Complejidad Computacional

Problemas y Algoritmos

Autor: Fausto David Hernández Jasso

September 11, 2022

1 Introducción

Los problemas y los algoritmos que lo resuelven *(si es que existe alguno)* actualmuente pueden ser formalizados y analizados matemáticamente. Consecuentemente, podemos pensar en las siguientes preguntas:

- **¿Qué es un algoritmo?**
- **¿Qué es un problema?**

A lo largo de éste texto daremos respuesta a las preguntas anteriores. Naturalmente podemos pensar en un problema de la siguiente forma:
Es una situación a la cual le queremos dar una solución, pero aún no sabemos cuál es dicha solución.
En términos simples, es algo que queremos resolver, sin embargo, en **complejidad computacional**, los problemas no son solamente cosas que tenemos que resolver sino que también pensamos en los problemas como entes matemáticos que son interesantes por sí mismos. Mientras que podemos pensar en un **algoritmo** como un método detallado paso a paso que nos sirve para resolver un problema.

2 Desarrollo

Hablaremos acerca de dos problemas, el primero es conocido como **graph reachability** *(en español se podría traducir como el alcance de una gráfica pero preferimos usar el término en su idioma original)*, el problema consiste en lo siguiente:
Dada una gráfica $G = (V, E)$ donde V es un conjunto finito de vértices y E es un conjunto de aristas *(pares de vértices)* y sean $u, v \in V$ **¿existe una ruta desde el vértice** u **hacia el vértice** v**?**.
Notemos que éste problema tiene un universo infinito **(numerable pero aún así infinito)** de instancias. Cada instancia de éste problema será tratado como un ente matemático, en el cual esperamos contestar la pregunta hecha en el párrafo anterior. Observamos que el problema no nos está solicitando que demos alguna ruta del vértice u al vértice v, sólo nos pide determinar sí dicha ruta existe, es decir, dada cualquier gráfica G tenemos que **decidir** sí para $v, u \in V$ existe una ruta entre ellos. Intituivamente llamaremos a ésta clase de problemas como **problemas de decisión**.
Para el problema anterior, existe un algoritmo que lo resuelve, y dicho algoritmo funciona de manera eficiente. Consecuentemente tenemos que definir qué es que un algoritmo sea eficiente. Sea A un algoritmo que resuelve el problema P, entonces A es eficiente sí y sólo sí A es de orden polinomial. Es decir, A es eficiente en cuanto al tiempo sí el función que describe cuánto tiempo tarda en ejercutarse A en el peor de los casos, $f(n)$ es un polinomio. Contamos el peor de los ya que podemos encontrarnos con instancias de los problemas que sean muy difíciles de tratar. A pesar de que consideremos que un algoritmo es eficiente cuando éste está en el orden polinomial no significa que éste sea la mejor opción siempre, supongamos que tenemos un problema P y tenemos los algoritmos A_1 y A_2 que resuelven a P, supongamos que A_1 es eficiente bajo nuestra definición pero A_2 no. Supongamos que $A_1 = \mathcal{O}\left(n^{100}\right)$ y $A_2 = \mathcal{O}\left(n^{\log n}\right)$, a pesar de que A_1 está acorde con nuestra definición A_1 no es muy buena opción para implementarlo ya que sería terriblemente lento, lo cual no pasa con A_2. En cambio, existen diversas razones por las cuales consideremos a los algoritmos de orden polinomial como eficientes, la más importante de éstas es que cualquier función polinomiar eventualmente se ve superada por una función exponencial, con esto, nos damos cuenta que un crecimiento polinomial siempre será superado por un crecimiento exponencial. Ya hemos hablado del tiempo pero ¿y el espacio que usan nuestros algoritmos?, en éste caso queremos que la función de nuestros algoritmo en espacio sea menor o igual que la que describe el tiempo.
El segundo problema es **Maximum Flow**.
Éste problema consiste en que dada una red N, encontrar el flujo del mayor valor posible. Notemos que éste problema no es de decisión, sino de **optimización**, ya que buscamos la mejor solución entre todas las posibles. Éste problema es una instancia del primer problema.
Es posible transformar cualquiero problema de optimización en un problema de decisión, simplemente en lugar de buscar el más valor que queremos damos una cota α y preguntamos sí α puede ser alcanzada.

3 Conclusión

Para un problema computacional P, es deseable que dicho problema sea de orden polinomial en tiempo ya que habría una forma eficiente de resolver a P, sin embargo no siempre existen estos algoritmos para cualquier problema P o en algunos casos, no se han encontrado aún. A pesar de que existen problemas de decisión y problemas de optimización, siempre podemos transformar éstos últimos en los primeros, éstas transformaciones son el pan de cada día de la complejidad computacional.