Identificación:

Santiago Pardo - 202013024

Kevin Cohen - 202011864

Algoritmo de Solución

El problema B busca hallar el diferencial mínimo que resulta de la suma de n grafos BC (Bipartitos Conexos) que son disyuntos por pares. Para sumar dos grafos BC diferentes, la manera más eficiente es la adición de un solo arco, pues conecta ambos grafos y dependiendo de par de vértices conecte, se termina minimizando la diferenciabilidad.

Para el algoritmo que desarrollamos tuvimos que crear una clase para representar el grafo. Esta clase se llama “Graph” y consta de dos argumentos: “Vertex” y “Edges”, los cuales representan los vértices (en forma de un ArrayList de Integer) y los ejes (en forma de un HashMap de Integer y ArrayList de Integer, donde cada llave es un vertice y su valor es una lista de adyacencias de este) de manera respectiva. Esta decisión se tomo con el fin de facilitar el proceso de escritura de código, simplificando mucho la replicación. Igualmente, para evitar que los vértices se repitieran entre grafos, se opto por cambiar su número indicador por la suma de este con la cantidad de vértices que se habían revisado hasta el momento, es decir, si en el grafo 1 tenemos 3 vértices (0,1,2) y en el grafo 2 tenemos otros 3 (0,1,2) a estos últimos se les adiciona la cantidad de vértices de todos los anteriores grafos (3,4,5). Como último paso, nos valemos de una función que desarrollamos llamada convert, para pasar la entrada que es un arreglo de int a una instancia de nuestra nueva clase Graph. Este proceso es iterativo.

Una vez se planteó esto, la función principal recibirá una lista de instancias de Graph llamada grafos. En dicha función, manejaremos una instancia de “Graph” llamada grafoControl e int minimal, para llevar registro del grafo mínimo y su valor de diferencial. También se inicia la variable grafo1 con el valor del primer grafo del arreglo grafos, para después iniciar un ciclo que recorre todos los grafos. En este ciclo fundamentalmente se juntan el grafo1 y el grafo2 de todas las maneras posibles, para luego buscar aquella con menor diferencial y continuar iterativamente el proceso con todos los grafos haciendo uso de las variables grafoControl y minimal. Este ciclo funciona pues el grafoControl reemplaza el valor del grafo1 y el grafo2 itera en el arreglo.

Como último detalle importante, la función para identificar el grafo Bipartito y su diferencial se llama Bip y funciona de la siguiente manera. Se basa en el algoritmo DFS en el que utilizamos una pila para ir recorriendo todos los vertices del grafo, según los ejes que tiene los mismos. Además de esto, utilizamos un arreglo de booleanos para registrar los colores de cada vertice, esto con el fin de determinar si es bipartito y a su vez el diferencial. Para saber los colores, asignamos arbitrariamente el primero y los demás deben ser del color opuesto, en caso de que no se cumpla el grafo no es Bipartito. Para saber el diferencial solo necesitamos el valor absoluto de la resta entre las cardinalidades de los dos colores.

Análisis de Complejidades

Para la complejidad espacial tenemos que revisar las estructuras de Datos utilizadas, las cuales son:

Un arreglo de tamaño V para cada grafo representando los vertices, al igual que un HashMap de tamaño V*E que representa los ejes.

Varios Arreglos y HashMap auxiliares que comparten complejidad con los presentados anteriormente.

Entonces podríamos decir que la complejidad sería de O(NV+NE) donde N representa la cantidad de grafos que utilizamos.

Para la complejidad temporal:

Este elemento es un poco más complejo que el anterior, esto debido a la manera en que recorremos cada pareja posible de vértices entre 2 grafos diferentes, las cuales se van sumando entre sí con tal de poder llegar a sumar todos los grafos que se entregan. En promedio, cada grafo tiene una cantidad de vértices V, entonces tendríamos en esa parte una complejidad V^2. Dicha complejidad se suma a la del DFS modificado que implementamos, la cual no difiere mucho de la implementación original que sería de V+E (vértices y ejes) adicionando la implementación de los colores que nos da una complejidad de V.

Entonces la complejidad total sería:

V^2 +V+E+V

Que en notación O sería:

O(V^2)

Comentarios Finales

La solución implementada se puede considerar como una de fuerza bruta, pues terminamos revisando todas las posibles combinaciones de Grafos para encontrar la suma con el mínimo diferencial, por lo que la complejidad termina siendo mayor que otras implementaciones del estilo, A pesar de esto, dicha implementación posiblemente más eficiente termina requiriendo más espacio en memoria, por lo que el balance final termina siendo similar (Suponiendo una posible implementación por programación dinámica).