Graph Report.

Adrian Jonsson Sjödin

Fall 2022

1 Task

In this task we will read a file in CSV format that contains cities, there neighboring city and the time it takes to travel between them. We will then implement different graph methods to find the shortest path between cities.

- Take the CSV file and turn it into a graph (map) that we will later use to find the shortest path between cities. For this you will need two other classes `City` and `Connection`. Also create a quick lookup method that will be used to add cities to the map and when traversing the graph.
- Implement a simple program that finds the shortest path between two cities, regardless if loops and double back paths are present. Do some benchmarks and present the minimum path found, and how long it took to find the path. What are the limitations of this implementation?
- Implement another program that finds the shortest path but that can avoid loops by keeping track of which cities we have already passed. Rerun the benchmarks from the previous program and see if there's any improvements.
-

2 Method & Theory

3 Result

Table 1: Linear search and binary search for when the zip code is in `String` format

linear 111 15	Stockholm	92 ns
linear 984 99	Pajala	32224 ns
binary 111 15	Stockholm	301 ns
binary 984 99	Pajala	141 ns

Table 2: Linear search and binary search for when the zip code is in `Integer` format

linear 111 15	Stockholm	53 ns
linear 984 99	Pajala	11318 ns
binary 111 15	Stockholm	126 ns
binary 984 99	Pajala	136 ns

Table 3: Lookup for when the zip-codes are used as indexes

lookup 111 15	Stockholm	51 ns
lookup 984 99	Pajala	53 ns

Table 4: Lookup for when using Hash Buckets

lookup 111 15	Stockholm	158 ns
lookup 984 99	Pajala	69 ns

Table 5: Number of collisions when using Hash Buckets

Modulo	Unique	2 on same	3 on same	4 on same	Utilization
31327	8961	688	25	0	29%
28627	8878	770	26	0	31%
27773	8820	841	13	0	32%

Table 6: Lookup for "Slightly better"

lookup 111 15	Stockholm	207 ns
lookup 984 99	Pajala	63 ns

Table 7: Steps in "Slightly better"

Max Steps	59
Average steps when stepping	8.9
Average numb. of times we need to step	1.1

4 Discussion

All the code can be found here: GitHub