# String Processing

Jill-Jênn Vie  Halim book

October 18, 2024

# Rabin-Karp: hashing

Looking for $s$ in $t$

## Idea

Comparing an updated rolling hash of every substring of t of size $|s|$ with the hash of s.

$$\text{hash}(x) = \sum_i x[i]A^i \pmod{P}$$

## Applications

- Pattern matching
- Lexicographical smallest rotation of a string

```
vector<H> getHashes(string& str, int length) {
	if (sz(str) < length) return {};
	H h = 0, pw = 1;
	rep(i,0,length)
		h = h * C + str[i], pw = pw * C;
	vector<H> ret = {h};
	rep(i,length,sz(str)) {
		ret.push_back(h = h * C + str[i] - pw * str[i-length]);
	}
	return ret;
}

struct HashInterval {
	vector<H> ha, pw;
	HashInterval(string& str) : ha(sz(str)+1), pw(ha) {
		pw[0] = 1;
		rep(i,0,sz(str))
			ha[i+1] = ha[i] * C + str[i],
			pw[i+1] = pw[i] * C;
	}
	H hashInterval(int a, int b) { // hash [a, b)
		return ha[b] - ha[a] * pw[b - a];
	}
};
```

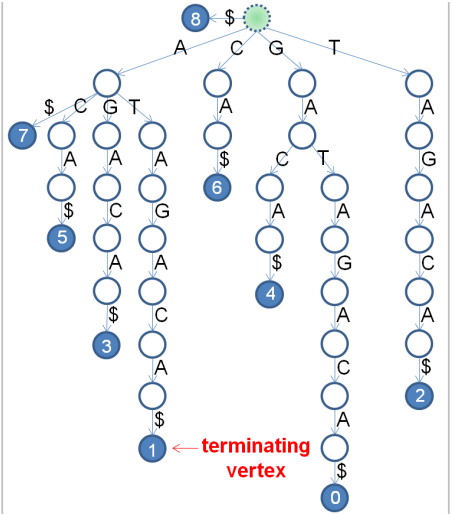
# 2D pattern matching: SWERC 2014's J: The Big Painting


https://open.kattis.com/problems/bigpainting

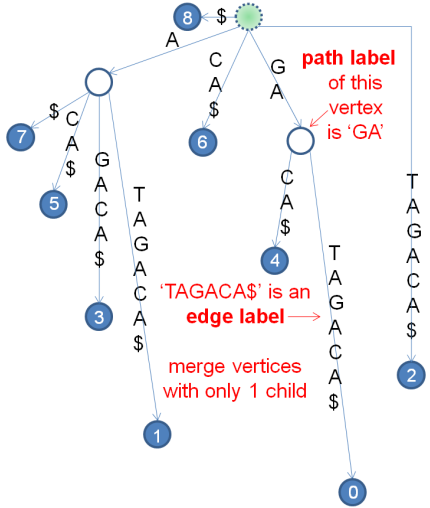
## Suffixes

| i | Suffix |
|---|---|
| 0 | GATAGACA$ |
| 1 | ATAGACA$ |
| 2 | TAGACA$ |
| 3 | AGACA$ |
| 4 | GACA$ |
| 5 | ACA$ |
| 6 | CA$ |
| 7 | A$ |
| 8 | $ |

## Suffix Trie



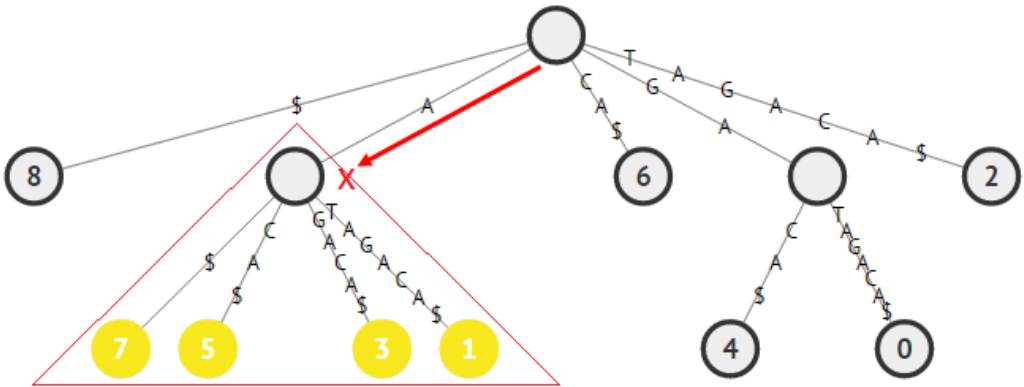

## Suffix Tree




String matching / longest repeated substring / longest common substring of multiple strings

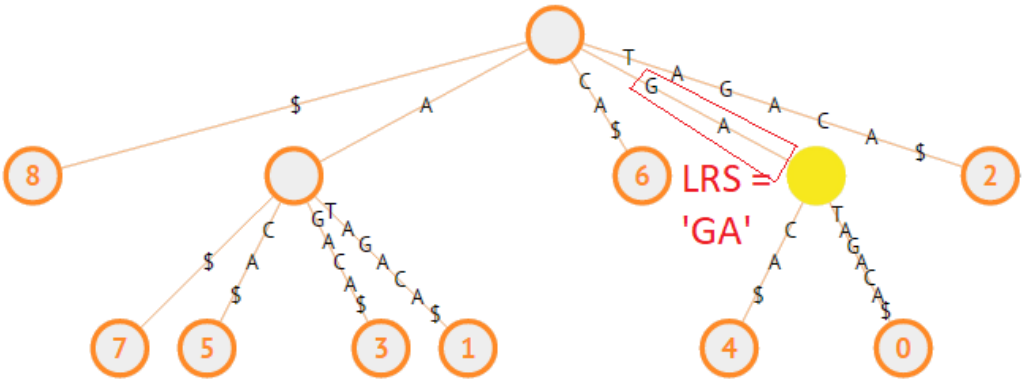
# String matching

A in GATAGACA



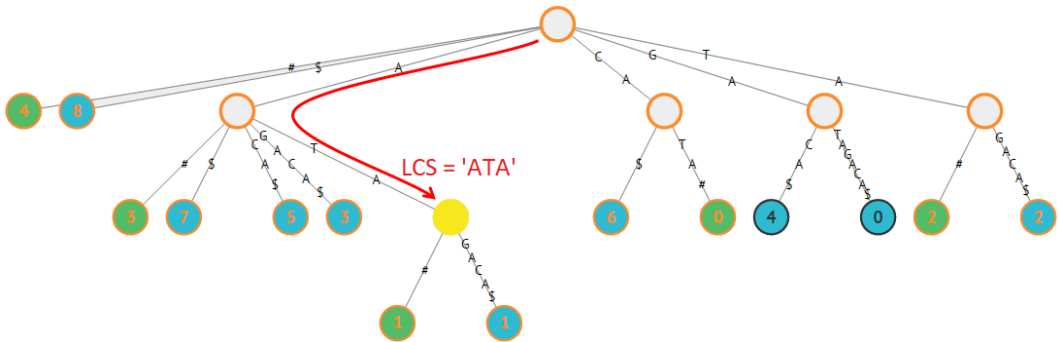
# Longest repeated substring

GATAGACA: it is GA



# Longest common substring

GATAGACA and CATA: it is ATA



# Suffix array

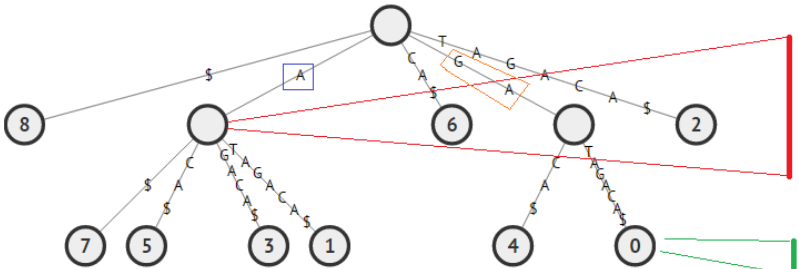
| i | Suffix |
|---|---|
| 0 | GATAGACA$ |
| 1 | ATAGACA$ |
| 2 | TAGACA$ |
| 3 | AGACA$ |
| 4 | GACA$ |
| 5 | ACA$ |
| 6 | CA$ |
| 7 | A$ |
| 8 | $ |

Sort →

| i | SA[i] | Suffix |
|---|---|---|
| 0 | 8 | $ |
| 1 | 7 | A$ |
| 2 | 5 | ACA$ |
| 3 | 3 | AGACA$ |
| 4 | 1 | ATAGACA$ |
| 5 | 6 | CA$ |
| 6 | 4 | GACA$ |
| 7 | 0 | GATAGACA$ |
| 8 | 2 | TAGACA$ |

Naive in $O(n^2 \log n)$, divide in conquer best in practice $O(n \log n)$, best in theory $O(n)$

# Suffix tree vs. suffix array




| i | SA[i] | LCP[i] | Suffix SA[i] |
|---|---|---|---|
| 0 | 8 | 0 | $ |
| 1 | 7 | 0 | A$ |
| 2 | 5 | 1 | ACA$ |
| 3 | 3 | 1 | AGACA$ |
| 4 | 1 | 1 | ATAGACA$ |
| 5 | 6 | 0 | CA$ |
| 6 | 4 | 0 | GACA$ |
| 7 | 0 | 2 | GATAGACA$ |
| 8 | 2 | 0 | TAGACA$ |

Longest common prefix of any two suffixes

# String applications

| | String Hashing | Suffix Array | Suffix Tree |
|---|---|---|---|
| Search for duplicate strings in array of strings | X | X | X |
| **Number of distinct substrings of given string** | **O(n²)** | **O(n log n)** | **O(n)** |
| Finding smallest cyclic shift | X | X | X |
| Finding substring in a string | X | X | X |
| *Comparing same-length substrings a < b* | X | X | X |
| Longest common prefix of substrings | X | X | X |
| Total length of all different substrings | | X | X |
| Lexicographically k-th substring | X | X | X |
| Shortest non-appearing string | | X | X |
| Longest common substring of multiple strings | X | X | X |

# Knuth-Morris-Pratt

Let $s$ a string of length $n$

## Prefix function

Array $p$ such that $p[i]$ is the length of the longest proper prefix of $s[0..i]$ which is also a suffix of $s[0..i]$.

## Idea

Build $p$ in $O(n)$ by dyn prog

## Applications

- String matching
- Know biggest $p$ such that $s = z^p$
- Find all palindrome prefixes
- Given $u, v$, find whether $\exists x, y, u = xy, v = yx$

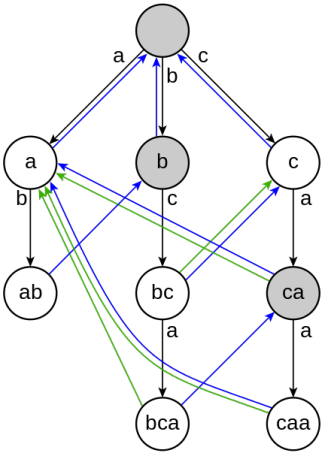
# Looking for a set of patterns $S$

## Generalization

But now, how to find all occurrences of a set of patterns in a string?

# Aho-Corasick

Look for all occurrences of a, ab, bc, bca, c, caa (white nodes)




Blue arrows: suffix links
Green arrows: terminal

## Complexity

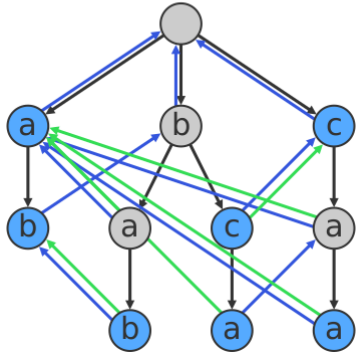
If $\sum$ strings is $m$, nb vertices $n$, alphabet size $k$

- $O(mk)$ thanks to dyn prog
- Can be sped up $O(n \log k)$ with a segment tree

## Note

- Generalization of KMP for several strings
- Notebook implementation is exactly cp-algorithms (wtf 445 pages of Stanford slides)

# Aho–Corasick




Ahocorasick.svg, CC-BY-SA 3.0,
Dllu, Wikimedia Commons

Dictionnaire : a, ab, bab, bc, bca, c, caa.

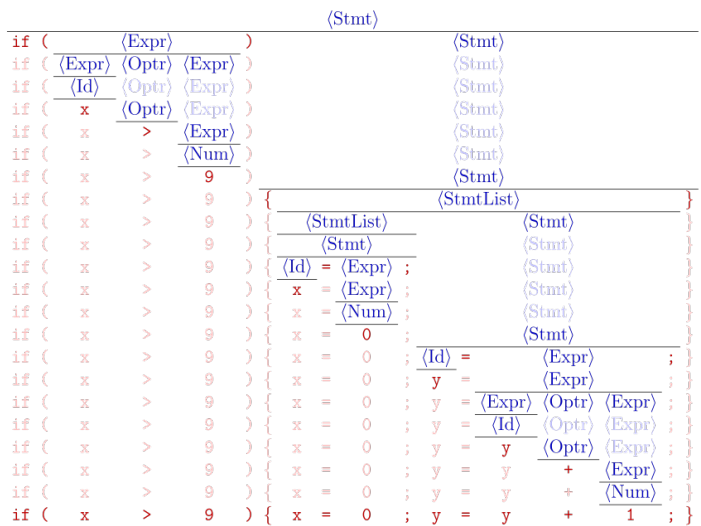
- Construire un **trie** du dictionnaire (liens en noir, mots sur fond bleu)
- Ajouter des **pointeurs** (en bleu) vers le plus grand suffixe strict dans le trie
- Ajouter des **raccourcis** (en vert) pour les mots du dictionnaire
- Exemple : **abccab** donne :
  - **a** (parcours normal),
  - **ab** (parcours normal),
  - **bc** (suivi du pointeur bleu) puis **c** (clôture par le pointeur vert),
  - **c** (suivi du pointeur bleu),
  - **ca** (parcours normal) puis **a** (clôture par le pointeur vert),
  - **ab** (suivi du pointeur bleu)

# Problems using Aho-Corasick

- Find all strings from a given set in a text
- Finding the lexicographical smallest string of a given length that doesn't match any given strings
- Finding the shortest string containing all given strings
- Finding the lexicographical smallest string of length $L$ containing $k$ strings

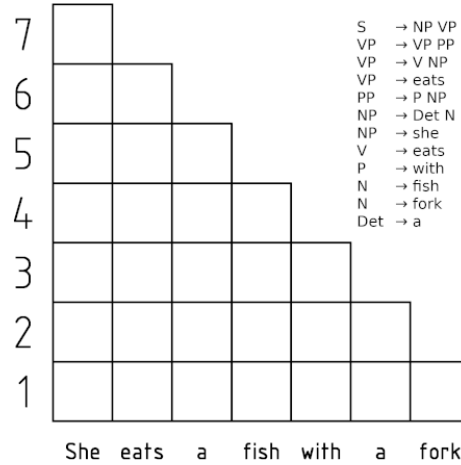
# Context-free grammar

⟨Stmt⟩ → ⟨Id⟩ = ⟨Expr⟩ ;
⟨Stmt⟩ → { ⟨StmtList⟩ }
⟨Stmt⟩ → if ( ⟨Expr⟩ ) ⟨Stmt⟩
⟨StmtList⟩ → ⟨Stmt⟩
⟨StmtList⟩ → ⟨StmtList⟩ ⟨Stmt⟩
⟨Expr⟩ → ⟨Id⟩
⟨Expr⟩ → ⟨Num⟩
⟨Expr⟩ → ⟨Expr⟩ ⟨Optr⟩ ⟨Expr⟩
⟨Id⟩ → x
⟨Id⟩ → y
⟨Num⟩ → 0
⟨Num⟩ → 1
⟨Num⟩ → 9
⟨Optr⟩ → >
⟨Optr⟩ → +



# Recognizing a context-free grammar in Chomsky normal form

$$
\begin{aligned}
\text{S} &\longrightarrow \text{NP VP} \\
\text{VP} &\longrightarrow \text{VP PP} \\
\text{VP} &\longrightarrow \text{V NP} \\
\text{VP} &\longrightarrow \text{eats} \\
\text{PP} &\longrightarrow \text{P NP} \\
\text{NP} &\longrightarrow \text{Det N} \\
\text{NP} &\longrightarrow \text{she} \\
\text{V} &\longrightarrow \text{eats} \\
\text{P} &\longrightarrow \text{with} \\
\text{N} &\longrightarrow \text{fish} \\
\text{N} &\longrightarrow \text{fork} \\
\text{Det} &\longrightarrow \text{a}
\end{aligned}
$$




## Complexity of CYK algorithm

$O(n^3|G|)$ for string of length $n$ and grammar of size $|G|$

# String applications

| | String Hashing | Suffix Array | Suffix Tree | Aho-Corasick |
|---|---|---|---|---|
| Search for duplicate strings in array of strings | X | X | X | |
| **Number of distinct substrings of given string** | **$O(n^2)$** | **O(n log n)** | **O(n)** | |
| Finding smallest cyclic shift | X | X | X | |
| Finding substring in a string | X | X | X | X |
| *Comparing same-length substrings a < b* | X | X | X | |
| Longest common prefix of substrings | X | X | X | |
| Total length of all different substrings | | X | X | |
| Lexicographically k-th substring | X | X | X | |
| Shortest non-appearing string | | X | X | |
| Longest common substring of multiple strings | X | X | X | |
| Find strings S in a text | | | | X |
| Finding lexicographically smallest string that doesn't contain S | | | | X |
| Finding lexicographically smallest string that contains k strings from S (or all) | | | | X |