Sequence Alignment: Motivation and Algorithms
Motivation and Introduction
Importance of Sequence Alignment

• For DNA, RNA and amino acid sequences, high sequence similarity usually implies significant functional or structural similarity.
  – imply structure from the sequence similarity
  – suggest gene function

• Frequently, we aren’t interested in exact matches of the sequences but approximate matches.
Inexact Alignment

Where is GATTACA approximately in the human genome? And how do we efficiently find them?

Answers to these questions depend on:

1. The definition of “approximately”
   - Hamming Distance, Edit distance, Sequence Similarity

2. Efficiency depends on the data characteristics and goals
In this lecture...

• Sequence alignment and alignment software are used all over bioinformatics for different purposes.

• The goal of this lecture is to understand sequence alignment, motivate it in the context of assembly, and seeing what software is out there.
Two examples where you would use sequence similarity

1. Use a known genome to help assemble an unknown genome.

2. Annotating a genome.
Two examples where you would use sequence similarity

1. Use a known genome to help assemble an unknown genome.

2. Annotating a genome.
What is a reference genome?

• A **reference genome** is a representative example of a species’ set of genes.

• As they are often assembled from the sequencing of DNA from a number of donors, reference genomes do not accurately represent the set of genes of any individual.

• BUT the reference is very, very, VERY similar in many locations to the individual.
Sample Preparation
Sample Preparation

Fragments
Sample Preparation → Fragments → Sequencing

Next Generation Sequencing (NGS)

Reads:
- ACGTAGAATCGACCATG
- GGGACGTAGAATACGAC
- ACGTAGAATACGTAGAA
Sample Preparation

Fragments

Sequencing

Reads

Assembly

ACGTAGAATACGTAGAA
ACGTAGAATCGACCATG
GGGACGTAGAATACGAC

Contigs

ACGTAGAATACGTAGAAACAGATTAGAGAG...
Sample Preparation

Fragments

Sequencing

Reads

Assembly

ACGTAGAATACGTAGAA
ACGTAGAATCGACCATG
GGGACGTAGAATACGAC

BUT WE DON’T HAVE TO BEGIN FROM SCRATCH!!!

Contigs

ACGTAGAATAACGTAGAAACAGATTAGAGA...

ACGTAGAATACGTAGAAACAGATTAGAGAG...
How do we use the reference?

• The reads from the individual will be very similar but not identical to the reference genome.

• Ideally, we would like to use the information from the reference! Otherwise, we are throwing up valuable information.

• Hence, the first step is to find where in the genome the sequences are similar to.
Two examples where you would use sequence similarity

1. Use a known genome to help assemble an unknown genome.

2. Annotating a genome.
What percent of your genes do you share?
Sequence Alignment

• Before we can make comparative statements about two sequences, we have to produce a pairwise sequence alignment
• What is the optimal alignment between two sequences?
• Match/mismatch? Gaps/extensions? Is an optimal alignment always significant?
Edit Distance

Levenshtein (1966) introduced edit distance between two strings as the minimum number of elementary operations (insertions, deletions, and substitutions) to transform one string into the other

\[ d(v, w) \]: the minimum number of operations required to transform \( v \) into \( w \).
**Edit Distance vs Hamming Distance**

**Hamming distance** always compares the $i^{\text{th}}$ letter of $v$ with the $j^{\text{th}}$ letter of $w$.

- $V = \text{ATATATAT}$
- $W = \text{TATATATA}$

- $d(v, w) = 8$
- computing Hamming distance is a trivial task

**Edit distance** may compare the $i^{\text{th}}$ letter of $v$ with the $j^{\text{th}}$ letter of $w$.

- $V = \text{--ATATATAT}$
- $W = \text{TATATATA--}$

- $d(v, w) = 2$
- computing edit distance is a non-trivial task
Edit Distance vs Hamming Distance

**Hamming distance** always compares the \( i^{th} \) letter of \( v \) with the \( j^{th} \) letter of \( w \)

\[
V = ATATATAT \\
W = TATATATA
\]

\[
d(v, w) = 8
\]

**Edit distance** may compare the \( i^{th} \) letter of \( v \) with the \( j^{th} \) letter of \( w \)

\[
V = -ATATATAT \\
W = TATATATA-
\]

\[
d(v, w) = 2 \\
\text{(one insertion and one deletion)}
\]

How to find what \( j \) goes with what \( i \) such that the “score” of the alignment is optimized?
Edit Distance: Example

TGCATAT $\rightarrow$ ATCCGAT in 5 steps

TGCATAT $\rightarrow$ (delete last T)
TGCATA $\rightarrow$ (delete last A)
TGCAT $\rightarrow$ (insert A at front)
ATGCAT $\rightarrow$ (substitute C for 3\textsuperscript{rd} G)
ATCCAT $\rightarrow$ (insert G before last A)
ATCCGAT $\rightarrow$ (Done)
Computing Edit Distance
Dynamic Programming Algorithm

• **Dynamic Programming** is a method for solving complex problems by breaking them down into simpler subproblems.

• Provides the best (optimal) alignment between two sequences.

• Includes matches, mismatches, and gaps to maximize the number of matched characters.

• Score: match, mismatch, gap (affine vs. non-affine)
Dynamic Programming Visualized

Corresponding path

(0,0), (1,1), (2,2), (2,3), (3,4), (4,5), (5,5), (6,6), (7,6), (7,7)
and \( \downarrow \) represent indels in \( v \) and \( w \) with score 0.

\( \downarrow \) and \( \implies \) represent matches with score 1.

The score of the alignment path is 5.
Every path in the edit graph corresponds to an alignment:

ATC GTTAT
ATCGT - A - C
Define Rules

• Score for match: +1
• Score for mismatch: 0
• Score for gap: 0

Let $s_{i,j}$ be the total score at position $i, j$ then we have that it is equal to the maximum of \{ $s_{i-1, j-1} + 1$ (if $v_i = w_j$), $s_{i-1, j}$ (otherwise: gap or mismatch) \}
Dynamic Programming Example

Initialize $1^{st}$ row and $1^{st}$ column to be all zeroes.

Or, to be more precise, initialize $0^{th}$ row and $0^{th}$ column to be all zeroes.
Dynamic Programming Example

\[ S_{i,j} = \text{maximum of :} \]
\[ S_{i-1, j-1} \leftarrow \text{value from NW +1, if } v_i = w_j \]
\[ S_{i-1, j} \leftarrow \text{value from North (top)} \]
\[ S_{i, j-1} \leftarrow \text{value from West (left)} \]
Alignment: Backtracking

Arrows show where the score originated.

- if from the top
- if from the left
- if $v_i = w_j$
Backtracking Example

Find a match in row and column 2.

\[ i=2, \; j=2,5 \text{ is a match (T).} \]

\[ j=2, \; i=4,5,7 \text{ is a match (T).} \]

Since \( v_i = w_j \), \( s_{i,j} = s_{i-1,j-1} + 1 \)

\[ s_{2,2} = \begin{cases} 1 \\ s_{1,1} = 1 \end{cases} + 1 \]
\[ s_{2,5} = \begin{cases} 1 \\ s_{1,4} = 1 \end{cases} + 1 \]
\[ s_{4,2} = \begin{cases} 1 \\ s_{3,1} = 1 \end{cases} + 1 \]
\[ s_{5,2} = \begin{cases} 1 \\ s_{4,1} = 1 \end{cases} + 1 \]
\[ s_{7,2} = \begin{cases} 1 \\ s_{6,1} = 1 \end{cases} + 1 \]
Backtracking Example

Continuing with the dynamic programming algorithm gives this result.
Scoring Matrices

• The alignment score represent odds of obtaining that score between sequences known to be related to that obtained by chance alignment between unrelated sequences

• When the correct scoring matrix is used, alignment statistics are meaningful

• Different scoring schemes for DNA and protein sequences
Examples of Scoring Matrices

• Amino acid substitution matrices:
  – PAM
  – BLOSUM

• DNA substitution matrices:
  – DNA is less conserved than protein sequences.
  – Less effective to compare coding regions at nucleotide level.
Scoring Indels: Naive Approach

• A fixed penalty $\sigma$ is given to every indel:
  – $-\sigma$ for 1 indel,
  – $-2\sigma$ for 2 consecutive indels
  – $-3\sigma$ for 3 consecutive indels, etc.

Can be too severe penalty for a series of 100 consecutive indels
Affine Gap Penalties

- A series of $k$ indels often come as a single event rather than a series of $k$ single nucleotide events:

\[
\begin{align*}
\text{ATA}_k\text{GC} & \quad \text{ATAG}_k\text{GC} \\
\text{ATATTGC} & \quad \text{AT}_k\text{GTGC}
\end{align*}
\]

This is more likely.

Normal scoring would give the same score for both alignments

This is less likely.
Accounting for Gaps

Score for a gap of length $x$ is $-(\rho + \sigma x)$, where $\rho > 0$ is the penalty for introducing a gap:

- gap opening penalty

$\rho$ will be large relative to $\sigma$:

- gap extension penalty

because you do not want to add too much of a penalty for extending the gap.
Affine Gap Penalties

• Gap penalties:
  - $-\rho-\sigma$ when there is 1 indel
  - $-\rho-2\sigma$ when there are 2 indels
  - $-\rho-3\sigma$ when there are 3 indels, etc.
  - $-\rho- x\cdot\sigma$ (-gap opening - x gap extensions)

• Somehow reduced penalties (as compared to naïve scoring) are given to runs of horizontal and vertical edges
• Make reasonable assumptions about nature of sequence alignments and try out only “most likely” alignments

• Full perfect match (word, k-tuple)
• Much faster than DP algorithms but less sensitive

• Extend alignment until:
Existing Tools
Different Sequence Alignment

• Database Search:
  – **BLAST**, FASTA, HMMER

• Multiple Sequence Alignment:
  – ClustalW, FSA

• Genomic Analysis:
  – BLAT

• Short Read Sequence Alignment:
  – **BWA**, **Bowtie**, drFAST, GSNAP, SHRiMP, SOAP, MAQ
BLAST

• Basic Local Alignment Search Tool
• BLAST is faster than Smith-Waterman (DP algorithm), it cannot “guarantee the optimal alignments of the query and database sequences”.
• A BLAST search enables a researcher to compare a query sequence with a library or database of sequences and identify library sequences that resemble the query sequence above a certain threshold.
• Different types of BLASTs are available according to the query sequences.
Blast Versions

- **BLASTN**: Compares a nucleotide query to a nucleotide database
- **BLASTP**: Compares a protein query to a protein database
- **BLASTX**: Compares a translated nucleotide query to protein database
- **TBLASTN**: Compares a protein query to a translated nucleotide database
- **TBLASTX**: Compares a translated nucleotide query to a translated nucleotide database
Typical Uses of BLAST

• Which bacterial species have a protein that is related in lineage to a certain protein with known amino-acid sequence?

• Where does a certain sequence of DNA originate?

• What other genes encode proteins that exhibit structures or motifs such as ones that have been determined?
Multiple Alignment

• A multiple sequence alignment is a sequence alignment of three or more biological sequences (protein, DNA, RNA)
• The query sequences are assumed to have some sort of evolutionary relationship by which they share some sort of lineage
• Many tools are available but this is a really complex computational problem
• Used in phylogenetics
Short Read Alignment SW

**Bowtie**: memory-efficient short read aligner. It aligns short DNA sequences (reads) to the human genome at a rate of over 25 million 35-bp reads per hours.

**Burrows-Wheeler Aligner (BWA)**: an aligner that implements two algorithms: bwa-short and BWA-SW. The former works for query sequences shorter than 200bp and the latter for longer sequences up to around 100kbp.
Sequence Alignment/Map Format

- Sequence Reads
- Alignment Software
- SAM File
  - Resequencing
  - RNA Seq
  - SNPs
SAM format

“A tab-delimited text format consisting of a header section, which is optional, and an alignment section”

Example Headers:

```
@HD VN:1.0 SO:coordinate
@SQ SN:1 LN:249250621 AS:NCBI37 UR:/data/local/ref/GATK/human_g1k_v37.fasta
M5:1b2b98cdeb4a9304cb5d48026a85128
@SQ SN:2 LN:243199373 AS:NCBI37 UR:/data/local/ref/GATK/human_g1k_v37.fasta
M5:a0d9851da00400dec1098a9255ac712e
@SQ SN:3 LN:198022430 AS:NCBI37 UR:/data/local/ref/GATK/human_g1k_v37.fasta
M5:fdfa811849cc2faebe9c929bb925902e5
@RG ID:UM0098:1 PL:ILLUMINA PU:HWUSI-EAS1707-615LHAAXX-L001 LB:80 DT:2010-05-05T20:00:00-0400 SM:SD37
```

Example Alignments:

```
1:497:R:-272+13M17D24M 113 1 497 37 37M 15 100338662 0
CGGGTCTGACCTGAGGAGAACTGTGCTCCGCCTTCAG
SM:i:37 AM:i:0 X0:i:1 X1:i:0
XT:A:U NM:i:0
MD:Z:37
19:20389:F:275+18M2D19M 99 1 17644 0 37M = 17919
TATGACTGCTAATAATACCTACACATGTTAGAACCAT
XM:i:0 XO:i:0 XG:i:0
MD:Z:37
19:20389:F:275+18M2D19M 99 1 17644 0 37M = 17919
TATGACTGCTAATAATACCTACACATGTTAGAACCAT
XM:i:0 XO:i:0 XG:i:0
MD:Z:37
```

Example Alignments:
1.4 The alignment section: mandatory fields

Each alignment line has 11 mandatory fields. These fields always appear in the same order and must be present, but their values can be ‘0’ or ‘*’ (depending on the field) if the corresponding information is unavailable. The following table gives an overview of the mandatory fields in the SAM format:

<table>
<thead>
<tr>
<th>Col</th>
<th>Field</th>
<th>Type</th>
<th>Regexp/Range</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>QNAME</td>
<td>String</td>
<td>![?-A~]{1,255}</td>
<td>Query template NAME</td>
</tr>
<tr>
<td>2</td>
<td>FLAG</td>
<td>Int</td>
<td>[0,2^{16}-1]</td>
<td>bitwise FLAG</td>
</tr>
<tr>
<td>3</td>
<td>RNAME</td>
<td>String</td>
<td>/*[-()+-&lt;&gt;-~]!**</td>
<td>Reference sequence NAME</td>
</tr>
<tr>
<td>4</td>
<td>POS</td>
<td>Int</td>
<td>[0,2^{20}-1]</td>
<td>1-based leftmost mapping POSition</td>
</tr>
<tr>
<td>5</td>
<td>MAPQ</td>
<td>Int</td>
<td>[0,2^{8}-1]</td>
<td>MAPping Quality</td>
</tr>
<tr>
<td>6</td>
<td>CIGAR</td>
<td>String</td>
<td>/*[0-9]+[MIDNSHPX=~]+</td>
<td>CIGAR string</td>
</tr>
<tr>
<td>7</td>
<td>RNEXT</td>
<td>String</td>
<td>/*[!-()+-&lt;&gt;-~]!**</td>
<td>Ref. name of the mate/next segment</td>
</tr>
<tr>
<td>8</td>
<td>PNEXT</td>
<td>Int</td>
<td>[0,2^{20}-1]</td>
<td>Position of the mate/next segment</td>
</tr>
<tr>
<td>9</td>
<td>TLEN</td>
<td>Int</td>
<td>[-2^{20}+1,2^{20}-1]</td>
<td>observed Template LENgth</td>
</tr>
<tr>
<td>10</td>
<td>SEQ</td>
<td>String</td>
<td>/*[A-Za-z=.]+</td>
<td>segment SEQuence</td>
</tr>
<tr>
<td>11</td>
<td>QUAL</td>
<td>String</td>
<td>![?-A~]+</td>
<td>ASCII of Phred-scaled base QUALity+33</td>
</tr>
</tbody>
</table>

1. **QNAME**: Query template NAME. Reads/segments having identical QNAME are regarded to come from the same template. A QNAME ‘*’ indicates the information is unavailable.
Bitwise Flags

FLAG: bitwise FLAG. Each bit is explained in the following table:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1</td>
<td>template having multiple segments in sequencing</td>
</tr>
<tr>
<td>0x2</td>
<td>each segment properly aligned according to the aligner</td>
</tr>
<tr>
<td>0x4</td>
<td>segment unmapped</td>
</tr>
<tr>
<td>0x8</td>
<td>next segment in the template unmapped</td>
</tr>
<tr>
<td>0x10</td>
<td>SEQ being reverse complemented</td>
</tr>
<tr>
<td>0x20</td>
<td>SEQ of the next segment in the template being reversed</td>
</tr>
<tr>
<td>0x40</td>
<td>the first segment in the template</td>
</tr>
<tr>
<td>0x80</td>
<td>the last segment in the template</td>
</tr>
<tr>
<td>0x100</td>
<td>secondary alignment</td>
</tr>
<tr>
<td>0x200</td>
<td>not passing quality controls</td>
</tr>
<tr>
<td>0x400</td>
<td>PCR or optical duplicate</td>
</tr>
</tbody>
</table>
Bitwise Representation

1 = 00000001 → paired-end read
2 = 00000010 → mapped as proper pair
4 = 00000100 → unmapped read
8 = 00001000 → read mate unmapped
16 = 00010000 → read mapped on reverse strand

Example:

The flag 11 → 1 + 2 + 8 = 00001011 (conditions 1, 2, 8)

• Flags 0, 4, and 16 are the flags most commonly used.
Mapping Quality

• Phred score, identical to the quality measure in the fastq file. quality $Q$, probability $P$:
  $$P = 10^{\frac{-Q}{10.0}}$$

• If $Q=30$, $P=1/1000$ → on average, one of out 1000 alignments will be wrong

• As good as this sounds it is not easy to compute such a quality.
Mapping Quality

• The repeat structure of the reference. Reads falling in repetitive regions usually get very low mapping quality
• The base quality of the read. Low quality means the observed read sequence is possibly wrong, and wrong sequence may lead to a wrong alignment
• The sensitivity of the alignment algorithm. The true hit is more likely to be missed by an algorithm with low sensitivity, which also causes mapping errors
• Paired end or not. Reads mapped in pairs are more likely to be correct
BWA Specific High Scores

A read alignment with a mapping quality 30 or above usually implies:

– The overall base quality of the read is good.
– The best alignment has few mismatches.
– The read has few or just one “good” hit on the reference, which means the current alignment is still the best even if one or two bases are actually mutations or sequencing errors.
BWA Specific Low Scores

Surprisingly difficult to track down the exact behavior

• Q=0 → if a read can be aligned equally well to multiple positions, BWA will randomly pick one position and give it a mapping quality zero.

• Q=25 → the edit distance equals mismatches and is greater than zero
What to do with low quality scores?

• Find repeat structures in the genome/contig.
• Determine if there is a problem with your alignment or data (i.e. all the reads mapped with low quality scores).
• Filter them out. Very common to write a perl/python script to filter out poorly aligned reads.
• Many, many, many other possibilities.
The Alignment Column

1.4 The alignment section: mandatory fields

Each alignment line has 11 mandatory fields. These fields always appear in the same order and must be present, but their values can be ‘0’ or ‘*’ (depending on the field) if the corresponding information is unavailable. The following table gives an overview of the mandatory fields in the SAM format:

<table>
<thead>
<tr>
<th>Col</th>
<th>Field</th>
<th>Type</th>
<th>Regexp/Range</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>QNAME</td>
<td>String</td>
<td>![?-A-] {1,255}</td>
<td>Query template NAME</td>
</tr>
<tr>
<td>2</td>
<td>FLAG</td>
<td>Int</td>
<td>[0,2^{16}-1]</td>
<td>bitwise FLAG</td>
</tr>
<tr>
<td>3</td>
<td>RNAME</td>
<td>String</td>
<td>*</td>
<td>![+-()]+-&lt;&gt;-<del>] ![+-</del>]*</td>
</tr>
<tr>
<td>4</td>
<td>POS</td>
<td>Int</td>
<td>[0,2^{20}-1]</td>
<td>1-based leftmost mapping POSition</td>
</tr>
<tr>
<td>5</td>
<td>MAPQ</td>
<td>Int</td>
<td>[0,2^{8}-1]</td>
<td>MAPping Quality</td>
</tr>
<tr>
<td>6</td>
<td>CIGAR</td>
<td>String</td>
<td>*</td>
<td>([0-9]+[MIDNSHXP=]) +</td>
</tr>
<tr>
<td>7</td>
<td>RNEXT</td>
<td>String</td>
<td>*</td>
<td>=</td>
</tr>
<tr>
<td>8</td>
<td>PNEXT</td>
<td>Int</td>
<td>[0,2^{20}-1]</td>
<td>Position of the mate/next segment</td>
</tr>
<tr>
<td>9</td>
<td>TLEN</td>
<td>Int</td>
<td>[-2^{20}+1,2^{20}-1]</td>
<td>observed Template LENgth</td>
</tr>
<tr>
<td>10</td>
<td>SEQ</td>
<td>String</td>
<td>![A-Za-z=.] +</td>
<td>segment SEQuence</td>
</tr>
<tr>
<td>11</td>
<td>QUAL</td>
<td>String</td>
<td>![+-~]*</td>
<td>ASCII of Phred-scaled base QUALity+33</td>
</tr>
</tbody>
</table>

1. **QNAME**: Query template NAME. Reads/segments having identical QNAME are regarded to come from the same template. A QNAME ‘*’ indicates the information is unavailable.
CIGAR String

• CIGAR string is a compact representation of how the read aligned to the reference genome at that exact position.

• More specifically, the CIGAR string is a sequence of base lengths and the associated operation.
  – match/mismatch with the reference.
  – deleted/inserted from the reference.
Example of CIGAR

RefPos: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Reference: C C A T A C T G A A C T G A C T A A C
Read: A C T A G A A T G G C T

In the SAM file you will have the following fields:

- POS: 5
- CIGAR: 3M1I3M1D5M
Final Comments

• BAM is a compressed version of the SAM file format. There are multiple programs that convert BAM files to SAM files and vice versa.
• Tablet (http://bioinf.scri.ac.uk/tablet/) is an easy to use, program that allows you to visualize an alignment.
  – You simply give it a sam file and a fasta file and it reads the sam file and shows you the alignment.