**Técnicas de Diseño de Algoritmos - Algoritmos III**

**1er Cuatrimestre 2024 - 2do Parcial**

| Nombre y Apellido | LU | #Ord |
|---|---|---|
| ...................................... | ........... | ....... |

| Tema | Ej A / B | #hojas |
|---|---|---|
| A | ........ | ........ |

*Duración: 4 horas. Este examen es a **libro cerrado**. Para aprobar se debe alcanzar 50 puntos.*

# 1. Preguntas opción múltiple (60 ptos.)

*Marcar con una cruz (X) **todas** las respuestas correctas.*
*Un ejercicio con todas sus opciones correctas marcadas y todas sus incorrectas **no** marcadas suma 6 puntos.*

**Pregunta 1** Sea $k \in \mathbb{R}$ un número real y $G$ un grafo pesado con $m$ aristas y $n$ vértices tal que toda arista tiene peso $k$. ¿Qué algoritmo/s puede/n utilizarse para encontrar un AGM de $G$ en menor complejidad?

- [ ] Kruskal
- [ ] Djikstra
- [x] DFS
- [ ] Prim
- [x] BFS

**Pregunta 2** Lean quiere armar un grafo muy simple para el segundo parcial de Algo III, por lo que crea un grafo $G$ conexo con $n$ vértices y $\frac{n(n-1)}{2}$ aristas. Define la longitud de cada arista según su índice de la siguiente manera: para todo $i$ tal que $1 \leq i \leq \frac{n(n-1)}{2}$, define la longitud $l$ de $e_i$ como $l(e_i) = i$. Pero tiene un problema: el grafo le quedó muy grande para las diapositivas debido a las aristas tan largas, por lo que quiere construir un subgrafo generador $H$ de $G$ que minimice la longitud total de las aristas que se muestran en pantalla. Naturalmente, también quiere que el grafo resultante sea conexo. ¿Qué algoritmo/s puede usar Lean para generar $H$?

- [x] Prim
- [ ] DFS
- [x] Kruskal
- [ ] Alcanza con devolver las $n-1$ aristas más livianas.
- [ ] Dijkstra
- [ ] BFS

Aparte, Lean se pregunta ¿Cuántos AGM distintos puede puede tener este grafo?

- [ ] Tiene exactamente 2 AGMs.
- [x] Solo tiene un AGM.
- [ ] Tiene más de 2 AGMs.

**Pregunta 3** Decidir cuáles de las siguientes afirmaciones son verdaderas para todo grafo pesado conexo $G = (V, E)$:

- [x] Si $e \in E$ es una de las aristas con peso mínimo de $G$ entonces pertenece a por lo menos un AGM.
- [x] Si $e \in E$ es una arista puente de $G$ entonces pertenece a todos los AGM.
- [x] Si $e \in E$ tiene peso menor estricto a todas las otras aristas de $G$ entonces $e$ pertenece a todos los AGMs de $G$.
- [ ] Si $e \in E$ tiene peso mayor estricto a todas las otras aristas de $G$ entonces $e$ no pertenece a ningún AGM.
- [ ] Si $e \in E$ pertenece a un ciclo $C$, entonces hay un AGM de $G$ que no contiene a $e$.
- [ ] Si $|E| > n - 1$ entonces hay alguna arista que no pertenece a ningún AGM.

**Pregunta 4** Facundo tiene un grafo conexo y quiere armar un AGM, pero no quiere usar ninguno de los algoritmos conocidos. Luego, se le ocurre el siguiente algoritmo: va a encontrar ciclos iterativamente, y cada vez que encuentre uno va a quitar el eje más pesado del mismo. Para buscar cada ciclo va a usar DFS, explotando la estructura que devuelve el algoritmo para encontrar un ciclo cualquiera en el menor tiempo posible. Este proceso continuará hasta que el grafo no tenga más ciclos. Decidir cuáles de las siguientes afirmaciones son verdaderas:

- [ ] El algoritmo propuesto por Facundo tiene complejidad $\Theta(nm)$ en peor caso (usando los algoritmos vistos en la materia).
- [x] El algoritmo propuesto por Facundo tiene complejidad $\Theta(m^2)$ en peor caso (usando los algoritmos vistos en la materia).
- [x] El algoritmo propuesto por Facundo es correcto (es decir, al final del procedimiento tendrá un AGM).
- [ ] El algoritmo propuesto por Facundo es incorrecto (es decir, al final del procedimiento no tendrá un AGM).

**Pregunta 5** Sea $G = (V, E)$ un grafo completo y pesado, con función de peso $w : E \to \mathbb{R}_{>0}$. Sean $s$ y $t$ dos vértices de $G$, y sea $D$ un DAG de caminos mínimos desde $s$ a $t$ de $G$. Decidir cuáles de las siguientes afirmaciones son verdaderas.

- [x] Una arista de $D$ puede pertenecer a más de un camino mínimo de $G$ entre $s$ y $t$.
- [ ] El camino máximo entre $s$ y $t$ no puede pertencer a $D$.
- [ ] Toda arista de $G$ aparece con exactamente una dirección definida en $D$.
- [ ] Toda arista $vz \in E$ que pertence al DAG de caminos mínimos satisface $d(s, v)+w(vz)+w(zt) = d(s, t)$.
- [ ] $D$ tiene a lo sumo $|V| - 1$ aristas.

**Pregunta 6** Tuki tiene un grafo pesado $G$ con $n$ nodos, $m_1$ aristas con peso positivo y $m_2$ aristas con peso negativo. En su modelo los pesos representan dinero: un eje con peso positivo indica que se debe pagar al recorrerlo (*ejes de pago*), mientras que uno con peso negativo indica que se cobra (*eje de cobro*). Naturalmente, los ejes de cobro no forman ciclos.
Dados dos nodos $v$ y $w$, está interesado en conocer el mayor monto final que se puede obtener yendo de $v$ a $w$ y luego volviendo a $v$, asumiendo que a la ida solo se pueden usar *ejes de pago*, y a la vuelta solo *ejes de cobro* ¿Cuál es la complejidad más ajustada en la que puede resolver el problema, usando los algoritmos vistos?

- [x] $O(n + m_1 \log n + m_2)$
- [ ] $O(m_1 \log n + m_2 n)$
- [ ] $O((m_1 + m_2)n)$
- [ ] $O((m_1 + m_2) \log n)$
- [ ] $O(n + m_1 + m_2)$

**Pregunta 7** Rocío tiene un grafo dirigido sin pesos $G = (V, E)$ con $|V| = n$ y $|E| = m$, y dados dos nodos $v$ y $w$ quiere encontrar el camino simple mínimo de longitud par que los une (es decir, si $C$ es el conjunto de caminos simples de $v$ a $w$ de longitud par en $G$, entonces Rocío quiere un camino simple de longitud mínima de $C$). Para eso plantea el siguiente modelo: como conjunto de nodos usa los pares $(v, b)$ con $v \in V$ y $b \in \{0, 1\}$, donde $b$ va a ser usado para indicar la paridad del camino, y por cada eje $xy \in E$ agrega los ejes $(x, 0) \to (y, 1)$ y $(x, 1) \to (y, 0)$. Luego, Rocío propone ejecutar BFS desde el nodo $(v, 0)$ y ver la distancia al nodo $(w, 0)$. Decidir cuáles de las siguientes afirmaciones son verdaderas:

- [ ] Rocío podría ejecutar BFS desde $(w, 0)$ y ver la distancia a $(v, 0)$ y la respuesta no cambiaría.
- [x] Rocío podría ejecutar BFS desde $(v, 1)$ y ver la distancia a $(w, 1)$ y la respuesta no cambiaría.
- [x] Su modelo es incorrecto, no le va a devolver la longitud de un camino simple de long. par mínima.
- [ ] La complejidad del algoritmo propuesto por Rocío es $\Theta((n + m)^2)$ en peor caso.
- [ ] Su modelo es correcto, y le va a indicar la longitud de un camino simple de long. par mínima.

**Pregunta 8** Dada $G$ una red de flujo con capacidades enteras y nodos distinguidos $s$ y $t$. Decidir cuáles de las siguientes afirmaciones son verdaderas.

- [ ] Si todos los arcos de $G$ tienen capacidad distinta entonces $G$ tiene un único corte mínimo.
- [x] Si $G$ tiene flujo máximo impar entonces al menos un arco tiene capacidad impar.
- [ ] Si $G$ es acíclico entonces para todos los flujos posibles el grafo residual $G_f$ también es acíclico.
- [ ] Hay un flujo de valor $k \iff$ los arcos que salen de la fuente $s$ tienen capacidades que suman al menos $k$ y los arcos que entran al sumidero $t$ tienen capacidades que suman al menos $k$.
- [ ] Si todos los arcos de $G$ tienen capacidad impar entonces existe un flujo máximo $f$ tal que $f(e)$ es impar para todo arco $e$.

**Pregunta 9** Sea $G$ una red con nodos distinguidos $s$ y $t$. Decidir cuáles de las siguientes afirmaciones son verdaderas:

- [ ] Sea $C$ un corte mínimo de $G$ y $e$ una arista de $G$ que cruza el corte, yendo de la componente de $s$ a la de $t$. Es cierto que si aumentamos su capacidad entonces el flujo máximo aumenta.
- [ ] Si se le suma una constante $\delta > 0$ a todas las capacidades de los arcos de $G$ entonces el flujo máximo no cambia.
- [x] Para todo $n \in \mathbb{N}$ existe una red de $n + 2$ vértices con $2^n$ cortes s-t distintos, todos con capacidad mínima.
- [ ] En cualquier flujo máximo no hay ciclos que lleven flujo mayor que 0.

**Pregunta 10** Gracias a la carrera de computación Sasha consiguió empleo en una fábrica que construye herramientas. En la fábrica tienen un conjunto $P$ de $p = |P|$ piezas de tipo 1 que se pueden acoplar a un conjunto $Q$ de $q = |Q|$ piezas de tipo 2. Lamentablemente, no toda pieza de tipo 1 puede acoplarse a una de tipo 2.
Más puntualmente, la fábrica conoce, para cada pieza $x \in P$ el conjunto $N_x \subseteq Q$ de piezas de tipo 2 a la que puede acoplarse $x$. Aparte, se sabe que $N_x$ siempre tiene a lo sumo 15 piezas.
Su jefe le pide que consiga con *algún algoritmo de flujo* la cantidad máxima de piezas acoplables. Sasha nos pide ayuda para modelar este problema y que le demos su complejidad como pista para que elle lo pueda resolver. ¿Cuál es la complejidad más ajustada para resolver este problema con un modelado de flujo, en términos de $p$ y $q$?

- [ ] $O(pq)$
- [x] $O(\text{mín}\{p, q\}(p + q))$
- [ ] $O(\text{mín}\{p, q\}pq)$
- [ ] $O((p + q)^3)$
- [ ] $O(p(p + q))$

# 2. Ejercicio a desarrollar (40 ptos.)

*Marcar con una cruz el cuadrado* ☐ *del ejercicio que vas a entregar.*
***Solo se entrega 1 de los 2 ejercicios.***

## 2.1. ☐ Problema A

La empresa *Tuki logística* está planificando rutas para sus repartidores, quienes deben entregar paquetes en diferentes puntos de una ciudad. Los repartidores tienen un límite de tiempo que pueden caminar sin descansar.

La empresa conoce el mapa de la ciudad, el cual se representa con un conjunto de nodos $V$ (ubicaciones) y un conjunto de aristas $E$ (calles) uniendo pares de nodos. También conocen un subconjunto $B \subseteq V$ de nodos donde los repartidores pueden tomar descansos y recargar energía. Para cada calle $e \in E$, se sabe el tiempo $t(e)$ que toma recorrerla, y cada repartidor es capaz de caminar $k$ minutos antes de necesitar un descanso.

El día de hoy, *Tuki logística* quiere descubrir el menor tiempo en el cual un repartidor puede llegar desde el club Luna de Avellaneda (LA) hasta Ciudad Universitaria (CU), asegurándose que no se agote en el camino.

a) Modele el problema como un problema de recorrido en un grafo ponderado. Justifique cómo un camino en su modelo representa la solución al problema original.

b) Describa un algoritmo eficiente para resolver el problema, con una complejidad temporal de $O(k|E|\log(k|V|))$. Incluya pseudocódigo si es necesario para mayor claridad. Describa las estructuras de datos utilizadas y justifique la correctitud y la complejidad del algoritmo.

## 2.2. ☐ Problema B

En el torneo de voley TYVA el organizador Ramiro se cansó de armar el fixture a mano, por lo que contrató a Sasha para que desarrolle un algoritmo que arme los fixtures automáticamente.

La liga consiste de un conjunto $E$ de equipos y un conjunto $P$ de partidos que deben disputarse, siendo un partido un par $p = (e, e')$ con $e, e' \in E$ y $e \neq e'$. El torneo además tiene un conjunto $T$ de turnos disponibles, siendo un turno $t \in T$, una tupla $(cancha, horario)$. En cada turno se puede jugar a lo sumo 1 partido. Cada equipo $e \in E$ tienen una disponibilidad $D_e \subseteq T$ de turnos en los que puede jugar (donde nunca informa que puede jugar en dos turnos que sean en el mismo horario). Aparte, para cada equipo $e$ hay un turno $f_e \in T$ que es su **turno favorito**.

Para cada partido $p = (e, e')$, se busca encontrar un turno $t \in T$ en el que pueda realizarse. Para que el partido se pueda jugar ambos equipos deben estar disponibles en ese turno, o bien ese turno debe ser uno de los favoritos de alguno de los dos equipos. Aparte, para evitar favoritismos se desea evitar que se jueguen más de $K$ partidos en una misma cancha.

En base a estos datos se busca armar un fixture, que consiste en encontrar turnos disponibles para todos los partidos. En caso de no ser posible se debe devolver la máxima cantidad de partidos que pueden organizarse siguiendo las restricciones.

Se pide:

1. Modelar el problema como un problema de flujo
2. Dar una interpretación a cada unidad de flujo y cada restricción de capacidad.
3. Justificar que el modelo es correcto. En particular, mostrar cómo se construye el fixture a partir del flujo.
4. Determinar la complejidad temporal de resolver el problema en base al tamaño de la entrada. Asumir que para resolver el modelo de flujo se emplea el algoritmo de Edmonds y Karp, y dar una cota ajustada. La entrada del algoritmo consiste en los conjuntos $E$, $P$, $T$ y $D_e$ para cada $e \in E$; los turnos $f_e$ para cada $e \in E$ y el número natural $K$.