Tlamati Sabiduría

Algoritmos para crear la matriz de adyacencia de las gráficas $S(G)$, $L(G)$, $R(G)$, $Q(G)$ y $T(G)$

Raúl Juárez-Morales[1*]
Luis Alberto Lucrecio-González[2]
Gerardo Reyna-Hernández[3]
Omar Rosario-Cayetano[3]

[1]Escuela Superior de Ciencias y Tecnologías de la Información, Universidad Autónoma de Guerrero. Av. Las Colinas No. 37A, Fraccionamiento Las Playas, Acapulco, Guerrero, México
[2]Centro Regional de Educación Superior de la Costa Chica, Universidad Autónoma de Guerrero. Carretera Cruz Grande – Ayutla, Cruz Grande, Municipio de Florencio Villareal
[3]Facultad de Matemáticas, Universidad Autónoma Guerrero. Carlos E. Adame 5, Col. La Garita, Acapulco, Guerrero, México

**Autor de correspondencia*
raul.juarezm@uagro.mx

Resumen

Sea $G = (V(G), E(G))$ una gráfica finita, simple y conexa, con conjunto de vértices $V(G)$ y conjunto de arista $E(G)$, donde $n = |V(G)|$ es el orden de G, $m = |E(G)|$ es el tamaño de G. La gráfica S(G) se obtiene insertando un vértice adicional en cada arista de G o equivalentemente reemplazando cada una de sus aristas por un camino de longitud dos. La gráfica R(G) es la que se obtiene a partir de G, agregando un nuevo vértice por cada arista de G y uniendo cada vértice nuevo a los extremos de la arista correspondiente a él. Q(G) se define como la gráfica obtenida a partir de G insertando un vértice adicional en cada arista de G y uniendo pares de estos nuevos vértices si y solo si las aristas correspondientes en G son incidentes. La gráfica total T(G) se define como la gráfica obtenida a partir de G insertando un vértice adicional en cada arista de G, con dos vértices de T(G) adyacentes si y solo si los elementos correspondientes de G son adyacentes o incidentes. En este trabajo se crearon algoritmos para construir la matriz de adyacencia para las gráficas S(G), L(G), R(G), Q(G) y T(G).

Palabras claves: Gráficas, Operadores de gráficas, Matriz de adyacencia, Algoritmo

Como citar el artículo:
Juárez-Morales, R., Lucrecio-González, L.A., Reyna-Hernández, G., Rosario-Cayetano, O. (2022). Algoritmos para crear la matriz de adyacencia de las gráficas $S(G)$, $L(G)$, $R(G)$, $Q(G)$ y $\boldsymbol{T(G)}$. *Tlamati Sabiduría*, 13, 17-25.

Abstract

Let $G = (V(G), E(G))$ be a finite, simple and connected graph, with set of vertices $V(G)$ and set of edges $E(G)$, when $n = |V(G)|$ is the order of G, $m = |E(G)|$ is the size of G. The graph $S(G)$ is obtained inserting an additional vertex in each edge of G or equivalently replacing each of its edges by a path of length two. The graph $R(G)$ is obtained from G by adding a new vertex corresponding to each edge of G and by joining each new vertex to the end points of the edge corresponding to it. $Q(G)$ is defined as the graph obtained from G by inserting a new vertex into every edge of G and by joining by edges those pairs of these new vertices if and only if the edges corresponding in G are incidents. The graph total $T(G)$ is defined as the graph obtained from G by inserting a additional vertex in each edge of G, with two vertex of $T(G)$ adjacent if and only if the elements corresponding of G are adjacent or incident. In this paper Algorithms were created to generate the adjacency matrix for the $S(G)$, $L(G)$, $R(G)$, $Q(G)$ and $T(G)$ graphs.

Keywords: Graphs, Graph operators, Adjacency matrix, Algorithm.

Introducción

Los algoritmos en teoría de graficas han ayudado a resolver problemas teóricos y prácticos muy importantes. Shafique (2004) realizó un trabajo de tesis de Alianzas de un grafo aplicado a Data Clustering. Hedetniemi *et al.* (1986) presentan un algoritmo de tiempo lineal para encontrar un conjunto dominante de orden mínimo en una gráfica cactus. Lenzen *et al.* (2013) reportan dos algoritmos, uno determinista y otro probabilístico, el cual da aproximaciones distribuidas de conjuntos dominantes mínimos.

En estas últimas décadas se han desarrollado trabajos sobre las gráficas en operadores unitarios de gráficas. Hande *et al.* (2013) determinan la gráfica derivada de la gráfica subdivisión. En Yan y Yeh, (2009); Ranjini *et al.* (2011); Nadeem *et al.* (2015); Klein y Yi, (2012); y, Ascioglu y Cangul, (2018), se pueden ver trabajos sobre la gráfica $S(G)$. Sigarreta y Rodríguez (2006) llevaron a cabo el estudio de las alianzas defensivas en la gráfica línea ($L(G)$). Castro et al. (2020) realizaron estudios del diferencial de la gráfica $Q(G)$. En Hernández (2020) se presenta una tesis doctoral en la cual se estudia el diferencial de la gráfica $R(G)$.

Los problemas de investigación en gráficas por lo regular tienen que ver con el número de subconjuntos de un conjunto de vértices V de la gráfica G, por lo tanto, una enumeración completa de todas las soluciones implicaría un coste exponencial $2^{|V|}$. Por ejemplo, para un grafo con solo 20 nodos, tal enumeración llevaría años en los ordenadores modernos. Se sabe que el problema es NP-duro y esto quiere decir que no existe ningún algoritmo exacto que encuentre una solución óptima en tiempo polinomial. Un algoritmo exacto de enumeración en tiempo exponencial fue sugerido por Gaspers *et al.* (2009); Van Rooij y Bodlaender, (2011); y, Fomin *et al.* (2009).

En este trabajo tenemos como objetivo construir los algoritmos que generan la matriz de adyacencia de las gráficas $S(G), L(G), R(G), Q(G)$ y $T(G)$ teniendo como entrada una gráfica G, estos algoritmos se pueden usar para desarrollar una herramienta de software que ayude a construir estas gráficas de una manera automática, para que los investigadores en esta rama puedan estudiar los diferentes parámetros de este tipo de gráficas.

Materiales y métodos

Una ***gráfica*** G es un par ordenado $(V(G), E(G))$ formado por un conjunto $V(G)$ de vértices y un conjunto $E(G)$, distinto de $V(G)$, de aristas, junto con una función de incidencia $\psi(G)$ que asocia a cada arista de G un par no ordenado de vértices (no necesariamente distintos) de G (Bondy y Murty, 2008).

Sea $G = (V(G), E(G))$ una gráfica, definiremos el ***orden*** de G como $|V(G)| = n$, el ***tamaño*** de G como $|E(G)| = m$.

Sea $G = (V(G), E(G))$ una gráfica, decimos que G es ***finita*** si el orden y el tamaño de G son finitos.

Sea $G = (V(G), E(G))$ una gráfica. Un ***lazo*** en G es una arista con extremos iguales. Una arista ***simple*** en G es aquella en la que sus extremos son distintos. Decimos que dos o más aristas son ***múltiples*** en G si tienen los mismos extremos en común.

Sea $G = (V(G), E(G))$ una gráfica, decimos que G es ***simple*** si no existen lazos o aristas múltiples en G.

Las definiciones de las gráficas $S(G), L(G), R(G), Q(G)$ y $T(G)$ fueron tomadas de Cvetkovi'c (1980).

La *gráfica subdivisión* $\boldsymbol{S(G)}$ se obtiene insertando un vértice adicional en cada arista de G o equivalentemente reemplazando cada una de sus aristas por un camino de longitud dos.

La *gráfica línea* $\boldsymbol{L(G)}$ de una gráfica G es la gráfica cuyos vértices corresponden a las aristas de G siendo dos vértices adyacentes si y solo si las aristas correspondientes en G tienen un vértice en común.

La gráfica $\boldsymbol{R(G)}$ es la que se obtiene a partir de G, agregando un nuevo vértice por cada arista de G y uniendo cada vértice nuevo a los extremos de la arista correspondiente a él.

La gráfica $\boldsymbol{Q(G)}$ se define como la gráfica obtenida a partir de G insertando un vértice adicional en cada arista de G y uniendo pares de estos nuevos vértices si y solo si las aristas correspondientes en G son incidentes.

La *gráfica total* $\boldsymbol{T(G)}$ se define como la gráfica obtenida a partir de G insertando un vértice adicional en cada arista de G, con dos vértices de $T(G)$ adyacentes si y solo si los elementos correspondientes de G son adyacentes o incidentes.

La ***matriz de adyacencia*** *MA(G)* de una gráfica G finita, simple y conexa es una matriz de tamaño $n \times n$ con entrada (i,j) igual a 1 si los vértices v_i y $v_j$son adyacentes y 0 en otro caso.

Ejemplo 1. Si G = (V(G), E(G)), con V(G) = {v_1, v_2, v_3, v_4, v_5} y $E(G)$ = {v_1v_2, v_1v_3, v_2v_4, v_2v_5,

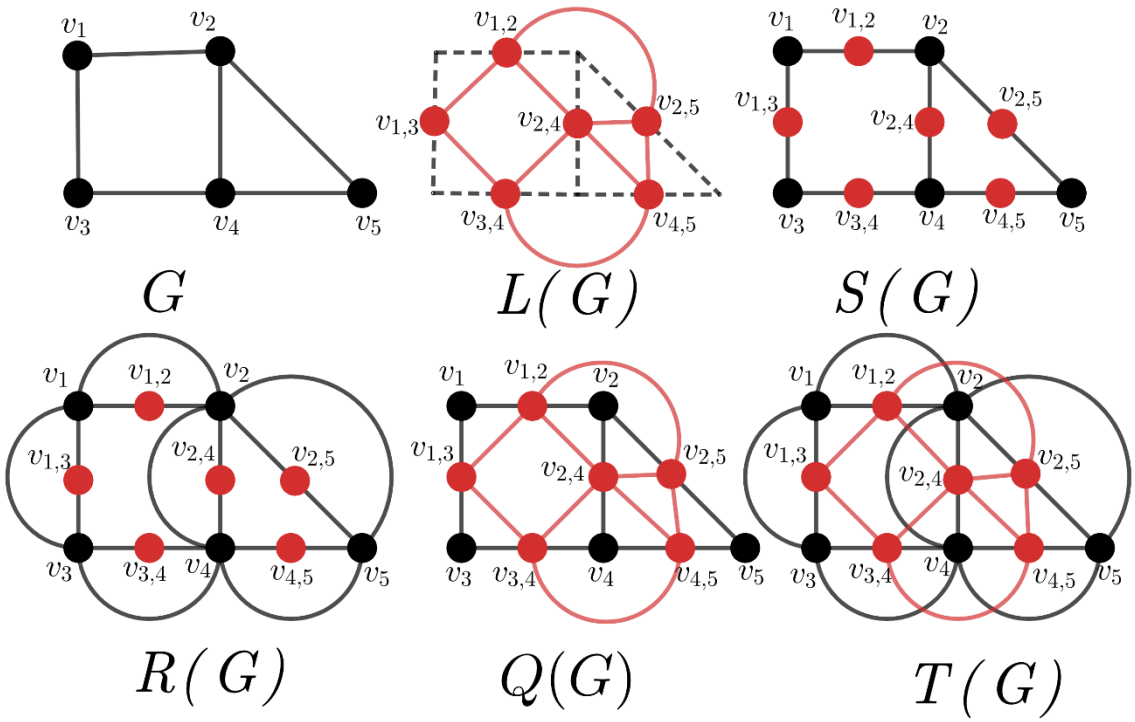

Figura 1. Diagrama de la gráfica *G, S(G), L(G), R(G), Q(G) y T(G).*

G	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	0	0
v_2	1	0	0	1	1
v_3	1	0	0	1	0
v_4	0	1	1	0	1
v_5	0	1	0	1	0

Tabla 1. Matriz de adyacencia de la gráfica G.

v_3v_4, $v_4v_5\}$, G tiene orden $n = 5$ y tamaño $m = 6$. En la Figura 1, se muestra el diagrama de la gráfica *G, S(G), L(G), R(G) y T(G)* en la Tabla 1 se muestra la matriz de adyacencia de *G*.

Ejemplo 2. Si tomamos G del **¡Error! No se encuentra el origen de la referencia.**, entonces *S(G) = (V(S(G), E(S(G))*, con *VS(G)* $\cup$ *E(G)* = $\{v_1, v_2, v_3, v_4, v_5, v_1v_2, v_1v_3, v_2v_4, v_2v_5, v_3v_4, v_4v_5\}$, y $E(S(G)) = \{v_1v_{1,2}, v_1v_{1,3}, v_2v_{1,2}, v_2v_{2,4}, v_2v_{2,5}, v_3v_{1,3}, v_3v_{3,4}, v_4v_{2,4}, v_4v_{3,4}, v_4v_{4,5}, v_5v_{2,5}, v_5v_{4,5}\}$. $|V(S(G))| = n + m$, y $|E(S(G))| = 2m$. En la Tabla 2 se muestra la matriz de adyacencia de la gráfica $S(G)$.

Ejemplo 3. Si tomamos G del **¡Error! No se encuentra el origen de la referencia.**, entonces $L(G) = (V(L(G)), E(L(G)))$, con $V(L(G)) = E(G) = \{ v_1v_2, v_1v_3, v_2v_4, v_2v_5, v_3v_4, v_4v_5\}$, y E(L(G))\ = $\{v_{1,2}v_{1,3}, v_{1,2}v_{2,4}, v_{1,2}v_{2,5}, v_{1,3}v_{3,4}, v_{3,4}v_{2,4}, v_{3,4}v_{4,5}, v_{4,5}v_{2,4}, v_{4,5}v_{2,5}, v_{2,5}v_{2,4}\}$. En la Tabla 3 se muestra la matriz de adyacencia de

S(G)	v_1	v_2	v_3	v_4	v_5	$v_{1,2}$	$v_{1,3}$	$v_{2,4}$	$v_{2,5}$	$v_{3,4}$	$v_{4,5}$
v_1	0	0	0	0	0	1	1	0	0	0	0
v_2	0	0	0	0	0	1	0	1	1	0	0
v_3	0	0	0	0	0	0	1	0	0	1	0
v_4	0	0	0	0	0	0	0	1	0	1	1
v_5	0	0	0	0	0	0	0	0	1	0	1
$v_{1,2}$	1	1	0	0	0	0	0	0	0	0	0
$v_{1,3}$	1	0	1	0	0	0	0	0	0	0	0
$v_{2,4}$	0	1	0	1	0	0	0	0	0	0	0
$v_{2,5}$	0	1	0	0	1	0	0	0	0	0	0
$v_{3,4}$	0	0	1	1	0	0	0	0	0	0	0
$v_{4,5}$	0	0	0	1	1	0	0	0	0	0	0

Tabla 2. Matriz de adyacencia de la gráfica S(G).

L(G)	$v_{1,2}$	$v_{1,3}$	$v_{2,4}$	$v_{2,5}$	$v_{3,4}$	$v_{4,5}$
$v_{1,2}$	0	1	1	1	0	0
$v_{1,3}$	1	0	0	0	1	0
$v_{2,4}$	1	0	0	1	1	1
$v_{2,5}$	1	0	1	0	0	1
$v_{3,4}$	0	1	1	0	0	1
$v_{4,5}$	0	0	1	1	1	0

Tabla 3. Matriz de adyacencia de la gráfica L(G).

la gráfica $L(G)$. En la Tabla 3 se muestra la matriz de adyacencia de la gráfica $L(G)$.

Ejemplo 4. Si tomamos G del **¡Error! No se encuentra el origen de la referencia.**, entonces $R(G) = (V(R(G)), E(R(G)))$, con $V(R(G)) = V(G) \cup E(G) = \{v_1, v_2, v_3, v_4, v_5, v_1v_2, v_1v_3, v_2v_4, v_2v_5, v_3v_4, v_4v_5\}$, y $E(R(G)) = E(G) \cup E(S(G)) = \{ v_1v_2, v_1v_3, v_2v_4, v_2v_5, v_3v_4, v_4v_5, v_1v_{1,2}, v_1v_{1,3}, v_2v_{1,2}, v_2v_{2,4}, v_2v_{2,5}, v_3v_{1,3}, v_3v_{3,4}, v_4v_{2,4}, v_4v_{3,4}, v_4v_{4,5}, v_5v_{2,5}, v_5v_{4,5}\}$. $|V(R(G))| = n + m$, y $|E(R(G))| = 3m$. En la Tabla 4 se muestra la matriz de adyacencia de la gráfica $R(G)$.

Ejemplo 5. Si tomamos G del **¡Error! No se encuentra el origen de la referencia.**, entonces $Q(G) = (V(Q(G)), E(Q(G)))$, con $V(Q(G)) = V(G) \cup E(G) = \{v_1, v_2, v_3, v_4, v_5, v_1v_2, v_1v_3, v_2v_4, v_2v_5, v_3v_4, v_4v_5\}$, y $E(Q(G)) = E(S(G)) \cup E(L(G)) = \{v_1v_{1,2}, v_1v_{1,3}, v_2v_{1,2}, v_2v_{2,4}, v_2v_{2,5}, v_3v_{1,3}, v_3v_{3,4}, v_4v_{2,4}, v_4v_{3,4}, v_4v_{4,5}, v_5v_{2,5}, v_5v_{4,5}, v_{1,2}v_{1,3}, v_{1,2}v_{2,4}, v_{1,2}v_{2,5}, v_{1,3}v_{3,4}, v_{3,4}v_{2,4}, v_{3,4}v_{4,5}, v_{4,5}v_{2,4}, v_{4,5}v_{2,5}, v_{2,5}v_{2,4}\}$. En la Tabla 5 se muestra la matriz de adyacencia de la gráfica $Q(G)$.

R(G)	v_1	v_2	v_3	v_4	v_5	$v_{1,2}$	$v_{1,3}$	$v_{2,4}$	$v_{2,5}$	$v_{3,4}$	$v_{4,5}$
v_1	0	1	1	0	0	1	1	0	0	0	0
v_2	1	0	0	1	1	1	0	1	1	0	0
v_3	1	0	0	1	0	0	1	0	0	1	0
v_4	0	1	1	0	1	0	0	1	0	1	1
v_5	0	1	0	1	0	0	0	0	1	0	1
$v_{1,2}$	1	1	0	0	0	0	0	0	0	0	0
$v_{1,3}$	1	0	1	0	0	0	0	0	0	0	0
$v_{2,4}$	0	1	0	1	0	0	0	0	0	0	0
$v_{2,5}$	0	1	0	0	1	0	0	0	0	0	0
$v_{3,4}$	0	0	1	1	0	0	0	0	0	0	0
$v_{4,5}$	0	0	0	1	1	0	0	0	0	0	0

Tabla 4. Matriz de adyacencia de la gráfica R(G).

Q(G)	v_1	v_2	v_3	v_4	v_5	$v_{1,2}$	$v_{1,3}$	$v_{2,4}$	$v_{2,5}$	$v_{3,4}$	$v_{4,5}$
v_1	0	0	0	0	0	1	1	0	0	0	0
v_2	0	0	0	0	0	1	0	1	1	0	0
v_3	0	0	0	0	0	0	1	0	0	1	0
v_4	0	0	0	0	0	0	0	1	0	1	1
v_5	0	0	0	0	0	0	0	0	1	0	1
$v_{1,2}$	1	1	0	0	0	0	1	1	1	0	0
$v_{1,3}$	1	0	1	0	0	1	0	0	0	1	0
$v_{2,4}$	0	1	0	1	0	1	0	0	1	1	1
$v_{2,5}$	0	1	0	0	1	1	0	1	0	0	1
$v_{3,4}$	0	0	1	1	0	0	1	1	0	0	1
$v_{4,5}$	0	0	0	1	1	0	0	1	1	1	0

Tabla 5. Matriz de adyacencia de la gráfica Q(G).

Ejemplo 6. Si tomamos G del **¡Error! No se encuentra el origen de la referencia.**, entonces $T(G) = (V(T(G)), E(T(G)))$, con $V(T(G)) = V(G) \cup E(G)$ ={v_1, v_2, v_3, v_4, v_5, v_1v_2, v_1v_3, v_2v_4, v_2v_5, v_3v_4, v_4v_5}, y $E(T(G)) = E(G) \cup E(S(G)) \cup E(L(G)) =$ { v_1v_2, v_1v_3, v_2v_4, v_2v_5, v_3v_4, v_4v_5, $v_1v_{1,2}$, $v_1v_{1,3}$, $v_2v_{1,2}$, $v_2v_{2,4}$, $v_2v_{2,5}$, $v_3v_{1,3}$, $v_3v_{3,4}$, $v_4v_{2,4}$, $v_4v_{3,4}$, $v_4v_{4,5}$, $v_5v_{2,5}$, $v_5v_{4,5}$, $v_{1,2}v_{1,3}$, $v_{1,2}v_{2,4}$, $v_{1,2}v_{2,5}$, $v_{1,3}v_{3,4}$, $v_{3,4}v_{2,4}$, $v_{3,4}v_{4,5}$, $v_{4,5}v_{2,4}$, $v_{4,5}v_{2,5}$, $v_{2,5}v_{2,4}$}. En la Tabla 6 se muestra la matriz de adyacencia de la gráfica $T(G)$.

T(G)	v_1	v_2	v_3	v_4	v_5	$v_{1,2}$	$v_{1,3}$	$v_{2,4}$	$v_{2,5}$	$v_{3,4}$	$v_{4,5}$
v_1	0	1	1	0	0	1	1	0	0	0	0
v_2	1	0	0	1	1	1	0	1	1	0	0
v_3	1	0	0	1	0	0	1	0	0	1	0
v_4	0	1	1	0	1	0	0	1	0	1	1
v_5	0	1	0	1	0	0	0	0	1	0	1
$v_{1,2}$	1	1	0	0	0	0	1	1	1	0	0
$v_{1,3}$	1	0	1	0	0	1	0	0	0	1	0
$v_{2,4}$	0	1	0	1	0	1	0	0	1	1	1
$v_{2,5}$	0	1	0	0	1	1	0	1	0	0	1
$v_{3,4}$	0	0	1	1	0	0	1	1	0	0	1
$v_{4,5}$	0	0	0	1	1	0	0	1	1	1	0

Tabla 6. Matriz de adyacencia de la gráfica T(G).

Resultados

El siguiente algoritmo muestra cómo se debe llevar a cabo el llenado de la matriz de adyacencia de la gráfica $S(G)$ (a esta matriz le llamaremos MSG) a partir de la matriz de adyacencia de la gráfica G. El procedimiento **LlenaMSG()** tiene como entrada una matriz G de tamaño $n \times n$, como valor de retorno arroja una matriz llamada MSG de tamaño $m+n \times m+n$. Este procedimiento hace uso de la subrutina **ObtenerAristas()**. Por la definición de la gráfica $S(G)$, sabemos que solo existen conexiones entre vértices de $V(G)$ y vértices de $E(G)$. Estas conexiones hacen que los primeros n × n elementos de la matriz de adyacencia sean 0, al igual que los últimos $m \times m$ elementos. La matriz MSG_Tem se obtiene con el procedimiento **GraficaSubdivisiónDeG()**, ésta contiene las adyacencias entre vértices de $V(G)$ y vértices de $E(G)$. La matriz $(MSG_Tem)^T$ es la matriz transpuesta de MSG_Tem. La Tabla 7 se muestra cómo se debe de llenar la matriz MSG. Podemos decir que MSG es simétrica si las submatrices $0_{n\times n}$ y $0_{m\times m}$ son simétricas, además MSG_Tem debe ser simétrica a $(MSG_Tem)^T$, todas estas condiciones son verdaderas en MSG.

$0_{n\times n}$	$[\,]_{n\times m} = (MSG_Tem)^T$
$[\,]_{m\times n} = MSG_Tem$	$0_{m\times m}$

Tabla 7. Representación abstracta de la matriz MSG.

PROCEDIMIENTO **LlenaMSG** ( )

Entrada: G, matriz de adyacencia de la gráfica G.
Salida: MSG, matriz de adyacencia de S(G)

n = |V(G)|;
m = |E(G)|;
MSG = $0_{m+n,m+n}$; (Crear la matriz MSG de tamaño $m+n \times m+n$, inicializada en 0)

MSG_Tem = GraficaSubdivisiónDeG (G, n, m);

```
For  i=n; i< m+n; i++
  For  j=0; j< n; j++
          MSG[i][j] = MSG_Tem[i-n][j];
          MSG[j][i] = MSG_Tem[i-n][j];
  End For
End For

Return MSG;
```

Observación 1: El procedimiento Llena*MSG*() retorna la matriz de adyacencia de la gráfica S(G) en tiempo $O(m^2 + 2n^2 + 5mn + 2m)$.

Los dos ciclos anidados del procedimiento **GraficaSubdivisiónDeG**() se encargan de agregar un 1 en caso de que el valor de la variable **j** coincida con uno de los dos valores de la posición de la fila **i**-esima de la matriz Aristas.

PROCEDIMIENTO **GraficaSubdivisiónDeG**()

Entrada: G, n, m (m y n es el número de vértices y de aristas respectivamente de G)
Salida: MSGTem

```
Aristas = ObtenerAristas(G, n, m);

MSG = 0m,n; (Crear la matriz MSGTem de tamaño m × n , inicializada en 0)

For i=0; i<m; i++
  For j=0; j<n; j++
      If Aristas[i][0] == j || Aristas[i][1] == j Then
            MSGTem[i][j] = 1;
      End If
  End For
End For

Return MSGTem;
```

Observación 2: El procedimiento **GraficaSubdivisiónDeG()** retorna la matriz MSGTem en tiempo $O(n^2 + 2mn + 2m)$.

El procedimiento **ObtenerAristas()** tiene como entrada la matriz G de tamaño $n \times n$, y retorna una matriz de tamaño $m \times 2$ llamada Aristas, esta matriz guarda las aristas de la gráfica G. En esta matriz estamos guardando las etiquetas de los vértices que pertenecen al conjunto $E(G)$, esto nos ayudará a conectar vértices del conjunto V(G) y del conjunto E(G).

PROCEDIMIENTO **ObtenerAristas()**

Entrada: G, n, m (n y m es el número de vértices y aristas de G respectivamente)

Salida: Aristas

```
a=0;

Aristas = 0m,2; (Crear la matriz Aristas de tamaño m × 2 , inicializada en 0)

For i=0; i < n; i++
  For j=0; j < n; j++
      If G[i,j] == 1 Then
          Aristas[a,0] = i;
          Aristas[a,1] =j;
          a++;
      End If
  End For
End For

Return Aristas;
```

Observación 3: El procedimiento **ObtenerAristas()** retorna la matriz Aristas en tiempo $O(n^2 + 2m)$.

En el siguiente algoritmo se muestra cómo llevar a cabo el llenado de la matriz de adyacencia de la gráfica $L(G)$ a partir de la matriz de adyacencia de la gráfica G. El procedimiento **GraficaLineaDeG()** recibe como entrada la matriz G de tamaño n × n, retorna como resultado la matriz de adyacencia de la gráfica línea de G la cual se guarda en la matriz con nombre MLG y es de tamaño $m \times m$. Por la configuración del algoritmo es claro que MLG es simétrica.

PROCEDIMIENTO **GraficaLineaDeG()**

```
Entrada: G, n, m (m y n es el número de vértices y de aristas respectivamente de G)
Salida: MLG, Matriz de adyacencia de L(G).

Aristas = ObtenerAristas(G, n, m);

MLG = 0m,m; (Crear la matriz MLG de tamaño m × m , inicializada en 0)

For i=0; i <m;  i++
  For j=0; j < i ; j++
      If Intersección(Aristas[i],Aristas[j]) && i != j)
      Then
            MLG[i,j]=1;
            MLG[j,i]=1;
      End If
  End For
End For
Return MLG;
```

Observación 4: El procedimiento **GraficaLineaDeG()** retorna la matriz de adyacencia de la gráfica L(G) en tiempo $O\left(n^2 + \frac{m(m+3)}{2}\right)$.

PROCEDIMIENTO **Interseccion()**

```
Entrada: Fila_i, Fila_j
Salida: Bolean

  For i=0; i< 2; i++
      For j=0; j<2; j++
            If Fila_i[i] == Fila_j[j] Then
                  Return VERDADERO;
            End If
      End For
End For

Return FALSO;
```

El algoritmo **LlenaMRG()** se encarga de construir la matriz de adyacencia de la gráfica $R(G)$ (a esta matriz le llamaremos MRG), este algoritmo tiene como entrada la matriz de adyacencia de la gráfica G. El algoritmo **LlenaMRG()** hace uso del procedimiento **LlenaMSG** (), sabemos que la matriz de adyacencia de la gráfica $R(G)$ es la unión entre la matriz de adyacencia de la gráfica G y la matriz de adyacencia de la gráfica $S(G)$. Igual que con la matriz de adyacencia de S(G), la matriz de adyacencia de R(G) es cuadrada. Al tener esto en cuenta, la unión de las matrices de G y S(G) es mucho más fácil, solo se tienen que actualizar los primeros n × n elementos de la matriz S(G) por la matriz G. En la Tabla 8 se muestra la representación abstracta de MRG. También podemos afirmar que MRG es simétrica, ya que, G y $0_{m\times m}$ son simétricas y MSG_Tem es simétrica a $(MSG_Tem)^T$.

$[\,]_{n\times n} = G$	$[\,]_{n\times m} = (MSG_Tem)^T$
$[\,]_{m\times n} = MSG_Tem$	$0_{m\times m}$

Tabla 8: Representación abstracta de la matriz MRG.

PROCEDIMIENTO **LlenaMRG( )**

```
Entrada: G, matriz de adyacencia de la gráfica G.
Salida: MRG, matriz de adyacencia de R(G).

n = |V(G)|;

MRG = LlenaMSG(G);  (Crear e inicializar la matriz MRG de tamaño m + n × m + n, mediante el procedimiento LlenaMSG)

For  i=0, i< n, i++
  For  j=0, j<n, j++
      MRG[i][j] = G[i][j];
  End For
End For

Return MRG;
```

Observación 5: El procedimiento **LlenaMRG()** retorna la matriz de adyacencia de la gráfica R(G) en tiempo $O(3n^2 + m^2 + 5mn + 2m)$.

El algoritmo **LlenaMQG()** obtiene la matriz de adyacencia de la gráfica $Q(G)$ (a esta matriz se le nombra MQG) a partir de la matriz de adyacencia de la gráfica G. Este algoritmo hace uso de los procedimientos **LlenaMSG()** y **GraficaLineaDeG**(), la matriz MQG se obtiene de la unión de la matriz de adyacencia de las gráficas $S(G)$ y $L(G)$.

El tamaño de la matriz MQG es de $(m+n) \times (m+n)$. En la Tabla 9 se muestra la representación abstracta de MQG. Podemos afirmar que MQG es simétrica, ya que, la matriz $0_{n\times n}$ es simétrica, lo mismo pasa con MLG y también sucede que MSG_Tem es simétrica a $(MSG_Tem)^T$.

$0_{n\times n}$	$[\,]_{n\times m} = (MSG_Tem)^T$
$[\,]_{m\times n} = MSG_Tem$	$[\,]_{m\times m} = MLG$

Tabla 9: Representación abstracta de MQG.

PROCEDIMIENTO **LlenaMQG( )**

Entrada: G, matriz de adyacencia de la gráfica G.
Salida: MQG, matriz de adyacencia de Q(G).

```
n = |V(G)|;
m = |E(G)|;

MQG = MSG(G); (Crear e inicializar la matriz MQG de tamaño m + n × m + n, mediante el procedimiento LlenaMSG)

MLG = GraficaLineaDeG(G, n, m); (Crear e inicializar la matriz MLG de tamaño m × m, mediante el procedimiento GraficaLineaDeG)

For  i=n; i< n+m; i++
  For  j=n; j<n+m; j++
      MQG[i][j] = MLG[i-n][j-n];
  End For
End For

Return MQG;
```

Observación 6: El procedimiento **LlenaMQG()** retorna la matriz de adyacencia de la gráfica Q(G) en tiempo $O\left(3n^2 + \frac{5m^2}{2} + 5mn + \frac{7m}{2}\right)$.

El procedimiento **LlenaMTG** obtiene la matriz de adyacencia de la gráfica total T(G) la cual es nombrada como MTG, este procedimiento hace uso de los procedimientos **LlenaMQG()**. La matriz MTG es la unión de las matrices G, S(G) y L(G). La matriz MTG tiene un tamaño de $(n+m) \times (n+m)$. Si se hace un análisis similar a los casos anteriores, la matriz MTG quedaría como en la Tabla 10, la cual también es simétrica.

$[\,]_{n\times n} = G$	$[\,]_{n\times m} = (MSG_Tem)^T$
$[\,]_{m\times n} = MSG_Tem$	$[\,]_{m\times m} = MLG$

Tabla 10: Representación abstracta de MTG.

PROCEDIMIENTO **LlenaMTG**( )

Entrada: G, matriz de adyacencia del grafo G.
Salida: MTG, matriz de adyacencia de T(G).

```
n = |V(G)|;
m = |E(G)|;

MTG = LlenaMQG(G);  (Crear e inicializar la matriz MTG de tamaño m + n × m + n, mediante el procedimiento LlenaMQG)

For  i=0; i< n; i++
  For  j=0; j< n; j++
      MTG[i][j] = G[i][j];
  End For
End For

Return MTG;
```

Observación 7: El procedimiento **LlenaMTG()** retorna la matriz de adyacencia de la gráfica T(G) en tiempo $O\left(4n^2 + \frac{5m^2}{2} + 5mn + \frac{7m}{2}\right)$.

Discusión y conclusiones

En este trabajo de investigación logramos construir la matriz de adyacencia de las gráficas S(G), L(G), R(G), Q(G) y T(G) mediante algoritmos computacionales, podemos decir que estos algoritmos se pueden programar en el lenguaje que el lector elija, ya que los algoritmos se escribieron en pseudocódigo. Con estos algoritmos se logra reducir el tiempo de construcción de estas gráficas y también evita el margen de error al momento de estar construyéndolas, es claro que entre más crece el número de vértices y de aristas más complejo se torna la construcción de este tipo de gráficas.

Bibliografía

Ascioglu, M., Cangul, I.N. (2018). Narumi–Katayama index of the subdivision graphs. Journal of Taibah University for science, 12(4), 401-408.

Bondy J.A., Murty U.S.R. (2008). Graph theory. San Francisco, USA: SPRINGER, 651p.

Castro, S.J., Leaños, J., Hernandez, B.L.A. Rosario, C.O. (2020). The Differential on Graph Operator Q (G). Symmetry, 12(5), 751.

Cvetkovi'c, D.M., Doob M., Sachs, H. (1980). Spectra of graphs: theory and application, New York, USA, Academic Press, 368p.

Fomin, F.V., Grandoni, F., Kratsch, D. (2009). A measure and conquer approach for the analysis of exact algorithms. Journal of the ACM, 56, 1–32.

Gaspers, S., Kratsch, D., Liedloff, M., Todinca, I. (2009). Exponential time algorithms for the minimum dominating set problem on some graph classes. ACM Transactions on Algorithms, 6, 1–21.

Hande, S.P., Jog, S.R., Ramane, H.S., Hampiholi, P.R., Gutman, I., Durgi, B.S. (2013). Derived graphs of subdivision graphs. Kragujevac Journal of Mathematics, 37(2), 319-323.

Hedetniemi, S.T., Laskar, R., Pfaff, J. (1986). A linear algorithm for finding a minimum dominating set in a cactus. Discrete applied mathematics, 13(2-3), 287-292.

Hernández, B.L.A. (2020). Propiedades del diferencial en gráficas. Tesis de Doctorado, Unidad Académica de Matemáticas, Universidad Autónoma de Zacatecas, 77p.

Klein, D.J., Yi, E. (2012). A comparison on metric dimension of graphs, line graphs, and line graphs of the subdivision graphs. European journal of pure and applied mathematics, 5(3), 302-316.

Lenzen, C., Pignolet, Y.A., Wattenhofer, R. (2013). Distributed minimum dominating set approximations in restricted families of graphs. Distributed computing, 26(2), 119-137.

Nadeem, M.F., Zafar, S., Zahid, Z. (2015). On certain topological indices of the line graph of subdivision graphs. Applied mathematics and computation, 271, 790-794.

Ranjini, P.S., Lokesha, V., Rajan, M.A., Raju, M. P. (2011). On the Shultz index of the subdivision graphs. Advanced Studies in Contemporary Mathematics, 21(3), 279-290.

Shafique, K.H. (2004). Partitioning a graph in alliances and its application to data clustering, Doctoral dissertation, University of Engineering and Technology, University of Central Florida, 154p.

Sigarreta, J.M., Rodríguez, J.A. (2006). On defensive alliances and line graphs. Applied Mathematics Letters, 19(12), 1345-1350.

Van Rooij, J.M., Bodlaender, H.L. (2011). Exact algorithms for dominating set. Discrete Applied Mathematics, 159, 2147–2164.

Yan, W., Yeh, Y.N. (2009). On the matching polynomial of subdivision graphs. Discrete applied mathematics, 157(1), 195-200.