1er Parcial AED3: Parte Domiciliaria

Manuel Panichelli, 72/18

October 10, 2020

Preámbulo

Teoremas / Ejercicios utilizados.

0.1 Arboles

Teorema 1 (Definiciones equivalentes de árbol)**.** Dado $G = (V, X)$ un grafo, las siguientes son equivalentes:

1. G es un árbol, un grafo conexo sin circuitos simples.
2. G es un grafo sin circuitos simples, y e arista tq $e \notin X$, el grafo $G + e$ tiene exactamente un circuito simple el cual contiene a e.
3. G es conexo, pero si se saca cualquier arista queda un grafo no conexo. Es decir, toda arista es puente.

Lema 1. Sea $G = (V, X)$ conexo y $e \in X$.

$$G - e \text{ es conexo } \iff \text{ e pertenece a un circuito simple de G}$$

Teorema 2. Sean $T = (V, X_T)$ un AG de $G = (V, X)$, $e \in X \setminus X_T$. Luego $T + e - f$ con f una arista del único circuito de $G + e$ es AG de G.

Resolución

Ejercicio 1

Ejercicio 2

Ejercicio 3

Ejercicio 4

*Un **puente** de un grafo es un eje del grafo tal que al removerlo se obtiene un grafo con más c.cs. Sea $G = (V, X)$ un grafo conexo y $e \in X$. Demostrar que e es un puente de G sii e pertenece a todo AG de G.*

Dem. Demostremos la ida y la vuelta.

$\Rightarrow$) Sea e puente de G. qvq pertenece a todo AG de G.
Va por el absurdo: supongo que no pertenece a ningún árbol generador de G, y sea $T = (V, X_T)$ uno cualquiera. Como $e \notin T$, por (1.2), $T + e$ tiene exactamente un circuito simple C, el cual contiene a e. Sea $f \in C, f \in X_T$. Como f pertenece al único circuito de $T + e$, por (2) el árbol $T' = T + e - f$ es árbol generador de G, y $e \in T'$. Pero estabamos suponiendo que e no pertenecía a ningún AG de G. Abs! Entonces pertenece a todos.

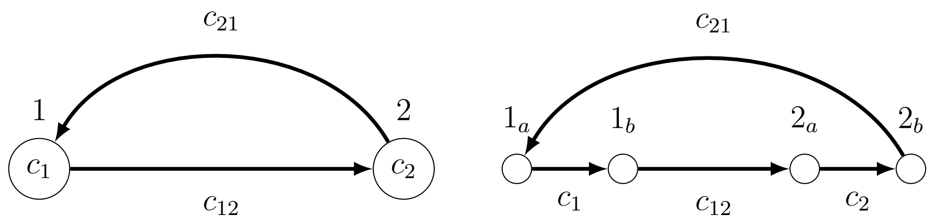
$\Leftarrow$) Como e pertenece a todo AG de G, y por (1.3) todas las aristas de un arbol son puente, en particular e es puente de G.

□

Ejercicio 5

Cabinas de peaje inverso, en las que se le paga un monto al conductor que pase. Cada cabina de peaje $i \in \{1 \dots C\}$ tiene un costo asociado c_i, el cual es negativo para las cabinas inversas, y el costo c_{ij} del viaje de la cabina i a la j, en caso de que se pueda de forma directa.

a. Para modelar el problema, me gustaría representar a las cabinas como nodos, donde una cabina es adyacente a otra si se puede viajar de forma directa. Pero tenemos un problema, las aristas y los vértices tendrían peso, cuando los grafos que modelamos solo tienen peso en las aristas. Vamos a tener que usar una representación muy similar a la de la clase de Mirko. Veamos un ejemplo



(a) Digrafo original (b) Digrafo adaptado

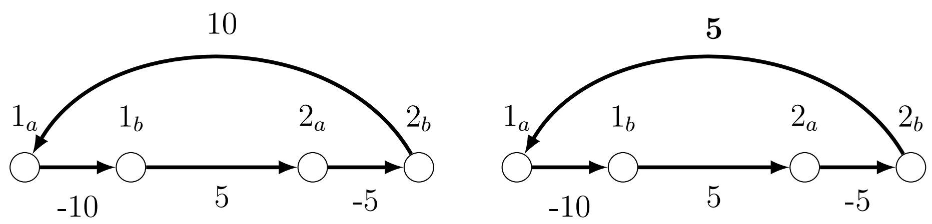
Figure 1: Ejemplo de representación de cabinas con digrafos

Separo cada nodo i con peso c_i en dos, i_a e i_b, con un arco de peso c_i de i_a a i_b. Todos los arcos salen de i_b, y llegan a i_a. De esta forma, me aseguro que cualquier camino que pase por el nodo pague el costo del peaje. Luego, habrá un arco de i_b a j_a con peso c_{ij} si es posible viajar de forma directa de la cabina i a la j.

Matemáticamente, $G = (V, X)$ donde

$$\begin{aligned} V &= \{1_a, 1_b, \dots C_a, C_b\}, \\ X &= \{(i_b, j_a, c_{ij}) \mid \text{se puede viajar de forma directa de i a j}\} \cup \\ &\quad \{(i_a, i_b, c_i) \mid i \in \{1, \dots, C\}\}. \end{aligned}$$

Con este modelo, se podrá obtener una ganancia recorriendo *eternamente* las cabinas si el grafo tiene **ciclos negativos**, circuitos que luego de recorrerlos dan una ganancia.



(a) Cabinas sin ciclos negativos (b) Cabinas con ciclos negativos

Figure 2: Ejemplo de cabinas con y sin ciclos negativos

En la figura 2 se pueden ver dos ejemplos de cabinas y montos. En 2a no hay ningún ciclo negativo, a pesar de que el camino $P : 1_a\ 1_b\ 2_a\ 2_b$ tiene costo -10, al volver a 1_a se vuelve 0, con lo cual recorrer infinitamente nunca daría ganancias. En cambio, en 2b se puede ver como con un pequeño ajuste, el ciclo $C : 1_a\ 1_b\ 2_a\ 2_b\ 1_a$ tiene costo -5, con lo cual se puede recorrer eternamente, generando ganancias.

b. Para resolver el problema del inciso anterior, basta con ver si hay ciclos negativos en algún camino de una cabina a la otra. Para ello, se puede usar el algoritmo de **Floyd**, pensado para resolver camino mínimo *múltiples orígenes, múltiples destinos*, que puede ser adaptado para detectar ciclos negativos. Este tiene una complejidad temporal de $O(n^3)$ con n la cantidad de vertices. Y en nuestro caso, $n = 2 \times |C|$, el doble de la cantidad de cabinas, ya que cada una se separa en dos nodos.