Identifying additional necessary changes from version control history

CPSC 503, Fall 2018 - Scott Newson

Supervised by Dr. Joel Reardon

Agenda

- Context
- Previous work
- Implementation notes
- Our work
- Results
- Future directions
- Questions

Context

- Software engineering: when teams and projects get large
- Adding new features is aided by understanding existing features
  - But identifying the full scope of existing features is hard
- Version control histories can provide useful information
  - But the version history is imperfect

Context

- Software engineering: when teams and projects get large
- Adding new features is aided by understanding existing features
  - But identifying the full scope of existing features is hard
- Version control histories can provide useful information
  - But the version history is imperfect

Our work

Automatically find related changes based on a search term and the existing version control history. Present those finding to support discovery and understanding.

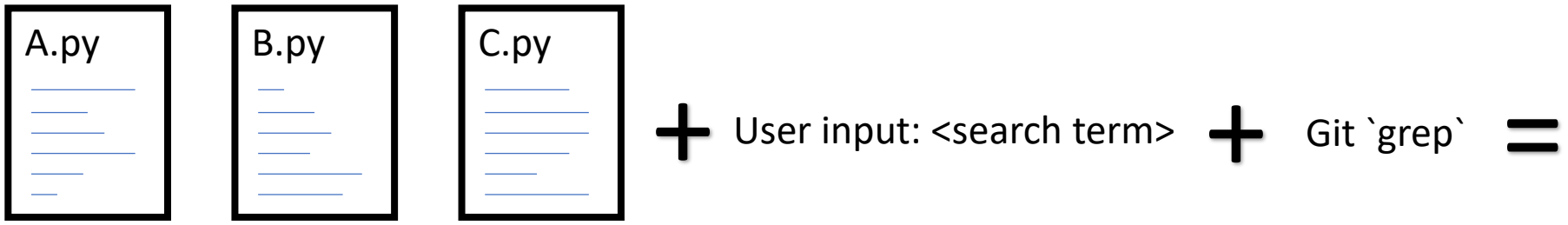
Previous Work

- Git version control system
  - Git's `blame`, `log` and `grep` commands
- Individual features are often inappropriately partitioned over multiple commits – (Arima, Higo, Kusumoto, 2018)
- Mining version histories to identify parts of projects that tend to change at the same time – (Ying, Murphy, Ng, Chu-Carroll, 2004) (Zimmermann, Zeller, Weissgerber, Diehl, 2005)
  - Identifies parts of code that change at the same time, where we identify related changes from different points in time

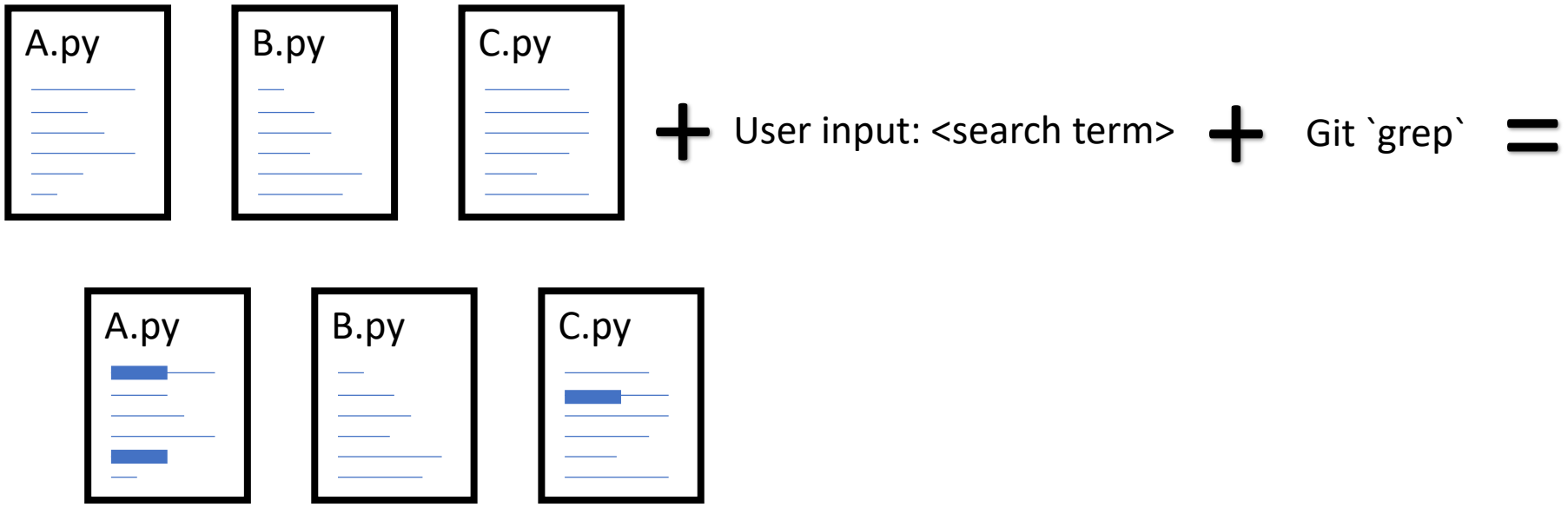
Implementation notes

- Git - 2.17.1 - as the version control system under analysis
- Mawk - 1.3.3 - for result formatting
- GNU Grep - 3.1 - for searching code-bases for fixed-string matches
- GNU Bash - 4.4.19 - as an interactive shell environment
- Python - 3.6.7 - as the general scripting wrapper in the work
- Ubuntu - 18.04.1 - for the operating system
- OpenSSL - 1.1.0g - as a test code-base for analysis

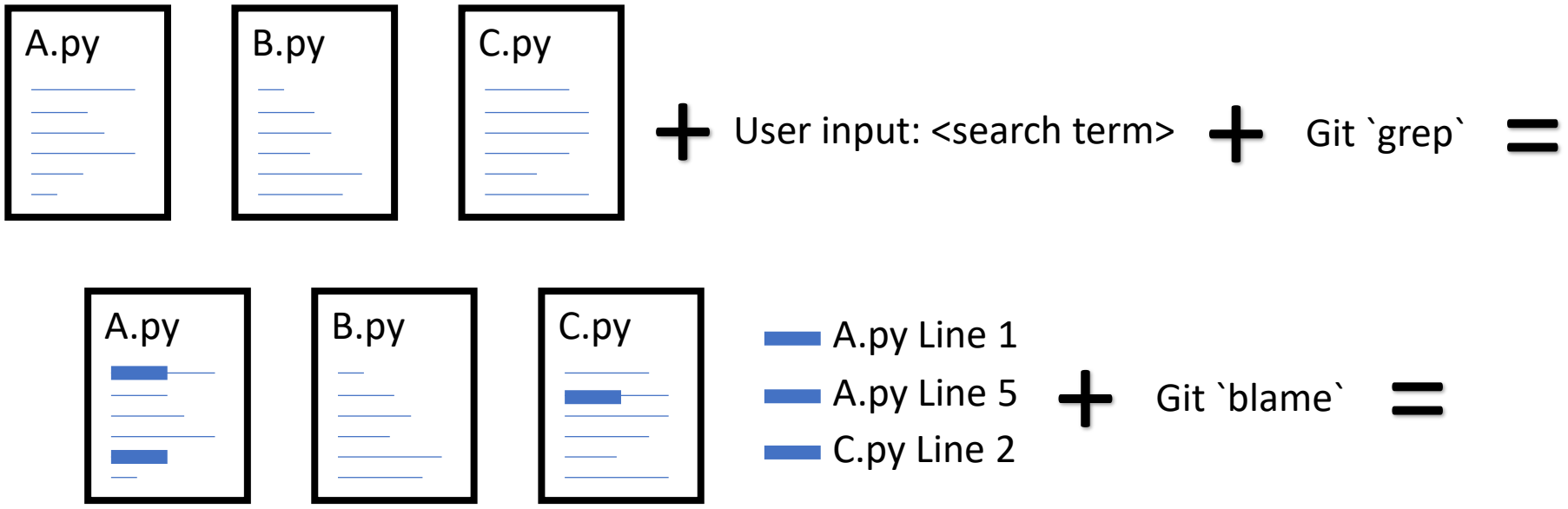
Our work – git ‘multi-blame’



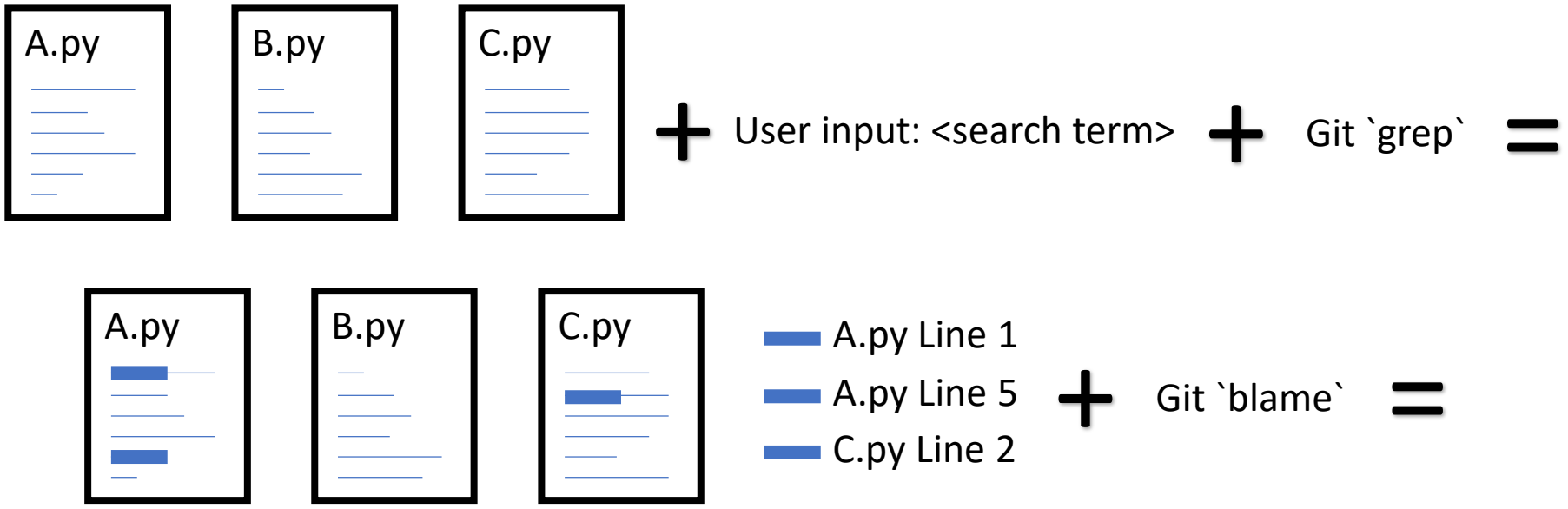
Our work – git ‘multi-blame’



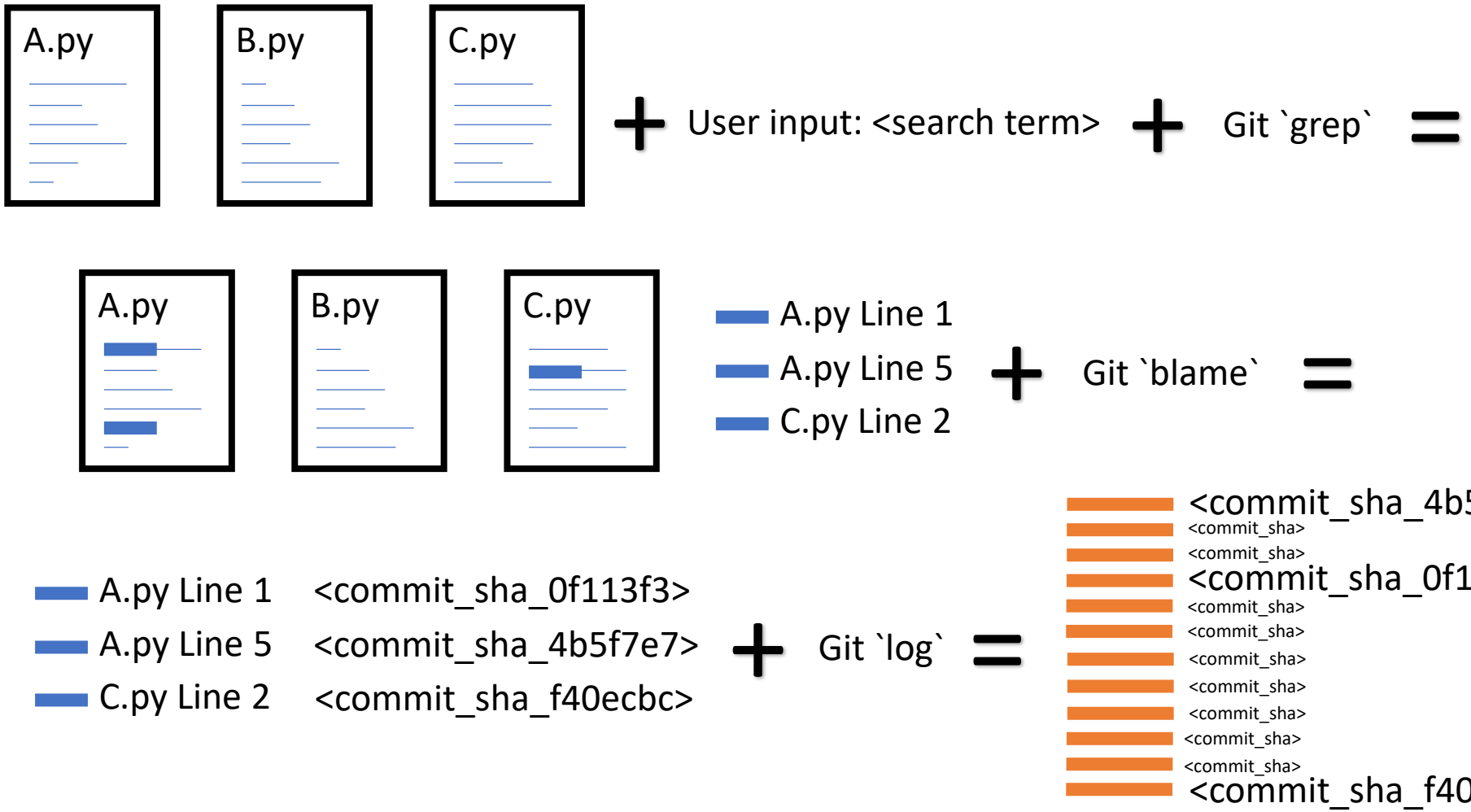
Our work – git ‘multi-blame’



Our work – git ‘multi-blame’

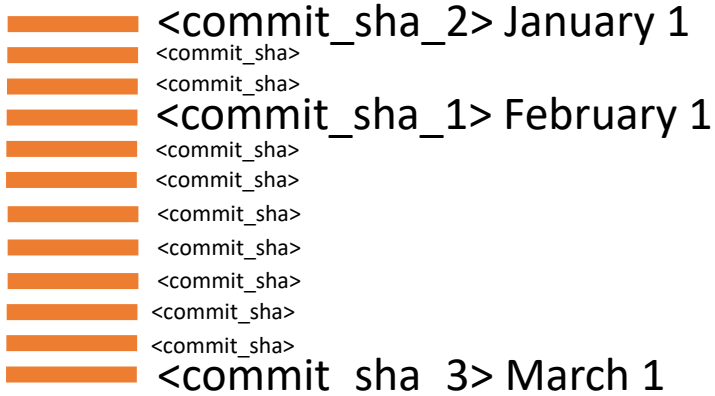


Our work – git ‘multi-blame’

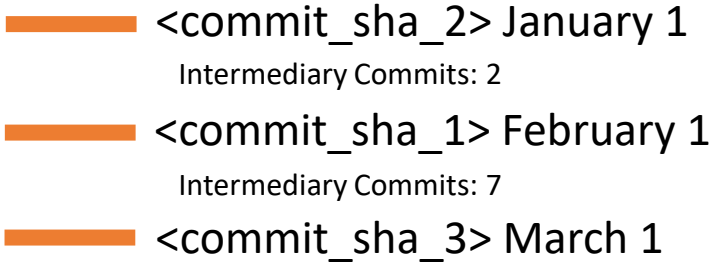


Our work – git ‘multi-blame’

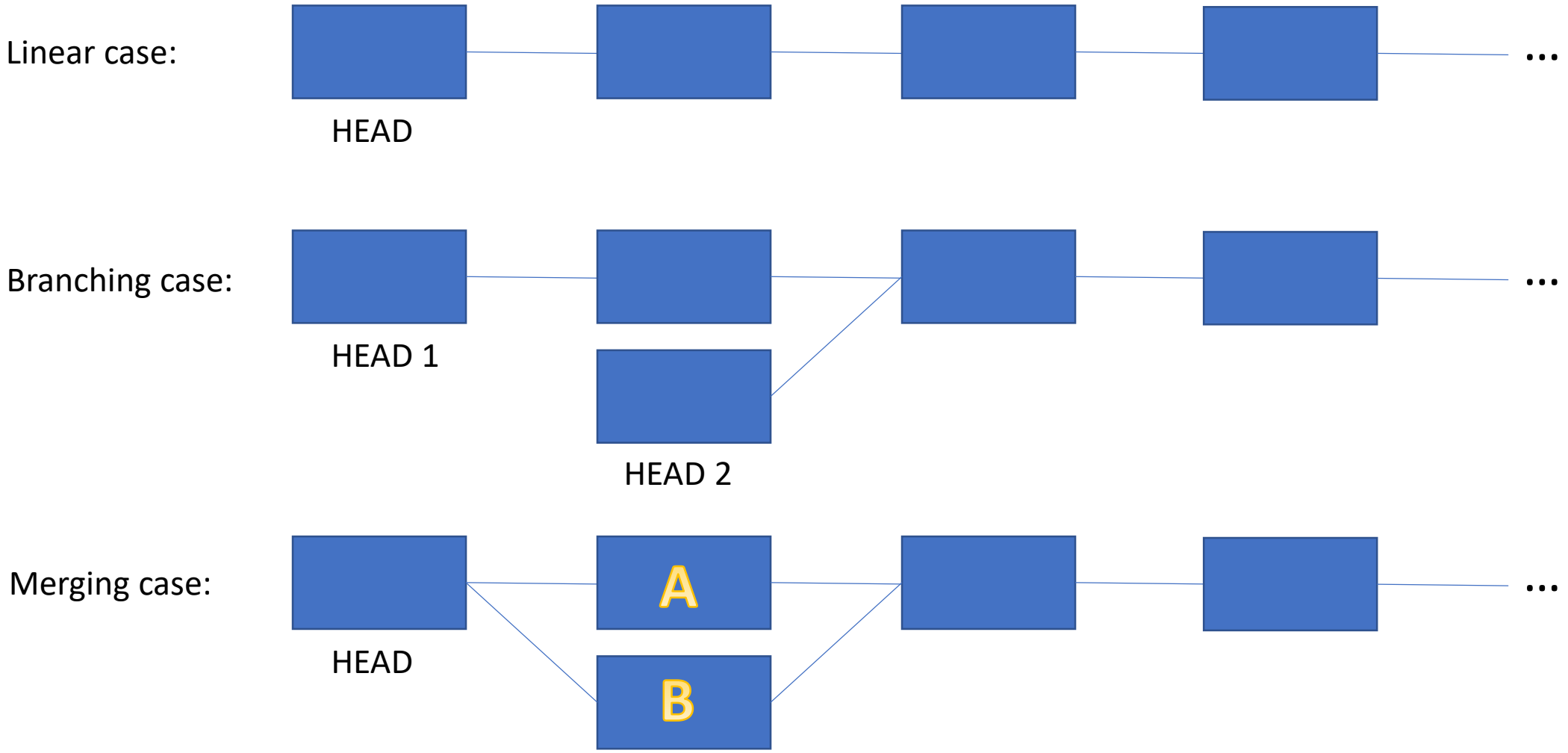
Full log output style:



Compressed log output style:



Partial ordering of commits



Results

- Input variants
  - File and line pairs
  - Search term
- Output variants
  - Full span of commits
  - Compressed intermediary commits
- Example usage on OpenSSL cipher suites
  - (Next page)

Results – OpenSSL cipher suites

```
# git multi-blame PSK-AES128-CBC-SHA
4b5f7e7 Update ossl_config.json for later BoringSSL commit
Intermediary count: 5692
b2f8ab8 Add PSK ciphersuites to docs
Intermediary count: 2
ea6114c Add RFC4279, RFC5487 and RFC5489 ciphersuites.
Intermediary count: 0
f40ecbc Initial new PSK ciphersuite defines
Intermediary count: 904
0f113f3 Run util/openssl -format -source -v -c
Intermediary count: 5480
ddac197 initial support for RFC 4279 PSK SSL ciphersuites
```

Results – OpenSSL cipher suites

```
# git multi-blame PSK-AES128-CBC-SHA
4b5f7e7 Update ossl_config.json for later BoringSSL commit
Intermediary count: 5692
b2f8ab8 Add PSK ciphersuites to docs
Intermediary count: 2
ea6114c Add RFC4279, RFC5487 and RFC5489 ciphersuites.
Intermediary count: 0
f40ecbc Initial new PSK ciphersuite defines
Intermediary count: 904
0f113f3 Run util/openssl -format -source -v -c
Intermediary count: 5480
ddac197 initial support for RFC 4279 PSK SSL ciphersuites
```

Results – OpenSSL cipher suites

```
# openssl ciphers 'ALL' | split ':' | xargs git multi-blame | statistics
```

- Total number of cipher suites analyzed: 125
- Total number of intermediary spans: 251
- Intermediary commit counts:
  - Minimum: 0
  - Maximum: 9215
  - Mean: 1551

Future Directions

- Application to tutorial generation
  - Example scenario: using this tool create a tutorial for adding a cipher suite to OpenSSL
- Validate with a user trial
- Handle merge cases
  - Likely just a change to the output format to support showing matches from multiple branches that have been merged

Future Directions

- Application to tutorial generation
  - Example scenario: using this tool create a tutorial for adding a cipher suite to OpenSSL
- Validate with a user trial
- Handle merge cases
  - Likely just a change to the output format to support showing matches from multiple branches that have been merged
- Applications to other persistent data-structures?

Questions?