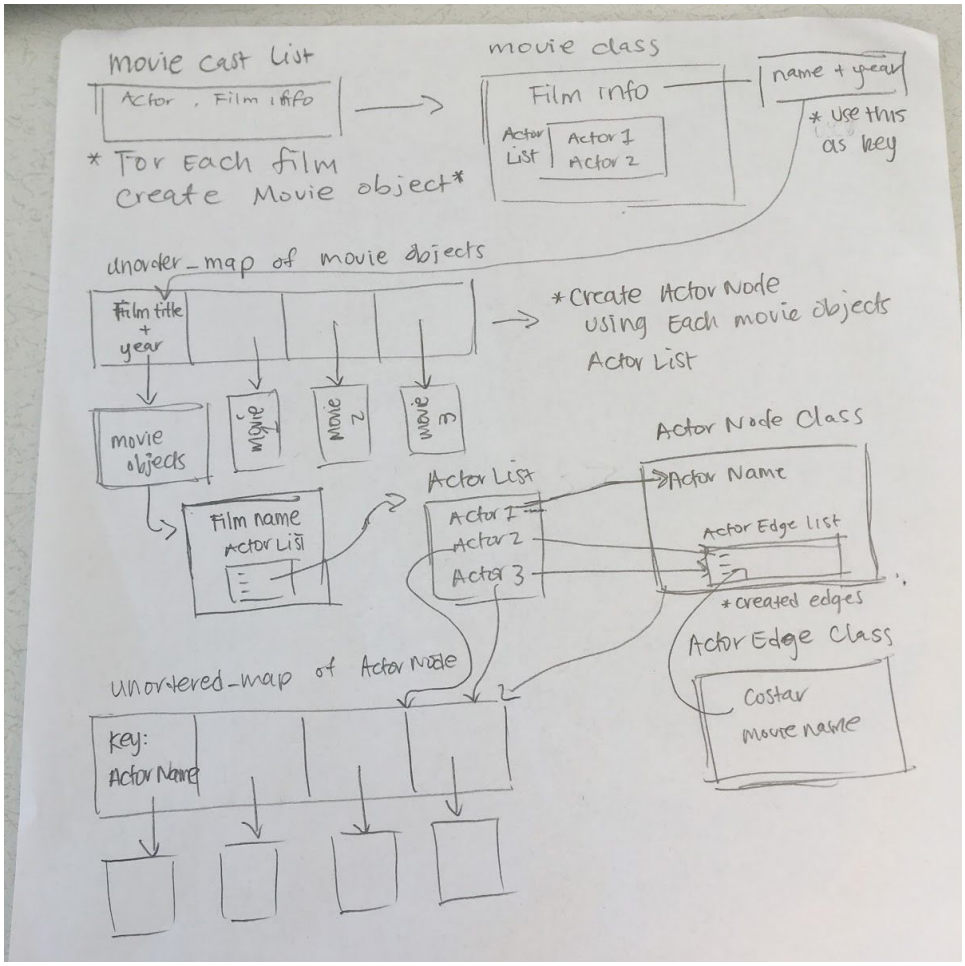
Graph Design Analysis

For my implementation of my graph structure I chose to make three classes: ActorNode class, ActorEdge class, and Movie class. The Movie class contains the name of film and a list of actors that have starred in the film. The ActorNode class contains the name of the actor and a set of ActorEdges. For storage of these object I chose unordered_map due to the fact that insert and find was rather speedy compared to a vector. I used the film name as the key in which to store each film object making it easier to find and retrieve. Likewise with the ActorNodes, I used the unique actor name as the key in which to store its corresponding ActorNode. Now when searching for a connection between the pair I can use the actors name and get the node corresponding to it in O(1) time.



Actorconnections running time comparison

Repeated Actor Pairs

- Breadth First Search: Runtime: **0m0.359s**
- Union Find: Runtime: **0m0.269s**

100 Unique Actor Pairs

- Breadth First Search: Runtime: **0m5.284s**
- Union Find: Runtime: Runtime: **0m0.346s**

Which implementation is better and by how much?

Without a doubt the union find implementation is faster than the BFS implementation. In my results the union find was about a little under 5 seconds faster than the BFS implementation. This time difference is significant and shows the increased input will not affect the runtime of union find by that much.

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

The union find data structure outperforms BFS when the number of unique test pairs increases. Both implementations were no different on small pair sets and repeated pair sets. Once the input grew, the time increased for BFS.

What arguments can you provide to support your observations?

The union find makes the lookup time for a single actor very fast (O(1)) since all actor nodes are connected directed to the sentinel node. As the number of searches increases, the sets become more directly connected to their sentinel nodes, thus making searching quite speedy. On the other hand BFS does not have this property, thus lookup requires traversing through the cluttered graph, giving it slower search times.