Jihoon You

A97097001

CSE 100

PA4 report

Graph Design Analysis

- Describe the implementation of your graph structure

Three files that newly created: pathfinder.cpp, ActorNode.h(ActorNode.cpp), ActorEdge.h

ActorNode class represents an actor. It has **string name** that represents name of the actor, **vector<ActorEdge*> edges** that represents movie list that the actor acted, **bool visit** that checks visit for BFS and dijkstra algorithm, **int dist** checks distance for BFS algorithm, string connection that holds output values for BFS function and dijkstra function**, int specialN** that represents special number for each nodes, **bool noexist** that checks if nodeActor exist or not, and **bool operator** for priority queue.

ActorEdge class represents a movie. It has **string title** that represents title of the movie, **int year** that represents the year of the movie filmed, **int weight** that is calculated 1 + (2015 – movie_year) for the weighted graph, and **vector<ActorNode*> cast** that contains list of casting of actors.

ActorGraph class. It has **unordered_map<string, ActorNode*> firstmap** that represents actors, **unordered_map<string, ActorEdge*> secondmap** that represents movies, **ActorNode* Dikjstra** function that checks connection between two actors on the weighted graph, **ActorNode* BFTraverse** function that checks connection between two actor on the unweighted graph, **void resetHelper** that resets values before BFTraverse and Dikjstra functions run, and given **loadFromFile** function.

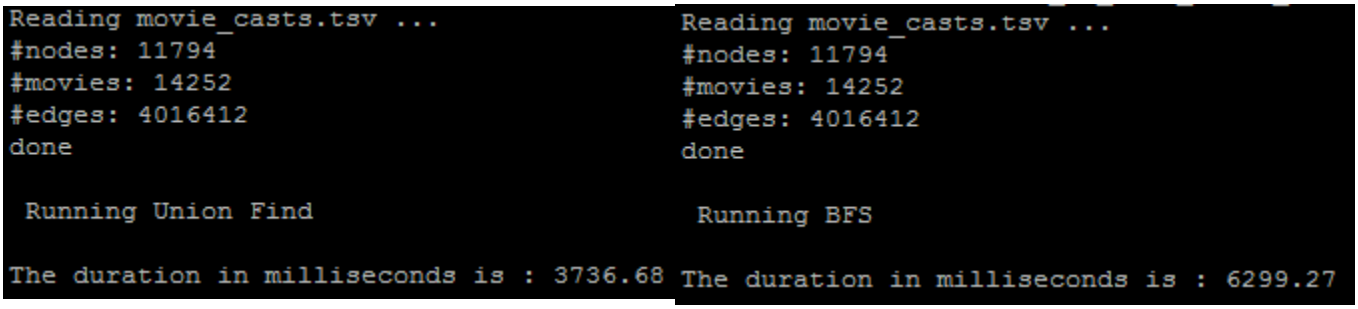
- Describe *why* you chose to implement your graph structure this way.

I choose to implement a graph with unordered_map because of the efficiency of lookup O(1). Also, it let us find wanted value with special keys(string movie_title + string movie_year). unordered_map takes an object of type key as argument and returns unique value of it.

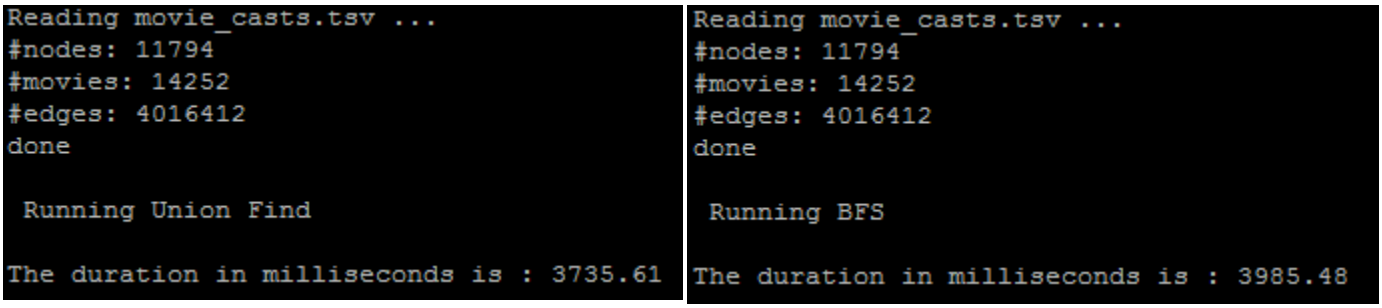
Actorconnections running time comparison

/**** My actorconnections function is not working so I'm writing this running time comparison by testing refactorconnections ****\

1. Which implementation is better and by how much?



Same actor pair appear 100 times: Union find is better than BFS by 6299.27 - 3736.68 = 2562.59



Different actor pairs 100 times: Union find is better than BFS by 3985.48 – 3735.61 = 249.87

2. When does the union-find data structure significantly outperform BFS?

The Union-find data structure significantly outperform BFS on same actor appear 100times.

3. What arguments can you provide to support your observations?

This significant change happens because of an implementation of path compression. The goal of find is to return sentinel node. On the path to the sentinel node, it make all nodes on the path points to the sentinel node. Therefore, when we perform union find the same actor pair 100 times, it outperforms BFS. However, for the different actor pairs 100, since each pair is different node, it does not affect running time much.