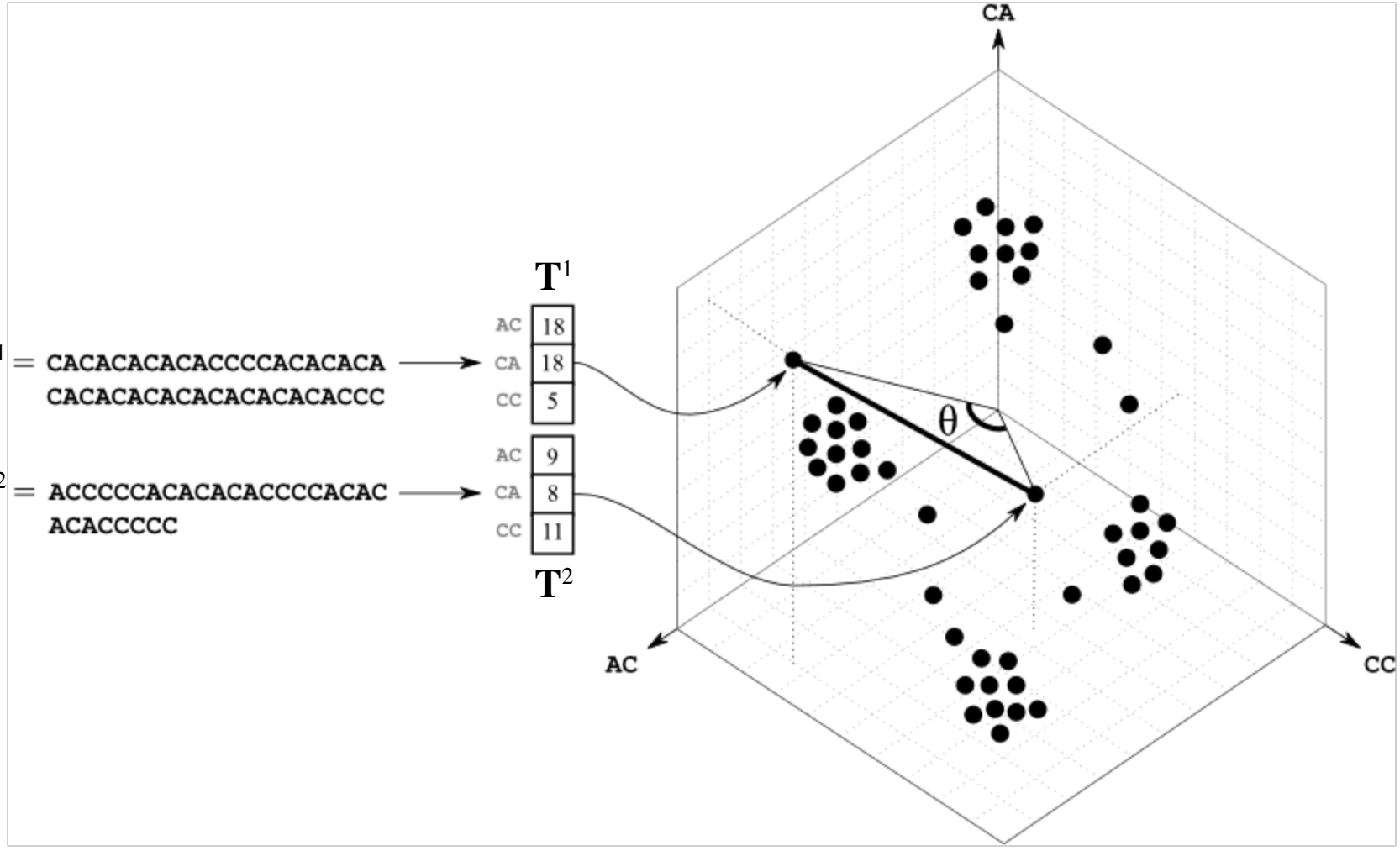
A framework for space-efficient string kernels

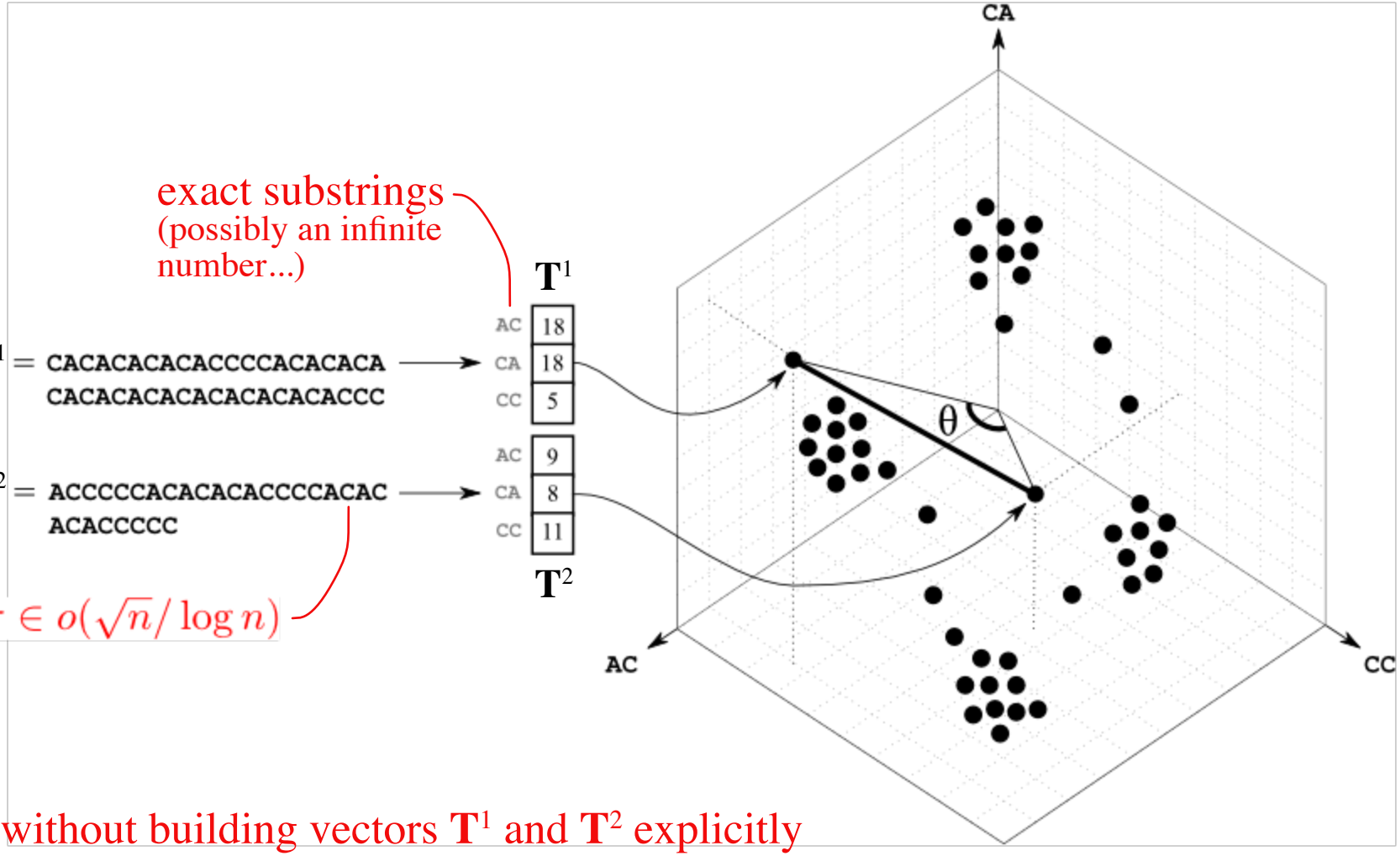
Djamal Belazzougui[1], **Fabio Cunial[2]**

(1) Department of Computer Science, University of Helsinki, Finland.
(2) Max Planck Institute for Molecular Cell Biology and Genetics, Dresden, Germany.

Substring kernels

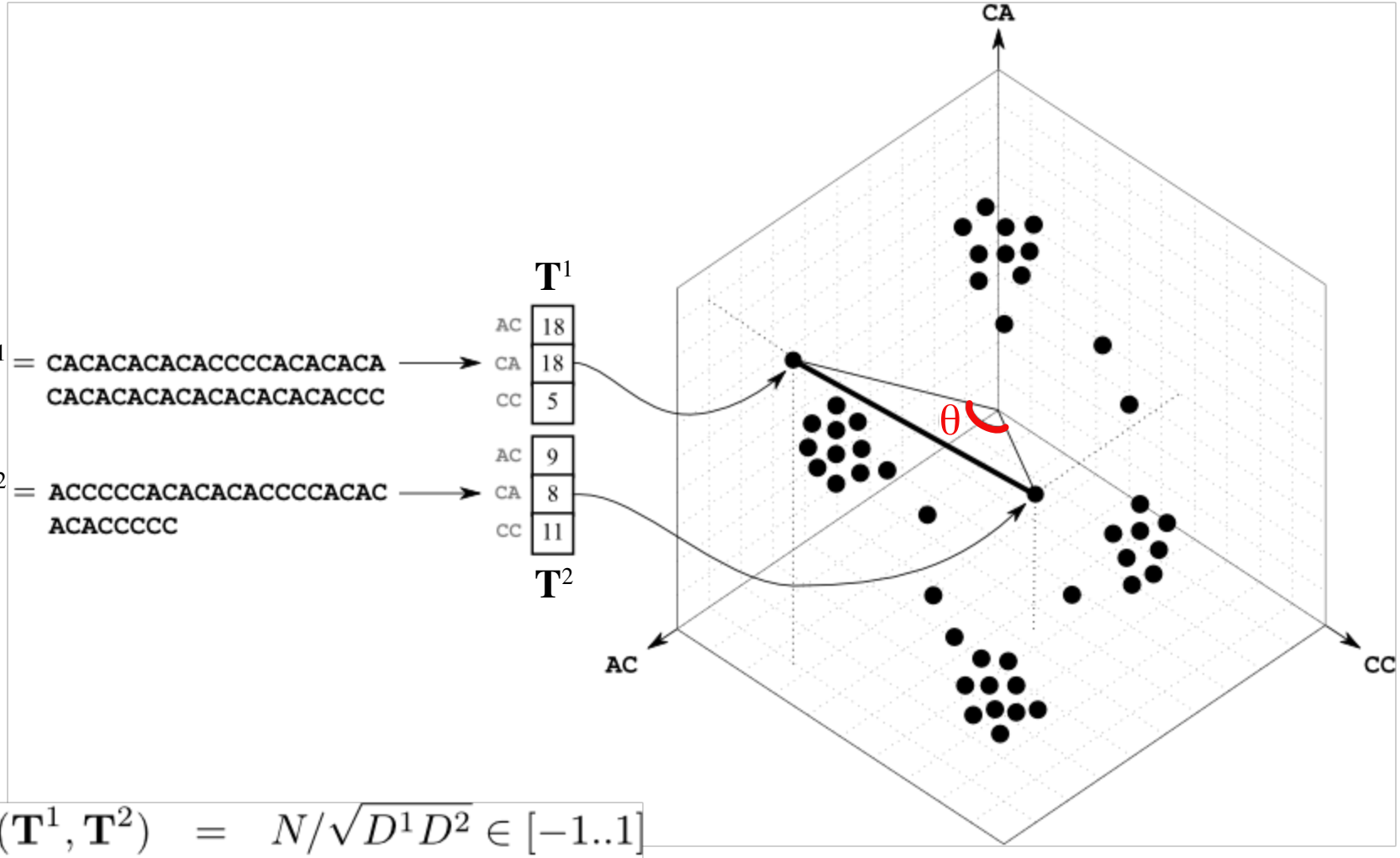


Substring kernels



...without building vectors $\mathbf{T}^1$ and $\mathbf{T}^2$ explicitly

Substring kernels



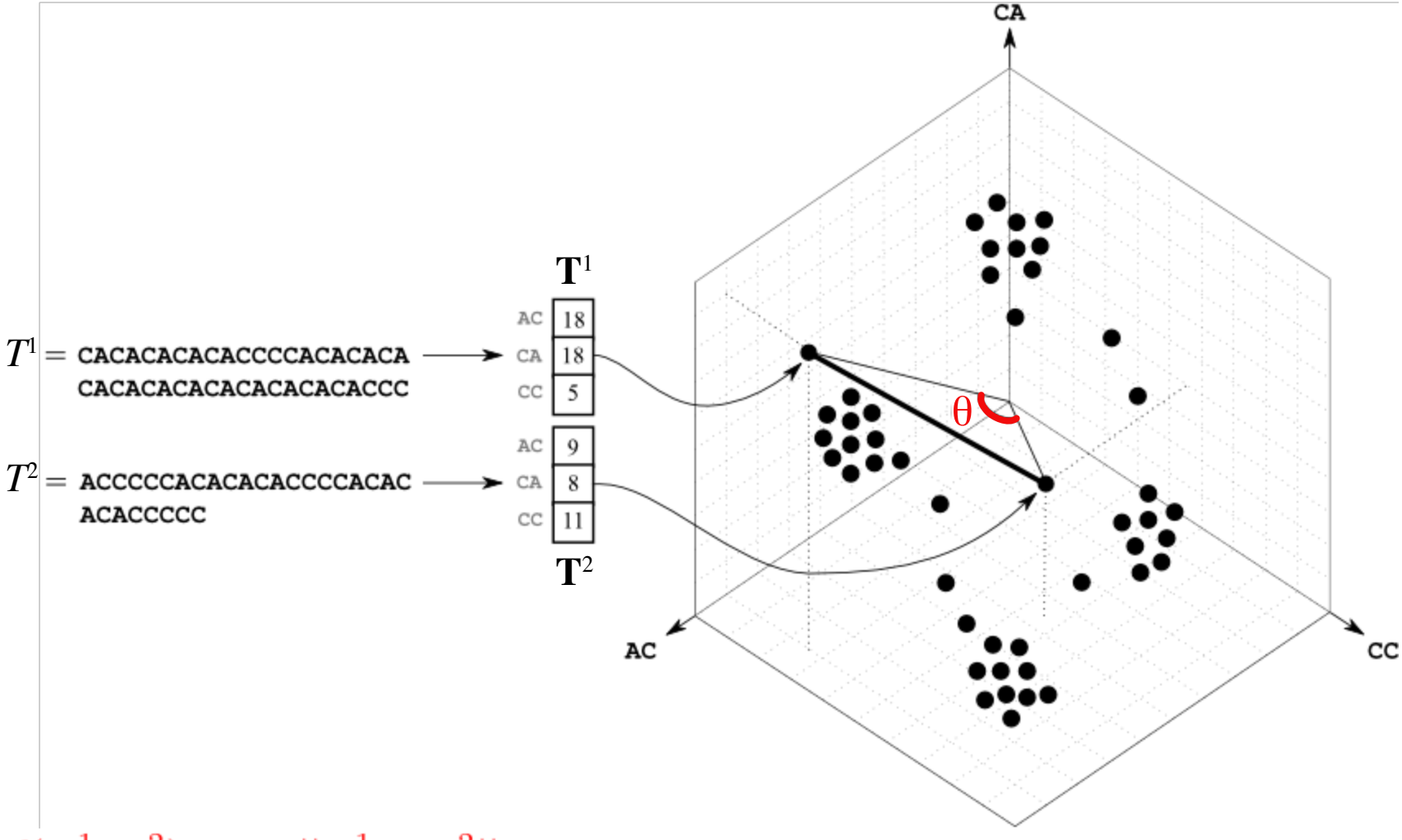
$$\kappa(\mathbf{T}^1, \mathbf{T}^2) = N/\sqrt{D^1 D^2} \in [-1..1]$$

$$N = \sum_W \mathbf{T}^1[W]\mathbf{T}^2[W]$$

$$D^i = \sum_W \mathbf{T}^i[W]^2$$

$$d(\mathbf{T}^1, \mathbf{T}^2) = (1 - \kappa(\mathbf{T}^1, \mathbf{T}^2))/2 \in [0..1]$$

Substring kernels



$$d(\mathbf{T}^1, \mathbf{T}^2) = ||\mathbf{T}^1 - \mathbf{T}^2||$$ e.g. *p*-norm, infinity norm

$$J(\mathbf{T}^1, \mathbf{T}^2) = \frac{||\mathbf{T}^1 \wedge \mathbf{T}^2||}{||\mathbf{T}^1 \vee \mathbf{T}^2||} = \frac{||\mathbf{T}^1 \wedge \mathbf{T}^2||}{(||\mathbf{T}^1|| + ||\mathbf{T}^2|| - ||\mathbf{T}^1 \wedge \mathbf{T}^2||)}$$

In this work...

***k*-mers**	**Substrings**
k-mer complexity	Substring complexity
k-mer kernel	Substring kernel
k-mer profiling	Length-weighted substring kernel
k-mer kernel profile	Character-weighted substring kernel
k-th order empirical entropy	Minimal absent words complexity
KL divergence profile	Minimal absent words distance
	Substring kernel with Markovian corrections
	KL divergence kernel

In this work...

***k*-mers**	**Substrings**
k-mer complexity	Substring complexity
k-mer kernel	Substring kernel
k-mer profiling	Length-weighted substring kernel
k-mer kernel profile	Character-weighted substring kernel
k-th order empirical entropy	Minimal absent words complexity
KL divergence profile	Minimal absent words distance
	Substring kernel with Markovian corrections
	KL divergence kernel

...in small space

Example applications

Alignment-free, whole-genome phylogeny reconstruction.

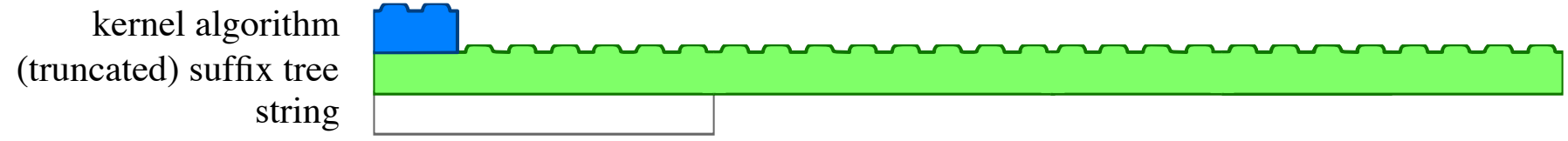
...for multiple values of *k* at once!

Calibrating the value of *k* for alignment-free distances, from a genome.

Calibrating the value of *k* for de Bruijn graphs, from a set of reads.

$O(n)$ time

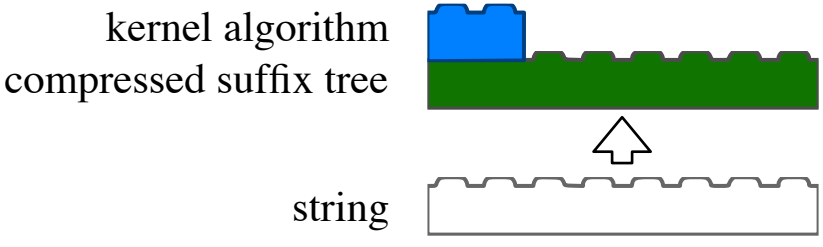
bits



$O(n \log n)$

$n \log \sigma$

$O(n \log^{\varepsilon} n)$ time [1]



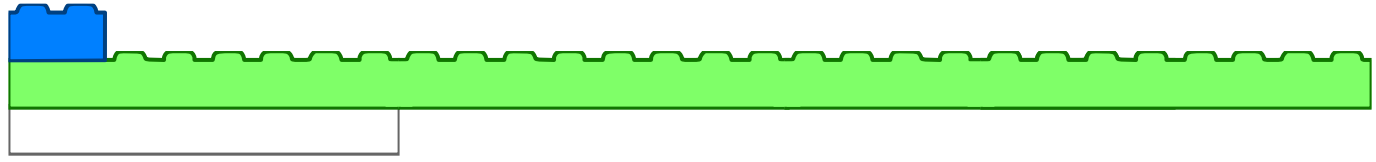
$O(n \log \sigma)$

[1] Simon Gog. *Compressed suffix trees: Design, construction, and applications*. PhD thesis, University of Ulm, Germany, 2011.

$O(n)$ time

bits

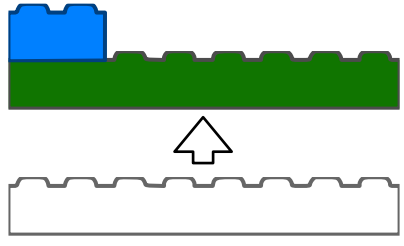
kernel algorithm
(truncated) suffix tree
string



$O(n \log n)$
$n \log \sigma$

$O(n \log^{\varepsilon} n)$ time [1]

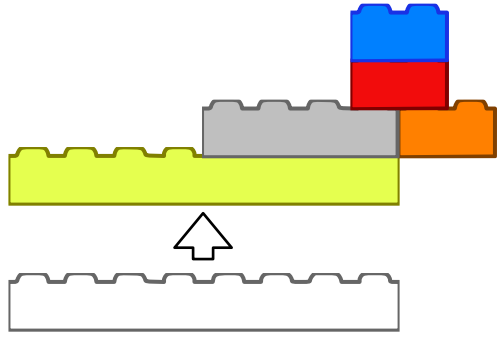
kernel algorithm
compressed suffix tree
string



$O(n \log \sigma)$

$O(n)$ time

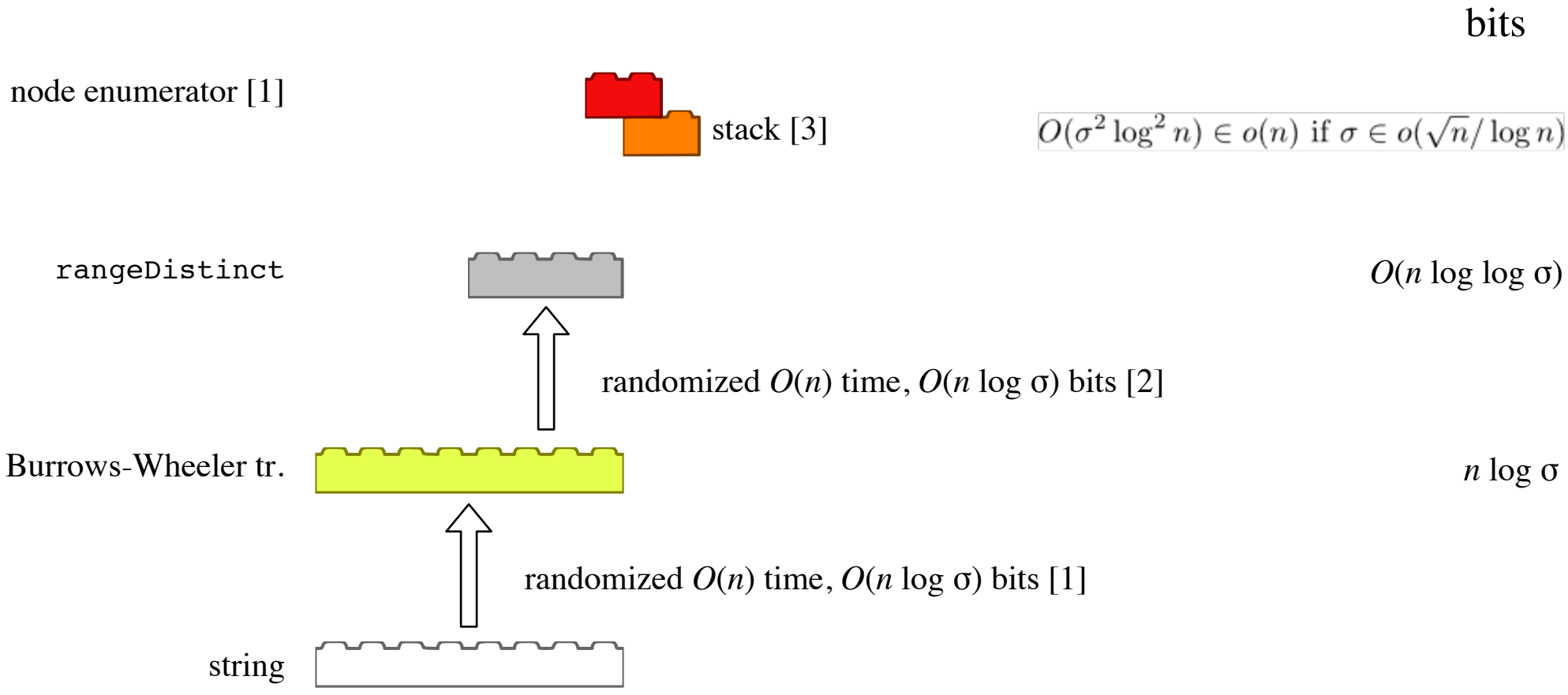
kernel algorithm
node enumerator
`rangeDistinct`
Burrows-Wheeler tr.
stack
string



$O(n \log \log \sigma)$, $O(\sigma^2 \log^2 n)$
$n \log \sigma$

[1] Simon Gog. *Compressed suffix trees: Design, construction, and applications*. PhD thesis, University of Ulm, Germany, 2011.

Randomized $O(n)$ time from the string



[1] D. Belazzougui. Linear time construction of compressed text indices in compact space. STOC 2014.
[2] D. Belazzougui, G. Navarro, D. Valenzuela. Improved compressed indexes for full-text document retrieval. JDA, 18:3–13, January 2013.
[3] D. Belazzougui, F. Cunial, J. Kärkkäinen, V. Mäkinen. Versatile succinct representations of the bidirectional Burrows-Wheeler transform. ESA 2013.

rangeDistinct queries

$$\texttt{rangeDistinct}(i,j) = \{(c, \texttt{rank}(c, p_c), \texttt{rank}(c, q_c)) : c \in \Sigma_{i,j}\}$$

unordered

$O(\texttt{out})$ time, σ log n bits of working space using [1]

let's say $O(\texttt{out}\ \boldsymbol{d})$ time in general

$p_{\texttt{G}}$

$$\begin{aligned} \texttt{range}(\texttt{T}) &= [12..26] \\ \Sigma_{12,26} &= \{\texttt{C}, \texttt{G}, \texttt{T}, \texttt{A}, \$\} \end{aligned}$$

$$\texttt{rangeDistinct}(12, 26) = \{(\texttt{G}, 2, 4), \ldots\}$$

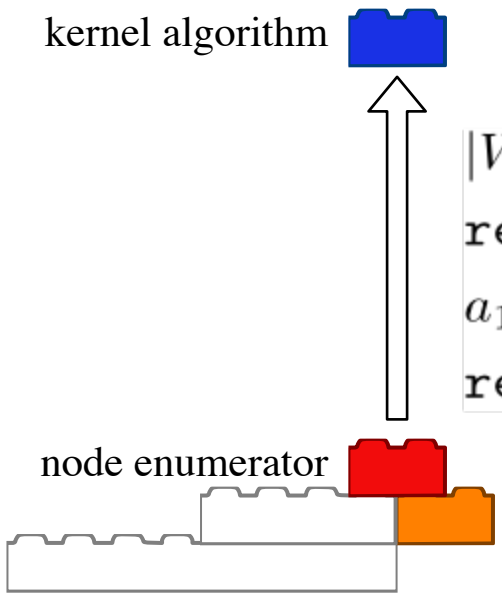
$q_{\texttt{G}}$

BWT_T

T	$
C	ATGTGTATTTGCT$
T	ATTTGCT$
T	CATGTGTATTTGCT$
G	CT$
T	CTGTTCATGTGTATTTGCT$
T	CTTTCTGTTCATGTGTATTTGCT$
T	GCT$
T	GTATTTGCT$
T	GTGTATTTGCT$
T	GTTCATGTGTATTTGCT$
C	T$
G	TATTTGCT$
T	TCATGTGTATTTGCT$
T	TCTGTTCATGTGTATTTGCT$
T	TCTTTCTGTTCATGTGTATTTGCT$
T	TGCT$
G	TGTATTTGCT$
A	TGTGTATTTGCT$
C	TGTTCATGTGTATTTGCT$
G	TTCATGTGTATTTGCT$
T	TTCTGTTCATGTGTATTTGCT$
$	TTCTTTCTGTTCATGTGTATTTGCT$
T	TTGCT$
C	TTTCTGTTCATGTGTATTTGCT$
A	TTTGCT$

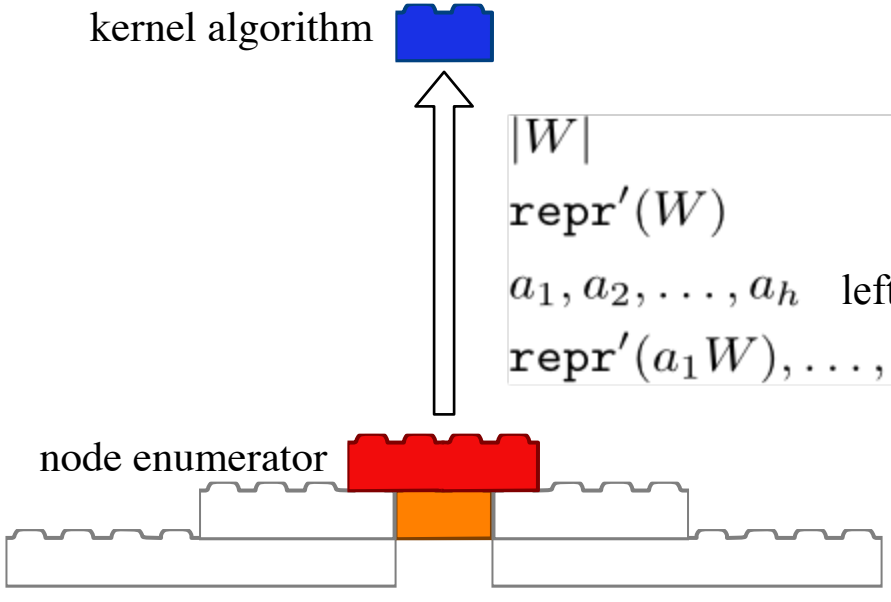
[1] D. Belazzougui, G. Navarro, D. Valenzuela. Improved compressed indexes for full-text document retrieval. JDA, 18:3–13, January 2013.

Enumerating nodes in no order



$|W|$
$\texttt{repr}(W)$
$a_1, a_2, \ldots, a_h$ left-extensions, **unordered**
$\texttt{repr}(a_1 W), \ldots, \texttt{repr}(a_h W)$

all right-maximal W of T
no order
$O(nd)$ time
$o(n)$ bits of working space



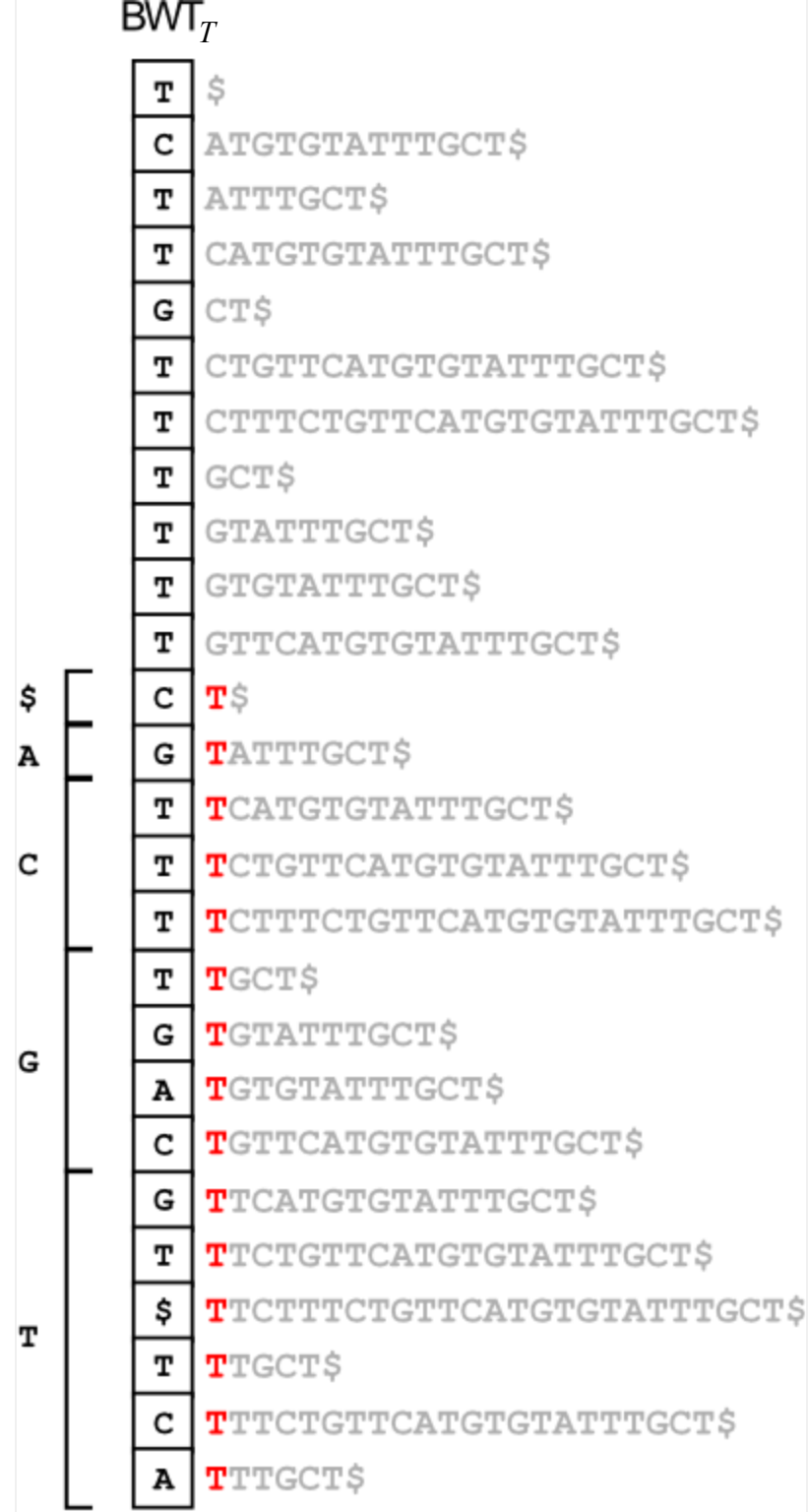
$|W|$
$\texttt{repr}'(W)$
$a_1, a_2, \ldots, a_h$ left-extensions, **unordered**
$\texttt{repr}'(a_1 W), \ldots, \texttt{repr}'(a_h W)$

all right-maximal W of $T^1 T^2$
no order
$O((n_1+n_2)d)$ time
$o(n_1+n_2)$ bits of working space

[1] D. Belazzougui. Linear time construction of compressed text indices in compact space. STOC 2014.
[2] D. Belazzougui, F. Cunial, J. Kärkkäinen, V. Mäkinen. Versatile succinct representations of the bidirectional Burrows-Wheeler transform. ESA 2013.

$$\texttt{repr}(\texttt{T}) = \big([\$, \texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}], [12, 13, 14, 17, 21, 26]\big)$$

ordered

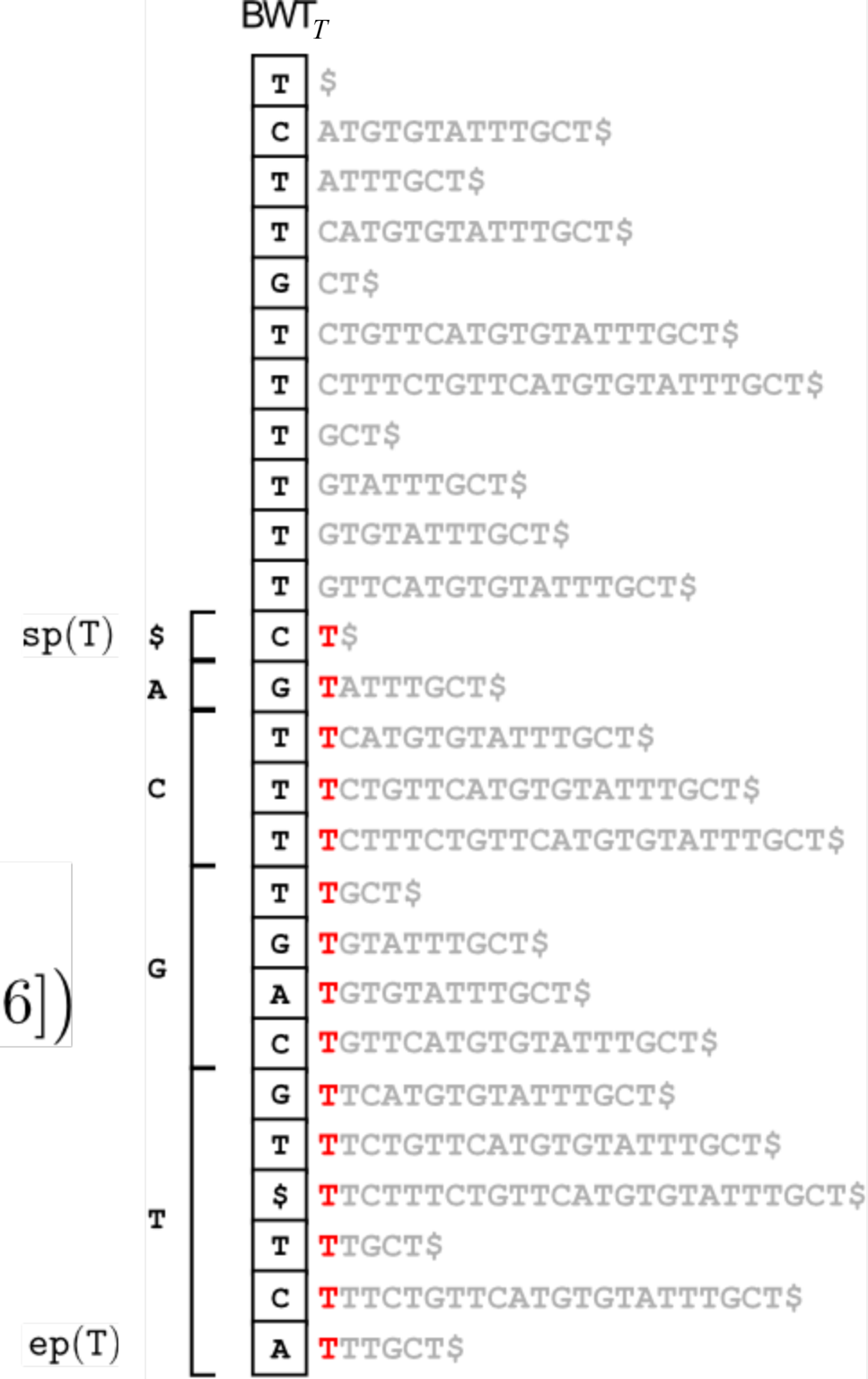


W is a maximal repeat of T iff:

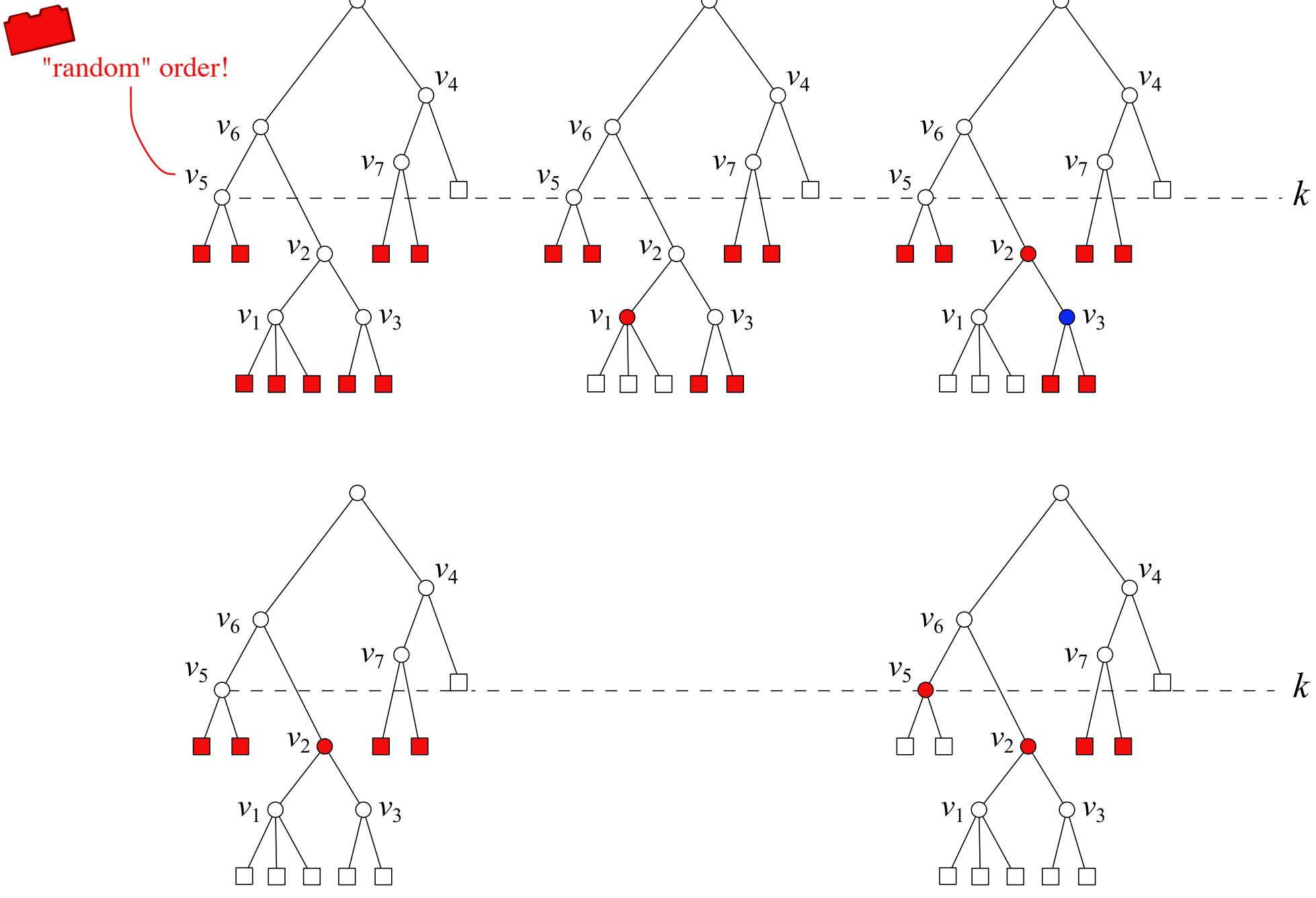
$$|\texttt{rangeDistinct}(\texttt{sp}(W), \texttt{ep}(W))| > 1$$

$$\texttt{repr}(\texttt{T}) = ([\$, \texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}], [12, 13, 14, 17, 21, 26])$$

ordered



k-mer complexity



k-mer kernel

$$\begin{aligned}\kappa(\mathbf{T}^1, \mathbf{T}^2) &= N/\sqrt{D^1 D^2} \in [-1..1] \\ N &= \sum_W \mathbf{T}^1[W]\mathbf{T}^2[W] \\ D^i &= \sum_W \mathbf{T}^i[W]^2\end{aligned}$$

from `repr`

$$\text{if } |\ell(u)| \geq k:$$

$$\begin{aligned}N &\leftarrow N + f_1(\ell(u)) \cdot f_2(\ell(u)) - \sum_v f_1(\ell(v)) \cdot f_2(\ell(v)) \\ D^i &\leftarrow D^i + f_i(\ell(u))^2 - \sum_v f_i(\ell(v))^2\end{aligned}$$

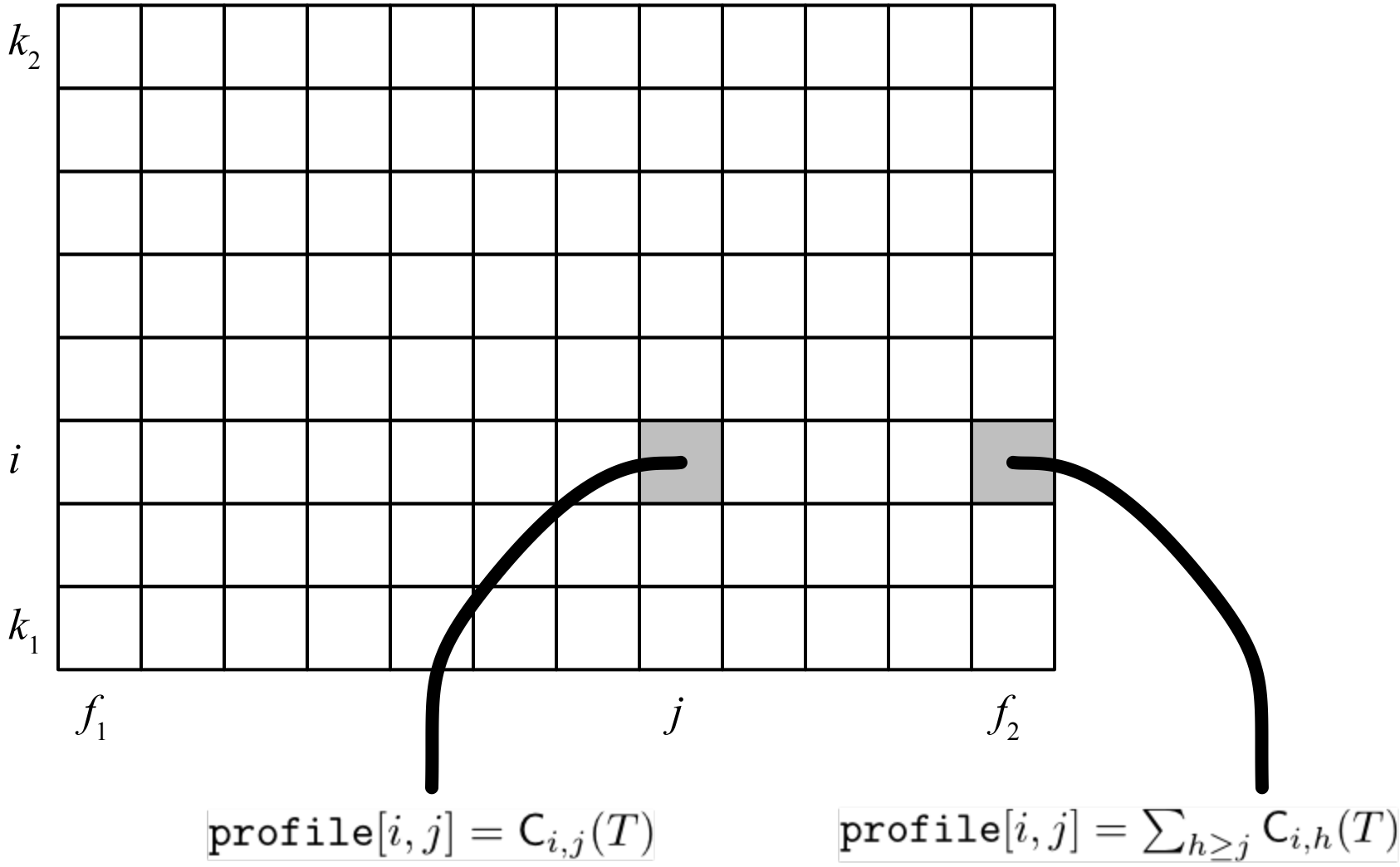
children of u, from `repr`

k-mer kernel

$$\begin{aligned}\kappa(\mathbf{T}^1, \mathbf{T}^2) &= N/\sqrt{D^1 D^2} \in [-1..1] \\ N &= \sum_W \mathbf{T}^1[W]\mathbf{T}^2[W] \\ D^i &= \sum_W \mathbf{T}^i[W]^2\end{aligned}$$

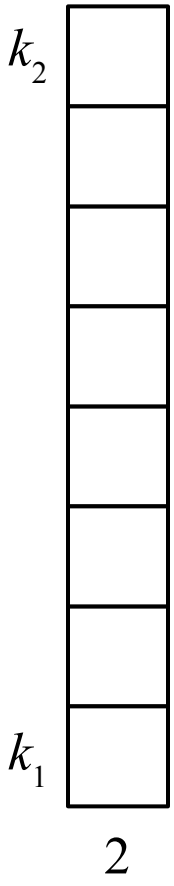
"telescoping"

k-mer profile



Rayan Chikhi and Paul Medvedev. Informed and automated *k*-mer size selection for genome assembly. Bioinformatics, 30(1):31–37, 2014.

Example application

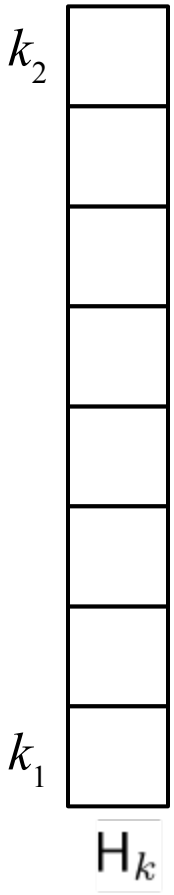


Find the value of $k \in [k_1..k_2]$ that maximizes $\mathsf{C}_{k,2}$

Smallest k to be used in k-mer kernels [1]

[1] G.E. Sims, S.-R. Jun, G.A. Wu, S.-H. Kim. Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. PNAS 106(8):2677–2682, 2009.

Similar problem: entropy



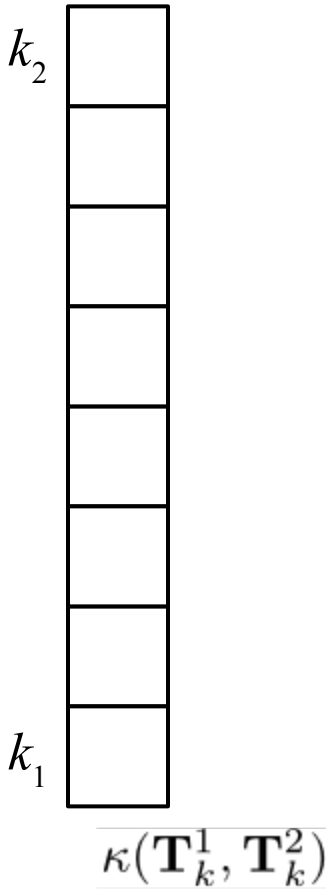
Compute the k-th order empirical entropy *for all* k:

$$\mathsf{H}_k(T) = \frac{1}{|T|} \cdot \sum_{W \in [1..\sigma]^k} \sum_{a \in \Sigma^r(W)} f_T(Wa) \cdot \log\left(\frac{f_T(W)}{f_T(Wa)}\right)$$

... in a single pass over BWT_T

Simon Gog. Compressed suffix trees: Design, construction, and applications. PhD thesis, University of Ulm, Germany, 2011.

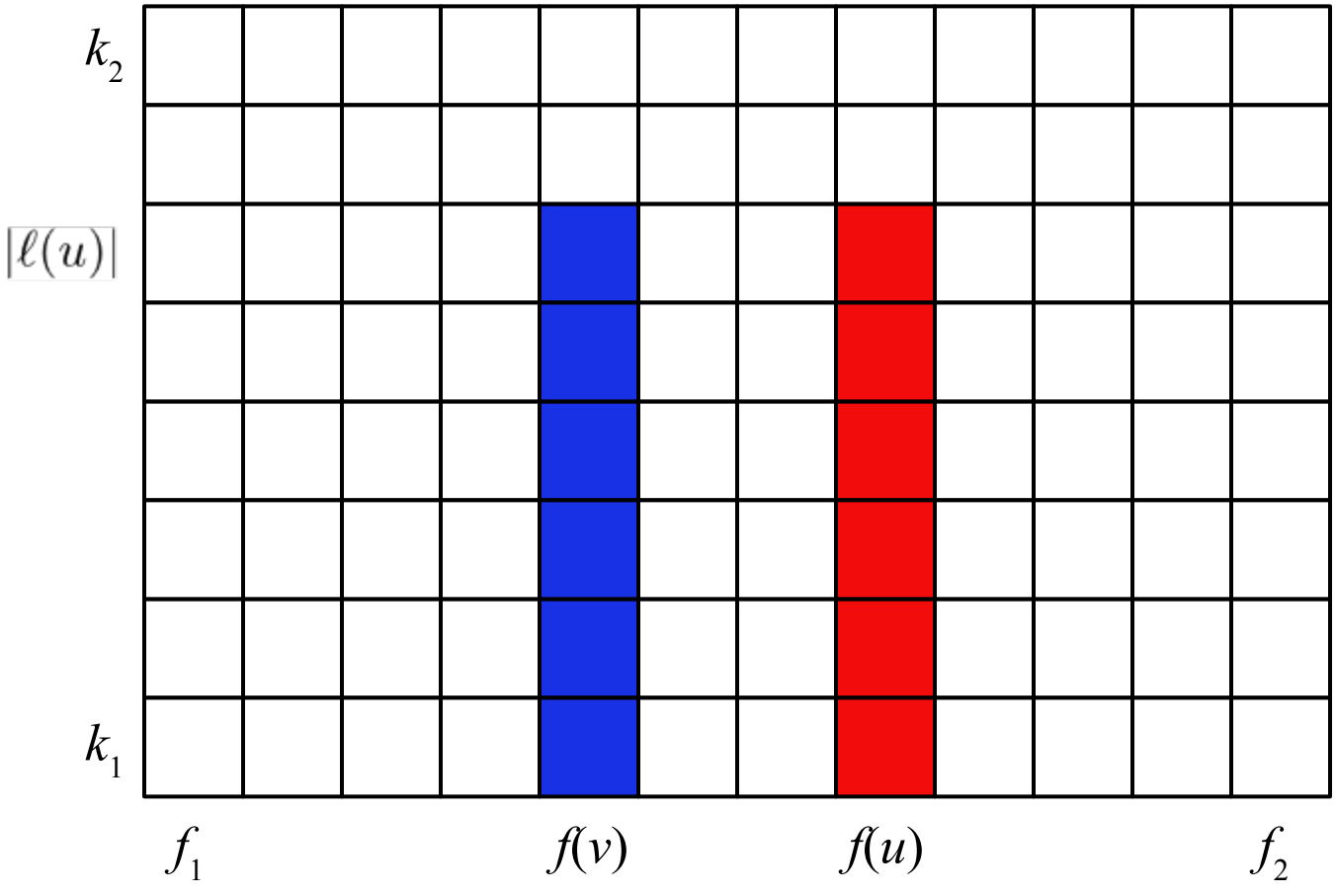
Similar problem: multiple kernels

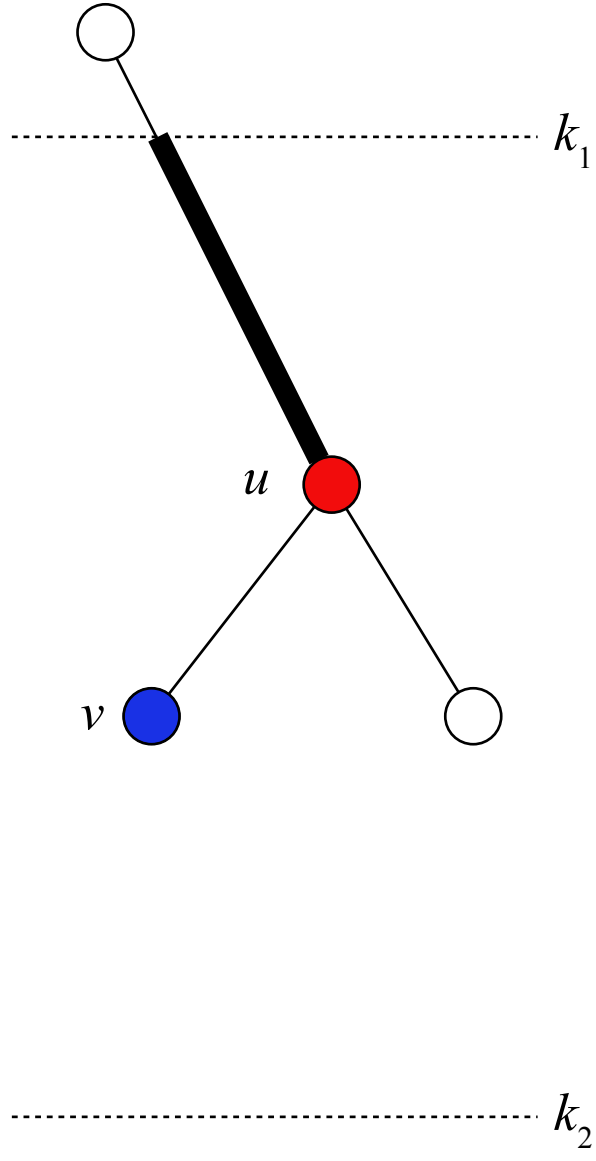


Compute $\kappa(\mathbf{T}_k^1, \mathbf{T}_k^2)$ for all $k \in [k_1..k_2]$

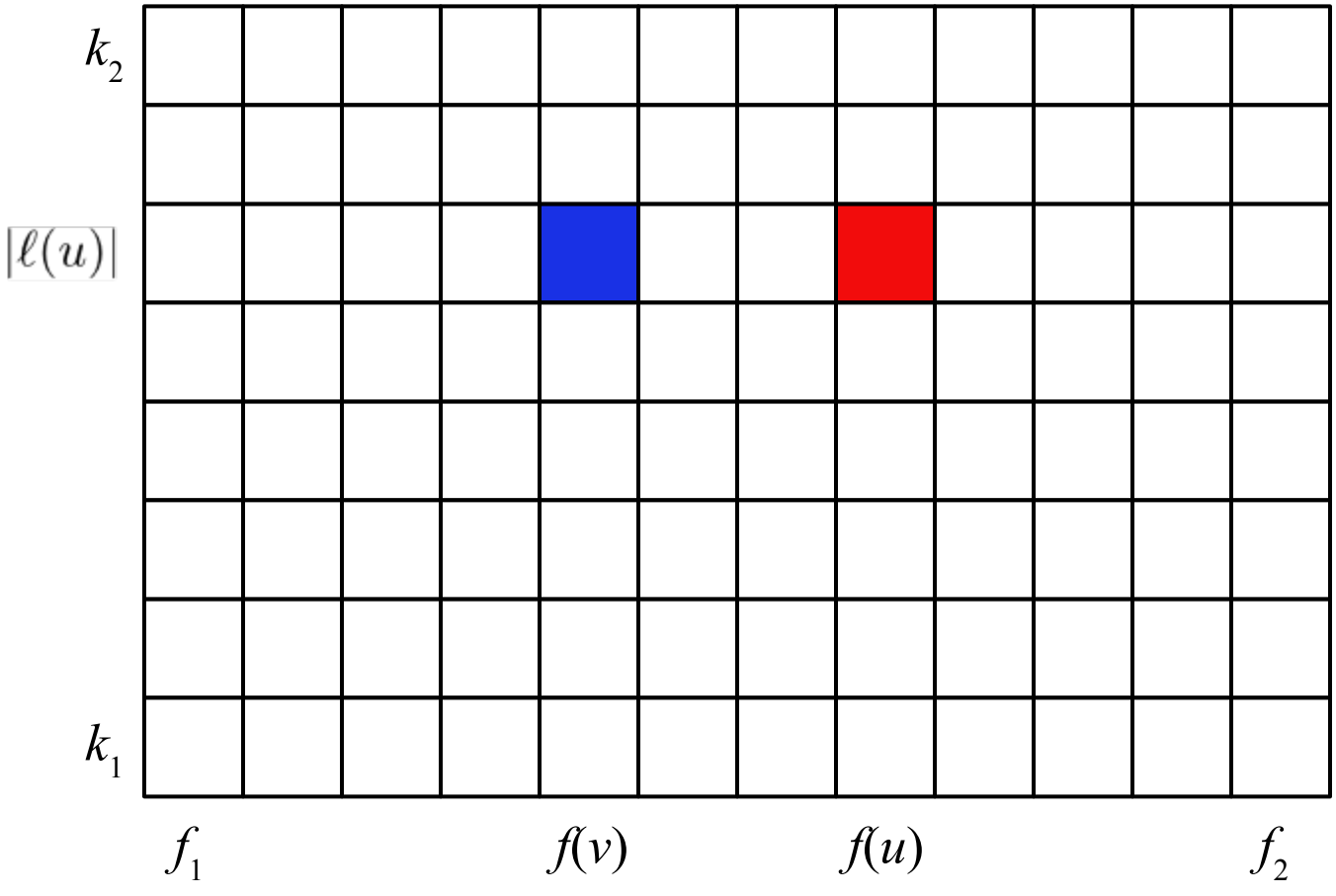
... in a single pass over BWT_T

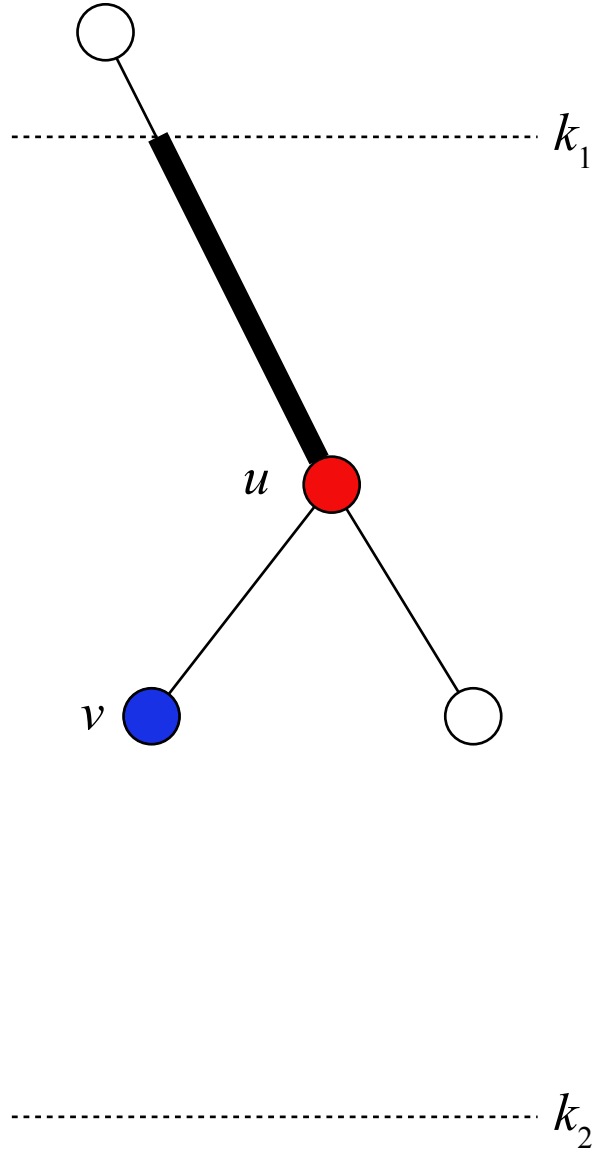
k-mer profile



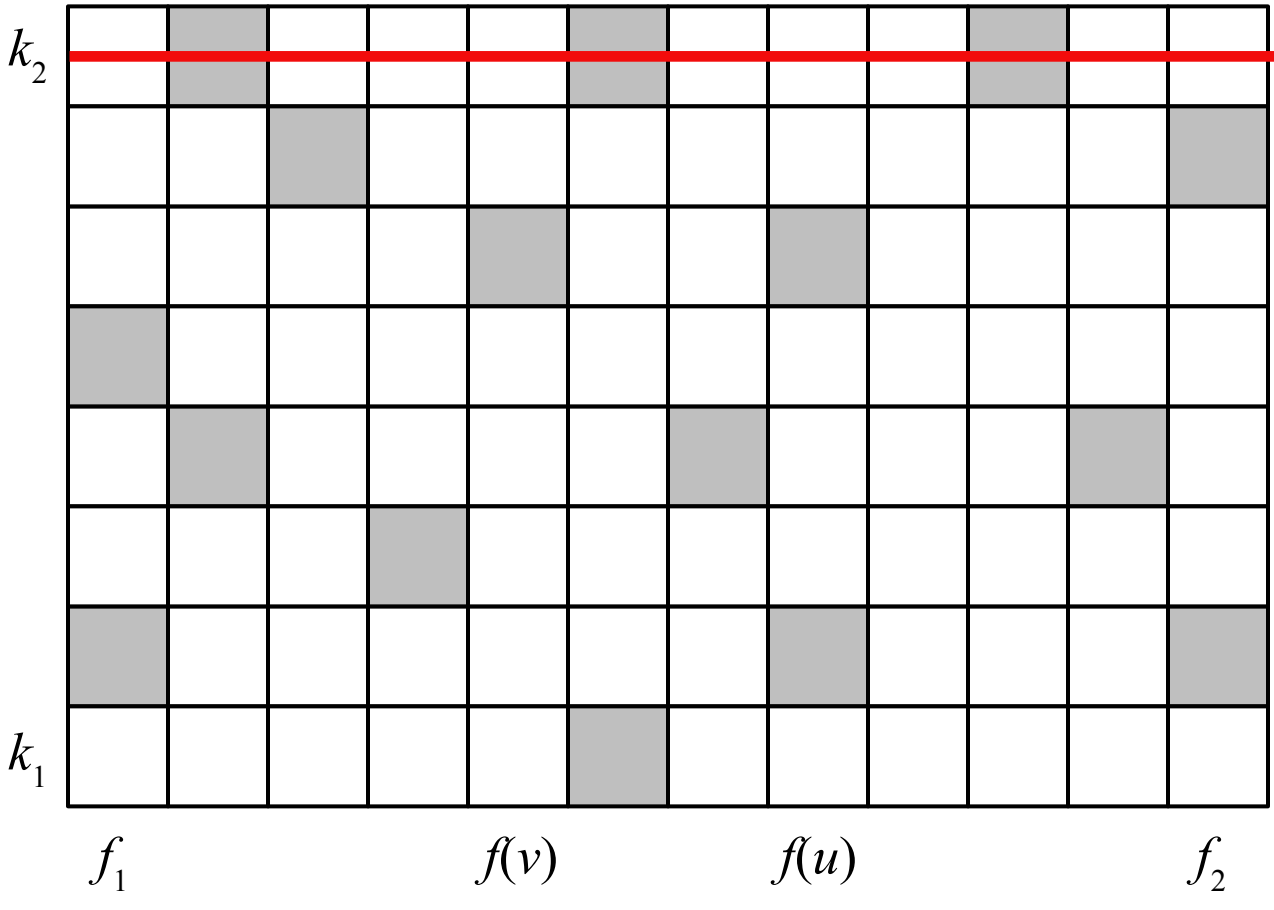


k-mer profile





k-mer profile



$$\texttt{profile}[i,j] = \texttt{profile}[i,j] + \texttt{profile}[i+1,j]$$

$O(nd + (k_2 - k_1)(f_2 - f_1))$ time, $o(n)$ bits of space

In this work...

***k*-mers**	**Substrings**
k-mer complexity	Substring complexity
k-mer kernel	Substring kernel
k-mer profiling	Length-weighted substring kernel
k-mer kernel profile	Character-weighted substring kernel
k-th order empirical entropy	Minimal absent words complexity
KL divergence profile	Minimal absent words distance
	Substring kernel with Markovian corrections
	KL divergence kernel

All-substrings kernel

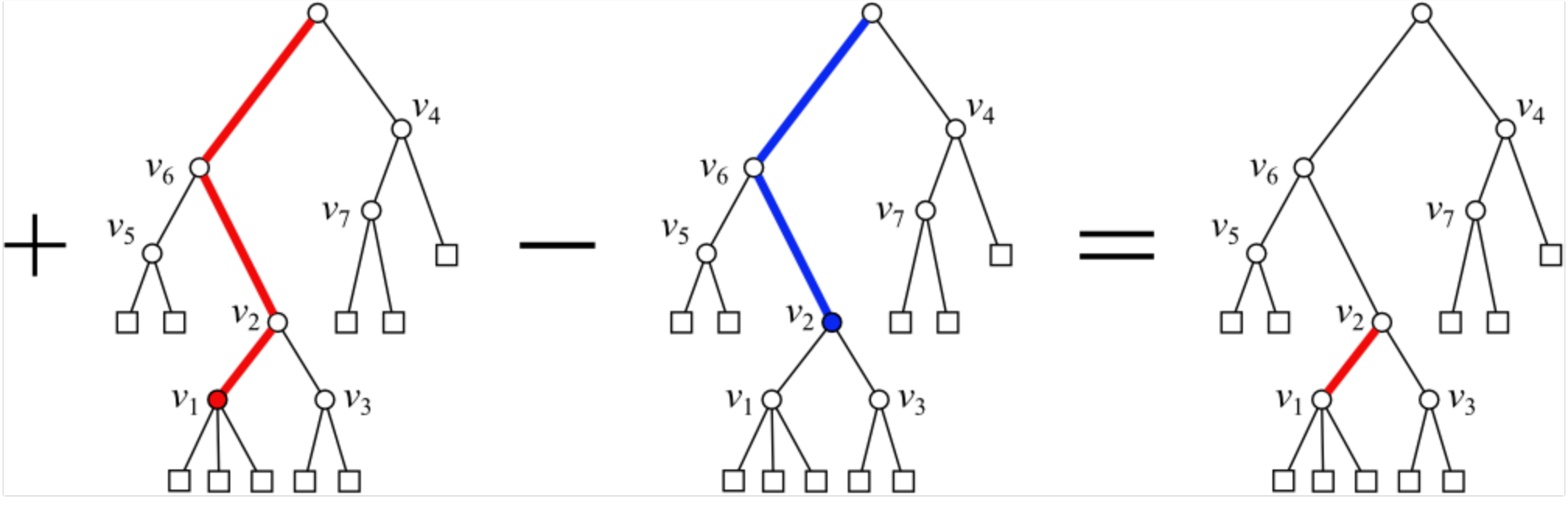
$$\kappa(\mathbf{T}^1_\infty, \mathbf{T}^2_\infty) \quad = \quad N/\sqrt{D^1 D^2}$$

from enumerator

$$N \quad \leftarrow \quad N + |\ell(u)| \cdot \left(f_1(\ell(u)) f_2(\ell(u)) - \sum_v f_1(\ell(v)) f_2(\ell(v)) \right)$$

$$D^i \quad \leftarrow \quad D^i + |\ell(u)| \cdot \left(f_i(\ell(u))^2 - \sum_v f_i(\ell(v))^2 \right)$$

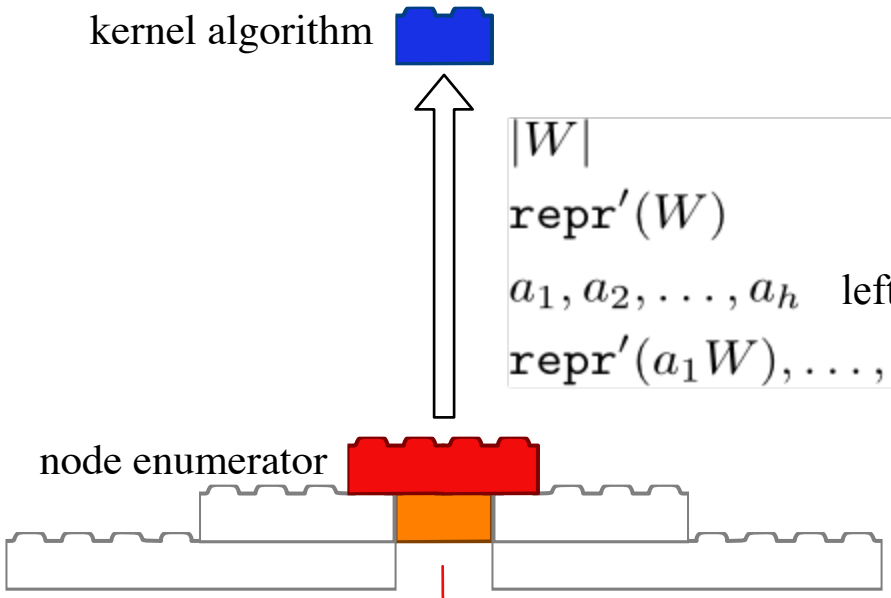
children of u, from `repr`



Using the stack

any function of string length

$$\mathbf{T}^i_{\infty,g}[W] = g(|W|) \cdot \mathbf{T}^i_{\infty}[W]$$



$|W|$

$\texttt{repr}'(W)$

$a_1, a_2, \ldots, a_h$ left-extensions, unordered

$\texttt{repr}'(a_1 W), \ldots, \texttt{repr}'(a_h W)$

push on the stack:

$$\texttt{prefixSum}(aW) = \sum_{i=1}^{|W|+1} g(i)^2 = \texttt{prefixSum}(W) + g(|W|+1)^2$$

and use it in place of $|aW|$

A.J. Smola, S.v.n. Vishwanathan. Fast kernels for string and tree matching. Advances in Neural Information Processing Systems 15, pages 585–592. 2003.

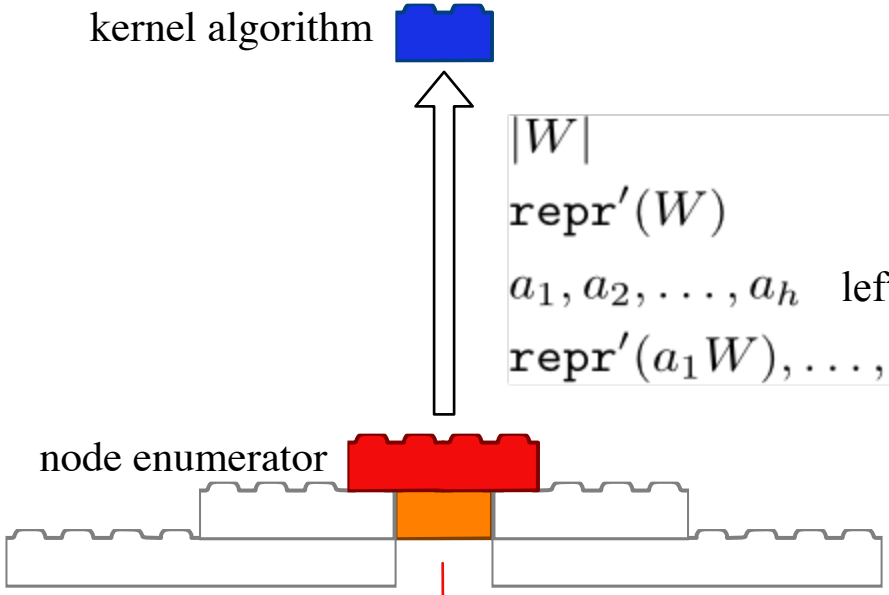
Using the stack

$$\mathbf{T}^i_{\infty,q}[W] = q(W) \cdot \mathbf{T}^i_{\infty}[W]$$

$$q(W) = \prod_{i=1}^{|W|} q(W[i])$$

any probability function on characters

kernel algorithm



$|W|$

$\texttt{repr}'(W)$

$a_1, a_2, \ldots, a_h$ left-extensions, unordered

$\texttt{repr}'(a_1 W), \ldots, \texttt{repr}'(a_h W)$

node enumerator

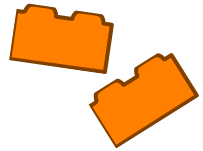
push on the stack:

$\texttt{prefixSum}(V) = \sum_{i=1}^{|V|} \prod_{j=1}^{i} q(V[j])^2 = q(a)^2 \cdot (1 + \texttt{prefixSum}(W))$

and use it in place of $|aW|$

A.J. Smola, S.v.n. Vishwanathan. Fast kernels for string and tree matching. Advances in Neural Information Processing Systems 15, pages 585–592. 2003.

Using two stacks



$$\mathbf{T}^{i}{}_{k,q}[W] = q(W) \cdot \mathbf{T}^{i}{}_{k}[W]$$

$$q(W) = \prod_{i=1}^{|W|} q(W[i])$$

just k-mers

push on another stack the characters that form W

$\lambda \log \sigma$ additional bits

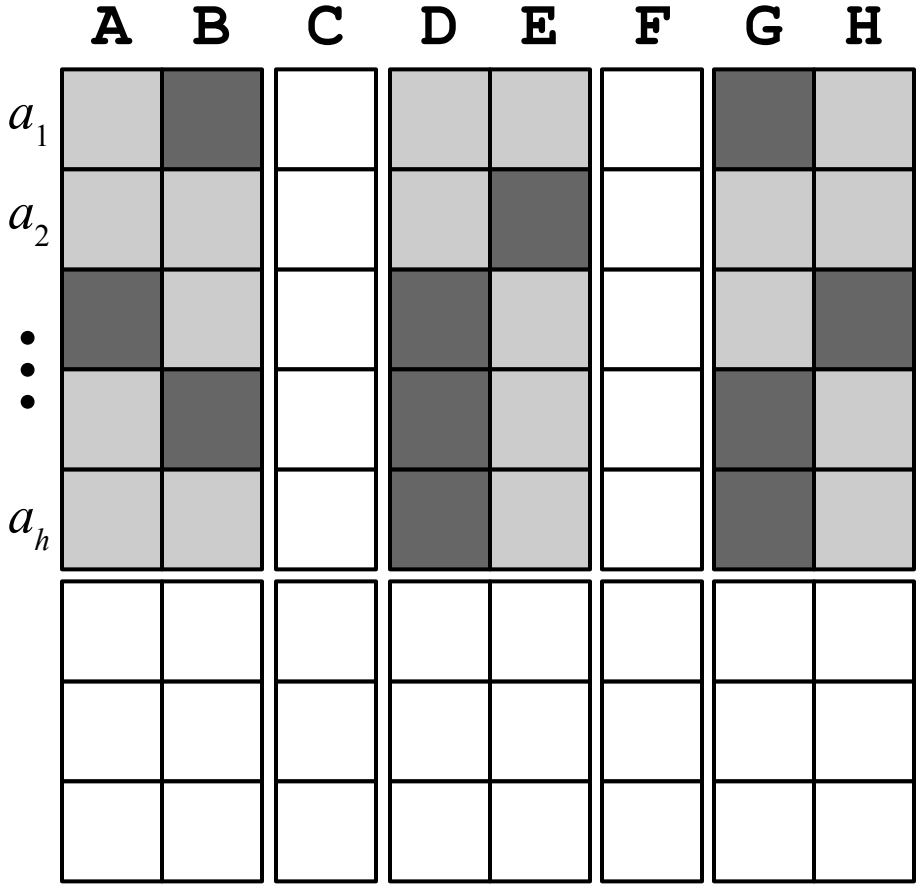
G. Reinert, D. Chew, F. Sun, M.S. Waterman. Alignment-free sequence comparison (I): statistics and power. Journal of Computational Biology, 2009.

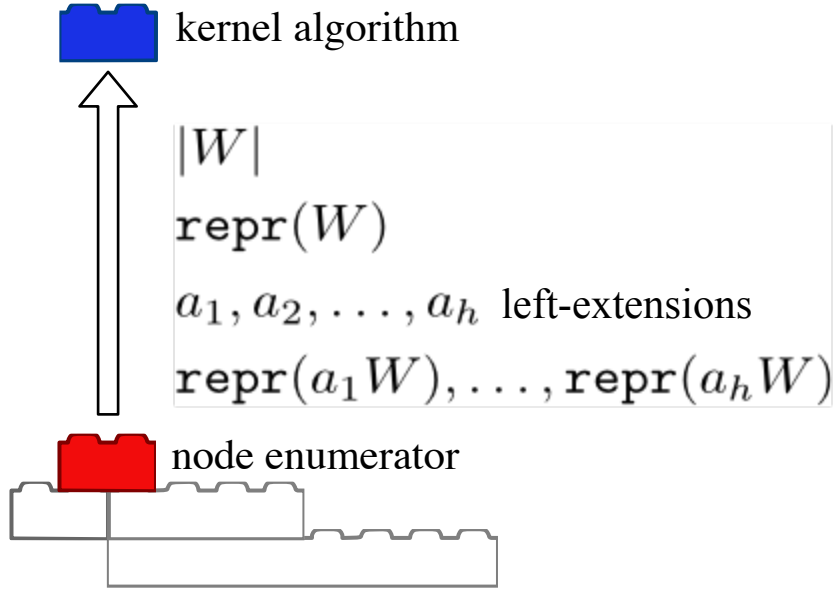
Minimal absent words complexity/Jaccard

Does not occur in T, but every substring occurs in T.

aWb : W is a maximal repeat of T

$\Theta(\sigma n)$





S. Chairungsee, M. Crochemore. Using minimal absent words to build phylogeny. Theoretical Computer Science, 450:109–116, 2012.

Markovian corrections

$$\mathbb{P}(W) = \frac{\mathbb{P}(W[1..k-1]) \cdot \mathbb{P}(W[2..k])}{\mathbb{P}(W[2..k-1])}$$

if T was generated by a Markov process of order $\leq k$ - 2

$$\tilde{p}_T(W) = \frac{p_T(W[1..k-1]) \cdot p_T(W[2..k])}{p_T(W[2..k-1])}$$

approximation with empirical probabilities

$$z_T(W) = \frac{p_T(W) - \tilde{p}_T(W)}{\tilde{p}_T(W)} \approx \frac{f_T(W) \cdot f_T(W[2..k-1])}{f_T(W[1..k-1]) \cdot f_T(W[2..k])} - 1$$

significance score

J. Qi, B. Wang, B.-L. Hao. Whole proteome prokaryote phylogeny without sequence alignment: a *k*-string composition approach. J. Mol. Evol., 2004.
A. Apostolico, O. Denas. Fast algorithms for computing sequence distances by exhaustive substring composition. Algorithms for Molecular Biology, 2008.

Markovian corrections

$$\mathbb{P}(W) = \frac{\mathbb{P}(W[1..k-1]) \cdot \mathbb{P}(W[2..k])}{\mathbb{P}(W[2..k-1])}$$

if T was generated by a Markov process of order $\leq k$ - 2

$$\tilde{p}_T(W) = \frac{p_T(W[1..k-1]) \cdot p_T(W[2..k])}{p_T(W[2..k-1])}$$

approximation with empirical probabilities

$$z_T(W) = \frac{p_T(W) - \tilde{p}_T(W)}{\tilde{p}_T(W)} \approx \frac{f_T(W) \cdot f_T(W[2..k-1])}{f_T(W[1..k-1]) \cdot f_T(W[2..k])} - 1$$

significance score

from `repr`

J. Qi, B. Wang, B.-L. Hao. Whole proteome prokaryote phylogeny without sequence alignment: a *k*-string composition approach. J. Mol. Evol., 2004.
A. Apostolico, O. Denas. Fast algorithms for computing sequence distances by exhaustive substring composition. Algorithms for Molecular Biology, 2008.

Markovian corrections

$$\mathbb{P}(W) = \frac{\mathbb{P}(W[1..k-1]) \cdot \mathbb{P}(W[2..k])}{\mathbb{P}(W[2..k-1])}$$

if T was generated by a Markov process of order $\leq k$ - 2

$$\tilde{p}_T(W) = \frac{p_T(W[1..k-1]) \cdot p_T(W[2..k])}{p_T(W[2..k-1])}$$

approximation with empirical probabilities

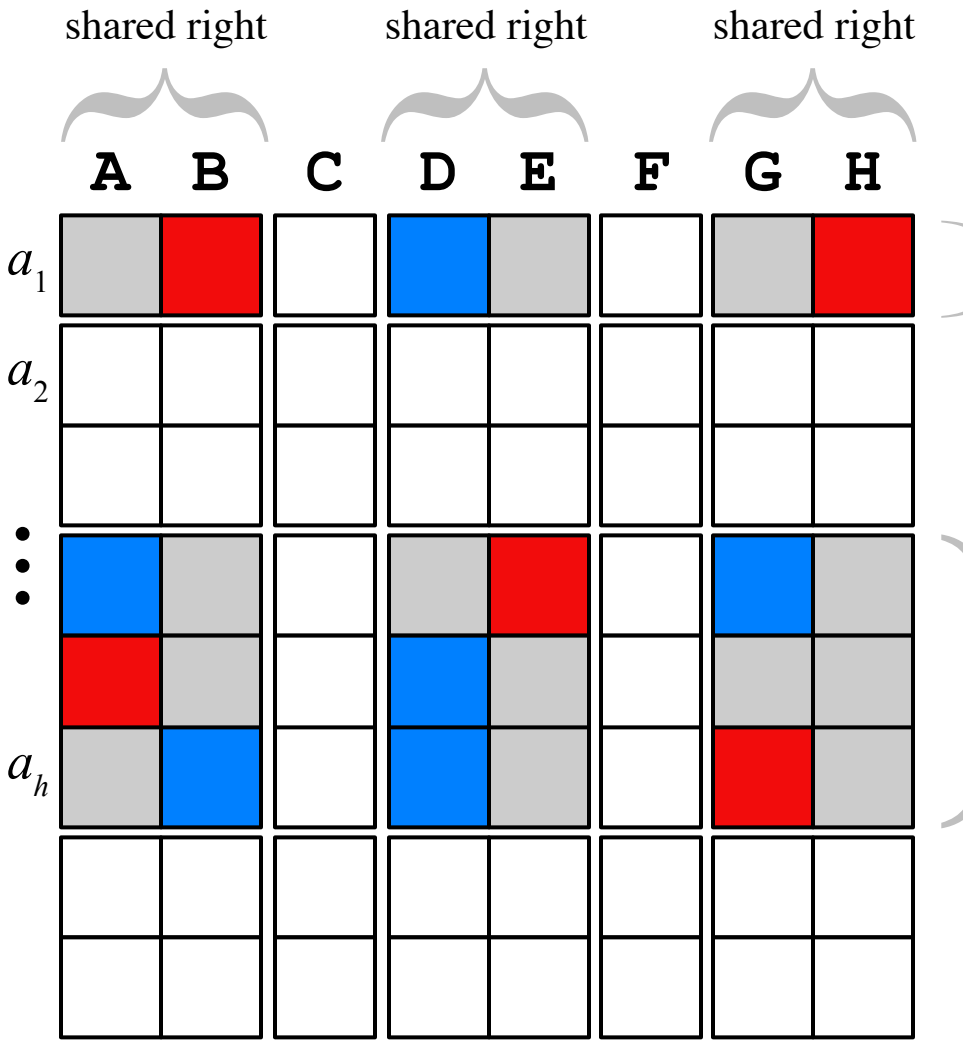
$$z_T(W) = \frac{p_T(W) - \tilde{p}_T(W)}{\tilde{p}_T(W)} \approx \frac{f_T(W) \cdot f_T(W[2..k-1])}{f_T(W[1..k-1]) \cdot f_T(W[2..k])} - 1$$

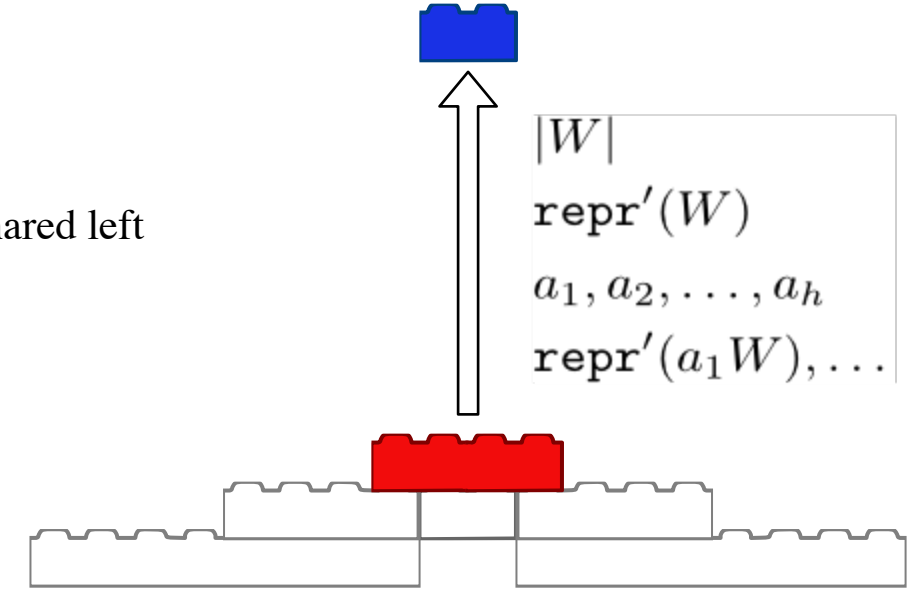
significance score

$$\begin{aligned} W \text{ present}, W[2..k-1] \text{ not a maximal repeat} &\Rightarrow z_T(W) = 0 \\ W \text{ absent, but not minimal absent} &\Rightarrow z_T(W) = 0 \\ W \text{ minimal absent} &\Rightarrow z_T = -1 \end{aligned}$$

J. Qi, B. Wang, B.-L. Hao. Whole proteome prokaryote phylogeny without sequence alignment: a *k*-string composition approach. J. Mol. Evol., 2004.
A. Apostolico, O. Denas. Fast algorithms for computing sequence distances by exhaustive substring composition. Algorithms for Molecular Biology, 2008.

Markovian corrections





Similar problem

Compute the Kullback-Leibler divergence vector for all k in a range $[k_1..k_2]$:

$$\mathsf{KL}_k = \sum_{W \in [1..\sigma]^k} p_T[W] \cdot (\log(p_T[W]) - \log(\tilde{p}_T[W]))$$

Only k-mers whose infix is a maximal repeat
can have nonzero contribution

Example application: find an upper bound on k for k-mer kernels.

G.E. Sims, S.-R. Jun, G.A Wu, S.-H. Kim. Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. PNAS, 2009.

In this work...

***k*-mers**	**Substrings**
k-mer complexity	Substring complexity
k-mer kernel	Substring kernel
k-mer profiling	Length-weighted substring kernel
k-mer kernel profile	Character-weighted substring kernel
k-th order empirical entropy	Minimal absent words complexity
KL divergence profile	Minimal absent words distance
	Substring kernel with Markovian corrections
	KL divergence kernel

all in $O(nd + \texttt{out})$ time and $o(n)$ bits

In this work...

***k*-mers**	**Substrings**
k-mer complexity	Substring complexity
k-mer kernel	Substring kernel
k-mer profiling	Length-weighted substring kernel
k-mer kernel profile	Character-weighted substring kernel
k-th order empirical entropy	Minimal absent words complexity
KL divergence profile	Minimal absent words distance
	Substring kernel with Markovian corrections
	KL divergence kernel

all in a single pass over the input

Algorithm design strategies

Independence on the order in which nodes of the suffix tree are enumerated

Telescoping

Batch processing to achieve output-sensitivity

Exploiting the traversal stack

Most "information" is "around" maximal repeats

A framework for space-efficient string kernels

Djamal Belazzougui[1], Fabio Cunial[2]

(1) Department of Computer Science, University of Helsinki, Finland.
(2) Max Planck Institute for Molecular Cell Biology and Genetics, Dresden, Germany.