Grafos Cordales

Problemas de Grafos y Tratabilidad Computacional

Definiciones

Sea $G = (V, E)$ un grafo.

- G es un grafo cordal si para todo ciclo C de al menos 4 vértices, existe una arista entre un par de vértices no consecutivos de C, esto quiere decir que G no tiene como subgrafo inducido a un ciclo inducido C_k con $k \geq 4$.
- La vecindad abierta de un vértice $v \in V$ se denota como $N_G(v) = \{w \in V / (v, w) \in E\}$ y su vecindad cerrada es $N_G[v] = N_G(v) \cup \{v\}$. En caso que el contexto del grafo es claro, se puede obviar el subíndice y quedan como $N(v)$ y $N[v]$, respectivamente.
- Dados 2 vértices no adyacentes a y b del grafo G. Un $a - b$ separador es un subconjunto de vértices $S \subseteq V \setminus \{a, b\}$ tal que a y b están en componentes conexas distintas de $G \setminus S$.
- Un vértice $v \in V$ es simplicial si su vecindad induce un completo.
- Sea $\sigma = [v_1, \cdots, v_n]$ un orden de los vértices de V es un esquema de eliminación perfecta de vértices si v_i es vértice simplicial en $G[v_i, \cdots, v_n]$.

Teorema

Dado un grafo $G = (V, E)$, G es cordal sii todo $a - b$ separador minimal es un completo.

- $\Rightarrow$) Supongamos que existe un $a - b$ separador minimal S no completo para un par de vértices a y b no adyacentes de G. Sean A y B las componentes conexas de $G \setminus S$ que contienen a a y b, respectivamente. Como S es minimal, entonces cualquier vértice $s \in S$, s tiene vecino tanto en A como en B. Como S no es completo, entonces existen $v, w \in S$ tal que $v \neq w$ y $(v, w) \notin E$. Sean G_1 el subgrafo inducido por vértices de A y $\{v, w\}$ y G_2 el inducido por B y $\{v, w\}$. Consideramos un camino mínimo C_1 en G_1 que conecta v con w y otro camino mínimo C_2 en G_2 que une también v con w. Claramente, $C_1 \cup C_2$ es un ciclo inducido de longitud al menos 4. Todos los vértices intermedios de C_1 son de A y hay al menos 1. Similarmente, los vértices intermedios de C_2 son de B y hay al menos 1. Contradicción.

- $\Leftarrow$) Supongamos que G no es cordal, C_k es un ciclo inducido de G con $k \geq 4$ y sus vértices son $v_1, v_2, v_3, v_4, \cdots, v_k$ de acuerdo al orden circular de C_k. Entonces cualquier $v_1 - v_3$ separador minimal S debe contener a v_2 y alguno de $v_4, \cdots, v_k$.Pero S no es completo ya que v_2 es adyacente solamente a v_1 y v_3 entre los vértices de este C_k. Contradicción.

Lema

Todo grafo cordal $G = (V, E)$ posee un vértice simplicial. Es más, si G no es completo entonces tiene 2 vértices simpliciales no adyacentes.

- Claramente, si G es completo cualquer vértice es simplicial. Supongamos que G no es completo y es cierto el lema para grafos cordales con menos vértices. Tomamos 2 vértices no adyacentes $a, b \in V$ y $a - b$ separador minimal S. Por inducción, el subgrafo G_1 inducido por vértices de A y de S tiene o bien (i) G_1 es completo entonces cualquier vértice de A es simplicial en G_1 o bien (ii) G_1 no es completo y tiene un par de vértices simpliciales no adyacentes. Al menos uno de ellos está en A ya que S es completo. Este vértice simplicial a' de A en G_1 es también es simplicial en G ya que $N_G(a') = N_{G_1}(a')$ (ningún vértice de A es adyacente a uno de B en G). Con el mismo argumento, existe otro vértice simplicial b' de B en G. Ambos vértices simpliciales no son adyacentes.

Teorema

Dado un grafo $G = (V, E)$, G es cordal sii G admite un esquema de eliminación perfecta de vértices.

- $\Rightarrow$) Por inducción, para el grafo trivial que tiene un solo vértice v, v es simplicial y $\sigma = [v]$ es el esquema de eliminación perfecta. Supongamos que vale la hipótesis inductiva para grafos cordales con menos vértices. Por el lema anterior, G tiene un vértice simplicial u y $G \setminus \{u\}$ es un grafo cordal con menos vértices, entonces admite un esquema de eliminación perfecta σ' de vértices en $G \setminus \{u\}$. Claramente, $\sigma = [u] \circ \sigma'$ es esquema de eliminación perfecta de vértices de G.

- $\Leftarrow$) Sea $\sigma = [v_1, \cdots, v_n]$ un esquema de eliminación perfecta de vértices de G. Supongamos que G no es cordal, C_k es un ciclo inducido de G con $k \geq 4$ y sea v_i el vértice de este C_k con menor subíndice. Ahora, sean v_j, v_l los dos vecinos de v_i en el C_k con $i < j < l$ y como C_k es un ciclo inducido, v_j y v_l no son adyacentes. Por otro lado v_i es vértice simplicial en $G[v_i, \cdots, v_n]$. Claramente, v_j, v_l están en $G[v_i, \cdots, v_n]$ y siguen siendo vecinos de v_i, deben ser adyacentes entre sí. Contradicción.

Algoritmos de reconocimiento

Pros y contras

- Usando la definición
- Usando la caracterización de $a - b$ separadores minimales
- Usando la caracterización de esquema de eliminación perfecta de vértices
  - intentar con distintos ordenes de los vértices
  - iterativamente encontrar un vértice simplicial e ir eliminándolo. $O(m * n^2)$
  - Observación: un vértice es simplicial en un grafo entonces también lo es en los subgrafos inducidos que lo contienen.

Algoritmos de reconocimiento

- Usar una lista de vértices simpliciales
- $\delta(v, w) = |N[v] \setminus N[w]| = |N[v]| - |N[v, w]|$ para una arista (v, w) como par ordenado. $N[v, w] = N[v] \cap N[w]$
- Usar $\mu(v)$, una variable que guarda iterativamente el valor de $|N[v]|$ para el vértice v y usar $\gamma(v, w)$, una variable que almacena el valor de $|N[v, w]|$. Se puede recalcular $\delta(v, w) = \mu(v) - \gamma(v, w)$ cada vez que haya modificado $\mu(v)$ y/o $\gamma(v, w)$. Cuando $\delta(v, w)$ valga 0, eso implica que todo vecino de v es vecino de w (se dice v es dominado por w). Un vértice es simplicial precisamente cuando es dominado por todos sus vecinos. Podemos utilizar una variable $\beta(v)$ para indicar la cantidad de vecinos de v que no dominan a v. Inicialmente, $\beta(v) = \mu(v) - 1$. Se descuenta 1 cada vez que encontramos un vecino w de v tal que $\delta(v, w)$ pasa a ser 0 o w es eliminado del grafo (w es elegido como vertice simplicial para el esquema).

- Obtener la lista de vértices simpliciales iniciales a partir del cálculo inicial de μ, γ, δ y β.
- Cuando se saca un vértice simplicial s de la lista, hay que actualizar μ, γ, δ y β.
- Momento para agregar vértices simpliciales nuevos a la lista.
- $O(m * n)$

Libro

https://www.ic.fcen.uba.ar/~mlin/docs/PGTC/LibroGolum.7z

La clave es segura para abrir el archivo 7z.

Algoritmo de reconocimiento usando definición

Sea $G = (V, E)$ un grafo y $e = (v, w)$ una arista de G. Existe un ciclo inducido C_k con $k \geq 4$ que contiene a e sii en $G \setminus (N[v] \cap N[w])$ existen $v' \in N[v] \setminus N[w]$ y $w' \in N[w] \setminus N[v])$ en una misma componente conexa.

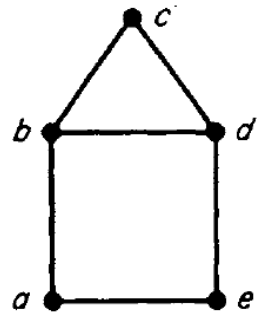
- $\Rightarrow$) Sea $e = (v, w)$ una arista de un ciclo inducido C_k de G con $k \geq 4$. Sean v' el vecino de v en C_k que no es w y w' el vecino de w en C_k que no es v. Es claro que v' no es vecino de w ni w' no es vecino de v porque C_k es ciclo inducido. Cualquier otro vértice de C_k que no están en $\{v, w, v', w'\}$ no es vecino de v ni de w. Por lo tanto, todos los vértices de C_k salvo v y w están en $G \setminus (N[v] \cap N[w])$ y están en una misma componente conexa.

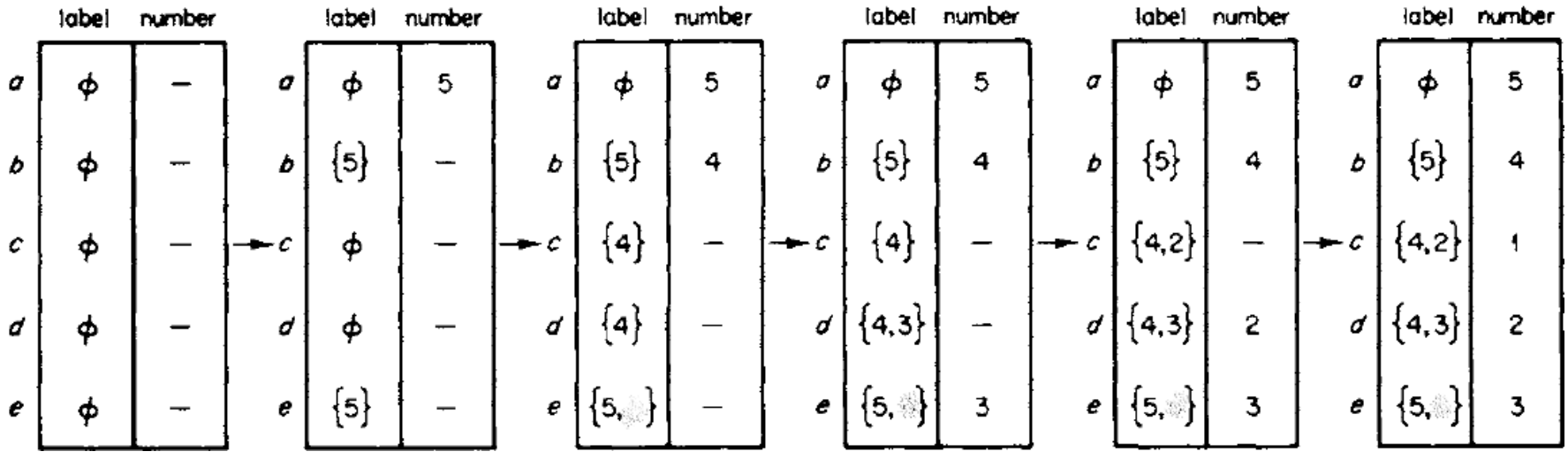
- $\Leftarrow$) Sea H la componente conexa de $G \setminus (N[v] \cap N[w])$ que contiene a $v' \in N[v] \setminus N[w]$ y $w' \in N[w] \setminus N[v]$. Consideramos el subgrafo inducido de G por los vértices de H y v, w y luego quitamos la arista (v, w). Este grafo resultante G' es conexo porque H es conexo, v es adyacente a v' de H y w es adyacente a w' de H. Sea P_k un camino de menor longitud de G' entre v y w. Claramente, P_k es inducido y $k \geq 4$ pués en H no hay vértice que sea vecino de v y de w al mismo tiempo. Podemos agregar la arista (v, w) a P_k y queda un ciclo inducido C_k de G.

Algoritmo LexBFS

for ($i := 1$; $i \leq n$; $i{+}{+}$) {
    $L(i) := \emptyset$;
    $\sigma(i) := 0$;
    $\sigma^{-1}(i) := 0$
}
for ($i := n$; $i \geq 1$; $i - -$) {
    Sea v el vértice no enumerado de mayor label;
    $\sigma(i) := v$;
    $\sigma^{-1}(v) := i$; /* v es enumerado */
    Actualizar label a cada vecino w no enumerado de v
        $L(w) := L(w) \cup \{i\}$
}

LexBFS:ejemplo





Propiedades de los Labels

$L_i(x)$ denota el valor del label de vértice x cuando $\sigma(i)$ se le asigna un vértice (ese vértice es enumerado con el número i).

1. $L_i(x) \leq L_j(x)$ para $j \leq i$. Significa que los labels solamente se pueden aumentar con el tiempo.
2. $L_i(x) < L_i(y) \Rightarrow L_j(x) < L_j(y)$ para $j \leq i$. Una vez que se hayan desempatado los labels de 2 vértices queda inclinada la balanza para el mismo lado de allí en adelante.
3. si $\sigma^{-1}(a) < \sigma^{-1}(b) < \sigma^{-1}(c)$ y $c \in N(a) \setminus N(b)$, entonces existe un vértice $d \in N(b) \setminus N(a)$ con $\sigma^{-1}(c) < \sigma^{-1}(d)$. Si b es enumerado antes que a y a cuenta con el apoyo diferencial de c donde c es enumerado antes que b entonces b debe haber recibido el apoyo diferencial de alguien (d) más influyente (enumerado antes) que c para ser elegido antes que a.

Teorema

Dado un grafo $G = (V, E)$, G es cordal sii el orden σ generado por el algoritmo LexBFS es esquema de eliminación perfecta (EEP).

- $\Rightarrow$) Por inducción. Claramente vale para el grafo trivial. Supongamos que vale para grafos cordales de menor tamaño: el orden σ generado por el algoritmo LexBFS es EEP. Ahora queremos probar para el grafo G con n vértices y $n \geq 2$ que el orden σ generado por el algoritmo LexBFS para este grafo es EEP. Sea $\sigma = [v_1, v_2, \cdots, v_n]$, consideramos ahora $\sigma' = [v_2, \cdots, v_n]$ y σ' es un orden de vértices del subgrafo inducido $G[v_2, \cdots, v_n]$ que es cordal. Este orden σ' también se puede obtener de una corrida del algoritmo LexBFS para $G[v_2, \cdots, v_n]$ haciendo las mismas elecciones de vértices para G salvo v_1 que fue el último para obtener σ. Como $G[v_2, \cdots, v_n]$ tiene menos vértices que G y por hipótesis inductiva, σ' es EEP para $G[v_2, \cdots, v_n]$. Entonces v_i es vértice simplicial para $G[v_i, \cdots, v_n]$, $2 \leq i \leq n$. Solamente hace falta probar que v_1 es vértice simplicial para concluir que σ es EEP.

- $\Leftarrow$) Trivial, si G tiene un esquema de eliminación perfecta entonces es cordal.

Lema

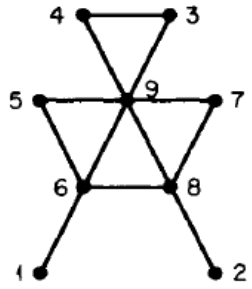
Dados un grafo cordal $G = (V, E)$ con al menos 2 vértices y $\sigma = [v_1, \cdots, v_n]$ el orden generado por el algoritmo LexBFS entonces v_1 es vértice simplicial de G.

- Supongamos $x = x_0 = v_1$ no es simplicial. Sean $x_1, x_2 \in N(x)$ tal que $(x_1, x_2) \notin E$ y con $\sigma^{-1}(x_2)$ mayor posible. Imponemos las siguientes reglas para inducir iterativamente vértices x_j, $1 \leq j$. Sea $x_1, \cdots, x_q$ una secuencia de estos vértices en cualquier momento.
  i. $(x, x_i) \in E \Leftrightarrow i \leq 2$
  ii. $(x_i, x_j) \in E \Leftrightarrow |i - j| = 2$ para $1 \leq i, j \leq q$
  iii. $\sigma^{-1}(x) = 1 < \sigma^{-1}(x_1) < \sigma^{-1}(x_2) < \cdots < \sigma^{-1}(x_q)$
  iv. para $j \geq 3$, se elije el vértice x_j maximizando $\sigma^{-1}(x_j)$ tal que $(x_{j-2}, x_j) \in E$ y $(x_{j-3}, x_j) \notin E$

  Los vértices x_{q-2}, x_{q-1} y x_q $(q \geq 2)$ cumplen la hipótesis de la propiedad (3) como a, b y c, respectivamente. Por lo tanto se puede aplicar la regla (iv) para elegir x_{q+1} tomando $j = q + 1$ y agregarlo a la secuencia. Para $q \geq 3$, veamos que

$(x_{q-3}, x_{q+1}) \notin E$. Supongamos que $(x_{q-3}, x_{q+1}) \in E$. Si $q = 3$, x_4 debe haber tomado el papel de x_2. Lo cual sería una contradicción. Consideramos ahora que $q \geq 4$ entonces x_{q-3}, x_{q-2} y x_{q+1} también cumplen la hipótesis de la propiedad (3) como a, b y c y debe existir un vértice y con $\sigma^{-1}(y) > \sigma^{-1}(x_{q+1})$ tal que $(y, x_{q-2}) \in E$ y $(y, x_{q-3}) \notin E$ contradiciendo que la maximidad de x_q en (iv). Por lo tanto, $(x_{q-3}, x_{q+1}) \notin E$. Las reglas (i) y (ii) se verifican para $x_1, \cdots, x_q$ por HI y $\{x_0, x_1, \cdots, x_q\}$ induce un camino P con extremos del camino x_{q-1} y x_q. Veamos las reglas también se verifican para $x_1, \cdots, x_{q+1}$. Sabemos que x_{q+1} es adyacentes a x_{q-1} y no adyacente a x_{q-2} ni x_{q-3}. Usando la arista (x_{q-1}, x_{q+1}), P se extiende y ahora tiene como extremos x_q y x_{q+1}. Falta ver que el camino extendido es inducido. Supongamos que x_{q+1} es adyacente a otro vértice x_j entonces forma un ciclo (no necesariamente inducido). Tomando el ciclo de menor longitud de este tipo. Ese ciclo es inducido y tiene longitud al menos 4 pero el grafo es cordal. Lo cual es una contradicción. Entonces siempre es posible usar este procedimiento para incorporar nuevos vértices a la secuencia y no para. Pero el grafo G es finito, absurdo.

LexBFS: implementación



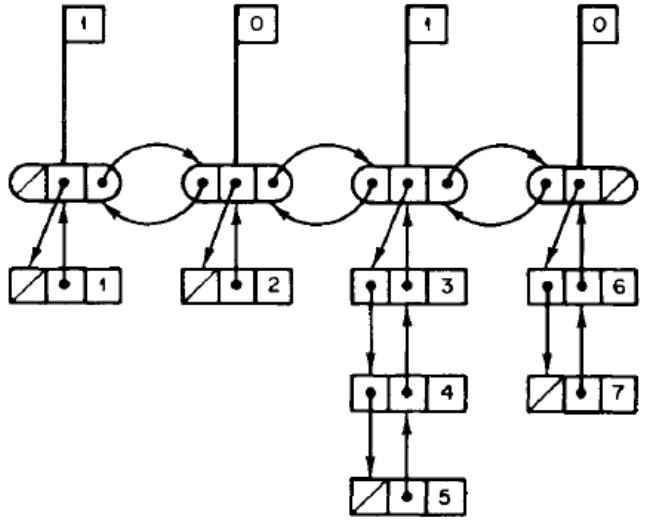
(a)

$Q_9 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$Q_8 = \{1, 2\} < \{3, 4, 5, 6, 7, 8\}$

$Q_7 = \{1\} < \{2\} < \{3, 4, 5\} < \{6, 7\}$

(b)



(c)

- Una lista doblemente encadenada donde cada elemento de esta lista es una clase de equivalencia de vértices no enumerados que tienen el mismo label, tiene un flag que toma valor true o false para indicar si está generada una clase nueva inmediatamente mayor para mudar a sus miembros que son vecinos del vértice seleccionado (inicialmente debe estar en false) y tiene un puntero a una lista también doblemente encadenada que contiene a los vértices de la clase. Los elementos de esta última lista debe tener la info. del vértice, punteros al elemento anterior y al siguiente, debe tener un puntero al elemente que representa a la clase que pertenece. Las clases están ordenadas de mayor a menor en la lista principal.
- Cada vértice no enumerado debe tener el puntero al elemento que lo representa dentro la estructura anterior.
- Una lista auxiliar para ir guardando referencia de las clases que tienen flag en true y para después apagarlo.

Motivación de LexBFS y MCS

From Lemma 4.2 we learned that the Fulkerson–Gross recognition procedure affords us a choice of at least two vertices for each position in constructing a perfect scheme for a triangulated graph. Therefore, we can freely choose a vertex v_n to *avoid* during the whole process, saving it for the last position in a scheme. Similarly, we can pick any vertex v_{n-1} adjacent to v_n to save for the $(n-1)$st position. If we continued in this manner, we would be constructing a scheme *backwards*! This is exactly what Leuker [1974] and Rose and Tarjan [1975] have done in order to give a linear-time algorithm for recognizing triangulated graphs. The version presented in Rose, Tarjan, and Leuker [1976] uses a *lexicographic* breadth-first search in which the

In an unpublished work, Tarjan [1976] has shown another method of searching a graph that can be used to recognize triangulated graphs. It is called *maximum cardinality search* (MCS), and it is defined as follows:

MCS: The vertices are to be numbered from n to 1.
The next vertex to be numbered is always one which is adjacent to the most numbered vertices, ties being broken arbitrarily.

Algoritmo $TesteoEEP(G = (V, E), \sigma)$

Para cada vértice $v \in V$, $A(v) \leftarrow \emptyset$
for $(i := 1;\ i \leq n-1;\ i{+}{+})$ {
    $v \leftarrow \sigma(i)$;
    $X_v \leftarrow \{x \in N(v) / \sigma^{-1}(v) < \sigma^{-1}(x)\}$;
    if $|X_v| \geq 2$ then {
        $u \leftarrow \sigma(min\{\sigma^{-1}(x) / x \in X_v\})$;
        $A(u) \leftarrow A(u) \circ (X_v \setminus \{u\})$
    }
    if $A(v) \setminus N(v) \neq \emptyset$ then {
        devolver Falso
    }
}
devolver Verdadero

Cuello de botella

Verificar $A(v) \setminus N(v) \neq \emptyset$ en forma eficiente.

adjacent to v, can be done in $O(|\mathrm{Adj}(v)| + |\mathrm{A}(v)|)$ time by using an array TEST of size n initially set to all zeros as follows:

```
8. begin
     for w ∈ Adj(v) do TEST(w) ← 1;;
     for w ∈ A(v) do
       if TEST(w) = 0 then
         return "nonempty";
     for w ∈ Adj(v) do TEST(w) ← 0;;
     return "empty";
   end
```

Thus, the entire algorithm can be performed in time and space proportional to

$$|V| + \sum_{v \in V} |\mathrm{Adj}(v)| + \sum_{u \in V} |A(u)|,$$

Teorema

El algoritmo *TesteoEEP* es correcto para determinar si σ, un orden de los vértice del grafo $G = (V, E)$, es o no EEP.

- Si el algoritmo devuelve Falso, es porque $A(v) \setminus N(v) \neq \emptyset$ para algún vértice $v \in V$. Es decir que existe un vértice $w \in A(v)$ tal que $w \notin N(v)$. Ahora, w está en $A(v)$ porque estaba en X_z de alguna iteración anterior donde v jugaba el paper de u y v era otro vértice z. Tanto w como u son vecinos de z y además $\sigma^{-1}(z) < \sigma^{-1}(u) < \sigma^{-1}(w)$. Claramente, z no es vértice simplicial de $G[z = \sigma(\sigma^{-1}(z)), \sigma(\sigma^{-1}(z) + 1), \cdots, \sigma(n)]$ porque u y w están en este grafo y no son adyacentes. Así que está bien que devuelva Falso.

- Ahora si el algoritmo devuelve Verdadero y supongamos que no es cierto. Por lo tanto, existe $i = max\{j/\sigma(j)$ no es vértice simplicial en $G[\sigma(j), \cdots, \sigma(n)]\}$ y $v = \sigma(i)$. Claramente, para $j > i$ y $\sigma(j)$ es vértice simplicial en $G[\sigma(j), \cdots, \sigma(n)]$. Sea z, w un par de vecinos de v en $G[\sigma(i), \cdots, \sigma(n)]$ que no son adyacentes y $\sigma^{-1}(z) < \sigma^{-1}(w)$. Es claro que en la iteración i, z y w están en X_v y $u \neq z$ sino w debe estar en $A(z)$ y hubiera dado $N(z) \setminus A(z) \neq \emptyset$ y retornaría False. De acuerdo la forma de elegir u, $\sigma^{-1}(v) < \sigma^{-1}(u) = j < \sigma^{-1}(z) < \sigma^{-1}(w)$ y z, w deben estar en $A(u)$ y estos 2 vértices deben ser vecinos de u ya que sino daría $N(u) \setminus A(u) \neq \emptyset$ (retornaría False). Por otro lado, u es vértice simplicial en $G[\sigma(j), \cdots, \sigma(n)]$ y z, w al estar allí, deben ser adyacentes, contradicción.

Observaciones

- ¿Cómo extender el algoritimo lineal para generar certificados tantos positivos como negativos?
- $X_v \cup \{v\}$ es completo del grafo G en cada iteración del algoritmo TesteoEEP si el resultado es positivo, inclusive podemos agregar una iteración más con $i = n$. ¿Todo clique de G es uno de estos completos?
- Supongamos que σ es EEP, ¿Cómo sabemos si $X_v \cup \{v\}$ es clique (completo maximal)?
- Si σ es EEP. ¿Qué significa $X_v \setminus \{u\}$ fue agregado a $A(u)$ en alguna iteración de TesteoEEP? ¿Vale la vuelta? ($v = \sigma(i)$, $u = \sigma(j)$, $i < j$ y
$\{u\} \subseteq N_{G[\sigma(i),\cdots,\sigma(n)]}(v) \subseteq N_{G[\sigma(j),\cdots,\sigma(n)]}[u])$

Ejercicios

1. Sean $G = (V, E)$ un grafo, $S \subset V$ es un completo de G y $G \setminus S$ es disconexo. Sean $G_1 = (V_1, E_1), \cdots, G_k = (V_k, E_k)$ sus componentes conexas, entonces probar
   - $\omega(G) = max\{\omega(G[S \cup V_1]), \cdots, \omega(G[S \cup V_k])\}$
   - $\chi(G) = max\{\chi(G[S \cup V_1]), \cdots, \chi(G[S \cup V_k])\}$
2. Probar si $G = (V, E)$ es un grafo cordal entonces $\omega(G) = \chi(G)$.
3. ¿Cómo se puede listar en forma eficiente los cliques de un grafo cordal G? ¿Sirve para calcular $\omega(G)$ y $\chi(G)$?
4. ¿Cómo se puede colorear óptimamente los vértices de un grafo cordal G?
5. Dado un grafo $G = (V, E)$, se denota como $\alpha(G)$ la cardinalidad de mayor conjunto independiente de G y $\kappa(G)$ el tamaño de menor clique-cover de G (un clique-cover de G es una partición de vértices de G en completos disjuntos). Más precisamente, $\alpha(G) = \omega(\overline{G})$ y $\kappa(G) = \chi(\overline{G})$. Probar que dado un grafo cordal G, $\alpha(G) = \kappa(G)$.