EDA 1

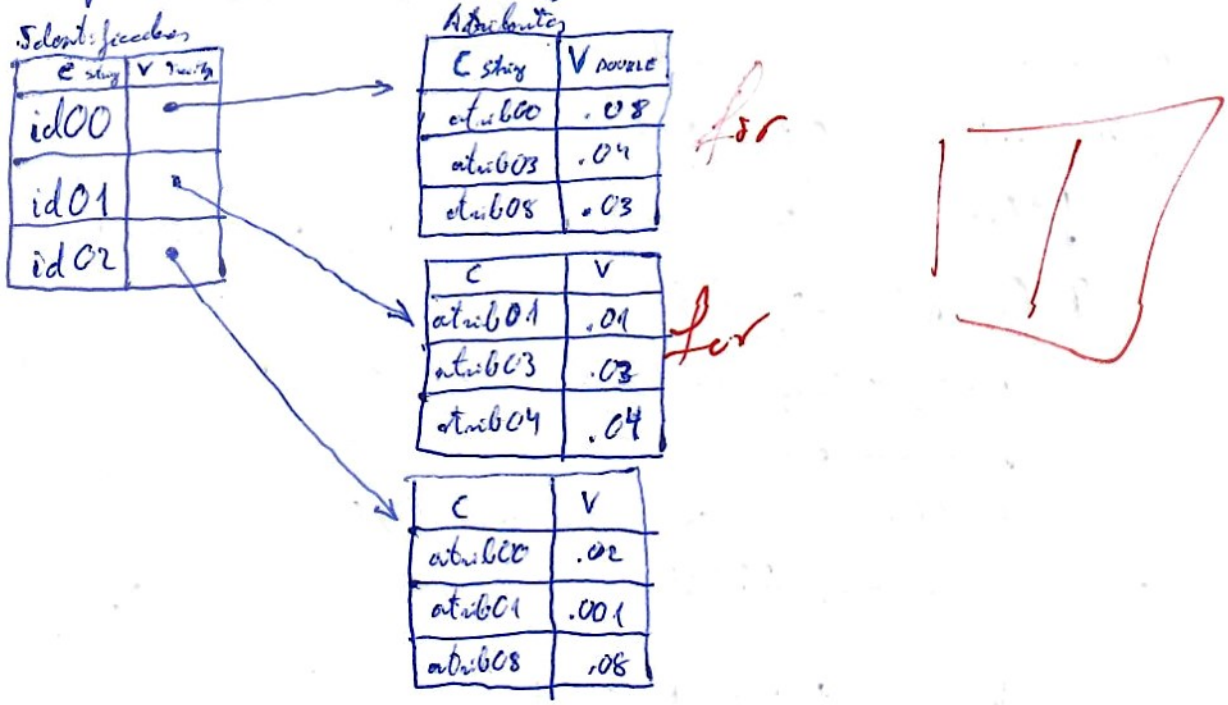
Andrés Xandro López

Ejercicio: Se pide realizar los métodos checkK(), repone() y distanciaEuclidea() de la siguiente tabla

id00	atrib00	.08
	atrib03	.04
	atrib08	.03
id01	atrib09	.01
	atrib03	.02
	atrib04	.09
id02	atrib00	.02
	atrib01	.001
	atrib08	.08

0) Estructura

TreeMap<String, TreeMap<String, Double>> estructura



1) Realiza el método check (String id). Este método debe devolver un boolean (True = 1, false ≠ 1). Para evitar controversia, recomiendo la estructura, si la suma de cada identificador da 1 devuelve true, si da distinto devuelve false. Se debe buscar por id.

```
public boolean check(String id) {
    if(! estructura.containsKey(id)) throw new IllegalArgumentException("No existe");
    TreeMap<String, Double> aux = estructura.get(id);
    double suma = 0.0;
    for (Double probabilidad : aux.values()) {
        suma += probabilidad;
    }
    return (suma == 1.0) ? true : false;
}
```

2) Realiza el método repara (String id). Este método debe reparar el resultado en caso de que el método anterior devuelva false. Debería sumar y dividir el valor para que al realizar la suma da 1.

```
public void repara(String id) {
    if(! estructura.containsKey(id)) throw new IllegalArgumentException("No existe");
    if (check(id) == false) {
        TreeMap<String, Double> aux = estructura.get(id);
        double suma = 0.0;
        for (Double probabilidad : aux.values()) {
            suma += probabilidad;
        }
        for (String k : aux.keySet()) {
            aux.put(k, aux.get(k) * (1.0/suma));
        }
    }
    for (String idActual : estructura.keySet()) {
        if (! idActual.equals(id)) {
            repara(idActual);
        }
    }
}
```

3) Realiza la distancia Euclídea (String idOrigen, idDestino) por su probabilidad, es decir, buscamos en cada identificador un atributo concreto, por ejemplo el atributo CC, si encuentra ese atributo en otro identificador, calculamos su distancia Euclídea.

```
public double distanciaEuclidea (String idOrigen, String idDestino) {
    if (!estructura.containsKey(idOrigen) || (!estructura.containsKey(idDestino)) throw new
        IllegalArgumentException ("No existe el identificador");
    TreeMap<String, Double> origen = estructura.get(idOrigen);
    TreeMap<String, Double> destino = estructura.get(idDestino);
    double suma = 0.0;
    for (String atributo : origen.keySet()) {
        if (destino.containsKey(atributo)) {
            double p1 = origen.get(atributo);
            double p2 = destino.get(atributo);
            suma += Math.pow(p1 - p2, 2);
        }
    }
    return Math.pow(suma, 0.5);
}
```

Se podría recibir a

```
for (String atributo : origen.keySet()){
    if (destino.containsKey(atributo))
    suma += Math.pow(origen.get(atributo)
    - destino.get(atributo), 2);
}
```

(Poco legible y muy largo)

NO

Implementar!!