**TP2 Update:**

* Using tkinter instead of pygame, and likely will not need sockets to implement multiplayer, so likely no modules.
* Because pygame is not being used, some user interface features such as toggling when hovering over a button and dragging tiles will not be possible.
* Although the AI plan has not changed, a strong AI may be less realistic as the runtime of the program is already significant, without great depth.

**Project Description**:

My term project is titled *Mahjong* and will make essentially create the Hong Kong variation of the game mahjong. It is a four player turn based tabletop game that involves swapping tiles to form a winning hand. It should provide playability for both multiplayer and against AI.

**Competitive Analysis**:

This has already been done as a 112 term project, seen here: <https://www.youtube.com/watch?v=ZW5ZfnYNE64>. I would like to implement the multiplayer and AI just as this project has. In addition, I would improve the user interface, using pygame. I hope to be able to display all the tiles rather than a static screen and have small animations to represent the tiles moving. There is a lot more information that can be shown in mahjong, such as tiles each player has thrown out, and what melds they have made. There are also many minor rules and points calculations that would be complex that I would implement. Ideally, the interface would look something more akin to

<http://www.mahjongready.com/Games/play/HongKong-Mahjong.htm>.

Longer term stretch objectives would be to have game settings such as to adjust minimum *faan* (number of points to win), AI difficulty, highlight the discarded tiles that match the one you are holding in your hand, and make a mode where play continues past a single game and the table rotates as in normal mahjong. One major goal would be to create an assist mode with hand auto-sorting that indicates completed melds in one’s hand and tiles that can potentially pair together to form a meld. This would likely involve a very complicated form of recursive backtracking. This could also pair with a system for tracking the many different point rules, showing the possible point values of each hand. Using this auto-sorting along with potential winning point values for heuristics could make a very strong AI (and possibly even alpha beta pruning).

**Structural Plan**:

There should be a main file, \_\_init\_\_.py, which runs everything. Graphics may be separated into a separate file, depending on how messy it ends up. A Player.py file will contain the Player class, as the four players in the game each have their own tiles and can take actions. Computers would be a subclass of Player, having an AI system. Pictures, primarily of the tiles, will be stored in an images folder and will be named such that they can be checked by string formatting. There will also be separate files checking for the legality of different melds and for the score of a winning hand.

**Algorithmic Plan**: A detailed algorithmic plan for how you will approach the trickiest part of the project.

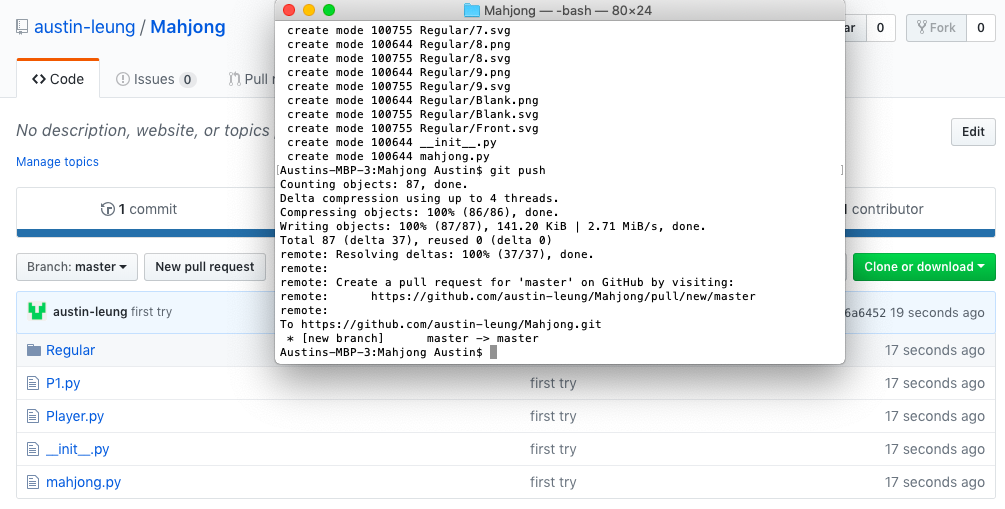
The most algorithmically difficult part of the program will be the assist mode utilizing recursive backtracking to return strong hand configurations. This is no clear answer for this, but the program should aim to give the few hands that would be strongest and explain why.

1. Create a function that identifies already completed melds in your hand
2. Sort the rest of the hand by pair to see how many extra tiles are needed to form a winning hand at minimum
3. Use recursive backtracking (storing solutions in a list as the function backtracks) to find all potential hands with the minimum number of extra tiles that could win, sorted by potential points.
4. Expand this to take into account that some tiles cannot be waited on as they have already been discarded or used in a meld.
5. Create a heuristic value for each possible hand configuration, based on the probability of finding the necessary tiles, and the point value payoff.
6. Use the heuristic value calculator to make an AI with min-maxing (and perhaps alpha-beta pruning)
7. Possibly add onto heuristic value calculator taking into account
8. Display what types people may be collecting and what tiles are left
9. Update AI to alpha-beta pruning

**Timeline Plan** [5 pts]: A timeline for when you intend to complete the major features of the project.

* By 11/21: Working Mahjong game against very basic computer
* By 11/23: Improved user interface using pygame (?) with small animations, also calculating points and more intricate rules
* By 11/25: Implement multiplayer
* By 11/27: Implement a few extra small features such as even better interface and small settings
* By 11/29: Steps 1-3 of the algorithmic plan
* By 12/1 Steps 4-6 of the algorithmic plan
* Rest of the time: Troubleshoot and prepare for presentation, perhaps implement steps 7+

**Version Control Plan**:

****

I am using Github to backup my code, making additional commits once I’ve completed a chunk of code.

**Module List**:

Pygame and perhaps sockets?