A SYSTEMATIC STUDY OF AUTOMATED PROGRAM REPAIR: FIXING 55 OUT OF 105 BUGS FOR $8 EACH

Claire Le Goues

Michael Dewey-Vogt

Stephanie Forrest

Westley Weimer

UNIVERSITY OF VIRGINIA 1819

THE UNIVERSITY of NEW MEXICO

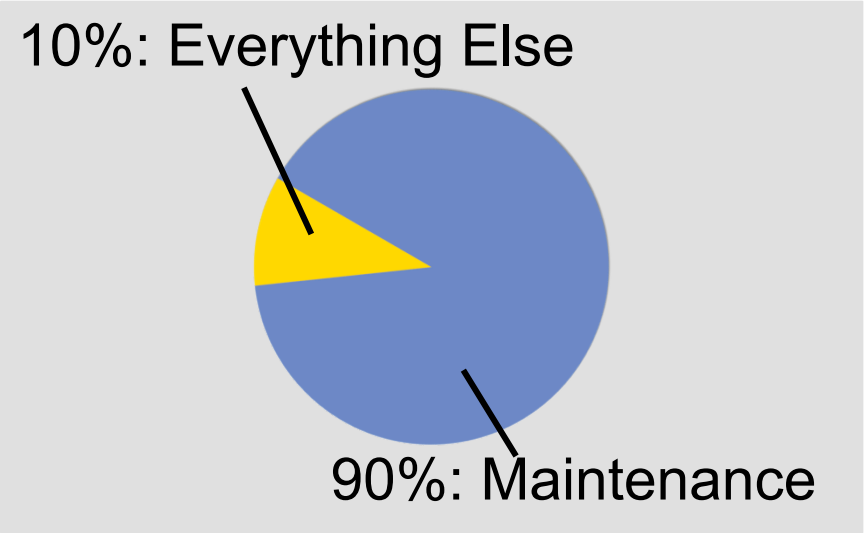
"Everyday, almost 300 bugs appear […] far too many for only the Mozilla programmers to handle."

– *Mozilla Developer, 2005*

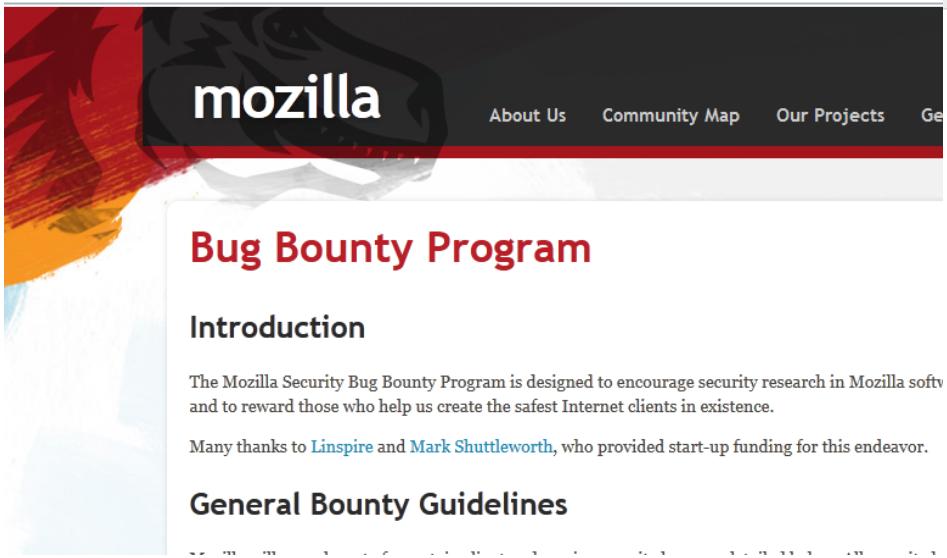
Annual cost of software errors in the US: $59.5 billion (0.6% of GDP).

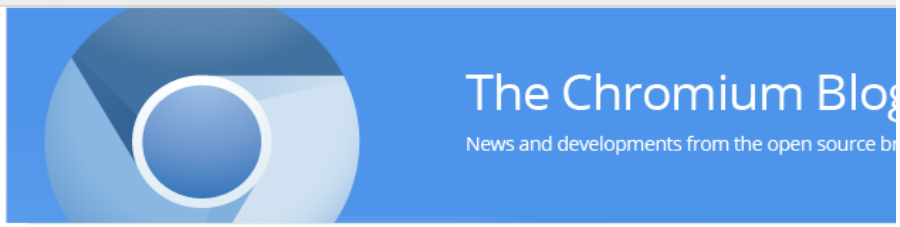
PROBLEM: BUGGY SOFTWARE

Average time to fix a security-critical error: 28 days.



HOW BAD IS IT?





Encouraging More Chromium Security Research

Thursday, January 28, 2010
Labels: googlechrome, security

In designing Chromium, we've been working hard to make the browser as secure as possible. We've made strong improvements with the integrated sandboxing and our up-to-date user base. We're always looking to stay on top of the latest browser security features. We've also worked closely with the broader security community to get independent scrutiny and to quickly fix bugs that have been reported.

Some of the most interesting security bugs we've fixed have been reported by researchers external to the Chromium project. For example, this same origin policy bypass from Isaac Dawson or this v8 engine bug found by the Mozilla Security Team. Thanks to the collaborative efforts of these people and others, Chromium security is stronger and our users are safer.

Today, we are introducing an experimental new incentive for external researchers to participate. We will be rewarding select interesting and original vulnerabilities reported to us by the security research community. For existing contributors to Chromium security — who would likely continue to contribute regardless — this may be seen as a token of our appreciation. In addition, we are hoping that the introduction of this program will encourage new individuals to participate in Chromium security. The more people involved in scrutinizing Chromium's code and behavior, the more secure our millions of users will be.

Such a concept is not new; we'd like to give serious kudos to the folks at Mozilla for their long-running and successful vulnerability reward program.

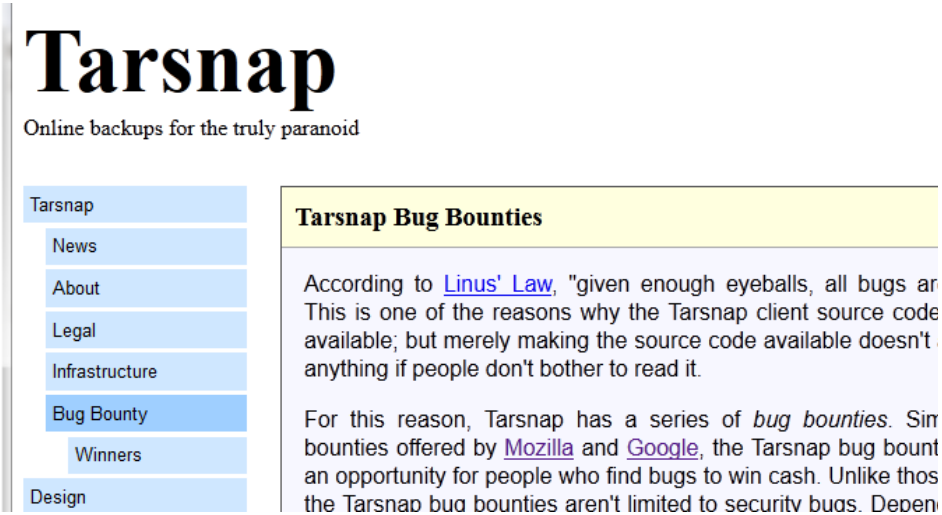
Any valid security bug filed through the Chromium bug tracker (under the template "Security Bug") will qualify for consideration. As this is an experimental program, here are some guidelines in the form of questions and answers:

Q) What reward might I get?
A) As per Mozilla, our base reward for eligible bugs is $500. If the panel finds a particular bug particularly severe or particularly clever, we envisage rewards of $1337. The panel may also decide a single report actually constitutes multiple bugs. As a consumer of the Chromium open source project, Google will be sponsoring the rewards.

Q) What bugs are eligible?
A) Any security bug may be considered. We will typically focus on High and Critical impact bugs, but any clever vulnerability at any severity might get a reward. Obviously, your bug won't be eligible if you worked on the code or review in the area in question.

Q) How do I find out my bug was eligible?
A) You will see a provisional comment to that effect in the bug entry once we have triaged the bug.

Q) What if someone else also found the same bug?
A) Only the first report of a given issue that we were previously unaware of is eligible. In the event of a duplicate submission, the earliest filed bug report in the bug tracker is considered the first report.



endangered the security of Mozilla's end users.

If two or more people report the bug together the reward will be divided among

Client Reward Guidelines

The bounty for valid critical client security bugs will be $3000 (US) cash reward
The bounty will be awarded for sg:critical and sg:high severity security bugs tha
criteria:

- Security bug is present in the most recent supported, beta or release cand
  Thunderbird, Firefox Mobile, or in Mozilla services which could compron
  products, as released by Mozilla Corporation or Mozilla Messaging.
- Security bugs in or caused by additional 3rd-party software (e.g. plugins,
  from the Bug Bounty program.

More information about this program can be found in the Client Security Bug B

Web Application and Services Reward Guideli

The bounty for valid web applications or services related securi
$500 (US) for high severity and, in some cases, may pay up t $3000 (US) for e
vulnerabilities. We will also include a Mozilla T-shirt. The bounty will be awarde
ws:high security bugs that meet the following criteria:

- Security bug is present in the web properties outlined in the Web Applica
- Security bug is on the list of sites which part of the bounty. See the eligibl
  Application Security Bounty FAQ for the list of sites which is included un

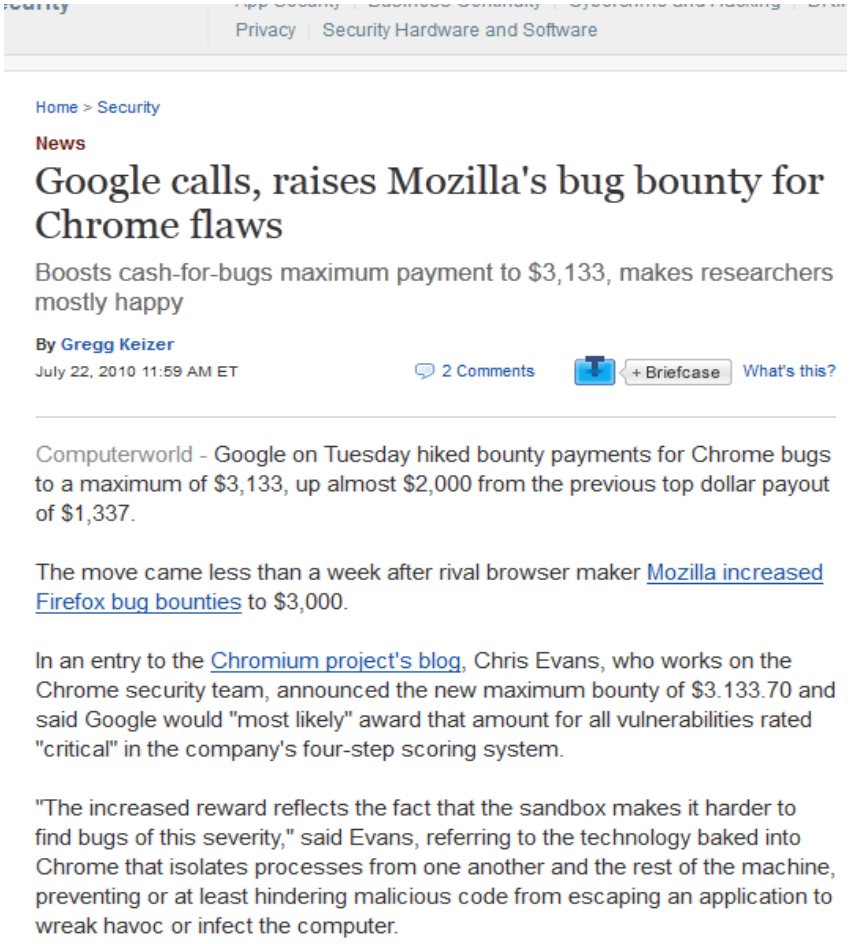
More information about this program can be found in the Web Application Security Bounty FAQ.

an opportunity for people who find bugs to win cash. Unlike those bounti
the Tarsnap bug bounties aren't limited to security bugs. Depending on t
type of bug and when it is reported, different bounties will be awarded:

Bounty value	Pre-release bounty value	Type of bug
$1000	$2000	A bug which allows someone intercepting Tarsna traffic to decrypt Tarsnap users' data.
$500	$1000	A bug which allows the Tarsnap service to decry Tarsnap users' data.
$500	$1000	A bug which causes data corruption or loss.
$100	$200	A bug which causes Tarsnap to crash (without corrupting data or losing any data other than an archive currently being written).
$50	$100	Any other non-harmless bugs in Tarsnap.
$20	$40	Build breakage on a platform where a previous Tarsnap release worked.
$10	$20	"Harmless" bugs, e.g., cosmetic errors in Tarsna output or mistakes in source code comments.
$1	$2	Cosmetic errors in the Tarsnap source code or website, e.g., typos in website text or source cod comments. Style errors in Tarsnap code qualify here, but usually not style errors in upstream cod (e.g., libarchive).

The pre-release bounty value will be awarded for bugs reported in t
interval between when a new Tarsnap release is sent to the tarsna

...REALLY?



Privacy | Security Hardware and Software

Home > Security

News

Google calls, raises Mozilla's bug bounty for Chrome flaws

Boosts cash-for-bugs maximum payment to $3,133, makes researchers mostly happy

By Gregg Keizer

July 22, 2010 11:59 AM ET

2 Comments + Briefcase What's this?

Computerworld - Google on Tuesday hiked bounty payments for Chrome bugs to a maximum of $3,133, up almost $2,000 from the previous top dollar payout of $1,337.

The move came less than a week after rival browser maker Mozilla increased Firefox bug bounties to $3,000.

In an entry to the Chromium project's blog, Chris Evans, who works on the Chrome security team, announced the new maximum bounty of $3.133.70 and said Google would "most likely" award that amount for all vulnerabilities rated "critical" in the company's four-step scoring system.

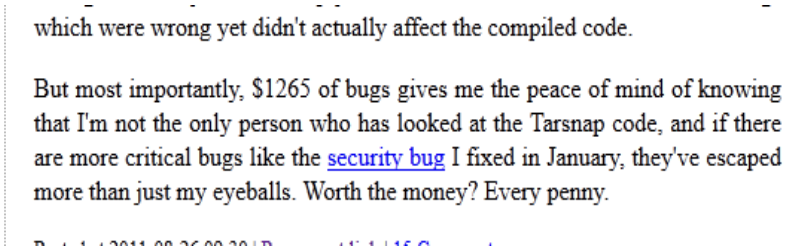
"The increased reward reflects the fact that the sandbox makes it harder to find bugs of this severity," said Evans, referring to the technology baked into Chrome that isolates processes from one another and the rest of the machine, preventing or at least hindering malicious code from escaping an application to wreak havoc or infect the computer.

Tarsnap:

125 spelling/style
63 harmless
11 minor
+ 1 major

75/200 = 38% TP rate

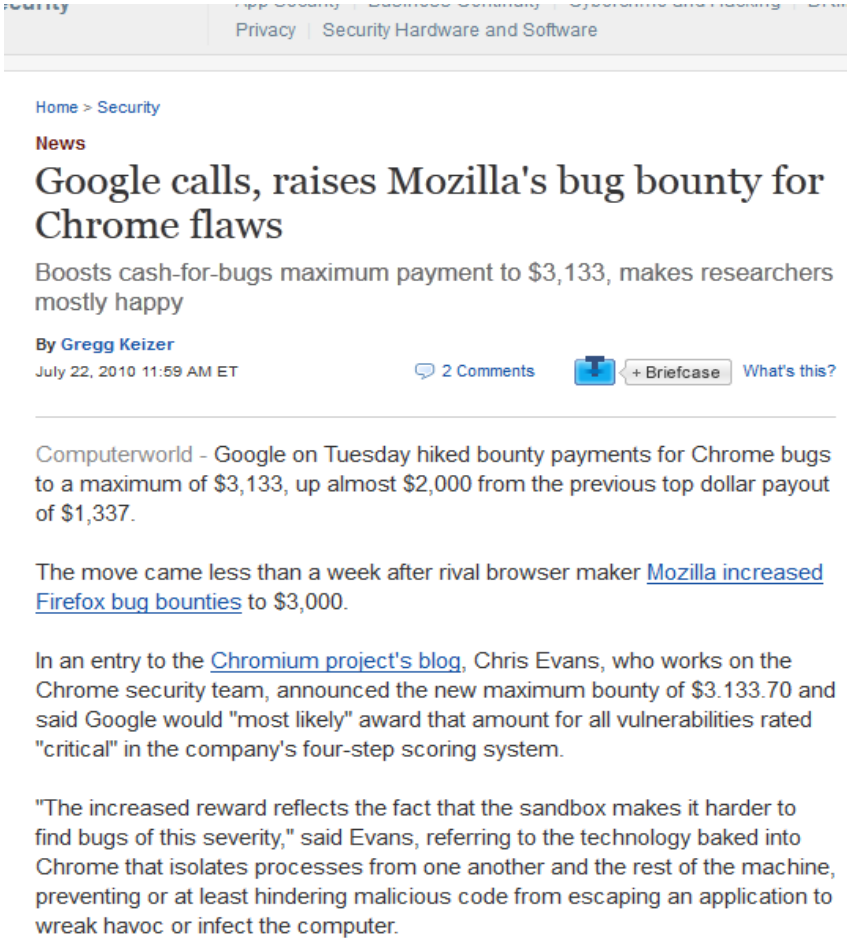
$17 + 40 hours per TP



which were wrong yet didn't actually affect the compiled code.

But most importantly, $1265 of bugs gives me the peace of mind of knowing that I'm not the only person who has looked at the Tarsnap code, and if there are more critical bugs like the security bug I fixed in January, they've escaped more than just my eyeballs. Worth the money? Every penny.

...REALLY?



Privacy | Security Hardware and Software

Home > Security

News

Google calls, raises Mozilla's bug bounty for Chrome flaws

Boosts cash-for-bugs maximum payment to $3,133, makes researchers mostly happy

By Gregg Keizer

July 22, 2010 11:59 AM ET

2 Comments + Briefcase What's this?

Computerworld - Google on Tuesday hiked bounty payments for Chrome bugs to a maximum of $3,133, up almost $2,000 from the previous top dollar payout of $1,337.

The move came less than a week after rival browser maker Mozilla increased Firefox bug bounties to $3,000.

In an entry to the Chromium project's blog, Chris Evans, who works on the Chrome security team, announced the new maximum bounty of $3.133.70 and said Google would "most likely" award that amount for all vulnerabilities rated "critical" in the company's four-step scoring system.

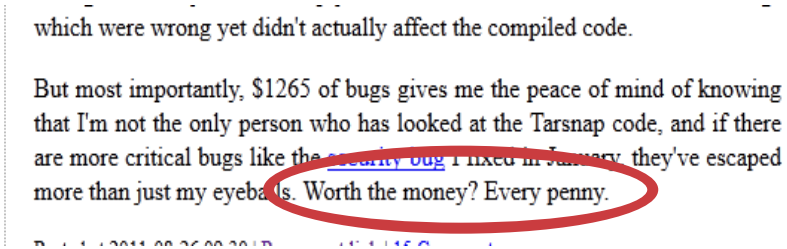
"The increased reward reflects the fact that the sandbox makes it harder to find bugs of this severity," said Evans, referring to the technology baked into Chrome that isolates processes from one another and the rest of the machine, preventing or at least hindering malicious code from escaping an application to wreak havoc or infect the computer.

Tarsnap:

125 spelling/style
63 harmless
11 minor
+ 1 major

75/200 = 38% TP rate

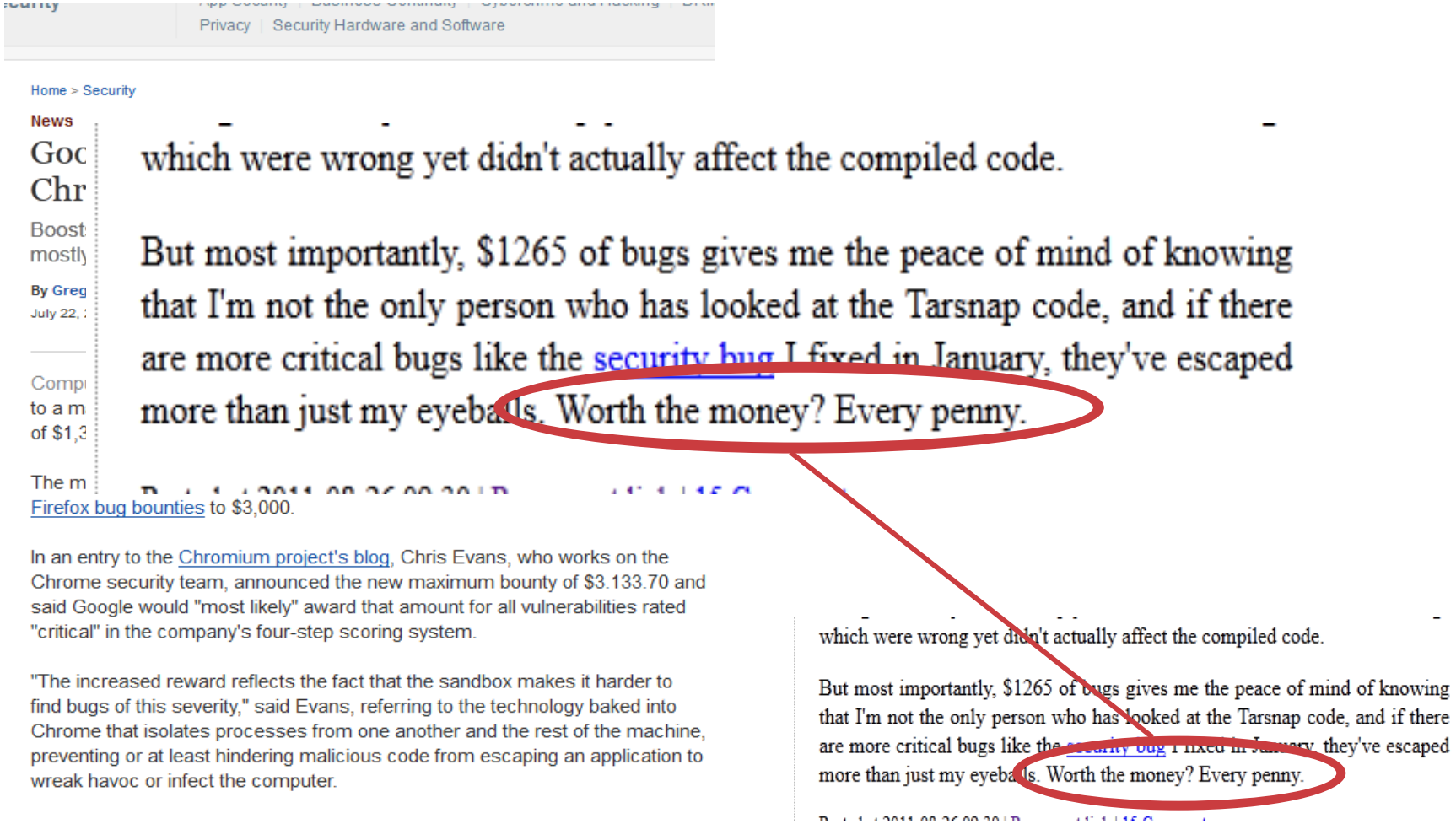
$17 + 40 hours per TP



which were wrong yet didn't actually affect the compiled code.

But most importantly, $1265 of bugs gives me the peace of mind of knowing that I'm not the only person who has looked at the Tarsnap code, and if there are more critical bugs like the security bug I fixed in January, they've escaped more than just my eyeballs. Worth the money? Every penny.

...REALLY?


Privacy | Security Hardware and Software
Home > Security
News
which were wrong yet didn't actually affect the compiled code.
But most importantly, $1265 of bugs gives me the peace of mind of knowing that I'm not the only person who has looked at the Tarsnap code, and if there are more critical bugs like the security bug I fixed in January, they've escaped more than just my eyeballs. Worth the money? Every penny.
Firefox bug bounties to $3,000.
In an entry to the Chromium project's blog, Chris Evans, who works on the Chrome security team, announced the new maximum bounty of $3.133.70 and said Google would "most likely" award that amount for all vulnerabilities rated "critical" in the company's four-step scoring system.
"The increased reward reflects the fact that the sandbox makes it harder to find bugs of this severity," said Evans, referring to the technology baked into Chrome that isolates processes from one another and the rest of the machine, preventing or at least hindering malicious code from escaping an application to wreak havoc or infect the computer.

SOLUTION: PAY STRANGERS

SOLUTION:
~~PAY STRANGERS~~

SOLUTION: AUTOMATE

AUTOMATED PROGRAM REPAIR

GENPROG: AUTOMATIC[1], SCALABLE, COMPETITIVE BUG REPAIR.

[1] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer, "GenProg: A generic method for automated software repair," *Transactions on Software Engineering,* vol. 38, no. 1, pp. 54– 72, 2012.
W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest, "Automatically finding patches using genetic programming," in *International Conference on Software Engineering,* 2009, pp. 364–367.

AUTOMATED PROGRAM REPAIR

GENPROG: AUTOMATIC[1], SCALABLE, COMPETITIVE BUG REPAIR.

[1] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer, "GenProg: A generic method for automated software repair," *Transactions on Software Engineering,* vol. 38, no. 1, pp. 54– 72, 2012.
W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest, "Automatically finding patches using genetic programming," in *International Conference on Software Engineering,* 2009, pp. 364–367.

AUTOMATED PROGRAM REPAIR

GENPROG:

AUTOMATIC,

SCALABLE,

COMPETITIVE

BUG REPAIR.

AUTOMATED PROGRAM REPAIR

GENPROG:

AUTOMATIC,

SCALABLE,

COMPETITIVE

BUG REPAIR.

AUTOMATED PROGRAM REPAIR

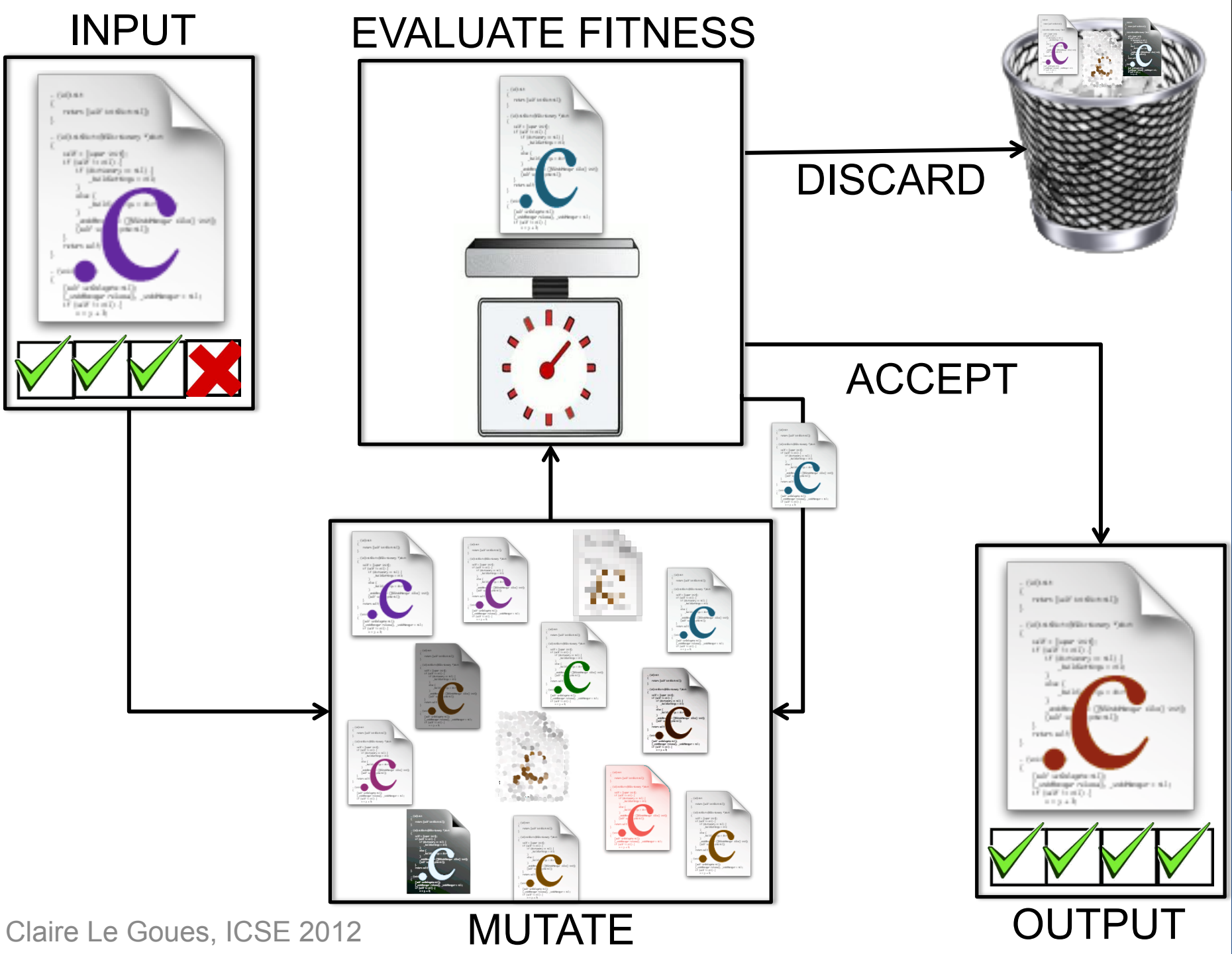
GENPROG:

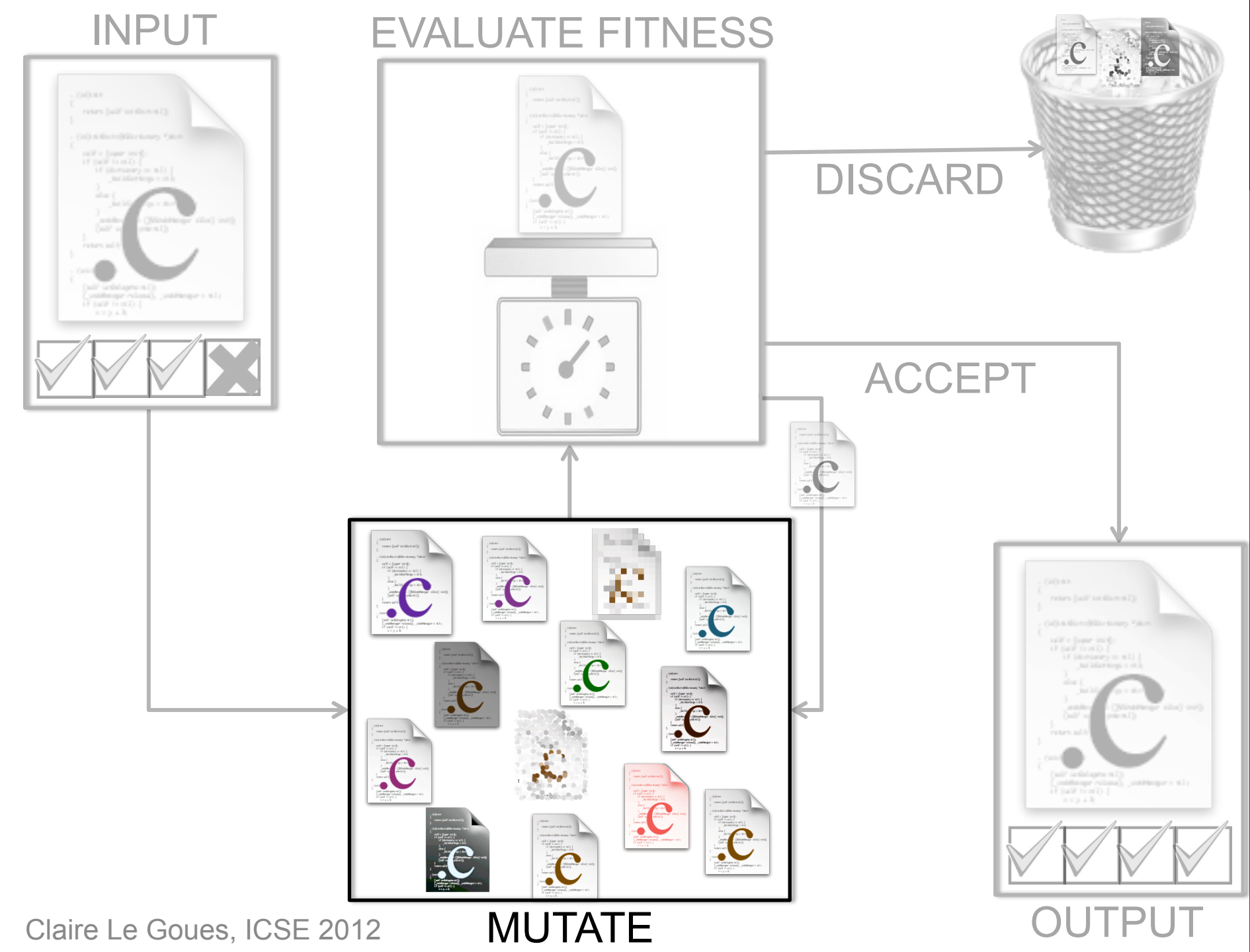
AUTOMATIC,

SCALABLE,

COMPETITIVE

BUG REPAIR.


INPUT
EVALUATE FITNESS
DISCARD
ACCEPT
MUTATE
OUTPUT


INPUT
EVALUATE FITNESS
DISCARD
ACCEPT
MUTATE
OUTPUT

BIRD'S EYE VIEW

Search: random (GP) search through nearby patches.

Approach: compose small random edits.

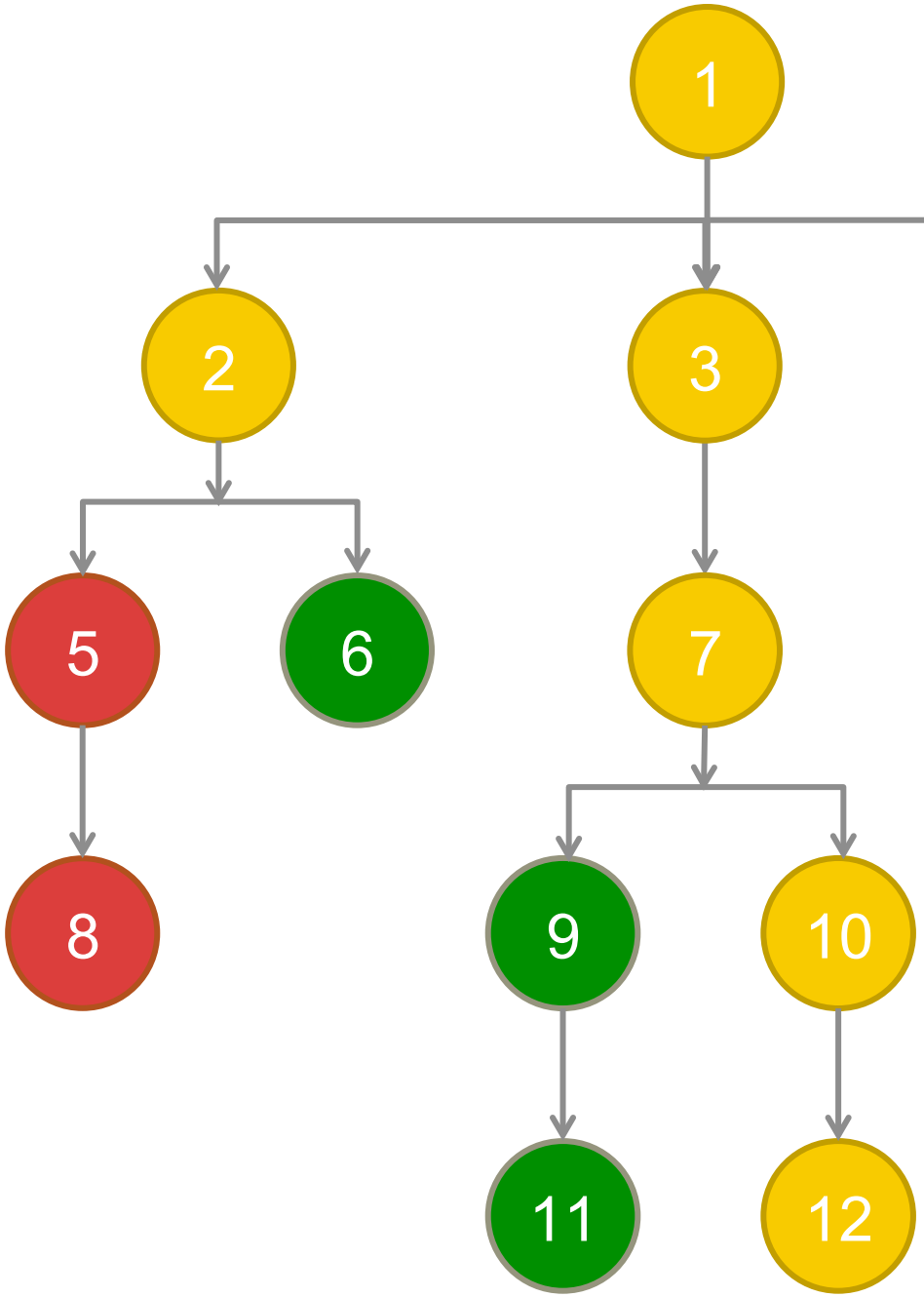
- Where to change?
- How to change it?

Input:

Input: 

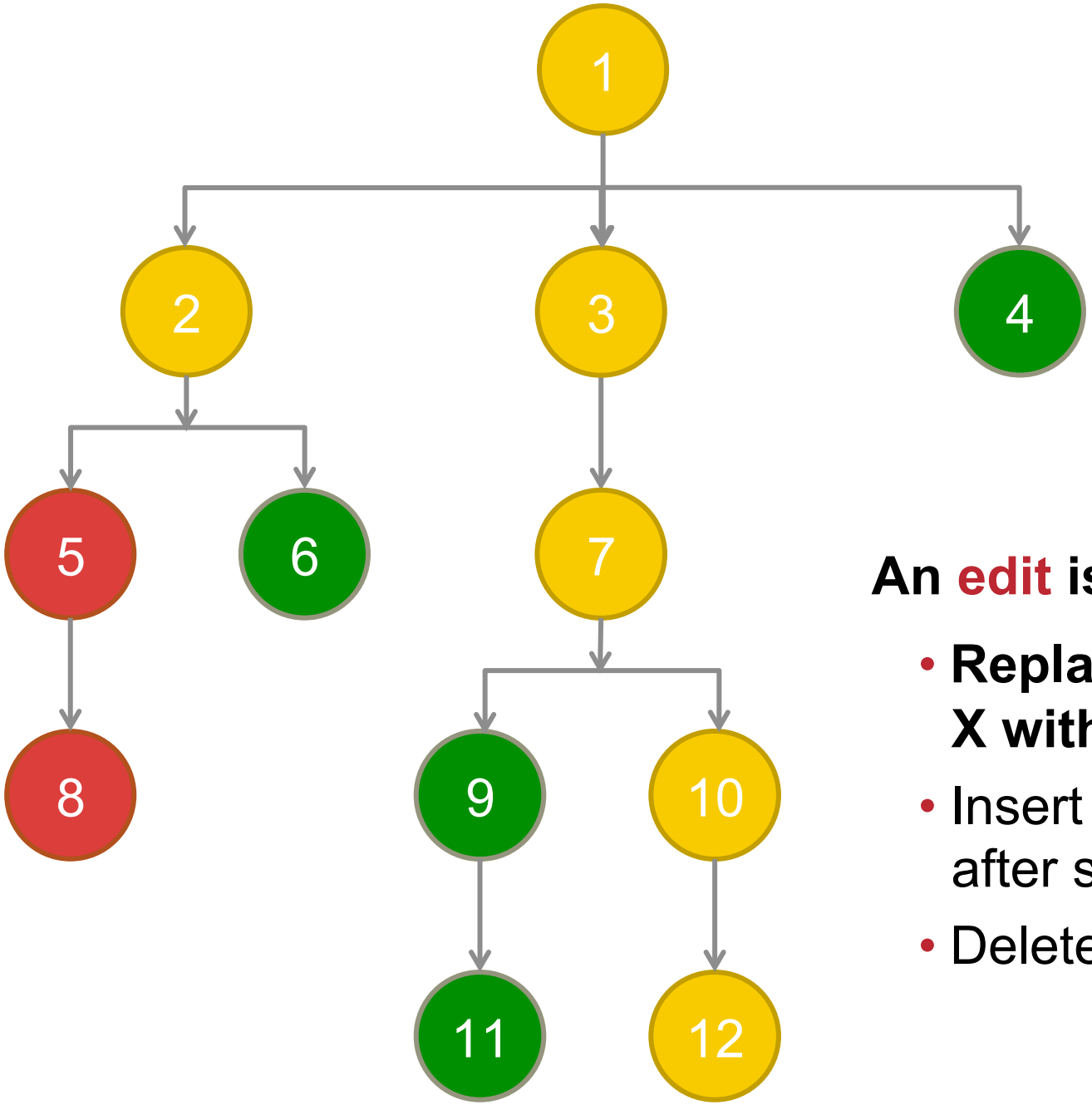
Legend:

- **High change probability.**
- **Low change probability.**
- **Not changed.**



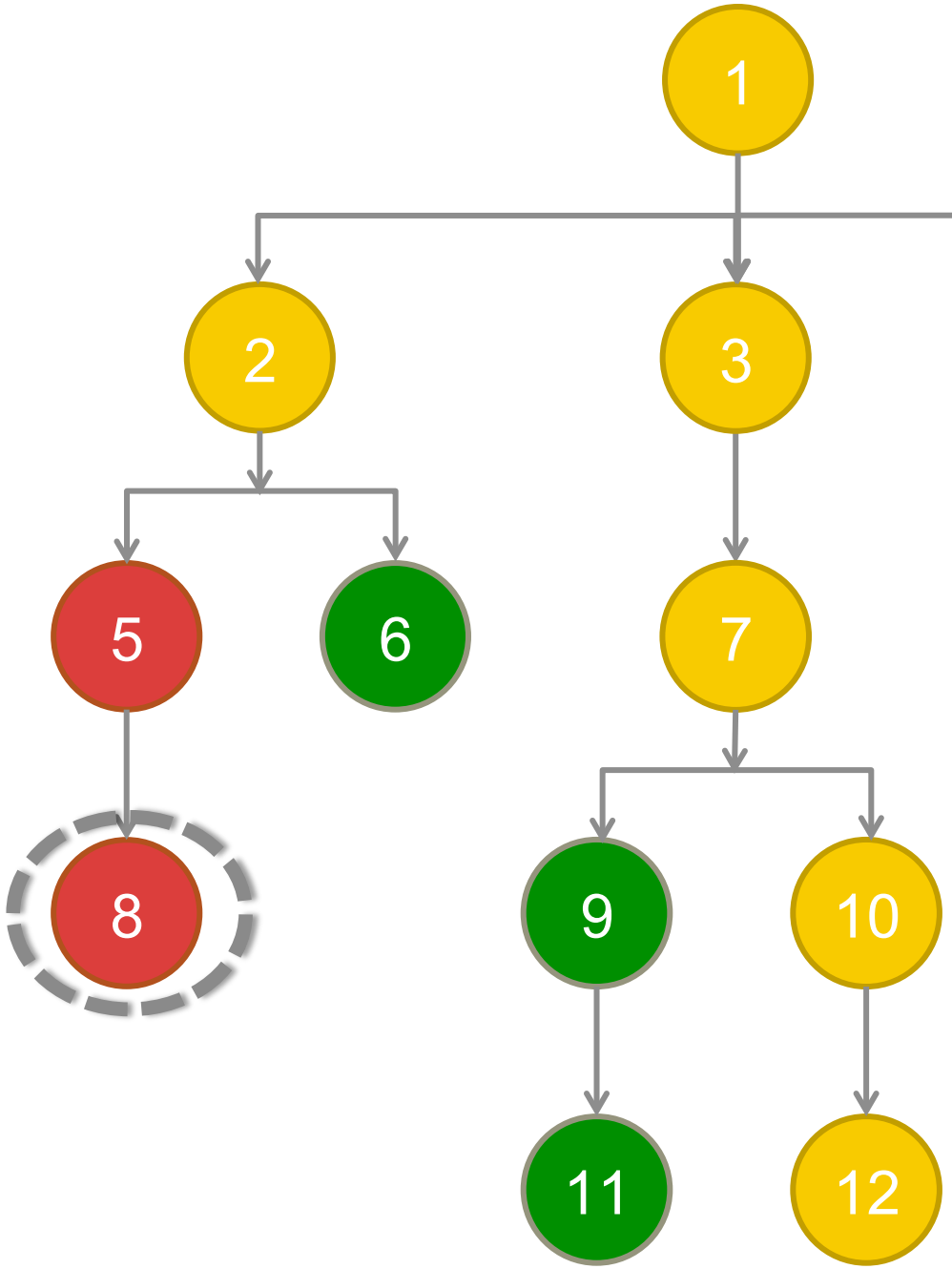
An edit is:

- Replace statement X with statement Y
- Insert statement X after statement Y
- Delete statement X



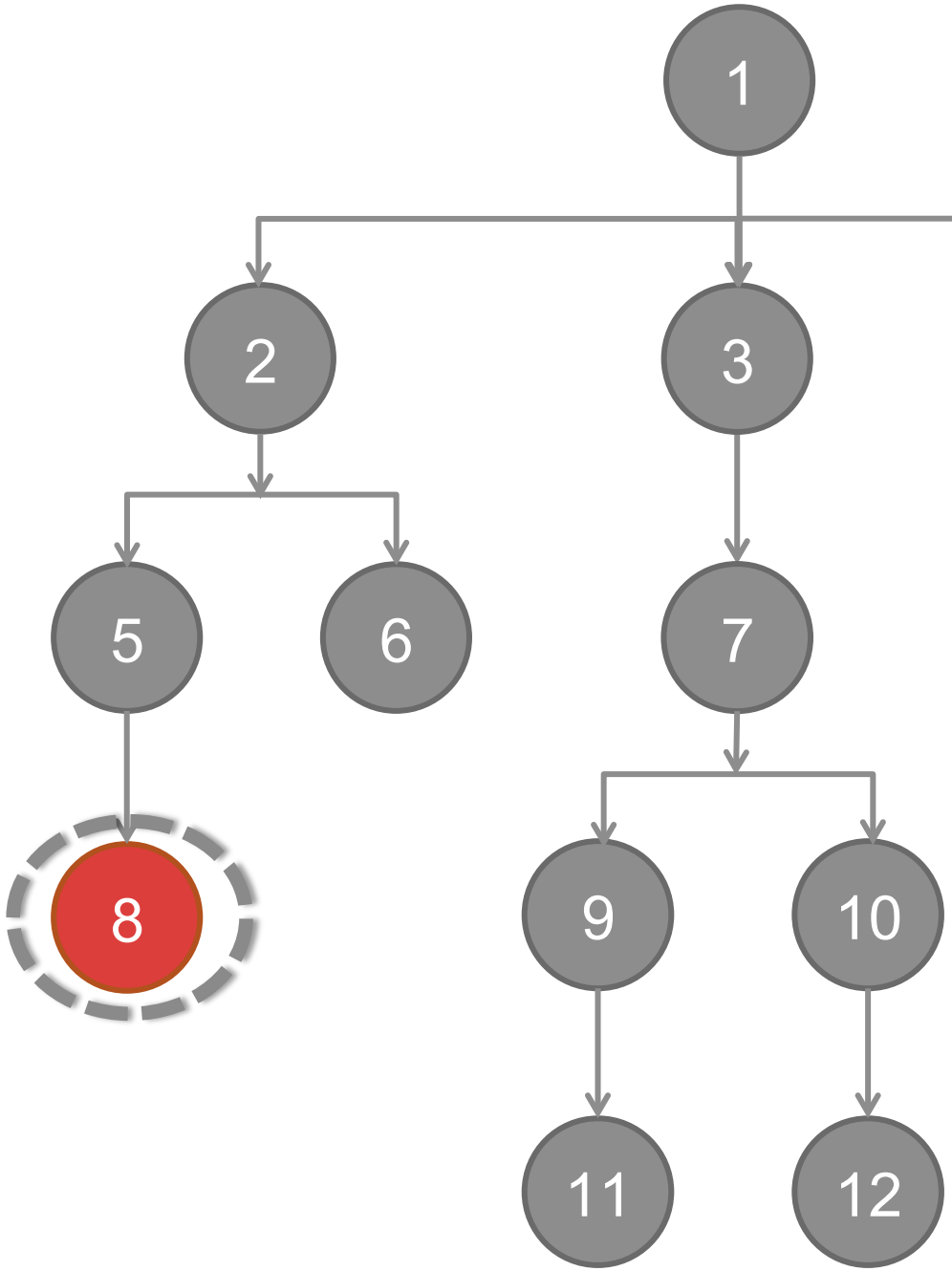
An edit is:

- **Replace statement X with statement Y**
- Insert statement X after statement Y
- Delete statement X



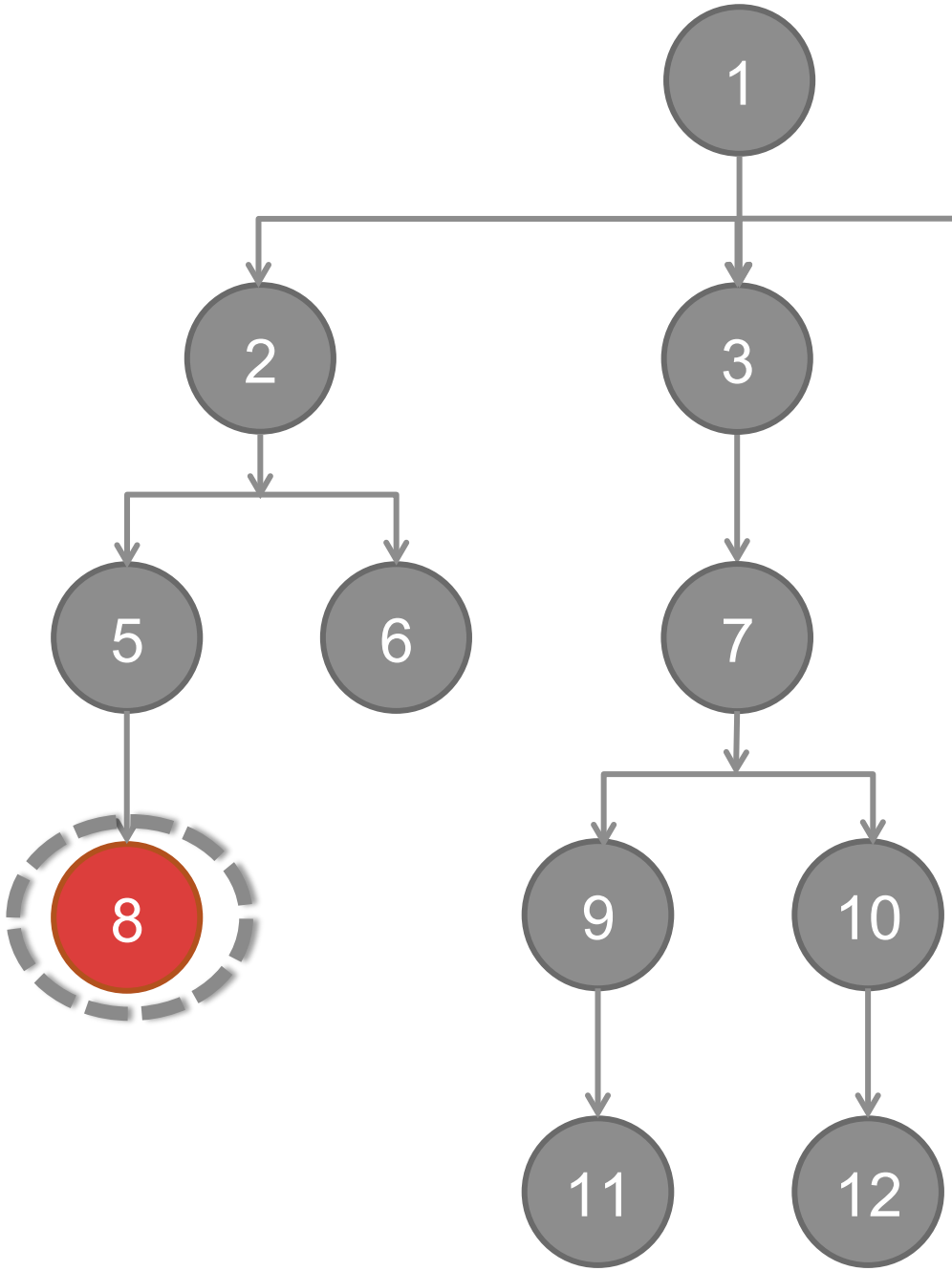
An edit is:

- **Replace statement X with statement Y**
- Insert statement X after statement Y
- Delete statement X



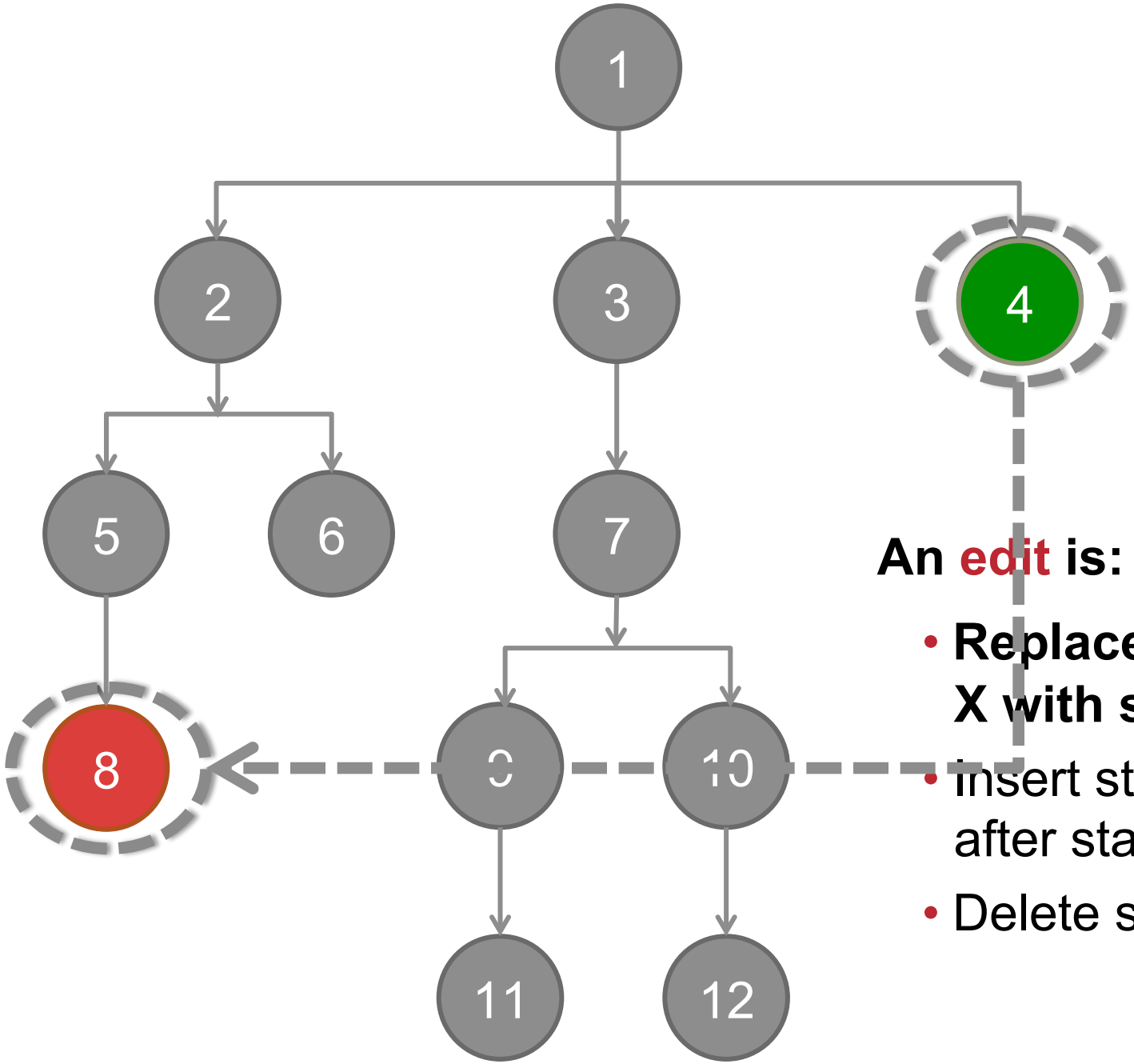
An edit is:

- **Replace statement X with statement Y**
- Insert statement X after statement Y
- Delete statement X



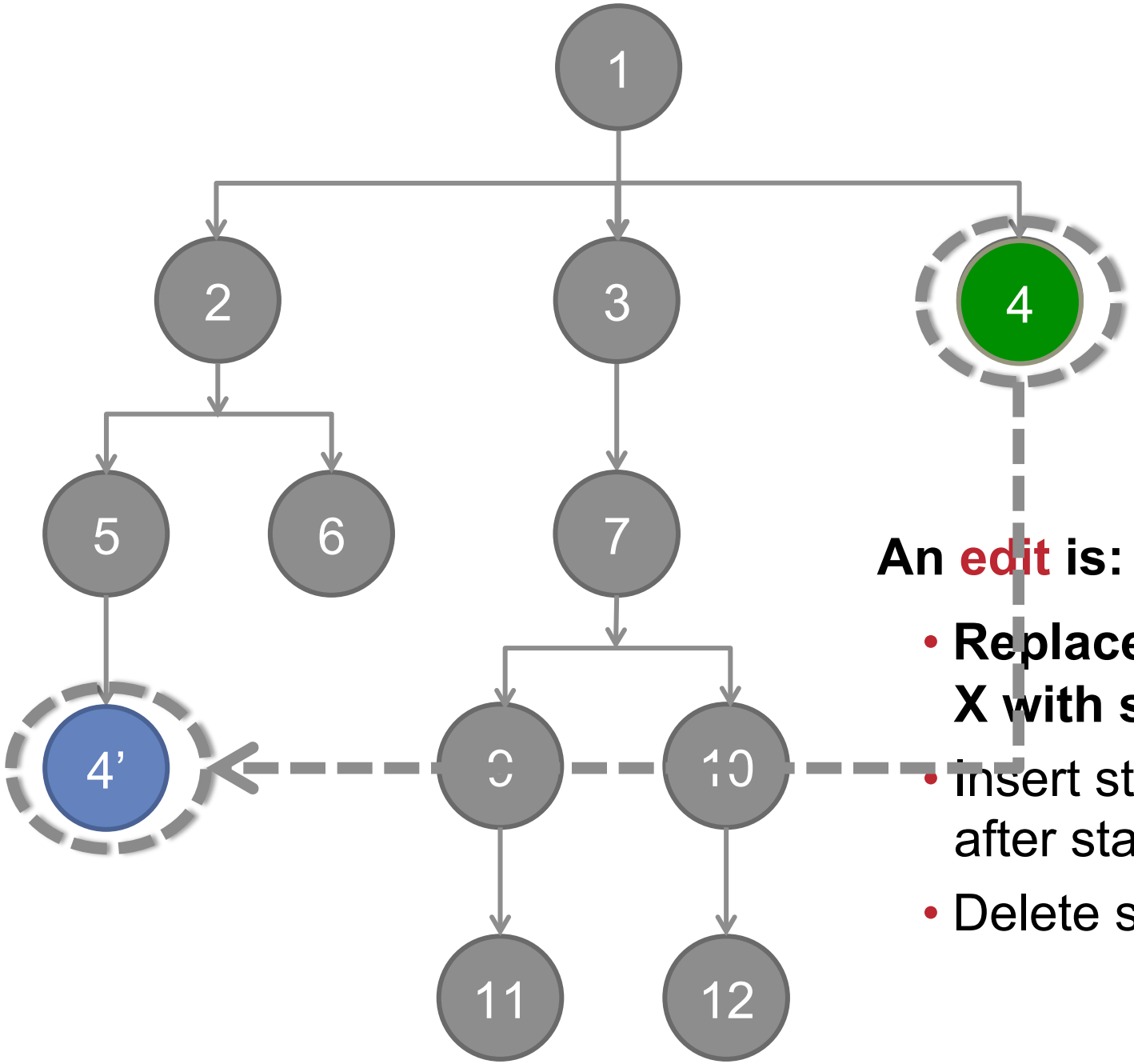
An edit is:

- **Replace statement X with statement Y**
- Insert statement X after statement Y
- Delete statement X



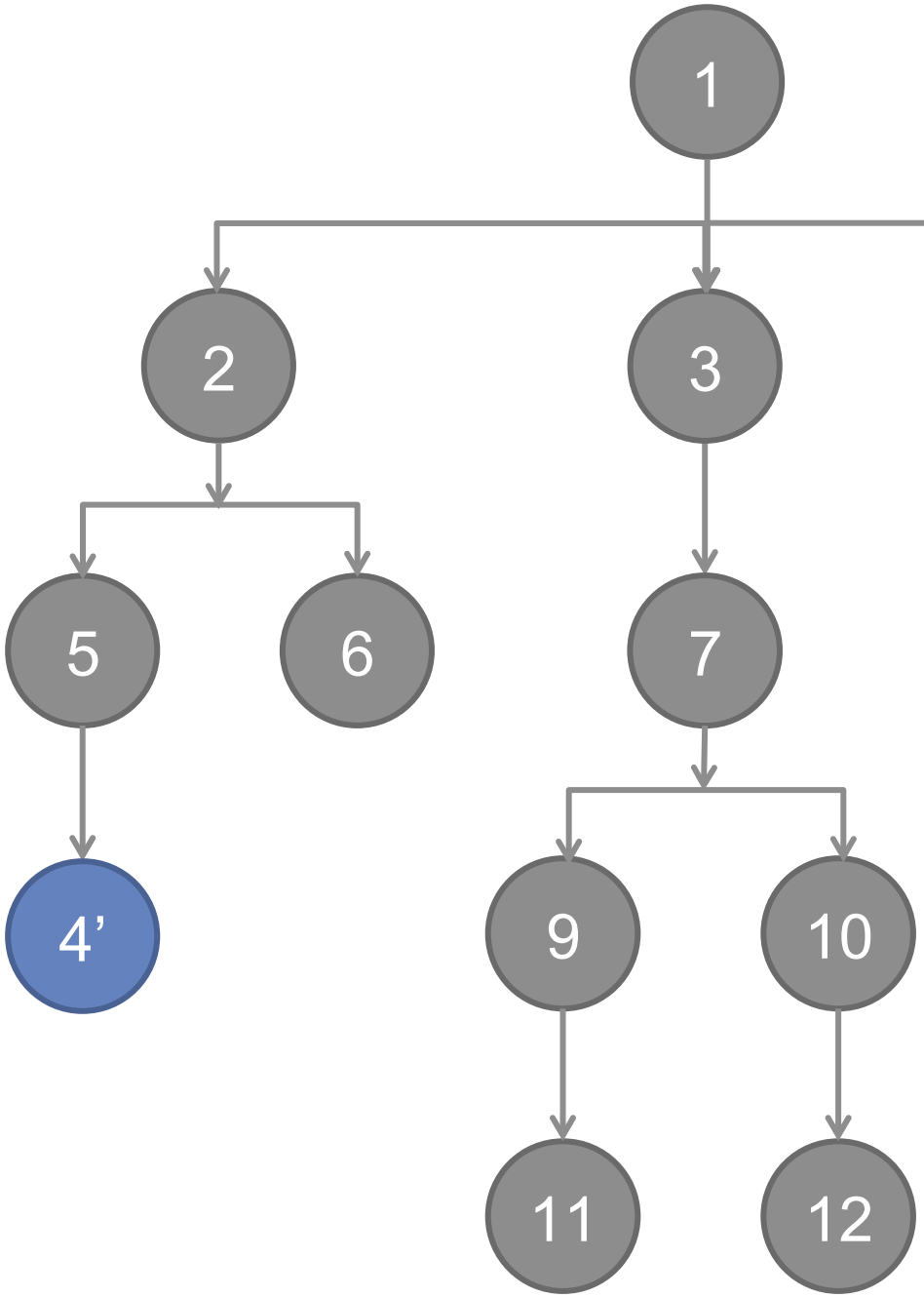
An edit is:

- **Replace statement X with statement Y**
- Insert statement X after statement Y
- Delete statement X



An edit is:

- **Replace statement X with statement Y**
- Insert statement X after statement Y
- Delete statement X



An edit is:

- **Replace statement X with statement Y**
- Insert statement X after statement Y
- Delete statement X

AUTOMATED PROGRAM REPAIR

GENPROG:

AUTOMATIC,

SCALABLE,

COMPETITIVE

BUG REPAIR.

AUTOMATED PROGRAM REPAIR

GENPROG:

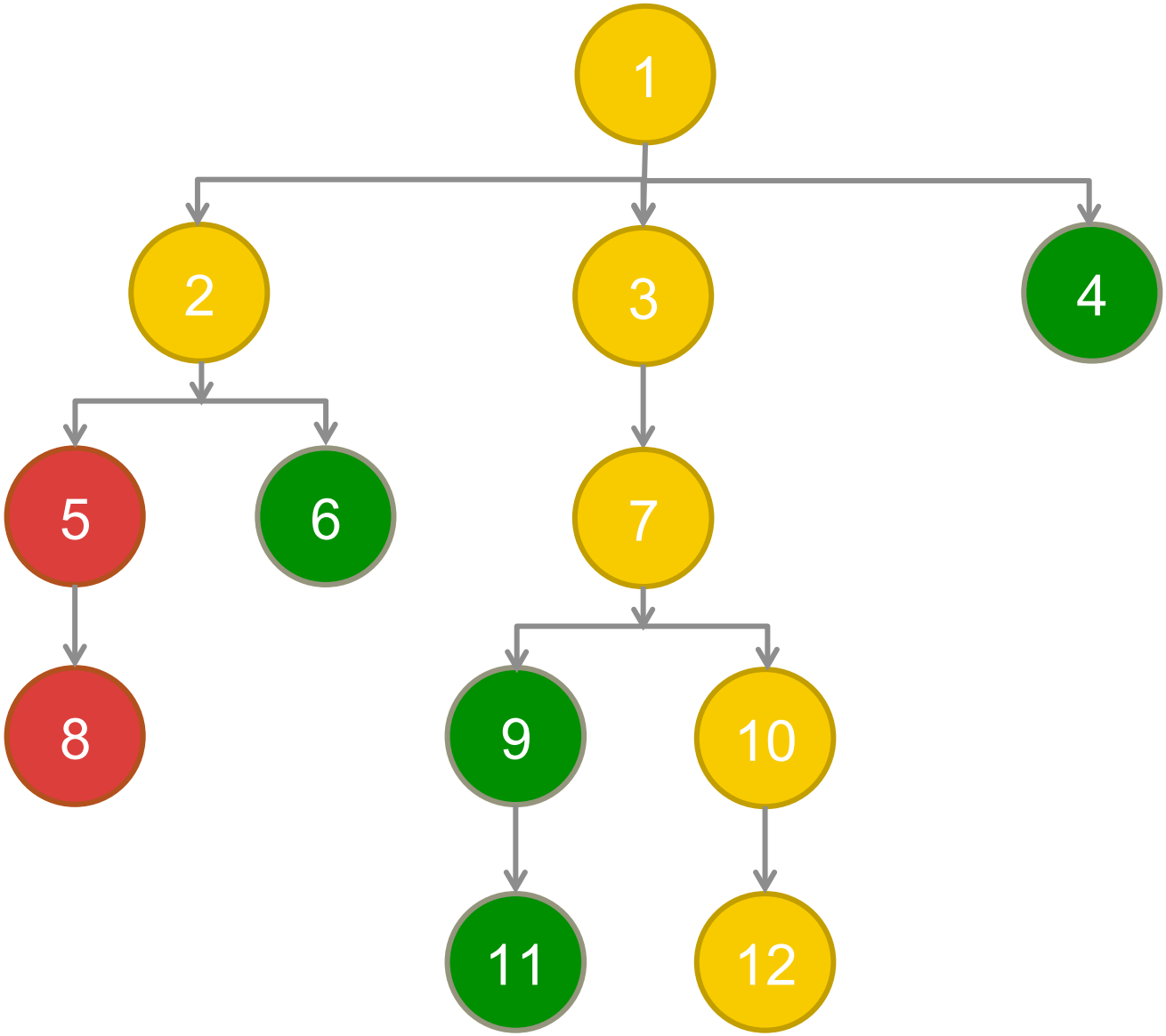
AUTOMATIC,

SCALABLE,

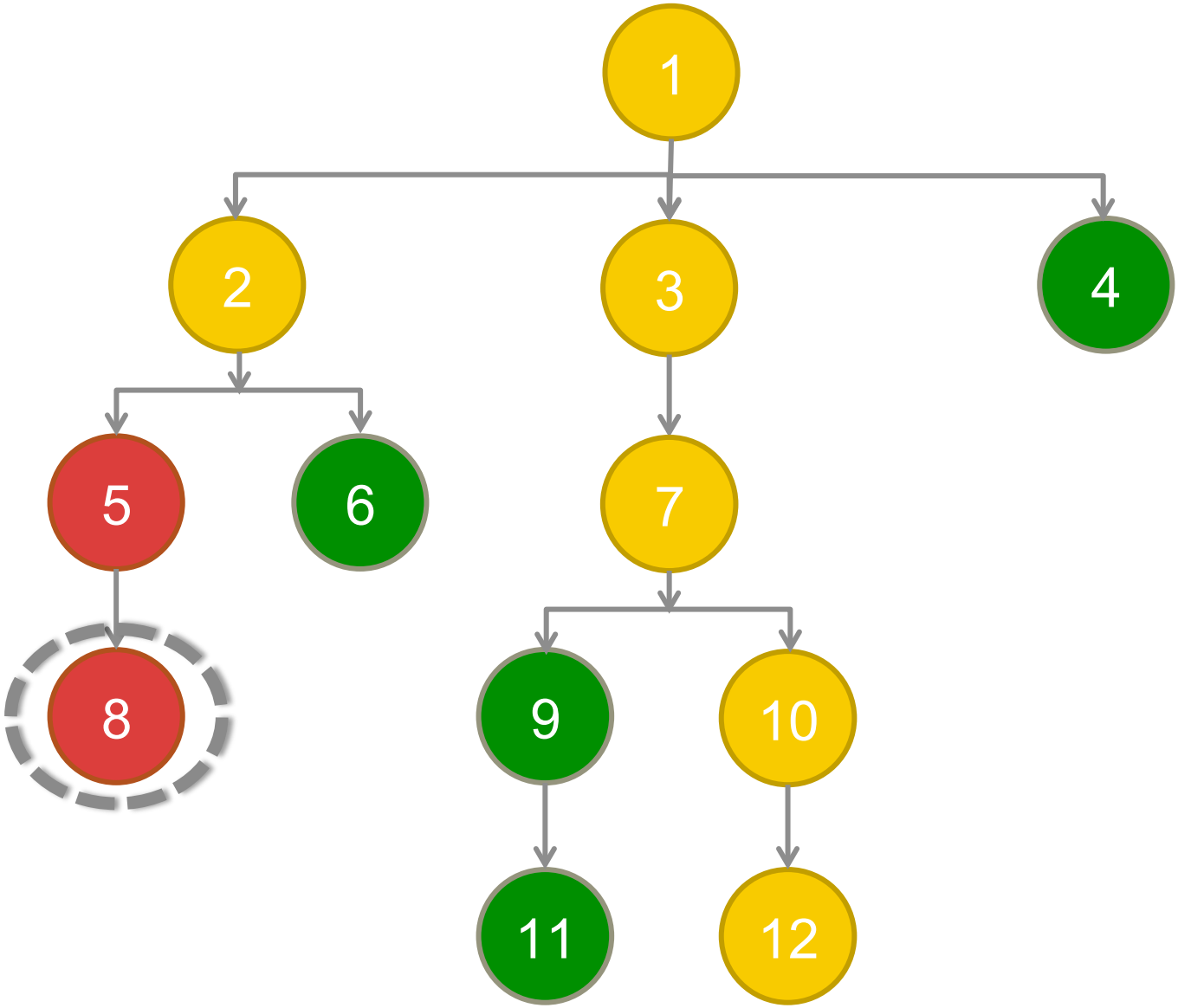
COMPETITIVE

BUG REPAIR.

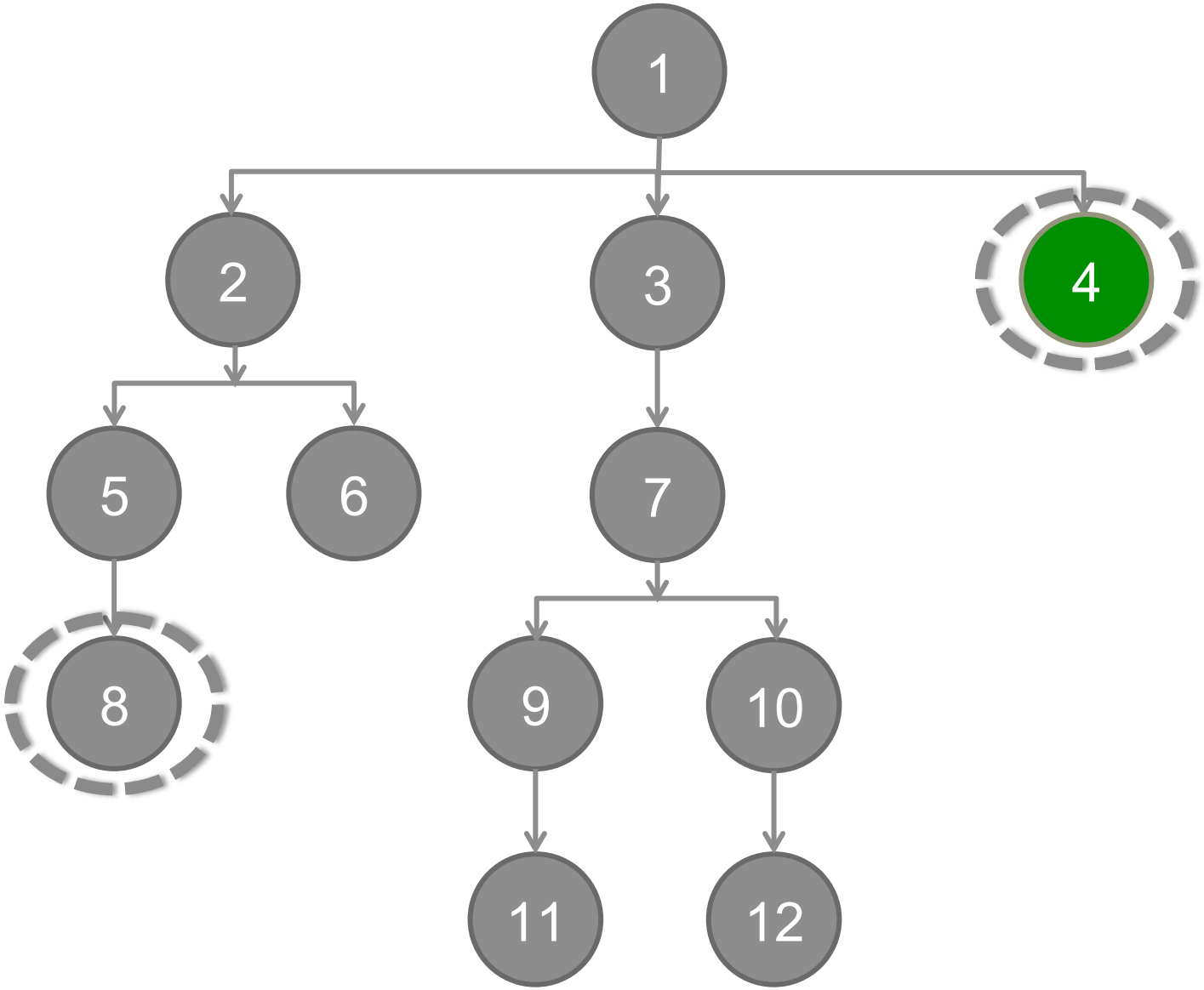
SCALABLE: SEARCH SPACE



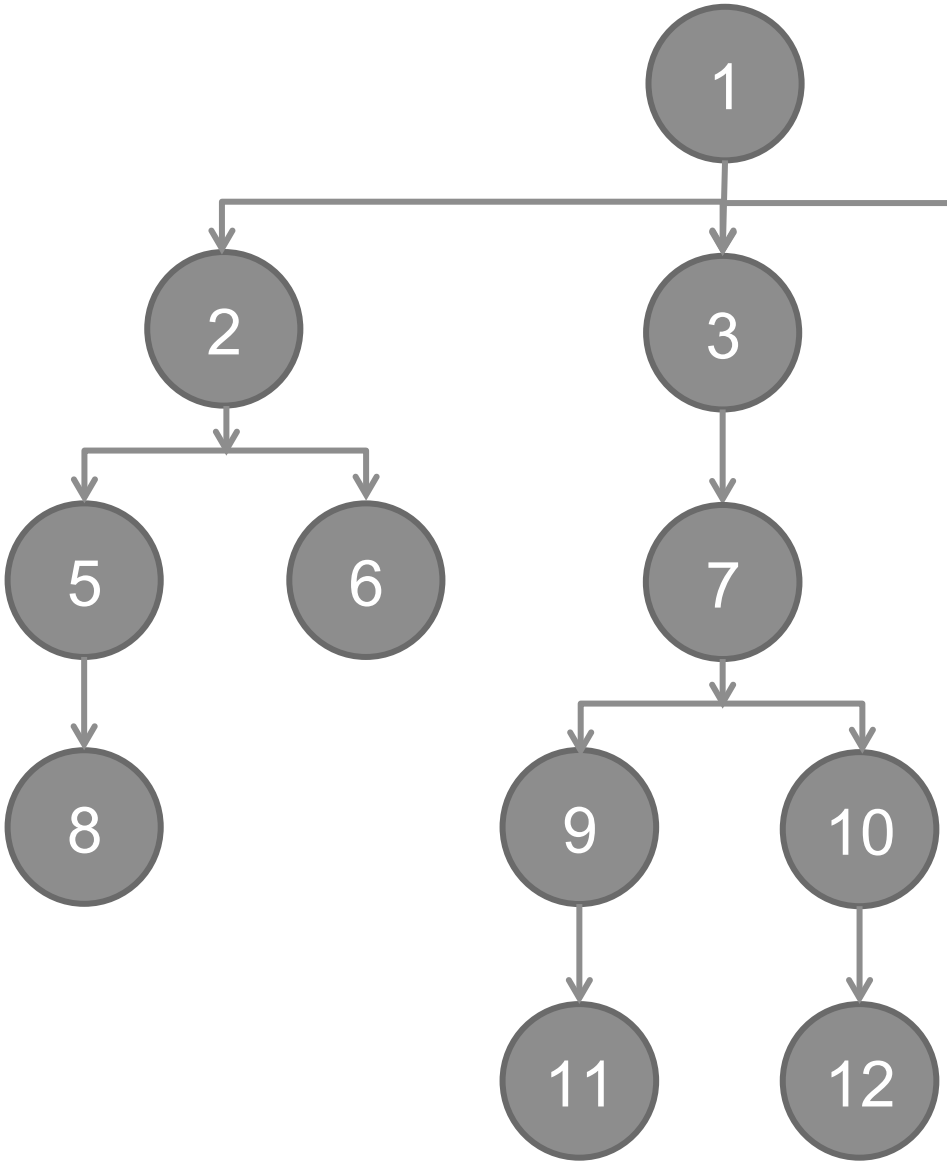
SCALABLE: SEARCH SPACE



SCALABLE: SEARCH SPACE



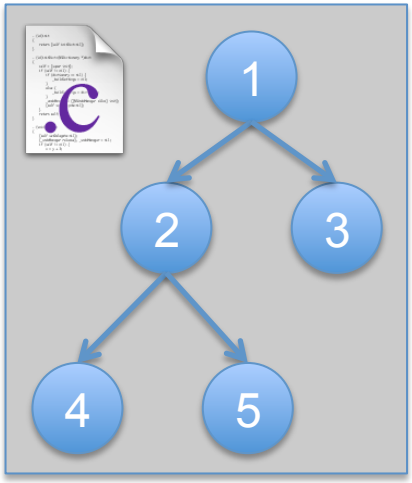
SCALABLE: SEARCH SPACE



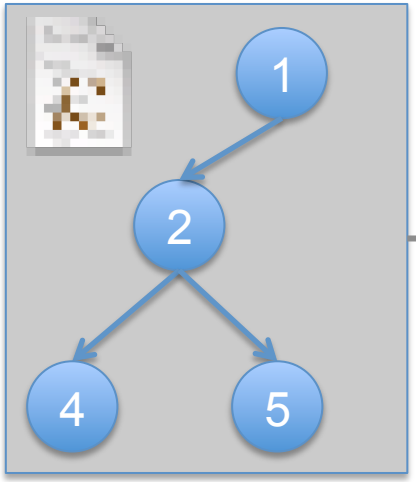
Fix localization: intelligently choose code to move.

SCALABLE: REPRESENTATION

Input:

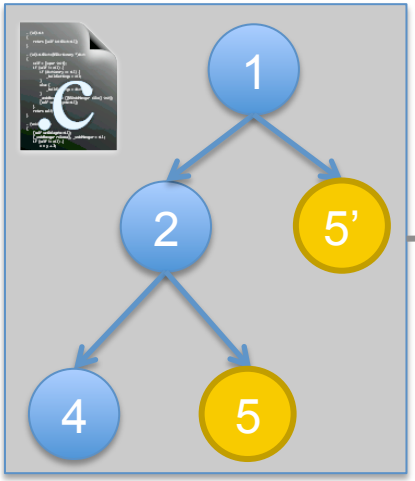


Naïve:



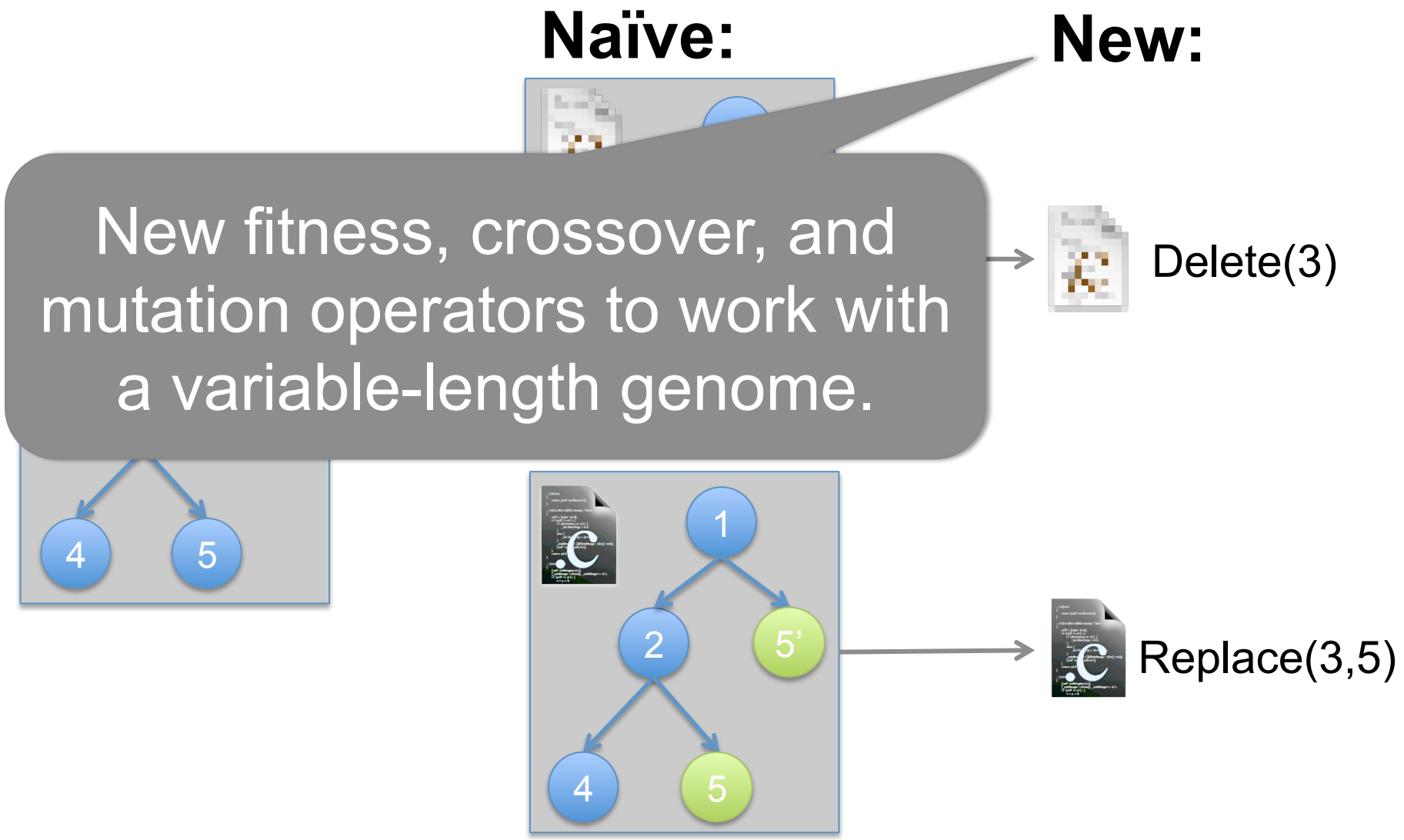
New:

Delete(3)



Replace(3,5)

SCALABLE: REPRESENTATION



SCALABLE: PARALLELISM

Fitness:

- Subsample test cases.
- Evaluate in parallel.

Random runs:

- Multiple simultaneous runs on different seeds.

AUTOMATED PROGRAM REPAIR

GENPROG:

AUTOMATIC,

SCALABLE,

COMPETITIVE

BUG REPAIR.

AUTOMATED PROGRAM REPAIR

GENPROG:

AUTOMATIC,

SCALABLE,

COMPETITIVE

BUG REPAIR.

How many bugs can GenProg fix?

COMPETITIVE

How much does it cost?

SETUP

Goal: systematically test GenProg on a general, indicative bug set.

General approach:

- Avoid overfitting: fix the algorithm.
- Systematically create a generalizable benchmark set.
- Try to repair every bug in the benchmark set, establish grounded cost measurements.

SETUP

Goal: systematically evaluate GenProg on a general, indicative bug set.

General approach:

- Avoid overfitting: fix the algorithm.
- Systematically create a generalizable benchmark set.
- **Try to repair every bug in the benchmark set, establish grounded cost measurements.**

CHALLENGE: INDICATIVE BUG SET

SYSTEMATIC BENCHMARK SELECTION

Goal: a large set of important, reproducible bugs in non-trivial programs.

Approach: use historical data to approximate discovery and repair of bugs in the wild.

SYSTEMATIC BENCHMARK SELECTION

Consider top programs from SourceForge, Google Code, Fedora SRPM, etc:

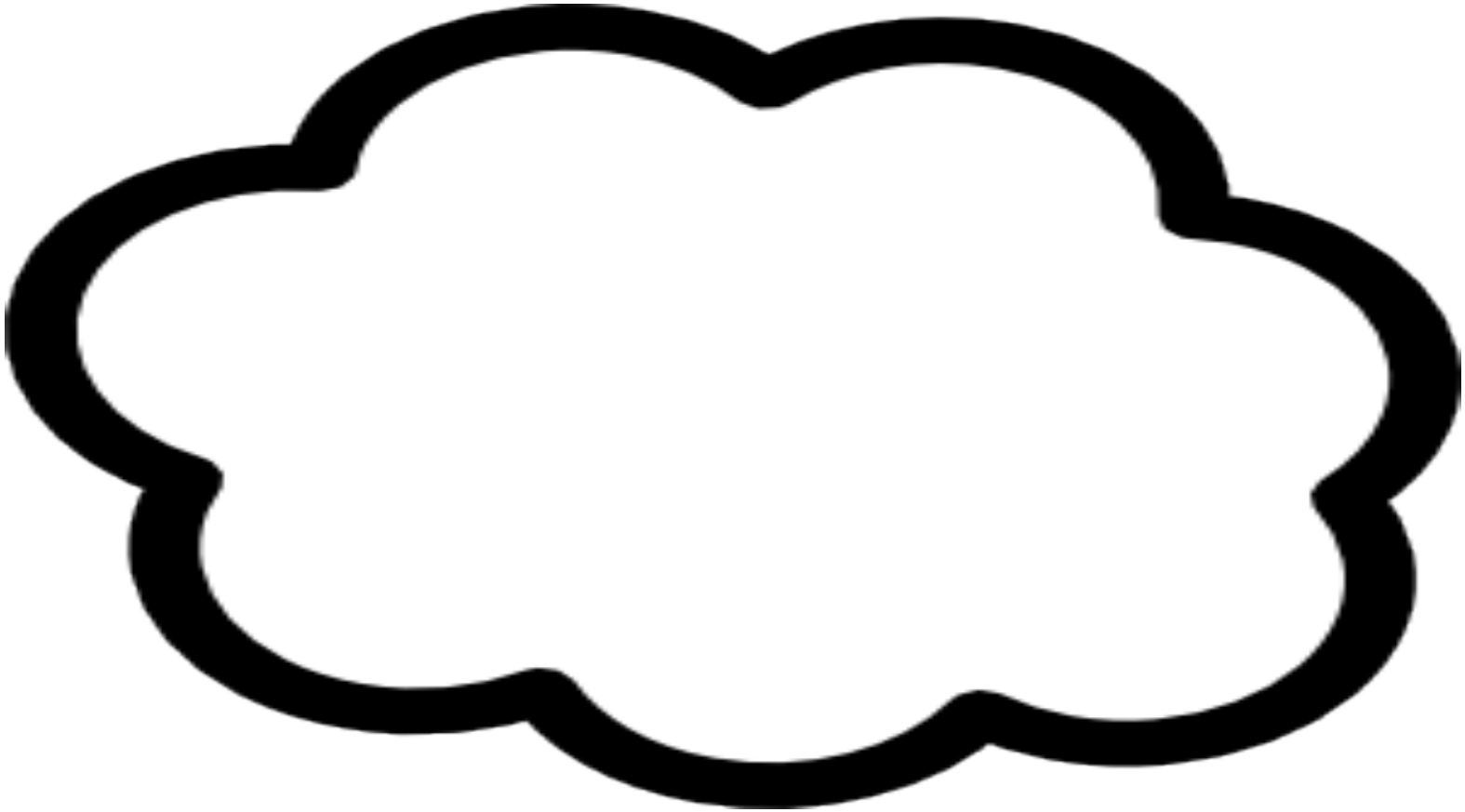
- Find pairs of viable versions where test case behavior changes.
- Take all tests from *most recent* version.
- Go *back in time* through the source control.

Corresponds to a human-written repair for the bug tested by the failing test case(s).

BENCHMARKS

Program	LOC	Tests	Bugs	Description
fbc	97,000	773	3	Language (legacy)
gmp	145,000	146	2	Multiple precision math
gzip	491,000	12	5	Data compression
libtiff	77,000	78	24	Image manipulation
lighttpd	62,000	295	9	Web server
php	1,046,000	8,471	44	Language (web)
python	407,000	355	11	Language (general)
wireshark	2,814,000	63	7	Network packet analyzer
Total	**5,139,000**	**10,193**	**105**	

CHALLENGE: GROUNDED COST MEASUREMENTS




amazon
web services™

READY

GO

13 HOURS LATER

SUCCESS/COST

Program	Defects Repaired	Cost per non-repair		Cost per repair	
		Hours	US$	Hours	US$
fbc	1/3	8.52	5.56	6.52	4.08
gmp	1/2	9.93	6.61	1.60	0.44
gzip	1/5	5.11	3.04	1.41	0.30
libtiff	17/24	7.81	5.04	1.05	0.04
lighttpd	5/9	10.79	7.25	1.34	0.25
php	28/44	13.00	8.80	1.84	0.62
python	1/11	13.00	8.80	1.22	0.16
wireshark	1/7	13.00	8.80	1.23	0.17
Total	**55/105**	**11.22h**		**1.60h**	

$403 for all 105 trials, leading to 55 repairs; $7.32 per bug repaired.

PUBLIC COMPARISON

JBoss issue tracking: median 5.0, mean 15.3 hours.[1]

IBM: $25 per defect during coding, rising at build, Q&A, post-release, etc.[2]

Tarsnap.com: $17, 40 hours per non-trivial repair.[3]

Bug bounty programs in general:

- At least $500 for security-critical bugs.
- One of our php bugs has an associated security CVE.

[1]C. Weiß, R. Premraj, T. Zimmermann, and A. Zeller, "How long will it take to fix this bug?" in *Workshop on Mining Software Repositories,* May 2007.

[2] L. Williamson, "IBM Rational software analyzer: Beyond source code," in *Rational Software Developer Conference,* Jun. 2008.

[3]http://www.tarsnap.com/bugbounty.html

CONCLUSIONS/CONTRIBUTIONS

GenProg: scalable, automatic bug repair.

- Algorithmic improvements for scalability: fix localization, internal representation, parallelism.

Systematic study:

- Indicative, systematically-generated set of bugs that humans care about.
- Repaired 52% of 105 bugs in 96 minutes, on average, for $7.32 each.

Benchmarks/results/source code/VM images available:

- http://genprog.cs.virginia.edu

I LOVE QUESTIONS.

(Examples: “Which bugs can GenProg fix?” “What happens if you run for more than 13 hours/change the probability distributions/ pick a different crossover/etc?” “How do you know the patches are any good?” “How do your patches compare to human patches?” …)

WHICH BUGS...?

Slightly more likely to fix bugs where the human:

- restricts the repair to statements.
- touched fewer files.

As fault space decreases, success increases, repair time decreases.

As fix space increases, repair time decreases.

FINDING BUGS IS HARD

Opaque or non-automated GUI testing.

- Firefox, Eclipse, OpenOffice

Inaccessible or small version control histories.

- bash, cvs, openssh

Few viable versions for recent tests.

- valgrind

Require incompatible automake, libtool

- Earlier versions of gmp

No bugs

- GnuCash, openssl

Non-deterministic tests ...

EXAMPLE: PHP BUG #54372

Relevant code: function `zend_std_read_property` in `zend_object_handlers.c`

Note: memory management uses reference counting.

Problem: this line:

`449.zval_ptr_dtor(object)`

If `object` points to `$this` and `$this` is global, its memory is completely freed, even though we could access `$this` later.

```
1.  class test_class {
2.    public function __get($n)
3.      { return $this; %$ }
4.      public function b()
5.       { return; }
6.  }
7.  global $test3;
8.  $test3 = new test_class();
9.  $test3->a->b();
```

Expected output: nothing

Buggy output: crash on line 9.

EXAMPLE: PHP BUG #54372

Human :

```
% 449c449,453
< zval_ptr_dtor(&object);
> if (*retval != object)
> { // expected
>  zval_ptr_dtor(&object);
> } else {
>  Z_DELREF_P(object);
> }
```

GenProg :

```
% 448c448,451
> Z_ADDROF_P(object);
> if (PZVAL_IS_REF(object))
> {
>  SEPARATE_ZVAL(&object);
> }
  zval_ptr_dtor(&object)
```

PATCH QUALITY

Is automatically-patched code more or less maintainable?

Approach: Ask 102 humans maintainability questions about patched code (human vs. GenProg).

Results:

- No difference in accuracy/time between human accepted and GenProg patches.
- Automatically-documented GenProg patches result in higher accuracy and lower effort than human patches.

Zachary P. Fry, Bryan Landau, Westley Weimer: A Human Study of Patch Maintainability. International Symposium on Software Testing and Analysis (ISSTA) 2012: to appear

PATCH REPRESENTATION

Program	Fault	LOC	Repair Ratio
gcd	infinite loop	22	1.07
uniq-utx	segfault	1146	1.01
look-utx	segfault	1169	1.00
look-svr	infinite loop	1363	1.00
units-svr	segfault	1504	3.13
deroff-utx	segfault	2236	1.22
nullhttpd	buffer exploit	5575	1.95
indent	infinite loop	9906	1.70
flex	segfault	18775	3.75
atris	buffer exploit	21553	0.97
Average		**6325**	**1.68**