1. Daten und Algorithmen

1.1. Sequenzalignments

Ein Sequenzalignment oder kurz Alignment (Alignierung) ist ein Vergleich von zwei oder mehreren Sequenzen. In der Bioinformatik sind dies oft Nukleotidsequenzen von verschiedenen Organismen, die auf funktionelle bzw. evolutionelle Verwandtschaft untersucht werden. In diesem Projektstudium wird auf paarweise Alignments fokussiert. Vor allem werden hier keine heuristischen Algorithmen wie dem Greedy-Algorithmus, BLAST oder FASTA implementiert, sondern exakte Algorithmen wie Smith-Waterman- oder Needleman-Wunsch-Algorithmus, sowie der Algorithmus basierend auf das Hidden Markov Modell. Folglich werden die drei Algorithmen vorgestellt.

1.1.1. Globales Aligment

Das globale Alignment beruht auf den Needleman-Wunsch-Algorithmus. Dieser berechnet dynamisch den optimalen globalen Score zwischen zwei Sequenzen. Dabei bedeutet optimal gleich maximal. Die Scores werden von Match (Kongruenz beider Sequenzen in Position i und j), Mismatch bzw. Substitution (Inkongruenz beider Sequenzen in Position i und j), Insertion (Sequenz hat ein Element eingefügt) oder Deletion (Sequenz hat ein Element gelöscht) beeinflusst. Dabei kann von allgemeinem fit- und *penalty*-Wert ausgegangen werden oder eine Matrix erstellt werden, welches je nach Fall der Mutation und Zeichen einen individuellen Wert besitzt. Ein wichtiger Spezialfall dabei ist eine kurze Sequenz in einer längeren Sequenz zu finden. Anders gesagt wird der Abschnitt mit größtmöglicher Ähnlichkeit zu der kurzen Sequenz gesucht. Das wird vor allem in der Genanalyse praktiziert.

Bei dem Algorithmus werden drei Matrizen erstellt - die (Mis-)Match-Matrix, die Score-Matrix und die Traceback-Matrix. In der (Mis-)Match-Matrix werden positive Werte bei Match und negative bei Mismatch eingeschrieben. In der Score-Matrix werden die Scores eingeschrieben, die durch Match, Insertion oder Deletion bestimmt

werden. Der größte Score vom Vorgänger - entweder von schräg hinten (Match oder Substitution), von links (Insertion) oder von oben (Deletion) - wird übernommen und der Wert in der (Mis-)Match-Matrix wird hinzuaddiert (siehe Gleichung 1 und 2). Der maximale Score der letzten Zeile wird bestimmt. Um den Pfad zu ermitteln, wie dieser Score enstanden ist, wird in jedem Scoreberechnung in der Traceback-Matrix gespeichert aus welcher Richtung dieser Score gerade kam - 0 bei schräg hinten (mit Abfrage ob Match oder Substitution), -1 bei Deletion, +1 bei Insertion. Somit kann nach der Ermittlung des globalen optimalen Alignments auch die Sequenz an der richtigen Stelle dargestellt werden.

Es werden zwei Sequenzen S_1 und S_2 mit der Länge n bzw m eingegeben. Dabei sei $S_1[i]$ das i-te Zeichen und $S_1[1..i]$ das Präfix der länge i - analog mit $S_2[j]$ und j. Dabei fangen die Sequenzen erst bei Position 1 an. Position 0 bleibt leer.

$$Score_{(0,0)} = 0 \tag{1}$$

$$Score_{(i,j)} = max \begin{cases} Score_{(i,j-1)} + insertion(S_2[j]) & j \geq 1 \\ Score_{(i-1,j)} + deletion(S_1[i]) & i \geq 1 \\ Score_{(i-1,j-1)} + substitution(S_1[i], S_2[j]) & i,j \geq 1 \end{cases} \tag{2}$$

Als Beispiel dienen die folgenden Sequenzen, dessen Alignment in den Abbildungen 1 und 2 berechnet wird:

$$S_1 = \{-, W, U, R, Z, E, L\} \text{ und}$$
$$S_2 = \{-, V, I, E, R, T, E, L\}.$$

Zu Beginn werden die Match- bzw. Mismatch-Werte eingegeben (Vgl. Abbildung 1). Danach wird eine zweite Tabelle erstellt - die Score-Matrix-Berechnung. Wie in der Abbildung 2 ersichtlich ist, werden bei horizontaler und vertikaler Bewegungsrichtung negative Werte und bei diagonaler Richtung positive Werte zu den bereits vorhandenen Werten aus der (Mis-)Match-Matrix addiert. Die Berechnung beginnt oben links und verläuft zeilenweise bis zur rechten untersten Position.

M	–	W	U	R	Z	E	L
–	+10	-10	-10	-10	-10	-10	-10
V	-10	-10	-10	-10	-10	-10	-10
I	-10	-10	-10	-10	-10	-10	-10
E	-10	-10	-10	-10	-10	+10	-10
R	-10	-10	-10	+10	-10	-10	-10
T	-10	-10	-10	-10	-10	-10	-10
E	-10	-10	-10	-10	-10	+10	-10
L	-10	-10	-10	-10	-10	-10	+10

Match: +10
Mismatch: -10
↘ : +10
→↓ : -10

Abbildung 1: (Mis-)Match-Matrix-Berechnung: In dieser Matrix werden bei Match +10 gespeichert, bei Mismatch -10. Die Pfeile daneben zeigen die Legende für Abbildung 2. Abbildung anlehnend an [**?**].

S	–	W	U	R	Z	E	L
–	+10	0	-10	-20	-30	-40	-50
V	0						
I	-10						
E	-20						
R	-30						
T	-40						
E	-50						
L	-60						

S/T	–	W	U	R	Z	E	L
–	+10	0	-10	-20	-30	-40	-50
V	0	0	-10	-20	-30	-40	-50
I	-10	-10	-10	-20	-30	-40	-50
E	-20	-20	-20	-20	-30	-30	-40
R	-30	-30	-30	-20	-30	-40	-40
T	-40	-40	-40	-30	-30	-40	-50
E	-50	-50	-50	-40	-40	-30	-40
L	-60	-60	-60	-50	-50	-40	-30

Abbildung 2: Score-Matrix-Berechnung: Zuerst werden die erste Zeile und die erste Spalte mit Werten befüllt. Dann werden die Scores zeilenweise von links nach rechts berechnet. Dabei werden sie entweder von den Vorgängern links, oben oder schräg links und der Score von der (Mis-)Match-Matrix berechnet. Wenn der maximale Score von schräg hinten kommt, werden +10 angerechnet, sonst -10 (siehe Legende in Abbildung 1. Die Pfade mit den roten Pfeilen zeigen den optimalen Pfad mit dem höchsten Score an. Abbildung anlehnend an [**?**].

1.1.2. Lokales Alignment

Häufig ist es nicht nur wichtig die Ähnlichkeit zwischen zwei Sequenzen zu bestimmen, sondern teilweise Übereinstimmungen zu finden. D.h. relativ kurze Teilabschnitte mit hoher Übereinstimmung sind von besonderem Interesse. Das Ziel des Smith-Waterman-Algorithmus ist es, die Teilsequenzen mit der größten Übereinstimmung zwischen zwei gegebenen Sequenzen zu finden. Ein solches Paar aus Teilsequenzen wird lokales Alignment genannt und ist somit eine Erweiterung des globalen Alignments. Das heißt aber auch, dass der Smith-Waterman-Algorithmus auf den

Needleman-Wunsch-Algorithmus basiert.

Der maximale Score kann überall in der Scorematrix stehen. Somit kann eine kurze Teilsequenz ermittelt werden, die eine hohe Ähnlichkeit zu der anderen (Teil-)Sequenz hat.

1.1.3. HMM-Alignment

Das Hidden Markov Modell ist ein statistisches Modell der Wahrscheinlichkeiten. Dieses ist bereits in der Signalverarbeitung stark verbreitet. So werden zum Beispiel Spracheingaben mittels dem Hidden Markov Modells durch den Baum-Welch-Algorithmus berechnet. Das Hidden Markov Modell wird meist schlicht am Beispiel mit der Wettervorhersage erklärt.

2. Ergebnisse

2.1. Localaligner

Der Localaligner wurde in den Sprachen Perl, Java und C geschrieben. Dabei war der Fokus zu ermitteln wie sehr sich die Sprachen in ihrer Geschwindigkeit unterscheiden. Der Sourcecode für die eben genannten liegt im Anhang A, B und C.

- in Perl
- in Java
- in C B

2.2. Erstellung eines WIOS-Moduls (HMM)

1. Definition des Formates $\longrightarrow$ Anbindung an das GNM-Format/HTK-Format?
   a) **GNM-Format:** *blub*
   b) **HTK-Format:** *blub*
2. Input: FASTA, Output: HMM: a/b-Matrix
3. Modul mit Reads-Input 50 x 3Milliarden Baum-Welchen $\longrightarrow$ Wie Beobachtung gegen Sparse?

   Sparse *sind Dateien in einer platzsparenden Speicherform, wo nur die Positionen der "1"gespeichert werden. Das ist vor allem praktisch für Dateien, die viele aufeinanderfolgende Bytes mit dem Wert 0 enthalten und somit wenig Informationsdichte aufweisen.*

A. Localaligner in Perl

```
#!/usr/bin/perl
use strict;
open( INS, "<seq1.txt" ) || die "Datei seq1.txt nicht gefunden\n";
my @Seq1;
my @Seq2;
my @seq1;
my @seq2;
while (<INS>) {
    push( @Seq1, $_ );
}
close(INS);
open( INS2, "<seq2.txt" ) || die "Datei seq2.txt nicht gefunden\n";
while (<INS2>) {
    push( @Seq2, $_ );
}
close(INS2);

@seq1 = ( ' ', split( //, $Seq1[0] ) );
@seq2 = ( ' ', split( //, $Seq2[0] ) );
my $Penalty = 10;
my $gp      = -$Penalty;          # gap/substituion penalty
my $fp      = $Penalty;           # fit penalty
my $n       = 0;
my $l       = 0;
my $N       = $#seq2 - $#seq1;
my $tmp     = 10;
my $tmp2    = -100000000000000;
my $help    = 1;
my @amatch  = ("");
my @aseq1   = ("");
my @aseq2   = ("");
my @M;
my @S;
my @T;

while ( $n != $N ) {
    my $t = $n + $#seq1 - 1;
    my $s = $#seq1 - 1;
    for ( my $i = 0 ; $i <= $s ; $i++ ) {
        for ( my $j = $n ; $j <= $t ; $j++ ) {
            if ( $seq1[$i] ne $seq2[$j] ) {
                $M[$i][$j] = $gp;
            }    #subsitution penalty
            else {
```

```
                $M[$i][$j] = $fp;
            }    #fit penalty
        }
    }
    my $MINSCORE = -1000000000;
    for ( my $i = 0 ; $i <= $s ; $i++ ) {
        for ( my $j = $n ; $j <= $t ; $j++ ) {
            $S[$i][$j] = $MINSCORE;
        }
    }
    my $k = 0;
    for ( my $j = $n ; $j <= $t ; $j++ ) {
        $S[0][$j] = $k * $gp + $M[0][$j];
        $T[0][$j] = -1;
        $k++;
    }

    for ( my $i = 0 ; $i <= $s ; $i++ ) {
        $S[$i][$n] = $i * $gp + $M[$i][$n];
        $T[$i][$n] = 1;
    }
    my $MaxScore   = $MINSCORE;
    my $TotalScore = 0;
    for ( my $i = 1 ; $i <= $s ; $i++ ) {
        for ( my $j = $n + 1 ; $j <= $t ; $j++ ) {
            my $sc  = $S[ $i - 1 ][ $j - 1 ] + $M[$i][$j];
            my $sc2 = $S[ $i - 1 ][ $j - 1 ] + $M[$i][$j];
            $S[$i][$j] = $sc;
            $T[$i][$j] = 0;
            $TotalScore += $sc;
            $sc = $S[ $i - 1 ][$j] + $gp;
            if ( $sc > $S[$i][$j] ) {
                $S[$i][$j]  = $sc;
                $T[$i][$j]  = 1;
                $TotalScore = $TotalScore - $sc2 + $sc;
            }
            $sc2 = $S[ $i - 1 ][$j] + $gp;
            $sc  = $S[$i][ $j - 1 ] + $gp;
            if ( $sc > $S[$i][$j] ) {
                $S[$i][$j]  = $sc;
                $T[$i][$j]  = -1;
                $TotalScore = $TotalScore - $sc2 + $sc;
            }
            if ( $S[$i][$j] > $MaxScore ) {
                $MaxScore = $S[$i][$j];
            }
        }
    }
    $a = 0;
```

```
    while ( $help == 1 ) {
        if ( $T[$s][$t] == 0 ) {
            $aseq1[$a]  = $seq1[$s];
            $amatch[$a] = '*';
            $aseq2[$a]  = $seq2[$t];
            $s--;
            $t--;
        }
        else {
            if ( $T[$s][$t] == 1 ) {
                $aseq1[$a]  = $seq1[$s];
                $amatch[$a] = ' ';
                $aseq2[$a]  = '-';
                $s--;
            }
            else {
                if ( $T[$s][$t] == -1 ) {
                    $aseq1[$a]  = '-';
                    $amatch[$a] = ' ';
                    $aseq2[$a]  = $seq2[$t];
                    $t--;
                }
            }
        }
        $a++;
        if ( $s == 0 && $t == $n ) { $help = 0; }
    }
    if ( $MaxScore >= $tmp && $TotalScore >= $tmp2 ) {
        $tmp  = $MaxScore;
        $tmp2 = $TotalScore;
        $l++;
        printf "\nn=$l, max= $tmp, total=$tmp2, Alignment an Position $n:\n\t";
        for ( my $i = 0 ; $i <= $a ; $i++ ) {
            printf "$aseq1[$i]";
        }
        print "\n\t";
        for ( my $i = 0 ; $i <= $a ; $i++ ) {
            print "$amatch[$i]";
        }
        print "\n\t";
        for ( my $i = 0 ; $i <= $a ; $i++ ) {
            print "$aseq2[$i]";
        }
    }
    $n++;
    $help = 1;
}
```

B. Localaligner in C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define L 4000

int penalty = 10;
int x, i, j, a, k, l, N, s, t, tmp, tmp2, minScore, maxScore, totalScore, sc, sc2,
    N1, N2, gp, fp;
char seq[100 + 1], seq1[100 + 2], seq2[3227 + 2], seq3[3227 + 1];
int M[L][L], S[L][L], T[L][L];
char align1[L], align2[L], amatch[L], merke1[L], merke2[L], mmatch[L];
enum boolean {
    FALSE, TRUE
} help;

int main(void) {
    gp = -penalty;
    fp = penalty;
    help = TRUE;
    x = 0, l = 0, tmp = 0, tmp2 = -100000;
    FILE *datei1, *datei2;
    datei1 = fopen("seq1.txt", "r");
    if (datei1 != NULL ) {
        fscanf(datei1, "%100c", seq);
        seq[101] = '\0';
        fclose(datei1);
    }
    seq1[0] = ' ';
    for (i = 1; i <= strlen(seq); i++) {
        seq1[i] = seq[i - 1];
    }
    datei2 = fopen("seq2.txt", "r");
    if (datei2 != NULL ) {
        fscanf(datei2, "%3227c", seq3);
        seq3[3227] = '\0';
        fclose(datei2);
    }
    seq2[0] = ' ';
    for (i = 1; i <= strlen(seq3); i++) {
        seq2[i] = seq3[i - 1];
    }
    int N1 = strlen(seq1), N2 = strlen(seq2);
    int N = N2 - N1 - 2;
    for (i = 0; i < L; i++) {
        align1[i] = align2[i] = amatch[i] = merke1[i] = merke2[i] = mmatch[i] =
                0;
```

```
        for (j = 0; j < L; j++) {
            M[i][j] = 0;
            S[i][j] = 0;
            T[i][j] = 0;
        }
    }
    printf("seq1:%s\nseq2:%s\n\nseq1laenge: %d\tseq2laenge: %d\tDiff N: %d\n\n",
        seq1, seq2, N1, N2, N);
    while (x != N) {
        s = N1 - 1;
        t = x + N1 - 1;
        for (i = 0; i <= s; i++) {
            for (j = x; j <= t; j++) {
                if (seq1[i] == seq2[j]) {
                    M[i][j] = fp;
                } else {
                    M[i][j] = gp;
                }
            }
        }
        minScore = -100000;
        for (i = 0; i <= s; i++) {
            for (j = x; j <= t; j++) {
                S[i][j] = minScore;
            }
        }
        k = 0;
        for (j = x; j <= t; j++) {
            S[0][j] = k * gp + M[0][j];
            T[0][j] = -1;
            k++;
        }
        for (i = 0; i <= s; i++) {
            S[i][x] = i * gp + M[i][x];
            T[i][x] = 1;
        }
        maxScore = minScore;
        totalScore = 0;
        for (i = 1; i <= s; i++) {
            for (j = x + 1; j <= t; j++) {
                sc = S[i - 1][j - 1] + M[i][j];
                S[i][j] = sc;
                T[i][j] = 0;
                sc = S[i - 1][j] + gp;
                if (sc > S[i][j]) {
                    S[i][j] = sc;
                    T[i][j] = 1;
                }
                sc = S[i][j - 1] + gp;
```

```
                if (sc > S[i][j]) {
                    S[i][j] = sc;
                    T[i][j] = -1;
                }
                if (S[i][j] > maxScore) {
                    maxScore = S[i][j];
                }
            }
        }
        totalScore = S[s][t];
        a = 0;
        while (help == TRUE) {
            if (T[s][t] == 0) {
                align1[a] = seq1[s];
                amatch[a] = '*';
                align2[a] = seq2[t];
                s--;
                t--;
            } else if (T[s][t] == 1) {
                align1[a] = seq1[s];
                amatch[a] = ' ';
                align2[a] = '-';
                s--;
            } else if (T[s][t] == -1) {
                align1[a] = '-';
                amatch[a] = ' ';
                align2[a] = seq2[t];
                t--;
            }
            a++;
            if (s == 0 && t == x) {
                help = FALSE;
            }
        }
        if (totalScore >= tmp2) {
            tmp = maxScore;
            tmp2 = totalScore;
            l++;
            for (i = a; i >= 0; i--) {
                merke1[a - i] = align1[i];
                mmatch[a - i] = amatch[i];
                merke2[a - i] = align2[i];
            }
            printf("n=%d, max= %d, total= %d, Alignment an Position %d:\n\t",l,
                maxScore, totalScore, x);
            for (i = 1; i <= a; i++) {
                printf("%c", merke1[i]);
            }
            printf("\n\t");
```

```
            for (i = 1; i <= a; i++) {
                printf("%c", mmatch[i]);
            }
            printf("\n\t");
            for (i = 1; i <= a; i++) {
                printf("%c", merke2[i]);
            }
            printf("\n\n");
        }
        x++;
        help = TRUE;
    }
    return 0;
}
```

C. Localaligner in Java

```
import java.io.*;

public class aligner {
    public static void main(String[] args) throws IOException {
        String filename1 = args[0] + ".txt";
        String filename2 = args[1] + ".txt";
        String Seq1 = getsequences(filename1);
        String Seq2 = getsequences(filename2);
        System.out.print("seq1: ");
        print(Seq1);
        System.out.print("seq2: ");
        print(Seq2);
        System.out.println();
        int penalty = 10;
        int fp = penalty; // fit
        int gp = -penalty;
        int sp = -penalty; // gap,substitution
        int a, k, l, n, N, s, t, tmp, tmp2, minScore, maxScore, totalScore, sc, sc2
            ;
        boolean help = true;
        int[][] M, S, T;
        char[] align1, align2, amatch, merke1, merke2, mmatch;
        n = 0;
        l = 0;
        tmp = 0;
        tmp2 = -1000000000;
        N = (Seq2.length() - Seq1.length());
        M = new int[Seq1.length()][Seq2.length()];
        S = new int[Seq1.length()][Seq2.length()];
```

```
        T = new int[Seq1.length()][Seq2.length()];
        align1 = new char[Seq2.length()];
        align2 = new char[Seq2.length()];
        amatch = new char[Seq2.length()];
        merke1 = new char[Seq2.length()];
        merke2 = new char[Seq2.length()];
        mmatch = new char[Seq2.length()];
        while (n != N) {
            s = Seq1.length() - 1;
            t = n + Seq1.length() - 1;
            for (int i = 0; i <= s; i++) {
                for (int j = n; j <= t; j++) {
                    if (Seq1.charAt(i) == Seq2.charAt(j)) {
                        M[i][j] = fp;
                    } else {
                        M[i][j] = sp;
                    }
                }
            }
            minScore = -1000000000;
            for (int i = 0; i <= s; i++) {
                for (int j = n; j <= t; j++) {
                    S[i][j] = minScore;
                }
            }
            k = 0;
            for (int j = n; j <= t; j++) {
                S[0][j] = k * gp + M[0][j];
                T[0][j] = -1;
                k++;
            }
            for (int i = 0; i <= s; i++) {
                S[i][n] = i * gp + M[i][n];
                T[i][n] = 1;
            }
            maxScore = minScore;
            totalScore = 0;
            for (int i = 1; i <= s; i++) {
                for (int j = n + 1; j <= t; j++) {
                    sc = sc2 = S[i - 1][j - 1] + M[i][j];
                    S[i][j] = sc;
                    T[i][j] = 0;
                    totalScore += sc;
                    sc = S[i - 1][j] + gp;
                    if (sc > S[i][j]) {
                        S[i][j] = sc;
                        T[i][j] = 1;
                        totalScore = totalScore - sc2 + sc;
                    }
```

```
                sc2 = S[i - 1][j] + gp;
                sc = S[i][j - 1] + sp;
                if (sc > S[i][j]) {
                    S[i][j] = sc;
                    T[i][j] = -1;
                    totalScore = totalScore - sc2 + sc;
                }
                if (S[i][j] > maxScore) {
                    maxScore = S[i][j];
                }
            }
        }
        a = 0;
        while (help == true) {
            if (T[s][t] == 0) {
                align1[a] = Seq1.charAt(s);
                amatch[a] = '*';
                align2[a] = Seq2.charAt(t);
                s--;
                t--;

            } else if (T[s][t] == 1) {
                align1[a] = Seq1.charAt(s);
                amatch[a] = ' ';
                align2[a] = '-';
                s--;
            } else if (T[s][t] == -1) {
                align1[a] = '-';
                amatch[a] = ' ';
                align2[a] = Seq2.charAt(t);
                t--;
            }
            a++;
            if (s == 0 && t == n) {
                help = false;
            }
        }
        if (maxScore >= tmp && totalScore >= tmp2) {
            tmp = maxScore;
            tmp2 = totalScore;
            l++;
            for (int i = a; i >= 0; i--) {
                merke1[a - i] = align1[i];
                mmatch[a - i] = amatch[i];
                merke2[a - i] = align2[i];
            }
            System.out.println("n= " + l + ", maxScore= " + tmp
                    + ", totalScore= " + tmp2 + ", Alignment an Position "
                    + n + ":");
```

```
                printc(merke1, a);
                printc(mmatch, a);
                printc(merke2, a);
                System.out.println();
            }
            n++;
            help = true;
        }
    }

    // ------------------------Printsubs------------------------
    public static void print(String seq) {
        for (int i = 0; i < seq.length(); i++) {
            System.out.print(seq.charAt(i));
        }
        System.out.println("");
    }

    public static void printc(char align[], int a) {
        System.out.print("\t");
        for (int i = 1; i <= a; i++) {
            System.out.print(align[i]);
        }
        System.out.println();
    }

    // ------------------------Einlesen------------------------
    public static String getsequences(String file) {
        String Seq = new String();
        Seq = " ";
        try {
            BufferedReader rdr = new BufferedReader(new FileReader(file));
            String strLine = null;
            while ((strLine = rdr.readLine()) != null) {
                Seq += strLine;
            }
            rdr.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return Seq;
    }
}
```

D. Globalaligner in C

E. Globalaligner in Java

```
import java.io.*;
import java.util.*;

/**
 * command:
 * <code>java Main read-file genome-file pruning[0:1] buffer[0:100]</code>
 **/

public class jglobalaligner {
    public static void main(String[] args) {
        final byte repos = 16;
        byte readcount = 0;
        boolean hilfe = false;
        int Nseq2 = 0; // position in chr
        long startTime = System.currentTimeMillis(), saveTime = startTime, midTime,
             readTime;
        long pos;
        PrintWriter out = new PrintWriter(System.out, true);
        PrintWriter err = new PrintWriter(System.err, true);
        int run = 0, time = 0;
        String file1 = args[0], file2 = args[1];
        String input1 = args[2], input2 = args[3];
        String[] wios = null, seq2 = null;
        err.printf("Start time: %tc\n\n", new Date());
        try {
            BufferedReader rdr1 = new BufferedReader(new FileReader(new File(
                    file1)));
            while ((wios = GetReads(rdr1)) != null) {
                readTime = System.currentTimeMillis();
                midTime = System.currentTimeMillis();
                err.println("read " + readcount + ":\t" + wios[0] + "\n\t 1\t"
                        + wios[1] + "\n\t-1\t" + wios[2]);
                // open second file
                RandomAccessFile raf = new RandomAccessFile(file2, "r");
                pos = 0;
                while ((seq2 = GetGenom(raf, pos, repos)) != null) {
                    if (seq2[1].length() <= 2)
                        break;
                    ++run; // align count
                    pos = Long.parseLong(seq2[3]);
                    if (run != 1)
                        Nseq2 = Nseq2 - repos;
                    // err.println("run " + run + ": from position " + Nseq2
```

```
                    // + " to " + (Nseq2 + seq2[1].length() - 1));
                    Nseq2 = align(wios, seq2, input1, input2, Nseq2, out, err,
                            hilfe, repos);
                    midTime = System.currentTimeMillis();
                    if ((midTime - startTime) >= (60 * 1000)) {// post every 2
                        minutes
                        ++time;
                        out.println("... " + time + " min, read " + readcount);
                        startTime = midTime;
                    }
                }
            //  out.println("run "+readcount+" needed "+((midTime - readTime)/1000)
                 +" seconds.");
                run = 0;
                raf.close();
                Nseq2 = 0;
            }
            rdr1.close();
        } catch (IOException e) {
            err.println("Could not read data!"); // System.exit(-1);
        }
        long endTime = System.currentTimeMillis();
        out.println("Total elapsed time: "
                + ((endTime - saveTime) / (60 * 1000)) + " minutes");
        err.printf("End time: %tc\n\n", new Date());
        err.println("Total elapsed time: "
                + ((endTime - saveTime) / (60 * 1000)) + " minutes");
    }

    public static int align(String[] seq1, String[] seq2, String input,
            String input2, int Nseq2, PrintWriter out, PrintWriter err,
            boolean h, short reposition) {
        // ------------------ Definition ----------------------
        final byte fit = 10;
        final byte penalty = -10;
        final byte buffer = Byte.valueOf(input2);
        final float pruning = Float.parseFloat(input);
        short N_seq1 = (short) seq1[1].length(), N_seq2 = (short) seq2[1]
                .length();
        byte it;
        short score = -1000, sc, ia, ja, imax, jmax, prebuffer;
        float output;
        boolean help = true;
        short[] S, T, S_sort;
        S = new short[N_seq1 * N_seq2];
        T = new short[N_seq1 * N_seq2];
        S_sort = new short[N_seq2];
        short[][] Mind = new short[2][N_seq1];
        String align1 = null, align2 = null, amatch = null;
```

```
90          // ------------------ Preparations ------------------
            for (short s = 1; s <= 2; s++) {
                Arrays.fill(S, score);  // Initializing
                for (short j = 0; j < N_seq2; j++) {
                    if (seq1[s].charAt(1) == seq2[s].charAt(j))
95                      S[j] = (short) (j * penalty + fit);
                    else
                        S[j] = (short) (j * penalty + penalty);
                    T[j] = -1;
                    if (S[j] < 0) { // only positive scores allowed!
100                     S[j] = 0;
                    }
                }
                for (short i = 1; i < N_seq1; i++) {
                    if (seq1[s].charAt(i) == seq2[s].charAt(1))
105                     S[i * N_seq2] = (short) (i * penalty + fit);
                    else
                        S[i * N_seq2] = (short) (i * penalty + penalty);
                    T[i * N_seq2] = 1;
                    if (S[i * N_seq2] < 0) { // only positive scores allowed!
110                     S[i * N_seq2] = 0;
                    }
                }
            // ------------------ Alignment ------------------
                sc = 0;
115             for (short i = 1; i < N_seq1; i++) {
                    for (short j = 1; j < N_seq2; j++) {
                        if (seq1[s].charAt(i) == seq2[s].charAt(j))
                            sc = (short) (S[(i - 1) * N_seq2 + j - 1] + fit);
                        else
120                         sc = (short) (S[(i - 1) * N_seq2 + j - 1] + penalty);
                        S[i * N_seq2 + j] = sc;
                        T[i * N_seq2 + j] = 0;
                        sc = (short) (S[(i - 1) * N_seq2 + j] + penalty);
                        if (sc > S[i * N_seq2 + j]) {
125                         S[i * N_seq2 + j] = sc;
                            T[i * N_seq2 + j] = 1;
                        }
                        sc = (short) (S[i * N_seq2 + j - 1] + penalty);
                        if (sc > S[i * N_seq2 + j]) {
130                         S[i * N_seq2 + j] = sc;
                            T[i * N_seq2 + j] = -1;
                        }
                        if (S[i * N_seq2 + j] < 0) {
                            // only positive scores allowed!
135                         S[i * N_seq2 + j] = 0;
                        }
                    }
                }
```

```
        // ---------- Storing max(score) ----------
imax = (short) (N_seq1 - 1);
for (short i = 0; i < N_seq2 - 1; i++) {    // S_sort: last row from S
    S_sort[i] = S[imax * N_seq2 + i];
}
Arrays.sort(S_sort);
output = (N_seq1 - 1) * 10 - (N_seq1 - 1) * pruning;
it = 0;
for (short j_sort = (short) (N_seq2 - 1); j_sort > 0; j_sort--) {
    if (S_sort[j_sort] < output) {
        break;
    } else if (S_sort[j_sort] == S_sort[j_sort - 1]) {
        continue;
    } else {
        for (short j = 0; j <= N_seq2 - 1; j++) {
            if (S[imax * N_seq2 + j] == S_sort[j_sort]) {
                Mind[0][it] = j;
                Mind[1][it] = S[imax * N_seq2 + j];
                it++;
            }
        }
    }
}
char mark;
// ---------- Print ----------
for (short n = 0; n < it; n++) {
    StringBuilder builder1 = new StringBuilder(), builder2 = new
        StringBuilder(), builder3 = new StringBuilder();
    jmax = Mind[0][n]; // position with best scores first
    score = Mind[1][n]; // score of position jmax
    ia = imax;
    ja = jmax;
    // ---------- Storing print alignment ----------
    while (help == true) {
        mark = ' ';
        if (T[ia * N_seq2 + ja] == 0) {
            if (seq1[s].charAt(ia) == seq2[s].charAt(ja))
                mark = ':';
            else
                mark = '*';
            builder1.insert(0, seq1[s].charAt(ia));
            builder2.insert(0, mark);
            builder3.insert(0, seq2[s].charAt(ja));
            ia--;
            ja--;
        } else if (T[ia * N_seq2 + ja] == 1) {
            builder1.insert(0, seq1[s].charAt(ia));
            builder2.insert(0, mark);
            builder3.insert(0, '-');
```

```
                ia--;
            } else if (T[ia * N_seq2 + ja] == -1) {
                builder1.insert(0, '-');
                builder2.insert(0, mark);
                builder3.insert(0, seq2[s].charAt(ja));
                ja--;
            }
            if (ia == 0 && ja < (jmax - imax + 1)) {
                help = false;
            }
        }
        mark = ' ';
        // ------------------------ Printing with buffer ------------------
        if (score >= output) {
            if (s == 1) // forward
                err.println("n=" + (n + 1) + " score=" + score
                        + ", read aligns from position "
                        + (Nseq2 + ja + 1) + " to "
                        + (Nseq2 + jmax + 1) + ":\n");
            else // backward
                err.println("n=" + (n + 1) + " score=" + score
                        + ", reversed read aligns from position "
                        + (Nseq2 + (1200 - 16) - ja + 1)
                        + " to "
                        + (Nseq2 + (1200 + 32 + 16) - jmax + 1)
                        + ":\n");
            prebuffer = buffer;
            if (ja - prebuffer < 0) {
                // prebuffer is too big -> set prebuffer=j
                prebuffer = ja;
            }
            for (short i = 0; i < prebuffer; i++) {
                // adds sequence before alignment
                builder1.insert(0, mark);
                builder2.insert(0, mark);
                builder3.insert(0, seq2[s].charAt(ja - i));
            }
            for (short j = (short) (jmax + 1); j < jmax + buffer; j++) {
                // adds sequence after alignment
                if (j == N_seq2)
                    break;
                builder3.append(seq2[1].charAt(j));
            }
            align1 = builder1.toString();
            amatch = builder2.toString();
            align2 = builder3.toString();
            err.println(align1);
            err.println(amatch);
            err.println(align2 + "\n");
```

```
                    help = true;
                }
            }
        }
        Nseq2 += N_seq2 - 1;
        return Nseq2;
    }

    // ---------------------------------- Input ----------------------------------------
    public static String[] GetReads(BufferedReader rdr) throws IOException {
        StringBuilder read = new StringBuilder(), read_rev = new StringBuilder();
        read.append(' '); // do not remove the space, this is for alignment
        read_rev.append(' '); // do not remove the space, this is for alignment
        String strLine = rdr.readLine();
        rdr.mark(1000);
        if (strLine != null) {
            String[] split = strLine.split("\t");
            String[] seq = new String[3];
            seq[0] = (split[0] + "\t" + split[11]);
            read.append(split[2]);
            for (short i = 1; i < (short) read.length(); i++) {
                if (read.charAt(i) == 'A')
                    read_rev.append('T');
                else if (read.charAt(i) == 'T')
                    read_rev.append('A');
                else if (read.charAt(i) == 'G')
                    read_rev.append('C');
                else
                    read_rev.append('G');
            }
            seq[1] = read.toString();
            seq[2] = read_rev.toString();
            return seq;
        } else
            return null;
    }

    public static String[] GetGenom(RandomAccessFile raf, long pos_beg,
            short repos) throws IOException {
        StringBuilder sequence = new StringBuilder(), sequence_rev = new
            StringBuilder();
        long pos;
        if (pos_beg != 0)
            pos = pos_beg - ((long) repos) - 1;
        else
            pos = 0; // first run
        short strlen;
        String[] seq = new String[4]; // 0:name 1:sequence 2:reversed 3:position
        sequence.append(' ');// do not remove the space, this is for alignment
```

```
        while (sequence.length() < 1200) {
285         raf.seek(pos);
            String strLine = raf.readLine();
            if (strLine == null) // EOF
                break;
            strlen = (short) strLine.length();
290         String strtrim = strLine.trim(); // trim: without space,tabs,breaks
            if ((pos + repos) >= (long) (raf.length() - repos))
                break;
            else if (strLine.charAt(0) == '>')
                seq[0] = strLine;
295         else {
                sequence.append(strtrim);
            }
            pos += strlen + 1;
        }
300     seq[1] = sequence.toString();
        sequence.append(' '); // do not remove the space, this is for alignment
        sequence_rev = sequence.reverse(); // reverse imported sequence
        sequence_rev.deleteCharAt(sequence_rev.length() - 1); // remove the space
            at the end
        seq[2] = sequence_rev.toString();
305     seq[3] = Long.toString(pos);
        return seq;
    }
}
```