Vincent Salinas
6-6-2017
CSE 100
Project Assignment 4
Report.pdf

Graph Design Analysis

The graph structure uses the provided template code (ActorGraph.cpp ActorGraph.h) as well as 2 new classes which include 4 files: ActorNode.h, ActorNode.cpp, Link.h, Link.cpp. These 6 program files together construct an actor graph from the actor nodes from the .tsv file, and utilizes the Link class to determine the pathway from the start actor (source) to the end actor (destination). Data structures utilized in this program included BFS, Dijkstra's algorithm, and union find. For the checkpiont, Dijkstra's algorithm was not needed here since the edge weights were all designated as "1", so BFS was used instead to determine the shortest path from the source to the destination. The advantage of using BFS was the run-time, which is O(E + V), which is faster than Dijkstra's algorithm, which is O((V+E*log(V)), and also under the condition that the edges (or connections) were unweighted. The goal here was run-time optimization. For the final submission, Dijkstra's algorithm was then used, due to the fact that the edges were now weighted.

Actor connections running time

1.
Union-Find is better. Run-times below:

BFS = **33375933947** ns
union-find = **17006055957** ns
Difference in favor of union-find: **16369877990** ns

2.
Between union-find and breadth first search algorithms, looking at their efficiencies suggests that breadth first search is better with its constant-time operation, versus union-find's logarithmic efficiency. But in practice, union-find is better, evident from the run-time analysis of the *actorconnections* program. The union-find data structure significantly outperforms BFS when the "relationship" is changing, or in other words, when there are "union" operations that need to be performed on a set of partitions.

Under the conditions of a fixed, undirected graph, the relationships are not changing and the edges are fixed. While BFS may be asymptotically faster than union-find, the important factor here really is the problem that is trying to be solved.

3. One piece of evidence of the better performance of union-find over BFS is the duration (nanoseconds) of the *actorconnections* program on the same set of data.

Between union-find and breadth first search algorithms, looking at their efficiencies suggests that breadth first search is better with its constant-time operation, versus union-find's logarithmic efficiency. But in practice, union-find is better, evident from the run-time analysis of the *actorconnections* program. Looking at their run-times: (1) union-find = **17006055957** ns, (2) BFS = **33375933947** ns on a certain set of data. The difference is fairly stark at **16369877990** ns. The union-find data structure significantly outperforms BFS when the "relationship" is changing, or in other words, when there are "union" operations that need to be performed on a set of partitions. Under the conditions of a fixed, undirected graph, the relationships are not changing and the edges are fixed. While BFS may be asymptotically faster than union-find, the important factor here really is the problem that is trying to be solved.