Persistent Data Structure Implementation for An Easy Code Repository
Optional Project for Course “Data Structure”

Junru Shao

1ACM Honored Class
Zhiyuan College
Shanghai Jiao Tong University

March 30, 2015
Outline

1. Aim of Project
2. Detailed Requirement
3. Time Phrase Partition
Aim of Project

- Another form of coding practice
- New challenge for debugging & testing
- Develop under capricious requirement
Another form of coding practice
New challenge for debugging & testing
Develope under capricious requirement
Aim of Project

- Another form of coding practice
- New challenge for debugging & testing
- Develop under capricious requirement
Form of Project

- 2~4 students in one group
  - Goldbricking is strictly prohibited
- Duration: This term + PPCA
Form of Project

- 2~4 students in one group
  - Goldbricking is strictly prohibited
- Duration: This term + PPCA
Form of Project

- 2~4 students in one group
  - Goldbricking is strictly prohibited
- Duration: This term + PPCA
Detailed Requirement I

In function

- Analogous to Git, not necessarily in implementation
  - Store the chronology of a code repository
    - Editing the current version of the code repository
    - Editing a specific historic version of the code repository
    - Forging history
    - Quick comparison between two versions
  - GUI is not required
  - Correctness comes first
Aim of Project

Detailed Requirement

Time Phrase Partition

Detailed Requirement I

In function

- Analogous to Git, not necessarily in implementation
- Store the chronology of a code repository
  - Editing the current version of the code repository
  - Editing a specific historic version of the code repository
  - Forging history
  - Quick comparison between two versions

- GUI is not required
- Correctness comes first
Detailed Requirement I
In function

- Analogous to Git, not necessarily in implementation
- Store the chronology of a code repository
  - Editing the current version of the code repository
  - Editing a specific historic version of the code repository
  - Forging history
  - Quick comparison between two versions

- GUI is not required
- Correctness comes first
Detailed Requirement I

In function

- Analogous to Git, not necessarily in implementation
- Store the chronology of a code repository
  - Editing the current version of the code repository
  - Editing a specific historic version of the code repository
    - Forging history
    - Quick comparison between two versions
- GUI is not required
- Correctness comes first
Detailed Requirement 1

In function

- Analogous to Git, not necessarily in implementation
- Store the chronology of a code repository
  - Editing the current version of the code repository
  - Editing a specific historic version of the code repository
  - Forging history
  - Quick comparison between two versions
- GUI is not required
- Correctness comes first
Detailed Requirement I

In function

- Analogous to Git, not necessarily in implementation
- Store the chronology of a code repository
  - Editing the current version of the code repository
  - Editing a specific historic version of the code repository
  - Forging history
  - Quick comparison between two versions
- GUI is not required
- Correctness comes first
Detailed Requirement I

In function

- Analogous to Git, not necessarily in implementation
- Store the chronology of a code repository
  - Editing the current version of the code repository
  - Editing a specific historic version of the code repository
  - Forging history
  - Quick comparison between two versions
- GUI is not required
  - Correctness comes first
Detailed Requirement I
In function

- Analogous to Git, not necessarily in implementation
- Store the chronology of a code repository
  - Editing the current version of the code repository
  - Editing a specific historic version of the code repository
  - Forging history
  - Quick comparison between two versions
- GUI is not required
- Correctness comes first
Detailed Requirement II
In Structure Design

- In Git
  - brute-force copying and real-time garbage collection is applied
- In our project
  - Emulating Git will not get full score
- Report the data structure you apply in presentation
In Git

- brute-force copying and real-time garbage collection is applied

In our project

- Emulating Git will not get full score

Report the data structure you apply in presentation
In Git
- brute-force copying and real-time garbage collection is applied

In our project
- Emulating Git will not get full score
- Report the data structure you apply in presentation
In Git
  - brute-force copying and real-time garbage collection is applied

In our project
  - Emulating Git will not get full score

Report the data structure you apply in presentation
In Git
- brute-force copying and real-time garbage collection is applied

In our project
- Emulating Git will not get full score

Report the data structure you apply in presentation
Detailed Requirement III
Teamwork

- Leader & coder & tester
- Unified coding style
Teamwork

- Leader & coder & tester
- Unified coding style
Detailed Requirement IV
Advanced (Not accounted for current score, but for PPCA)

- Inner interface designing
- Code refactoring
Detailed Requirement IV
Advanced (Not accounted for current score, but for PPCA)

- Inner interface designing
- Code refactoring
## Time Phrase Partition (Tentative)

<table>
<thead>
<tr>
<th>Phrase</th>
<th>Description</th>
<th>Week</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrase 1</td>
<td>Team formation</td>
<td>6</td>
</tr>
<tr>
<td>Phrase 2</td>
<td>Coding &amp; Testing</td>
<td>7~12</td>
</tr>
<tr>
<td>Phrase 3</td>
<td>Presentation</td>
<td>13</td>
</tr>
</tbody>
</table>