Steven Su

# AMS 550.400 HW SET 1 Due Date: Oct 8

Last Compiled on October 9, 2012

## Problem 1 (10 pts):

To produce the two branches with the desired commit histories, the following GIT Bash commands can be executed:

```
$ mkdir hondacivic.git
$ cd hondacivic.git
$ git init
$ vi main.txt #Create main.txt and add Line A
$ git add main.txt
$ git commit -m "A is Done"
$ vi main.txt #Add Line B to main.txt
$ git add .
$ git commit -m "B is Done"
$ git branch alt #Create branch named alt that has A and B commits
$ vi main.txt #Add Line C to main.txt
$ git add .
$ git commit -m "C is Done"
$ git checkout alt #Switch to alt branch
$ vi main.txt #Add Line X to main.txt in the alt branch
$ git add .
$ git commit -m "X is Done"
$ git checkout master #Switch to master branch
$ git merge alt #Merge the master branch with the alt branch
$ vi main.txt #Resolve the merge conflict
$ git add .
$ git commit -m "D is Done"
```

The following figure represents the commit history graph when on the `master` branch:

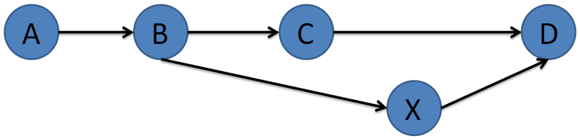

**Figure 1:** The commit history graph when on the `master` branch

The following figure represents the commit history graph when on the `alt` branch:

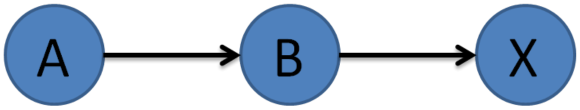

**Figure 2:** The commit history graph when on the `alt` branch

**Problem 2 (10 pts):**
Assume that you have created a new github repository at:

`https://github.com/ssu7/PoemCollection.git`

The following GIT Bash commands will produce what the problem asks for:

```
$ git config --global user.name "Steven" #Sets the user name
$ git config --global user.email "stevenemail@yahoo.com" #Sets the email of the user
$ mkdir hondacrv.git #Makes new git folder
$ cd hondacrv.git #Changes the directory to the git folder
$ git init #Initializes the git folder
$ git remote add Stanza1 git://github.com/nhlee/550400.stanza1.git #Creates alias for stanza 1
$ git remote add Stanza2 git://github.com/nhlee/550400.stanza2.git #Creates alias for stanza 2
$ git remote add Stanza3 git://github.com/nhlee/550400.stanza3.git #Creates alias for stanza 3
$ git pull Stanza1 master
$ vi main.txt #Add Title to Poem in VIM
$ git add .
$ git commit -m "Stanza1 Added"
$ git pull Stanza2 master
$ vi main.txt # Resolve Merge Conflict in VIM
$ git add .
$ git commit -m "Stanza2 Added"
$ git pull Stanza3 master
$ vi main.txt # Resolve Merge Conflict in VIM
$ git add .
$ git commit -m "Stanza3 Added"
$ git remote add origin https://github.com/ssu7/PoemCollection.git
$ git push origin master #Pushes main.txt to the previously setup github repository
```

**Problem 3 (40 pts):**

The issue of asynchronous collaboration on a class project is a common problem faced by many college students. Git, a free and open source version control software, offers a possible solution to this issue. In the proposed situation, there are 4 students, A, B, C, and D, who will all be working on different portions of a LaTeX/Beamer presentation. The main problem in this situation is to model a GIT workflow process that minimizes the amount of merge conflict that need to be resolved.

As stated in the problem statement, we will assume that each student will only edit the section of the Beamer presenation in `main.txt` which they are responsible for, and will NOT edit any other sections. Furthermore, as previously stated in the problem statement, none of the sections in `main.txt` overlap. In addition, if a student's section requires additional LaTeX packages, that student will add the necessary packages to the preamble portion of `main.txt`. Under these assumptions, the preamble part of `main.txt` is an exogenous variable. The endogenous variables (i.e. what we are interested in studying) are how effective different git workflows are at minimizing merge conflict resolutions. More specifically the endogenous variables can be thought of as how changes and edits to the the different sections of the Beamer presentation (*Introduction*, *Problem Statement*, *Timeline*, and *Deliverables*) are comitted using git. The endogenous and exogenous variables are interrelated in that the workflow for committing changes to `main.txt` (i.e the endogenous variable) will affect how often and how smoothly the preamble portion of `main.txt` is merged (i.e. the exogenous variable). In addition, we will further assume that all students are proficient at using git, and as such, the git proficiency of the students is an unimportant and ignored variable.

One possible workflow process that could be employed in this situation uses a remote git repository with a `master` branch and 4 other branches, say `A`, `B`, `C`, and `D`, one for each student. In this workflow, each student will clone the remote git repository onto their computer and (i.e. create a local git repository). Each student will then edit/create their respective section of `main.txt` while commiting these changes to only their assigned branch in their local repository. After a student is satisfied with his/her section of `main.txt`, he/she can then merge his/her final local branch to his/her local `master` branch. After this, the student will push both his/her local assigned branch to the corresponding remote assigned branch as well as his/her local `master` branch to the remote `master` branch. After a student pushes his/her local `master` branch to the remote `master` branch, he/she will be responsible for merging their section into the existing file. Since each student only edits his/her respective section of `main.txt`, the merge process should be fairly straight-forward and easy. Each section of `main.txt` in the `master` branch of the remote repository should be empty up until the student who is responsible for that section pushes his/her version of the presentation to the `master` branch in the remote repository. A student will only need to check to see if his/her section has been added correctly, and that the other sections haven't been affected (which they shouldn't). The only portion of `main.txt` that this is not true of is the preamble portion in which case a merge conflict may arise. However merging this section should be easy as well; each student should keep what is in the preamble of the version in the remote repository and add in the changes they made.

Using this workflow, the `master` branch in the remote repository should be very clean: the only nodes that appear in the commit history should be when each student adds his/her finalized section of the presentation. This workflow also allows for the edits and commits made by each student for their section of the presentation to be remembered through the commit history of the 4 student branches. This workflow also minimizes the number of merges that need to be made since each student will only merge their contributions to the remote `master` branch once.

Another possible workflow process uses one remote group repository with a single `master` branch. Each student will again clone the remote repository onto their own computer and create a local repository. However in this workflow, each student will edit and commit changes directly to his/her local `master` branch and every so often push and merge these commits to the remote `master` branch. Students will continue doing this until they are all satisfied with their sections. At the end of all the commits, the final, completed version of `main.txt` will be in the final commit of the remote `master` branch.

This workflow involves more pushes and merges to the remote `master` branch then necessary. Most of these merges should still be straight-forward since each student will only make changes to their own section of `main.txt` which does not overlap with any other section. However merges that involve the preamble portion will again require special attention since all 4 students can potentially change this portion of the LaTeX file. This workflow also mixes the commit histories of each section into the remote `master` branch's commit history which could potentially become very messy. This workflow also fails to utilize branching, which is a main feature of git.

There is a causal relationship between the number of times merges occur and the number of merge conflicts: the greater the number of merges, the greater the possible number of merge conflicts. The mechanism by which this causal relationship happens can be easily understood: anytime a merge happens, there is a chance that a merge conflict will arise which then needs to be resolved. This causal relationship is strong and consistent in its association. The causal relationship is also temporal in its nature: the cause (merges) precedes the effect (merge conflicts). Therefore by mimimizing the total times the students merge files, the first proposed workflow in essence minimizes the potential number of merge conflicts that need to be resolved. As such, the first proposed workflow model solves the previously formulated problem which was to develop a workflow strategy which minimizes merge conflict resolutions, and is thus a useful workflow model. We therefore can recommend using the first proposed workflow over the second.

We can test the recommended workflow model to see if it is indeed better then the second workflow by having 4 students working on a group presentation employ the recommended workflow scheme and having another group of 4 students use the second workflow scheme. Whichever group of students encounters less merge conflicts and has less trouble with merge conflict resolutions is the group that used the better workflow. In theory, the group using the recommended workflow should only have 4 merge conflicts to resolve in a worst case scenario. (Each student will only push their finalized version of their section to the remote `master` branch once.) The other group could potentially have far more merge conflicts since they are continually pushing and merging their unfinished work to the same remote `master` branch. Thus by using logic we can conclude that our recommended workflow does indeed minimize merge conflicts and is better than the other proposed workflow.

**Problem 4 (aka. Fair Play, 40 pts):**

Tennis is an exciting sport to watch once a person understands its rules. In a tennis match, one player, say Player A, starts out as a server and the other player, say Player B, starts as the reciever. The winner of a *game* is the first player to 4 points. The players switch serve after each *game*. The first player to win 6 out of 11 *games* wins the *set*. Note that a player must win by 2 *games* to win a *set*, so more then 11 *games* could be played per *set*. The first player to win 3 out of 5 *sets* wins the entire *match*.

The question we are trying to answer is whether serving first gives an advantage to a player. This is the same as asking whether the player who serves first is more likely to win the match then the other player who recieves first. In order to simplify the model, we will neglect any innate skill difference between the two players and assume that each player is of equal skill as the other player. In other words, we will assume that the probability of Player A winning a game when Player A is serving is equal to the probabiity that Player B will win a game when Player B is serving. Let us denote this probability as $P_{game}$.

By making this assumption, we are making the players' skill difference a unimportant variable and ignoring it. Another unimportant variable that we will ignore is any and all psychological factors that may impact players when they are competing. The exogenous variable in this problem is the probability of a player winning a game when he is serving, which as previously stated, is $P_{game}$ for both players. The endogenous variable is the probability that a player wins the match; let us denote this $P_{matchA}$ for Player A and $P_{matchB}$ for Player B.

To simplify our model, we will consider an abbreviated tennis match and analyze the match at the *games* level. Suppose that the first player to win 2 games wins a set, and that the first player to win 1 set wins the match. For now we will ignore the "win by 2 games rule." Imagine that each game is a Bernouli trial (i.e. a coinflip): the serving player has $P_{game}$ chance of winning and $1 - P_{game}$ chance of losing. Note that this means that games are independent of one another. Suppose Player A serves the first game. He has $P_{game}$ chance of winning that match, while Player B (the reciever) has $1 - P_{game}$ chance of winning the game. In the next game Player B serves and Player A receives, and thus, Player B now has $P_{game}$ chance of winning the second game and Player A now has $1 - P_{game}$ chance of winning the second game. The probabilities once again switch for the third game and so on.

Note that the set (and thus the match) could end after 2 games or all 3 games could be needed to decide the winner of the set. Thus we have the following equations:

$$P_{matchA} = P_{game}(1 - P_{game}) + P_{game}(P_{game})P_{game} + (1 - P_{game})(1 - P_{game})P_{game} \tag{1}$$

$$P_{matchB} = (1 - P_{game})P_{game} + (1 - P_{game})(1 - P_{game})(1 - P_{game}) + P_{game}(P_{game})(1 - P_{game}) \tag{2}$$

The first product term in (1) represents the probability of Player A winning the set by winning the first 2 games. The second product term in (1) represents the probability of Player A winning the first game, losing the second, and winning the third. The third term in (1) represents the probability of Player A losing the first game and winning the second and third games. Similarily the first product term in (2) represents the probability of Player B winning the first 2 games. The second term in (2) represents the probability of Player B winning the first game, losing the second game, and winning the third game. The last term in (2) represents the probability of Player B losing the first game and winning the second and third games. Careful analysis of (1) and (2) shows that if $P_{game} > 0.5$ then $P_{matchA} > P_{matchB}$, and serving first is advantageous. If $P_{game} < 0.5$, then $P_{matchA} < P_{matchB}$, and serving first is disadvantageous. Note that if $P_{game} = 0.5$, then $P_{matchA} = P_{matchB}$, and we find that serving first does not affect the fairness of tennis.

In the prevous analysis we ignored the "win by 2 games rule"; we now add it to our analysis. By adding this rule into our model, we can now see just how important the "win by 2 games rule" truly is. A set (and thus the match) can ONLY be decided when a player has won 2 games in a row. Note that if a player is down one game to his opponent, he can win the set by winning 3 games in a row. However, winning 3 games in a row entails winning 2 games in a row. (i.e. winning 1 game and then winning the next 2 games in a row.) Note that when a player is up by 1 game, he is in the process of winning 2 games in a row; he has won the previous game and just needs to win the upcoming game. This logic shows that it is not necessary to consider situations where one player is ahead by 1 game. Being ahead by 1 game simply means that a player is in the process of trying to win 2 games in a row. As such, we will only consider situations where the two players are tied in games.

Note that the probability of Player A winning 2 games in a row is equal to the probability of Player B winning 2 games in a row:

$$P_{Win\ 2\ Consecutive\ Games,\ Player\ A} = P_{game}(1 - P_{game}) = P_{Win\ 2\ Consecutive\ Games,\ Player\ B} \tag{3}$$

Because serving alternates between games, we can see that (3) is memoryless: regardless of how many games have been played in the past, the probability of winning the next 2 games in a row is always $P_{game}(1 - P_{game})$ for either player. A player must always serve one game and recieve the other. Consider the simple case when we are just starting a match and 0 games have been played. The probabilty of A winning the first two games and thus the set and match is $P_{game}(1-P_{game})$. The probability of B winning the first two games (and thus the set and match) is also $P_{game}(1 - P_{game})$. Taking this to the extreme, when the players are tied 1001 games to 1001 games, each player still has a $P_{game}(1 - P_{game})$ probability of winning the next two games and thus the set and match.

Based on (3) and our analysis above we can see that when the "win by 2 games rule" is in place, for all $P_{game} \in (0, 1)$, $P_{matchA} = P_{matchB}$, and thus serving first gives no advantage to a player. This result is different from our previous analysis when we ignored the "win by 2 rule", which concluded that if $P_{game} > 0.5$, then serving first is advantageous, and if $P_{game} < 0.5$, then then serving first is disadvantageous. When ignoring the "win by 2 rule", the only time serving first is not advantageous or disadvantageous is when $P_{game} = 0.5$. We see that the "win by 2 rule" makes it so that serving first offers no advantage or disadvantage to a player.

The model of fairness we outlined above is useful if we can find an empirical value of $P_{game}$. We can find this value by obtaining a large data set of professional tennis match statistics. $P_{game}$ will be equal to the fraction of games that professional players win when they are serving, averaged for all players. After coming up with this value for $P_{game}$, we can input it into (3) and get a value for $P_{game}(1 - P_{game})$. We can then compare this predicted value to the data set value (i.e the fraction of the time that 2 games are won in a row). We can also use the actual data to see if our prediction that serving first is neither advantageous or disadvantageous is true by computing the percentage of actual matches that are won by the player who serves first.