MA4701. Optimización Combinatorial. 2013.
Profesor: José Soto

Problemas Controlables (parte 1).

Problema 1.
Sea $G = (V, E)$ un grafo.

a) Demuestre que si el grado de cada vértice es igual a 2 entonces las componentes conexas de G son todas ciclos.

b) Para el caso general, diseñe un algoritmo que encuentre un ciclo contenido en G o reporte que G no tiene ciclos. Pruebe correctitud y determine su complejidad. ¿Puede lograr un algoritmo $o(n^3)$?
Indicación: Intente obtener un algoritmo $O(n + m)$.

Problema 2
Dado un grafo $G = (V, E)$ conexo y una funcion de peso $w: E \to \mathbb{R}$ que puede tomar valores negativos. Considere el problema de encontrar un subgrafo $G' \subseteq G$ conexo y generador de peso mínimo.

a) Muestre con un ejemplo que si T es árbol generador de peso mínimo de G entonces T no es necesariamente óptimo para el problema considerado.

b) Diseñe un algoritmo que resuelva el problema. Demuestre que el algoritmo es correcto y evalúe su complejidad en función de n y m. Éste debe ser fuertemente polinomial. ¿Puede lograr $O(n^2)$?

Problema 3
Un digrafo (o grafo dirigido) es un par $G = (V, E)$ donde $E \subseteq V \times V$. Los elementos de E se llaman arcos y los elementos de V se llaman nodos. Análogamente al caso de grafos no dirigidos se definen los conceptos de paseo, caminos y ciclos dirigidos (pero ahora todos ellos tienen "dirección").

Formalmente, un u-v camino **dirigido** es una secuencia de nodos distintos $x_1x_2 \ldots x_k$, con $x_1 = u$, $x_k = v$, tal que para todo i, (x_i, x_{i+1}) es arco en E.

Por otro lado un u-v camino **no dirigido** es una secuencia de nodos distintos $x_1x_2 \ldots x_k$, con $x_1 = u$, $x_k = v$, tal que para todo i, al menos uno de (x_i, x_{i+1}) y (x_{i+1}, x_i) es arco en E. En otras palabras, un camino no dirigido es un camino en el grafo obtenido al remover las direcciones de los arcos.

Decimos que u y v están **débilmente conectados** en G si existe un u-v camino **no** dirigido.

Decimos que u y v están **fuertemente conectados** en G si existen ambos un u-v camino dirigido y un v-u camino dirigido.

Ambas relaciones (débilmente conectado o fuertemente conectado) son de equivalencia en V. Las clases de equivalencia se conocen como componentes débilmente conexas de G y componentes fuertemente conexas de G.

a) Diseñe un algoritmo para encontrar la componente débilmente conexa de G que contiene a un nodo dado v en tiempo $O(|V| + |E|)$.

b) Modifique el algoritmo anterior para encontrar todas las componentes débilmente conexas en tiempo $O(|V|+|E|)$.

c) Diseñe un algoritmo para encontrar la componente fuertemente conexa de G que contiene a un nodo dado v en tiempo $O(|V| + |E|)$.

d) (No controlable, pero buen ejercicio) Modifique el algoritmo anterior para encontrar todas las componentes fuertemente conexas en tiempo $O(|V| + |E|)$.

Problema 4

Sea $G = (V, E)$ conexo y $w\colon E \to \mathbb{R}^+$ estrictamente positiva. Sea $\mathcal{X}$ el conjunto de los árboles generadores de G.

a) Demuestre que $T \in \mathcal{X}$ minimiza $w(E(T)) = \sum_{e \in E(T)} w(e)$ si y solo si minimiza $\prod_{e \in E(T)} w(e)$.

b) Demuestre que el siguiente algoritmo calcula un **bosque**[1] generador de costo mínimo, para cualquier grafo $G = (V, E)$ no necesariamente conexo y cualquier función de costo $w\colon E \to \mathbb{R}$ no necesariamente positiva.

Algorithm 1: Kruskal reverso (G, w)

Ordenar y renombrar las aristas de E de modo en orden decreciente de costo, de modo que $w(e_1) \geq w(e_2) \geq \cdots \geq w(e_m)$, donde $E = \{e_1, \ldots, e_m\}$;
$F \leftarrow E$; `// Aristas de solución`
for $i = 1$ *to* m **do**
  if $(F - e_i)$ *genera* e_i **then**
    $F \leftarrow F - e_i$
  end
end
return (V, F)

Problema 5

(a) Sea $M = (E, \mathcal{I})$ una matroide. Un **circuito** $C \subseteq E$ es un conjunto dependiente minimal (es decir, $C \notin \mathcal{I}$, y $\forall e \in C$, $C - e \in \mathcal{I}$). Sea $\mathcal{C}$ el conjunto de circuitos de M. Demuestre que la condición (AX-C) siguiente se satisface.

$$\text{Si } C_1, C_2 \in \mathcal{C},\ C_1 \neq C_2 \text{ y } z \in C_1 \cap C_2 \text{ entonces existe } C \in \mathcal{C} \text{ con } C \subseteq (C_1 \cup C_2) - z. \tag{AX-C}$$

(b) ¿A qué corresponden los circuitos de una matroide gráfica? Demuéstrelo.

(c) Sea ahora E un conjunto finito cualquiera, y $\mathcal{C} \subseteq 2^E$ un conjunto que satisface (AX-C), $\emptyset \notin \mathcal{C}$ y la condición (Clutter) siguiente

$$(\forall C_1, C_2 \in \mathcal{C}), C_1 \subseteq C_2 \text{ implica que } C_1 = C_2 \text{ (i.e., ningún conjunto contiene a otro).} \tag{Clutter}$$

Defina el conjunto $\mathcal{I} = \{F \subseteq E\colon F \text{ no contiene ningún } C \in \mathcal{C} \text{ como subconjunto}\}$.

Demuestre que $M = (E, \mathcal{I})$ es una matroide y que $\mathcal{C}$ es exactamente su conjunto de circuitos.

(d) Concluya que dos matroides con igual conjunto de referencia E e igual conjunto de circuitos $\mathcal{C}$ son iguales.

Problema 6

Sea $M = (E, \mathcal{I})$ una matroide.

(a) Demuestre que si I es independiente, y $e \in E \setminus I$ entonces $I + e$ contiene a lo más un circuito.

(b) Sea $e_1, \ldots, e_m$ un ordenamiento **cualquiera** de E y sea $w\colon E \to \mathbb{R}$ una función de peso. Considere el siguiente algoritmo de intercambios:

[1] Notar la diferencia con el problema 2

Algorithm 2: Intercambios (max)

```
F ← ∅; // Conjunto solución
for i = 1 to m do
    if F + e_i ∈ I then
        F ← F + e_i
    end
    else
        Sea C ⊆ F + e_i el único circuito creado y sea f ∈ C el elemento de menor peso de C.
        F ← F + e_i − f; // Intercambiamos f por e_i en la solución
    end
end
return F
```

Sea $E_i = \{e_1, \dots e_i\}$. Demuestre que al final de la iteración i, F contiene una base de peso máximo de la matroide $M|_{E_i}$. Concluya que el algoritmo devuelve una base de peso máximo de F.

(c) Sea $G = (V, E)$ un grafo con n vértices y con peso en sus m aristas. Considere el siguiente juego entre usted y un adversario. Usted parte con un conjunto F de aristas de un bosque generador de peso **mínimo**. En cada etapa el adversario **disminuye** el peso de una arista cualquiera y usted debe actualizar F (intercambiando aristas) para que siga siendo de peso mínimo. Diseñe un algoritmo (y pruebe que es correcto) para esta labor. En cada etapa, su algoritmo debe tomar tiempo $O(n)$.
Indicación: Modifique el algoritmo anterior.
Observación: Si permitimos al adversario disminuir o aumentar el peso de una arista el problema se hace un poco más difícil. (No controlable: ¿Puede lograr un algoritmo $O(n + m)$ para este caso?)