CSC111 Project 2 Proposal: TODO FILL IN YOUR PROJECT TITLE HERE

Hsin Kuang (Joseph) Cheung
Tsz Kan Charlotte Wong
Yan Lam
Chun Yin Liang

March 4, 2025

Problem Description and Research Question

TBD

Computational Plan

We are implementing a Binary Tree to match users to generate a list of matches based on the users' preference for attributes. Before using the app, each user is asked to choose whether they are looking for: friends, potential romantic partners, or both. Two trees can be generated based on the user's answer: one for matching friends, and one for matching potential romantic partners. Then, the user is asked to enter their answer for each attribute. Each user ranks their priorities for different attributes (e.g., common interests, program of study, personality traits).

According to their priorities, a binary decision tree is built using these ranked attributes as decision nodes. For example, if a user prioritizes hobbies the most, then hobbies will be at the first level of the tree. Then, the most important attribute is at the root of the tree, and the least important attribute is at the bottom of the tree. The root of the tree is " ", and each node could be either: `Yes` (match), or `No` (mismatch), except the leafs where nodes are representing users. The program will systematically traverse through the tree (taking the `Yes` path). After reaching a leaf, the user will be added to `result`, which is a list of potential matches in order. If there is more than one leaf with common attributes, start from the leftmost node. (For example, if user A and user B share the same attributes, user A will be added to `result` first.) After the first all-`Yes` path, the second priority would be given to the path where the last level is `No` (which means the least important attribute is mismatched, but other higher-priority attributes are matched). The final full list `result`, including all users, is generated and ordered according to ranking based on compatibility.

The user will be first shown with the first matching user in the `result` list (i.e., `result[0]`). If the user "refreshes" the page, then the next matching user (`result[1]`) will be suggested, and so on. If the user successfully matches with another user, they will be stored in `romantic_matches` and/or `friend_matches`, which are lists of sets, with the data type of `list[set[User]]`. Each set contains two users, which are instances of the `User` class. Also, `user_list` is a list of all users in the system, with the data type of `list[User]`.

After users are matched through the system, they can be represented as a graph. Network graphs will be used to visualize the connections between all users. There will be two graphs, one targeted at people looking for romantic relationships and one for friendships. Each node represents one user, and each edge represents a match between two users. The size of each node is directly proportional to its degree, i.e., a user with more friends will appear as a larger node in the graph. To distinguish between current and past relationships, the edges for current couples and friends will be in red, while the edges for ex-couples and ex-friends will be in blue.

These two network graphs are created from `user_list` (for generating all the nodes on the graphs), `romantic_matches`, and `friend_matches` (for generating the edges, and showing their connections). We will generate these interactive graphs using plotly, an interactive graphing Python library (Plotly, 2025). Then, as the developers of the application, we can zoom in and out of the graph, and view the detailed information of different users in our system.

References

Plotly. (2025). *Plotly.* https://plotly.com/python/
University of Toronto. (2024). *Quick facts.* https://www.utoronto.ca/about-u-of-t/quick-facts