Flake-Aware Culprit Finding

A Bayesian Approach

Bobby Dorward, Collin Johnston, Eric Nickell, Tim Henderson

2021-April-16

dorward at google dot com, collinj at google dot com, esnickell at google dot com, tadh at google dot com

Problem

Find culprits even when a test is flaky.

Background

CI System

We use a mono-repo, so all commits are submitted to a single branch. The commits are linearly ordered so a test failure can be attributed to a single commit.

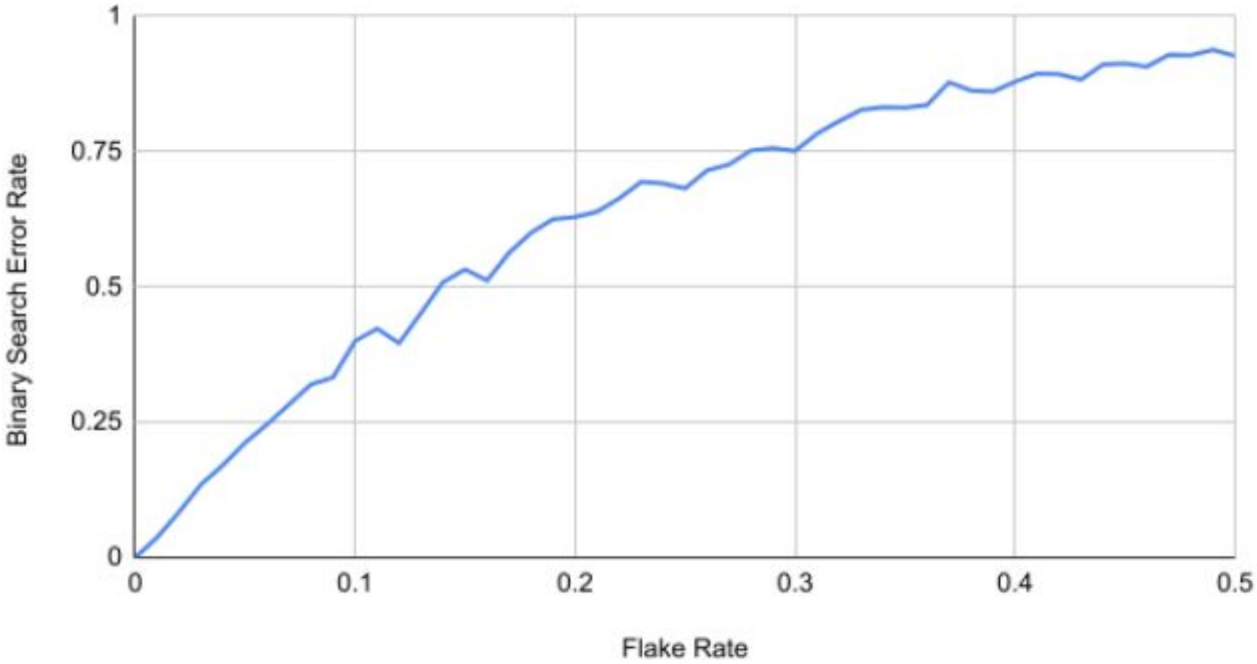
Commits are not tested exhaustively before or after submit. Instead, we rely on culprit finding to pinpoint the exact commit which caused a regression.

Flaky Tests

Test may fail non-deterministically. The non-determinism "at scale" can be from the test infrastructure as well as non-determinism in either the test code, or the code under test.

For our culprit finders, we assume that a "failing test" cannot have flaky passes, but a "passing test" can have flaky failures.

Error rate of naïve binary search vs Flake Rate



Culprit-finding flaky targets: Existing solutions

Deflaked Binary Search

One solution is to run binary search, and deflake the result at each pivot point. Instead of running the test once and recursing on the left/right half based on the result, we could run the test N times, and recurse based on the aggregated result: Passed if any result passed, Failed if all results failed.

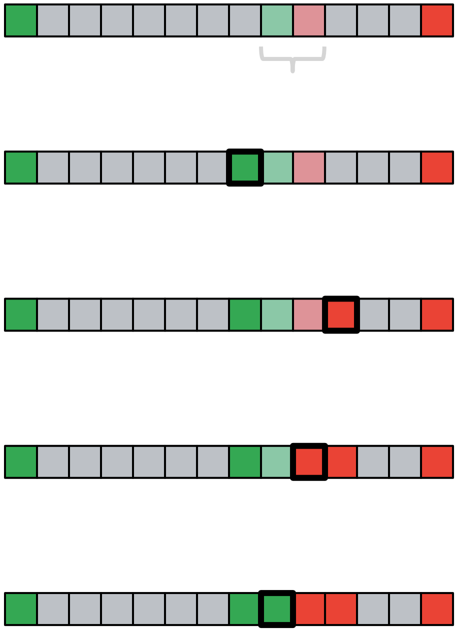
Flake-aware Culprit-finding

FACF Key Idea

Track probability that each commit is the culprit and use Bayes' rule to update results

Naïve binary search

Naïve binary search



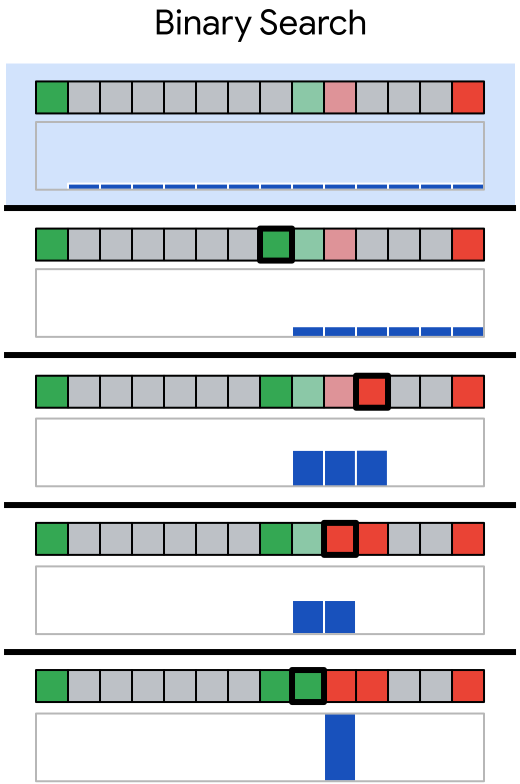
Binary Search

While we normally view the progress of CF as bounding the possible locations of the pass-to-fail transition, we could alternatively view it as shifting a probability distribution.

Legend

- 🟥 Saw FAIL
- 🟩 Saw PASS
- ▨ Culprit Commit
- ▧ Last pass before culprit
- ⬜ Commit w/ no results

Naïve binary search ... as probability redistribution


Binary Search

Start with a uniform distribution.

Legend

🟥 Saw FAIL

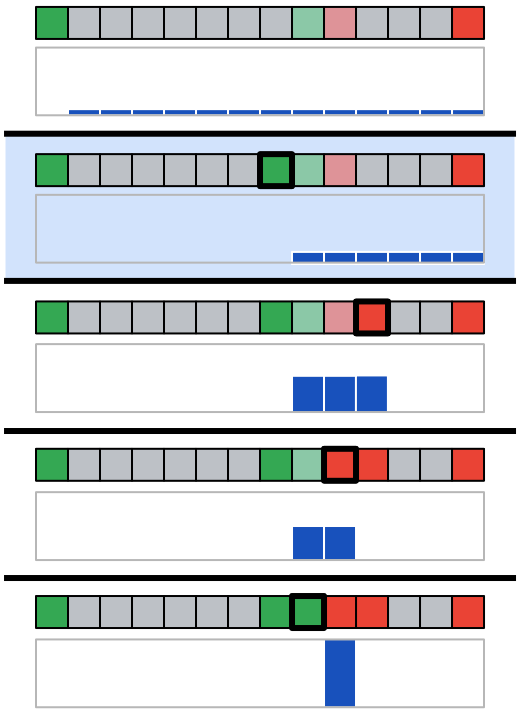
🟩 Saw PASS

🟥 Culprit Commit

🟩 Last pass before culprit

⬜ Commit w/ no results

Naïve binary search ... as probability redistribution



Binary Search

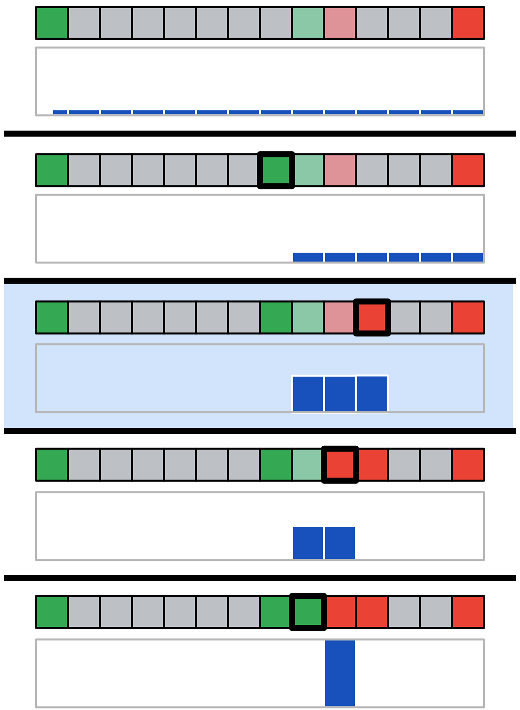
Start with a uniform distribution.

On seeing a passing result, transfer all the left-hand probability to the right.

Legend

- 🟥 Saw FAIL
- 🟩 Saw PASS
- ▨ Culprit Commit
- ▧ Last pass before culprit
- ⬜ Commit w/ no results

Naïve binary search ... as probability redistribution



Binary Search

Start with a uniform distribution.

On seeing a passing result, transfer all the left-hand probability to the right.

On seeing a failing result, transfer all the right-hand probability to the left.

Legend

🟥 Saw FAIL

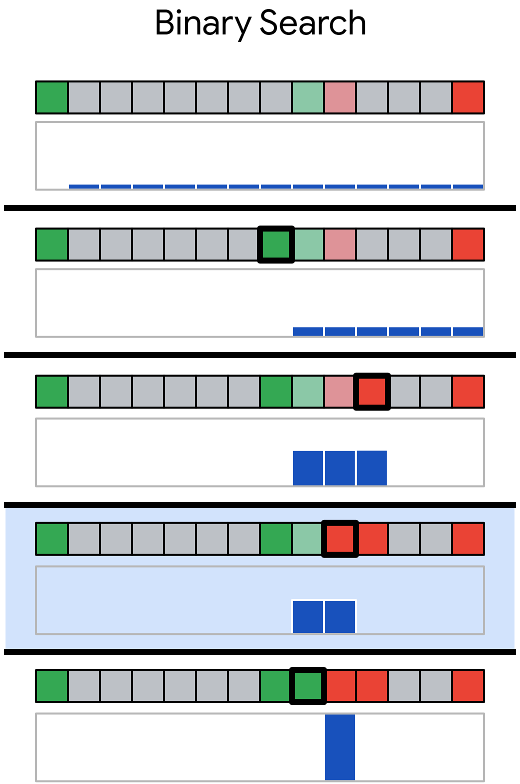
🟩 Saw PASS

▨ Culprit Commit

▧ Last pass before culprit

⬜ Commit w/ no results

Naïve binary search ... as probability redistribution


Binary Search

Start with a uniform distribution.

On seeing a passing result, transfer all the left-hand probability to the right.

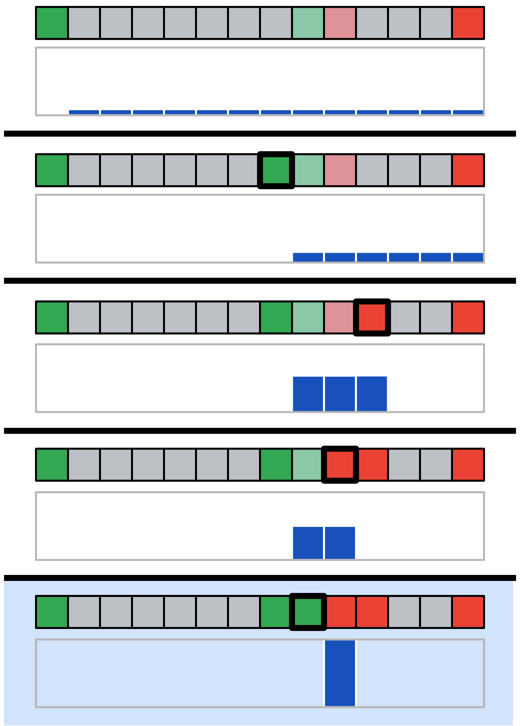
On seeing a failing result, transfer all the right-hand probability to the left.

Legend

- 🟥 Saw FAIL
- 🟩 Saw PASS
- 🟥 Culprit Commit
- 🟩 Last pass before culprit
- ⬜ Commit w/ no results

Naïve binary search ... as probability redistribution

Binary Search



Start with a uniform distribution.

On seeing a passing result, transfer all the left-hand probability to the right.

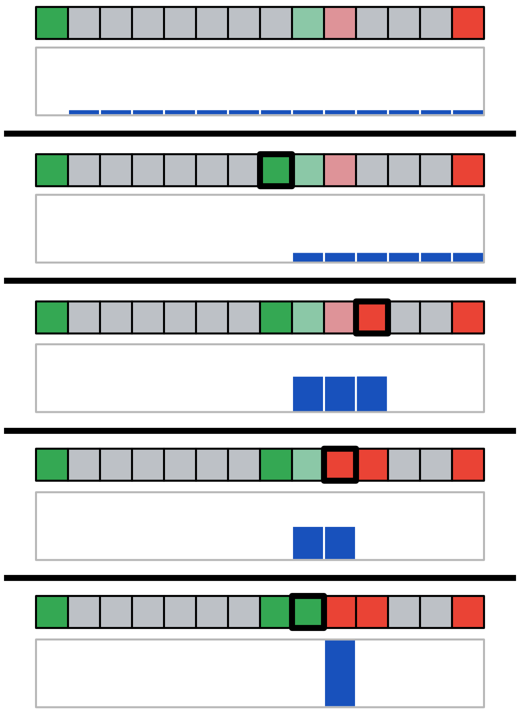
On seeing a failing result, transfer all the right-hand probability to the left.

Legend

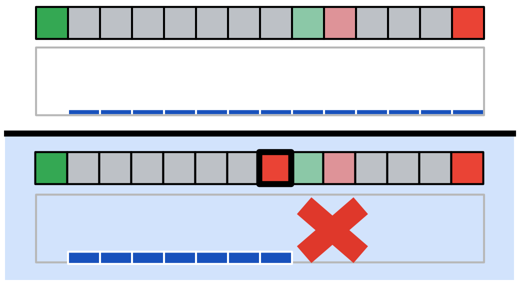
- 🟥 Saw FAIL
- 🟩 Saw PASS
- 🟥 Culprit Commit
- 🟩 Last pass before culprit
- ⬜ Commit w/ no results

Naïve binary search ... and flaky failures

Binary Search



A flaky target...



If we move all the right-hand probability to the left on a flake, we can never find the culprit.

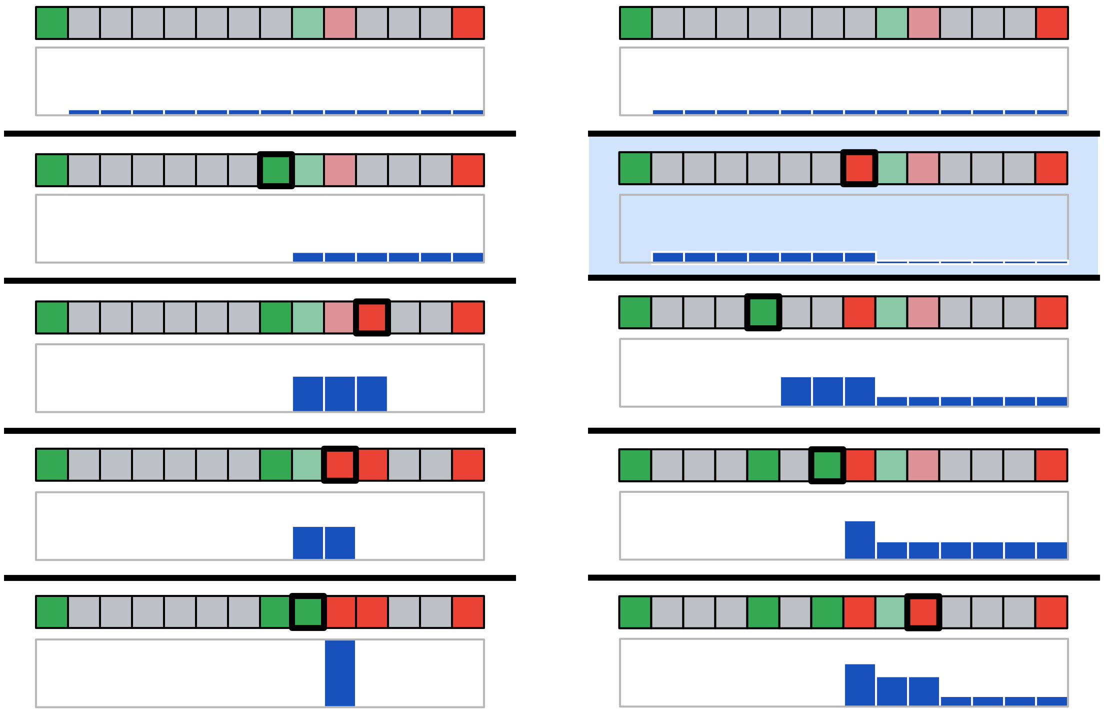
(If the likelihood for a commit ever drops to zero, it can never recover.)

Legend

- 🟥 Saw FAIL
- 🟩 Saw PASS
- ▨ Culprit Commit
- ▨ Last pass before culprit
- □ Commit w/ no results

Flake-aware probability distribution

Flake-aware Culprit Finder



Binary Search

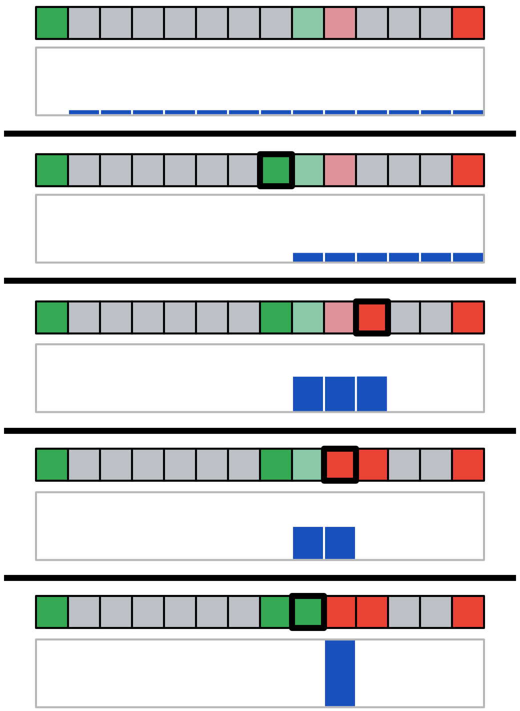
A flaky target...

If we move all the right-hand probability to the left on a flake, we can never find the culprit.

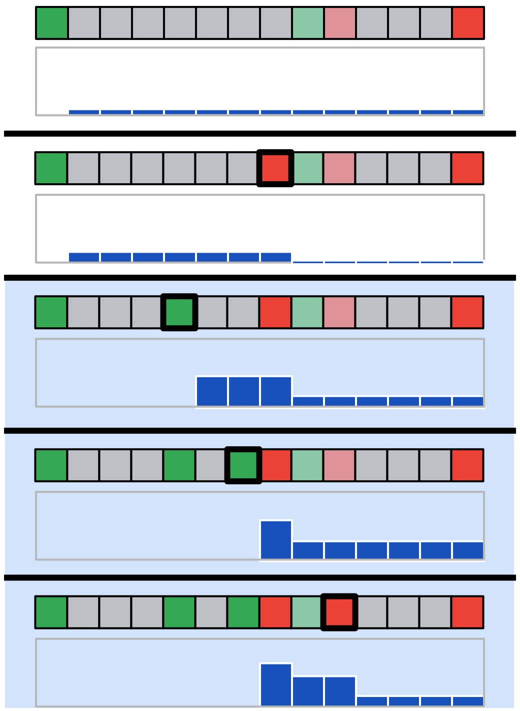
So we only shift *some* of the probability. What's left behind is the probability that the failure was a flake.

Flake-aware Culprit Finder

Binary Search



A flaky target...

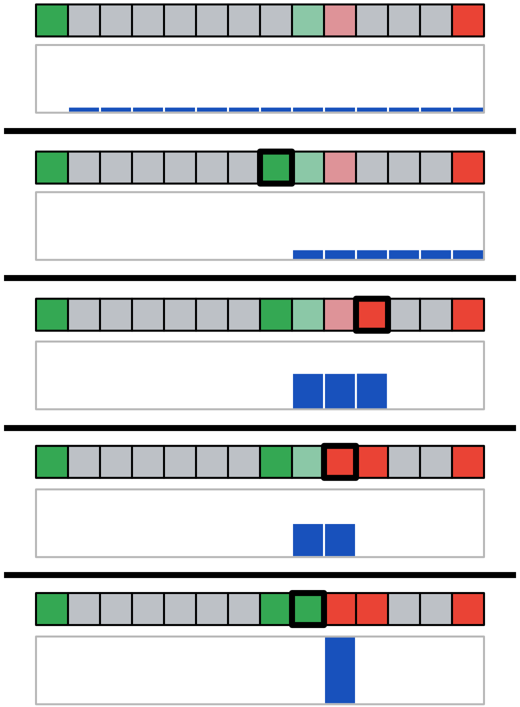


The probabilities will self-correct as long as there aren't too many flakes.

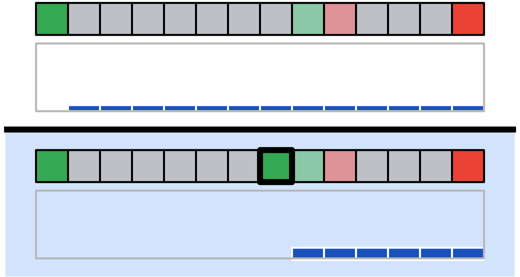
Flake-aware when no flakes

Flake-aware Culprit Finder

Binary Search

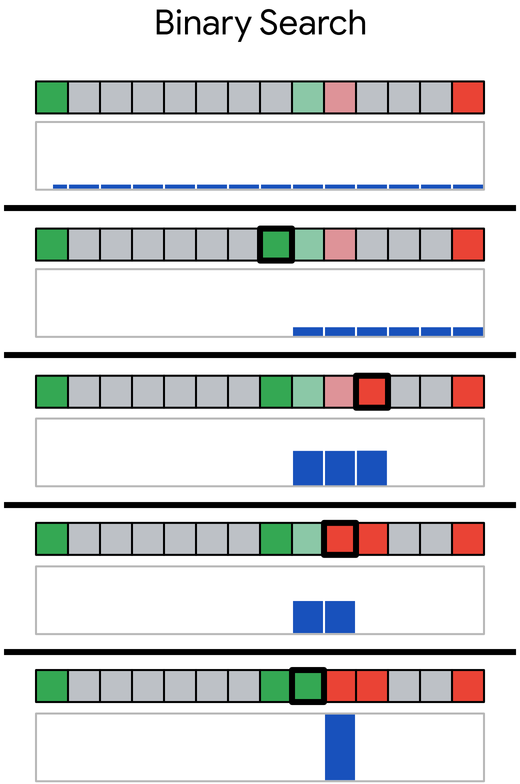


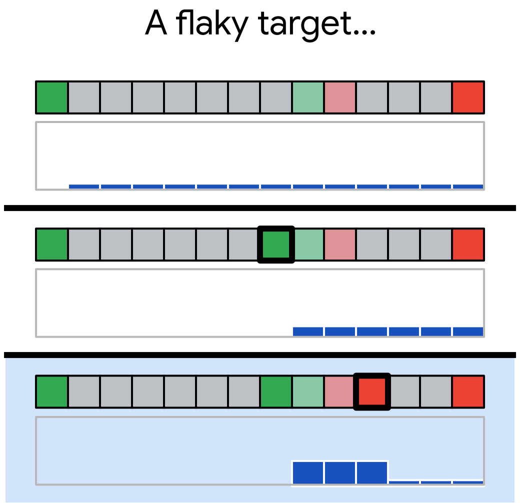
A flaky target...



Since we assume are no false PASSes, we can still move *all* left-hand probability to the right on a pass.

Flake-aware Culprit Finder

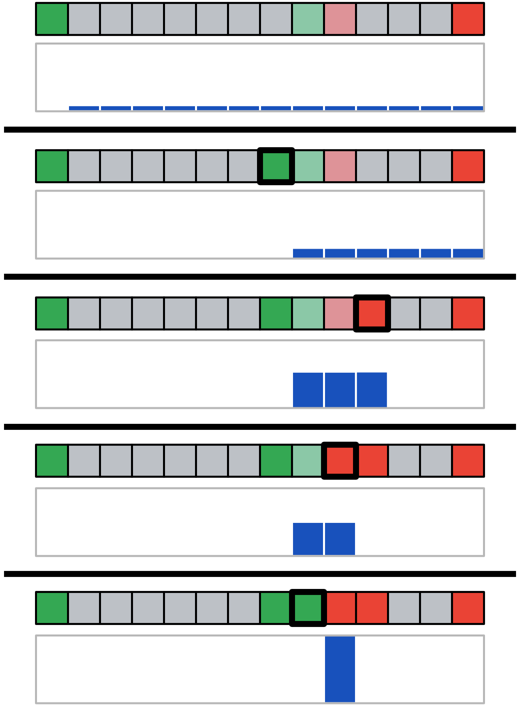




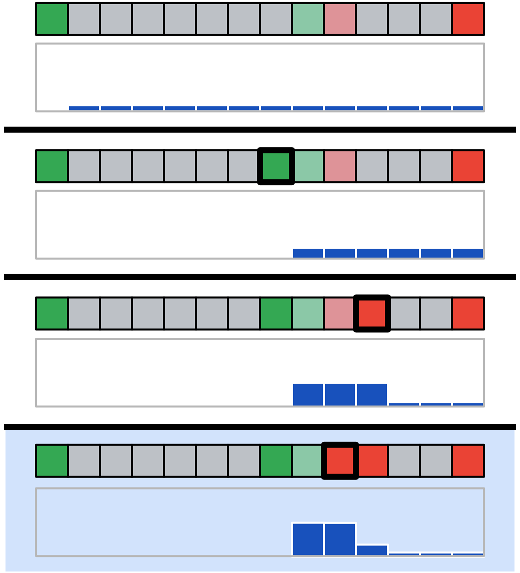
Shift some probability to commits left on FAIL.

Flake-aware Culprit Finder

Binary Search



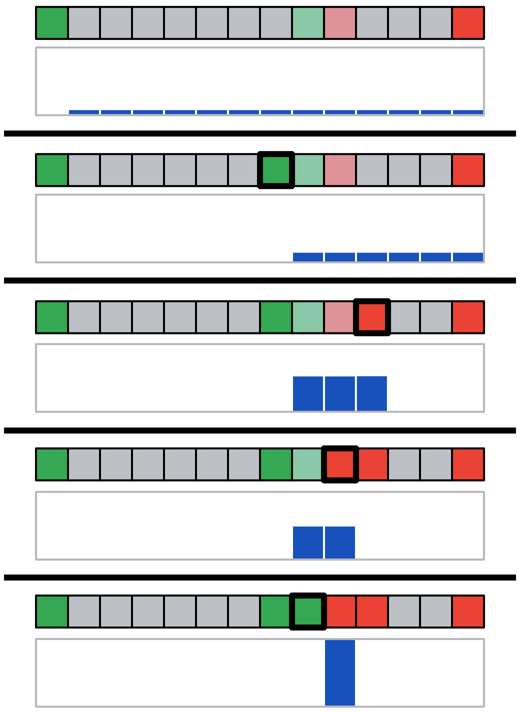
A flaky target...



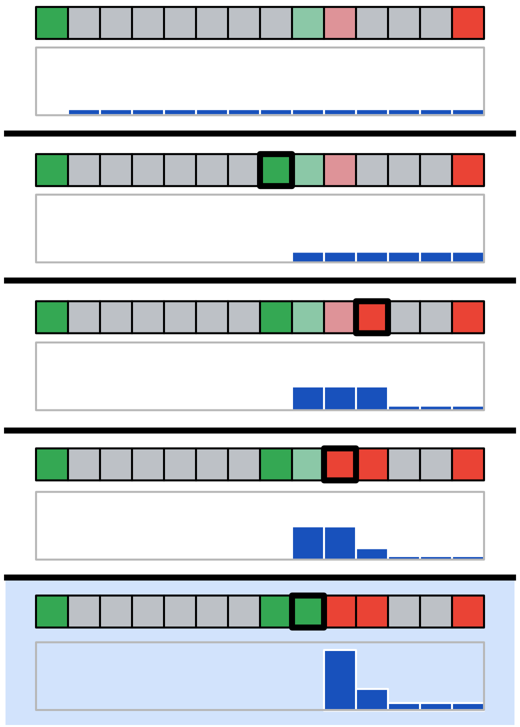
Shift more probability left on FAIL.

Flake-aware Culprit Finder

Binary Search



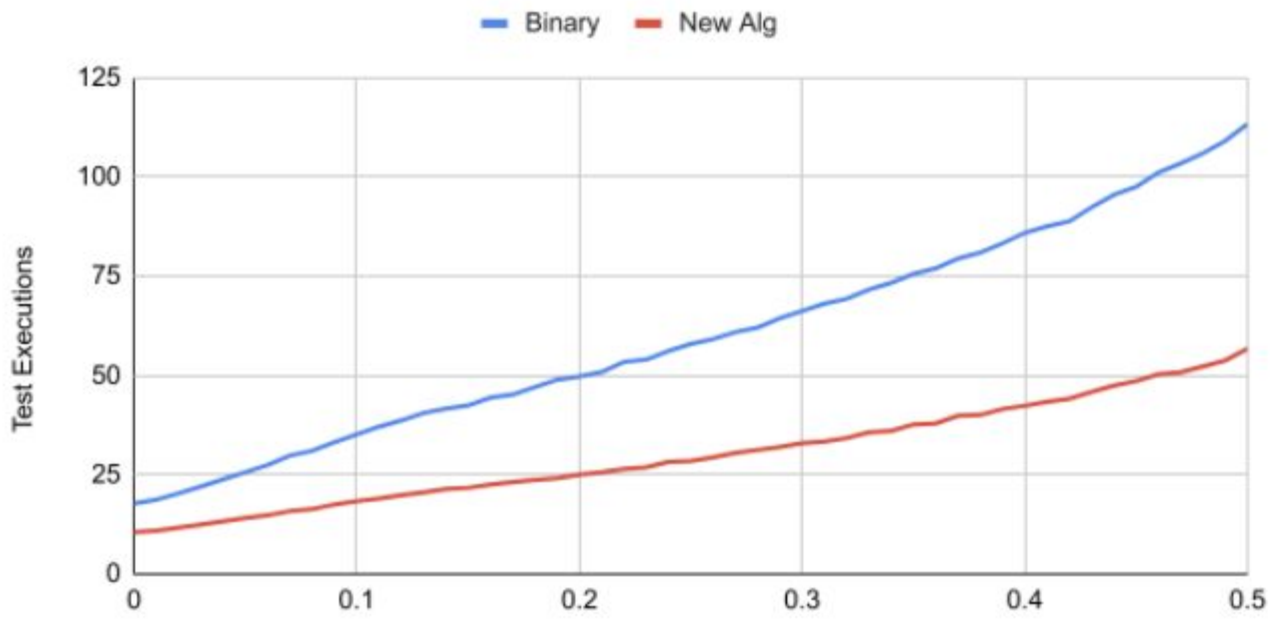
A flaky target...



Continue until a single transition is the culprit, with sufficiently high likelihood.

Features of flake-aware culprit-finding

Deflaked Binary search vs Flake Aware Culprit Finder



At each flakiness, both algos achieve 99.9% correctness.

Flake-aware Culprit Finder

- Can find culprits even for flaky targets in *O(log N)* time & resources.
- Splits culprit range to minimize expected number of iterations.
- Number of test executions auto-scales to desired correctness.
- Algorithm becomes binary search as flakiness drops to zero.

Flake-aware Culprit Finder: Notes, Caveats

Prior Distribution

FACF can start with a prior distribution, coming from heuristics, an ML model, or another culprit finder which produces a probability distribution.

Flaky Trigger

The initial failing edge could have been a flake. This can be handled by including an extra suspect commit representing "no culprit", on the right hand end. This is initially set to an estimate based on an initial estimate.

Build Cost

Although FACF results in less test executions, deflaked binary search might be cheaper in situations where the cost of building the test heavily outweighs the cost of running the test, since FACF runs at more commits.

Questions?