AMS 550.400 HW SET 1 Due Date: Oct 8

Last Compiled on October 10, 2012

General Instruction: To complete the homework set, you are required to do the followings. Your solutions must be typed in LaTeX using the course homework template. The progression of your homework solution is to be "recorded" by making a git folder specifically for this homework set. The burden of proof is on you, and if your git commit history is sparse, then you may be liable for a penalty. A paper copy of the PDF output of your LaTeX file is to be submitted to your instructor in class on the due date. *After* submitting the paper copy, but *before* the end of the due date, you will upload your work to your github by making a remote repository specifically for the homework, and post the link to the repository at the designated *Discussion* forum in Blackboard by making a thread just for you. The repository name in your github should be `550400.homeworkset.1` and the discussion forum thread should be named `YourFirstNameMiddleInitialLastName`, e.g., `BaracHObama` and `WillardMRommey`. You have till the end of the due date to finalize your github repository. However, any commit made after the class time of the due date will be inadmissible. *Your attention to details in following this instruction will be critical, and if not followed exactly at the time of collection, the homework set may be graded at* 90% *of the full score.*

Problem 1 (10 pts): Assume that you are starting from "scratch" at the directory `~/`. Provide a sequence of git/bash commands that yields a git folder with a commit history such that:

- the *master* branch has commits A, B, C, X and D,
- the *alt* branch has commits A, B, X,

Suppose that you are currently working on `master` branch. Draw its commit history graph (i.e., the graph portion of the output of `git log --graph --oneline`). Next, assume that you are on `alt` branch. Draw its commit history graph.

Solution:
Key code for this problem:

```
mkdir scratch
cd scratch
git init
vi main.txt
git add .
git commit -m ”A is done”
vi main.txt
git add .
git commit -m ”B is done”
git branch alt
git checkout alt
vi main.txt
git add .
git commit -m ”X in alt is done”
git checkout master
```

vi main.txt
git add .
git commit -m ”C in master is done”
git merge alt
vi main.txt
git add .
git commit -m ”merge from alt and conflicts resolved”
vi main.txt
git add .
git commit -m ”D in master is done”
git log –graph –oneline

Problem 2 (10 pts): Assume that you are starting from “scratch” at the directory `~/`. Provide a sequence of git/bash commands that yields a git folder and

- configure your git with your name and your email address,
- set up an alias for each of the git remotes listed below:

  ```
  git://github.com/nhlee/550400.stanza1.git
  git://github.com/nhlee/550400.stanza2.git
  git://github.com/nhlee/550400.stanza3.git
  ```

  Assume that each remote contains exactly single commit with a txt file for a single (different) stanza,
- pull to combine three stanzas of a poem,
- after the first pull, add the title of the poem,
- after the second and third pull, resolve the merge conflict,
- after resolving the third pull merge conflict, push the result to your (newly created) remote repository.

Solution:
The outcome is in https://github.com/Shihongli/550400.homeworkset
branch alt1
Key code:
mkdir scratch2
cd scratch2
git config –global user.name ”Shihong Li”
git config –global user.email lshzju@gmail.com
git init
git remote add stanza1 git://github.com/nhlee/550400.stanza1.git
git remote add stanza2 git://github.com/nhlee/550400.stanza2.git
git remote add stanza3 git://github.com/nhlee/550400.stanza3.git

```
git pull stanza1 master
vi main.txt
git add .
git commit -m "title added"
git checkout -b alt1
git pull stanza2 master
vi main.txt
git add .
git commit -m "conflict resolved"
git pull stanza3 master
vi main.txt
git add .
git commit -m "second conflict resolved"
git push https://github.com/Shihongli/550400.homeworkset master
git push https://github.com/Shihongli/550400.homeworkset alt1
```

Problem 3 (40 pts): Consider a team of four students, say, A, B, C and D, who just started working on writing a `latex/beamer` file, say `main.tex`, for a class presentation of their work statement. Assume that they do not wish to coordinate their schedules for a concurrent group meeting (both virtually and physically). Assume that:

- A is in charge of *Introduction*,
- B is of *Problem Statement*,
- C is of *Timeline*,
- D is of *Deliverable* part of the presentation.

In other words, their contributions to `main.tex` do not overlap. Then,

- first, devise a work flow strategy for the team so that they can collaborate asynchronously using `git`,
- next, devise yet another `git` strategy different from your earlier proposal.

Finally,

- discuss the strength and weakness of each of your proposed strategies in terms of merge conflicts resolution,
- make the final recommendation.

In order to answer this question, *build* a mathematical model, *following* the guideline from IMM. Use Section 1.4 and Section 1.5 of IMM as *role models*. For example, you are to identify which variables are exogenous and which are endogenous. More specifically, among other things, in your model, is the preamble part of `main.tex` an endogenous or exogenous variable? Note also that in addition to this issue, there are other issues that you are to consider. So, *be sure to consult IMM.*

Solution:

Strategy 1

Strategy one is to let A B C D do their own work asynchronously. Each worker follows the former one. The process is explained as below: // A initiates a git folder as master to write introduction. After A's done, B will set up a branch named B which has already contained A's work in it. Then B begins to do his work. After B's done, he needs to merge his branch into master and resolve conflicts. Afterwards, C sets up a branch from master named C, and writes timeline in it. After he finishes, C merges his branch into master and resolves conflicts. At last, D sets up a branch from master, and the branch D contains the work of A, B, and C. Then D writes his work. Finally, D merges with master and resolves conflicts.

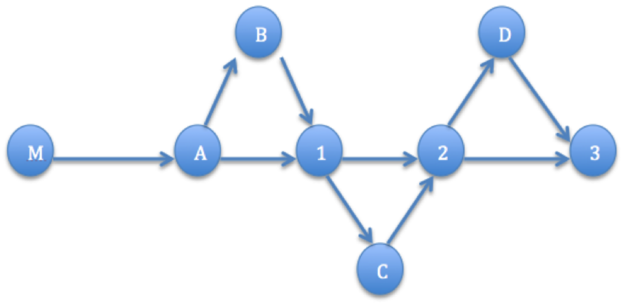

Figure 1: The graph for Strategy 1

Strategy 2

In Strategy two, each worker can do his work anytime he wishes. And each of them is responsible for a merge and resolving conflicts. The prcess is explained as below: // Someone initiates a git folder as master. Then four branches corresponding to four workers are set up. A works on branch A to write introduction; B works on branch B to write problem statement; C works on branch C to write timeline; D works on branch D to write deliverable. Each of them, after finishing their own work, should pull from master to merge what others have done and then resolve the conflicts. Afterwards, they should push back to master. Finally, on the final note of master branch, we will ask someone to reorganize the sequence and check if there still are any conflicts. The process can be illustrated by the figure below.

These two strategies both work and have their strength and weakness. When considering these two strategies, we treat other things such as the preamble part of main.tex as exogenous. Our endogenous are the length of time, the efficiency of work, and the number of merge conflicts resolution.

In the first strategy, workers are required to do their work in sequence. The number of merge conflicts resolution is three. So the strength is that it has lighter workload for workers, because they can process fewer conflicts in total. However, the disadvantage is that it can be time-consuming, since each worker has to wait until the former finishes his work. As for the second strategy, workers can start their work at the same time, but each has to do both pull and merge. Another weakness is that someone is required to rearrange the outcome when all of them have finished their work.

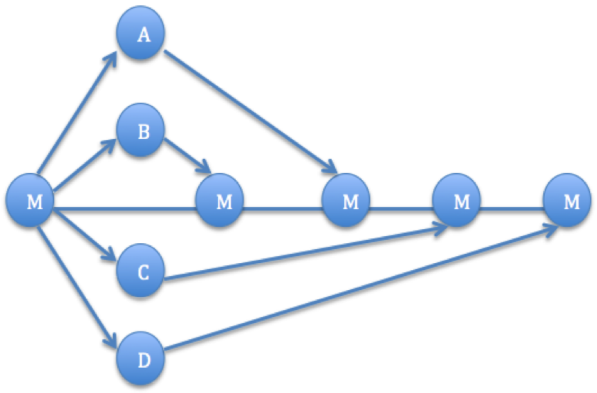

Figure 2: The graph for Strategy 2

In this case, I would like to recommend Strategy 1 for the following reasons. First, the workload for each person is not very heavy, so there will not be too much delay for each one to finish his work. Therefore, time is not a big issue. Second, resolving conflicts sometimes is not easy. Strategy 1 wins by reducing the workload for all the workers. However, if we are given another case with complicated work for each person, it is possible that I will suggest Strategy 2 to save time. that the only things we are concerned about are the merging of four parts: Introduction, Problem Statement, Timeline as well as deliverable. We do not focus on the other parts of main.txt which might be a short beginning or a conclusion. So those are considered exogenous variables in this context. Therefore we ignore the other parts of main.txt than the four core parts.

Problem 4 (aka. Fair Play, 40 pts): Answer the following question:

> Is the tennis game fair?

Note that unlike Problem 3, this question is vaguely stated. This is intensional, whence to begin, you will first need to clarify what exactly your question is. You may use the class discussion on this particular problem, but you *may not* directly refer to our discussion. Instead, formulate the model carefully but concisely in your own words.

Solution:

Final Remarks about Problem 3 & Problem 4: They are open-ended problems. However, your scores will be determined by how well do you follow the exposition style outlined by IMM and WMA. For both problems, your write-up should be

- self-contained,
- covering all four parts of Section 1.3 of IMM,
- paying a particular attention to any causal relation that you might be investigating, following Chapter 3 of WMA,
- answering questions that are explicitly asked in the problem statements.

For Problem 3, focus mostly on Step 2 and Step 3 of Section 1.3 of IMM. For Problem 4, focus mostly on Step 1 and Step 2. For each problem, minimum 1 pages and maximum 2 pages.