CS 202 Iditarod - Shaktoolik

Bryan Beus

April 22, 2020

Source Code Link: `https://github.com/siddhartha-crypto/cs202/tree/master/iditarod/06-shaktoolik`

1 Design

1.1 SVG Output - Part II

I designed this assignment in tandem with Part I, as the functionality is very similar between the two.

2 Post Mortem

2.1 SVG Output - Part II

The primary challenge I had was the issue of getting the binary paths to write the path instructions. The solution I found was to save the path to a separate file and then import it into the main file (and delete the separate file) afterwards.

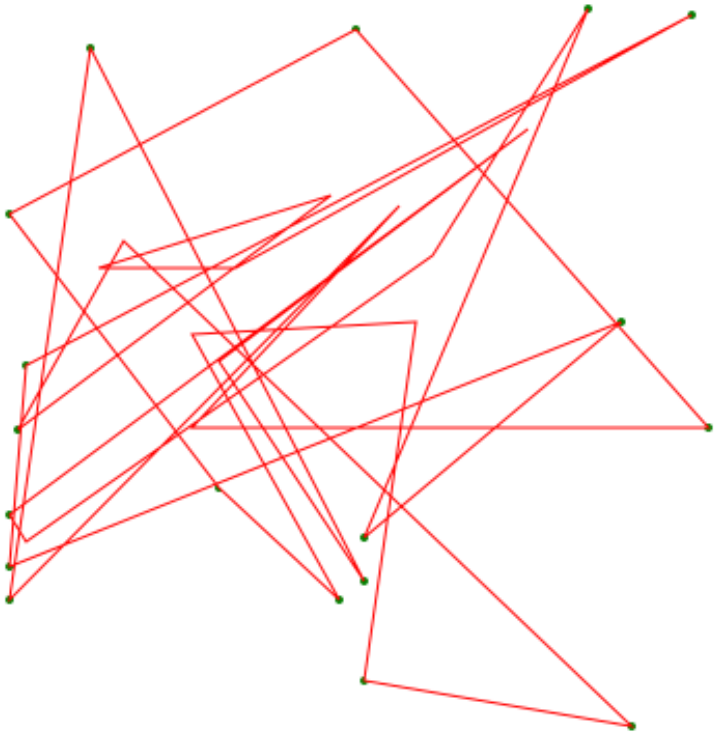
3 Commit History

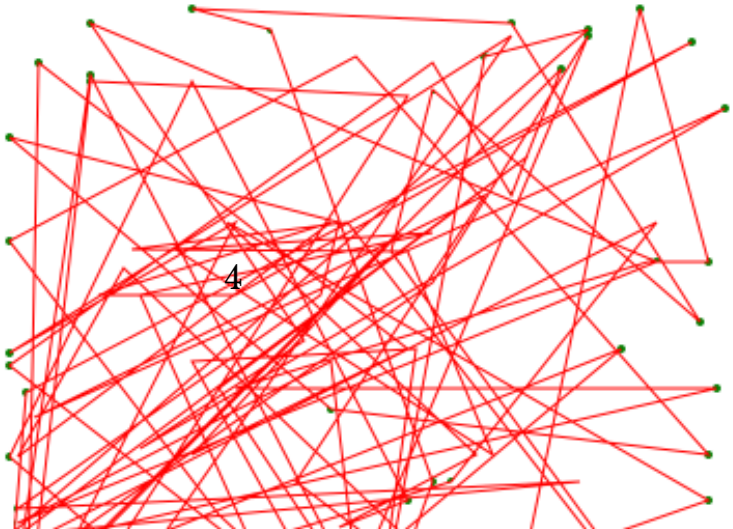
3.1 (Second Half)

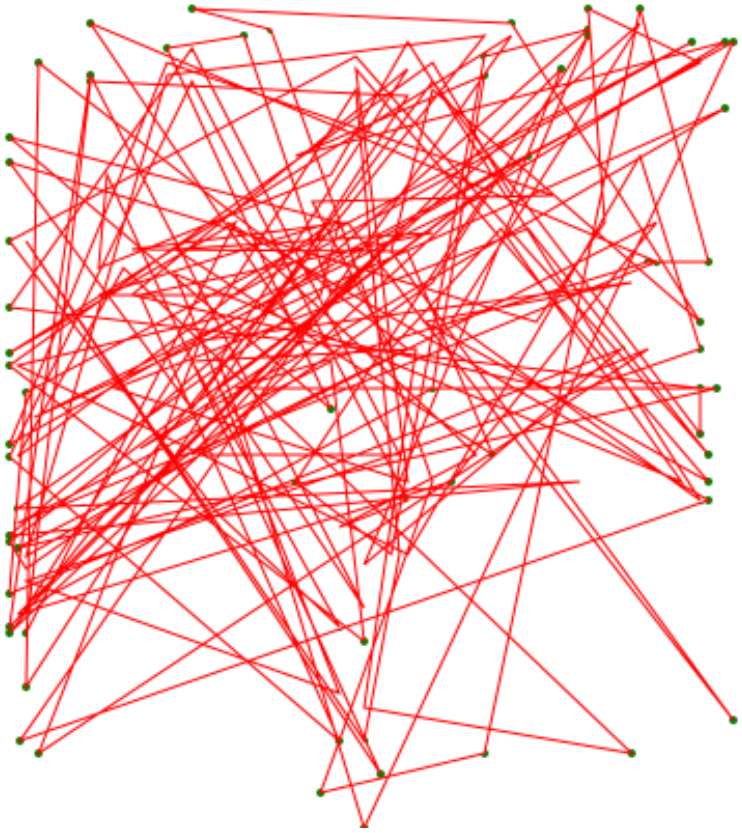
2020-04-22 Update greedy image
2020-04-22 convert Tsp render method to WIP
2020-04-22 extend WIP prints to all three methods
2020-04-22 deactive two of the three WIP solvers
2020-04-22 Reduce WIP count and improve naming accuracy
2020-04-22 Kaltag: Debug and small fixes
2020-04-22 Initiate Shaktoolik
2020-04-22 rename subdir
2020-04-22 rename repo

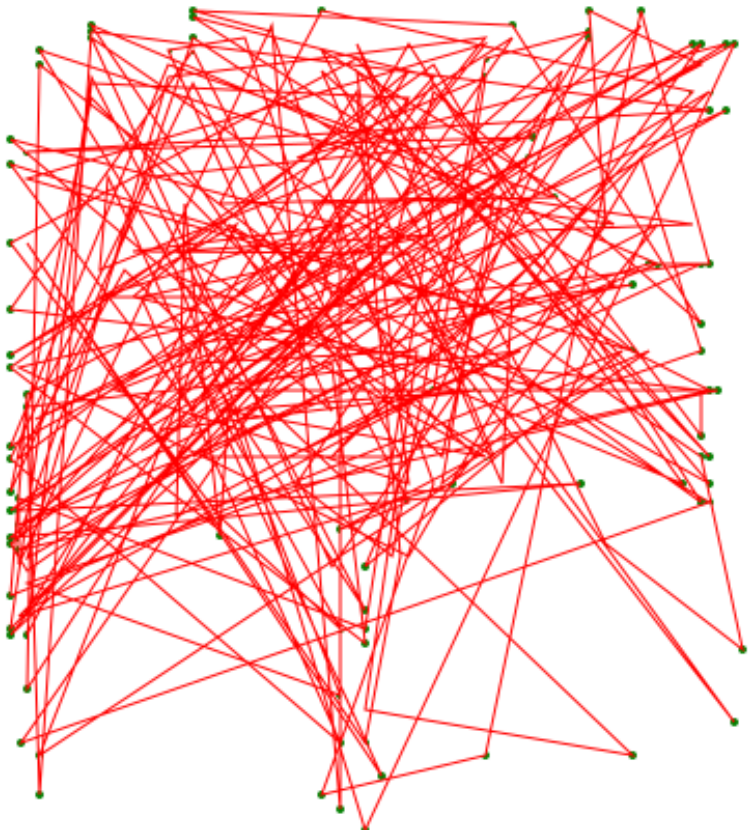
4 Sample Output

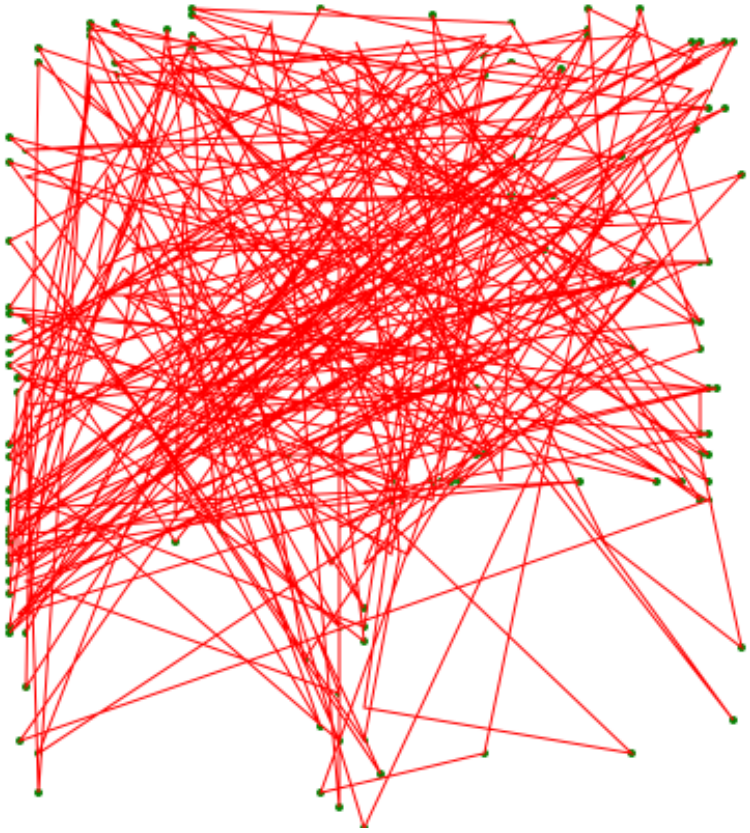
4.1 Shaktoolik

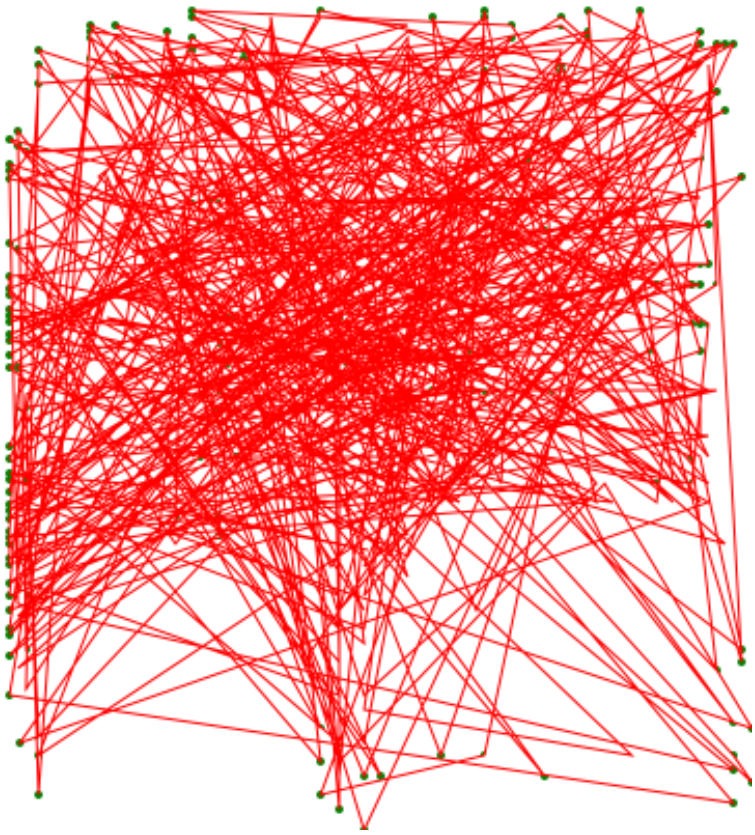


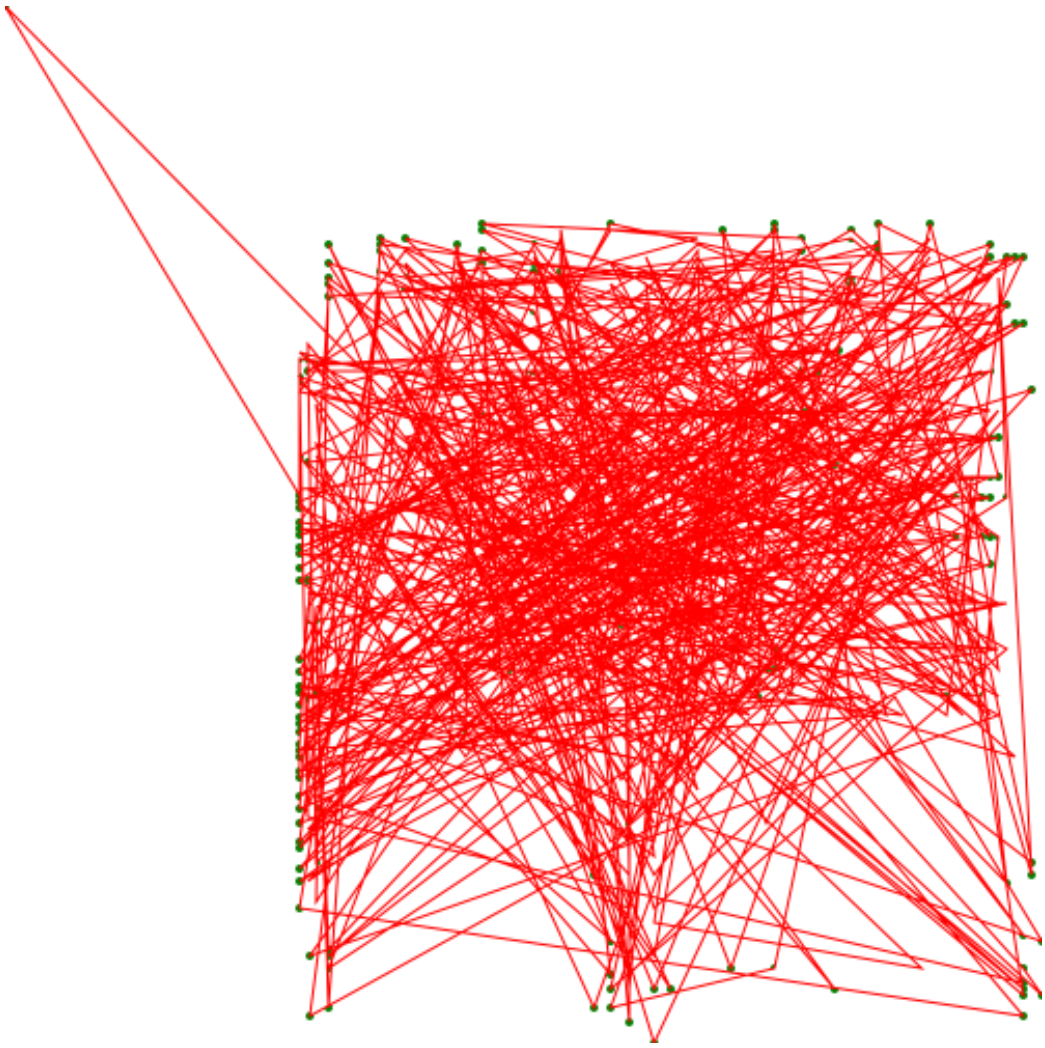


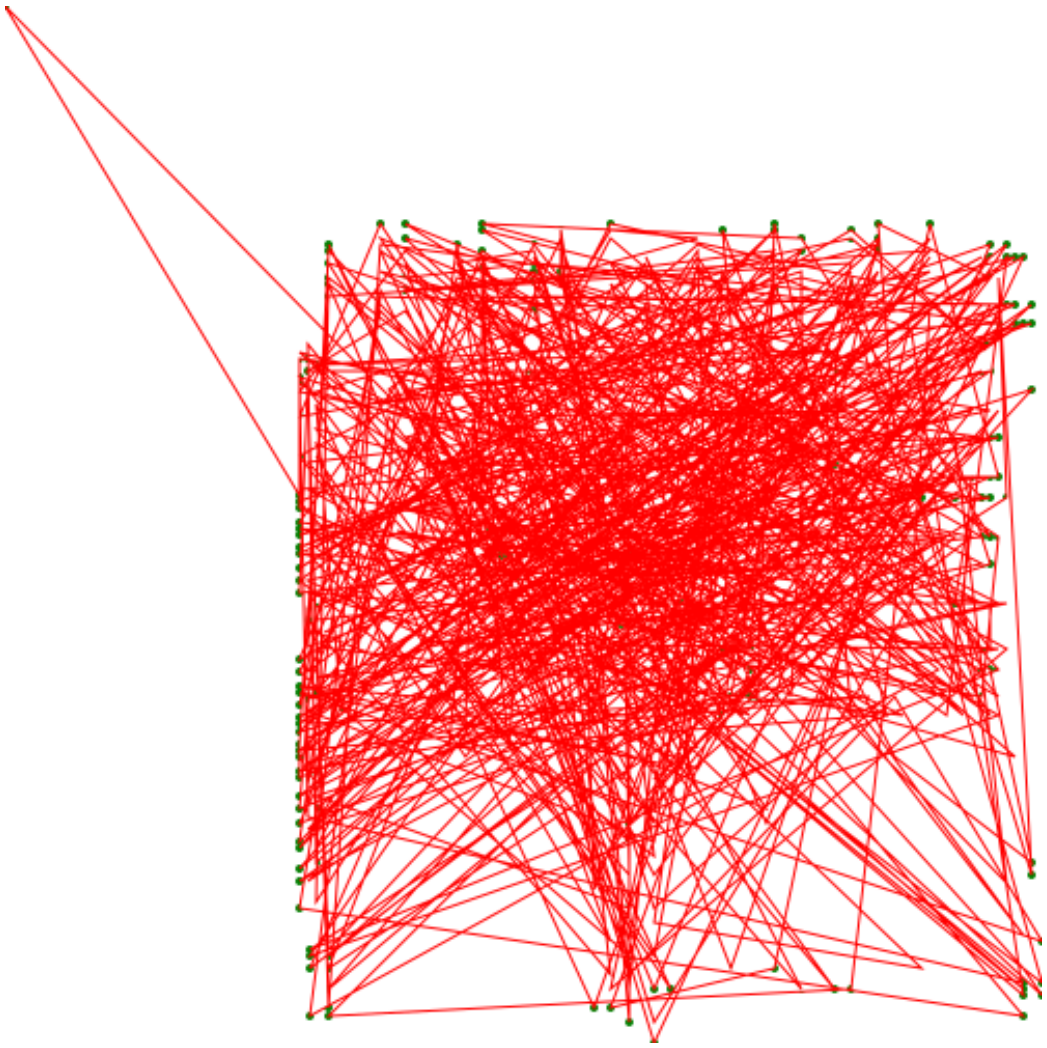


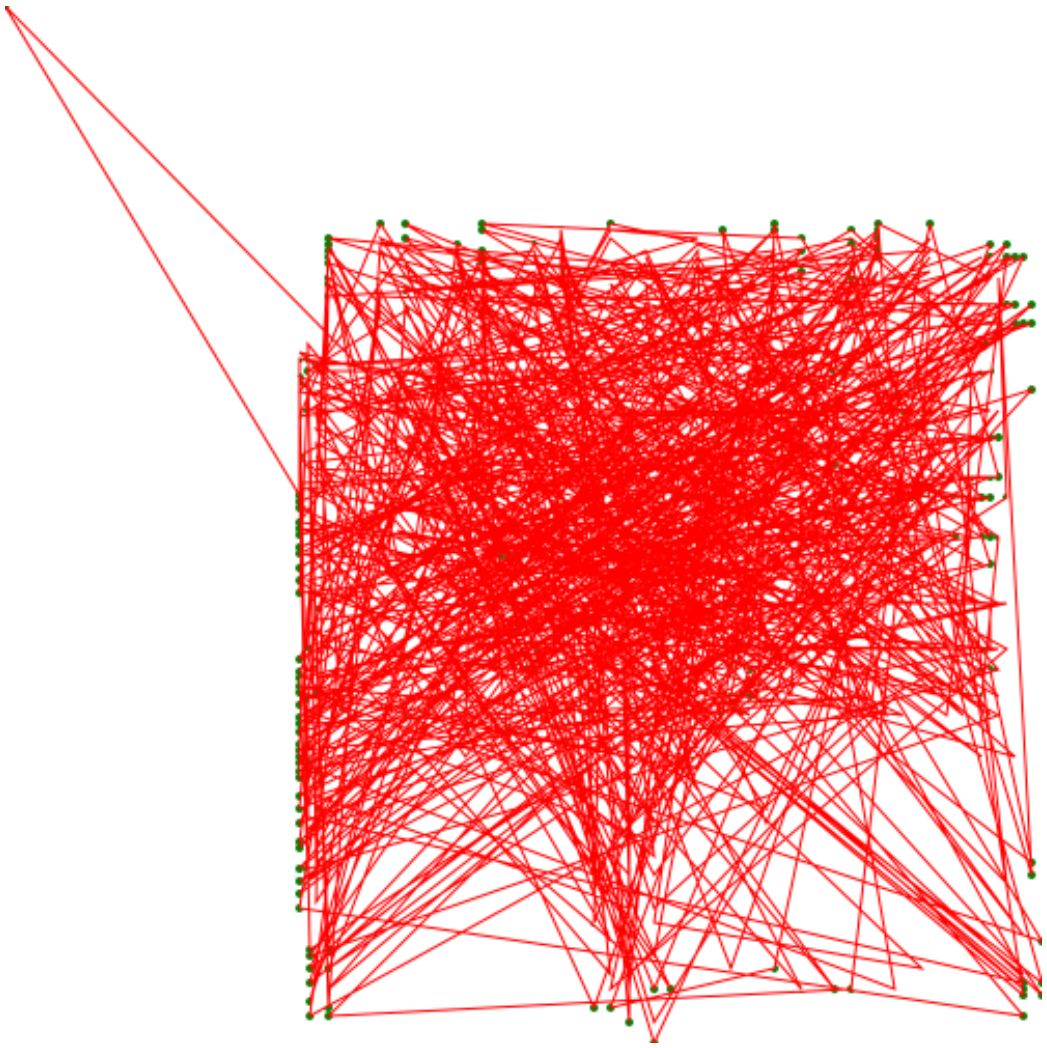


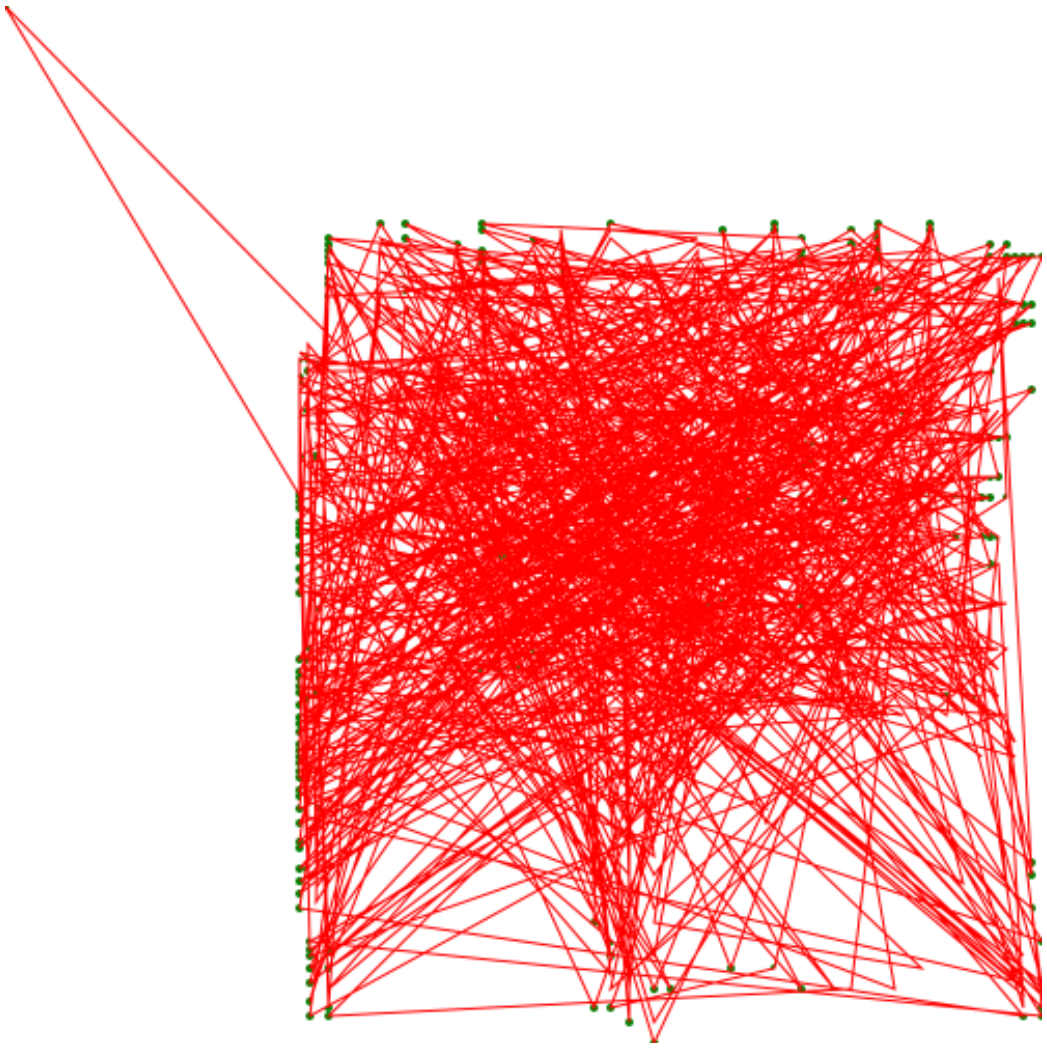


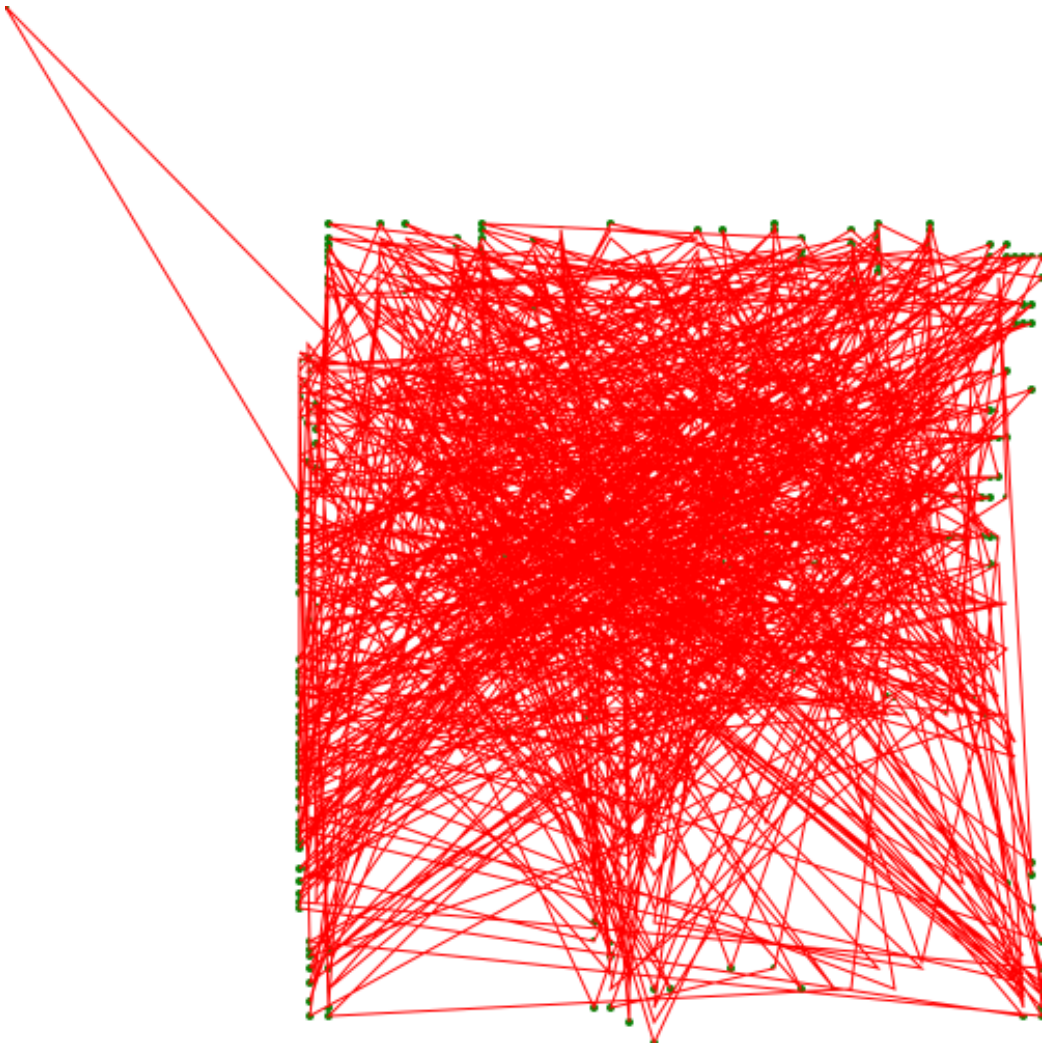


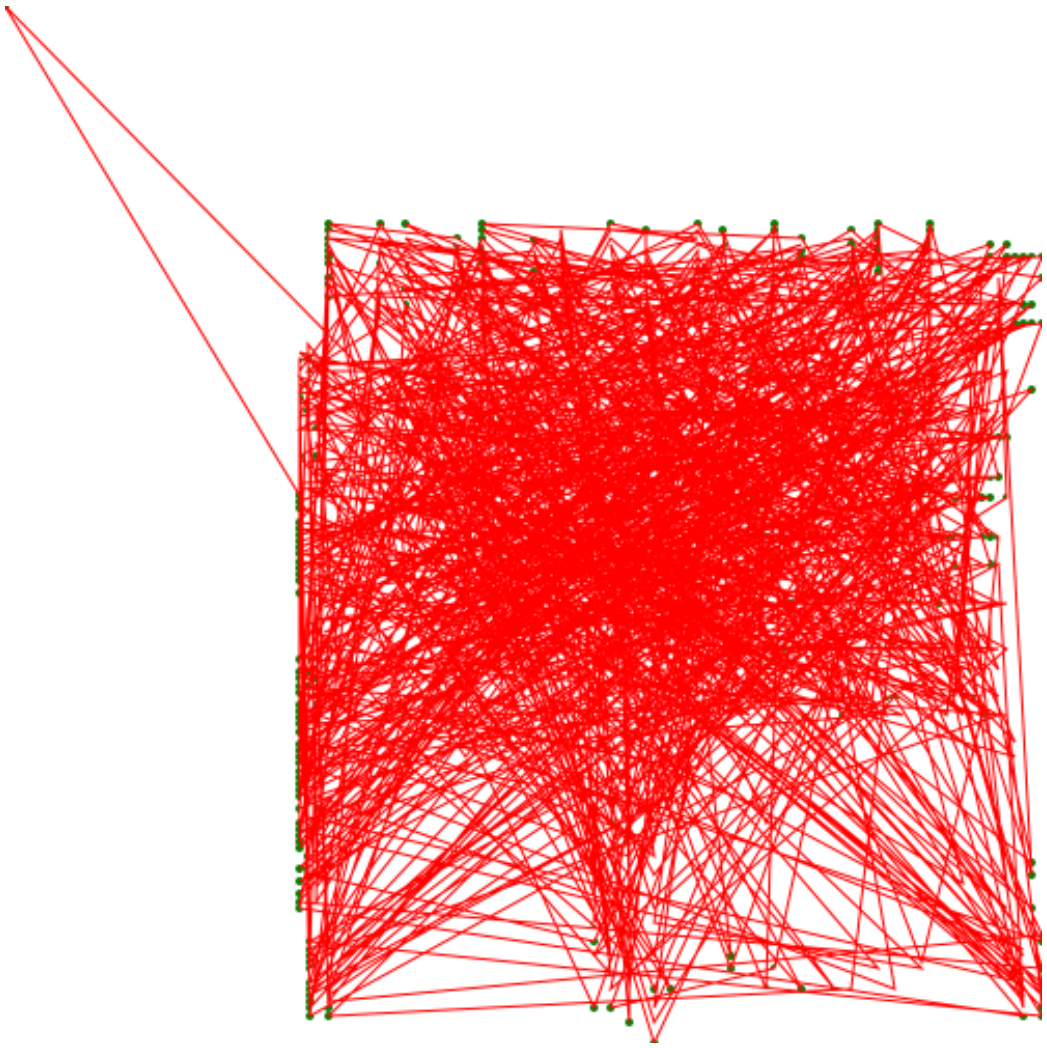


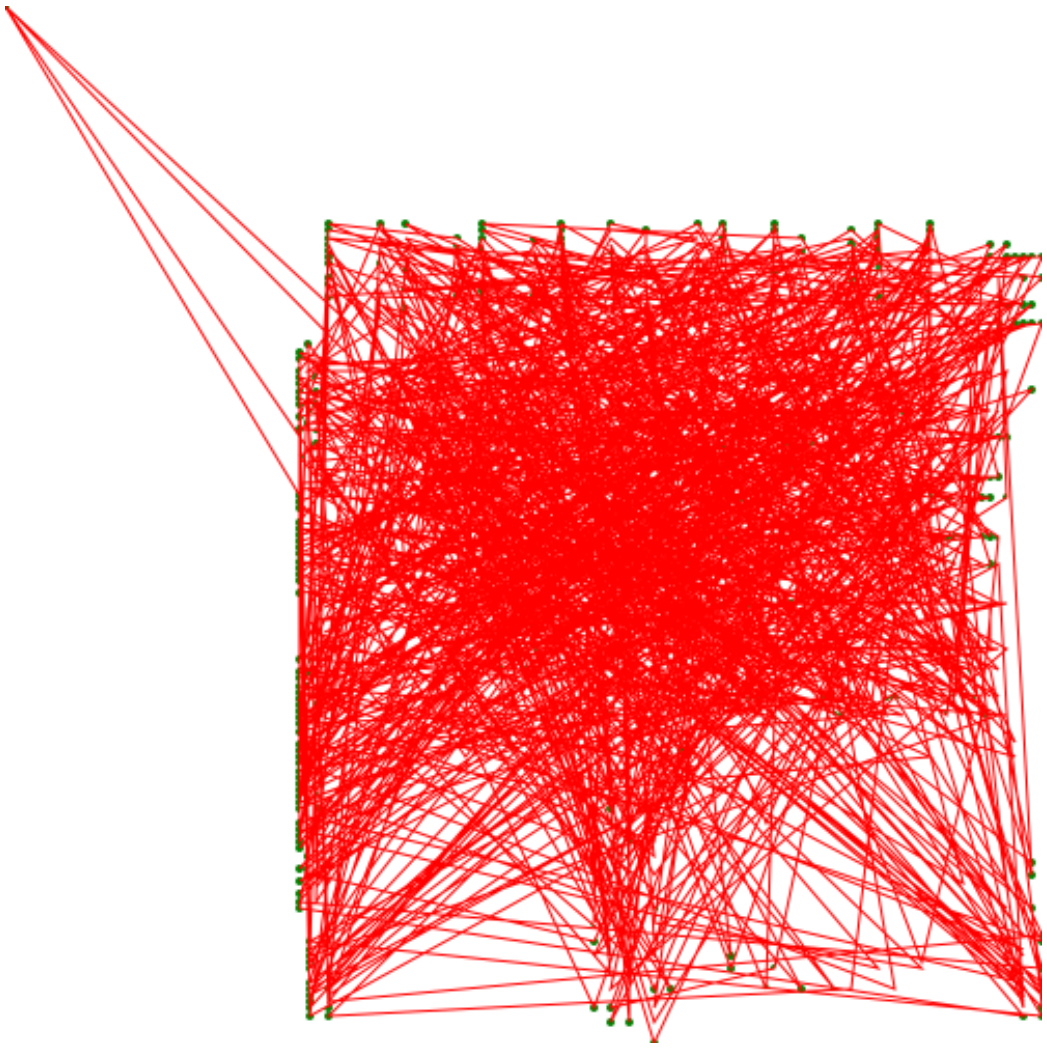


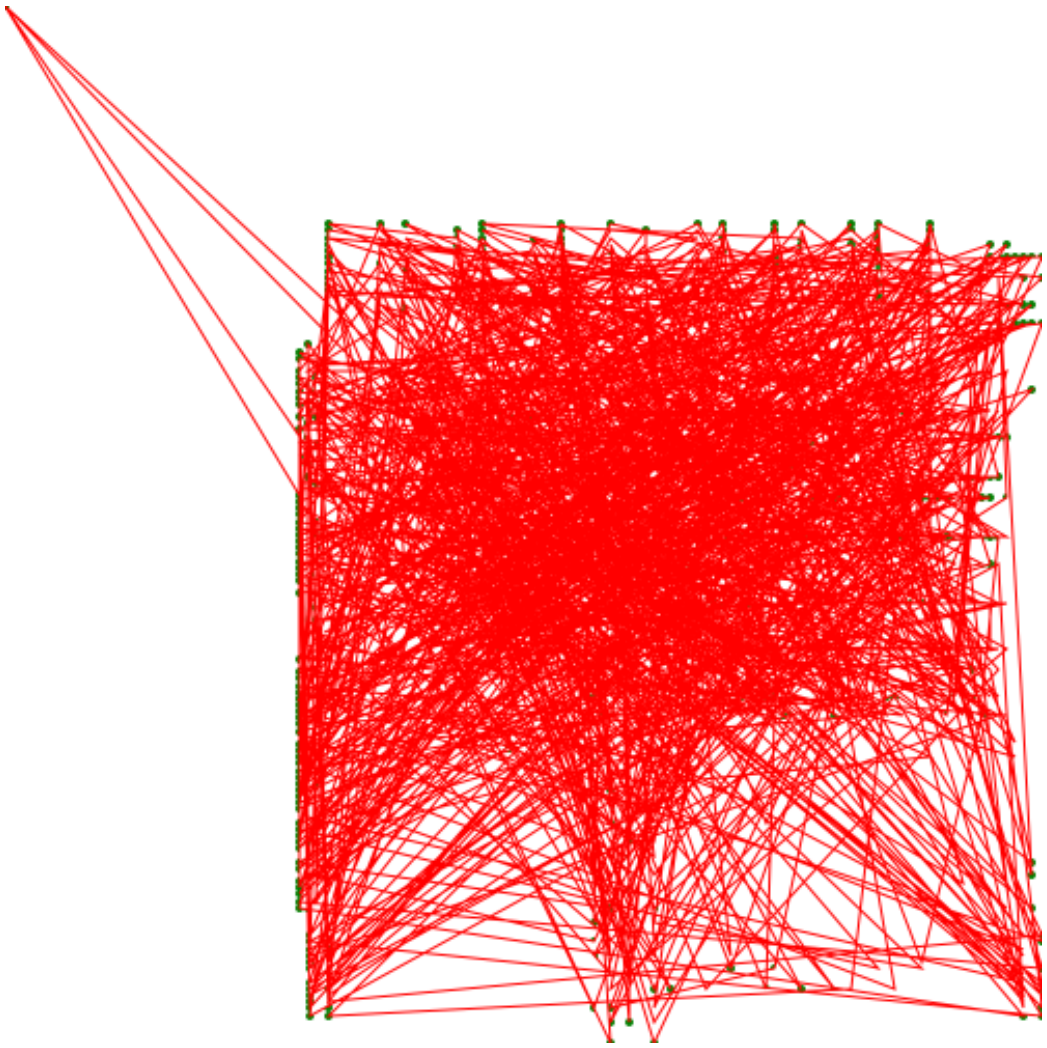


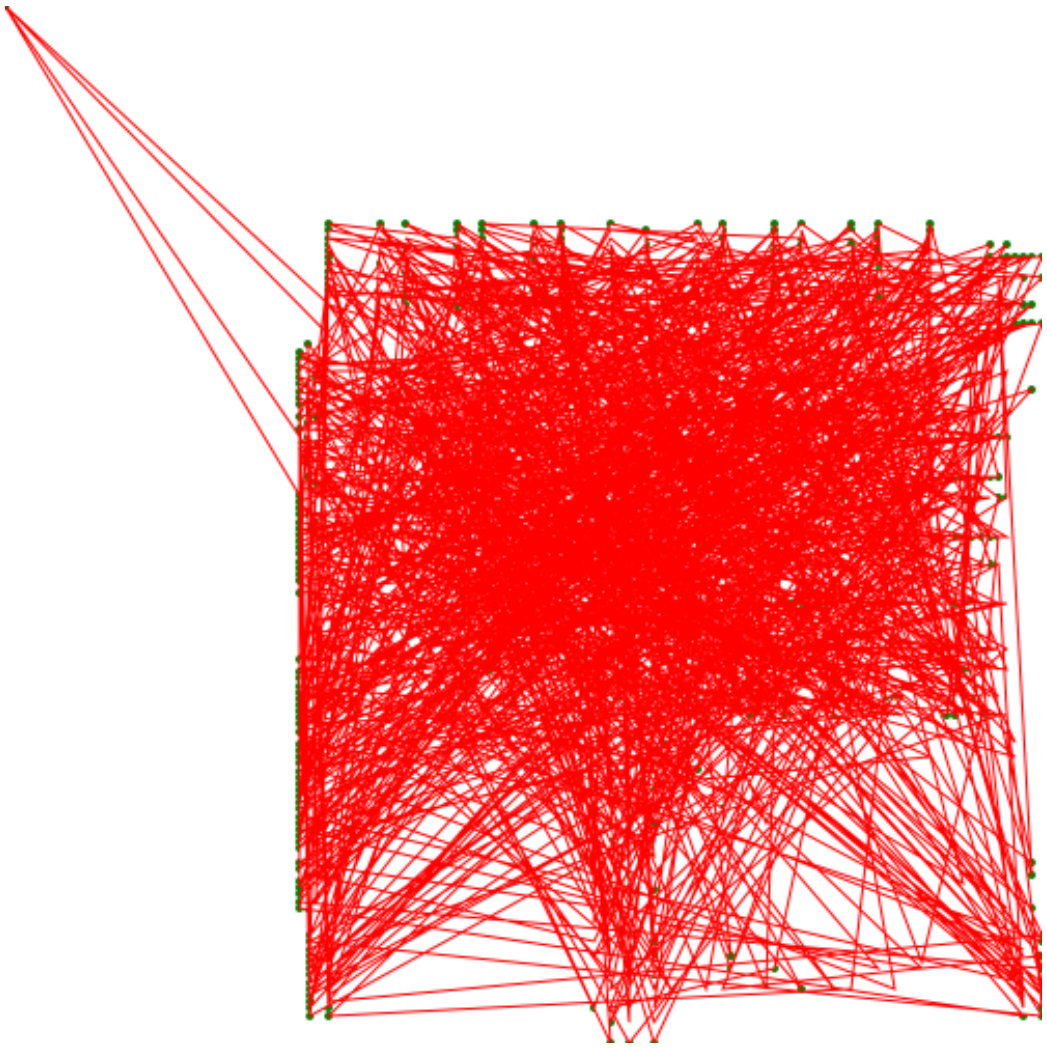


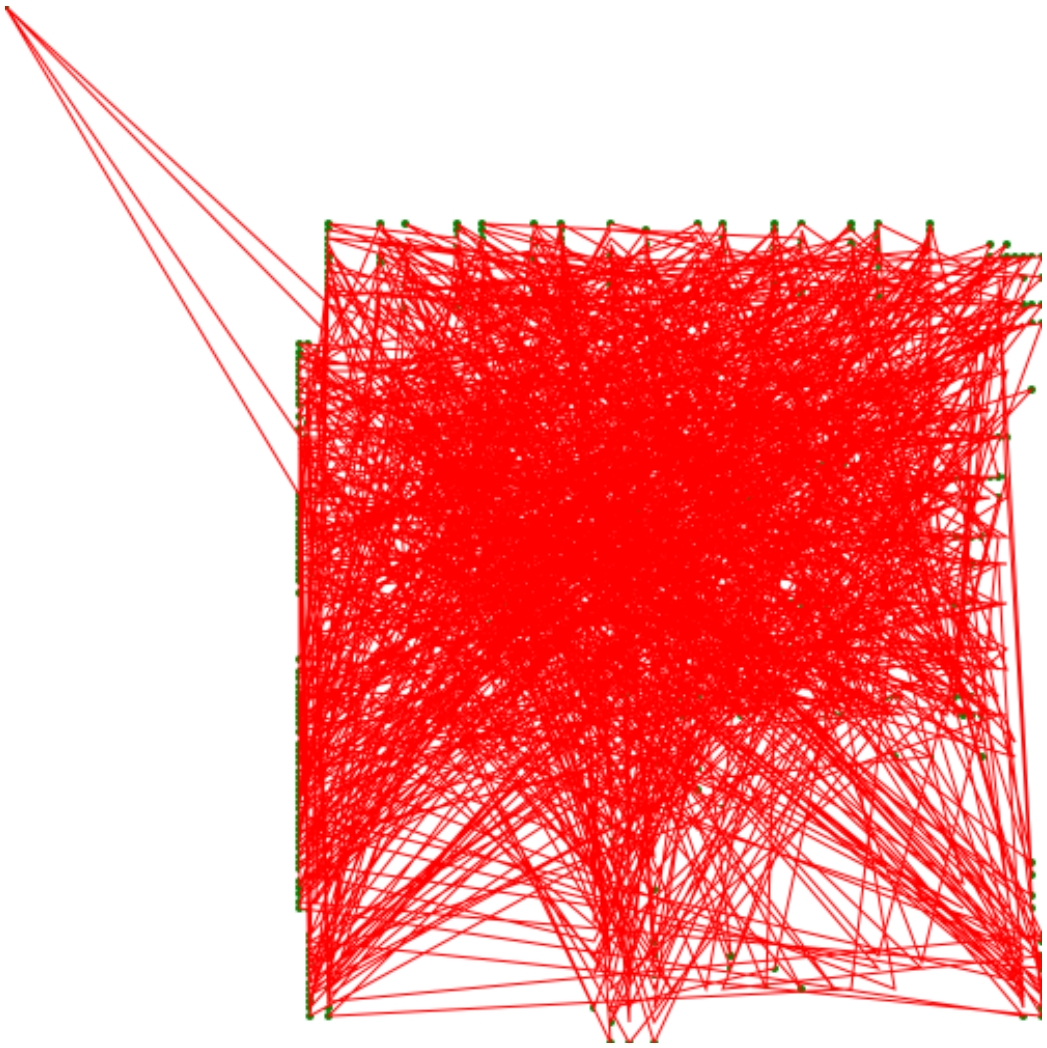












5 My Programs

5.1 SVG main.cpp

```
/*
 * main.cpp
 * CS202
 * April 22, 2020
 * Bryan Beus
 * Shaktoolik station for Iditarod Challenge
 */

#include <iomanip>
#include <vector>
#include <string>
#include <iostream>
#include <fstream>
#include <filesystem>
#include <stdlib.h>
#include <memory>
// #define BOOST_NO_CXX11_SCOPED_ENUMS
// #include <boost/thread.hpp>
// #undef BOOST_NO_CXX11_SCOPED_ENUMS

#include "Takotna.hpp"
#include "CityNode.hpp"
#include "CityPath.hpp"
#include "CityList.hpp"
#include "Miscellaneous.hpp"
#include "TspSolver.hpp"

using std::cin;
using std::cout;
using std::to_string;
using std::endl;
using std::vector;
using std::string;
using std::ofstream;
using std::ifstream;
using std::istringstream;
using std::pair;
using std::make_pair;
using std::setw;
using std::right;
using std::left;

namespace fs = std::filesystem;
// using namespace boost;
// using namespace boost::this_thread;

```

```cpp
int main() {

    clearConsole();

    vector<string> fileNames;
    callFileNames(fileNames);
    vector<CityList> citylists;

    // Parse File
    // for (size_t i = 0; i < fileNames.size(); i++) {
    for (size_t i = 0; i < 1; i++) {
        cout << "Parsing file: " << fileNames.at(i) << endl;
        string file = "./big/" + fileNames.at(i);
        ifstream fin(file);
        if (!fin) {
            cout << "Error loading file: " << file << endl;
            exit(0);
        }

        CityList newList;
        newList.parseFile(fin);
        citylists.push_back(newList);
    }

    // SolveRandomly()
    double bestDistanceRandom = 1000000000000;
    string filepath = "./output_images/randomly";
    fs::create_directory(filepath);
    for (size_t i = 0; i < citylists.size(); i++) {
    // for (size_t i = 0; i < 1; i++) {
        CityPath citypath;
        TspSolver tsp;
       double randomDistance = tsp.SolveRandomly(citylists.at(i),
        ↪  citypath, to_string(i));
        if (bestDistanceRandom > randomDistance)
        ↪  bestDistanceRandom = randomDistance;
    }
    cout << "Best Distance for SolveRandomly: " <<
    ↪  bestDistanceRandom << endl;

    // SolveMyWay()
    double bestDistanceMyWay = 1000000000000;
    filepath = "./output_images/my_way";
    fs::create_directory(filepath);
    for (size_t i = 0; i < citylists.size(); i++) {
    // for (size_t i = 0; i < 1; i++) {
        CityPath citypath;
        TspSolver tsp;
        double MyWayDistance = tsp.SolveMyWay(citylists.at(i),
        ↪  citypath, to_string(i));
        if (bestDistanceMyWay > MyWayDistance) bestDistanceMyWay
        ↪  = MyWayDistance;
```

```
94      }
95      cout << "Best Distance for SolveMyWay: " << bestDistanceMyWay
        ↪  << endl;
96
97      // SolveGreedy()
98      double bestDistanceGreedy = 1000000000000;
99      filepath = "./output_images/greedy";
100     fs::create_directory(filepath);
101     for (size_t i = 0; i < citylists.size(); i++) {
102     // for (size_t i = 0; i < 1; i++) {
103         CityPath citypath;
104         TspSolver tsp;
105         double greedyDistance = tsp.SolveGreedy(citylists.at(i),
            ↪  citypath, to_string(i));
106         if (bestDistanceGreedy > greedyDistance)
            ↪  bestDistanceGreedy = greedyDistance;
107     }
108     cout << "Best Distance for SolveGreedy: " <<
        ↪  bestDistanceGreedy << endl;
109
110     return 0;
111 }
```