Debug Report

`main` — 2026-02-28

delve-helper

Contents

1 Debug Report — main — 2026-02-28

1.1 Hypothesis

Suspected location: `pipeline.go:27-28`

Expected: end clamped to len(data) so all 16 filtered values are reachable

Actual: end clamped to len(data)-1, dropping the last element; window[6] and window[7] get wrong data

1.2 Debugging Trace

#	Action	Location	Reasoning
1	set	`pipeline.go:27`	guard condition uses len(data)-1; suspect off-by-one clamp
2	set	`pipeline.go:27 if start==12`	window[6] starts at index 12; this is where truncation should manifest
3	hit	`pipeline.go:27 (bp 3, start=12)`	end=16 clamped to 15 by len(data)-1 guard; window[6] loses its last element
4	verify	`pipeline.go:27 (post-fix, start=12)`	end=16 after next; no clamping; go test passes all 8 windows

1.3 Breakpoints & Evidence

1.3.1 pipeline.go:27

Source context:

```
25 for start := 0; start < len(data); start += step {
26     end := start + size
27     if end > len(data)-1 {
28         end = len(data) - 1
29     }
```

Args:

```
data =
size = 4
step = 2
~r0 =
```

Locals:

```
windows =
start = 12
end = 16
```

Stack:

```
#0 main.Window
↪  /Users/agent/Desktop/go-debug-skill/examples/failing_test/pipeline.go:27
```

```
#1 main.main /Users/agent/Desktop/go-debug-skill/examples/failing_test/main.go:45
#2 runtime.main /usr/local/go/src/runtime/proc.go:283
#3 runtime.goexit /usr/local/go/src/runtime/asm_arm64.s:1223
```

Print `end, len(data), len(data)-1`:

```
end=end = 16  len(data)=len(data) = 16
```

Observation: At start=12, end=16 (12+4), len(data)=16, len(data)-1=15. Guard 'end > 15' fires and clamps end to 15, slicing data[12:15] (3 elements) instead of data[12:16] (4 elements). window[6] is truncated by 1.

Root Cause

pipeline.go:27-28 — Window() uses 'len(data)-1' as both the guard threshold and the clamp value. When end == len(data) (a valid full boundary), the guard fires and reduces end by 1, producing a slice that is one element too short. The correct boundary is len(data) (no -1).

Fix Applied

pipeline.go:27-28 — changed guard from 'end > len(data)-1' to 'end > len(data)' and clamp from 'len(data)-1' to 'len(data)'. This allows end to reach the true slice boundary, restoring all 4 elements in window[6] and the correct 2-element window[7].

1.4 Post-fix Verification

Confirmed: end correctly equals 16 at start=12. go test -count=1 ./... passes all 8 windows (ok 0.005s).