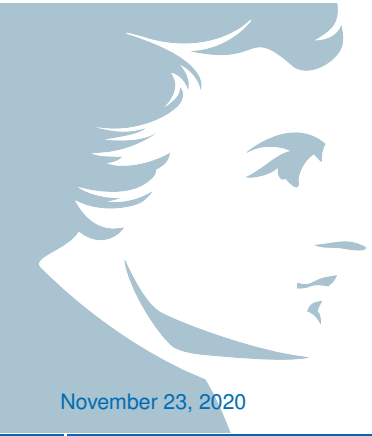
hhu.

Correcting Long-Reads with *k*-mers: A Dream Comes True

Pierre Marijon, Philipp Spohr, Antoine Limasset



November 23, 2020

Table of Contents

Introduction

Heinrich Heine
Universität Düsseldorf

Long-read are usefull

Introduction

Long-read are usefull

But they have a high error rate

Introduction

Long-read are usefull

But they have a high error rate

Self-correction is efficient (low error rate) but:

Introduction

Long-read are usefull

But they have a high error rate

Self-correction is efficient (low error rate) but:

- does not consider heterozygote variants

Introduction

Heinrich Heine
Universität Düsseldorf

Long-read are usefull

But they have a high error rate

Self-correction is efficient (low error rate) but:

- does not consider heterozygote variants
- huge computation time (quadratic)

Introduction

Long-read are usefull

But they have a high error rate

Self-correction is efficient (low error rate) but:

- does not consider heterozygote variants
- huge computation time (quadratic)

Remark: *k*-mers based methods work well on short-read and on hybrid correction data

Table of Contents

PanCov-Correct: Context

PanCov project goal: detect variants in COVID-19 samples

PanCov-Correct: Context

Heinrich Heine
Universität Düsseldorf

PanCov project goal: detect variants in COVID-19 samples

With unusual reads:

PanCov-Correct: Context

Heinrich Heine Universität Düsseldorf

PanCov project goal: detect variants in COVID-19 samples

With unusual reads:

- reverse transcript amplified COVID-19

PanCov-Correct: Context

Heinrich Heine Universität Düsseldorf

PanCov project goal: detect variants in COVID-19 samples

With unusual reads:

- reverse transcript amplified COVID-19
- Nanopore R9

PanCov-Correct: Context

Heinrich Heine Universität Düsseldorf

PanCov project goal: detect variants in COVID-19 samples

With unusual reads:

- reverse transcript amplified COVID-19
- Nanopore R9
- 300bp !!! (due to lab protocol)

PanCov-Correct: Context

Heinrich Heine Universität Düsseldorf

PanCov project goal: detect variants in COVID-19 samples

With unusual reads:

- reverse transcript amplified COVID-19
- Nanopore R9
- 300bp !!! (due to lab protocol)
- 300x coverage (coverage drop $\approx$ 20x)

PanCov-Correct: Context

Heinrich Heine Universität Düsseldorf

PanCov project goal: detect variants in COVID-19 samples

With unusual reads:

- reverse transcript amplified COVID-19
- Nanopore R9
- 300bp !!! (due to lab protocol)
- 300x coverage (coverage drop $\approx$ 20x)
- $\approx$ 7% error

PanCov-Correct: Context

Heinrich Heine Universität Düsseldorf

PanCov project goal: detect variants in COVID-19 samples

With unusual reads:

- reverse transcript amplified COVID-19
- Nanopore R9
- 300bp !!! (due to lab protocol)
- 300x coverage (coverage drop $\approx$ 20x)
- $\approx$ 7% error
- strand bias

PanCov-Correct: Context

PanCov project goal: detect variants in COVID-19 samples

With unusual reads:

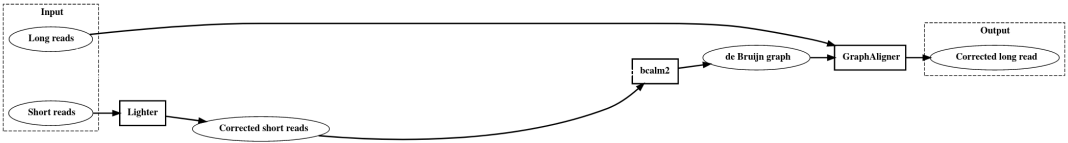
- reverse transcript amplified COVID-19
- Nanopore R9
- 300bp !!! (due to lab protocol)
- 300x coverage (coverage drop $\approx$ 20x)
- $\approx$ 7% error
- strand bias
- strain mixture

PanCov-Correct goal: correct reads will keeping variants, especially low-abundancy strains

PanCov-Correct: Overview

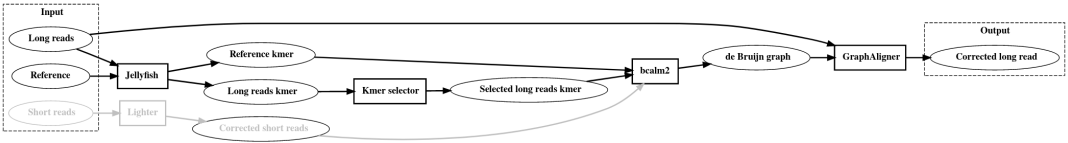
Heinrich Heine Universität Düsseldorf

GraphAligner hybrid correction pipeline:

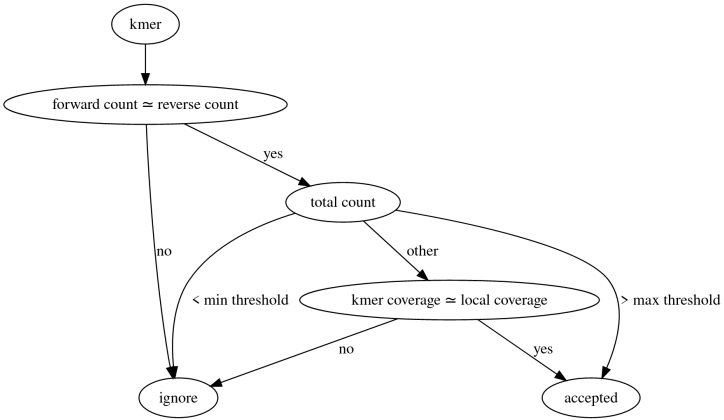


PanCov-Correct: Overview

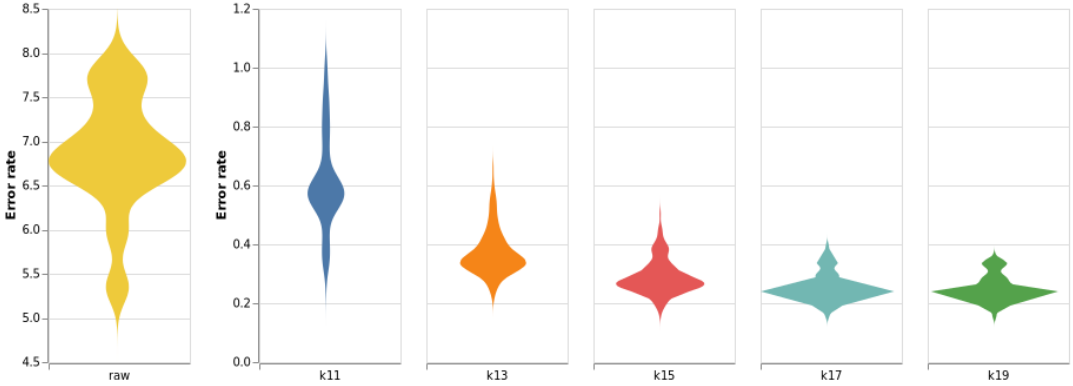
GraphAligner hybrid correction pipeline:



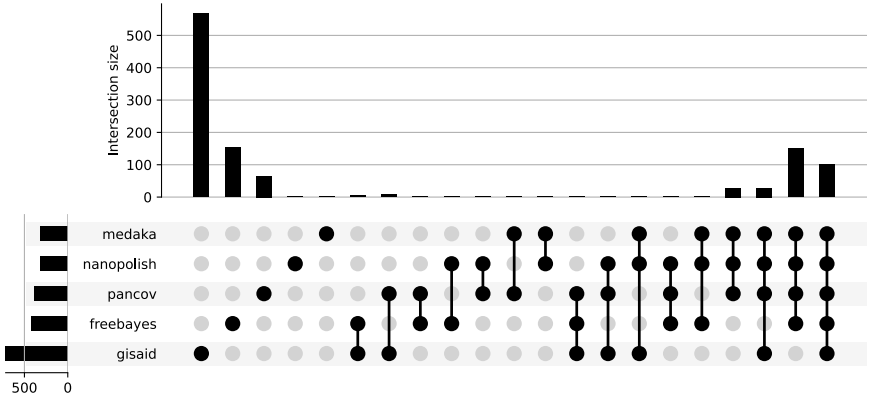
PanCov-Correct: *k*-mers selection



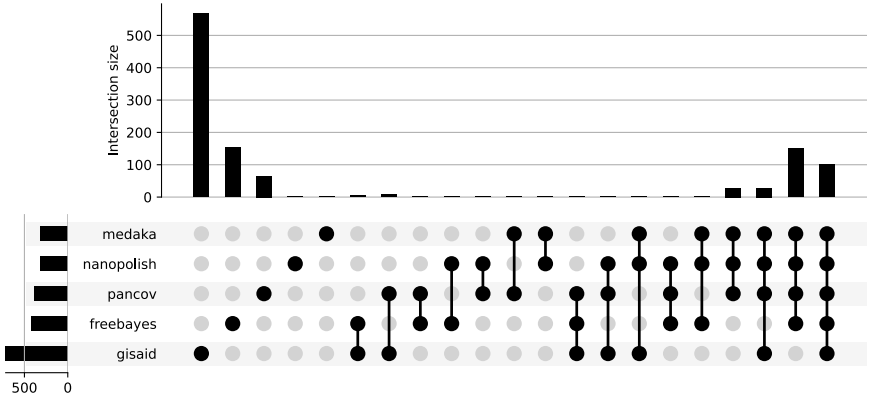
PanCov-Correct: Error rate



PanCov-Correct: Variant calling



PanCov-Correct: Variant calling



Awaiting illumina sequencing for confirmation

`PanCov-Correct`: Conclusion

Heinrich Heine Universität Düsseldorf

On **our** data `PanCov-Correct`:

- Reduces error rate
- Retains low covered variants
- Retains *heterozygote* variants

PanCov-Correct: Conclusion

Heinrich Heine Universität Düsseldorf

On **our** data PanCov-Correct:

- Reduces error rate
- Retains low covered variants
- Retains *heterozygote* variants

Future:

- Standalone tools
- Reference free usage
- Test on other organisms and common reads
- Running time optimization

Table of Contents

Pcon: Overview

Heinrich Heine Universität Düsseldorf

Pcon uses a hash function k-mer $\rightarrow [0, \frac{4^k}{2}[$, if k is odd
We define several functions:

- *kmer2bin* converts a DNA string into two bits A $\rightarrow$ 00, C $\rightarrow$ 01, G $\rightarrow$ 11 and T $\rightarrow$ 10
- *revcomp* performs the reverse complement for a binary representation of k-mer
- *popcount* count number of 1 in binary representation of a number

Pcon: Overview

Heinrich Heine Universität Düsseldorf

Pcon uses a hash function k-mer $\rightarrow [0, \frac{4^k}{2}[$, if k is odd
We define several functions:

- *kmer2bin* converts a DNA string into two bits A $\rightarrow$ 00, C $\rightarrow$ 01, G $\rightarrow$ 11 and T $\rightarrow$ 10
- *revcomp* performs the reverse complement for a binary representation of k-mer
- *popcount* count number of 1 in binary representation of a number

kmer2bin(**AGC**) $\rightarrow$ **00 11 01**
revcomp(**00 11 01**) $\rightarrow$ **11 01 10**
popcount(**00 11 01**) $\rightarrow$ **3**
popcount(**11 01 10**) $\rightarrow$ **4**

Pcon: Overview

Heinrich Heine Universität Düsseldorf

Pcon uses a hash function k-mer $\rightarrow [0, \frac{4^k}{2}[$, if k is odd
We define several functions:

- *kmer2bin* converts a DNA string into two bits A $\rightarrow$ 00, C $\rightarrow$ 01, G $\rightarrow$ 11 and T $\rightarrow$ 10
- *revcomp* performs the reverse complement for a binary representation of k-mer
- *popcount* count number of 1 in binary representation of a number

```
hash (kmer)
    bin = kmer2bin(kmer)
    if popcount(bin) % 2 == 0 then
        return bin » 1
    else
        return revcomp(bin) » 1
```

```
kmer2bin(AGC)        →   00 11 01
revcomp(00 11 01)    →   11 01 10
popcount(00 11 01)   →         3
popcount(11 01 10)   →         4
hash(AGC)            →    11 01 1
```

Pcon: Overview

Heinrich Heine Universität Düsseldorf

Pcon uses a hash function k-mer $\rightarrow [0, \frac{4^k}{2}[$, if k is odd
We define several functions:

- *kmer2bin* converts a DNA string into two bits A $\rightarrow$ 00, C $\rightarrow$ 01, G $\rightarrow$ 11 and T $\rightarrow$ 10
- *revcomp* performs the reverse complement for a binary representation of k-mer
- *popcount* count number of 1 in binary representation of a number

```
hash (kmer)
    bin = kmer2bin(kmer)
    if popcount(bin) % 2 == 0 then
        return bin » 1
    else
        return revcomp(bin) » 1
```

```
count (sequences)
    count = [0] × 4^k / 2
    foreach sequence ∈ sequences do
        foreach kmer ∈ sequence do
            count[hash(kmer)] += 1
    return count
```

`Br`: Overview

1. Count *k*-mers

`Br`: Overview

1. Count *k*-mers
2. Analyse *k*-mers spectrum to find minimal abundance threshold

`Br`: Overview

1. Count *k*-mers
2. Analyse *k*-mers spectrum to find minimal abundance threshold
3. Creation of a bitfield set.

`Br`: Overview

1. Count *k*-mers
2. Analyse *k*-mers spectrum to find minimal abundance threshold
3. Creation of a bitfield set.
4. If *k*-mer count is higher than the minimal abundance threshold *k*-mer is added to bitfield.

`Br`: Overview

1. Count *k*-mers
2. Analyse *k*-mers spectrum to find minimal abundance threshold
3. Creation of a bitfield set.
4. If *k*-mer count is higher than the minimal abundance threshold *k*-mer is added to bitfield.
5. Over each sequence apply correction algorithms

`Br`: Algorithms

One: correct isolate error (musket algorithm with modification to support indel)

Br: Algorithms

Heinrich Heine Universität Düsseldorf

One: correct isolate error (musket algorithm with modification to support indel)

TGGTAGTAGTTACGA

Graph: search a simple path in *DeBruijn* graph between *k*-mer around error

Br: Algorithms

Heinrich Heine Universität Düsseldorf

One: correct isolate error (musket algorithm with modification to support indel)

TGGTAGTAGTTACGA

Graph: search a simple path in *DeBruijn* graph between *k*-mer around error

GapLength: use distance between *k*-mer around error to correct good number of base

Br: Algorithms

One: correct isolate error (musket algorithm with modification to support indel)

TGGTAGTAGTTACGA

Graph: search a simple path in *DeBruijn* graph between *k*-mer around error

GapLength: use distance between *k*-mer around error to correct good number of base

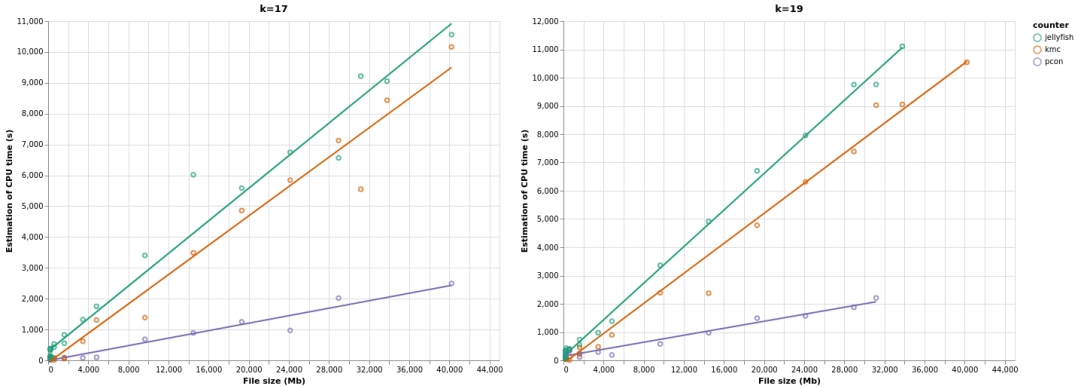
Greedy: get *N* in *DeBruijn* graph perform a pairwise alignment to check it's correct

Pcon & Br Dataset

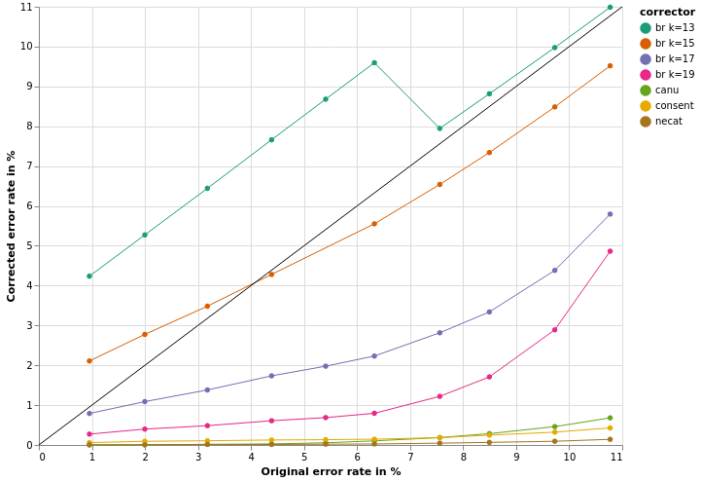
Codename	Organism	Technology	Error rate	Coverage
bacteria	*E. coli*	ONT R10.0	14.7%	$\approx 127x$
bacteria5	*E. coli*	ONT R10.3	5.9%	$\approx 54x$
bacteria7	*E. coli*	ONT R10.3	7.7%	$\approx 127x$
metagenome	metagenome	ONT R10.3	10.8%	
yeast	*S. cerevisiae*	ONT R10.3	8.3 %	$\approx 283x$
synthetic	*E. coli*	Badreads	*	$\approx 50x$
celegans	*C. elegans*	Badreads	5 %	**

*: 1% to 10% per 1% step
**: 16x, 20x, 50x to 400x per 50x step

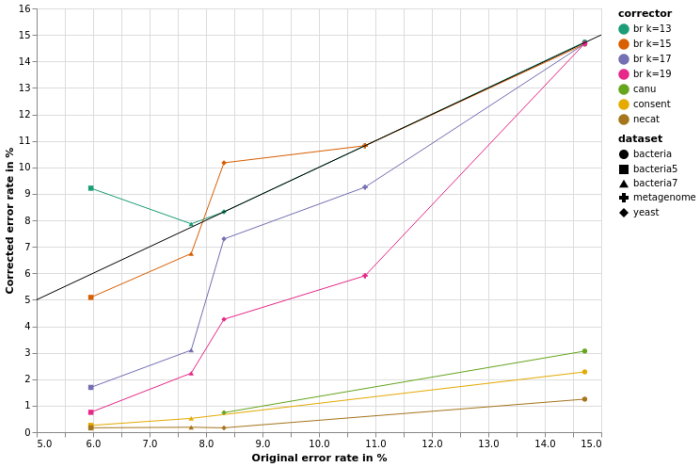
Pcon: Runtime and memory



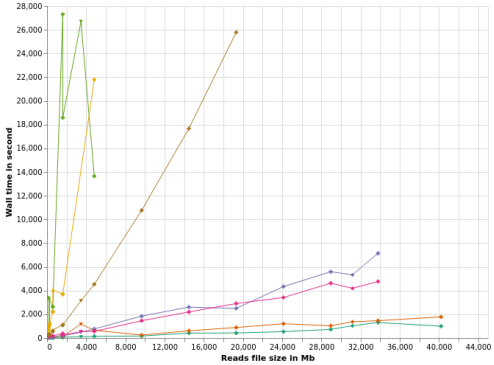
Br: Error rate

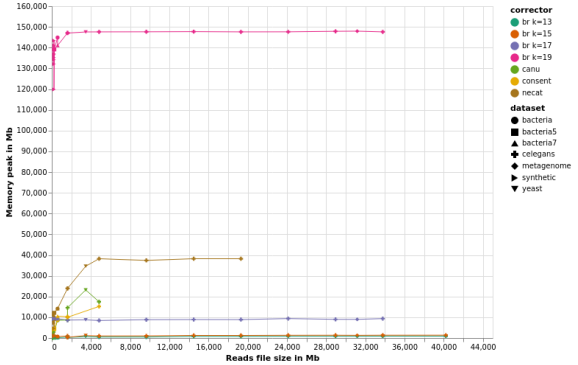


Br: Error rate



Br: Runtime and memory





Conclusion: `Pcon` & `Br`

Heinrich Heine Universität Düsseldorf

`Pcon` count *k*-mer faster than other tools but:

- only odd *k*-mer size
- small *k*-mer k=13 $\rightarrow$ 32Mb k=21 $\rightarrow$ 2Tb
- only canonical form

Conclusion: `Pcon` & `Br`

`Pcon` count *k*-mer faster than other tools but:

- only odd *k*-mer size
- small *k*-mer k=13 $\rightarrow$ 32Mb k=21 $\rightarrow$ 2Tb
- only canonical form

`Br` very fast but less efficient than other, future:

- other algorithms, **Two**, **Greedier**,
- use different set structure $\rightarrow$ use larger kmer
- heterozygosity should by preserved, we have to check
- read filtering, scrubbing, contig polishing,
- hybrid correction

Table of Contents

Take Home Message

Heinrich Heine
Universität Düsseldorf

We can correct long-reads with long-reads *k*-mers:

- with long-read hybrid correction method → `PanCov-Correct`
- with short-read correction method → `Pcon` & `Br`
- do you have any new ideas?

Take Home Message

We can correct long-reads with long-reads *k*-mers:

- with long-read hybrid correction method → `PanCov-Correct`
- with short-read correction method → `Pcon` & `Br`
- do you have any new ideas?

Improvement of raw reads quality will help us

Take Home Message

We can correct long-reads with long-reads *k*-mers:

- with long-read hybrid correction method → `PanCov-Correct`
- with short-read correction method → `Pcon` & `Br`
- do you have any new ideas?

Improvement of raw reads quality will help us

No biorxiv, github link or bioconda logo here we are still in write/development

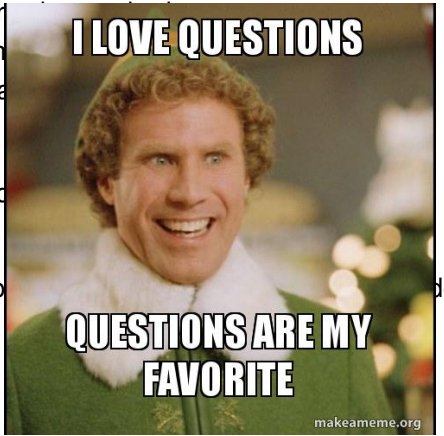
Take Home Message

We can correct long-reads with long-reads *k*-mers:

- with long-read hybrid corr
- with short-read correction
- do you have any new idea

Improvement of raw reads c

No biorxiv, github link or bio development



`Br`: One

Heinrich Heine
Universität Düsseldorf

Input: **TGGTAGTCCTTACGA**

✓ **TGGTA**

Br: One

Heinrich Heine
Universität Düsseldorf

Input: **TGGTAGTCCTTACGA**

✓ **TGGTA**
✓ **GGTAG**

`Br`: One

Heinrich Heine
Universität Düsseldorf

Input: **TGGTAGTCCTTACGA**

✓ **TGGTA**
✓ **GGTAG**
✗ **GTAGT**

Br: One

Heinrich Heine
Universität Düsseldorf

Input: **TGGTAGTCCTTACGA**

✓	**TGGTA**
✓	**GGTAG**
✗	**GTAGT**
✗	**GTAGC**
✗	**GTAGA**
✓	**GTAGG**

Br: One

Heinrich Heine
Universität Düsseldorf

Input: **TGGTAGTCCTTACGA**

✓	**TGGTA**
✓	**GGTAG**
✗	**GTAGT**
✗	**GTAGC**
✗	**GTAGA**
✓	**GTAGG**
✓	**TAGGC**
✓	**AGGCC**

Br: One

Heinrich Heine
Universität Düsseldorf

Input: TGGTAGTCCTTACGA

✓ TGGTA
✓ GGTAG
✗ GTAGT
✗ GTAGC
✗ GTAGA
✓ GTAGG
✓ TAGGC
✓ AGGCC

Output: TGGTAGGCCTTACGA

Br: One

Heinrich Heine Universität Düsseldorf

Input: TGGTAGTCCTTACGA

✓ TGGTA
✓ GGTAG
✗ GTAGT
✗ GTAGC
✗ GTAGA
✓ GTAGG
✓ TAGGC
✓ AGGCC

Output: TGGTAGGCCTTACGA

Input: TGGTAGTCGTTACGA

✓ TGGTA
✓ GGTAG
✗ GTAGT
✗ GTAGC
✗ GTAGA
✓ GTAGG

Br: One

Heinrich Heine Universität Düsseldorf

Input:	TGGTAGTCCTTACGA	Input:	TGGTAGTCGTTACGA
✓	TGGTA	✓	TGGTA
✓	GGTAG	✓	GGTAG
✗	GTAGT	✗	GTAGT
✗	GTAGC	✗	GTAGC
✗	GTAGA	✗	GTAGA
✓	GTAGG	✓	GTAGG
✓	TAGGC	✓	TAGGC
✓	AGGCC	✗	AGGCG

Output: TGGTAGGCCTTACGA

Br: One

Heinrich Heine Universität Düsseldorf

Input:	TGGTAGTCCTTACGA	Input:	TGGTAGTCGTTACGA
✓	TGGTA	✓	TGGTA
✓	GGTAG	✓	GGTAG
✗	GTAGT	✗	GTAGT
✗	GTAGC	✗	GTAGC
✗	GTAGA	✗	GTAGA
✓	GTAGG	✓	GTAGG
✓	TAGGC	✓	TAGGC
✓	AGGCC	✗	AGGCG
Output:	TGGTAGGCCTTACGA	Output:	TGGTAGTCGTTACGA

Br: One

Heinrich Heine Universität Düsseldorf

Input:	TGGTAGTCCTTACGA
✓	TGGTA
✓	GGTAG
✗	GTAGT
✗	GTAGC
✗	GTAGA
✓	GTAGG
✓	TAGGC
✓	AGGCC

Output: TGGTAGGCCTTACGA

Input:	TGGTAGTCGTTACGA
✓	TGGTA
✓	GGTAG
✗	GTAGT
✗	GTAGC
✗	GTAGA
✓	GTAGG
✓	TAGGC
✗	AGGCG

Output: TGGTAGTCGTTACGA

Input:	TGGTAGTCCTTACGA
✓	TGGTA
✓	GGTAG
✗	GTAGT
✗	GTAGC
✓	GTAGA
✓	GTAGG

Br: One

Heinrich Heine Universität Düsseldorf

Input:	TGGTAGTCCTTACGA	Input:	TGGTAGTCGTTACGA	Input:	TGGTAGTCCTTACGA
✓	TGGTA	✓	TGGTA	✓	TGGTA
✓	GGTAG	✓	GGTAG	✓	GGTAG
✗	GTAGT	✗	GTAGT	✗	GTAGT
✗	GTAGC	✗	GTAGC	✗	GTAGC
✗	GTAGA	✗	GTAGA	✓	GTAGA
✓	GTAGG	✓	GTAGG	✓	GTAGG
✓	TAGGC	✓	TAGGC		
✓	AGGCC	✗	AGGCG		
Output:	TGGTAGGCCTTACGA	Output:	TGGTAGTCGTTACGA	Output:	TGGTAGTCCTTACGA

Br: One

Heinrich Heine Universität Düsseldorf

Input:	TGGTAGTCCTTACGA
✓	TGGTA
✓	GGTAG
✗	GTAGT
✗	GTAGC
✗	GTAGA
✓	GTAGG
✓	TAGGC
✓	AGGCC
Output:	TGGTAGGCCTTACGA

Input:	TGGTAGTCGTTACGA
✓	TGGTA
✓	GGTAG
✗	GTAGT
✗	GTAGC
✗	GTAGA
✓	GTAGG
✓	TAGGC
✗	AGGCG
Output:	TGGTAGTCGTTACGA

Input:	TGGTAGTCCTTACGA
✓	TGGTA
✓	GGTAG
✗	GTAGT
✗	GTAGC
✓	GTAGA
✓	GTAGG
Output:	TGGTAGTCCTTACGA

$\mathcal{O}(\text{set access}) = 4 + N$

`Br`: Method **Graph**

Heinrich Heine
Universität Düsseldorf

Input: **TGGTAGTAGTTACGA**

GGTAG **TTACG**

Br: Method **Graph**

Input: **TGGTAGTAGTTACGA**

```
      GGTAG    TTACG
✓      GTAGG
✓       TAGGA
✓        AGGAC
✓         GGACT
✓          GACTT
✓           ACTTA
✓            CTTAC
✓             TTACG
```

Output: **TGGTAGGACTTACGA**

`Br`: Method **Graph**

Input: **TGGTAGTAGTTACGA**

GGTAG TTACG

✓ **GTAGG**
✓ **TAGGA**
✓ **AGGAC**
✓ **GGACT**
✓ **GACTT**
✓ **ACTTA**
✓ **CTTAC**
✓ **TTACG**

Output: **TGGTAGGACTTACGA**

Criteria to stop graph exploration:

- number of successor $\neq 1$
- back on a k-mer seen before

`Br`: Method **Graph**

Input: **TGGTAGTAGTTACGA**

GGTAG TTACG

✓ **GTAGG**
✓ **TAGGA**
✓ **AGGAC**
✓ **GGACT**
✓ **GACTT**
✓ **ACTTA**
✓ **CTTAC**
✓ **TTACG**

Output: **TGGTAGGACTTACGA**

Criteria to stop graph exploration:

- number of successor $\neq 1$
- back on a k-mer seen before

$|\text{set access}| = 2 \times 8 = 2 \times (5 + 3)$

`Br`: Method **Graph**

Input: **TGGTAGTAGTTACGA**

GGTAG TTACG

✓ **GTAGG**
✓ **TAGGA**
✓ **AGGAC**
✓ **GGACT**
✓ **GACTT**
✓ **ACTTA**
✓ **CTTAC**
✓ **TTACG**

Output: **TGGTAGGACTTACGA**

Criteria to stop graph exploration:

- number of successor $\neq 1$
- back on a k-mer seen before

|set access| = $2 \times 8 = 2 \times (5 + 3)$

$\mathcal{O}$(set access) = $2 \times$ (k-mer size + error length)

`Br`: Method **GapLength**

Input: **TGGTAGTAGTTACGA**

GGTAG **TTACG**

Br: Method **GapLength**

Input: **TGGTAGTAGTTACGA**

GGTAG **TTACG**

GapLength(*begin*, *distance*)
if *distance == kmer size* **then**
  One()
else if *distance < kmer size* **then**
  Graph()
else
  get_kmers(begin, distance - kmer size)

If distance > *k*-mer size:
$\mathcal{O}$(set access) = *k*-mer size + 2 $\times$ error length

Br: Method **GapLength**

Input: TGGTAGTAGTTACGA

GGTAG TTACG

✓ GTAGG

✓ TAGGA

✓ AGGAC

GapLength(*begin*, *distance*)
if *distance == kmer size* **then**
  One()
else if *distance < kmer size* **then**
  Graph()
else
  get_kmers(begin, distance - kmer size)

If distance > *k*-mer size:
$\mathcal{O}$(set access) = *k*-mer size + 2 $\times$ error length

Br: Method **GapLength**

Input: TGGTAGTAGTTACGA

GGTAG TTACG

✓ GTAGG

✓ TAGGA

✓ AGGAC

Output: TGGTAGGACTTACGA

GapLength(*begin*, *distance*)
if *distance == kmer size* **then**
  One()
else if *distance < kmer size* **then**
  Graph()
else
  get_kmers(begin, distance - kmer size)

If distance > *k*-mer size:
$\mathcal{O}$(set access) = *k*-mer size + 2 $\times$ error length

Br: Method **Greedy**

Heinrich Heine
Universität Düsseldorf

Input: **TGGTAGTAGTTACGA**

Br: Method **Greedy**

Heinrich Heine
Universität Düsseldorf

Input: **TGGTAGTAGTTACGA**

✓ **GTAGG**
✓ **TAGGA**
✓ **AGGAC**
✓ **GGACT**
✓ **GACTT**

Br: Method **Greedy**

Input: TGGTAGTAGTTACGA

✓ GTAGG
✓ TAGGA
✓ AGGAC
✓ GGACT
✓ GACTT

```
GTAGTAGTT
|||| | ||
GTAGGACTT
```

Br: Method **Greedy**

Heinrich Heine Universität Düsseldorf

Input: **TGGTAGTAGTTACGA**

✓ **GTAGG**
✓ **TAGGA**
✓ **AGGAC**
✓ **GGACT**
✓ **GACTT**

GTAGTAGTT
|||| | ||
GTAGGACTT

Output: **TGGTAGGACTTACGA**

$\mathcal{O}$(set access) = M + $\mathcal{O}$(2 $\times$ (K+M))