Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Primer Semestre de 2022

IIC 2213 – Lógica para ciencia de la Computación
Tarea 5 - Entrega Viernes 3 de Abril a las 20:00 - via canvas

Recuerda que esta tarea es individual. Puedes discutir sobre la respuesta con tus compañeros (¡y eso está muy bien!), pero no puedes enviar la respuesta a nadie o utilizar la respuesta de alguien más. Esta tarea simula una situación de análisis de algoritmos en la que podrías encontrarte en tu vida futura, ¡practícala ahora!
Importante. Para efectos de esta tarea, puedes considerar como NP-completos a SAT, CNF SAT, 3-CNF SAT, o a cualquiera de los 21 problemas que Richard Karp demuestra como np-completos el 72 (`https://es.wikipedia.org/wiki/Lista_de_21_problemas_NP-completos_de_Karp`).

Problema Un almacén de productos solicita una consultoría con tu equipo para que le ayuden a resolver el siguiente problema.

Todos los años, los robots del almacén deben ser actualizados. Actualizar robots es caro, requiere traer a estudiantes de doctorado del DCC, tenerlos todo el día en la bodega. Pero además no se pueden actualizar todos los robots al mismo tiempo, por que hay restricciones operativas, siempre tienen que haber robots capaces de cubrir todo el flujo de operación.

Entonces, el problema se puede abstraer así. Hay una lista $P = \{p_1, \ldots, p_{|P|}\}$ de robots. Una lista $D = \{d_1, \ldots, d_{|D|}\}$ de días en los que los estudiantes del DCC pueden venir a actualizarlos, y un conjunto $F = \{f_1, \ldots f_{|D|}\}$ de listas de robots, donde cada $f_i \subseteq P$ dice que en el día d_i ninguno de los robots en f_i se pueden actualizar.

La meta es organizar la actualización de forma de minimizar la cantidad de días totales en los que se están actualizando los robots: encontrar el subconjunto $\hat{D}$ de D más pequeño de forma que se puede asignar cada robot p_i en P a al menos un día d_j en $\hat{D}$ en donde p_i no esté en f_j.

Primera parte Dados P, D y F como arriba, una calendarización para (P, D, F) es una función $c : P \to D$ de tal forma que para todo $p_i \in P$, si $c(p_i) = d_j$, entonces p_i no está en f_j. Decimos que la calendarización *usa* k días si el recorrido de la función c es de tamaño k, es decir, si hay exactamente k días d_j en D para los que existe un p_i tal que $c(p_i) = d_j$.

- Muestra que el siguiente lenguaje L es NP-completo[1].

$$L = \{(P, D, F, k) \mid \text{ existe una calendarización para } (P, D, F) \text{ que usa } k \text{ días}\}.$$

- Considera el problema de recibir una tupla (P, D, F) y encontrar la cantidad mínima de días necesarios para calendarizar (P, D, F). Muestra que existe un algoritmo polinomial para este problema si y solo si $P = NP$.

[1] recuerda que los problemas son lenguajes sobre un alfabeto, por lo que si fuéramos súper formales, deberíamos trabajar con codificaciones de P, D, F y k en algún alfabeto fijo, por ejemplo $\{0, 1\}$.

Segunda parte OK, a la empresa del almacen le queda claro que va a estar difícil encontrar la menor cantidad de días. Pero te preguntan ahora para ver alguna métrica para ver cuando saber si la actualización va a ser compleja. En particular, considera ahora el lenguaje

$$L = \{(P, D, F, k) \mid \text{ toda calendarización para } (P, D, F) \text{ usa al menos } k \text{ días}\}.$$

Argumenta por que tampoco deberías poder encontrar un algoritmo polinomial para esto.

Formato de entrega Aceptamos documentos pdf escrito en latex, o, excepcionalmente, imágenes escaneadas o fotografiadas en buena calidad. La nota de esta tarea va desde un 1 a un 7.