IN7K3-1 Flujos en Redes - Verano 2011 26 de octubre 2011

Auxiliar 2-Pauta

Profesor: Fernando Ordóñez *Auxiliar: Renaud Chicoisne*

Ejercicio 4.18

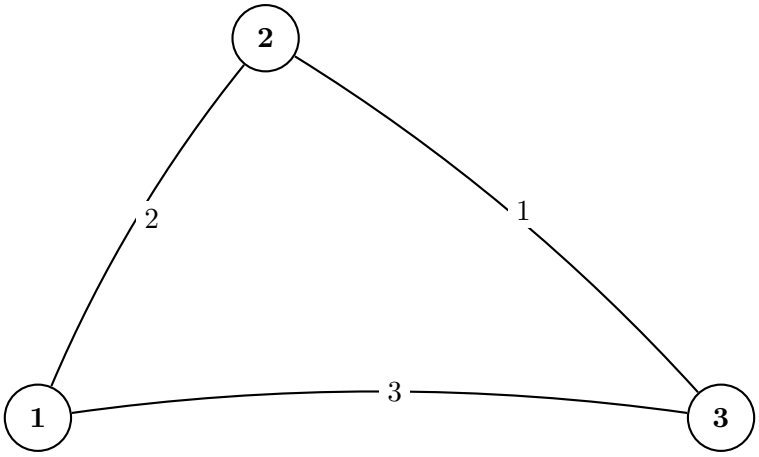
Unit length network A cada iteración del *breadth-first search algorithm* se puede notar que se marca uno de los nodos no marcados lo más cercanos al nodo raíz (*breadth-first search algorithm* = "Búsqueda en ancho primero") Durante una iteración de Dijkstra, cuando se elige el nodo que entra a S se escoge aquel tiene la etiqueta más chica. Dado que todos los arcos tienen largo 1, el nodo escogido es uno de los nodos de S que están a la distancia más chica -en número de arcos- del nodo origen. Lo que es exactamente una iteración del algoritmo (bfs).

Dado que el algoritmo (bfs) verifica a lo máximo una vez cada arco, tiene una complejidad en $O(m)$. Por lo tanto, en un grafo con largos unitarios, encontrar un camino más corto puede ser efectuado en $O(m)$.

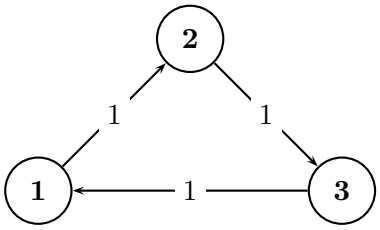
Ejercicio 4.21

Verdadero/falso

(a) El grafo siguiente tiene dos árboles de caminos mínimos desde 1: $\{(1,2),(2,3)\}$ o $\{(1,2),(1,3)\}$

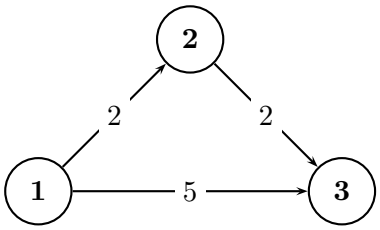


(b) En el siguiente grafo, el largo del camino mínimo entre 1 y 3 es 3, pasando por el camino $\{1, 2, 3\}$.



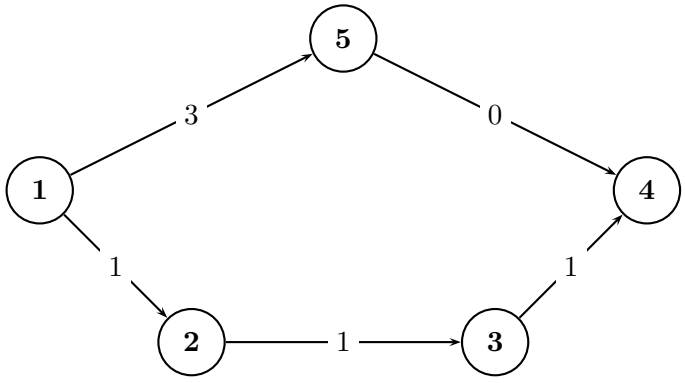
Cuando se eliminan las direcciones de los arcos, el camino mas corto de 1 a 3 es directo pasando por el arco $(1, 3)$ y tiene largo 1.

(c) En el siguiente grafo, el camino más corto de 1 a 3 vale 4, pasando por $\{1, 2, 3\}$.



Ahora si aumentamos de $k = 2$ unidades el largo de cada arco, el camino más corto de 1 a 3 vale 7, pasando directo por el arco $(1, 3)$. $7 - 4 = 3$ no es multiple de $k = 2$.

(d) Ejecutamos el algoritmo de Dijkstra en el siguiente ejemplo a partir del nodo 1:



Iteración 1: $d_1 = 0/pred_1 = 1$ y $S = \{1\}$
Entonces: $d_5 = 3/pred_5 = 1$, $d_2 = 1/pred_2 = 1$

Iteración 2: $S = \{1, 2\}$
Solo se modifica d_3: $d_3 = 2/pred_2 = 2$

Iteración 3: $S = \{1, 2, 3\}$
Solo se modifica d_4: $d_4 = 3/pred_2 = 3$

Iteración 4: $S = \{1, 2, 3, 5\}$
No se modifica ninguna distancia

Iteración 5: $S = \{1, 2, 3, 5, 4\}$
No se modifica ninguna distancia, y tenemos $\bar{S} = \emptyset$: el algoritmo terminó dando como camino más corto entre 1 y 4 el camino $\{1, 2, 3, 4\}$. Sin embargo el camino más corto que contiene el menor número de arcos es {1,5,4}

Ejercicio 5.10
In-tree of shortest paths La siguiente modificación del algoritmo generico de *label-correcting* produce un *in-tree* teniendo raíz en t.

$d_t := 0$, y $p_t := 0$;
$d_j := \infty$, para cada $j \in N \setminus \{t\}$
Mientras existe (i, j) tal que $d_i > d_j + c_{ij}$
    $d_i := d_j + c_{ij}$;
    $p_i := j$;

(BONUS: Truco Tarea) One-to-subset Dijkstra Se puede ver que $\forall t \in T$:

$$d_t \leqslant \max\{d_s : s \in T\} < d_k + \bar{c} < \infty$$

Entonces todos los elementos de T han sido etiquetados (porque tienen etiquetas finitas). Ahora supongamos que se puede encontrar un mejor camino llegando a t del actual, es decir:

$$\exists i \neq t, d_i + c_{i,t} < d_t$$

Aislamos tres casos:
1) Si i ya salió de V, deberíamos tener $d_t \leftarrow d_i + c_{it}$: contradicción.

2) Si i está en V, por definición de k tenemos: $d_i \geqslant d_k$. Ademas, tenemos $c_{it} \geqslant \bar{c}$. Entonces se tiene:

$$d_k + \bar{c} \leqslant d_i + c_{it} < d_t$$

Contradice $d_k + \bar{c} > d_t, \forall t \in T$

3) Si i aún no ha entrado en V, por propiedad de Dijkstra:

$$d_i \geqslant d_v, \forall v \in V$$

Entonces: $d_i \geqslant d_k$. Se llega a la misma conclusión que en el caso 2).