Lucas Blumhardt
A12020745

Analysis of Timing:

BFS User times

- ran with pair.tsv: 3.275 seconds
- ran with 100 of the same actor pair: 2.471 seconds

UFIND User times

- ran with pair.tsv 1.085 seconds
- ran with 100 of the same actor pair: 0.734 seconds

Both buildBFS and buildUFIND push every Movie* onto the priority_queue and pops all of them off. Therefore there is no difference in runtime due to that because they are identical. BuildBFS doesn't build as many edges as buildGraph does, because all that's required in BuildBFS is to find A path, not THE SHORTEST path. Therefore it's more time efficient to just add the minimum amount of edges per movie (just make sure every actor has SOME path to each other). This makes buildBFS have a quicker build component than buildGraph had, but buildGraph needed all these edges in order to compute the SHORTEST path. buildUFIND doesn't even use edges. Whenever actors starred in a movie together their graph was merged. Merging was just setting the source (ActorNode pointer member var) of one ActorNode to the source of the other which is O(1).

It makes me think that edges are only advantageous when knowing the path is required, otherwise you don't even need them.

The find component of buildBFS calls Dijkstra (unweighted) on every actor pair. This has a runtime of O(V + ElogE). I reduced the amount of edges slightly to improve time, but it's still slower than buildUFIND's find component. The find method there (I structured it as a helper method called find(ActorNode* a) in my ActorGraph class) is O(log V). But I also optimized it with path compression, so the runtime should be closer to O(1) after the first find.

1) Which implementation is better and by how much?
UFIND is better than BFS because BFS keeps track of edges which takes unnecessary time when we aren't concerned with the specific path, but rather just that one could exist. UFIND is about 3 times as fast in the average case.

2) When does the union-find data structure significantly outperform BFS (if at all)?
UFIND was still around 3 times as fast when finding the same actor pair repeatedly. My implementation of UFIND doesn't outperform BFS significantly in this case.

3) What arguments can you provide to support your observations?
As I stated above, the lack of edges in UFIND and the presence of pathcompression make UFIND a much quicker algorithm