

# DNA

## SEQUENCE ALIGNMENT

# **DNA SEQUENCE ALIGNMENT**

## **Design & Analysis of Algorithms (CS311) Project**

**Lama Alghzzi | 220410092**

**Norah Alrubayan | 220410543**

**Dr. Huda Alrammah**

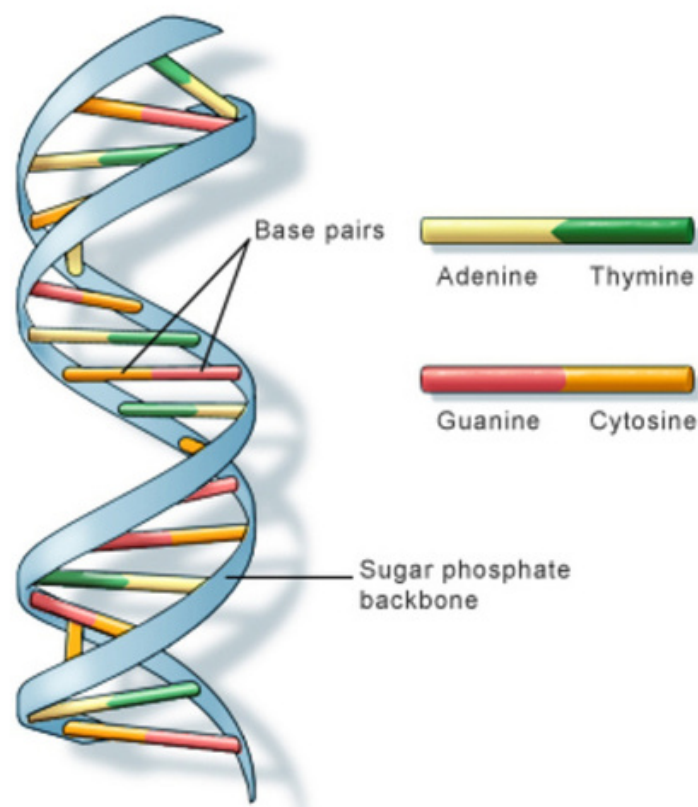


# TABLE OF CONTENTS

04	WHAT IS THE DNA?
05	DEFINITION OF THE PROBLEM
05	TYPES OF SEQUENCE ALIGNMENT
07	DYNAMIC PROGRAMMING METHOD
10	NEEDLEMAN-WUNSCH ALGORITHM STEPS & PSEUDOCODE & IMPLEMENTATION
16	SMITH-WATERMAN ALGORITHM STEPS & PSEUDOCODE & IMPLEMENTATION
19	EFFICIENCY
20	APPLICATIONS IN REAL LIFE
21	REFERENCES

## What is the DNA?

DNA, the genetic blueprint in cells, is uniform across most living organisms and resides in the cell nucleus as nuclear DNA. Comprised of four chemical bases, adenine, guanine, cytosine, and thymine, human DNA holds 3 billion bases, with over 99 percent similarity among individuals. This sequence of bases acts as a code, dictating an organism's development and functions, resembling the order of letters forming language. The pairing of these bases, A with T and C with G; forms nucleotides and ultimately, as iconic double helix structure.



## Definition of the problem

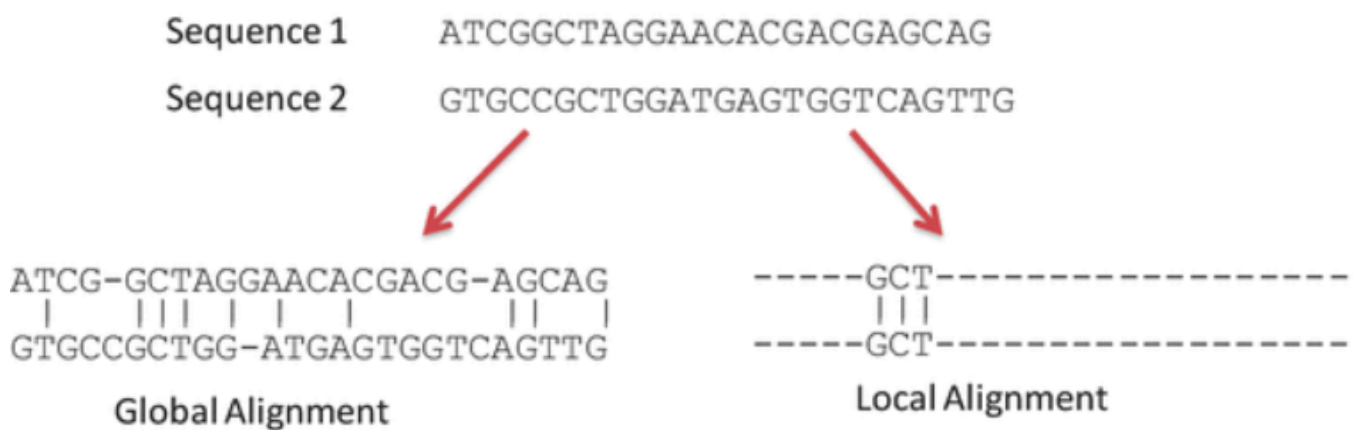
In biological sequence comparison, sequence alignment is considered to be the most crucial step. Sequence alignment is a method used to arrange two or more amino acid or nucleotide sequences in order to find similar areas. Understanding the structural, functional, and evolutionary relationships between the sequences is made easier by looking at these areas of similarity. We will use two algorithms for this problem, Needleman-Wunsch and Smith-Waterman, both algorithms use dynamic programming

## Types of Sequence Alignment

Algorithm	Needleman-Wunsch	Smith-Waterman
Type of alignment	Global alignment	Local alignment
How it works	Comparing two sequences, that aligns the whole length of the sequences by maximizing the overall similarity.	Rather than attempting to align the whole length of the sequences, only the parts with the greatest number of matches are aligned.

## Types of Sequence Alignment

Algorithm	Needleman-Wunsch	Smith-Waterman
Strengths	Optimal global alignment	Faster for long sequences, conserved domains
Weaknesses	Computationally expensive	Alignment may not cover entire length



## **Dynamic Programming Method**

Dynamic programming is utilized to find the optimal alignment between two proteins or nucleic acid sequences by comparing all feasible pairs of characters in the sequences. Both global and local alignments can be achieved through dynamic programming. While dynamic programming in local alignment is based on the Smith-Waterman algorithm, it is based on the Needleman-Wunsch algorithm in the global pairwise alignment algorithm.

### **How this algorithms works:**

- 1. Creating the score matrix:** The first step is to write the two sequences that need to be aligned along the top and left sides of a two-dimensional matrix.
- 2. Matrix filling with maximum scores:** The following step includes using a scoring matrix to fill the matrix with scores. For every match, a positive value is returned, and for every mismatch, a negative value. The algorithm aligns the corresponding residues to fill each cell in the matrix with the maximum score possible.

## **How this algorithms works:**

**3. Traceback to determine ideal alignment:** Following matrix filling, the algorithm traces back to determine the ideal alignment path.

**4. To find the optimal path with the highest total score,** adjacent cells are checked in reverse order, starting from the bottom-right corner and moving towards the top-left corner. The alignment path with the highest score is the best one.

- **Match**

A match occurs when the corresponding characters at a particular position in both sequences are the same. Matches contribute positively to the alignment score.

- **Mismatch**

A mismatch occurs when the corresponding characters at a particular position in both sequences are different. Mismatches usually result in a penalty to the alignment score.

- **Gap**

A gap occurs when there is an insertion or deletion of a character in one of the sequences to achieve the alignment. Gaps usually result in a penalty to the alignment score.



Let's start by explaining the concept of alignment and then we will take a simple example to explain the two algorithms separately.

**Match +2**

**Mismatch -1**

**Gap -2**

Seq1:	A	T	C	G	
Seq2:		T	C	A	
	-2	+2	+2	-1	

= **+1** **Alignment 1**

Seq1:	A	T	C	G	
Seq2:	T	C	A		
	-1	-1	-1	-2	

= **-5** **Alignment 2**

Seq1:	A	T	C	G	
Seq2:	T		C	A	
	-1	-2	+2	-1	

= **-2** **Alignment 3**

The first alignment is the best alignment because it gave the best result, which is +1.

# Needleman-Wunsch algorithm

## 1- Scoring Matrix

The value of each cell can come from:

- Cell beside (Gap)
- Cell Bottom (Gap)
- Diagonal cell (Match / Mismatch)

Match +2  
Mismatch -1  
Gap -2

<b>G</b>	-8	-5	-2	+1
<b>C</b>	-6	-3	0	-2
<b>T</b>	-4	-1	-2	-4
<b>A</b>	-2	-1	-3	-5
<b>----</b>	0	-2	-4	-6
	<b>----</b>	<b>T</b>	<b>C</b>	<b>G</b>

Arrows to facilitate the next step, to know the source cell for each number in each cell

## Needleman-Wunsch algorithm

### 2- TraceBack

<b>G</b>	-8	-5	-2	+1
<b>C</b>	-6	-3	0	-2
<b>T</b>	-4	-1	-2	-4
<b>A</b>	-2	-1	-3	-5
<b>----</b>	0	-2	-4	-6
	<b>----</b>	<b>T</b>	<b>C</b>	<b>G</b>

- Start at the Matrix Cell with the Highest Score

Trace Back through the Matrix: At each step during the traceback process, we move to the adjacent cell with the highest score among the three possibilities:

**Diagonal Move:** Indicates a match or mismatch between the two sequences. If the score comes from a diagonal move, we align the corresponding characters from both sequences.

**Upward Move:** Indicates a gap in the first sequence

**Leftward Move:** Indicates a gap in the second sequence

## Needleman-Wunsch algorithm

### 3- Alignment

We choose each value and its corresponding value based on the direction of the arrow.

<b>G</b>	-8	-5	-2	+1
<b>C</b>	-6	-3	0	-2
<b>T</b>	-4	-1	-2	-4
<b>A</b>	-2	-1	-3	-5
<b>----</b>	0	-2	-4	-6
	<b>----</b>	<b>T</b>	<b>C</b>	<b>G</b>

### Output:

**A T C G**

**- T C G**

## Pseudocode of Needleman-Wunsch algorithm

```
function needlemanWunsch(sequence1, sequence2, gapPenalty,
matchScore, mismatchPenalty):
    scoreMatrix = initializeMatrix(sequence1, sequence2, gapPenalty)
    for i from 1 to length(sequence1):
        for j from 1 to length(sequence2):
            matchMismatchScore = matchScore if sequence1[i - 1] equals
sequence2[j - 1] else mismatchPenalty

            diagonalScore = scoreMatrix[i - 1][j - 1] + matchMismatchScore
            upScore = scoreMatrix[i - 1][j] + gapPenalty
            leftScore = scoreMatrix[i][j - 1] + gapPenalty
            scoreMatrix[i][j] = maximum(diagonalScore, upScore, leftScore)

    alignedSequence1, alignedSequence2 = traceback(scoreMatrix,
sequence1, sequence2, gapPenalty)

    print("Aligned Sequence 1: " + alignedSequence1)
    print("Aligned Sequence 2: " + alignedSequence2)

function initializeMatrix(sequence1, sequence2, gapPenalty):
    scoreMatrix = new matrix(length(sequence1) + 1, length(sequence2) +
1)

    for i from 0 to length(sequence1):
        scoreMatrix[i][0] = i * gapPenalty
    for j from 0 to length(sequence2):
        scoreMatrix[0][j] = j * gapPenalty
    return scoreMatrix
```

## Pseudocode of Needleman-Wunsch algorithm

```
function traceback(scoreMatrix, sequence1, sequence2, gapPenalty):  
    alignedSequence1 = ""  
    alignedSequence2 = ""  
    i = length(sequence1)  
    j = length(sequence2)  
  
    while i > 0 or j > 0:  
        if i > 0 and j > 0 and sequence1[i - 1] equals sequence2[j - 1]:  
            alignedSequence1 = sequence1[i - 1] + alignedSequence1  
            alignedSequence2 = sequence2[j - 1] + alignedSequence2  
            i = i - 1  
            j = j - 1  
        else if i > 0 and (j == 0 or scoreMatrix[i][j] equals scoreMatrix[i - 1]  
[j] + gapPenalty):  
            alignedSequence1 = sequence1[i - 1] + alignedSequence1  
            alignedSequence2 = "-" + alignedSequence2  
            i = i - 1  
        else:  
            alignedSequence1 = "-" + alignedSequence1  
            alignedSequence2 = sequence2[j - 1] + alignedSequence2  
            j = j - 1  
  
    return alignedSequence1, alignedSequence2  
  
needlemanWunsch("AGTACGC", "ATAGC", -2, 2, -1)
```

## Implementation of Needleman-Wunsch algorithm

```
1 package needlemanwunsch;
2
3
4 public class NeedlemanWunsch {
5
6     public static void main(String[] args) {
7         String sequence1 = "AGTACGC";
8         String sequence2 = "ATAGC";
9
10        int gapPenalty = -2; // Penalty for gap
11        int matchScore = 2; // Score for match
12        int mismatchPenalty = -1; // Penalty for mismatch
13
14        alignSequences(sequence1, sequence2, gapPenalty, matchScore, mismatchPenalty);
15    }
16
17    public static void alignSequences(String sequence1, String sequence2, int gapPenalty, int matchScore, int mismatchPenalty) {
18        int[][] scoreMatrix = new int[sequence1.length() + 1][sequence2.length() + 1];
19
20        // Initialization
21    }
22 }
```

Find:

Output - NeedlemanWunsch (run)

```
run:
Aligned Sequence 1: AGTACGC
Aligned Sequence 2: A-TA-GC
BUILD SUCCESSFUL (total time: 0 seconds)
```

The input size for the first sequence is 7 and the second sequence is 5

[illegible]

The input size for the first sequence is 12386 and the second sequence is 12435

**There is a direct relationship between data size and running time**

# Smith-Waterman algorithm

## 1- Scoring Matrix

There is no negative score with local alignment, any negative value will be replaced with 0

Match +1

Mismatch -1

Gap -2

	---	A	T	G	C	T
---	0	0	0	0	0	0
A	0	1	0	0	0	0
G	0	0	0	1	0	0
C	0	0	0	0	2	0
T	0	0	0	0	0	3



## Smith-Waterman algorithm

### 2- TraceBack

	---	A	T	G	C	T
---	0	0	0	0	0	0
A	0	1	0	0	0	0
G	0	0	0	1	0	0
C	0	0	0	0	2	0
T	0	0	0	0	0	3


### 3- Alignment

**Output:**

**A T G C T**

**- A G C T**

## Implementation of Smith-Waterman algorithm



The screenshot shows an IDE with two panels. The top panel displays the source code for the SmithWaterman algorithm. The code is in Java and includes a package declaration, a class definition, a main method, and an alignSequences method. The bottom panel shows the output of the program, which includes the aligned sequences and a success message.

```
1 package smithwaterman;
2 public class SmithWaterman {
3
4     public static void main(String[] args) {
5         String sequence1 = "AGTACGC";
6         String sequence2 = "ATAGCG";
7
8         alignSequences(sequence1, sequence2);
9     }
10
11     public static void alignSequences(String sequence1, String sequence2) {
12         int[][] scoreMatrix = new int[sequence1.length() + 1][sequence2.length() + 1];
13         int maxScore = 0;
14         int maxI = 0, maxJ = 0;
15     }
16
17 }
```

Find:  Previous Next Select

Output - SmithWaterman (run)

```
run:
Aligned Sequence 1: TACGC
Aligned Sequence 2: TA-GC
BUILD SUCCESSFUL (total time: 0 seconds)
```

The input size for the first sequence is 7 and the second sequence is 5

[illegible]

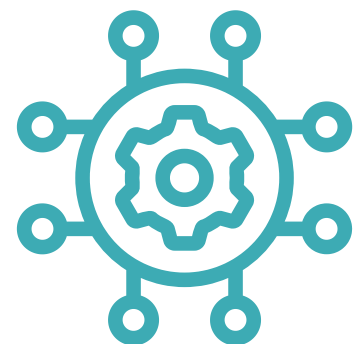
The input size for the first sequence is 12386 and the second sequence is 12452

**There is a direct relationship between data size and running time**

## Efficiency

The efficiency of the Smith-Waterman and Needleman-Wunsch algorithms can depend on several factors:

- **Length of Sequences:** Both algorithms have a time complexity  $O(m \cdot n)$ , where  $m$  and  $n$  are the lengths of the sequences that aligned.
- **Local or Global Alignment:** Needleman-Wunsch is used for global sequence alignment, aligning the entire sequences, while Smith-Waterman focuses on local alignments, identifying regions of similarity. In cases where the sequences have large regions of dissimilarity, Smith-Waterman might be more efficient and relevant.



## **Applications in real life**

1- DNA sequence alignment aids in identifying genes within a genome. This is crucial for understanding the genetic basis of various traits and diseases.

2- Comparing DNA sequences helps identify variations associated with diseases. This is vital for studying genetic predispositions and developing targeted treatments.

3- In pharmaceutical research, DNA sequence alignment aids in identifying potential drug targets. Understanding the genetic basis of diseases helps in developing drugs that target specific genetic factors.

## References