SimpleGraph.cpp

```cpp
//
// Created by Nikolay Yakovets on 2018-01-31.
//

#include "SimpleGraph.h"

SimpleGraph::SimpleGraph(uint32_t n)   {
    setNoVertices(n);
}

uint32_t SimpleGraph::getNoVertices() const {
    return V;
}

void SimpleGraph::setNoVertices(uint32_t n) {
    V = n;
    adj.resize(V);

}

uint32_t SimpleGraph::getNoEdges() const {
    uint32_t sum = 0;
    for (const auto & l : adj)
        sum += l.size();
    return sum;
}

// sort on the second item in the pair, then on the first (ascending order)
bool sortPairs(const std::pair<uint32_t,uint32_t> &a, const std::pair<uint32_t,uint32_t> &b) {
    if (a.second < b.second) return true;
    if (a.second == b.second) return a.first < b.first;
    return false;
}

uint32_t SimpleGraph::getNoDistinctEdges() const {

    uint32_t sum = 0;

    for (auto sourceVec : adj) {

        std::sort(sourceVec.begin(), sourceVec.end(), sortPairs);

        uint32_t prevTarget = 0;
        uint32_t prevLabel = 0;
        bool first = true;

        for (const auto &labelTgtPair : sourceVec) {
            if (first || !(prevTarget == labelTgtPair.second && prevLabel == labelTgtPair.first)) {
                first = false;
                sum++;
                prevTarget = labelTgtPair.second;
                prevLabel = labelTgtPair.first;
            }
        }
    }

    return sum;
}

uint32_t SimpleGraph::getNoLabels() const {
    return L;
}

void SimpleGraph::setNoLabels(uint32_t noLabels) {
    L = noLabels;
}

void SimpleGraph::addEdge(uint32_t from, uint32_t to, uint32_t edgeLabel) {
    if(from >= V || to >= V || edgeLabel >= L)
        throw std::runtime_error(std::string("Edge data out of bounds: ") +
                                         "(" + std::to_string(from) + "," + std::to_string(to) + "," +
                                         std::to_string(edgeLabel) + ")");
    adj[from].emplace_back(std::make_pair(edgeLabel, to));
    //reverse_adj[to].emplace_back(std::make_pair(edgeLabel, from));
}
```

```cpp
void SimpleGraph::readFromContiguousFile(const std::string &fileName) {

    std::string line;
    std::ifstream graphFile { fileName };

    std::regex edgePat (R"((\d+)\s(\d+)\s(\d+)\s\.)"); // subject predicate object .
    std::regex headerPat (R"((\d+),(\d+),(\d+))"); // noNodes,noEdges,noLabels

    // parse the header (1st line)
    std::getline(graphFile, line);
    std::smatch matches;
    if(std::regex_search(line, matches, headerPat)) {
        uint32_t noNodes = (uint32_t) std::stoul(matches[1]);
        uint32_t noLabels = (uint32_t) std::stoul(matches[3]);

        setNoVertices(noNodes);
        setNoLabels(noLabels);
    } else {
        throw std::runtime_error(std::string("Invalid graph header!"));
    }

    // parse edge data
    while(std::getline(graphFile, line)) {

        if(std::regex_search(line, matches, edgePat)) {
            uint32_t subject = (uint32_t) std::stoul(matches[1]);
            uint32_t predicate = (uint32_t) std::stoul(matches[2]);
            uint32_t object = (uint32_t) std::stoul(matches[3]);

            addEdge(subject, object, predicate);
        }
    }

    graphFile.close();

}
```