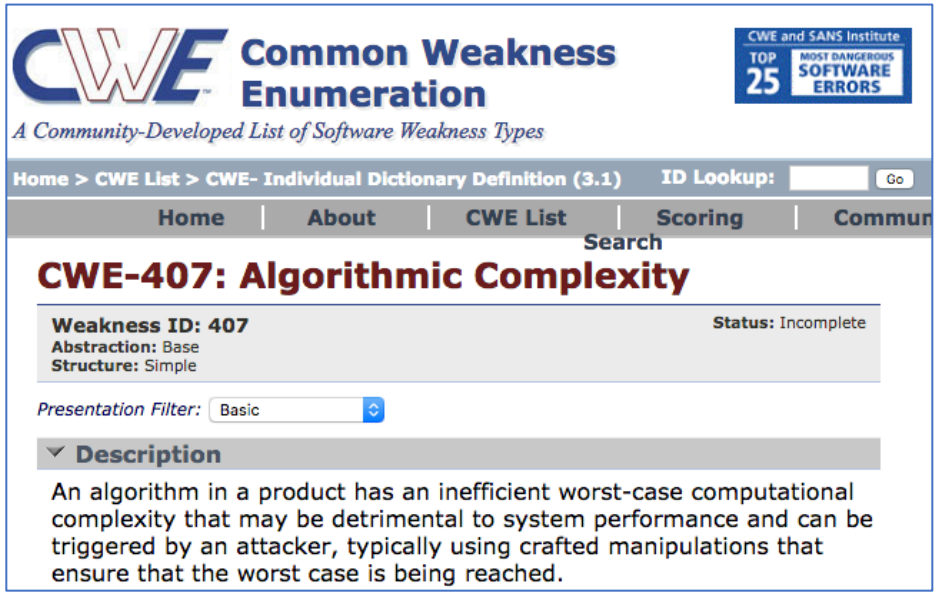
PerfFuzz: Automatically Generating Pathological Inputs

Caroline Lemieux, Rohan Padhye, Koushik Sen, Dawn Song
University of California, Berkeley

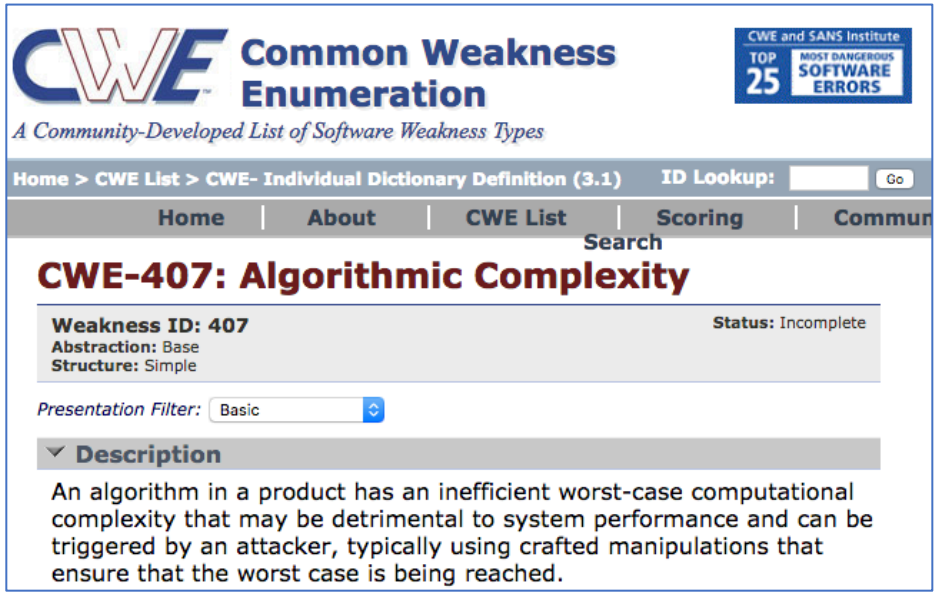
source: https://github.com/carolemieux/perffuzz

Nobody Expects Performance Problems

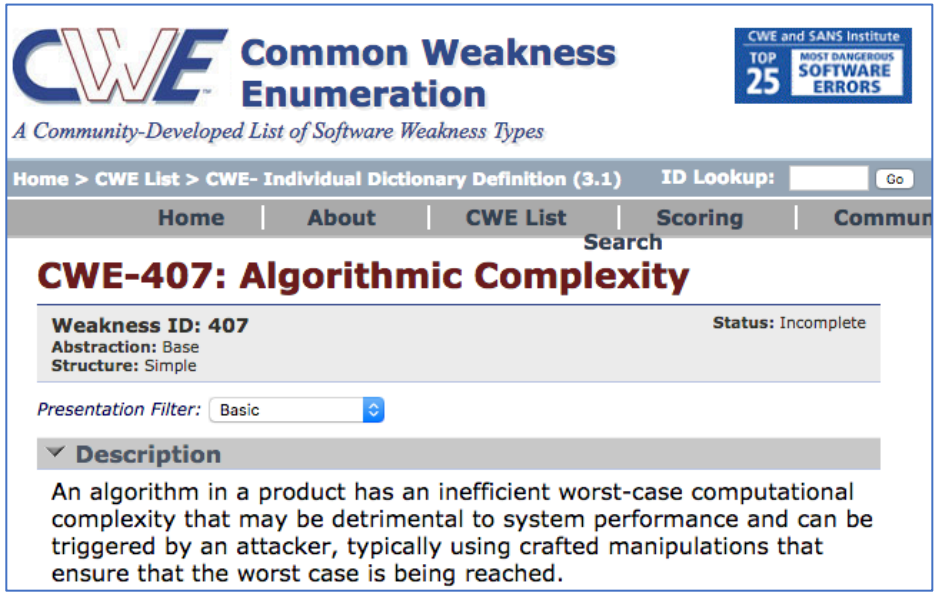
Performance Problems Have Consequences



Performance Problems Have Consequences

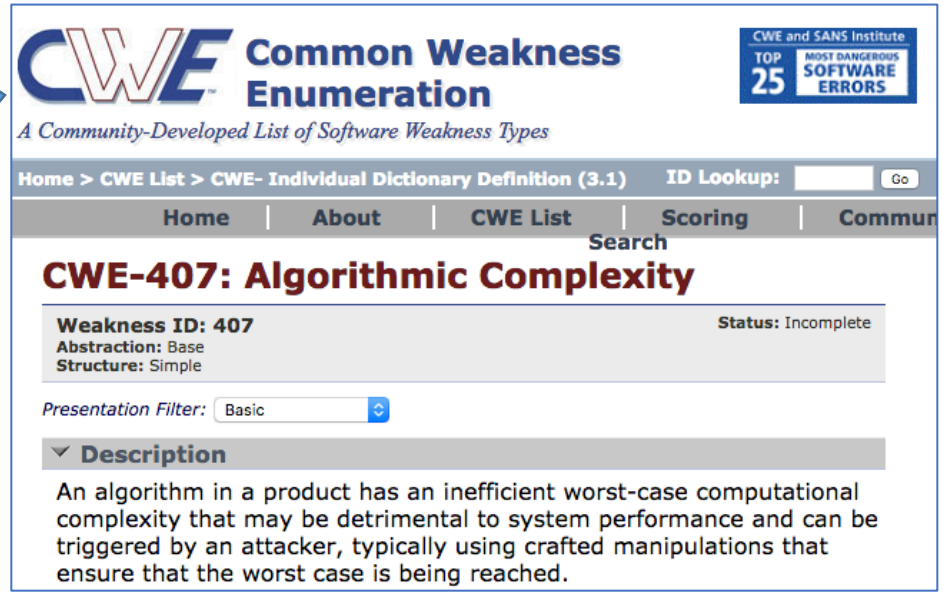


Performance Problems Have Consequences



Performance Problems Have Consequences

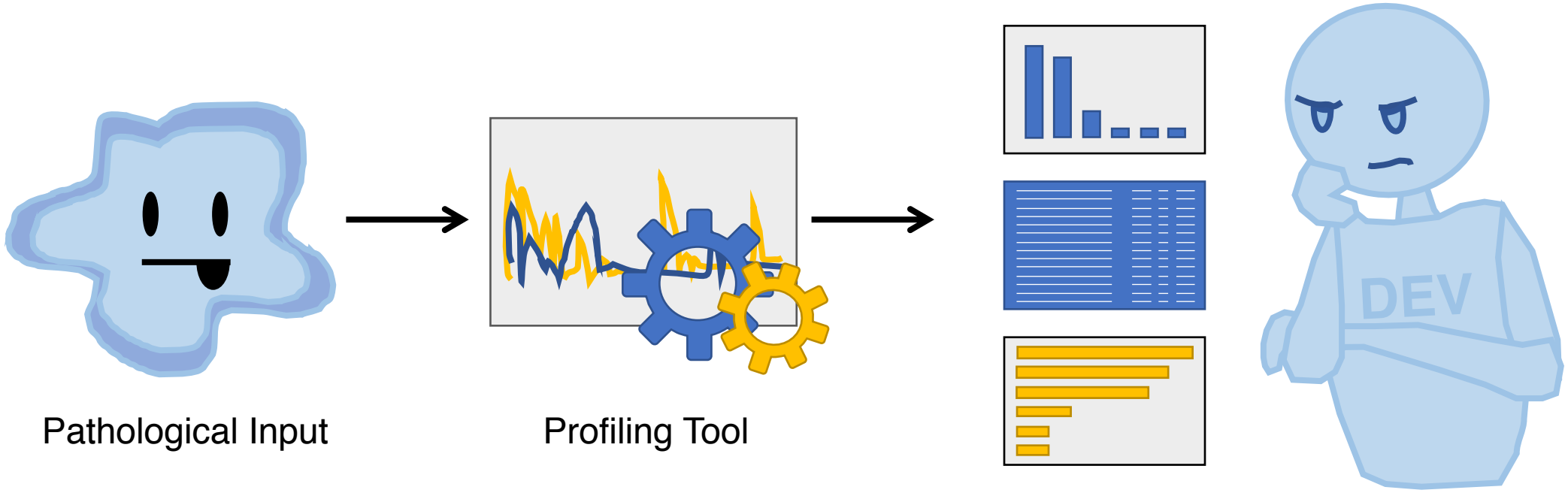
security vulnerabilities (DoS)



poor user experience

excessive resource consumption

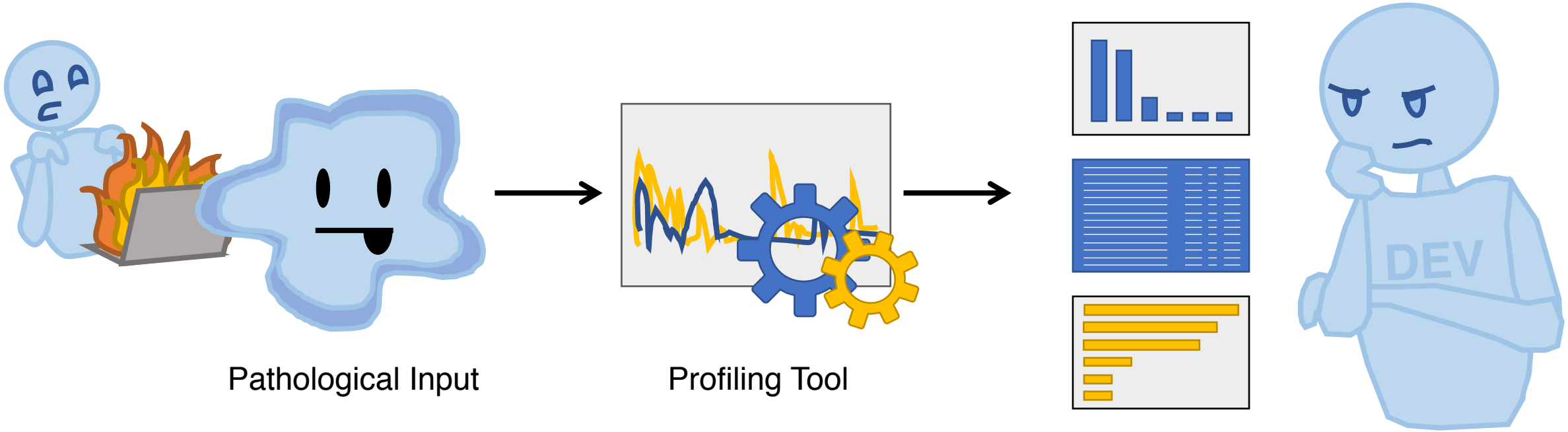
Alleviating Performance Problems



Pathological Input

Profiling Tool

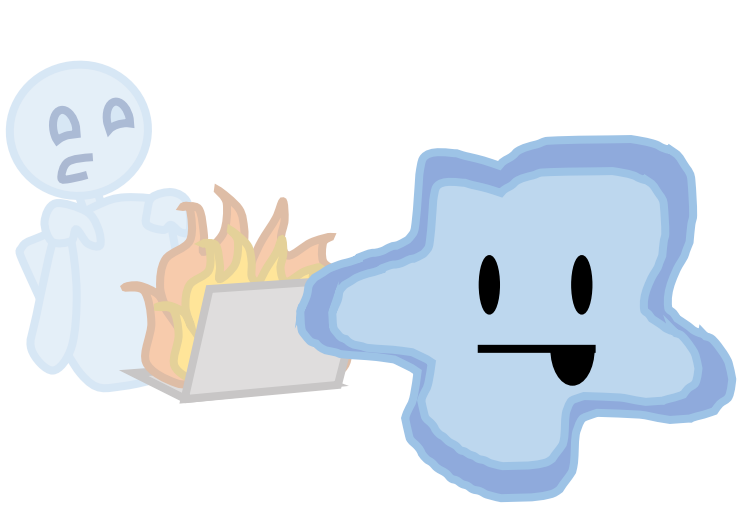
Alleviating Performance Problems



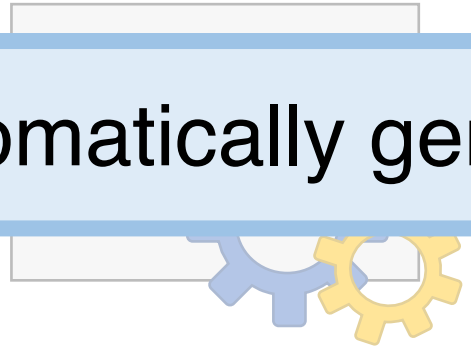
Pathological Input

Profiling Tool

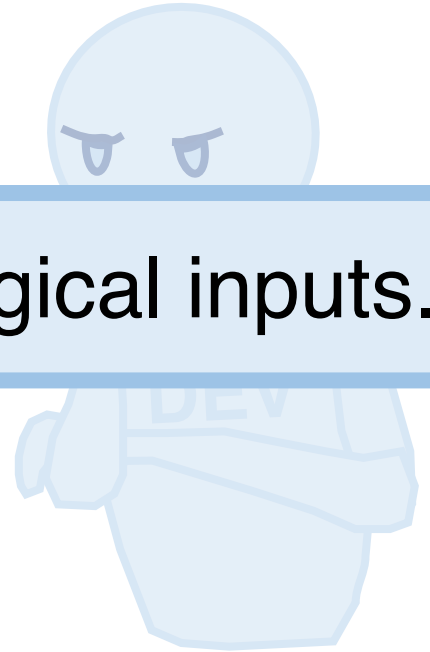
PerfFuzz Goal



Pathological Input



Profiling Tool



Automatically generate pathological inputs.

PerfFuzz

- A **feedback-directed mutational fuzzing** tool
- Uses **performance feedback** to produce pathological inputs

PerfFuzz

- A **feedback-directed mutational fuzzing** tool
  - **Fuzzing**: sends inputs to program
  - **Mutational**: creates new inputs by mutating saved inputs
  - **Feedback-directed**: saves inputs if program gives *interesting* feedback

PerfFuzz

- A **feedback-directed mutational fuzzing** tool
- Uses **performance feedback** to produce pathological inputs
  - **First idea**: interesting if longer execution time, path length [1]
  - **PerfFuzz**: interesting if higher execution count of any given CFG edge

[1] T. Petsios, J. Zhao, A. D. Keromytis, and S. Jana. 2017. SlowFuzz: Automated Domain-Independent Detection of Algorithmic Complexity Vulnerabilities. CCS '17.

Example Program: Word Frequency (`wf`)

- Count # occurrences of words in a string

  input:

```
the quick brown the dog
```

  output:

```
brown: 1
dog: 1
quick: 1
the: 2
```

- `wf` shipped with Fedora Linux had real performance bugs

Example Program: Word Frequency (wf)

- Count # occurrences of words in a string

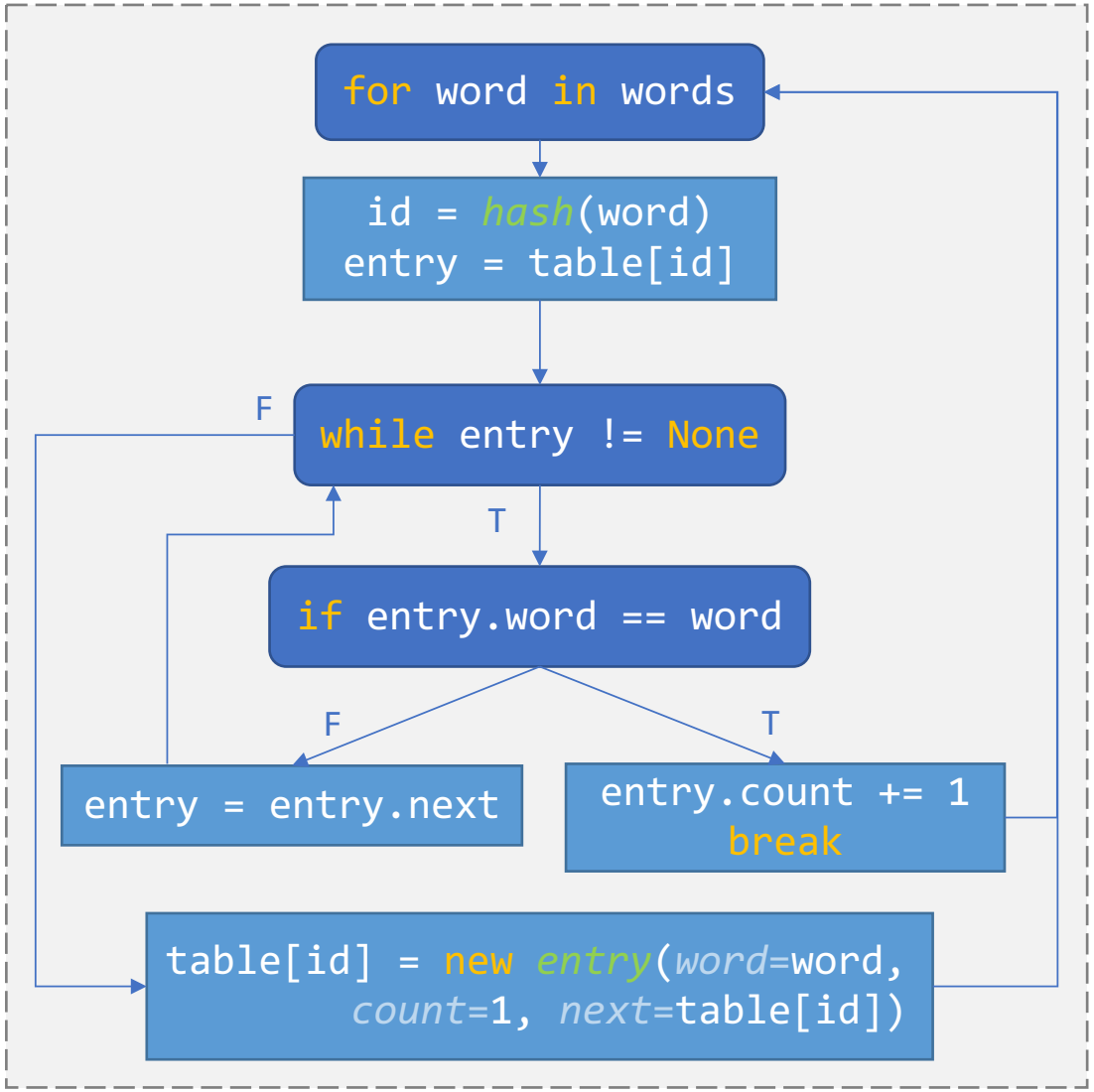
input:

```
the quick brown the dog
```

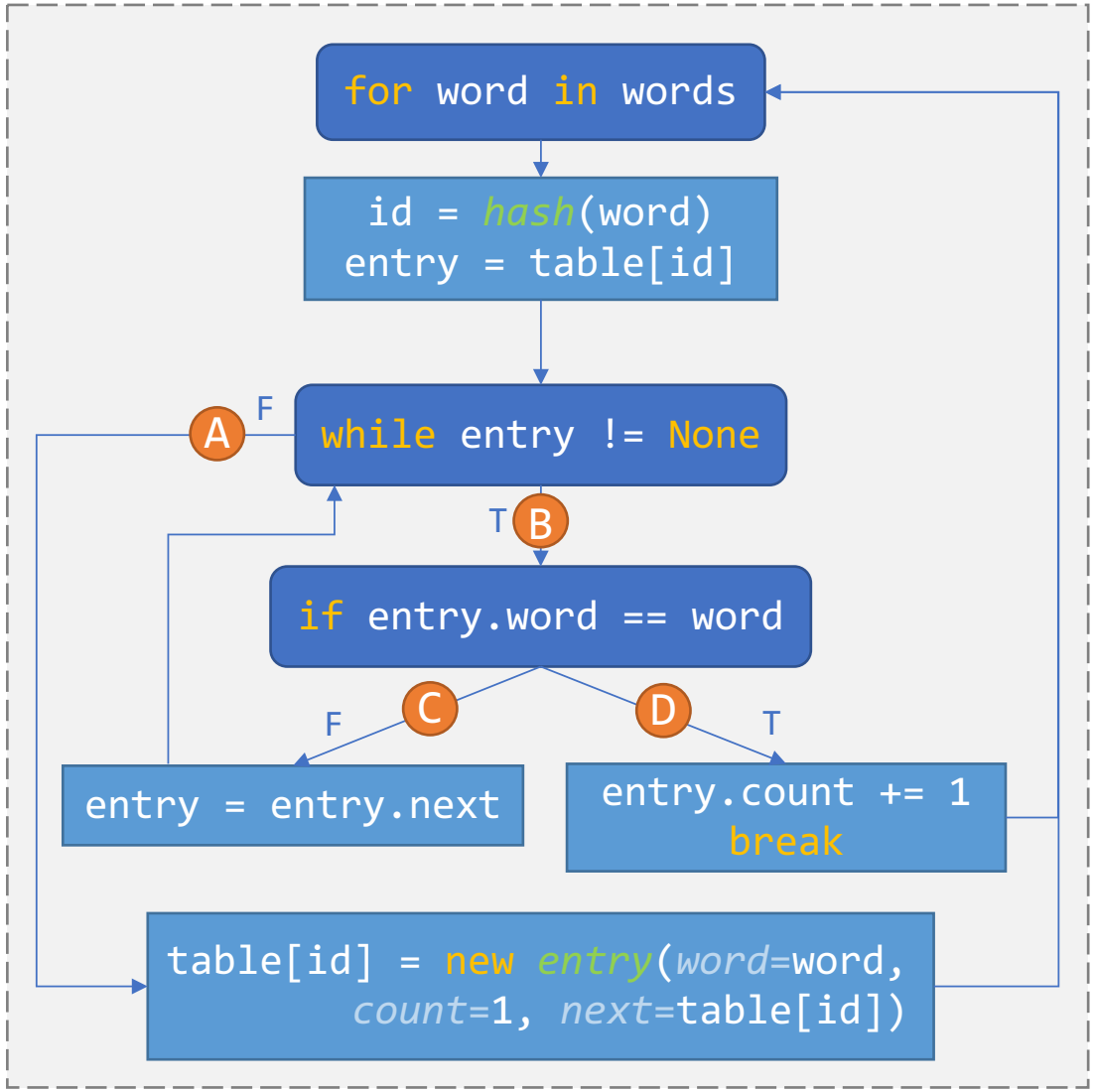
output:

```
brown: 1
dog: 1
quick: 1
the: 2
```

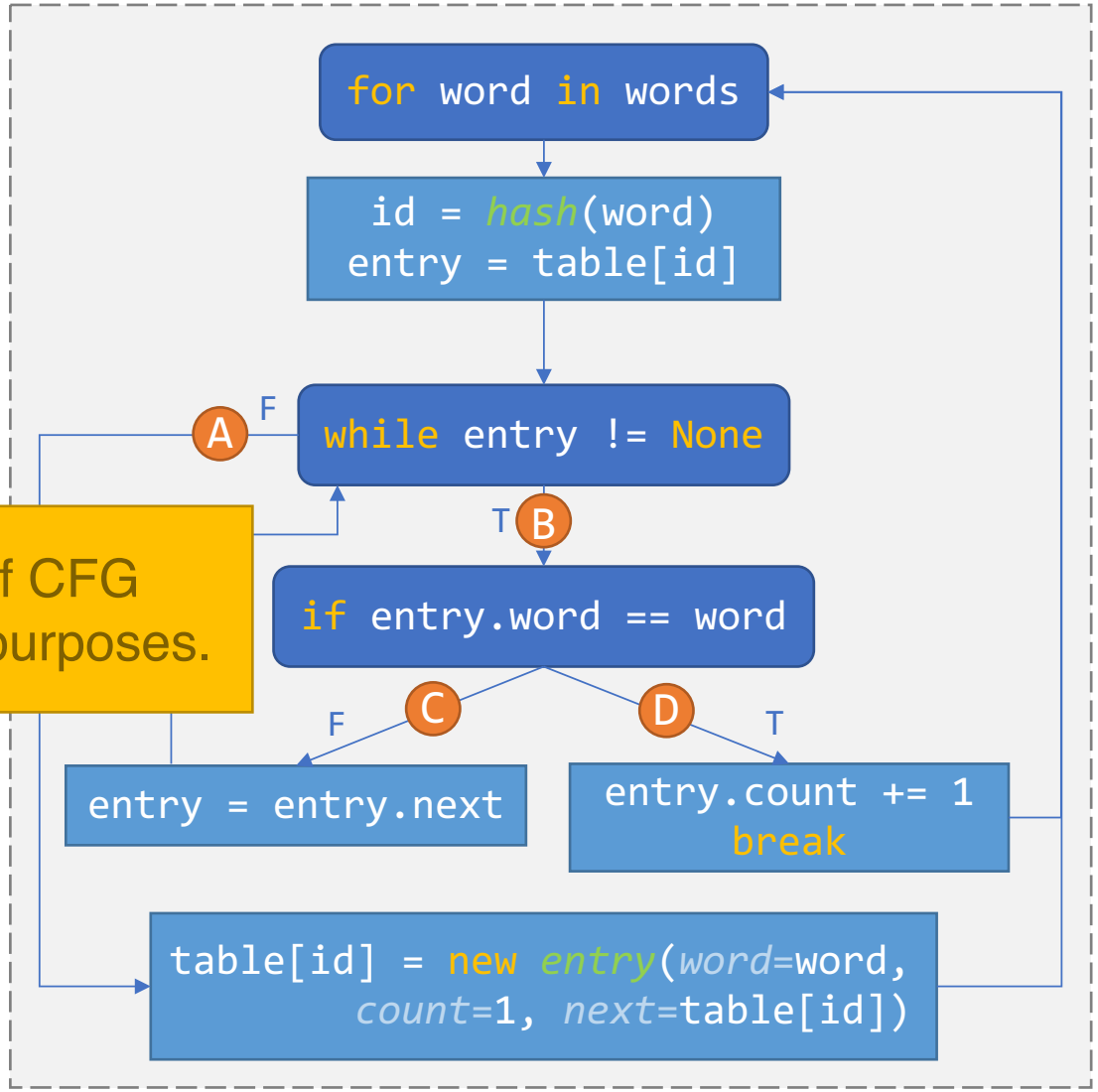
- wf shipped with Fedora Linux had real performance bugs



wf Performance Response



wf Performance Response



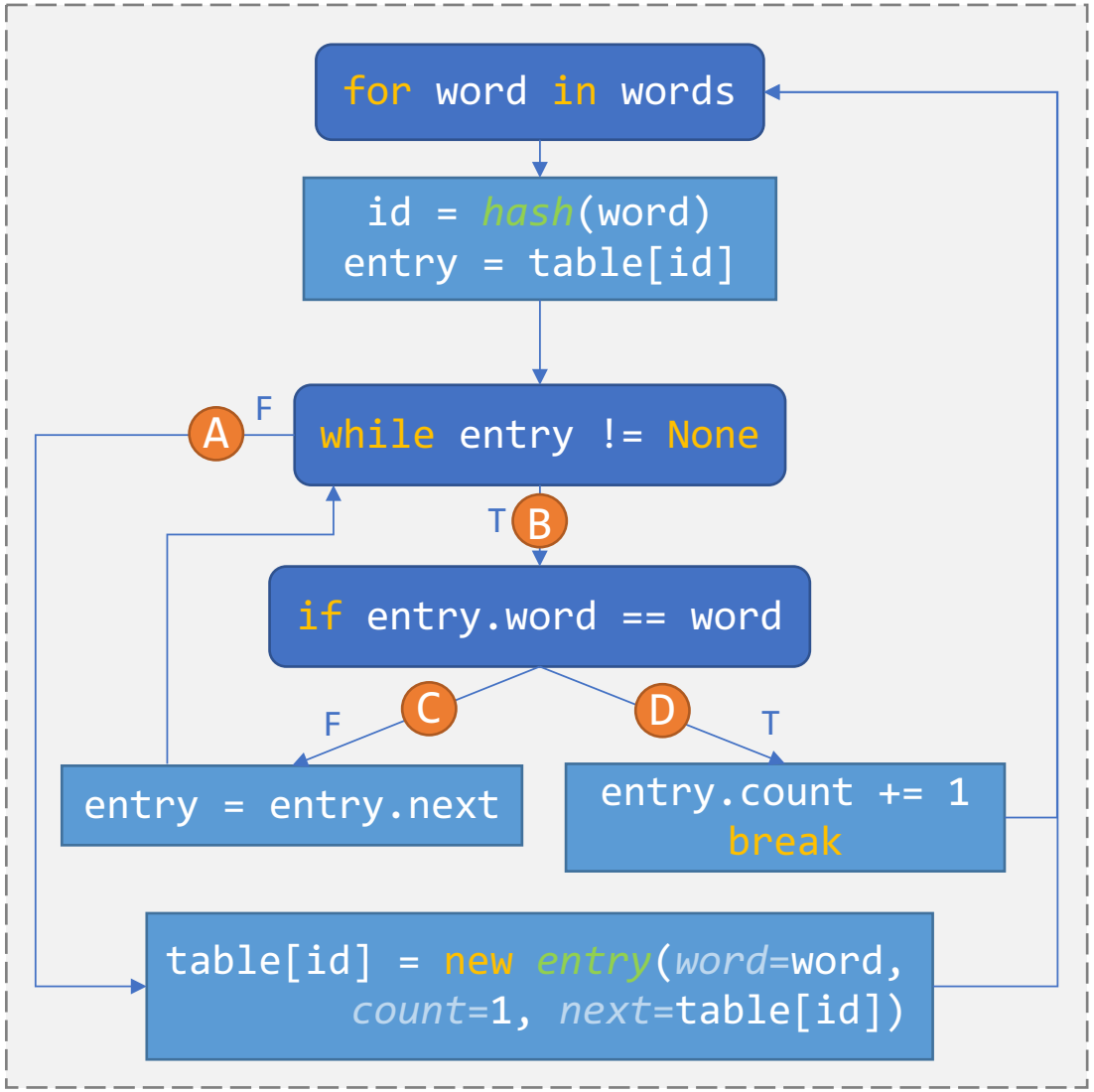
→ We look at a subset of CFG edges for illustration purposes.

wf Performance Response

- Usual case:

```
the quick brown the dog
```

Edge	# Hits
A	
B	
C	
D	

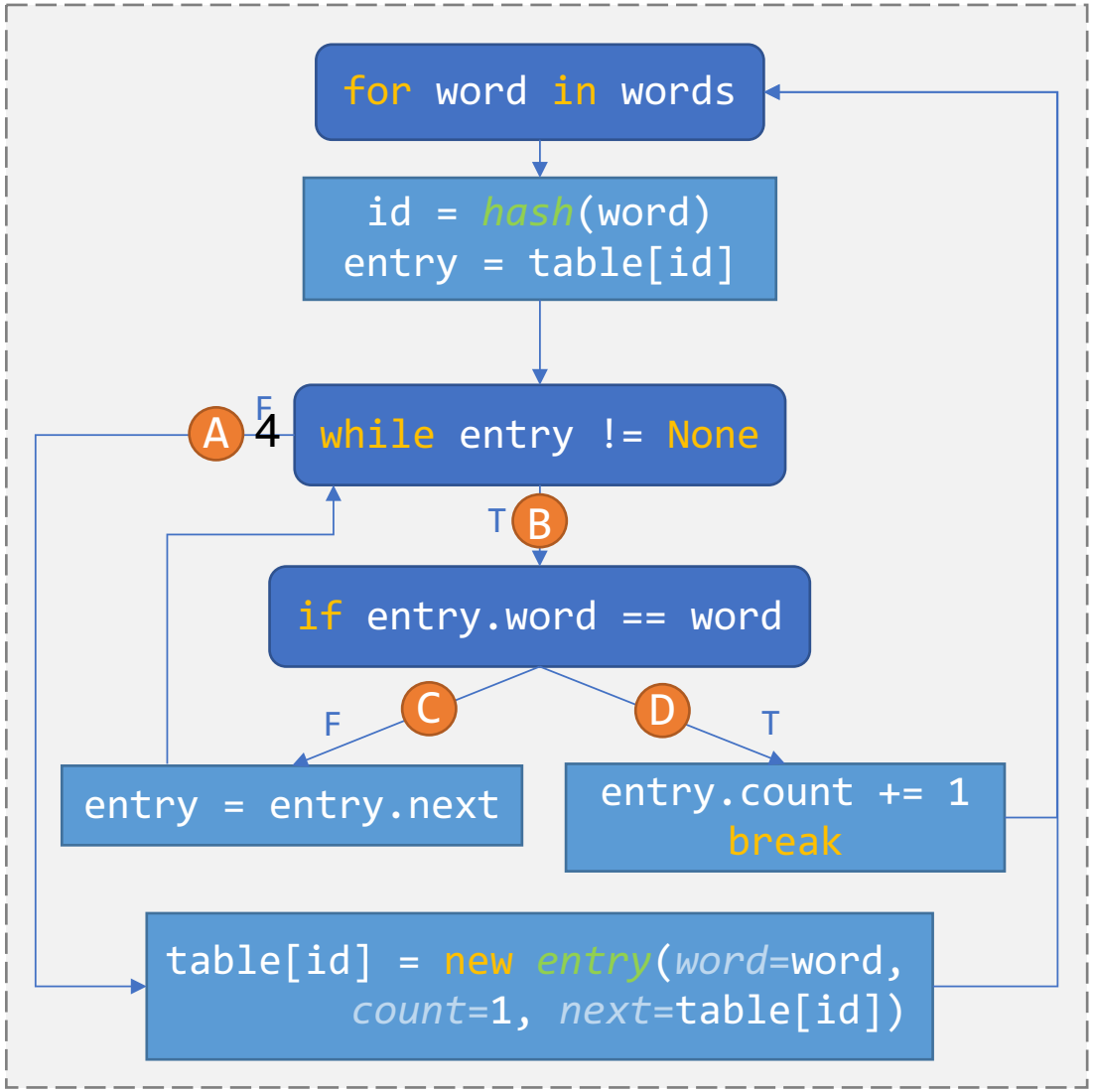


wf Performance Response

- Usual case:

```
the quick brown the dog
```

Edge	# Hits
A	4
B	
C	
D	

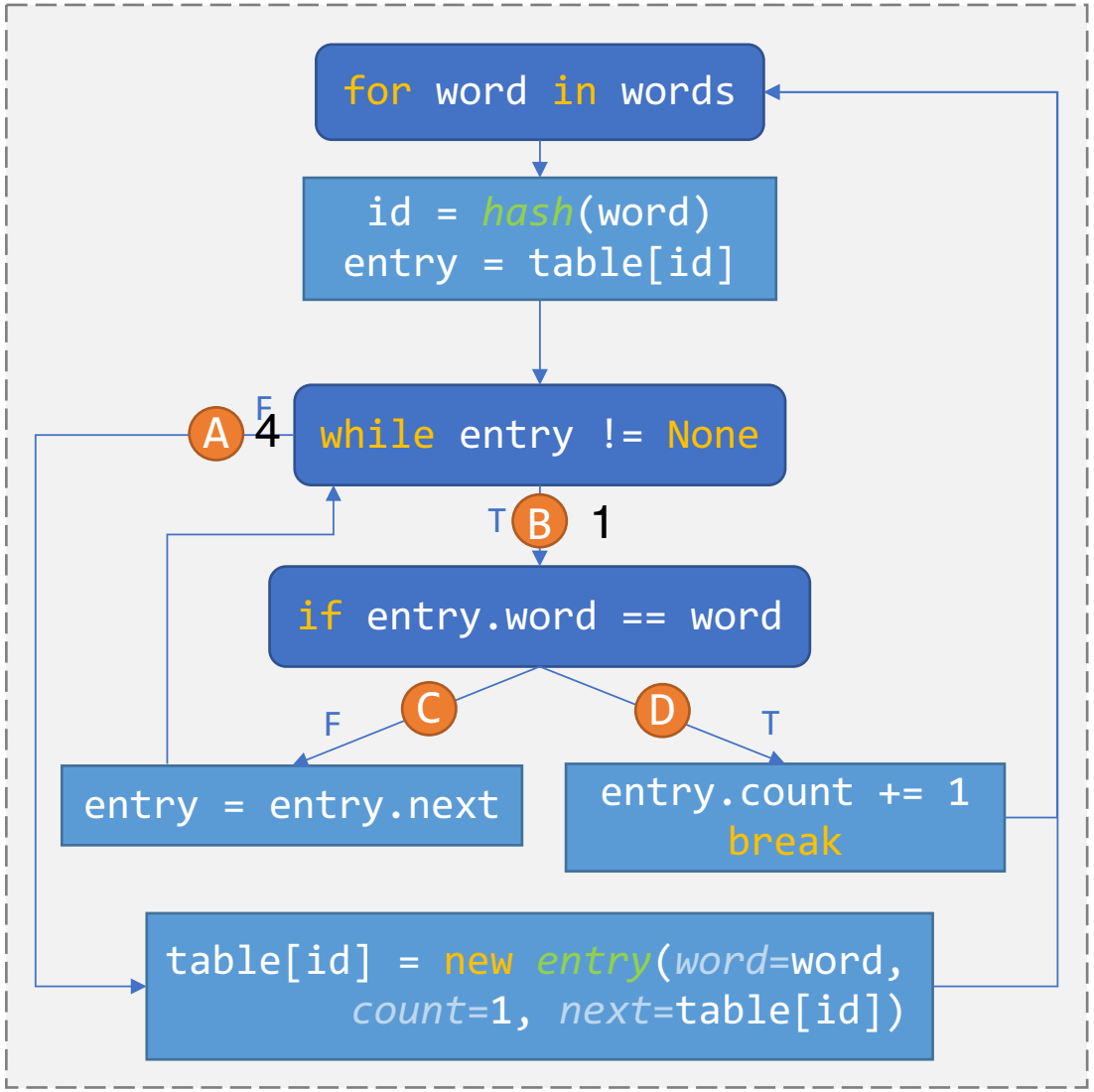


wf Performance Response

- Usual case:

```
the quick brown the dog
```

Edge	# Hits
A	4
B	1
C	
D	

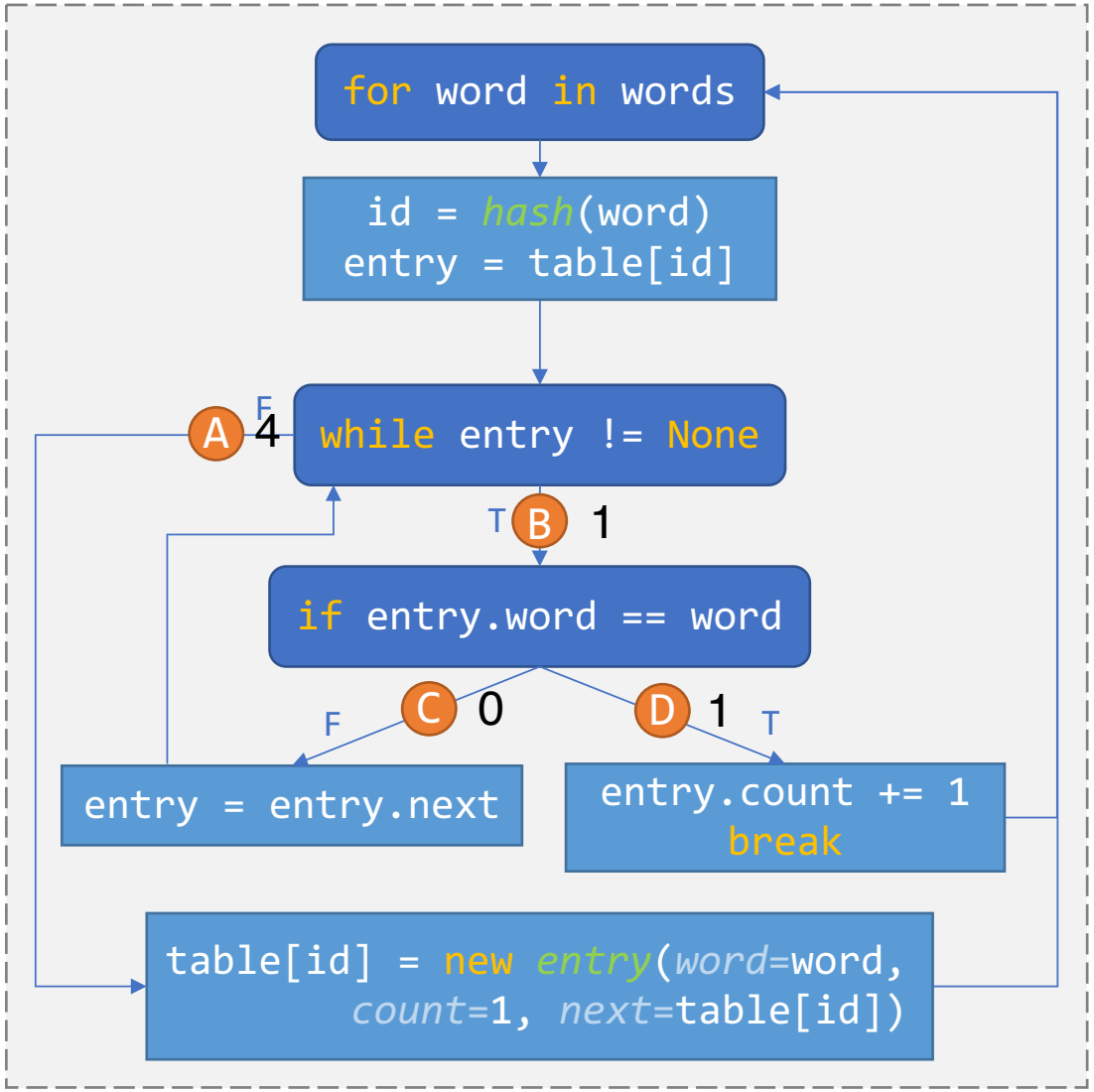


wf Performance Response

- Usual case:

```
the quick brown the dog
```

Edge	# Hits
A	4
B	1
C	0
D	1



wf Performance Response

- Usual case:

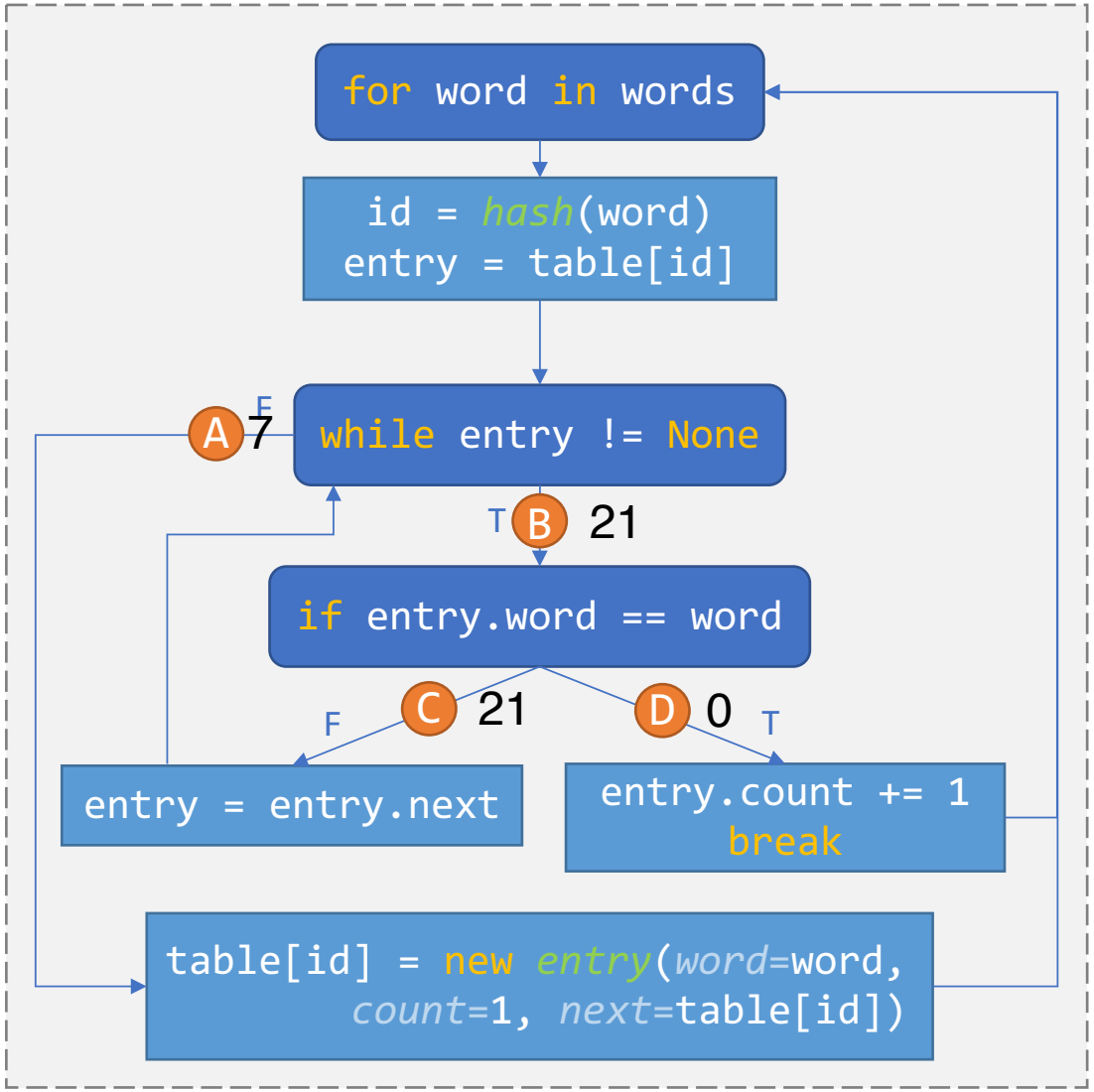
```
the quick brown the dog
```

Edge	# Hits
A	4
B	1
C	0
D	1

- Hash collisions:

```
t ?t xt at$ #a ))t Qwaa
```

Edge	# Hits
A	7
B	21
C	21
D	0



wf Performance Response

- Usual case:

```
the quick brown the dog
```

Edge	# Hits
A	4
B	1
C	0
D	1

- Hash collisions:

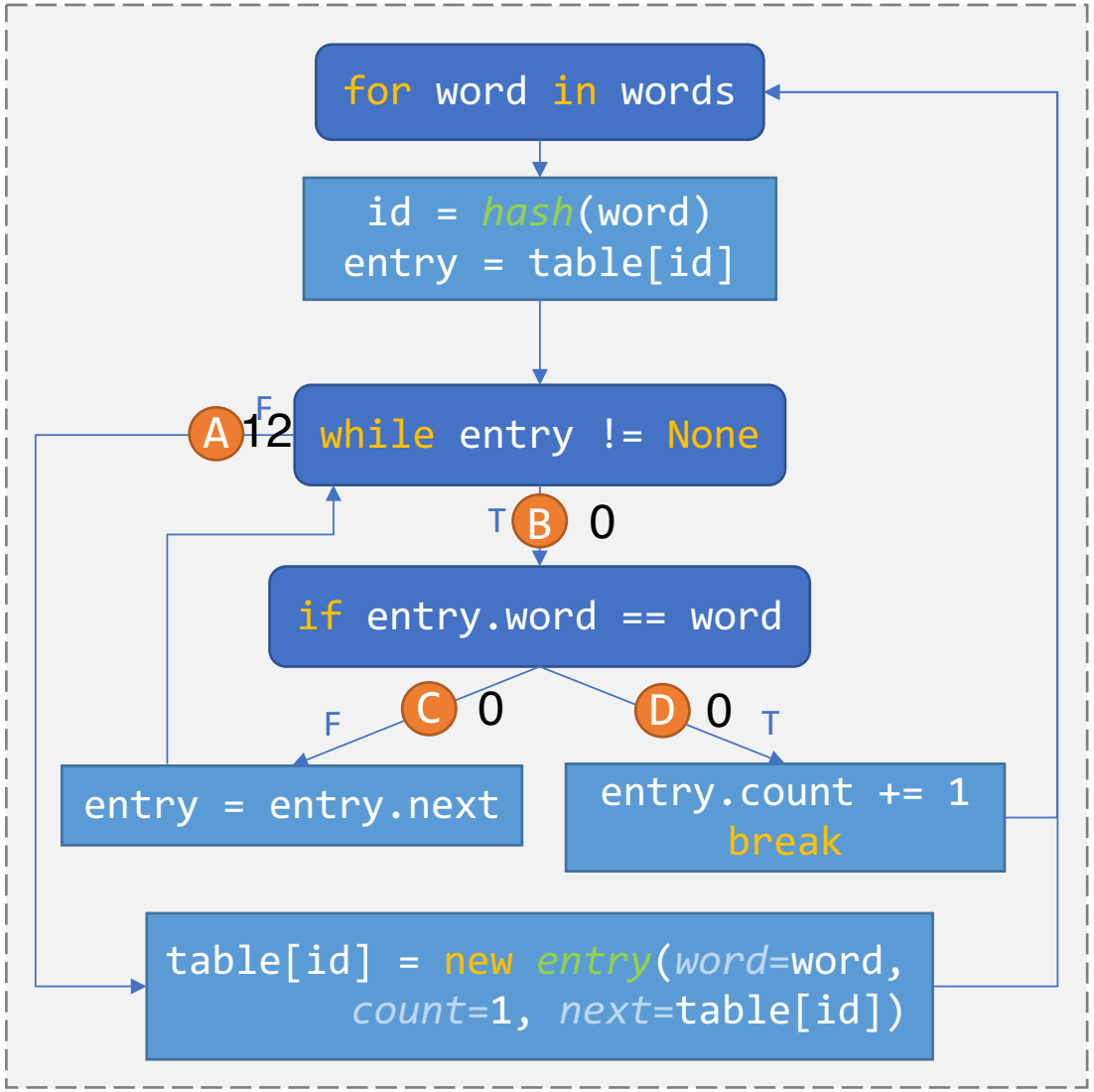
```
t ?t xt at$ #a ))t Qwaa
```

Edge	# Hits
A	7
B	21
C	21
D	0

- Small words:

```
t h e q u i c k b r o w
```

Edge	# Hits
A	12
B	0
C	0
D	0



wf Performance Response

- Usual case:

```
the quick brown the dog
```

Edge	# Hits
A	4
B	1
C	0
D	1

- Hash collisions:

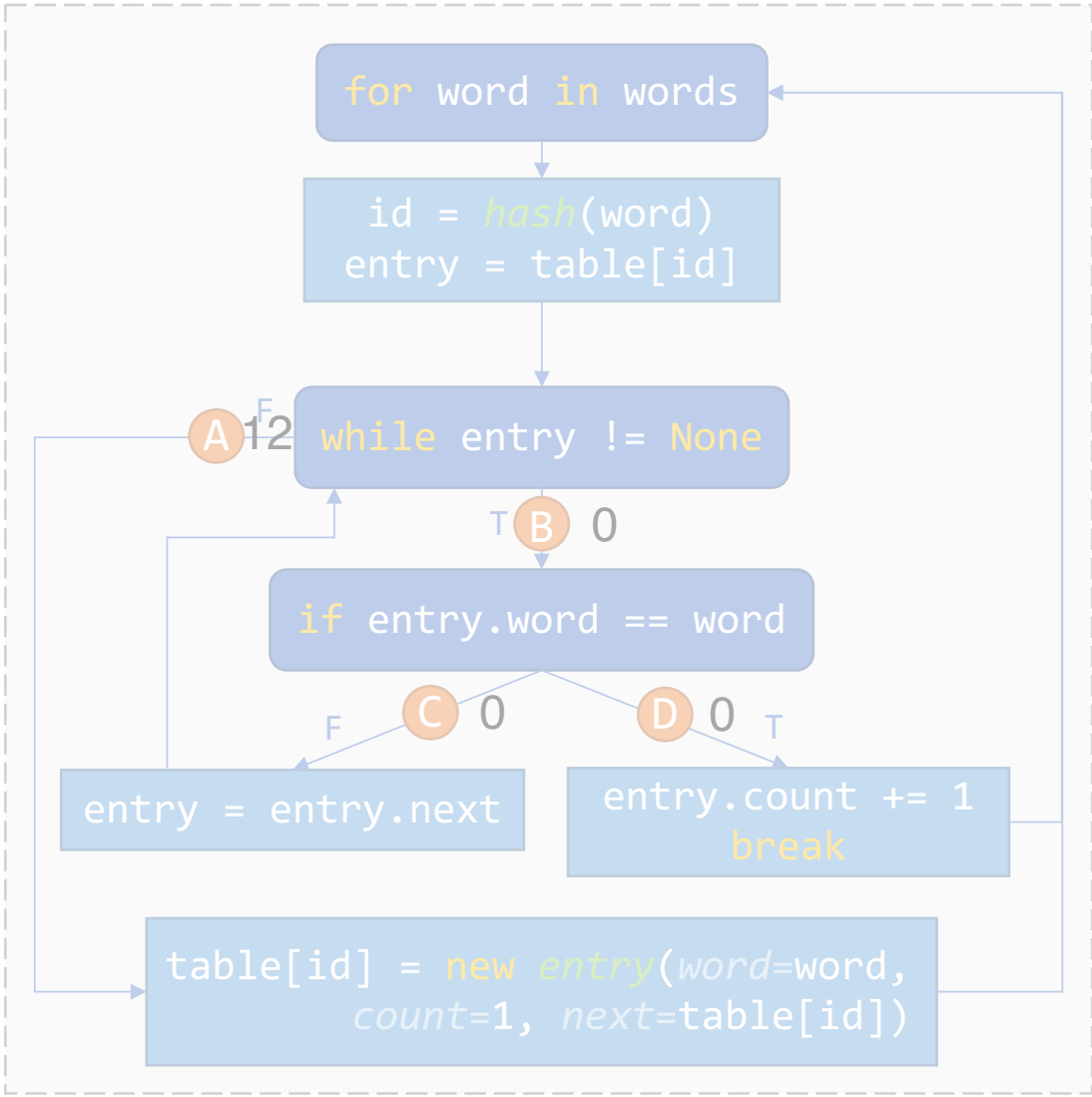
```
t ?t xt at$ #a ))t Qwaa
```

Edge	# Hits
A	7
B	21
C	21
D	0

- Small words:

```
t h e q u i c k b r o w
```

Edge	# Hits
A	12
B	0
C	0
D	0



wf Performance Response

- Usual case:

```
the quick brown the dog
```

Edge	# Hits
A	4
B	1
C	0
D	1

- Hash collisions:

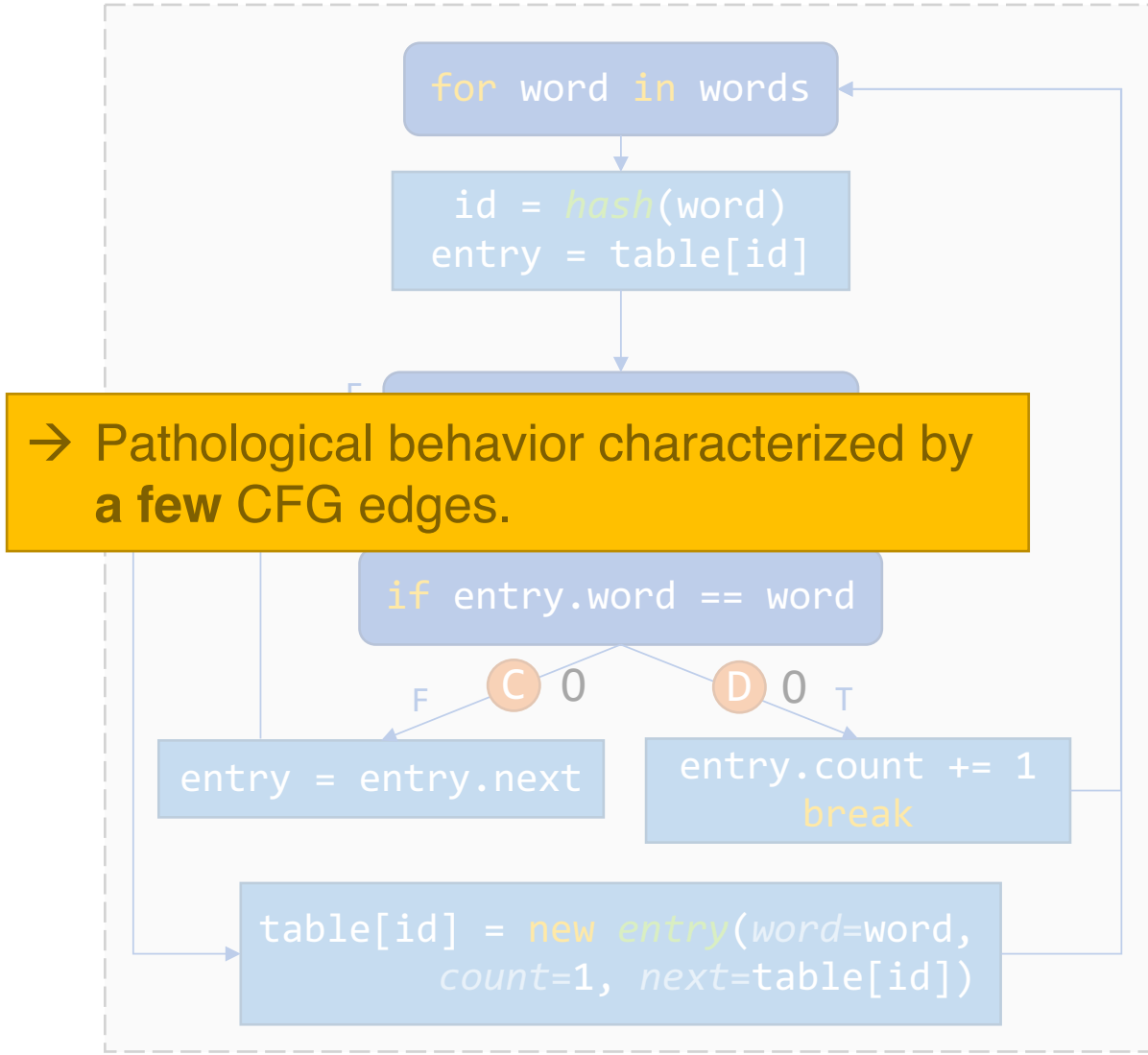
```
t ?t xt at$ #a ))t Qwaa
```

Edge	# Hits
A	7
B	21
C	21
D	0

- Small words:

```
t h e q u i c k b r o w
```

Edge	# Hits
A	12
B	0
C	0
D	0



wf Performance Response

- Usual case:

```
the quick brown the dog
```

Edge	# Hits
A	4
B	1
C	0
D	1

- Hash collisions:

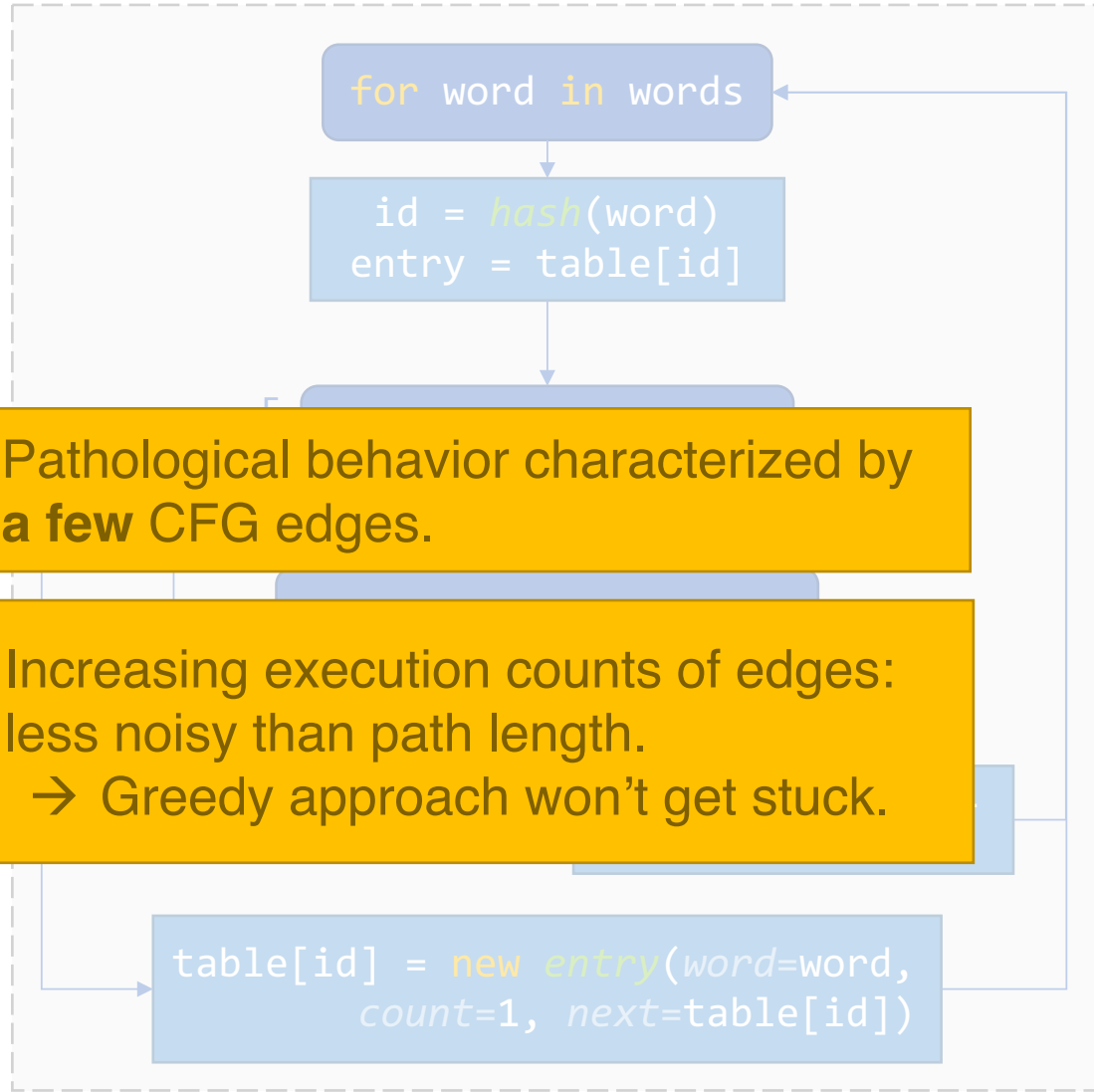
```
t ?t xt at$ #a ))t Qwaa
```

Edge	# Hits
A	7
B	21
C	21
D	0

- Small words:

```
t h e q u i c k b r o w
```

Edge	# Hits
A	12
B	0
C	0
D	0



→ Pathological behavior characterized by **a few** CFG edges.

→ Increasing execution counts of edges: less noisy than path length.
→ Greedy approach won't get stuck.

PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input

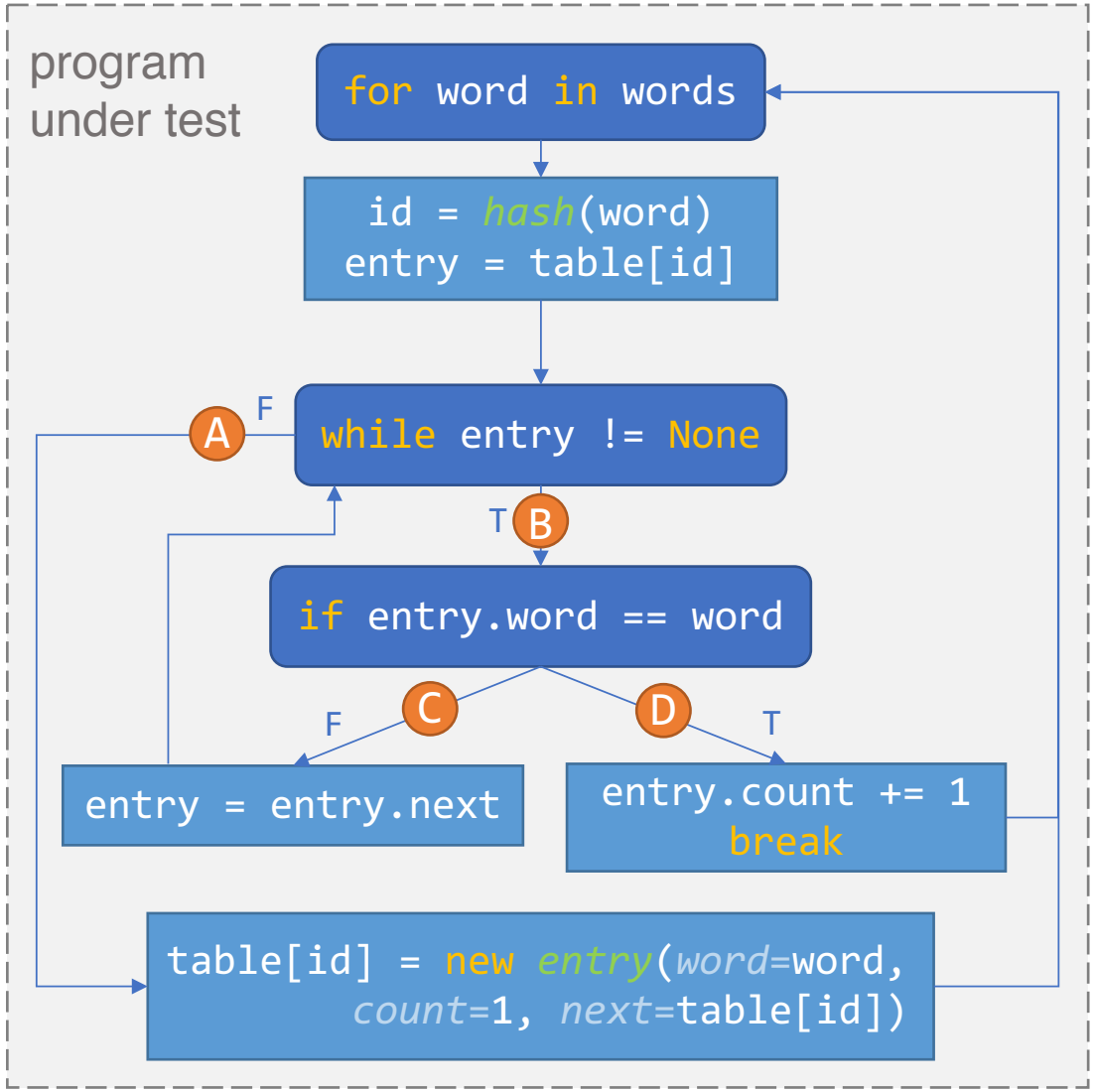
mutation engine

Parent to Mutate

input running, feedback analysis

Current Input

Edge	# Hits



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input

mutation engine

Parent to Mutate

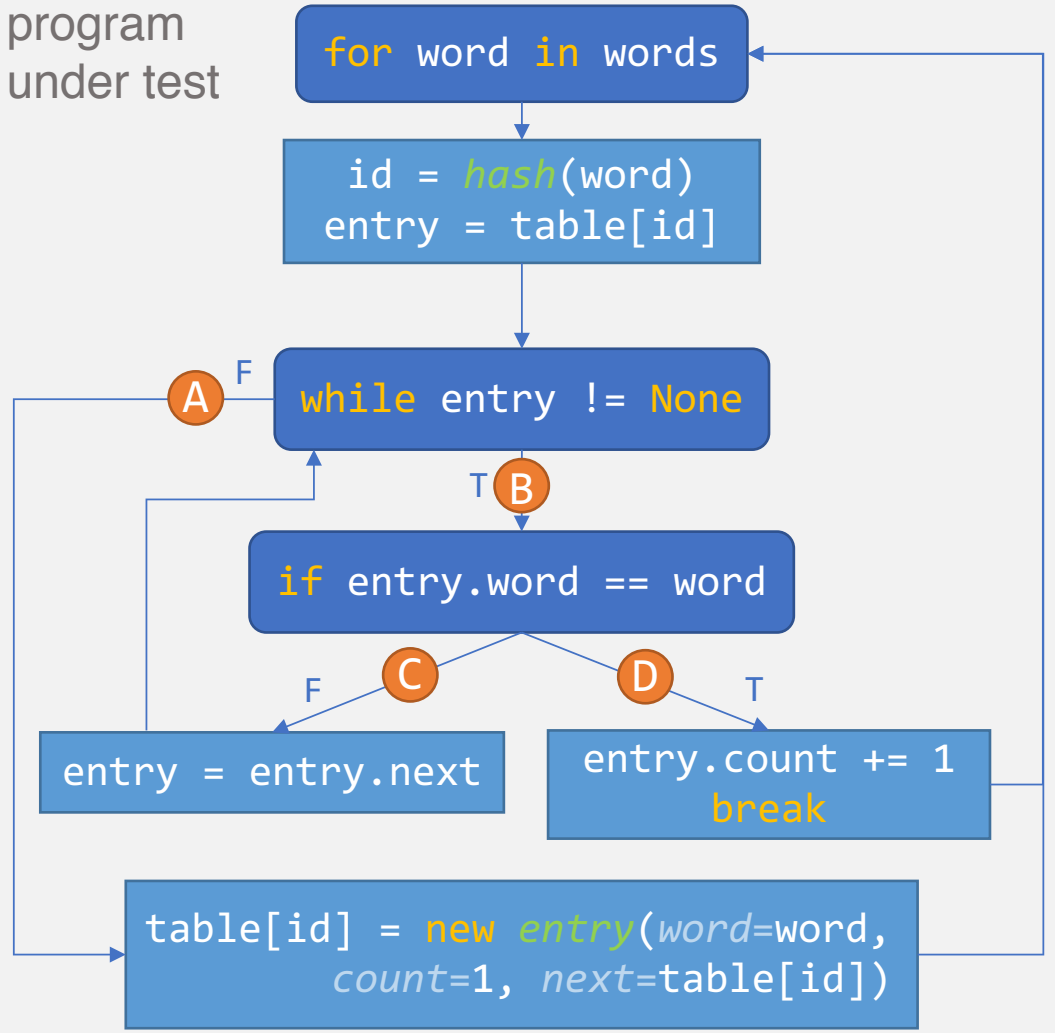
input running, feedback analysis

seed input

Current Input
the quick brown the dog

Edge	# Hits

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input

mutation engine

Parent to Mutate

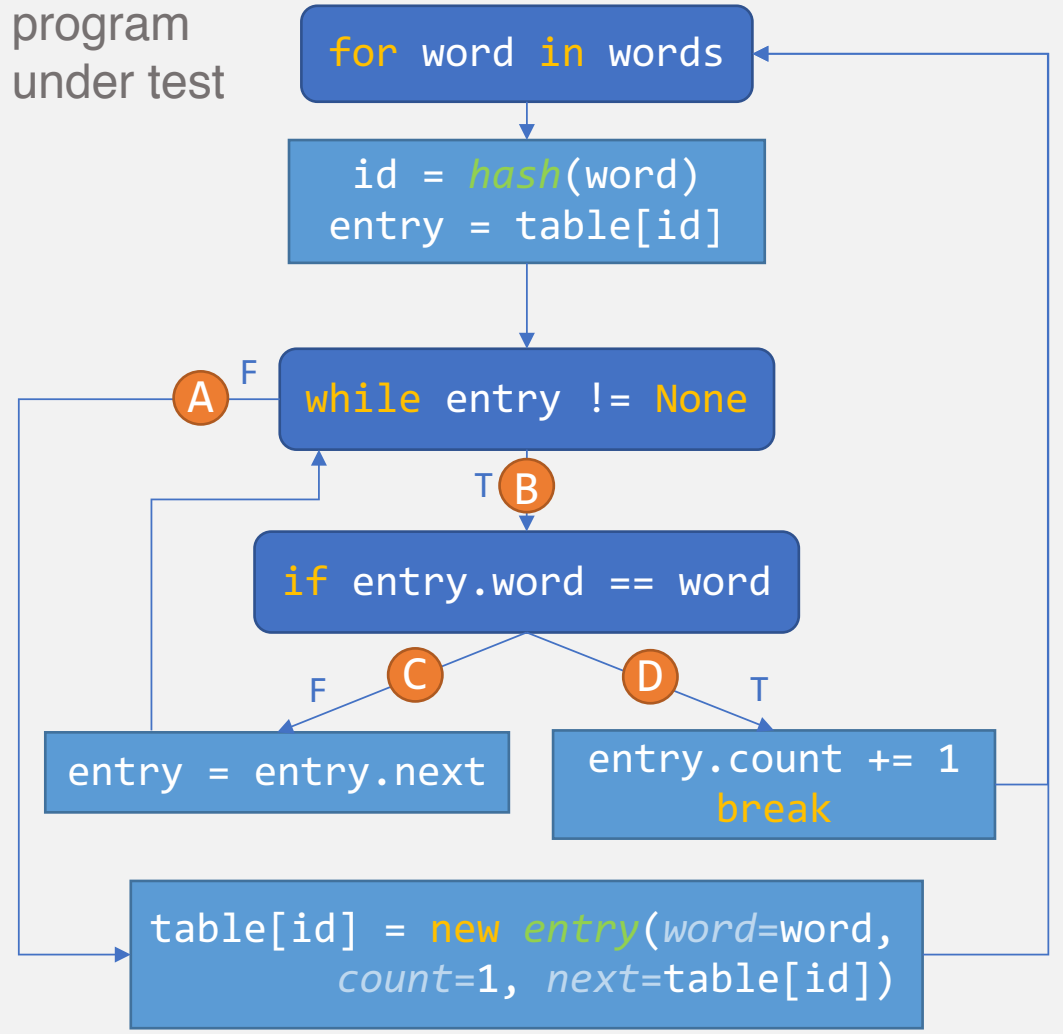
input running, feedback analysis

Current Input
the quick brown the dog

Edge	# Hits

input

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input

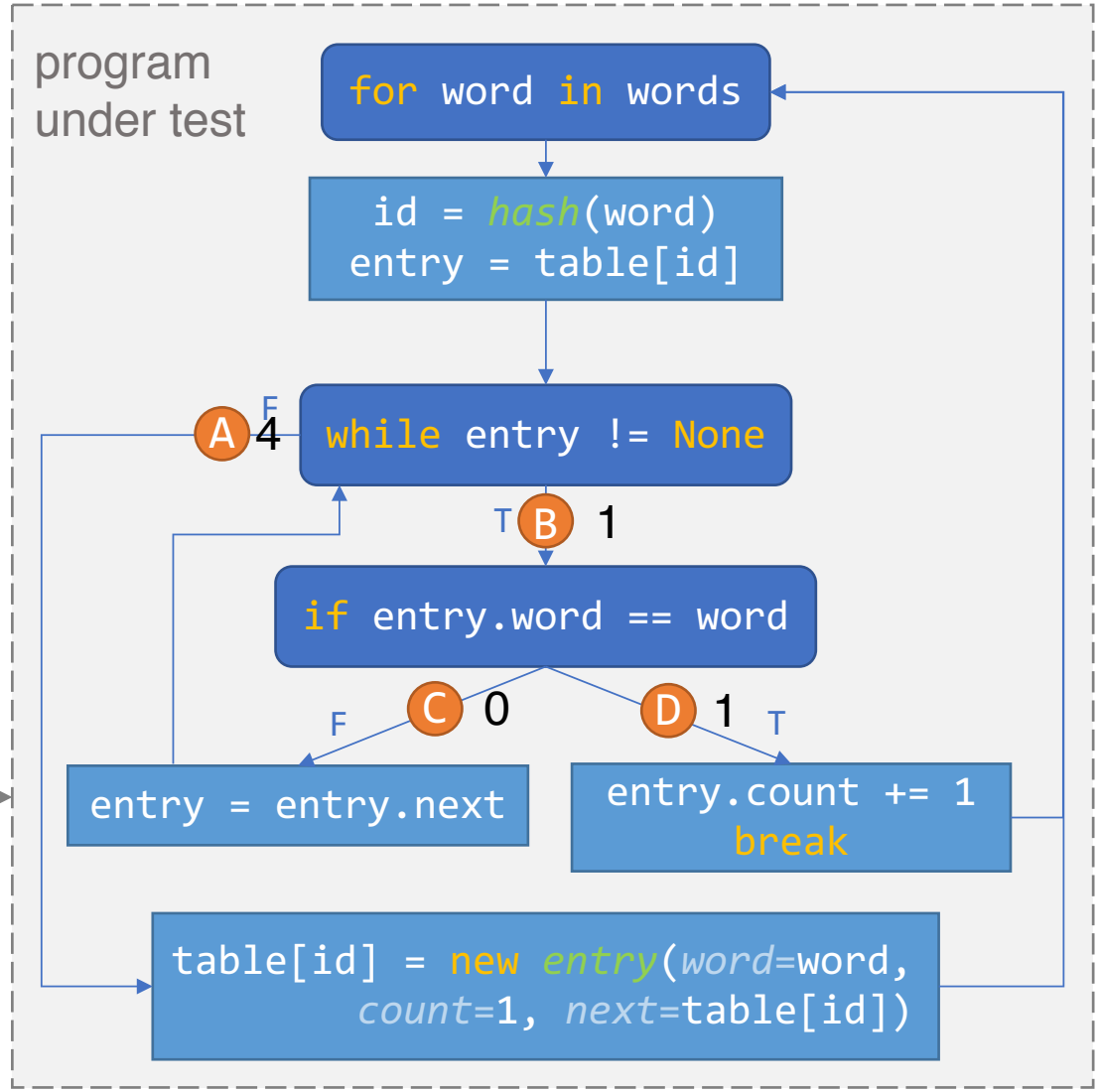
mutation engine

Parent to Mutate

input running, feedback analysis

Current Input
the quick brown the dog

Edge	# Hits



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input

mutation engine

Parent to Mutate

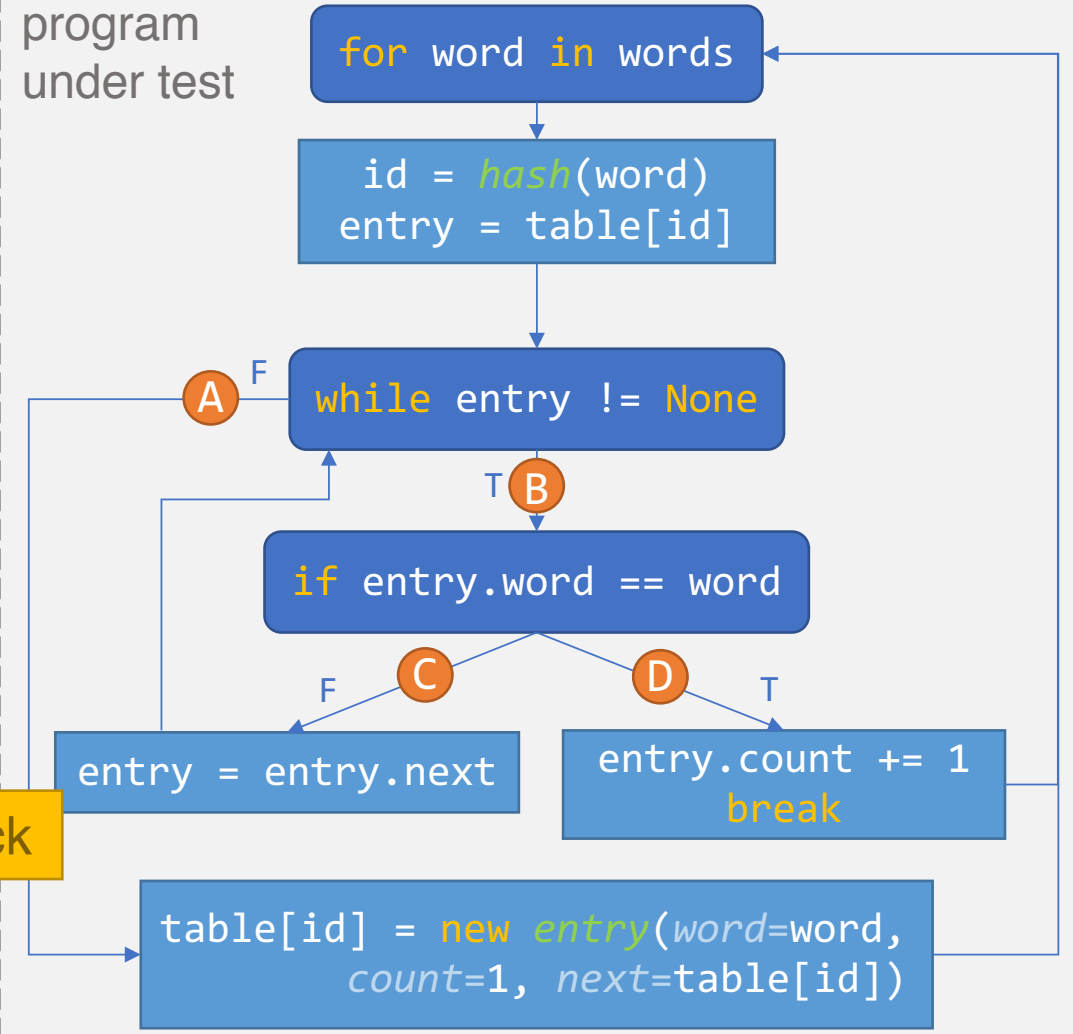
input running, feedback analysis

Current Input
the quick brown the dog

Edge	# Hits
A	4
B	1
D	1

feedback

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate

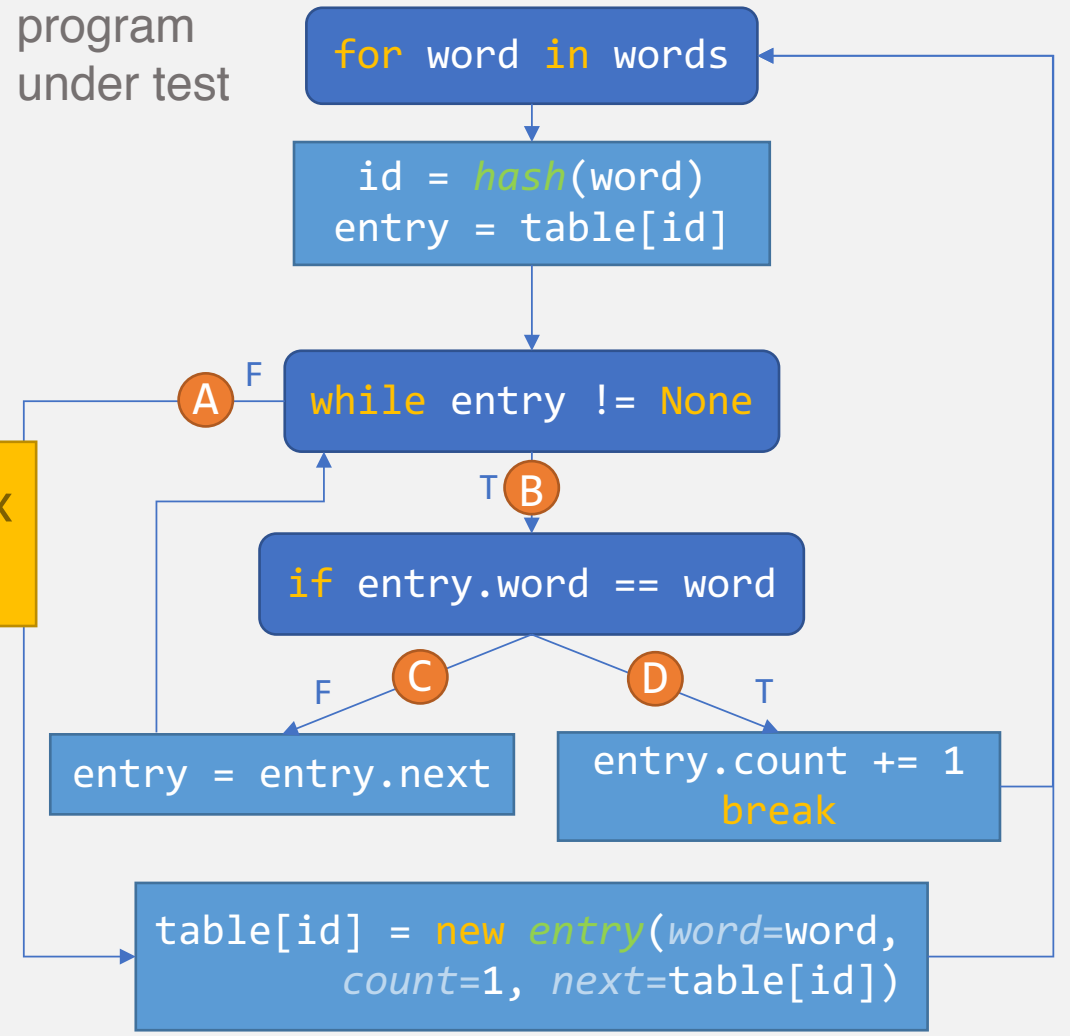
save new max for A, B, D

input running, feedback analysis

Current Input
the quick brown the dog

Edge	# Hits
A	4
B	1
D	1

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

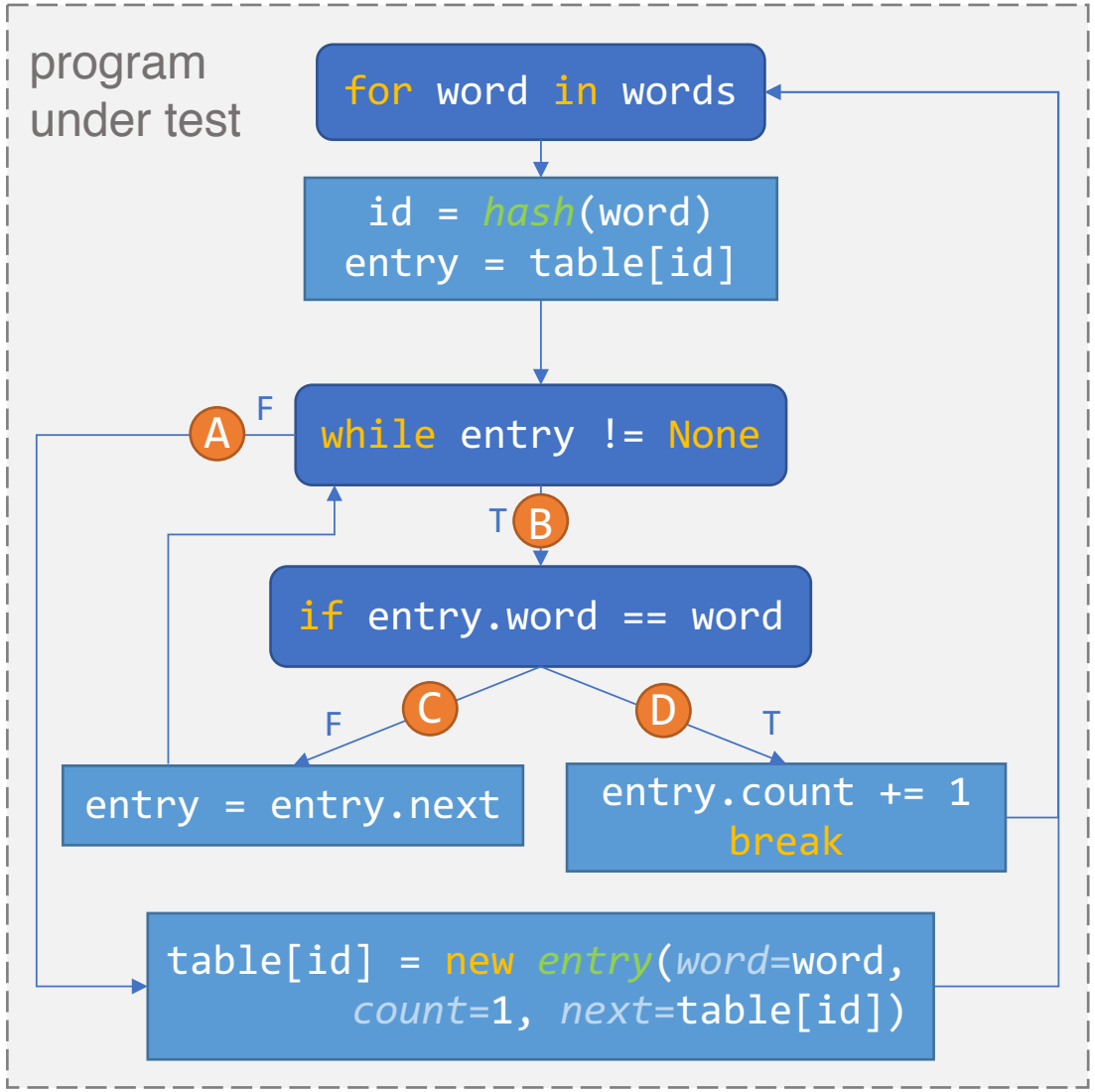
mutation engine

Parent to Mutate

input running, feedback analysis

Current Input

Edge	# Hits



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

choose parent

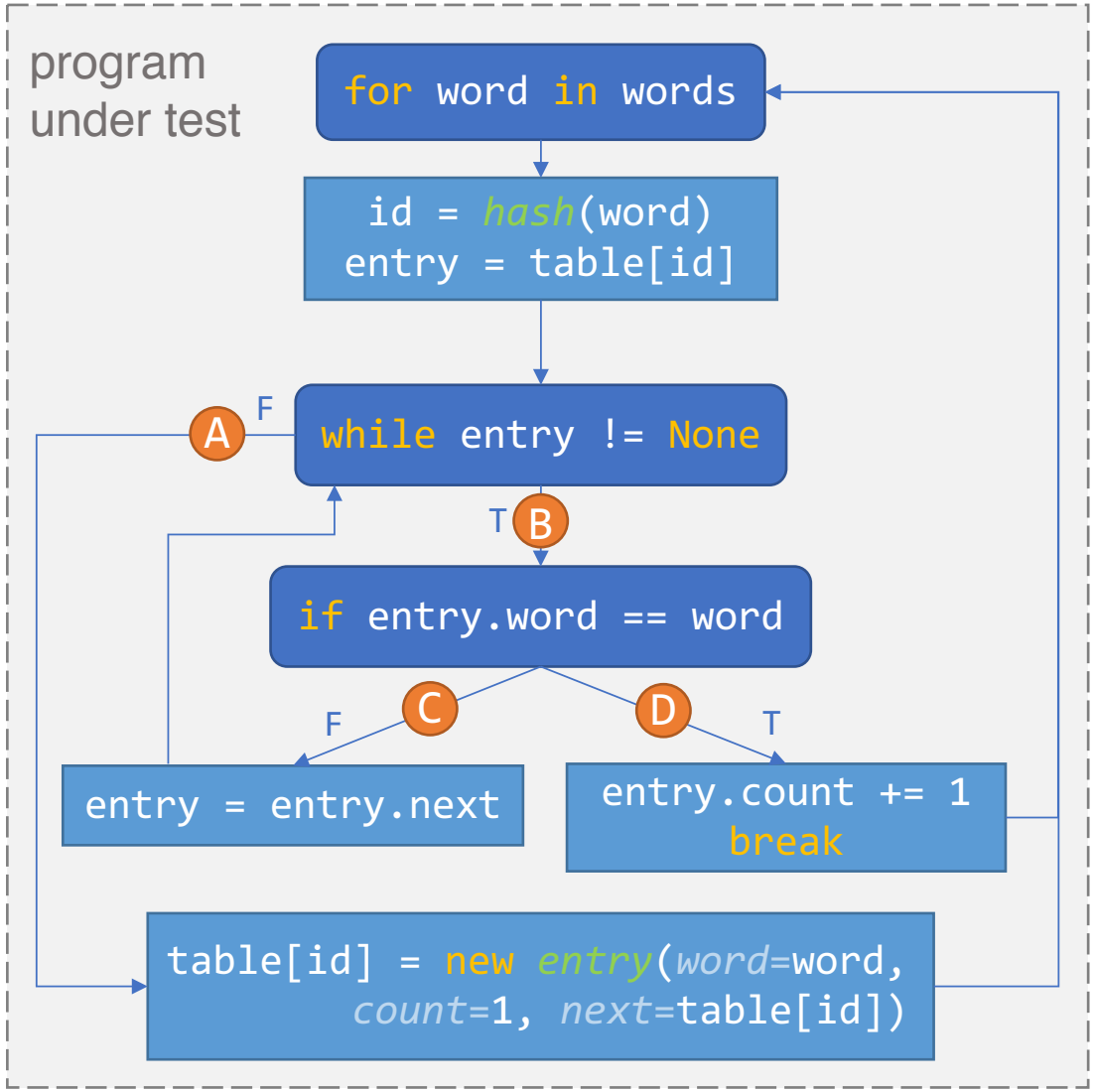
mutation engine

Parent to Mutate
the quick brown the dog

input running, feedback analysis

Current Input

Edge	# Hits



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

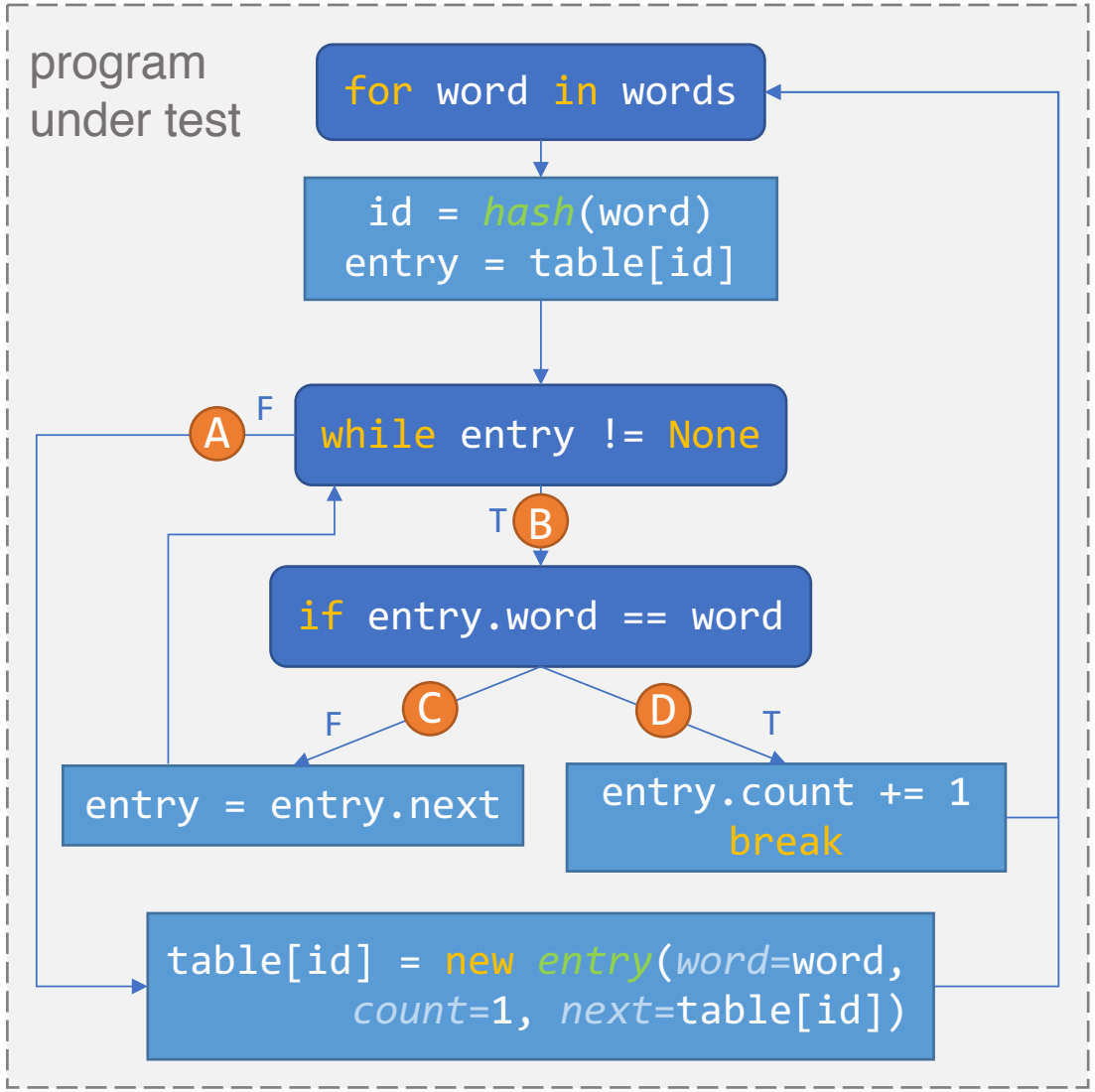
mutation engine

Parent to Mutate
the quick brown the dog

input running, feedback analysis

Current Input

Edge	# Hits



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

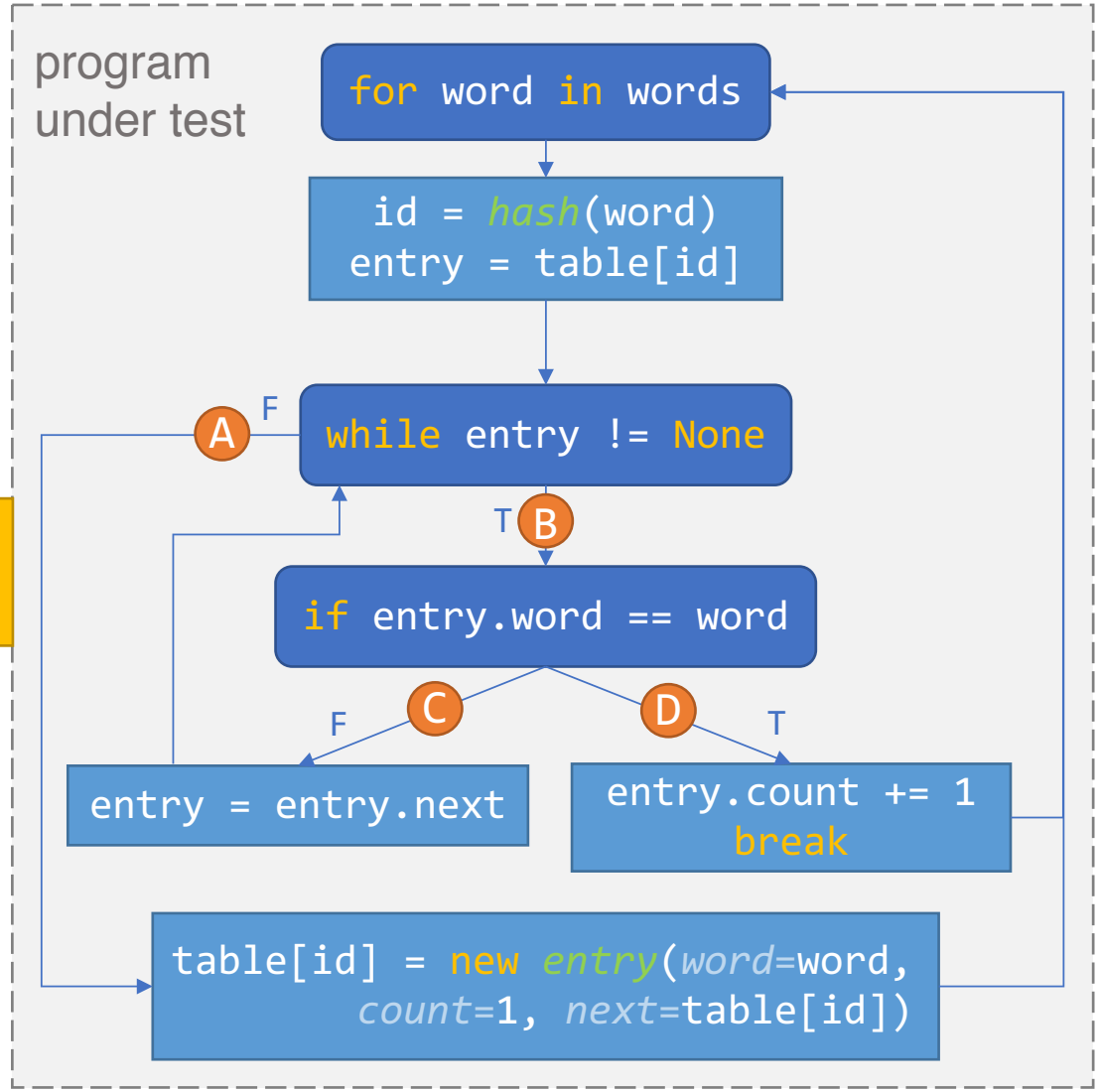
Parent to Mutate
the quick brown the dog

let's mutate this many times

input running, feedback analysis

Current Input

Edge	# Hits



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

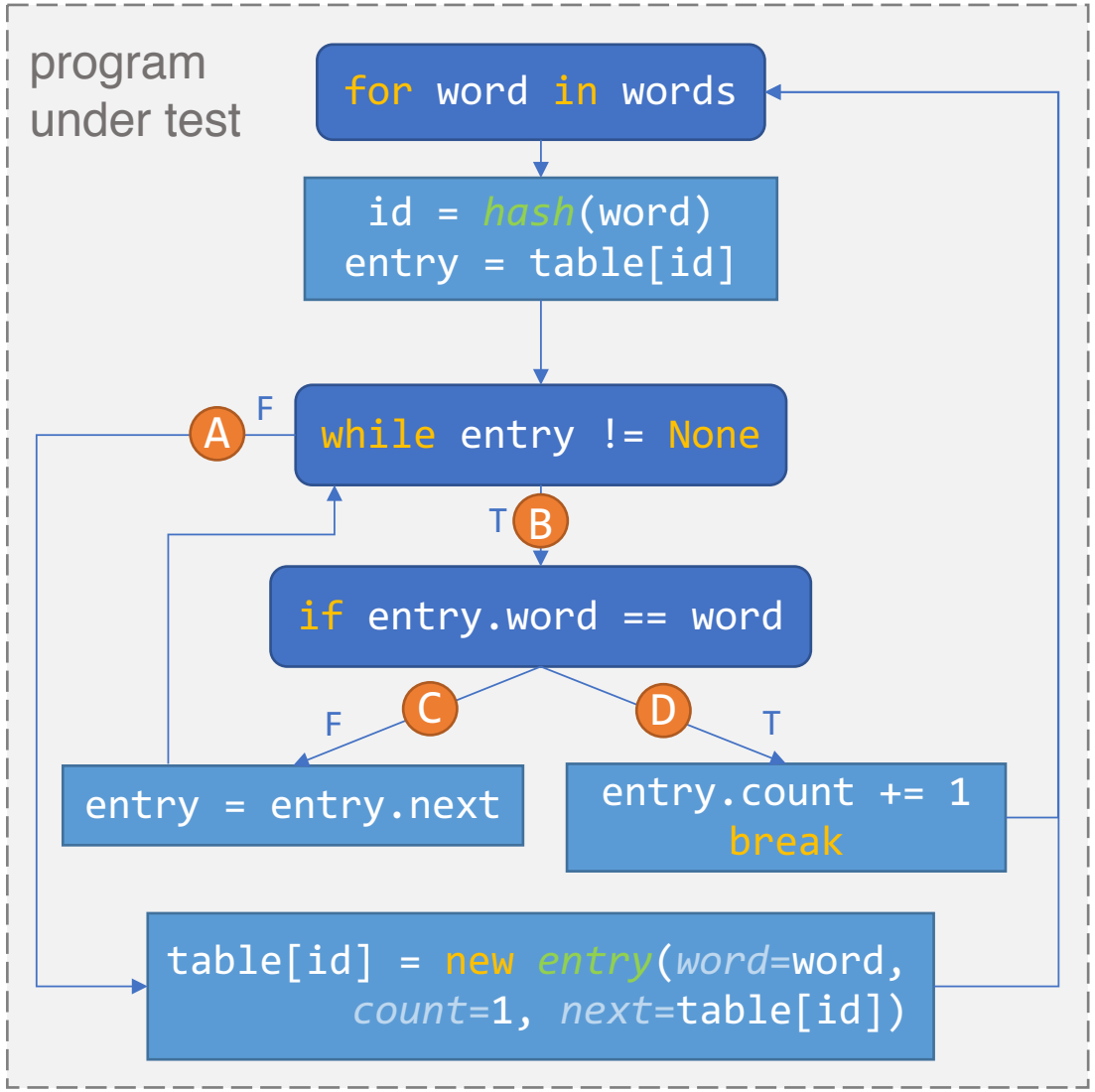
Parent to Mutate
the quick brown the dog

input running, feedback analysis

new mutant

Current Input
the quack brown the dog

Edge	# Hits



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

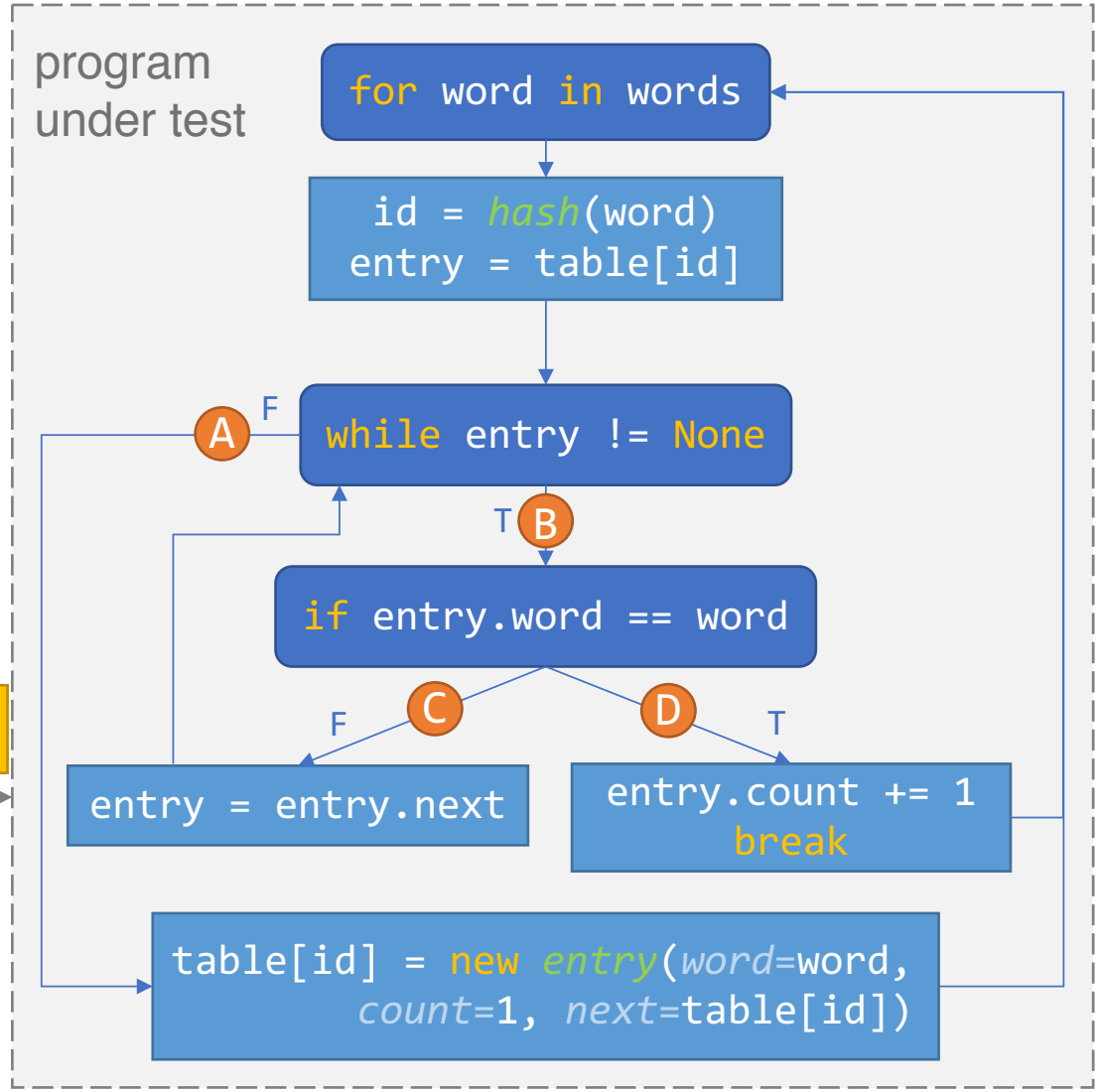
Parent to Mutate
the quick brown the dog

input running, feedback analysis

Current Input
the quack brown the dog

Edge	# Hits

input



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

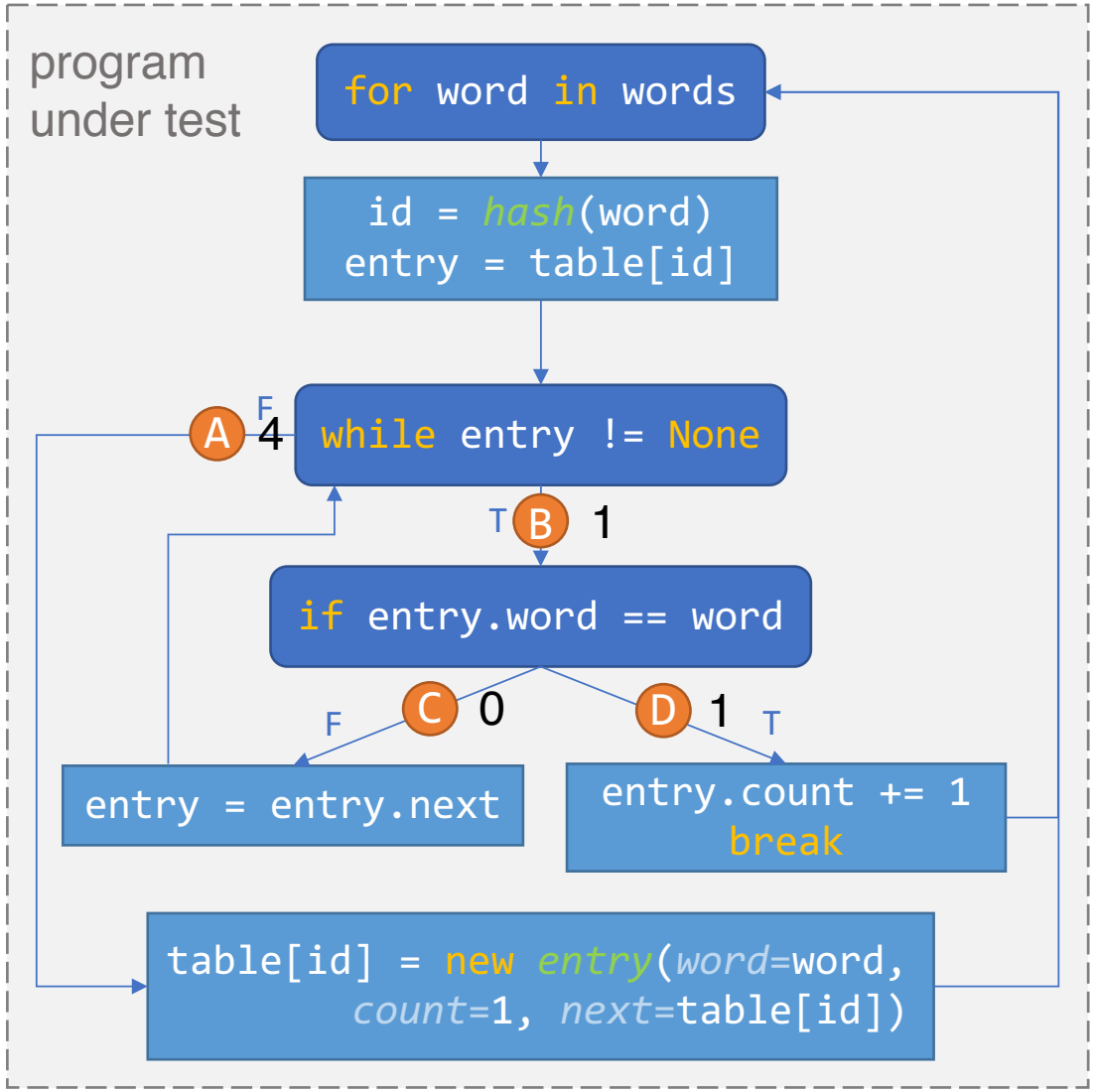
mutation engine

Parent to Mutate
the quick brown the dog

input running, feedback analysis

Current Input
the quack brown the dog

Edge	# Hits



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown the dog

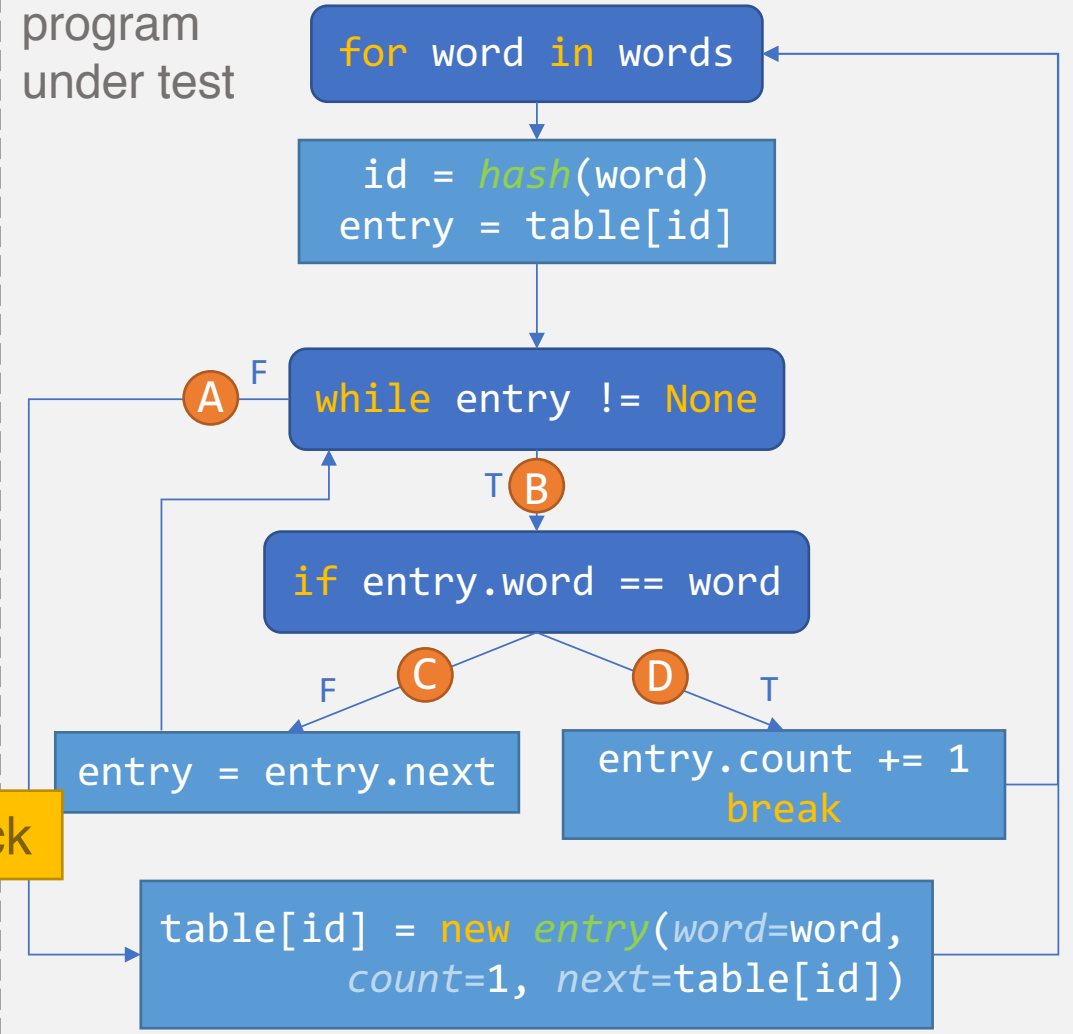
input running, feedback analysis

Current Input
the quack brown the dog

Edge	# Hits
A	4
B	1
D	1

feedback

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

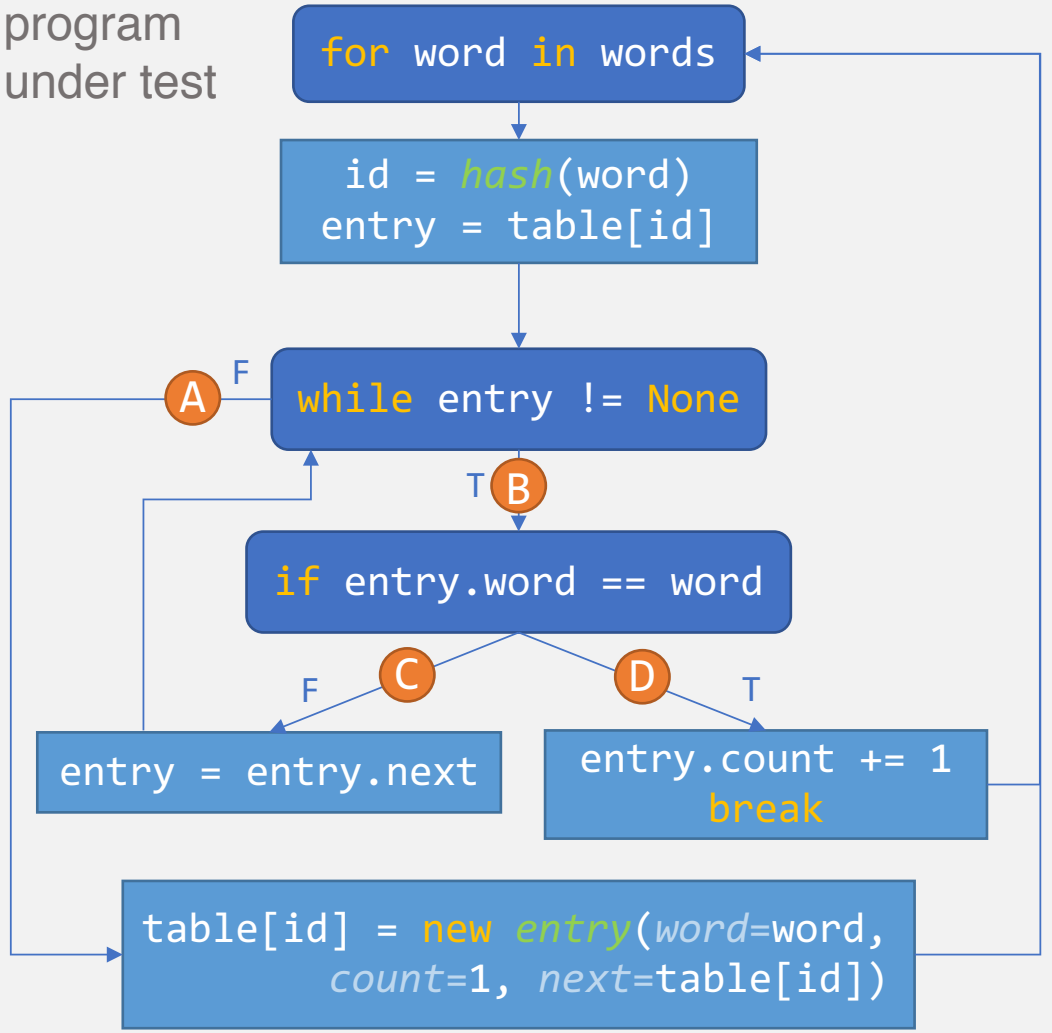
Parent to Mutate
the quick brown the dog

input running, feedback analysis

Current Input
the quack brown the dog

Edge	# Hits
A	4
B	1
D	1

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown the dog

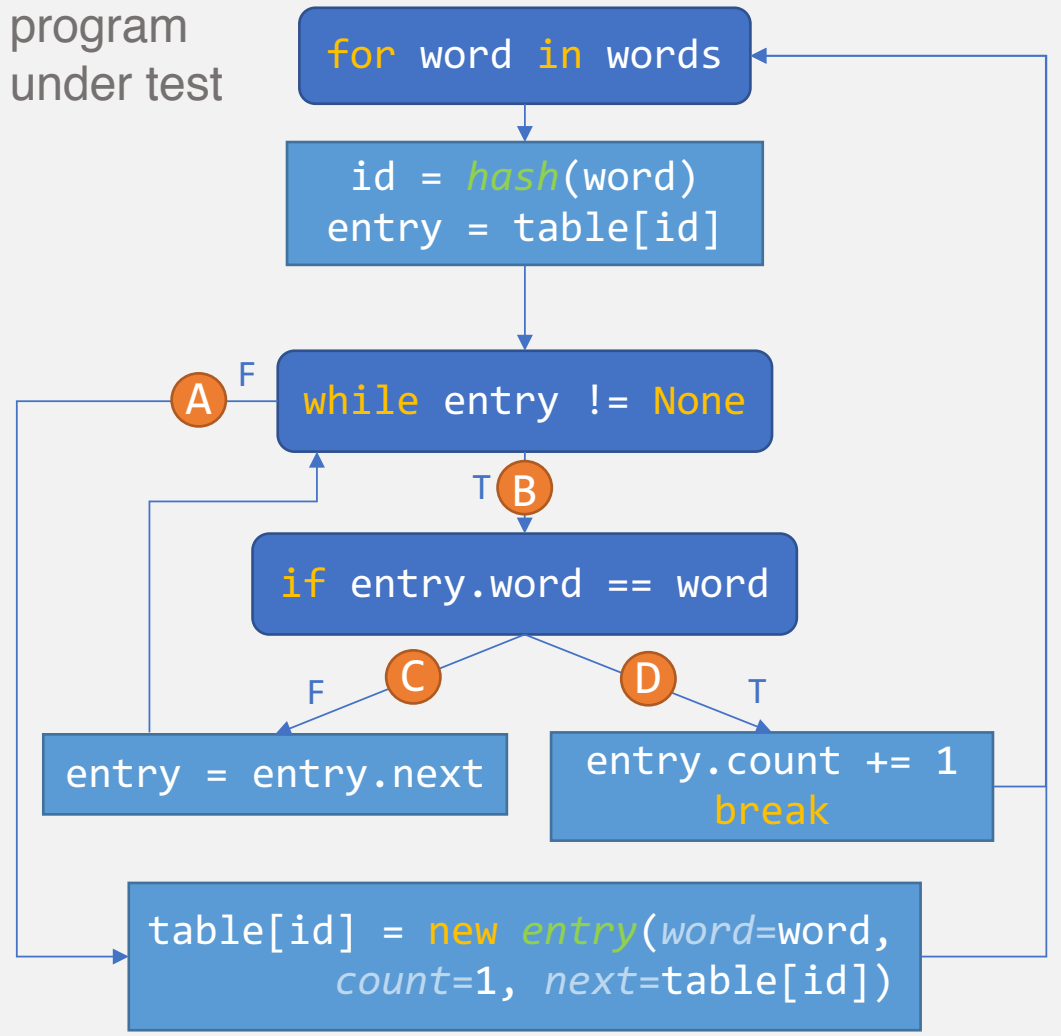
input running, feedback analysis

Current Input
the quack brown the dog

Edge	# Hits
A	4
B	1
D	1

no new max

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown the dog

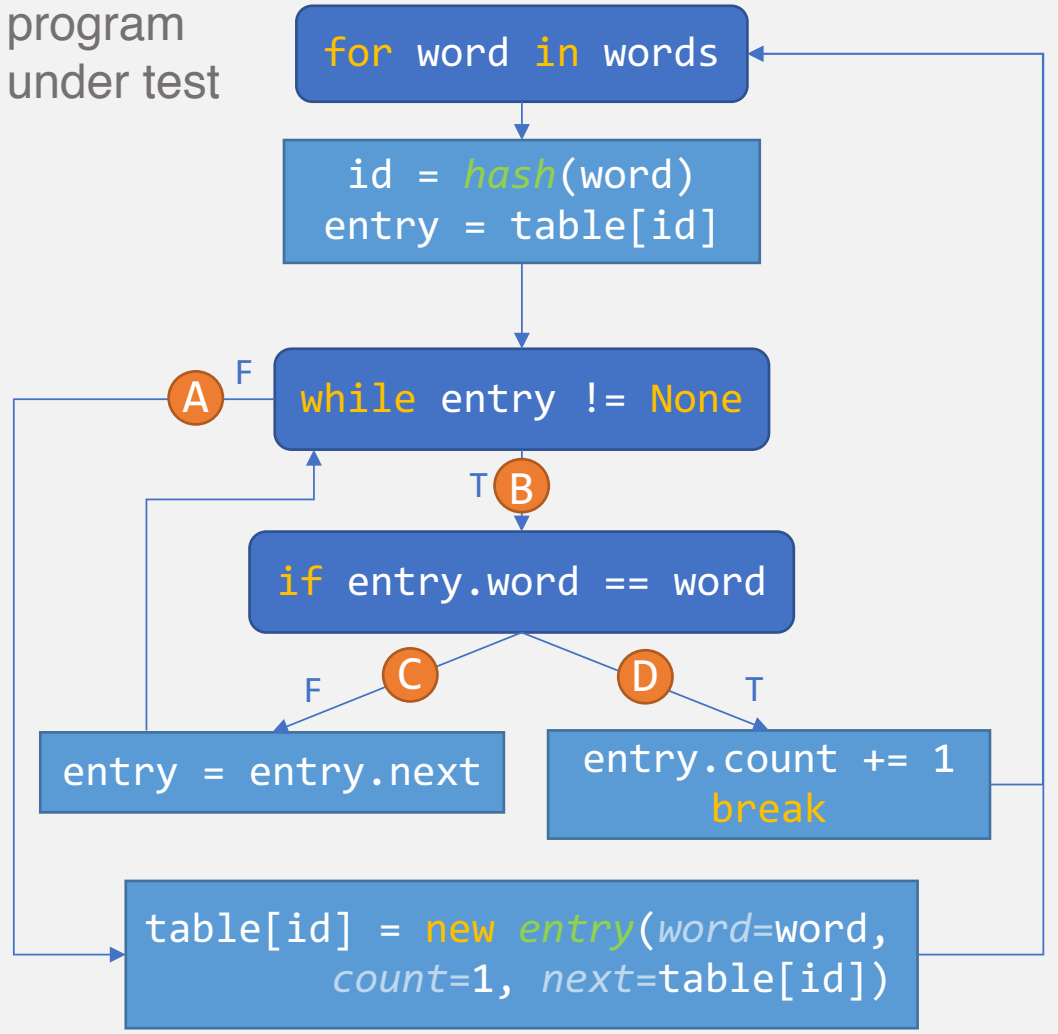
input running, feedback analysis

new mutant

Current Input
the quick brown t e dog

Edge	# Hits

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown the dog

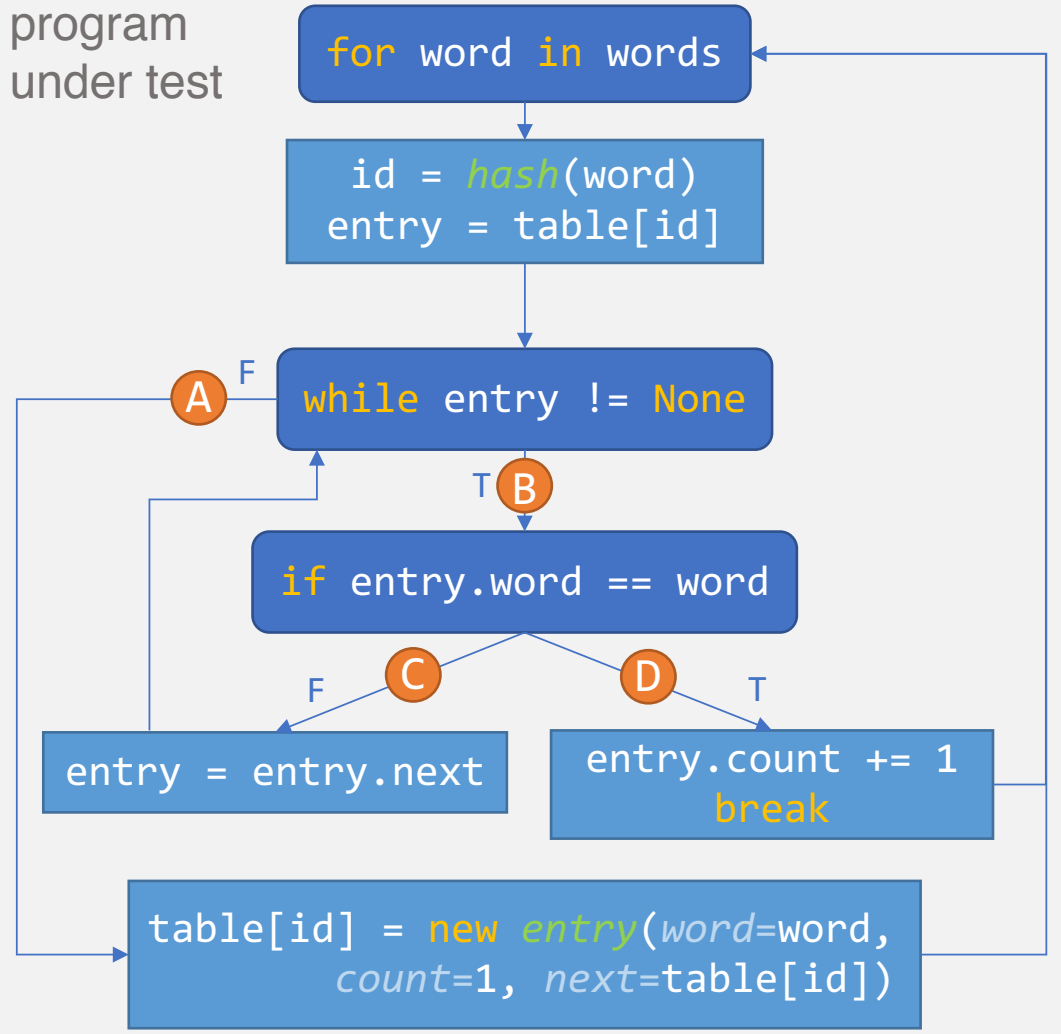
input running, feedback analysis

Current Input
the quick brown t e dog

Edge	# Hits

input

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

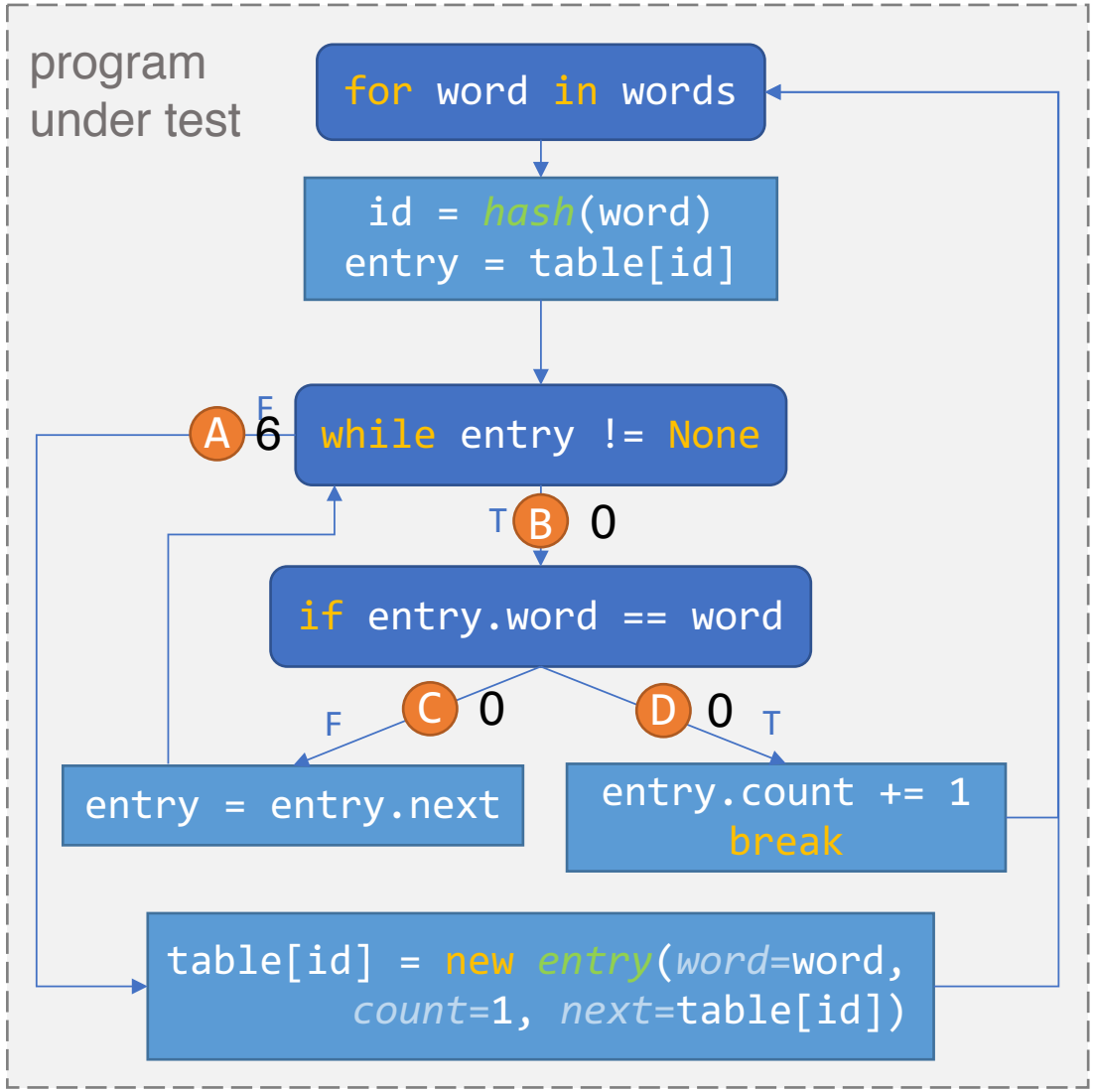
mutation engine

Parent to Mutate
the quick brown the dog

input running, feedback analysis

Current Input
the quick brown t e dog

Edge	# Hits



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	4	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown the dog

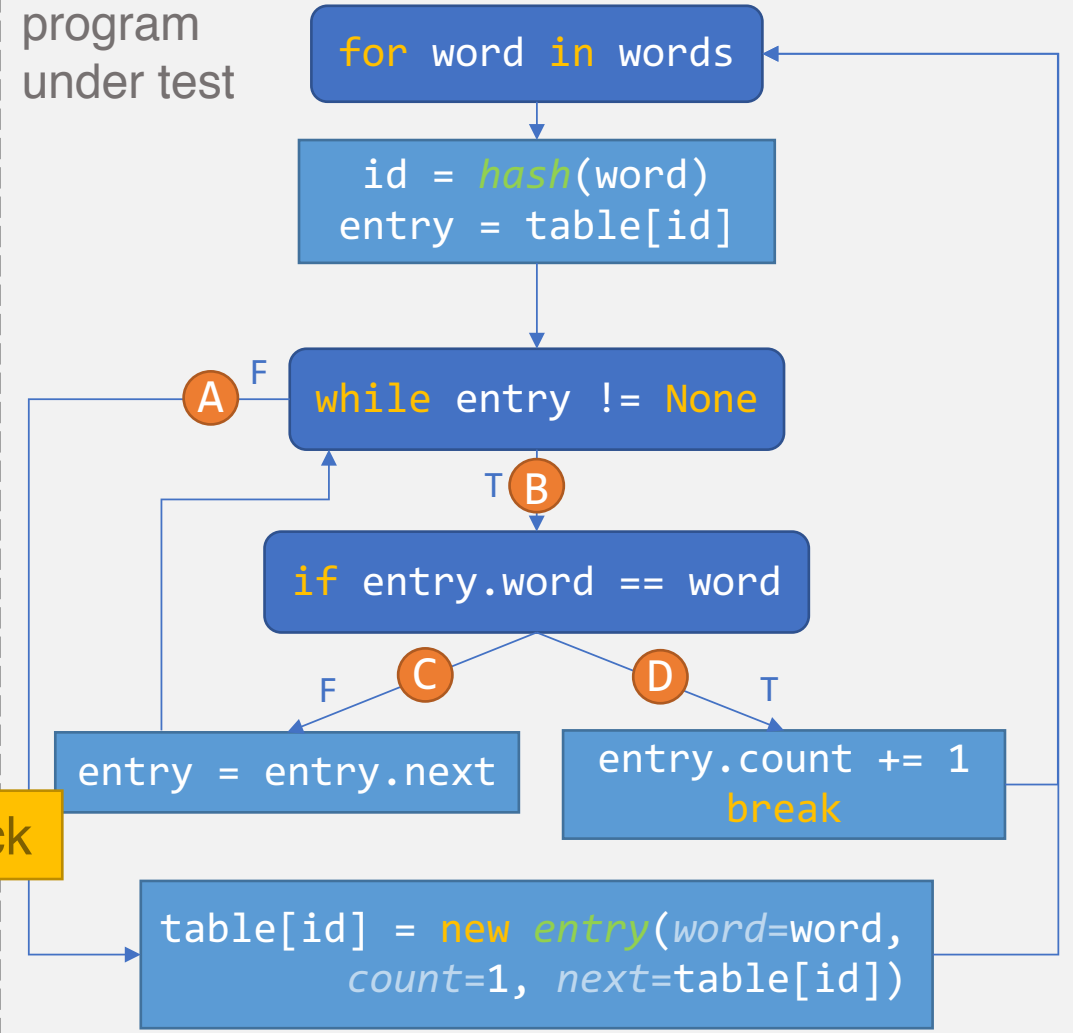
input running, feedback analysis

Current Input
the quick brown t e dog

Edge	# Hits
A	6

feedback

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown t e dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown the dog

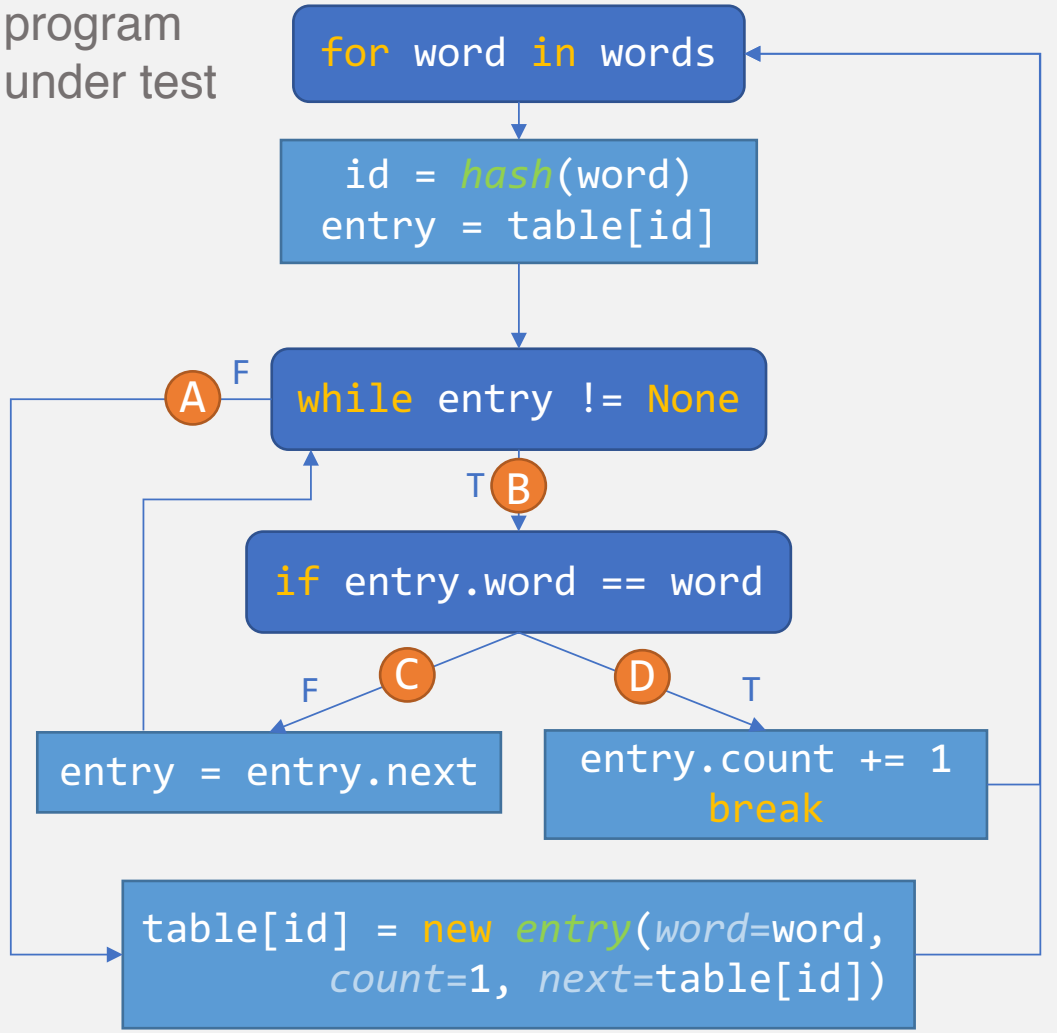
save new max for A

input running, feedback analysis

Current Input
the quick brown t e dog

Edge	# Hits
A	6

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown the dog

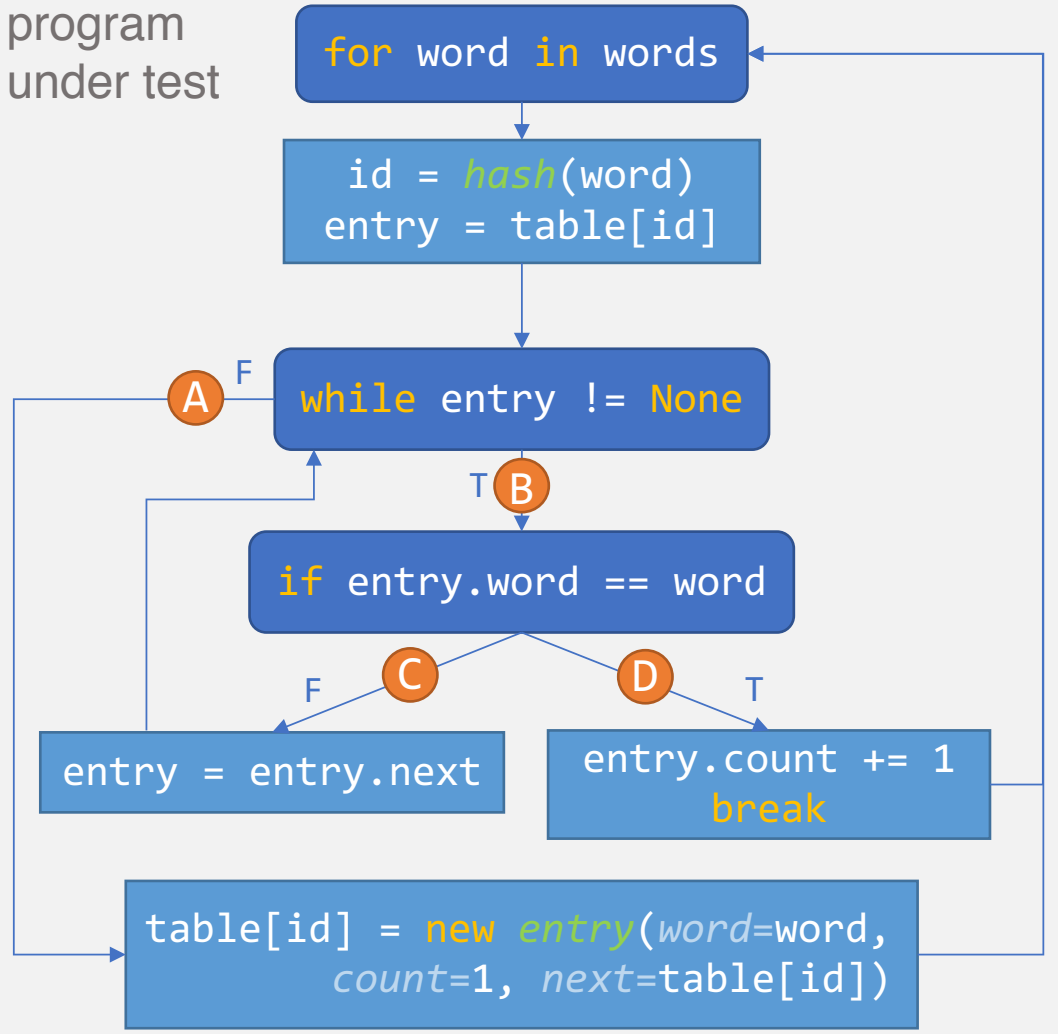
input running, feedback analysis

new mutant

Current Input
the quick_brown the dog

Edge	# Hits

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown t e dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown the dog

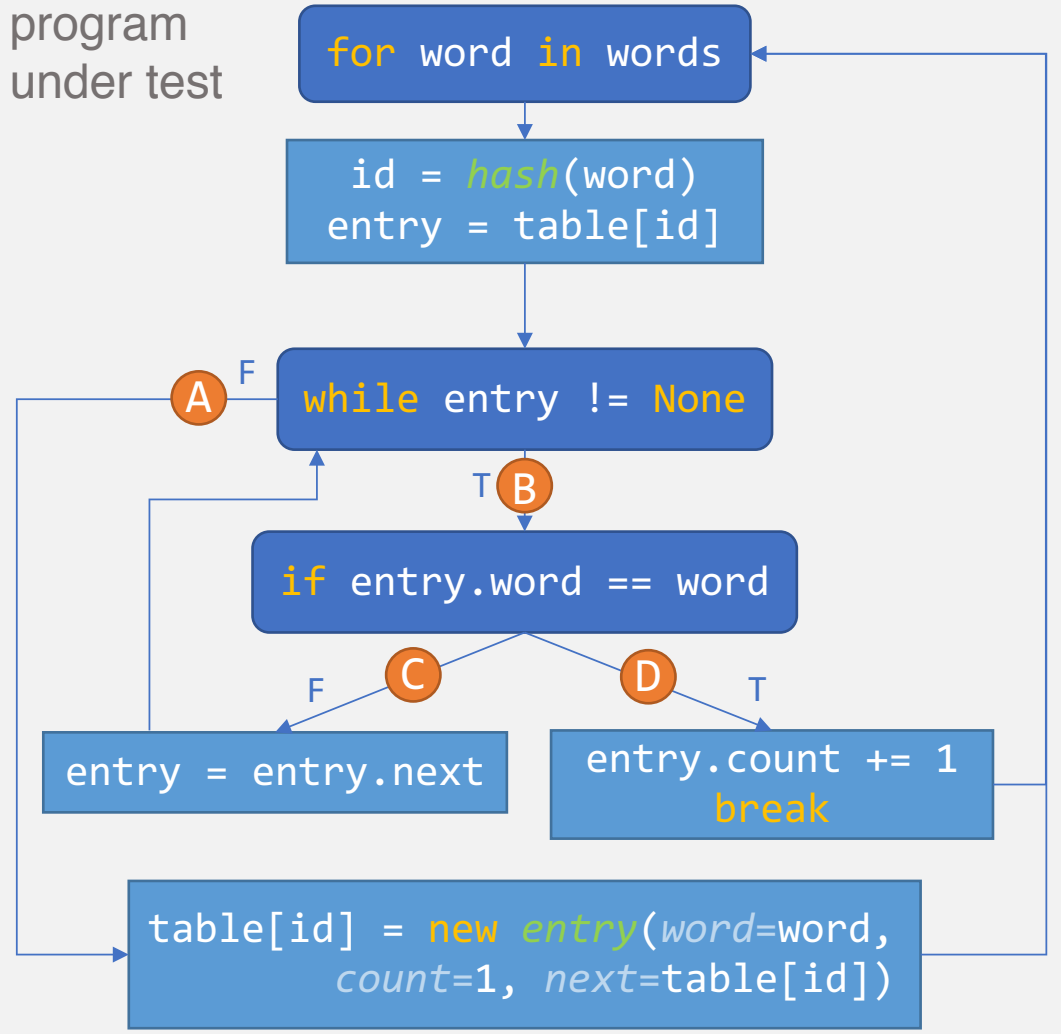
input running, feedback analysis

Current Input
the quick_brown the dog

Edge	# Hits

input

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

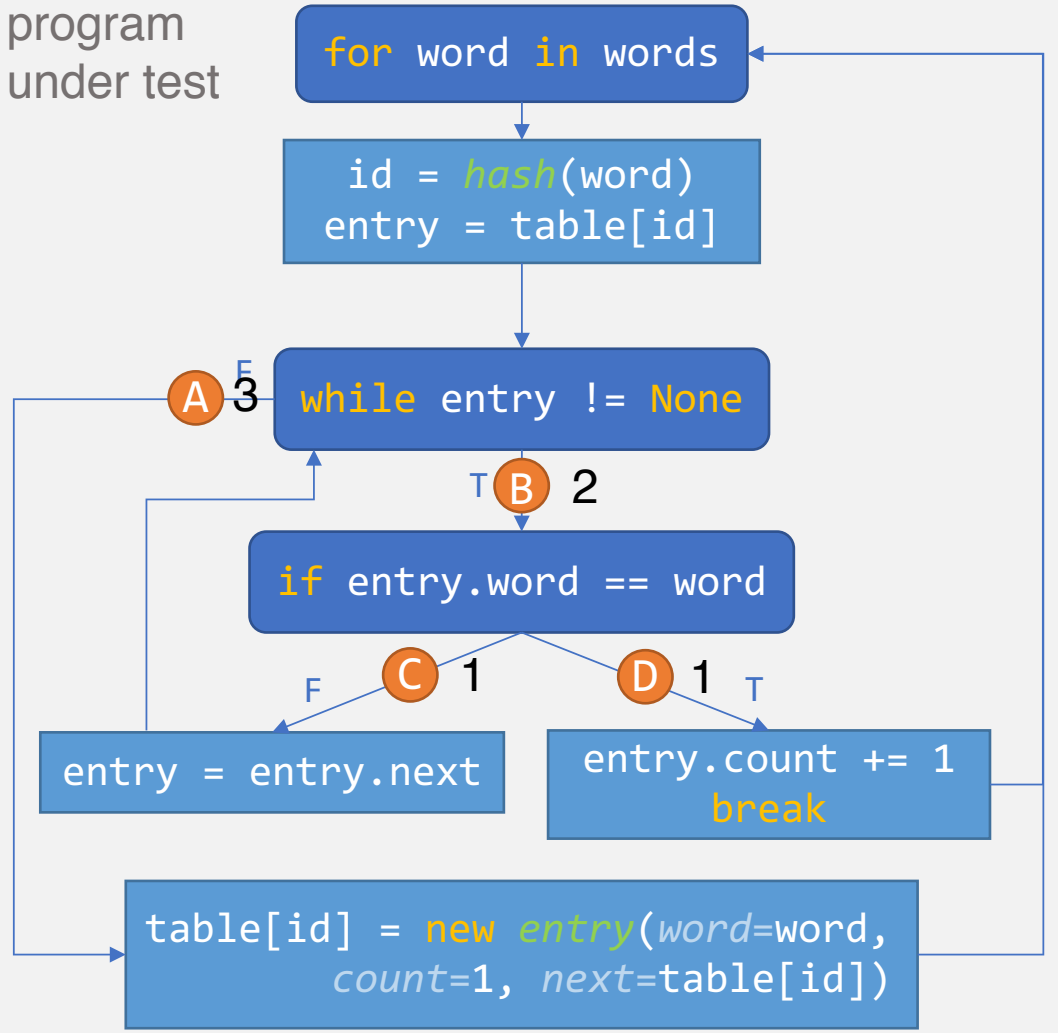
Parent to Mutate
the quick brown the dog

input running, feedback analysis

Current Input
the quick_brown the dog

Edge	# Hits

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown the dog
B	1	the quick brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown the dog

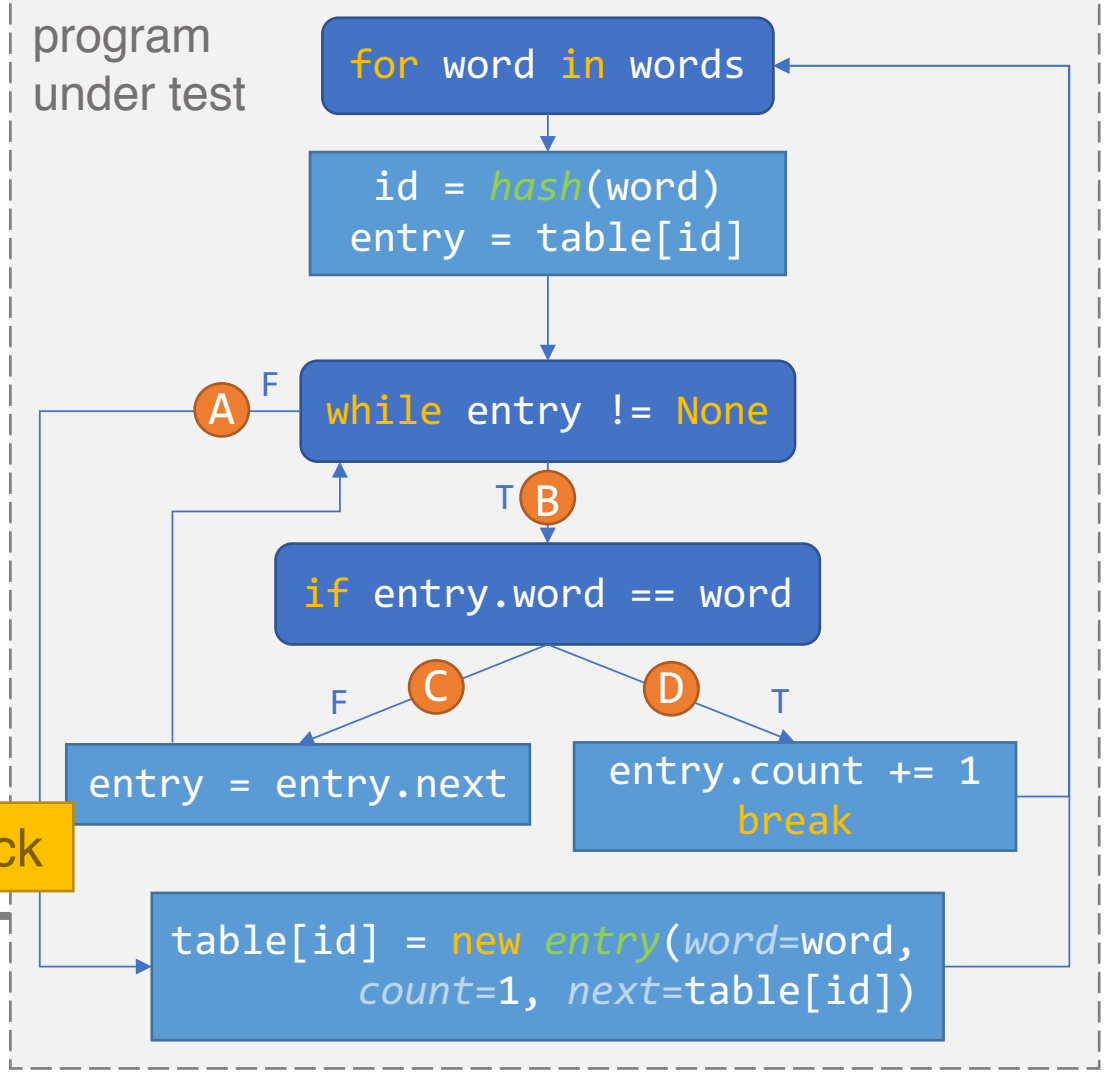
input running, feedback analysis

Current Input
the quick_brown the dog

Edge	# Hits
A	3
B	2
C	1
D	1

feedback

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown t e dog
B	1	the quick brown the dog
D	1	the quick brown the dog

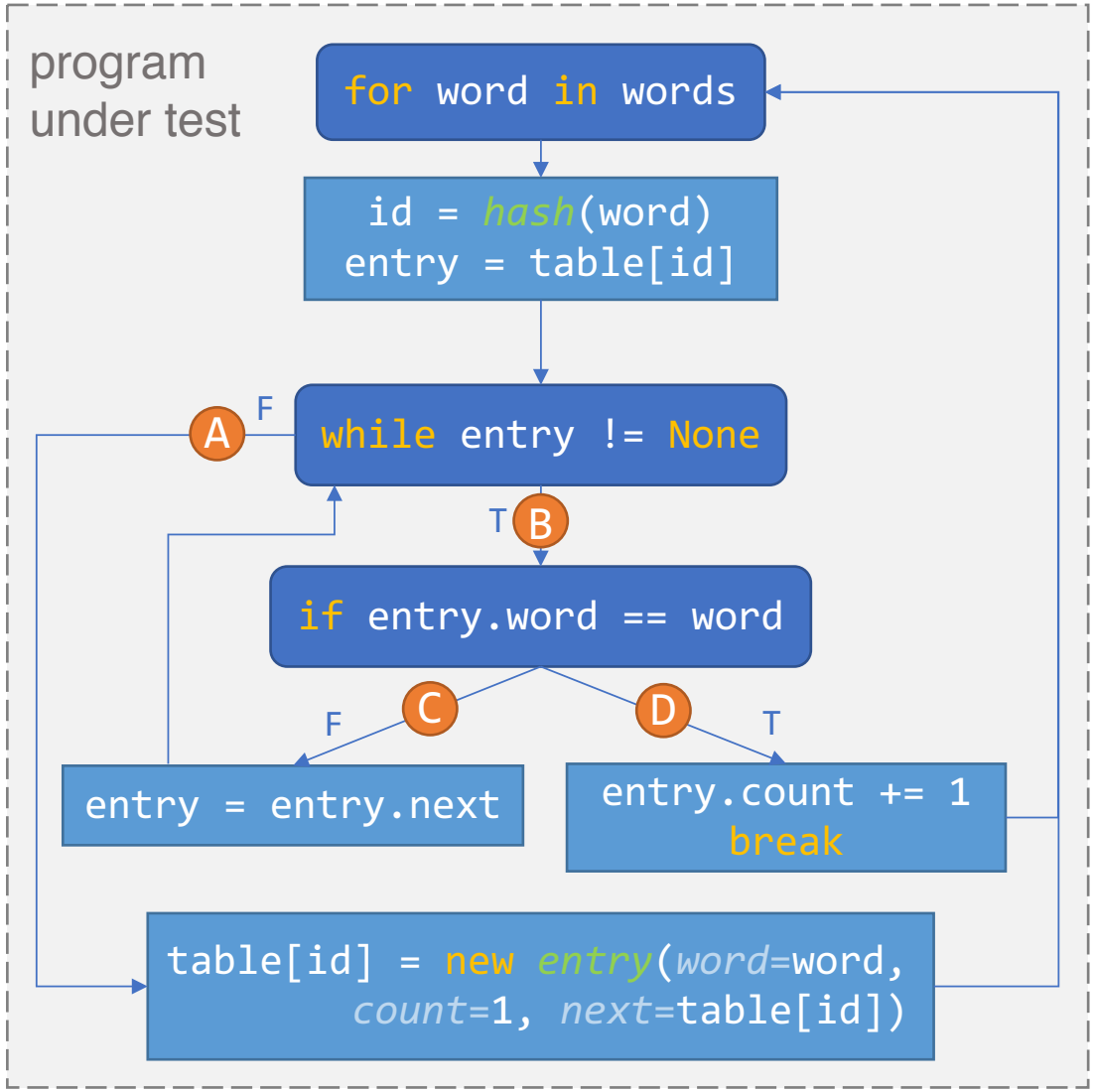
mutation engine

Parent to Mutate
the quick brown the dog

input running, feedback analysis

Current Input
the quick_brown the dog

Edge	# Hits
A	3
B	2
C	1
D	1



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown the dog
B	2	the quick_brown the dog
C	1	the quick_brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown the dog

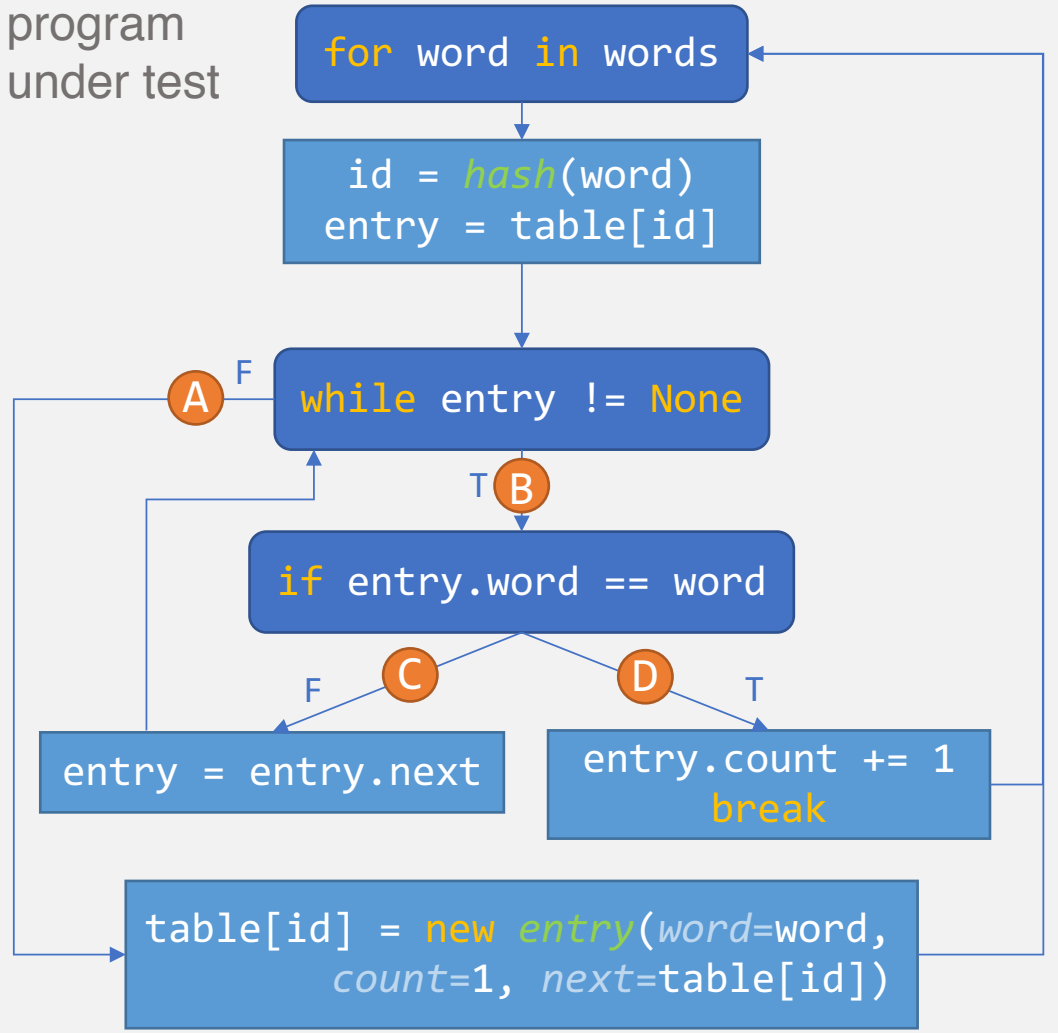
save new max for B,C

input running, feedback analysis

Current Input
the quick_brown the dog

Edge	# Hits
A	3
B	2
C	1
D	1

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown the dog
B	2	the quick_brown the dog
C	1	the quick_brown the dog
D	1	the quick brown the dog

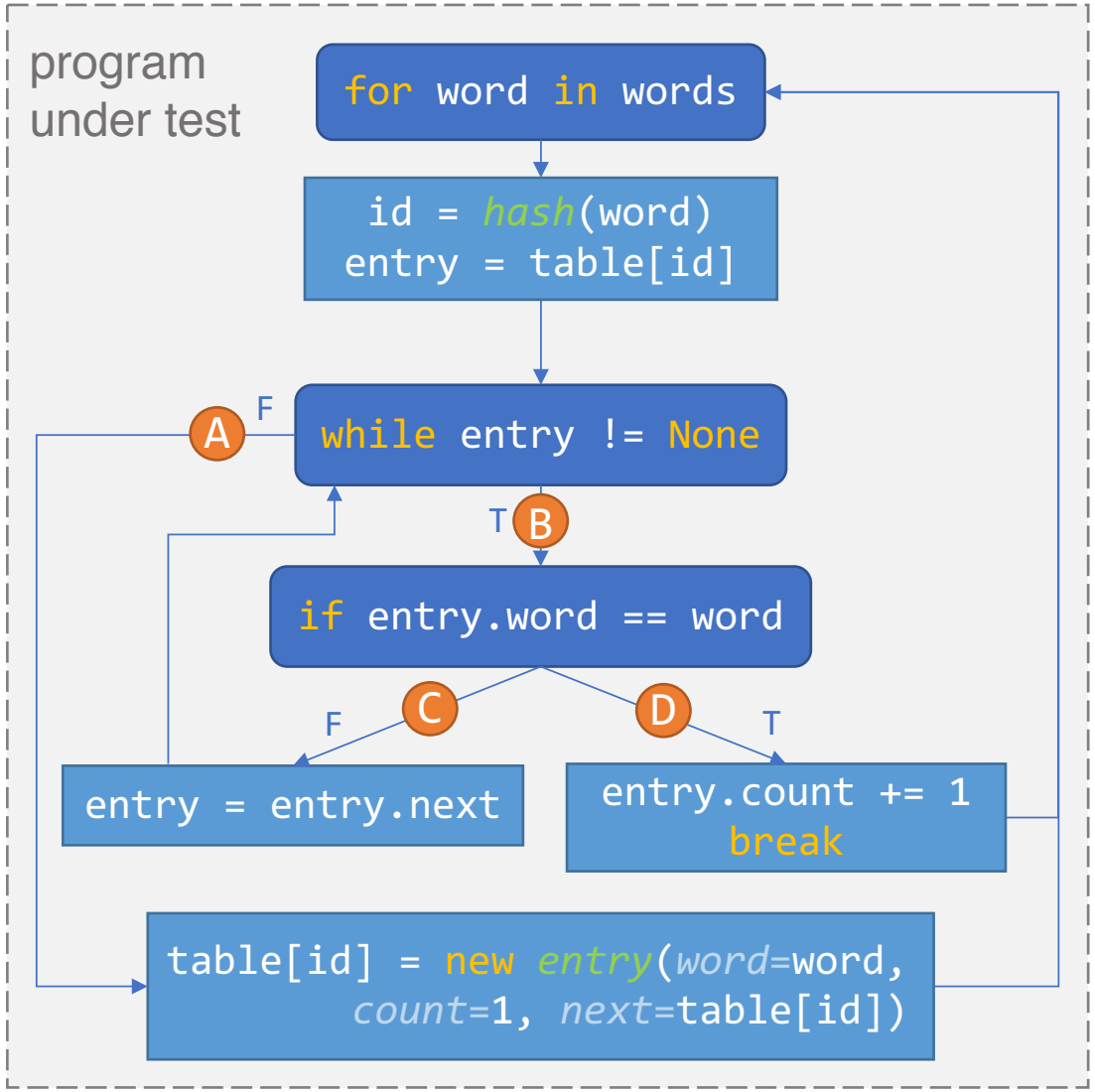
mutation engine

Parent to Mutate
the quick brown the dog

input running, feedback analysis

Current Input

Edge	# Hits



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown t e dog
B	2	the quick_brown the dog
C	1	the quick_brown the dog
D	1	the quick brown the dog

choose parent

mutation engine

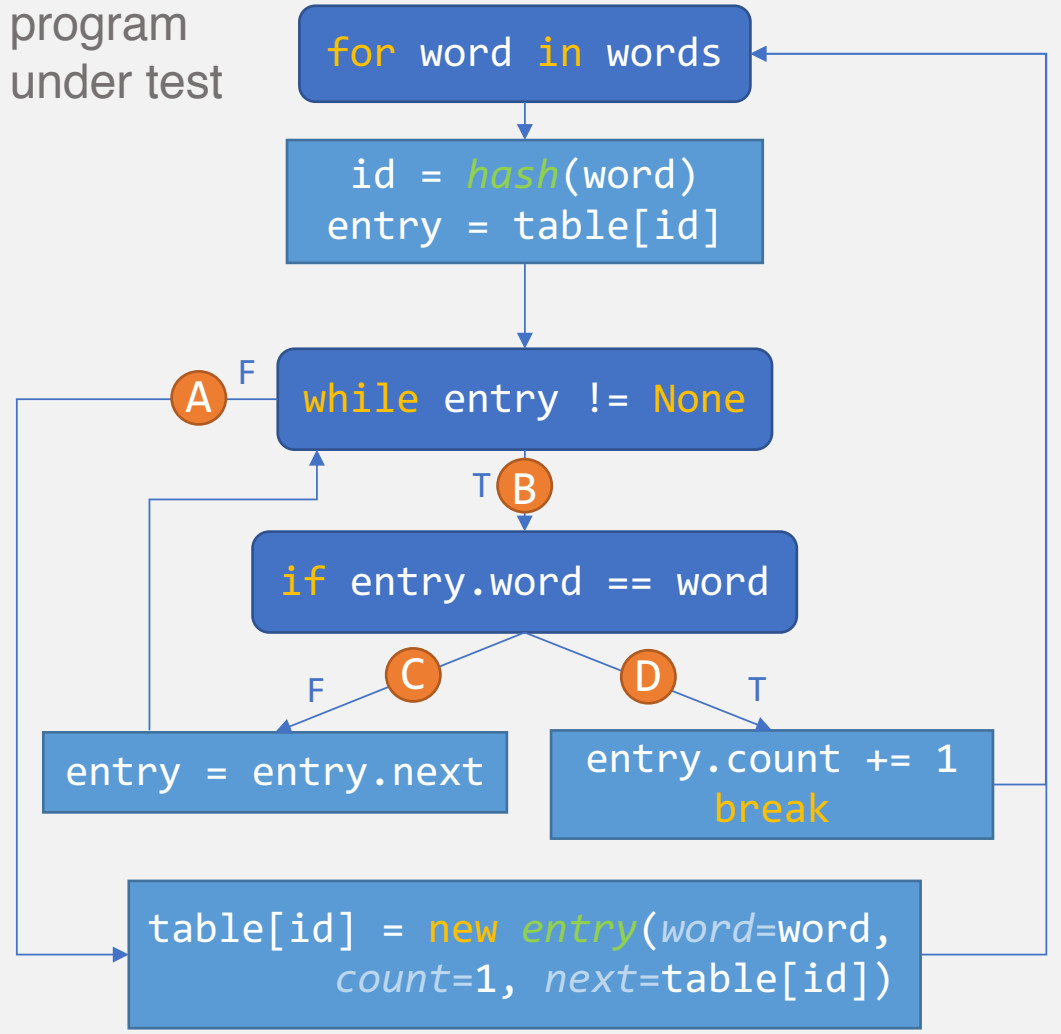
Parent to Mutate
the quick brown t e dog

input running, feedback analysis

Current Input

Edge	# Hits

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown t e dog
B	2	the quick_brown the dog
C	1	the quick_brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutate
the quick brown t e dog

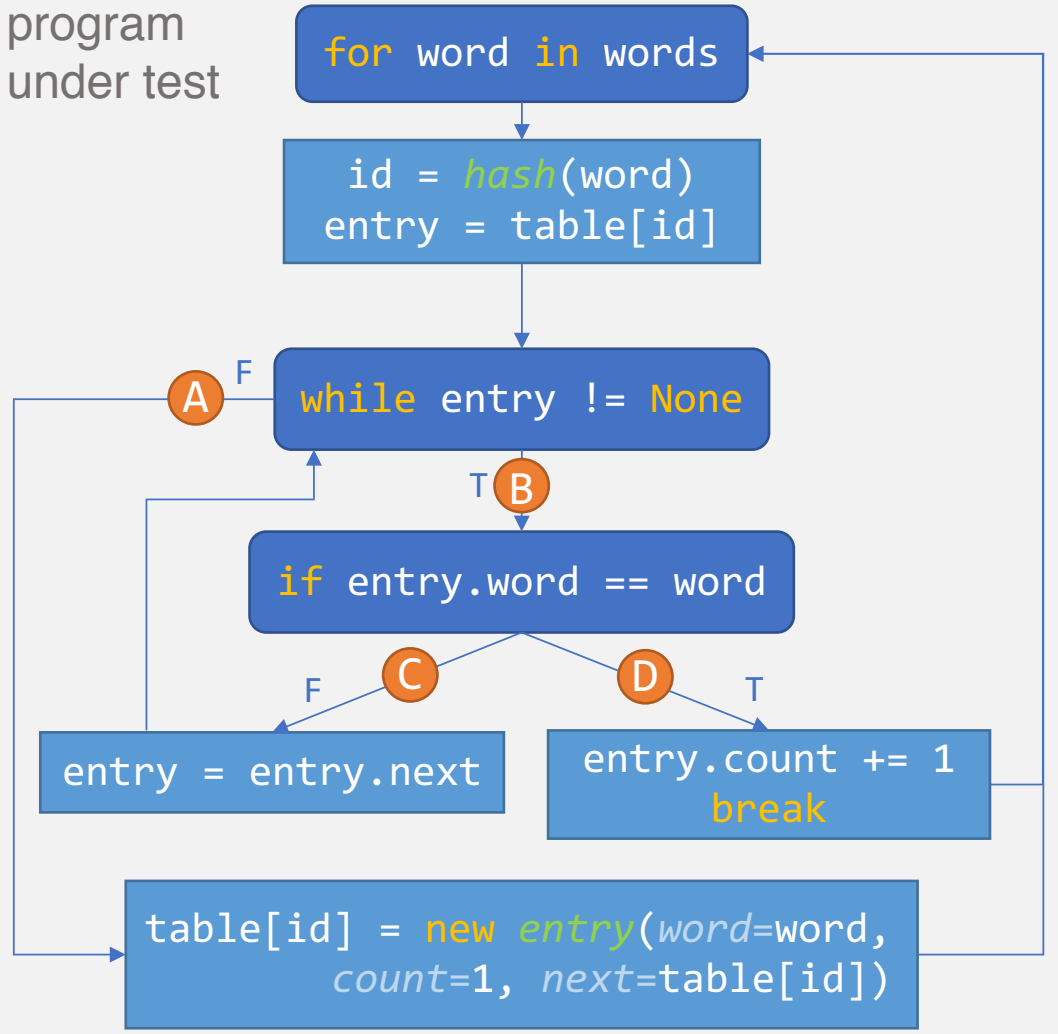
let's mutate this many times

input running, feedback analysis

Current Input

Edge	# Hits

program under test



PerfFuzz Algorithm

input corpus

Edge	Max # Hits	Maximizing Input
A	6	the quick brown t e dog
B	2	the quick_brown the dog
C	1	the quick_brown the dog
D	1	the quick brown the dog

mutation engine

Parent to Mutat
the quick brown t

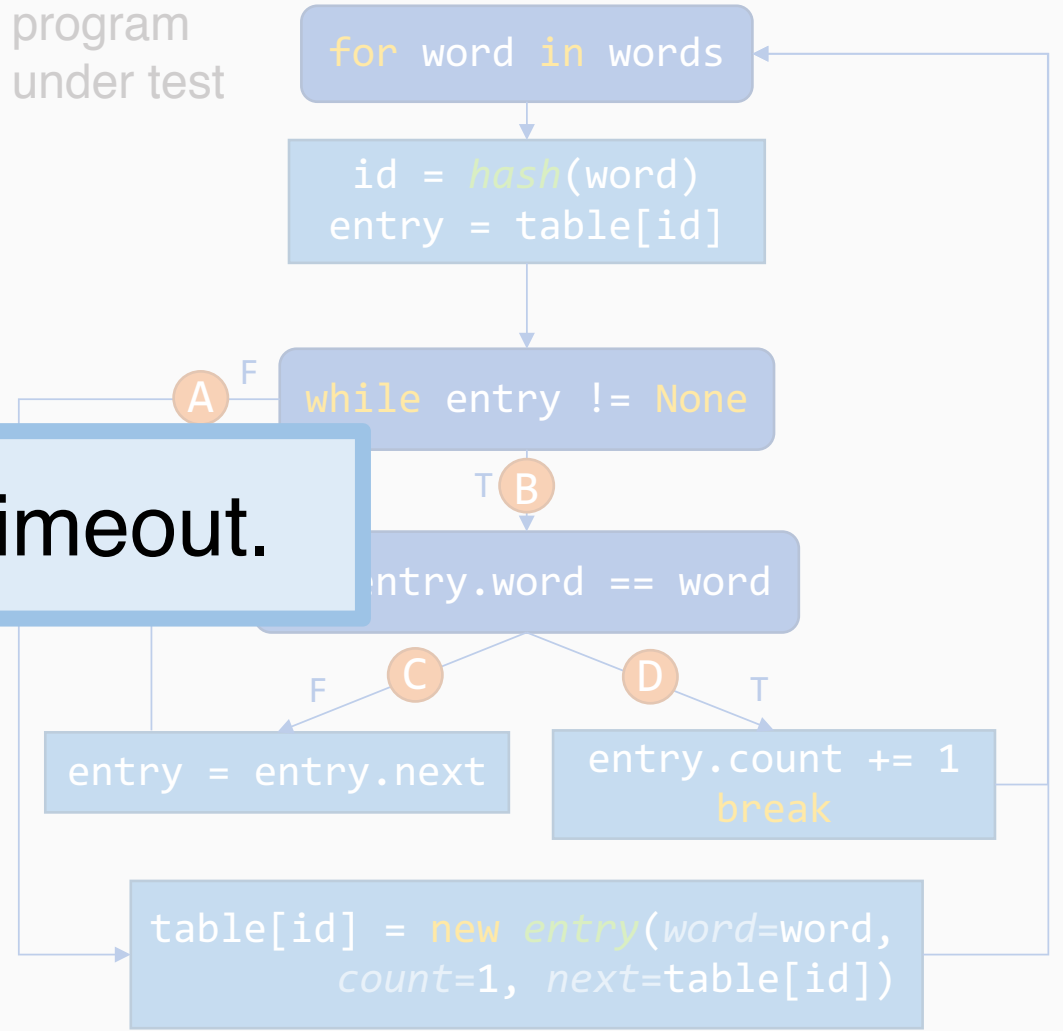
input running, feedback analysis

Current Input

Edge	# Hits

Repeat until timeout.

program under test



PerfFuzz Algorithm: Results

input corpus

Edge	Max # Hits	Maximizing Input
A	12	t h e q u i c k b r o w
B	21	t ?t xt at$ #a))t Qwaa
C	21	t ?t xt at$ #a))t Qwaa
D	11	t t t t t t t t t t t t

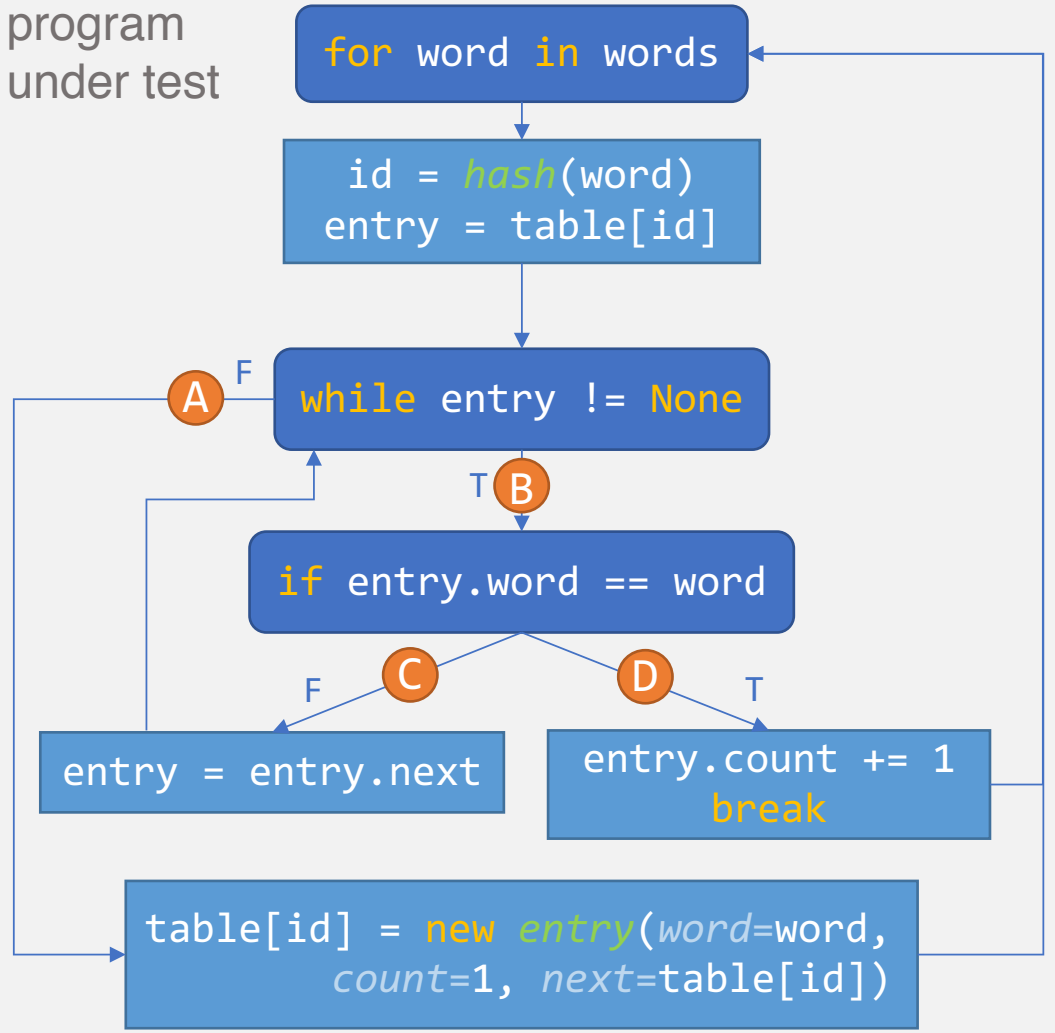
mutation engine

input running, feedback analysis

Current Input

Edge	# Hits

program under test



PerfFuzz Algorithm: Results

input corpus

Edge	Max # Hits	Maximizing Input
A	12	t h e q u i c k b r o w
B	21	t ?t xt at$ #a))t Qwaa
C	21	t ?t xt at$ #a))t Qwaa
D	11	t t t t t t t t t t t t

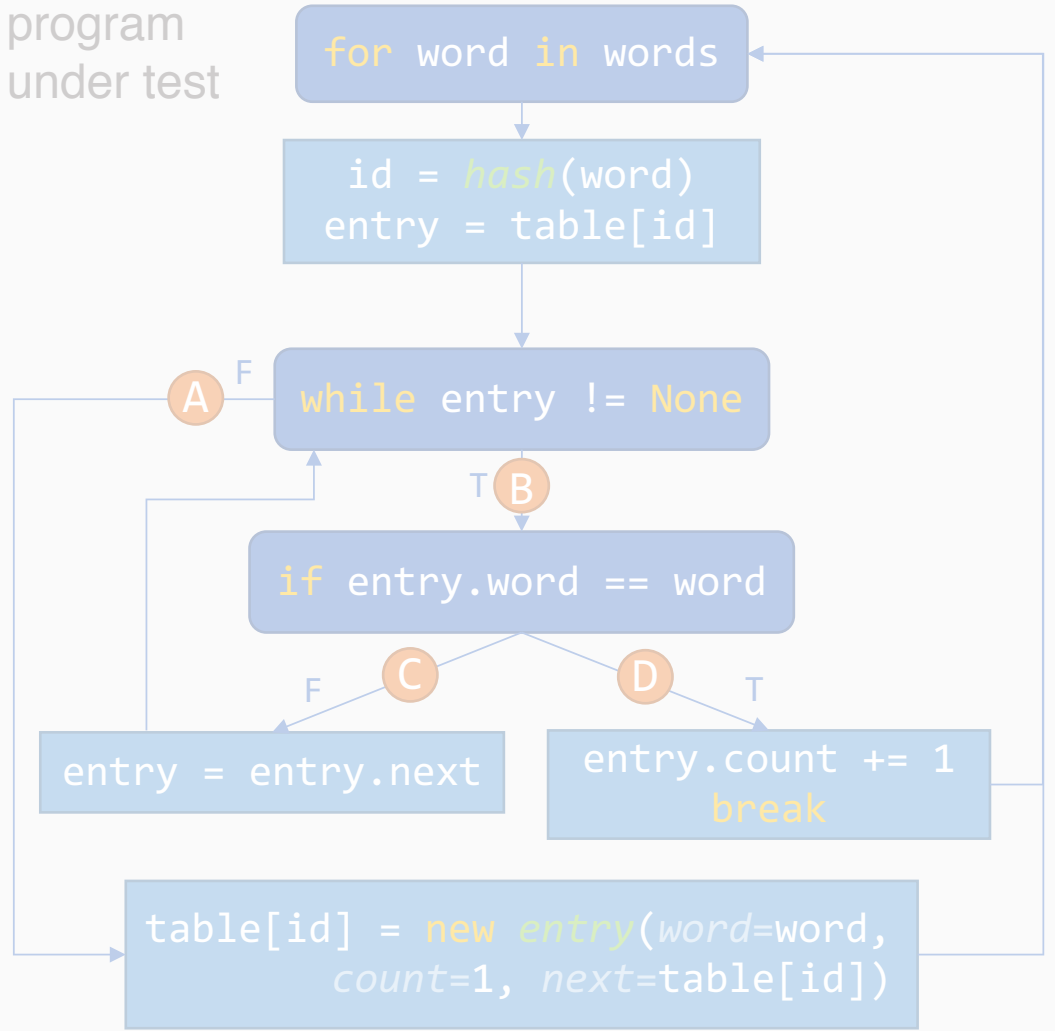
mutation engine

input running, feedback analysis

Current Input

Edge	# Hits

program under test



PerfFuzz Algorithm: Results

input corpus

Edge	Max # Hits	Maximizing Input
A	12	t h e q u i c k b r o w
B	21	t ?t xt at$ #a))t Qwaa
C	21	t ?t xt at$ #a))t Qwaa
D	11	t t t t t t t t t t t t

many unique words

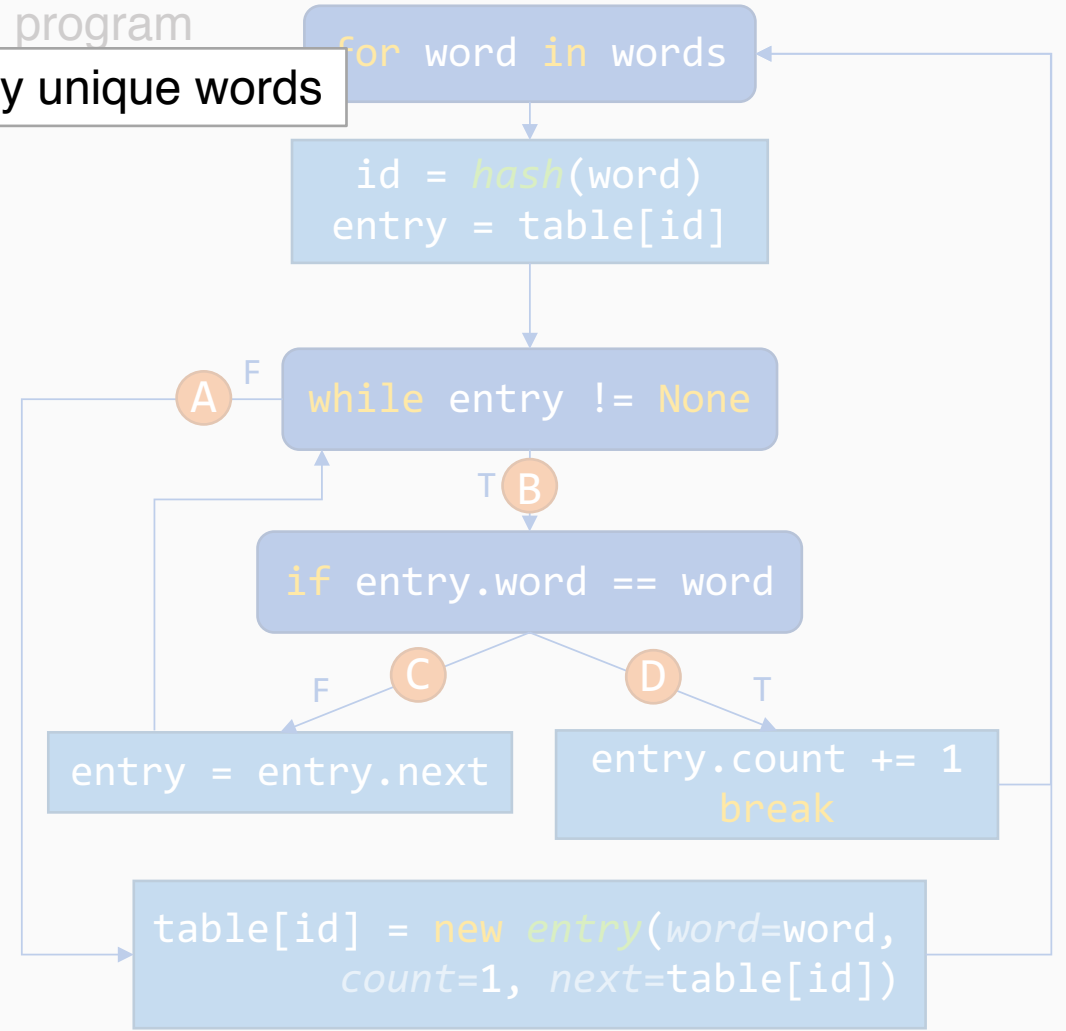
mutation engine

input running, feedback analysis

Current Input

Edge	# Hits

program



PerfFuzz Algorithm: Results

input corpus

Edge	Max # Hits	Maximizing Input
A	12	t h e q u i c k b r o w
B	21	t ?t xt at$ #a))t Qwaa
C	21	t ?t xt at$ #a))t Qwaa
D	11	t t t t t t t t t t t t

many unique words

many hash collisions in same bucket

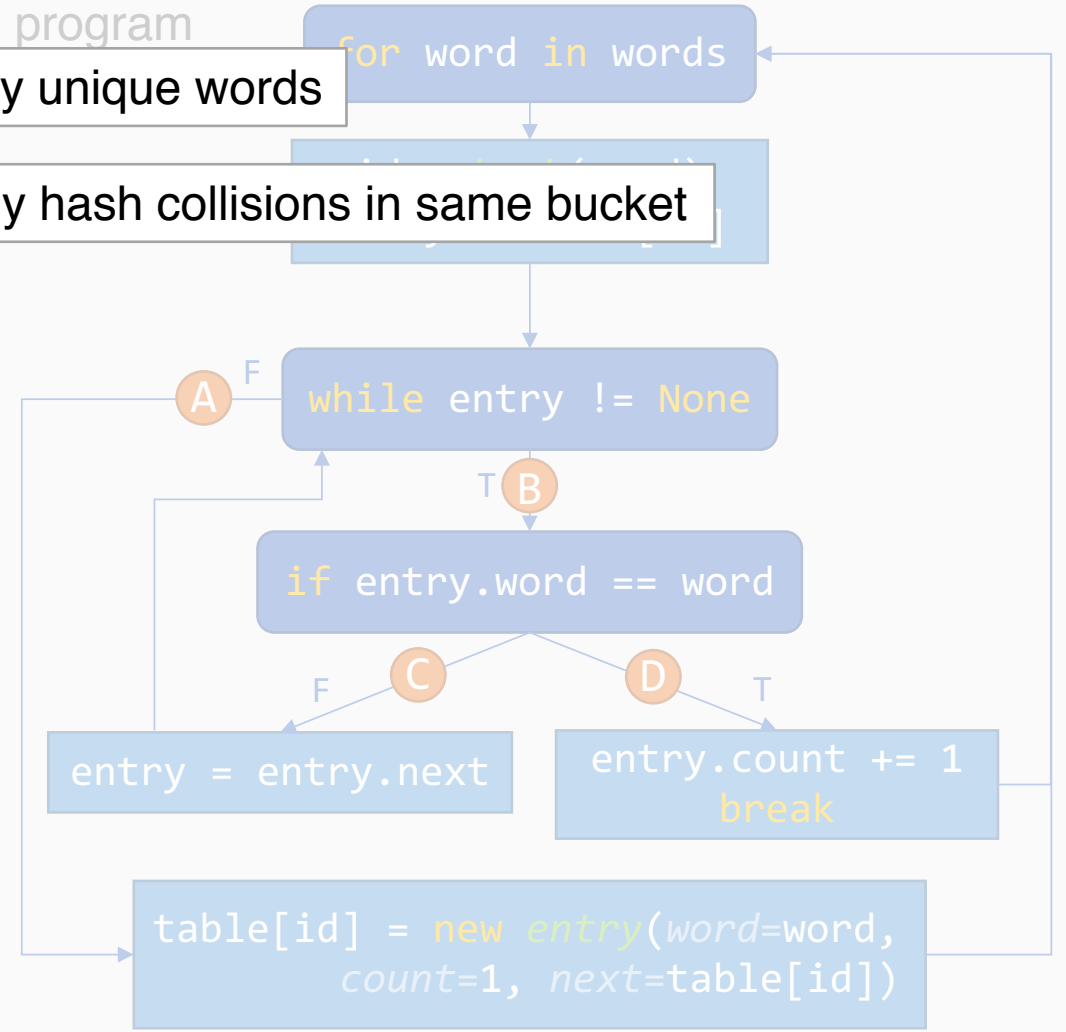
mutation engine

input running, feedback analysis

Current Input

Edge	# Hits

program



PerfFuzz Algorithm: Results

input corpus

Edge	Max # Hits	Maximizing Input
A	12	t h e q u i c k b r o w
B	21	t ?t xt at$ #a))t Qwaa
C	21	t ?t xt at$ #a))t Qwaa
D	11	t t t t t t t t t t t t

many unique words

many hash collisions in same bucket

many occurrences of same word

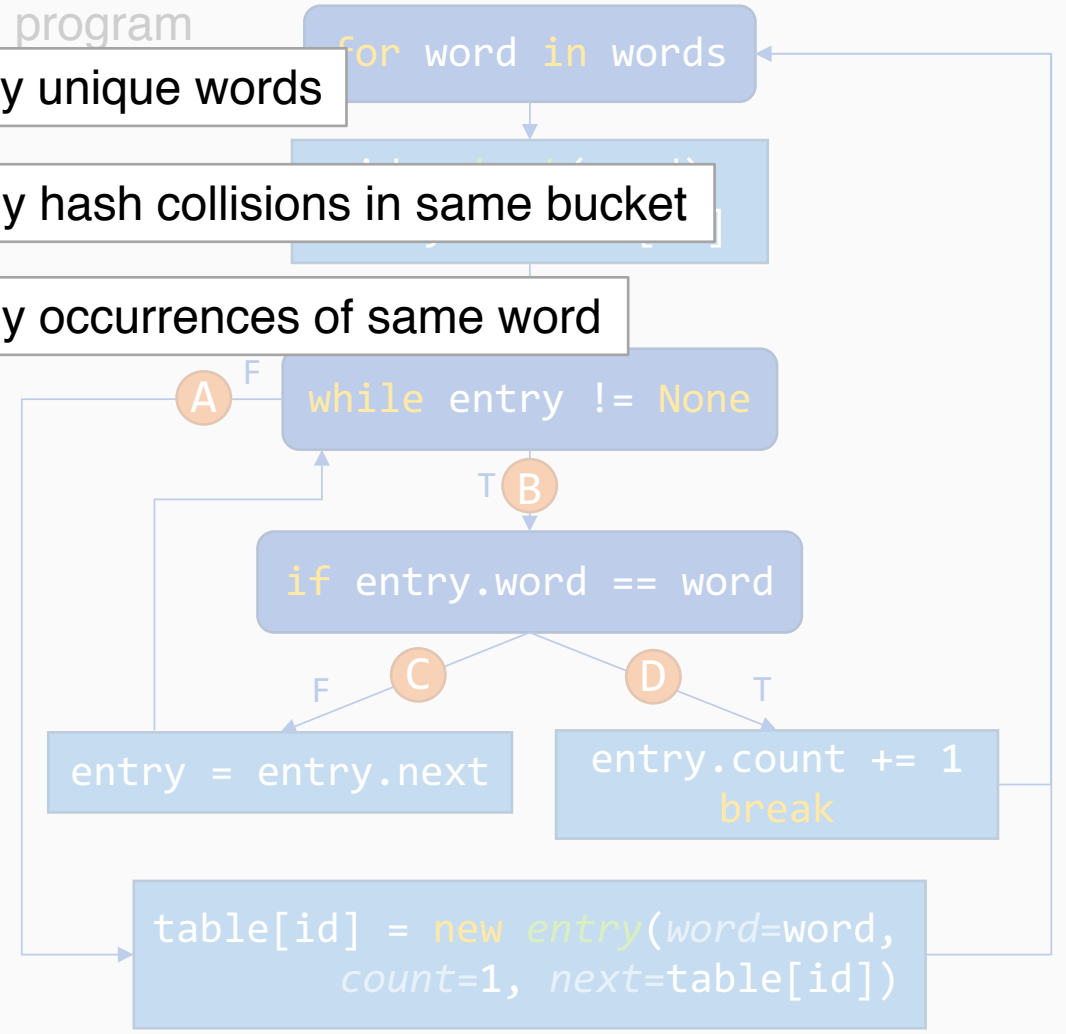
mutation engine

input running, feedback analysis

Current Input

Edge	# Hits

program



PerfFuzz Algorithm: Results

input corpus

Edge	Max # Hits	Maximizing Input
A	12	t h e q u i c k b r o w
B	21	t ?t xt at$ #a))t Qwaa
C	21	t ?t xt at$ #a))t Qwaa
D	11	t t t t t t t t t t t t

many unique words

many hash collisions in same bucket

many occurrences of same word

→ variety of performance behaviors

mutation engine

input running, feedback analysis

Current Input

Edge	# Hits

program

```
for word in words
    while entry != None
        if entry.word == word
            entry.count += 1
            break
        entry = entry.next
    table[id] = new entry(word=word,
                          count=1, next=table[id])
```

A F C D F T

Evaluation Outline

- Compare to SlowFuzz
  - Macro-benchmarks
  - Micro-benchmarks
- Compare to AFL
- Case Studies

Evaluation Outline

- Compare to SlowFuzz
  - Macro-benchmarks
  - Micro-benchmarks
- Compare to AFL
- Case Studies (more in paper)

Prior Work

SlowFuzz

→ Fuzzing to find algorithmic complexity vulnerabilities

- Saves inputs that increase *total* path length
- Randomly chooses parent
- Prioritizes mutations that increase path length
- Faster than PerfFuzz (based on LibFuzzer)

T. Petsios, J. Zhao, A. D. Keromytis, and S. Jana. 2017. SlowFuzz: Automated Domain-Independent Detection of Algorithmic Complexity Vulnerabilities. In Proceedings of CCS '17. DOI: https://doi.org/10.1145/3133956.3134073

Prior Work

SlowFuzz

→ Fuzzing to find algorithmic complexity vulnerabilities

- Saves inputs that increase *total* path length
- Randomly chooses parent
- Prioritizes mutations that increase path length
- Faster than PerfFuzz (based on LibFuzzer)

T. Petsios, J. Zhao, A. D. Keromytis, and S. Jana. 2017. SlowFuzz: Automated Domain-Independent Detection of Algorithmic Complexity Vulnerabilities. In Proceedings of CCS '17. DOI: https://doi.org/10.1145/3133956.3134073

Experimental Setup: Macro-Benchmarks

- Max input size: 500 bytes
- Seeds: AFL default seed for each format
- Run each tool for 6 hours
- Repeat 6-hour runs 20 times

Library	LoC	Function Exercised
`libpng`	30k	PNG read
`libxml2`	70k	XML read
`libjpeg-turbo`	30k	JPEG decompress
`zlib`	9k	GZIP decombress

Macro-Benchmarks: Maximum Path Length

- Path length: total number of hits of CFG edges by an input

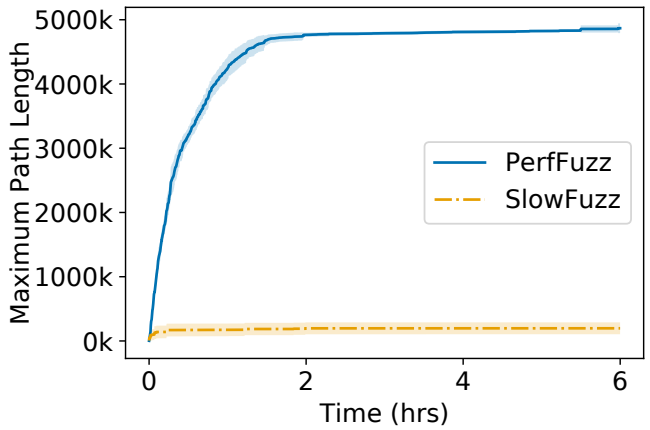
Edge	# Hits
A	1
B	11
C	0
D	11

path len: 23

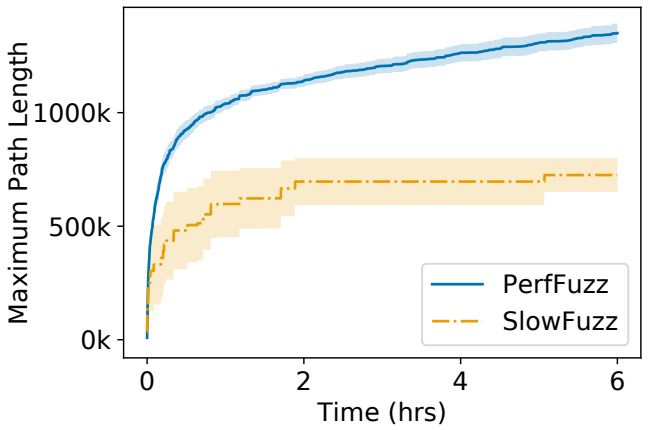
Macro-Benchmarks: Maximum Path Length

- Path length: total number of hits of CFG edges by an input

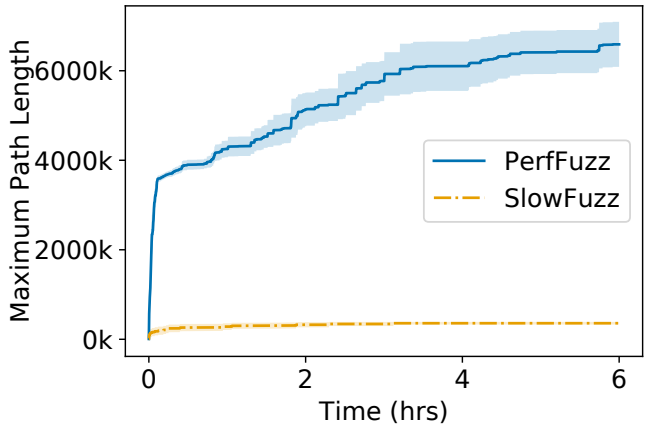
`libpng`



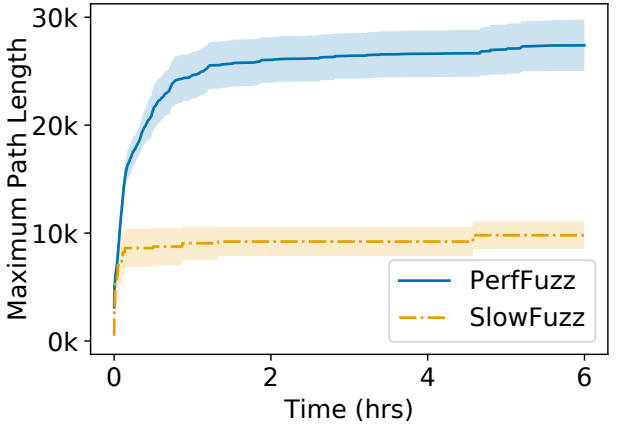
`libxml2`



`libjpeg-turbo`



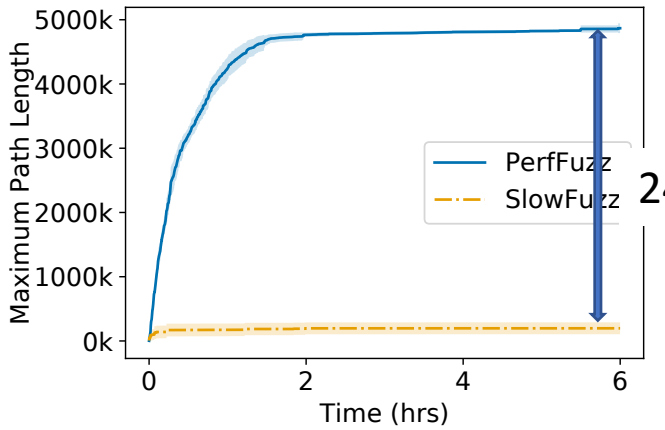
`zlib`



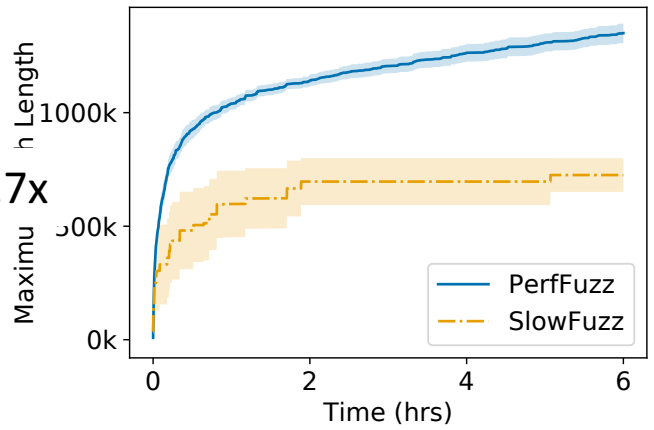
Macro-Benchmarks: Maximum Path Length

- Path length: total number of hits of CFG edges by an input

libpng

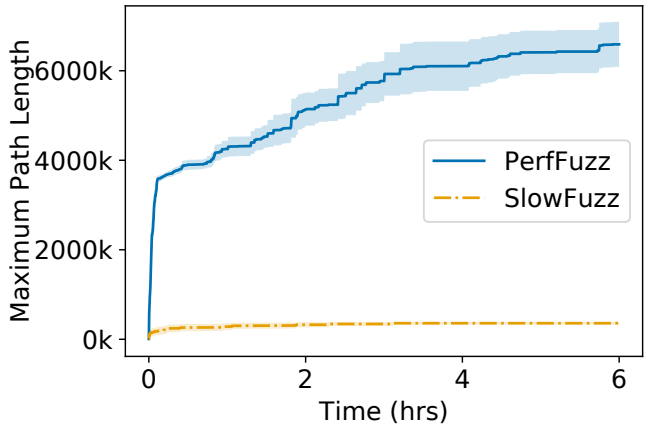


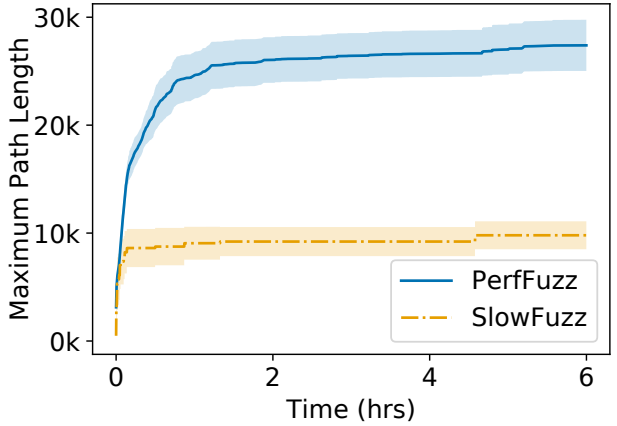
24.7x



libxml2

libjpeg-turbo





zlib

Macro-Benchmarks: Maximum Hot Spot

- Hot spot: maximum # hits of a CFG edge by an input

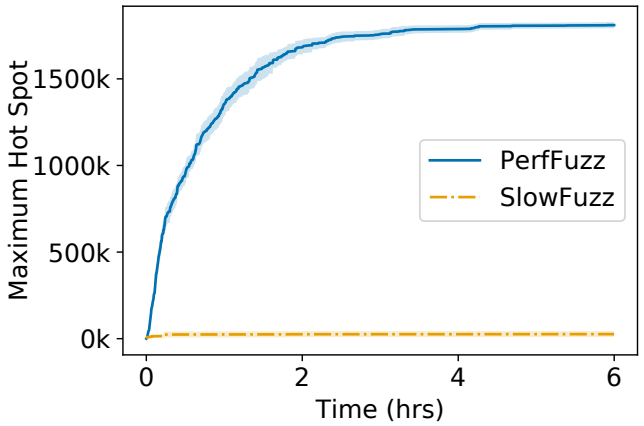
Edge	# Hits
A	1
B	11
C	0
D	11

hot spot: 11

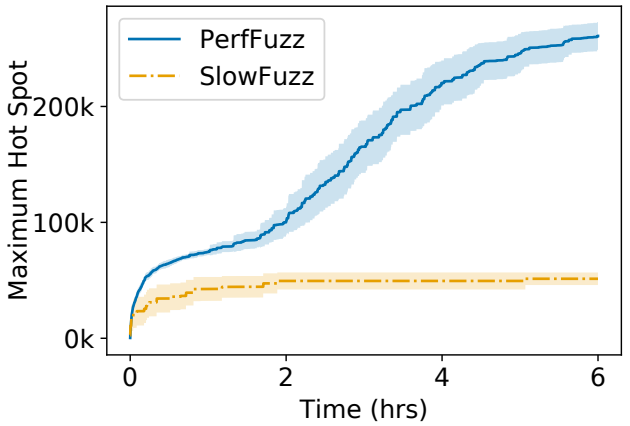
Macro-Benchmarks: Maximum Hot Spot

- Hot spot: maximum # hits of a CFG edge by an input

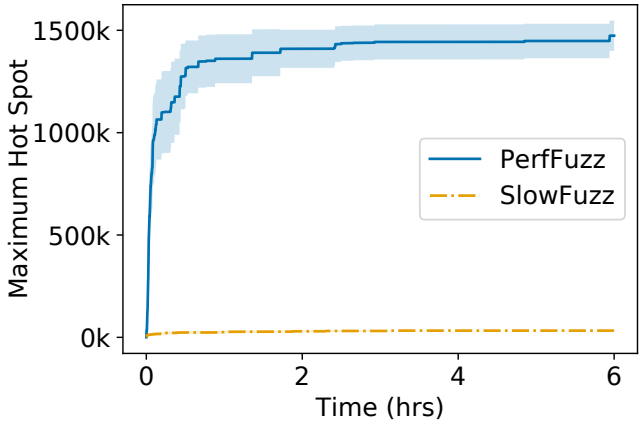
`libpng`



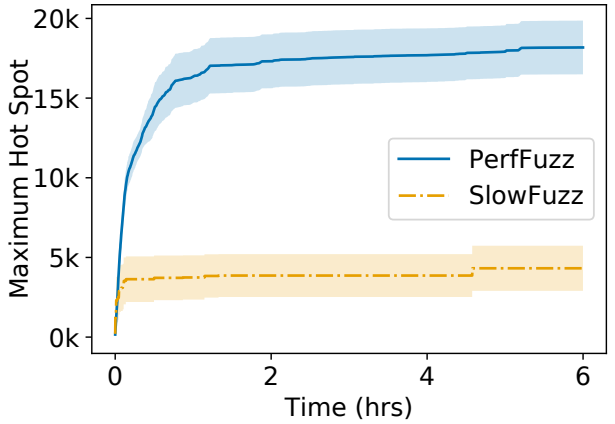
`libxml2`



`libjpeg-turbo`



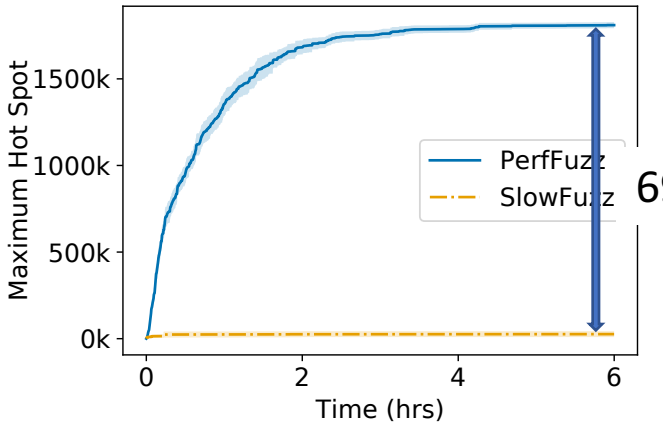
`zlib`



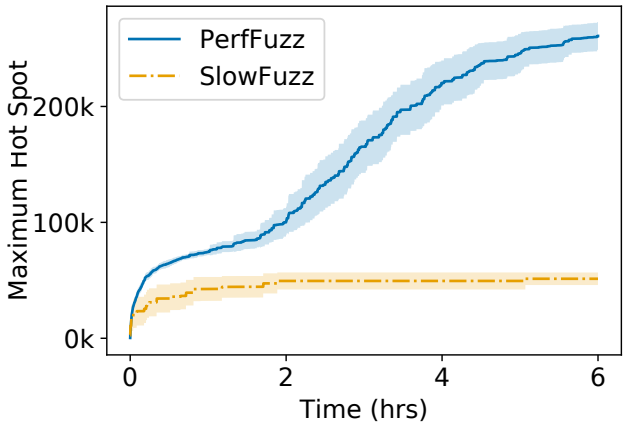
Macro-Benchmarks: Maximum Hot Spot

- Hot spot: maximum # hits of a CFG edge by an input

`libpng`

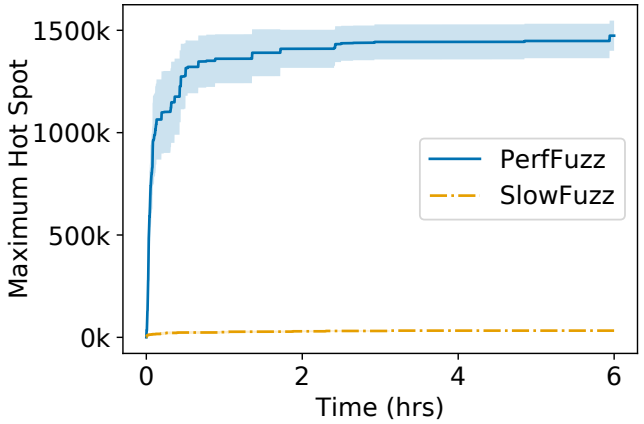


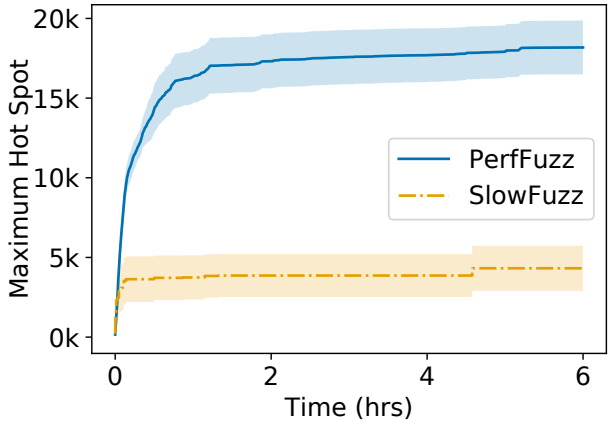
69x



`libxml2`

`libjpeg-turbo`

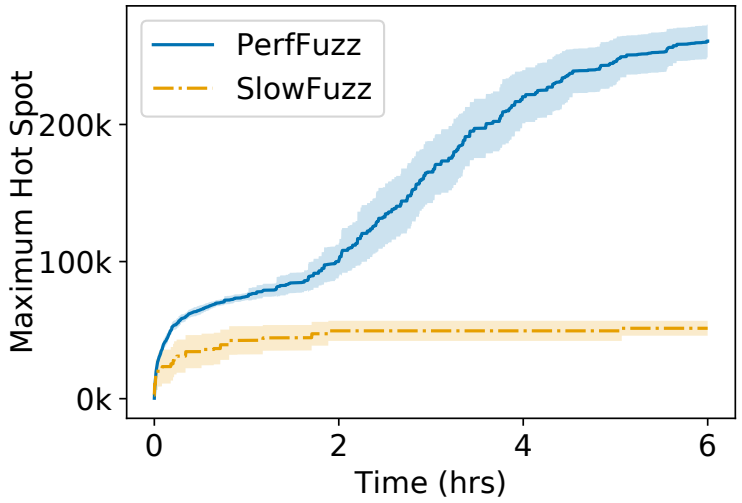




`zlib`

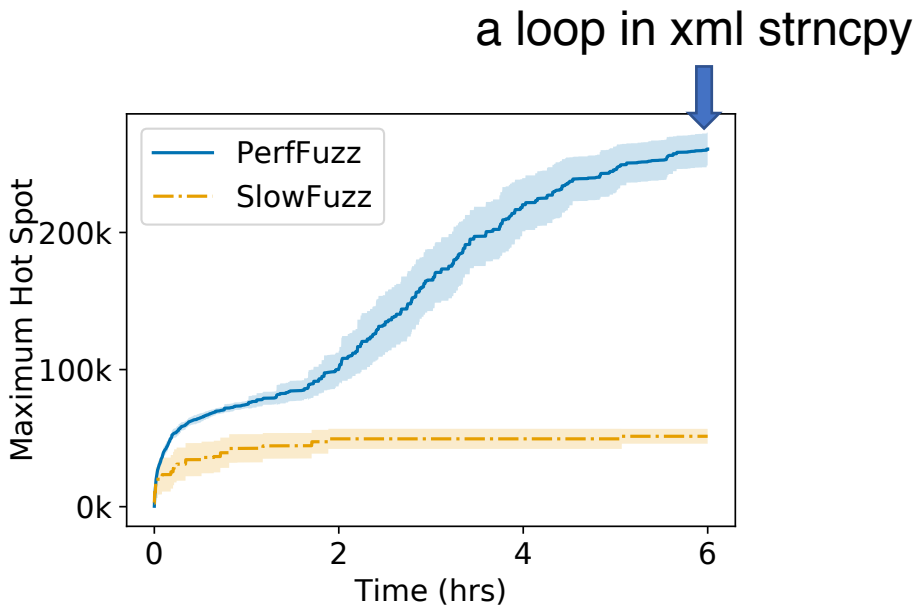
What Does It Mean?

libxml2 case study:



What Does It Mean?

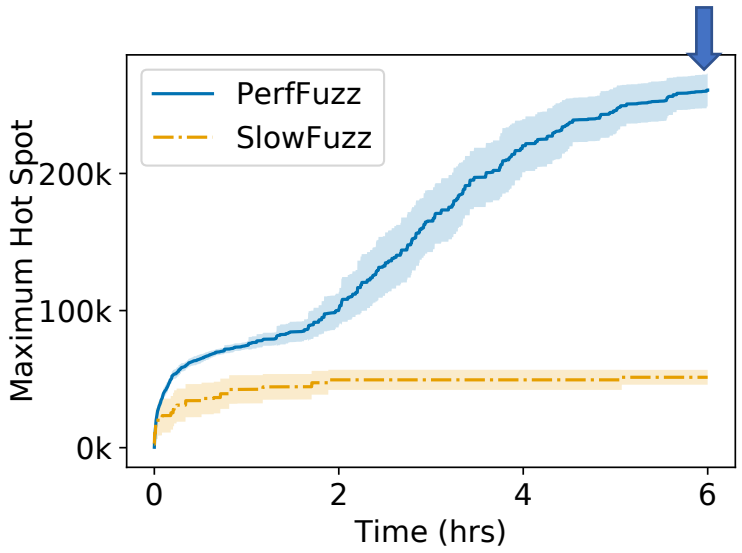
libxml2 case study:



What Does It Mean?

libxml2 case study:

a loop in xml strncpy



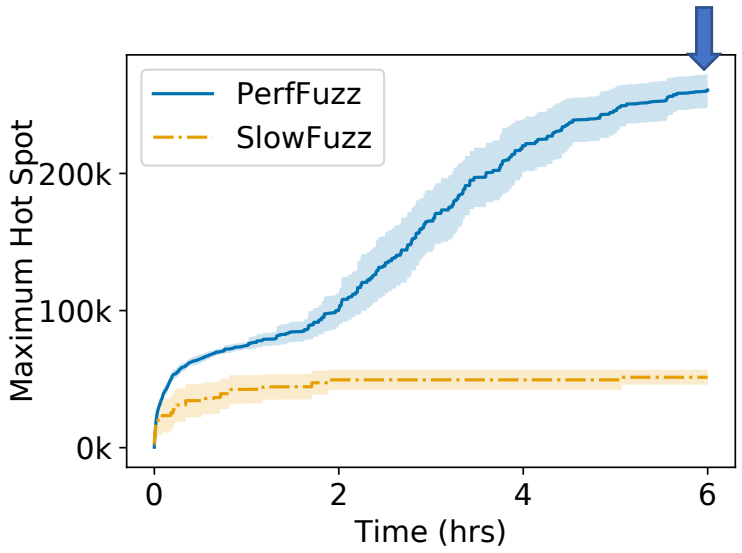
output of read XML on that input:

```
parser error : Double hyphen within comment: <!--3
<a>>>>0>>>#>G<!--3---6----------------------------4-------------
                  ^
parser error : Double hyphen within comment: <!--3---6
<a>>>>0>>>#>G<!--3---6----------------------------4-------------
                    ^
parser error : Double hyphen within comment: <!--3---6--
<a>>>>0>>>#>G<!--3---6----------------------------4-------------
                      ^
parser error : Double hyphen within comment: <!--3---6----
<a>>>>0>>>#>G<!--3---6----------------------------4-------------
                        ^
parser error : Double hyphen within comment: <!--3---6------
<a>>>>0>>>#>G<!--3---6----------------------------4-------------
                          ^
parser error : Double hyphen within comment: <!--3---6--------
<a>>>>0>>>#>G<!--3---6----------------------------4-------------
                            ^
parser error : Double hyphen within comment: <!--3---6----------
<a>>>>0>>>#>G<!--3---6----------------------------4-------------
                              ^
parser error : Double hyphen within comment: <!--3---6-----------
<a>>>>0>>>#>G<!--3---6----------------------------4-------------
                                ^
```

What Does It Mean?

libxml2 case study:

a loop in xml strncpy



output of read XML on that input:

```
parser error : Double hyphen within comment: <!--3
<a>>>>0>>>#>G<!--3---6----------------------------4------------
                  ^
parser error : Double hyphen within comment: <!--3---6
<a>>>>0>>>#>G<!--3---6----------------------------4------------
                    ^
parser error : Double hyphen within comment: <!--3---6--
<a>>>>0>>>#>G<!--3---6----------------------------4------------
                      ^
parser error : Double hyphen within comment: <!--3---6----
<a>>>>0>>>#>G<!--3---6----------------------------4------------
                        ^
parser error : Double hyphen within comment: <!--3---6------
<a>>>>0>>>#>G<!--3---6----------------------------4------------
                          ^
parser error : Double hyphen within comment: <!--3---6--------
<a>>>>0>>>#>G<!--3---6----------------------------4------------
                            ^
parser error : Double hyphen within comment: <!--3---6----------
<a>>>>0>>>#>G<!--3---6----------------------------4------------
                              ^
parser error : Double hyphen within comment: <!--3---6------------
<a>>>>0>>>#>G<!--3---6----------------------------4------------
                                ^
```

quadratic complexity

Experimental Setup: Micro-Benchmarks

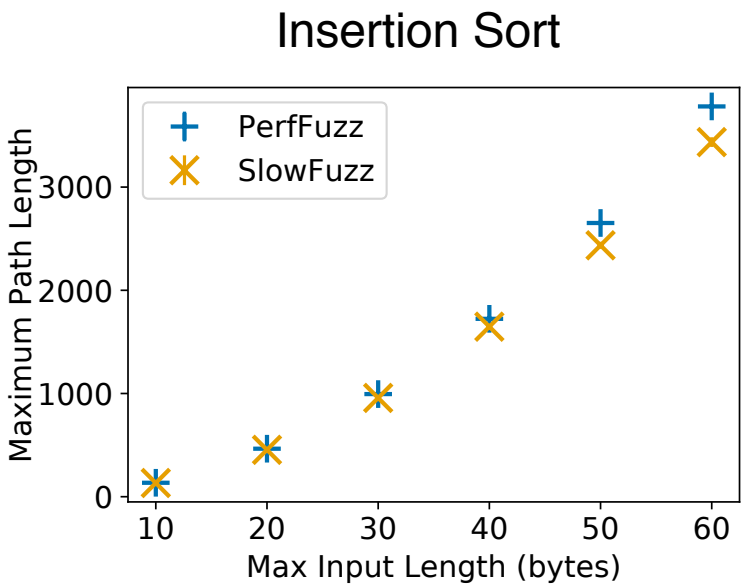
- Choose benchmarks with known worst-case complexity:

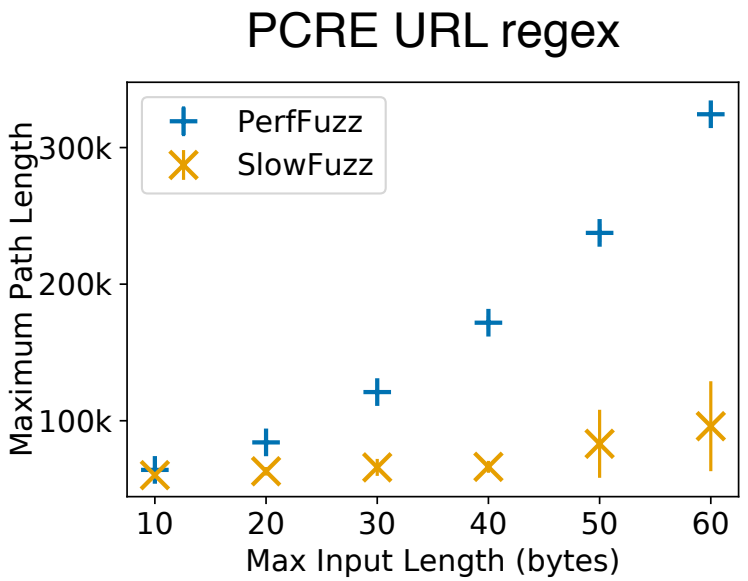
Micro-benchmark	Complexity	Seed	Timeout
Insertion sort (SlowFuzz example)	$\mathbf{n^2}$ n = input len	List of 0's	10 min
PCRE regex match (URL regex)	$\mathbf{n^2}$ n = input len	Null string	60 min
wf-0.41 (Fedora Linux)	$\mathbf{m^2}$ m = num words	"the quick brown fox jumps over the lazy dog"	60 min

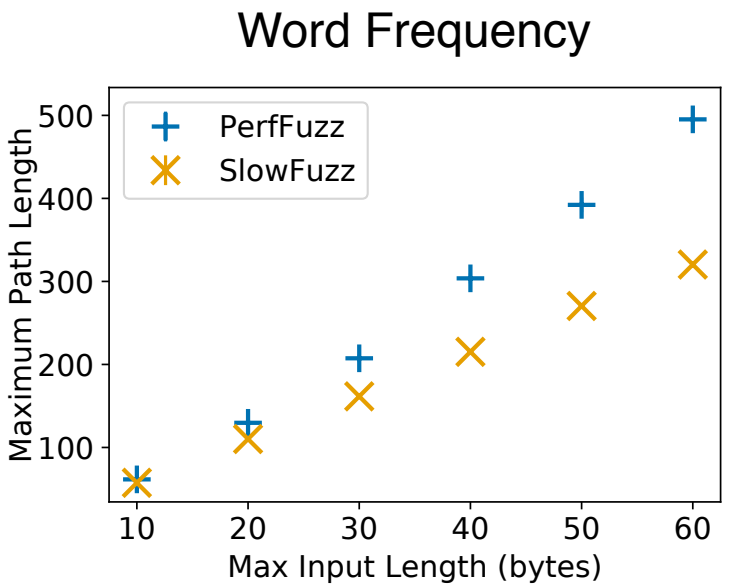
- Repeat 20 runs for each input length: 10, 20, …, 60 bytes.

Micro-Benchmarks: Algorithmic Complexity

- Maximum path length for varying input sizes

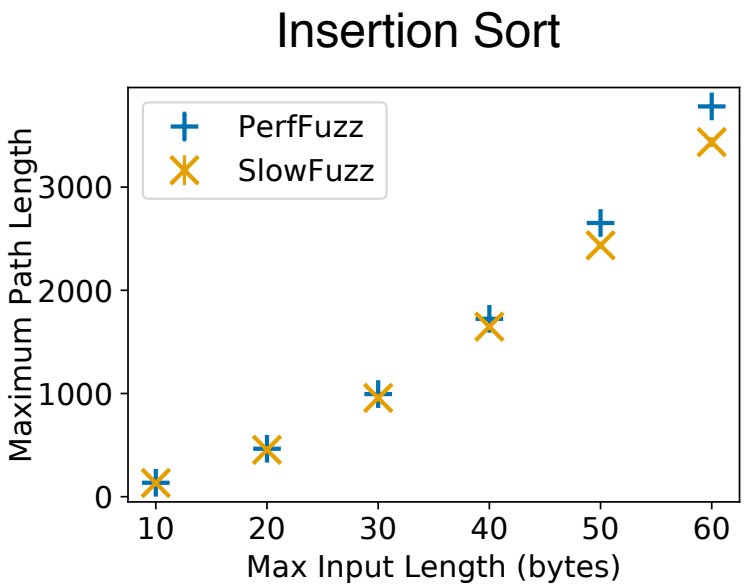


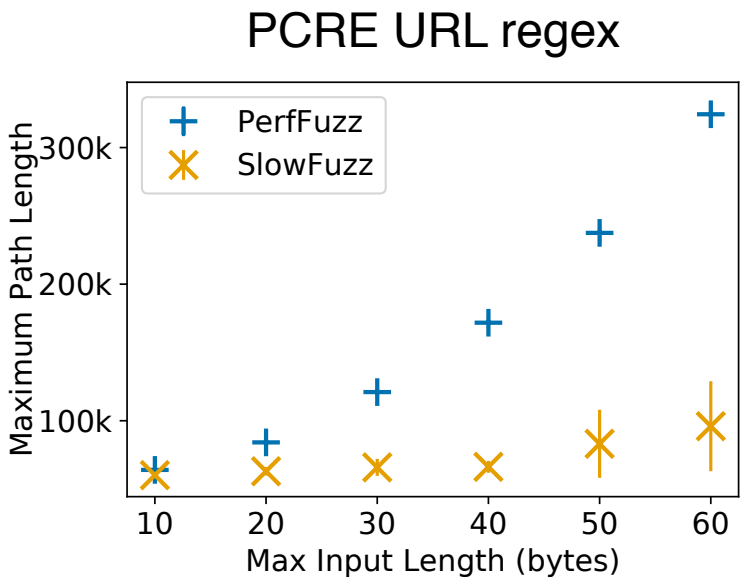


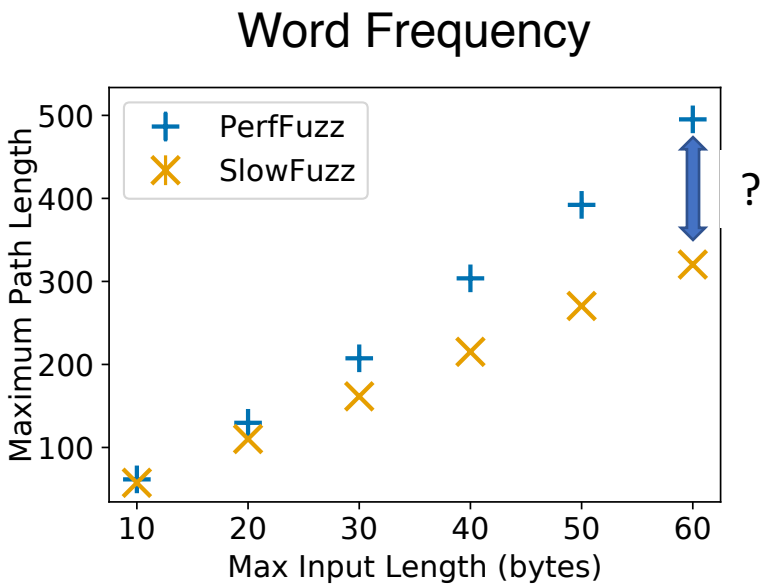


Micro-Benchmarks: Algorithmic Complexity

- Maximum path length for varying input sizes







Back to our Motivating Example

- SlowFuzz worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

- PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```

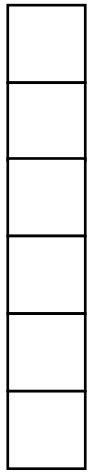
Back to our Motivating Example

- SlowFuzz worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

- PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```



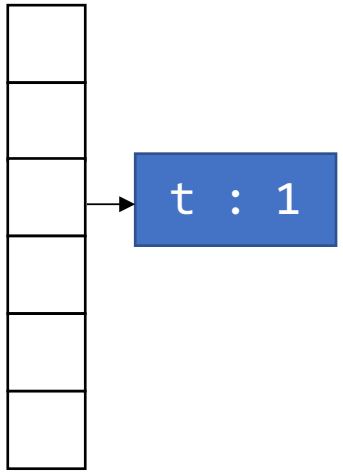
Back to our Motivating Example

- SlowFuzz worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

- PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```



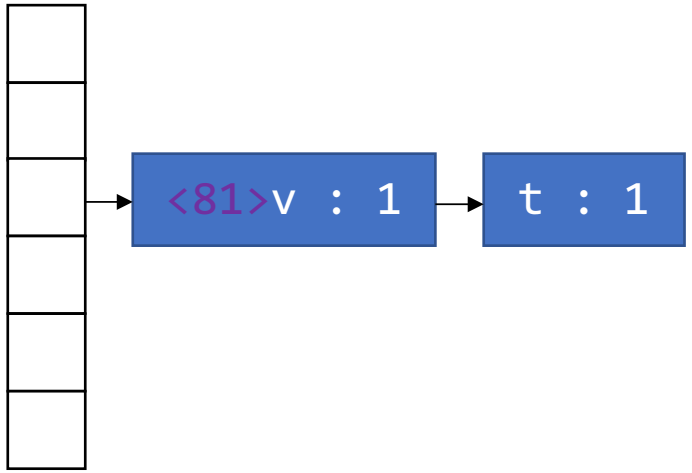
Back to our Motivating Example

- SlowFuzz worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

- PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```



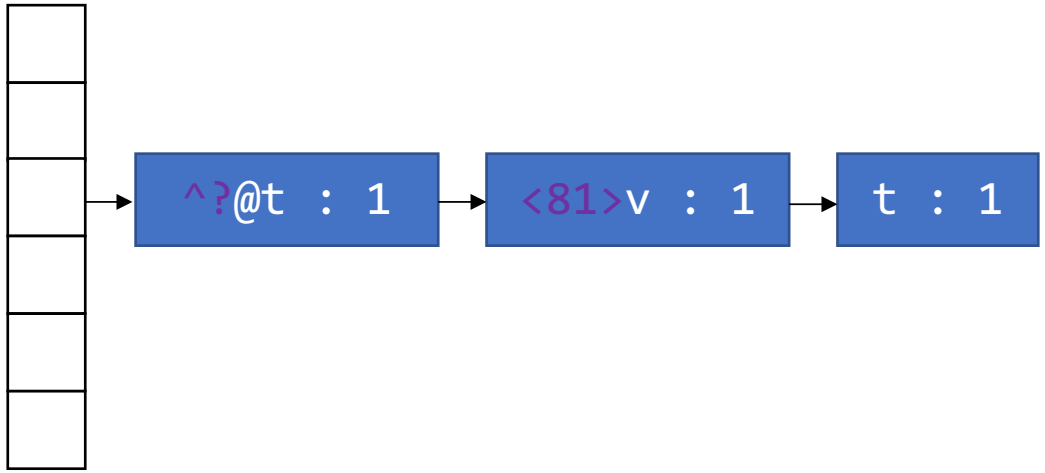
Back to our Motivating Example

- SlowFuzz worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

- PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```



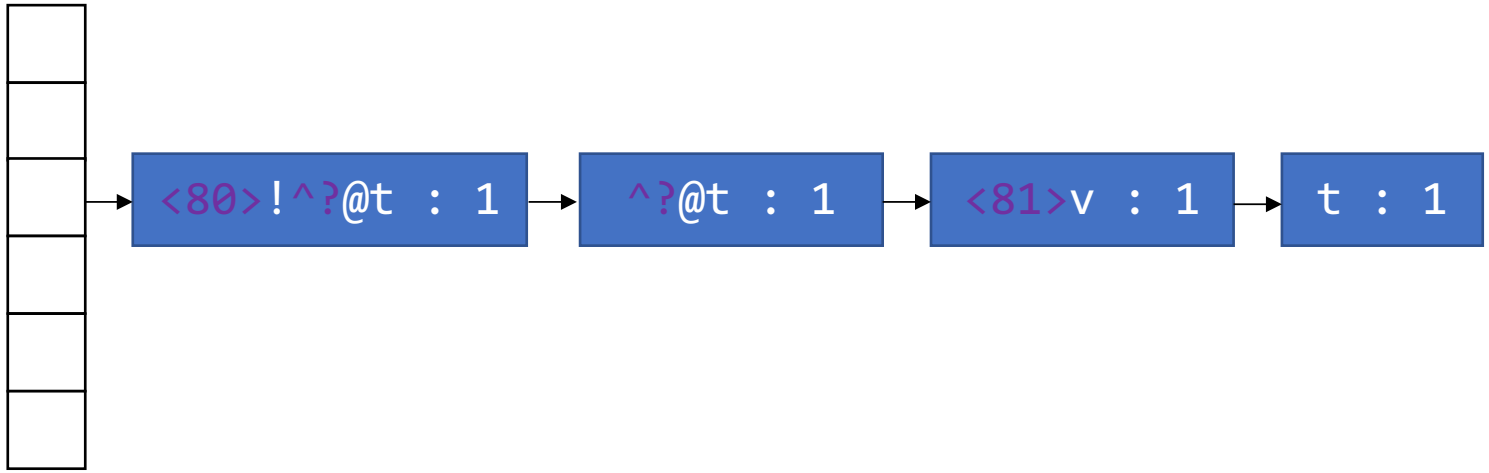
Back to our Motivating Example

- SlowFuzz worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

- PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```



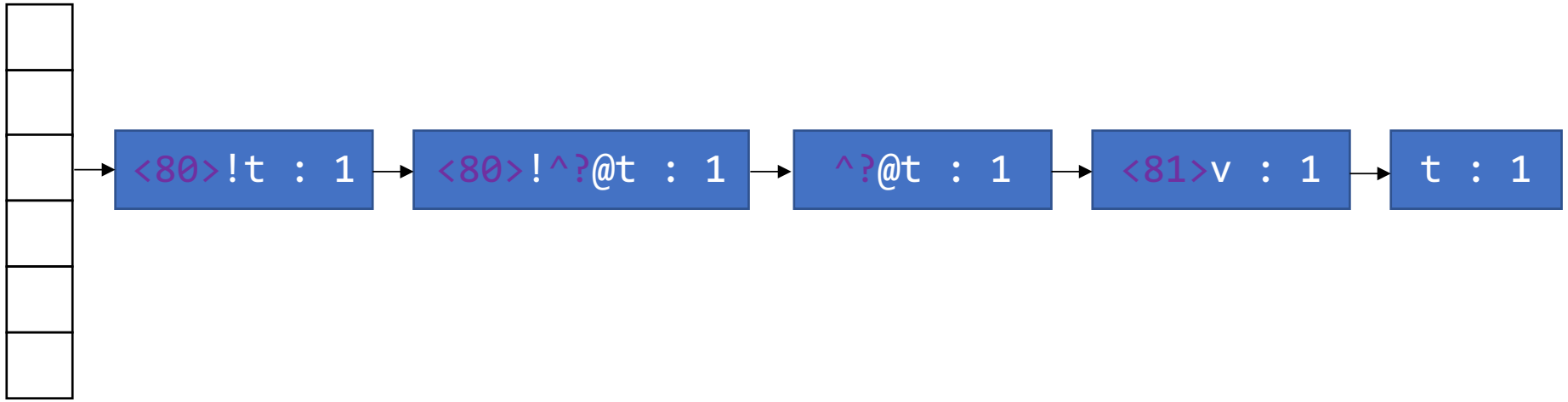
Back to our Motivating Example

- SlowFuzz worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

- PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```



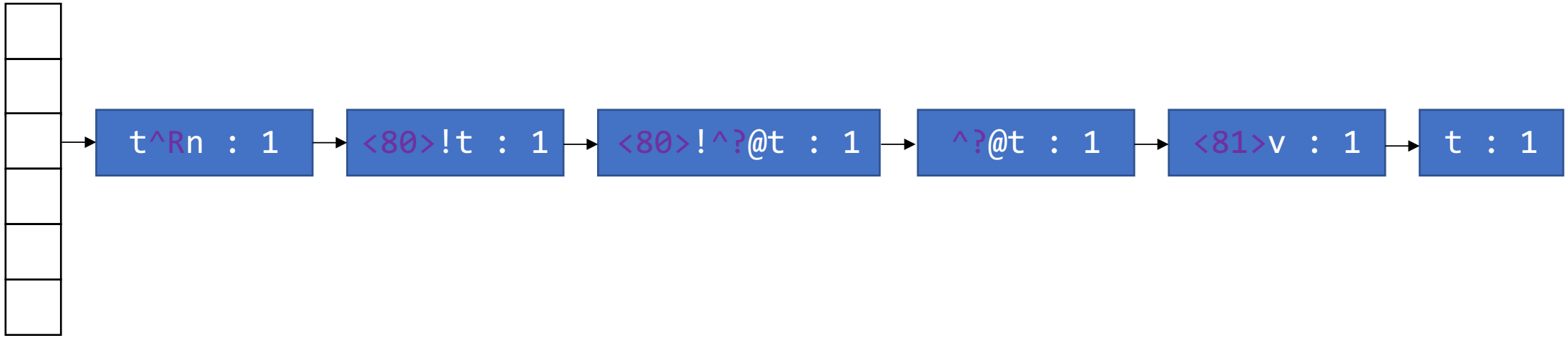
Back to our Motivating Example

- SlowFuzz worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

- PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```



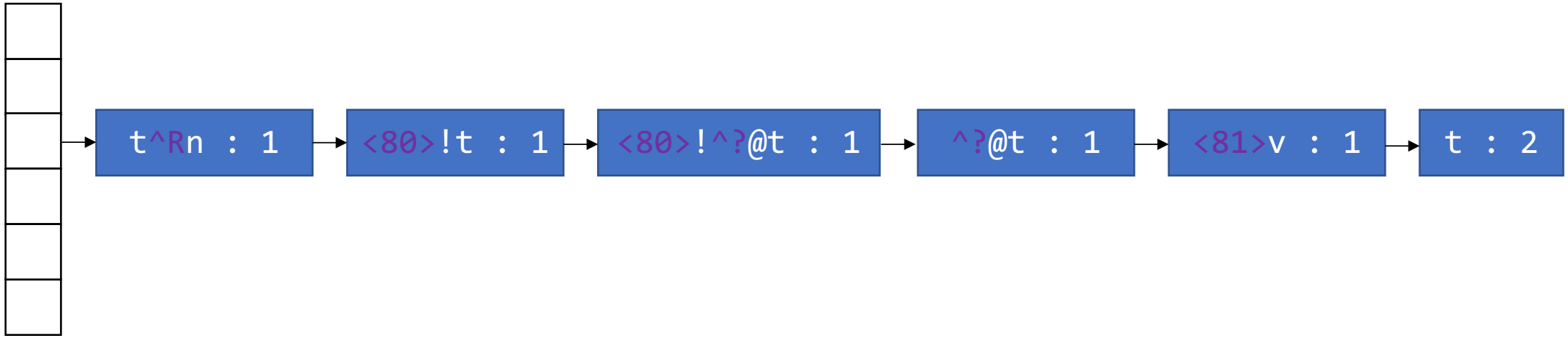
Back to our Motivating Example

- SlowFuzz worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

- PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```



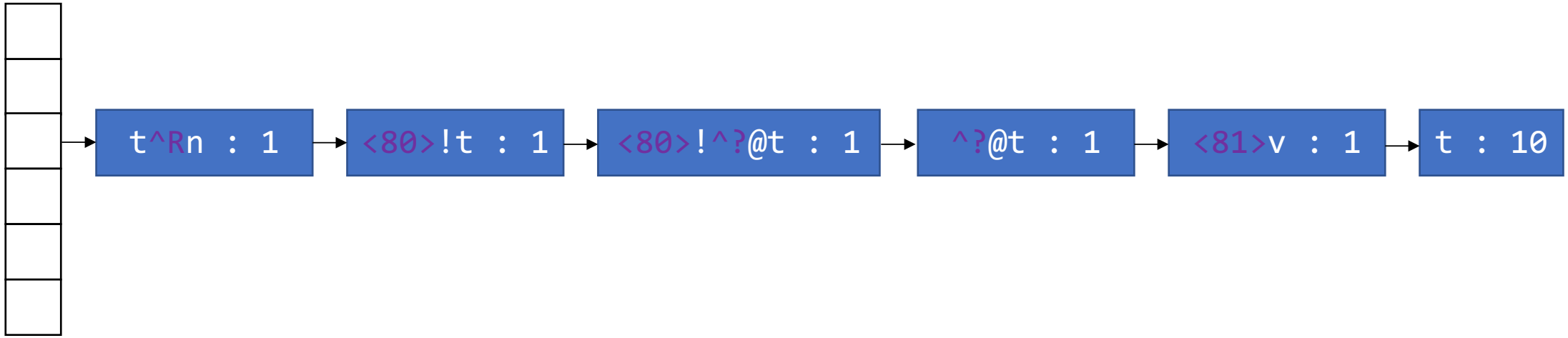
Back to our Motivating Example

- SlowFuzz worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

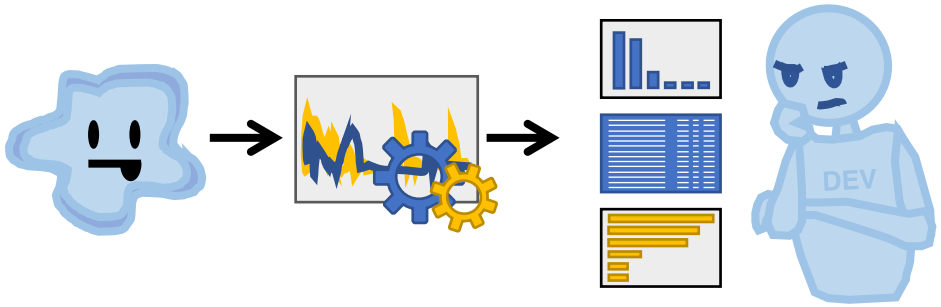
- PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```



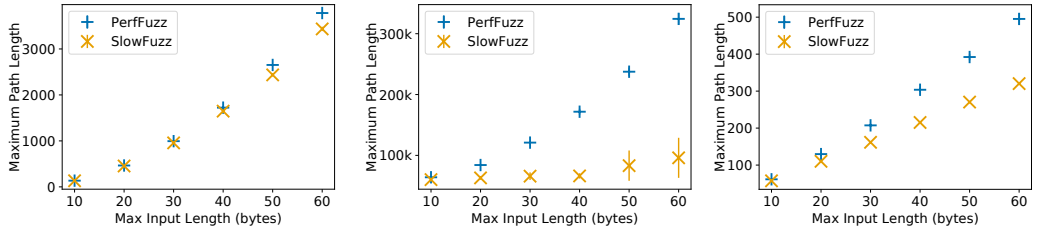
Conclusion

How to find **pathological inputs**?



Use **feedback-directed mutational fuzzing!**

Multi-dimensional feedback more effective.



Where's the code?

https://github.com/carolemieux/perffuzz