UNIVERSIDAD AUTÓNOMA DE MÉXICO
Facultad de Ciencias

Autores:
Fernanda Villafán Flores
Fernando Alvarado Palacios
Adrián Aguilera Moreno

Gráficas y Juegos

Tarea 9 y 10

Puntos Extra

1. (2 puntos) Sea G una gráfica conexa y e una arista de G que no sea un lazo. Exhiba una biyección entre el conjunto de árboles generadores de G que contienen a e y el conjunto de árboles generadores de G/e.

   Demostración: Sea G un conjuntos de arbloes generadores y T⊂G tal que e∈T y T'⊂G tal que T'=G\e.

   Definamos la funcion f: T→T' tal que si g∈T $\implies$ f(g)=g\e.

   Observemos que como g∈T (g es un arbol generador) $\implies$ f(g) tambien será un arbol generador $\implies$ la función f preseva conexidad.

   Pd) f es inyectiva

   Sea g_1, g_2 $\implies$ si f(g_1)=f(g_2) $\implies$ el subconjunto A y A' de aristas en G que induce a f(g_1) y f(g_2) respectivamente son iguales $\implies g_1 = A \setminus e = A' \setminus e = g_2 \implies g_1 = g_2$.

   Por lo tanto f es inyectiva

   Pd) f es suprayectiva

   Sea b∈T' $\implies$ por definicion b=b\e $\implies$ sea w un vértice que une a las aristas a_1 y a_2 $\implies$ "Partimos a w en dos vértices u,v" y en medio de u y v colocamos a e y denostamos a este nuevo árbol como g_3 $\implies$ existe $g_3 \in T$ tal que f(g_3)=b.

   Por lo tanto f es sobreyectiva.

   Por lo tanto f es un biyección.

   □

2. (2 puntos) Sea T un árbol de DFS de una gráfica conexa no trivial G, y sea v la raiz de un bloque B de G. Demuestre que el grado de v en $T \cap B$ es uno.

   Demostración: Observemos el caso particualar para k_2, este caso siempre se cumple sin importar en que vértice se jecute DFS, nuestro árbol siempre será de tipo k_2, donde los dos vértices tendran grado 1.

   Sea G una gráfica conexa con al menos 3 vértices y v∈V $\implies$ ejecutando DFS en v y s.p.g supongamos v es la raiz de algun bloque T de G. $\implies$ si w es el primer vecino del mismo bloque de v que visitamos.

   Como para cualesquiera tres vértices de un bloque de G existe una trayectoria que una a cualesquiera dos de ellos y que no pasa por el tercero, entonces siempre podemos garantizar que el siguiente vértice que visitemos después de w no saldrá de v. Además DFS ya no agrega a la pila los vértices previamente visitados. Así, en el árbol DFS de G, $G \bigcap T$ sólo tendrá los vértices de T y ahí la raíz tendrá $d(v) = 1$ por lo mencionado anteriormente.

   □

3. (2 puntos) Si f es la función de tiempo de entrada del algoritmo DFS, defina $f^*\colon V \to \mathbb{N}$ de la siguiente forma. Si algún ancestro propio de v puede ser alcanzado desde v mediante una trayectoria dirigida que consista de flechas del árbol (posiblemente ninguna) seguida de una flecha que no está en el árbol (que va hacia arriba), $f^*(v)$ se define como el menor valor de f de un ancestro de este tipo; si no, $f^*(v) = f(v)$. Observe que un vértice v es la raíz de un bloque si y sólo si tiene un hijo w tal que $f^*(w) \geq f(v)$. Modifique el algoritmo DFS para que regrese los vértices de corte y los bloques de una gráfica conexa.

Primero describiremos un algoritmo que encuentra los vértices de corte de una gráfica.

Algorithm 1: DFS-CutVertex

```
Input: Una gráfica G, un vértice distinguido r y el tiempo t de entrada de vértice en
  cuestión.
Output: Una colección C con los vértices de corte.
i ← t;
i ← i + 1;
C ← [];
f(r) ← i, f*(r) ← f(r)
while r tenga vecinos do
    Sea v un vecino de r.
    if v no está visitado then
        p(v) ← r
        C ← DFS-CutVertex(G, v, i);
        if r no tiene padre, y tiene mas de dos hijos then
            añadir r a la colección C;
        end
        else
            f*(r) ← min(f*(r), f*(v));
            if f*(v) ≥ f(r) then
                añadir r a la colección C.
            end
        end
    end
    else if v no es el padre de r y f(v)<f(r) then
        f*(r) ← min(f*(r), f(v))
    end
end
i ← i + 1;
l(r) ← i;
return C;
```

Nótese que el tiempo t en la primer ejecución es igual a 0, como el código es recursivo, eventualmente nos servirá introducir un nuevo tiempo que corresponda a la entrada del vértice en cuestión a la pila. Una vez teniendo el algoritmo para obtener los vértices de corte de una gráfica, podemos usarlo en un nuevo algoritmo para encontrar tanto los vértices de corte, junto con los bloques en una gráfica G conexa.

Algorithm 2: DFS-CutVertexBlock.

```
Input: Una gráfica G, un vértice distinguido r y el tiempo t de entrada de vértice en
  cuestión.
Output: Una colección C de vértices de corte y un conjunto B de bloques de la gráfica G.
S ← []; i ← t;
i ← i + 1;
C ← []; B ← ∅;
marcar a r como visitado.
f(r) ← i, f*(r) ← f(r);
while r tenga vecinos do
    Sea v un vecino de r.
    if v no ha sido visitado then
        p(v) ← r;
        metemos a la arista (r, v) a S en el tope;
        C ← DFS-CutVertex(G, v, i);
        f*(r) ← min(f*(r), f*(v));
        if f*(v) ≥ f(u) then
            Sacamos el tope de la pila y lo insertamos en S, tantas veces hasta que saquemos
              a (u, v) y estás serán las aristas de una componente b y añadimos b a B.
        end
    end
    else if v no es el padre de r y f(v)<f(r) then
        Metemos a la arista (r, v) en S;
        f*(r) ← min(f*(r), f(v));
    end
end
i ← i + 1;
l(r) ← i;
return C, B;
```

Así obtenemos los vértices de corte y los bloques de una gráfica G conexa.

4. (2 puntos) Sea T un árbol óptimo en una gráfica conexa ponderada (G, w) (con pesos positivos), y sean x y y vértices adyacentes en T. Demuestre que la trayectoria $xTy = xy$ es una xy-trayectoria de peso mínimo en G.

Demostración: (Demostración por reduccion al absurdo)

Sea (G,w) una gráfica conexa ponderada, T un árbol optimo y x,y $\in$V(G,W), supongamos que xTy no es una trayectoria optima en (G,w) $\Longrightarrow$ Existe t_0 tal que t_0 sea un árbol generador, t_0 sea diferente a T $\Longrightarrow$ y la trayectoria optima sea xt_0y $\Longrightarrow$ t_0 tiene mayor peso que T, ya que la trayectoria xTy es mas pesada $\Longrightarrow$ que T no es un arbol optimo (Lo que es una contradiccion a nuestra hipotesis).

Por lo tanto xTy es una trayectoria optima.

□

5. (2 puntos) Demuestre que si todos los pesos de una gráfica ponderada G son distintos, entonces G tiene un único árbol óptimo.

Demostración: Supongamos a G una gráfica ponderada con el costo de sus aristas distintos entre si, sean T_1 y T_2 arboles tales que

$$T_1 \neq T_2$$

y ambos son arboles óptimos en G (P.D., la unicidad de T_1), entonces consideremos a e una arista de peso mínimo que está contenida en T_1 o T_2 (esto por que las aristas tienen pesos distintos entonces solo habrá una con peso mínimo), llamémosla e_1, luego s.p.g. que e_1 está contenida en T_1.

$$\Rightarrow e_1 \notin T_2$$

por lo que si unimos $e_1 + T_2$ es claro que existirá al menos un ciclo C el cual contendrá al menos una arista digamos e_2 que no está contenida en T_1 (Esto por que si lo estuviera, entonces T_1 formaría un ciclo con e_2, pero T_1 es un árbol).
Entonces llegamos a que e_1 y e_2 contenidas en T_1 o T_2 por construcción de las mismas tenemos que

$$w(e_1) < w(e_2)$$

luego podemos notar que se si

$$(T_2 \cup e_1) - e_2$$

es claro que genera la construcción de T_1 por la construcción de T_1 y T_2 implica que

$$W(T_1) < W(T_2)!!$$

$$\therefore \quad G \text{ es un árbol óptimo único.}$$ □

6. (2 puntos) Modifique el algoritmo de Borůvka-Kruskal para que en cada iteración vértices en la misma componente del bosque F reciban el mismo color y vértices en componentes distintas reciban colores distintos.

   A continuación se muestra la modificación del algoritmo de Borůvka-Kruskal:

Algorithm 3: Borůvka-Kruskal-Coloration

Input: G una gráfica ponderada
Output: Un bosque óptimo $B = (F, V)$ de G donde cada árbol tiene una coloración distinta.
1. $F \leftarrow \emptyset$, $w(B) \leftarrow \emptyset$, $i \leftarrow 0$;
2. **for** $v \in V_G$ **do**
    3. asignarle $i++$ como color a v.
end
4. **while** exista una arista $e \in E - F$ tal que $F \cup \{e\}$ induce a un bosque **do**
    5. Elegir e de peso mínimo con dicha característica
    6. **if** los extremos de e tienen colores diferentes **then**
        7. Asignamos como el color de ambas, al color mas mínimo entre las dos.
        8. F←F $\cup\{e\}$, $w(B) \leftarrow w(B) + w(e)$
    end
    .
end
return $(F, w(T))$;

7. (2 puntos) Demuestre que el problema de encontrar un árbol generador de peso máximo en una gráfica conexa puede resolverse eligiendo iterativamente una arista de peso máximo, con la condición de que la subgráfica resultante siga siendo un bosque. (Proponga un algoritmo y demuestre que es correcto.)

8. (2 puntos) Escriba una versión del algoritmo BFS para digráficas. Utilice esta versión de BFS dirigida para describir un algoritmo que encuentre un ciclo dirigido de longitud mínima en una digráfica. Su versión dirigida de BFS debe de correr en tiempo $\mathcal{O}(|V| + |E|)$, y el algoritmo para encontrar el ciclo dirigido más corto debe correr en tiempo a lo más $\mathcal{O}(|V|^2 + |V||E|)$.