



UNIVERSITÄT
DES
SAARLANDES



ZBI

ZENTRUM FÜR
BIOINFORMATIK

Genome-Wide DNA Read Mapping (or DNA Database Search)

Algorithms for Sequence Analysis

Sven Rahmann

Summer 2021

Overview

Previously: Error Tolerant Pattern Matching

- Many online algorithms
- Index-based: **Error-tolerant backward search (FM-index + NFA-like table)**

Overview

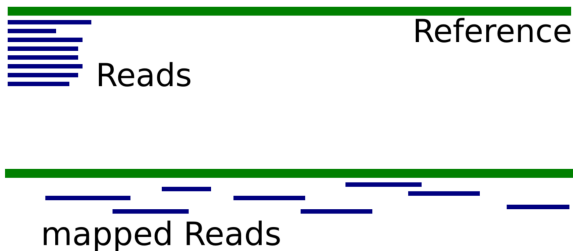
Previously: Error Tolerant Pattern Matching

- Many online algorithms
- Index-based: **Error-tolerant backward search (FM-index + NFA-like table)**

Today's Lecture: Genome-Wide DNA Sequence Search

- Read Mappers: bwa-sw, bwa-mem, bowtie2
- Seed-and-extend principle (anchors)

Read Mapping and DNA Database Search Problems



Read Mapping

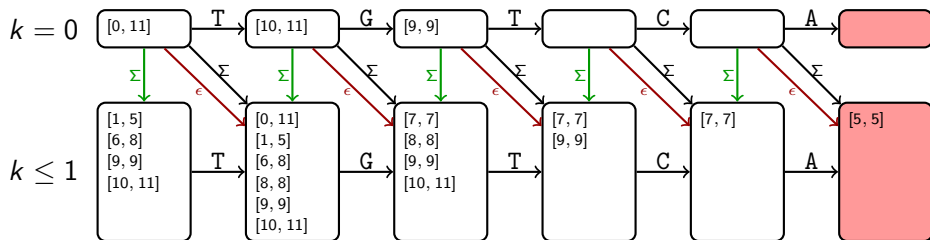
Find (possible) origin(s) of (short) piece of DNA (“read”) within genome;
Find (possible) origin(s) of parts of (long) piece of DNA within genome.

DNA Database Search

Find (parts of) DNA query in huge DNA database (NCBI GenBank)

Reminder: Error Tolerant Backward Search

$T = \text{AAAACGTACCT\$}$, $P = \text{ACTGT}$, $\Sigma = \{A, C, G, T\}$:



- Green edges: insertions
- Red edges (ϵ): deletions
- Black edges: matches (horizontal) and mismatches (diagonal)
- Note: numbers for illustration only, not necessarily correct.

Alternative: Branch on FM Index (no storage in states)

(Example: cta, Hamming distance 1.)

	F	L
0	\$ ctatata	t
1	a t\$ctata	t
2	a tat\$cta	t
3	a tatat\$c	t
4	c tatatat	\$
5	t \$ctatat	a
6	t at\$ctat	a
7	t atat\$ct	a
8	t atatat\$	c

Alternative: Branch on FM Index (no storage in states)

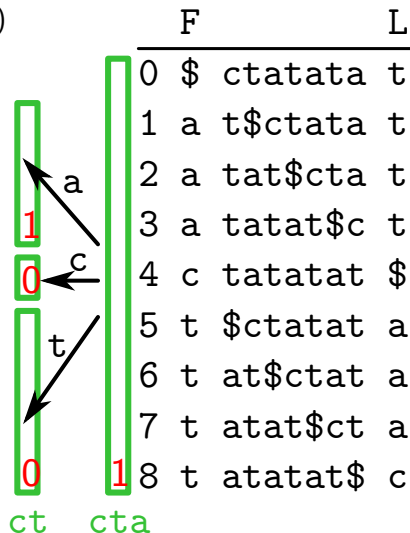
(Example: cta, Hamming distance 1.)

	F	L
0	\$ ctatata	t
1	a t\$ctata	t
2	a tat\$cta	t
3	a tatat\$c	t
4	c tatatat	\$
5	t \$ctatat	a
6	t at\$ctat	a
7	t atat\$ct	a
1	8 t atatat\$	c

cta

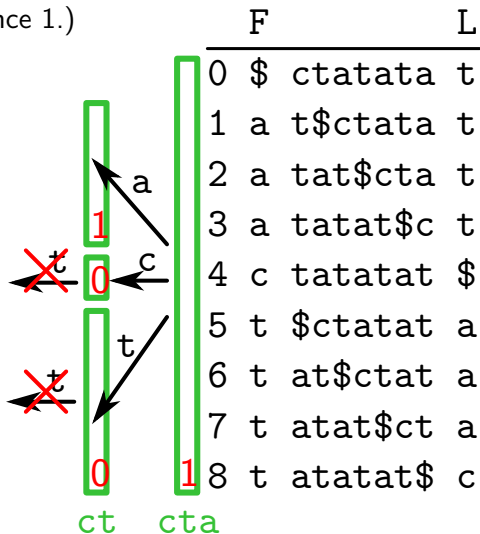
Alternative: Branch on FM Index (no storage in states)

(Example: cta, Hamming distance 1.)



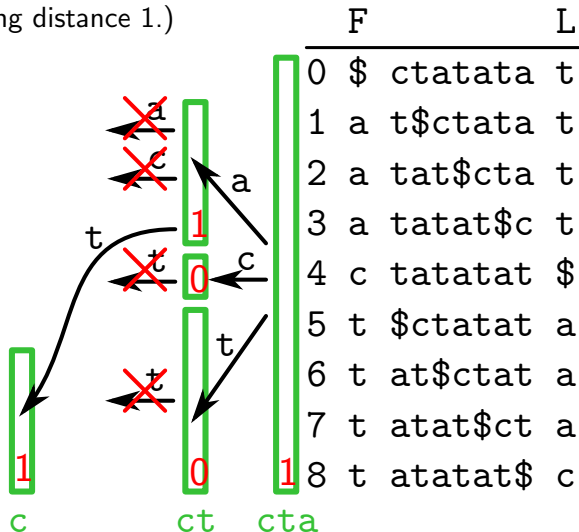
Alternative: Branch on FM Index (no storage in states)

(Example: cta, Hamming distance 1.)



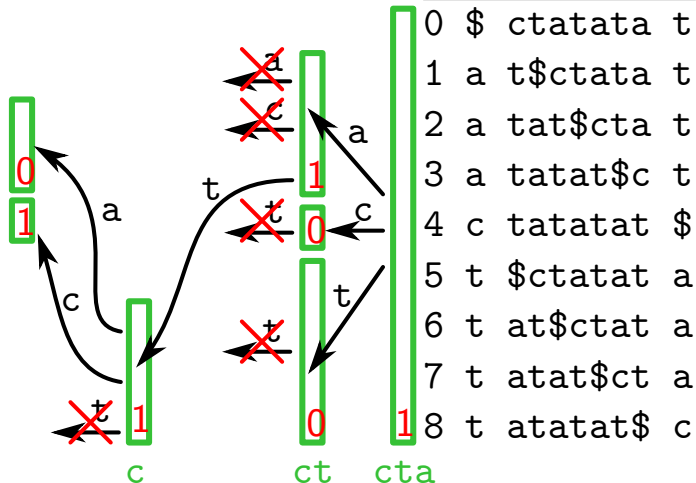
Alternative: Branch on FM Index (no storage in states)

(Example: cta, Hamming distance 1.)



Alternative: Branch on FM Index (no storage in states)

(Example: cta, Hamming distance 1.)



Running time of Approximate Backward Search

(with NFA states or by branching on-the-fly)

Worst case

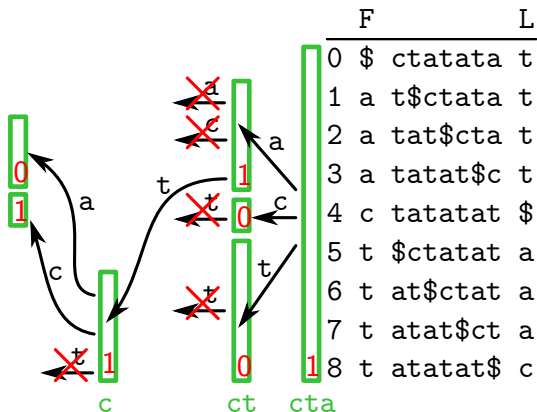
- All strings in edit/Hamming neighborhood are enumerated, i.e. the set

$$\{P' \in \Sigma^* \mid d(P, P') \leq k\},$$

where $P \in \Sigma^m$ is the pattern and k a distance threshold.

- \Rightarrow Running time is **exponential** in k in the worst case.
- In practice, it might be less bad, depending on the string that is searched.

Getting the Actual Positions of Matches



Question

Given: interval on BWT

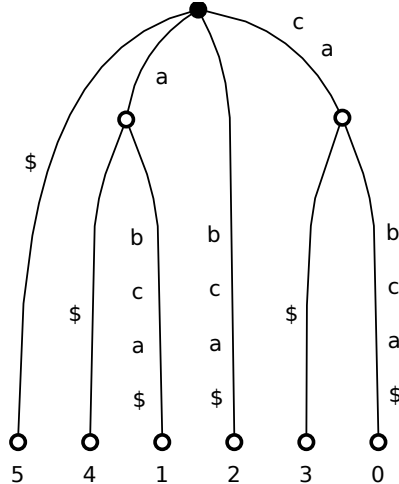
Sought: position of matches in original string

Answer

Use **(sparse) suffix array** pos

Alternative: Branch on Suffix Tree (Forward Search)

Examples (Hamming distance 1 for simplicity): (1) acc, (2) bab



Approaches in Practice

- Suffix tree (forward search) not used: memory footprint too large
- NFA states not used (memory overhead)

Approaches in Practice

- Suffix tree (forward search) not used: memory footprint too large
- NFA states not used (memory overhead)
- **bwa-sw**: approximate backward search with branching on-the-fly, intervals examined using depth-first search.
- **bwa-sw** starts at several locations in the read (local matches), allows for smaller edit distance thresholds
- **bwa-sw** collects a large number of intervals first, then performs batch lookup in suffix array

Literature: **bwa-sw**

Heng Li & Richard Durbin:

Fast and accurate short read alignment with Burrows–Wheeler transform.

Bioinformatics 25(14):1754–1760 (2009).

Seed and Extend Idea for Filter-based Approaches

Motivation

Branching on errors is computationally expensive

- Running time exponential in the number of errors
- Exact matches are inexpensive to compute
- **Idea:** Find exact matches
 - of fixed length q (q -grams) or k (k -mers)
 - of maximal length (MEMs, maximal exact matches) using FM-index
- Then perform anchored local alignment around the match (seed).

Motivation

Branching on errors is computationally expensive

- Running time exponential in the number of errors
- Exact matches are inexpensive to compute
- **Idea:** Find exact matches
 - of fixed length q (q -grams) or k (k -mers)
 - of maximal length (MEMs, maximal exact matches) using FM-index
- Then perform anchored local alignment around the match (seed).

Literature: bwa-mem

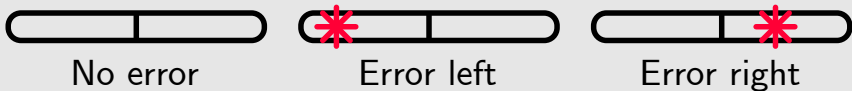
Heng Li:

Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM
arXiv:1303.3997 [q-bio.GN]

Pigeon Hole Principle, Seed and Extend

Example

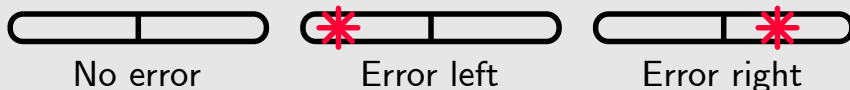
When we allow a read to contain up to 1 error, the following scenarios are possible:



Pigeon Hole Principle, Seed and Extend

Example

When we allow a read to contain up to 1 error, the following scenarios are possible:



More than one error?

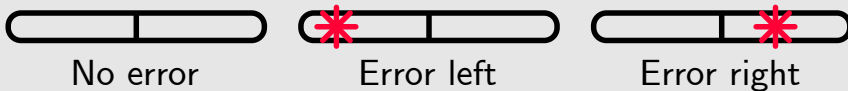
Pigeon hole principle holds for $d \geq 1$ errors:

- Cut pattern into $d + 1$ parts
- Then, at least one part without error has to exist.

Pigeon Hole Principle, Seed and Extend

Example

When we allow a read to contain up to 1 error, the following scenarios are possible:



More than one error?

Pigeon hole principle holds for $d \geq 1$ errors:

- Cut pattern into $d + 1$ parts
- Then, at least one part without error has to exist.

Seed and Extend

- **Seed:** Find origins of exact match
- **Extend:** Continue search **allowing errors** around exact match

Bowtie

Implementation of seed-extend in bowtie

- Use two indexes: for searching forward/backward, respectively
- **Seed:** search for left/right half **without errors** using FM index
- **Extend:** continue search using branching in FM index

Bowtie

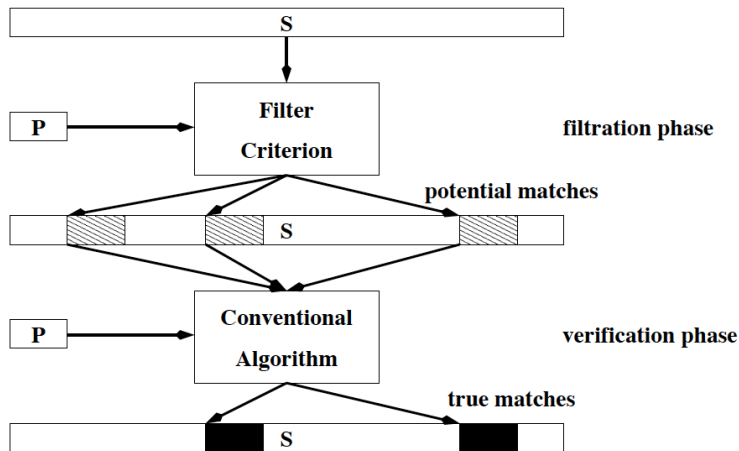
Implementation of seed-extend in bowtie

- Use two indexes: for searching forward/backward, respectively
- **Seed:** search for left/right half **without errors** using FM index
- **Extend:** continue search using branching in FM index

Variants used in other read mappers or search tools

- seed phase using index, extend phase using alignment
- multiple seeds, then alignment

Abstraction: Filter-based approximate search



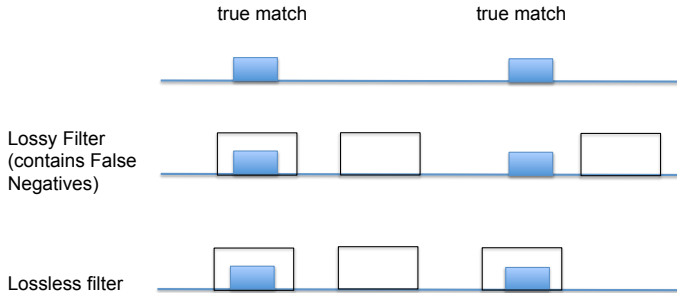
[Stefan Burkhardt, Ph.D. Thesis 2002]

Filter-based Search

A filter restricts the expensive verification to promising regions of the target sequence.

- A **lossy** filter may miss true matches in the target.
- A **lossless** filter contains all true matches.

A good filter always balances sensitivity and speed.



Definitions

q -gram

A **q -gram** of a string s is a substring of length q of s .

Matching Problem

Given a reference text T and a query string s , the **approximate matching problem with d differences and window length w** consists of finding a pair of substrings $(s[i \dots i + w - 1], t)$ such that

- 1 $s[i, i + w - 1]$ is a length- w substring of s , and t is a substring of T ,
- 2 $d_{\text{edit}}(s[i \dots i + w - 1], t) \leq d$, i.e., edit distance is at most d .

Q-gram lemma

Lemma

Let an occurrence of $s[i, i + w - 1]$ with at most d differences end at position j in T . Then at least $w + 1 - (d + 1)q$ q -grams of $s[i, i + w - 1]$ occur in $T[j - w + 1, j]$.

Q-gram lemma

Lemma

Let an occurrence of $s[i, i + w - 1]$ with at most d differences end at position j in T . Then at least $w + 1 - (d + 1)q$ q -grams of $s[i, i + w - 1]$ occur in $T[j - w + 1, j]$.

Proof

- each error destroys at most q q -grams (Y positions below)
- the last $q - 1$ positions have no q -grams (X positions below)

intact q -grams: $w - dq - (q - 1) = w + 1 - (d + 1)q$

- Example: $q=3$ (valid positions of 3-grams are M below)

target	ATTGACAC
query	ATT C ACAC
3-grams	MYYYYMMXX

QUASAR algorithm

Task: Find approximate matches of s in T

Seed-and-extend based on q -gram lemma

- 1 Pre-compute the threshold α for a window of length w using the q -gram lemma.
- 2 Partition T into blocks (larger than window length w)
- 3 Count all q -grams in $s[1 \dots w]$ for each block along T
- 4 Each block that contains an approximate match has a counter of at least α .
(The reverse is not true. We may have false positive blocks)
- 5 Advance the window in s from $s[i \dots j]$ to $s[i + 1 \dots j + 1]$

QUASAR - counting q-grams

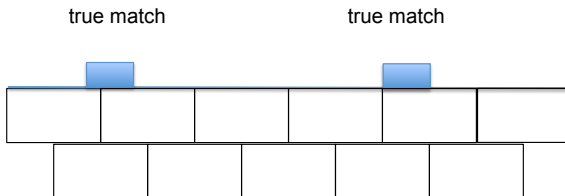
In order to count the q-gram matches in a window, we need an efficient data structure.

In order to count the q-gram matches in a window, we need an efficient data structure.
The **q-gram index**:

- 1 Construct the suffix array pos of T
- 2 Create a table of size $|\Sigma|^q$ that maps a q -gram to its starting rank in pos, to allow constant time lookup.
- 3 q -grams are integer-encoded as numbers $0 \dots |\Sigma|^q - 1$,
e.g. DNA: TTGCCA = $(332110)_4 = 3988$ (base-4 number)

QUASAR - Blockwise counting

- Keeping a counter for each possible window of length w of the text would lead to a huge array of size $\mathcal{O}(|T|)$
- To save space, we define two arrays of blocks of size $b \geq 2w$. The first block array is shifted by $b/2$ positions against the second.



QUASAR - Efficient counting and shifting

When the query window $s[i \dots j]$ is moved to $s[i + 1 \dots j + 1]$, the only difference for the block counters are two q -grams, one that leaves (blue) and one that enters the window (red).

i j
ACTGTAA**GAT**

QUASAR - Efficient counting and shifting

When the query window $s[i \dots j]$ is moved to $s[i + 1 \dots j + 1]$, the only difference for the block counters are two q -grams, one that leaves (blue) and one that enters the window (red).

i j
ACTGTAA**GAT**

So for each shift update, change the block counters as follows:

- subtract the count of the leading q -gram (blue) from blocks, unless the block counter is $\geq \alpha$ (lock-in)
- add a count to all blocks that contain the trailing q -gram (red)

Gapped q -Grams

Definition

Gapped q -grams are specified with a **mask**, a string of

- # characters denote required matches
- . characters denote “don’t care” positions

For example **##.##** denotes a gapped 4-gram of **span** (length) 6.

Gapped q -Grams

Definition

Gapped q -grams are specified with a **mask**, a string of

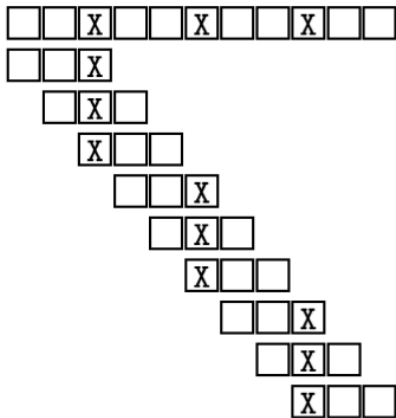
- # characters denote required matches
- . characters denote “don’t care” positions

For example `##.##` denotes a gapped 4-gram of **span** (length) 6.

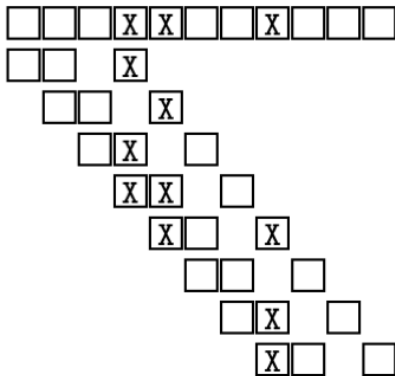
- Gapped q -grams can improve the filter efficiency by orders of magnitude.
- The q -gram lemma is not tight for gapped q -grams and there is no closed formula for computing the threshold for a given edit distance d .

Gapped Q-Grams: Example

$w=11, k=3$ ###



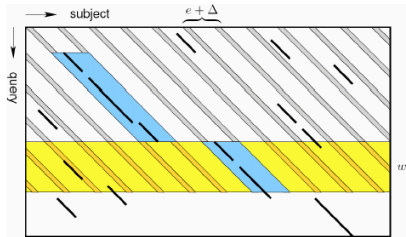
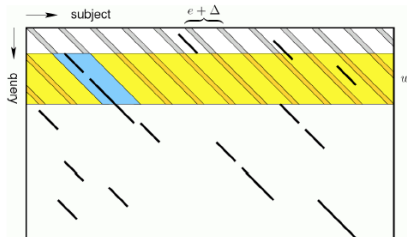
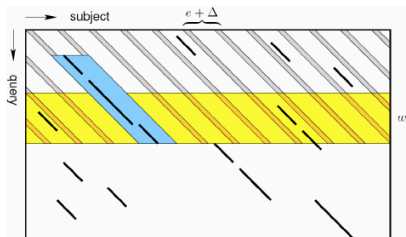
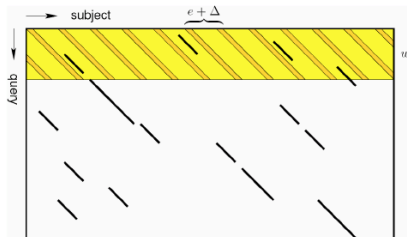
###.#



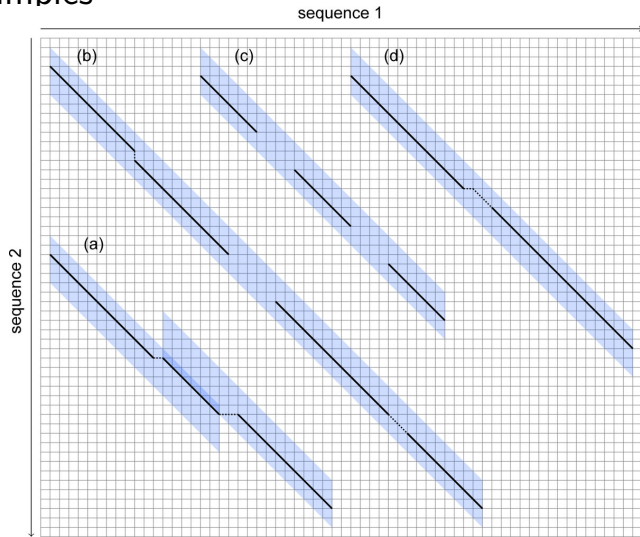
[Stefan Burkhardt and Juha Kärkkäinen, Fundamenta Informaticae, 2003]

SWIFT algorithm: Parallelograms instead of blocks

Slide length- w window over query, count q -grams in diagonals



SWIFT - Examples



from Kehr, Weese, Reinert, BMC Bioinformatics 2011

Other Tools

There are a number of read mappers that were made for the analysis of next-generation sequencing reads against large genomes and use variants of filter algorithms with q-grams

- RMAP
- RazerS and RazerS3
- Hobbes
- GEM
- Stellar
- and many more

Literature

- Q-gram Based Database Searching Using a Suffix Array (QUASAR).
Burkhardt et al., RECOMB 1999
- Efficient q -Gram Filters for Finding All ϵ -Matches over a Given Length.
Rasmussen, Stoye, Myers, Journal of Computational Biology, 2006
- Better filtering with gapped q -grams.
Stefan Burkhardt and Juha Kärkkäinen, Fundamenta Informaticae, 2003

Summary

Approximate pattern matching **with index**

- NFA or branching on FM index
- Branching on suffix tree
- Examples: bwa-sw
- Seed-extend principle: avoid branching by using exact matches
- Examples: bwa-mem, bowtie
- q -gram lemma
- Fast q -gram access: q -gram index
- Filtration approaches (QUASAR, SWIFT)
- Idea of gapped q -grams

Possible Exam Questions

- How can a suffix tree be used to search for approximate pattern occurrences?
- What is the resulting running time?
- What is the benefit of using an FM index instead of a suffix tree?
- Explain approximate search on an FM index by means of an example.
- What's the main idea behind the “seed and extend” paradigm?
- What is the purpose of a filter?
- What is a lossy vs. a lossless filter?
- Explain the q -gram lemma.
- What is a q -gram index? How is it related to the suffix array?
- What is the idea of gapped q -grams?