1. **Graph Edit Distance (GED)**

Finds out how many operations (add/delete node or edge, relabel, etc.) are needed to convert Graph A into Graph B.

Extract a graph from a new image and compare it to a “reference clean water graph” to see how much the structure differs.

Create cost functions to quantify how expensive a certain operation is, eg. Adding node (1.0), deleting node (1.2)

* The library **JGraphT** with custom comparison logic.

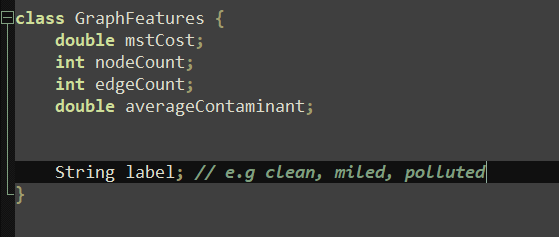
Our cost constants will differ depending on the reason the GED is being calculated (see classification and similarity)

**2. k-Nearest Neighbours (k-NN) Graph Matching**

- create a database (text file) of known clean and polluted water samples, each sample having features such as number of regions, mst cost, average contamination level, Graph Edit Distance (GED) (**measured similarity between two graphs**)

- compare the new samples features (variables) to all known samples and pick the **k closest samples**

- graphs are not just plain values than can be compared, so you can make an object that has the features discussed above (e.g. number of regions, mst cost, etc….)

****

* when comparing the new graph we extract all its features, and compare with all previous graphs, we can use Euclidian distance for this : e.g distance = squaretroot(G1\_FEATURE– G2\_FEATURE)^2 + next feature ^2 etc….
* Store the datasets as a list, iterate through and compare, the label that is the most common wins

Instead of extract features we can use **Graph Edit Distance (GED)** to determine how many operations it would take to convert graph A to graph B

Then we pick the K number of graphs that have the lowest GED operations, the label that appears the most wins