Tercer examen parcial

AA 361075 Carlos Eduardo Sánchez Torres

12 de noviembre de 2021

1. Existen dos tipos de luchadores profesionales: los “babyfaces” (técnicos) y los “heels” (rudos). Entre cada par de luchadores puede o no existir una rivalidad. Suponga que tenemos n luchadores profesionales y una lista de r pares de luchadores para los cuales existe rivalidad. Proporcione un algoritmo de tiempo O(n+r) que determine si es posible etiquetar a los n luchadores ya sea como “babyfaces” o “heels” tal que las rivalidades siempre sean entre luchadores de diferente tipo. De ser posible ese etiquetado, su algoritmo debe proporcionar el etiquetado como salida, de lo contrario debe dar como salida FALSE. (25 puntos)

```
fight = {BABYFACE: HEEL, HEEL: BABYFACE}

SearchFighters(G):
  for each vertex u in G.V
    u.color = WHITE
    u.d = inf
    u.pi = nil
  s = GetRandomVertex(G)
  s.color = BABYFACE
  s.d = 0
  Q = Queue()
  Q.enqueue(s)
  while not Q.empty()
    u = Q.dequeue()
    for each v in G.adj[u]
       if v.color == u.color
         return FALSE
       if v.color == WHITE
            v.color = fight[u.color]
            v.d = u.d + 1
            v.pi = u
            Q.enqueue(v)
  return G
```

Listing 1: BabyFaces vs Heels

Análisis. Sabemos que $|V| = n$ y $|E| = r$, también que el máximo de coste de la inicialización es $O(V)$ y por análisis amortizado las operaciones de encolar y desconsolar toman tiempo $O(1)$, la suma del tamaño de las listas adjuntas es $\Theta(E)$, entonces el tiempo total esta dado por $O(E)$, por lo tanto, $O(V + E) = O(n + r)$.

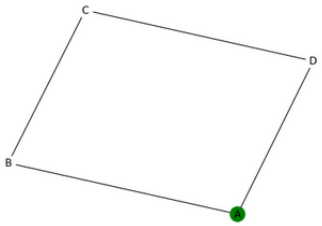

Figura 1: Verdes: técnicos; rojos: rudos; blanco: por determinar.

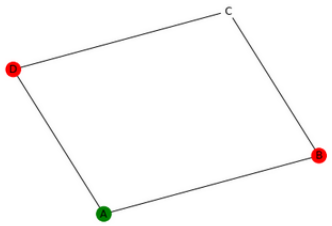

Figura 2: Verdes: técnicos; rojos: rudos; blanco: por determinar.

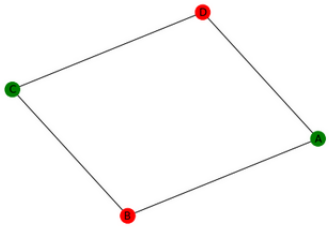

Figura 3: Verdes: técnicos; rojos: rudos; blanco: por determinar.

2. El profesor Borden propone un nuevo algoritmo divide y vencerás para calcular el árbol de esparcimiento mínimo que consiste en lo siguiente: (25 puntos)

Dividir: Dado un grafo dirigido G = (V, E), crear una bipartición de los vértices en dos conjuntos V_1 y V_2 tal que $|V_1|$ y $|V_2|$ difieran en máximo 1.

Vencer: Sea E_1 el conjunto de aristas que inciden en V_1 y E_2 el conjunto de las que inciden en V_2. Resolver recursivamente encontrando el árbol de esparcimiento mínimo para los sub-grafos G_1 = (V_1, E_1) y G_2 = (V_2, E_2).

Combinar: Seleccionar la arista de menor costo del conjunto E que cruce el corte (V_1, V_2) y utilícela para unir los dos árboles de esparcimiento mínimo resultantes de resolver los dos subproblemas para crear uno solo.

Demuestre si este algoritmo calcula correctamente el árbol de esparcimiento mínimo, o proporcione un contraejemplo para el cual falle.
Este algoritmo es incorrecto: Sea el grafo dirigido $G = (\{(A, B, C)\}, \{(A, C, 2), (A, B, 1), (B, C, 1)\})$ donde el tercer parámetro indica el peso de la arista. Siguiendo el algoritmo:

$$G_1 = (V_1, E_1) = G_1 = (\{A, C\}, \{(A, C, 2)\}) \tag{1}$$

$$G_2 = (V_2, E_2) = G_2 = (\{B\}, \{\}) \tag{2}$$

Seleccionamos la arista y nos queda como árbol de esparcimiento:

$$T = (\{A, B, C\}, \{(A, C, 2), (A, B, 1)\}) \tag{3}$$

Donde el peso total es $\sum_{i=1}^{2} w_i = 2 + 1 = 3$. Sin embargo, el MST es $T = (\{A, B, C\}, \{(A, B, 1), (B, C, 1)\})$, con peso total $\sum_{i=1}^{2} w_i = 1 + 1 = 2$. Por lo tanto, el algoritmo falla.

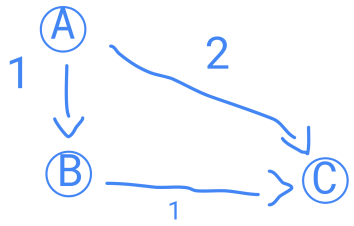

Figura 4: G

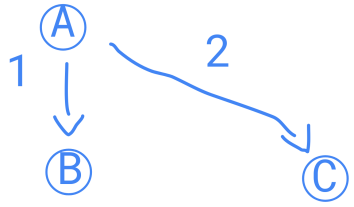

Figura 5: T por el algoritmo de Borden.

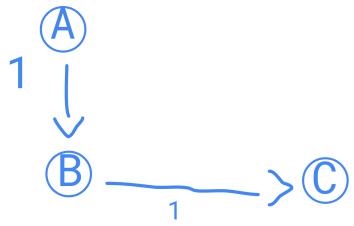

Figura 6: MST

3. Dado un grafo dirigido G = (V, E) con función de costo w: E $\rightarrow$ R, sea un árbol de esparcimiento mínimo un subconjunto acíclico $T \subseteq E$ que conecta a todos los vértices y cuyo costo es mínimo. Si A es un subconjunto de E que se encuentra incluido en algún árbol de esparcimiento mínimo para G: (20 puntos) ¿Qué significa que una arista sea segura para A? Proporcione un grafo dirigido a partir del cual ejemplifique un sub-conjunto de aristas diferentes, un corte que lo respete, y diga cuál es la arista ligera cruzando ese corte.

Se dice que una arista es segura para A si cumple el teorema 23.1 de [1]. Siendo A nuestro árbol de esparcimiento, necesitamos añadir nuevas aristas que cumplan ciertas condiciones para que A sea el árbol de esparcimiento mínimo, a estas aristas se le denomina seguras por darnos la confianza de que: no formara ciclos -solo una arista para cruzar el corte (S,V-S)-, el peso son los mínimos -aristas ligeras-, y no se repiten las aristas porque ninguna de ellas cruza el corte -el respeto de A, para dar paso a $A \cup (u, v)$-.

Para el ejemplo puede verse en la figura 7: $S = \{(A, C), (B, D)\} \subseteq E$ y $(\{A, C\}, \{B, D\})$ un corte que lo respeta y (C, D) la arista ligera que cruza el corte.

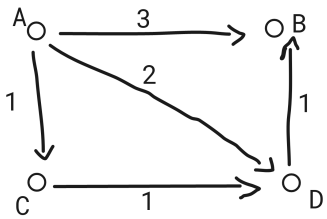

Figura 7: $G = (\{A, B, C, D\}, \{(A, B), (A, D), (A, C), (C, D), (D, B)\})$

4. El profesor Newman propone una nueva demostración para la corrección del algoritmo de Dijkstra. Afirma que el algoritmo relaja las aristas de cada ruta más corta en el grafo en el orden en el que aparecen en la ruta, y por lo tanto la propiedad de la relajación de la ruta (Lemma 24.15) aplica para todos los vértices a los que tenemos acceso desde la fuente. Demuestre que el profesor está en un error proporcionando como contraejemplo un grafo dirigido para el cual el algoritmo de Dijkstra podría relajar las aristas de una ruta más corta en desorden. (25 puntos)

El profesor Newman afirma: sea $p =< v_0, v_1, ..., v_k >$ la ruta más corta, el algoritmo de Dijkstra relaja las aristas en el orden $\{(v_0, v_1), (v_1, v_2), ...\}$. Podemos decir que la afirmación es errónea, si el algoritmo de Dijkstra al menos da un orden de relajación de aristas distinto para alguna p:

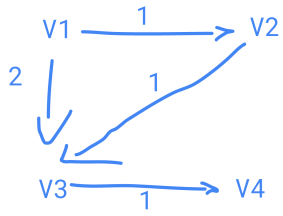

Figura 8: $G = (\{V1, V2, V3, V4\}, \{(V1, V2), (V2, V3), (V3, V4), (V1, V3)\})$

El algoritmo de Dijkstra podría relajar las aristas en el orden: $(V1, V2), (V1, V3), (V3, V4), (V2, V3)$, lo cual la lugar a dos rutas más cortas (con igual peso total), considerando la de nuestro interés $p =< V1, V2, V3, V4 >$. Entonces la afirmación es errónea.

5. Dado un grafo dirigido G = (V, E) con función de costo w: E $\rightarrow$ R, sea $p =< v_0, v_2, \ldots, v_k >$ una ruta más corta del vértice v_0 al vértice v_k. Demuestre que para todo valor de i y j tal que $0 \leq i \leq j \leq k$, si $p_{ij} =< v_i, v_{i+1}, \ldots, v_j >$ es una sub-ruta de p que va del vértice v_i al vértice v_j, entonces p_{ij} es una ruta más corta de v_i a v_j. (15 puntos)
Demostración.

Se define el peso $w(p)$ de una ruta $q =< v_0, v_1, ..., v_k >$:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \tag{4}$$

El peso de la ruta más corta de u a v por:

$$\delta(u, v) = \begin{cases} min\{w(p) : u \xrightarrow{p} v\} & \text{si existe una ruta de u a v} \\ \infty & \text{de otra manera} \end{cases} \tag{5}$$

Entonces definimos la ruta más corta desde u a v para cualquier p con peso $w(p) = \delta(u, v)$.

Así, para la ruta más corta:

$$\delta(u,v) = \sum_{l=1}^{k} w(v_{l-1}, v_l) \tag{6}$$

$$\delta(u,v) = w(v_0,v_1) + w(v_1,v_2) + \sum_{m=i}^{m=j} w(v_m, v_{m+1}) + w(v_{k-1}, v_k), 0 \leq i \leq j \leq k \tag{7}$$

Por definición 4

$$w(p_{ij}) = \sum_{m=i}^{m=j} w(v_m, v_{m+1}) \tag{8}$$

Como el término $w(v_{l-1}, v_l)$ es el mínimo, entonces su suma también lo es: dado que $f(x) = min(a) = a, a \in R$, por lo tanto, $min(a) + min(b) = a + b = min(a + b), a, b \in R$. Concluyendo que $w(p_{ij})$ es el peso mínimo para la sub-ruta. Es decir, por definición 5, que p_{ij} es una ruta más corta de v_i a v_j.

Referencias

[1] Cormen, T. H., & Leiserson, C. E. (2009). In Introduction to algorithms (3rd Edition).