Análisis y diseño de algoritmos avanzados

Gpo 570

Evidencia 2

Actividad Integradora

Alejandra Coeto Sánchez

A01285221

Monterrey, NL.

30 de noviembre, 2024

Reflexión

Considero que a lo largo de esta evidencia pude aprender varias cosas. Primero que nada, me interesó aprender más sobre CMake y en lugar de utilizar un template, en nuestro equipo decidimos implementar el uso de esta herramienta desde cero. Esto fue considerablemente más sencillo, pero me permitió comprender de mejor manera la manera en que se estructura el código al usar CMake, así como el proceso de *build*, compilación y correr tanto el ejecutable como las pruebas. También me pareció interesante integrar esto con Github Actions, ya que una vez listas las pruebas podemos asegurarnos de que si realizamos algún cambio, este no interfiere con la funcionalidad de alguna función o programa.

Por otra parte, en cuanto a las soluciones que programamos, me tocó enfocarme en el programa de flujo máximo, para lo cual usamos Ford Fulkerson. De esta manera es posible obtener la máxima cantidad de flujo que puede enviarse en un grafo donde cada *edge* tiene un límite de capacidad. Por lo tanto, se utiliza algún algoritmo para explorar, en este caso BFS, y encontrar *augmenting paths*. Adicionalmente, de manera conceptual hay un grafo residual para poder "deshacer" el uso de alguna *edge* y así optimizar el flujo. Personalmente, ya conocía el propósito de este algoritmo, pero no la manera en la que se implementa, por lo que tuve la oportunidad de explorar el algoritmo más a detalle.

En cuanto al resto de problemas, el primero consistía en obtener el *minimum spanning tree*, es decir la manera menos costosa de conectar un grafo. Para esto se utilizó el algoritmo Kruskal, lo cual requiere de la estructura DSU (*Disjoint Set Union)*. Este algoritmo ordena las *edges* y va agregando una al resultado si es que esta une dos nodos diferentes, para lo cual se usa el DSU. El segundo problema es el *Traveling Salesman Problem*, donde se debe hallar la ruta de menor costo que une todos los nodos y regresa al punto inicial. Para esto, se utilizó DP (*Dynamic Programming)* y así reutilizar los resultados que ya habían sido calculados. Finalmente, para el último problema se realizó una búsqueda lineal, la cual es bastante directa pero eficiente.