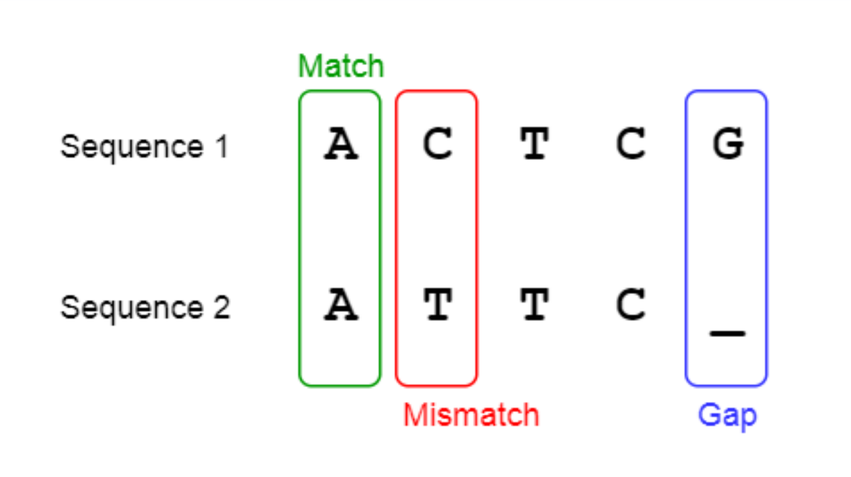
Alineación de Distancia para Secuencias de ADN y RNA

Nicole M. Ramirez Mulero
CCOM 6885
Prof. Humberto Ortiz
Mayo 5 de 2025

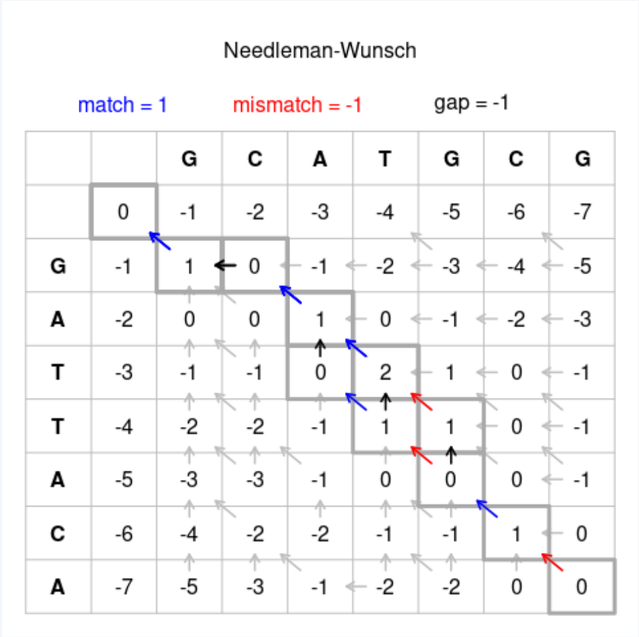
Introducción



- Comparar secuencias es importante para muchas aplicaciones en la vida real como identificar anormalidades en secuencias, e identificar zonas biológicamente significativas.
- Diversas herramientas se han creado para cumplir el objetivo de alineamiento (distancia por edición, alineamiento solapado, alineamiento de ajuste, etc.).
- El alineamiento de edición implica encontrar el mínimo número de ediciones con tal de que 2 secuencias estén lo más alineadas posible.
- Para alinear, tomamos en cuenta los caracteres iguales, los desiguales y los espacios.

El Algoritmo: Needleman-Wunsch

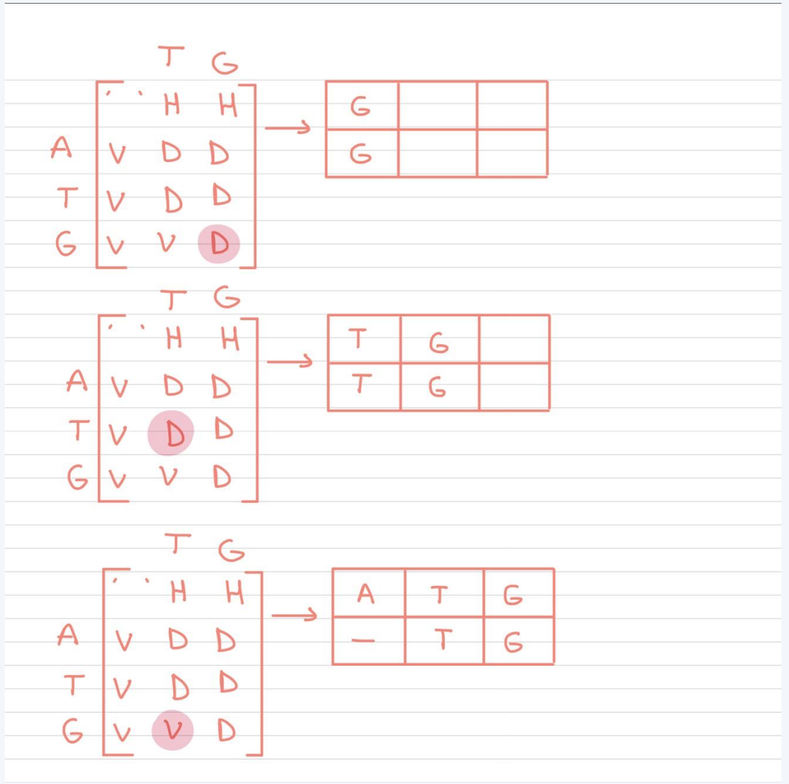
- Algoritmo usado en bioinformática para alinear secuencias de proteínas o nucleótido.
- Una de las primeras aplicaciones de programación dinámica para comparar secuencias biológicas.
- Este algoritmo es una manera inteligente para procesar todas las posibilidades de secuencias que se deben considerar. Reduce significativamente el tiempo requerido y garantiza una solución óptima.
- La idea es construir la alineación más optima utilizando la optimización optima de secuencias más pequeñas.



		G	C	A	T	G	C	G
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

$$D(i,j) = \max \begin{cases} D(i-1,j-1) + s(x_i, y_j) \\ D(i-1,j) + g \\ D(i,j-1) + g \end{cases}$$

Ejemplo de Alineación



El Código

```python
def Edit_Distance_Aligment(seq1, seq2):

    # Puntuaciones
    match = 1
    mismatch = -1
    gap = -2


    #Iniciar Matrices
    score = np.zeros((len(seq1) + 1, len(seq2) + 1))
    backtrack = np.zeros((len(seq1) + 1, len(seq2) + 1), dtype="str")

    # Inicializar puntajes
    score[0, :] = np.arange(len(seq2) + 1) * gap
    score[:, 0] = np.arange(len(seq1) + 1) * gap

    # Initialar backtrack
    backtrack[0, 1:] = "H"
    backtrack[1:, 0] = "V"
```

```python
    for i in range(1, len(seq1) + 1):
        for j in range(1, len(seq2) + 1):
            # Puntaje diagonal
            if seq1[i - 1] == seq2[j - 1]:
                scoreD = score[i - 1, j - 1] + match
            else:
                scoreD = score[i - 1, j - 1] + mismatch

            # puntaje horizotal
            scoreH = score[i, j - 1] + gap

            # puntaje vertical
            scoreV = score[i - 1, j] + gap

            # elejir mejor puntaje
            best_score = np.max([scoreD, scoreH, scoreV])
            if scoreD == best_score:
                score[i, j] = scoreD
                backtrack[i, j] = "D"
            elif scoreH == best_score:
                score[i, j] = scoreH
                backtrack[i, j] = "H"
            elif scoreV == best_score:
                score[i, j] = scoreV
                backtrack[i, j] = "V"
```

El Código

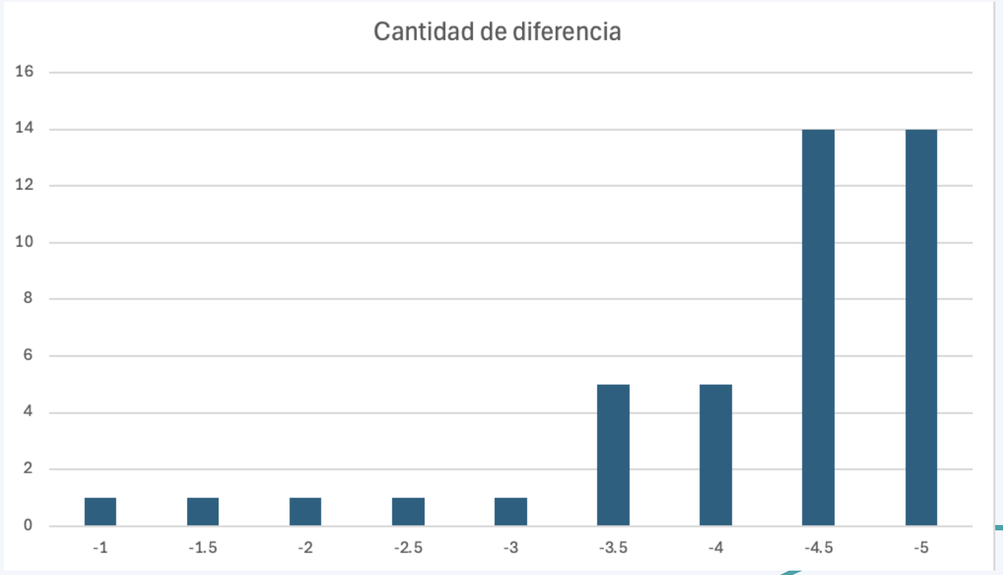
```python
# Variables para sequencias alineadas
aligned_seq1 = ""
aligned_seq2 = ""


i = len(seq1)
j = len(seq2)


while i != 0 or j != 0:
    # Si es diagonal, simplemente escribe el caracter
    if backtrack[i, j] == "D":
        aligned_seq1 = seq1[i - 1] + aligned_seq1
        aligned_seq2 = seq2[j - 1] + aligned_seq2
        i -= 1
        j -= 1

    # Si es horizontal, alinea sequencia 1
    elif backtrack[i, j] == "H":
        aligned_seq1 = "-" + aligned_seq1
        aligned_seq2 = seq2[j - 1] + aligned_seq2
        j -= 1

    # Si es vertical, alinea sequencia 2
    elif backtrack[i, j] == "V":
        aligned_seq1 = seq1[i - 1] + aligned_seq1
        aligned_seq2 = "-" + aligned_seq2
        i -= 1


print(aligned_seq1)
print(aligned_seq2)
```

Pruebas y Resultados

```
test1_1 = "ATG"
test1_2 = "TG"

test2_1 = "ACGATACG"
test2_2 = "TACGTCG"

test3_1 = "CTGGTAGTTG"
test3_2 = "CTAGTG"

test4_1 = "GAGCCGCATAAATATTGTCA"
test4_2 = "GAGGCATAATATTGTCATC"

test5_1 = "GGTCATATTAGTTCGGACCGGATGAGGCGGCGGTTACGGGTGACGAGGGGCCCTTGGAACCTAGAGAAACACAGTGTGGAAAAGTCGCGTTGTAACACAA"
test5_2 = "GGTCATATTTGTTCGGACCGGATGAGGCGGCGGTTACGGGTGACGAGGGGCCCTTGAACCTAGAGAAACACAGTGTGGAAAAGTCGCGTTGTAACACAAAAA"
```

```
ATG
-TG

-ACGATACG
TACG-T-CG

CTGGTAGTTG
C---TAG-TG

GAGCCGCATAAATATTGTCA--
GAG--GCAT-AATATTGTCATC

GGTCATATTAGTTCGGACCGGATGAGGCGGCGGTTACGGGTGACGAGGGGCCCTTGGAACCTAGAGAAACACAGTGTGGAAAAGTCGCGTTGTAACAC---AA
GGTCATATTTGTTCGGACCGGATGAGGCGGCGGTTACGGGTGACGAGGGGCCCTT-GAACCTAGAGAAACACAGTGTGGAAAAGTCGCGTTGTAACACAAAAA
```

```python
seq1 = ["ATG","CTGGTAGTTG","ACGATACG","GAGCCGCATAAATATTGTCA","GGTCATATTAGTTCGGACCGGATGAGGCGGCGGTTA(
seq2 = ["TG","CTAGTG","TACGTCG","GAGGCATAATATTGTCATC","GGTCATATTTGTTCGGACCGGATGAGGCGGCGGTTACGGGTGA(

diferencias = [0,0,0,0,0,0,0,0,0]

for i in range(len(seq1)):
    j = -1
    x = 0
    while(j > -5.5):
        print(j)
        aligned_seq1, aligned_seq2 = Edit_Distance_Aligment(seq1[i], seq2[i], 1, -1, j)
        print(aligned_seq1)
        print(aligned_seq2)

        diferencias[x] += CheckDiference(aligned_seq1, aligned_seq2)
        j += -0.5
        x +=1

        print(diferencias)
        print("----------")

    print(" ")
    print("~~~~~~~~~~~~~~~~~~~~~~~~")

print(diferencias)
```



Conclusión

- El alineamiento de secuencias es una herramienta fundamental en la bioinformática, ya que permite identificar similitudes, diferencias, mutaciones e inserciones entre cadenas genéticas como el ADN y ARN.
- Los algoritmos de programación dinámica son esenciales para problemas como este para poder realizar tareas complejas de manera efectiva bajo un tiempo considerable sin requerir demasiados recursos.
- Las penalizaciones y recompensas dentro del algoritmo (por ejemplo, el valor asignado al *gap*) tienen un impacto significativo en los resultados. Como se demostró en el experimento, penalizaciones demasiado altas pueden producir alineamientos incorrectos, lo que resalta la importancia de ajustar estos valores según el tipo de secuencia y el contexto del análisis.

Bibliografía

[1] Compeau, P., & Pevzner, P. (2018). Bioinformatics algorithms: an active learning approach. La Jolla, California: Active Learning Publishers. ISBN-13978-0990374633 https://www.bioinformaticsalgorithms.org/

[2] Likic, V. (2008). The Needleman-Wunsch algorithm for sequence alignment. *Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne*, 1-46.

[3] Rashed, A. E. E. D., Amer, H. M., El-Seddek, M., & Moustafa, H. E. D. (2021). Sequence alignment using machine learning-based needleman–wunsch algorithm. *IEEE Access*, *9*, 109522-109535.

[4] Kenneth H. Rosen. 2010. Handbook of Discrete and Combinatorial Mathematics, Second Edition (2nd. ed.). Chapman & Hall/CRC.