3.2) lo que hacen es, en el AM (que utiliza matrices) lo que hace es crear una matriz de size*size, y en esta le va agregando los valores de si existe o no un arco entre ellos, mientras que el AL (el de listas) lo que hace es agregar a una lista la pareja con el peso del arco, por lo que tiene una complejidad diferente.

3.3) para grafos con pocos arcos es mejor utilizar el de listas anexadas, ya que como son pocos, se pueden agregar rápido y buscar rápido y no gastaría mucha memoria, mientras que, si son muchos, es mejor usar la matriz, ya que como son muchos significa que la matriz va a estar llena de datos y podrá buscar las relaciones de manera más fácil.

3.4 para el mapa de Medellín, lo mejor sería usar listas, ya que hay mucha gente en Medellín que no tiene nada que ver con otra, por lo que crear una mega matriz para tener muchos espacios vacíos es muy ineficiente.

3.5) depende del uso que se le quiere dar, pero la mayoría del tiempo es mejor usar listas, ya que normalmente hay nodos que tienen muy pocas relaciones, son pocos los casos que todos los nodos se relacionen entre si

3.6) lo mejor sería usar listas, ya que lo que hace es calcular el peso del arco, y además no necesita compararlo con todos los otros dispositivos electrónicos existentes, simplemente con los que tiene una relación el dispositivo

3.7)

```
import java.util.*;

public class BiColor // C0

{

   List<Integer> numeros = new ArrayList<Integer>(); // C1

   private int size; // C2

   private int numeroDeArcos; // C3

   private int[][] matriz; // C4

   private int[] BicolorMatriz; // C5

   private int size() // C6

   {

      Scanner reader = new Scanner(System.in); // C7

      size = 0; // C8


      System.out.println("Introduce el número de nodos y el cero para salir"); // C9
```

```
        try { // C10
            size = reader.nextInt(); // C11

        } catch (InputMismatchException ime){ // C12
            System.out.println("¡Cuidado! Solo puede insertar números. "); // C13
            reader.next(); // C14
        }
    matriz = new int [size][size]; // C15
    BicolorMatriz = new int [size]; // C16
    return size; // C17
}
private void Arcos() // C18
{
    Scanner reader = new Scanner(System.in); // C19
    numeroDeArcos = 0; // C20

    System.out.println("introdusca el numero de relaciones"); // C21


        try { // C22
            numeroDeArcos = reader.nextInt(); // C23

        } catch (InputMismatchException ime){ // C24
            System.out.println("¡Cuidado! Solo puede insertar números. "); // C25
            reader.next(); // C26
        }

        for(int i=0;i<=numeroDeArcos-1;i++){ // T(n)+C27
            int primerNodo=0; // T(n)+ C28
```

```
        int segundoNodo=0; // T(n)+C29
        System.out.println("introdusca el nodo que quiere relacionar"); // T(n)+C30
        Scanner reader1 = new Scanner(System.in); // T(n)+C31
        try { // T(n)+C32
            primerNodo = reader1.nextInt(); // T(n)+C33

        } catch (InputMismatchException ime){ // T(n)+C34
           System.out.println("¡Cuidado! Solo puede insertar números. "); // T(n)+C35
           reader.next(); // T(n)+C36
        }
        System.out.println("Este nodo se relaciona con:"); // T(n)+C37
        Scanner reader2 = new Scanner(System.in); // T(n)+C38
        try { // T(n)+C39
            segundoNodo = reader2.nextInt(); // T(n)+C40

        } catch (InputMismatchException ime){ // T(n)+C41
           System.out.println("¡Cuidado! Solo puede insertar números. "); // T(n)+C42
           reader.next(); // T(n)+C43
        }
        matriz[primerNodo][segundoNodo] = 1; // T(n)+C44
        matriz[segundoNodo][primerNodo] = 1; // T(n)+C45
      }
}
   private boolean Bicolor() // C46
{
   boolean Bicolor = true; // C47
   for(int k=0; k<=size-1; k++){ // T(m) + C48
      BicolorMatriz[k] = 0; // T(m) + C49
   }
```

```
for(int i = 0; i<=size-1 ;i++){ // T(m) + C50
   for(int j = 0; j<=size-1 ;j++){ // T(m*m) + C51
      if(matriz[i][j] == 1){ // T(m*m) + C52
        if(BicolorMatriz [i] == 0 && BicolorMatriz [j] == 0){ // T(m*m) + C53
           BicolorMatriz [i] = 1; // T(m*m) + C54
           BicolorMatriz [j] = 2; // T(m*m) + C55
        }
        if(BicolorMatriz [i] == 1 && BicolorMatriz [j] == 0){ // T(m*m) + C56
           BicolorMatriz [j] = 2; // T(m*m) + C57
        }
        if(BicolorMatriz [i] == 2 && BicolorMatriz [j] == 0){ // T(m*m) + C58
           BicolorMatriz [j] = 1; // T(m*m) + C53
        }
        if(BicolorMatriz [i] == 0 && BicolorMatriz [j] == 1){ // T(m*m) + C60
           BicolorMatriz [i] = 2; // T(m*m) + C61
        }
        if(BicolorMatriz [i] == 0 && BicolorMatriz [j] == 2){ // T(m*m) + C62
           BicolorMatriz [i] = 1; // T(m*m) + C63
        }
        if((BicolorMatriz [i] == BicolorMatriz [j] ) && BicolorMatriz [j] != 0){ // T(m*m) + C64
           Bicolor = false; // T(m*m) + C65
        }
      }
   }
}
if(Bicolor ==false){ // C66
   System.out.println("NO SE PUEDE BICOLOR"); // C67
}else{ // C68
   System.out.println("SE PUEDE BICOLOR"); // C69
```

```
        }
      return Bicolor; // C70
   }
   public void analizar(){ // C71
      do { // T(k) + C72
         size(); // T(k) + C73
         if(size == 0){ // T(k) + C74
            break; // T(k) + C75
         }
         Arcos(); // T(k) + C76
         Bicolor(); // T(k) + C77
      } while (size!=0); // T(k) + C78
   }
}
```

Siendo n el numero el número de relaciones que va a poner el usuario, m el número de nodos que pone el usuario, y k es el número de veces que el usuario coloca un nuevo tamaño

Por lo que es

3.8) $O(n+m^2+k)$

4.1)

	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)

0 = [3,4]

1 = [0,2,5]

2 = [1,4,6]

3 = [7]

4 = [2]

5 = []

6 = [2]

7 = []

4.3)

b)n^2