

Text buffer evaluation

Evaluating and improving data structures for high-performance text editing

Sanyam Singh Brar

Supervisor: Joel Mackenzie

INSERT XY

ab|cd



2 bytes inserted

Edit locality and accumulated changes

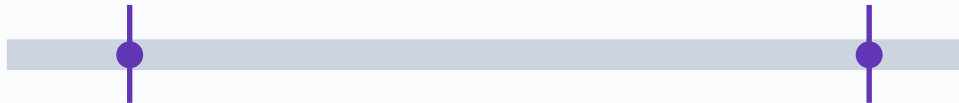
Hold the edit fixed. Vary locality and accumulated history.

01 EDIT LOCALITY

Nearby typing



Distant corrections



02 EDIT HISTORY

Original text



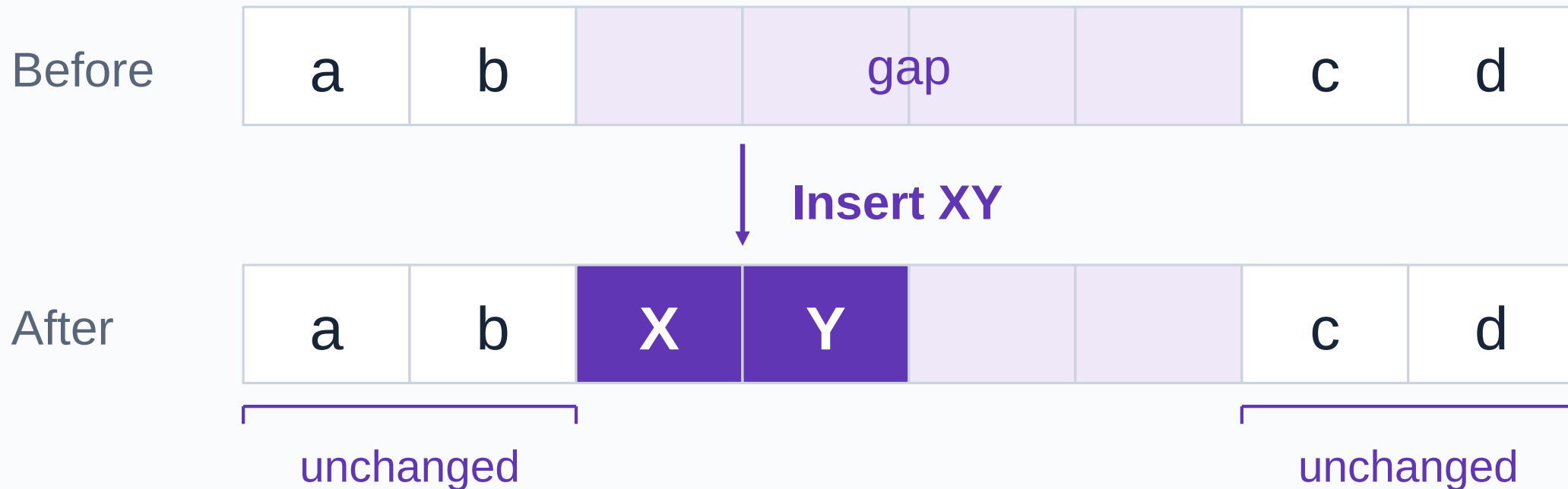
Same text

**Insert, then delete.
Repeat.**

Measure latency, bytes moved and retained storage.

Gap buffer insertion

Gap buffer / spare capacity at the edit position



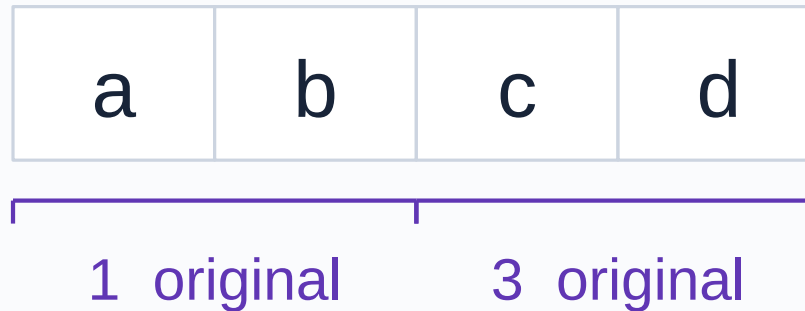
Distant insertion relocates the gap through intervening text.

Piece table insertion

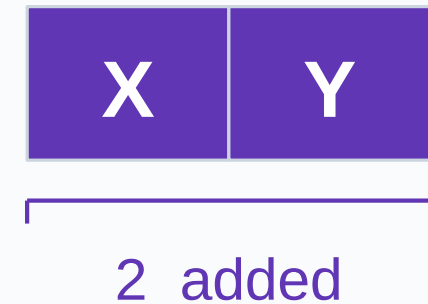
Output: abXYcd / original bytes stay in place

STORAGE

Original



Added



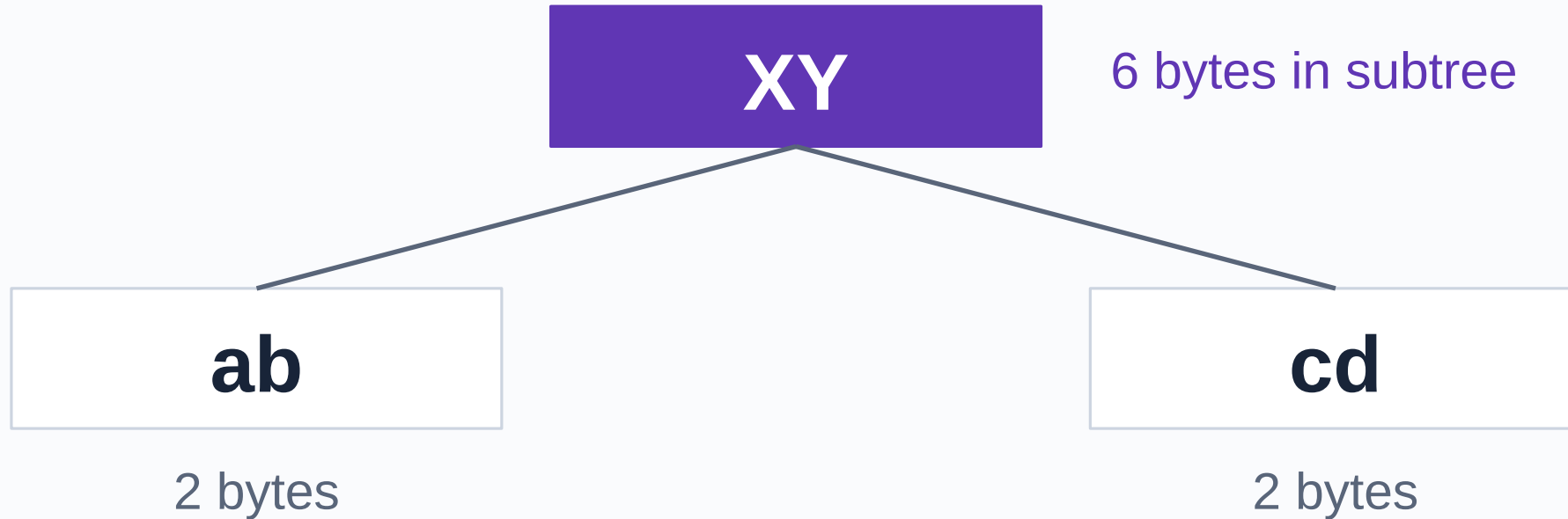
READ IN ORDER



Deleted text can disappear from the sequence but remain in storage.

Rope chunk lookup

Treap chunks and cached subtree summaries



In-order text: ab XY cd

Subtree counts guide text and line lookup.

Editor design choices

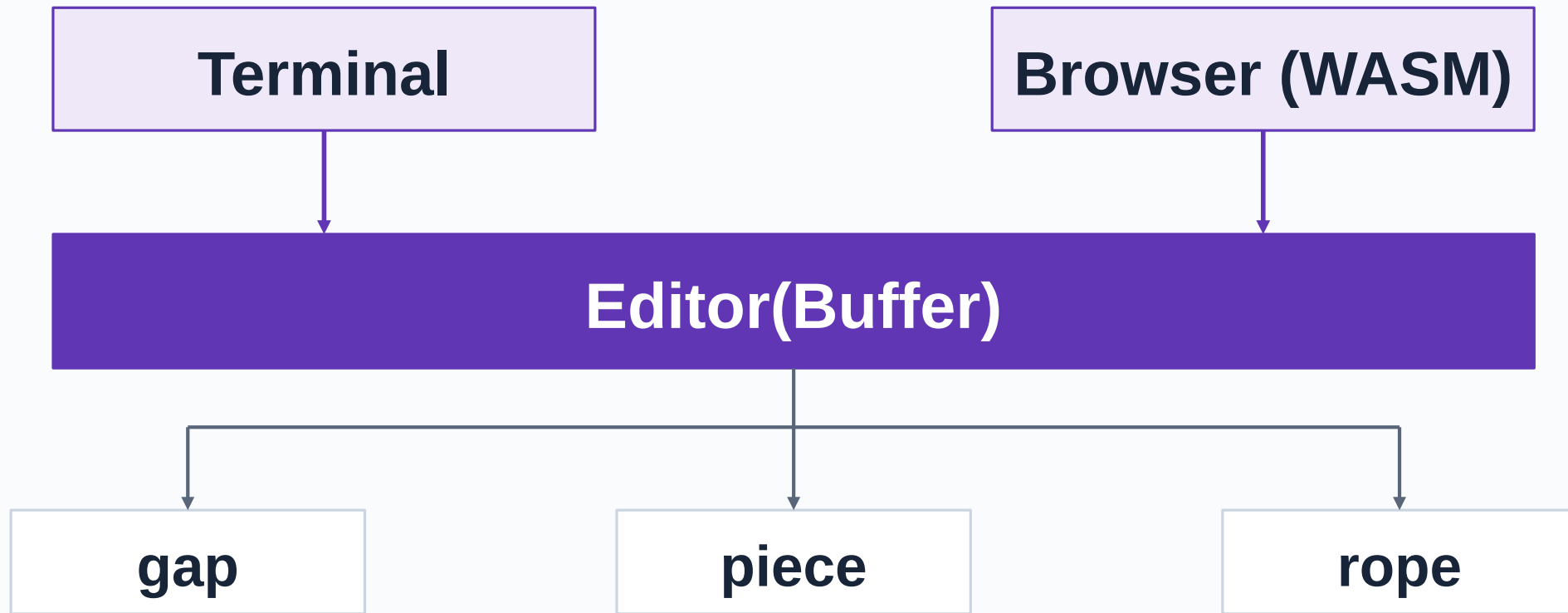
Compare storage and indexing with the same editing operations.

SYSTEM	DESIGN CHOICE	QUESTION TO TEST
VS Code	Tree-indexed pieces	How is text located?
JumpRope	Gap-buffer leaves	Where does text move?
Zed	Subtree summaries	What must be updated?

Crowley: compare structures through the same editing operations.

Editor implementation

UTF-8 editing, file handling and twelve compiled buffer variants



3 backends × 4 cache layouts

Buffer API checks

Generic reference checks across all twelve buffer types

- 01 Insert and delete text
- 02 Replace a byte range
- 03 Read bytes and line starts
- 04 Test allocation failures



REFERENCE MODEL

Byte output
LF metadata

12 buffer variants

Generic checks cover all twelve variants.

Terminal validation

Recorded PTY output / rope backend / ReleaseSafe

```
Buffer experiment
```

```
abXYcd
```

```
Same text. Different storage.
```

```
~
```

```
~
```

```
reit | rope | demo.txt | 3:5 | modified
```

```
^S save | ^Q quit | arrows move | Home/End |
```

01

Insert XY

02

Cursor at 3:5

03

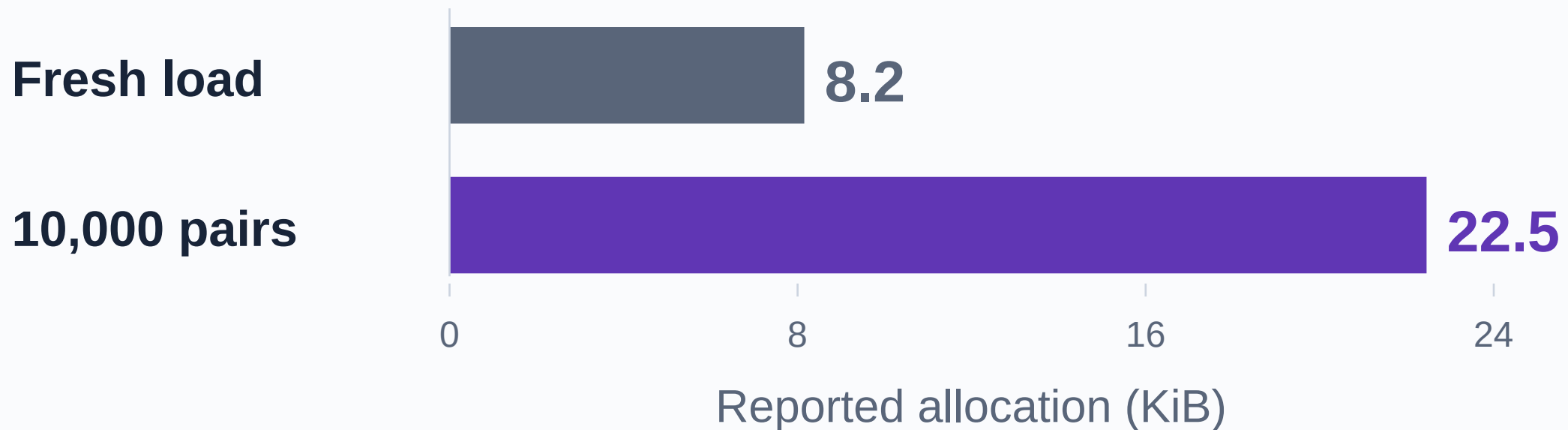
Modified state

Insert XY: `ab|cd` → `abXY|cd`

Piece table retained storage

Recorded PieceTable retained capacity / 5 October 2026

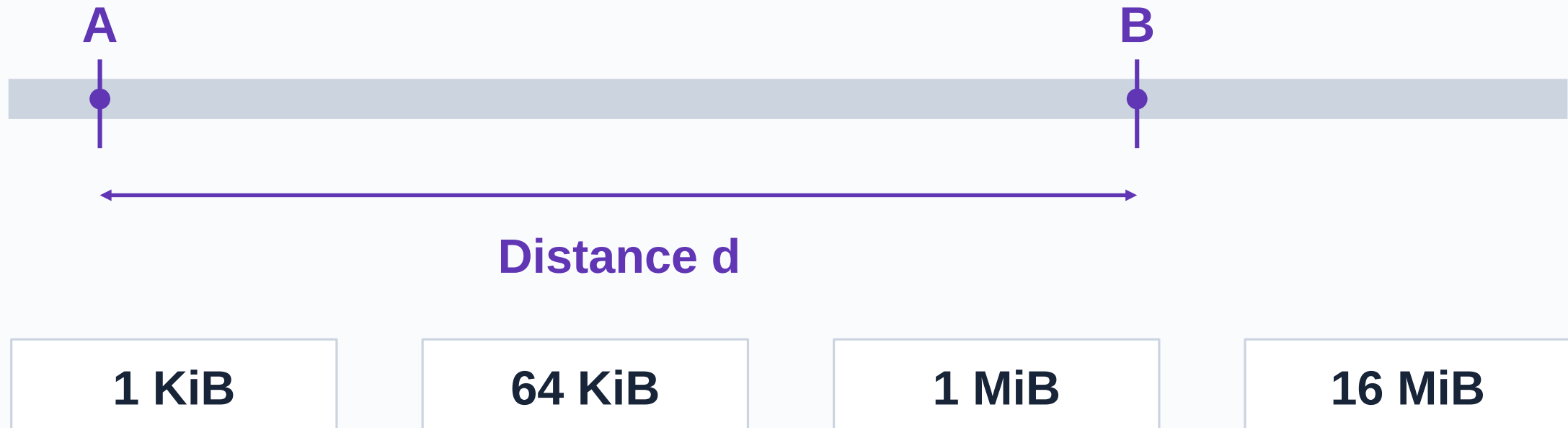
Same 8 KiB document · One live piece before and after



Fixed position · insert x, then backspace · 10,000 repetitions

Edit-distance experiment

Proposed: alternate insert/backspace pairs between A and B (32 MiB file)



Measure latency. Test whether bytes moved explain the trend.

Compaction experiment

Proposed variant / test only if retained storage matters at target scale

Retained storage



Live text + unreferenced storage

After candidate compaction



Same logical text

Possible saving

Rebuild work + pauses

Compare retained storage, latency and semantic equivalence.

Project deliverables

Planning estimates from the proposal

01 Nov–Dec 2026

Validate the harness

Operation timing + counters

02 Dec 2026–Jan 2027

Explain the baselines

Distance, file size and edit history

03 Jan–Mar 2027

Test one intervention

Benefit, regressions and conditions

Then: loading and persistence costs. Prioritise one explained result.

Progress and next steps

01

Built

Native and browser
buffer editing

02

Observed

Identical text;
retained storage

03

Next

Measure distance;
test one change

Questions

Progress checkpoint · implemented testbed, bounded observation, planned experiments

Structural probe method

Historical 5 October probe / eight runs across four then-active backends

INPUT

8,192 ASCII bytes
Cursor 4,097

CONDITIONS

Fresh load
10,000 insert/backspace pairs

8,352  **22,997 bytes**

$\text{original.len} + \text{add.capacity} + \text{pieces.capacity} \times \text{sizeof(Piece)}$

Capacity snapshot; excludes RSS, allocator overhead and peaks.

Checks and limitations

Current generic API and frontend checks

12	Compiled buffer variants
3	Generic Zig test blocks
24	PTY sessions per build mode

UTF-8 scalars · explicit file save · three backends

Build and run

Native editor

```
zig build install -Doptimize=ReleaseSafe  
./zig-out/bin/reit --backend rope notes.txt
```

Browser editor

```
zig build web  
https://reit4842-editor.pages.dev
```

Primary sources

[1] Crowley · 1998

Data Structures for Text Sequences

[2] Lyu · 2018

Text Buffer Reimplementation

[3] JumpRope · 1.1.2

Library documentation

[4] Ball et al. · 2024

Rope & SumTree

Preparation and review

Course guidance

15-minute seminar

Clear visuals, relevant evidence, concise wording

Preparation record

AI-assisted code, diagrams, notes and independent reviews

Student source review and timed rehearsal remain