# Enhanced de Bruijn Graphs

**Pierre MORISSE**
pierre.morisse2@univ-rouen.fr

Supervisors: Thierry LECROQ and Arnaud LEFEBVRE

Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes

September 14, 2017


# Plan

# Next Generation Sequencing

- NGS technologies allow to produce millions of short sequences (100-300 bases), called reads
- These reads contain sequencing errors ($\sim 1\%$)
- Efficient algorithms and data structures are required to process these reads
- Main focus: error correction and assembly

# Next Generation Sequencing

- NGS technologies allow to produce millions of short sequences (100-300 bases), called reads
- These reads contain sequencing errors ($\sim$ 1%)
- Efficient algorithms and data structures are required to process these reads
- Main focus: error correction and assembly

# Next Generation Sequencing

- NGS technologies allow to produce millions of short sequences (100-300 bases), called reads
- These reads contain sequencing errors ($\sim$ 1%)
- Efficient algorithms and data structures are required to process these reads
- Main focus: error correction and assembly

# Next Generation Sequencing

- NGS technologies allow to produce millions of short sequences (100-300 bases), called reads
- These reads contain sequencing errors ($\sim$ 1%)
- Efficient algorithms and data structures are required to process these reads
- Main focus: error correction and assembly

# Next Generation Sequencing

- Recently, Third Generation Sequencing technologies started to develop
- Two main technologies: Pacific Biosciences and Oxford Nanopore
- Allow the sequencing of longer reads (several thousand of bases)
- Very useful to resolve assembly problems for large and complex genomes
- Much higher error rate, around 15% for Pacific Biosciences and up to 30% for Oxford Nanopore

# Next Generation Sequencing

- Recently, Third Generation Sequencing technologies started to develop
- Two main technologies: Pacific Biosciences and Oxford Nanopore
- Allow the sequencing of longer reads (several thousand of bases)
- Very useful to resolve assembly problems for large and complex genomes
- Much higher error rate, around 15% for Pacific Biosciences and up to 30% for Oxford Nanopore

# Next Generation Sequencing

- Recently, Third Generation Sequencing technologies started to develop
- Two main technologies: Pacific Biosciences and Oxford Nanopore
- Allow the sequencing of longer reads (several thousand of bases)
- Very useful to resolve assembly problems for large and complex genomes
- Much higher error rate, around 15% for Pacific Biosciences and up to 30% for Oxford Nanopore

# Next Generation Sequencing

- Recently, Third Generation Sequencing technologies started to develop
- Two main technologies: Pacific Biosciences and Oxford Nanopore
- Allow the sequencing of longer reads (several thousand of bases)
- Very useful to resolve assembly problems for large and complex genomes
- Much higher error rate, around 15% for Pacific Biosciences and up to 30% for Oxford Nanopore

# Next Generation Sequencing

- Recently, Third Generation Sequencing technologies started to develop
- Two main technologies: Pacific Biosciences and Oxford Nanopore
- Allow the sequencing of longer reads (several thousand of bases)
- Very useful to resolve assembly problems for large and complex genomes
- Much higher error rate, around 15% for Pacific Biosciences and up to 30% for Oxford Nanopore

# Overlap graph

An overlap graph is a graph structure that allows to compute overlaps of variable length between the reads of a given set.

**Formal definition**

For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $OG(R) = (V, E)$ such as:

- $V : \{r_i; i = 1, ..., n\}$
- $E : \{(s, l, d); s, d \in V \text{ and } \text{suff}_l(s) = \text{pref}_l(d)\}$

# Overlap graph

An overlap graph is a graph structure that allows to compute overlaps of variable length between the reads of a given set.

## Formal definition

For a set of reads $R = \{r_1, r_2, ..., r_n\}, OG(R) = (V, E)$ such as:

- $V : \{r_i; i = 1, ..., n\}$
- $E : \{(s, l, d); s, d \in V \text{ and } \text{suff}_l(s) = \text{pref}_l(d)\}$

# Overlap graph

An overlap graph is a graph structure that allows to compute overlaps of variable length between the reads of a given set.

## Formal definition

For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $OG(R) = (V, E)$ such as:

- $V : \{r_i; i = 1, ..., n\}$
- $E : \{(s, l, d); s, d \in V \text{ and } \text{suff}_l(s) = \text{pref}_l(d)\}$

# Overlap graph

An overlap graph is a graph structure that allows to compute overlaps of variable length between the reads of a given set.

## Formal definition

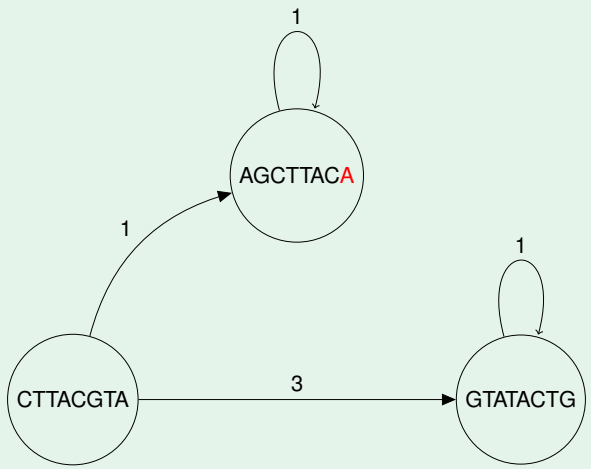
For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $OG(R) = (V, E)$ such as:

- $V : \{r_i; i = 1, ..., n\}$
- $E : \{(s, l, d); s, d \in V \text{ and } \text{suff}_l(s) = \text{pref}_l(d)\}$

# Overlap graph

## Example

With the set of reads $S = \{\text{AGCTTACA}, \text{CTTACGTA}, \text{GTATACTG}\}$, we obtain the following overlap graph:



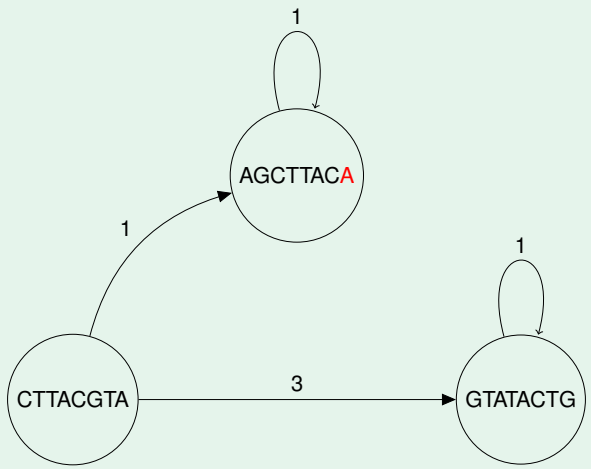

## Drawback

Faces difficulties with sequencing errors.

# Overlap graph

## Example

With the set of reads $S = \{\text{AGCTTACA}, \text{CTTACGTA}, \text{GTATACTG}\}$, we obtain the following overlap graph:




## Drawback

Faces difficulties with sequencing errors.

# de Bruijn graph

A de Bruijn graph of order $k$ is a graph structure that allows to compute overlaps of constant length $k-1$ between the $k$-mers of the reads of a given set.

**Formal definition**

For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $DBG_k(R) = (V, E)$ such as:

- $V : \{w; |w| = k \text{ and } \exists i; w \in Fact(r_i)\}$
- $E : \{(s, d); s, d \in V \text{ and } \text{suff}_{k-1}(s) = \text{pref}_{k-1}(d)\}$

# de Bruijn graph

A de Bruijn graph of order $k$ is a graph structure that allows to compute overlaps of constant length $k-1$ between the $k$-mers of the reads of a given set.

## Formal definition

For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $DBG_k(R) = (V, E)$ such as:

- $V : \{w; |w| = k \text{ and } \exists i; w \in Fact(r_i)\}$
- $E : \{(s, d); s, d \in V \text{ and } \text{suff}_{k-1}(s) = \text{pref}_{k-1}(d)\}$

# de Bruijn graph

A de Bruijn graph of order $k$ is a graph structure that allows to compute overlaps of constant length $k-1$ between the $k$-mers of the reads of a given set.

## Formal definition

For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $DBG_k(R) = (V, E)$ such as:

- $V : \{w; |w| = k \text{ and } \exists i; w \in Fact(r_i)\}$
- $E : \{(s, d); s, d \in V \text{ and } \text{suff}_{k-1}(s) = \text{pref}_{k-1}(d)\}$

# de Bruijn graph

A de Bruijn graph of order $k$ is a graph structure that allows to compute overlaps of constant length $k-1$ between the $k$-mers of the reads of a given set.

## Formal definition

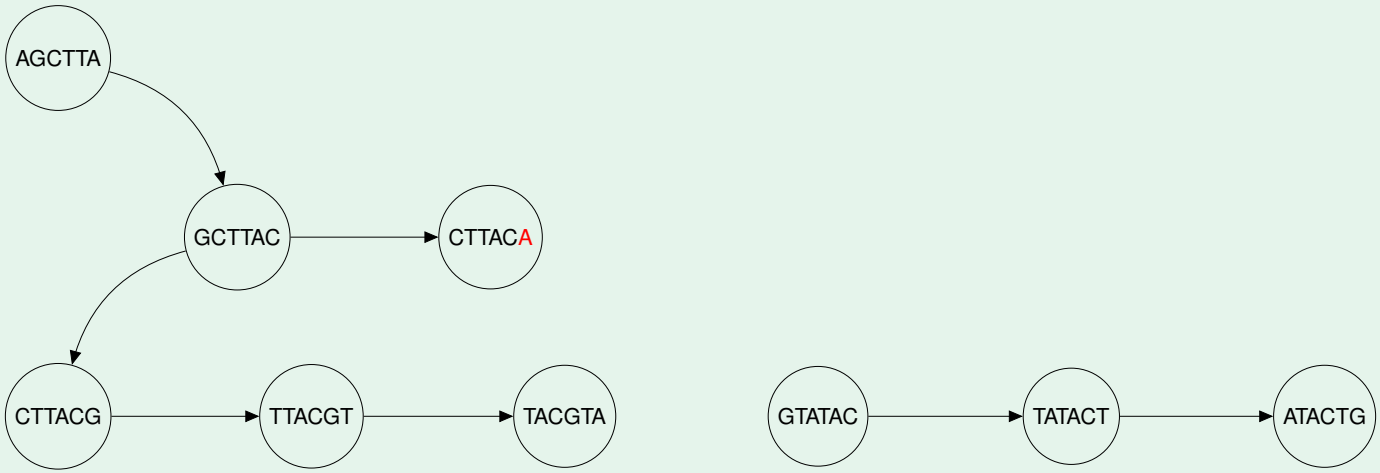
For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $DBG_k(R) = (V, E)$ such as:

- $V : \{w; |w| = k \text{ and } \exists i; w \in Fact(r_i)\}$
- $E : \{(s, d); s, d \in V \text{ and } \text{suff}_{k-1}(s) = \text{pref}_{k-1}(d)\}$

# de Bruijn graph

## Example

With the set of reads $S = \{\text{AGCTTACA}, \text{CTTACGTA}, \text{GTATACTG}\}$, we obtain the following de Bruijn graph of order 6:



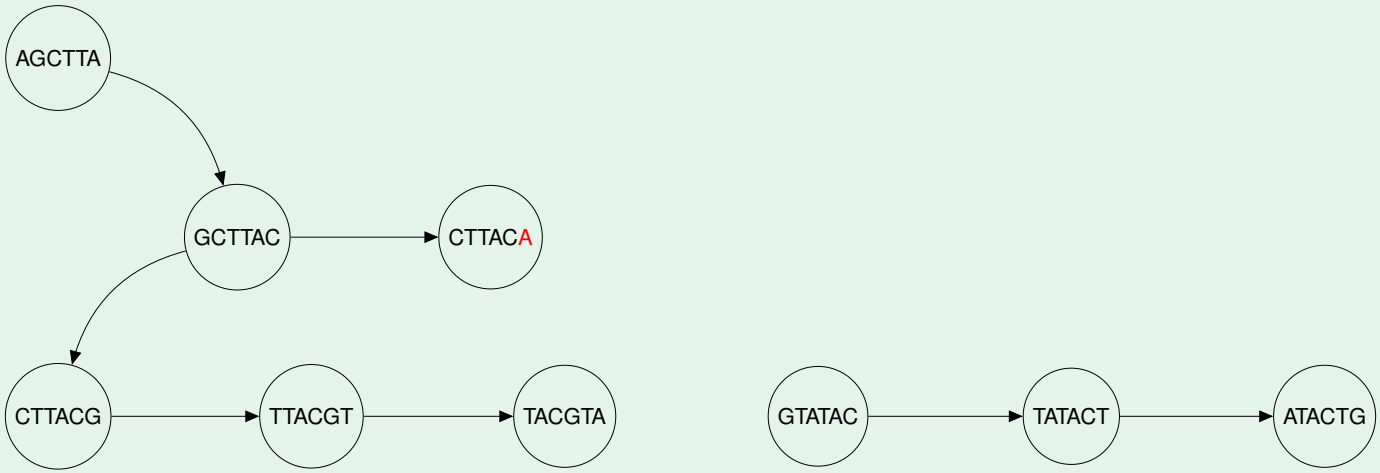

## Drawback

Faces difficulties with locally insufficient coverage.

# de Bruijn graph

## Example

With the set of reads $S = \{\text{AGCTTACA}, \text{CTTACGTA}, \text{GTATACTG}\}$, we obtain the following de Bruijn graph of order 6:




## Drawback

Faces difficulties with locally insufficient coverage.

# Multiple de Bruijn graphs

- Usually, multiple de Bruijn graphs of different orders are built
- Requires a different graph for each order
- Consumes large amounts of memory

# Multiple de Bruijn graphs

- Usually, multiple de Bruijn graphs of different orders are built
- Requires a different graph for each order
- Consumes large amounts of memory

# Multiple de Bruijn graphs

- Usually, multiple de Bruijn graphs of different orders are built
- Requires a different graph for each order
- Consumes large amounts of memory

# Enhanced de Bruijn graph

## Idea

Enhance the de Bruijn graph with the capability of computing overlaps of variable lengths between the $k$-mers, in an overlap graph fashion, in order to avoid building multiple de Bruijn graphs of different orders.

## Formal definition

For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $eDBG_{k,m}(R) = (V, E)$ such as:

- $V : \{w; |w| = k \text{ and } \exists i; w \in Fact(r_i)\}$
- $E : \{(s, l, d); s, d \in V; m \leq l \leq k-1 \text{ and } \text{suff}_l(s) = \text{pref}_l(d)\}$

# Enhanced de Bruijn graph

## Idea

Enhance the de Bruijn graph with the capability of computing overlaps of variable lengths between the $k$-mers, in an overlap graph fashion, in order to avoid building multiple de Bruijn graphs of different orders.

## Formal definition

For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $eDBG_{k,m}(R) = (V, E)$ such as:

- $V : \{w; |w| = k \text{ and } \exists i; w \in Fact(r_i)\}$
- $E : \{(s, l, d); s, d \in V; m \leq l \leq k-1 \text{ and } suff_l(s) = pref_l(d)\}$

# Enhanced de Bruijn graph

## Idea

Enhance the de Bruijn graph with the capability of computing overlaps of variable lengths between the $k$-mers, in an overlap graph fashion, in order to avoid building multiple de Bruijn graphs of different orders.

## Formal definition

For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $eDBG_{k,m}(R) = (V, E)$ such as:

- $V : \{w; |w| = k \text{ and } \exists i; w \in Fact(r_i)\}$
- $E : \{(s, l, d); s, d \in V; m \leq l \leq k-1 \text{ and } suff_l(s) = pref_l(d)\}$

# Enhanced de Bruijn graph

## Idea

Enhance the de Bruijn graph with the capability of computing overlaps of variable lengths between the $k$-mers, in an overlap graph fashion, in order to avoid building multiple de Bruijn graphs of different orders.

## Formal definition

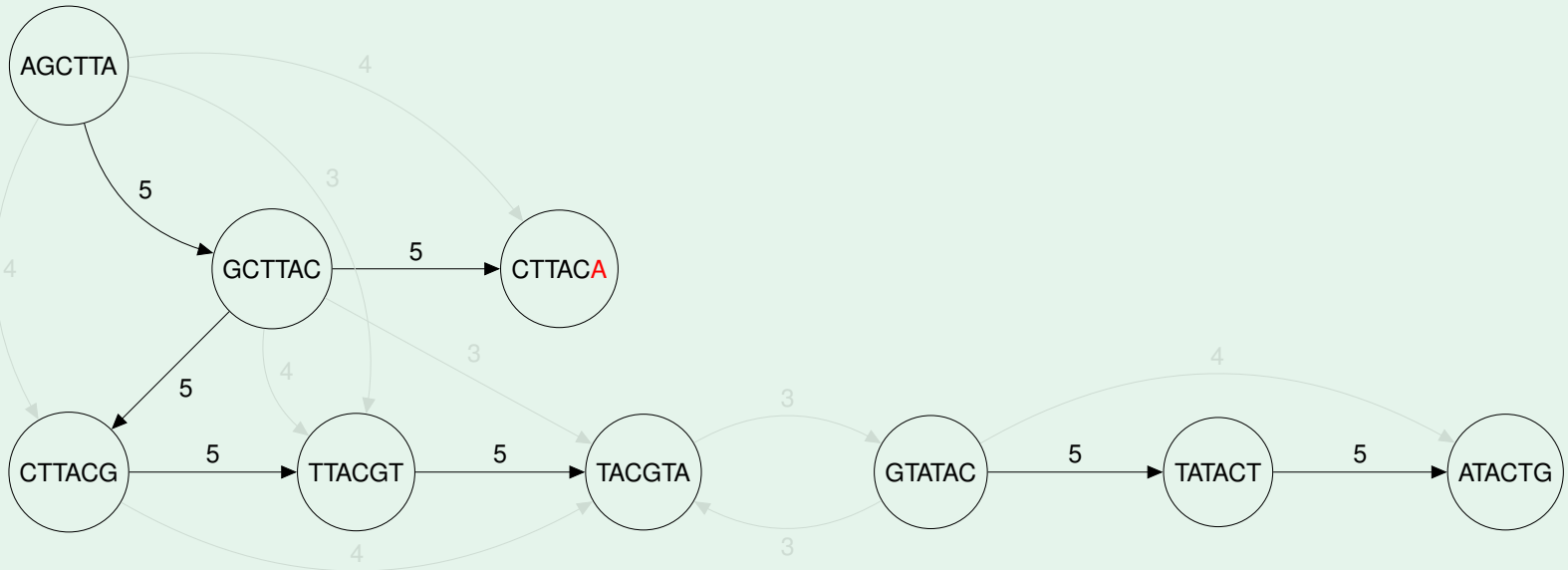
For a set of reads $R = \{r_1, r_2, ..., r_n\}$, $eDBG_{k,m}(R) = (V, E)$ such as:

- $V : \{w; |w| = k \text{ and } \exists i; w \in Fact(r_i)\}$
- $E : \{(s, l, d); s, d \in V; m \leq l \leq k-1 \text{ and } \text{suff}_l(s) = \text{pref}_l(d)\}$

# Enhanced de Bruijn graph

## Example

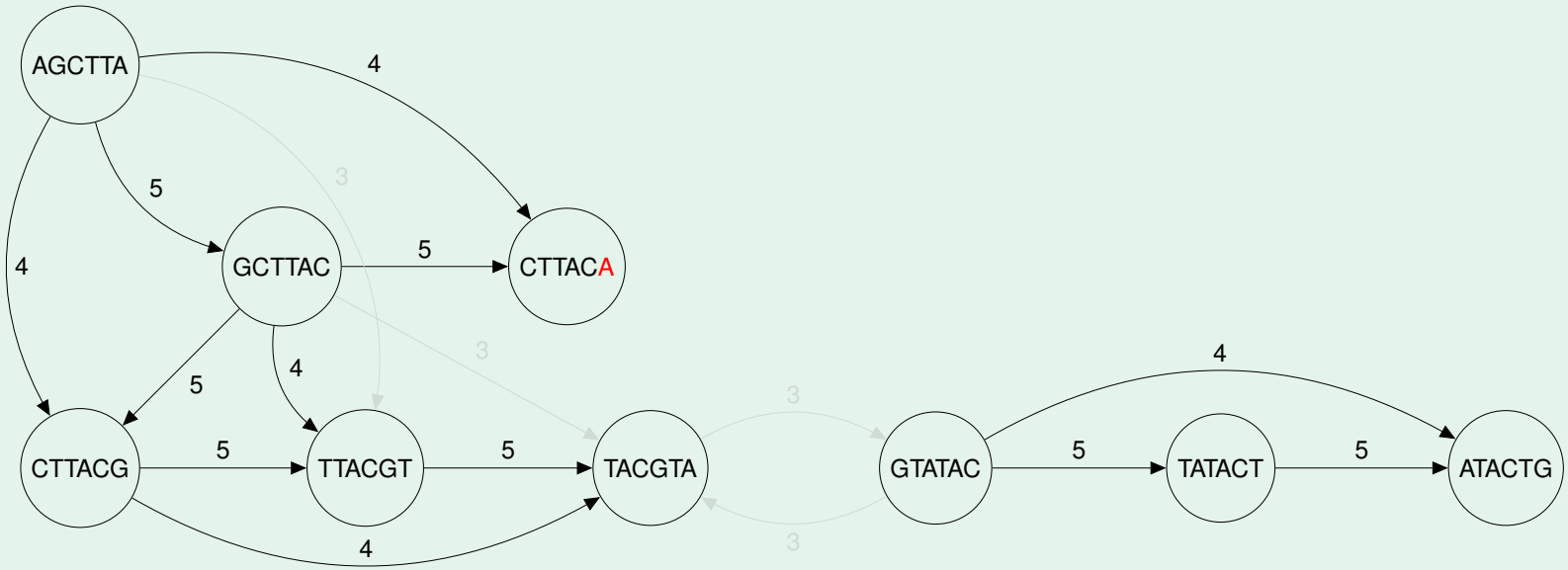
With the set of reads $S = \{\text{AGCTTACA}, \text{CTTACGTA}, \text{GTATACTG}\}$, we obtain the following enhanced de Bruijn graph of order 6,3:



# Enhanced de Bruijn graph

## Example

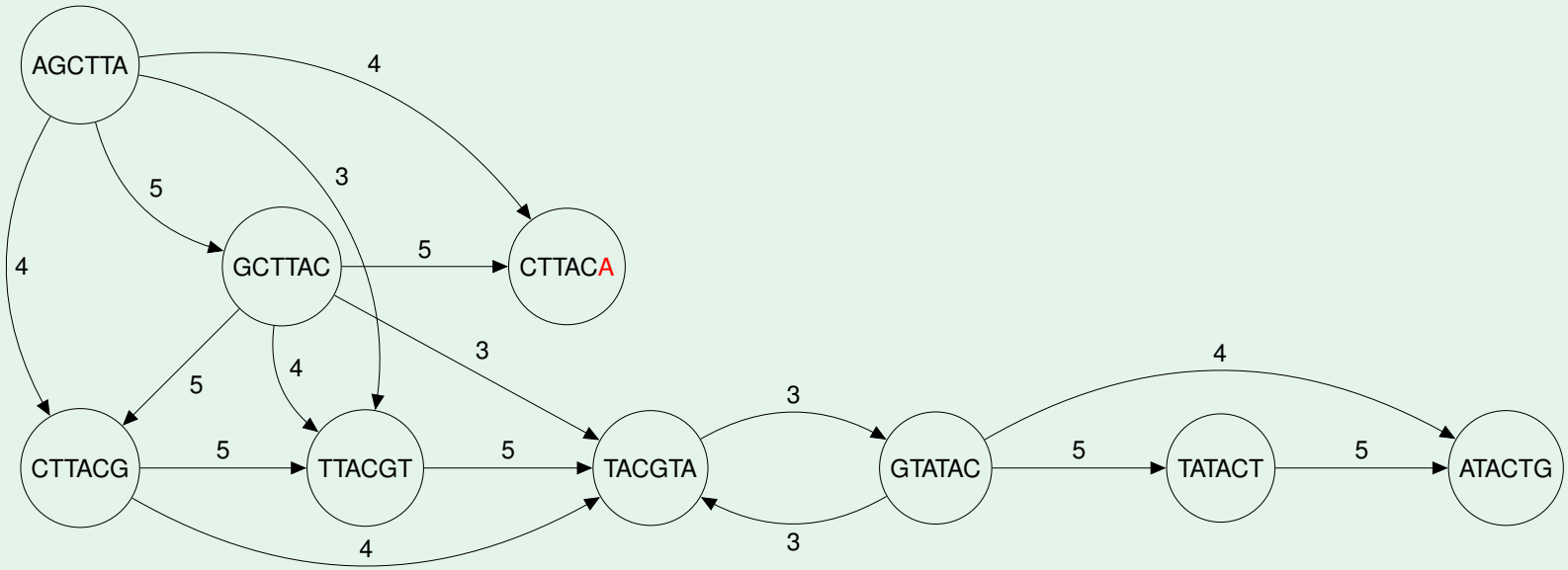
With the set of reads $S = \{\text{AGCTTACA}, \text{CTTACGTA}, \text{GTATACTG}\}$, we obtain the following enhanced de Bruijn graph of order 6,3:



# Enhanced de Bruijn graph

## Example

With the set of reads $S = \{\text{AGCTTACA}, \text{CTTACGTA}, \text{GTATACTG}\}$, we obtain the following enhanced de Bruijn graph of order 6,3:



# Construction

- The enhanced de Bruijn graph does not need to be explicitly built
- It can be traversed with the help of an index structure:
  - All the $k$-mers from the reads are stored in the index
  - The index is queried to retrieve the edges
- Makes backwards traversal easy

# Construction

- The enhanced de Bruijn graph does not need to be explicitly built
- It can be traversed with the help of an index structure:
  - All the $k$-mers from the reads are stored in the index
  - The index is queried to retrieve the edges
- Makes backwards traversal easy

# Construction

- The enhanced de Bruijn graph does not need to be explicitly built
- It can be traversed with the help of an index structure:
  - All the $k$-mers from the reads are stored in the index
  - The index is queried to retrieve the edges
- Makes backwards traversal easy

# Construction

- The enhanced de Bruijn graph does not need to be explicitly built
- It can be traversed with the help of an index structure:
  - All the $k$-mers from the reads are stored in the index
  - The index is queried to retrieve the edges
- Makes backwards traversal easy

# Construction

- The enhanced de Bruijn graph does not need to be explicitly built
- It can be traversed with the help of an index structure:
  - All the $k$-mers from the reads are stored in the index
  - The index is queried to retrieve the edges
- Makes backwards traversal easy

# Definition

PgSA [Kowalski et al., 2015] is a data structure that allows the indexing of a set of reads, in order to answer the following queries on the reads, for a given string $f$:

1. In which reads does $f$ occur?
2. In how many reads does $f$ occur?
3. What are the occurrences positions of $f$?
4. What is the number of occurrences of $f$?
5. In which reads does $f$ occur only once?
6. In how many reads does $f$ occur only once?
7. What are the occurrences positions of $f$ in the reads where it occurs only once?

# Definition

PgSA [Kowalski et al., 2015] is a data structure that allows the indexing of a set of reads, in order to answer the following queries on the reads, for a given string $f$:

1. In which reads does $f$ occur?
2. In how many reads does $f$ occur?
3. What are the occurrences positions of $f$?
4. What is the number of occurrences of $f$?
5. In which reads does $f$ occur only once?
6. In how many reads does $f$ occur only once?
7. What are the occurrences positions of $f$ in the reads where it occurs only once?

# Definition

PgSA [Kowalski et al., 2015] is a data structure that allows the indexing of a set of reads, in order to answer the following queries on the reads, for a given string $f$:

1. In which reads does $f$ occur?
2. In how many reads does $f$ occur?
3. What are the occurrences positions of $f$?
4. What is the number of occurrences of $f$?
5. In which reads does $f$ occur only once?
6. In how many reads does $f$ occur only once?
7. What are the occurrences positions of $f$ in the reads where it occurs only once?

# Definition

PgSA [Kowalski et al., 2015] is a data structure that allows the indexing of a set of reads, in order to answer the following queries on the reads, for a given string $f$:

1. In which reads does $f$ occur?
2. In how many reads does $f$ occur?
3. What are the occurrences positions of $f$?
4. What is the number of occurrences of $f$?
5. In which reads does $f$ occur only once?
6. In how many reads does $f$ occur only once?
7. What are the occurrences positions of $f$ in the reads where it occurs only once?

# Definition

PgSA [Kowalski et al., 2015] is a data structure that allows the indexing of a set of reads, in order to answer the following queries on the reads, for a given string $f$:

1. In which reads does $f$ occur?
2. In how many reads does $f$ occur?
3. What are the occurrences positions of $f$?
4. What is the number of occurrences of $f$?
5. In which reads does $f$ occur only once?
6. In how many reads does $f$ occur only once?
7. What are the occurrences positions of $f$ in the reads where it occurs only once?

# Definition

PgSA [Kowalski et al., 2015] is a data structure that allows the indexing of a set of reads, in order to answer the following queries on the reads, for a given string $f$:

1. In which reads does $f$ occur?
2. In how many reads does $f$ occur?
3. What are the occurrences positions of $f$?
4. What is the number of occurrences of $f$?
5. In which reads does $f$ occur only once?
6. In how many reads does $f$ occur only once?
7. What are the occurrences positions of $f$ in the reads where it occurs only once?

# Definition

PgSA [Kowalski et al., 2015] is a data structure that allows the indexing of a set of reads, in order to answer the following queries on the reads, for a given string $f$:

1. In which reads does $f$ occur?
2. In how many reads does $f$ occur?
3. What are the occurrences positions of $f$?
4. What is the number of occurrences of $f$?
5. In which reads does $f$ occur only once?
6. In how many reads does $f$ occur only once?
7. What are the occurrences positions of $f$ in the reads where it occurs only once?

# Definition

PgSA [Kowalski et al., 2015] is a data structure that allows the indexing of a set of reads, in order to answer the following queries on the reads, for a given string $f$:

1. In which reads does $f$ occur?
2. In how many reads does $f$ occur?
3. What are the occurrences positions of $f$?
4. What is the number of occurrences of $f$?
5. In which reads does $f$ occur only once?
6. In how many reads does $f$ occur only once?
7. What are the occurrences positions of $f$ in the reads where it occurs only once?

# Index construction

- Concatenation of the reads, with respect to their overlaps

  Ex: ACGT + GTGG $\Rightarrow$ ACGTGG

- Construction of the sparse suffix array of the obtained pseudogenome
- Construction of an auxiliary array
- Queries are handled by a binary search over the suffix array, and with the help of the auxiliary array

# Index construction

- Concatenation of the reads, with respect to their overlaps

  Ex: ACGT + GTGG $\Rightarrow$ ACGTGG

- Construction of the sparse suffix array of the obtained pseudogenome
- Construction of an auxiliary array
- Queries are handled by a binary search over the suffix array, and with the help of the auxiliary array

# Index construction

- Concatenation of the reads, with respect to their overlaps

  Ex: ACGT + GTGG $\Rightarrow$ ACGTGG

- Construction of the sparse suffix array of the obtained pseudogenome
- Construction of an auxiliary array
- Queries are handled by a binary search over the suffix array, and with the help of the auxiliary array

# Index construction

- Concatenation of the reads, with respect to their overlaps

  Ex: ACGT + GTGG $\Rightarrow$ ACGTGG

- Construction of the sparse suffix array of the obtained pseudogenome

- Construction of an auxiliary array

- Queries are handled by a binary search over the suffix array, and with the help of the auxiliary array

# Index construction

- Concatenation of the reads, with respect to their overlaps

  Ex: ACGT + GTGG $\Rightarrow$ ACGTGG

- Construction of the sparse suffix array of the obtained pseudogenome

- Construction of an auxiliary array

- Queries are handled by a binary search over the suffix array, and with the help of the auxiliary array

# Index construction

- Concatenation of the reads, with respect to their overlaps

  Ex: ACGT + GTGG $\Rightarrow$ ACGTGG

- Construction of the sparse suffix array of the obtained pseudogenome
- Construction of an auxiliary array
- Queries are handled by a binary search over the suffix array, and with the help of the auxiliary array

# Traversal of the enhanced de Bruijn graph

- Extract the $k$-mers of the reads
- Build the index of the $k$-mers
- Query the index, looping over the third query (what are the occurrences positions of $f$?), to retrieve the edges

# Traversal of the enhanced de Bruijn graph

- Extract the $k$-mers of the reads
- Build the index of the $k$-mers
- Query the index, looping over the third query (what are the occurrences positions of $f$?), to retrieve the edges

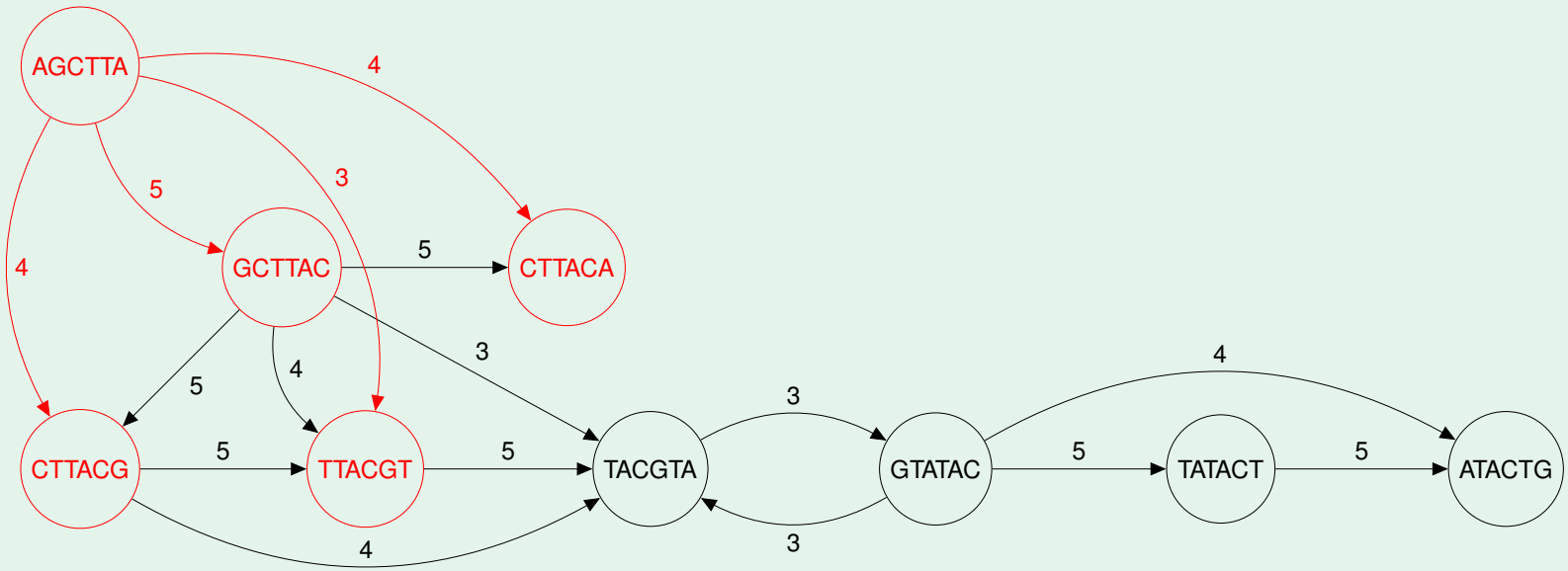
# Traversal of the enhanced de Bruijn graph

- Extract the $k$-mers of the reads
- Build the index of the $k$-mers
- Query the index, looping over the third query (what are the occurrences positions of $f$?), to retrieve the edges

# Traversal of the enhanced de Bruijn graph

## Example

Traversing the previous enhanced de Bruijn graph:



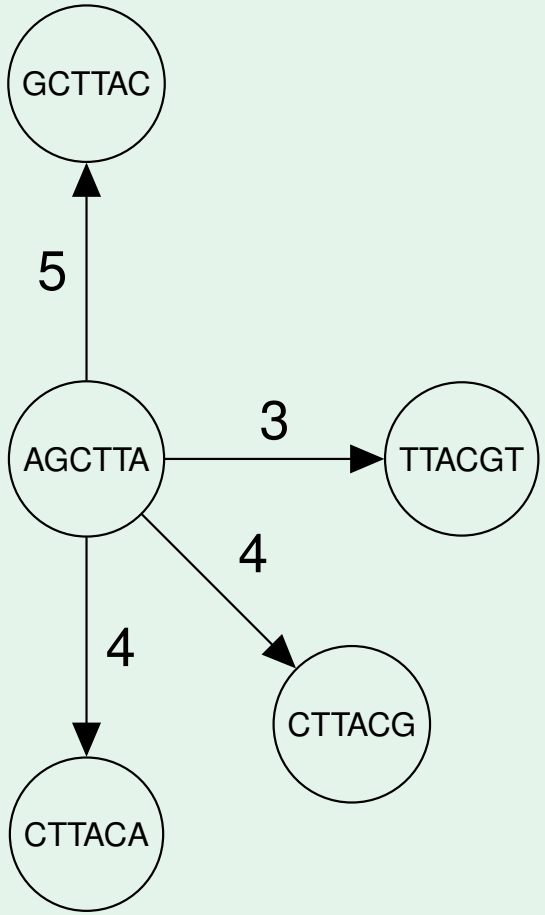
# Traversal of the enhanced de Bruijn graph

## Example

*k*-mers set

1: AGCTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index



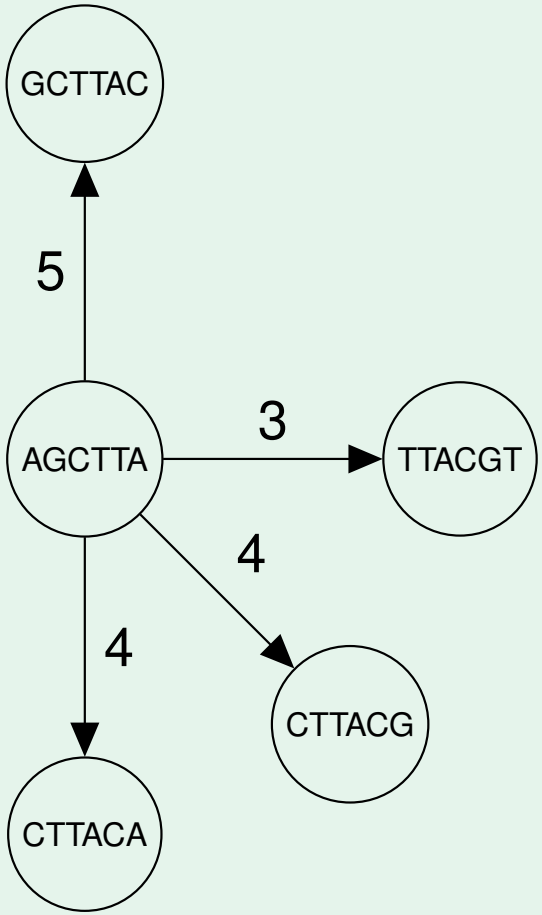
# Traversal of the enhanced de Bruijn graph

## Example

*k*-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index



# Traversal of the enhanced de Bruijn graph

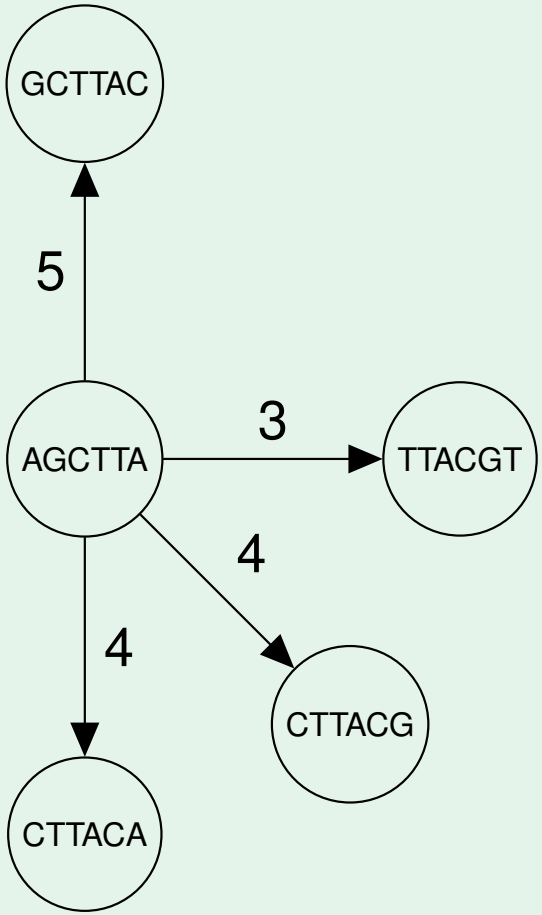
## Example

*k*-mers set

1: **A**GCTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

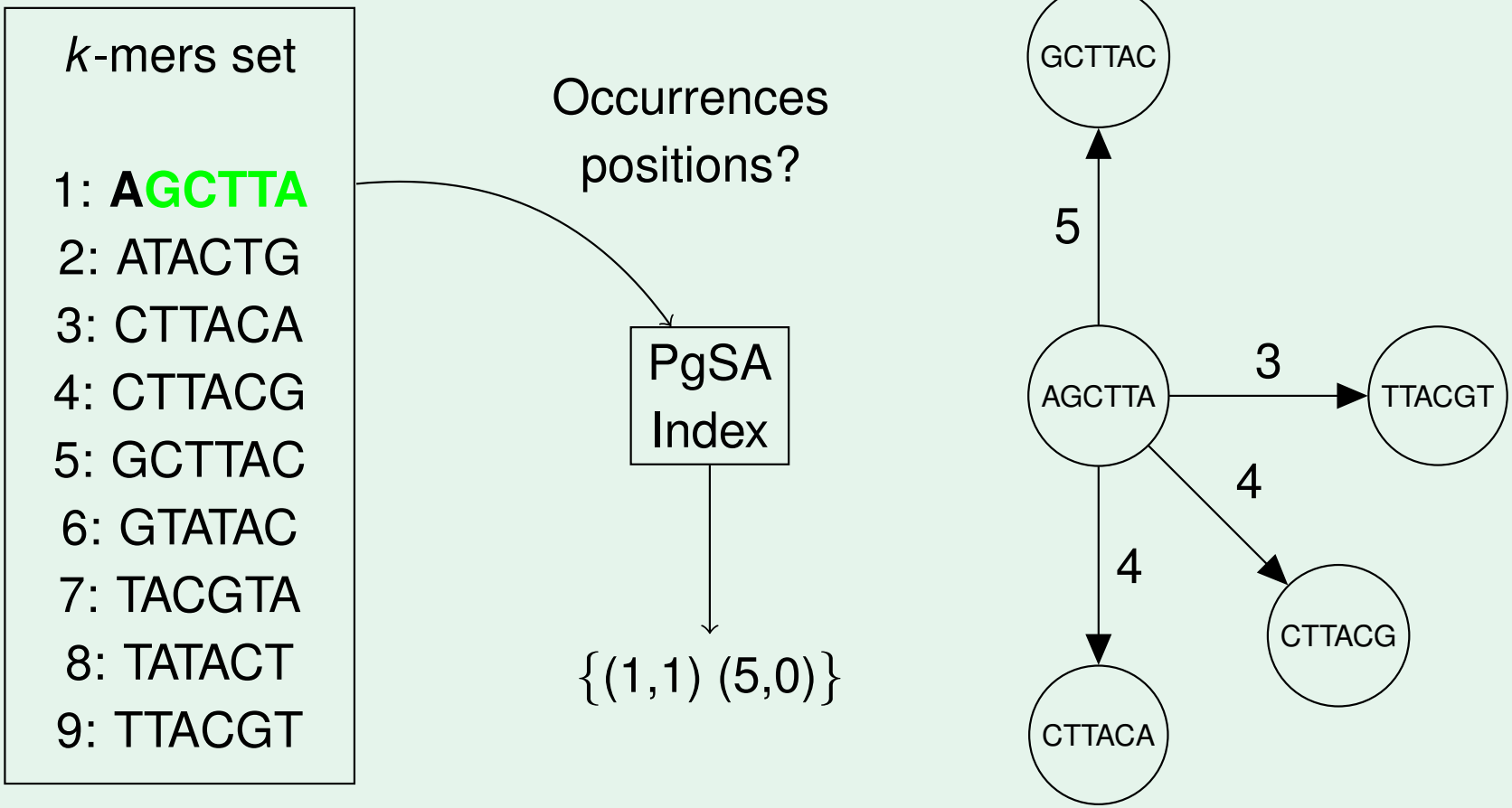
Occurrences positions?

PgSA Index



# Traversal of the enhanced de Bruijn graph

## Example



# Traversal of the enhanced de Bruijn graph

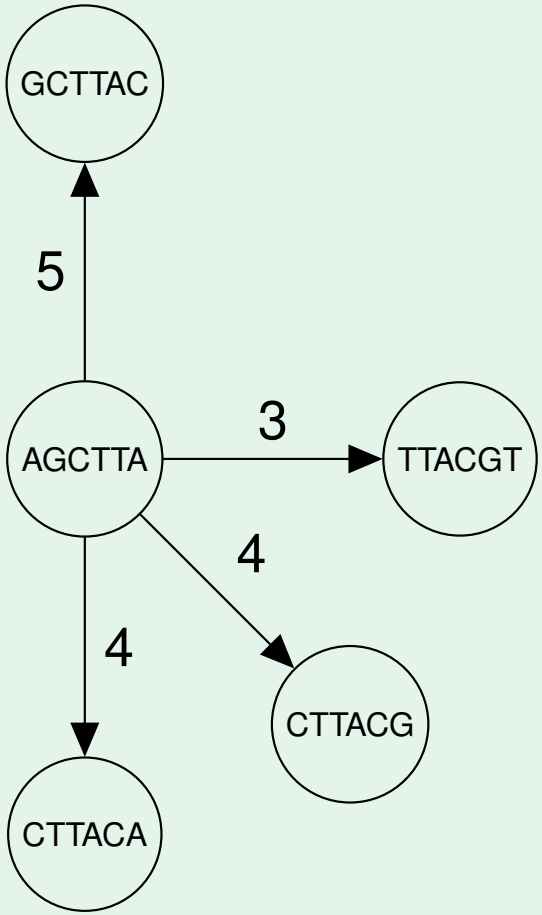
## Example

*k*-mers set

1: **A**GCTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,1) (5,0)}



# Traversal of the enhanced de Bruijn graph

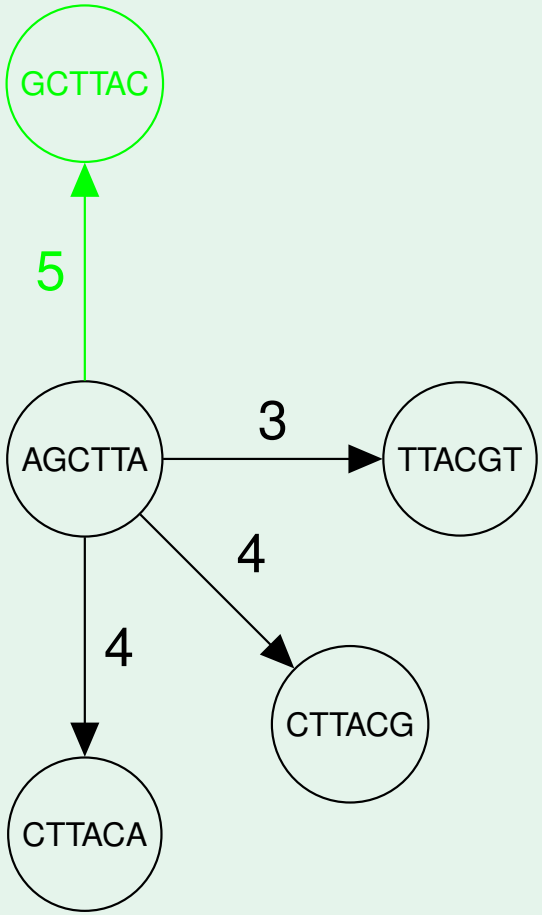
## Example

*k*-mers set

1: **A**GCTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,1) (5,0)}



# Traversal of the enhanced de Bruijn graph

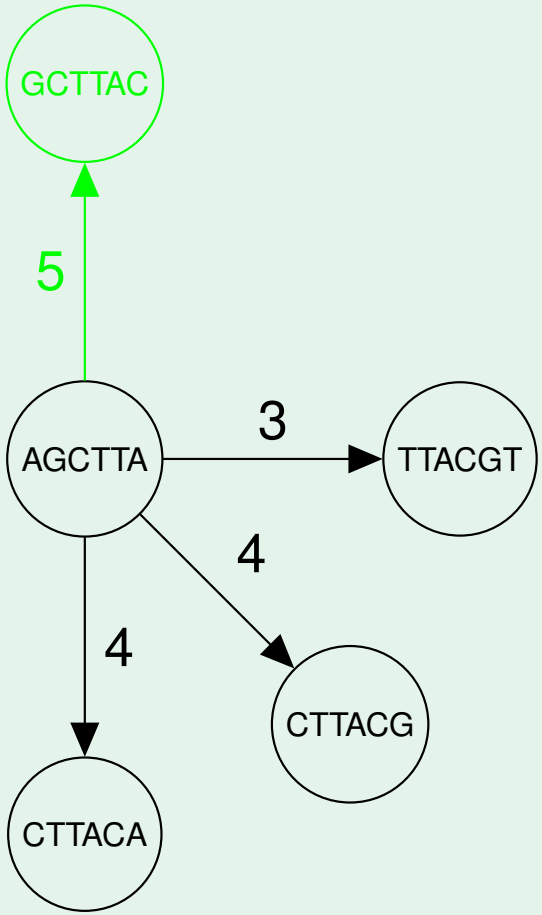
## Example

*k*-mers set

1: **AG**CTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

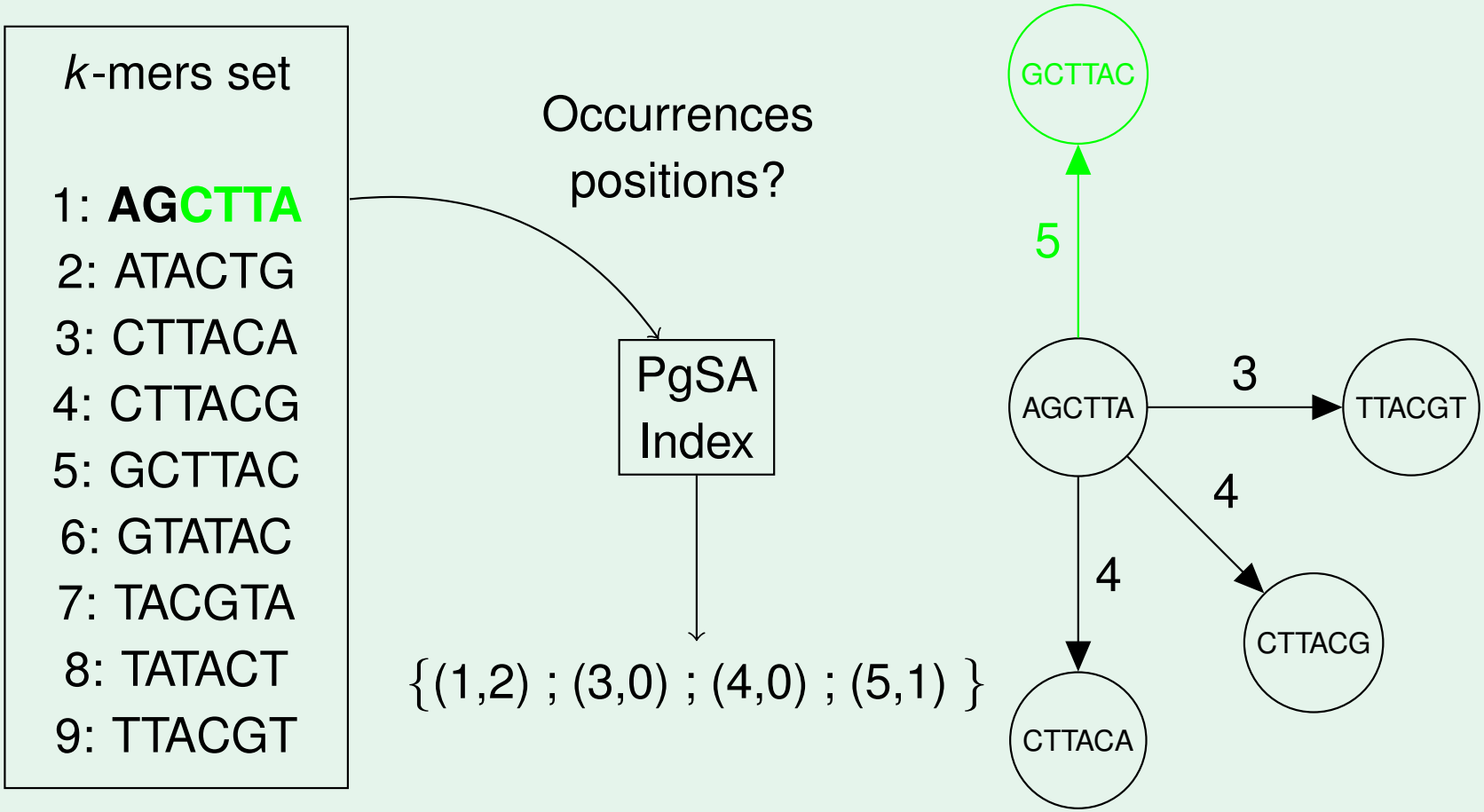
Occurrences positions?

PgSA Index



# Traversal of the enhanced de Bruijn graph

## Example



# Traversal of the enhanced de Bruijn graph

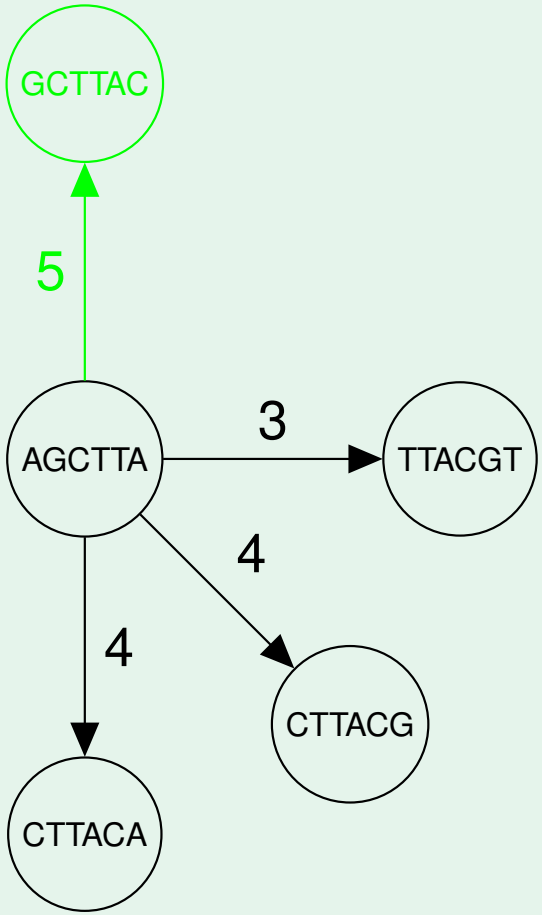
## Example

*k*-mers set

1: **AG**CTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,2) ; (3,0) ; (4,0) ; (5,1) }



# Traversal of the enhanced de Bruijn graph

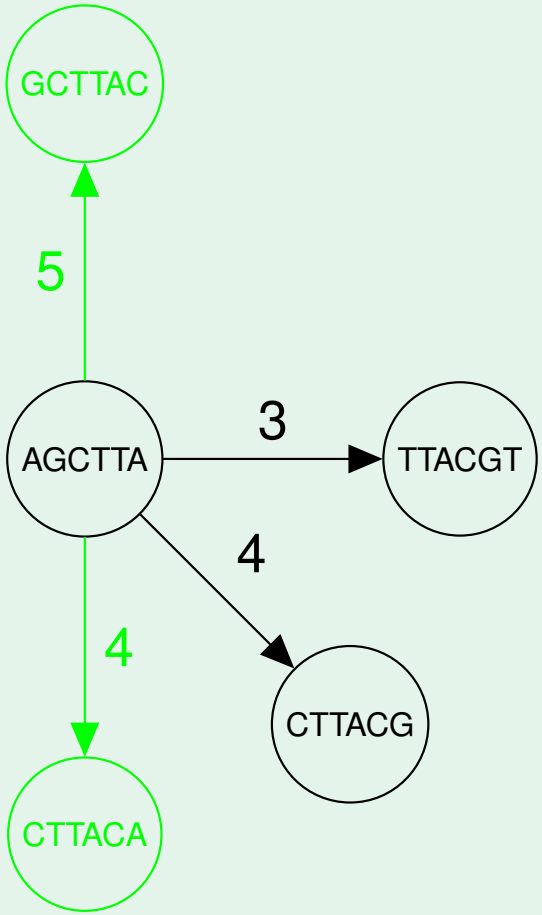
## Example

*k*-mers set

1: **AG**CTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,2) ; (3,0) ; (4,0) ; (5,1) }



# Traversal of the enhanced de Bruijn graph

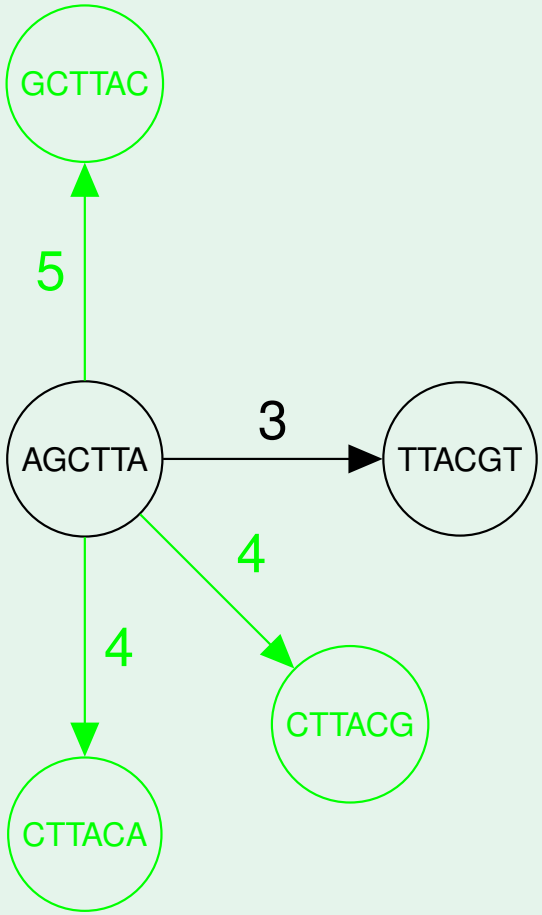
## Example

*k*-mers set

1: **AG**CTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,2) ; (3,0) ; (4,0) ; (5,1) }



# Traversal of the enhanced de Bruijn graph

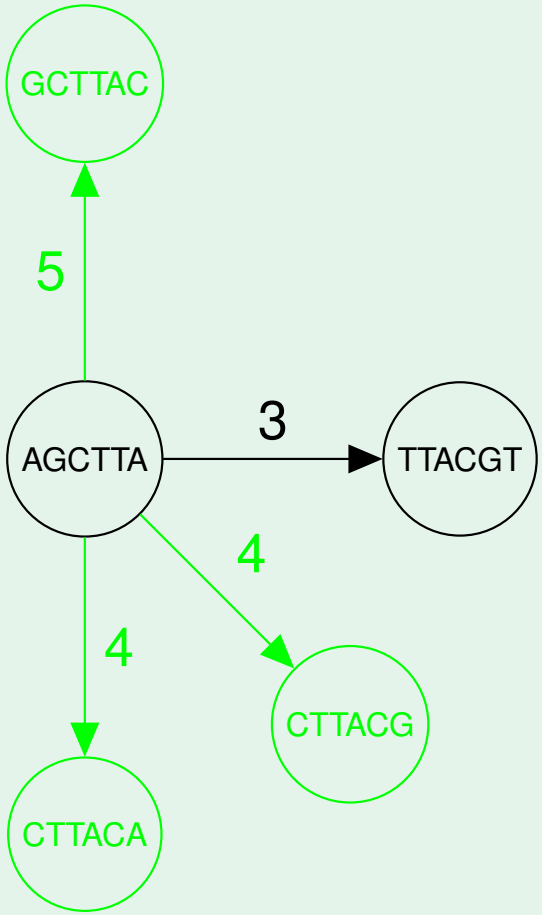
## Example

*k*-mers set

1: **AG**CTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,2) ; (3,0) ; (4,0) ; (5,1) }



# Traversal of the enhanced de Bruijn graph

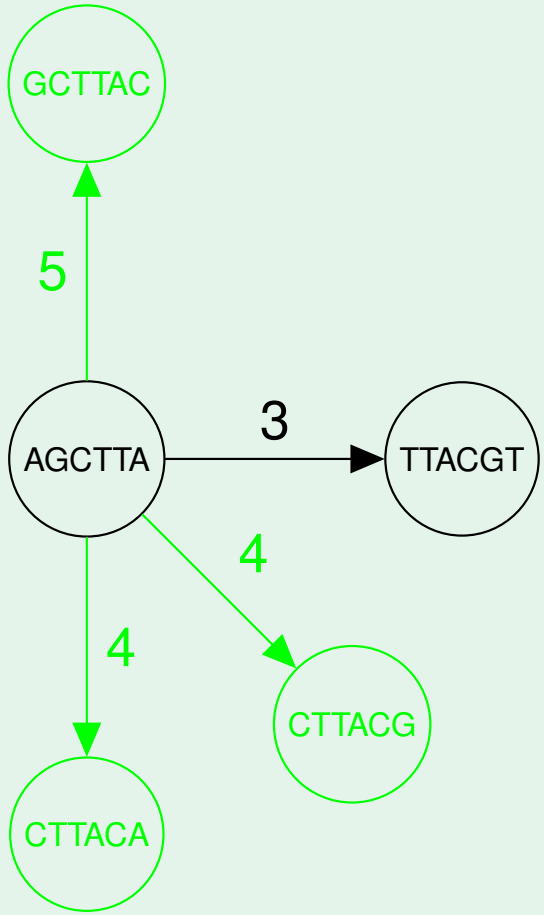
## Example

*k*-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

Occurrences positions?

PgSA Index



# Traversal of the enhanced de Bruijn graph

## Example

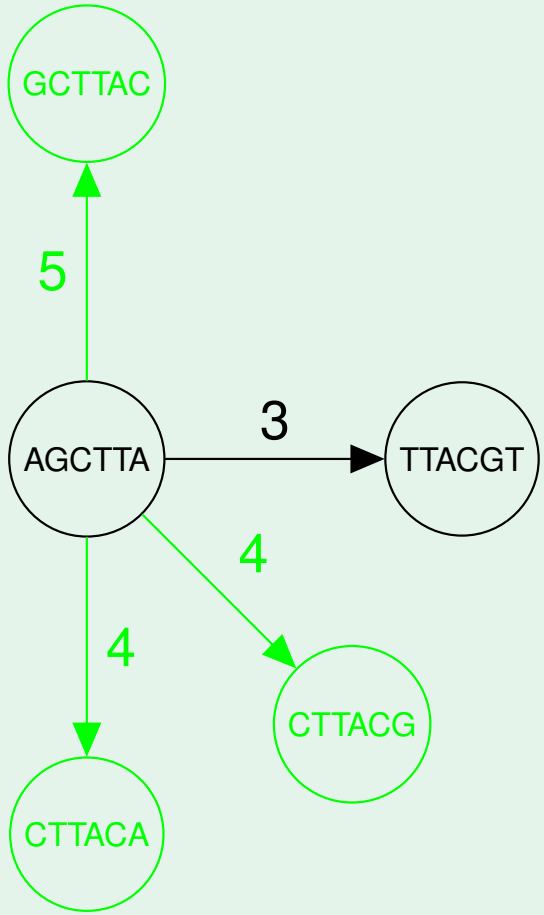
*k*-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

Occurrences positions?

PgSA Index

{(1,3) ; (3,1) ; (4,1) ; (5,2) ; (9,0)}



# Traversal of the enhanced de Bruijn graph

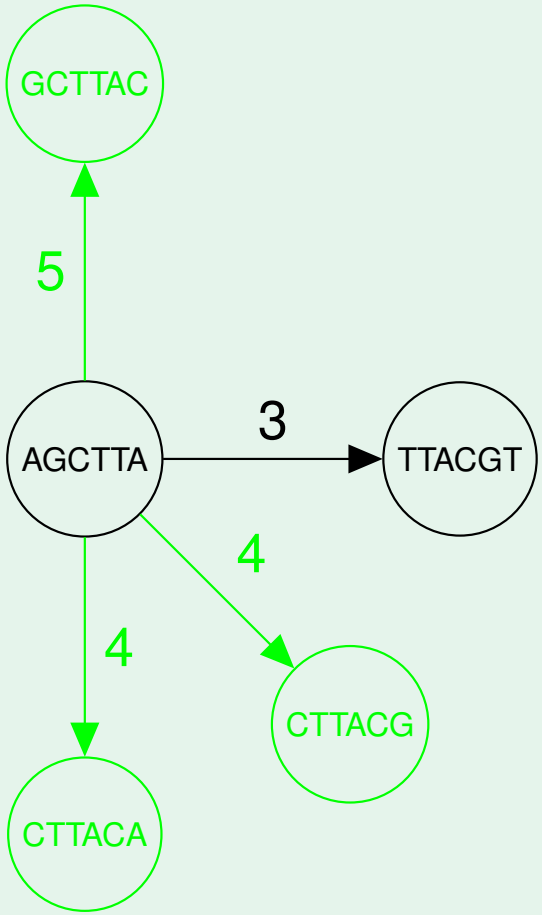
## Example

*k*-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,3) ; (3,1) ; (4,1) ; (5,2) ; (9,0)}



# Traversal of the enhanced de Bruijn graph

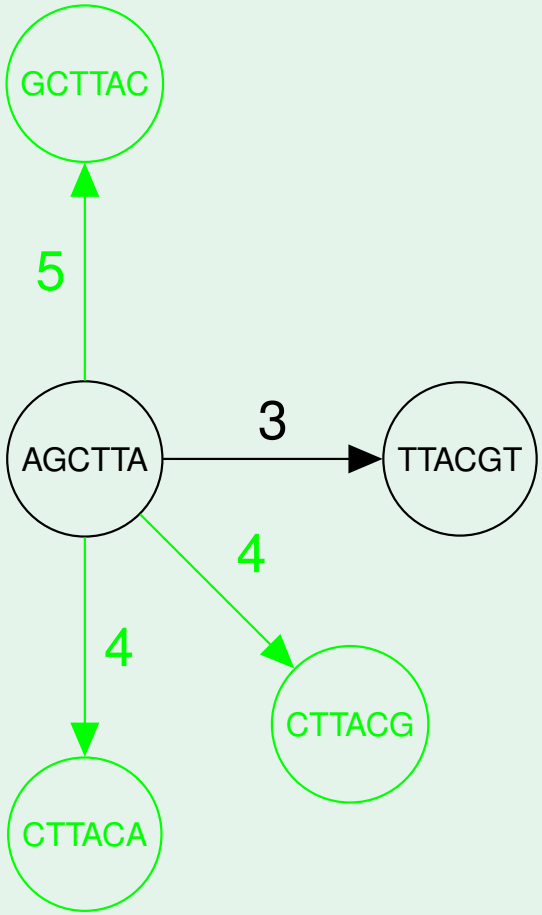
## Example

*k*-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,3) ; (3,1) ; (4,1) ; (5,2) ; (9,0)}



# Traversal of the enhanced de Bruijn graph

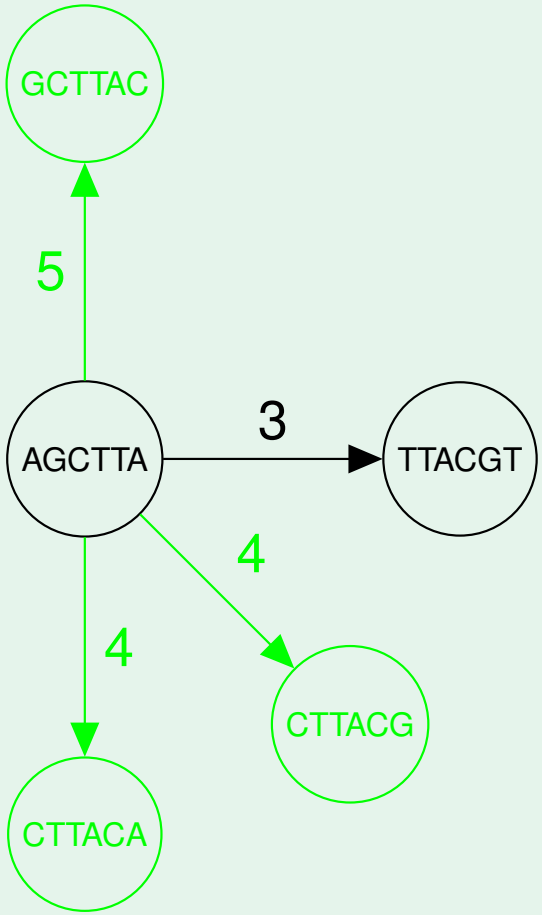
## Example

*k*-mers set

1: **AGC**TTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,3) ; (3,1) ; (4,1) ; (5,2) ; (9,0)}



# Traversal of the enhanced de Bruijn graph

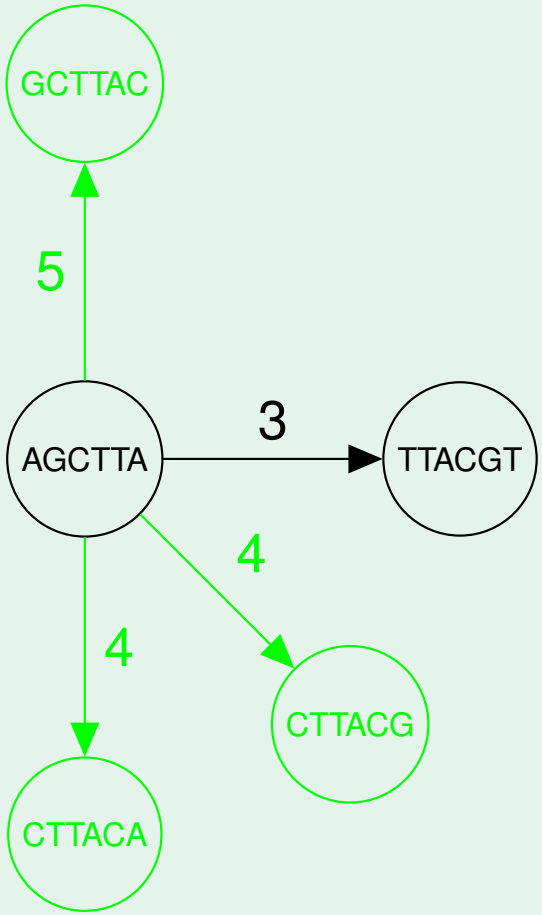
## Example

*k*-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,3) ; (3,1) ; (4,1) ; (5,2) ; (9,0)}



# Traversal of the enhanced de Bruijn graph

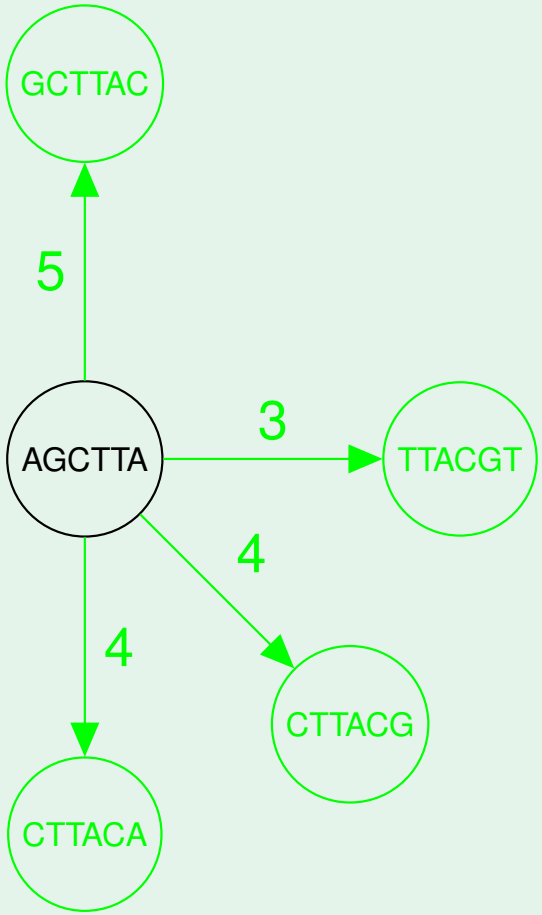
## Example

*k*-mers set

1: **AGC**TTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,3) ; (3,1) ; (4,1) ; (5,2) ; (9,0)}



# Context

- Due to their high error rate, error correction of long reads is mandatory
- Various methods already exist for the correction of short reads, but are not applicable to long reads
- Forces the development of new error correction methods
- Two main categories: self-correction and hybrid correction

# Context

- Due to their high error rate, error correction of long reads is mandatory
- Various methods already exist for the correction of short reads, but are not applicable to long reads
- Forces the development of new error correction methods
- Two main categories: self-correction and hybrid correction

# Context

- Due to their high error rate, error correction of long reads is mandatory
- Various methods already exist for the correction of short reads, but are not applicable to long reads
- Forces the development of new error correction methods
- Two main categories: self-correction and hybrid correction

# Context

- Due to their high error rate, error correction of long reads is mandatory
- Various methods already exist for the correction of short reads, but are not applicable to long reads
- Forces the development of new error correction methods
- Two main categories: self-correction and hybrid correction

# Workflow

5 steps:

1. Correct the short reads
2. Align the short reads on the long reads, to find seeds
3. Merge the overlapping seeds
4. Link the seeds, by traversing the enhanced de Bruijn graph
5. Extend the obtained corrected long read, on the left (resp. right) of the leftmost (resp. rightmost) seed

# Workflow

5 steps:

1. Correct the short reads
2. Align the short reads on the long reads, to find seeds
3. Merge the overlapping seeds
4. Link the seeds, by traversing the enhanced de Bruijn graph
5. Extend the obtained corrected long read, on the left (resp. right) of the leftmost (resp. rightmost) seed

# Workflow

5 steps:

1. Correct the short reads
2. Align the short reads on the long reads, to find seeds
3. Merge the overlapping seeds
4. Link the seeds, by traversing the enhanced de Bruijn graph
5. Extend the obtained corrected long read, on the left (resp. right) of the leftmost (resp. rightmost) seed

# Workflow

5 steps:

1. Correct the short reads
2. Align the short reads on the long reads, to find seeds
3. Merge the overlapping seeds
4. Link the seeds, by traversing the enhanced de Bruijn graph
5. Extend the obtained corrected long read, on the left (resp. right) of the leftmost (resp. rightmost) seed

# Workflow

5 steps:

1. Correct the short reads
2. Align the short reads on the long reads, to find seeds
3. Merge the overlapping seeds
4. Link the seeds, by traversing the enhanced de Bruijn graph
5. Extend the obtained corrected long read, on the left (resp. right) of the leftmost (resp. rightmost) seed

# Step 4: Seeds linking

- Seeds are used as anchor points on the enhanced de Bruijn graph
- The graph is traversed to link together the seeds and assemble the $k$-mers

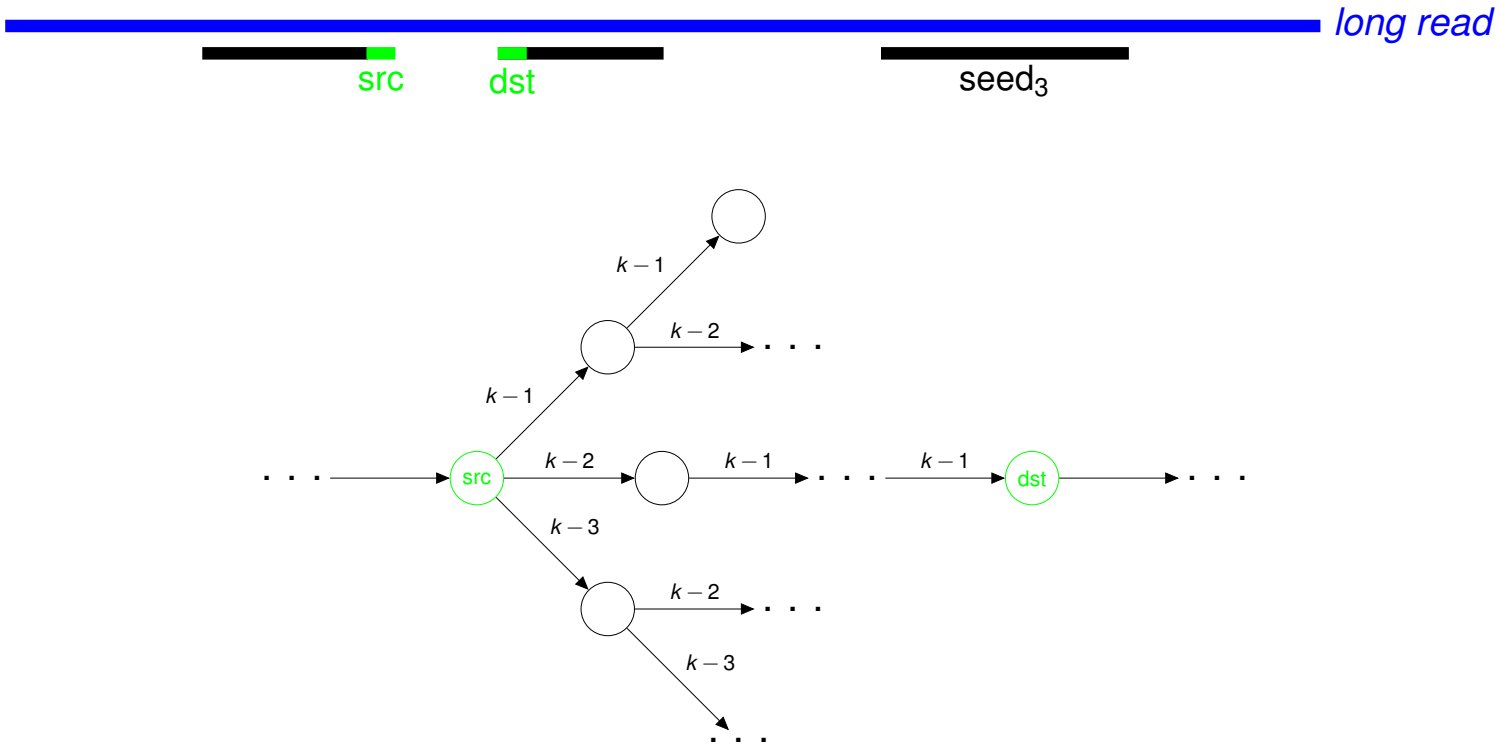
# Step 4: Seeds linking

- Seeds are used as anchor points on the enhanced de Bruijn graph
- The graph is traversed to link together the seeds and assemble the $k$-mers

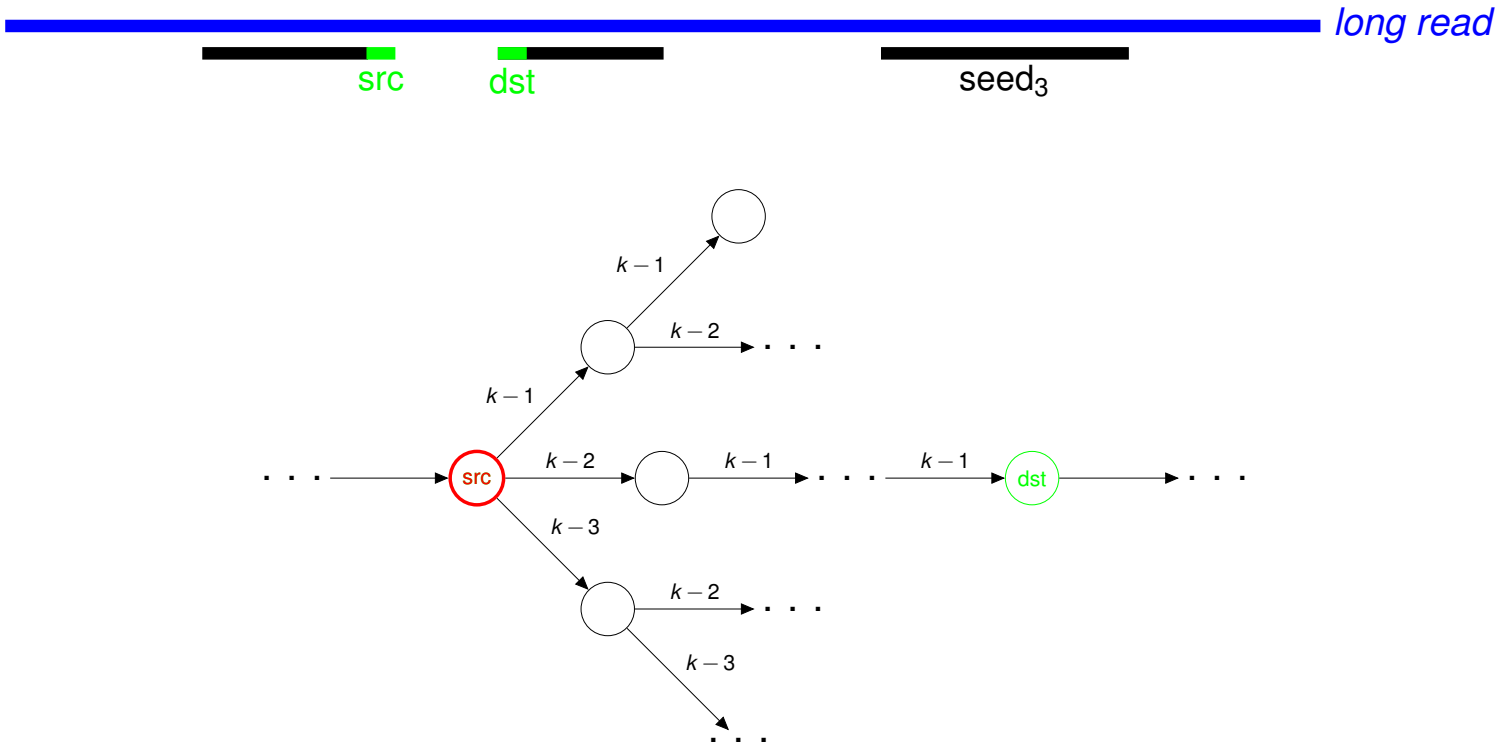
# Step 4: Seeds linking

# Step 4: Seeds linking

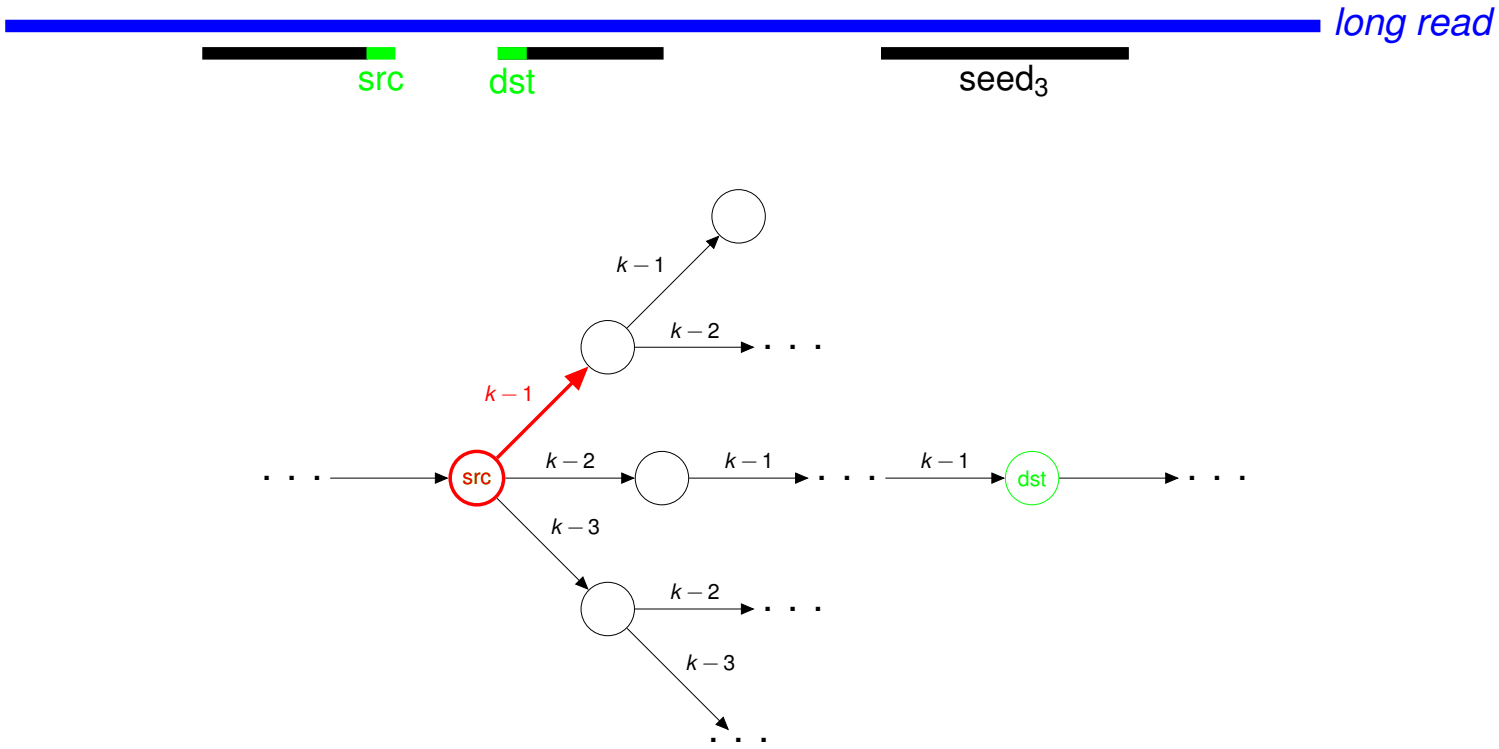
# Step 4: Seeds linking



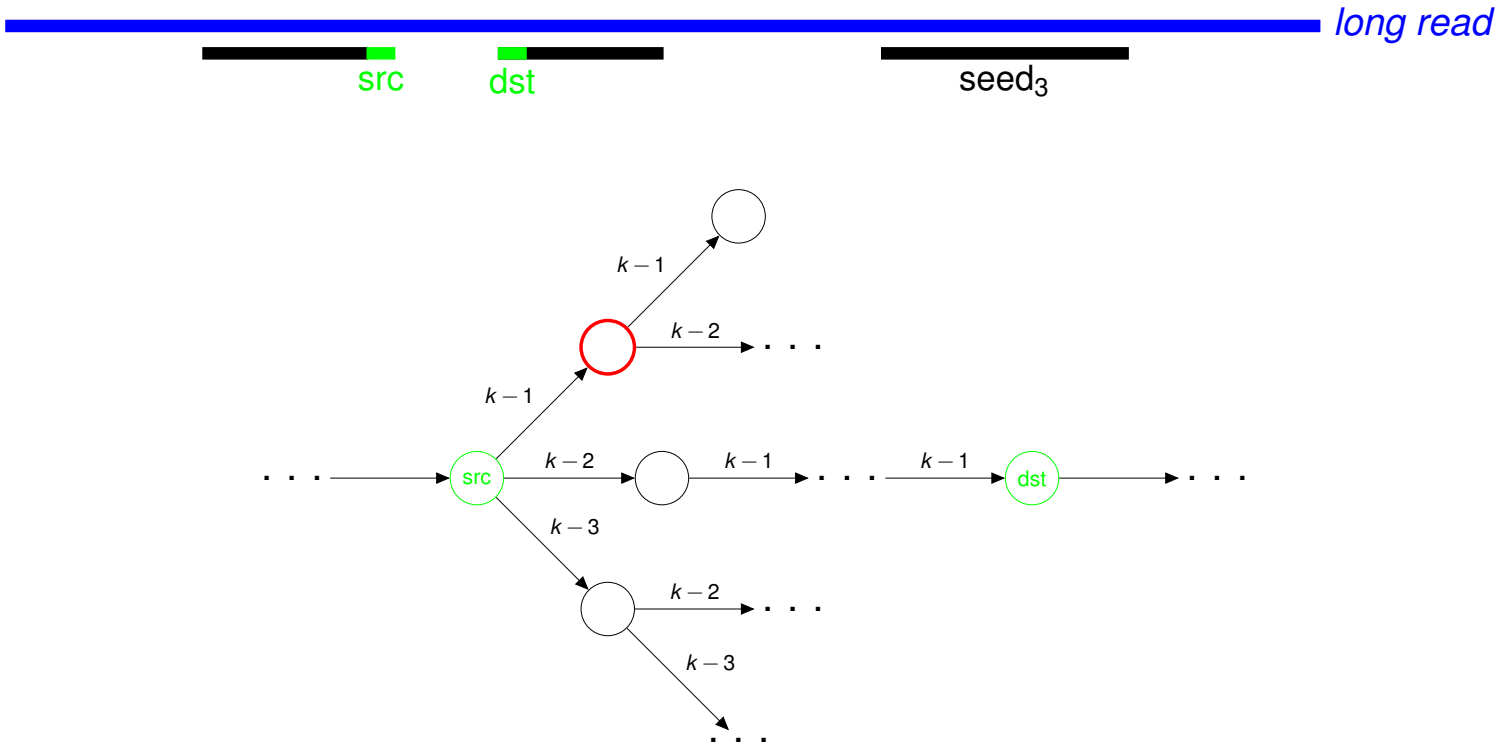
# Step 4: Seeds linking



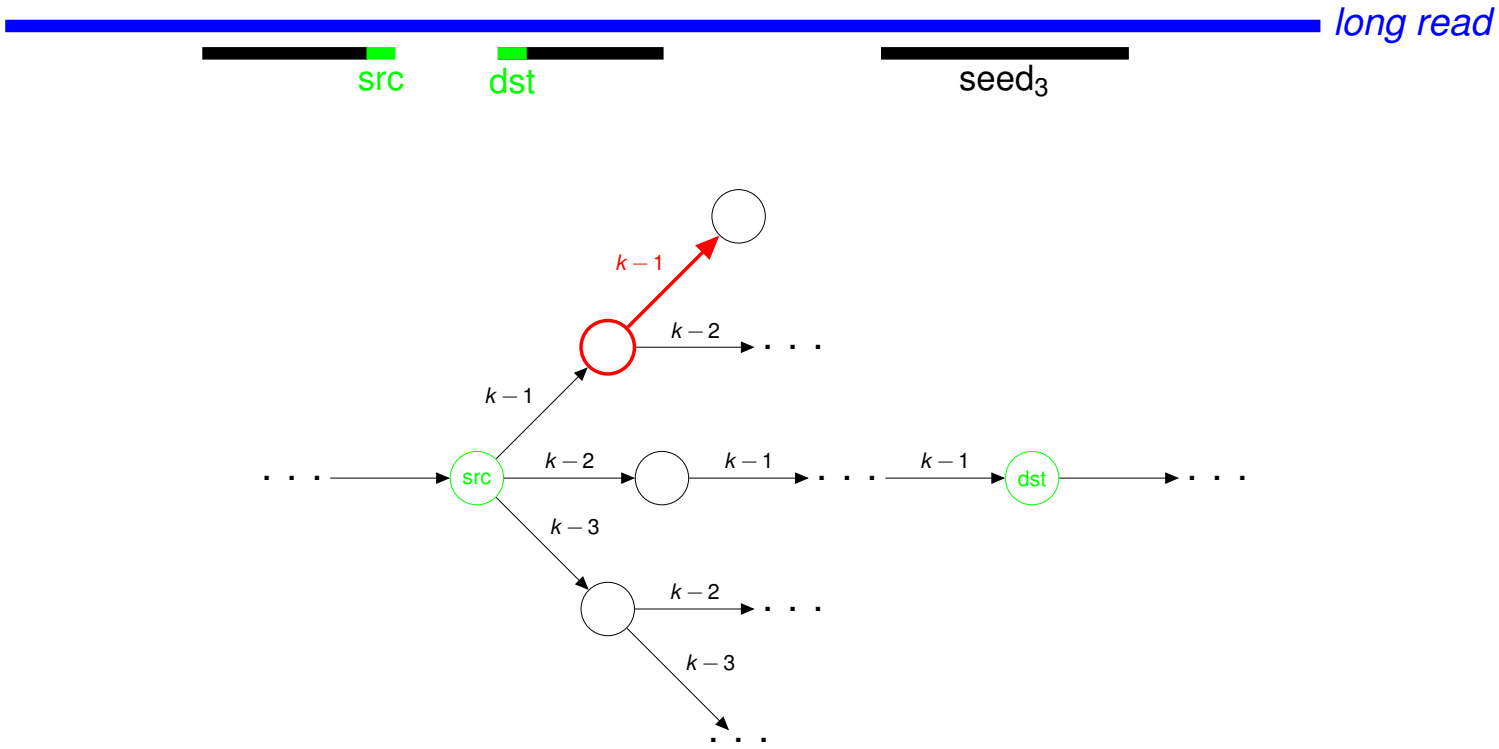
# Step 4: Seeds linking



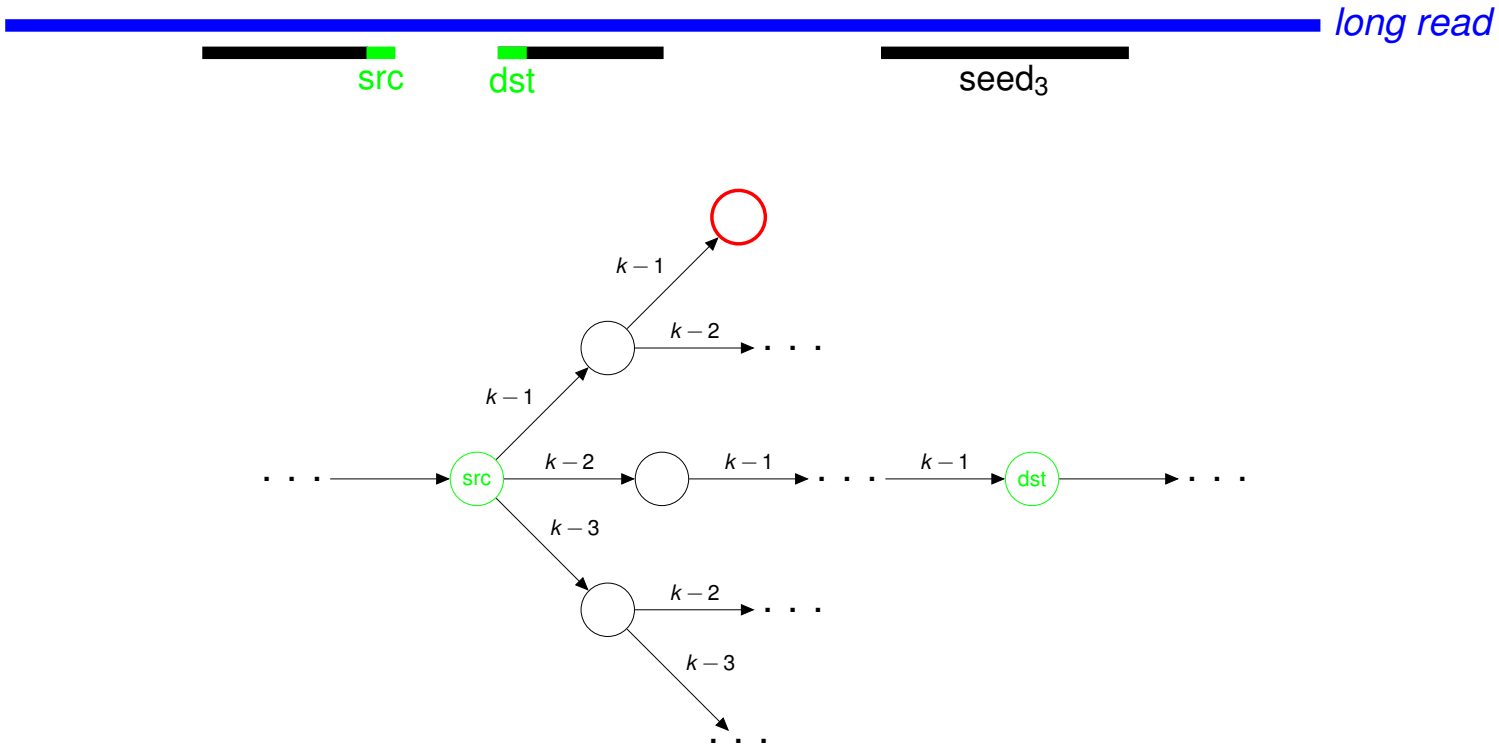
# Step 4: Seeds linking



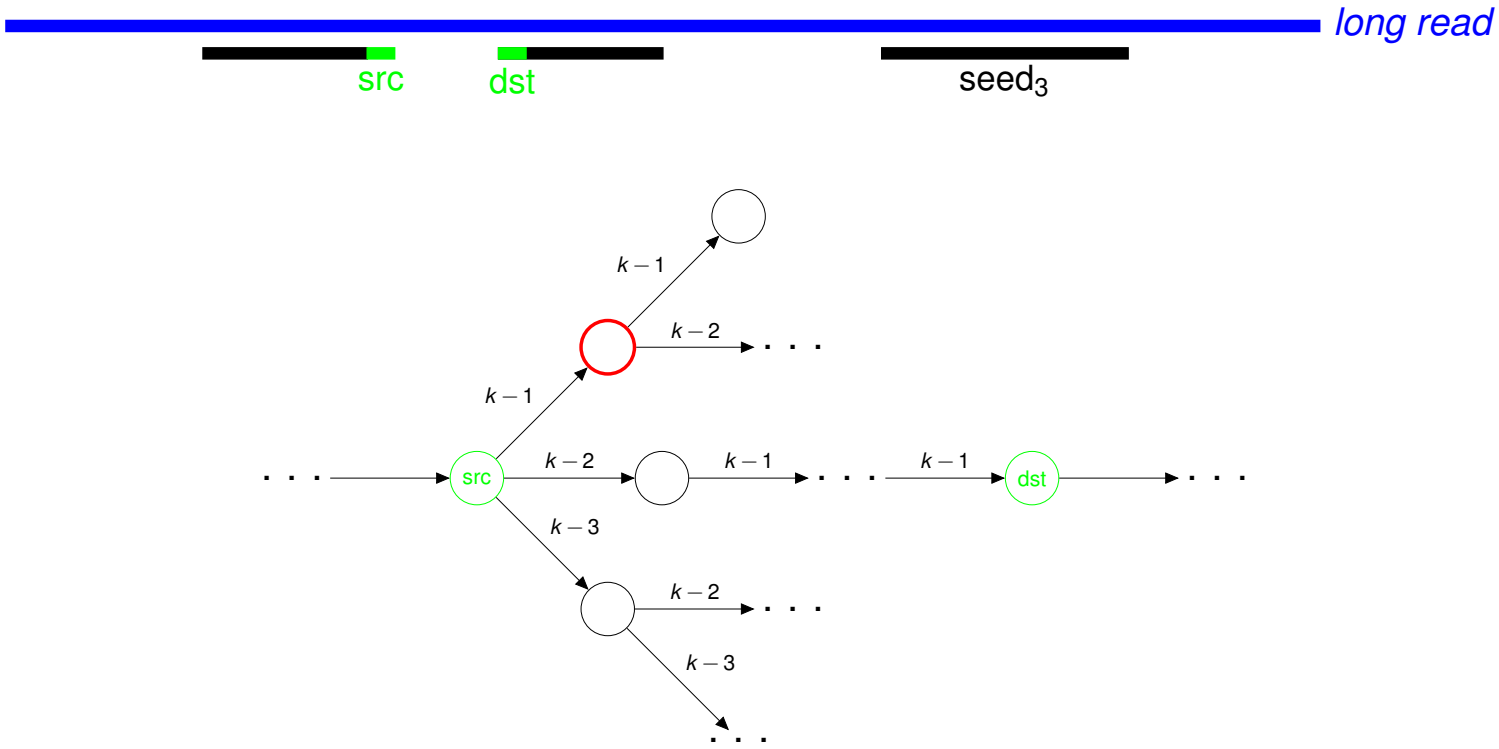
# Step 4: Seeds linking



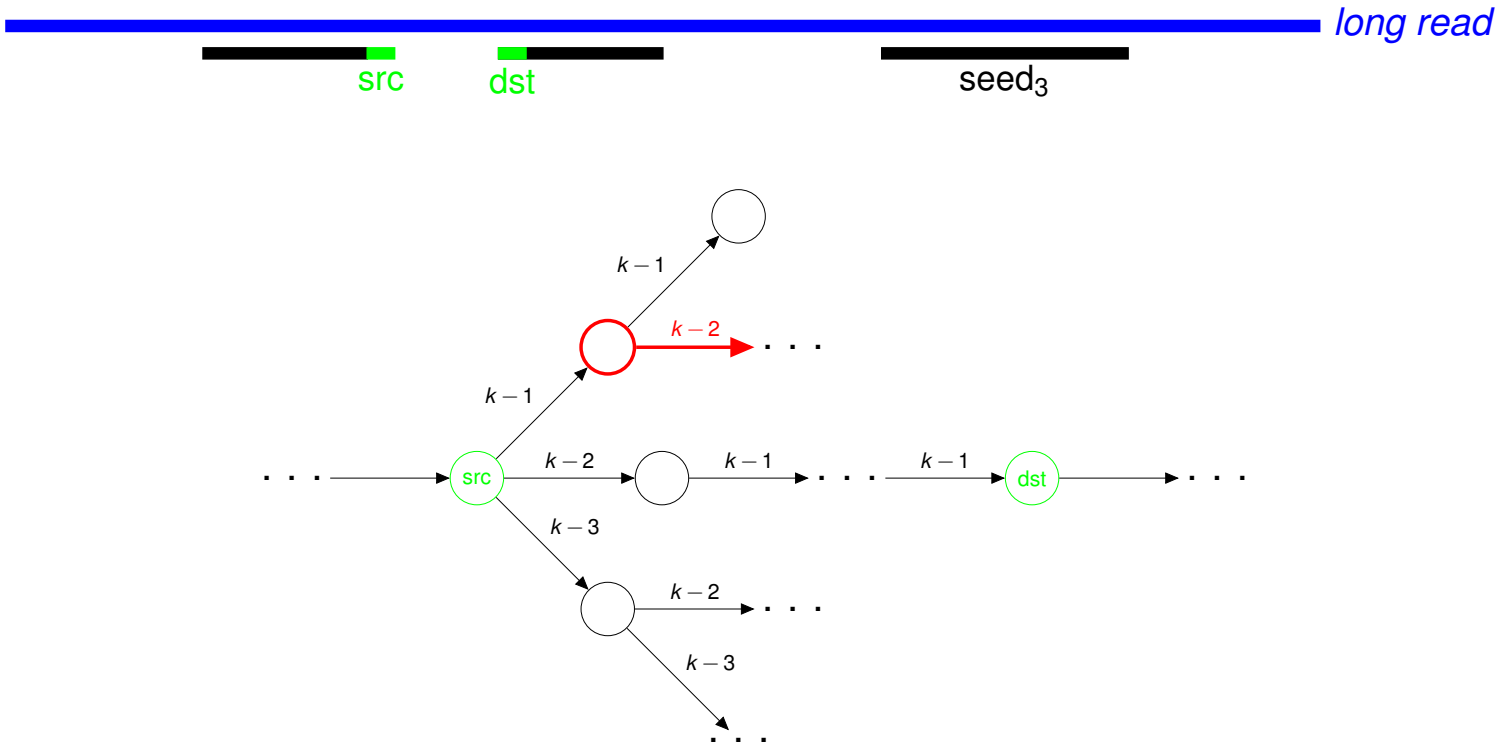
# Step 4: Seeds linking



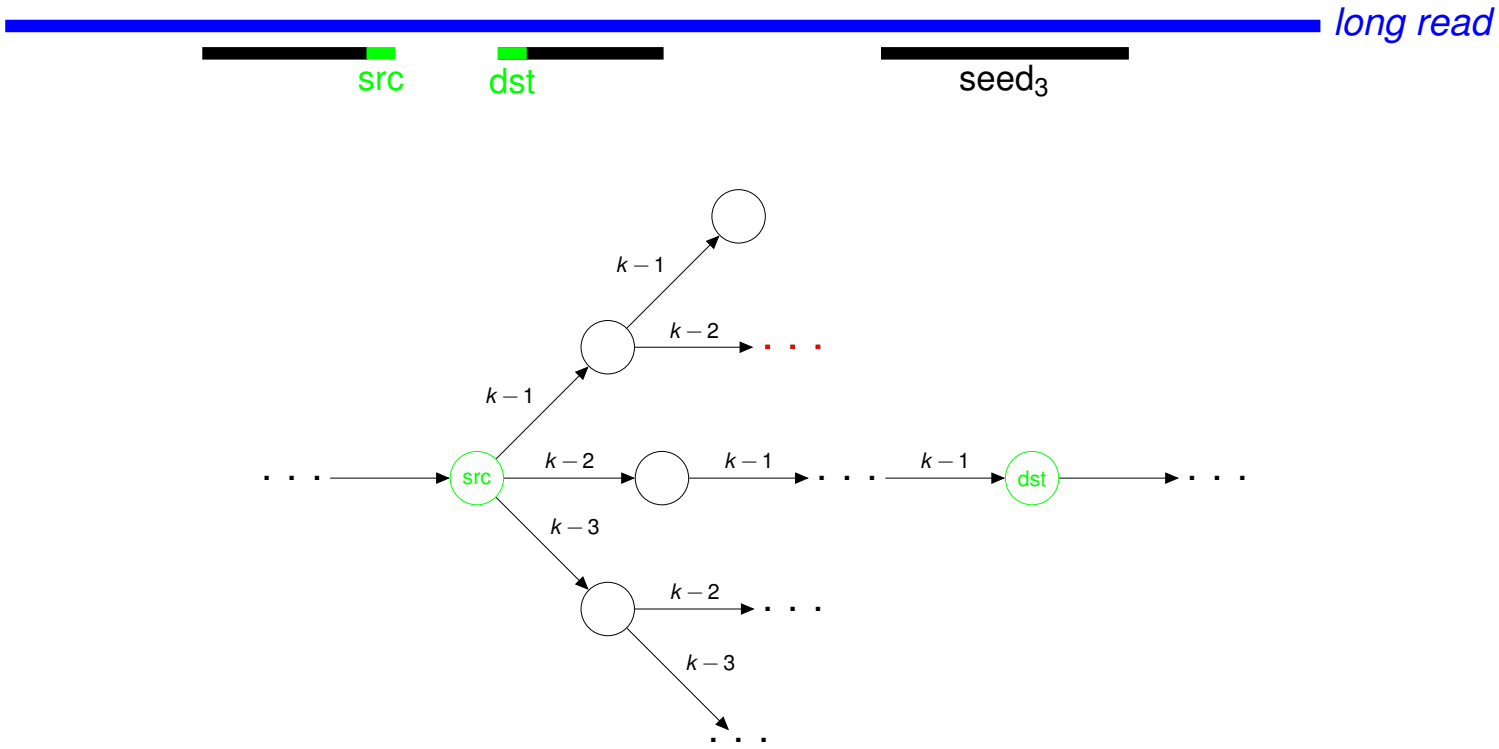
# Step 4: Seeds linking



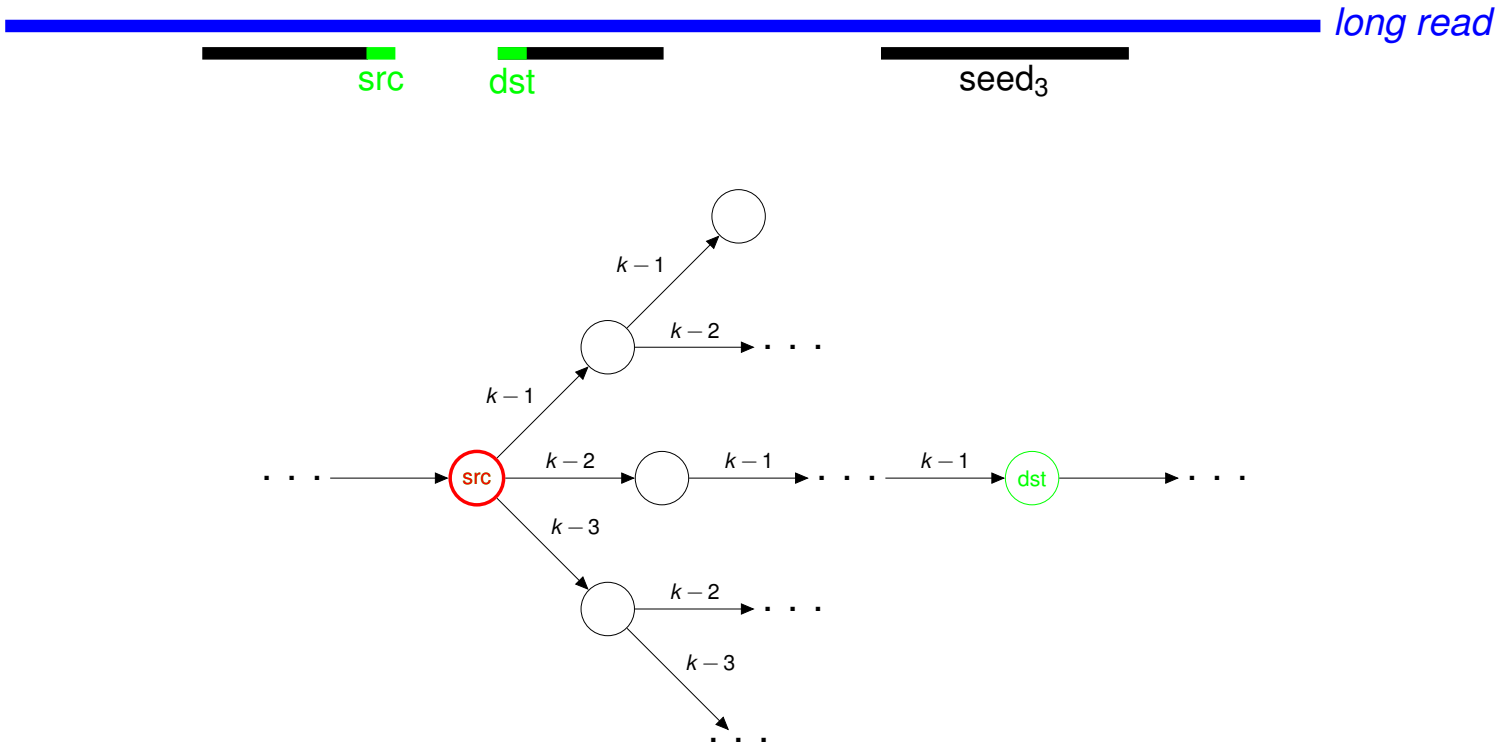
# Step 4: Seeds linking



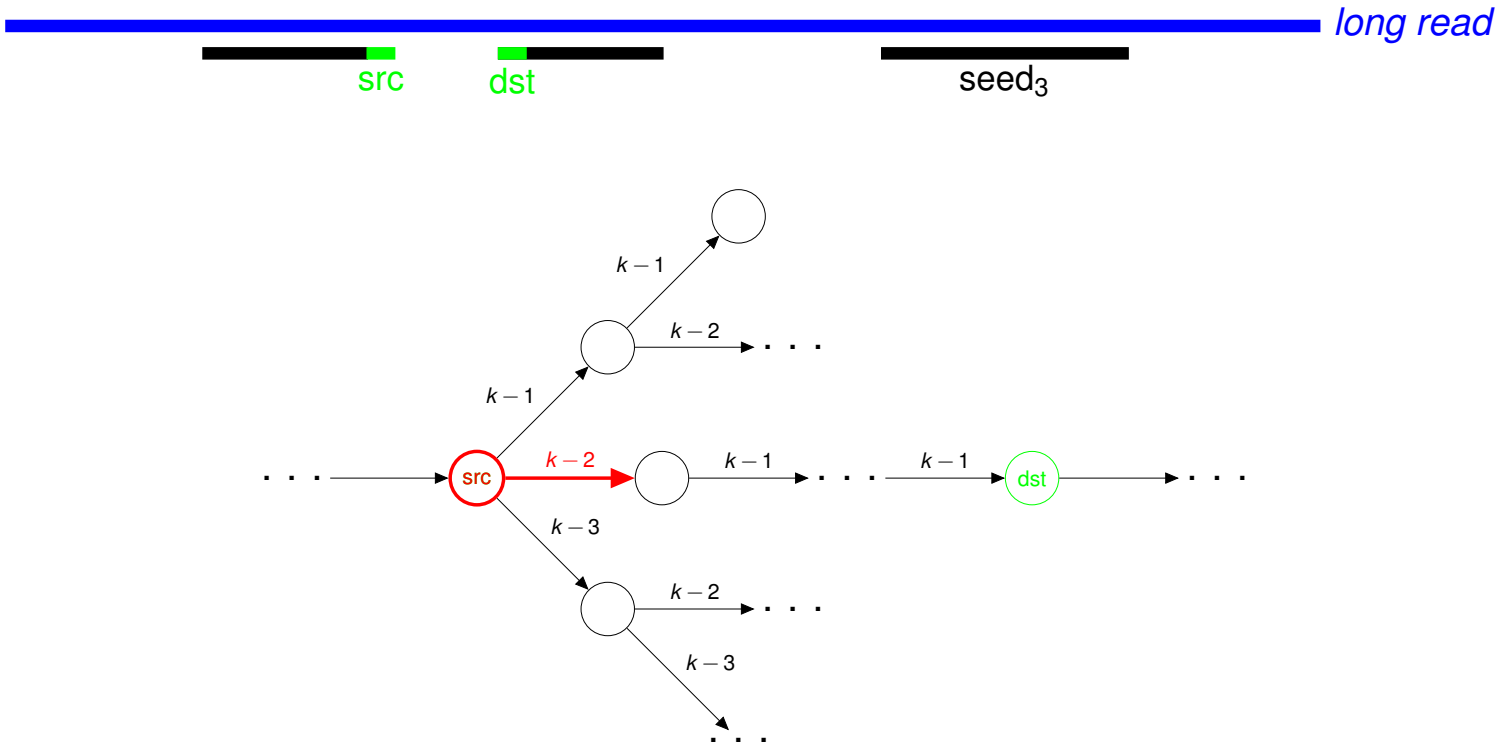
# Step 4: Seeds linking



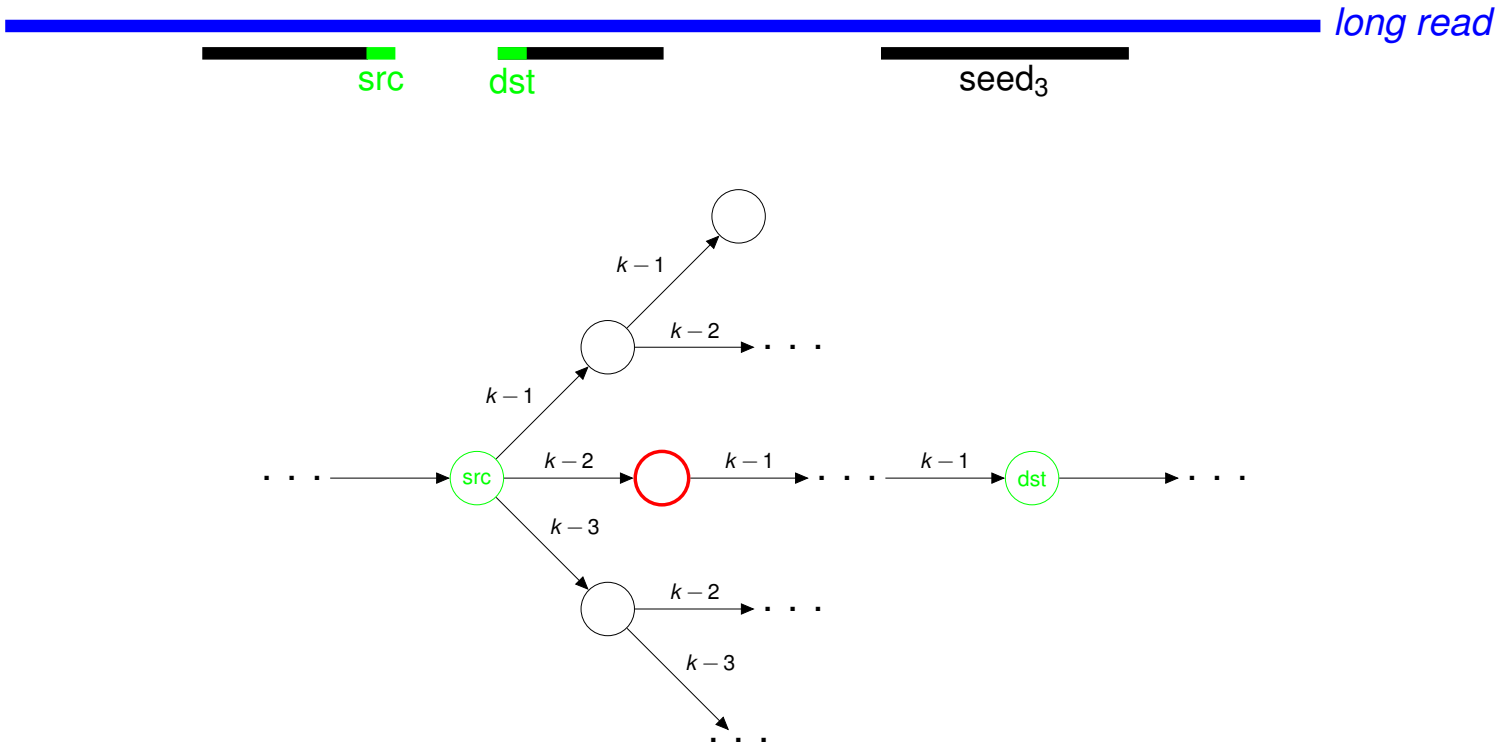
# Step 4: Seeds linking



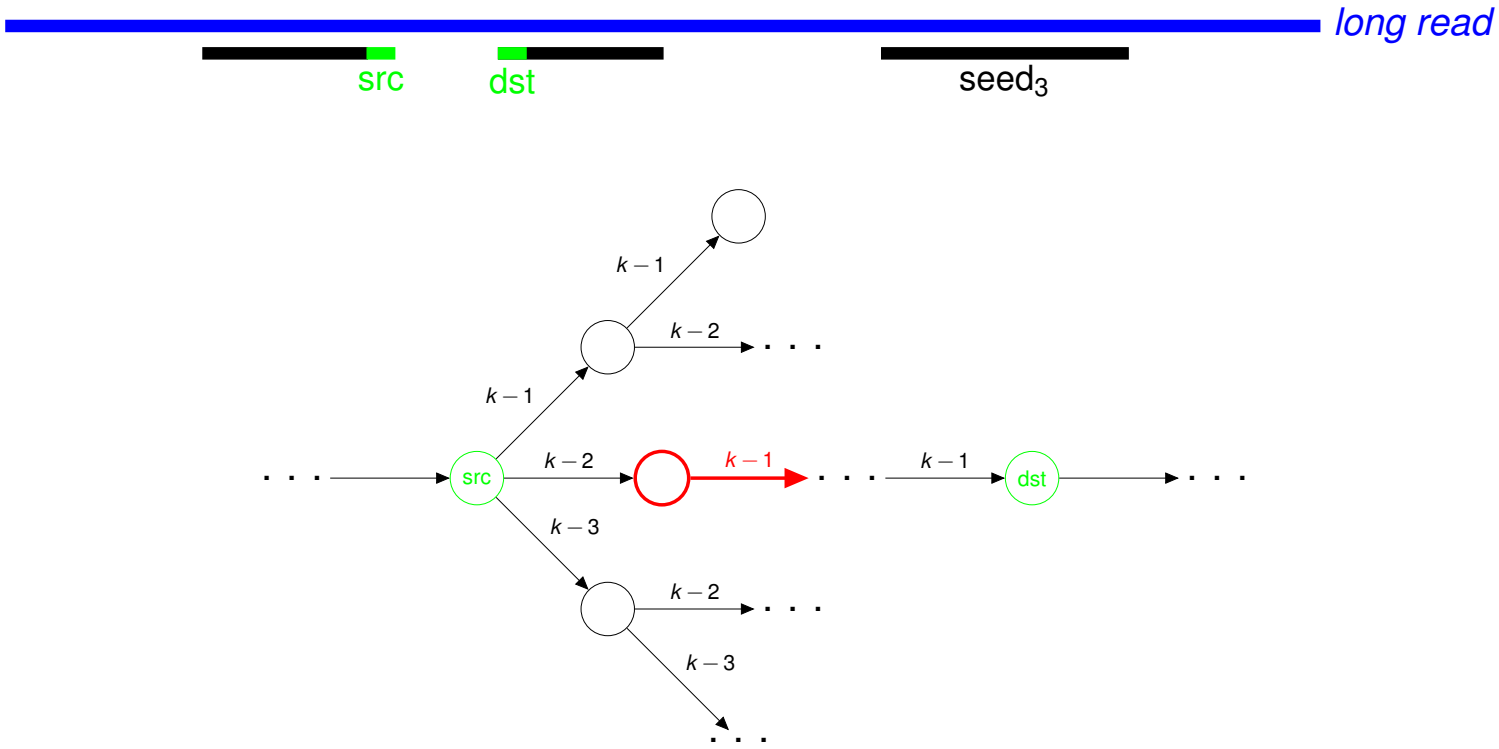
# Step 4: Seeds linking



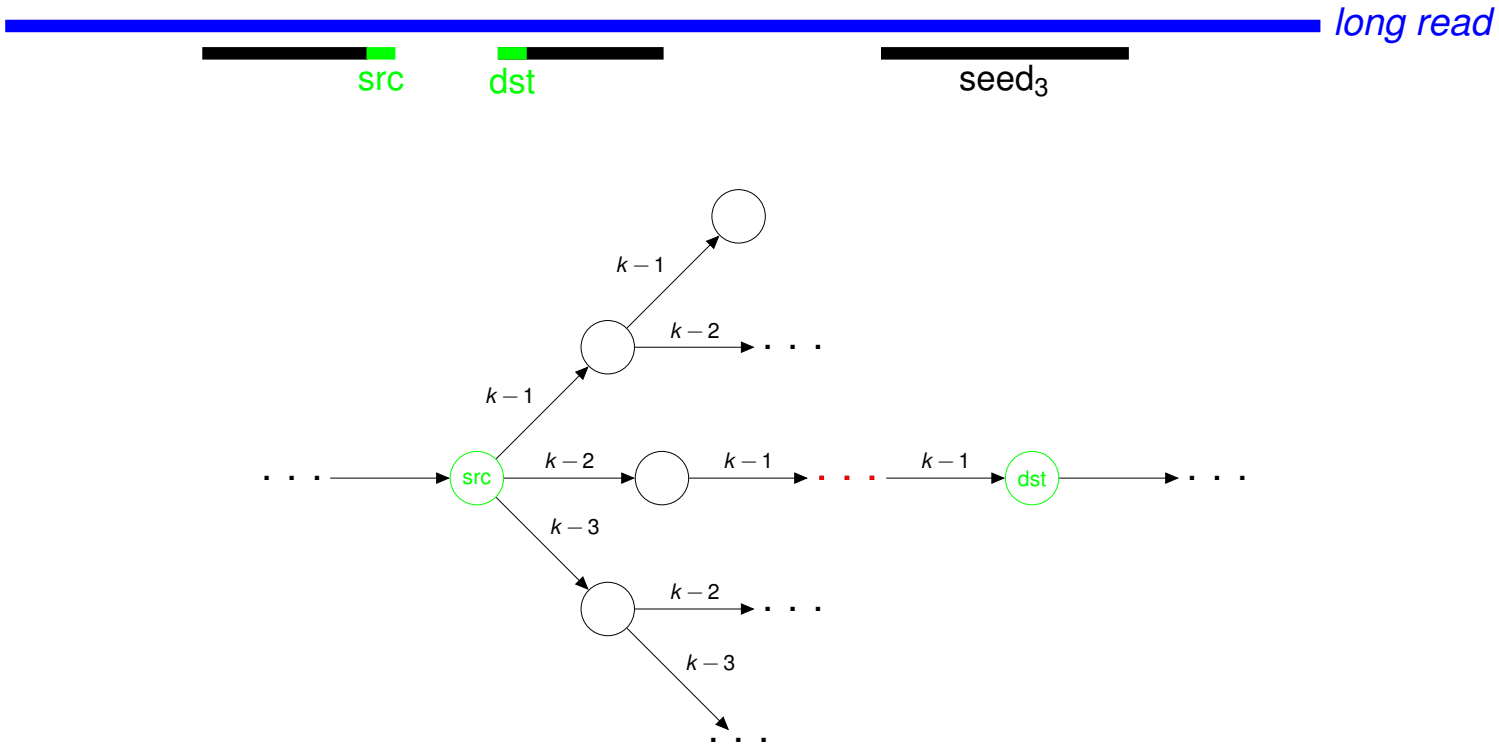
# Step 4: Seeds linking



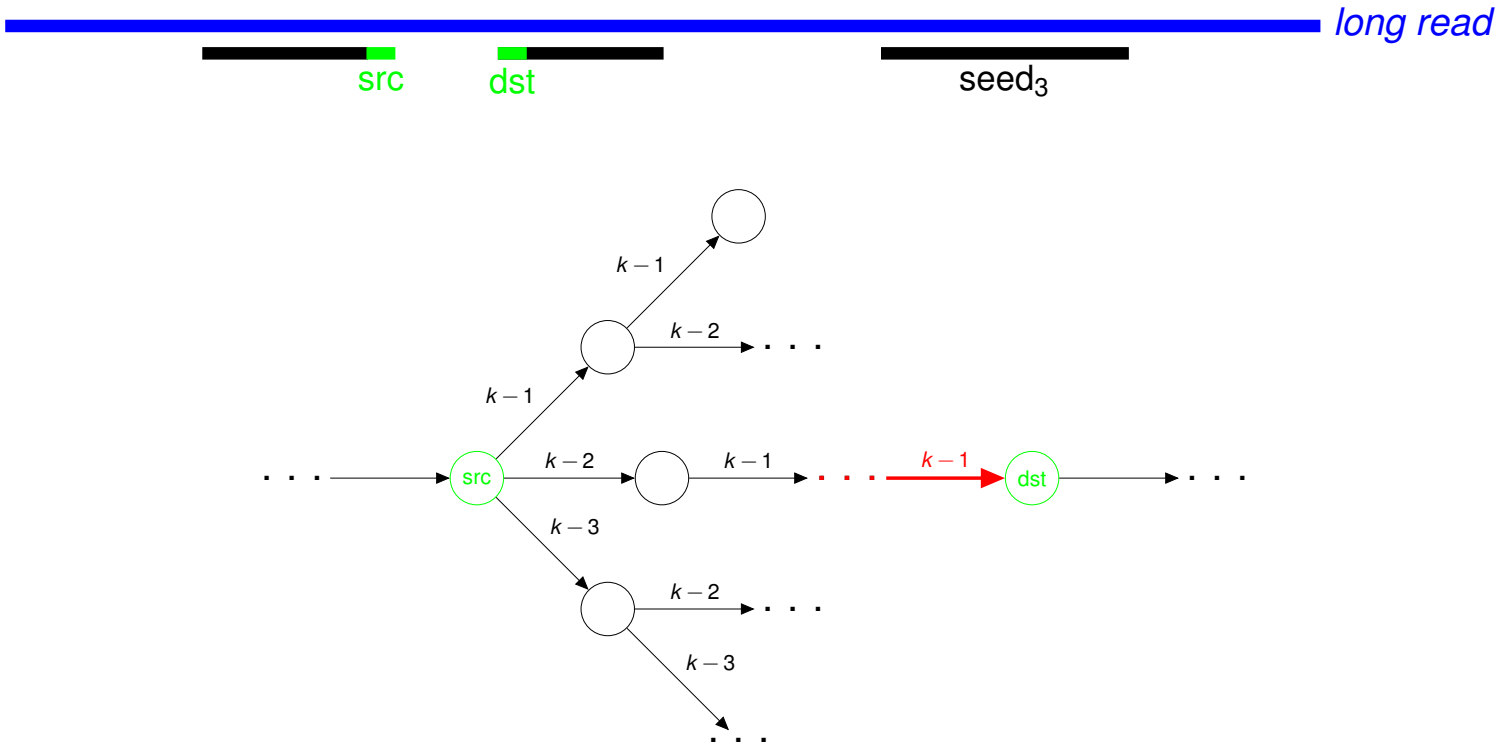
# Step 4: Seeds linking



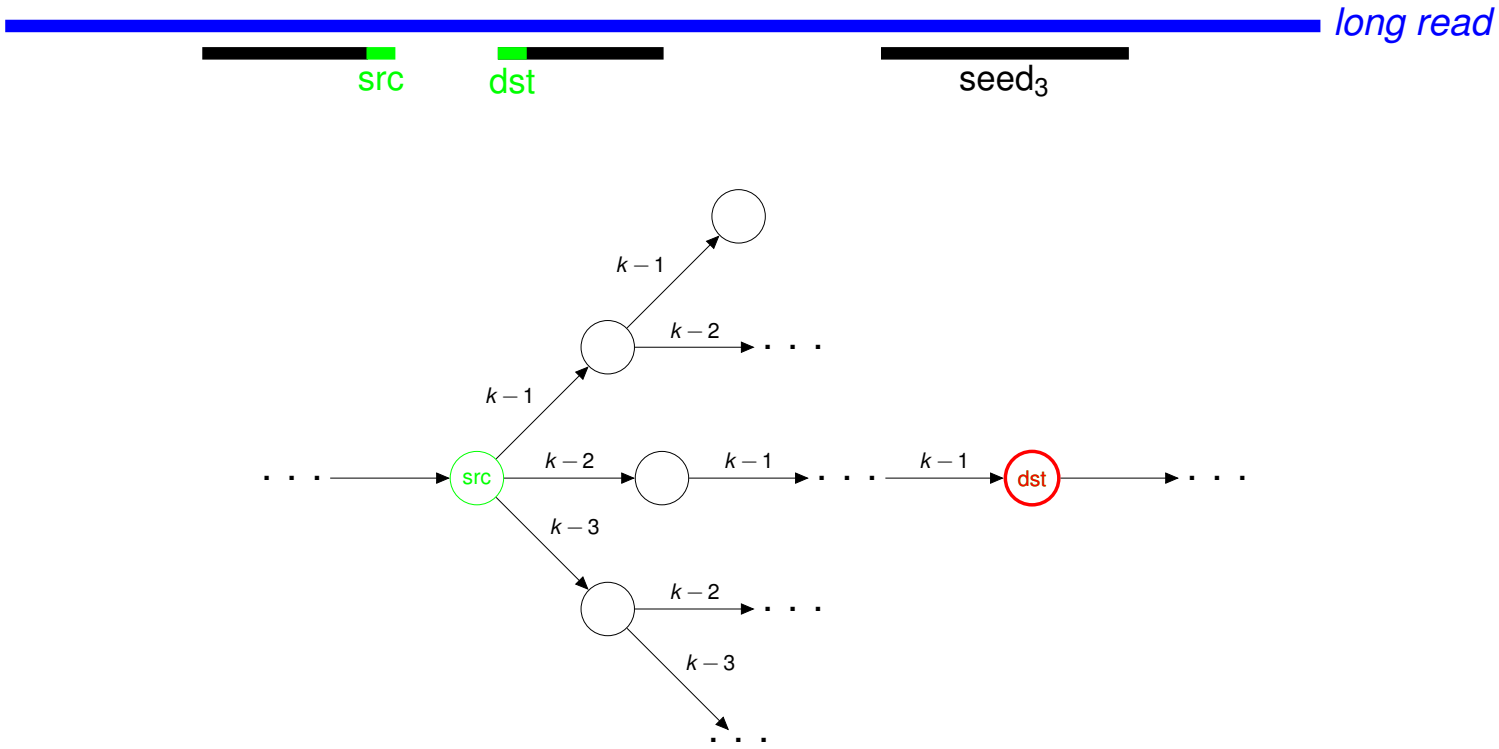
# Step 4: Seeds linking



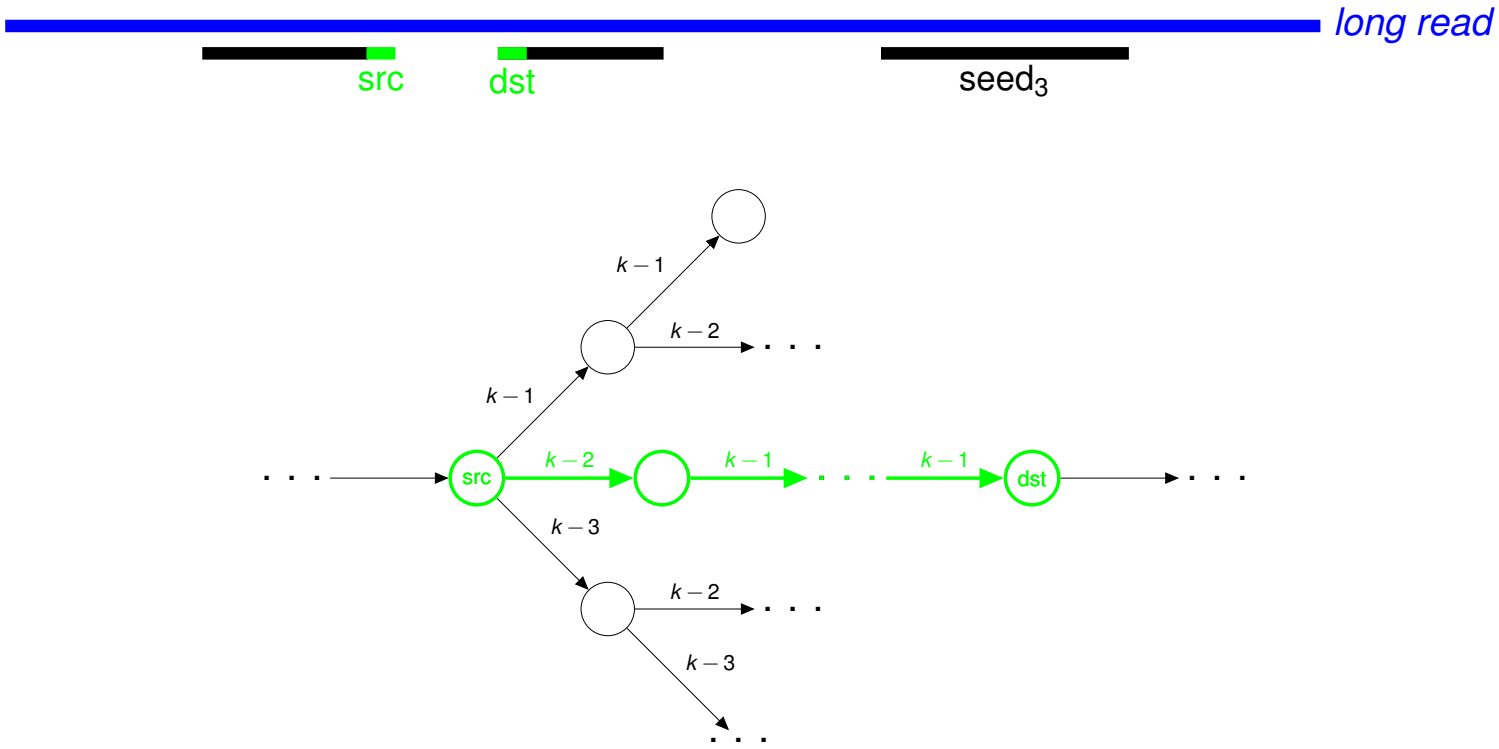
# Step 4: Seeds linking



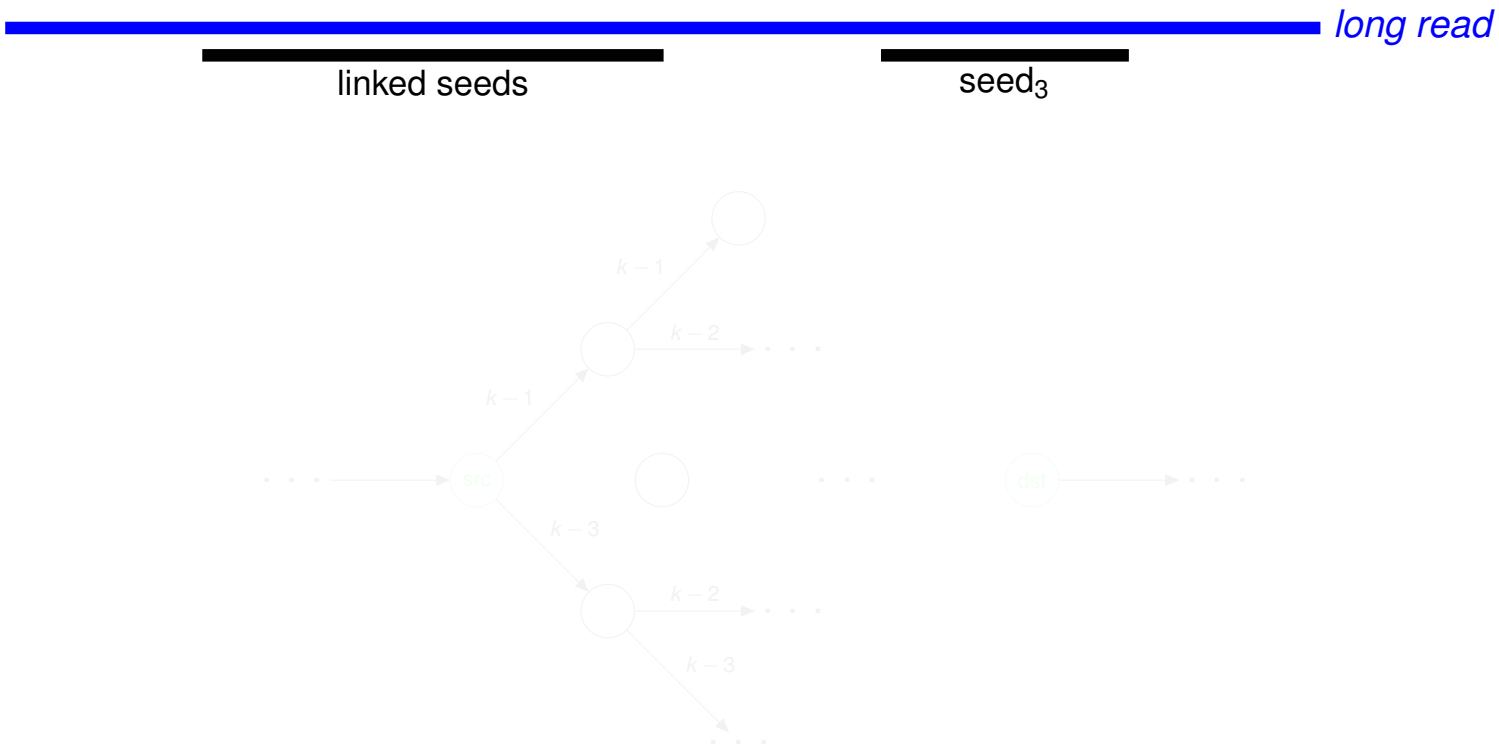
# Step 4: Seeds linking



# Step 4: Seeds linking

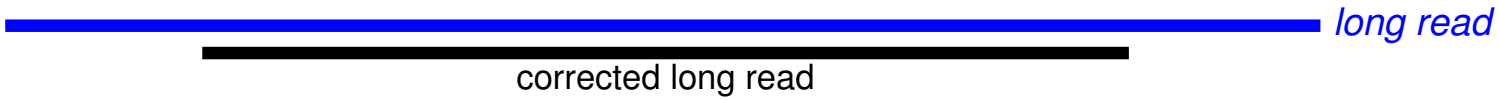


# Step 4: Seeds linking



# Step 4: Seeds linking

# Step 4: Seeds linking



# Step 5: Tips extension

- Seeds don't always map right at the beginning or until the end of the long read
- Once all the seeds have been linked, HG-CoLoR keeps on traversing the graph
- The traversal stops when the borders of the long read or a branching path are reached

# Step 5: Tips extension

- Seeds don't always map right at the beginning or until the end of the long read
- Once all the seeds have been linked, HG-CoLoR keeps on traversing the graph
- The traversal stops when the borders of the long read or a branching path are reached

# Step 5: Tips extension

- Seeds don't always map right at the beginning or until the end of the long read
- Once all the seeds have been linked, HG-CoLoR keeps on traversing the graph
- The traversal stops when the borders of the long read or a branching path are reached

# Remark

- Some seeds might be impossible to link together
- $\Rightarrow$ Production of a corrected long read fragmented in multiple parts

# Remark

- Some seeds might be impossible to link together
- $\Rightarrow$ Production of a corrected long read fragmented in multiple parts

# Datasets

We replaced the enhanced de Bruijn graph in the HG-CoLoR implementation with an overlap graph and with a classical de Bruijn graph, in order to compare the obtained results.

Experiments were run on the following datasets

| Dataset | Reference genome | | Oxford Nanopore data | | | Illumina data | | |
|---|---|---|---|---|---|---|---|---|
| | Name | Genome size | # Reads | Average length | Coverage | # Reads | Read length | Coverage |
| *E. coli* | *E. coli* | 4.6 Mbp | 22,270 | 5,999 | 28x | 465,000 | 300 | 30x |
| Yeast | *S. cerevisae* | 12.4 Mbp | 118,763 | 5,512 | 34x | 2,500,000 | 250 | 50x |

# Alignment-based comparison

| Dataset | Graph | # Reads | # Fragmented reads | Average length | Average identity | Runtime |
|---|---|---|---|---|---|---|
| *E. coli* | Raw reads | 22,270 | N/A | 5,999 | 79.46% | N/A |
| | Overlap graph | 19,592 | 1,319 | 5,979 | 99.91% | 40min |
| | de Bruijn graph ($k = 100$) | 21,782 | 132 | 6,144 | 99.75% | 1h53 |
| | Enhanced de Bruijn graph ($k = 100, m = 50$) | 21,786 | 40 | 6,174 | 99.72% | 1h46 |
| *Yeast* | Raw reads | 118,763 | N/A | 5,512 | 68.63% | N/A |
| | Overlap graph | 60,649 | 14,095 | 4,694 | 99.42% | 6h10 |
| | de Bruijn graph ($k = 100$) | 69,610 | 11,763 | 6,060 | 98.61% | 18h20 |
| | Enhanced de Bruijn graph ($k = 100, m = 50$) | 69,784 | 11,567 | 6,078 | 99.03% | 17h58 |

# Assembly-based comparison

| Dataset | Graph | # Expected contigs | # Obtained contigs |
|---|---|---|---|
| *E. coli* | Overlap graph | 1 | 20 |
| | de Bruijn graph ($k = 100$) | 1 | 4 |
| | Enhanced de Bruijn graph ($k = 100, m = 50$) | 1 | 1 |
| Yeast | Overlap graph | 16 | 197 |
| | de Bruijn graph ($k = 100$) | 16 | 124 |
| | Enhanced de Bruijn graph ($k = 100, m = 50$) | 16 | 103 |

# Conclusion

- We showed that multiple de Bruijn graphs of different orders can be combined into a single enhanced de Bruijn graph
- We showed how to traverse an enhanced de Bruijn graph without explicitly building it
- We introduced a new long read hybrid error correction method relying on an enhanced de Bruijn graph
- We proved the usefulness of enhanced de Bruijn graphs by comparing them with overlap graphs and classical de Bruijn graphs on the HG-CoLoR implementation
- HG-CoLoR is available from: https://github.com/pierre-morisse/HG-CoLoR

# Conclusion

- We showed that multiple de Bruijn graphs of different orders can be combined into a single enhanced de Bruijn graph
- We showed how to traverse an enhanced de Bruijn graph without explicitly building it
- We introduced a new long read hybrid error correction method relying on an enhanced de Bruijn graph
- We proved the usefulness of enhanced de Bruijn graphs by comparing them with overlap graphs and classical de Bruijn graphs on the HG-CoLoR implementation
- HG-CoLoR is available from: https://github.com/pierre-morisse/HG-CoLoR

# Conclusion

- We showed that multiple de Bruijn graphs of different orders can be combined into a single enhanced de Bruijn graph
- We showed how to traverse an enhanced de Bruijn graph without explicitly building it
- We introduced a new long read hybrid error correction method relying on an enhanced de Bruijn graph
- We proved the usefulness of enhanced de Bruijn graphs by comparing them with overlap graphs and classical de Bruijn graphs on the HG-CoLoR implementation
- HG-CoLoR is available from: https://github.com/pierre-morisse/HG-CoLoR

# Conclusion

- We showed that multiple de Bruijn graphs of different orders can be combined into a single enhanced de Bruijn graph
- We showed how to traverse an enhanced de Bruijn graph without explicitly building it
- We introduced a new long read hybrid error correction method relying on an enhanced de Bruijn graph
- We proved the usefulness of enhanced de Bruijn graphs by comparing them with overlap graphs and classical de Bruijn graphs on the HG-CoLoR implementation
- HG-CoLoR is available from: https://github.com/pierre-morisse/HG-CoLoR

# Conclusion

- We showed that multiple de Bruijn graphs of different orders can be combined into a single enhanced de Bruijn graph
- We showed how to traverse an enhanced de Bruijn graph without explicitly building it
- We introduced a new long read hybrid error correction method relying on an enhanced de Bruijn graph
- We proved the usefulness of enhanced de Bruijn graphs by comparing them with overlap graphs and classical de Bruijn graphs on the HG-CoLoR implementation
- HG-CoLoR is available from: `https://github.com/pierre-morisse/HG-CoLoR`

# Future work

- Use a greedy selection at branching paths
- Run HG-CoLoR on larger genomes
- Build a proper assembly tool relying on enhanced de Bruijn graphs
- Compare it with already existing assemblers using multiple de Bruijn graphs of different orders

# Future work

- Use a greedy selection at branching paths
- Run HG-CoLoR on larger genomes
- Build a proper assembly tool relying on enhanced de Bruijn graphs
- Compare it with already existing assemblers using multiple de Bruijn graphs of different orders

# Future work

- Use a greedy selection at branching paths
- Run HG-CoLoR on larger genomes
- Build a proper assembly tool relying on enhanced de Bruijn graphs
- Compare it with already existing assemblers using multiple de Bruijn graphs of different orders

# Future work

- Use a greedy selection at branching paths
- Run HG-CoLoR on larger genomes
- Build a proper assembly tool relying on enhanced de Bruijn graphs
- Compare it with already existing assemblers using multiple de Bruijn graphs of different orders

# References I


Kowalski, T., Grabowski, S., and Deorowicz, S. (2015).
Indexing arbitrary-length k-mers in sequencing reads.
*PLoS ONE*, 10(7):1–14.

Morisse, P., Lecroq, T., and Lefebvre, A. (2017).
HG-CoLoR: Hybrid Graph for the error Correction of Long Reads.
In *Proceedings of the Journées Ouvertes en Biologie, Informatique et Mathématiques*.