**Estructura de datos y algoritmos**

**Taller 3**

* **Santiago Torres**
* **Juan David Orozco**

**Ejercicio 2.** *Distancia de Hamming.*

**2.** Dé la complejidad en tiempo y la complejidad espacial del algoritmo.

En tiempo O(n), y en complejidad O(n).

**3.** ¿Cuál es el principal inconveniente del uso de esta distancia en el contexto de secuencias ADN?

No todas las secuencias tienen la misma distancia y por ello el algoritmo tiende a fallar. Además no tiene en cuenta información como mutaciones o estructura.

Para almacenar los *k*-mers de forma eficiente, seleccione una estructura de datos adecuada.

1. Escriba un algoritmo para descomponer en *k*-mers las secuencias de un fasta y guardarlas.
2. ¿Cuántos *k*-mers se generan por una secuencia de longitud *|S|* ?
3. ¿ Cual es la complejidad en tiempo del algoritmo de descomposición en *k*-mers de una secuencia *S* dada ?  
     
   **Solucion:**  
     
   1. Este es un Algoritmo para descomponer en k-mers las secuencias de un archivo fasta y guardarlas:  
     
   def decompose\_to\_kmers(sequence, k):

    kmers = []

    for i in range(len(sequence) - k + 1):

        kmers.append(sequence[i:i+k])

    return kmers

def fasta\_to\_kmers\_dict(fasta\_sequences, k):

    kmers\_dict = {}

    for header, sequence in fasta\_sequences.items():

        kmers\_dict[header] = decompose\_to\_kmers(sequence, k)

    return kmers\_dict

2. se puede calcular utilizando la fórmula: |S| - k + 1. Esto se debe a que al descomponer una secuencia en k-mers, se generan subsecuencias de longitud k, y el número total de k-mers será igual a la longitud de la secuencia menos la longitud de los k-mers más uno.  
  
3. depende del tamaño de la secuencia y del valor de k. En general, la complejidad es O(|S| - k + 1), ya que se debe recorrer la secuencia una vez para generar los k-mers. Sin embargo, es importante considerar que el acceso y almacenamiento de los k-mers en la estructura de datos seleccionada también puede influir en la complejidad del algoritmo.

▶ **Ejercicio 4.** *Similitud genética y índice de Jaccard.*

La similitud entre dos conjuntos *A* y *B* se calcula mediante el índice de Jaccard *J*. La fórmula se da en la ecuación (2). En el contexto de las secuencias de ADN, para determinar la similitud genética utilizamos los *k*-mers de las especies.

*|A ∩ B|*

*J*(*A,B*) = (2)

*|A ∪ B|*

1. Escriba un método para calcular el índice de Jaccard de dos listas A y B.
2. Escriba el método que, dados dos conjuntos de *k*-mers, calcula el índice de Jaccard asociado.
3. Escriba un método que, dado un conjunto de fasta, retorna la similitud entre las secuencias como una matriz.

Dé los resultados sobre el conjunto de datos dado en forma de matriz de similitud para *K* = 7 y *K* = 11.

Indique el tiempo de ejecución para cada *K*.  
  
**Solucion:**  
  
1, Método para calcular el índice de Jaccard de dos listas A y B:

def jaccard\_index(list\_a, list\_b):

    set\_a = set(list\_a)

    set\_b = set(list\_b)

    intersection = len(set\_a.intersection(set\_b))

    union = len(set\_a.union(set\_b))

    return intersection / union if union != 0 else 0

2. Método para calcular el índice de Jaccard dado dos conjuntos de k-mers:  
  
def jaccard\_index\_kmers(kmers\_a, kmers\_b):

    return jaccard\_index(kmers\_a, kmers\_b)  
  
3. Método para calcular la matriz de similitud entre secuencias fasta para un valor dado de k:  
  
def similarity\_matrix(fasta\_sequences, k):

    num\_sequences = len(fasta\_sequences)

    similarity\_mat = [[0] \* num\_sequences for \_ in range(num\_sequences)]

    for i in range(num\_sequences):

        for j in range(i, num\_sequences):

            kmers\_i = fasta\_to\_kmers(fasta\_sequences[i], k)

            kmers\_j = fasta\_to\_kmers(fasta\_sequences[j], k)

            similarity\_mat[i][j] = jaccard\_index\_kmers(kmers\_i, kmers\_j)

            similarity\_mat[j][i] = similarity\_mat[i][j]

    return similarity\_mat  
  
Este método toma una lista de secuencias en formato fasta y un valor k como entrada, y devuelve una matriz de similitud entre todas las combinaciones de estas secuencias. Itera sobre todas las posibles combinaciones de pares de secuencias y calcula el índice de Jaccard entre los conjuntos de k-mers correspondientes.