Problema B

0. Identificación:

Autores del Problema B

- Tony Santiago Montes Buitrago [202014562] <t.montes@uniandes.edu.co>
- Juan Carlos Marin Morales [202013973] <j.marinm@uniandes.edu.co>

1. Algoritmo de Solución:

El problema de la BC-Suma se dividió en 2 subproblemas que cubren todos los requerimientos del problema. Para esto inicialmente después de varias pruebas denotamos que para calcular el diferencial mínimo del grafo BC-Suma no era necesario calcular más que el diferencial de cada uno de los grafos BC-Sumandos. Para esto se definió una función que calcula el diferencial de cada BC-Sumando.

Para el cálculo del diferencial del grafo, se tomó un vértice cualquiera del grafo completo y se eliminó, junto con todos sus vértices conectados; dichos vértices conectados se agregaron a una lista de los vértices coloreados. Sobre los vértices restantes se realiza el mismo proceso hasta que ya no queden nodos por revisar. Cada vez que se debe tomar un vértice nuevo, se suma 1 a la longitud de los vértices no coloreados. Finalmente se aplica $df = (|coloured| - |not\_coloured|)$

Luego de calcular el diferencial de cada grafo, lo que se hace es hallar la "mínima suma absoluta" de la lista de diferenciales, que es tomar cada número y sumarlo o restarlo para conseguir el número más cercano a 0 en valor absoluto. Este algoritmo fue el más difícil de resolver, dado que a pesar de que se consiguieron varias aproximaciones Greedy (incluso muy cortas, como la del anexo 1), no había una que funcionara para todos los casos, y que no tomara tiempos exponenciales.

Finalmente se consiguió un algoritmo simple y rápido que realiza el siguiente proceso:

1. Ordenar el arreglo de menor a mayor.
2. Obtener la suma de la lista e ir agotando la lista ordenada (pop) restando los valores hasta obtener el mínimo.

En este último paso es importante recalcar dos cosas; primero, que al poner un valor de la lista negativo (restarlo) este se multiplica por 2 al restar (ej. $sum([1,2,3]) = 6\,; sum([1,-2,3] = 6 - 2 \cdot 2 = 2)$); y segundo, que cabe la posibilidad de que, al ser un valor absoluto, la mínima suma absoluta sea un valor negativo, por ende, es importante hacer una revisión negativa y positiva y luego hallar el mínimo absoluto de estas.

Métodos:

1. `dif_graph`

Precondición: "El grafo es bipartito" = "Existe algún subgrafo tal que dos elementos no compartan arcos"

$$Q_1 : \{\ (\exists\ subg : 2^{g.v} \mid : (\forall\ \langle a, b\rangle : g.e \mid : (a \in subg \Rightarrow b \notin subg) \wedge (b \in subg \Rightarrow a \notin subg)))\ \}$$

Postcondición: $R_1 : \{\ df = \big||g.v| - 2 \cdot |subg|\big|\ \}$

2. `min_dif`

Precondición: $Q_2 : \{\ (\forall\ x : d \mid : x \geq 0)\ \}$

Postcondición: "La mínima resta absoluta de 2 particiones de la lista (p y $d \setminus p$)"

$$sum(set) = (+i : set \mid : i)$$

$$R_2 : \{\, m = \left(min\, p : 2^d \middle| : |sum(p) - sum(d \setminus p)|\right) \}$$

2. Análisis de Complejidad:

a. Temporal: $\boldsymbol{O\left(n \cdot \left(\frac{v \cdot e}{4} + logn\right)\right)}$

1. Para hallar el diferencial de cada grafo, se recorren todos los vértices y sobre estos se recorren todos los arcos; sin embargo, sobre cada iteración de los vértices se elimina (en el peor caso) un arco y un vértice por ende cada vez serán menos iteraciones. Además, remover un elemento toma $O(1)$ ya que se utilizó la estructura de datos set [2].

De este modo, sacando un promedio de iteraciones en el peor caso se obtiene: $O\left(\frac{v}{2} \cdot \frac{e}{2}\right) = O\left(\frac{v \cdot e}{4}\right)$ para cada grafo.

2. Para calcular la mínima suma absoluta, se realiza un ordenamiento (Python toma $O(n \cdot logn)$ para ordenar, ya que efectúa merge sort [1]) y luego se hace un recorrido simple por la lista $O(n)$.

Finalmente, la complejidad es: $O\left(n \cdot \left(\frac{v \cdot e}{4} + logn\right)\right)$

* NOTA: n es el número de grafos BC-Sumandos, e es el máximo de ejes y v es el máximo de vértices.

b. Espacial: ***O(n)***

La única lista que se utiliza es la lista de diferenciales. Un diferencial por cada grafo.

* NOTA: n es el número de grafos BC-Sumandos.

3. Comentarios Finales:

Se puede evidenciar y concluir que la solución propuesta es bastante eficiente para los dos algoritmos. Ya que, en el algoritmo del cálculo del diferencial se implementó un algoritmo que reduce el número de ejes y vértices a revisar (en promedio más de una vez) lo cual disminuye mucho el tiempo de complejidad en grafos densos; y, por otra parte, el algoritmo de cálculo del mínimo no llega a ser cuadrático y menos exponencial, razón por la cuál logra una excelente aplicación sin necesidad de ocupar espacio adicional.

4. Bibliografía:

[1] Time Complexity Python. Disponible en: https://wiki.python.org/moin/TimeComplexity

[2] Python Set Discard (Remove) Complexity. Disponible en: https://blog.finxter.com/python-set-discard/

5. Anexos:

```
@timer()
def min_dif(d:list) -> int:
```

```
    """Dados los diferenciales de todos los grafos (difs) hallar el mínimo
diferencial del BC-Suma"""
    d.sort()
    f:int = d.pop()
    while d:
        f = abs(f-d.pop())
    return f
```

Anexo 1. Aproximación Greedy a la minima suma absoluta.