# Analysis of Next Generation Sequencing data: Read Mapping

Philipp Rescheneder

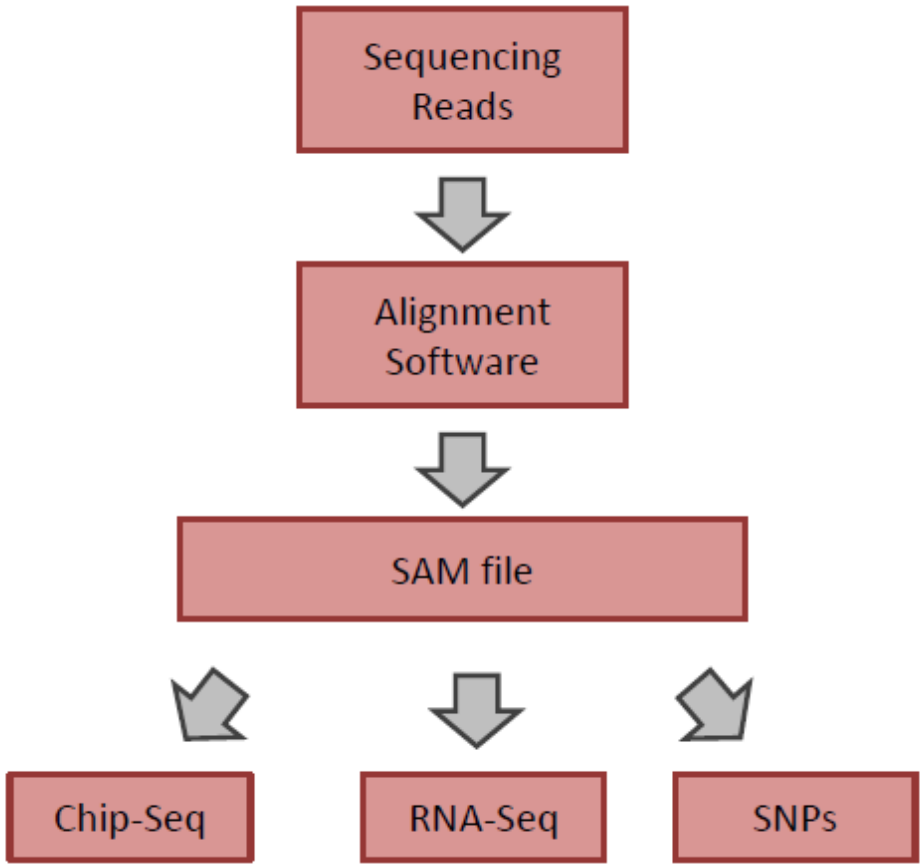
Center for Integrative Bioinformatics Vienna

December 3, 2014

# Center for Integrative Bioinformatics Vienna

CIBIV MFPL
Center for Integrative Bioinformatics Vienna

# Next generation sequencing data analysis



# Overview

1. **General:**
   1. What is read mapping?
   2. Why do we need it?
   3. Sequence alignments (optimal)
   4. Seed and extend (heuristic)
   5. Tree based (heuristic)
2. **Seed and extend:**
   1. How does it work?
   2. Hash-table
   3. Sequence alignment optimizations
   4. Applications

# Making sense of NGS data

DNA copies of the genome

Sequence Reads

The Call–Chronicle–Examiner

SAN FRANCISCO, THURSDAY, APRIL 19, 1906.

EARTHQUAKE AND FIRE: SAN FRANCISCO IN RUINS

NO HOPE LEFT FOR SAFETY OF ANY BUILDINGS

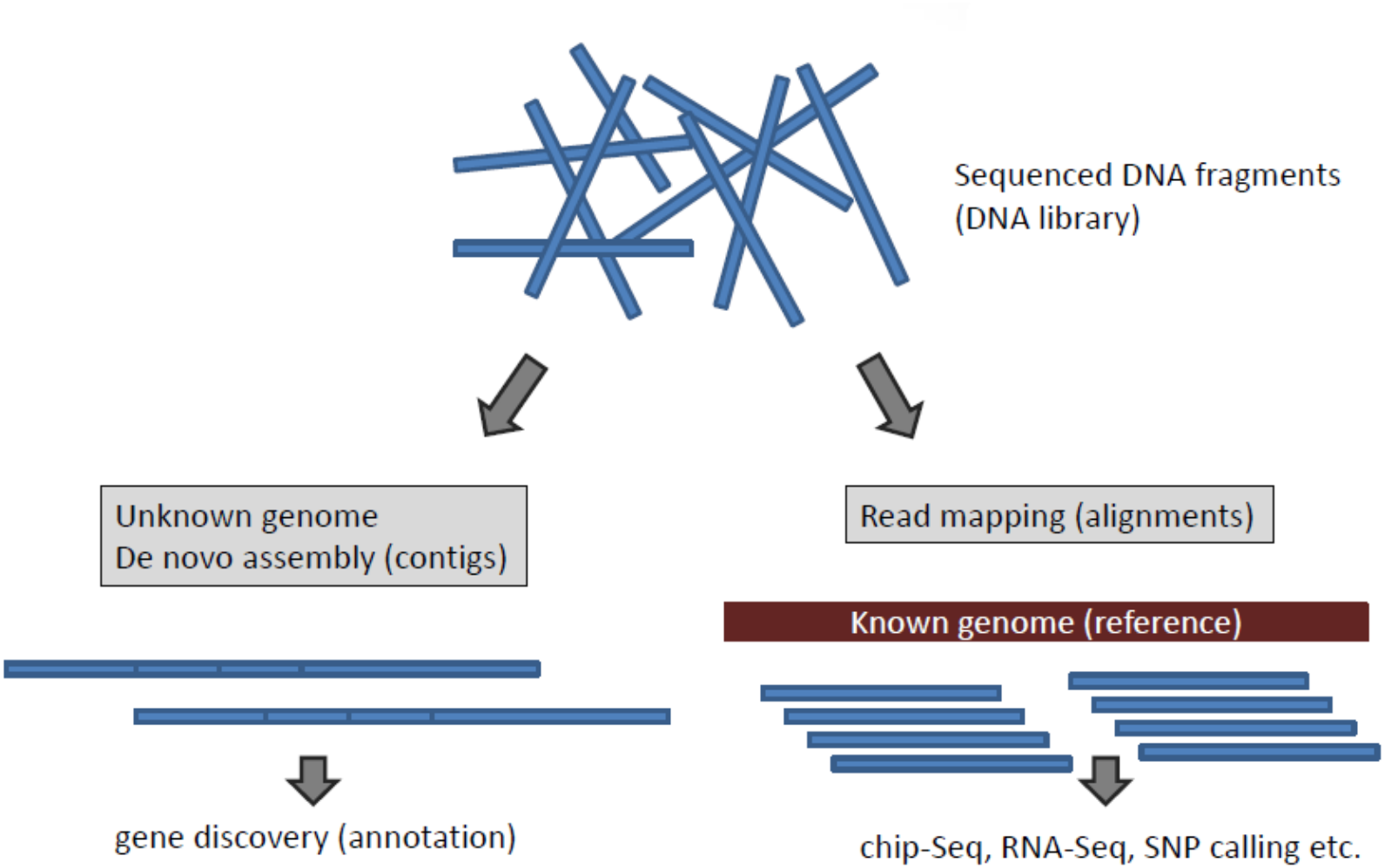
BLOW BUILDINGS UP TO CHECK FLAMES

WHOLE CITY IS ABLAZE

CHURCH OF SAINT IGNATIUS IS DESTROYED

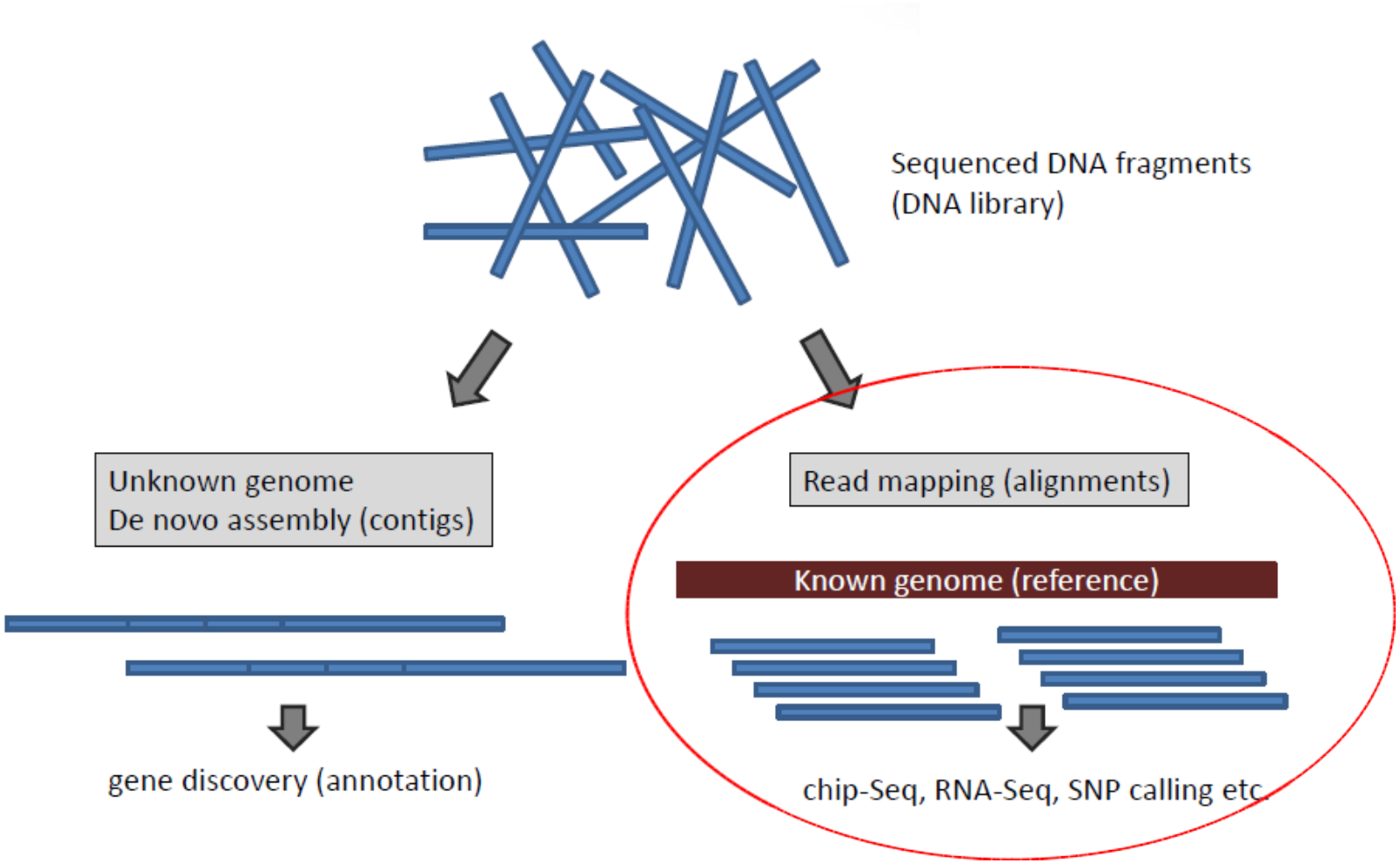
MAYOR CONFERS WITH MILITARY AND CITIZENS

# Making sense of NGS data



# Making sense of NGS data





# Input data

- FASTQ file:

```
$ head SRR094861.fastq
@SRR094861.1 HWI-EAS206_23_30H0YAAXX:5:1:709:724 length=36
ACCTCCCGCCCCCNACNGCNCCCCCGTGCACCTCCC
+SRR094861.1 HWI-EAS206_23_30H0YAAXX:5:1:709:724 length=36
555555555555!%5!55!555555$555(.$2.2
@SRR094861.2 HWI-EAS206_23_30H0YAAXX:5:1:1460:1251 length=36
AAACAAGCTAACATGACTAACACCCTTAATTCCATC
+SRR094861.2 HWI-EAS206_23_30H0YAAXX:5:1:1460:1251 length=36
555555555555555555555555555555121-11
@SRR094861.3 HWI-EAS206_23_30H0YAAXX:5:1:899:1936 length=36
AACAGTCTGATTAAAAAATGGGCCAAAGAGCTTAAC
```

# Input data

- FASTQ file:

```
$ head SRR094861.fastq
@SRR094861.1 HWI-EAS206_23_30H0YAAXX:5:1:709:724 length=36
ACCTCCCGCCCCCNACNGCNCCCCCGTGCACCTCCC
+SRR094861.1 HWI-EAS206_23_30H0YAAXX:5:1:709:724 length=36
55555555555555!%5!55!555555$555(.$2.2
@SRR094861.2 HWI-EAS206_23_30H0YAAXX:5:1:1460:1251 length=36
AAACAAGCTAACATGACTAACACCCTTAATTCCATC
+SRR094861.2 HWI-EAS206_23_30H0YAAXX:5:1:1460:1251 length=36
5555555555555555555555555555121-11
@SRR094861.3 HWI-EAS206_23_30H0YAAXX:5:1:899:1936 length=36
AACAGTCTGATTAAAAAATGGGCCAAAGAGCTTAAC
```

1. sequence id
2. sequence content
3. + (placeholder)
4. quality string

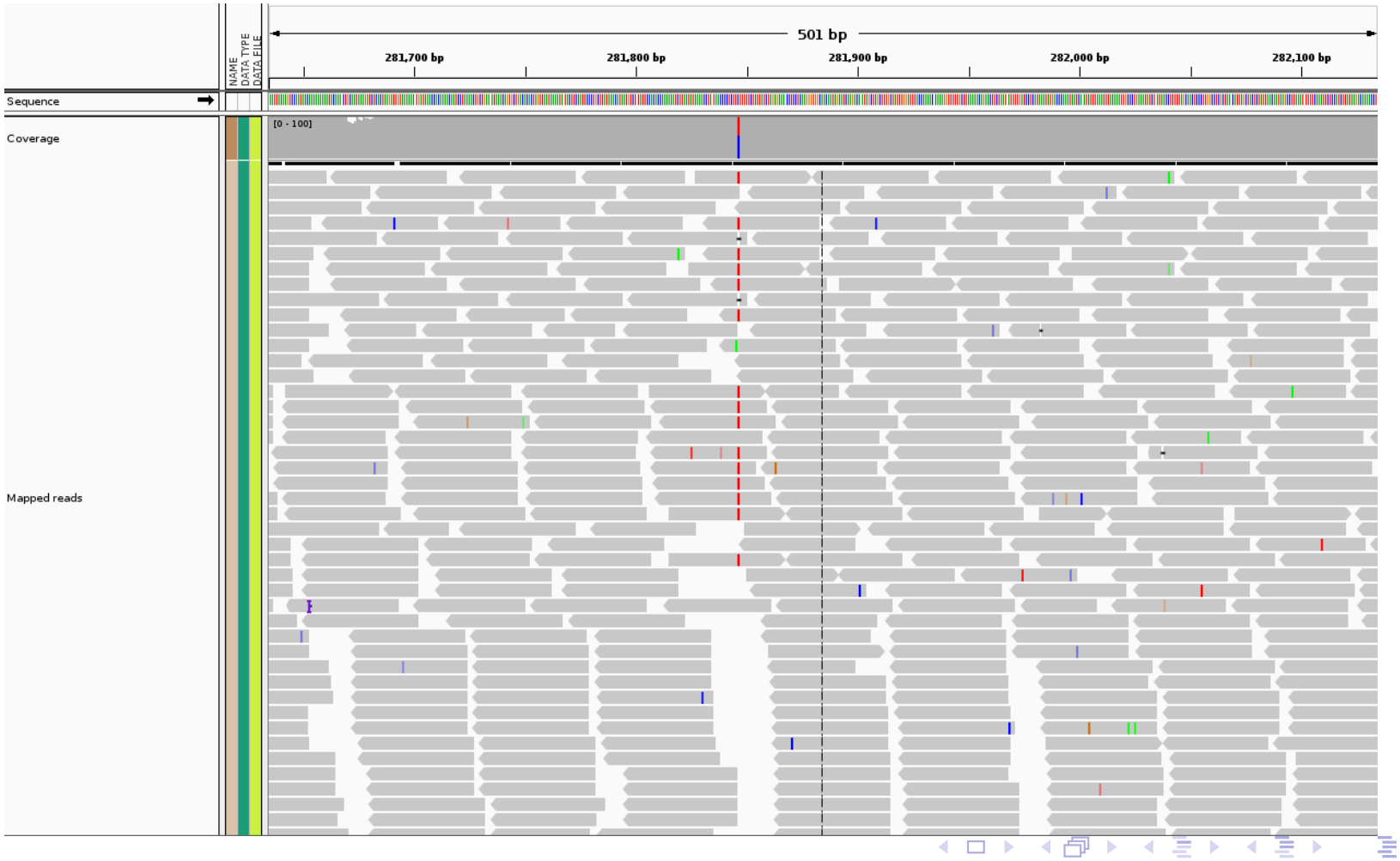
# Read mapping (1)



- Finding the region of the reference genome that shows the highest similarity to a given read

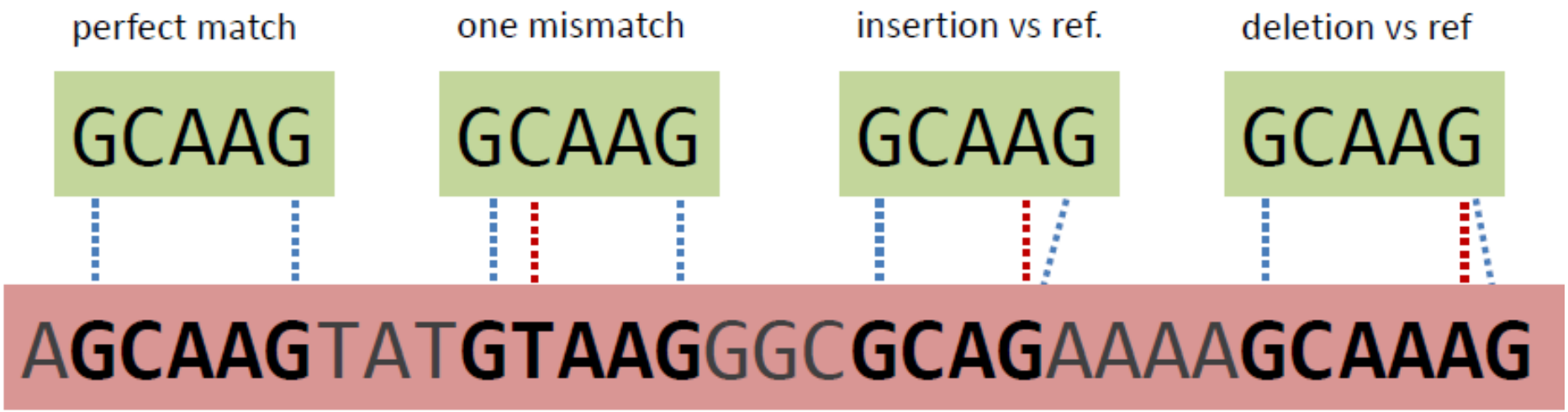
# Read mapping (1)

- Finding the region of the reference genome that shows the highest similarity to a given read



# Read mapping (2)





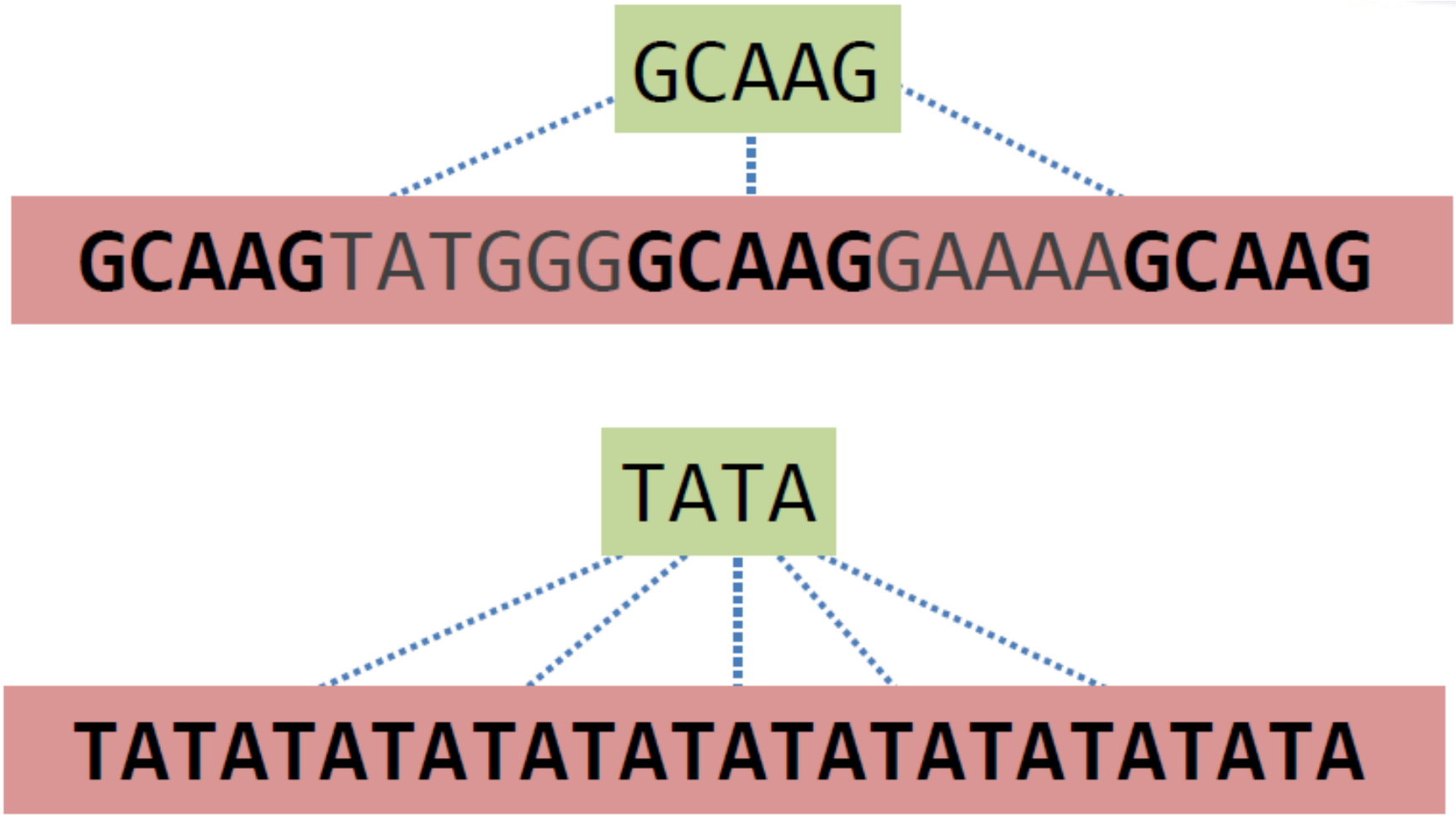

**NOTE: mismatches or indels can be longer than 1 base!**

**It gets complicated very quickly**

Alignment scoring depends on mismatch scoring (different across bases!), gap open, gap extension penalties

# Read mapping (3)





# Difficulties



- Size of reference genome (human: 3 GB, some plant genomes even bigger)
- Number of reads ($> 100,000,000$)
- Differences between read and reference sequence. How to weight them.
- Repeats! Some parts of the genome are unique, some are repeated thousands of times

# Difficulties



- Size of reference genome (human: 3 GB, some plant genomes even bigger)
- Number of reads ($> 100,000,000$)
- Differences between read and reference sequence. How to weight them.
- Repeats! Some parts of the genome are unique, some are repeated thousands of times

# Difficulties



- Size of reference genome (human: 3 GB, some plant genomes even bigger)
- Number of reads ($> 100,000,000$)
- Differences between read and reference sequence. How to weight them.
- Repeats! Some parts of the genome are unique, some are repeated thousands of times

# Difficulties



- Size of reference genome (human: 3 GB, some plant genomes even bigger)
- Number of reads ($> 100,000,000$)
- Differences between read and reference sequence. How to weight them.
- Repeats! Some parts of the genome are unique, some are repeated thousands of times

# Sequence Alignments



- Arranging two or more sequences such as to maximize the length of the common regions between the two
- Operations: insert gap into sequence 1, insert gap into sequence 2, accept mismatch
- Well developed field of research

# Sequence Alignments



- Arranging two or more sequences such as to maximize the length of the common regions between the two
- Operations: insert gap into sequence 1, insert gap into sequence 2, accept mismatch
- Well developed field of research

# Sequence Alignments



- Arranging two or more sequences such as to maximize the length of the common regions between the two
- Operations: insert gap into sequence 1, insert gap into sequence 2, accept mismatch
- Well developed field of research

```
Seq1: GTATACGATTCTTATCTTCTCGTTGTGGTTTCTTA
Seq2: GTATACGATTCTTATCTTCTGTGTTGTGGTATCTTA
```

# Sequence Alignments

- Arranging two or more sequences such as to maximize the length of the common regions between the two
- Operations: insert gap into sequence 1, insert gap into sequence 2, accept mismatch
- Well developed field of research

```
Seq1: GTATACGATT-CTTATCTTCTCGTTGTGGTTTCTTA
      ||||||||||  |    |    ||||||||| |||||
Seq2: GTATACGATTCTTATCTTCTGTGTTGTGGTATCTTA
```

# Sequence Alignments



- Arranging two or more sequences such as to maximize the length of the common regions between the two
- Operations: insert gap into sequence 1, insert gap into sequence 2, accept mismatch
- Well developed field of research

```
Seq1: GTATACGATTCTTATCTTCTC-GTTGTGGTTTCTTA
      |||||||||||||||||||  ||||||||||| |||
Seq2: GTATACGATTCTTATCTTCTGTGTTGTGGTATCTTA
```

# Sequence Alignments

- Arranging two or more sequences such as to maximize the length of the common regions between the two
- Operations: insert gap into sequence 1, insert gap into sequence 2, accept mismatch
- Well developed field of research

```
Seq1: GTATACGATTCTTATCTTCTC-GTTGTGGTTTCTTA
      |||||||||||||||||||  ||||||||||| |||
Seq2: GTATACGATTCTTATCTTCTGTGTTGTGGTATCTTA
```

Given two sequences $A$ and $B$ and a scoring function for two characters $a$ and $b$

$$S(a,b) = \begin{cases} +5 & \text{if } a = b \text{ (match)} \\ -2 & \text{if } a \neq b \text{ (mismatch)} \\ -6 & \text{if } a \text{ or } b \text{ indel (gap)} \end{cases}$$

to score each alignment column. Then we are looking for that alignment, that gives us the highest score $S(A,B)$.

For example:

**T G C T C G T A**

**T – – T C A T A**

+5−6−6+5+5−2+5+5 =11

# Global Alignment: Needleman-Wunsch

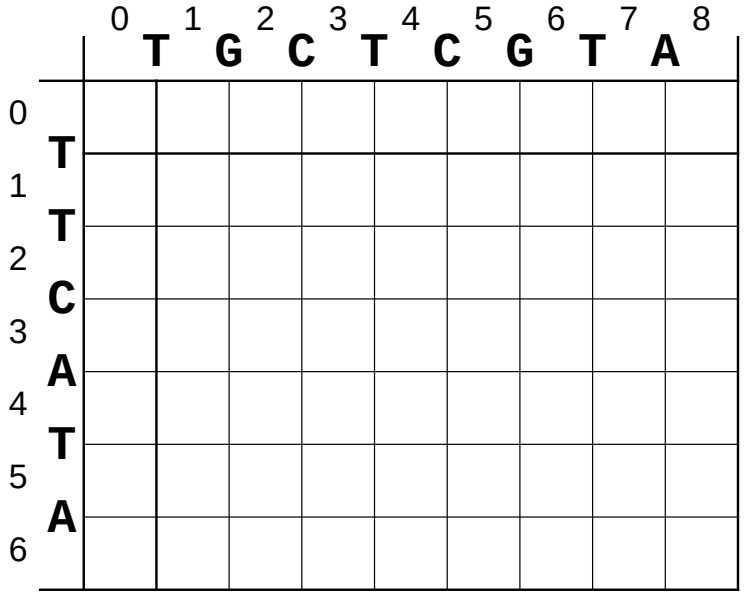
Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

Resulting alignment and score:

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) = \max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$
- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$




Resulting alignment and score:

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) = \max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$
- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 | 1 T | 2 G | 3 C | 4 T | 5 C | 6 G | 7 T | 8 A |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | | | | | | | | |
| 2 | T | −12 | | | | | | | | |
| 3 | C | −18 | | | | | | | | |
| 4 | A | −24 | | | | | | | | |
| 5 | T | −30 | | | | | | | | |
| 6 | A | −36 | | | | | | | | |

Resulting alignment and score:

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) = \max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$
- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch



Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 T | 1 G | 2 C | 3 T | 4 C | 5 G | 6 T | 7 A | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | 5 | | | | | | | |
| 2 | T | −12 | | | | | | | | |
| 3 | C | −18 | | | | | | | | |
| 4 | A | −24 | | | | | | | | |
| 5 | T | −30 | | | | | | | | |
| 6 | A | −36 | | | | | | | | |

Resulting alignment and score:

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) =$

$$\max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$$

- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 T | 1 G | 2 C | 3 T | 4 C | 5 G | 6 T | 7 A | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | 5 | −1 | | | | | | |
| 2 | T | −12 | | | | | | | | |
| 3 | C | −18 | | | | | | | | |
| 4 | A | −24 | | | | | | | | |
| 5 | T | −30 | | | | | | | | |
| 6 | A | −36 | | | | | | | | |

Resulting alignment and score:

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) =$

$$\max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$$

- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 T | 1 G | 2 C | 3 T | 4 C | 5 G | 6 T | 7 A | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | 5 | −1 | −7 | | | | | |
| 2 | T | −12 | | | | | | | | |
| 3 | C | −18 | | | | | | | | |
| 4 | A | −24 | | | | | | | | |
| 5 | T | −30 | | | | | | | | |
| 6 | A | −36 | | | | | | | | |

Resulting alignment and score:

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) =$

$$\max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$$

- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 T | 1 G | 2 C | 3 T | 4 C | 5 G | 6 T | 7 A | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | 5 | −1 | −7 | −13 | | | | |
| 2 | T | −12 | | | | | | | | |
| 3 | C | −18 | | | | | | | | |
| 4 | A | −24 | | | | | | | | |
| 5 | T | −30 | | | | | | | | |
| 6 | A | −36 | | | | | | | | |

Resulting alignment and score:

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) =$

$$\max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$$

- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 T | 1 G | 2 C | 3 T | 4 C | 5 G | 6 T | 7 A | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | 5 | −1 | −7 | −13 | −19 | −25 | −31 | −37 |
| 2 | T | −12 | | | | | | | | |
| 3 | C | −18 | | | | | | | | |
| 4 | A | −24 | | | | | | | | |
| 5 | T | −30 | | | | | | | | |
| 6 | A | −36 | | | | | | | | |

Resulting alignment and score:

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) =$

$$\max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$$

- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 T | 1 G | 2 C | 3 T | 4 C | 5 G | 6 T | 7 A | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | 5 | −1 | −7 | −13 | −19 | −25 | −31 | −37 |
| 2 | T | −12 | −1 | 3 | −3 | −2 | −8 | −14 | −20 | −26 |
| 3 | C | −18 | −7 | −3 | 8 | 2 | 3 | −3 | −9 | −15 |
| 4 | A | −24 | −13 | −9 | 2 | 6 | 0 | 1 | −5 | −4 |
| 5 | T | −30 | −19 | −15 | −4 | 7 | 4 | −2 | 6 | 0 |
| 6 | A | −36 | −25 | −21 | −10 | 1 | 5 | 2 | 0 | 11 |

Resulting alignment and score:

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) =$

$$\max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$$

- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 T | 1 G | 2 C | 3 T | 4 C | 5 G | 6 T | 7 A | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | 5 | −1 | −7 | −13 | −19 | −25 | −31 | −37 |
| 2 | T | −12 | −1 | 3 | −3 | −2 | −8 | −14 | −20 | −26 |
| 3 | C | −18 | −7 | −3 | 8 | 2 | 3 | −3 | −9 | −15 |
| 4 | A | −24 | −13 | −9 | 2 | 6 | 0 | 1 | −5 | −4 |
| 5 | T | −30 | −19 | −15 | −4 | 7 | 4 | −2 | 6 | 0 |
| 6 | A | −36 | −25 | −21 | −10 | 1 | 5 | 2 | 0 | 11 |

Resulting alignment and score:

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) =$
$$\max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$$
- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 T | 1 G | 2 C | 3 T | 4 C | 5 G | 6 T | 7 A | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | 5 | −1 | −7 | −13 | −19 | −25 | −31 | −37 |
| 2 | T | −12 | −1 | 3 | −3 | −2 | −8 | −14 | −20 | −26 |
| 3 | C | −18 | −7 | −3 | 8 | 2 | 3 | −3 | −9 | −15 |
| 4 | A | −24 | −13 | −9 | 2 | 6 | 0 | 1 | −5 | −4 |
| 5 | T | −30 | −19 | −15 | −4 | 7 | 4 | −2 | 6 | 0 |
| 6 | A | −36 | −25 | −21 | −10 | 1 | 5 | 2 | 0 | 11 |

Resulting alignment and score:

**T G C T C G T A**
**T − − T C A T A**
+5−6−6+5+5−2+5+5 =11

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) = \max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$
- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 T | 1 G | 2 C | 3 T | 4 C | 5 G | 6 T | 7 A | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | 5 | −1 | −7 | −13 | −19 | −25 | −31 | −37 |
| 2 | T | −12 | −1 | 3 | −3 | −2 | −8 | −14 | −20 | −26 |
| 3 | C | −18 | −7 | −3 | 8 | 2 | 3 | −3 | −9 | −15 |
| 4 | A | −24 | −13 | −9 | 2 | 6 | 0 | 1 | −5 | −4 |
| 5 | T | −30 | −19 | −15 | −4 | 7 | 4 | −2 | 6 | 0 |
| 6 | A | −36 | −25 | −21 | −10 | 1 | 5 | 2 | 0 | 11 |

Resulting alignment and score:

**T G C T C G T A**
**T − − T C A T A**
+5−6−6+5+5−2+5+5 =11

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) =$
$$\max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$$
- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Global Alignment: Needleman-Wunsch

Given sequences $A$ and $B$ and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

| | | 0 T | 1 G | 2 C | 3 T | 4 C | 5 G | 6 T | 7 A | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | −6 | −12 | −18 | −24 | −30 | −36 | −42 | −48 |
| 1 | T | −6 | 5 | −1 | −7 | −13 | −19 | −25 | −31 | −37 |
| 2 | T | −12 | −1 | 3 | −3 | −2 | −8 | −14 | −20 | −26 |
| 3 | C | −18 | −7 | −3 | 8 | 2 | 3 | −3 | −9 | −15 |
| 4 | A | −24 | −13 | −9 | 2 | 6 | 0 | 1 | −5 | −4 |
| 5 | T | −30 | −19 | −15 | −4 | 7 | 4 | −2 | 6 | 0 |
| 6 | A | −36 | −25 | −21 | −10 | 1 | 5 | 2 | 0 | 11 |

Resulting alignment and score:

**T G C T C G T A**
**T − − T C A T A**
+5−6−6+5+5−2+5+5 =11

- Initialize an $N \times M$ matrix with the sequences $A$ and $B$ of length $M$ and $N$.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) = \max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$
- Optimal score for can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

# Local alignment (Smith-Waterman algorithm)



```
Seq1: GTATACGATTCTTATCTTCTCGTTGTGGTTTCTTA

Seq2: AAAAAAAAAAAAATTCTGTGTTGAAAAAAAAA
```

- Instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure
- Negative scoring matrix cells are set to zero
- Backtracking starts at the highest scoring matrix cell and proceeds until a cell with score zero

# Local alignment (Smith-Waterman algorithm)



```
Seq1: GTATACGATTCTTATCTTCTC-GTTGTGGTTTCTTA
           |   |     |  ||||  ||||          |
Seq2: ---AAAAAAAAAAAAATTCTGTGTTGAAAAAAAA-A
```

- Instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure
- Negative scoring matrix cells are set to zero
- Backtracking starts at the highest scoring matrix cell and proceeds until a cell with score zero

# Local alignment (Smith-Waterman algorithm)



```
Seq1: GTATACGATTCTTATCTTCTC-GTTGTGGTTTCTTA
                           ||||  ||||
Seq2:                      TTCTGTGTTG
```

- Instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure
- Negative scoring matrix cells are set to zero
- Backtracking starts at the highest scoring matrix cell and proceeds until a cell with score zero

# Local alignment (Smith-Waterman algorithm)



```
Seq1: GTATACGATTCTTATCTTCTC-GTTGTGGTTTCTTA
                           ||||  ||||
Seq2:                      TTCTGTGTTG
```

- Instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure
- Negative scoring matrix cells are set to zero
- Backtracking starts at the highest scoring matrix cell and proceeds until a cell with score zero

# Local alignment (Smith-Waterman algorithm)



```
Seq1: GTATACGATTCTTATCTTCTC-GTTGTGGTTTCTTA
                           ||||  ||||
Seq2:                      TTCTGTGTTG
```

- Instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure
- Negative scoring matrix cells are set to zero
- Backtracking starts at the highest scoring matrix cell and proceeds until a cell with score zero

# Local alignment (Smith-Waterman algorithm)



```
Seq1: GTATACGATTCTTATCTTCTC-GTTGTGGTTTCTTA
                           ||||  ||||
Seq2:                      TTCTGTGTTG
```

- Instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure
- Negative scoring matrix cells are set to zero
- Backtracking starts at the highest scoring matrix cell and proceeds until a cell with score zero

# Using alignments for read mapping



- Gives optimal results $\Rightarrow$ local alignment between read and reference genome, but:
- Alignment matrix: $3,000,000,000 * 100 \Rightarrow 600 GB$ of main memory
- Assume time needed for alignment is 0.1 seconds
- For $100,000,000 \Rightarrow$ 115 days
- We need methods that can map at least $10,000$ reads per second on standard hardware

# Using alignments for read mapping



- Gives optimal results $\Rightarrow$ local alignment between read and reference genome, but:
- Alignment matrix: $3,000,000,000 * 100 \Rightarrow 600GB$ of main memory
- Assume time needed for alignment is 0.1 seconds
- For $100,000,000 \Rightarrow$ 115 days
- We need methods that can map at least $10,000$ reads per second on standard hardware

# Using alignments for read mapping



- Gives optimal results $\Rightarrow$ local alignment between read and reference genome, but:
- Alignment matrix: $3,000,000,000 * 100 \Rightarrow 600GB$ of main memory
- Assume time needed for alignment is 0.1 seconds
- For $100,000,000 \Rightarrow$ 115 days
- We need methods that can map at least $10,000$ reads per second on standard hardware

# Using alignments for read mapping



- Gives optimal results $\Rightarrow$ local alignment between read and reference genome, but:
- Alignment matrix: $3,000,000,000 * 100 \Rightarrow 600GB$ of main memory
- Assume time needed for alignment is 0.1 seconds
- For $100,000,000 \Rightarrow$ 115 days
- We need methods that can map at least $10,000$ reads per second on standard hardware

# Using alignments for read mapping



- Gives optimal results $\Rightarrow$ local alignment between read and reference genome, but:
- Alignment matrix: $3,000,000,000 * 100 \Rightarrow 600GB$ of main memory
- Assume time needed for alignment is 0.1 seconds
- For $100,000,000 \Rightarrow$ 115 days
- We need methods that can map at least $10,000$ reads per second on standard hardware

# Heuristic approaches



- Heuristic refers to experience-based techniques for problem solving that find a solution which is not guaranteed to be optimal, but good enough for a given set of goals
  - Seed and extend (BLAST)
  - Tree/Trie based approach (suffix/prefix trees, burrows wheeler transformation, etc.)

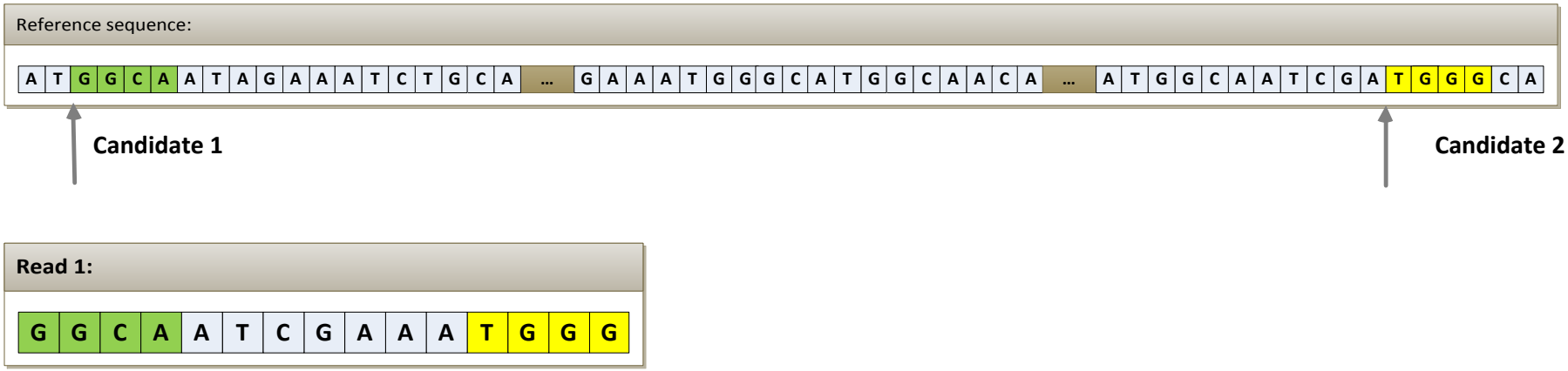
# Heuristic approaches



- Heuristic refers to experience-based techniques for problem solving that find a solution which is not guaranteed to be optimal, but good enough for a given set of goals
  - Seed and extend (BLAST)
  - Tree/Trie based approach (suffix/prefix trees, burrows wheeler transformation, etc.)

# Seed and extend

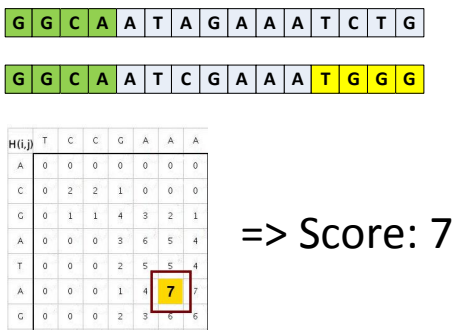
1. Candidate mapping region search: identification of perfectly matching sub regions using an index data structure
2. Computation of Smith-Waterman alignment scores for candidates
3. Computation of full alignment only for the highest scoring candidate



# Seed and extend

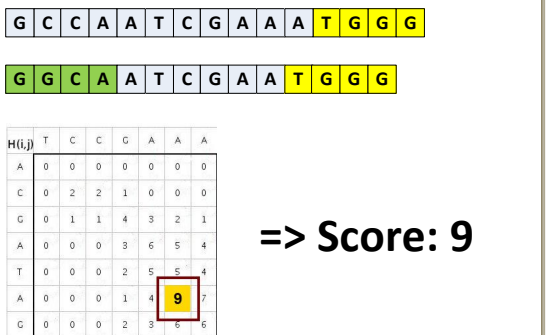
1. Candidate mapping region search: identification of perfectly matching sub regions using an index data structure
2. Computation of Smith-Waterman alignment scores for candidates
3. Computation of full alignment only for the highest scoring candidate

**Candidate 1**




=> Score: 7

**Candidate 2**




**=> Score: 9**

...

1. Candidate mapping region search: identification of perfectly matching sub regions using an index data structure
2. Computation of Smith-Waterman alignment scores for candidates
3. Computation of full alignment only for the highest scoring candidate

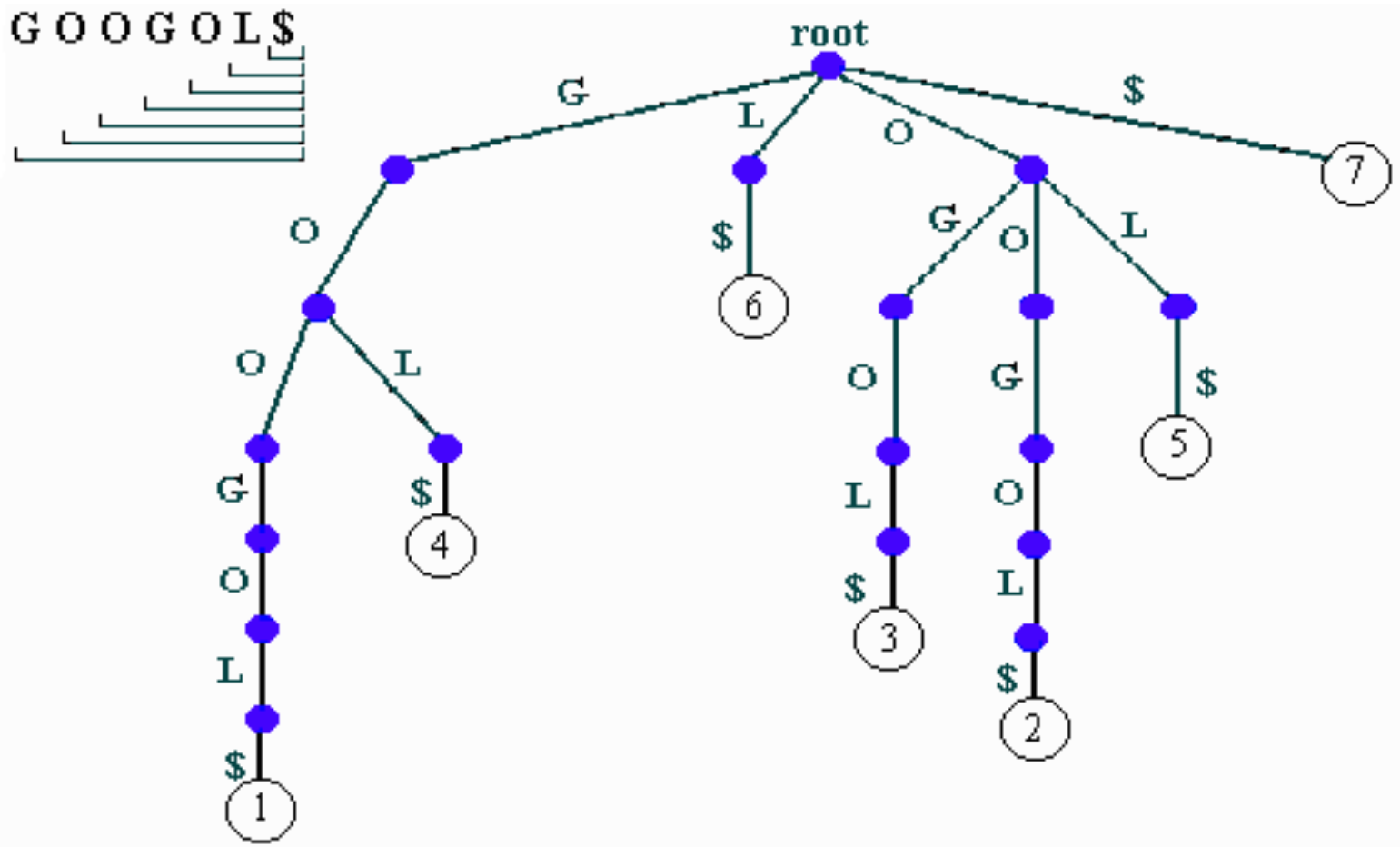
| H(i,j) | T | C | C | G | A | A | A |
|---|---|---|---|---|---|---|---|
| G | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 0 | 0 | 0 | 2 | 2 | 2 |
| T | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| C | 0 | 2 | 2 | 1 | 0 | 0 | 0 |
| C | 0 | 2 | 4 | 3 | 2 | 1 | 0 |
| A | 0 | 1 | 3 | 3 | 5 | 4 | 3 |
| A | 0 | 0 | 2 | 2 | 5 | 7 | 6 |
| A | 0 | 0 | 1 | 1 | 4 | 7 | 9 |
| T | 0 | 0 | 0 | 0 | 3 | 6 | 8 |
| G | 0 | 2 | 2 | 1 | 2 | 5 | 7 |

**Candidate 2**

| G | C | C | A | A | T | C | G | A | A | A | T | G | G | G |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G | G | C | A | A | T | C | G | A | A | - | T | G | G | G |

# Suffix tree

- Search “GOL” in “GOOGOL”



# Suffix tree: problems



- Only for exact matching strings
- But, we have mismatches, insertions and deletions

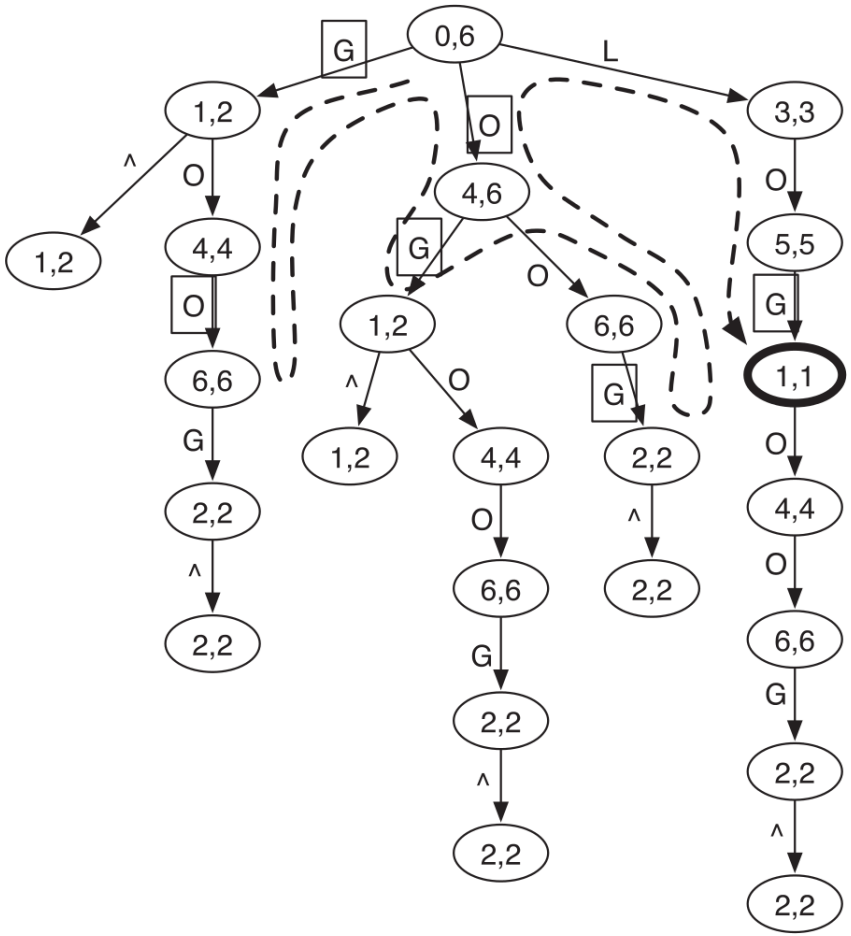
# Suffix tree: problems



- Only for exact matching strings
- But, we have mismatches, insertions and deletions

# Prefix tree

- Search “LOL” in “GOOGOL”



# Seed and extend vs. Trees



Seed and extend:

- High memory usage
- Typically slow(er) due to time consuming alignment computation
- High sensitivity
- Can handle a high number of differences between the read and the references

Trees:

- Fast
- Memory efficient
- Better handling of repeating regions
- Not well suited for a high number of mismatches and insertions/deletions

# Seed and extend vs. Trees



Seed and extend:

- High memory usage
- Typically slow(er) due to time consuming alignment computation
- High sensitivity
- Can handle a high number of differences between the read and the references

Trees:

- Fast
- Memory efficient
- Better handling of repeating regions
- Not well suited for a high number of mismatches and insertions/deletions

# Seed and extend vs. Trees



Seed and extend:

- High memory usage
- Typically slow(er) due to time consuming alignment computation
- High sensitivity
- Can handle a high number of differences between the read and the references

Trees:

- Fast
- Memory efficient
- Better handling of repeating regions
- Not well suited for a high number of mismatches and insertions/deletions

# Seed and extend vs. Trees



Seed and extend:

- High memory usage
- Typically slow(er) due to time consuming alignment computation
- High sensitivity
- Can handle a high number of differences between the read and the references

Trees:

- Fast
- Memory efficient
- Better handling of repeating regions
- Not well suited for a high number of mismatches and insertions/deletions

# Seed and extend vs. Trees



Seed and extend:

- High memory usage
- Typically slow(er) due to time consuming alignment computation
- High sensitivity
- Can handle a high number of differences between the read and the references

Trees:

- Fast
- Memory efficient
- Better handling of repeating regions
- Not well suited for a high number of mismatches and insertions/deletions

# Seed and extend vs. Trees



Seed and extend:

- High memory usage
- Typically slow(er) due to time consuming alignment computation
- High sensitivity
- Can handle a high number of differences between the read and the references

Trees:

- Fast
- Memory efficient
- Better handling of repeating regions
- Not well suited for a high number of mismatches and insertions/deletions

# Seed and extend vs. Trees



Seed and extend:

- High memory usage
- Typically slow(er) due to time consuming alignment computation
- High sensitivity
- Can handle a high number of differences between the read and the references

Trees:

- Fast
- Memory efficient
- Better handling of repeating regions
- Not well suited for a high number of mismatches and insertions/deletions

# Seed and extend vs. Trees



Seed and extend:

- High memory usage
- Typically slow(er) due to time consuming alignment computation
- High sensitivity
- Can handle a high number of differences between the read and the references

Trees:

- Fast
- Memory efficient
- Better handling of repeating regions
- Not well suited for a high number of mismatches and insertions/deletions

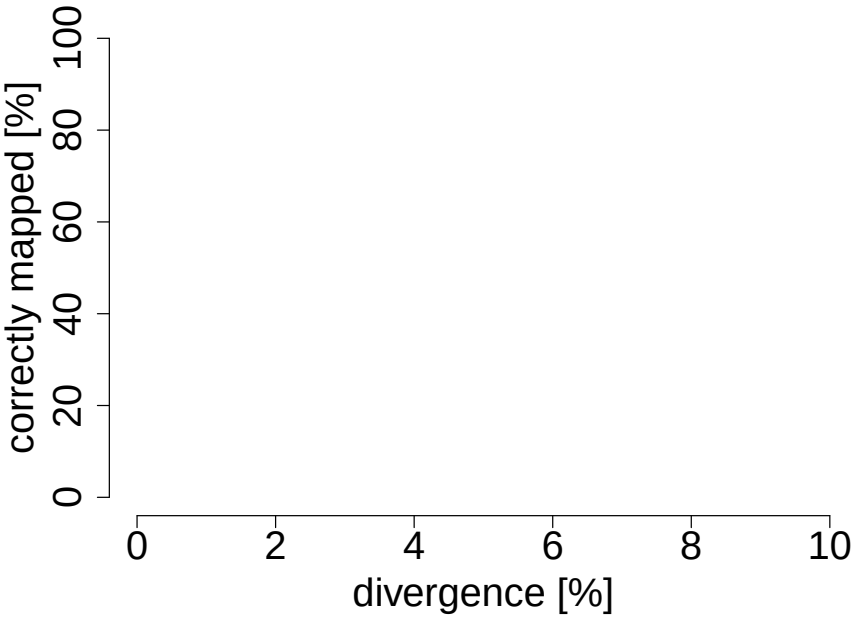
# Read mapper comparison (1)



- Reference genome: *A. thaliana*
- Genomes size: $\sim$ 110 MB
- We simulated 11 data sets with 5 million, 100 bp reads
- 0 to 10 % difference between sequenced and reference genome
- 2% simulated sequencing error
- Programs: BWA(-SW), Bowtie2, Stampy
- Comparison of runtime and number of correctly mapped reads

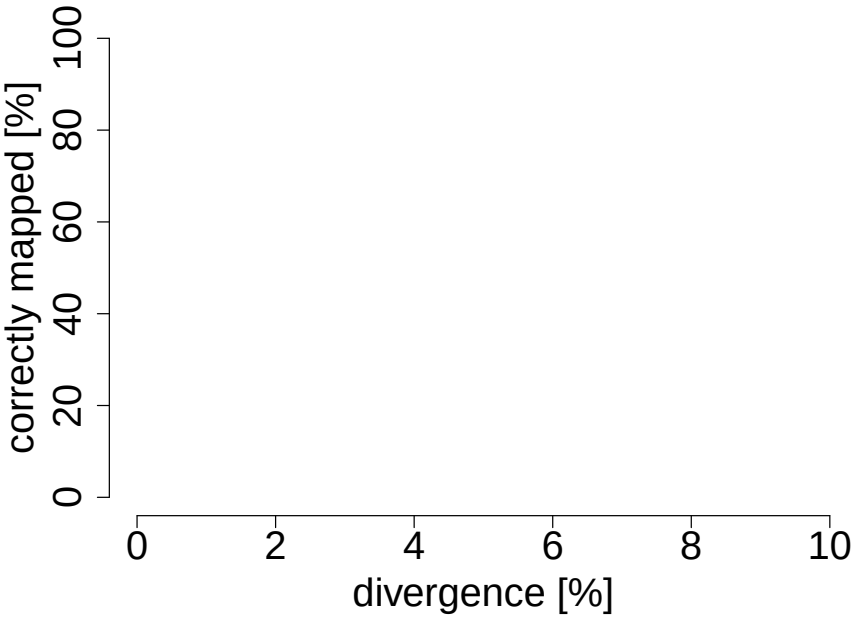
# Read mapper comparison (2)

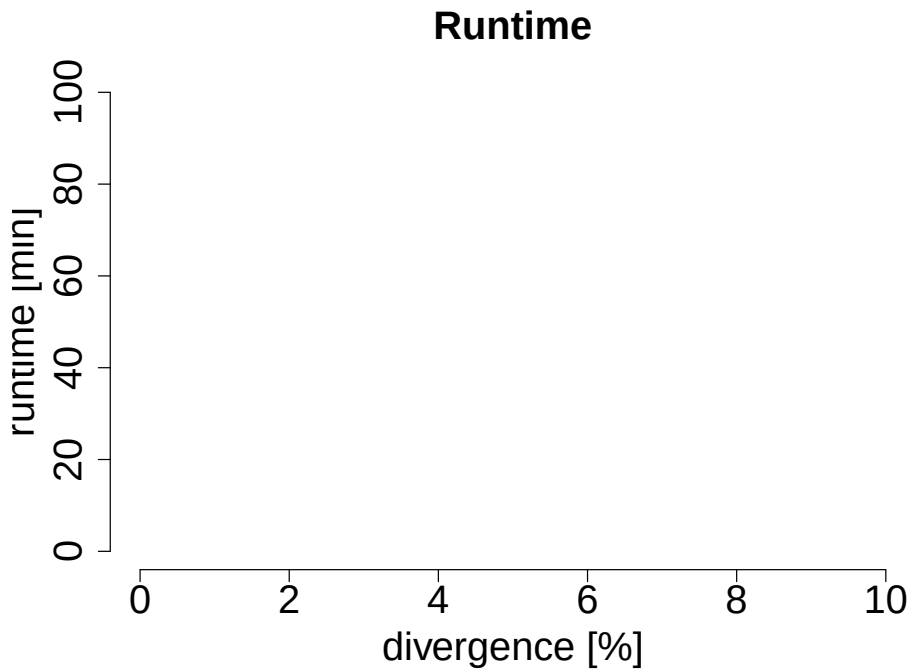




# Read mapper comparison (2)

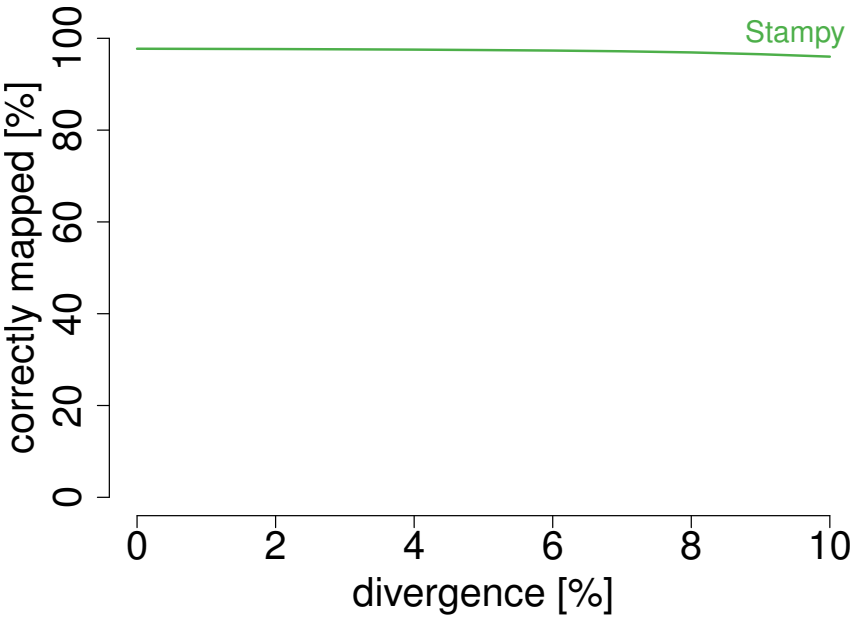


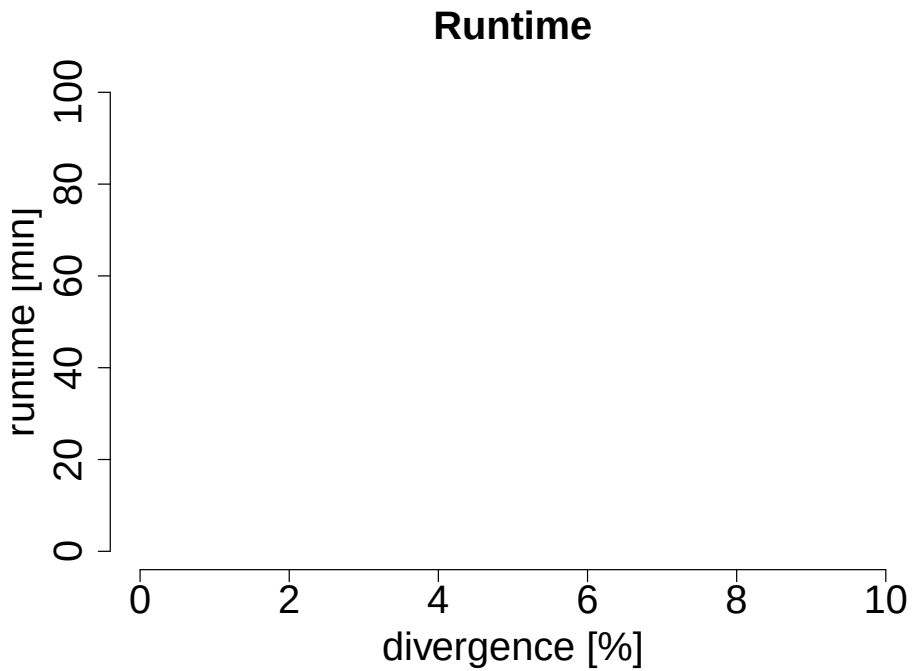


# Read mapper comparison (2)

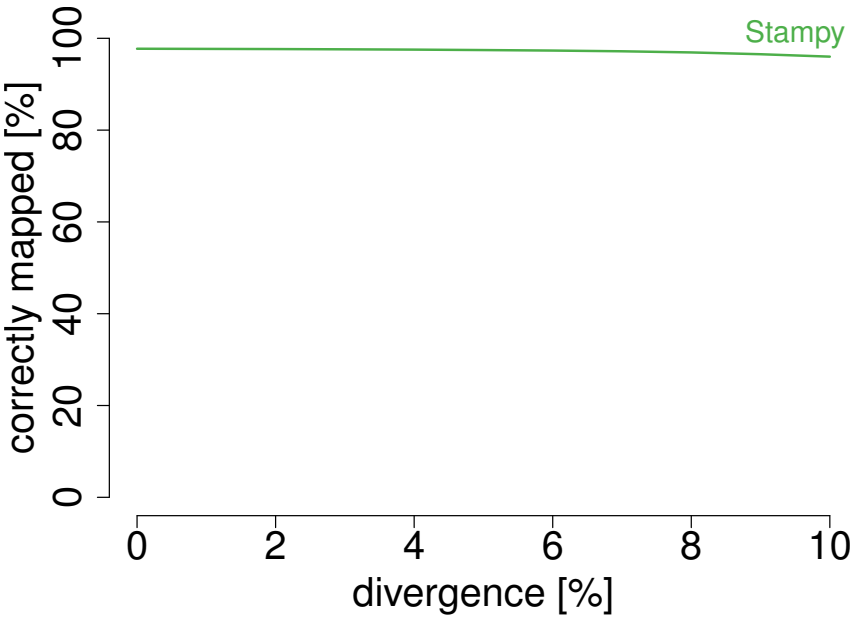


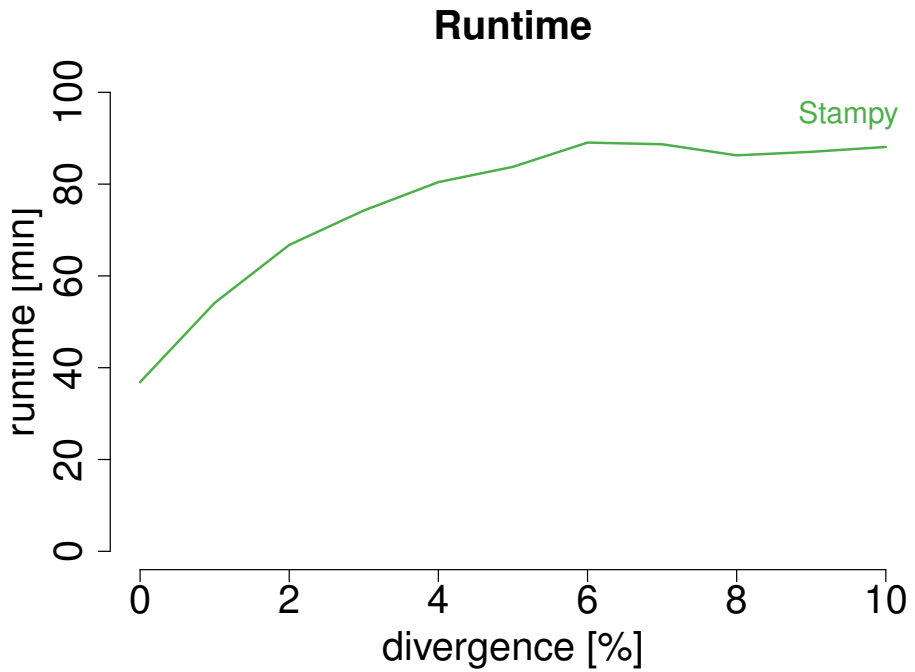


# Read mapper comparison (2)

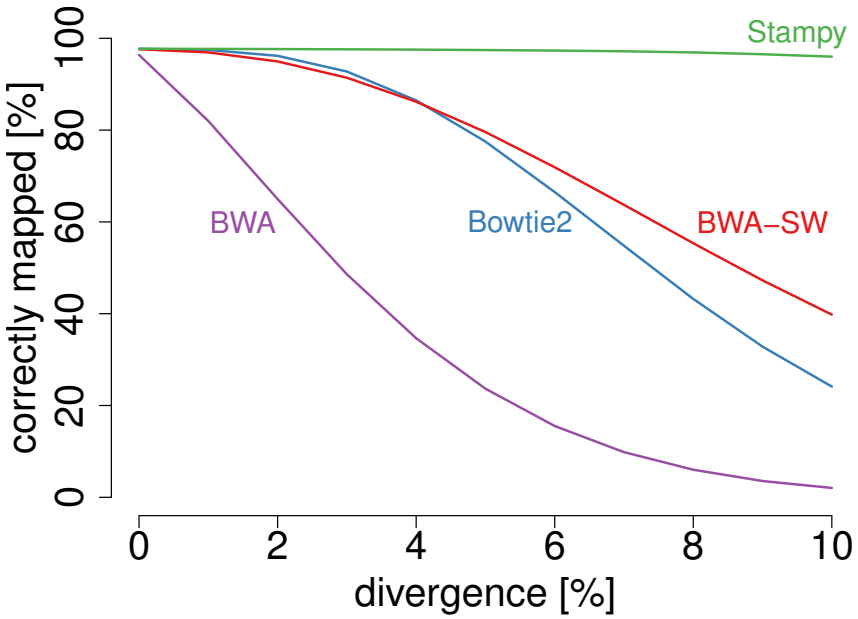


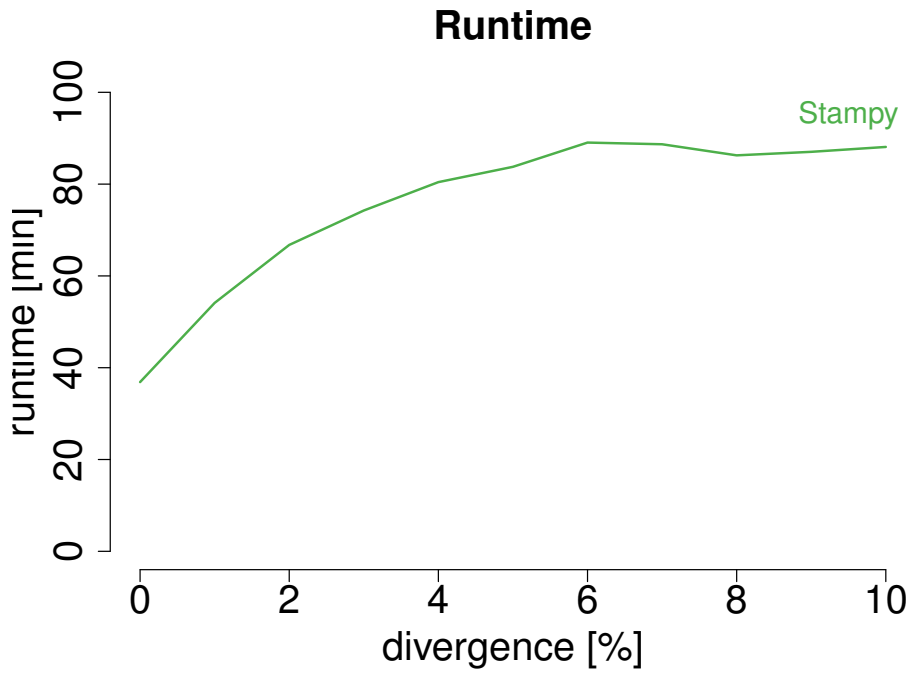


# Read mapper comparison (2)

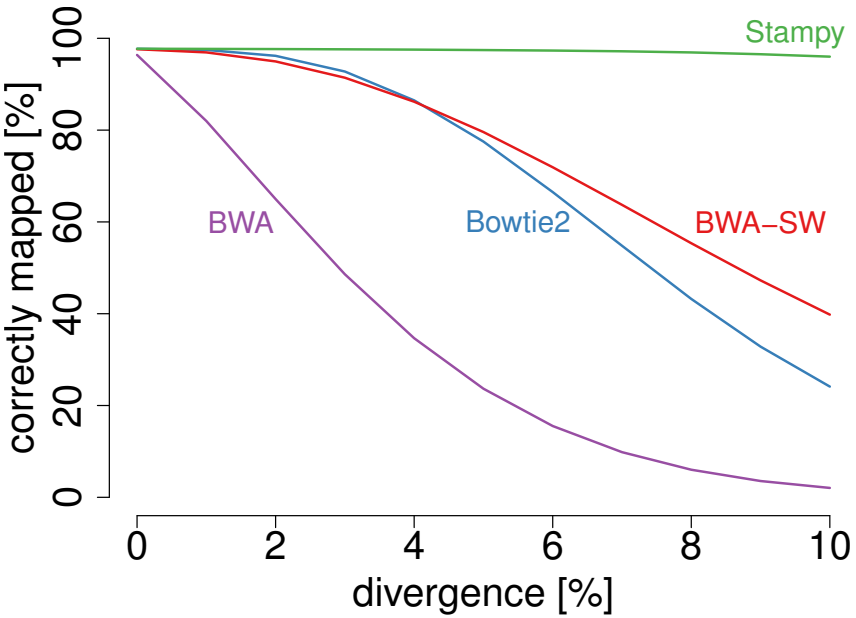


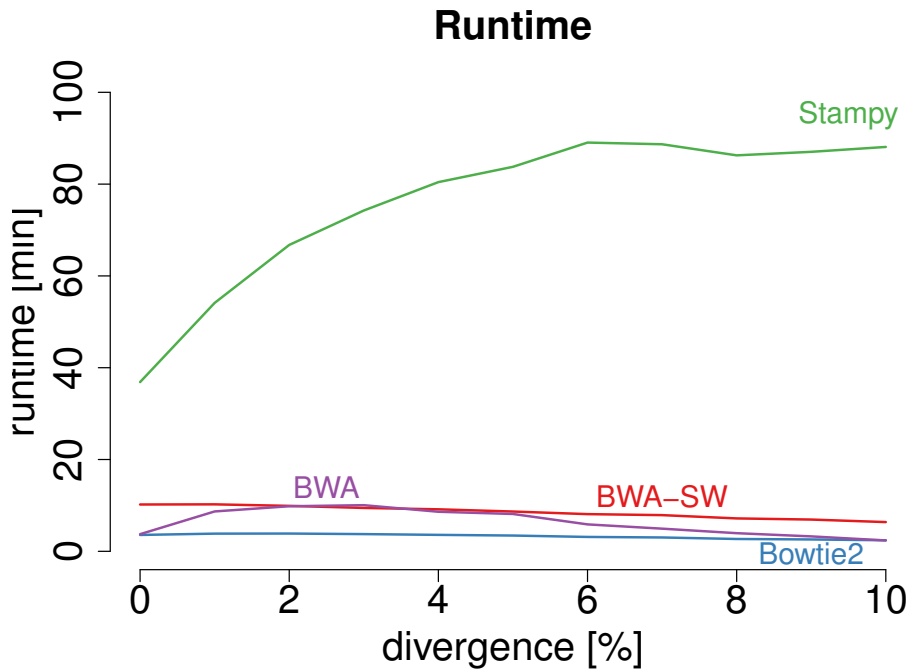


# Read mapper comparison (2)





- Read mapping program
- Based on seed and extend approach
- Uses sequence alignments to get high mapping sensitivity
- Goal: as fast or faster then tree based approaches
- Optimizations:
  - Technical
  - Algorithmic

- Read mapping program
- Based on seed and extend approach
- Uses sequence alignments to get high mapping sensitivity
- Goal: as fast or faster then tree based approaches
- Optimizations:
  - Technical
  - Algorithmic

- Read mapping program
- Based on seed and extend approach
- Uses sequence alignments to get high mapping sensitivity
- Goal: as fast or faster then tree based approaches
- Optimizations:
  - Technical
  - Algorithmic

- Read mapping program
- Based on seed and extend approach
- Uses sequence alignments to get high mapping sensitivity
- Goal: as fast or faster then tree based approaches
- Optimizations:
  - Technical
  - Algorithmic

- Read mapping program
- Based on seed and extend approach
- Uses sequence alignments to get high mapping sensitivity
- Goal: as fast or faster then tree based approaches
- Optimizations:
  - Technical
  - Algorithmic

# NextGenMap

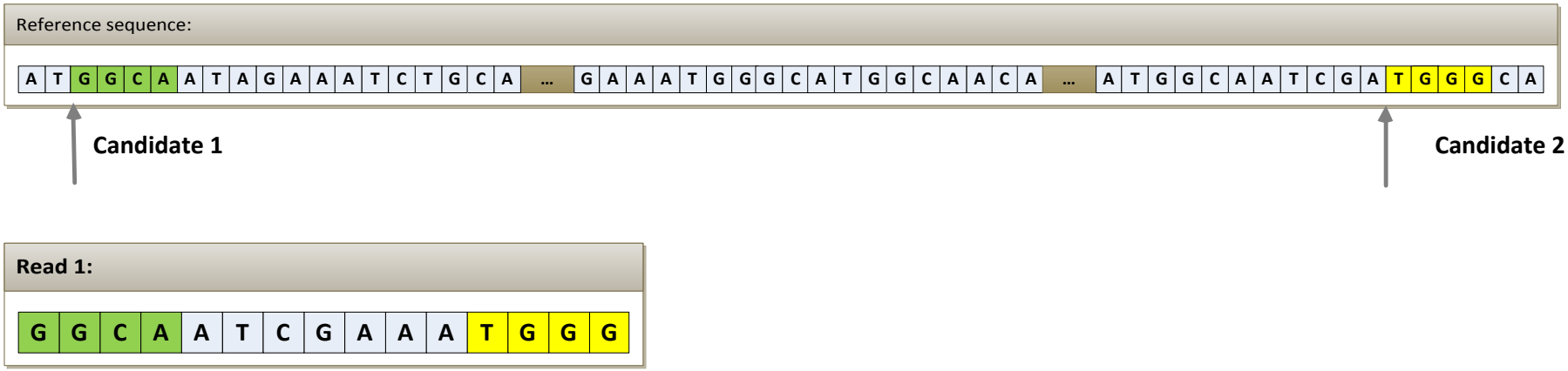
- Read mapping program
- Based on seed and extend approach
- Uses sequence alignments to get high mapping sensitivity
- Goal: as fast or faster then tree based approaches
- Optimizations:
  - Technical
  - Algorithmic

# NextGenMap

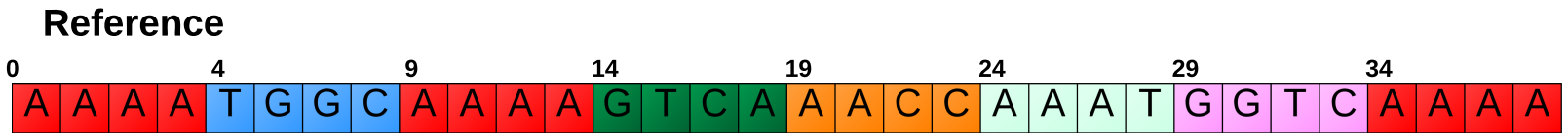
- Read mapping program
- Based on seed and extend approach
- Uses sequence alignments to get high mapping sensitivity
- Goal: as fast or faster then tree based approaches
- Optimizations:
  - Technical
  - Algorithmic

# Seed and extend

1. Identification of perfectly matching sub regions using an **index data structure**: Candidate mapping region (CMR) search
2. Computation of Smith-Waterman alignment scores for candidates
3. Computation of full alignment only for the highest scoring candidate



# Indexing the genome




**Objective:**

Fast identification of genomic positions given a k-mer

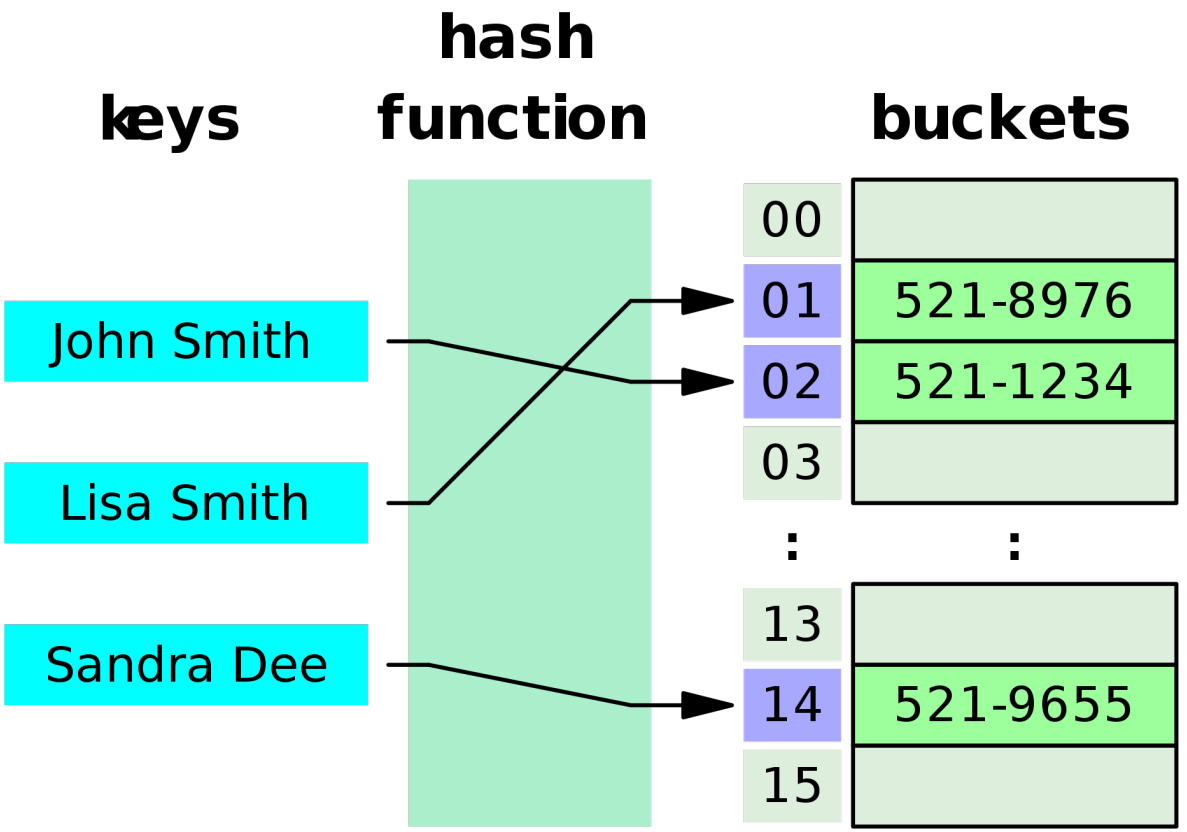
# Hash-table



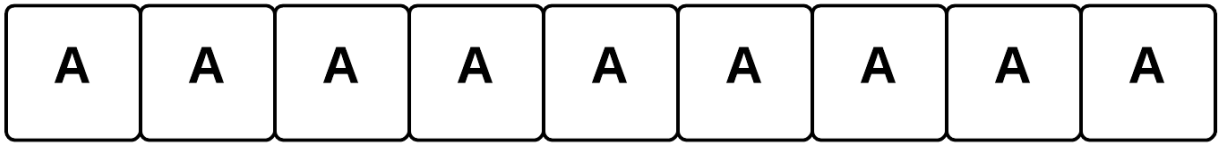
- A hash function maps data (e.g. strings) of arbitrary size to data of fixed size, with slight differences in input data producing very big differences in output data.

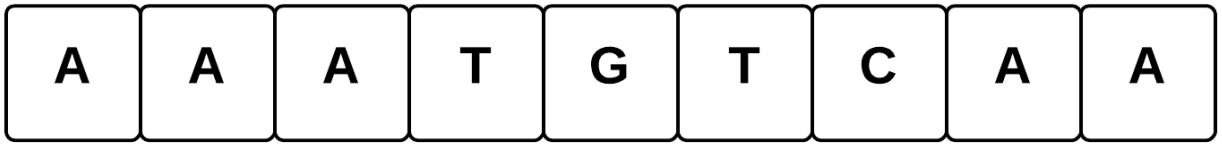
# Hash-table

- A hash function maps data (e.g. strings) of arbitrary size to data of fixed size, with slight differences in input data producing very big differences in output data.



# Hashing sequencing data




**Rule:**

**A-> 00**
**C-> 01**
**G-> 10**
**T-> 11**

# Hashing sequencing data

| A | A | A | A | A | A | A | A | A |
|---|---|---|---|---|---|---|---|---|
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

| A | A | A | T | G | T | C | A | A |
|---|---|---|---|---|---|---|---|---|

**Rule:**

**A-> 00**

**C-> 01**

**G-> 10**

**T-> 11**

# Hashing sequencing data



| A | A | A | A | A | A | A | A | A |
|---|---|---|---|---|---|---|---|---|
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

| A | A | A | T | G | T | C | A | A |
|---|---|---|---|---|---|---|---|---|
| 00 | 00 | 00 | 11 | 10 | 11 | 01 | 00 | 00 |

**Rule:**

**A-> 00**
**C-> 01**
**G-> 10**
**T-> 11**

# Hashing sequencing data



| A | A | A | A | A | A | A | A | A |
|---|---|---|---|---|---|---|---|---|
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

= 0

**Rule:**

**A-> 00**
**C-> 01**
**G-> 10**
**T-> 11**

| A | A | A | T | G | T | C | A | A |
|---|---|---|---|---|---|---|---|---|
| 00 | 00 | 00 | 11 | 10 | 11 | 01 | 00 | 00 |

= 3,792

# Construct hash-table (1)



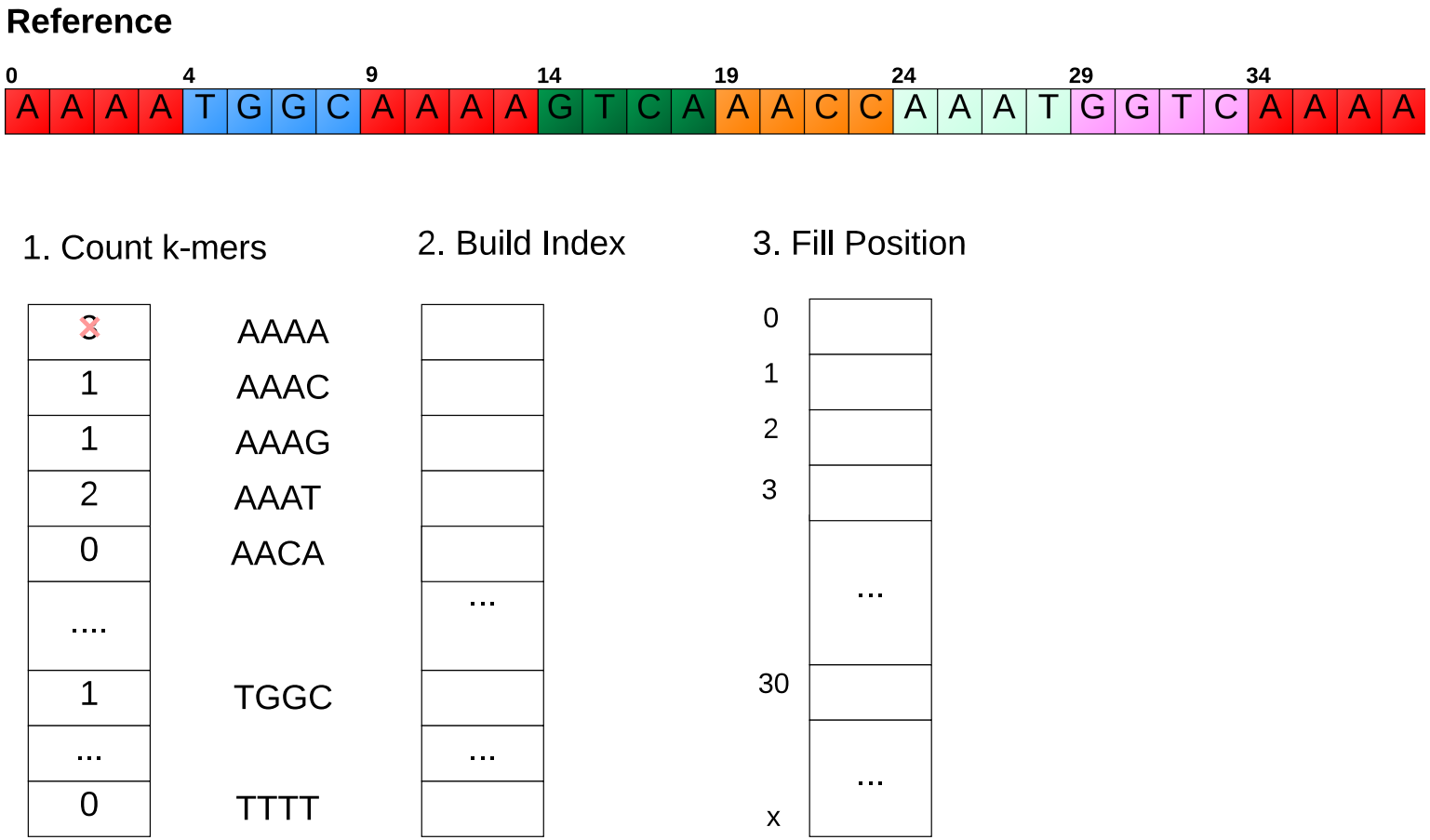
**Reference**


1. Count k-mers

| | |
|---|---|
| 3 | AAAA |
| 1 | AAAC |
| 1 | AAAG |
| 2 | AAAT |
| 0 | AACA |
| .... | |
| 1 | TGGC |
| ... | |
| 0 | TTTT |

# Construct hash-table (2)





# Construct hash-table (2)



**Reference**


1. Count k-mers

| | |
|---|---|
| ~~3~~ | AAAA |
| 1 | AAAC |
| 1 | AAAG |
| 2 | AAAT |
| 0 | AACA |
| .... | |
| 1 | TGGC |
| ... | |
| 0 | TTTT |

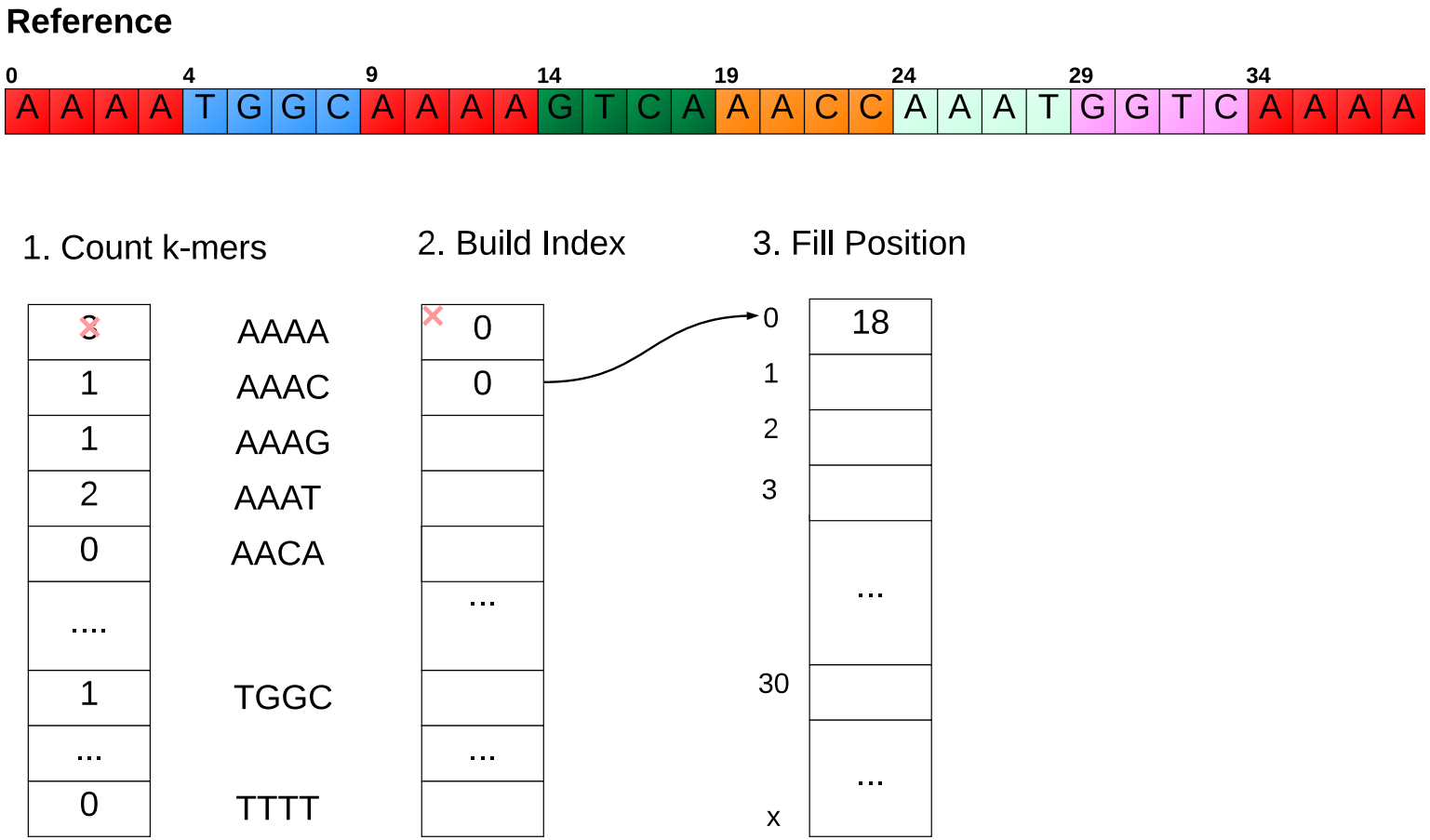
2. Build Index

| |
|---|
| ~~0~~ |
| |
| |
| |
| |
| ... |
| |
| ... |
| |

3. Fill Position

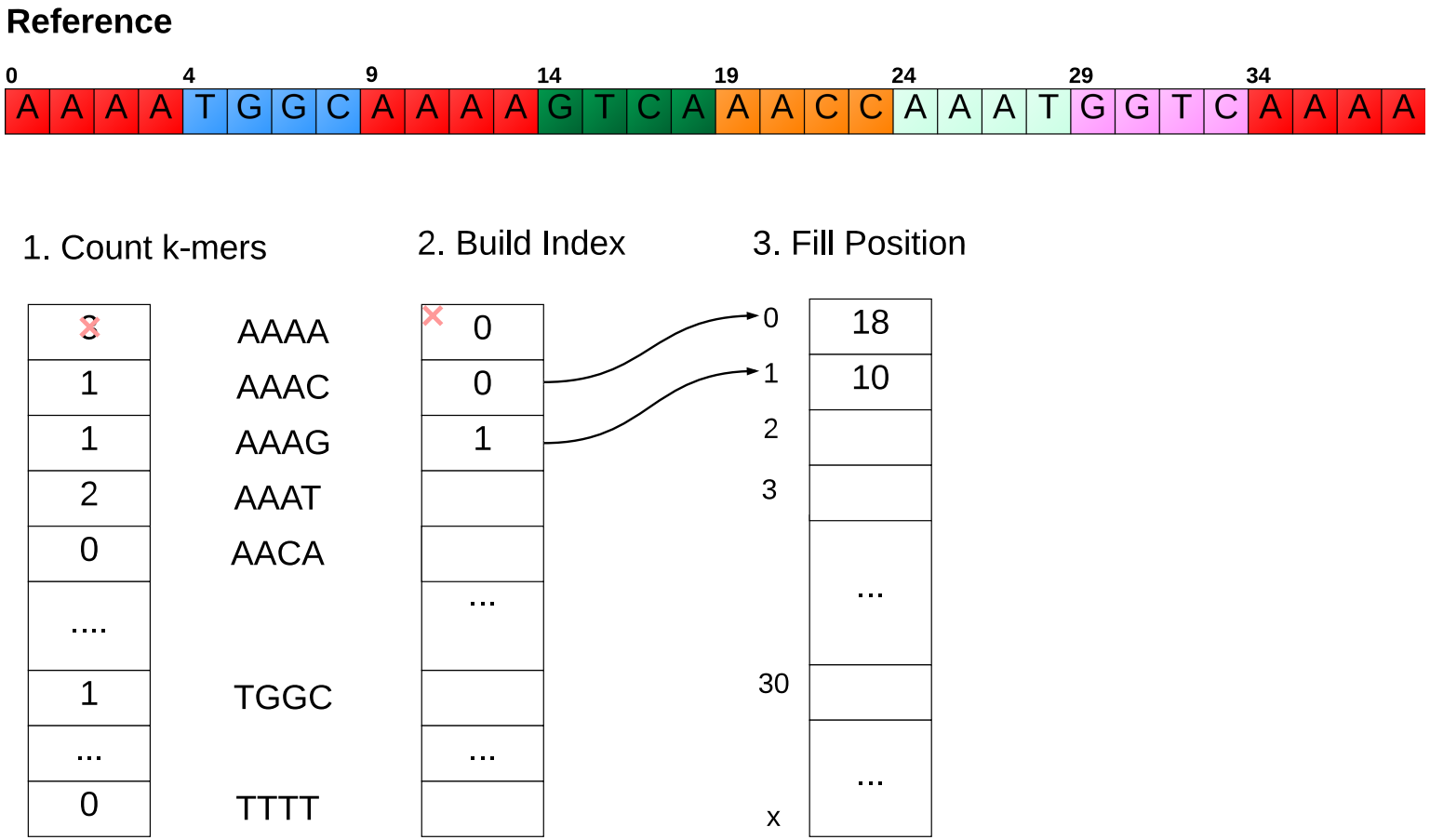
| | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| | ... |
| 30 | |
| | ... |
| x | |

# Construct hash-table (2)



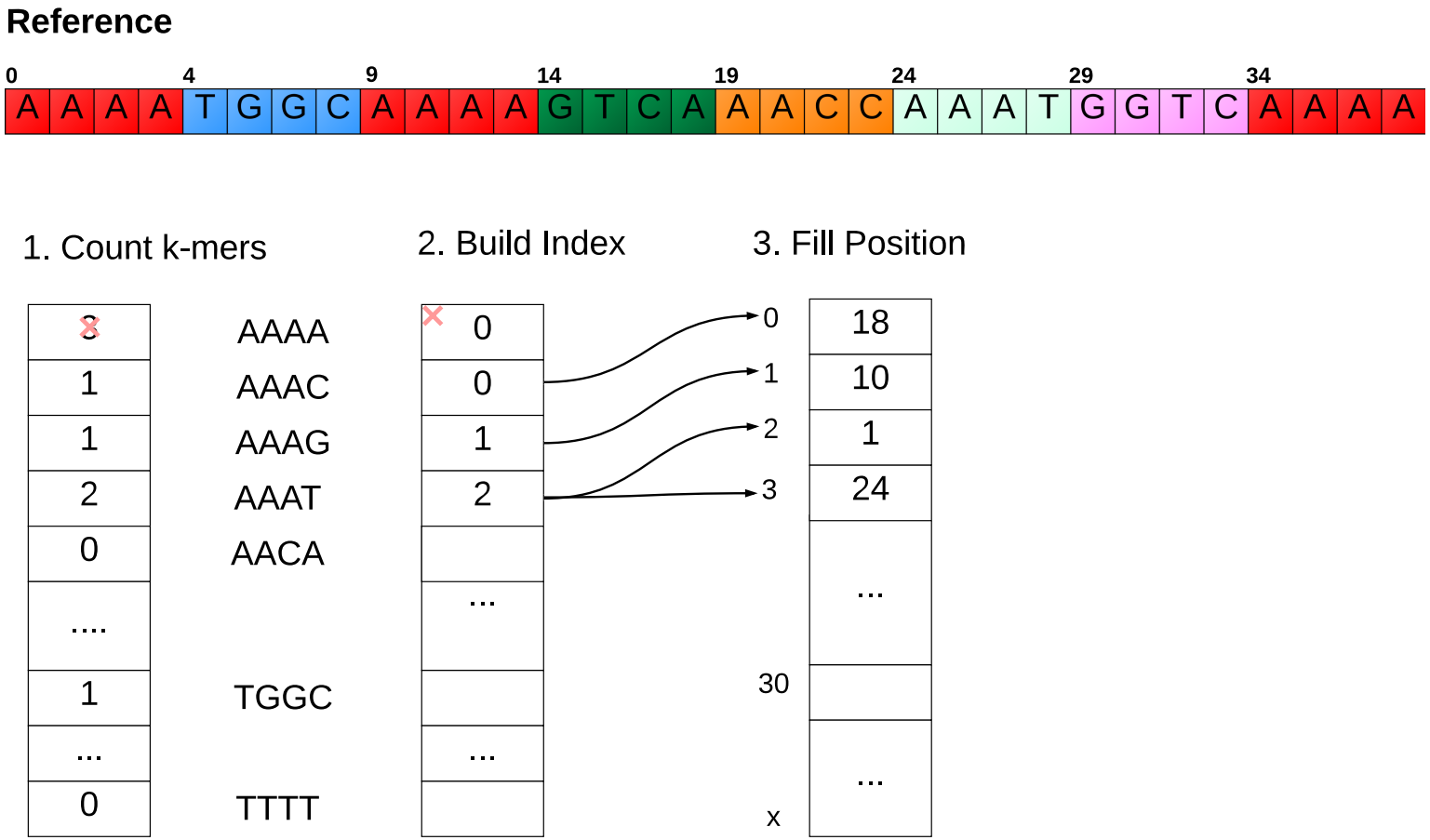
# Construct hash-table (2)



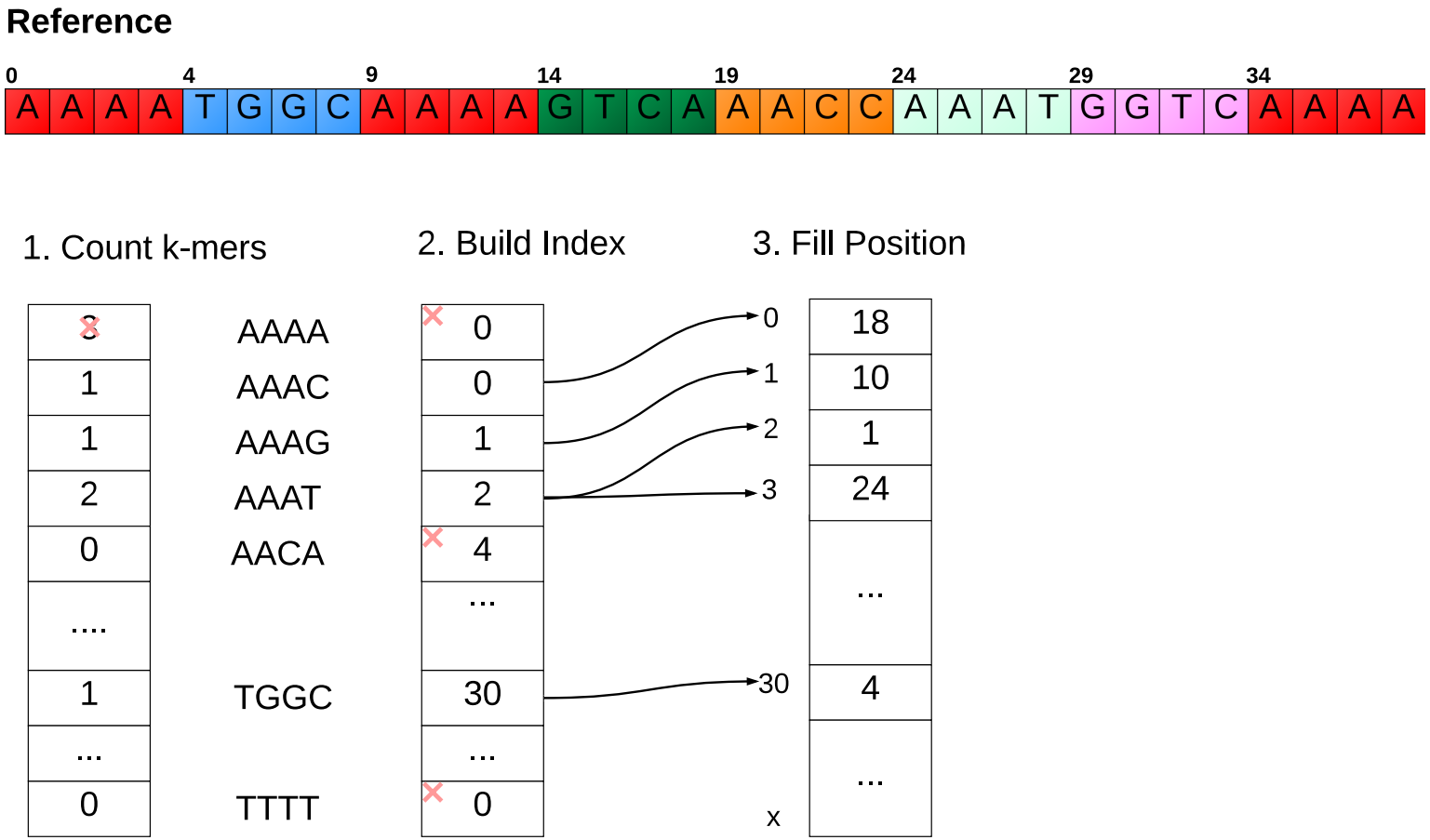


# Construct hash-table (2)





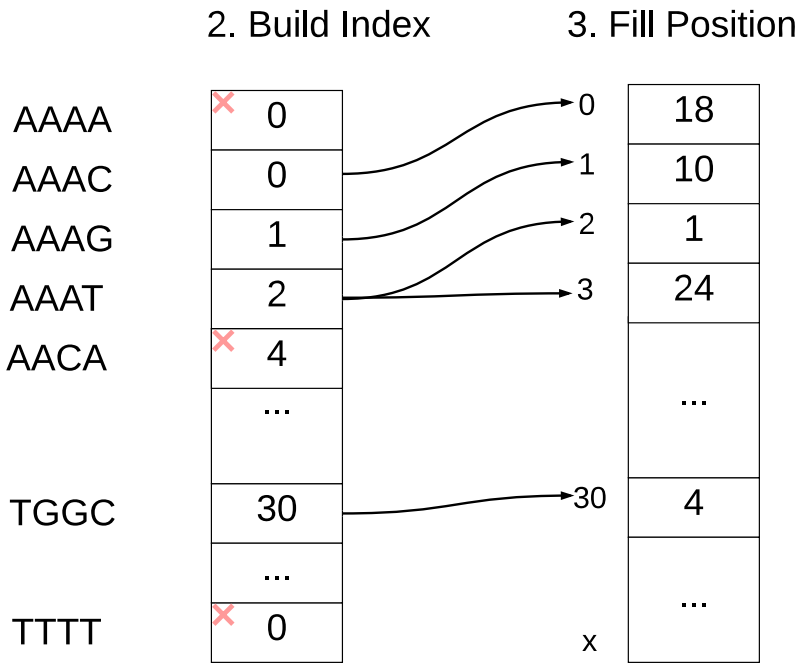
# Construct hash-table (2)



# Construct hash-table (3)

**Advantages:**

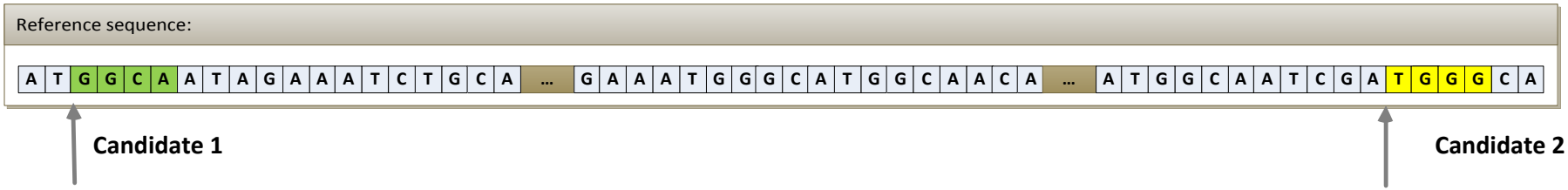
1. Fast access (continuous memory access)
2. Memory efficient ($\sim$ 3 GB for human)

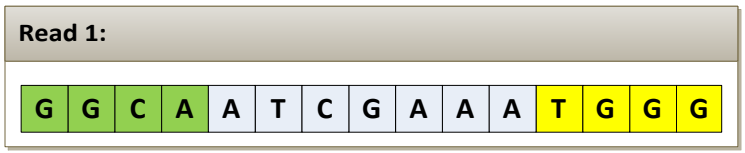


# Candidate region search



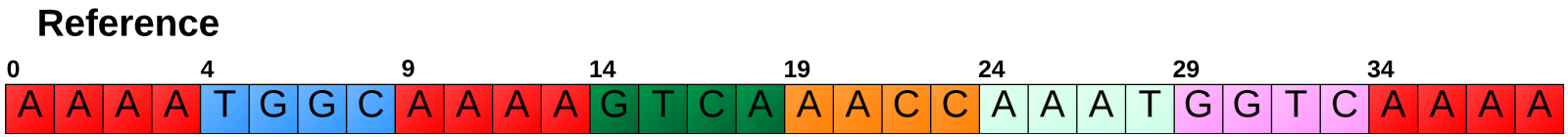
1. Identification of perfectly matching sub regions using an index data structure: **Candidate mapping region (CMR) search**
2. Computation of Smith-Waterman alignment scores for candidates
3. Computation of full alignment only for the highest scoring candidate



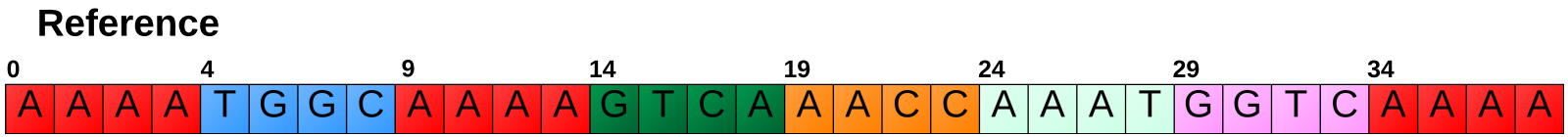



# Collect candidate regions





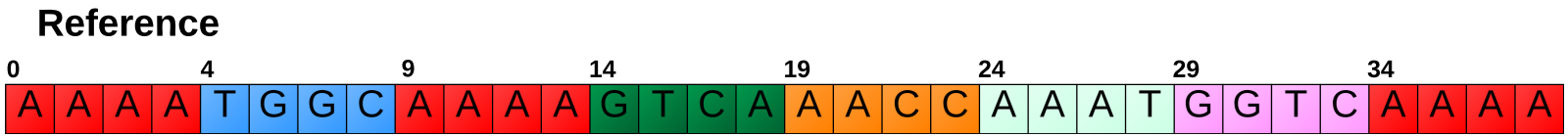
# Collect candidate regions




| K-mers | genomic positions |
|---|---|
| A A A A | |
| A A A T | 1, 24 |
| A A T G | 2, 25 |
| A T G G | 3, 26 |
| T G G C | 4 |

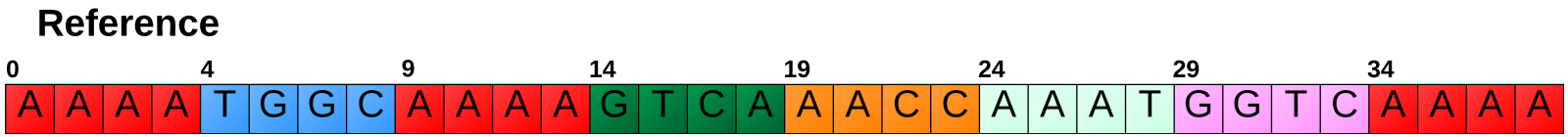
| K-mers | genomic positions |
|---|---|
| A A A T | 1, 24 |
| A A T G | 2, 25 |
| A T G T | |
| T G T C | |
| G T C A | 14, 30 |

# Correct candidate regions




| K-mers | genomic positions | | K-mers | genomic positions | |
|---|---|---|---|---|---|
| A A A A | | -0 | A A A T | 1, 24 | -0 |
| A A A T | 1, 24 | -1 | A A T G | 2, 25 | -1 |
| A A T G | 2, 25 | -2 | A T G T | | -2 |
| A T G G | 3, 26 | -3 | T G T C | | -3 |
| T G G C | 4 | -4 | G T C A | 14, 30 | -4 |

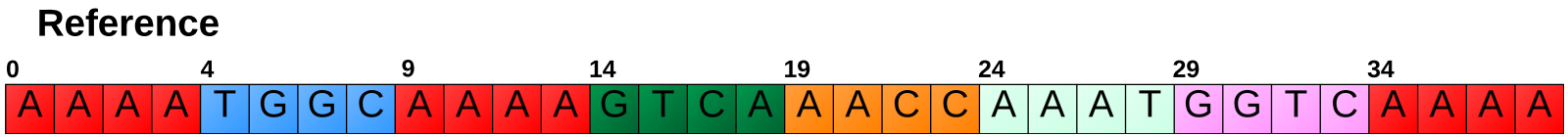
# Correct candidate regions




| K-mers | genomic positions |
|---|---|
| A A A A | |
| A A A T | 0, 23 |
| A A T G | 0, 23 |
| A T G G | 0, 23 |
| T G G C | 0 |

| K-mers | genomic positions |
|---|---|
| A A A T | 1, 24 |
| A A T G | 1, 24 |
| A T G T | |
| T G T C | |
| G T C A | 9, 25 |

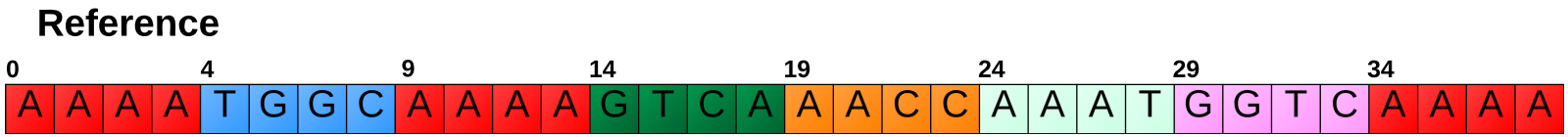
# Account for indels

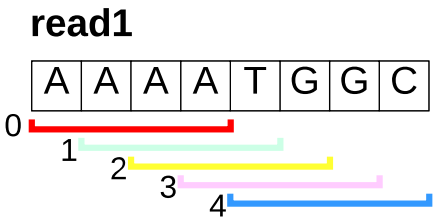


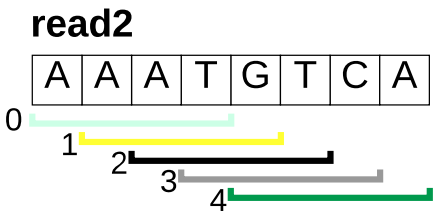

| K-mers | genomic positions | | K-mers | genomic positions | |
|---|---|---|---|---|---|
| A A A A | | /8 | A A A T | 1, 24 | /8 |
| A A A T | 0, 23 | /8 | A A T G | 1, 24 | /8 |
| A A T G | 0, 23 | /8 | A T G T | | /8 |
| A T G G | 0, 23 | /8 | T G T C | | /8 |
| T G G C | 0 | /8 | G T C A | 9, 25 | /8 |

# Account for indels





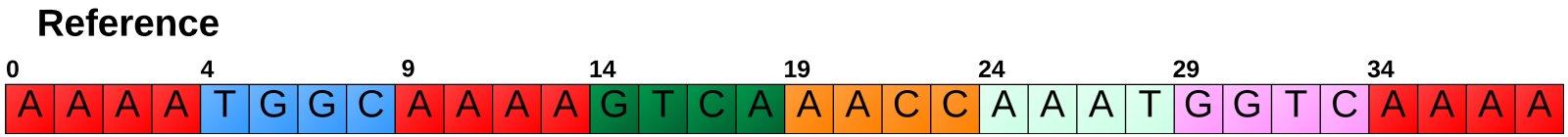



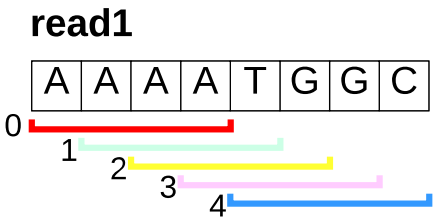

| K-mers | genomic positions |
|---|---|
| A A A A | |
| A A A T | 0, 2 |
| A A T G | 0, 2 |
| A T G G | 0, 2 |
| T G G C | 0 |

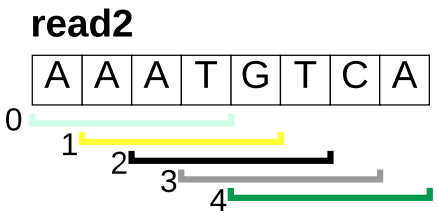
| K-mers | genomic positions |
|---|---|
| A A A T | 0, 3 |
| A A T G | 0, 3 |
| A T G T | |
| T G T C | |
| G T C A | 1, 3 |

# Account for indels





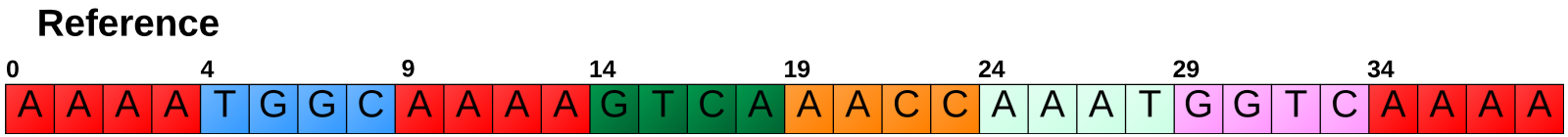



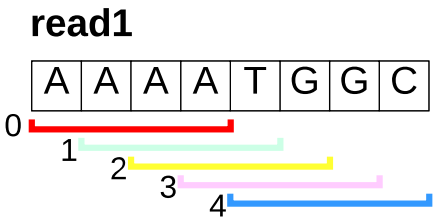



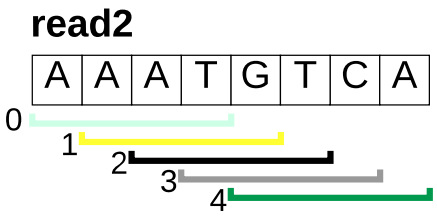

| K-mers | genomic positions | |
|---|---|---|
| A A A A | | *8 |
| A A A T | 0, 2 | *8 |
| A A T G | 0, 2 | *8 |
| A T G G | 0, 2 | *8 |
| T G G C | 0 | *8 |

| K-mers | genomic positions | |
|---|---|---|
| A A A T | 0, 3 | *8 |
| A A T G | 0, 3 | *8 |
| A T G T | | *8 |
| T G T C | | *8 |
| G T C A | 1, 3 | *8 |

# Account for indels




| K-mers | genomic positions |
|---|---|
| A A A A | |
| A A A T | 0, 16 |
| A A T G | 0, 16 |
| A T G G | 0, 16 |
| T G G C | 0 |

| K-mers | genomic positions |
|---|---|
| A A A T | 0, 24 |
| A A T G | 0, 24 |
| A T G T | |
| T G T C | |
| G T C A | 8, 24 |

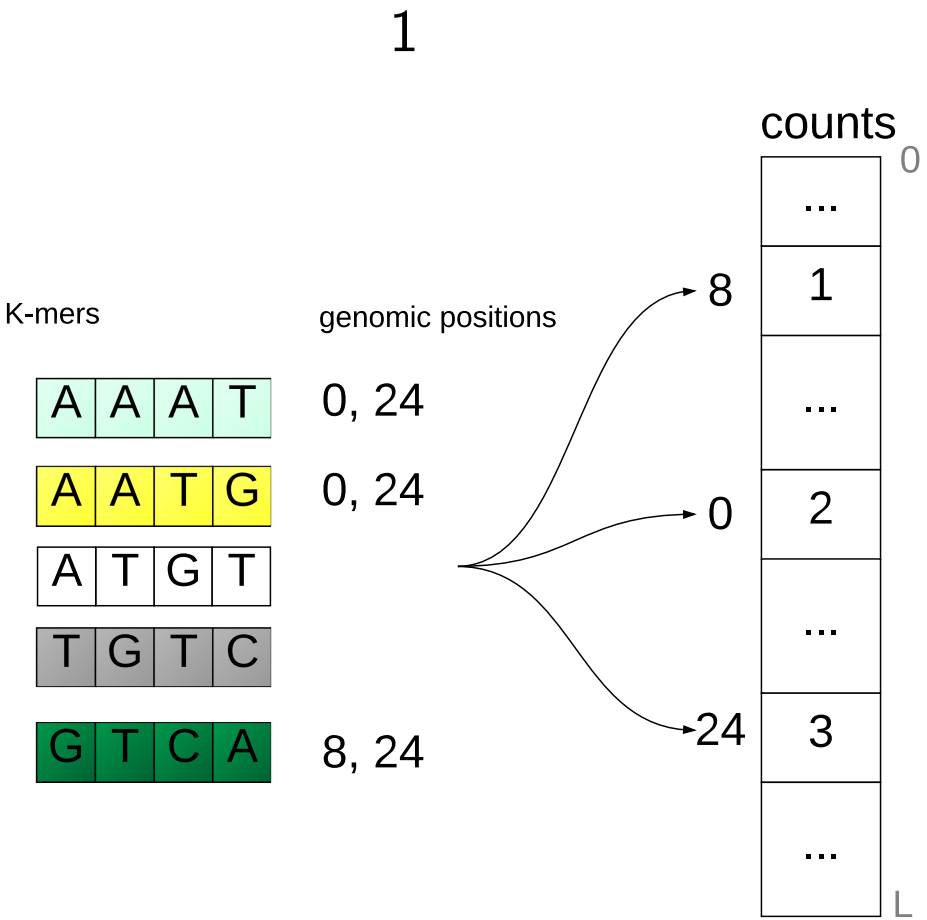
# Summarize candidate regions

**Multiplication hashing**

$$key_j = (p'_j * prim) \bmod L$$

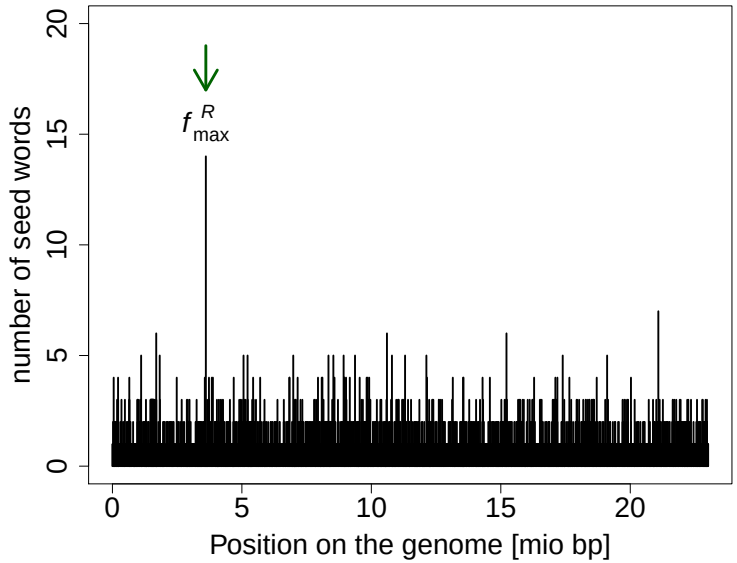
$$prim = 2654435761$$

Donald E. Knuth,1997



# Choosing candidate regions

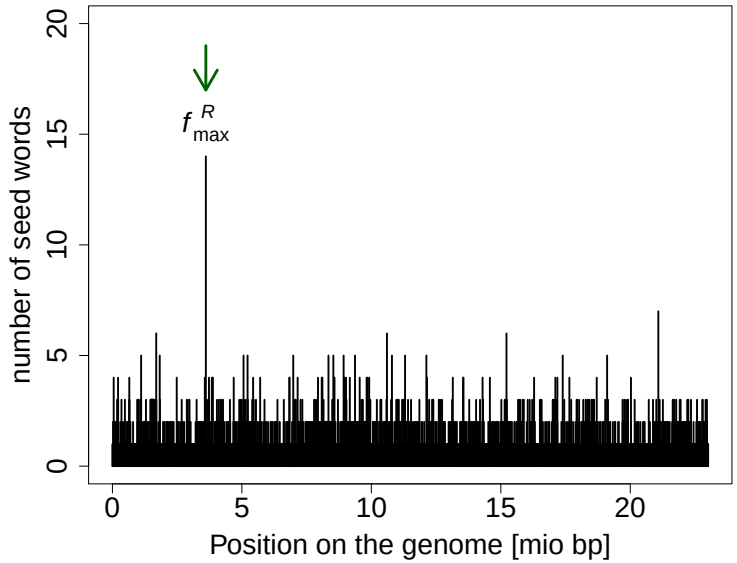


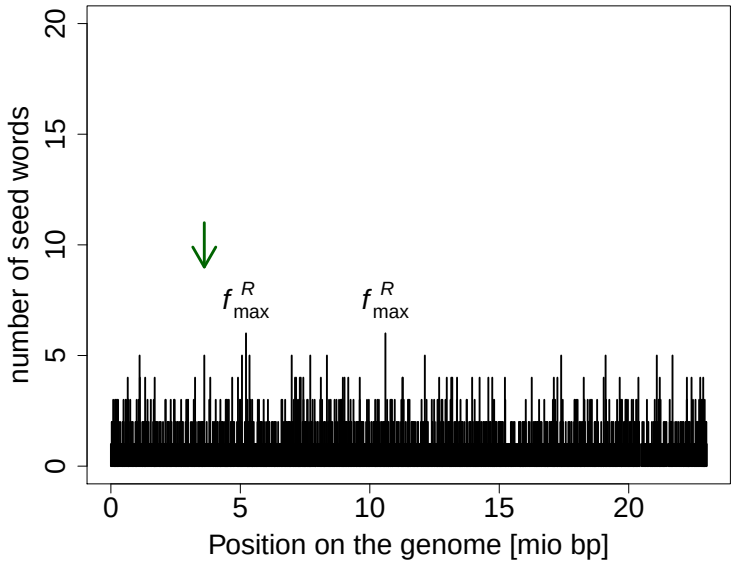



- Fixed threshold ($t = 2$) for all reads: 1614 alignment computations
- Choose $t$ relative to best matching position: 214 alignment computations

# Choosing candidate regions



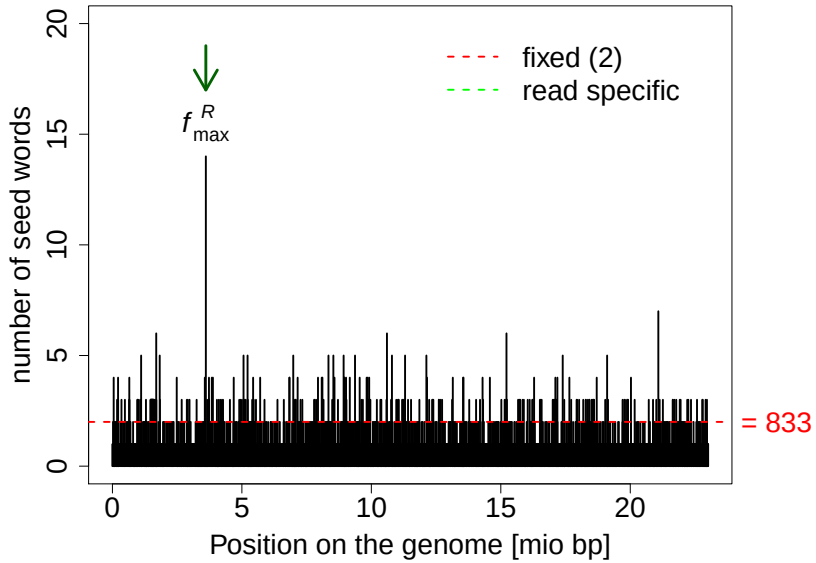


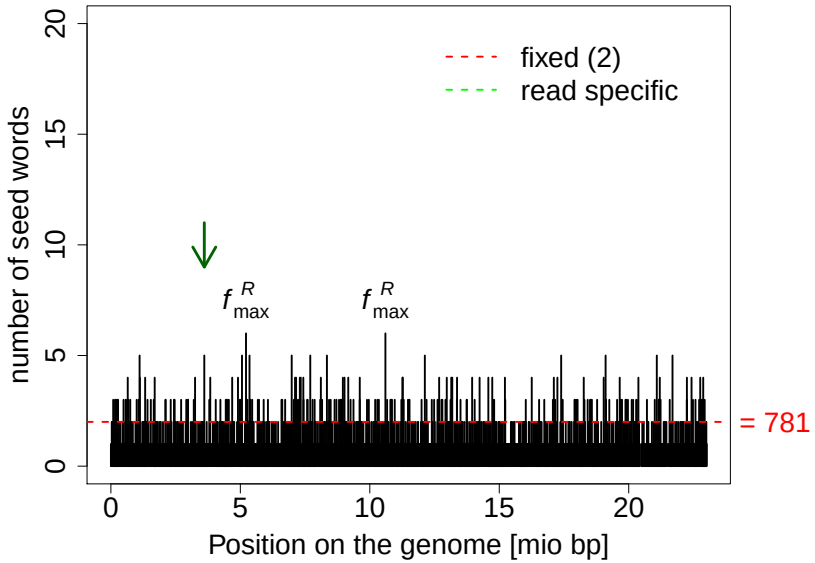




- Fixed threshold ($t = 2$) for all reads: 1614 alignment computations
- Choose $t$ relative to best matching position: 214 alignment computations

# Choosing candidate regions



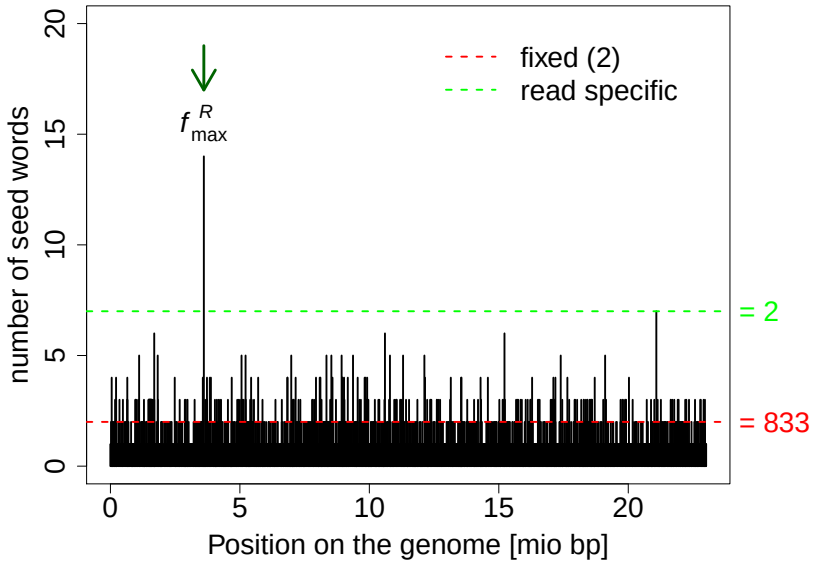


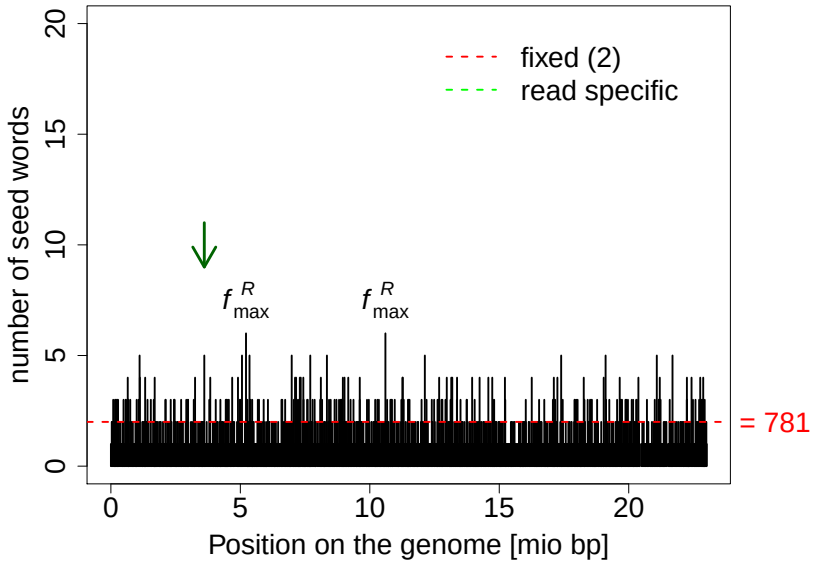




- Fixed threshold ($t = 2$) for all reads: 1614 alignment computations
- Choose $t$ relative to best matching position: 214 alignment computations

# Choosing candidate regions



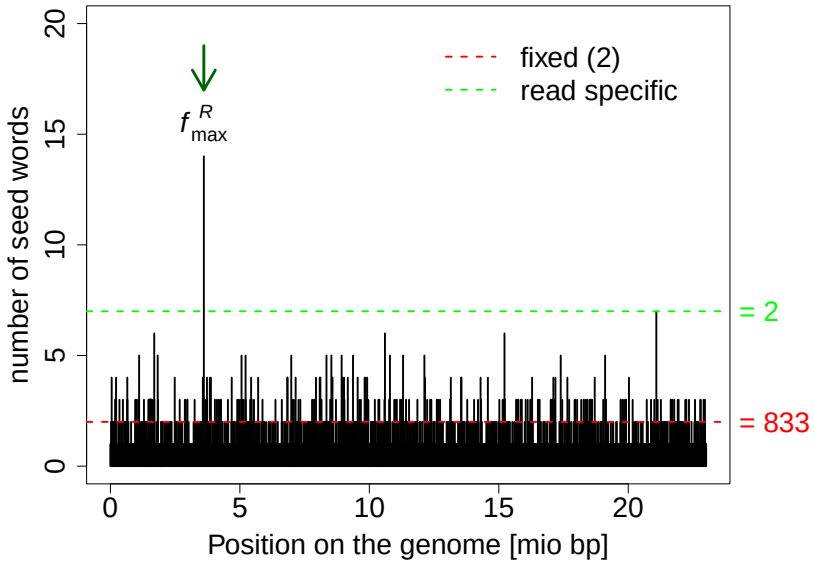


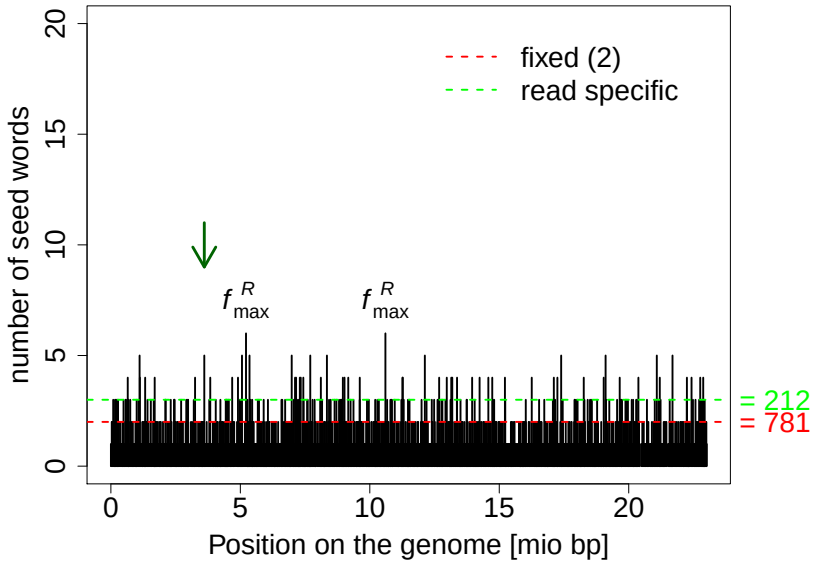




- Fixed threshold ($t = 2$) for all reads: 1614 alignment computations
- Choose $t$ relative to best matching position: 214 alignment computations

# Choosing candidate regions





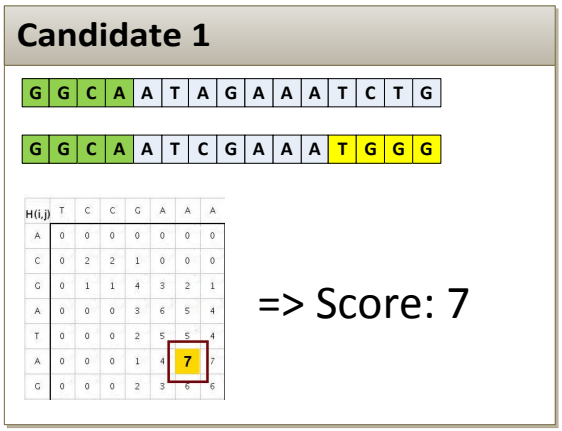



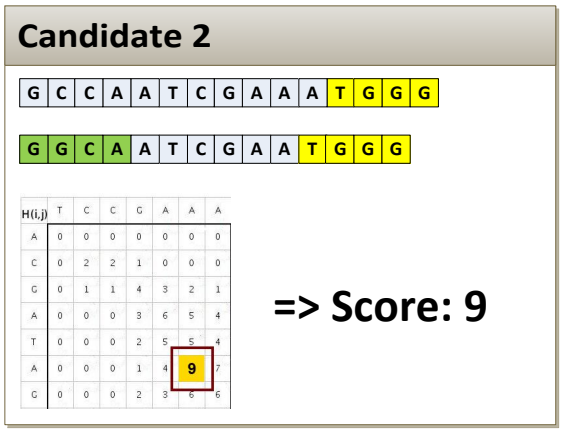

- Fixed threshold ($t = 2$) for all reads: 1614 alignment computations
- Choose $t$ relative to best matching position: 214 alignment computations

# Step 2 & 3: alignment computation



1. Candidate mapping region search: identification of perfectly matching sub regions using an index data structure
2. Computation of Smith-Waterman alignment scores for candidates
3. Computation of full alignment only for the highest scoring candidate



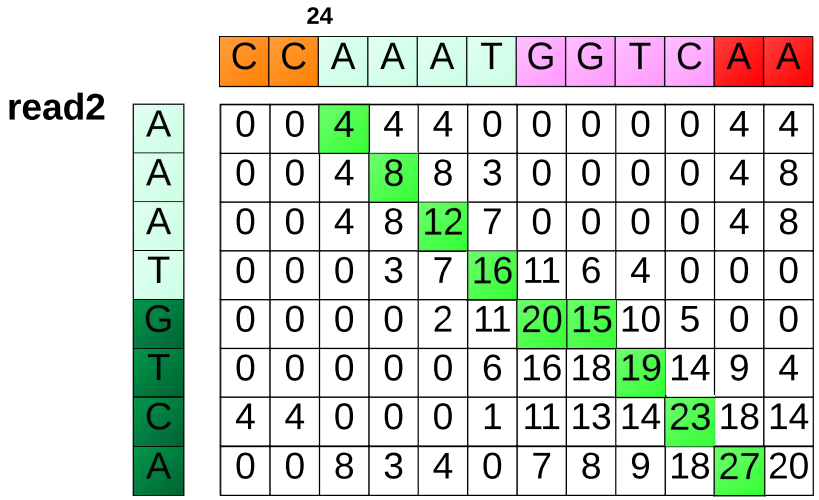

...

# Pairwise alignment

Complexity: $n \times m$

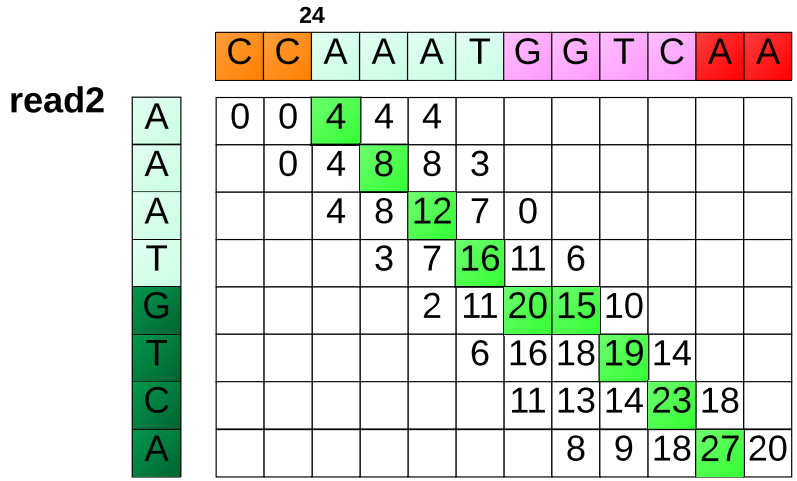
$n = 120bp$

$m = 100bp$



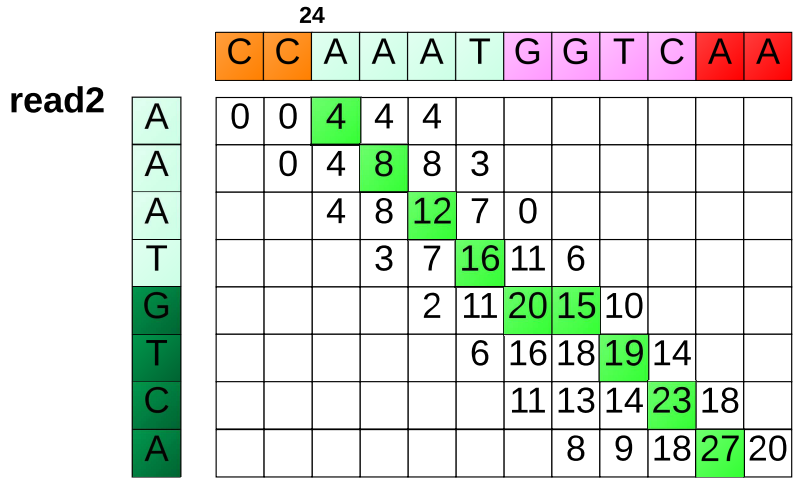
# Algorithmic optimizations

1. Banded alignment
   Complexity: $m \times c$
   $m = 100bp$
   $c \in \{1, \ldots, 40\}$
2. Split score and alignment computation
3. For score computation
   Computations, $m \times c$
   Memory, $c$



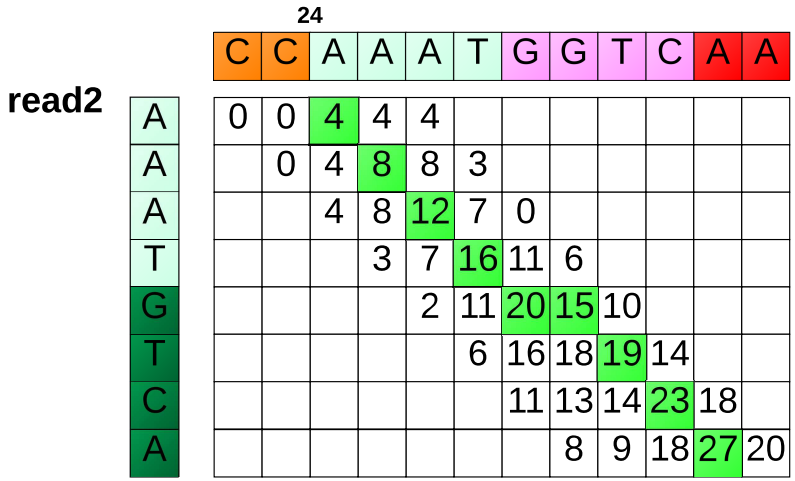
# Algorithmic optimizations

1. Banded alignment
   Complexity: $m \times c$
   $m = 100bp$
   $c \in \{1, \ldots, 40\}$
2. Split score and alignment computation
3. For score computation
   Computations, $m \times c$
   Memory, $c$

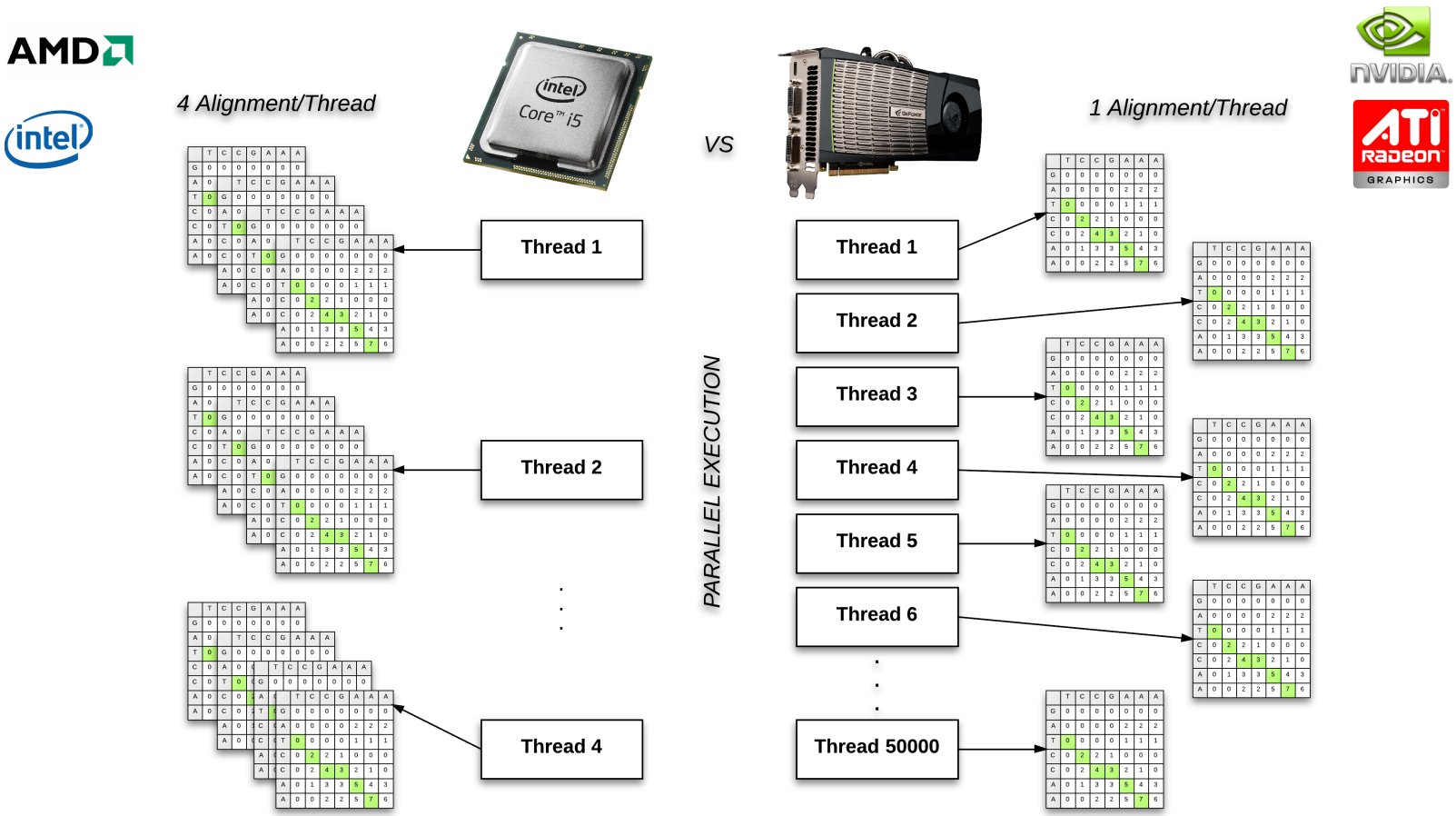


# Algorithmic optimizations

1. Banded alignment
   Complexity: $m \times c$
   $m = 100bp$
   $c \in \{1, \ldots, 40\}$
2. Split score and alignment computation
3. For score computation
   Computations, $m \times c$
   Memory, $c$



# Hardware optimizations

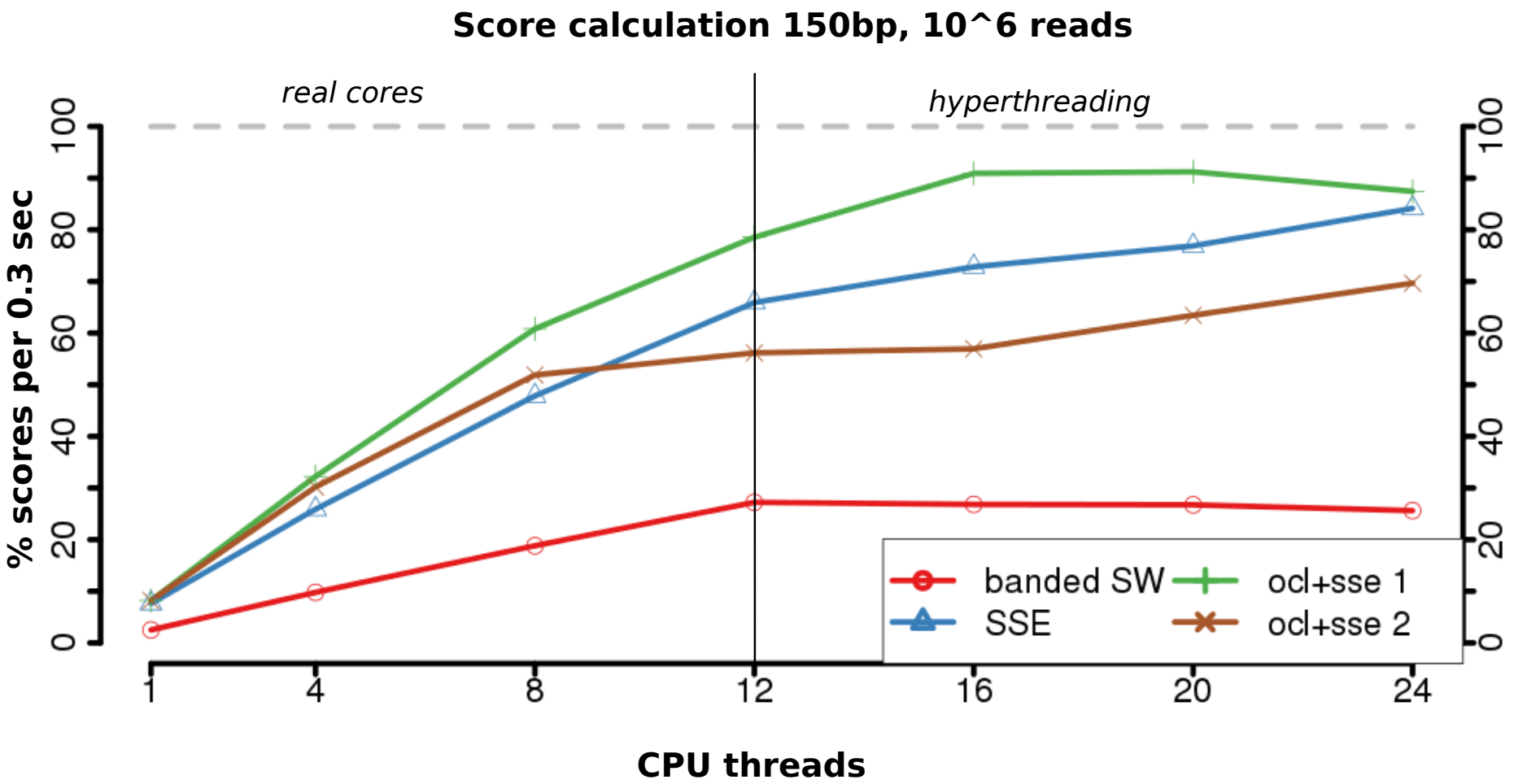



| OpenCL | |
|---|---|
| Hardware: | Intel i5-2500k |
| Sequences: | 1 million |
| Time (Scores): | 0.48 s |
| Time (Alignmets): | 2.18 s |

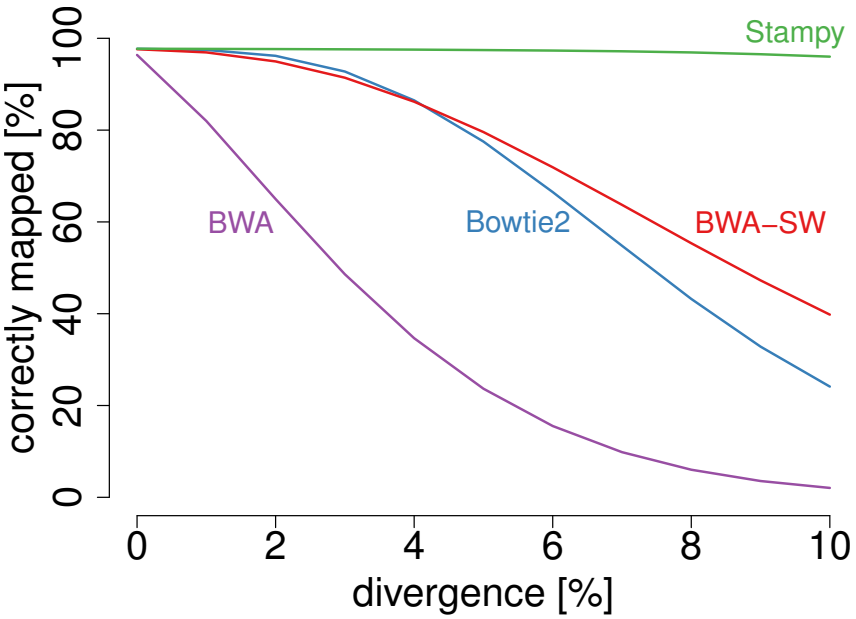
OpenCL

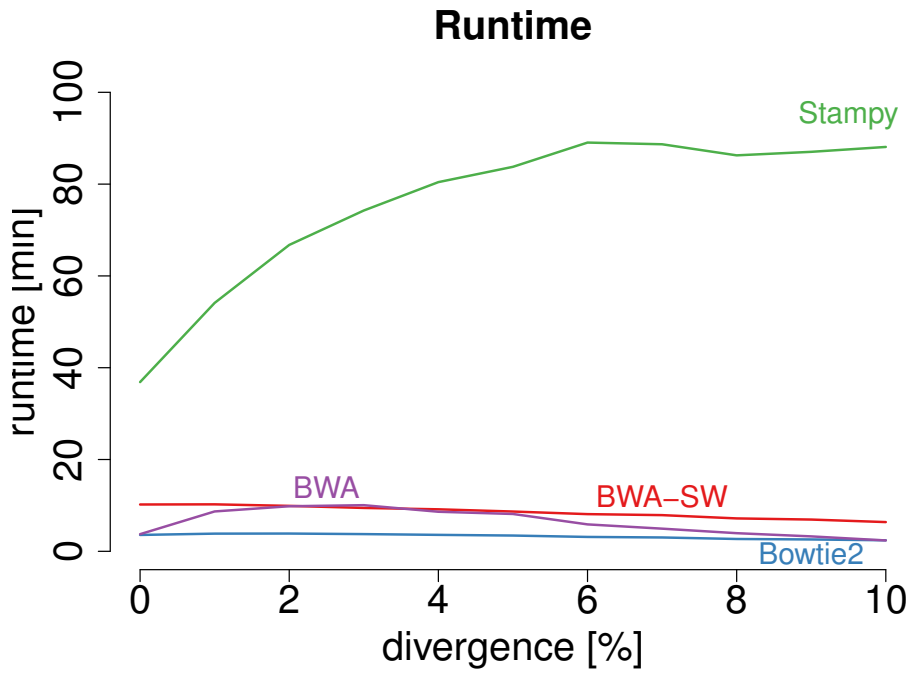
| OpenCL | |
|---|---|
| Hardware: | Nvidia GTX480 |
| Sequences: | 1 million |
| Time (Scores): | 0.25 s |
| Time (Alignmets): | 1.14 s |

# Hardware optimizations

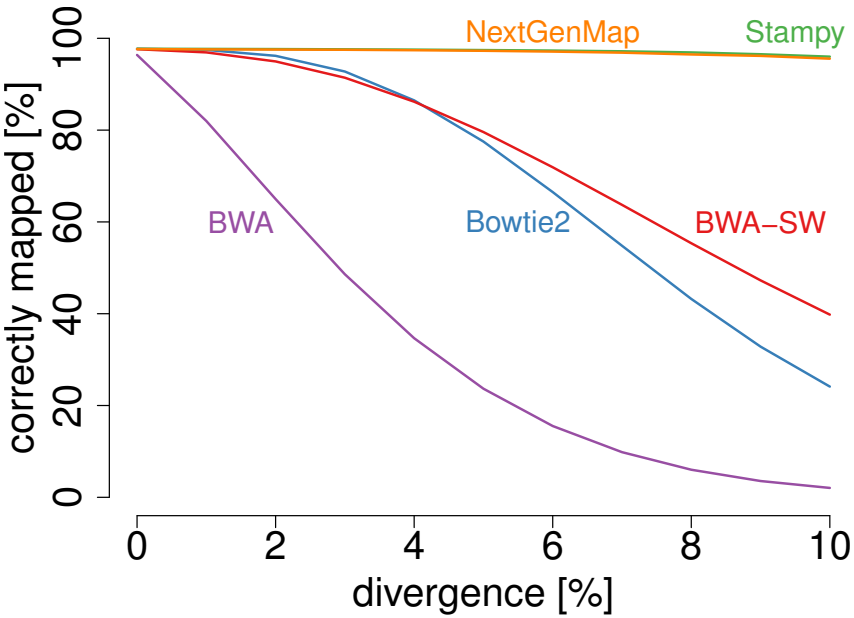


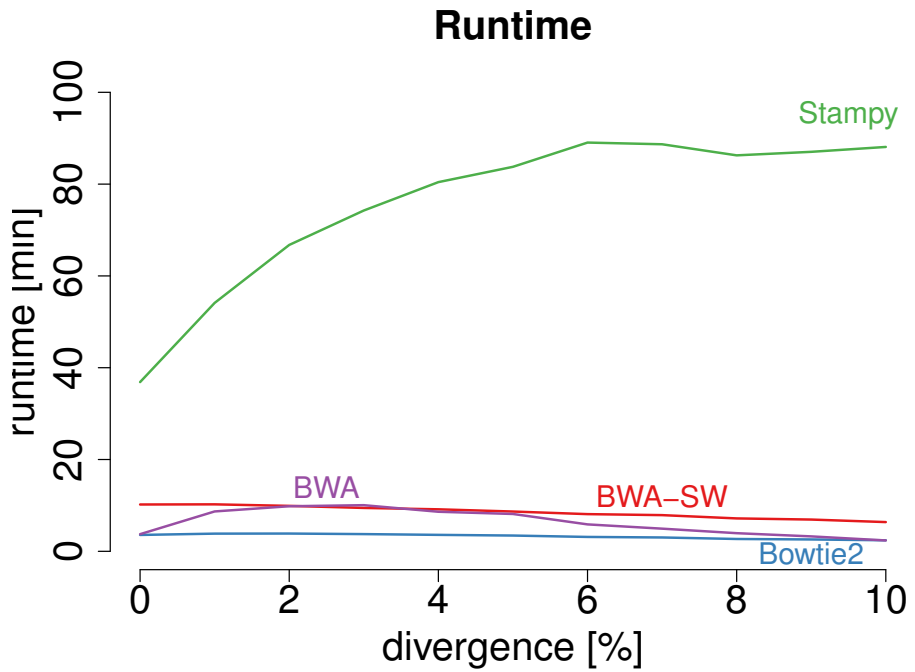
# Results


correctly mapped [%]
0
20
40
60
80
100
0
2
4
6
8
10
divergence [%]
Stampy
BWA
Bowtie2
BWA–SW

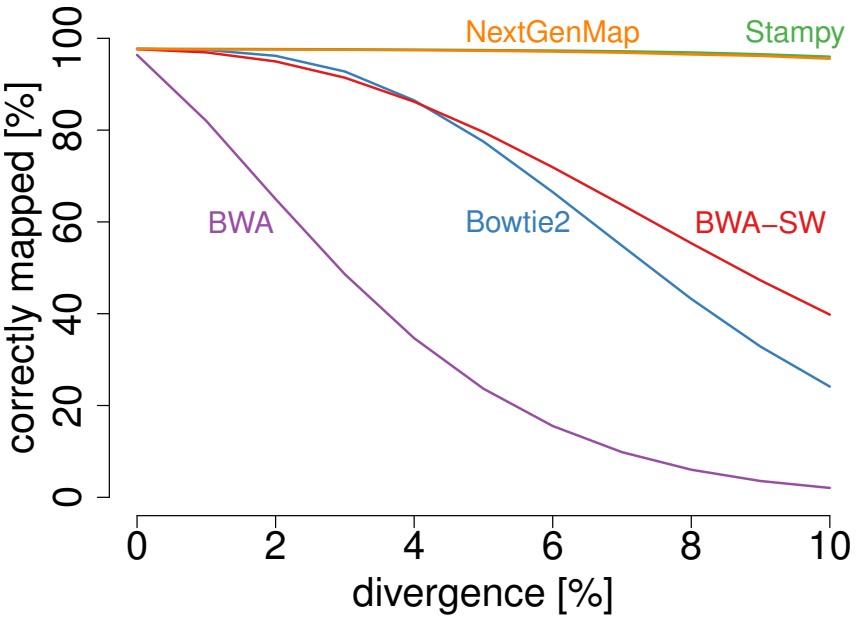

Runtime
runtime [min]
0
20
40
60
80
100
0
2
4
6
8
10
divergence [%]
Stampy
BWA
BWA–SW
Bowtie2

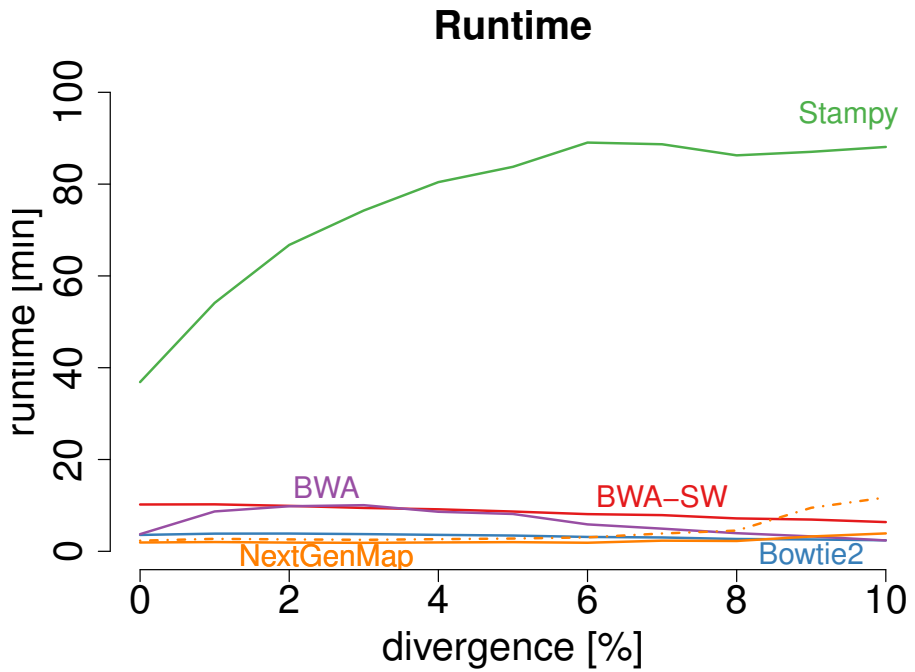
# Results


NextGenMap
Stampy
BWA
Bowtie2
BWA–SW
correctly mapped [%]
0
20
40
60
80
100
0
2
4
6
8
10
divergence [%]


Runtime
Stampy
BWA
BWA–SW
Bowtie2
runtime [min]
0
20
40
60
80
100
0
2
4
6
8
10
divergence [%]

# Results


NextGenMap
Stampy
BWA
Bowtie2
BWA–SW
correctly mapped [%]
0
20
40
60
80
100
0
2
4
6
8
10
divergence [%]


Runtime
Stampy
BWA
BWA–SW
NextGenMap
Bowtie2
runtime [min]
0
20
40
60
80
100
0
2
4
6
8
10
divergence [%]

# Applications



- Finding sRNAs in borrelia
- Identifying mutations in tumor tissues
- Sequencing based test for sepsis in patients

# Applications



- Finding sRNAs in borrelia
- Identifying mutations in tumor tissues
- Sequencing based test for sepsis in patients

# Applications



- Finding sRNAs in borrelia
- Identifying mutations in tumor tissues
- Sequencing based test for sepsis in patients

# Conclusion



**Read mapping:**

1. Finding the region of the reference genome that shows the highest similarity to a given read
2. It is one of the central steps in NGS data analysis
3. Size of genome, number of reads and repeating regions make it difficult to find the correct mapping location in reasonable time
4. Several approaches: sequence alignments (optimal), seed and extend (heuristic), tree based methods (heuristic)

**Seed and extend methods:**

1. Hash-table is used to quickly identify regions on the genome that are similar to a given read
2. Sequences alignments are used to find the region that shows the highest similarity to a read
3. Exploiting modern hardware reduces runtime required for alignment computations significantly