001000000010010000000010000000110000000010100000110000001000001010011111
```

Referencias

[1] Gonzalez,D.A.(2017)*Introducción a la Teoría de las Gráficas*.Departamento de Matemáticas Aplicadas y Sistemas Universidad Autónoma Metropolitana - Cuajimalpa, 67-112

[2] W. McCuaig (1984) *A simple proof of Menger's theorem*, J. Graph Theory 8 , 427–429.

[3] R. J. Wilson (1998) *Introduction to Graph Theory*, 4th Edition, Longman Group Ltd.