3.1 Escriban una explicación entre 3 y 6 líneas de texto **del código del numeral *1***. Digan cómo funciona, cómo está representado el mapa de la ciudad, por ejemplo, utilizaron matrices, listas, tablas de hash, ¿por qué? Pueden apoyarse realizando una imagen para facilitar su explicación.

Nota: La explicación del código debe ir dentro del informe y no dentro del código

Para resolver este problema usamos tablas de hash, dentro de esta hay una pareja como key la cual es **Id** del vértice y **nombre** del vértice, y el valor es una LinkedList de parejas la cual es el **ID** del otro vértice y a que **distancia** esta de ese vértice. Usamos bufferReader y un StringTokenizer para poder separar los datos dentro del txt, además usamos Longs en vez de int porque algunas id eran demasiado grandes para poder ser almacenadas en un int.

3.2 Si representamos el mapa de Medellín del numeral 1 con matrices de adyacencia, ¿Cuánta memoria consumiría? Tengan en cuenta que hay alrededor de 300,000 vértices.

Una matriz de adyacencia es una la cual se comporta por el numero de elementos al cuadrado por lo que si son 300,000 serian: 90’000.000.000

3.3 ¿Cómo solucionaron el problema de que los identificadores de los puntos del mapa no empiezan en cero?

No nos ocurrió este problema, no se si fue debido a que utilizamos long en vez de int.

3.4 Expliquen con sus propias palabras la estructura de datos que utilizan para resolver el problema, y cómo funcionan los algoritmos realizados en el numeral 2.1 y los ejercicios opcionales que hayan hecho del punto 2. Esto en 3 a 6 líneas de texto.

Para la implementación de los grafos usamos matrices de adjacencia. Inicializamos todos pesos en 0, asi sabemos que si dos vertices no estan conectados tendran peso 0.

El metodo para determinar si es Bicolorable o no, se basa en la teoria de grafos, con los grafos bipartitos La clase a travez de una lista saca los sucesores de cada nodo y revisa por cada sucesor que los sucesores de el mismo no esten en los sucesores del nodo inicial.

3.5 Calculen la complejidad del ejercicio 2.1 y, si los hicieron, de los Ejercicios Opcionales.

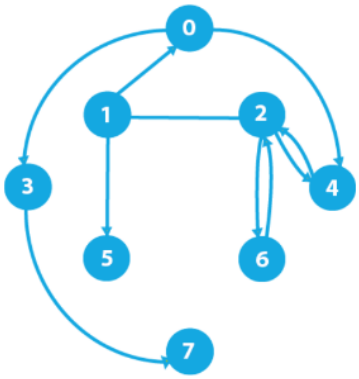
El algoritmo que estamos utilizando revisa los sucesores de cada nodo para ver si coinciden con los sucesores de otro de los nodos, teniendo esto en cuenta, la complejidad del mismo dependeria de la cantidad de nodos y de la cantidad de sucesores de cada nodo, entonces esta estaria dada por n*m.

3.6 Expliquen con sus palabras las variables (*qué es 'n', qué es 'm', etc.*) del cálculo de complejidad del numeral 3.5. Ver ejemplo a continuación:

n: es la cantidad de nodos del grafo.

m: es la cantidad de sucesores de cada uno de los nodos.

4.1 **[Opc]** Consideren el no hay arco, por simplicidad, deje el espacio en blanco.



	0	1	2	3	4	5	6	7
0				1	1			
1	1		1			1		
2		1			1		1	
3								1
4			1					
5								
6			1					
7								

4.2 Para el mismo grafo, completen la representación de **listas de adyacencia**. Como el grafo no tiene pesos, sólo se colocan los sucesores en la lista de adyacencia.

0 ->[3,4]

1 ->[0,2,5]

2 ->[1,4,6]

3 ->[7]

4 ->[2]

5 ->[]

6 ->[2]

7 ->[]

4.3 ¿Cuánta memoria (ojo, no tiempo sino memoria) ocupa una representación usando listas de adyacencia para un grafo dirigido con *n* vértices en el peor de los casos?

a) $O(n)$
b) $O(n^2)$ ←
c) $O(1)$
d) $O(\log n)$
e) $O(n.\log n)$