Taller individual

Valentina Samaniego

Desarrolle unan aplicación que permita encontrar el árbol de recubrimiento mínimo a partir de un grafo no dirigido. Debe desarrollar el algoritmo de Kruskal o Prim. La aplicación deberá presentar los diferentes valores que se están generando al aplicar el algoritmo (Prueba de escritorio).

```java
import java.util.*;

class Arista implements Comparable<Arista> {
    int u, v, peso;

    public Arista(int u, int v, int peso) {
        this.u = u;
        this.v = v;
        this.peso = peso;
    }

    @Override
    public int compareTo(Arista otra) {
        return Integer.compare(x: this.peso, y: otra.peso);
    }

    @Override
    public String toString() {
        return "(" + u + ", " + v + ") peso: " + peso;
    }
}

public class Kruskal {

    static int[] conjunto;

    public static int buscar(int x) {
        if (conjunto[x] != x) {
            conjunto[x] = buscar(conjunto[x]);
        }
        return conjunto[x];
    }
    public static void fusionar(int a, int b) {
        int i = Math.min(a, b);
        int j = Math.max(a, b);
        System.out.println("Fusionar conjuntos " + i + " y " + j);
        for (int k = 1; k < conjunto.length; k++) {
            if (conjunto[k] == j) {
                conjunto[k] = i;
            }
        }
    }

    public static void mostrarConjuntos() {
        System.out.print(s: "Conjuntos actuales: ");
        for (int i = 1; i < conjunto.length; i++) {
            System.out.print("[" + i + "->" + conjunto[i] + "] ");
        }
        System.out.println();
    }

    public static void main(String[] args) {
        int n = 6; // numero de nodos
        conjunto = new int[n + 1];

        for (int i = 1; i <= n; i++) {
            conjunto[i] = i;
        }

        List<Arista> A = new ArrayList<>();
        A.add(new Arista(u: 1, v: 3, peso: 1));
        A.add(new Arista(u: 3, v: 4, peso: 2));
        A.add(new Arista(u: 6, v: 4, peso: 2));
        A.add(new Arista(u: 2, v: 5, peso: 3));
        A.add(new Arista(u: 3, v: 2, peso: 5));
        A.add(new Arista(u: 1, v: 4, peso: 5));
        A.add(new Arista(u: 2, v: 3, peso: 6));
        A.add(new Arista(u: 3, v: 5, peso: 6));
```

```
        Collections.sort(list: A);

        System.out.println(x: "Lista de aristas ordenadas por peso:");
        for (Arista a : A) {
            System.out.println(x: a);
        }

        System.out.println(x: "\n--- Inicio del algoritmo de Kruskal ---\n");

        List<Arista> T = new ArrayList<>();
        int paso = 1;

        for (Arista e : A) {
            System.out.println("Paso " + paso + ":");
            System.out.println("Evaluar arista " + e);

            int compu = buscar(x: e.u);
            int compv = buscar(x: e.v);

            System.out.println("Buscar conjunto de " + e.u + ": " + compu);
            System.out.println("Buscar conjunto de " + e.v + ": " + compv);

            if (compu != compv) {
                System.out.println(x: "Los nodos pertenecen a conjuntos diferentes. No hay ciclo.");
                fusionar(a: compu, b: compv);
                T.add(e);
                System.out.println(x: "Arista agregada al arbol de recubrimiento.");
            } else {
                System.out.println(x: "Los nodos ya estan conectados. Se forma ciclo. Arista descartada.");
            }

            mostrarConjuntos();
            System.out.println(x: "-------------------------");
            if (T.size() == n - 1) {
                System.out.println(x: "Se han agregado n - 1 aristas. Fin del algoritmo.");
                break;
            }

            paso++;
        }

        System.out.println(x: "\n--- Arbol de recubrimiento minimo ---");
        int pesoTotal = 0;
        for (Arista e : T) {
            System.out.println(x: e);
            pesoTotal += e.peso;
        }
        System.out.println("Peso total del arbol: " + pesoTotal);
    }
}
```

Output - Kruskal (run)

```
Paso 5:
Evaluar arista (3, 2) peso: 5
Buscar conjunto de 3: 1
Buscar conjunto de 2: 2
Los nodos pertenecen a conjuntos diferentes. No hay ciclo.
Fusionar conjuntos 1 y 2
Arista agregada al arbol de recubrimiento.
Conjuntos actuales: [1->1] [2->1] [3->1] [4->1] [5->1] [6->1]
-------------------------
Se han agregado n - 1 aristas. Fin del algoritmo.

--- Arbol de recubrimiento minimo ---
(1, 3) peso: 1
(3, 4) peso: 2
(6, 4) peso: 2
(2, 5) peso: 3
(3, 2) peso: 5
Peso total del arbol: 13
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
package kruskal;

import java.util.*;

class Arista implements Comparable<Arista> {
    int u, v, peso;

    public Arista(int u, int v, int peso) {
        this.u = u;
        this.v = v;
        this.peso = peso;
    }

    @Override
    public int compareTo(Arista otra) {
        return Integer.compare(this.peso, otra.peso);
    }

    @Override
    public String toString() {
        return "(" + u + ", " + v + ") peso: " + peso;
    }
}

public class Kruskal {

    static int[] conjunto;

    public static int buscar(int x) {
        if (conjunto[x] != x) {
            conjunto[x] = buscar(conjunto[x]);
```

```
    }
    return conjunto[x];
}

public static void fusionar(int a, int b) {
    int i = Math.min(a, b);
    int j = Math.max(a, b);
    System.out.println("Fusionar conjuntos " + i + " y " + j);
    for (int k = 1; k < conjunto.length; k++) {
        if (conjunto[k] == j) {
            conjunto[k] = i;
        }
    }
}

public static void mostrarConjuntos() {
    System.out.print("Conjuntos actuales: ");
    for (int i = 1; i < conjunto.length; i++) {
        System.out.print("[" + i + "->" + conjunto[i] + "] ");
    }
    System.out.println();
}

public static void main(String[] args) {
    int n = 6; // numero de nodos
    conjunto = new int[n + 1];

    for (int i = 1; i <= n; i++) {
        conjunto[i] = i;
    }
```

```
List<Arista> A = new ArrayList<>();
A.add(new Arista(1, 3, 1));
A.add(new Arista(3, 4, 2));
A.add(new Arista(6, 4, 2));
A.add(new Arista(2, 5, 3));
A.add(new Arista(3, 2, 5));
A.add(new Arista(1, 4, 5));
A.add(new Arista(2, 3, 6));
A.add(new Arista(3, 5, 6));

Collections.sort(A);

System.out.println("Lista de aristas ordenadas por peso:");
for (Arista a : A) {
   System.out.println(a);
}

System.out.println("\n--- Inicio del algoritmo de Kruskal ---\n");

List<Arista> T = new ArrayList<>();
int paso = 1;

for (Arista e : A) {
   System.out.println("Paso " + paso + ":");
   System.out.println("Evaluar arista " + e);

   int compu = buscar(e.u);
   int compv = buscar(e.v);

   System.out.println("Buscar conjunto de " + e.u + ": " + compu);
   System.out.println("Buscar conjunto de " + e.v + ": " + compv);
```

```
            if (compu != compv) {
                System.out.println("Los nodos pertenecen a conjuntos diferentes. No hay ciclo.");
                fusionar(compu, compv);
                T.add(e);
                System.out.println("Arista agregada al arbol de recubrimiento.");
            } else {
                System.out.println("Los nodos ya estan conectados. Se forma ciclo. Arista
descartada.");
            }

            mostrarConjuntos();
            System.out.println("------------------------");

            if (T.size() == n - 1) {
                System.out.println("Se han agregado n - 1 aristas. Fin del algoritmo.");
                break;
            }

            paso++;
        }

        System.out.println("\n--- Arbol de recubrimiento minimo ---");
        int pesoTotal = 0;
        for (Arista e : T) {
            System.out.println(e);
            pesoTotal += e.peso;
        }
        System.out.println("Peso total del arbol: " + pesoTotal);
    }
}
```