Simulacro de segundo parcial de Programación 3
noviembre de 2018

Ejercicio 1 (18 puntos)

Para los monitoreos de cierto curso de la carrera de Ing. en Computación, que tiene un conjunto $A = \{a_1, \ldots, a_n\}$ de alumnos inscriptos, se ha definido un conjunto $H = \{h_1, \ldots, h_m\}$ de m horarios diferentes. A través de una encuesta, cada alumno a_i, $1 \leq i \leq n$, ha seleccionado un conjunto no vacío de horarios H_i, $H_i \subseteq H$, en los cuales tiene disponibilidad para asistir. Para cada horario h_j, $1 \leq j \leq m$, se dispone de un salón de clases con capacidad para c_j alumnos. El problema consiste en asignar alumnos a horarios de modo que todos sean asignados a alguno de los horarios que eligió y no se exceda la capacidad de ningún salón. Si dicha asignación no es factible, queremos poder determinar correctamente que no existe una solución.

(a) Especifique una red de flujo que modele esta realidad y permita solucionar el problema mediante una maximización de flujo.

(b) Dé un algoritmo que determina si existe una solución al problema y, en ese caso, la construye. Recuerde que puede usar algoritmos vistos en el material teórico del curso sin reescribirlos.

(c) Muestre que en los casos en que su algoritmo devuelve una asignación de alumnos a horarios, dicha asignación satisface las restricciones del problema: todo alumno es asignado a uno y solo un horario y en ningún caso se excede la capacidad de un salón.

Solución:

(a) El conjunto de vértices de la red lo definimos como $V = \{s, t\} \cup A \cup H$, donde s es un nodo *fuente* y t un *sumidero*. El conjunto de aristas se construye de la forma que se describe a continuación. Para cada i, $1 \leq i \leq n$, agregamos una arista de s a a_i con capacidad 1 y agregamos también una arista de capacidad 1 desde a_i a cada uno de los vértices que pertenecen a H_i. Para cada horario h_j agregamos una arista desde h_j a t con capacidad c_j.

(b)
1. Ejecutar el algoritmo de Ford-Fulkerson sobre la red de la parte anterior para obtener un flujo integral, f, de máximo valor.
2. Si el valor de f es menor que n, responder que no existe solución.
3. En caso contrario, asignar a cada alumno a_i un horario h_j tal que (a_i, h_j) es una arista con flujo positivo en f.

(c) El algoritmo devuelve una asignación cuando el valor de f es mayor o igual a n. Como desde s solo parten n aristas de capacidad 1, todas ellas deben estar saturadas de flujo. Esto implica que todo vértice $a_i \in A$ recibe una unidad de flujo y, por la propiedad de conservación, debe salir una unidad de flujo desde a_i. Más aún, como el flujo es integral, esto implica que para cada alumno $a_i \in A$ existe una única arista saliente con flujo unitario. Por lo tanto, todos los alumnos son asignados a uno y solo un horario. Por otra parte, el flujo entrante a cada horario h_j es la suma de flujos unitarios, que por definición de la red solo pueden provenir de alumnos, por lo cual coincide con la cantidad de alumnos asignados a ese horario. Por la propiedad de conservación, el flujo saliente de h_j debe ser igual a esta cantidad de alumnos asignados, que a su vez debe ser menor o igual que la capacidad c_j de la única arista saliente de h_j. En consecuencia, la solución no excede la capacidad de ninguno de los salones.

Ejercicio 2 (20 puntos)

En una fábrica tenemos que realizar n tareas de corte, $1 \dots n$, en ese orden, sobre materiales potencialmente diferentes, $m_1 \dots m_n$. Para esto contamos con una máquina cortadora, la cual lleva instalada una cuchilla que puede ser de tipo A o de tipo B. Dependiendo del material que cortemos, cada tipo de cuchilla sufre un desgaste diferente que provoca un cierto costo económico; sean $C_A(m)$ y $C_B(m)$ los costos generados por cortar el material m con cuchillas de tipo A y B, respectivamente. Al momento de iniciar cada tarea somos libres de instalar el tipo de cuchilla que nos convenga. Sin embargo, cambiar un tipo de cuchilla por otro implica detener la máquina por cierto tiempo, lo cual ocasiona un costo económico T. Nuestro problema consiste en elegir con qué tipo de cuchilla realizar cada una de las n tareas de modo de minimizar el costo total. Asumimos que al momento de comenzar a trabajar la máquina tiene instalada una cuchilla de tipo A.

(a) ¿Cuál es el costo de realizar la primera tarea con una cuchilla de tipo A? ¿Y con una cuchilla de tipo B? Recuerde que inicialmente la máquina tiene instalada una cuchilla de tipo A.

(b) Escriba una relación de recurrencia que permita resolver el problema mediante un algoritmo de programación dinámica. Especifique claramente los parámetros y la semántica de la función que defina. Explique la recurrencia.

(c) Escriba un algoritmo iterativo de programación dinámica para calcular el costo total óptimo.

(d) Escriba un algoritmo para obtener una solución óptima.

Solución:

(a) Como inicialmente la máquina tiene instalada una cuchilla de tipo A, si se usa el tipo A para la primera tarea el costo es simplemente el del desgaste, que es $C_A(m_1)$. Si en cambio se usa el tipo B, además del costo de desgaste incurrimos el de intercambio de cuchillas, de modo que el costo en este caso es $C_B(m_1) + T$.

(b) Para un entero i, $1 \leq i \leq n$, y un tipo de cuchilla X, $X \in \{A, B\}$, definimos $OPT(i, X)$ como el costo óptimo para realizar las primeras i tareas terminando con una cuchilla de tipo X. A partir de la parte a concluimos que para $i = 1$ se cumplen las ecuaciones (2) y (3). Para cada paso i, $1 < i \leq n$, tenemos dos opciones: usamos la cuchilla que ya está instalada o la cambiamos. En cualquiera de los dos casos, si llamamos X al tipo de cuchilla que decidimos usar en el paso i, incurrimos un costo de desgaste $C_X(m_i)$; en el segundo caso incurrimos un costo adicional T por el cambio de cuchilla. Observamos que una solución óptima que usa una cuchilla de tipo X para la tarea i debe minimizar el costo incurrido en las primeras $i - 1$ tareas y terminar usando la cuchilla de tipo X en el primer caso y del tipo opuesto en el segundo caso. Denotando $\bar{X}$ el tipo de cuchilla opuesto a X, vemos que la función OPT satisface (1).

$$OPT(i, X) = C_X(m_i) + \min\{OPT(i-1, X), OPT(i-1, \bar{X}) + T\}, \quad 1 < i \leq n, X \in \{A, B\}, \quad (1)$$

$$OPT(1, A) = C_A(m_1), \quad (2)$$

$$OPT(1, B) = C_B(m_1) + T. \quad (3)$$

(c) El algoritmo para calcular el costo total mínimo se presenta en la figura 1.

```
1 Hacer OPT[1, A] = C_A(m_1)
2 Hacer OPT[1, B] = C_B(m_1) + T
3 for i = 2 to n do
4     Hacer OPT(i, A) = C_A(m_i) + mín{OPT(i − 1, A), OPT(i − 1, B) + T}
5     Hacer OPT(i, B) = C_B(m_i) + mín{OPT(i − 1, B), OPT(i − 1, A) + T}
6 end
7 Devolver mín{OPT(n, A), OPT(n, B)}
```

Figura 1: Algoritmo para calcular el costo total mínimo. Como producto secundario calcula $OPT(i, X)$, para $1 \leq i \leq n$, $X \in \{A, B\}$.

(d) El algoritmo de la figura 2 construye una solución óptima en un arreglo de tamaño n, SOL, donde el contenido de cada posición i, $1 \leq i \leq n$, indica el tipo de cuchilla que se usa para la tarea i. El algoritmo utiliza el arreglo OPT producido por el algoritmo de la parte anterior.

```
1 if OPT(n, A) < OPT(n, B) then Hacer X = A else Hacer X = B
2 for i = n downto 2 do
3     Hacer SOL[i] = X
4     if OPT(i − 1, X̄) + T < OPT(i − 1, X) then Hacer X = X̄
5 end
6 Hacer SOL[1] = X
7 Devolver SOL
```

Figura 2: Algoritmo para construir una solución óptima.

Ejercicio 3 (20 puntos)

Un ciclo en un grafo dirigido es Hamiltoniano si pasa por todos los vértices y no repite ninguno, salvo el primero y el último que coinciden. De forma similar, un camino es Hamiltoniano si pasa por todos los vértices y no repite ninguno. Sabemos que el problema de determinar si un grafo dirigido tiene algún ciclo Hamiltoniano (problema *Ciclo H*) es NP-completo. Ahora consideramos el problema *Camino H*, que consiste en determinar si un grafo dirigido tiene algún camino Hamiltoniano.

(a) Muestre que el problema *Camino H* pertenece a NP.

(b) Muestre que el problema *Camino H* es NP-completo.

Sugerencia: Dado un grafo dirigido $G = (V, E)$ considere un grafo $G' = (V', E')$ que se construye a partir de G sustituyendo un vértice arbitrario, u, por dos vértices nuevos, v ,w, y redirigiendo todas las aristas entrantes a u hacia v y todas las salientes de u desde w (ver figura 3).

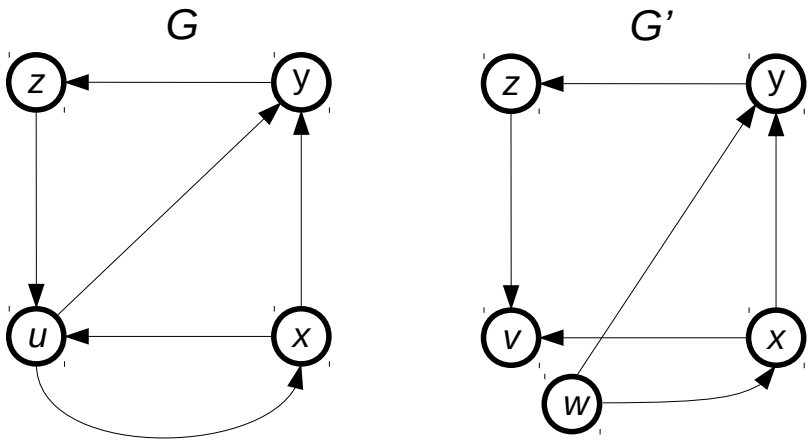

Figura 3: Transformación de un grafo G en otro G' según la sugerencia. En G hay un ciclo Hamiltoniano: u, x, y, z, u. En G' hay un camino Hamiltoniano: w, x, y, z, v.

Solución:

(a) Un certificado es la lista de nodos que componen un camino Hamiltoniano. La representación de este certificado requiere una cantidad de bits que es de orden polinomial en el tamaño del grafo (por ejemplo un arreglo de tamaño n que contiene los índices de los vértices en el orden en que ocurren en el camino requiere $O(n \log n)$ bits). La verificación del certificado puede implementarse recorriendo en orden los vértices que componen el camino candidato controlando en cada paso:

1. que vértices consecutivos están unidos por una arista,
2. que no se pasa más de una vez por un mismo vértice.

El primer control requiere un tiempo total que es $O(m + n)$, si el grafo está representado mediante lista de adyacencia, y el segundo requiere un tiempo total que es $O(n)$, si mantenemos un arreglo booleano donde marcamos los vértices que se visitan.

(b) Reducimos el problema *Ciclo H* a *Camino H* a través de la reducción sugerida. Esta reducción requiere un tiempo de ejecución polinomial en el tamaño del grafo. En efecto, sea $G = (V, E)$ un grafo dirigido, con n nodos y m aristas, que constituye una instancia del problema *Ciclo H*. Si usamos, por ejemplo, una representación de lista de adyacencia (con dos listas por nodo: una para las aristas salientes y otra para las entrantes), necesitamos copiar las listas de adyacencia de $n - 1$ vértices, reemplazando las referencias a u por v o w según corresponda, y generar las listas de adyacencia para v y w a partir

de las listas de aristas entrantes y salientes de u, respectivamente. Claramente este proceso requiere un tiempo de ejecución que es $O(m+n)$.

Veamos ahora que G contiene un ciclo Hamiltoniano si y solo si G' contiene un camino Hamiltoniano. Supongamos que G contiene un ciclo Hamiltoniano. Recorriendo este ciclo a partir de u, podemos escribirlo como $u, x_1, \ldots, x_{n-1}, u$. Por definición de G', tanto (w, x_1) como (x_{n-1}, v) son aristas de G', así como también (x_{i-1}, x_i), $1 < i < n$, por lo cual $w, x_1, \ldots, x_{n-1}, v$ es un camino Hamiltoniano en G'. Por otra parte, si G' tiene un camino Hamiltoniano, necesariamente parte de w, porque w no tiene aristas entrantes, y necesariamente termina en v, porque v no tiene aristas salientes. Entonces, dicho camino es de la forma $w, x_1, \ldots, x_{n-1}, v$ y, como tanto (u, x_1) como (x_{n-1}, u) son aristas de G, así como también (x_{i-1}, x_i), $1 < i < n$, concluimos que $u, x_1, \ldots, x_{n-1}, u$ es un ciclo Hamiltoniano de G.