Team Project & Code Review

SWPP

April 15th

Juneyoung Lee

Project

- In the project, you're going to work as a team
- Each team will have one main repository
  - One of your teammates should create it & invite TAs
    (not yet; after team formation is finished)
  - Please create it as a private repo
- Each teammate should fork the main repository & work on it
  - Fork: create a copy of an existing repo (e.g. SWPP202101) under your GitHub account

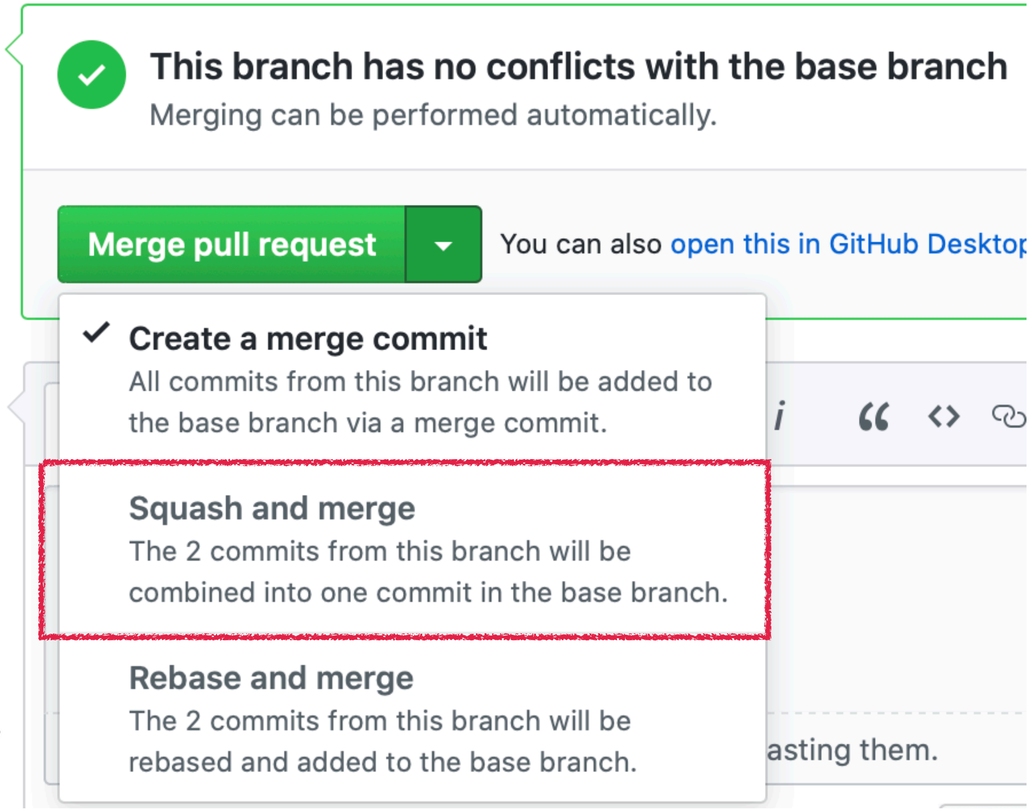
Collaborating via Sending Pull Requests

- You'll implement your new algorithm / fix bugs / do something on your forked repo
- After you finish your implementation, you send a pull request to the main repo
  - Pull request: "I want to merge my changes at branch A to the main repo's branch B"

- N Reviewers are appropriately designated (either online / offline)
  - Reviewers should understand how the code works & leave questions
  - Of course, reviewee can contact reviewer and kindly explain the code :)
  - The goal is to collaborate!

Pull Request

- A pull request is a unit of working feature
  - After it is merged, it should be compiled & all existing unit tests should pass
- Using ‘squash and merge’ will help this
  - It makes commit history linear
  - Having a linear history is important for finding out a buggy pull request
  - The main repo cannot be “intermediate state”



Before Sending Pull Request..

- Before *starting* implementation: please share what you're going to do with people & upload it on the main repo's Github Issue
  - This will help reduce burden of reviewers
  - Will prevent collision
- If you're going to implement a complex algorithm..
  - Having short slides for reviewers will be great (imagine you are a reviewer!)
  - This will help finding potential bugs in advance as well

Code Review Process

- Understanding the submitted patch & giving feedbacks (or accepting with ‘LGTM’)
- It is often slow & take longer time than writing a code alone
- It takes your emotional energy as well (Imagine someone attacked your code)
- But in the long run, the program becomes less buggy & simpler, so it takes less time
- Code review is essential when you are writing a very complex program

Reviewer's Work

- Reviewers can do 5 things:
  1. Check whether the PR correctly addresses the issue
  2. Point out possible correctness/performance issues (e.g. UB, value copy)
  3. Check whether existing libraries can be used (e.g. std::sort())
  4. See whether better C++ idiom can be used (e.g. using template)
  5. Check whether there is a missing test (e.g. FileCheck, GoogleTest)

Reviewer & Review

- Reviewee should write the description of PR succinctly
  - Imagine you are a reviewer! :)
- Reviewers can request reviewee to explain it via Skype/Phone call
- Reviewers can ask ‘why implement this way?’ and ‘why not implement this way?’
- Actually, this pass alone can find numerous bugs from the patch

N Reviewers, 1 Reviewee

- Reviewee should try hard to reduce burden of reviewers
  1. One PR should have one subject; split it if PR is too big
  2. Minimize diff before-after the pull request (very important!!)
  3. Describe the results of experiment (if possible)

One PR should have one subject

Common mistakes:

- Add feature A + format code in feature B
- Add feature A + make tweaks in B to make A faster
- Fix a bug in A, B where A and B are irrelevant
- Add a big feature A + enable it by default
  - Ideally, enabling it should be a separate patch, so reverting it can still preserve code

If PR is too big, split it

- 200 lines without tests / comments is assumed to be ‘big’
- You may think your PR cannot be splitted, but it is often true that it can be.
- Common cases:
  - Implement A with complex algorithm -> Implement A + add a better algorithm
  - Fix bugs A, A’, A’’ -> Fix A first + expand it to A’ + A’’

Minimize diffs

- Very important for reviewers to understand the code
- Whenever possible, try to stay succinct
  - Use for-each statement
  - Use `auto` type to hide unnecessary details
  - Use standard libraries often (e.g. std::copy, std::sort, etc)
  - Remove unnecessary changes in irrelevant code (e.g. space)