BLAST: Basic Local Alignment Search Tool

1705066 - Ataf Fazledin Ahamed
1705067 - Nishat Farhana Purbasha

Department of CSE
Bangladesh University of Engineering & Technology

July 13, 2021

What is BLAST?

What is BLAST

- Basic Local Alignment Search Tool
- Heuristic Algorithm
- Faster
  - FASTA
- Five Types

What is BLAST

- Basic Local Alignment Search Tool
- Heuristic Algorithm
- Faster
  - FASTA
- Five Types

What is BLAST

- Basic Local Alignment Search Tool
- Heuristic Algorithm
- Faster
  - FASTA
- Five Types

What is BLAST

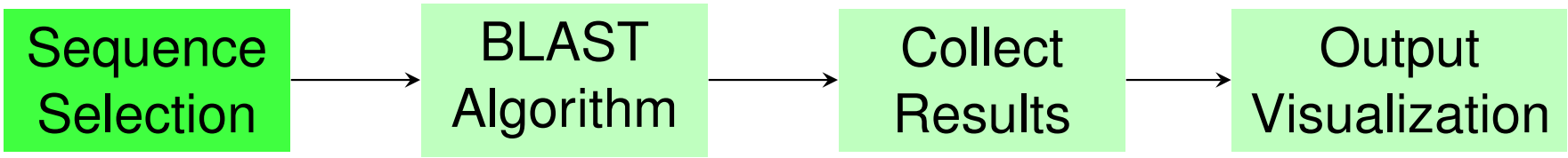
- Basic Local Alignment Search Tool
- Heuristic Algorithm
- Faster
  - FASTA
- Five Types

What is BLAST

Nature	Program	Query	Database
Nucleotide BLAST	blastn	Nucleotide (DNA, RNA)	Nucleotide (DNA, RNA)
Protein BLAST	blastp	Protein	Protein
Mixed BLAST	blastx	Translated Nucleotide	Protein
	tblastn	Protein	Translated Nucleotide
	tblastx	Translated Nucleotide	Translated Nucleotide

Table: Different types of BLAST

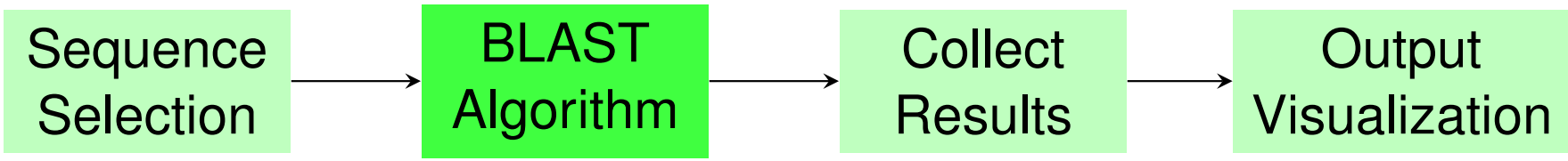
Overview



Database Selection:
nt (nucleotide)

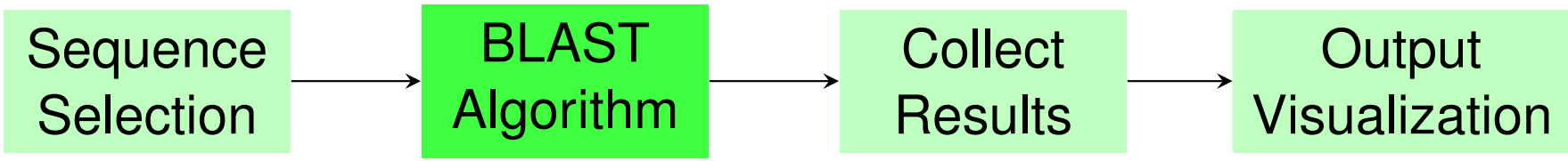
Query Sequence:
`ACTGAATCGCTA`

Overview



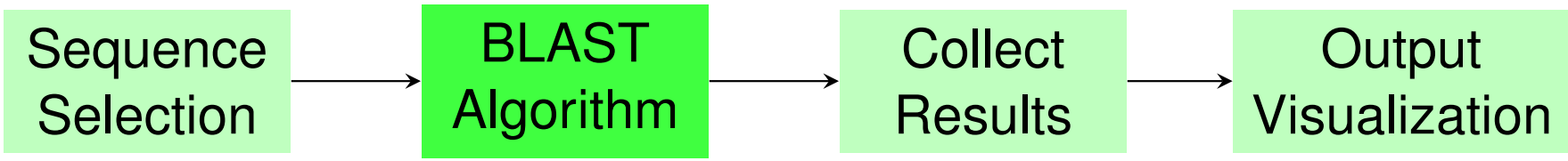
- Query & Database Sequence
- High-scoring Segment Pair (**HSP**)
- Extension of Local Alignment

Overview



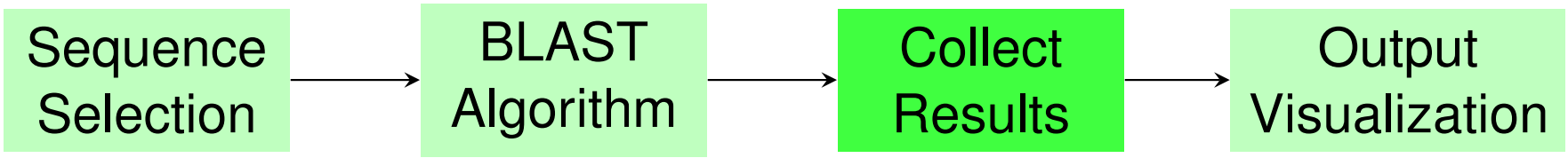
- Query & Database Sequence
- High-scoring Segment Pair (**HSP**)
- Extension of Local Alignment

Overview



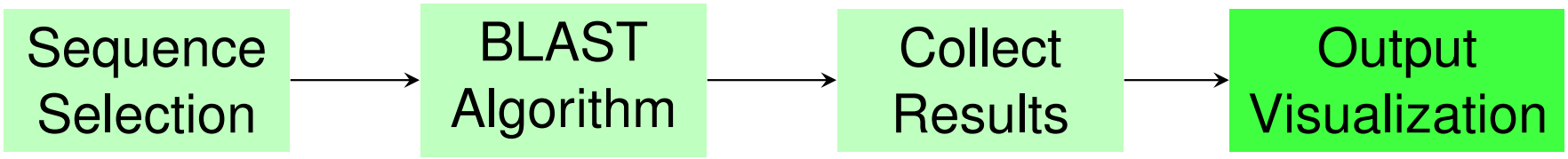
- Query & Database Sequence
- High-scoring Segment Pair (**HSP**)
- Extension of Local Alignment

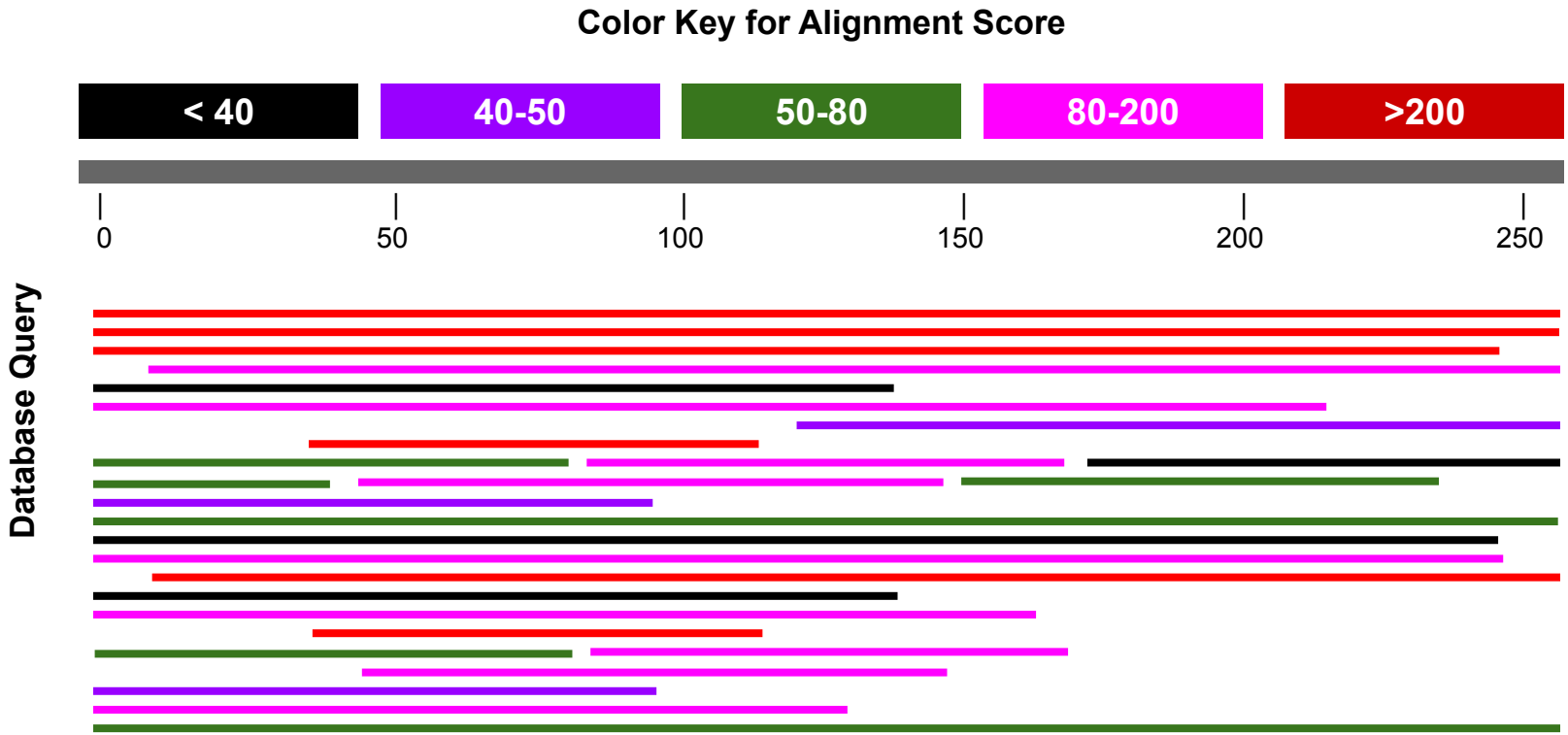
Overview




```
      Query: unknown_yakuba_sequence
        Hit: gi|1976360323|ref|XM_015191338.2| PREDICTED: Drosophila yakuba p...
Query range: [1014:2552] (1)
  Hit range: [3198:4736] (-1)
Quick stats: evalue 0; bitscore 2774.87
  Fragments: 1 (1538 columns)
      Query - TTATTTGTTGACAAAGAACGCTGGATTCGGGGATAAATTCGGCGGCATTGTTATCATGT~~~ACGCT
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||~~~|||||
        Hit - TTATTTGTTGACAAAGAACGCTGGATTCGGGGATAAATTCGGCGGCATTGTTATCATGT~~~ACGCT
```

Overview


Sequence Selection
BLAST Algorithm
Collect Results
Output Visualization


Color Key for Alignment Score
< 40
40-50
50-80
80-200
>200
0
50
100
150
200
250
Database Query

Basic Terminologies

Numeric Representation

Letter	Number
A	0
C	1
G	2
T	3

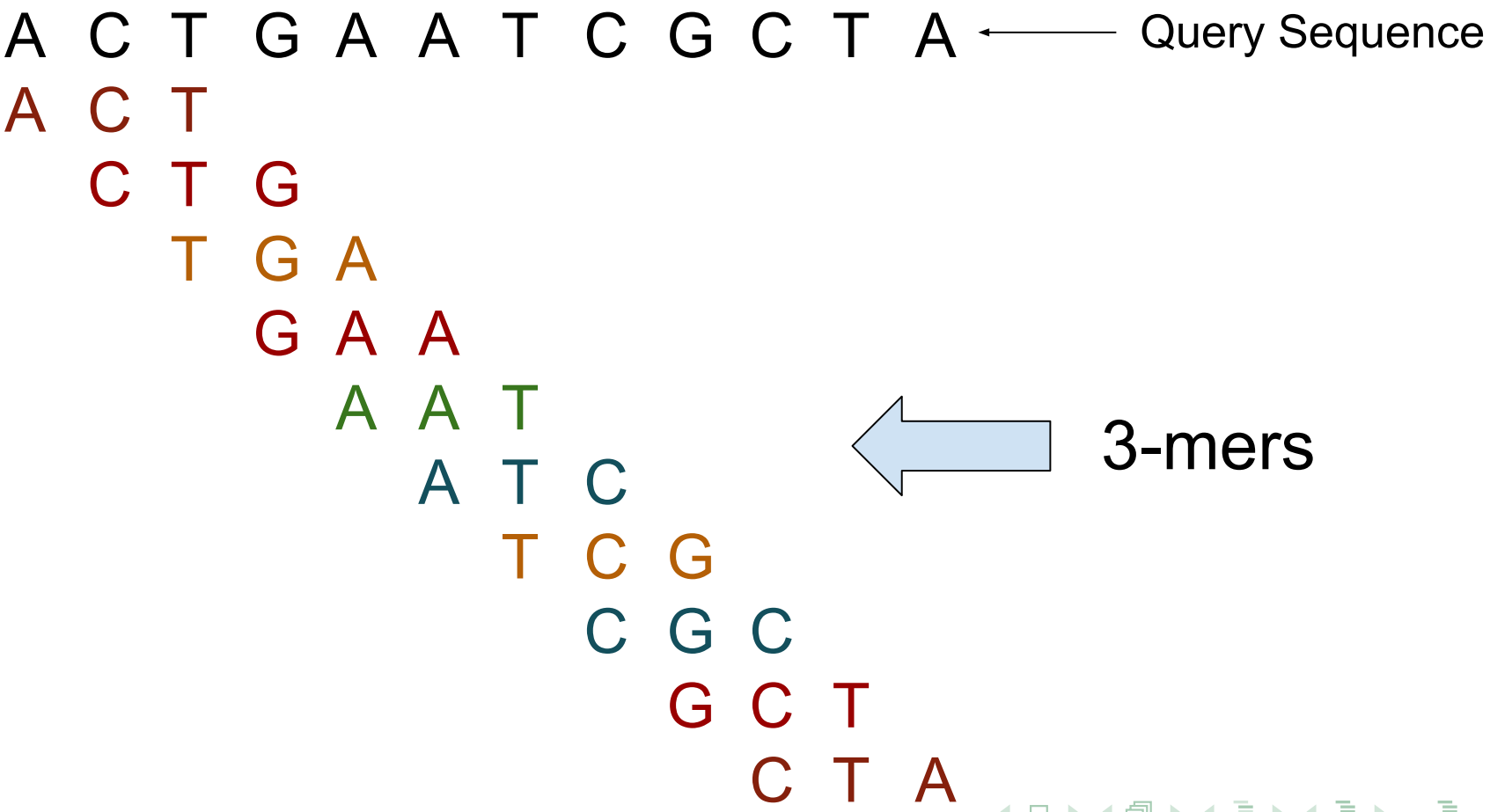
Table: Base 4 Representation

CTAG is represented as:

$$1 \times 4^3 + 3 \times 4^2 + 0 \times 4^1 + 2 \times 4^0 = 114$$

Basic Terminologies

K-mer



Example

BLAST Algorithm

Let us take an example!

Example: Pre-processing the database

Database Sequence: `GGACGGATTCC`

3-mers	Position	Key
GGA	1,5	40
GAC	2	33
ACG	3	6
CGG	4	26
GAT	6	35
ATT	7	15
TTC	8	61
TCC	9	53

Table: Generated 3-mers from DB sequence

3-mers	Position	Key
ACG	3	6
ATT	7	15
CGG	4	26
GAC	2	33
GAT	6	35
GGA	1,5	40
TCC	9	53
TTC	8	61

Table: Sorted 3-mers with respect to key

Example: Pre-processing the database

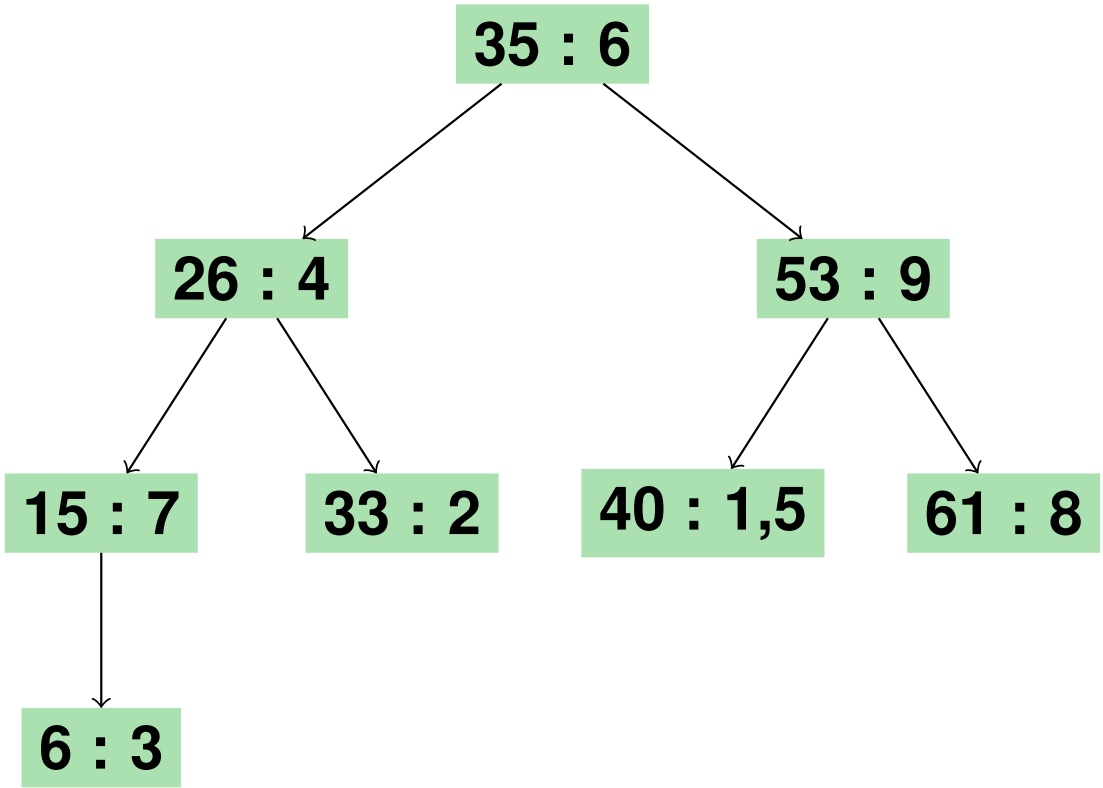


Figure: Constructed binary search tree

Example: Make k-mer of query sequence

Query Sequence: `ATCG`

3-mers	Key
ATC	13
TCG	54

Scoring Scheme	
Match	1
Mismatch	-1
Gap Insertion	-1
HSP Threshold	1

Example: Make k-mer of query sequence

Query Sequence: `ATCG`

Selected K-mers with minimum HSP threshold

3-mers	Key
ATC	13
CTC	45
GTC	49
TTC	61
AAC	1
ACC	5
AGC	9
ATA	12
ATG	14
ATT	15

3-mers	Key
TCG	54
ACG	6
CCG	24
GCG	38
TAG	50
TGG	58
TTG	62
TCA	52
TCC	53
TCT	59

Scoring Scheme	
Match	1
Mismatch	-1
Gap Insertion	-1
HSP Threshold	1

Example: Comparing k-mer with pre-processed data

Selected K-mers with minimum HSP threshold

3-mers	Key
ATC	13
CTC	45
GTC	49
TTC	61
AAC	1
ACC	5
AGC	9
ATA	12
ATG	14
ATT	15

3-mers	Key
TCG	54
ACG	6
CCG	24
GCG	38
TAG	50
TGG	58
TTG	62
TCA	52
TCC	53
TCT	59

Sorted k-mer of the database sequence

3-mers	Position	Key
ACG	3	6
ATT	7	15
CGG	4	26
GAC	2	33
GAT	6	35
GGA	1,5	40
TCC	9	53
TTC	8	61

Example: Comparing k-mer with pre-processed data

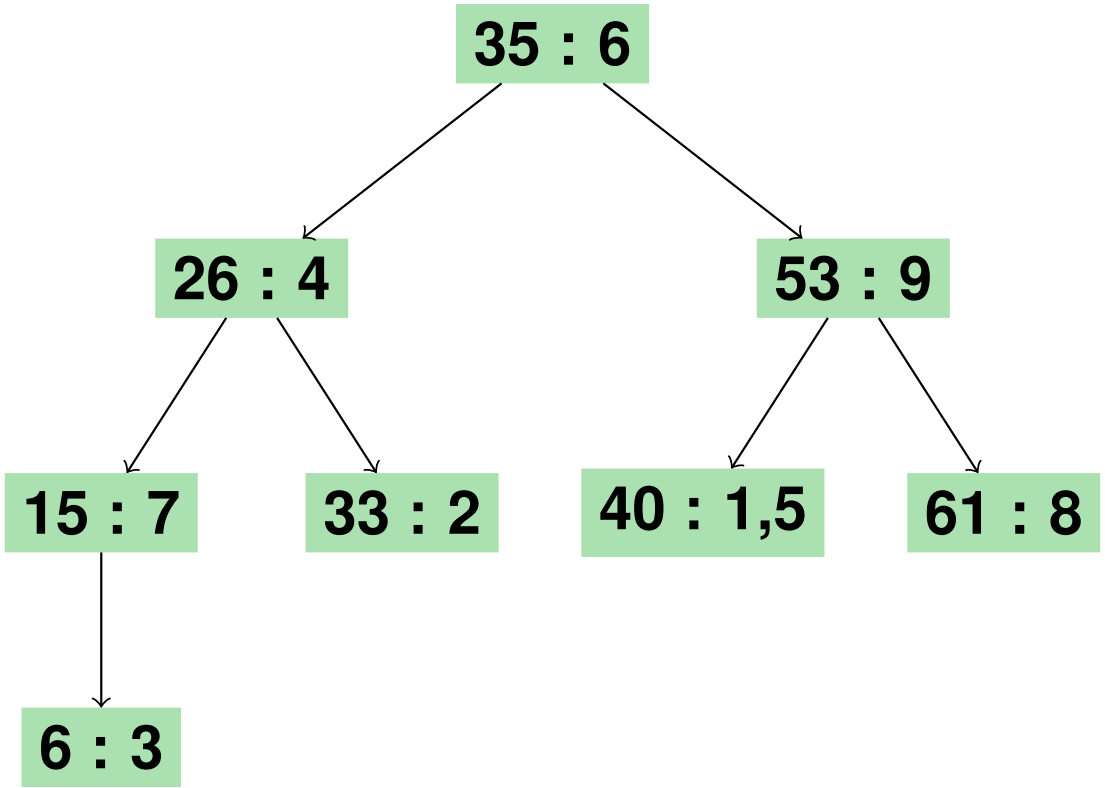


Figure: Constructed binary search tree

Selected k-mers:

Example: Comparing k-mer with pre-processed data

Selected K-mers with minimum HSP threshold

3-mers	Key
ATC	13
CTC	45
GTC	49
TTC	61
AAC	1
ACC	5
AGC	9
ATA	12
ATG	14
ATT	15

3-mers	Key
TCG	54
ACG	6
CCG	24
GCG	38
TAG	50
TGG	58
TTG	62
TCA	52
TCC	53
TCT	59

Sorted k-mer of the database sequence

3-mers	Position	Key
ACG	3	6
ATT	7	15
CGG	4	26
GAC	2	33
GAT	6	35
GGA	1,5	40
TCC	9	53
TTC	8	61

Example: Comparing k-mer with pre-processed data

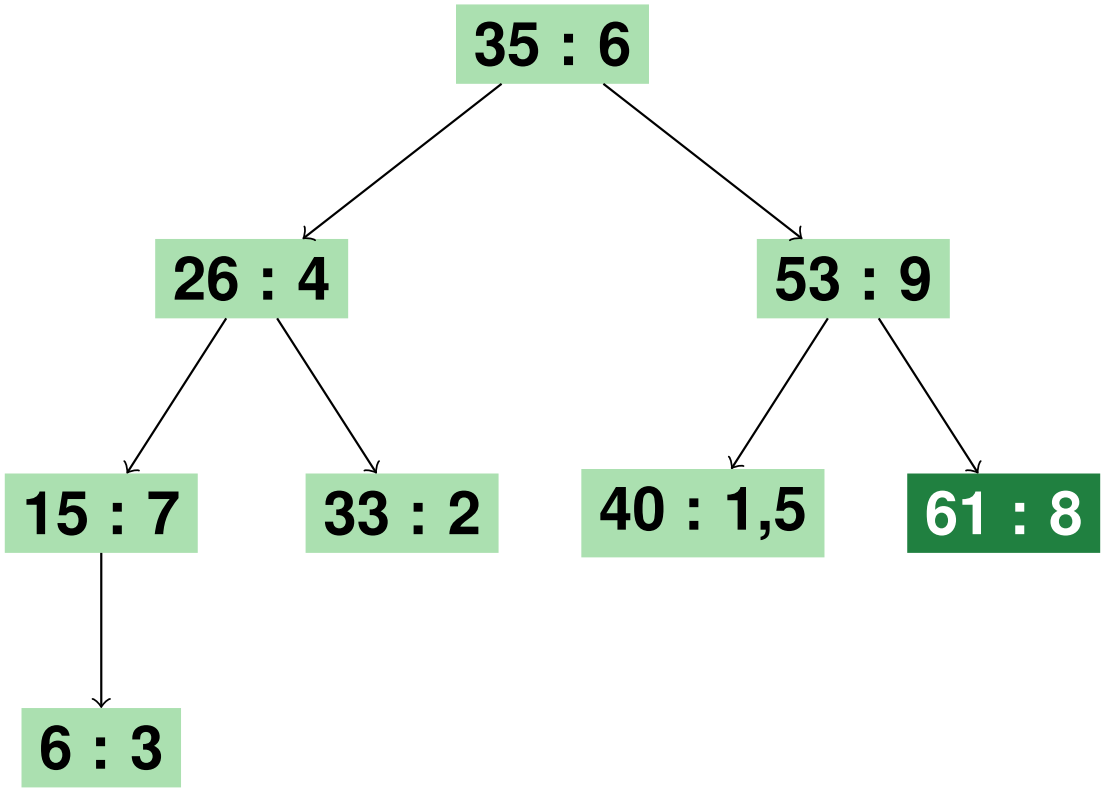


Figure: Constructed binary search tree

Selected k-mers: `(61:8)`

Example: Comparing k-mer with pre-processed data

Selected K-mers with minimum HSP threshold

3-mers	Key
ATC	13
CTC	45
GTC	49
TTC	61
AAC	1
ACC	5
AGC	9
ATA	12
ATG	14
ATT	15

3-mers	Key
TCG	54
ACG	6
CCG	24
GCG	38
TAG	50
TGG	58
TTG	62
TCA	52
TCC	53
TCT	59

Sorted k-mer of the database sequence

3-mers	Position	Key
ACG	3	6
ATT	7	15
CGG	4	26
GAC	2	33
GAT	6	35
GGA	1,5	40
TCC	9	53
TTC	8	61

Example: Comparing k-mer with pre-processed data

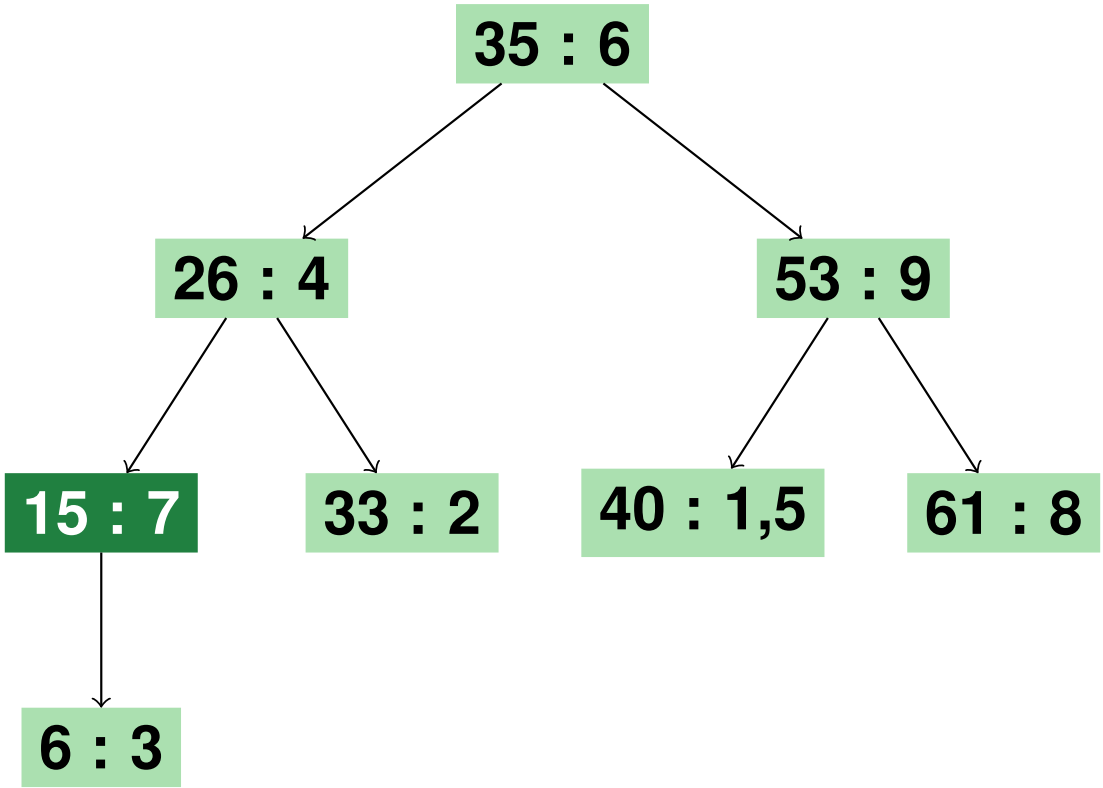


Figure: Constructed binary search tree

Selected k-mers: `(61:8), (15:7)`

Example: Comparing k-mer with pre-processed data

Selected K-mers with minimum HSP threshold

3-mers	Key
ATC	13
CTC	45
GTC	49
TTC	61
AAC	1
ACC	5
AGC	9
ATA	12
ATG	14
ATT	15

3-mers	Key
TCG	54
ACG	6
CCG	24
GCG	38
TAG	50
TGG	58
TTG	62
TCA	52
TCC	53
TCT	59

Sorted k-mer of the database sequence

3-mers	Position	Key
ACG	3	6
ATT	7	15
CGG	4	26
GAC	2	33
GAT	6	35
GGA	1,5	40
TCC	9	53
TTC	8	61

Example: Comparing k-mer with pre-processed data

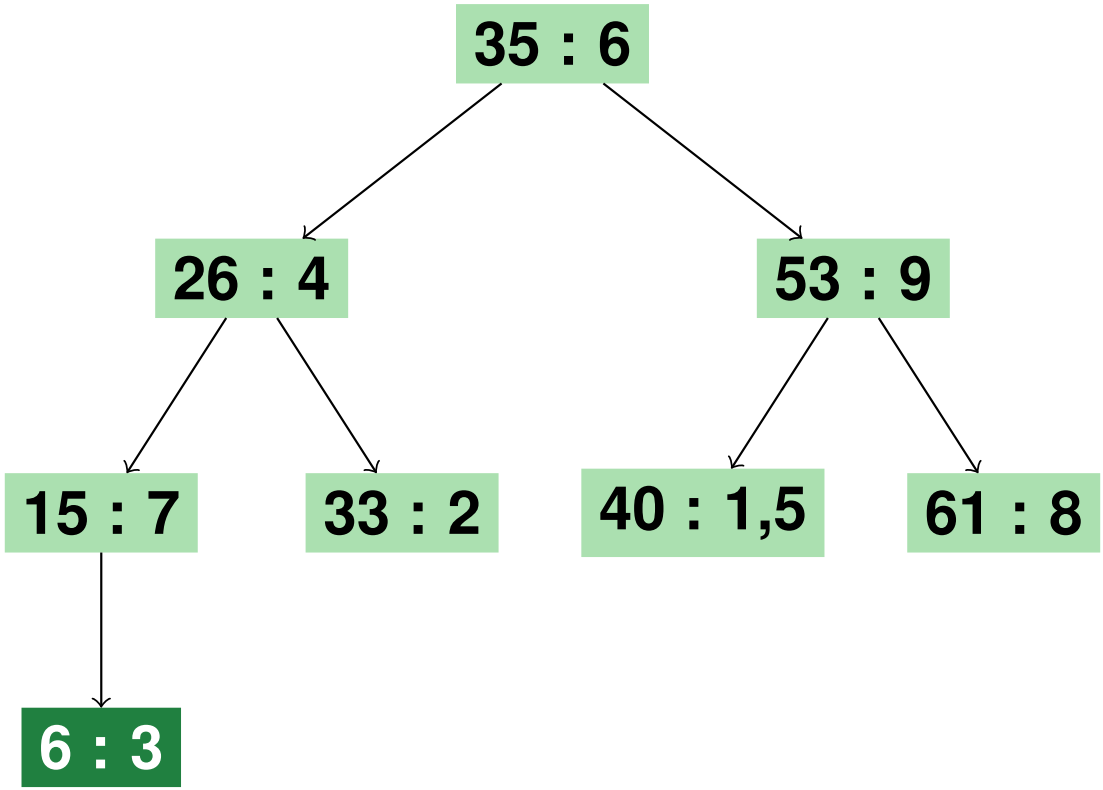


Figure: Constructed binary search tree

Selected k-mers: `(61:8), (15:7), (6:3)`

Example: Comparing k-mer with pre-processed data

Selected K-mers with minimum HSP threshold

3-mers	Key
ATC	13
CTC	45
GTC	49
TTC	61
AAC	1
ACC	5
AGC	9
ATA	12
ATG	14
ATT	15

3-mers	Key
TCG	54
ACG	6
CCG	24
GCG	38
TAG	50
TGG	58
TTG	62
TCA	52
TCC	53
TCT	59

Sorted k-mer of the database sequence

3-mers	Position	Key
ACG	3	6
ATT	7	15
CGG	4	26
GAC	2	33
GAT	6	35
GGA	1,5	40
TCC	9	53
TTC	8	61

Example: Comparing k-mer with pre-processed data

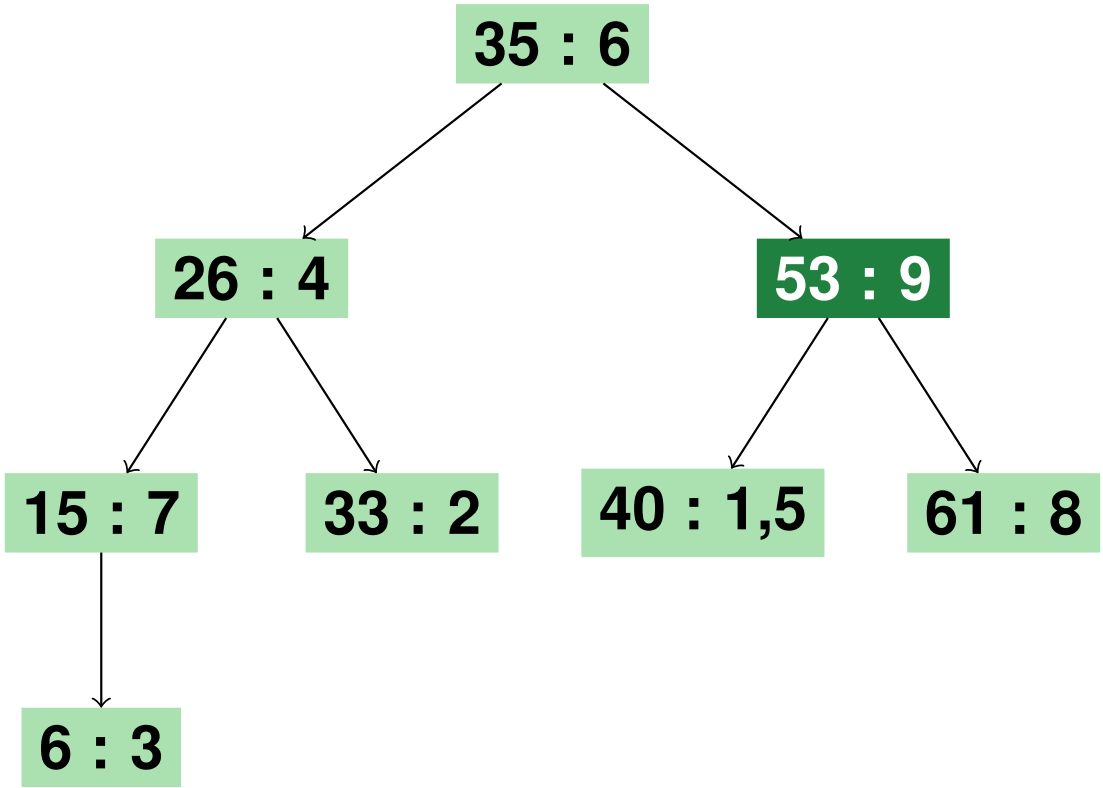


Figure: Constructed binary search tree

Selected k-mers: `(61:8), (15:7), (6:3), (53:9)`

Example: Alignment Extension

	G	G	A	C	G	G	A	T	T	C	C
A							X	X			
T			X					X	X		
C				X					X	X	
G					X						X

Table: Seeding

Example: Alignment Extension

	-	G	G	A	C	G	G	A	T	T	C	C
-												
A								1				
T									2	1		
C					1				1	1	2	1
G						2	1				1	1

Table: Extension of alignment

Alignment Extension

Smith-Waterman algorithm is performed on selected cells for determining similar regions between two sequences.

Example: Alignment Extension

	-	G	G	A	C	G	G	A	T	T	C	C
-												
A								1				
T									2	1		
C					1				1	1	2	1
G						2	1				1	1

Table: Extension of alignment

Alignment Extension

C G G

C G –

Example: Alignment Extension

	-	G	G	A	C	G	G	A	T	T	C	C
-												
A								1				
T									2	1		
C					1				1	1	2	1
G						2	1				1	1

Table: Extension of alignment

Alignment Extension

A T –

A T C

Example: Alignment Extension

	-	G	G	A	C	G	G	A	T	T	C	C
-												
A								1				
T									2	1		
C					1				1	1	2	1
G						2	1				1	1

Table: Extension of alignment

Alignment Extension

A T T

A T C

Example: Alignment Extension

	-	G	G	A	C	G	G	A	T	T	C	C
-												
A								1				
T									2	1		
C					1				1	1	2	1
G						2	1				1	1

Table: Extension of alignment

Alignment Extension

A T T C C

A T – C G

Example: Alignment Extension

	-	G	G	A	C	G	G	A	T	T	C	C
-												
A								1				
T									2	1		
C					1				1	1	2	1
G						2	1				1	1

Table: Extension of alignment

Alignment Extension

A T T C C

A T – C –

Example: Alignment Extension

	-	G	G	A	C	G	G	A	T	T	C	C
-												
A								1				
T									2	1		
C					1				1	1	2	1
G						2	1				1	1

Table: Extension of alignment

Alignment Extension

A T T C –

A T – C G

Example: Alignment Extension

C G G C G –	A T T C C A T – C G
A T T A T C	A T T C C A T – C –
A T – A T C	A T T C – A T – C G

Alignments

Algorithm Complexity

$$n_{k-mer}$$

in X length string = X - K + 1

M length Database,

$$n_{k-mer} = M - K + 1$$

L length Query,

$$n_{k-mer} = L - K + 1$$

Algorithm Complexity

$$n_{k-mer}$$

in X length string = X - K + 1

M length Database,

$$n_{k-mer} = M - K + 1$$

L length Query,

$$n_{k-mer} = L - K + 1$$

Algorithm Complexity

$$n_{k-mer}$$

in X length string = X - K + 1

M length Database,

$$n_{k-mer} = M - K + 1$$

L length Query,

$$n_{k-mer} = L - K + 1$$

Algorithm Complexity

$$O(\{L\text{-}K+1\} \times \log_2\{M - K + 1\})$$

K = k-mer length
L = Length of query sequence
M = Total length of database sequences

Algorithm Complexity

Since K = constant,

$$O(L \times \log_2 M)$$

L = Length of query sequence
M = Total length of database sequences

Resources

- https://doi.org/10.1016/S0022-2836(05)80360-2
- https://doi.org/10.1016/0022-2836(81)90087-5
- https://blast.ncbi.nlm.nih.gov/Blast.cgi
- https://www.youtube.com/channel/UC8kHK9I5NxHmW0j-RcWQ8cg

End of Slide

Thank You