CS315002 – Homework 7

Tristan Basil

**Analysis of provided solution:**

The provided solution implements the Floyd-Marshall algorithm, although the weights are all set to 1, as the spec for the problem does not indicate any particular weight between each node. It then tests various pairs of nodes, although after employing the algorithm these are simply as fast as it takes to access the locations within the modified graph.

**Plan to test running time:**

In order to test the running time, I modified the program to accept various sizes of graphs, and then removed the functionality of testing data points, as they were not part of the actual algorithm. I then created a program that generated test inputs for these sizes by picking a random number of higher numbered neighbors for each country, and then assigning those neighbors randomly. This did not guarantee that all countries were connected and included repeats, but this did not seriously hinder the progression of the algorithm and the likelihood of ending with a separated graph was extremely low.

Using the modified original program and my own input generator, I wrote a script to automate the process of testing different inputs and concatenated the results into 1 file for analysis via excel.

**Theoretical results for the number of min operations:** The theoretical number of operations for each size m input is m^3.

**Experimental Results:**

|  |  |  |
| --- | --- | --- |
| m | Min Operations | Running Time (s) |
| 10 | 1000 | 3.30E-05 |
| 20 | 8000 | 0.000125 |
| 30 | 27000 | 0.000422 |
| 40 | 64000 | 0.000917 |
| 50 | 125000 | 0.001887 |
| 60 | 216000 | 0.00286 |
| 70 | 343000 | 0.004549 |
| 80 | 512000 | 0.006381 |
| 90 | 729000 | 0.008801 |
| 100 | 1000000 | 0.011252 |
| 110 | 1331000 | 0.0137 |
| 120 | 1728000 | 0.016854 |
| 130 | 2197000 | 0.01969 |
| 140 | 2744000 | 0.024136 |
| 150 | 3375000 | 0.029945 |
| 160 | 4096000 | 0.035673 |
| 170 | 4913000 | 0.043217 |
| 180 | 5832000 | 0.050908 |
| 190 | 6859000 | 0.061063 |
| 200 | 8000000 | 0.070935 |
| 210 | 9261000 | 0.081953 |
| 220 | 10648000 | 0.09513 |
| 230 | 12167000 | 0.107645 |
| 240 | 13824000 | 0.122905 |
| 250 | 15625000 | 0.13922 |

**Discussion of results:**

The instruction count as a function of m was exactly as expected; the amount of instructions was equal to m^3 for each test case.

The time as a function of m was as expected as well, although slightly faster at higher values of m. I think this can be attributed to the overhead generated in looping through the various numbers: the actual min comparison takes up proportionally much more time than any other overhead in the algorithm, so the appearance that the instruction is speeding up as the input size increases is actually due to the actual time per instruction being less and less affected by overhead.

The ratio of time to count as a function of m was expected to be constant and was far from it, but can still be explained. The smaller m values yielded larger time to count ratios, while as m increased in size, the ratio decreased, until it remained about constant after m > 130. The larger m values generating a more constant ratio support my thoughts about the last paragraph: as m increased, the overhead calculation times proportionally decreased, resulting in a more exact representation of time per instruction with a larger graph size.

The ratio of time to theoretical running time displays a nearly identical trend to the last graph. It should be noted that the theoretical running time was calculated by using the provided time/instruction value from the gp.script file. Again, as m increased, the running time decreased, until it remained relatively constant after >130. This indicates the overhead becoming less significant with larger input size, and overall, supports the idea of the theoretically constant graph.

In conclusion, my experimental results tend to agree with the theoretical running times for my programs. While the smaller m values were not always representative of the theoretical trend, the larger sample sizes proved that with a sufficient input size, the theoretical running times hold true for the Floyd-Marshall algorithm.