Dephanie Ho
Peter Phan
CSE 100 PA 4

Report

Graph Design Analysis

The graph structure contains the following classes: Movie, ActorNode, and ActorGraph. Movie contains the string year and name of the movie, and a hash set that contains a set of strings of the actors in the movie. ActorNode contains the string name of the actor, and a hash set that contains a set of strings of the year+name of the movie. ActorGraph contains a hash map movies, that takes in the string of a movie year+name and returns the corresponding Movie *, and a hash map actors, that that takes in the string of the actor's name and returns the corresponding ActorNode*. Overall actors are the nodes, movies are the edges, and the year of the movie is the weight of the edges. We set ActorGraph to hold the data of both movies and actors to optimize memory storage. ActorNode and Movie can just access the data they need from one source, instead of storing repetitive data in their respective classes. All we need to do is get the string corresponding with the Actor or Movie and we can quickly access it from the ActorGraph

Actor Connections Running Time

1. Which implementation is better and by how much?
   Disjoint set is faster and in our implementation case by average 50 seconds faster.

2. When does the union-find data structure significantly outperform BFS (if at all)?

It significantly outperforms BFS when we check multiple sets of data because we compress the path such that if the node shares the same root, we are done. While BFS has to find a path every time from start to finish, so there are more edges that BFS has to traverse than the disjoint set.

3. What arguments can you provide to support your observations? (run time analysis)

Let N be number of nodes and M be number of edges

Everything in finding out whether or not the actor nodes are connected is pretty much the same except when we add edges in the BFS and union in disjoint set as well as doing a bfs in BFS and find in disjoint set. Adding edges takes O(M) average case and doing a BFS has worst case O(N) for looking at one pair. On the other hand, disjoint set is O(log n) for union, find run time is O(N) for the first pair, but for the other pairs that are in the same path of the first pair, it will take

O(1) because of path compression. As we can see, find reduces the runtime significantly in a disjoint set whereas in BFS, it will stay relatively the same. So comparing P number of pairs, BFS will take O(P(M+N)) while Disjoint Set will run O(P(N)) or O(P).

O(P(N)) and O(P) are definitely faster than O(P(M+N)) meaning that Disjoint set are definitely faster.