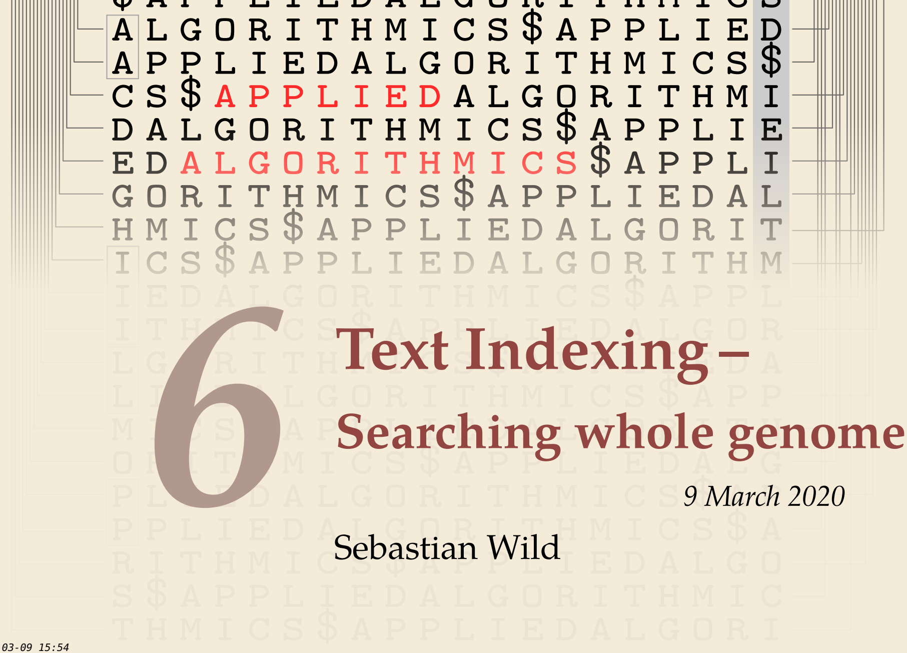


6 Text Indexing – Searching whole genomes

9 March 2020

Sebastian Wild

version 2020-03-09 15:54

Outline

6 Text Indexing

6.1 Motivation

6.2 Suffix Trees

6.3 Applications

6.4 Longest Common Extensions

6.5 Suffix Arrays

6.6 The LCP Array

6.1 Motivation

Text indexing

- *Text indexing* (also: *offline text search*):
  - case of string matching: find $P[0..m-1]$ in $T[0..n-1]$
  - but with *fixed* text ⇝ preprocess T (instead of P)
  - ⇝ expect many queries P, answer them without looking at all of T
  - ⇝ essentially a data structuring problem: “building an *index* of T”

    Latin: “one who points out”

- application areas
  - web search engines
  - online dictionaries
  - online encyclopedia
  - DNA/RNA data bases
  - . . . searching in any collection of text documents (that grows only moderately)

Inverted indices

same as “indexes”

- original indices in books: list of (key) words ↦ page numbers where they occur
- assumption: searches are only for **whole** (key) **words**

⇝ often reasonable for natural language text

Inverted indices

same as "indexes"

- original indices in books: list of (key) words ↦ page numbers where they occur
- assumption: searches are only for **whole** (key) **words**

⇝ often reasonable for natural language text

Inverted index:

- collect all words in T
    - can be as simple as splitting T at whitespace
    - actual implementations typically support *stemming* of words
      `goes` ⟶ `go`, `cats` ⟶ `cat`

not here

- store mapping from words to a list of occurrences ⇝ *how?* — BST

$\text{go} \longmapsto \{5, 10, 20\}$

$\text{cat} \longmapsto \{4, 21\}$

Clicker Question

Do you know what a *trie* is?

- **A** A what? No!
- **B** I have heard the term, but don’t quite remember.
- **C** I remember hearing about it in a module.
- **D** Sure.

`pingo.upb.de/622222`

Tries

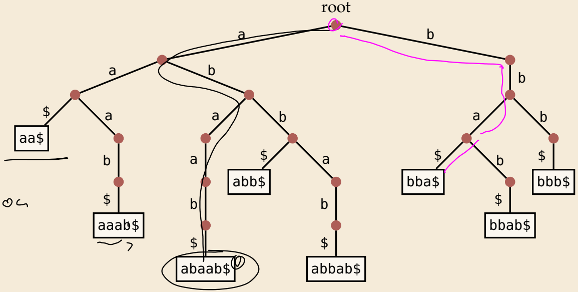
{aa, a}

- efficient dictionary data structure for strings
- name from re**trie**val, but pronounced “try”
- tree based on symbol comparisons

- **Assumption:** stored strings are *prefix-free* (no string is a prefix of another)
  - strings of same length ✓
  - strings have “end-of-string” marker $ ✓

some character ∉ Σ

- **Example**:
  {aa$, aaab$, abaab$, abb$, abbab$, bba$, bbab$, bbb$}

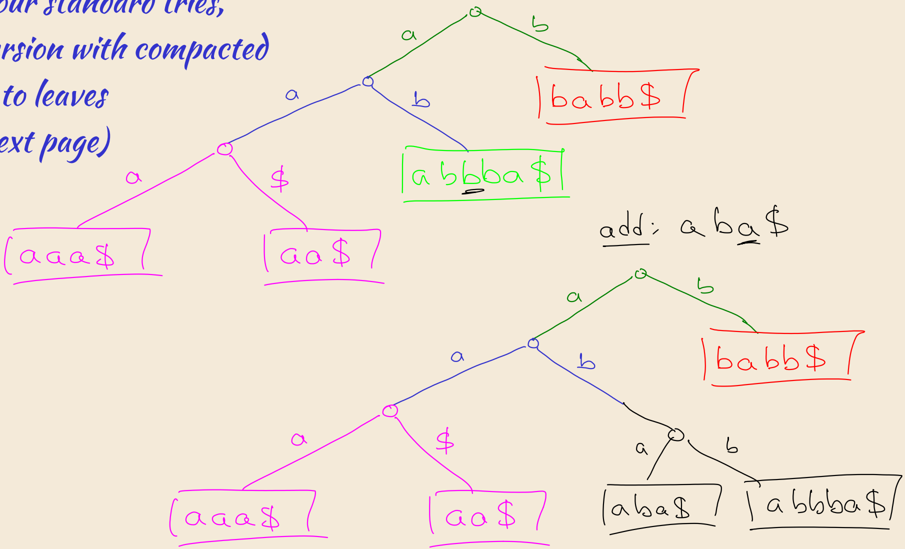


○ construction: top-down
independent of order of insertion

○ query: (get) ex: bba$

Construction {aaa$, babb$, aa$, abbba$}

NOT our standard tries, but version with compacted paths to leaves (see next page)



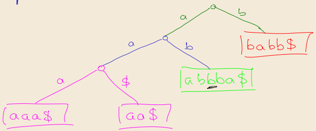
add: aba$

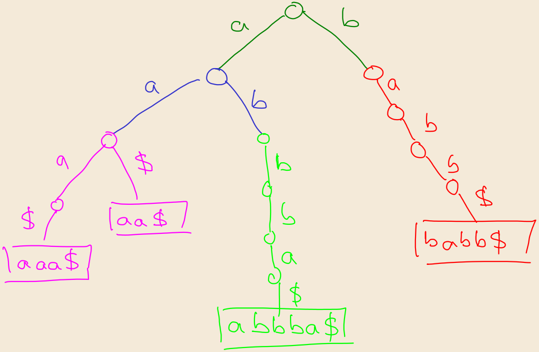
Trie construction (correct version)

{aaa$, babb$, aa$, abbba$}

standard trie

trie with compacted paths to leaves





Clicker Question

Suppose we have a trie that stores n strings over $\Sigma = \{\texttt{A}, \ldots, \texttt{Z}\}$.
Each stored string consists of m characters.
We now search for a query string Q with $|Q| = q$.
How many **nodes** in the trie are **visited** during this **query**?

- **A** $\Theta(\log n)$
- **B** $\Theta(\log(nm))$
- **C** $\Theta(m \cdot \log n)$
- **D** $\Theta(m + \log n)$
- **E** $\Theta(m)$
- **F** $\Theta(\log m)$
- **G** $\Theta(q)$
- **H** $\Theta(\log q)$
- **I** $\Theta(q \cdot \log n)$
- **J** $\Theta(q + \log n)$

pingo.upb.de/622222

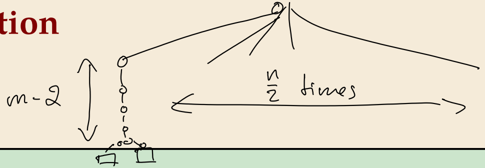
Clicker Question

Suppose we have a trie that stores n strings over $\Sigma = \{\texttt{A}, \ldots, \texttt{Z}\}$.
Each stored string consists of m characters.
We now search for a query string Q with $|Q| = q$.
How many **nodes** in the trie are **visited** during this successful **query**?

- **A** ~~$\Theta(\log n)$~~
- **B** ~~$\Theta(\log(nm))$~~
- **C** ~~$\Theta(m \cdot \log n)$~~
- **D** ~~$\Theta(m + \log n)$~~
- **E** ~~$\Theta(m)$~~
- **F** ~~$\Theta(\log m)$~~
- **G** $\Theta(q)$ ✓
- **H** ~~$\Theta(\log q)$~~
- **I** ~~$\Theta(q \cdot \log n)$~~
- **J** ~~$\Theta(q + \log n)$~~

`pingo.upb.de/622222`

Clicker Question



$S_1 = \$_1$ aaaaaaa b

$S_2 = \$_1$ aaaaaaa a

$S_3 = \$_2$

$S_4 = \$_2$ . . .

Suppose we have a trie that stores n strings over $\Sigma = \{\mathtt{A}, \dots, \mathtt{Z}\}$.
Each stored string consists of m characters.
How many **nodes** does the trie have **in total** *in the worst case*?

A $\Theta(n)$

B $\Theta(n + m)$

C $\Theta(n \cdot m)$

D $\Theta(n \log m)$

E $\Theta(m)$

F $\Theta(m \log n)$

`pingo.upb.de/622222`

Clicker Question

Suppose we have a trie that stores n strings over $\Sigma = \{\mathtt{A}, \ldots, \mathtt{Z}\}$.
Each stored string consists of m characters.
How many **nodes** does the trie have **in total** *in the worst case*?

A ~~$\Theta(n)$~~

B ~~$\Theta(n + m)$~~

[✓ C $\Theta(n \cdot m)$]

D ~~$\Theta(n \log m)$~~

E ~~$\Theta(m)$~~

F ~~$\Theta(m \log n)$~~

pingo.upb.de/622222

Compact tries

∘ △ get needs extra check

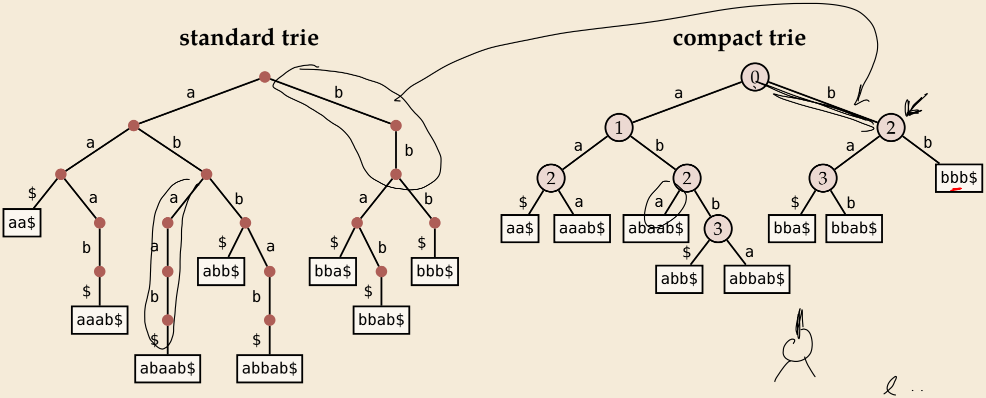
bab$

=1 child

- ► compress paths of unary nodes into single edge
- ► nodes store index of next character

standard trie

compact trie



⇝ searching slightly trickier, but same time complexity as in trie

- ► all nodes ≥ 2 children ⇝ #nodes ≤ #leaves = #strings ⇝ linear space

$\Theta(n)$ not $\Theta(nm)$