CS 202 Iditarod - Galena

Bryan Beus

April 14, 2020

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

1 Design

1.1 TSPLIB - Part 2

For the first two solutions, I plan to keep things simple and follow the instructions.

For the final solution, my way, I am going to try to increase the number of random attempts my algorithm makes, just a few times, to add a slightly reduced time.

2 Post Mortem

2.1 TSPLIB

The only issue I really had here was in the fact that I do not have enough time before the due date to calculate the greedy method to conclusion. In fact, I think that might take several weeks.

I assume there are efficiencies I could add to speed things up, but the ”greed” in this algorithm is still far off the available clock. This was a fun learning experience.

3 Commit History

3.1 Galena

2020-04-14 Iditarod: initiate Galena
2020-04-14 Create CityPath class
2020-04-14 Galena: Develop SolveRandomly; Takotna: Add comments
2020-04-14 Create TspSolver SolveRandomly function
2020-04-14 Comment source files
2020-04-14 Galena: Create SolveGreedy function; create SolveMyWay() function

4 Sample Output

4.1 TSPLIB Part 1

```
Calculating distance between nodes: 2974 and 5729
Calculating distance between nodes: 5729 and 8579
Calculating distance between nodes: 8579 and 13184
Calculating distance between nodes: 13184 and 17546
Calculating distance between nodes: 17546 and 18002
Calculating distance between nodes: 18002 and 8457
Calculating distance between nodes: 8457 and 13575
Calculating distance between nodes: 13575 and 17346
```

5 My Programs

5.1 TSPLIB Part 1 - main.cpp

```
/*
 * main.cpp
 * CS202
 * April 14, 2020
 * Bryan Beus
 * Galena station for Iditarod Challenge
 */

#include <iomanip>
#include <vector>
#include <string>
#include <iostream>
#include <fstream>
#include <filesystem>
#include <stdlib.h>
#include <memory>

#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::endl;
```

```
28 using std::vector;
29 using std::string;
30 using std::ofstream;
31 using std::ifstream;
32 using std::istringstream;
33 using std::pair;
34 using std::make_pair;
35 using std::setw;
36 using std::right;
37 using std::left;
38
39 namespace fs = std::filesystem;
40
41 int main() {
42
43     clearConsole();
44
45     vector<string> fileNames;
46     callFileNames(fileNames);
47     vector<CityList> citylist;
48
49     // Parse File
50     for (size_t i = 0; i < fileNames.size(); i++) {
51         cout << "Parsing file: " << fileNames.at(i) << endl;
52         string file = "./big/" + fileNames.at(i);
53         ifstream fin(file);
54         if (!fin) {
55             cout << "Error loading file: " << file << endl;
56             exit(0);
57         }
58
59         CityList newList;
60         newList.parseFile(fin);
61         citylist.push_back(newList);
62     }
63
64     // SolveRandomly()
65     double bestDistanceRandom = 1000000000000;
66     for (size_t i = 0; i < citylist.size(); i++) {
67         CityPath citypath;
68         TspSolver tsp;
69         double randomDistance = tsp.SolveRandomly(citylist.at(i),
           ↪  citypath);
70         if (bestDistanceRandom > randomDistance)
           ↪  bestDistanceRandom = randomDistance;
71     }
72     cout << "Best Distance for SolveRandomly: " <<
       ↪  bestDistanceRandom << endl;
73
74     // SolveMyWay()
75     double bestDistanceMyWay = 1000000000000;
76     for (size_t i = 0; i < citylist.size(); i++) {
```

```
77          CityPath citypath;
78          TspSolver tsp;
79          double MyWayDistance = tsp.SolveMyWay(citylist.at(i),
            ↪  citypath);
80          if (bestDistanceMyWay > MyWayDistance) bestDistanceMyWay
            ↪  = MyWayDistance;
81      }
82      cout << "Best Distance for SolveMyWay: " << bestDistanceMyWay
        ↪  << endl;
83
84      // SolveGreedy()
85      // (This is too long to actually complete!
86      double bestDistanceGreedy = 1000000000000;
87      for (size_t i = 0; i < citylist.size(); i++) {
88          CityPath citypath;
89          TspSolver tsp;
90          double greedyDistance = tsp.SolveGreedy(citylist.at(i),
            ↪  citypath);
91          if (bestDistanceGreedy > greedyDistance)
            ↪  bestDistanceGreedy = greedyDistance;
92      }
93      cout << "Best Distance for SolveGreedy: " <<
        ↪  bestDistanceGreedy << endl;
94
95      return 0;
96  }
```