COMP26120 Lab 10 - Small World Hypothesis Test Report

Tejas Chandrasekar

March 6, 2019

1 The small-world hypothesis

The small world hypothesis states that in a social network graph, any pair of nodes is connected via a short path, which must not exceed 6 edges in length. This special property is known as the Six Degrees of Separation, which states that of all socially connect people, any two are six or fewer social connections away.
This hypothesis will be tested by solving the all-pairs shortest path problem using Dijkstra's algorithm, which will involve finding the shortest paths between all possible pairs of nodes in the graph. Note that these paths must have finite lengths. For graphs which are too big to implement Dijkstra's or Floyd's algorithms, heuristic path following will be used, where a certain heuristic, i.e. an interesting constraint among nodes, can be used to find short paths similar to that of Dijkstra's algorithm.

2 Complexity Arguments

For an all-pairs solution, Dijkstra's algorithm complexity is $O(n*(n+e)log(n))$ where n is number of nodes and e is number of edges, if the algorithm uses an optimised heap structure to implement its priority queue. This is because for a single source solution, the complexity would be $O((n+e)log(n))$, due to $O(e*log(n))$ runtime for all priority value updates and $O(n*log(n))$ runtime for all n removals of the minimum element from the heap, starting from a single node. Since this implementation uses a simple queue with adjacency lists, the asymptotic complexity would most likely be $O((n^2)*log(n))$.
Floyd's algorithm complexity is $O(n^3)$ but it is more suitable for an adjacency matrix than a group of adjacency lists, which is what these graphs have. Hence in this case, Dijkstra's algorithm is more suitable because there is less risk of wasting memory, and (when optimised fully) it is theoretically more computationally efficient than Floyd's algorithm. However, the mathematical analysis of Floyd's algorithm appearing slower fails to consider the practical factors, e.g. processor pipelining and caching of instructions and data. Since processors are skilled in optimising nested for loops and therefore have the algorithm making better use of temporal and spatial locality than Dijkstra's algorithm, there are in fact strong chances that in practice, Floyd's algorithm may run faster, despite its seemingly slower algorithm complexity.

3 Part 2 results

During the implementation of Dijkstra's algorithm, it was not necessary to use a priority queue for unweighted graphs such as Caltech.gx and Oklahoma.gx. A priority queue stores as keys all the distances between some source node and its adjacent nodes, with the keys' values being the actual nodes. When a node is yet be to assigned to the 'cloud' of visited nodes, it is implied that the distance from the source to its target adjacent node is 1, in an unweighted

graph. Hence, after the first iteration of the outer while loop, a priority queue would subsequently store all its keys as 1. Dequeuing the minimum value (i.e. 1 by default) before iteratively relaxing edges would not be effective, therefore, since there is no inherent 'priority' amongst the distance values.
The algorithm I implemented was 'greedy' due to the use of edge relaxation. This involved assigning every non-source node distance value to an infinite quantity, giving the source node a zeroed value, and updating the distance from source to target with a new, more accurate, finite distance, for every time Dijkstra's algorithm was run.
The average execution times (given in ms), longest path (in edges) and average distances for running Dijkstra's algorithm on the following files are given below (note that DNF = Did Not Finish):

1. Caltech.gx (770 nodes, 762 edges)
2. email.gx (1134 nodes, 1133 edges)
3. small.gx (11 nodes, 5 edges)
4. Oklahoma.gx (17425 nodes)

Table 1: Runtime data

Filename	Average Exec. Time	Longest Path (edges)	Average Distance
Caltech.gx	1.523 sec	6	2.32
email.gx	3.71 sec	8	3.60
small.gx	0.002 sec	3	0.91
Oklahoma.gx	83 mins	DNF	DNF

The above results have demonstrated that while one graph had taken too long to give conclusive results, there was one violation of the Six Degrees of Separation Rule with email.gx. This was most likely due to its larger size than Caltech and small.gx, and resultant higher average shortest distance between two nodes. Alternative measures included connectedness of a graph, which is shown in the following table:

Table 2: Connectedness data

Filename	Connectedness (percentage)
Caltech.gx	97.8
email.gx	99.7
small.gx	29.8
Oklahoma.gx	DNF

4 Part 3 results

In this part of the experiment, I chose the heuristic of finding the shortest path by traversing different node's adjacency lists via the node of that list with the highest number of outgoing connections (i.e., its outdegree). The algorithm can be described in the following steps, to find the shortest path from a source node to a target node given a certain graph:

1. Assign a current node to the beginning (source) node
2. If the target node isn't in the current node's adjacency list, which should also have at least one item in it, the path is modified with the current node
3. Assign a node in the current node's adjacency list to that with the largest outdegree
4. Reassign current to this node
5. Repeat from step 2 until either current node's adjacency list is empty or target node is found in that same adjacency list
6. At this point, modify and complete the path with the current node

Inherently, this approach alone is not guaranteed to find the shortest path. This is partly because there is a risk of cycling between a small subset of nodes, if at any point, the adjacency list of a node A has its maximum outdegree point to another node B, but node B's adjacency list has its maximum outdegree point back to node A. Therefore, as the distance of the path keeps getting updated, an infinite distance will be produced as a result. Hence, it is usually used in conjunction with Dijkstra's algorithm as part of a more computationally efficient algorithm, e.g. the A* search algorithm. However, the graph data for this experiment is all unweighted so this being an informed search algorithm need not apply here.

5 Conclusions

From part 2, it is clear that on average, the email.gx graph breaks the small world hypothesis. Hence the small world hypothesis is not true for all graphs that were tested with Dijkstra's algorithm.
From part 3, it is clear by comparison to Dijkstra's algorithm, the average shortest distance was not consistent with my results from part2, and therefore heuristics alone are an unreliable method of calculating the average shortest path between two nodes in a graph.