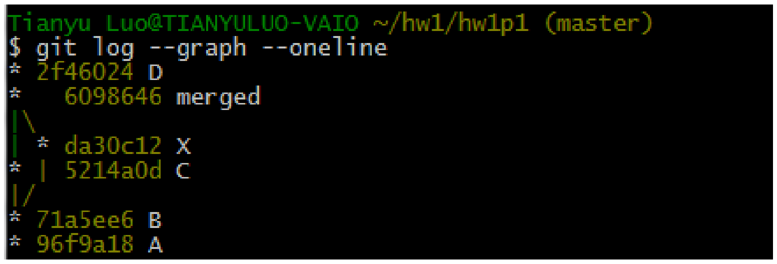
AMS 550.400 HW SET 1 Due Date: Oct 10

Author: Tianyu Luo, Last Compiled on October 10, 2012

Problem 1 (10 pts): A sequence of git/bash commands that yields a git folder with a commit history as required in Problem 1 is as follows:

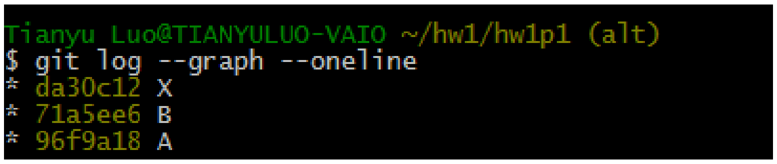
```
mkdir hw1
cd hw1
git init .
mkdir hw1p1
cd hw1p1
git init .
vi main.txt
git add .
git commit -m"A"
vi main.txt
git add .
git commit -m"B"
git branch alt
vi main.txt
git add .
git commit -m"C"
git checkout alt
vi main.txt
git add .
git commit -m"X"
git checkout master
git merge alt
vi main.txt
git add .
git commit -m"merged"
vi main.txt
git add .
git commit -m"D"
git log --graph --oneline
git checkout alt
git log --graph --oneline
```

Its commit history graphs are as follows:



```
Tianyu Luo@TIANYULUO-VAIO ~/hw1/hw1p1 (master)
$ git log --graph --oneline
* 2f46024 D
*   6098646 merged
|\
| * da30c12 X
* | 5214a0d C
|/
* 71a5ee6 B
* 96f9a18 A
```

Figure 1: Commit History on `master`



```
Tianyu Luo@TIANYULUO-VAIO ~/hw1/hw1p1 (alt)
$ git log --graph --oneline
* da30c12 X
* 71a5ee6 B
* 96f9a18 A
```

Figure 2: Commit History on `alt`

Problem 2 (10 pts): A sequence of git/bash commands which will conduct the task in Problem 2 is as follows:

```
git config --global user.name"tluo3"
git config --global user.email"luoty07@gmail.com"
cd hw1
mkdir hw1p2
cd hw1p2
git init .
git remote add stanza1 git://github.com/nhlee/550400.stanza1
git remote add stanza2 git://github.com/nhlee/550400.stanza2
git remote add stanza3 git://github.com/nhlee/550400.stanza3
git pull stanza1 master
vi main.txt
git add .
git commit -m"A"
git pull stanza2 master
vi main.txt
git add .
git commit -m"B"
git pull stanza3 master
vi main.txt
git add .
git commit -m"C"
git remote add origin https://github.com/tluo3/550400.homeworkset.1.git
git push origin master
```

Problem 3 (40 pts): In Problem 3, my task is to devise an effective workflow strategy for a team of four students to collaborate asynchoronously on their workstatement using Git.

`main.tex` includes two parts: the preamble part and the document part. As the preamble part shoud be same for all the team members to wirte their `main.tex`, so it doesn't depend on any other variables(i.e., it is exogenous). In both of my models, the origin `main.tex` file (i.e., the tex file at t=0) will only include the preamble part, which is also exogenous. The `main.tex` file at time t will depend on the changes A, B, C, D makes from time 0 to time t, and the original copy. Thus, it is endogenous.

My first workflow strategy is as follows:

- D prepares and uploads the original tex file to the shared repository,
- A, B, C, D pull the original tex file, and work on developing their own tex file,
- Before pushing their tex file to the shared repository, A, B, C, D should first pull the texfile from the shared repository, fix the merged conflict by hand, and finally push their own file to the shared repository,
- If they want to make changes, A, B, C, D should firstly change the file in the way they want, pull the newest tex file from the shared repository, fix the conflicts mannually, and finally push their own file to the share repository,
- Repeat step 4

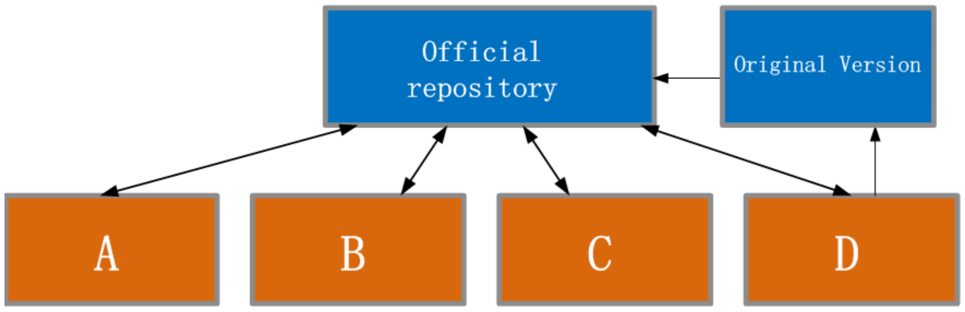


Figure 3: Strategy 1

We can see in this strategy, every time a team member wants to submit something new, he or she should firstly pull from the shared repository and fix the merged conflicts if there are any. The following example will illustrate why they should do so: A works on the original file and submits his or her version at time t. B works on the original file and submits his or her version an hour later. Actually, the submission of B will be turned down by Git, because Git won't let B's version overwrite A's version because their versions have conflicts (the details are described in Pro Git http://git-scm.com/book/en/Distributed-Git-Distributed-Workflows).

The strength of this method is that it is very simple and straightforward. However, the shortcoming is that it is very time consuming and ineffective because team members should manually fix conflicts again and again. This situation will become even worse if all the team members actively partcipate in working on the workstatement, and always update their file. The following example will illstrate my point: A makes change to his or her part, pulls the tex file from the shared repository, and is trying to fix conflicts by hand. During the time when A is fixing conflicts, B pushes a new version to the shared repository. Then A has to pull from the shared repository, and fixes conflicts again. Before A fixes all the conflict, C pushes a new tex file. A, who is very unlucky, should repeat what he or she did again.

Another workflow strategy is as follows. Here, D will act as a team leader. Besides the *Deliverable* part, D will be in charge of pulling files from other team members, merging files and fixing conflicts, and uploading the newest version:

- D prepares and uploads the original tex file to the official repository,
- A, B, C, D clone the official repository to their local repositories, develop their own tex files based on the original copy, and push the newest versions to their own public repositories,
- D pulls files from A, B, C, D's public repositories, merges the file and fixes the conflicts, and pushes the merged version to the official repository,
- Team members make changes to their files, push them to their public repositories, and informs D about the change,
- D will pull from team members' repositories, fixes conflicts and proofread the file, uploads it to the official repository regularly, and inform team members of the change regularly,
- Repeat step 4 and 5.

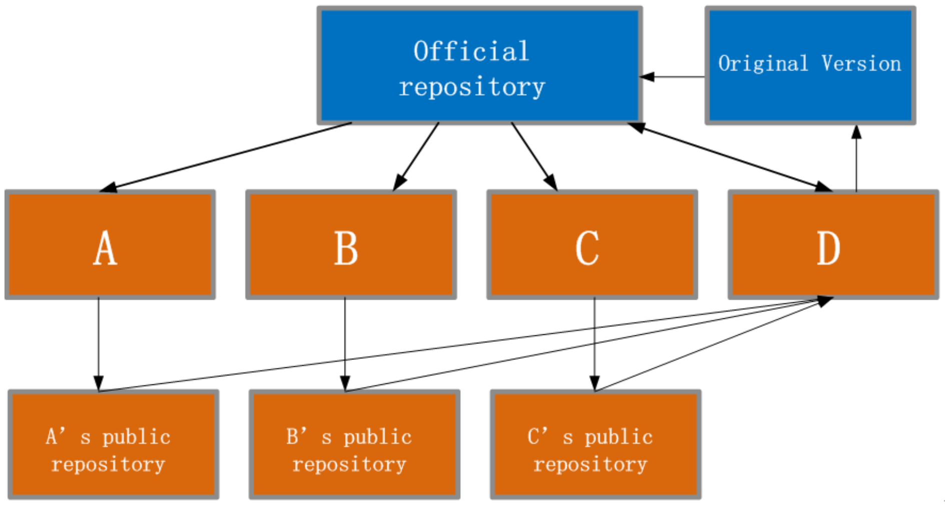


Figure 4: Strategy 2

In this strategy, team memembers can work on their own schedule and won't be worried about the change made by others. Our team leader D do not have to merge files and fix conflicts every time team members make a change. Actually, D can decide when he or she want to upload the file in the official repository. Thus, I recommend strategy 2. However, when the workstatement is simple and doesn't need too many changes, strategy 1 is all right.

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

The problem stated in Problem 4 is vague. First, what does " fairness" mean in tennis? Here, a clearer definition of "fairness" is " the possiblity of a more skilled player to win the whole game is 1." This definition introduces one more questions:"What are the requirements for a player to win?"

Firstly, it is useful to introduce some tennis background.The players (or teams) start on opposite sides of the net. One player is designated the server, and the opposing player is the receiver. The choice to be server or receiver in the first game and the choice of ends is decided by a toss before the warm-up starts. Service alternates game by game between the two players (or teams.) The outcome of a tennis match is determined through a best of three or five sets system. A set consists of a sequence of games played with service alternating between games, ending when the count of games won meets certain criteria. Typically, a player wins a set by winning at least six games and at least two games more than the opponent.A game consists of a sequence of points played with the same player serving. A game is won by the first player to have won at least four points in total and at least two points more than the opponent. (Resource:http://en.wikipedia.org/wiki/Tennis)

According to the background, we know to win a match, the only requirement is to win at least N sets first. However, when it comes to win a set or a game, there are two requirements: 1.the player should win at least N games or points at first; 2.the player should ahead by at least 2 games or points.Thus, to simpify our problem, we can just consider the separate influences of these two requirements on fairness, and then consier these two requirements together.

As to the first requirement, firstly we do not consider the server's advantage and assume a player's possibility to win a single game is p. If this player is more skilled, then $p > 0.5$. Thus, the fairness here turns into the following question: "Will a player, who's possibility to win a single game is larger than 0.5, win at least N games before his or her competitor?" In an ideal situation (we call it "absolute fairness" here), the relationship between the possibility to win a single game and the possibility to win the whole game is as follows:

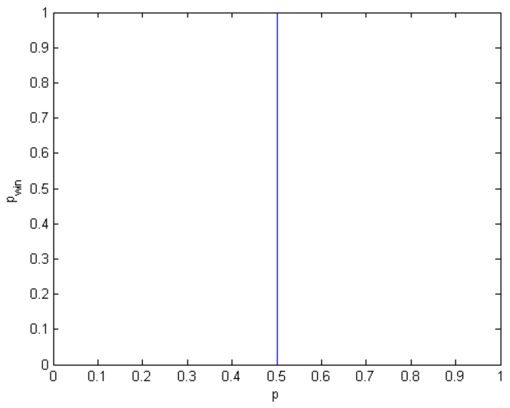


Figure 5: A plot of absolute fairness

Thus, our task is to see whether the first requirment will make the relationship between the skill level and the winning possiblity a good approximation of the absolute fairness plot. Because we do not consider the second part and server's advantage, so the maximum of total games played wil be 2N-1, and p stays the same. Thus, the probability that the player will win the whole game is as follows:

$p_{win} = p^N + \sum_{k=1}^{k=2N-1} \binom{N+k-1}{k} p^N (1-p)^k$

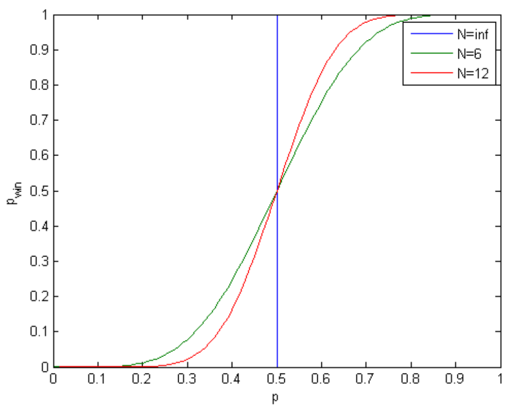


Figure 6: Plots when N=6,N=12,N=∞

I use Matlab to draw plots when N=6,N=12. We can see as N gets larger, it becomes more and more close to the absolute fairness plot.

Then let's take server's advantage into consideration. Assume server's advantage will improve the possiblity to win a single game by a ratio r. Then if the player is a server, the possiblity to win is pr. When the player is a receiver, the possiblity to win is $1 - r(1 - p)$. Let's start from the simplest case. If it is in the absolute fairness case, the server's advantage won't hurt the fairness if pr and $1 - r(1 - p)$ are both larger than 0.5. That is to say: when $rp <= 1$ holds, this relationship $r(1 - p) < 0.5$ always holds. Then we get when p is larger than $\frac{2}{3}$, no matter how r changes, the player will definitely win at least N games before his or her competitor. When the situation is N=6, we can see if $p > 0.7$, the player has the approximately 100% possiblity to win at least N games, when we don't consider server's advantage.When we consider server's advantage, there isno possible r to hurt the fairness when $p > 0.7$. The reason is as follows: $r < \frac{1}{p} = \frac{10}{7}$. Because $1 - p < 0.3$, so $(1 - p)r < \frac{3}{7} < 0.7$.

If we take the second part into consideration, then the maximum of total games won't be $2N - 1$.The number of total games a player win n is not fixed. It can go to infinity. And when n is given, the number of games is $2n - 2$. Rectify the formula above, we get

$$p_{win} = \sum_{n=N}^{n=infinity}(p^n + \sum_{k=1}^{k=2n-2}\binom{n+k-1}{k}p^N(1-p)^k)$$

The plot of p and p_{win} will be an approximate combination of plots of p and p_{win} plots generated by the first formula when n changes from N to infinity. That is to say, the second part will make the plot more close to the absolute fairness plot.

To summarize, the two requirements for winning improve the fairness of tennis, while the server's advantage decreases the fairness. When one player is much better than the other, the server's advantage won't hurt fairness much, and the game is relatively fair. When two players are close, the tennis game tends to be unfair. In my opinion, tennis is a relatively fair game compared to soccer in which it is common to see a strong team loses to a weak one.This model is useful to analyze the fairness of the tennis theoretically. However, there are two shortcomings of it. First, in this model, I assume that the server advantage will improve the stronger player and weaker player by the same ratio. However, it may not be true in reality. Besides, the value of p is hard to get. First, we cannot have two players play for large enough times to get an approximation of p. Besides, this p is for the winning a single game without the influence of the server's adavantage. In reality, the server's advantage always exists. As mentioned in class, one approach to overcome the second shortcoming is using the video game to simulate, if the video game doesn't consider server's advantage.