MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE 4065 – Computational Genomics

Programming Assignment # 2

Group Members

150116012 Rahim Gün

150116056 Emre YİĞİT

150116016 Berk Köylü

1 - Implementation Details

In this assignment, we implemented pairwise sequence alignment by using Needleman–Wunsch algorithm.

In out project, we created SequenceAlignment class and inside the class we have alignSequences method that makes the sequence alignment. Method takes the sequences, match score, mismatch score and indel score as arguments. Inside the method, firstly, we created two-dimensional matrix with length of the strings. And then, we filled the first row and first column with the indel score by summing cumulatively. And then, we filled the other cells of the matrix by selecting the highest cumulative score that comes from match/mismatch, insert or delete one by one. Score of each cell is calculated depending on the match, mismatch and indel score given by the user. Finally, the lower right cell of the alignment matrix gives the optimum alignment between the given sequences. Code snipped below shows the implementation of the algorithm inside the project.

```
public int[][] alignSequences(String sequenceOne, String sequenceTwo, int matchScore, int mismatchScore, int indelScore){

    int[][] alignmentMatrix = new int[sequenceOne.length() + 1][sequenceTwo.length() + 1];
    alignmentMatrix[0][0] = 0;
    int matchOrMismatchScore = 0;

    for (int i = 1; i < sequenceOne.length() + 1; i++) {
        alignmentMatrix[i][0] = alignmentMatrix[i - 1][0] + indelScore;
    }
    for (int i = 1; i < sequenceTwo.length() + 1; i++) {
        alignmentMatrix[0][i] = alignmentMatrix[0][i - 1] + indelScore;
    }

    for (int i = 1; i < sequenceOne.length() + 1; i++) {
        for (int j = 1; j < sequenceTwo.length() + 1; j++) {

            if(sequenceOne.charAt(i - 1) == sequenceTwo.charAt(j -1)){
                matchOrMismatchScore = matchScore;
            }else {
                matchOrMismatchScore = mismatchScore;
            }

            alignmentMatrix[i][j] = Math.max(Math.max(alignmentMatrix[i][j - 1] + indelScore , alignmentMatrix[i- 1][j] + indelScore), alignmentMatrix[i - 1][j - 1] + matchOrMismatchScore);
        }
    }

    return alignmentMatrix;
}
```

Figure 1 - Implementation of Needleman Wunsch Algorithm

After calculation of the sequence matrix, we need to backtrack from sink to source to find the optimum alignment sequence. Inside the SequenceAlignment class, we created backtrackMatrix method that traverse the matrix backward from sink to source by comparing the cell value with three possible sources match/mismatch, insert, delete to see where it comes from. If it is match/mismatch, then nucleotides are aligned. If it is insert, first sequence is aligned with gap. If it is delete, second sequence is aligned with gap. Code below shows the implementation of the backtracking algorithm and its arguments. And then, method prints the aligned sequences on the terminal.

```java
public void backtrackMatrix(String sequenceOne, String sequenceTwo, int[][] sequenceMatrix, int indelScore){
    StringBuilder alignmentOne = new StringBuilder();
    StringBuilder alignmentTwo = new StringBuilder();
    int i = sequenceOne.length();
    int j = sequenceTwo.length();

    while (i > 0 || j > 0) {
        if((i > 0) && (j > 0) && (sequenceMatrix[i][j] == sequenceMatrix[i-1][j -1] + matchOrMismatchCheck(sequenceOne.charAt(i - 1),sequenceTwo.charAt(j-1)))){
            alignmentOne.insert( offset: 0, sequenceOne.charAt(i - 1));
            alignmentTwo.insert( offset: 0, sequenceTwo.charAt(j - 1));
            i = i - 1;
            j = j - 1;
        }else if(i > 0 && sequenceMatrix[i][j] == sequenceMatrix[i -1][j] + indelScore ){
            alignmentOne.insert( offset: 0, sequenceOne.charAt(i - 1));
            alignmentTwo.insert( offset: 0,  str: "_");
            i = i - 1;
        }else{
            alignmentOne.insert( offset: 0,  str: "_");
            alignmentTwo.insert( offset: 0, sequenceTwo.charAt(j - 1));
            j = j - 1;
        }
    }

    System.out.println(alignmentOne);
    System.out.println(alignmentTwo);
}
```

Figure 2 – Implementation of the Backtracking Algorithm

2 – Program Outputs

Sequence-alignment-result-output.txt file also shared with submitted assignment folder.

Output of test1.seq

```
CGG__GTAGTTAACCCT_ACA_GCATAGAGTCGCGAGATAAAGTGCAGGA_GTCTTTCGCGGCAGATTCGTACCTCAACCACGTGCTACTT_TCTGGCA_TCACGAAT_CTGCCGCATAGGTCCGTGAGT_CCATATGA
AGGAAGTAGTT_AGCCTAACAGGCATAGAGTCGCGACAT_ATGTG_AAGATGTCATT__CGG__TATTCAAACCTCATGCA__T_C_A_TTGCCTTG_AGT__CG_CTCCTGGAGCATA_GTCCCTGAGTGCCATATGA
Score of the Alignment:
149
```

Output of test2.seq

```
_C_GAGACCGA_C_GAAGAGGTTTGGC_CCCAA_CCAGGTTCC_CTGATCACGTAACTTACCGGCCAAAAGGACTG__GCCTT_ACTAAGGC__CTT__TGTCTA_CTGC_G_G__GT_C__CGG_G_GGC_CGTT__GGT_T_TC_GGCAGAACACTTC
CCTG_GACCGAGCTTAA_A___TT_GCTAGCAATACAGATGCCGCT__TC_C_TTGGGGA_GGGTGTGTAGGA_TGTAG_GTTAACGAATGCAAGTTCCGGGGTATC_GCAGAGTCGTGCTACGGCGTGGCAC_TTAGGGTCTCTCGGGAAAAAGAGTAG
Score of the Alignment:
95
```

Output of test3.seq

```
GTGT_GGTTGCTTGCATCACTCCGTGTACATGTGACAACCGAACGAGTTGATCCAGCTTTGTTAAGTCAGCTTCGAATG_CGGTAGCTCTCAA_ATATGAT_ATG_AC_TCTTGGGGTAGAT__GCTGG__GGACCTATTGCGC_C_CAAAGCGATAT__TCGGGGCA_CCGGTTTA
G_GTAGGTT_CGTGAAGCACTCC_TGGAC_TCTGACCA_C_AAC_A__TAATCAAGC___G__CAG__AG_TT_GAATGACCAAAGCGCTCAAGCT_T__TCACGAACGTC_T____CATATCCGC_GGTCGTA_CAAACGCGCTCTTAAAAC_A_ATCGTCTATCCATCCGG____

GGGTACCCTATCAAGGCGAT_AC_TTCGA_TGCAGTGTGATGCCGG_A_G_GTGTCG_GTCAGCATGTGAGA__GCT
GGATA_CC_A__ATGG_G_TGACATT_AACTG__G_G_CAGGCCGGCACGCG_GACGCGACAG_AT_TGAAATGGAT


Score of the Alignment:
184
```

Output of test4.seq

Sequences do not fit the screen shot but we provide the outputs of the program as sequence-alignment-result-output.txt file inside the submitted folder.

```
CG_GGGAAAGA_CGG__A_ATGCATCG_ACCATCGGACAAT_GC_CTCACTGGAAGCGCTG__CTGATTTTTGCGCA_ACGAGCCTCGGA_CCTCCCG_C_TCAAA_CT_TACGAA__A_ATGACTCCACC_____AG_CACTGAA_CCAACTGGCCTCCAG_TGA_AGAT___AGTAT_CT__
AGTGAG_AAGACCGGATATATGCAACGAACAATCGCAAAATAGCTC_CA_TGTACGC_CTGTCCCG_TAATACCGCATGCGAGCCTCGAAGCC_CCCGACGTCAAACCTCAAC_AAGTATAGGGTTCCACCGTATGAGCCACAGAATTC_AGT_TCCTCC_GCTGATAG_TCCCGGT_TCCTCC
Score of the Alignment:
605
```

Output of test5.seq

```
AGGC__CGAAAACGTCGCGAAT_T_GACCCTGGCGACGCCGCCGAACGGGACCTCCGTTAG_TGT_GGGAGGTCATCAATCTCGTTCGCTAGCGGCTGAC_ACCAATCACTATAAG_TCTGTCATGAC
___CTTC___AA_GT__CAAATATAGATCCTGGC__CG_CTCC__ACGGG_CTTAAG_TCGTTCTCCGAAGGT_A_CGATCTGGTTGGAT_GCTTCCGTCTA__AA_CA__AGAAGAT_AATC__G__
Score of the Alignment:
82
```