COMP26120 Lab 10: Testing the small world hypothesis

Haorui Chen

March 5, 2019

1 The small-world hypothesis

Statement: If we represent each person as node and acquaintance of two as edge with length of 1, then for any two nodes there will be a path in between with its length no longer than 6. This shold apply to the majority part of this graph, yet there wlll be a few exceptions.
Test: I am going to test the hypothesis on two given databases, Caltech.gx and Oaklahoma.gx, using 2 methods, Dijkstra's algorithm and Heuristic method which pseudo code has been given in the lab manual. For every node in graph, set it as starting point and traverse the whole graph, see if there are often nodes that has to be reached by 7 steps or more from starting point, or even unreachable.

2 Complexity Arguments

Time complexity of Dijkstra's algorithm is O(n*(n+e)*log(n)); that of Floyd's algorithm is O(N**3). For Oklahoma.gx, there are 17425 nodes and 1785057 edges, 442,489M time units is expected for Dijkstra's algorithm, 5,290,763M time units is expected for Floyd's algorithm. For Caltech.gx, there are 769 nodes and 33313 edges, 251,262K time units is expected for Dijkstra's algorithm, 454,756K time units is expected for Floyd's algorithm. We can see that for both two graphs Dijkstra's algorithm is expected to be more effcient.

3 Part 1 results

For Dijkstra' algorithm, I use a priority queue instead of a queue to store select the new minimum distance and store newly discovered edges. Though we are applying Dijkstra's algorithm on unweighted graphs, a priority queue implemented by heap guarantees its application on weighted graphs in which case a simple queue is unusable, heap implmentation with heap scales down time complexity of the algorithm to O(n*(n+e)*log(n)).

For each starting node, I listed out nodes that have a distance larger than 6 with it and those unreachable and counted number of them respectively, also I gave out average distance from each starting node to give a direct idea of overall circumstance of length of nodes that is reachable from current one, if the figure is low, say 2 or 3, we believe that at least most of nodes is reachable in 6 steps, thus hypothesi is met. To further prove this hypothesis, I also give out the longest shortest path, if the figure is shorter than 6 than we can definitely say that all reachable nodes can be reachable within 6 steps. I also gave global average distance to give a more general sense of the graph.
For both Caltech and Oklahoma The whole graph is very likely to meet the small-world hypotheis.
For Oklahoma, we can see only a few paths with its length larger than 6(given that they are mutually reachable), that's 3324, very few compared to 1785057 total edges. We can see that for most nodes there are 5 nodes that are unreachable and vise versa, by investigating the database I know that these nodes are not connected to the lagrest one but they themselves are mutual connected. Since paths with length bigger than 6 and number of nodes unreachable are both very small I will say that the hypothesis is met on Oklahoma.gx.
For Caltech, we can see no path has it length larger than 6(given that they are mutually reachable), also we discover that for most nodes there are 7 nodes that are unreachable and vise versa, by investigating the database I know that these nodes are divided into 3 connective groups and not connected to the lagrest one. Since paths with length bigger than 6 and number of nodes unreachable are both very small I will say that the hypothesis is met on Caltech.gx.
Running time on Caltech.gx of all-pairs Dijkstra's algorithm is about 0.341s, on Oklahoma.gx is 2310.931s, 6776 times higher than that of Caltech.gx. If we put relation of nodes and edges of Caltech.gx and Oklahoma.gx to compute time complexity relationship, we get log(17425)/log(769)*17425/769*(17425+1785057)/(769+33313)=1761.135, not that far from actual result, thus execution times agree with the complexity characteristics found in Part 1.

4 Part 2 complexity analysis

My implementaton of heuristic route finder is basically based on BFS. In the worst case(or say ideally), all edges and nodes will be visited exactly once, so time complexity of single-source heuristic algorithm is O(n+e), for all-pairs it's O(n*(n+e)).

5 Part 2 results

My heuristic is choosing the node among its source node with the largest out degree, because I guess that the larger the outdegree in nodes of path, the more likely and earlier I may get to the target, thus those with less out degree will not be searched from in my algorithm. Also, in this kind of greedy method, we

don't consider edge length equally important as out degree, so I don't need to pick out the edge with shortest path anymore, and a simple queue is applicable. For the above heuristic search, running time is much faster than Dijkstra's algorithm, that's 0.057s for Caltech.gx and 208.20s for Oklahoma.gx, but its results get much less accurate, as for both graph we got much more distances that are larger than 6, 147552026 in Oklahoma and 3324 in Caltech respectively, and unreachable that don't appear under Dijkstra's searching, also, average distance, nomatter for each node as starting point or the global one, grows considerably and exceed limit of 6.

However, we cannot sayb that we can get a conclusion on whether the ablove hypothesis is met. Though It seem that the pypotheis is not met if we look at these results individually, yet Dijkstra algorithm we used previously has told us it indeed has been met. Takeing in to account that there're actually a lot of nodes that we have not searched in heuristic method even though they are actually reachable from starting point, we cannot simply tell whether it is met from this result, or I will say it is met, it's just we are getting incomplete result by heuristic. Thus there's a trade-off between accuracy and effciency: If running time is not a issue and enough space is given, then Dijkstra's algorithm is the best to get most accurate result; if effciency matters most and fallacies can be tolerated, then heuristic serach(probably a refined version, i.e. do search based on several nodes with larger out degree instead of just with the largest out degree)shall be a good choice.