Final Bechdel Test Project

Authors: Sophie Lin, Rachel Hu, and Lilymoon Whalen

Introduction:

In this project, we examined gender imbalance in films by examining a real data set of 50 Hollywood movies and running them through a quantitative test. The original Bechdel-Wallace test which checks if a movie has two named female characters, and if those two characters have at least one conversation that is not about a man. However, this original test does not uncover the deeper core inequalities imbedded in Hollywood films. Hickey et al., reached out to women in film and television and developed 11 new Bechdel Test's to test whether the film industry has improved in representation of women (2017).

Inspired by Hickey et al., we look at the same data set of 50 Hollywood movies and create our own Bechdel test (2017). We first create a visual representation of the movies and actors examined by using a graph. Our test is similar to the uphold test created by Hickey et al.. The uphold test proposed by Rory Uphold tests whether the on-stage crew is at least 50% women (2017). Rather than looking at on-stage crew, our test looks at the set of actors for a specific movie and checks whether at least 50% of the cast are female.

Method (1/2 page):

To create a visual representation of the Hollywood movies in relationship to the actors who played in those movies, we implemented Graph with the nodes representing the actors and movies. Additionally, we created an Actor and Movie class, which represents the actor and movie. Both the Actor and Movie objects extend from the FilmElement class. We decided to utilize polymorphism since the HollywoodGraph class would take both Actors and Movie objects, so using a parent class that encapsulates the two into one type makes it possible for our vertices Vector that can only hold one type hold both Actors and Movies. We also later added a Roles class, which contains the information of each role that the actor played in, including the gender, what type of role, etc. We decided to utilize the Roles class as it would be easier to access specific information correlated with a specific movie. We then, in Actor, used a Hashtable to store a movie's name and a role object for the role the actor played in that movie. This allows easy accessibility of the actor's role in the movie without having to iterate through the entire list of an actor's roles. For the method that calculates the number of movies that separate two actors, we saw two approaches—keeping track of every possible path from the first actor to the second using DFS, and then counting the path length of each to find the shortest, or using BFS to keep track of how deep the search is at. We went with the second option because the first option with DFS would create an extremely large data structure holding every single permutation of paths between the two actors, which would be space and time inefficient, especially when the provided data set is 2075 lines long.

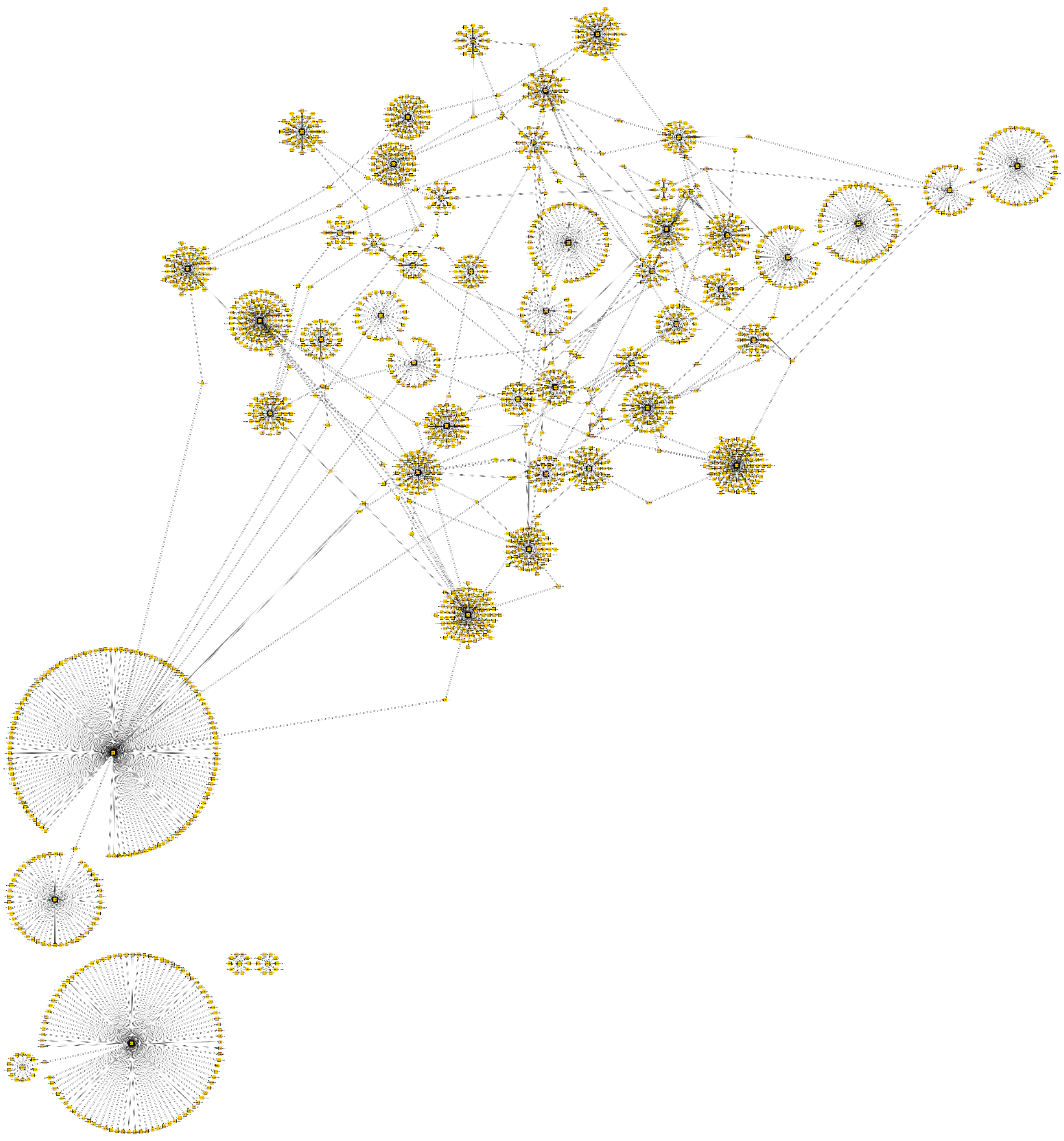

Figure 1: Graph generated with yED from the provided dataset of 50 movies and the actors that played in them, showing circles of actors around each movie, and connections between movies formed by actors that acted in multiple movies.

Conclusions (1 page):

Our own new Bechdel-like test was to calculate the percentage of female roles that appear on screen, while the Bechdel test incorporated every employee involved in the making of the film. Calculating the percentage of female roles is a meaningful test to include because the cast are the people who actually appear on screen and are exposed to the audience. This isn't to dismiss the significance of having a support team for the film that also has gender equality, and attitudes towards in background staff can be reflected in the final film, however this is concealed from the the conclusions that an audience member forms based purely on what they see on screen. The provided reading for the project explained how surprisingly few films pass the original Bechdel test, and we found the same outcome in our own test with the given data. Only 3 out of the 50 movies, or 6%, passed our test in having 50% or greater female roles.

Hacksaw Ridge was the most disappointing by far, with only 5.56% women, while *Lights Out* has the highest percentage of female roles, with 71.43%. The other two movies that passed our test were *Don't Breathe* with exactly 50% female roles, and *Bad Moms* with a 62.5%.The average percentage of female characters across all the films in the data set was 29.48%. Our data only accounts for roles that were female, and roles described with unknown genders weren't included in the percentage of women. Therefore, the actual percentage of non-male roles would be higher. However, only 248 out of the provided 2075 roles in the data set were attributed to having unknown genders, which is only 11.95% of all the data, and divided over 50 movies, it would only cause an expected 0.239% difference in a given movie's percentage of female roles.

We also discussed about more effective tests that would allow us to analyze and assess the movies more accurately. Another more effective method of testing that we discussed about was looking at the screen time of each female actress, which wasn't included in the given dataset. This would give us a clearer picture of how much involvement female actresses actually had, while our current testing method overlooks the possibilities of all female actresses being background characters. We also thought to look at the specific roles of the female actresses, whether that included stereotypical roles or more main roles. All these ideas would give a more specific and accurate result on analyzing the gender equality(or inequality) in modern day movies.

Not only did we learn how to implement a test to analyze the ratio of female to male actors, but we also learned how to collaborate, working together to manage a GitHub repository. We came across many merge conflicts, and we had to work through them together to ensure that we didn't accidentally overwrite each other's code. Moreover, we also learned how to take a large problem, splitting it into smaller manageable parts, and discussing the most effective and efficient way of solving our problems.

Overall, we were able to not only apply our technical knowledge to this project, but also learn how to collaborate and analyze our test results.

Collaboration (1/2 page):

Task 1: We scheduled times to meet up and discussed how to implement the project, considering different design choices. During the design process, we drew diagrams of the classes' relationship with each other using a shared freeform document. After we finished discussing, we broke the tasks down and started implementing them. We all implemented the methods required by the java foundations.Graph interface. Initially, we copied our implementations from our LabM9 code, and adding implementation for edges—since HollywoodGraph is undirected—was trivial because it just involves calling on our arc methods once for each direction. Writing the constructor for HollywoodGraph was a team effort. We drew pseudocode and diagrams for it in Freeform and debugged it together.

Task 2: Besides the constructor in Task 1, the methods in Task 2 were less trivial. We discussed pseudocode and approaches to solving all the problems together, drawing and writing ideas out on a Freeform board, this included changes we needed to make to previous classes. Although we discussed and planned all the tasks together, we separated the implementation, utilizing Github similar to a professional setting. Sophie implemented tasks 2.0 and 2.1, Rachel implemented task 2.2, and Lilymoon implemented task 2.3. When we ran into any issues, we would consult and help each other. Then we debugged the tasks together, and collaborated on figuring out cleaner ways to write results from the 4 tasks to the testing output file.

Works Cited

Hickey, W., Koeze, E., Dottle, R., & Wezerek, G. (2017, December 21). *Creating the next Bechdel Test*. FiveThirtyEight. https://projects.fivethirtyeight.com/next-bechdel/