Puntos de articulación y componentes biconexas

Colectivo Estructuras de Datos y Algoritmos

Octubre 2021

1. Determine si las siguientes afirmaciones son verdaderas o falsas. Justifique su respuesta.

   a) ___ En un grafo no dirigido una arista puente siempre une dos puntos de articulación.

   b) ___ La condición necesaria y suficiente para que un vértice v sea punto de articulación en un grafo no dirigido G es que, en un árbol abarcador en profundidad G_π, el vértice v tenga un hijo w tal que $low[w] \geq low[v]$.

   c) ___ Un grafo no dirigido es $k-$conexo si y solo si no tiene puntos de articulación.

   d) ___ Hallar las aristas puente de un grafo no dirigido equivale a hallar las aristas que nunca serán clasificadas como **arista de retroceso** en todo recorrido *DFS*.

2. Implemente un algoritmo que permita determinar si un grafo no dirigido y conexo $G =< V, E >$ es triconexo en $O(|V| * |E|)$. Justifique complejidad temporal y correctitud de su algoritmo.

Respuesta

De la conferencia se pueden extraer los siguientes planteamientos:

a) $G =< V, E >$ es triconexo si es biconexo y al eliminar del mismo cualquier conjunto de 2 vértices con las aristas correspondientes, el grafo resultante es conexo.

b) $G =< V, E >$ es triconexo si para todo $v \in V(G)$ se cumple que $G^{'} =< V - v, E - E(v) >$ es biconexo.

c) $G =< V, E >$ es biconexo si es conexo y no tiene puntos de articulación.

En conferencia se tiene un algoritmo para grafos conexos que permite computar una lista con los puntos de articulación y una lista con las aristas puentes. Definamos este algoritmo como **PA_DFS**.

El siguiente pseudocódigo brinda solución al problema apoyándose en los razonamientos anteriores:

```
def is_biconex(G):
    if not is_connected(G):
        return False
    ap, bridges = PA_DFS(G, V(G)[0])
    return |ap| == 0

def is_triconex(G):
    if not is_biconex(G):
        return False
    for v in V(G):
        Gp = G - v
        if not is_biconex(Gp):
            return False
    return True
```

Para analizar la complejidad temporal del algoritmo **is_triconex** se tienen dos casos luego de aplicar la regla de la suma y el producto:

a) G no es biconexo: $O(|V| + |E|)$

b) G es biconexo: $O(|V|(|V|+|E|)) \Rightarrow O(|V|^2+|V||E|)$. Como G es conexo entonces $|V| \leq |E|$, por tanto $|V|^2+|V||E| \leq 2*|V||E|$

Aplicando la regla de la suma se tiene que este algoritmo tiene complejidad temporal $O(|V||E|)$

3. Implemente un algoritmo que dado un grafo conexo y no dirigido $G =< V, E >$ determine en $O(|V|+|E|)$ una secuencia en que se pueden ir eliminando todos los vértices de G de manera tal que nunca se desconecte la parte de G que va quedando. Explique por qué su algoritmo es correcto y justifique su complejidad temporal.

Respuesta

La idea para resolver este ejercicio se basa en una relajación del problema. Se analizan 2 casos:

a) G es un árbol :
Al eliminar una hoja en ese árbol no se afecta la conexidad del grafo resultante y el mismo sigue siendo un árbol. (queda pendiente su demostración, se basa en la idea de que las *degree* de una hoja es 1)
Siguiendo una idea inductiva este planteamiento nos da un algoritmo, basado en que todo árbol tiene un vértice con *degree* 1 (El caso base sería un árbol con un solo vértice, el cual se puede eliminar. En el caso de un árbol con n vértices, se sabe que siempre existe un nodo con *degree* 1 que puede ser eliminado y el grafo resultante sigue siendo un árbol de $n-1$ vértices, donde puedes apoyarte en la hipótesis.)

b) G no es un árbol:
Se aplica el análisis anterior sobre un árbol abarcador de G resultado de un BFS o DFS.

4. Implemente un algoritmo con complejidad temporal $O(|E|)$ que le asigne a cada arista e de un grafo no dirigido y conexo G, un número positivo $Num(e)$ tal que $Num(e) = Num(e')$ si y solo si e y e' pertenecen a la misma componente biconexa. Justifique la complejidad temporal y explique de forma intuitiva por qué su algoritmo funciona.

5. Sean $G =< V, E >$ un grafo no dirigido y conexo y $a, b \in V$. Se dice que un nodo $v \in V$ es $a-b$ *crítico* si v se encuentra en todo camino de a a b en G . Implemente un algoritmo que permita determinar en $O(|V|+|E|)$ el conjunto de nodos $a-b$ *críticos* dados G, a y b. Explique por qué su algoritmo funciona correctamente y justifique la complejidad temporal.

Respuesta

Si $< a, b >\in E(G)$ el conjunto *a-b críticos* es el vacío.
En otro caso:
Sea C el camino entre a y b resultado del $PA\_DFS(G, a)$, donde para cada nodo $v \in V(G)$ se tiene correctamente calculado el low, d y f.

$$C = \{a, c_1, ..., c_t, b\},\ 1 \leq t \leq n-2$$

La propia definición de *a-b críticos* plantea:

a) $v \in$ *a-b críticos* $\Rightarrow v \in C$

b) $v \in$ *a-b críticos* $\Rightarrow v \in AP(G)$

c) $c_i \in$ *a-b críticos* $\Leftrightarrow low[c_{i+1}] \geq d[c_i]$ (pendiente de demostración)

El siguiente pseudocódigo brinda solución al problema apoyándose en los razonamientos anteriores:

```
def ab_critico(G, a, b):
    d, f, low, pi = PA_DFS(G, a)
    s = []
    p = pi[b]
    while p != a:
        if(low[b] >= d[p]):
            s.push(p)
        b = p
        p = pi[b]
    return s
```

6. Dado un grafo no dirigido G decimos que los caminos simples $(u, a_1), (a_1, a_2)...(a_p, v)$ y $(u, b_1), (b_1, b_2)...(b_q, v)$ son vértice disjuntos si los dos son caminos válidos en G y los conjuntos $\{a_1, a_2, ..., a_p\}$ y $\{b_1, b_2, ..., b_q\}$ son disjuntos. Demuestre que G no tiene puntos de articulación si y solo si para cada par de vértices u y v de G existen dos caminos vértice disjuntos de u a v.

7. Se desea implementar una estructura de datos que dado un grafo no dirigido y conexo $G =< V, E >$ permita realizar consultas de la forma $CANT\_PUENTES(G, a, b)$ para determinar la cantidad de aristas puentes que hay en todo camino simple de a a b.

   a) Sean $G =< V, E >$ un grafo no dirigido y conexo y a, $b \in V$. Demuestre que todos los caminos simples entre a y b tienen la misma cantidad de aristas puentes.

   b) Diseñe una estructura de datos que permita responder a estas consultas en $O(P)$, donde P es la cantidad de aristas puentes de G. La complejidad temporal de inicializar tal estructura debe ser $O(|V| + |E|)$. Explique la correctitud y complejidad temporal de la implementación de los métodos de su estructura de datos.

   Hint

   1) $e \in E(G)$ es puente $\Leftrightarrow$ No pertenece a ningún ciclo en G

   2) Sean:
   bridges el conjunto de aristas puentes de G
   $G^{'} = G - bridges$
   $cc^{'}$ las componentes conexas de $G^{'}$
   $Gb =< Vb, Eb >$ donde Vb representa cada componente conexa de $G^{'}$

   $$< Vb_x, Vb_y > \in Eb \Leftrightarrow \exists e =< u, v > \in E(G) \text{ tal que } cc[u] = x \text{ y } cc[v] = y$$

   Demuestre que Gb es un árbol.