**Time Analysis :**

The time complexity of the algorithm is determined by the nested loops which used to fill the scoring matrix , assuming the n to be the length of the sequence X and m to be the length of sequence Y we can divide the time into 3 main stages :

1 - Initialization of the **charToIndexMap** is **O(1)** .

2 - Constructing of the **derivedScoreMatrix** consists of two nested loops ,but each iteration of the inner loop contain constant time operation hence the time complexity of this part is **O(n\*m)** .

3- Reconstructing part (**Traceback** ) consists also of a loop , and in each iteration of this loop , there is constant time operation . No. of iteration of the worst case depends on the length of both sequence length **O(n+m)** so the Reconstructing part time complexity is **O(n+m)**

Therefore the time complexity of the **sequenceAlignment()** method is controlled by the initialization of the **derivedScoreMatrix** which is **O(n\*m)** .

**Approach :** using the principle of the **Dynamic Programming Approach** .

**Initialization** : 1- **charToIndexMap** is created to map the nucleotides to their corresponding indices .

2- Score Matrix which called **derivedScoreMatrix** is initialized with the dimensions (n+1)(m+1) where n is length of the sequence X and m is the length of the sequence Y.

**derivedScoreMatrix Filling up :** as the nested loops iterate over each cell of the scoring matrix , we calculate 3 different possibilities for the scores 1- matching characters from both sequences called **matchedScore** 2-Inserting dashed (gap) in sequence X which called **dashXScore** 3- inserting dashed (gap) in sequence Y which called **dashYScore** .

**TraceBack (Reconstructing ) :** the process start from the bottom right corner of the matrix and follow up to the path with highest scores ,if a match is found the corresponding characters are appended to both sequences but if a gap in sequence X is found , then a gap is appended to **row\_X** and if a gap in sequence Y is found , then a gap is appended to **row\_Y**

**Output :** then for the result , i created a **result** array of size 2 contain both **row\_X** and **row\_Y** that is returned And to test the correctness , i have added a method which called **calculateAlignmentScore()** that take the result array outputs both **row\_X** and **row\_Y** and the scoring matrix (which is already given in the main method before and the assignment ) that calculates the alignment score between sequence X and sequence Y , based on a given scoring matrix .

**The output given in the assignment is** String row\_X = “ - -T- TATGT-CAGGGGACACG-A-GCAT GCAGA-GAC” , String row\_Y= “AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C"; which has scored -5.89999999 and my output that my code has generated is row\_X

= “- - -TCCCAGTTATGTCAGGGGACACG -AG-CATG-CAGAGAC" , And

row\_Y =”AATTGCC-G-C-CGTC-GTTTTCA-GCAGTTATGTCAGAT-C", Which scored 5 (which score an alignment higher than the alignment of the solution test case given in the assignment ).