Project Proposal: 6Degrees

Casey Olsen (ceo2132), Jorge Raad (jar2356)

Introduction

Inspired by the popular game, “Six Degrees of Kevin Bacon,” our team plans to implement an algorithm that finds the minimum number of degrees (i.e. the shortest path) separating two actors, where actors who have appeared in the same movie are one degree apart from each other. The game claims that there is a maximum of six degrees of separation between any actor and Kevin Bacon, who has appeared in over 60 movies and co-starred with over 3,000 actors. We plan on implementing an algorithm that finds the minimum degrees of separation between any two actors, which would be a generalization of this game.

We are currently planning to use a dataset from IMDB consisting of movies and actors that appear in them, but we plan on making our code general so that we can substitute the dataset in the case that we run into issues. In this case, we could substitute the individuals (in this case actors) and the groups to which they belong (in this case movies), or we could be even more general and have no distinction between these two entities (e.g. for movies vs. actors, you can imagine the sets of movies and actors as bipartite subgraphs of the same graph). This would allow us to model “degrees of separations” through other relationships like friendships.

Parallelization

Our project has two opportunities for parallelism and performance optimization: graph construction and graph traversal. A significant amount of the work will go into reading our datasets and constructing our graph. There is likely a hard limit to how much reading in the data can be sped up due to the bottleneck we would likely face with I/O. So, we would most likely need to focus on how to parallelize the creation and traversal of our graph. Because this problem is only concerned with whether vertices are connected or not, we will have our edge weights equal 1, which simplifies this problem. If we focus on getting the fastest possible algorithm, we would simply use BFS since it is a linear-time algorithm. To begin parallelizing this, we could begin by executing in parallel both the popping of the vertices off of the queue in the while loop and the iteration through edges in the for loop.

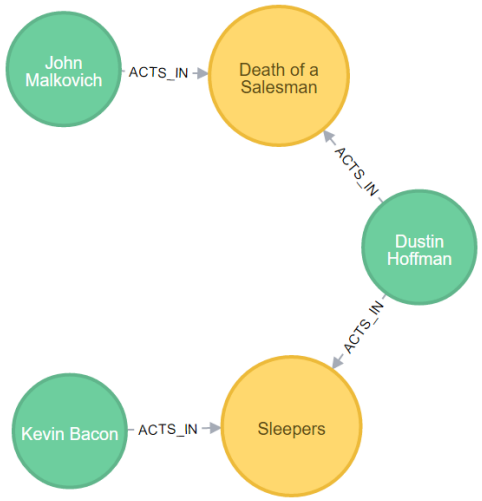
Sequential pseudo code:

```
bfs(V, E):
      for all v in V:
            dist[v] = infinity
      dist[s] = 0
      Q = emptyQueue()
```

```
Q.add(s)
while Q is not empty:
      u = Q.pop()
      for all edges (u, v) in E:
            if dist[v] = infinity:
                  Q.add(v)
                  dist[v] = dist[u] + 1
```

Example

For example, if we input the names “Kevin Bacon” and “John Malkovich” to our test program, we would find that they are a single degree of separation apart. We would find that they both acted in a movie with Dustin Hoffman. As this graph demonstrates, each path in our traversal will alternate between actors and films they have starred in.



Resources

We are planning on using free, public, IMDb datasets with over 12 million actors and the movies they are famous for. The dataset does include other movie industry individuals such as directors, cinematographers, etc, that we will filter out for our project. Additionally, the movies are currently stored using generated ids, rather than title, so we will need to do a join with an additional table to get the name of the movies. Our program will read in these tables to construct a graph with actor and movie nodes, that we will traverse with our shortest path algorithm.