Building Deduplicated Model Repositories to Assess Domain-Specific Language Evolution

Alexandre Lachance, Sébastien Mosser

MODELS 2024, Linz, Austria

24/09/2024

Context

- Evolving tooling for P4

- P4 is a DSL to model software-defined network behaviour by Intel and the Linux Foundation

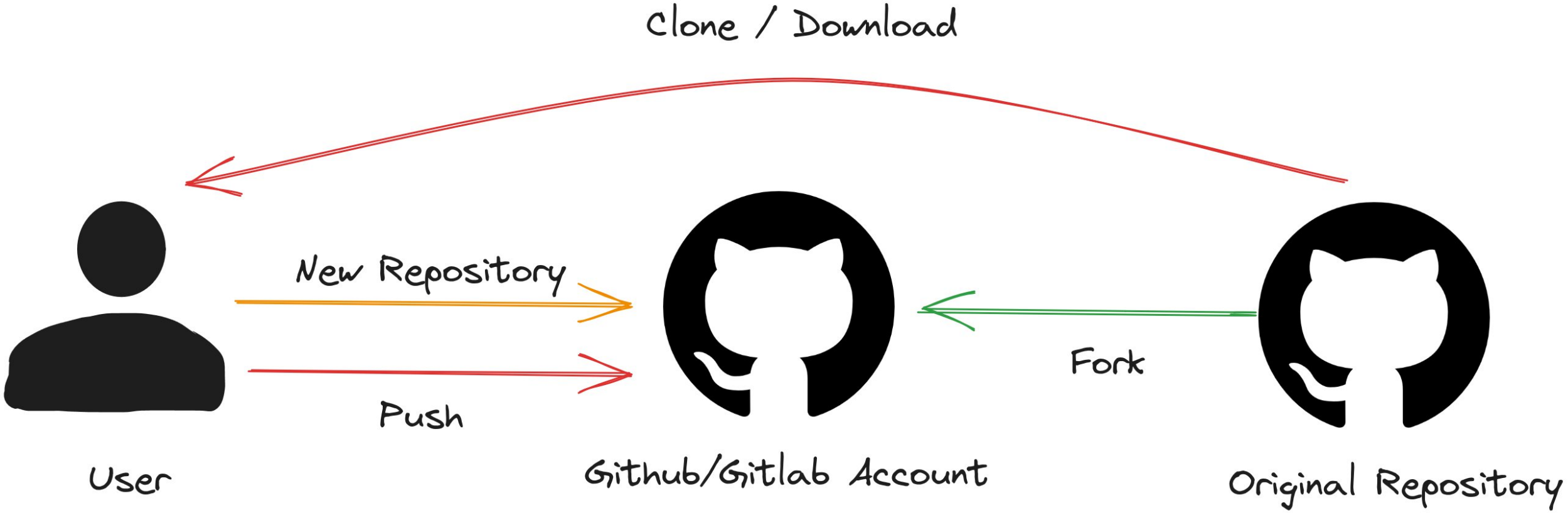
The Problem

- P4 language lacks a usable grammar (available only in PDF form or compiled in the existing compiler)
- So build our own grammar...
- How do we verify that this grammar is retrocompatible with previous models?
- Build a representative dataset of models from open source repositories to validate against
- Duplicates skew results...

Why is deduplicating hard?

- Duplicates are hard to find
- Very costly to compare different models together
- Need a way to avoid the need for exponential comparisons
- Need to reduce the search space

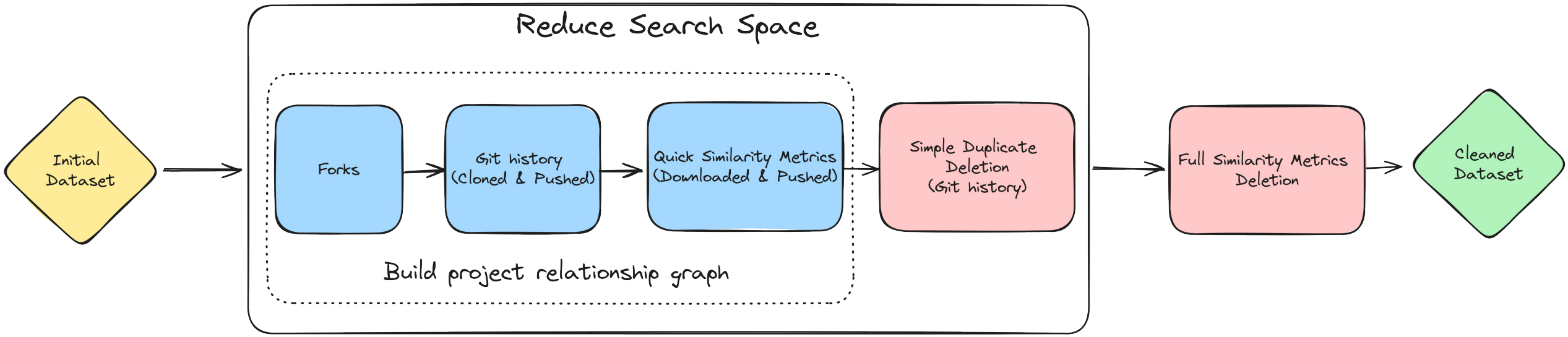
How duplicates happen



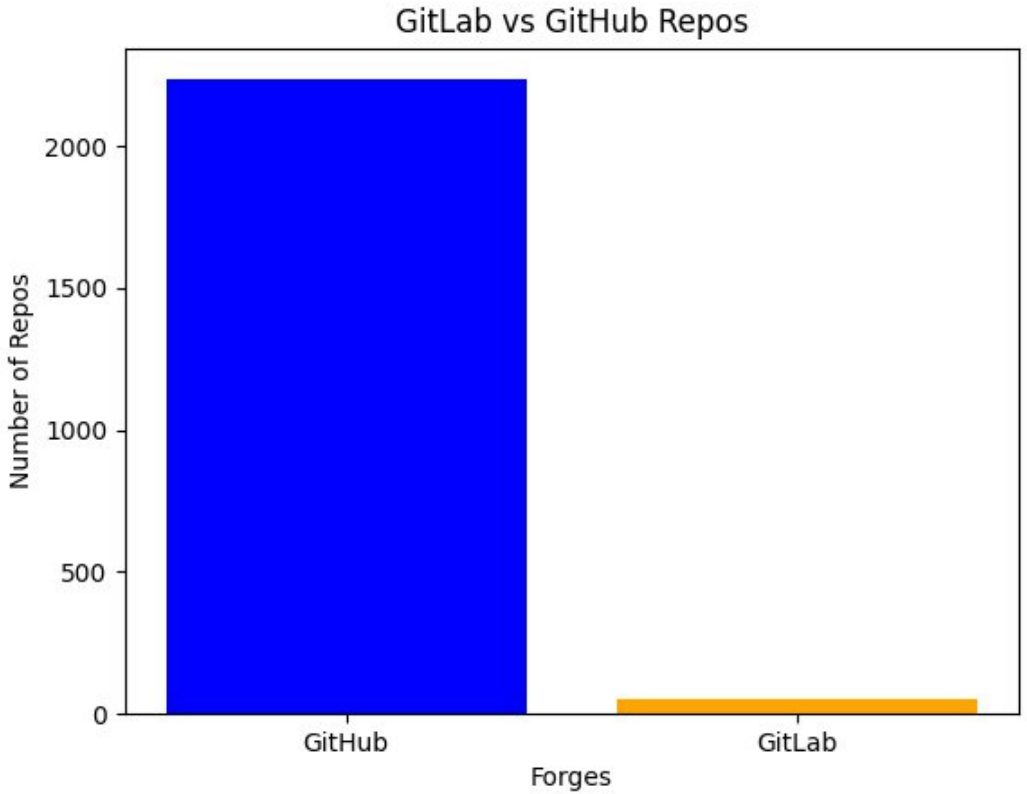
How to identify each type of duplication?

1	Forks	Use existing fork relations in metadata
2	Cloned & Pushed	Use commit ids to identify projects that originated from others
3	Downloaded	Use quick project comparison metrics to ~~identify projects that were downloaded~~ and pushed

Overview of Approach



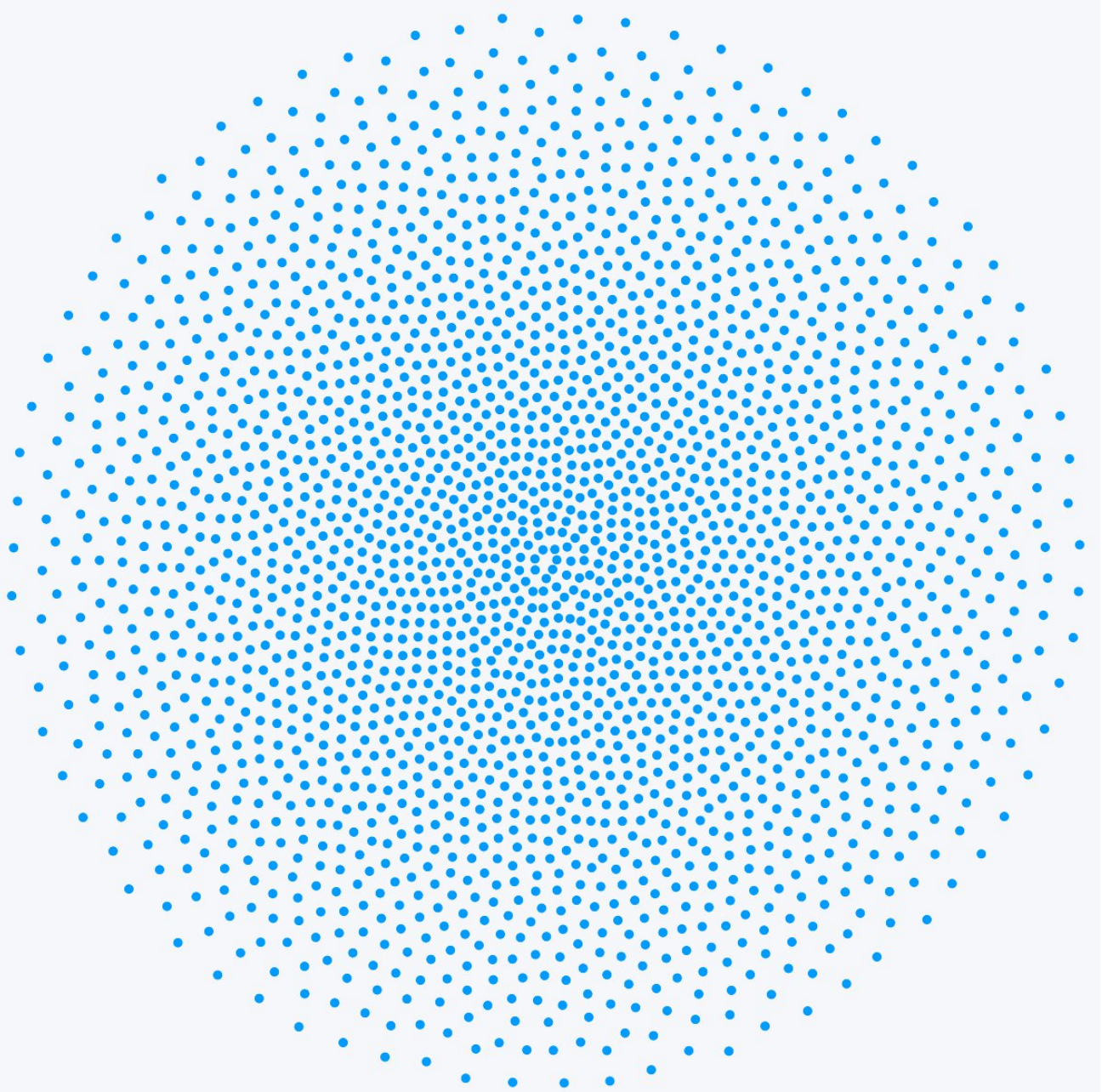
Dataset



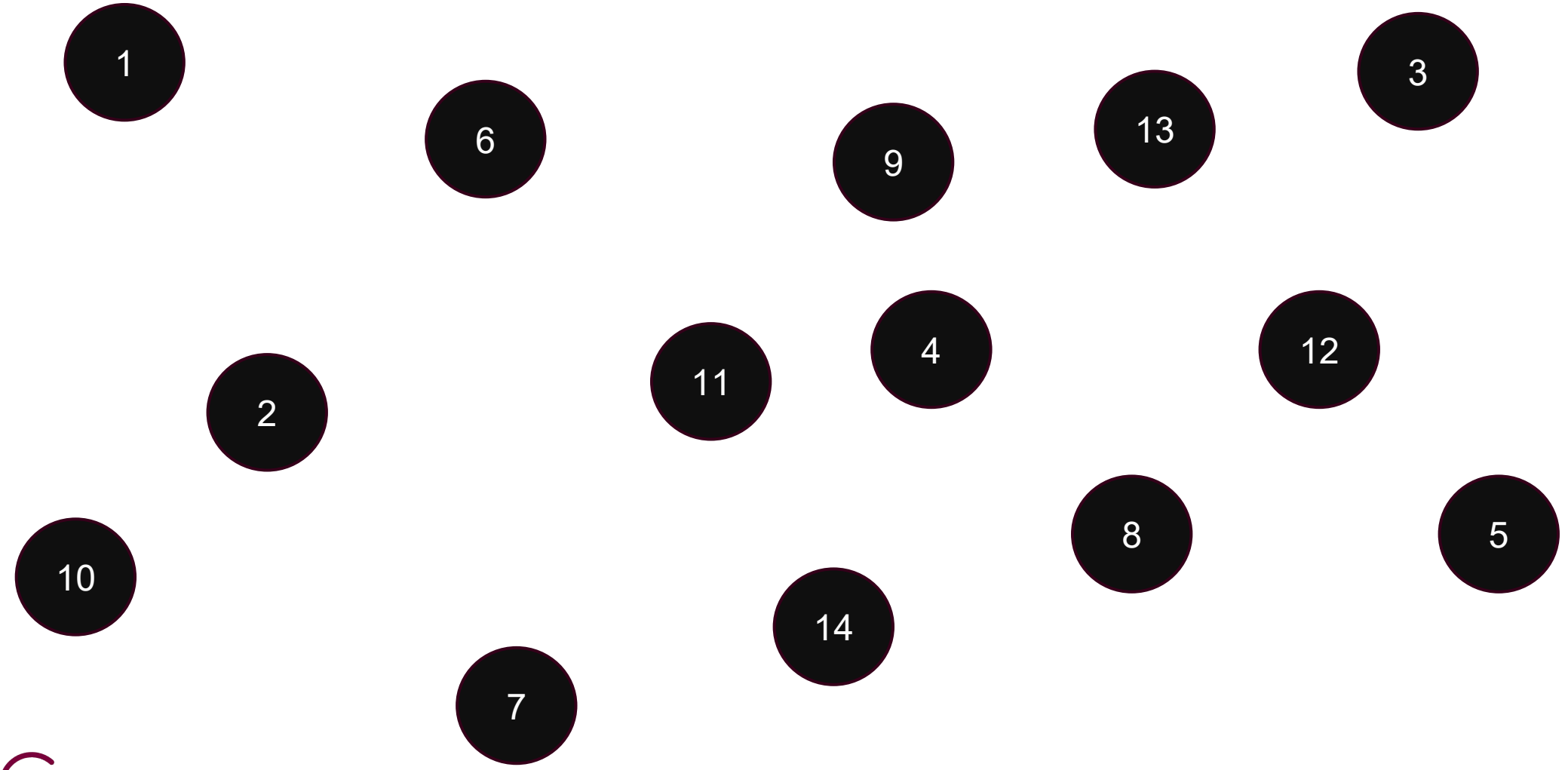
- Dataset of entire ecosystem
- 2234 GitHub repositories
- 49 GitLab repositories
- GitHub API 1000 repositories limit
- GitLab API returning errors on requests bigger than 50 repositories

The Starting Point

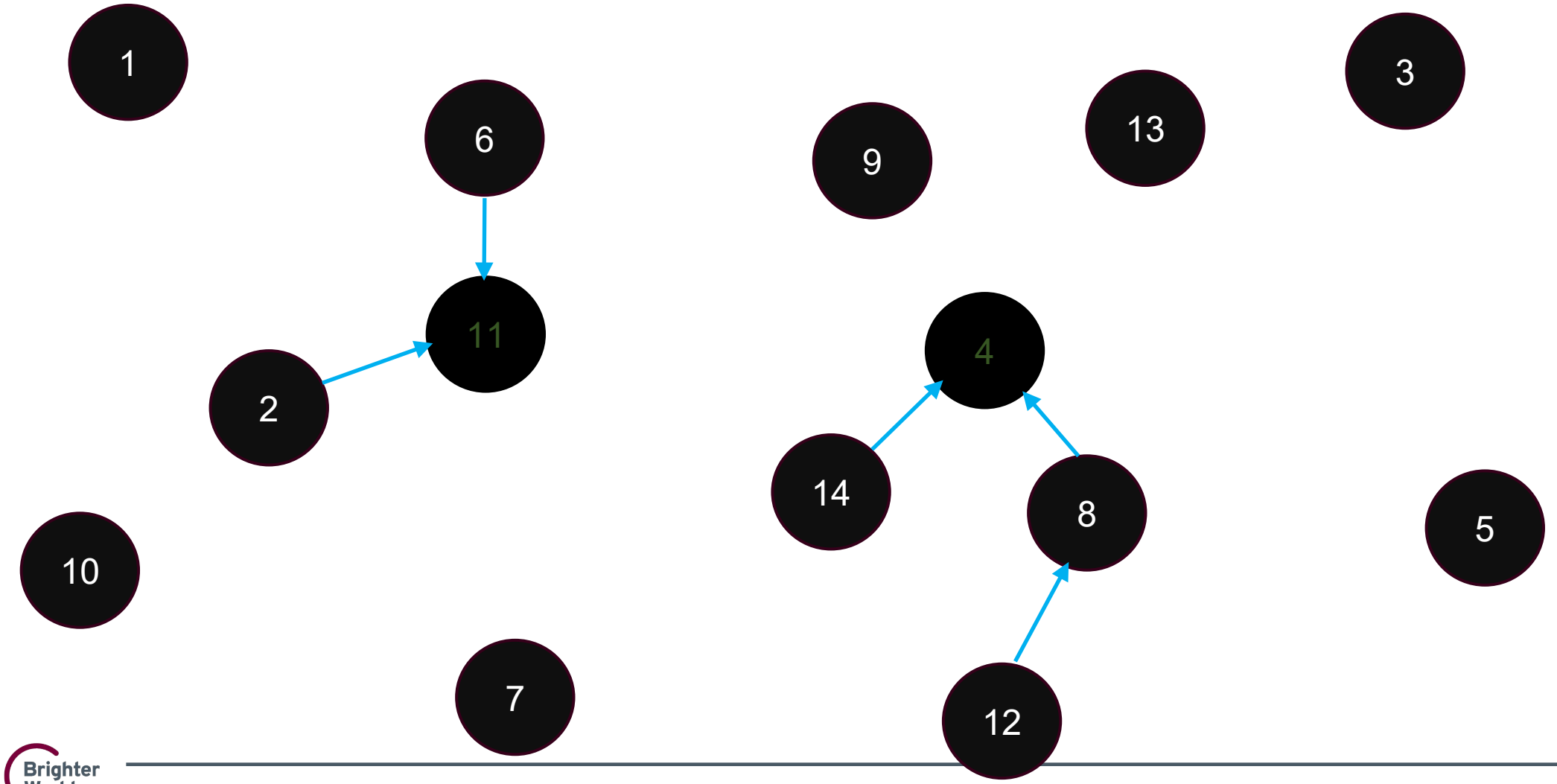
– 2315 Repositories



Step 1 - FORKS



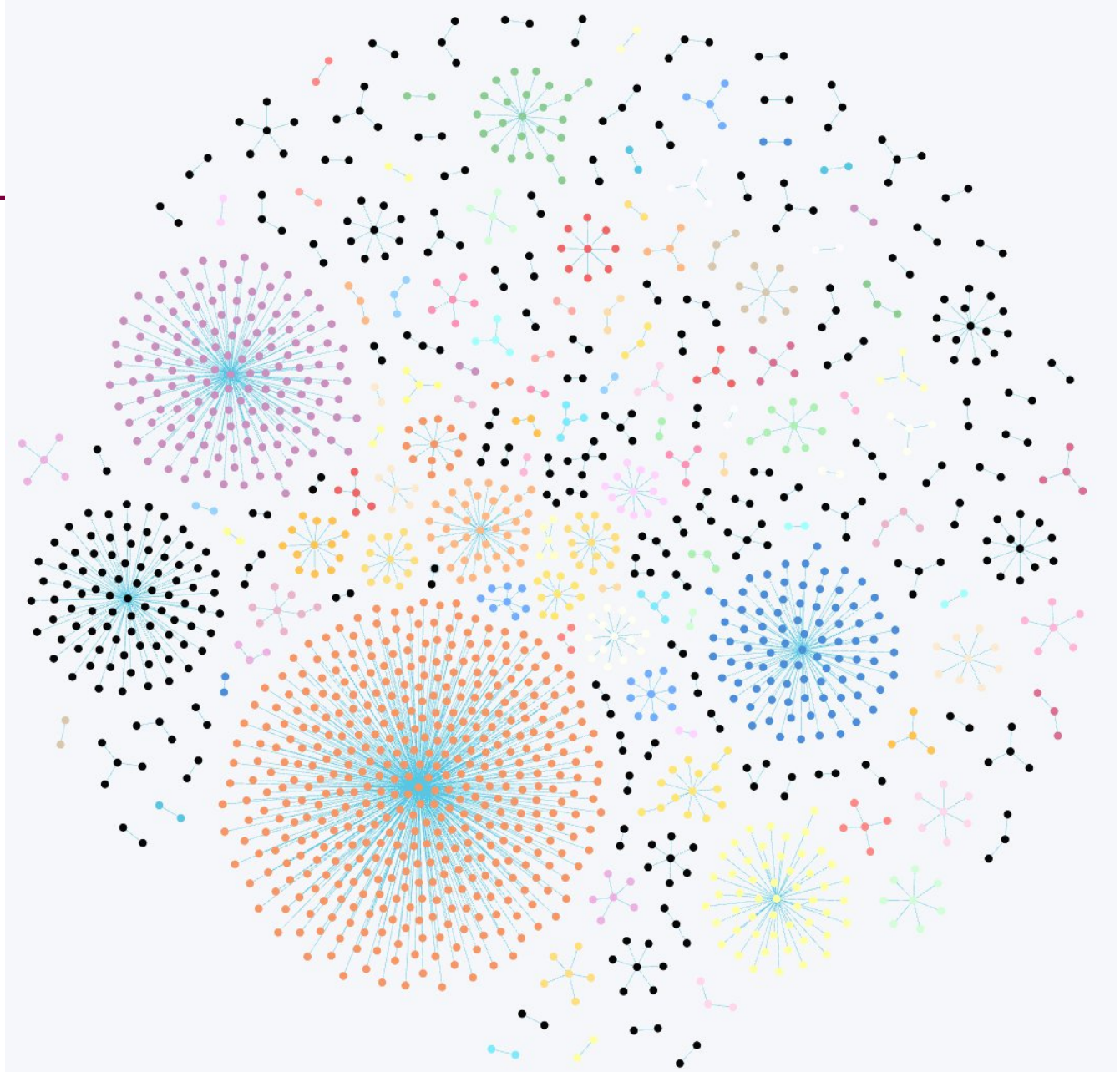
Step 1 - FORKS



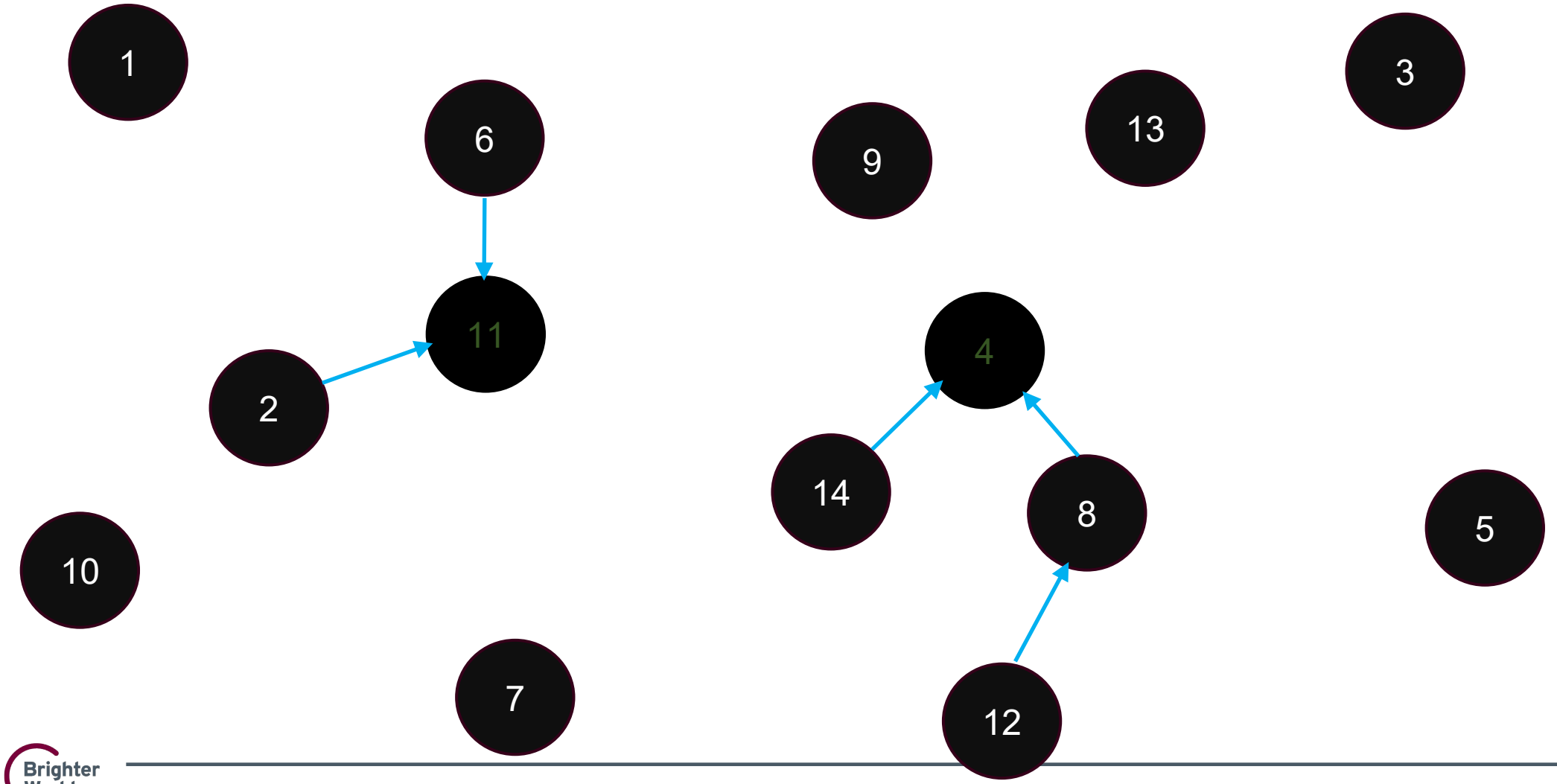
Results

- 1679 relationships created
- 72.53% of repositories connected

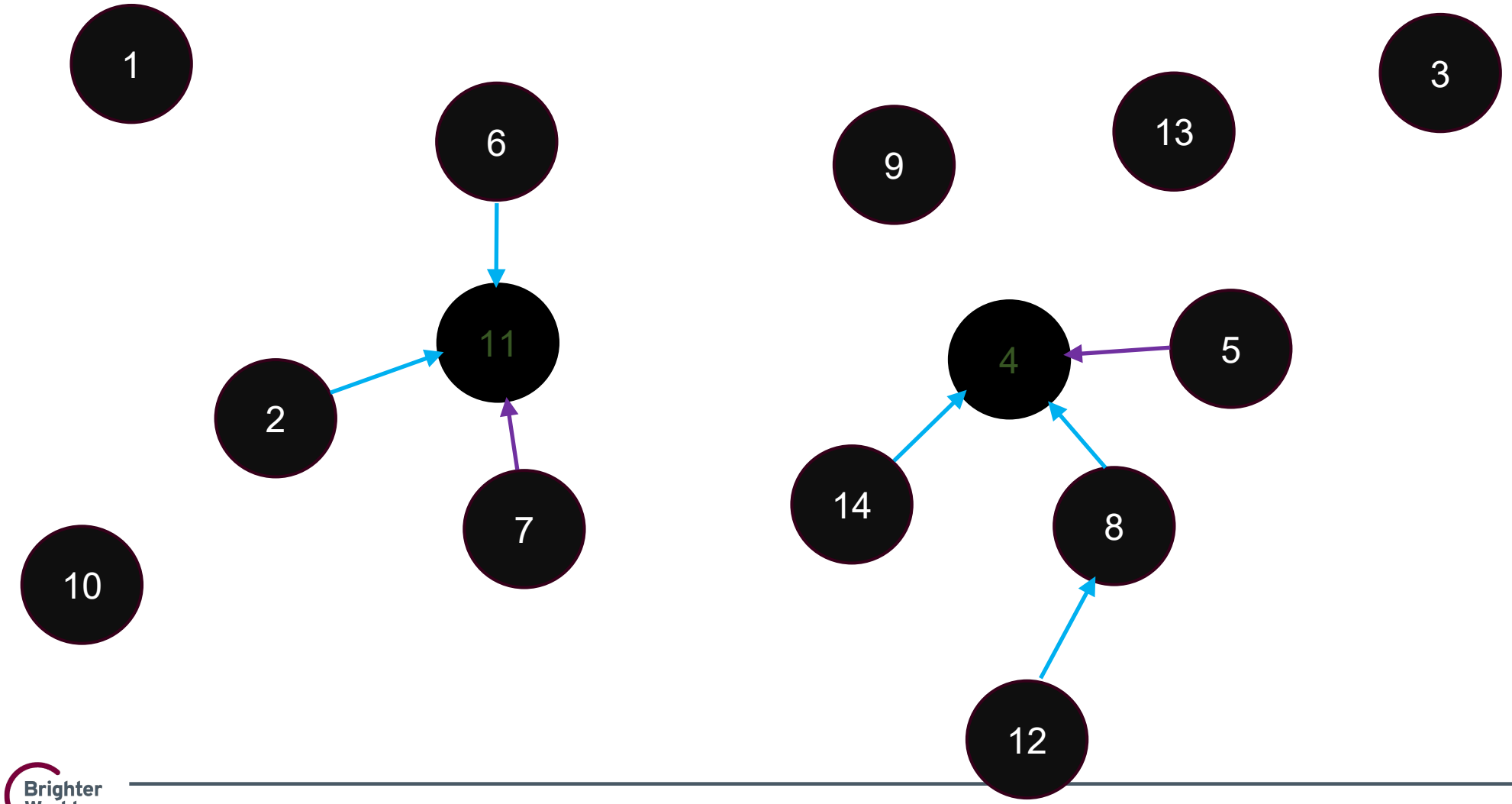
(Coloring done using Weakly Connected Components algorithm)



Step 2 - CLONED & PUSHED

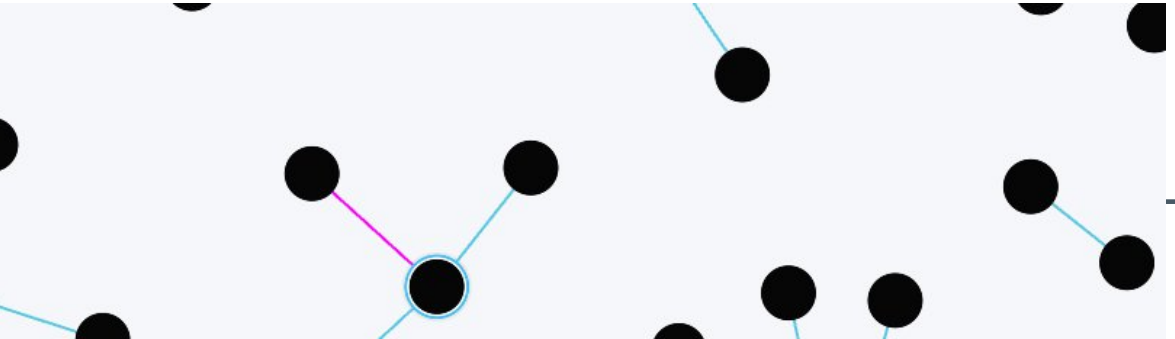


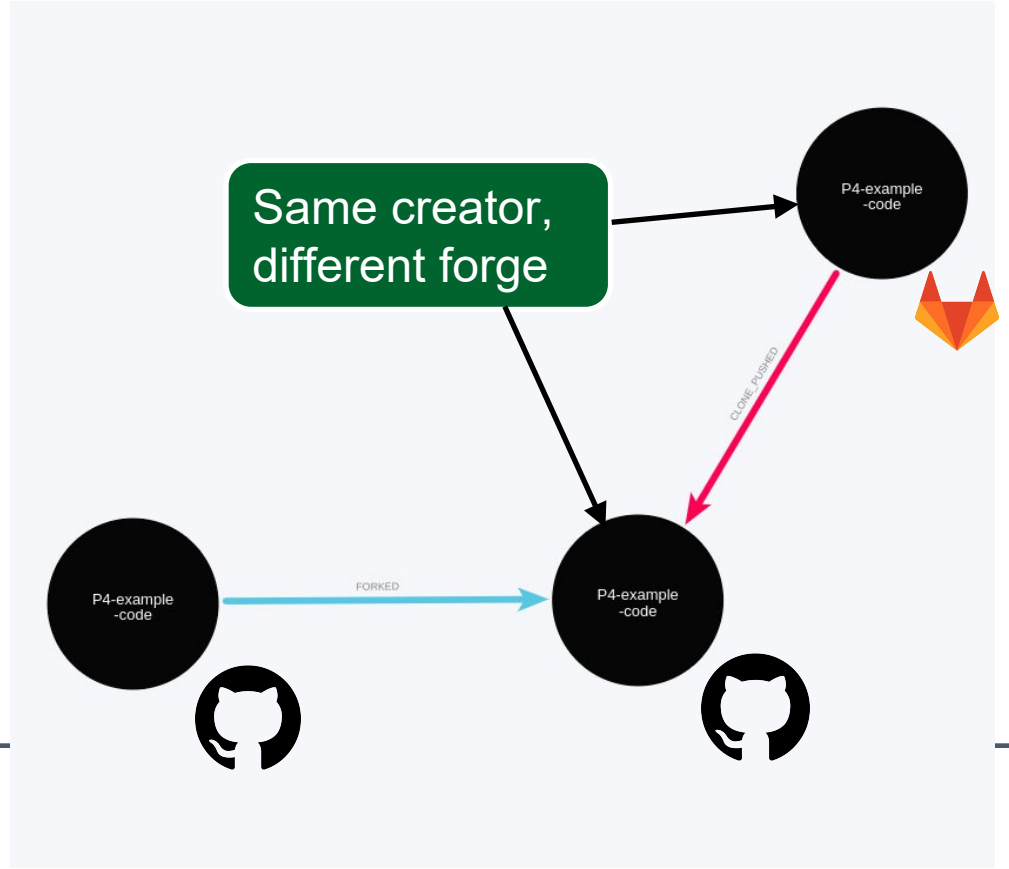
Step 2 - CLONED & PUSHED



How does it work?

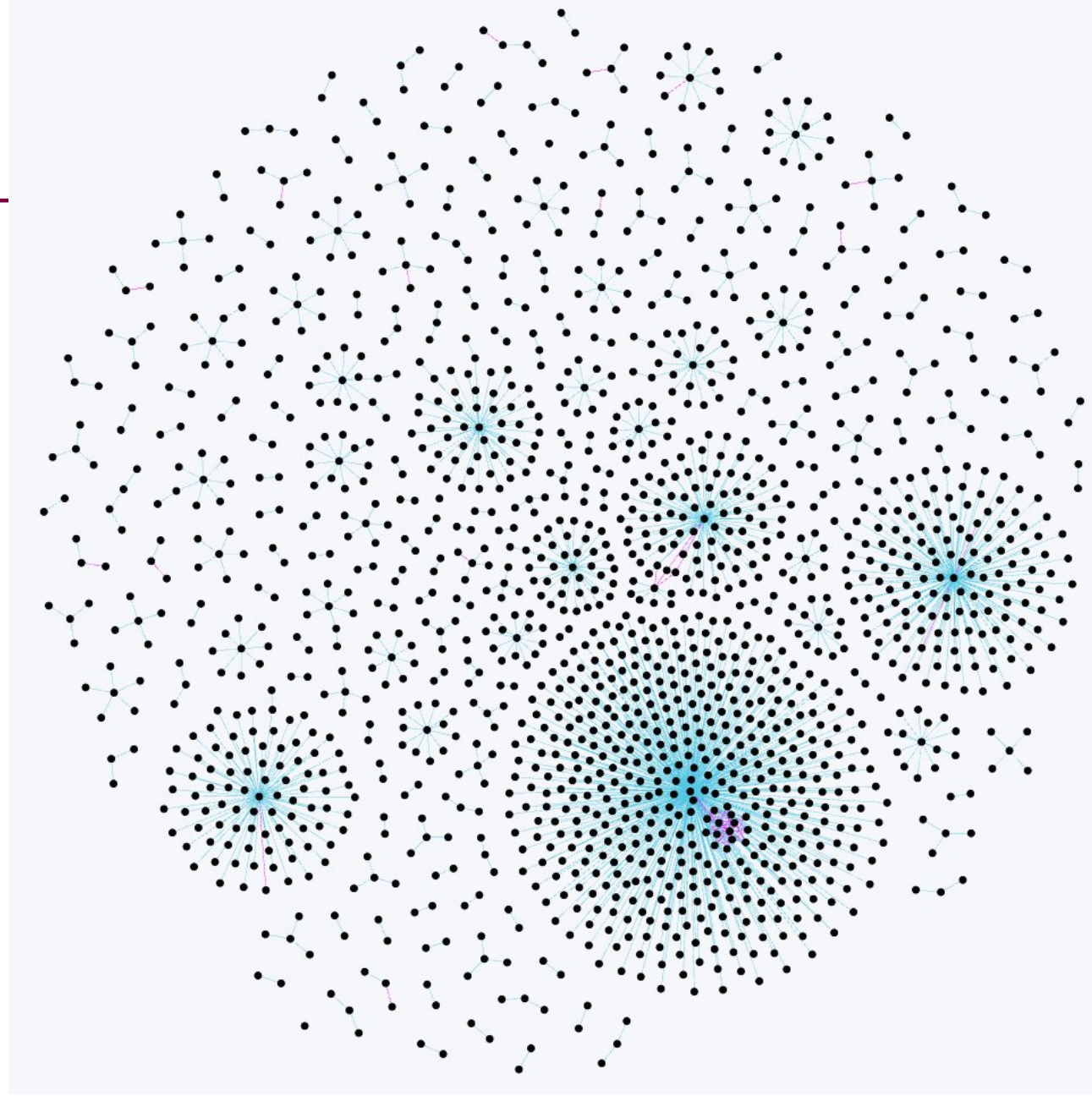
– Look at first commit id of each repository and find ones that match.



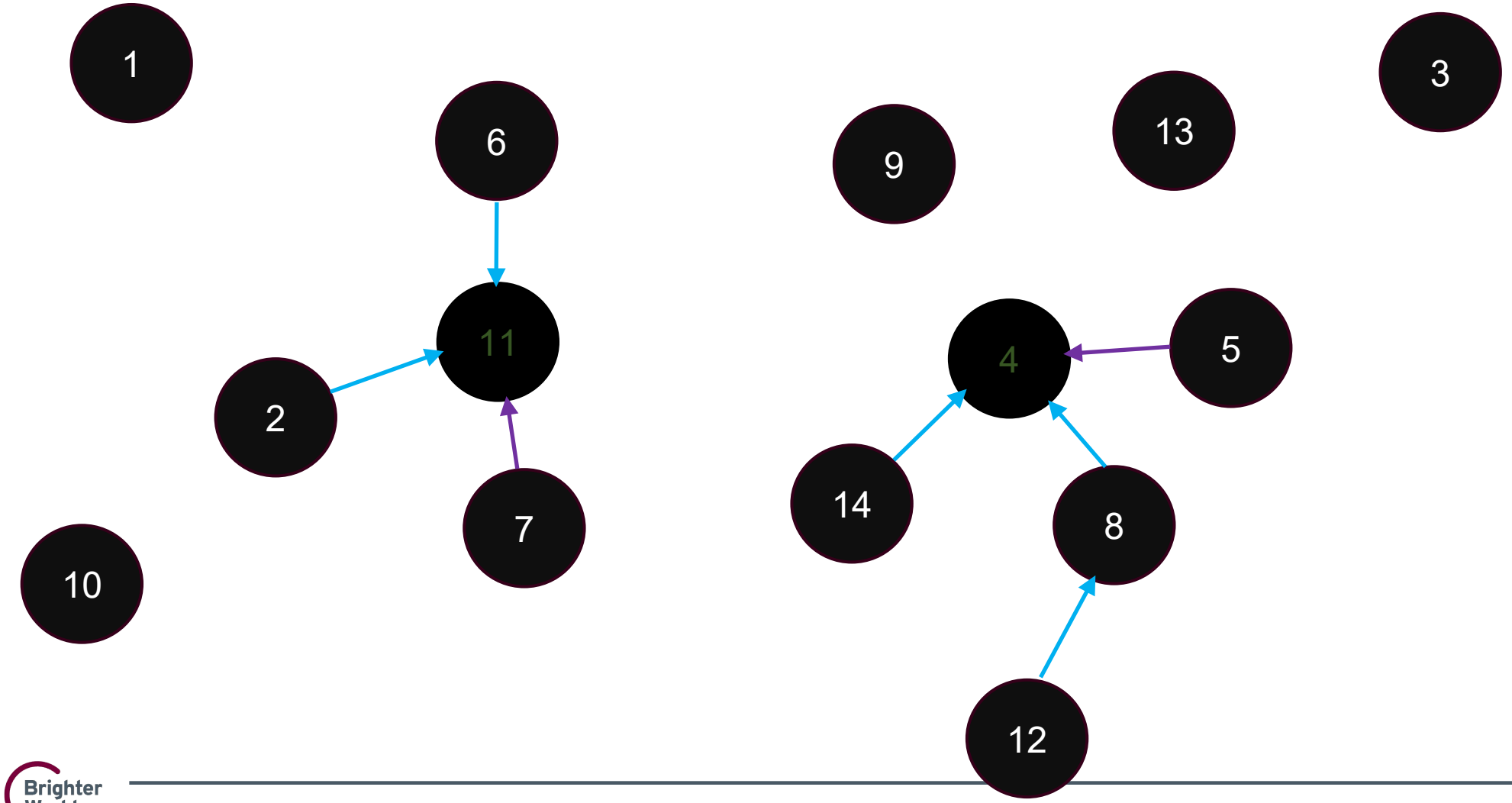


Results

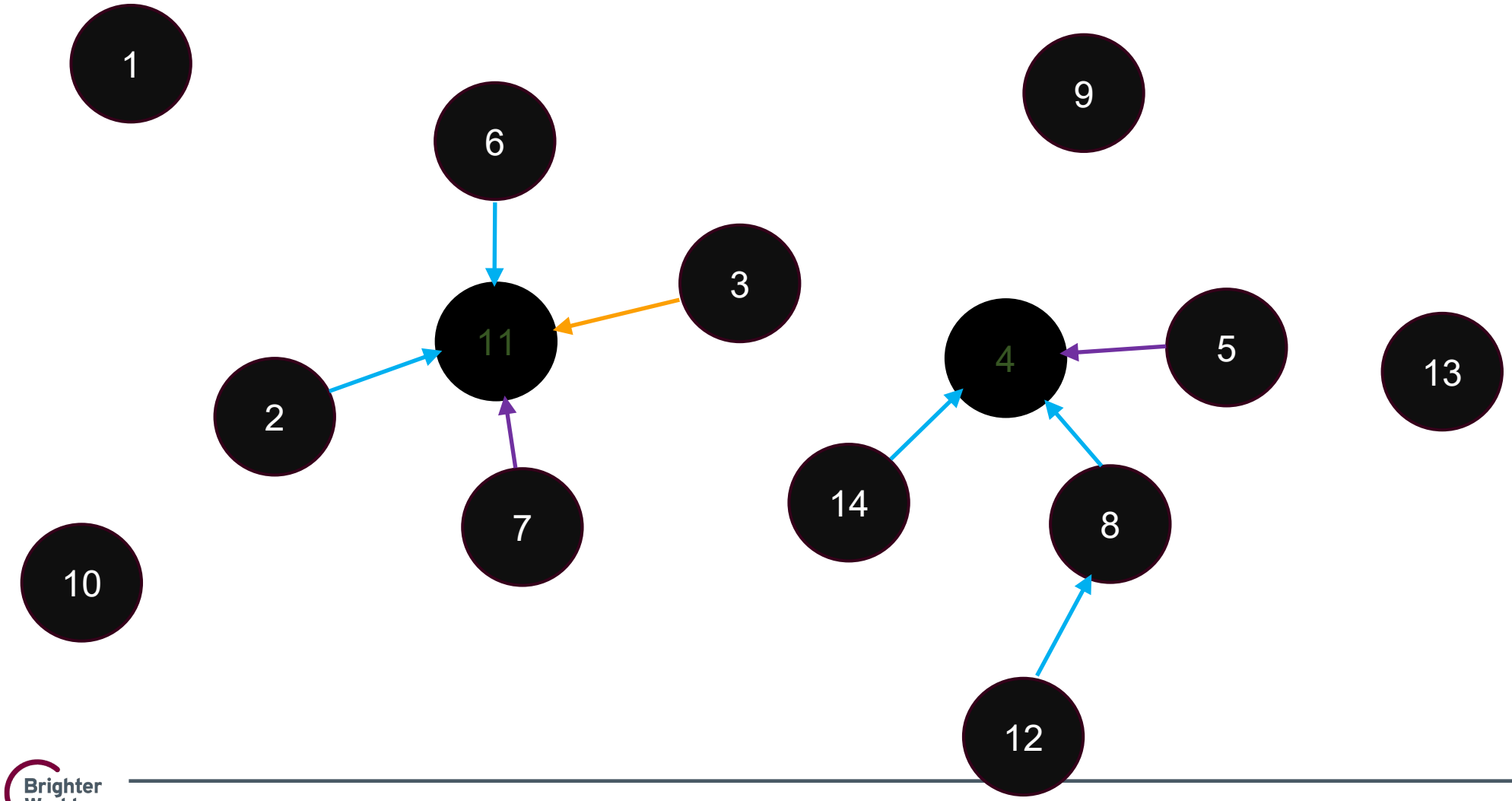
- 1712 relationships (+54)
- 73.95% of repositories connected



Step 3 - DOWNLOADED & PUSHED



Step 3 - DOWNLOADED & PUSHED



Quickly evaluate similarity between 2 projects

$$\text{Similarity} = w_0 \cdot D_{\text{tree}} + w_1 \cdot D_{\text{name}}$$

- w_0: Weight of tree edit distance
- D_{tree}: Normalized tree edit distance.
- w_1: Weight of name edit distance
- D_{name}: Normalized name edit distance

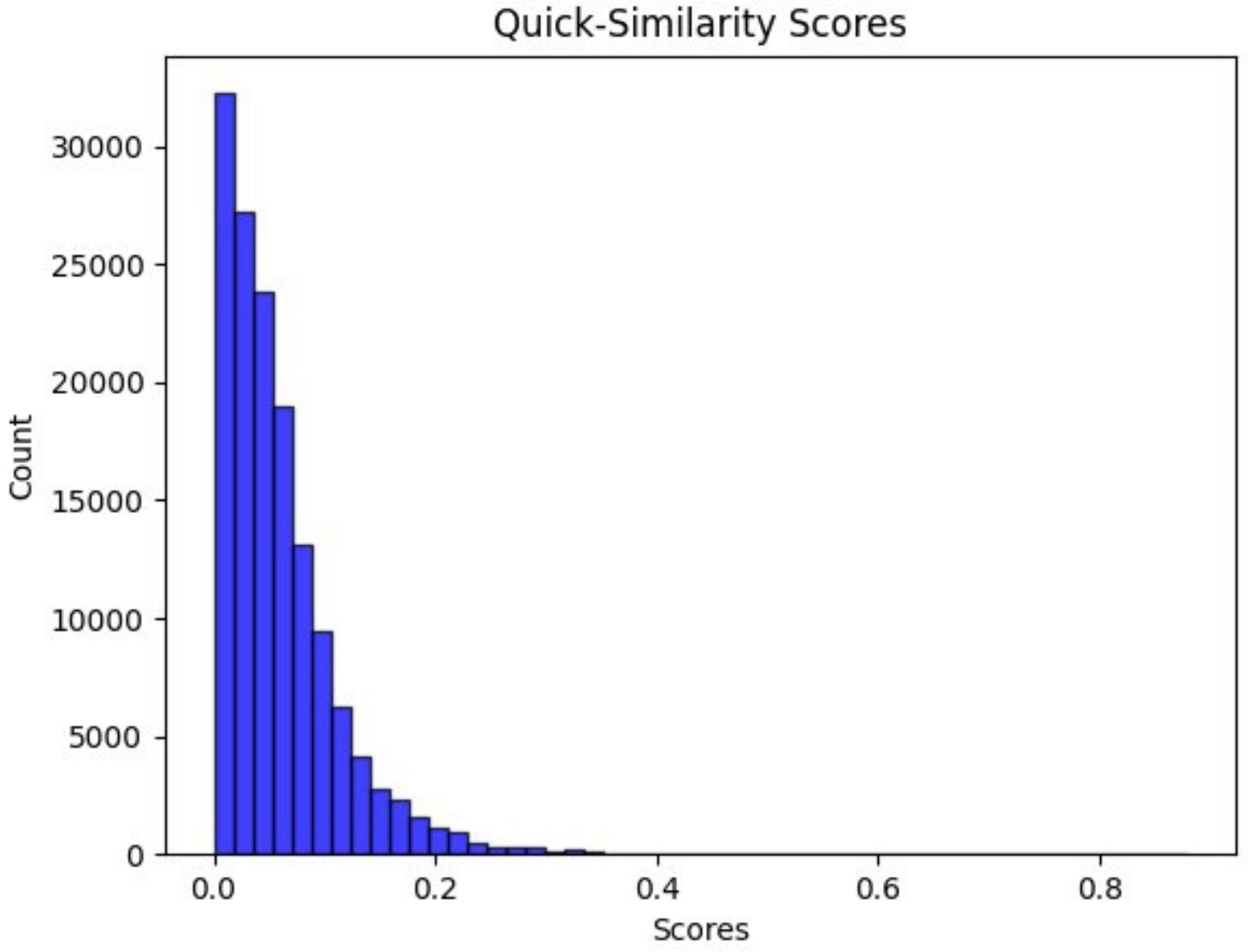
Algorithms chosen:

- APTED[1] for tree edit distance
- Levenshtein for name edit distance

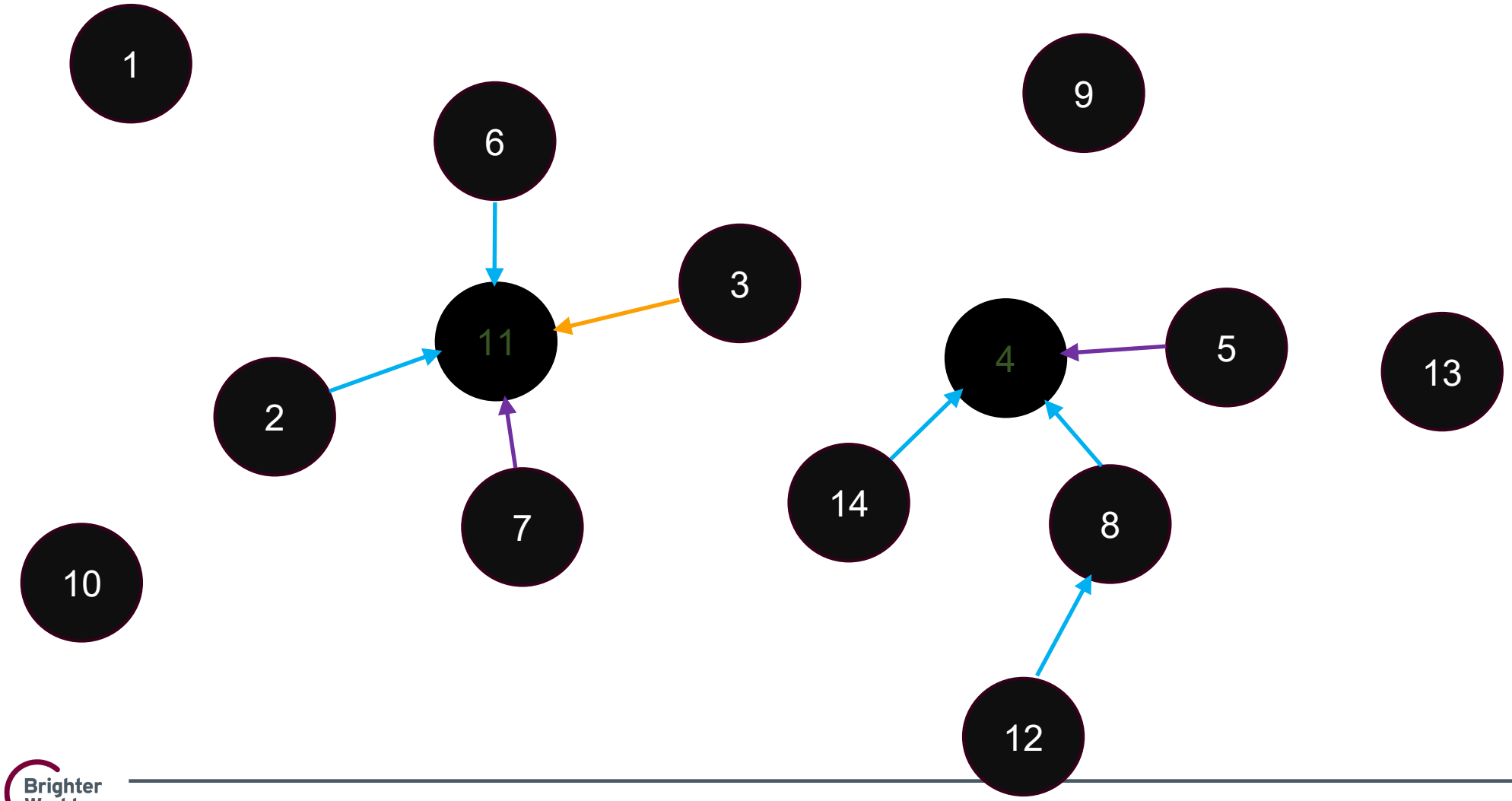
[1]http://tree-edit-distance.dbresearch.uni-salzburg.at/

Results

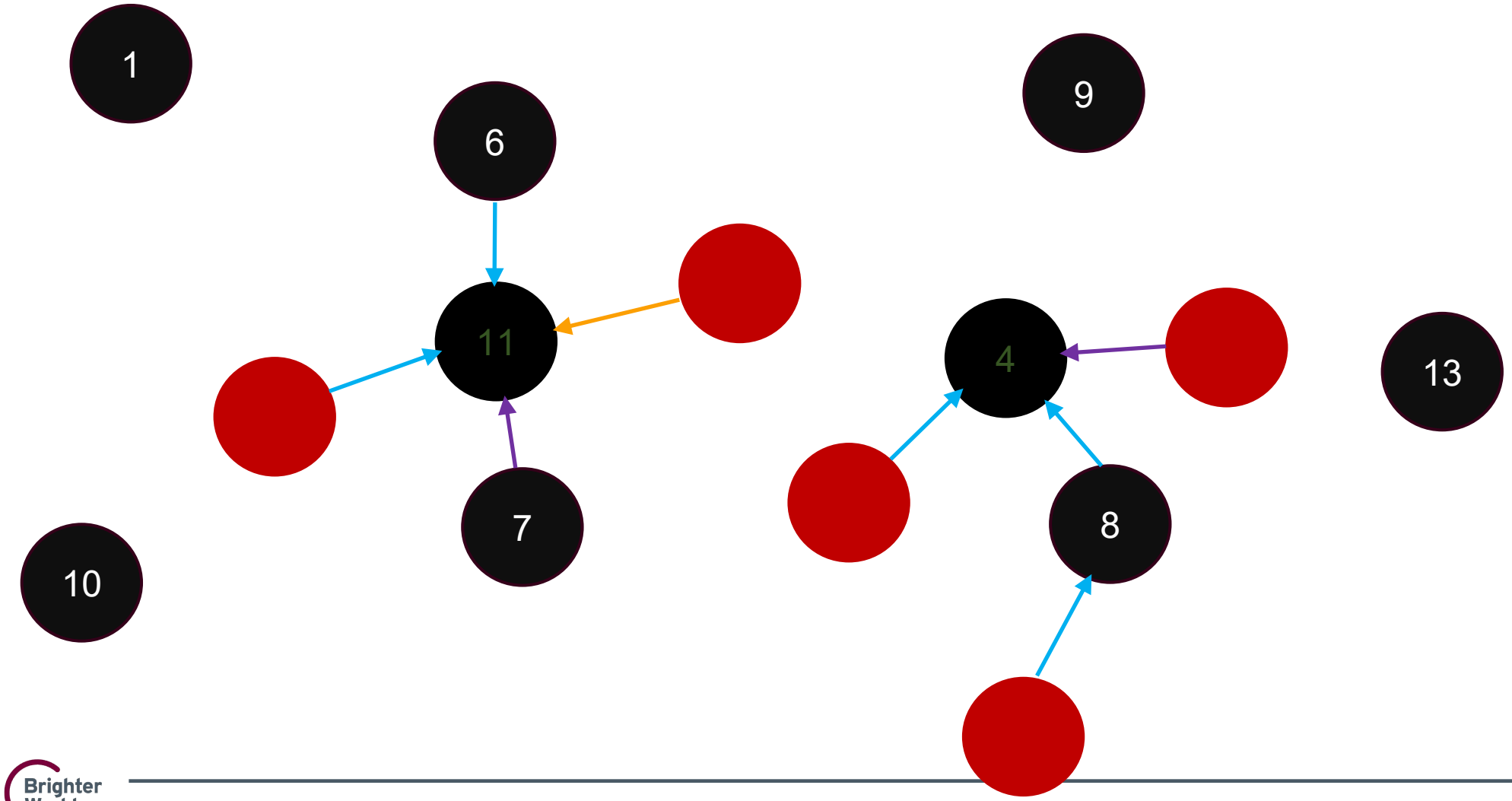
- 140,499 pairs to compare
- 35x decrease in number of computations needed
- ~17 minutes of computation
- 12 pairs with a threshold of a similarity score of >0.7
- 1724 connected nodes (+12)



Step 4 - DELETION



Step 4 - DELETION

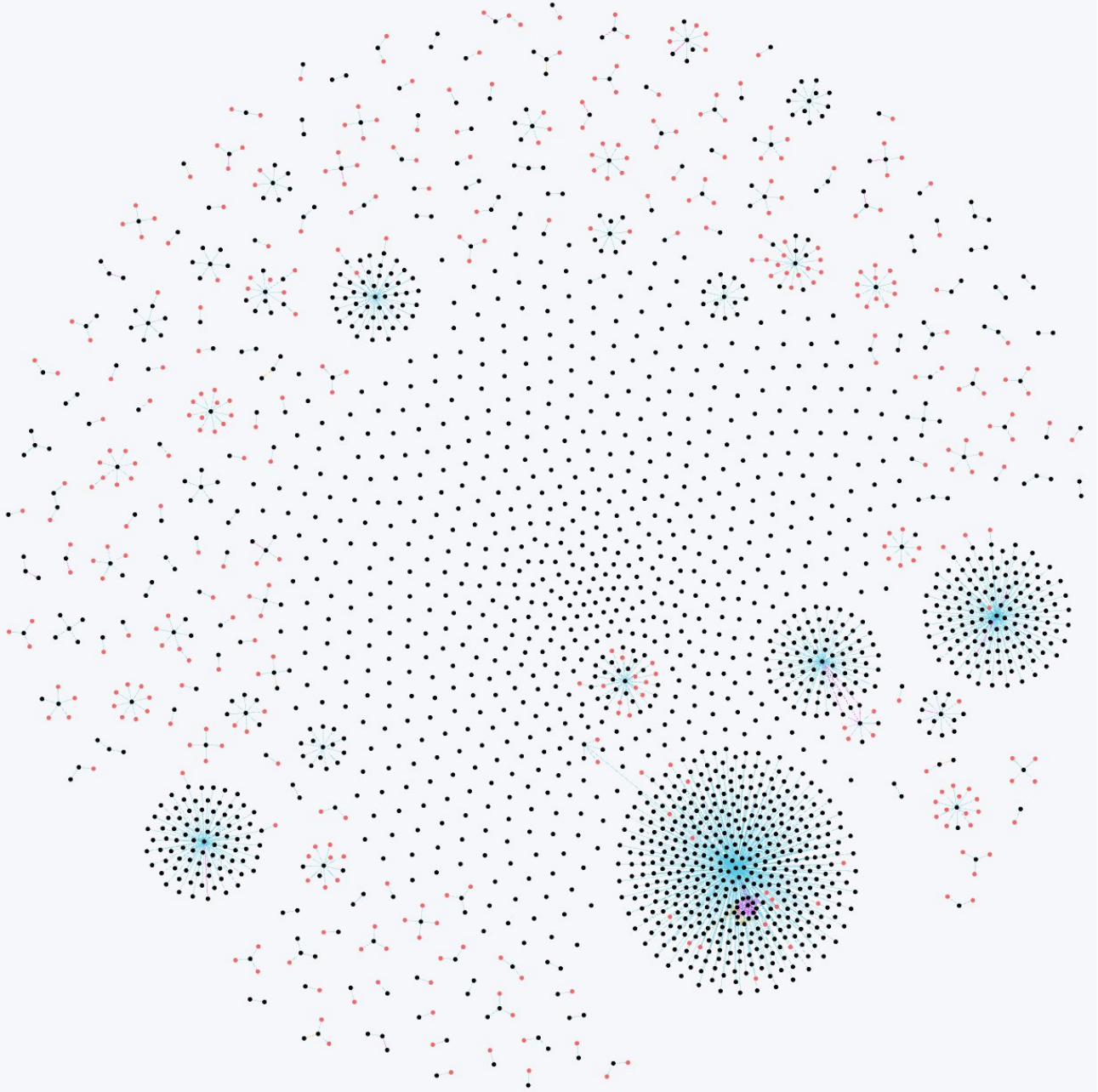


Simple Deletion

- Check if the latest commit of a project is the same as the parent's latest commit
- If it is, no changes have been made
- 100% duplicate code

Result:

- 195 duplicates found



Fully evaluate similarity between 2 projects

$$Similarity = \frac{S_1 + S_2 + \ldots + S_n}{n}$$

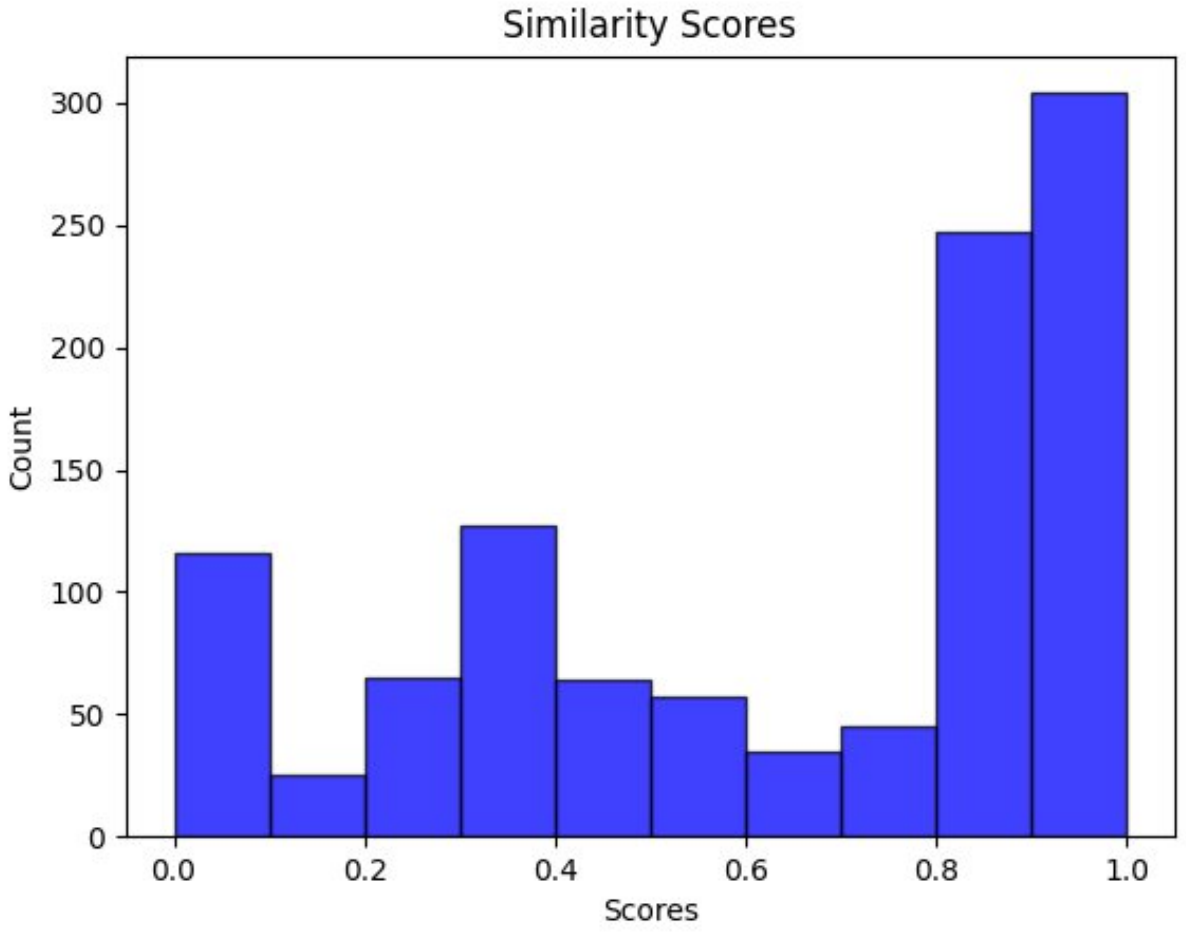
- S_1,S_2,…,S_n: Similarity scores of n files
- n: number of files

Algorithm chosen:

- Python difflib's SequenceMatcher
- Based on Ratcliff/Obershelp's algorithm

Results

- 386 pairs to compare
- ~1h30 of computation
- 286 duplicates (+91) with the threshol set at 0.75
- Out of 2315 (**~12.35%**)



Future Work

Potential Improvements in Pipeline

- Potential improvements of database queries
- Could lead to more duplicates detected

Better full project comparison technique

- All current solutions are language specific
- Test this system with a language specific comparison technique

Better full project comparison technique

- Using state-of-the-art solutions like Repo2Vec
- Deduplication using embeddings

Conclusion

- Solution is far from perfect
- Good results
- 12.35% duplicates

McMaster University

Questions?

REFERENCES

- Allamanis, Miltiadis. “The Adverse Effects of Code Duplication in Machine Learning Models of Code.” In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, 143–53. Athens Greece: ACM, 2019. https://doi.org/10.1145/3359591.3359735.
- Lopes, Cristina V., Petr Maj, Pedro Martins, Vaibhav Saini, Di Yang, Jakub Zitny, Hitesh Sajnani, and Jan Vitek. “DéjàVu: A Map of Code Duplicates on GitHub.” *Proceedings of the ACM on Programming Languages* 1, no. OOPSLA (October 12, 2017): 1–28. https://doi.org/10.1145/3133908.
- Rokon, Md Omar Faruk, Pei Yan, Risul Islam, and Michalis Faloutsos. “Repo2Vec: A Comprehensive Embedding Approach for Determining Repository Similarity.” arXiv, July 11, 2021. http://arxiv.org/abs/2107.05112.
- Sajnani, Hitesh, Vaibhav Saini, Jeffrey Svajlenko, Chanchal K. Roy, and Cristina V. Lopes. “SourcererCC: Scaling Code Clone Detection to Big Code.” In *Proceedings of the 38th International Conference on Software Engineering*, 1157–68, 2016. https://doi.org/10.1145/2884781.2884877.
- Spinellis, Diomidis, Zoe Kotti, and Audris Mockus. “A Dataset for GitHub Repository Deduplication.” In *Proceedings of the 17th International Conference on Mining Software Repositories*, 523–27. Seoul Republic of Korea: ACM, 2020. https://doi.org/10.1145/3379597.3387496.