Constructing phylo k-mers

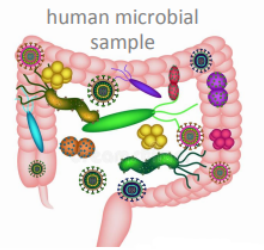
Phylogenetically-informed k-mers for phylogenetic placement and recombination detection

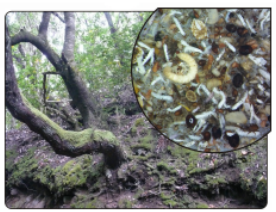
Nikolai Romashchenko

November 24, 2020

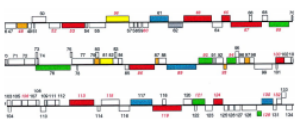
Phylo k-mers: Introduction

Many key bioinformatic tasks: how are new sequences related to those that we already know?





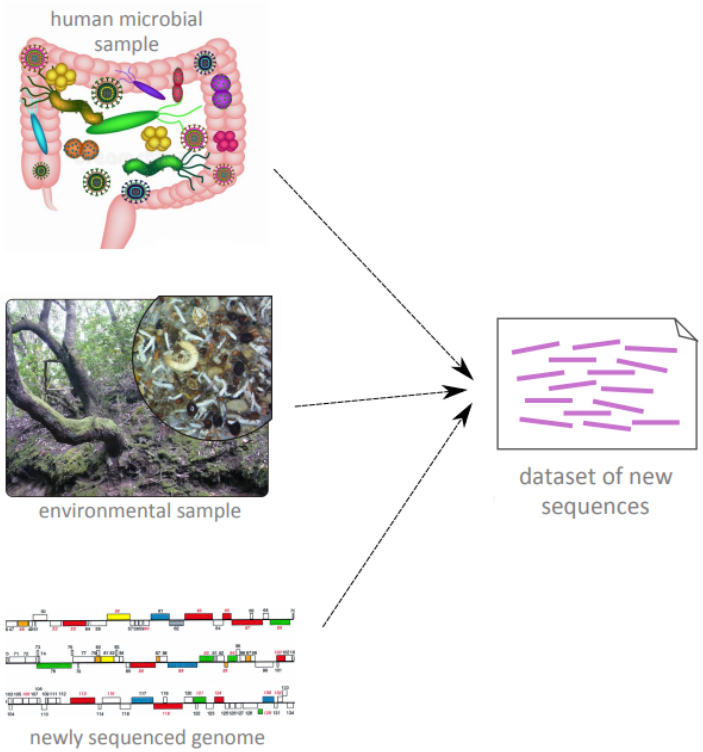
environmental sample



newly sequenced genome

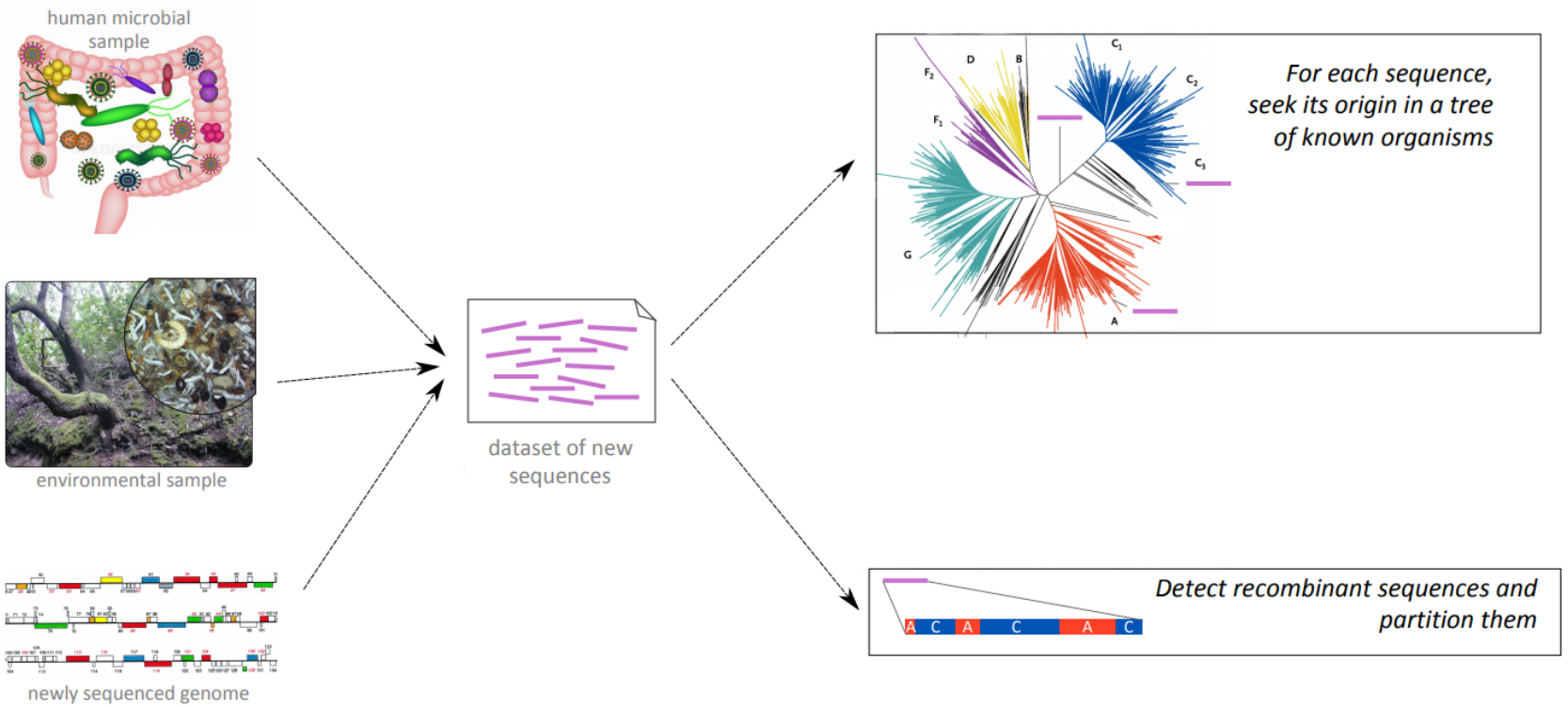
Phylo k-mers: Introduction

Many key bioinformatic tasks: how are new sequences related to those that we already know?



Phylo k-mers: Introduction

Many key bioinformatic tasks: how are new sequences related to those that we already know?

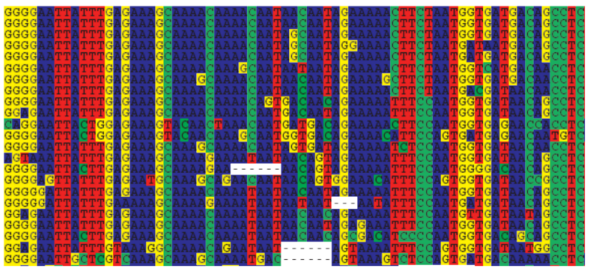


Phylo k-mers: Introduction (contd.)

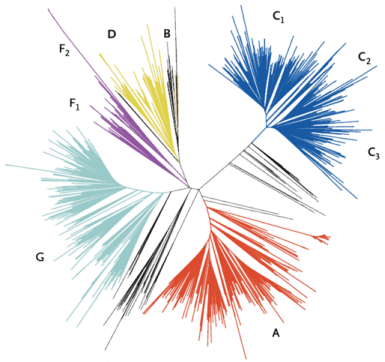
Many key bioinformatic tasks: how are new sequences related to those that we already know?

Alignment-based approaches:

known reference



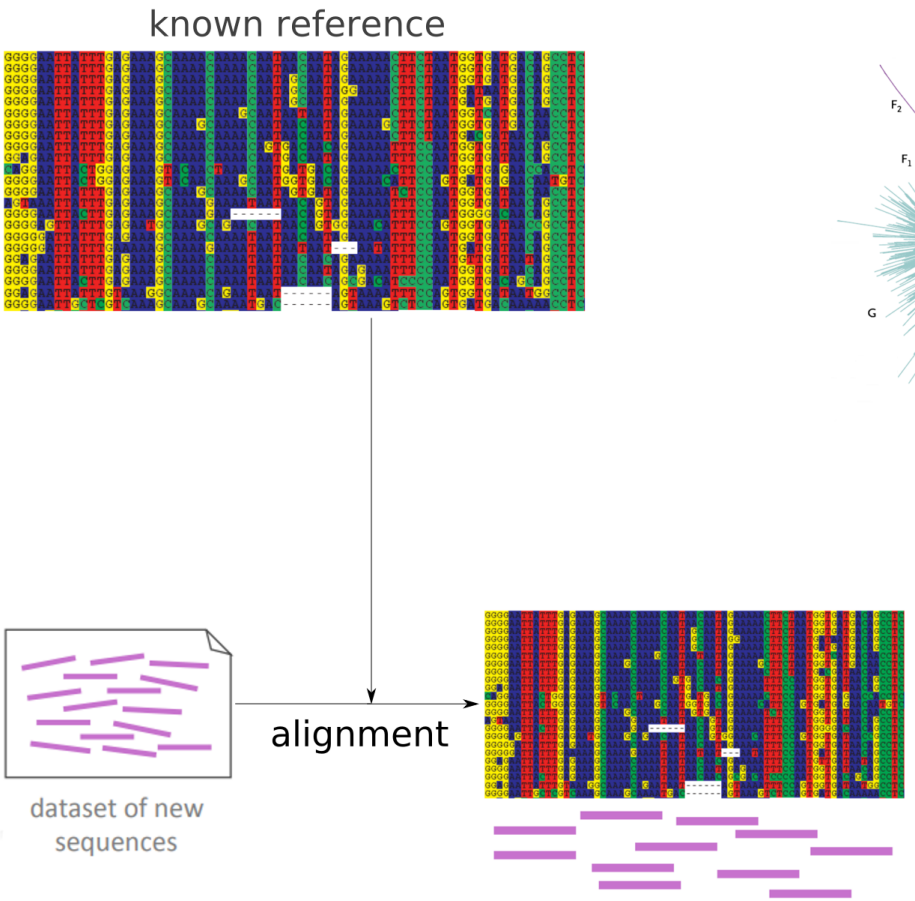
known tree



Phylo k-mers: Introduction (contd.)

Many key bioinformatic tasks: how are new sequences related to those that we already know?

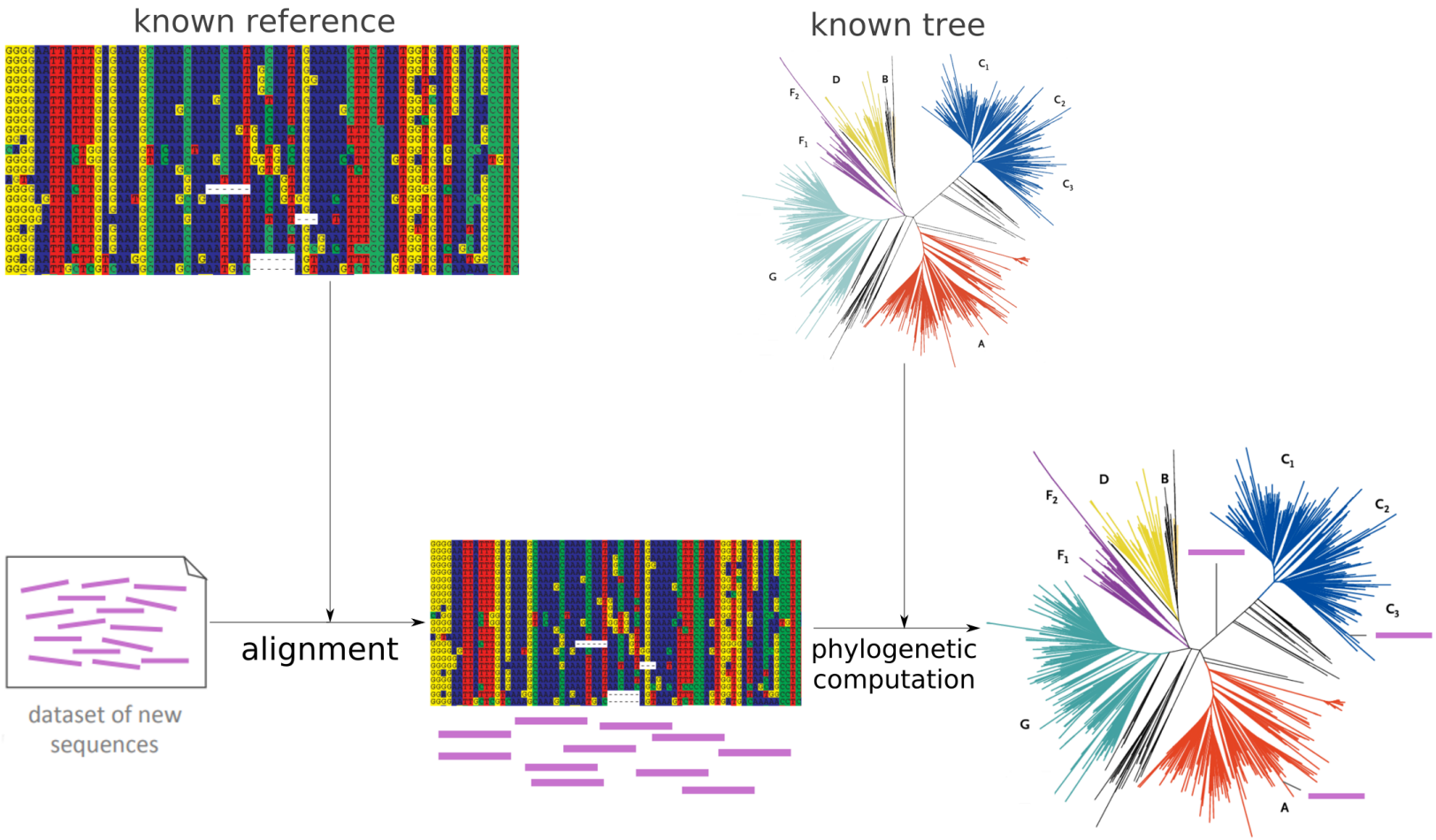
Alignment-based approaches:



Phylo k-mers: Introduction (contd.)

Many key bioinformatic tasks: how are new sequences related to those that we already know?

Alignment-based approaches:

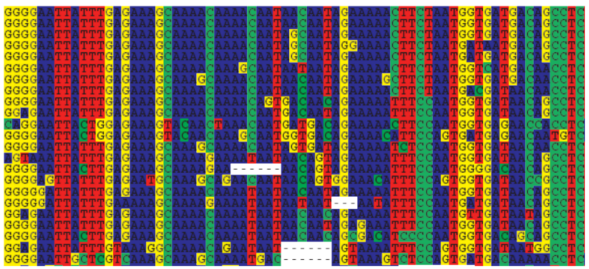


Phylo k-mers: Our new approach

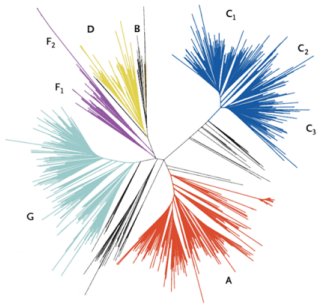
Many key bioinformatic tasks: how are new sequences related to those that we already know?

Our new approach:

known reference



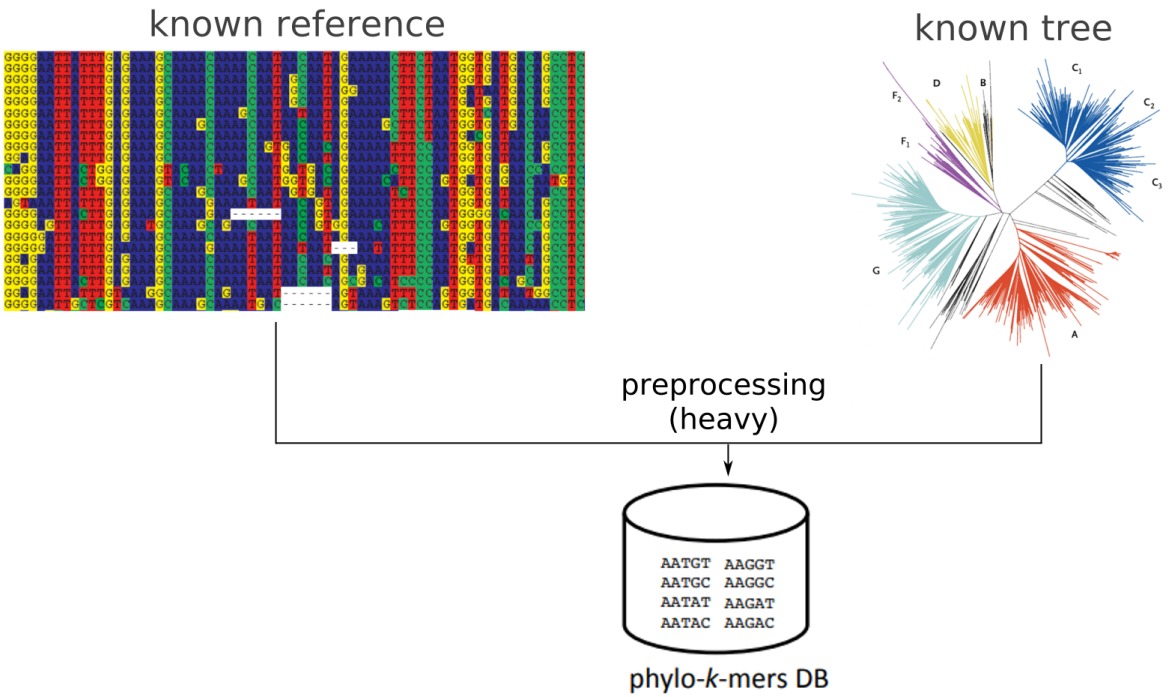
known tree



Phylo k-mers: Our new approach

Many key bioinformatic tasks: how are new sequences related to those that we already know?

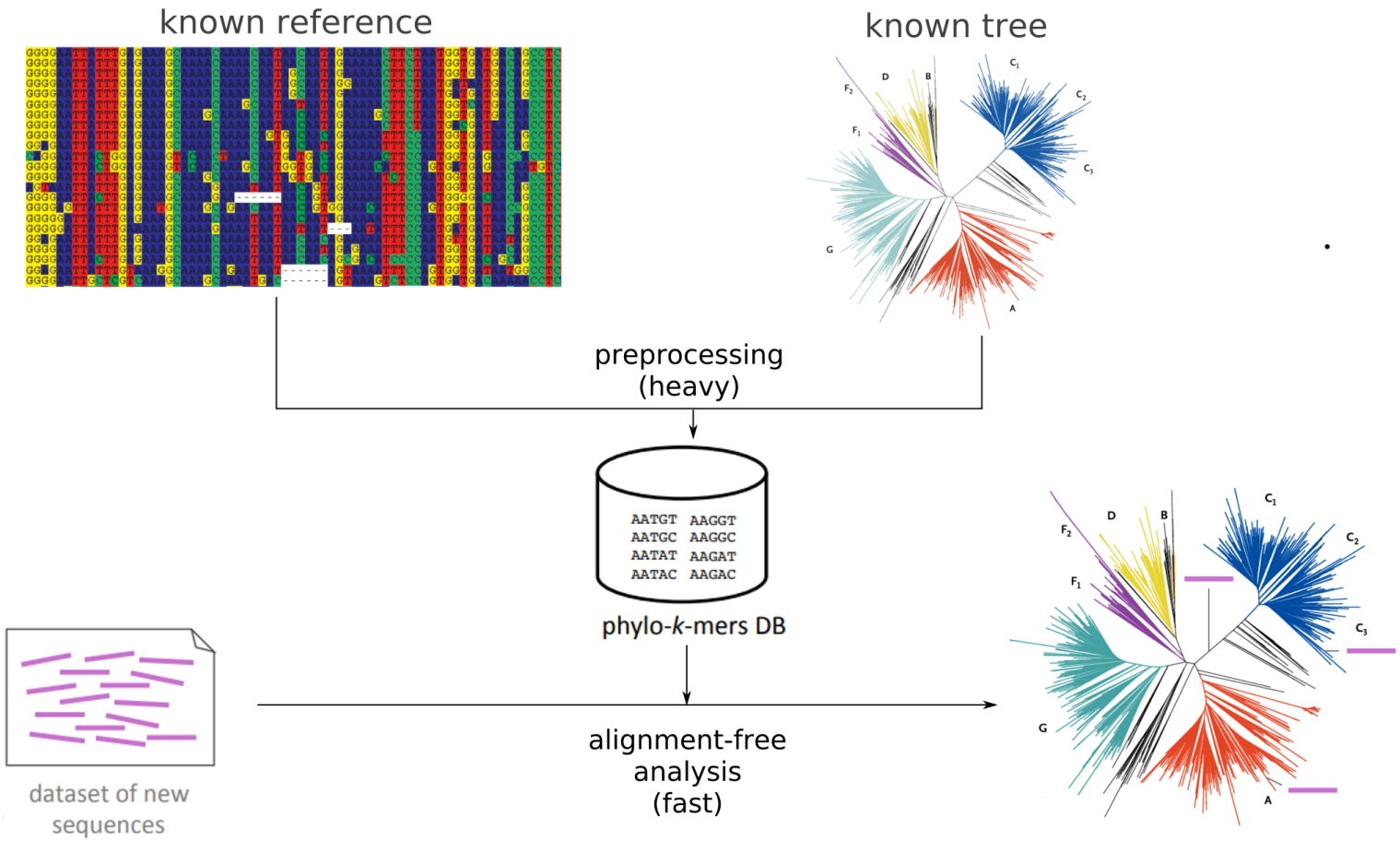
Our new approach:



Phylo k-mers: Our new approach

Many key bioinformatic tasks: how are new sequences related to those that we already know?

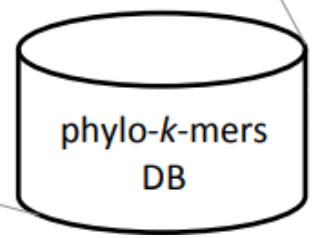
Our new approach:

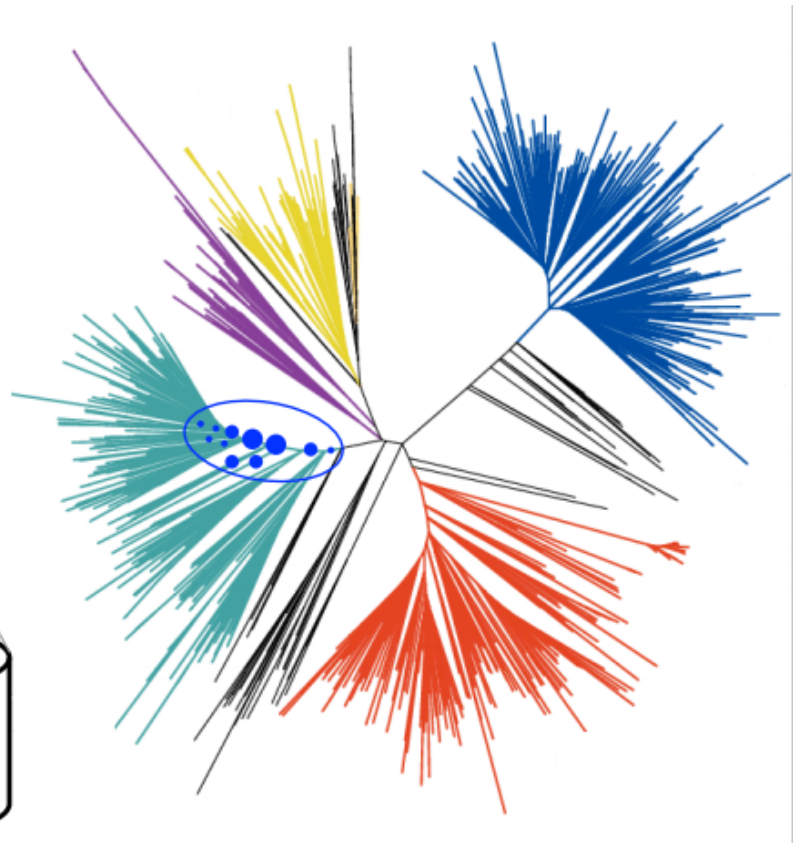


Phylo k-mers: Content of databases

"Phylo *k*-mers" = sequences of length *k* that are phylogenetically informative: we know where they could have come from, and with what probability.

k-mer	edge	probability
AACAAC	e_{27}	0.17
	e_{29}	0.11
	e_{42}	0.07
	⋮	⋮
AACATC	e_{53}	0.13
	e_{61}	0.29
	⋮	⋮
GATATT	e_{98}	0.01
	e_{192}	0.34
	e_{193}	0.19
	⋮	⋮
...	⋮	⋮



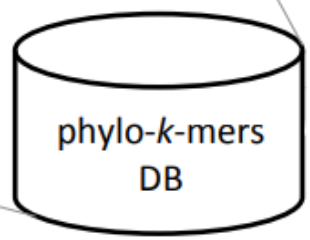


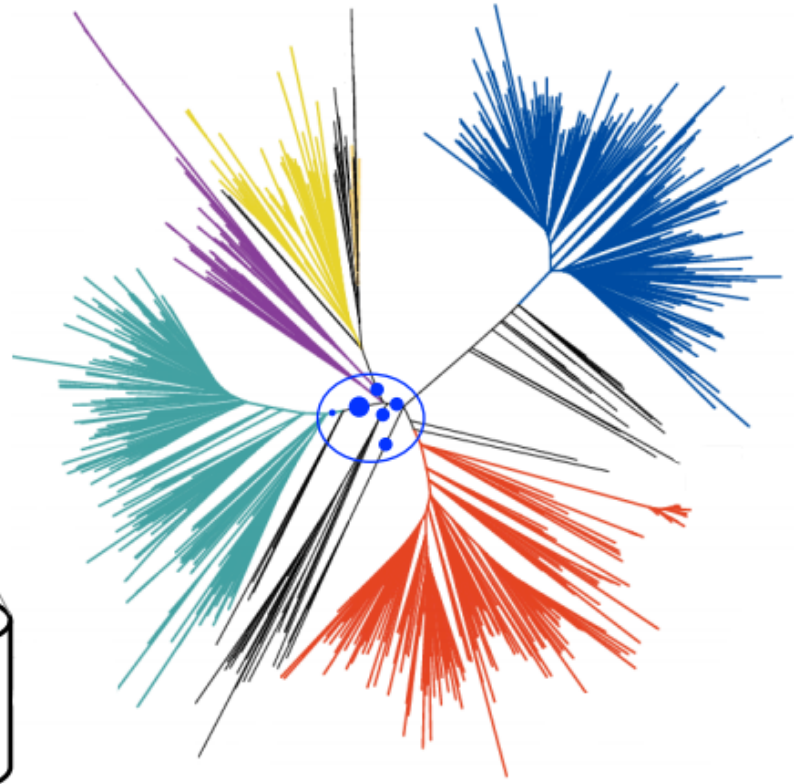
In practice we use longer *k*-mers ($k = 8, 10, 12 \ldots$)

Phylo k-mers: Content of databases

"Phylo *k*-mers" = sequences of length *k* that are phylogenetically informative: we know where they could have come from, and with what probability.

k-mer	edge	probability
AACAAC	e_{27}	0.17
	e_{29}	0.11
	e_{42}	0.07
	⋮	⋮
AACATC	e_{53}	0.13
	e_{61}	0.29
	⋮	⋮
GATATT	e_{98}	0.01
	e_{192}	0.34
	e_{193}	0.19
	⋮	⋮
...	⋮	⋮



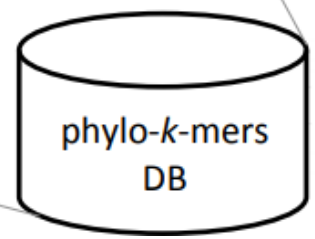


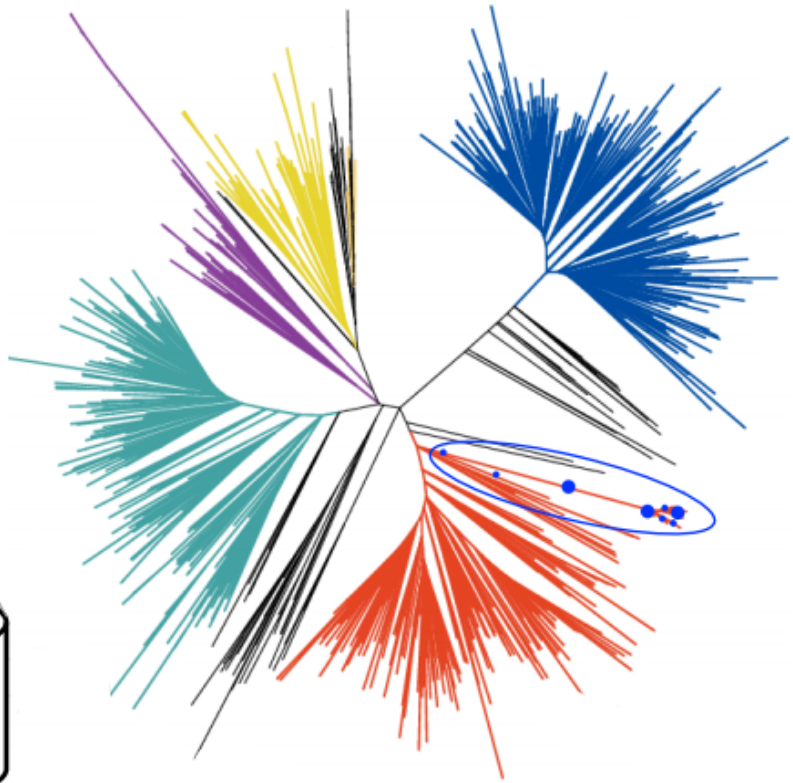
In practice we use longer *k*-mers ($k = 8, 10, 12 \ldots$)

Phylo k-mers: Content of databases

"Phylo *k*-mers" = sequences of length *k* that are phylogenetically informative: we know where they could have come from, and with what probability.

k-mer	edge	probability
AACAAC	e_{27}	0.17
	e_{29}	0.11
	e_{42}	0.07
	⋮	⋮
AACATC	e_{53}	0.13
	e_{61}	0.29
	⋮	⋮
GATATT	e_{98}	0.01
	e_{192}	0.34
	e_{193}	0.19
	⋮	⋮
...	⋮	⋮





In practice we use longer *k*-mers ($k = 8, 10, 12 \ldots$)

Phylo k-mer based tools

RAPPAS: phylogenetic placement

Rapid alignment-free phylogenetic identification of metagenomic sequences. B. Linard, K. Swenson, F. Pardi. Bioinformatics 2019:
doi.org/10.1093/bioinformatics/btz068

SHERPAS: viral recombination detection

[Accepted] Rapid screening and detection of inter-type viral recombinants using phylo-*k*-mers. G. Scholz, B. Linard, N. Romashchenko, E. Rivals, F. Pardi. Bioinformatics 2020.
Preprint: doi.org/10.1101/2020.06.22.161422

xpas: Phylo k-mer construction library

xpas is a library for phylo *k*-mer construction in SHERPAS and the new version of RAPPAS. Github: github.com/phylo42/xpas

[1]Yang, Z. (2007) PAML 4: Phylogenetic Analysis by Maximum Likelihood. Mol. Biol. Evol., 24, 1586–1591.

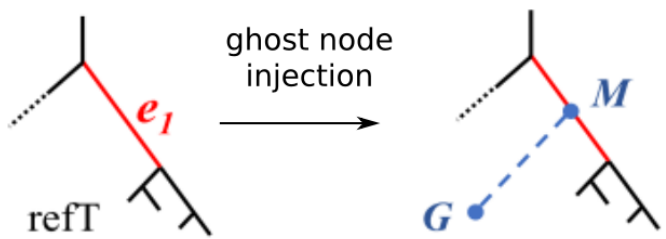
[2]Guindon, S. et al. (2010) New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. Systematic Biology 59 (3): 307–21.

[3]Kozlov, A. et at. (2019) RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. Bioinformatics, 35 (21). 4453–4455.

xpas: Phylo k-mer construction library

xpas is a library for phylo *k*-mer construction in SHERPAS and the new version of RAPPAS. Github: github.com/phylo42/xpas

1. Introducing ghost nodes (RAPPAS, xpas)



[1]Yang, Z. (2007) PAML 4: Phylogenetic Analysis by Maximum Likelihood. Mol. Biol. Evol., 24, 1586–1591.

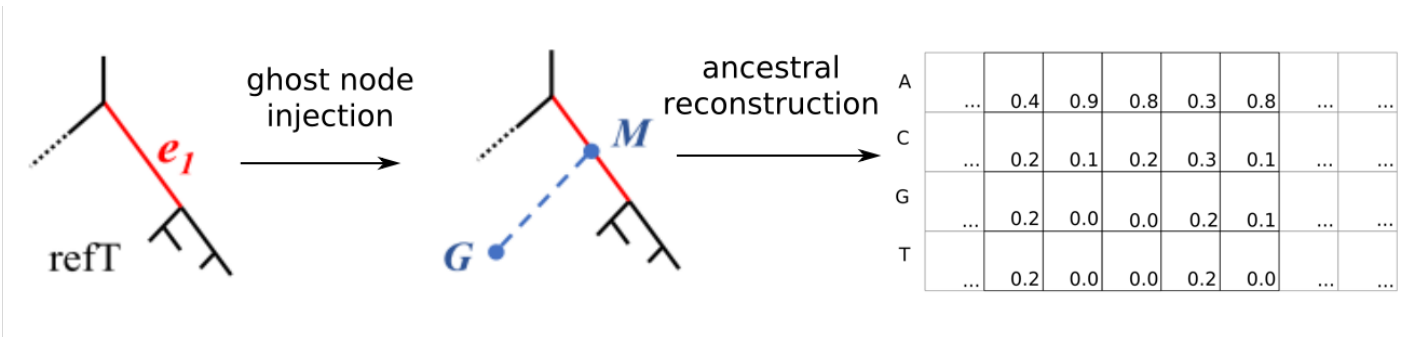
[2]Guindon, S. et al. (2010) New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. Systematic Biology 59 (3): 307–21.

[3]Kozlov, A. et at. (2019) RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. Bioinformatics, 35 (21). 4453–4455.

xpas: Phylo k-mer construction library

xpas is a library for phylo *k*-mer construction in SHERPAS and the new version of RAPPAS. Github: github.com/phylo42/xpas

1. Introducing ghost nodes (RAPPAS, xpas)
2. Ancestral reconstruction (PAML[1], PhyML[2], RAxML-NG[3])2



[1]Yang, Z. (2007) PAML 4: Phylogenetic Analysis by Maximum Likelihood. Mol. Biol. Evol., 24, 1586–1591.

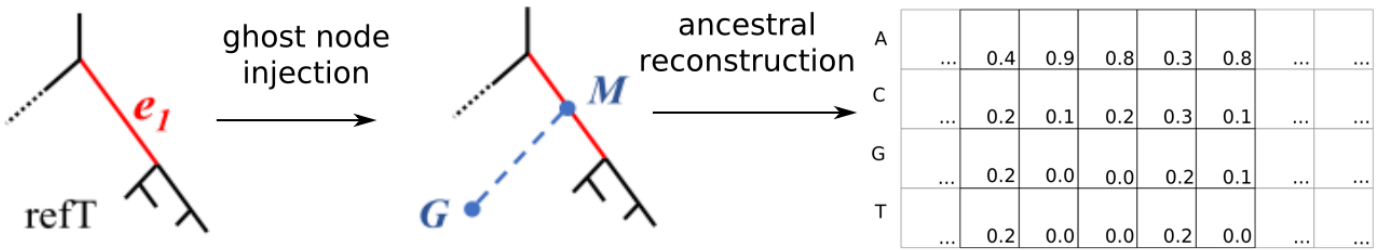
[2]Guindon, S. et al. (2010) New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. Systematic Biology 59 (3): 307–21.

[3]Kozlov, A. et at. (2019) RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. Bioinformatics, 35 (21). 4453–4455.

xpas: Phylo k-mer construction library

xpas is a library for phylo *k*-mer construction in SHERPAS and the new version of RAPPAS. Github: github.com/phylo42/xpas

1. Introducing ghost nodes (RAPPAS, xpas)
2. Ancestral reconstruction (PAML[1], PhyML[2], RAxML-NG[3])2
3. Calculating of phylo *k*-mers (RAPPAS, xpas)



[1]Yang, Z. (2007) PAML 4: Phylogenetic Analysis by Maximum Likelihood. Mol. Biol. Evol., 24, 1586–1591.

[2]Guindon, S. et al. (2010) New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. Systematic Biology 59 (3): 307–21.

[3]Kozlov, A. et at. (2019) RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. Bioinformatics, 35 (21). 4453–4455.

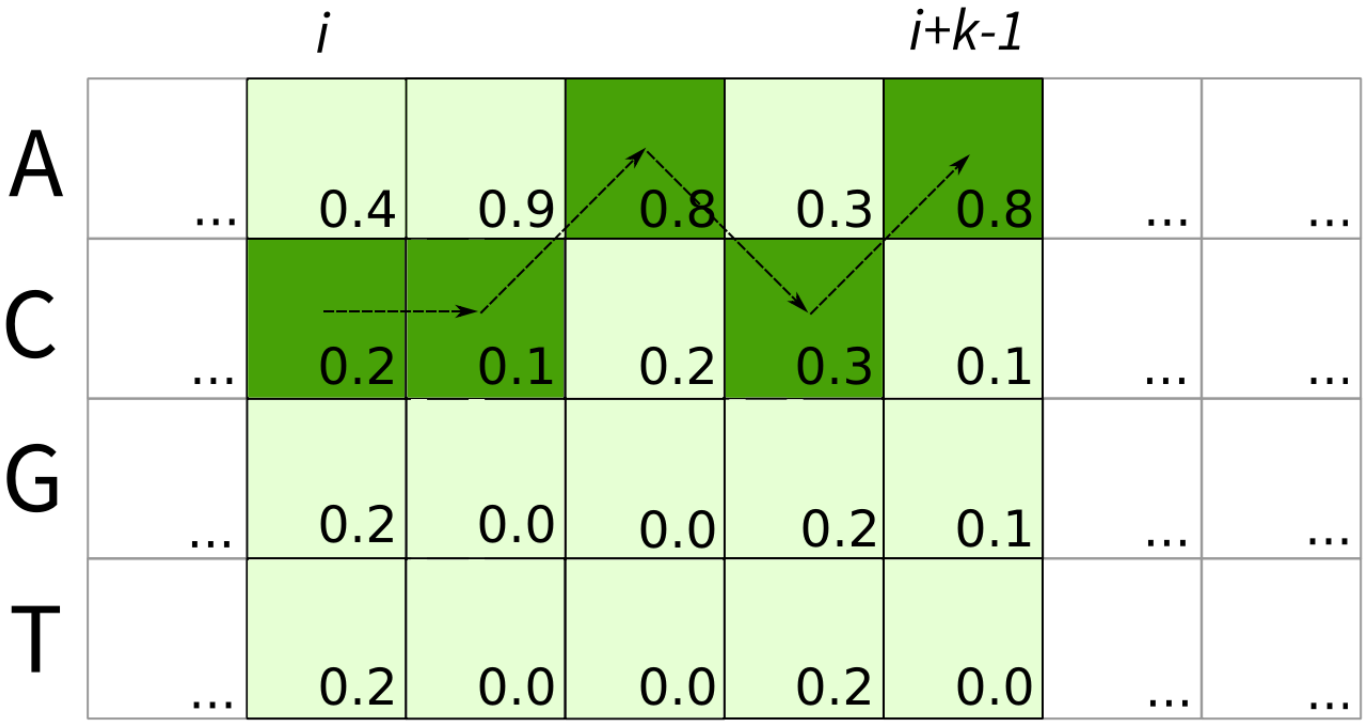
Phylo k-mer construction: Intuition

We can straightforwardly compute scores of *k*-mers for a given position of the alignment, by just multiplying corresponding values.

		i				*i+k-1*		
A	...	0.4	0.9	0.8	0.3	0.8	...	...
C	...	0.2	0.1	0.2	0.3	0.1	...	...
G	...	0.2	0.0	0.0	0.2	0.1	...	...
T	...	0.2	0.0	0.0	0.2	0.0	...	...

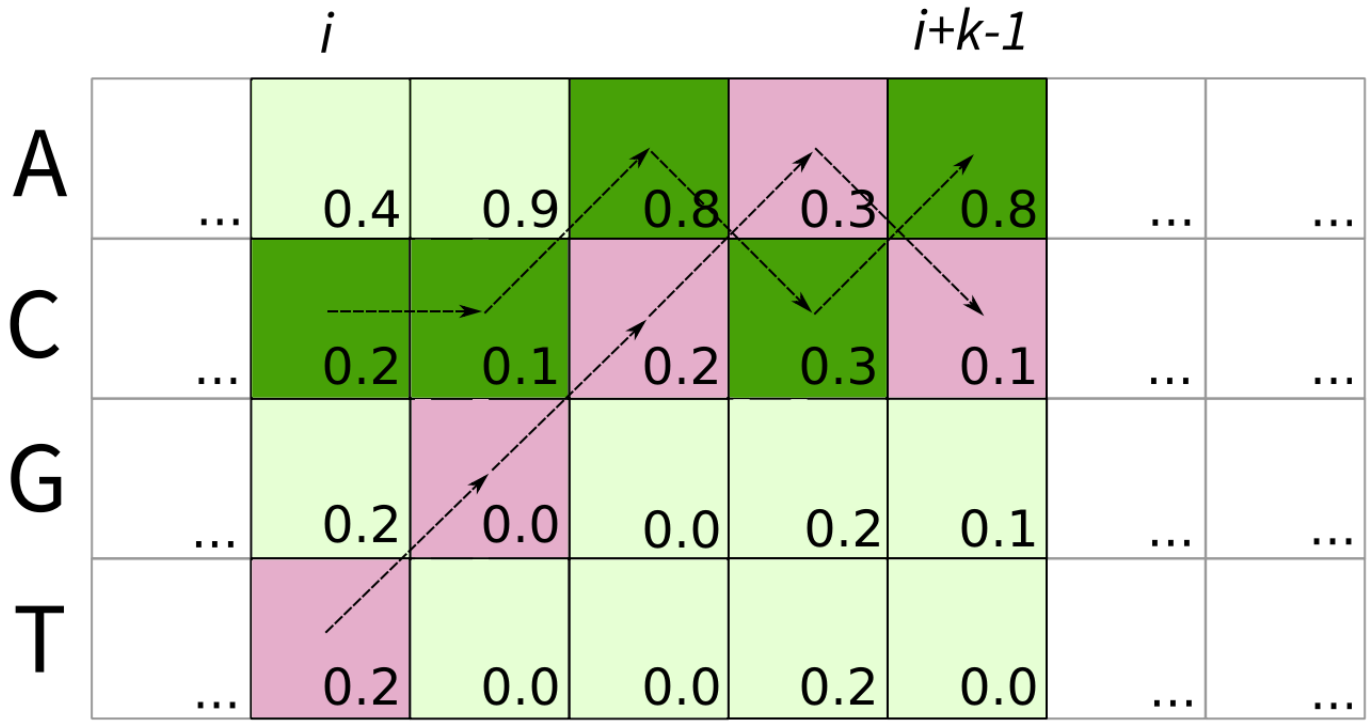
Phylo k-mer construction: Intuition

We can straightforwardly compute scores of *k*-mers for a given position of the alignment, by just multiplying corresponding values.



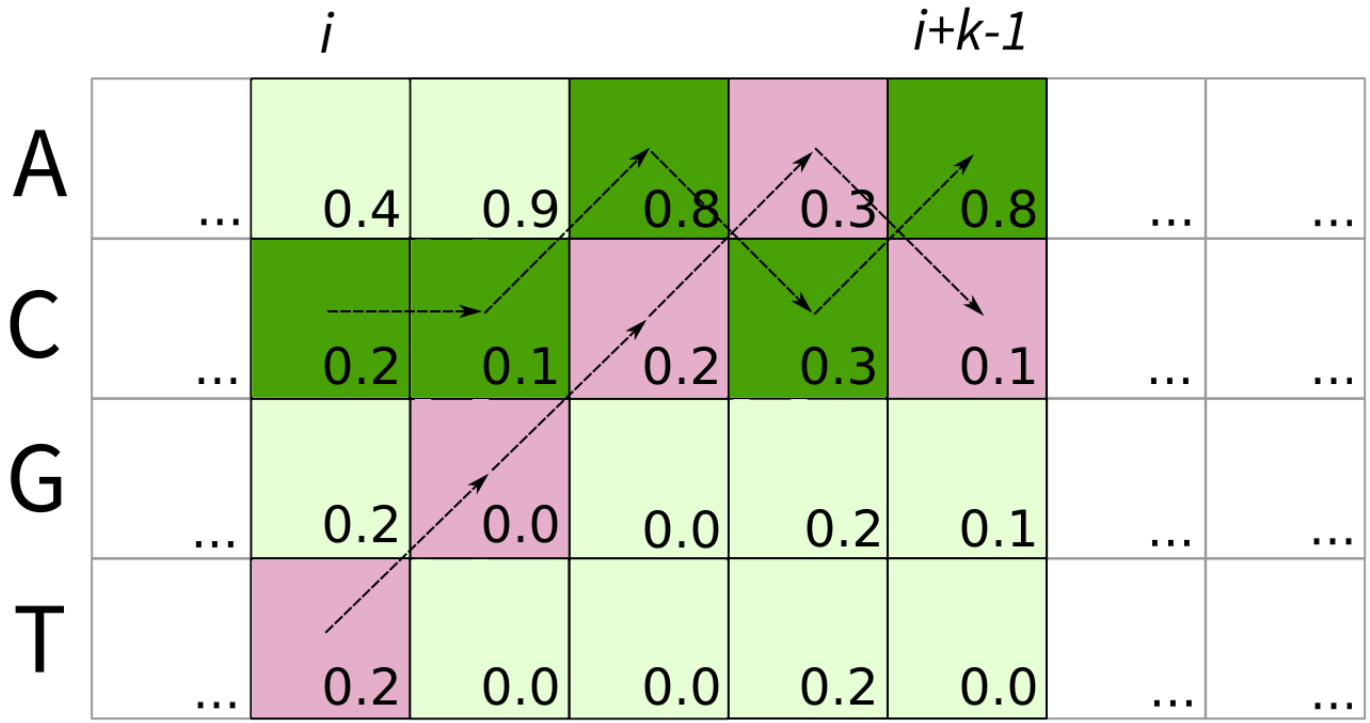
Phylo k-mer construction: Intuition

We can straightforwardly compute scores of *k*-mers for a given position of the alignment, by just multiplying corresponding values.



Phylo k-mer construction: Intuition

We can straightforwardly compute scores of *k*-mers for a given position of the alignment, by just multiplying corresponding values.

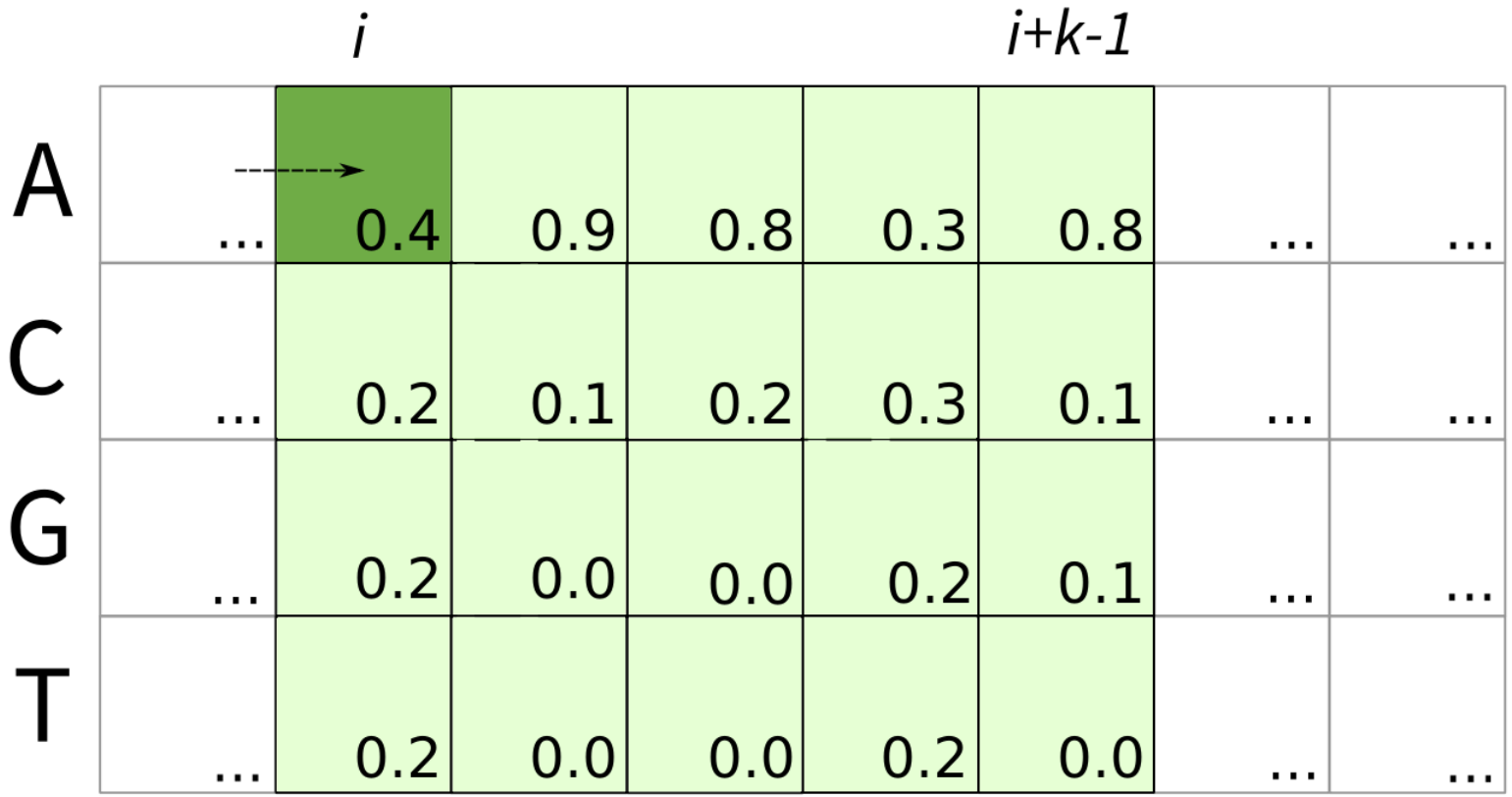


Problem

There are 4^k different combinations, and it must be done for every window of size *k* across the alignment.

Phylo k-mer construction: Naive algorithm

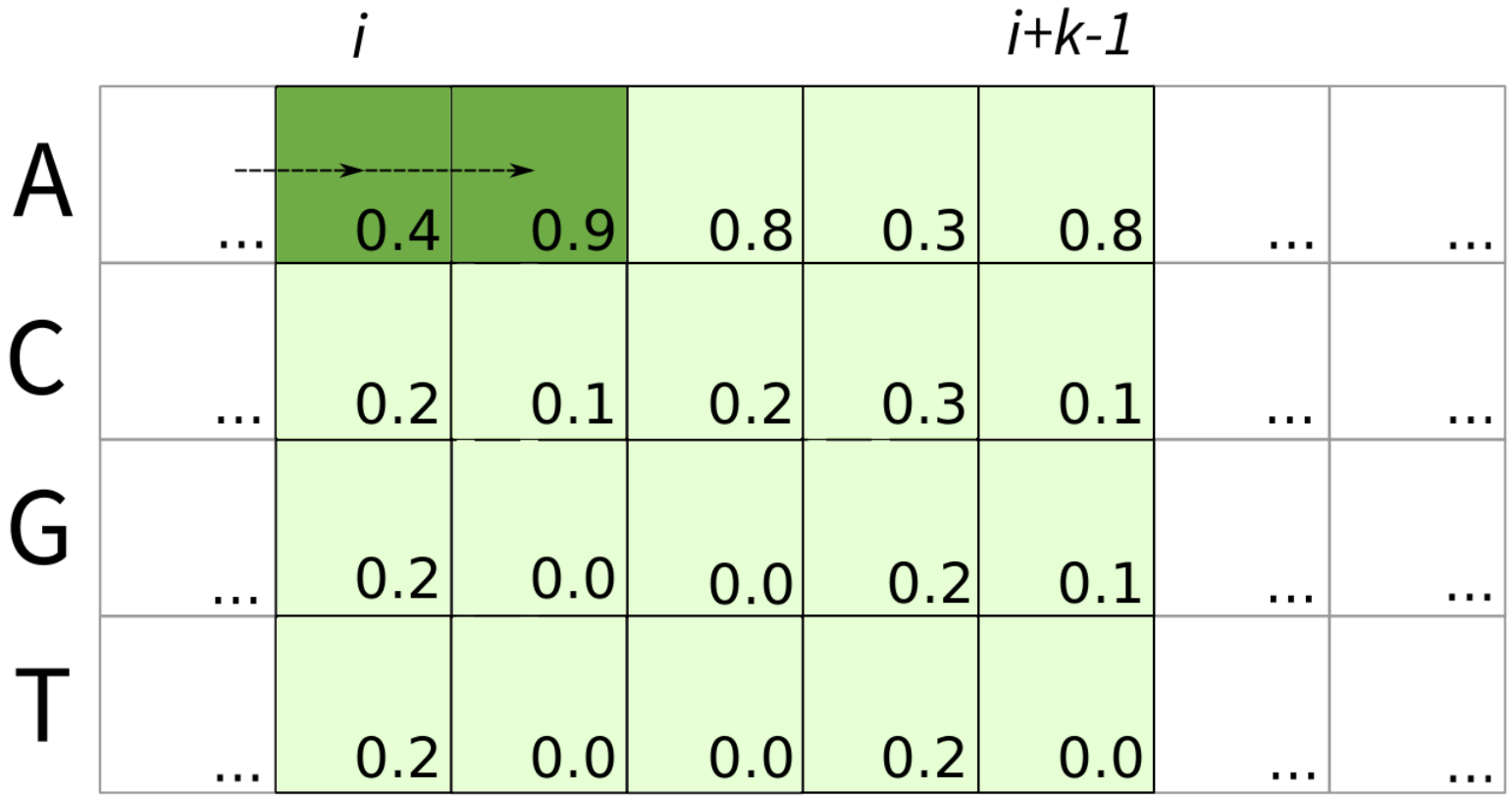
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *A*

Phylo k-mer construction: Naive algorithm

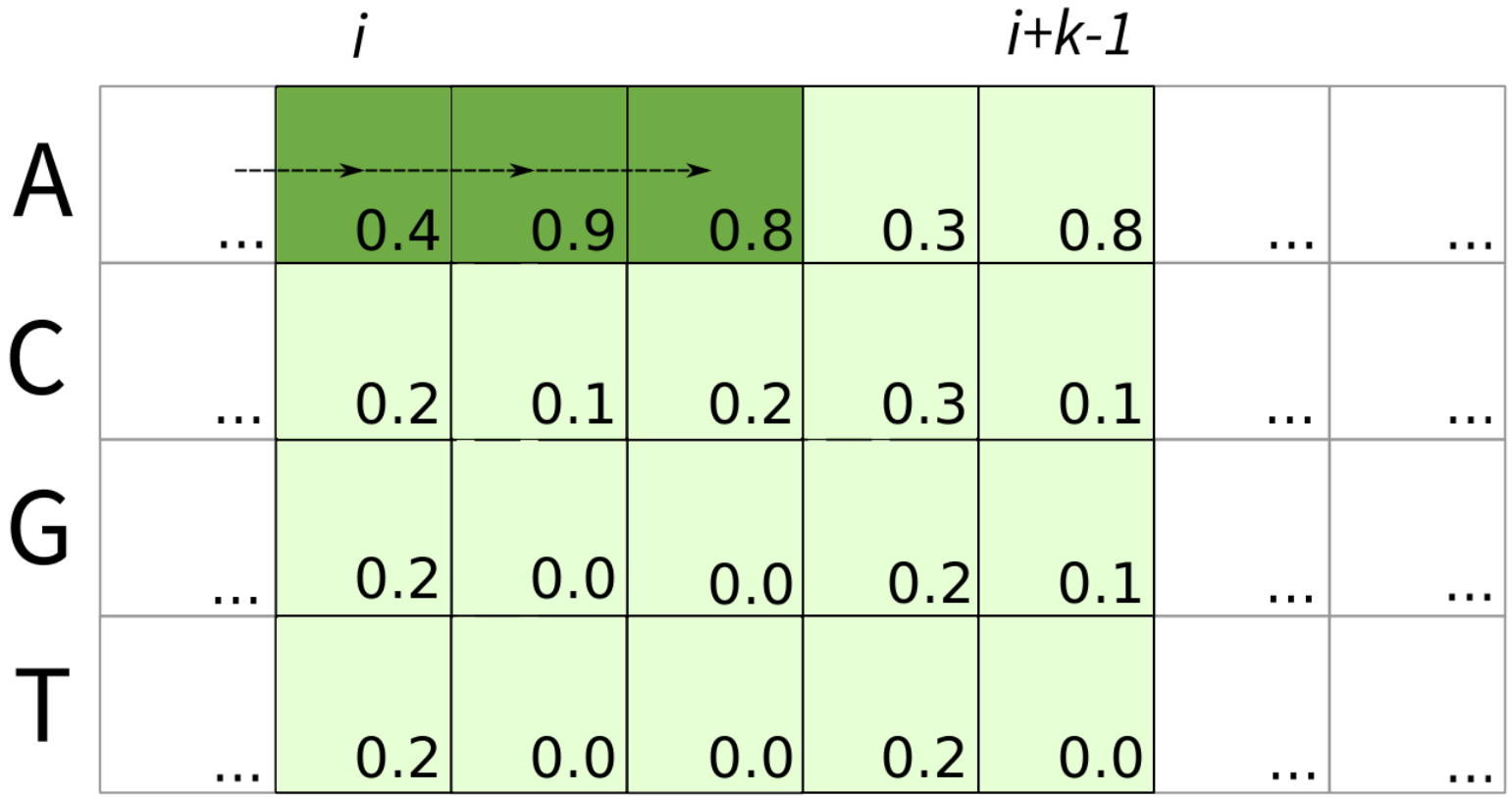
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AA*

Phylo k-mer construction: Naive algorithm

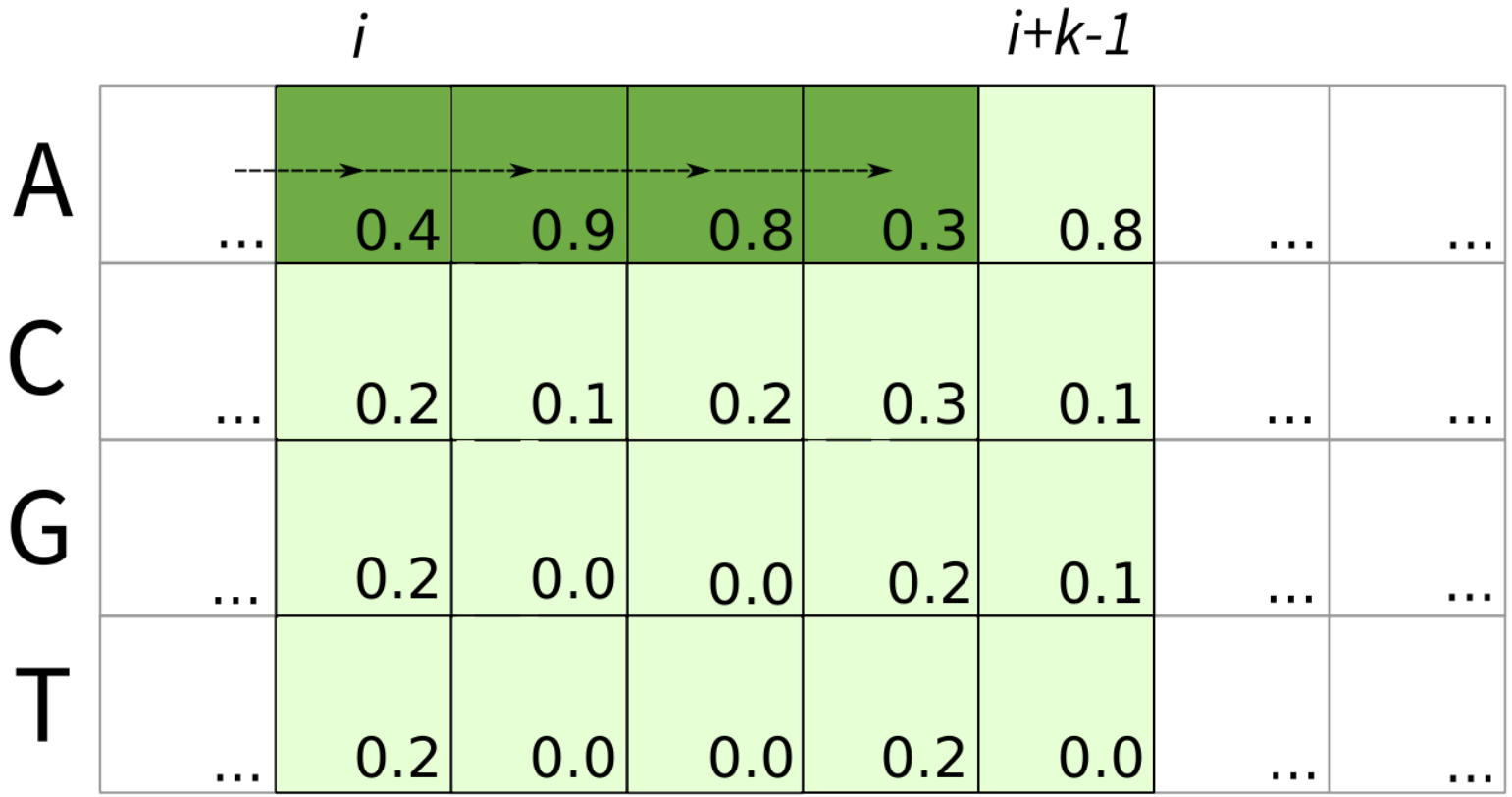
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAA*

Phylo k-mer construction: Naive algorithm

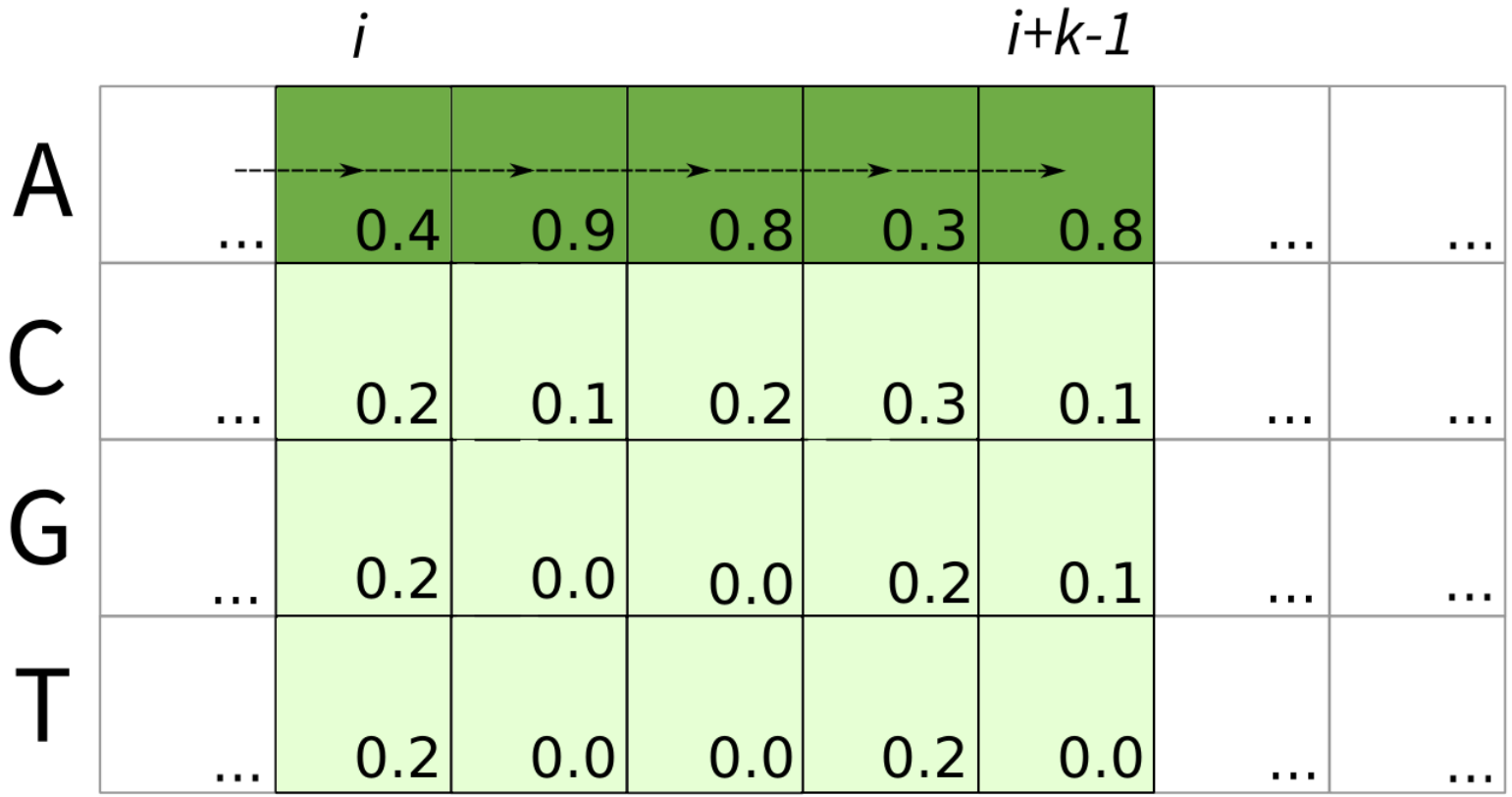
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAAA*

Phylo k-mer construction: Naive algorithm

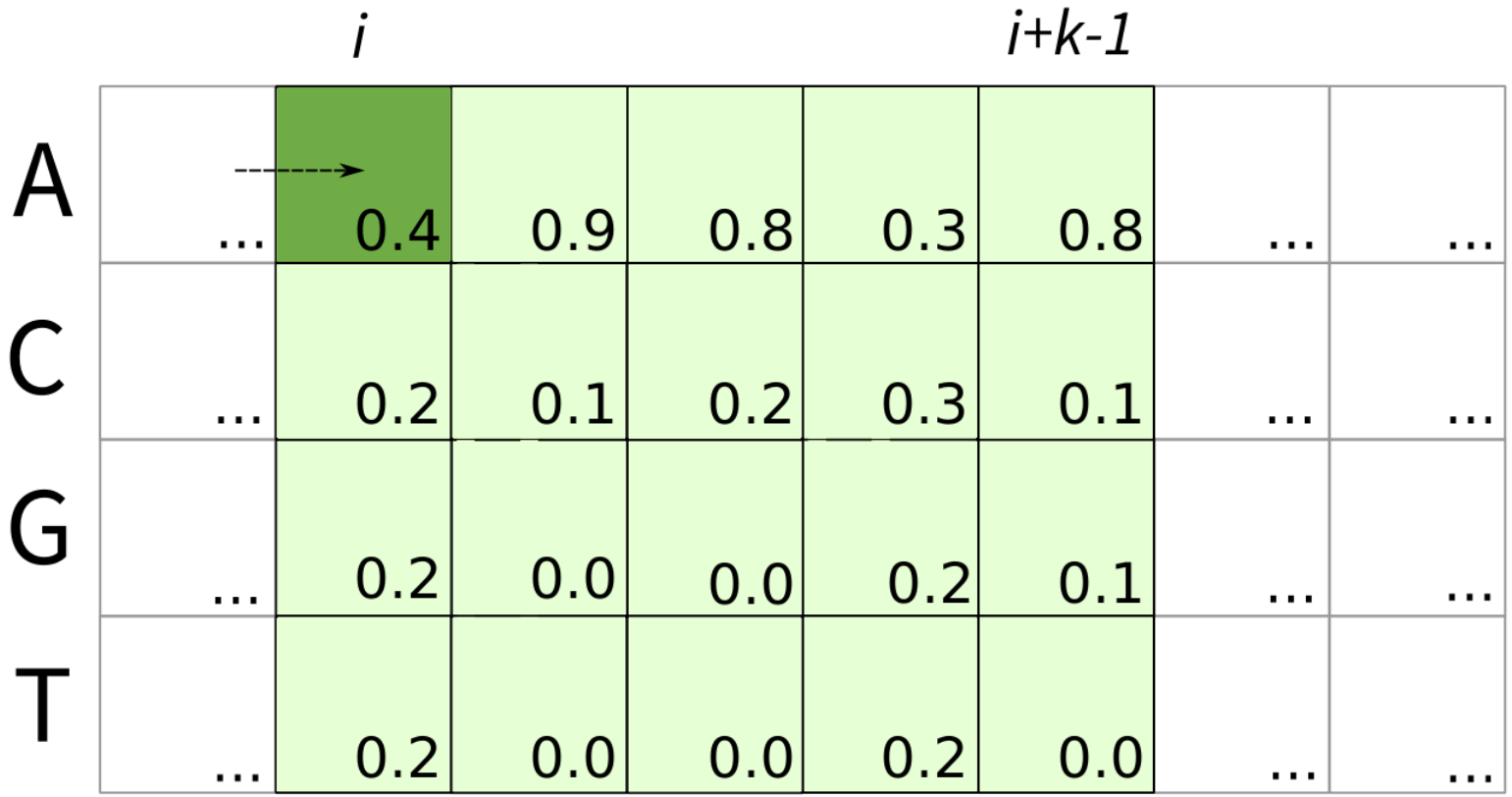
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAAAA* $\xrightarrow{\text{save}}$ `DB`

Phylo k-mer construction: Naive algorithm

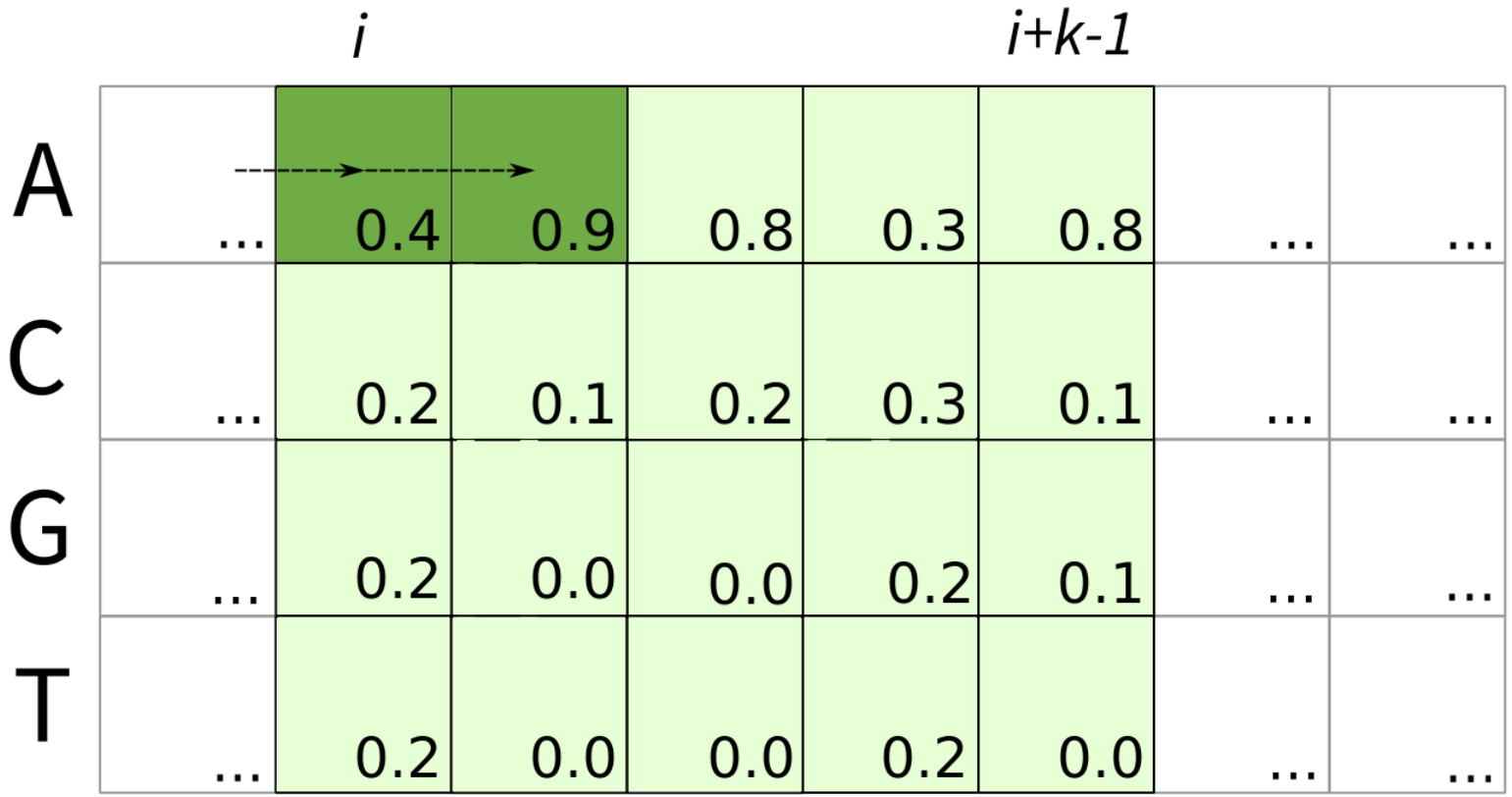
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *A*

Phylo k-mer construction: Naive algorithm

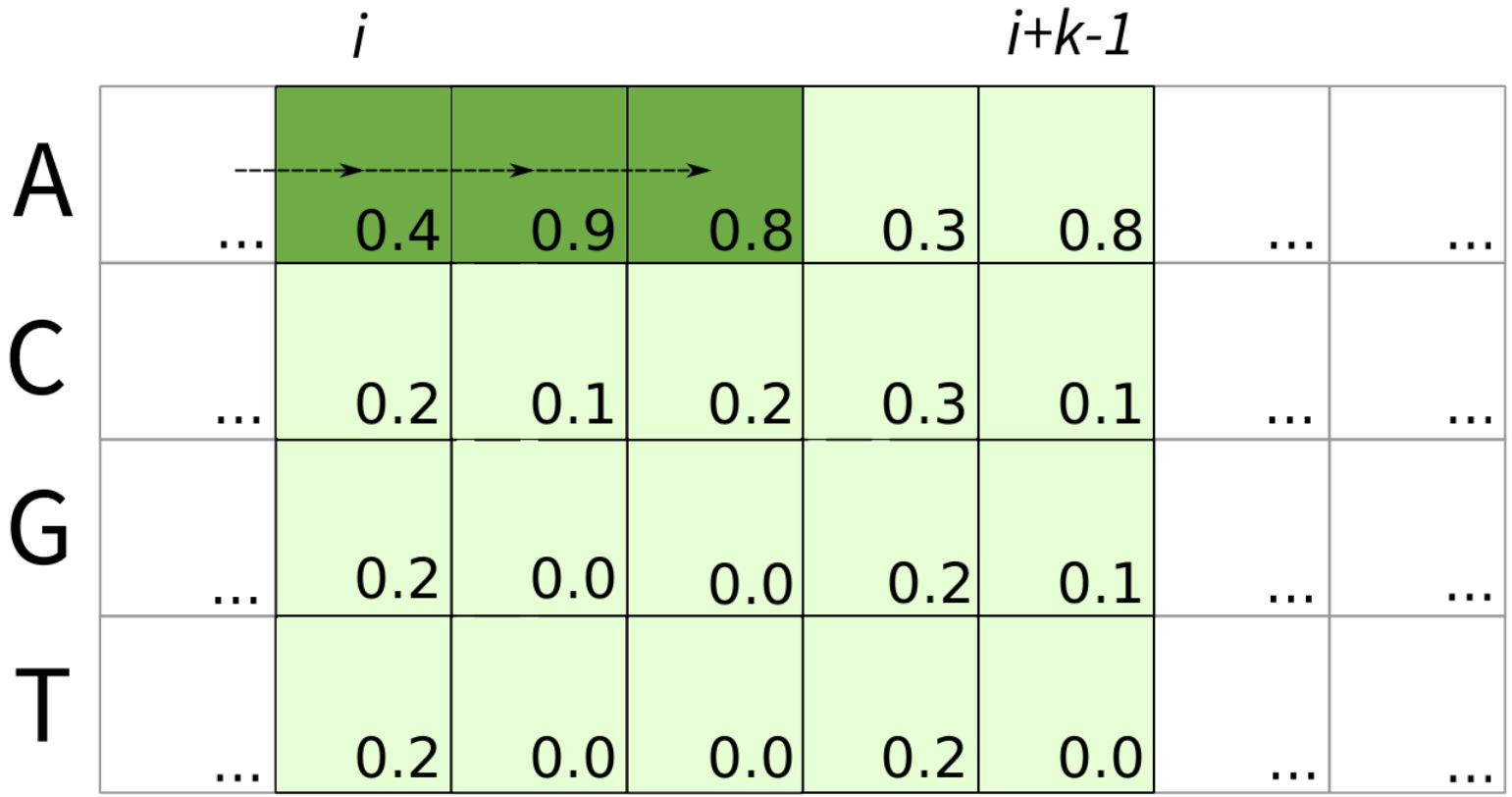
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAAA*

Phylo k-mer construction: Naive algorithm

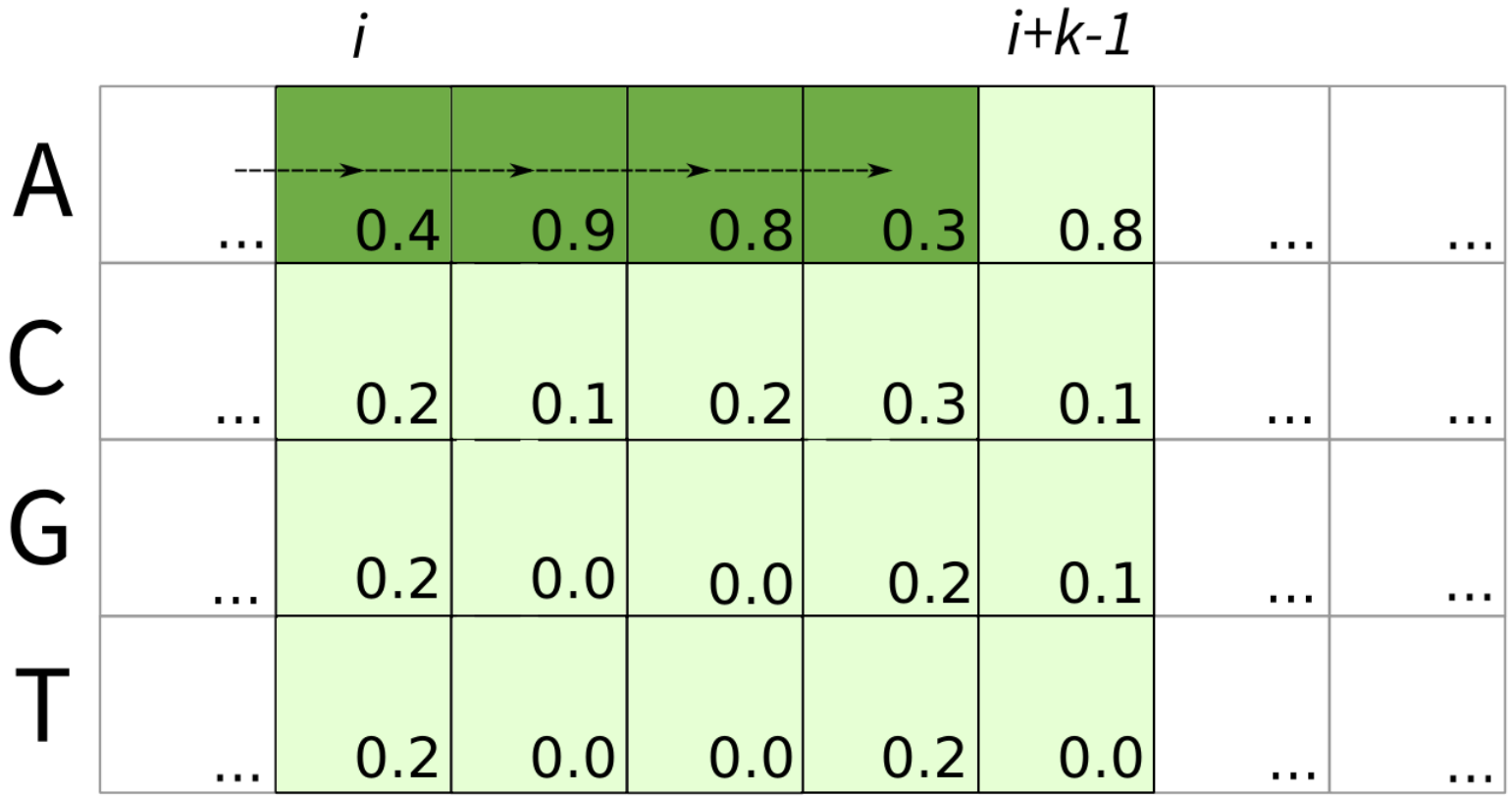
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAA*

Phylo k-mer construction: Naive algorithm

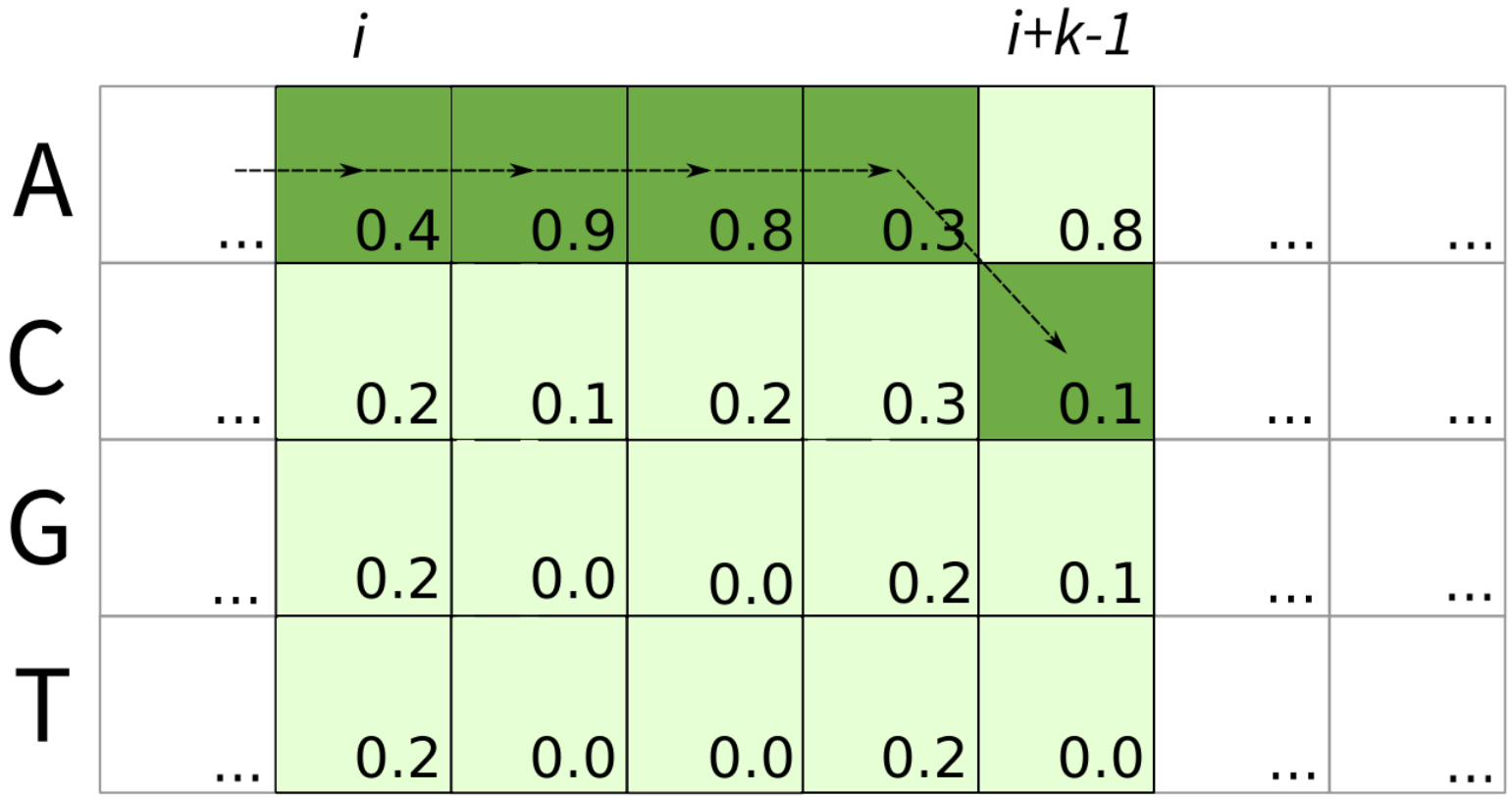
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAAA*

Phylo k-mer construction: Naive algorithm

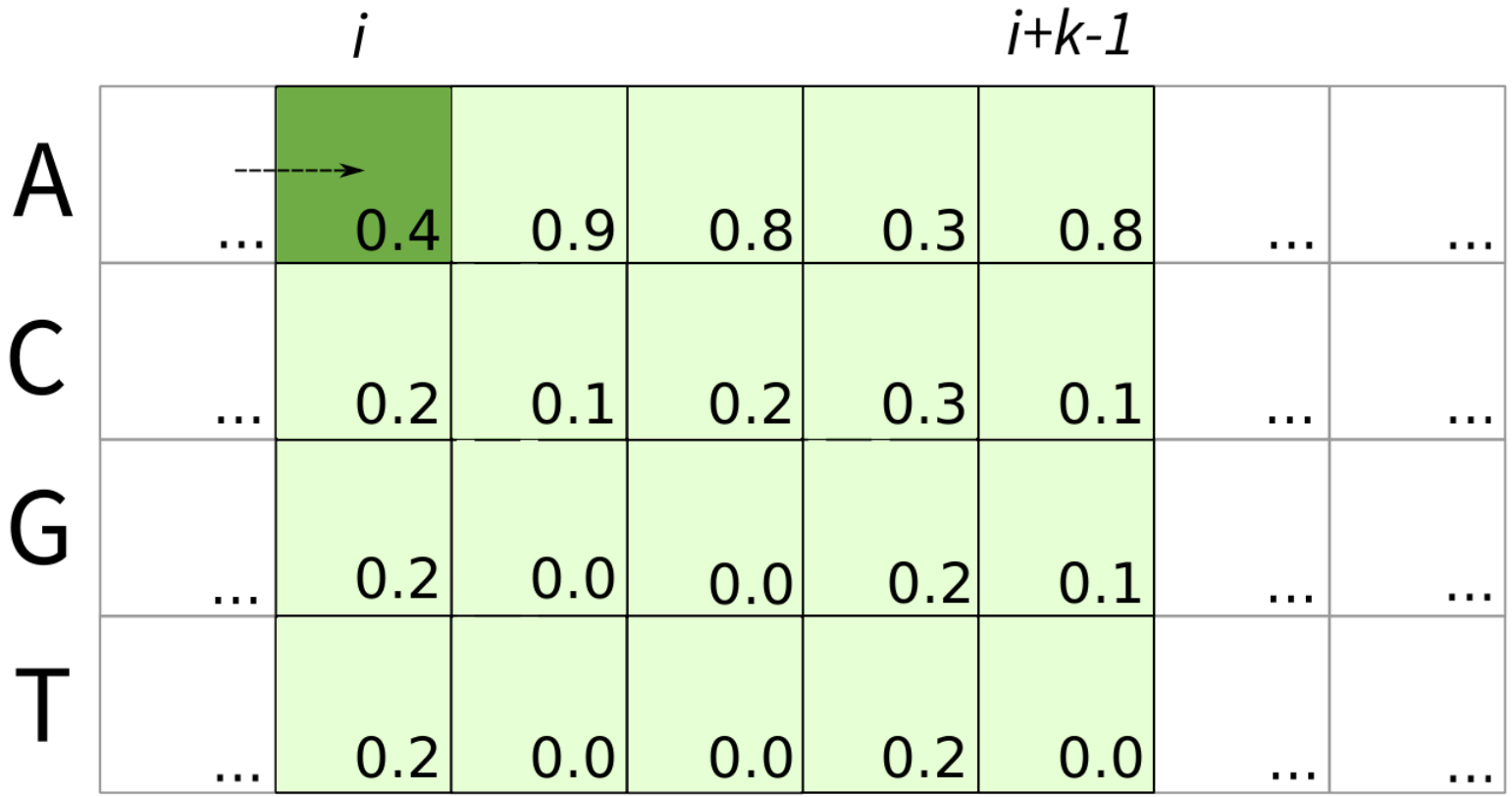
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAAAC* $\xrightarrow{\text{save}}$ `DB`

Phylo k-mer construction: Naive algorithm

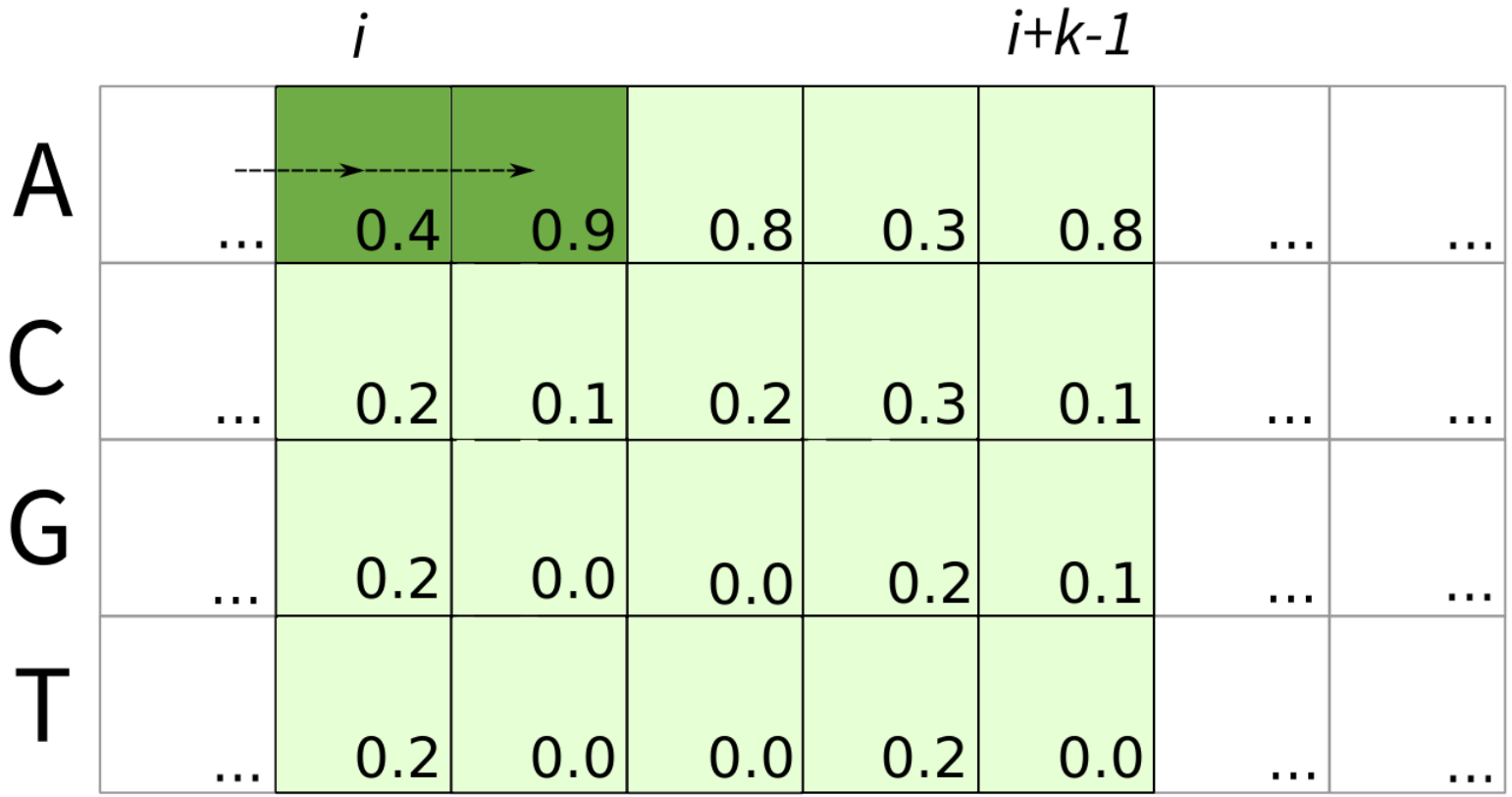
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *A*

Phylo k-mer construction: Naive algorithm

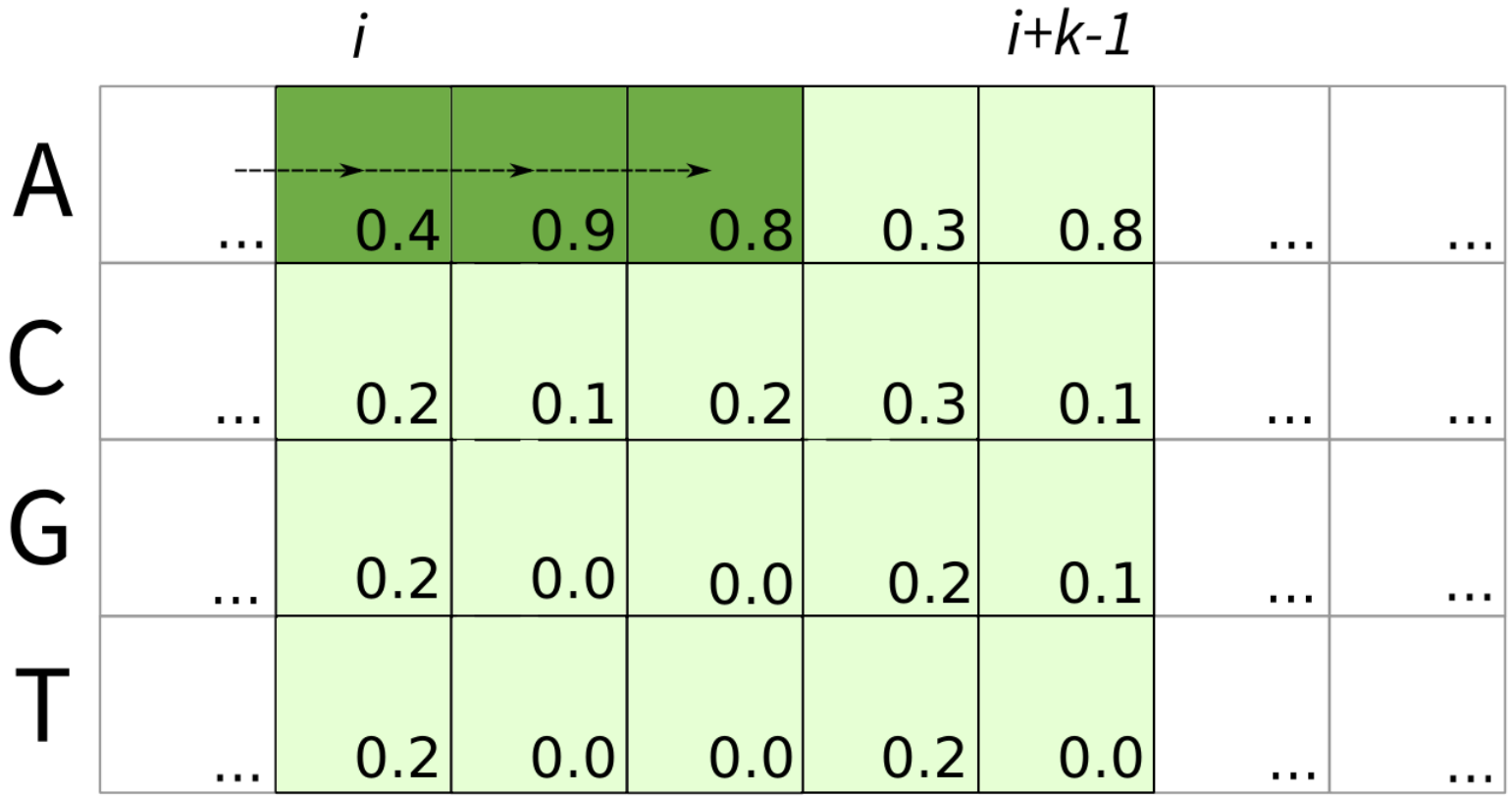
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AA*

Phylo k-mer construction: Naive algorithm

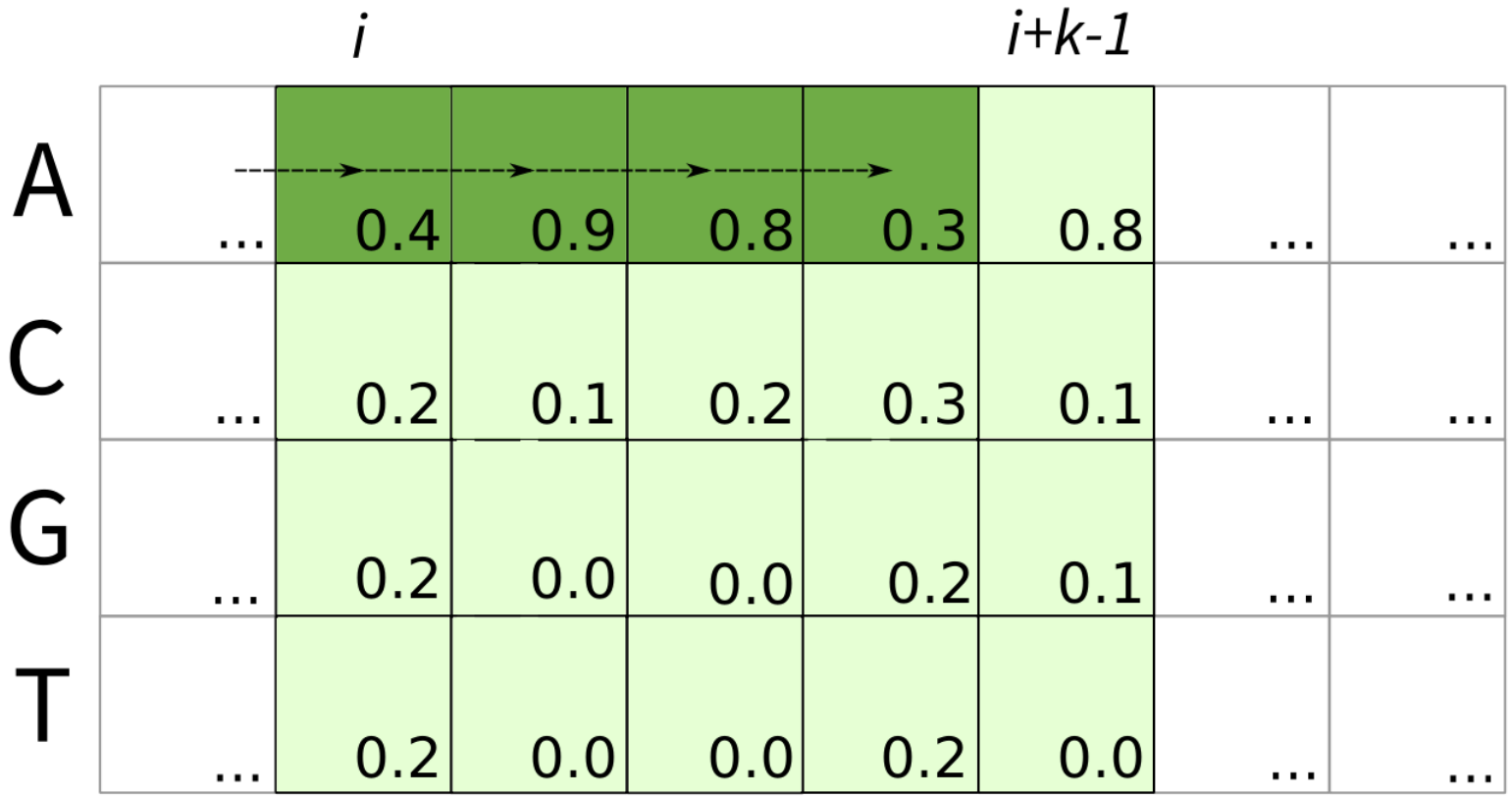
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAA*

Phylo k-mer construction: Naive algorithm

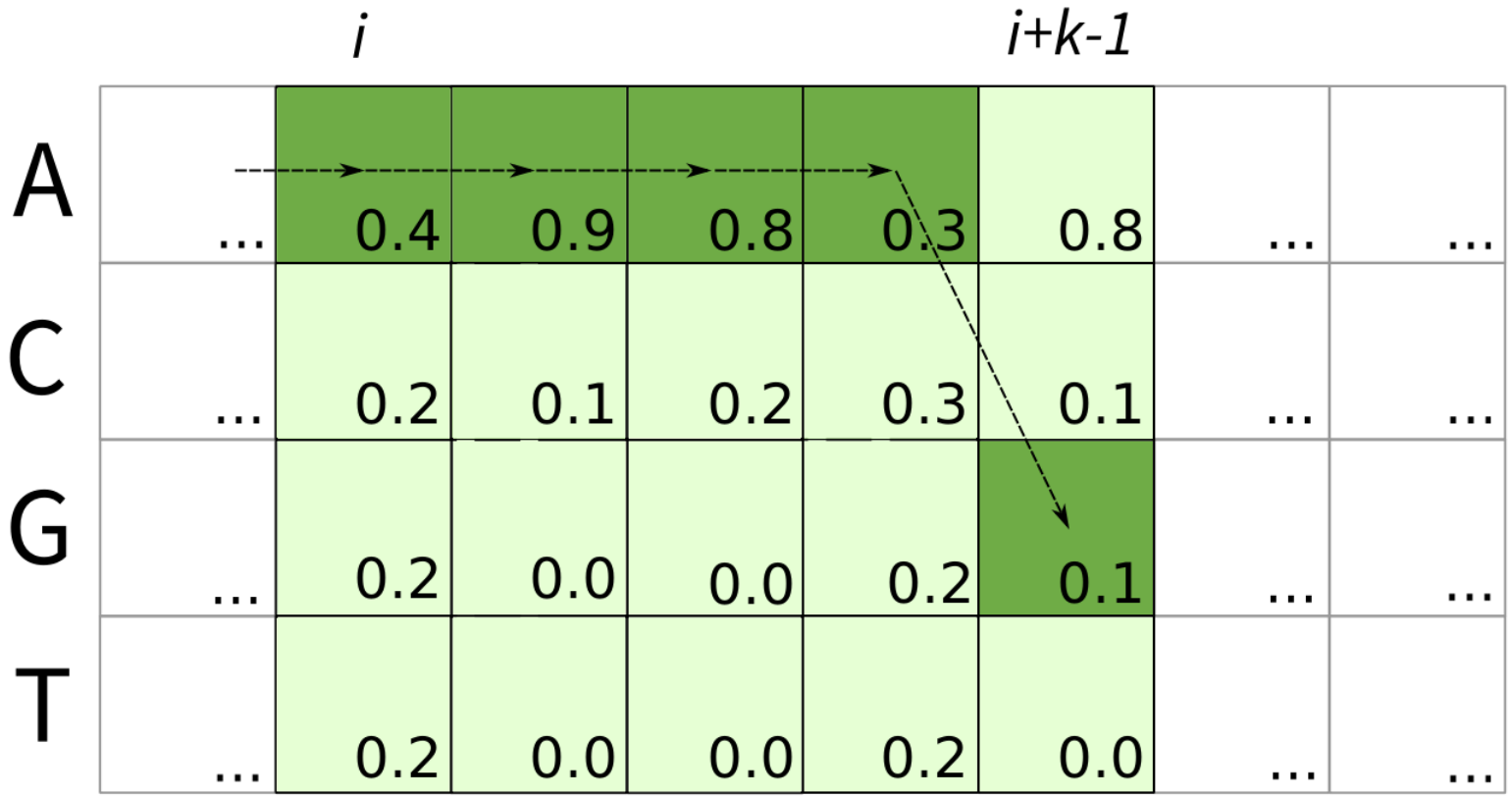
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAAA*

Phylo k-mer construction: Naive algorithm

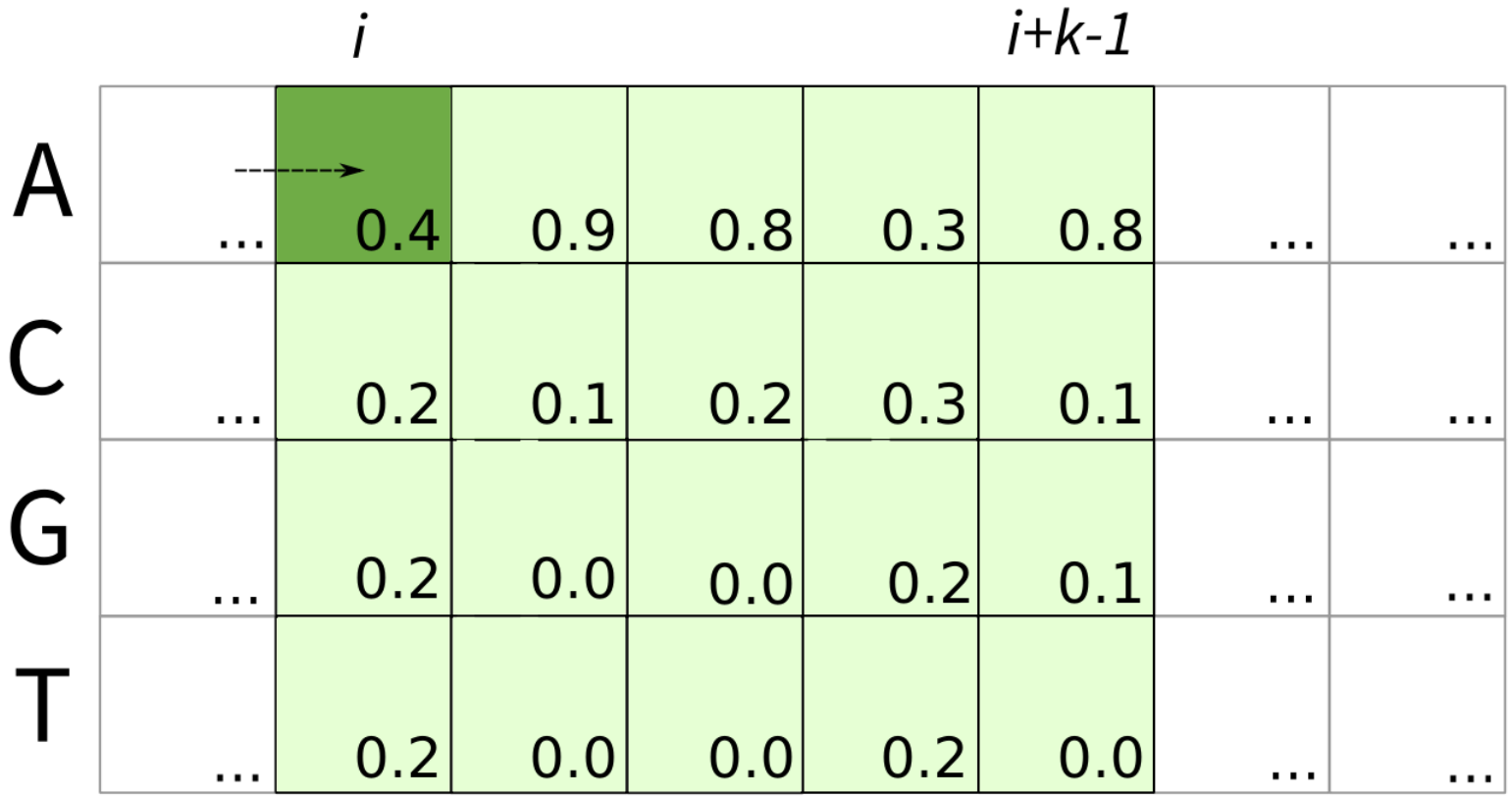
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAAAG* $\xrightarrow{\text{save}}$ `DB`

Phylo k-mer construction: Naive algorithm

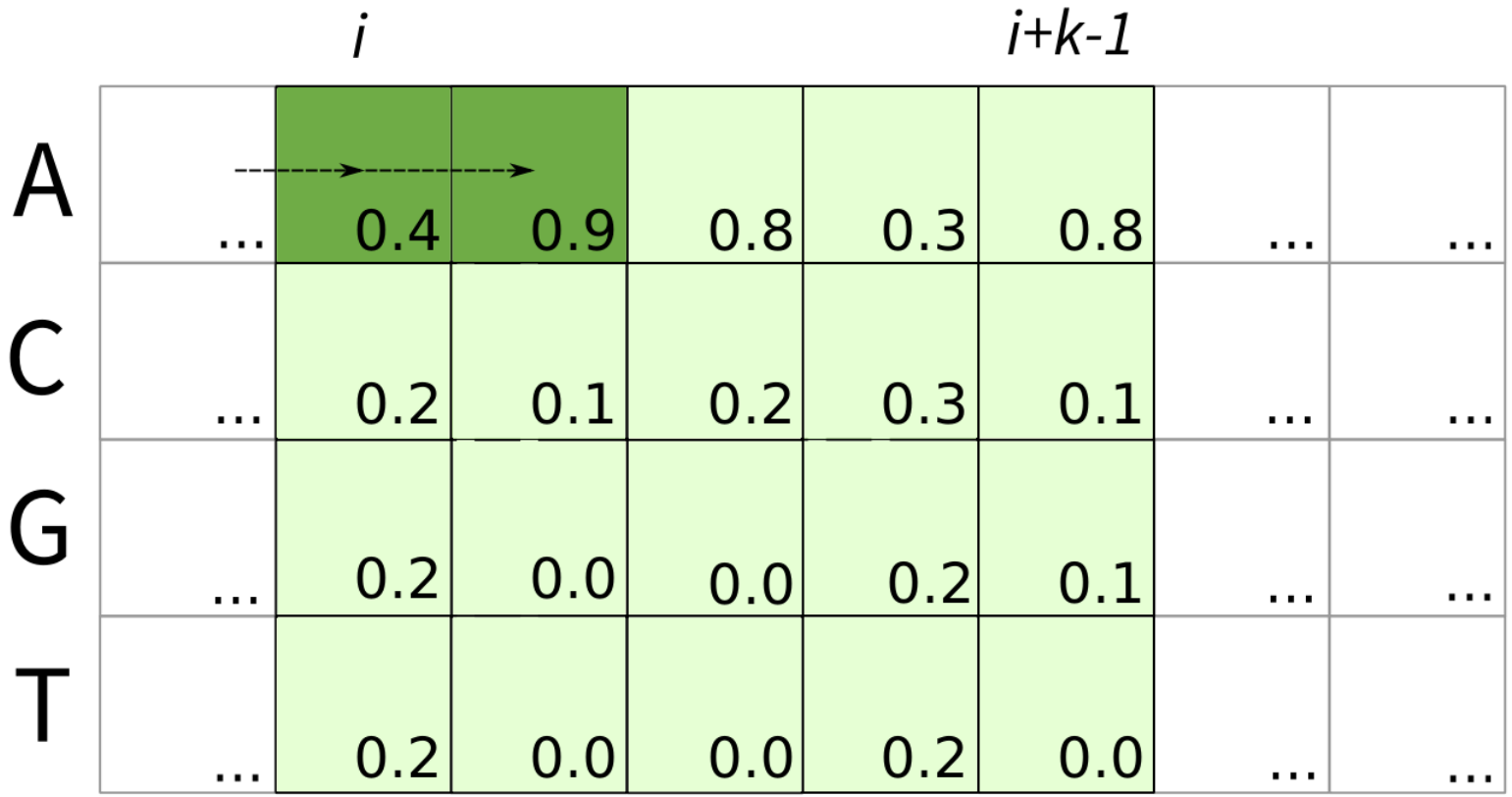
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *A*

Phylo k-mer construction: Naive algorithm

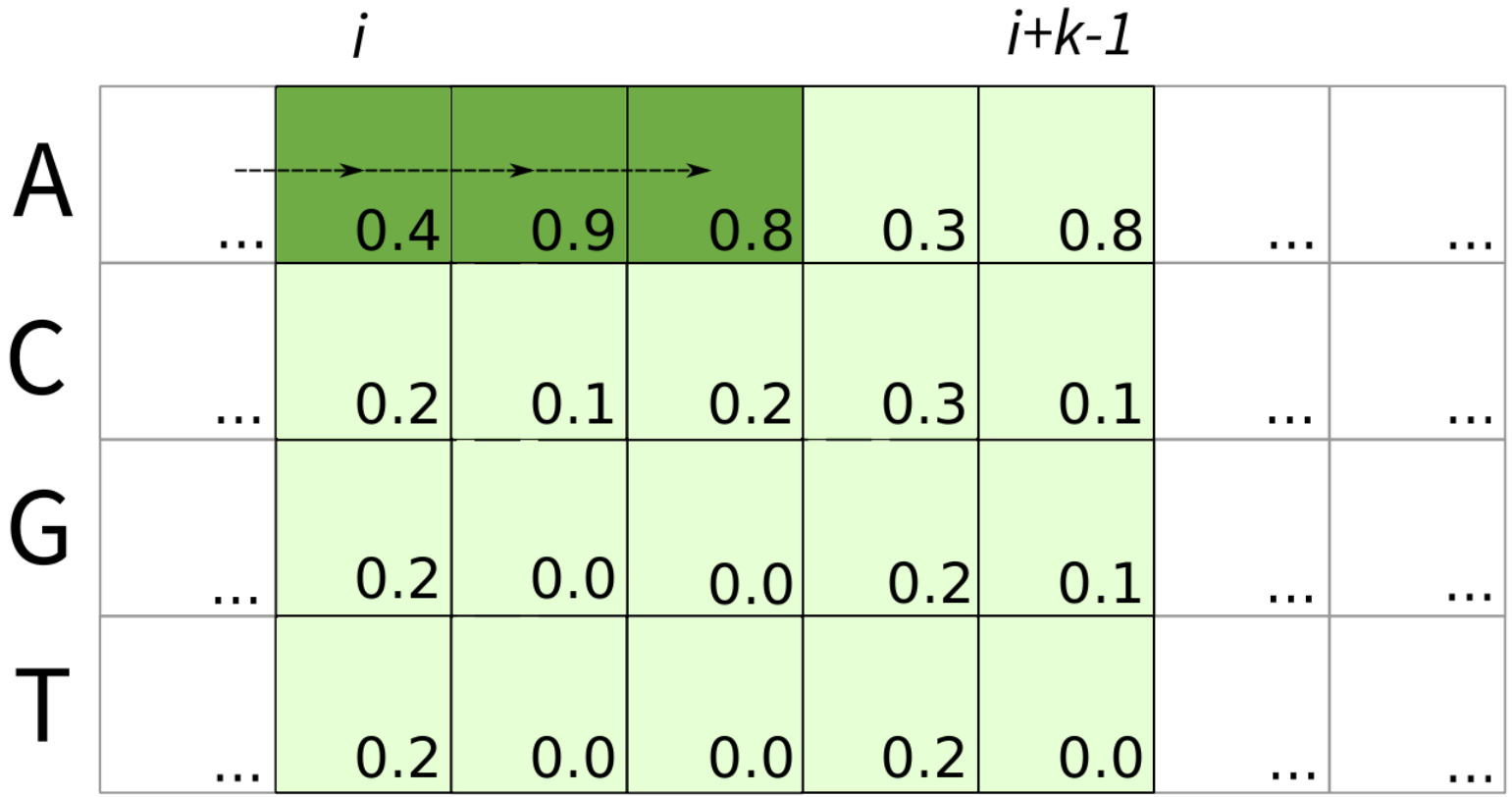
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AA*

Phylo k-mer construction: Naive algorithm

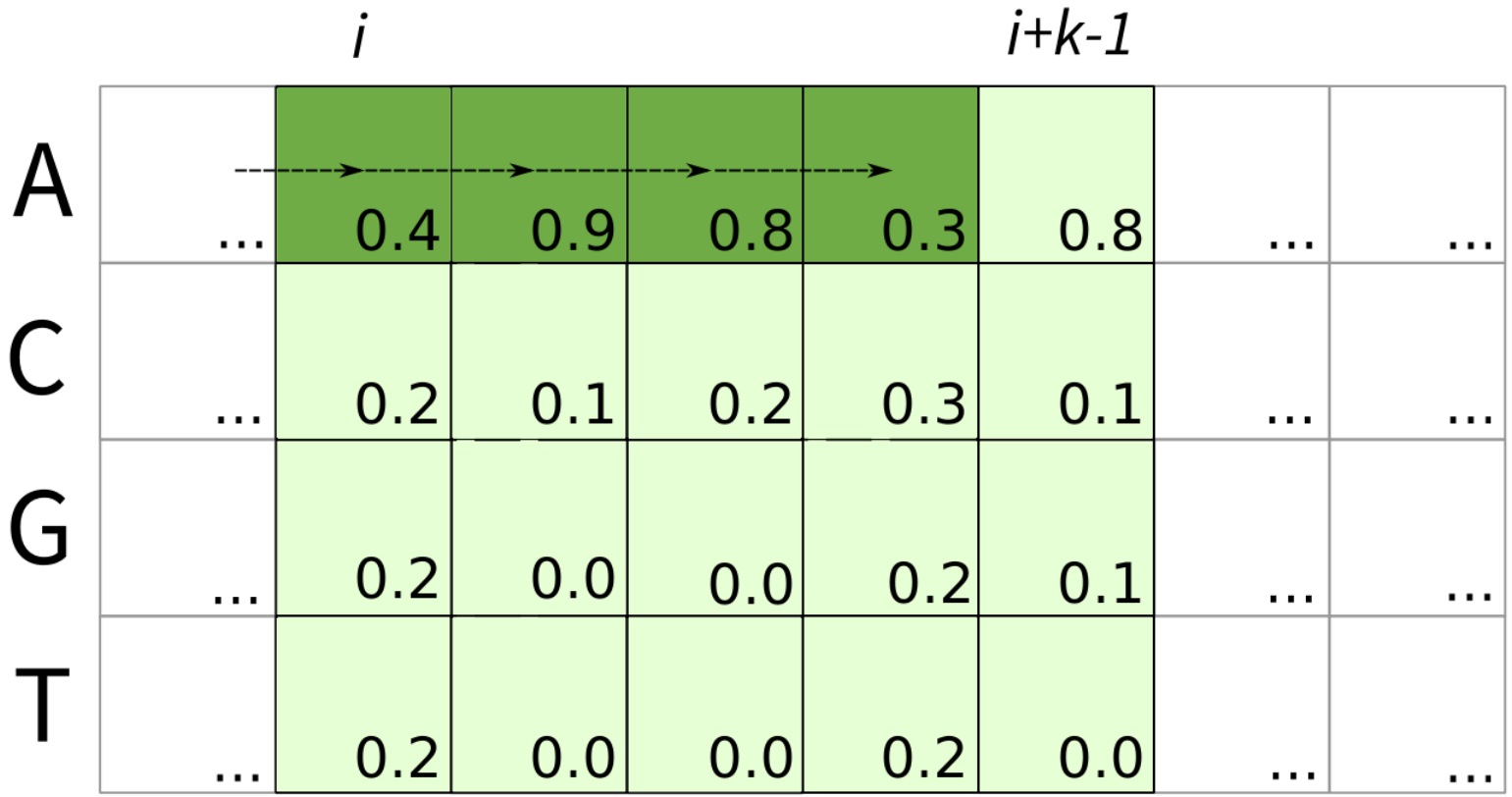
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAA*

Phylo k-mer construction: Naive algorithm

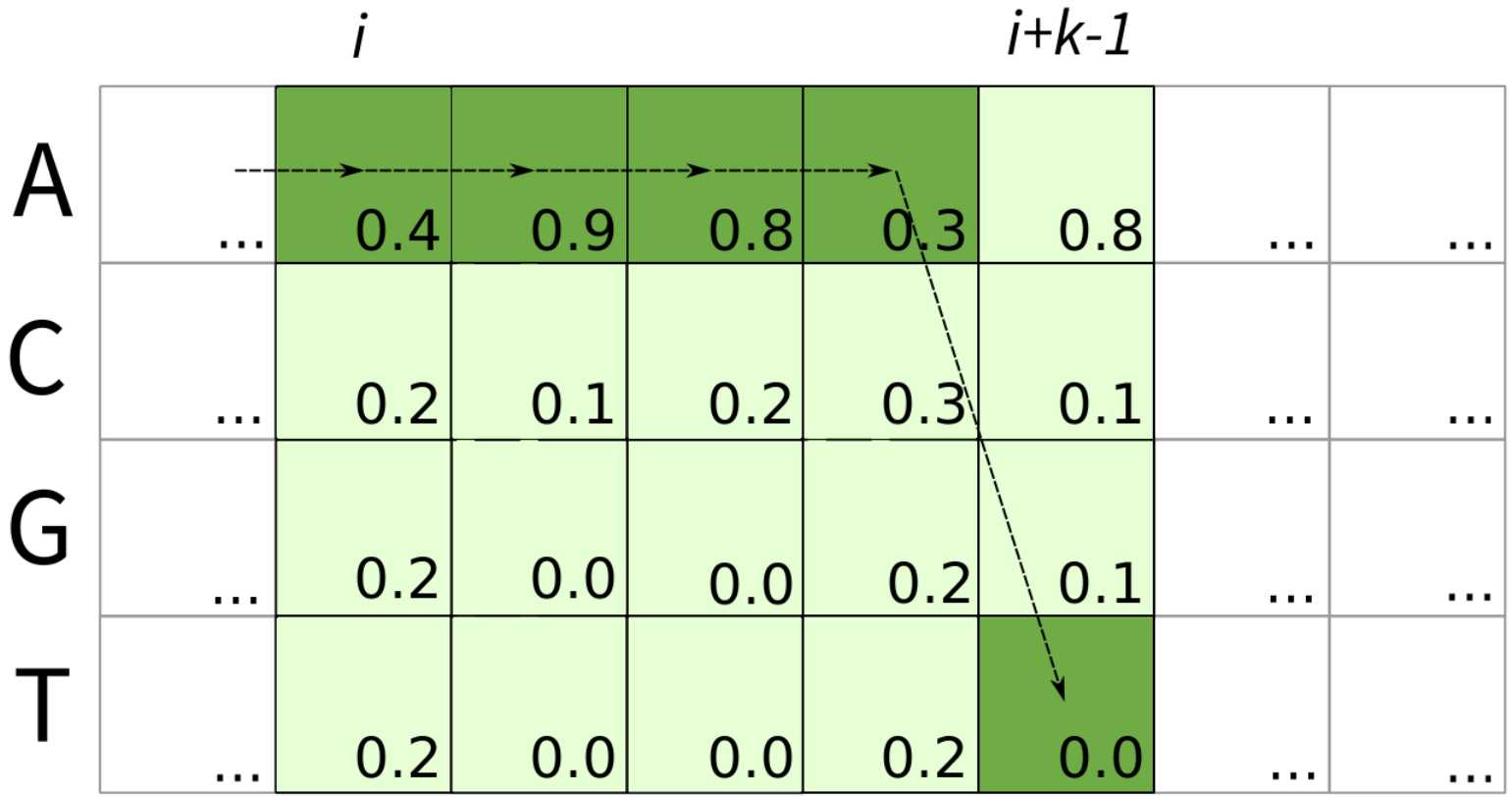
In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAAA*

Phylo k-mer construction: Naive algorithm

In the naive solution, we calculate scores of *k*-mers in the lexicographic order.



Current *l*-mer: *AAAAT* $\xrightarrow{\text{save}}$ `DB`

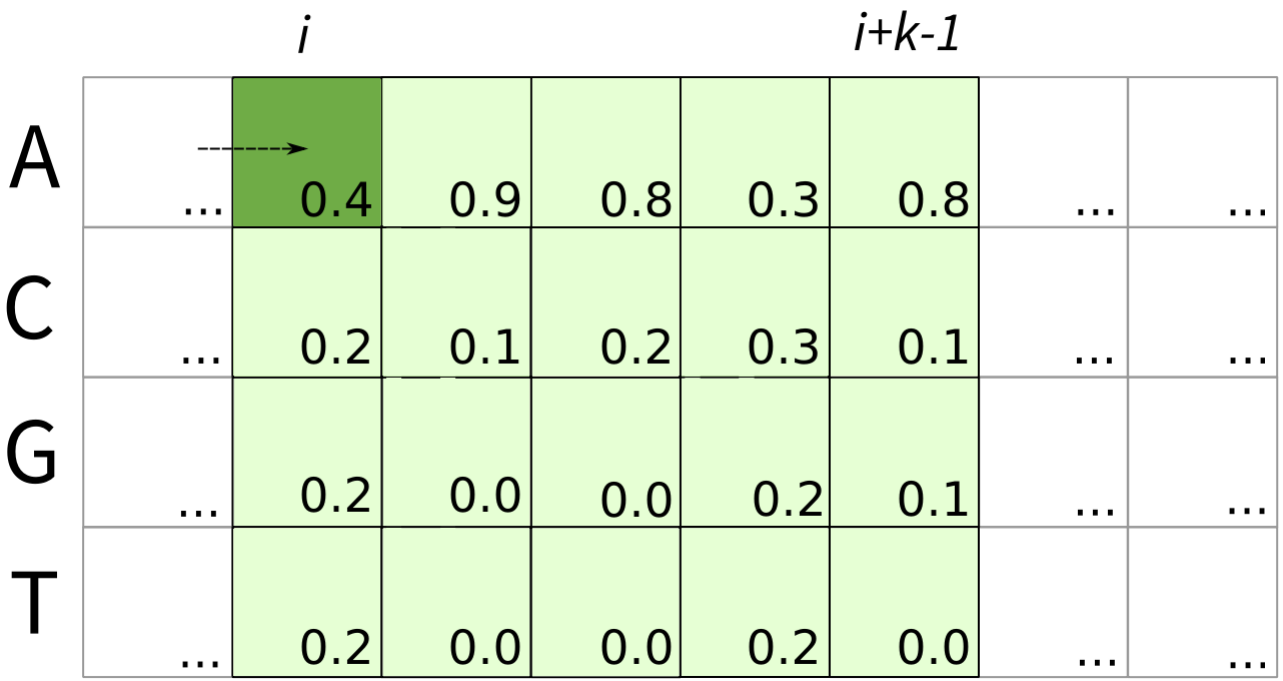
That must be frustratingly long.

That must be frustratingly long.

Can we do any better?

Phylo k-mer construction: Dynamic algorithm

Let's save the current *l*-mer's score in a stack.

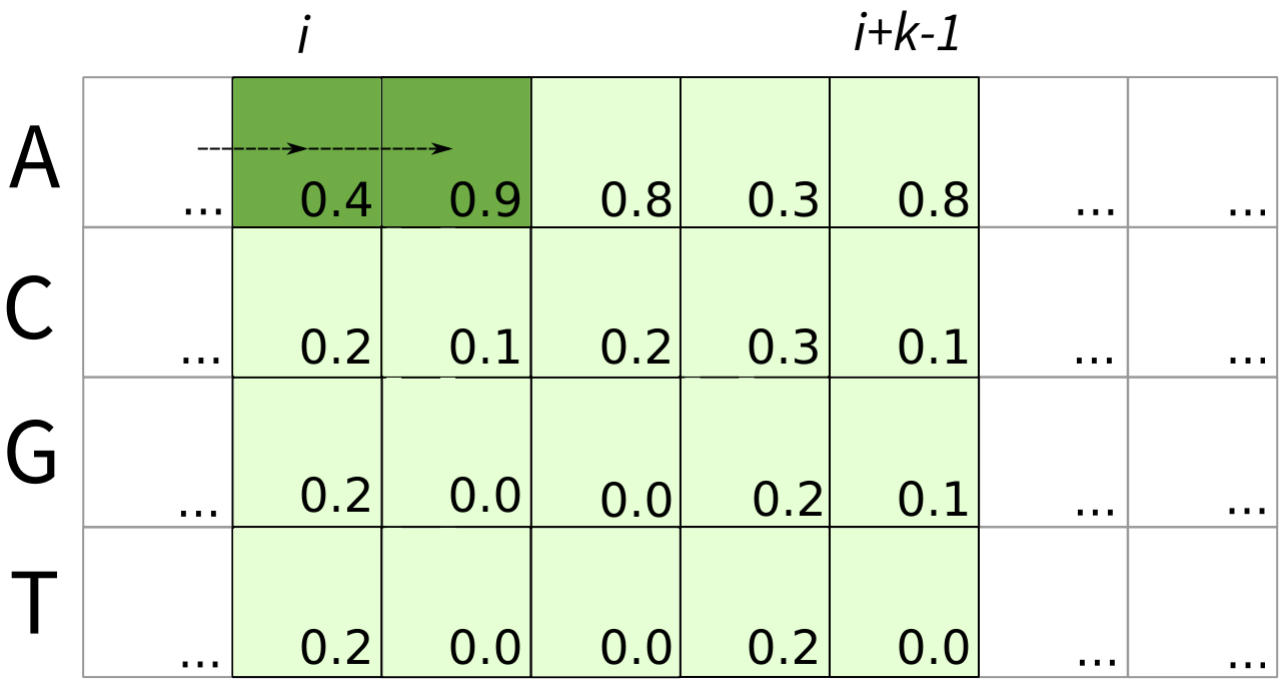


	l-mer	score
1.	A	0.4

Current *l*-mer: *A*

Phylo k-mer construction: Dynamic algorithm

Let's save the current *l*-mer's score in a stack.



	l-mer	score
1.	A	0.4
2.	AA	0.36

Current *l*-mer: *AA*

Phylo k-mer construction: Dynamic algorithm

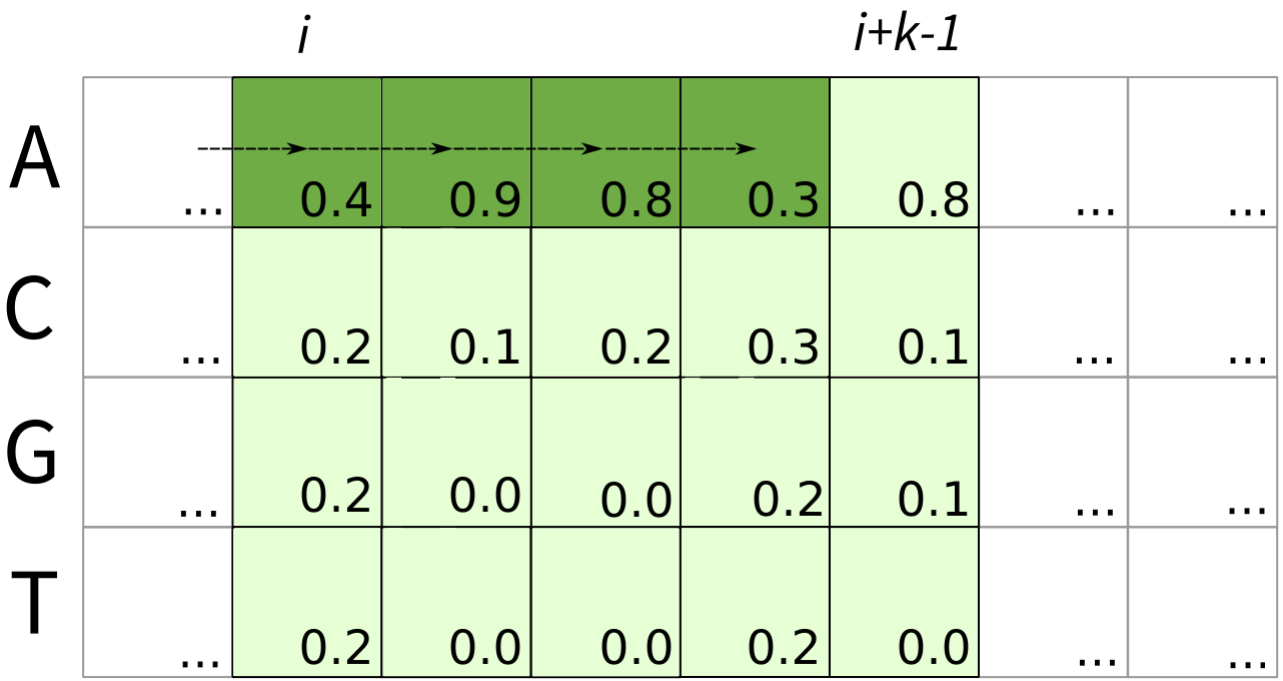
Let's save the current *l*-mer's score in a stack.

	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288

Current *l*-mer: *AAA*

Phylo k-mer construction: Dynamic algorithm

Let's save the current *l*-mer's score in a stack.



	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288
4.	AAAA	0.0864

Current *l*-mer: *AAAA*

Phylo k-mer construction: Dynamic algorithm

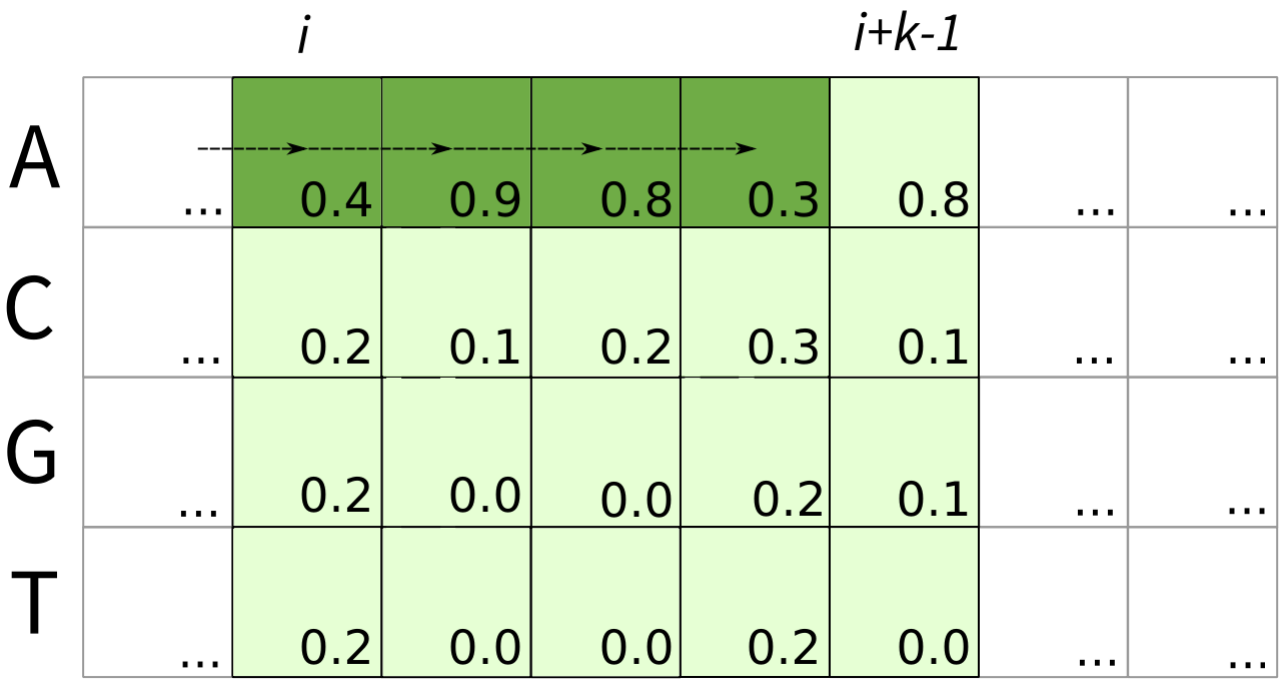
Let's save the current *l*-mer's score in a stack.

	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288
4.	AAAA	0.0864
5.	AAAAA	0.06912

Current *l*-mer: *AAAAA*

Phylo k-mer construction: Dynamic algorithm

Let's save the current *l*-mer's score in a stack.

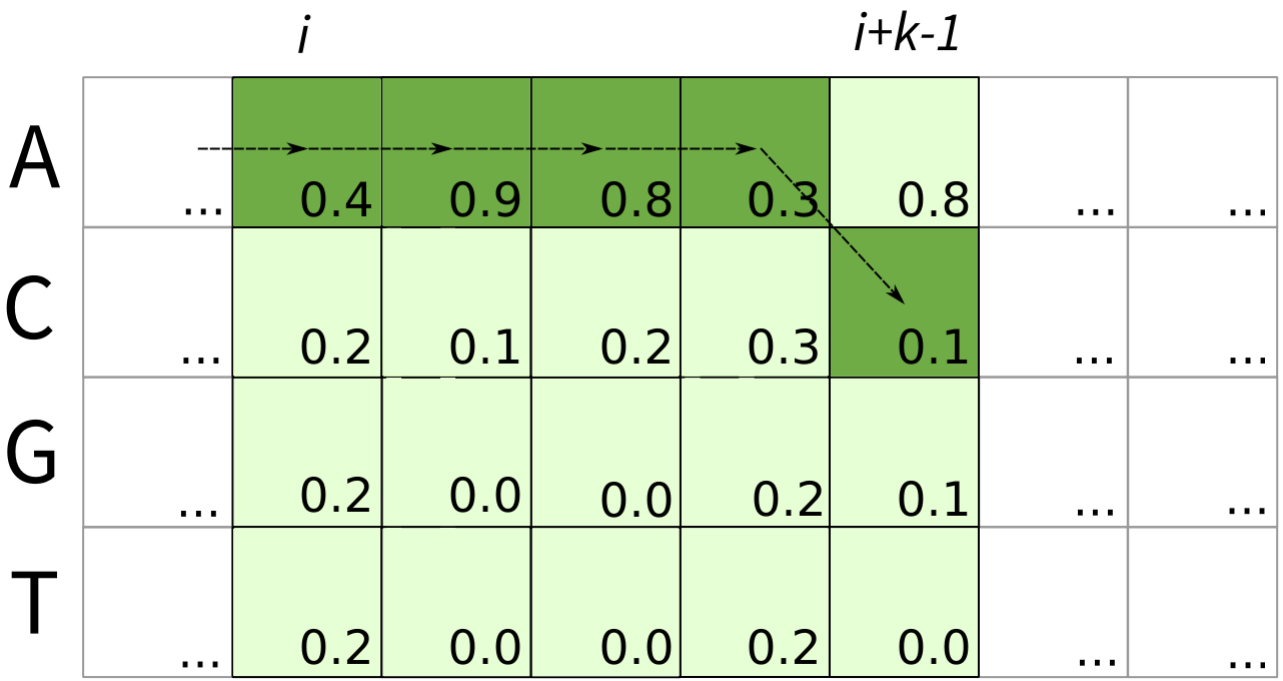


	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288
4.	AAAA	0.0864

Current *l*-mer: *AAAA*

Phylo k-mer construction: Dynamic algorithm

Let's save the current *l*-mer's score in a stack.

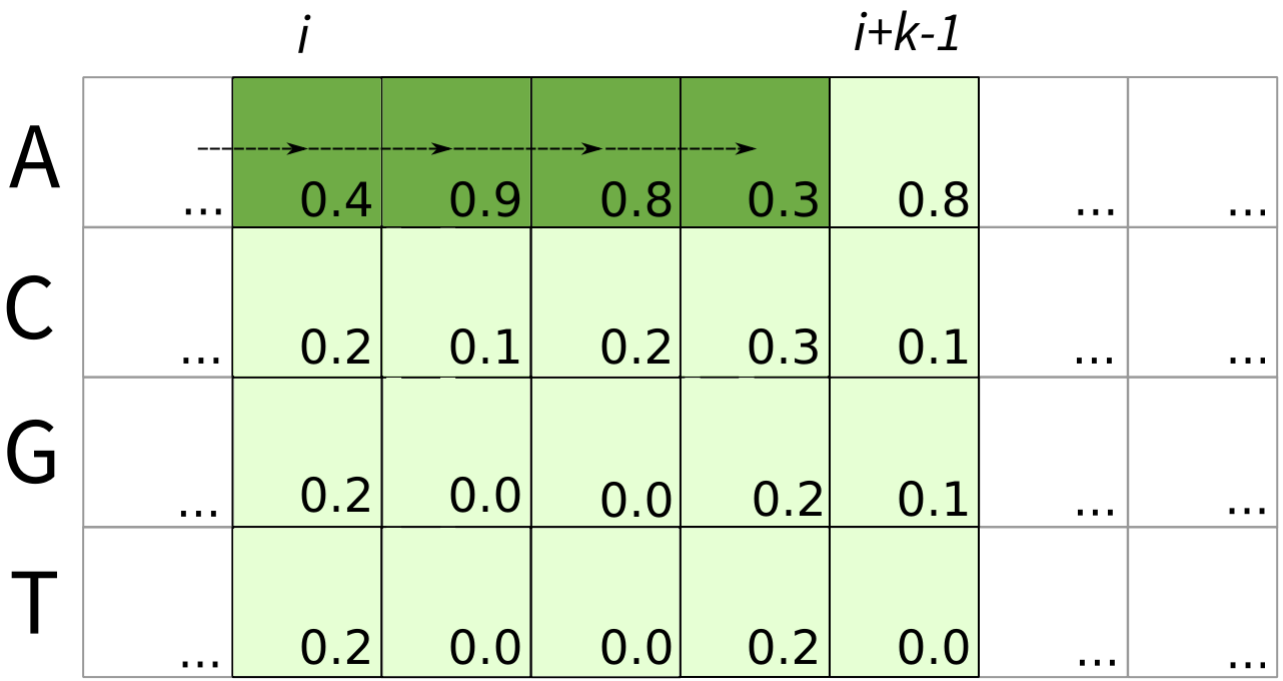


	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288
4.	AAAA	0.0864
5.	AAAAC	0.00864

Current *l*-mer: *AAAAC*

Phylo k-mer construction: Dynamic algorithm

Let's save the current *l*-mer's score in a stack.

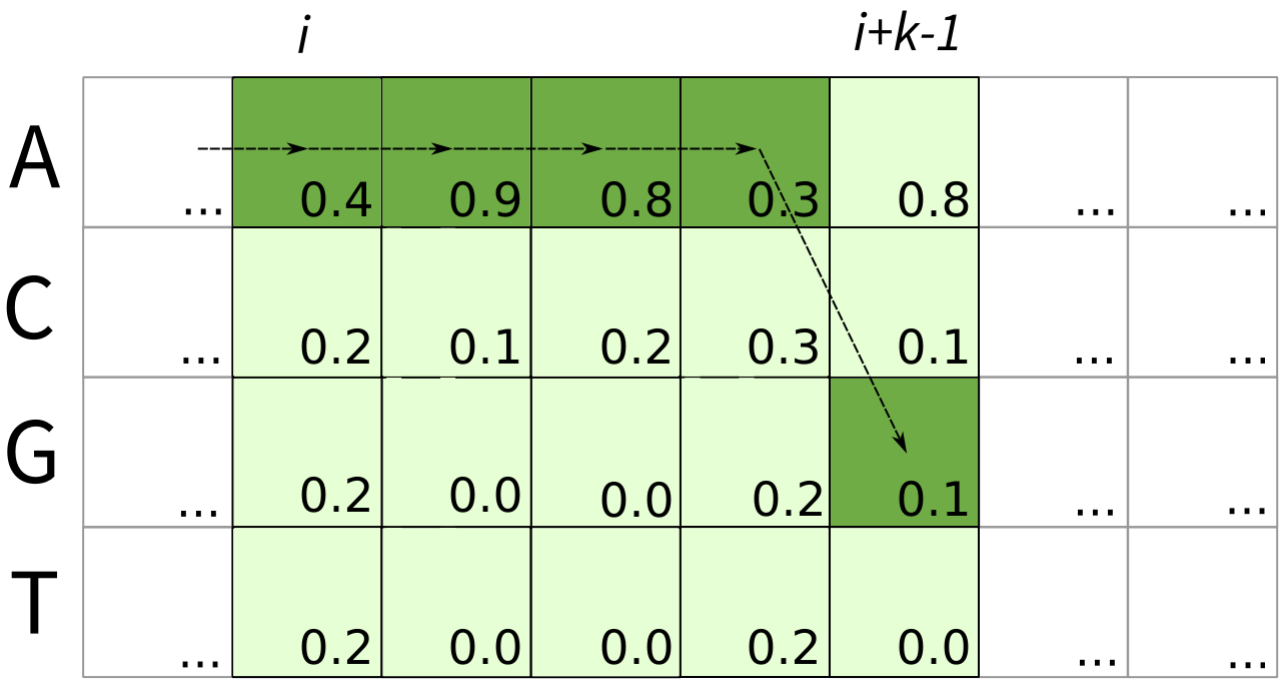


	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288
4.	AAAA	0.0864

Current *l*-mer: *AAAA*

Phylo k-mer construction: Dynamic algorithm

Let's save the current *l*-mer's score in a stack.

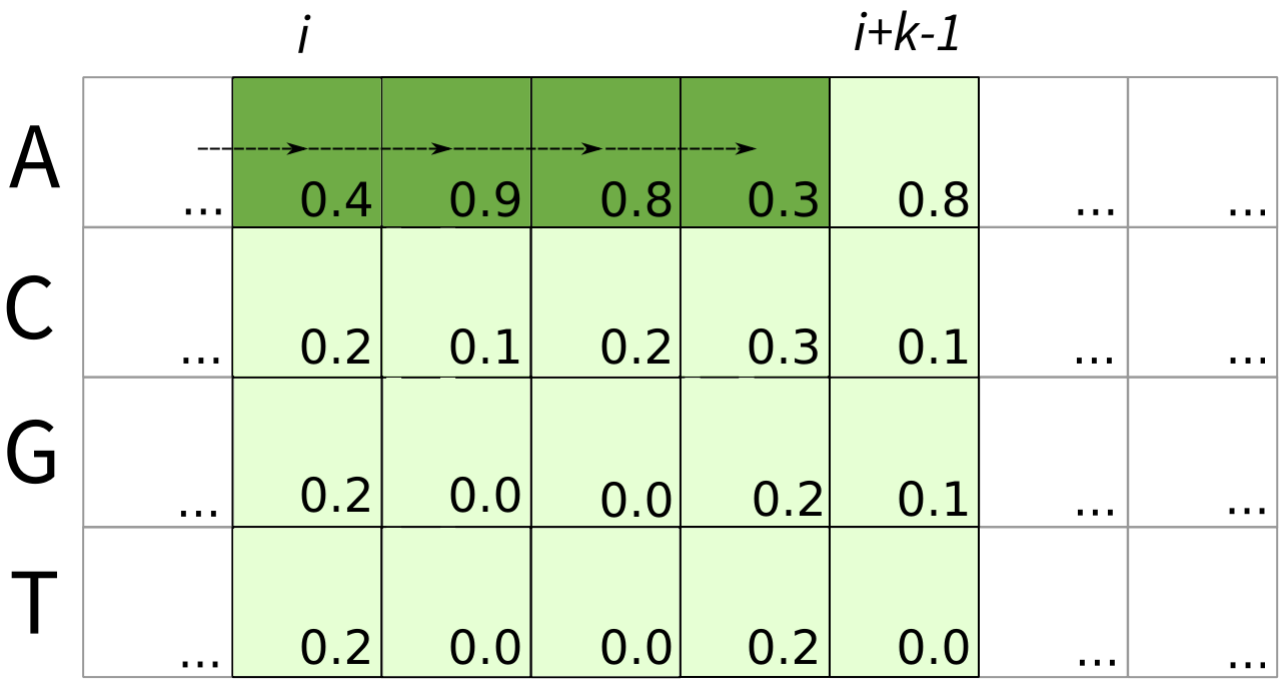


	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288
4.	AAAA	0.0864
5.	AAAAG	0.00864

Current *l*-mer: *AAAAG*

Phylo k-mer construction: Dynamic algorithm

Let's save the current *l*-mer's score in a stack.

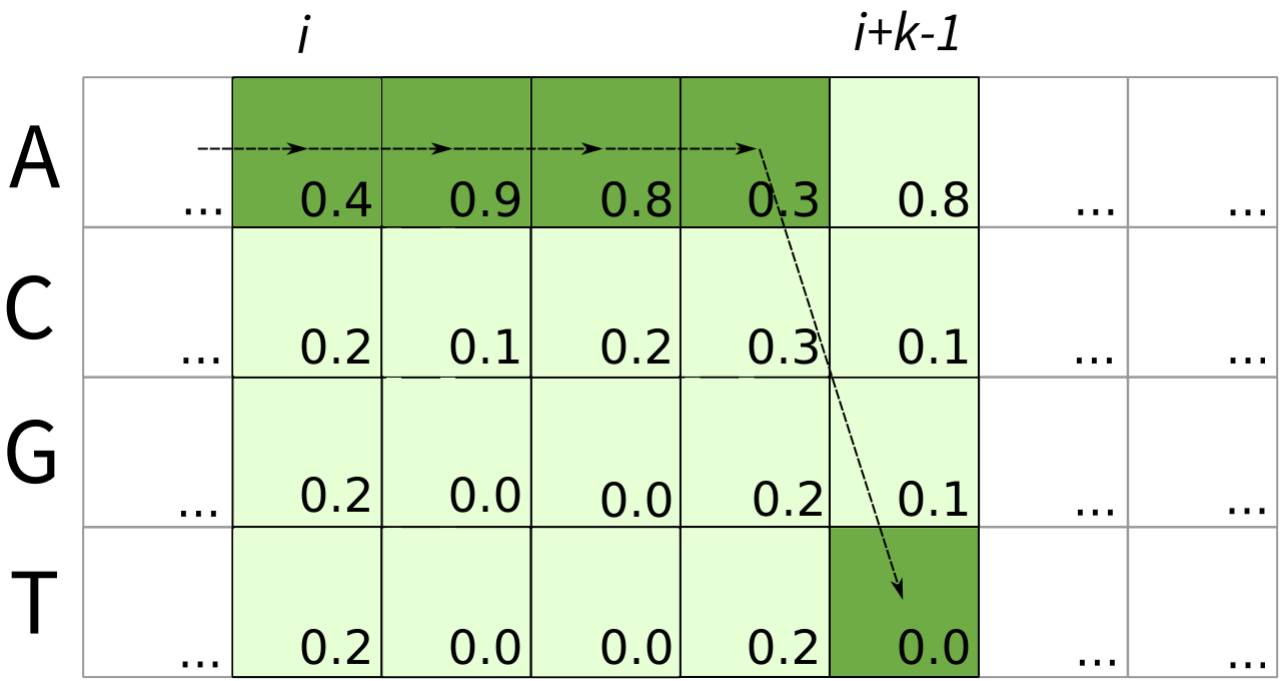


	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288
4.	AAAA	0.0864

Current *l*-mer: *AAAA*

Phylo k-mer construction: Dynamic algorithm

Let's save the current *l*-mer's score in a stack.



	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288
4.	AAAA	0.0864
5.	AAAAT	0.0

Current *l*-mer: *AAAAT*

Phylo k-mer construction: Dynamic algorithm

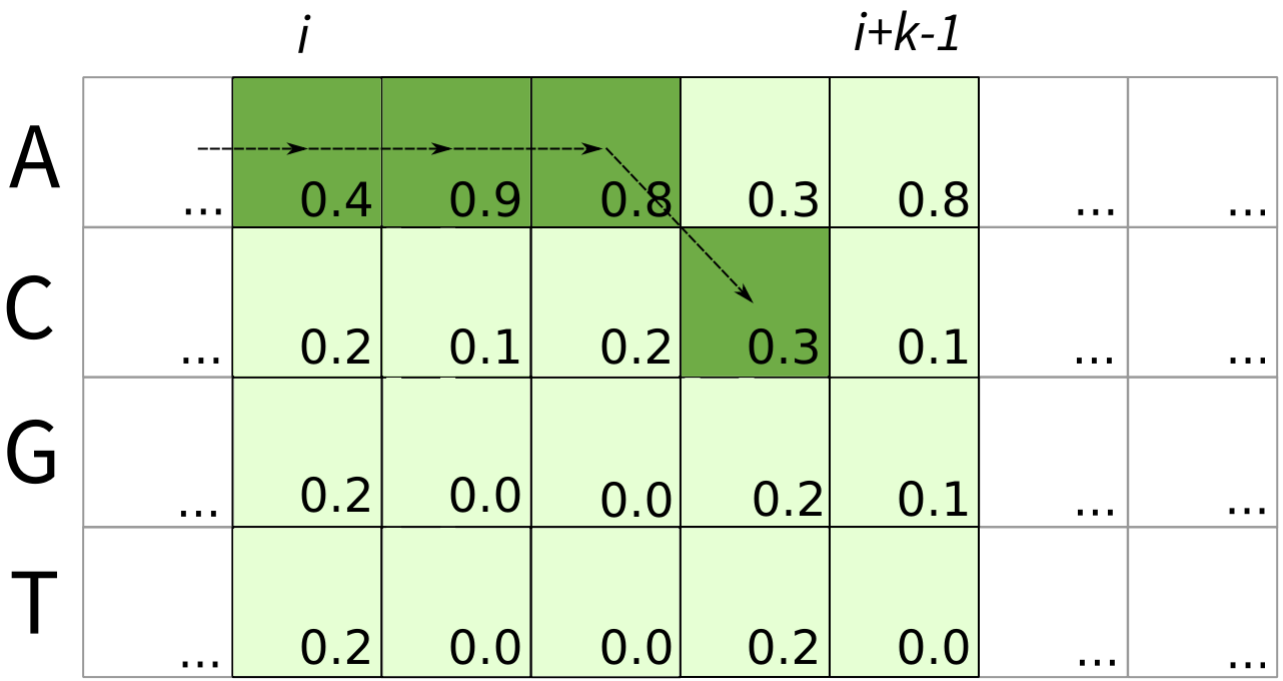
Let's save the current *l*-mer's score in a stack.

	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288

Current *l*-mer: *AAA*

Phylo k-mer construction: Dynamic algorithm

Let's save the current *l*-mer's score in a stack.



	l-mer	score
1.	A	0.4
2.	AA	0.36
3.	AAA	0.288
4.	AAAC	0.0864

Current *l*-mer: *AAAC*

Phylo k-mer construction: Score threshold heuristic

In practice, we are interested in phylo *k*-mers with high final scores. The vast majority of phylo *k*-mers have low scores.

Phylo k-mer construction: Score threshold heuristic

In practice, we are interested in phylo *k*-mers with high final scores. The vast majority of phylo *k*-mers have low scores.

Important optimization: do not store values less than a certain *threshold* value.

$$score(w) = \max(score(w), threshold)$$

threshold becomes the default value of phylo *k*-mers in the database.

Phylo k-mer construction: Score threshold heuristic

In practice, we are interested in phylo *k*-mers with high final scores. The vast majority of phylo *k*-mers have low scores.

Important optimization: do not store values less than a certain *threshold* value.

$$score(w) = \max(score(w), threshold)$$

threshold becomes the default value of phylo *k*-mers in the database.

Benefits

- **Significantly** less values to store in the database
- Allows to give up with calculating whole sets of *k*-mers

Phylo k-mer construction: Branch-and-bound algorithm

Let’s sort every column by score value.

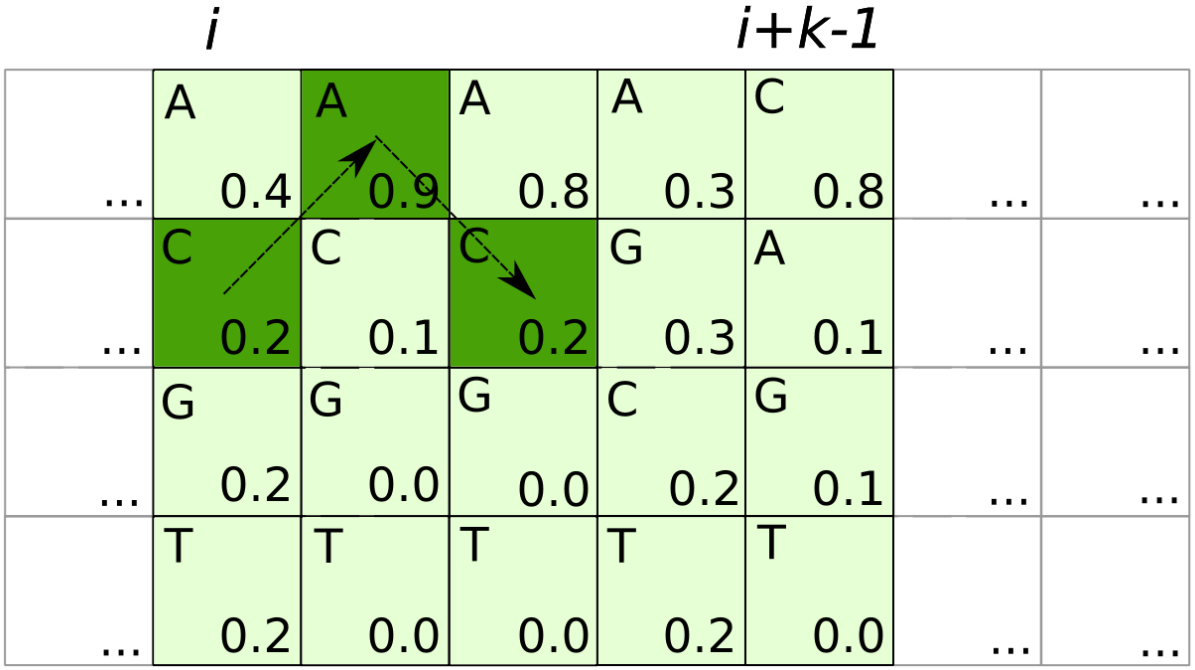
	i			*i+k-1*			
	A 0.4	A 0.9	A 0.8	A 0.3	C 0.8	...	...
	C 0.2	C 0.1	C 0.2	G 0.3	A 0.1	...	...
	G 0.2	G 0.0	G 0.0	C 0.2	G 0.1	...	...
	T 0.2	T 0.0	T 0.0	T 0.2	T 0.0	...	...

l-mer score

Phylo k-mer construction: Branch-and-bound algorithm

Let’s sort every column by score value.

The order of *k*-mers is now different, but we can cut out whole "branches" of *k*-mers if the score of a prefix is low enough.

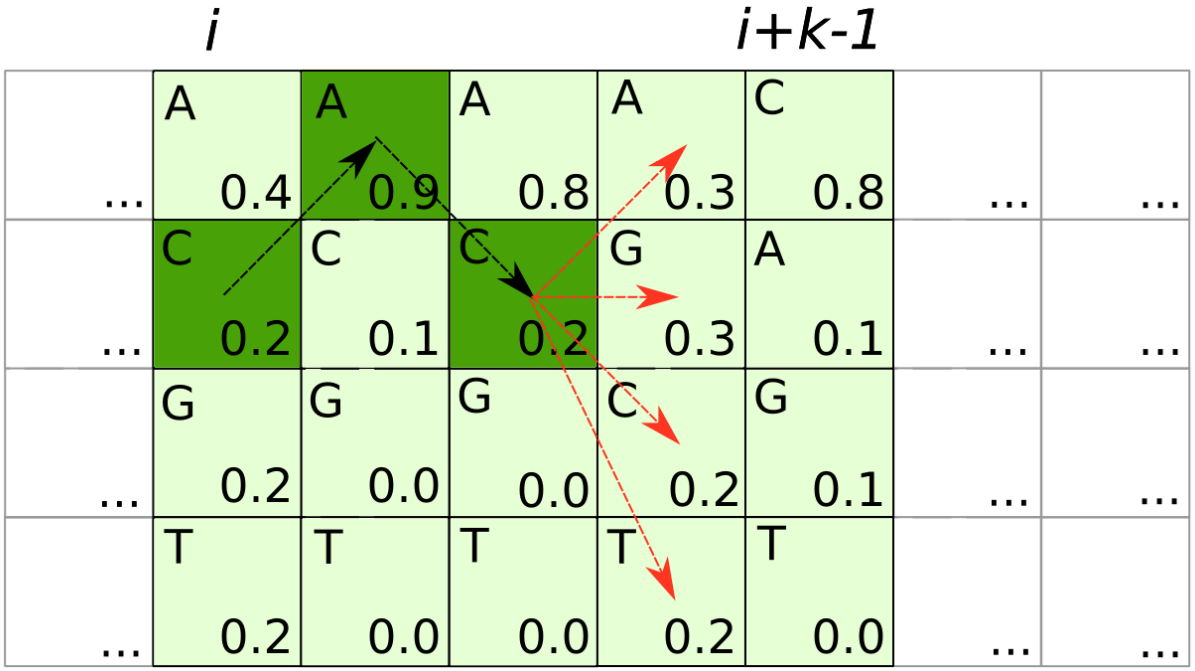


	l-mer	score
1.	C	0.2
2.	CA	0.18
3.	CAC	0.036

Phylo k-mer construction: Branch-and-bound algorithm

Let's sort every column by score value.

The order of *k*-mers is now different, but we can cut out whole "branches" of *k*-mers if the score of a prefix is low enough.



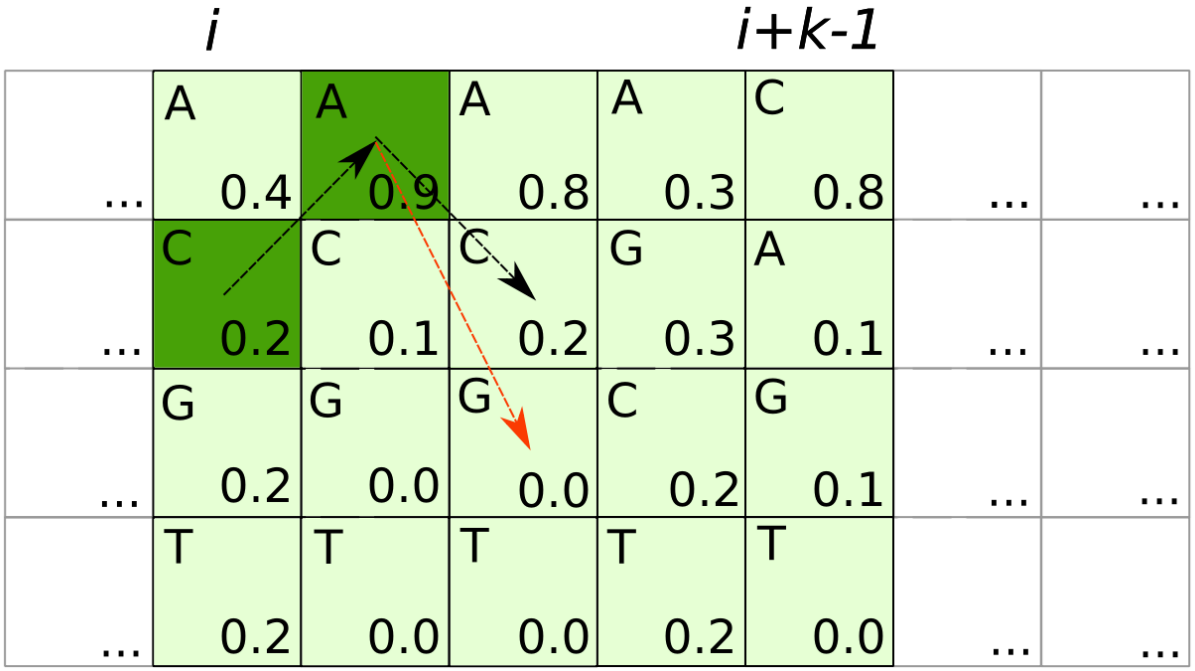
	l-mer	score
1.	C	0.2
2.	CA	0.18
3.	CAC	0.036

$s(CAC) = 0.036 \leq thr$

Phylo k-mer construction: Branch-and-bound algorithm

Let's sort every column by score value.

The order of *k*-mers is now different, but we can cut out whole "branches" of *k*-mers if the score of a prefix is low enough.



	l-mer	score
1.	C	0.2
2.	CA	0.18
3.	CAC	0.036

$s(CAC) = 0.036 \leq thr$

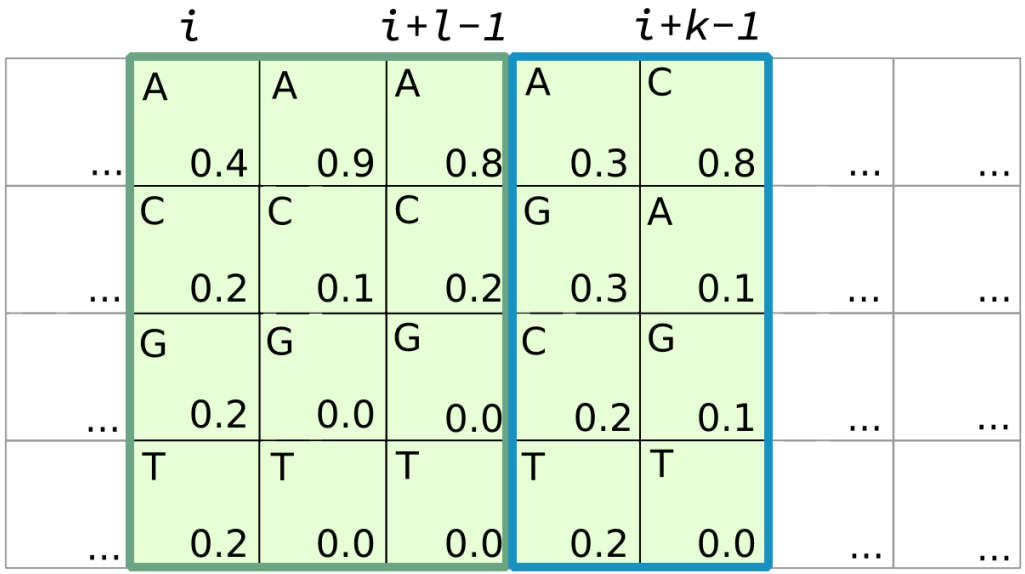
Phylo k-mer construction: Branch-and-bound algorithm

We still recalculate the same suffixes.

Phylo k-mer construction: Divide-and-conquer algorithm

The branch-and-bound algorithm reuses prefix scores, but for two k-mers with the same suffix, it would calculate the suffix score twice.

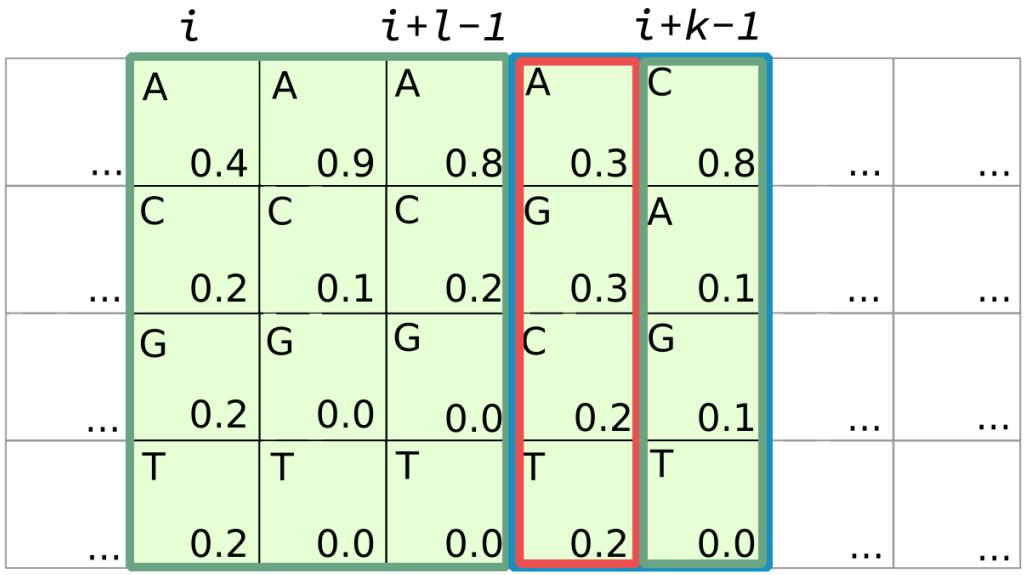
To avoid this, we split the window into subwindows: $l = \left\lceil \frac{k}{2} \right\rceil, r = \left\lfloor \frac{k}{2} \right\rfloor$



Phylo k-mer construction: Divide-and-conquer algorithm

The branch-and-bound algorithm reuses prefix scores, but for two *k*-mers with the same suffix, it would calculate the suffix score twice.

To avoid this, we split the window into subwindows: $l = \left\lceil \frac{k}{2} \right\rceil, r = \left\lfloor \frac{k}{2} \right\rfloor$



Prefixes and suffixes are constructed in subwindows recursively, and then merged into phylo *k*-mers.

Windows of size 1 already list all possible phylo 1-mers.

Phylo k-mer construction: Window merge

l-window contains prefixes, *r*-window contains suffixes. We can take the cartesian product of scores, but it will produce a lot of *k*-mers with scores $\leq$ *threshold*.

prefix	score
AAA	0.288
CAA	0.144
...	
TAA	0.144

$\times$

suffix	score
AC	0.24
GC	0.24
...	
TC	0.16

=

k-mer	score
AAAAC	0.069
AAAGC	0.069
CAAAC	0.034
CAAGC	0.034
...	
TAATC	0.023

Instead, we sort suffixes, and calculate the *residual threshold* `th(prefix)` for every prefix. Suffix is taken to the product only if `score(suffix)` $\geq$ `th(prefix)`.

Phylo k-mer construction: Window merge

l-window contains prefixes, *r*-window contains suffixes. We can take the cartesian product of scores, but it will produce a lot of *k*-mers with scores $\leq$ *threshold*.

prefix	score
AAA	0.288
CAA	0.144
...	
TAA	0.144

$\times$

suffix	score
AC	0.24
GC	0.24
...	
TC	0.16

=

k-mer	score
AAAAC	0.069
AAAGC	0.069
CAAAC	0.034
CAAGC	0.034
...	
TAATC	0.023

Instead, we sort suffixes, and calculate the *residual threshold* `th(prefix)` for every prefix. Suffix is taken to the product only if `score(suffix)` $\geq$ `th(prefix)`.

Phylo k-mer construction: Window merge

l-window contains prefixes, *r*-window contains suffixes. We can take the cartesian product of scores, but it will produce a lot of *k*-mers with scores $\leq$ *threshold*.

prefix	score
AAA	0.288
CAA	0.144
...	
TAA	0.144

$\times$

suffix	score
AC	0.24
GC	0.24
...	
TC	0.16

=

k-mer	score
AAAAC	0.069
AAAGC	0.069
CAAAC	0.034
CAAGC	0.034
...	
TAATC	0.023

Instead, we sort suffixes, and calculate the *residual threshold* `th(prefix)` for every prefix. Suffix is taken to the product only if `score(suffix)` $\geq$ `th(prefix)`.

Future Other applications

- RAPPAS2, phylogenetic placement
- CLAPPAS, protein family classification
- Other applications?