BAB III IMPLEMENTASI

Dalam bab ini dijelaskan mengenai implementasi sistem yang dikembangkan untuk melakukan percobaan terhadap metode yang diteliti. Penjelasan meliputi perangkat keras, perangkat lunak, alur kerja sistem dan implementasi sistem pengukuran kemiripan DNA.

III.1. Spesifikasi Perangkat

Beberapa perangkat keras dan lunak digunakan dalam pengembangan sistem pengukuran DNA ini. Perangkat-perangkat tersebut dirinci pada subbab ini.

III.1.1. Perangkat Keras

Perangkat keras komputer yang digunakan untuk mengembangkan sistem ini terdiri dari satu buah komputer *desktop* dan satu buah *notebook*. Spesifikasi *notebook* adalah sebagai berikut:

- Acer Aspire 2920
- Prosessor : Intel® Core ™ 2 Duo T5500 @ 1.83 GHz
- Memori : 1GB

Spesifikasi komputer *desktop* adalah sebagai berikut:

- Processor : Intel® Celeron™ D 3.00 Ghz
- Memori : 1GB

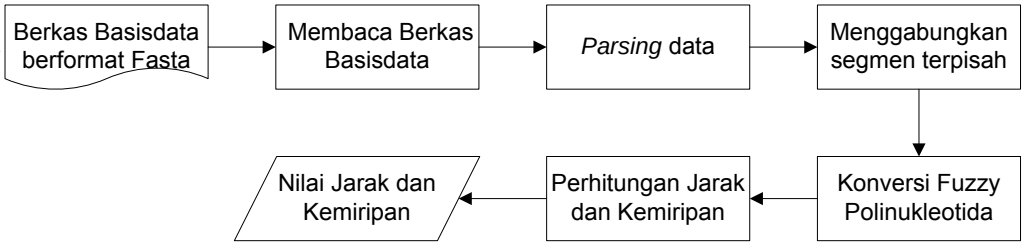
III.1.2. Perangkat Lunak

Perangkat Lunak yang digunakan untuk mengembangkan sistem ini adalah sebagai berikut:

Sistem Operasi	: Microsoft Windows XP Professional SP 3
Editor	: Eclipse Ganymede
Bahasa Pemograman	: Java
Java Standar Development Kit	: Java 1.6

III.2. Alur Proses Sistem

Sistem yang dibuat memiliki proses sebagai berikut.



Gambar 10. Alur Kerja Sistem Pengukuran Jarak Rantai Polinukleotida

Sistem menerapkan metode perhitungan jarak dan kemiripan yang telah dijelaskan pada Bab II.

III.3. Implementasi Algoritma

Implementasi sistem untuk mengukur kemiripan DNA ini masih cukup sederhana.

III.3.1. Pembacaan Basisdata Berformat Fasta

Basisdata berformat fasta dapat menyimpan banyak rantai DNA dalam satu *file*. Pada implementasi membaca *file* dengan format fasta ini, setiap rantai akan dijadikan struktur data masing-masing. Sebelumnya, maka perlu dilakukan dahulu pemisahan setiap rantai dalam *file* berformat fasta. Fungsi untuk memisahkan masing-masing rantai adalah sebagai berikut.

```
/*
 * Fungsi mengembalikan array of string.
 * Fungsi ini untuk memisahkan keseluruhan string dalam file berformat
FASTA.
 * Pemisahan dilakukan berdasarkan tanda ">".
 * Sehingga menghasilkan array of String yang memiliki anggota rantai
DNA.
 * Setiap rantai dan definisinya dalam sebuah anggota array.
 * @param    paramString String yang berisi terdiri dari banyak rantai
dalam format FASTA
 * @return   array yang berisi satu buah FASTA dalam satu anggota
 */
public static String[] splitMultiStringFasta(String paramString){
       StringTokenizer strToken = new StringTokenizer(paramString, ">");
       int sumToken = strToken.countTokens();
```

```
        String[] strFasta = new String[sumToken];
        for(int ii = 0; strToken.hasMoreTokens(); ii++) {
                strFasta[ii] = ">"+strToken.nextToken();
        }
        return strFasta;
}
```

Dari hasil pemisahan string yang telah dilakukan, masing-masing string berformat fasta tersebut dirubah menjadi struktur data untuk fasta dengan fungsi sebagai berikut.

```
/*
 * Fungsi mengembalikan struktur data FASTA.
 * Fungsi mengkonversi FASTA yang berbentuk string menjadi
struktur data FASTA.
 * @param   paramString berisi string dalam format FASTA
 * @return  Mengembalikan struktur data FASTA
 * @see     Fasta
 */
public static Fasta stringToFasta(String paramString){
        String def = "", seq = "",temp ="";
        StringTokenizer strToken = new StringTokenizer(paramString,"\n");
        while (strToken.hasMoreTokens()) {
                temp = strToken.nextToken();
                if(temp.charAt(0)== '>'){
                        def = temp.substring(1);
                } else {
                        seq = seq.concat(temp);
                }
        }
        return new Fasta(def, seq);
}
```

Fungsi untuk merubah masing-masing *string* fasta berformat fasta menjadi struktur data fasta akan dipanggil berulangkali pada sebuah *looping* apabila dalam yang dibaca terdapat beberapa rantai yang didefinisikan.

III.3.2. Penggabungan segmen DNA terpisah dalam satu genom

Rantai DNA dalam sebuah makhluk hidup seringkali tersimpan dalam banyak segmen atau dalam banyak format fasta. Perlu dilakukan penggabungan setiap

segmen tersebut. Berikut fungsi yang bertugas menggabungkan dua buah rantai DNA yang telah berstruktur data fasta.

```
/*
 * Menambahkan string pada rantai DNA.
 *
 * @return  mengembalikan FASTA yang telah ditambah sequencenya
 */
public Fasta concat(String paramString){
      this.sequence = this.sequence.concat(paramString);
      return this;
}
```

Rantai yang digabungkan biasanya lebih dari dua buah rantai. Oleh karena itu, penggabungan dilakukan dengan memanggil beberapa kali fungsi ini. Dalam data percobaan semua rantai dalam milik sabuah virus tertentu disimpan dalam satu *file*. Hal ini dilakukan secara manual sehingga setiap *file* yang menjadi masukan adalah sebuah *file* berformat fasta yang terdiri dari segmen-segmen dalam satu virus yang telah diurutkan.

III.3.3. Implementasi struktur data Fasta.

Setiap rantai DNA berformat fasta disimpan dalam sebuah kelas Fasta. Kelas Fasta tersebut didefinisikan sebagai berikut (potongan kode).

```
/*
 * Representasi data setiap rantai DNA dalam format fasta.
 */
public class Fasta {
      private String definition;
      private String sequence;
...
       * Class Constructor dengan parameter definisi dan sequence
       */
      public Fasta(String paramString1, String paramString2){
          this.definition = paramString1;
          this.sequence = paramString2;
      }
...
      /*
       * Menambahkan string pada rantai DNA.
       * @return    mengembalikan FASTA yang telah ditambah sequencenya
       */
      public Fasta concat(String paramString){
```

```
            this.sequence = this.sequence.concat(paramString);
            return this;
        }
        /*
         * Menghitung panjang rantai DNA
         * @return    mengembalikan besar panjang DNA.
         */
        public int getLength(){
            return sequence.length();
        }
}
```

III.3.4. Konversi Fasta Menjadi Fuzzy Polinukleotida

Setiap struktur data fasta dikonversi menjadi struktur data Nukleotida kemudian dikonversi menjadi struktur data Fuzzy Nukleotida.

```
/*
 * Merubah Nukleotida menjadi fuzzy nukleotida.
 * Panjang array yang diperoleh adalah n*4 (panjang nukleotida*4).
 * Satu Nukleotida dibutuhkan 4 array Fuzzy Nukleotida
 * @return    Mengembalikan array bertipe double
 */
public double[] toFuzzyNucleotide(){
double[] result = new double[length*4];
int indexNuclei = 0, kk = 0;
        for(int ii = 0; ii < length; ii++){
            if(seqs[ii] == 'T'){indexNuclei = 0;}
            else if(seqs[ii] == 'C'){indexNuclei = 1;}
            else if(seqs[ii] == 'A'){indexNuclei = 2;}
            else if(seqs[ii] == 'G'){indexNuclei = 3;}
        result[kk+indexNuclei] = 1;
        kk+=4;
        }
return result;
}
```

Dalam mengkonversi polinukleotida menjadi *fuzzy* nukleotida sebelumnya dihitung dahulu jumlah masing-masing elemen setiap triplet kodon. Perhitungan itu menggunakan fungsi *countStatisticNucleotide*.

```
/*
 * Menghitung jumlah nukelotida yang bertepatan dalam bentuk fuzzy
polinukleotida
```

```
 * Nukleotida n bertepatan dengan n%paramLength(n modulo paramLength).
 *
 * @return    Mengembalikan array bertipe integer dengan panjang
paramLength*4
 * @param     Panjang nukleotida yang akan dihasilkan untuk dihitung
 */
public int[] countStatisticNucleotide(int paramLength) {
	int kk = 0, indexNuclei = 0, result[] = new int[paramLength*4];
	for(int ii = 0; ii < length; ii++){
		if(seqs[ii] == 'T'){indexNuclei = 0;}
		else if(seqs[ii] == 'C'){indexNuclei = 1;}
		else if(seqs[ii] == 'A'){indexNuclei = 2;}
		else if(seqs[ii] == 'G'){indexNuclei = 3;}

		result[(kk+indexNuclei)] += 1;
		if((ii+1)%paramLength == 0){
			kk = 0;
		} else {
			kk+=4;
		}
	}
	return result;
}
```

Setelah menghitung banyaknya nukleotida pada setiap kodon maka selanjutnya dapat dikonversi menjadi *fuzzy* polinukleotida dengan membagi jumlah yang telah dihitung dengan banyaknya kodon yang dihitung. Fungsi yang digunakan untuk ini adalah fungsi toFuzzyNucleotide.

```
/*
 * Konversi Nukleotida menjadi fuzzy Nukleotida dengan menetapkan panjang
fuzzy nukelotida.
 * Nilainya yang diperoleh berdasarkan hasil rata-rata statistik.
 *
 * @return    Mengembalikan array bertipe double dengan panjang
paramLenght*4
 * @param     Menentukan panjang nukleotida yang dihasilkan (untuk kodon
panjangnya 3)
 */
public double[] toFuzzyNucleotide(int paramLength){
	if(paramLength <= this.length/2){
		double[] result = new double[paramLength*4];
```

```
                int[] statNucleotide =
countStatisticNucleotide(paramLength);
                int divisorStat = length/paramLength;
                int remainderStat = length%paramLength;
                divisorStat++;
                for(int ii = 0; ii<paramLength*4; ii++){
                        if(ii == (4*remainderStat)){divisorStat--;}
                        result[ii] =
(double)statNucleotide[ii]/(double)divisorStat;
                }

                return result;
        }else{
                return null;
        }
}
```

Setelah melalui beberapa pengolahan data dari fungsi yang disebutkan pada subbab ini, hasil akhirnya yaitu sebuah vektor berdimensi 12. Vektor tersebut direpresentasikan dalam tipe data array double. Vektor inilah yang menjadi salah satu ciri dari sebuah rantai polinukleotida.

III.3.5. Perhitungan Jarak Fuzzy Polinukleotida

Pengukuran jarak yang digunakan ada dua macam yang berbeda. Pengukuran pertama yaitu mengukur dengan fungsi *similar* dan *differ* yang ditawarkan Zadeh. Perhitungan kedua menggunakan perhitungan *sim* dan *dif* yang ditawarkan oleh Tores et al.

Berikut ini implementasi dari perhitungan similar yang ditawarkan oleh Zadeh.

```
/*
 * Definisi Similar from Zadeh 2000 menghitung dalam Fuzzy Nucleitide
Rumus (2.19)
 */
public static double getSimilar(FuzzyNucletiode paramFuzzyNucletiode1,
FuzzyNucletiode paramFuzzyNucletiode2) {
        double result = 0, dividen = 0, divisor = 0;
        for(int ii = 0; ii < paramFuzzyNucletiode1.length; ii++){
                dividen +=
Math.min(paramFuzzyNucletiode1.getNucleotideValue(ii),
paramFuzzyNucletiode2.getNucleotideValue(ii));
```

```
                divisor += (paramFuzzyNucletiode1.getNucleotideValue(ii) +
paramFuzzyNucletiode2.getNucleotideValue(ii))/2;
        }
        result = dividen/divisor;
        return result;
}
```

Berikut ini implementasi dari perhitungan similar yang ditawarkan oleh Tores et al.

```
/*
 * Menghitung dif(p, q) atau d(p,q) berdasarkan Nieto Tores
 */
public static double getDistance(FuzzyNucletiode paramFuzzyNucletiode1,
FuzzyNucletiode paramFuzzyNucletiode2){
        double result = 0, dividen = 0, divisor = 0;
        for(int ii = 0; ii < paramFuzzyNucletiode1.getlength(); ii++ ){
                dividen +=
Math.abs(paramFuzzyNucletiode1.getNucleotideValue(ii)-
paramFuzzyNucletiode2.getNucleotideValue(ii));
                divisor +=
Math.max(paramFuzzyNucletiode1.getNucleotideValue(ii),
paramFuzzyNucletiode2.getNucleotideValue(ii));
        }
        if(divisor != 0){
                result = dividen/divisor;
        }else{
                result = 0;
        }
        return result;
}
```

BAB IV UJI COBA DAN ANALISIS

Bab ini menjelaskan hasil uji coba yang dilakukan dengan menggunakan sistem yang dibuat. Penjelasan meliputi data yang digunakan, skenario uji coba dan hasil uji coba beserta analisis dari hasil uji coba yang telah dilakukan.

IV.1. Data Uji Coba

Penelitian ini mengambil data virus dari database dari *National Center for Biotechnology Information* (NCBI) (Bao, et al. 2008). Jumlah sampel virus yang diambil adalah 70 virus. Setiap virus tersimpan dalam beberapa segmen yang terpisah. Segmen-segmen terpisah itu digabungkan dalam sebuah *file* secara manual. Segmen disimpan dalam format fasta yang berurutan. Virus utama sebagai pembanding adalah virus Influenza A H5N1 dari situs NCBI (http://www.ncbi.nlm.nih.gov) dengan nomor akses segmen pertama yaitu CY016794.

Semua data virus yang digunakan adalah virus yang telah dipetakan genomnya secara keseluruhan. Rantai DNA yang diambil untuk perbandingan adalah rantai DNA dalam *coding region*. *coding region* yaitu bagian dari rantai DNA yang dapat ditranskripsi menjadi mRNA atau Protein.

IV.2. Skenario Uji Coba

Beberapa skenario uji dilakukan untuk mengetahui lebih dalam mengenai pengukuran dilakukan. Skenario tersebut diantaranya Pengukuran rantai DNA pada sesama virus H5N1, pengukuran virus influenza H5N1 dengan virus influenza yang memiliki subtipe berbeda dan pengukuran virus H5N1 dengan virus lainnya.

IV.2.1. Pengukuran DNA Virus Influenza Bertipe Sama

Skenario Pertama mencoba membandingkan sebuah virus influenza dengan virus yang bertipe sama. Virus yang diambil adalah virus bertipe H5N1 kemudian dibandingkan dengan virus H5N1 Lainnya. Percobaan skenario pertama ini bertujuan untuk mengukur tingkat persamaan dan perbedaan virus influenza yang bertipe sama.

IV.2.2. Pengukuran DNA Virus Influenza Berbeda Subtipe

Skenario kedua melakukan pengukuran rantai DNA penyandi H5N1 dengan rantai DNA penyandi virus influenza dengan tipe selain H5N1. Perbandingan ini ingin mengukur tingkat kemiripan antar virus dengan tipe yang berbeda. Hal ini dilakukan untuk melihat kemungkinan dapat membedakan antara subtipe virus influenza dengan pengukuran jarak yang dilakukan.

IV.2.3. Pengukuran DNA Virus Influenza Dengan Virus Lain

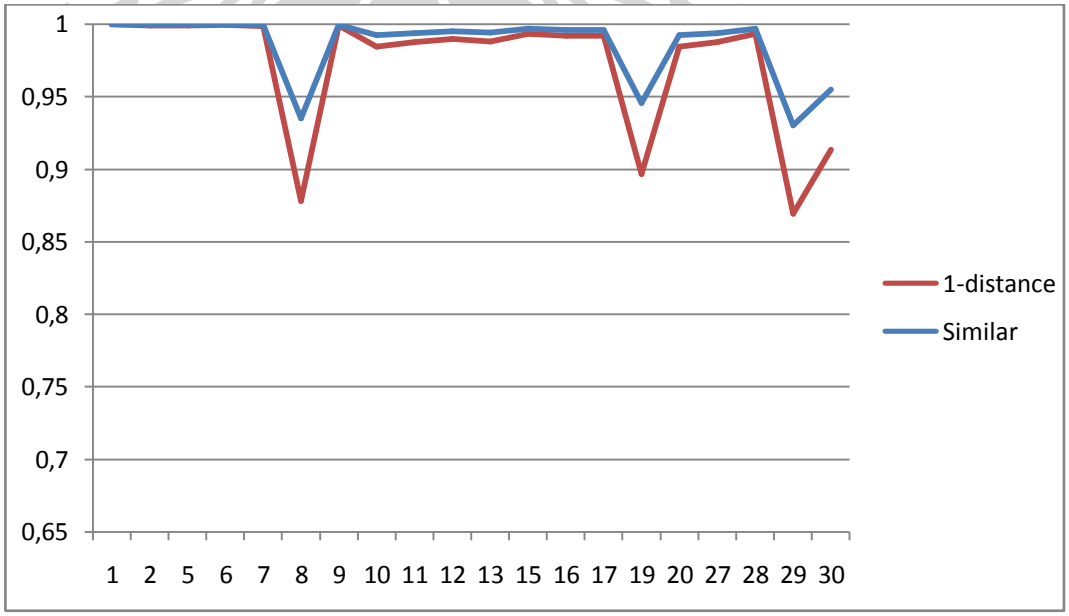
Skenario ketiga ini dilakukan untuk menguji kemampuan metode pengukuran ini untuk membadakan virus influenza H5N1 dengan virus jenis lainya. Hasil yang diharapkan nilai yang didapat memiliki kemiripan yang kecil dan jarak yang besar.

IV.3. Hasil Uji Coba

Pada bagian ini dijelaskan hasil uji coba yang telah dilakukan dengan skenario yang telah dibuat. Gambaran hasil yang diperoleh disajikan dalam bentuk grafik.

IV.3.1. Pengukuran DNA Virus Influenza Bertipe Sama

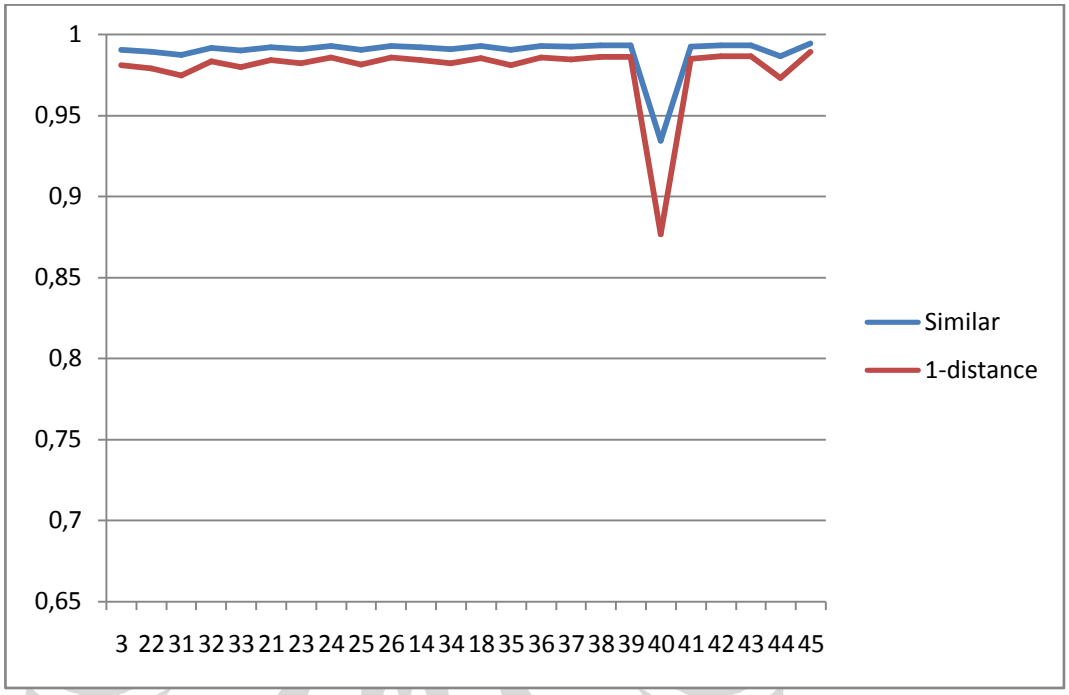
Hasil yang diperoleh adalah sebagai berikut. Data lebih lengkap dapat dilihat di lampiran.



Gambar 11. Grafik Hasil Pengukuran Sesama Virus Influenza H5N1.

IV.3.2. Pengukuran DNA Virus Influenza Berbeda Tipe

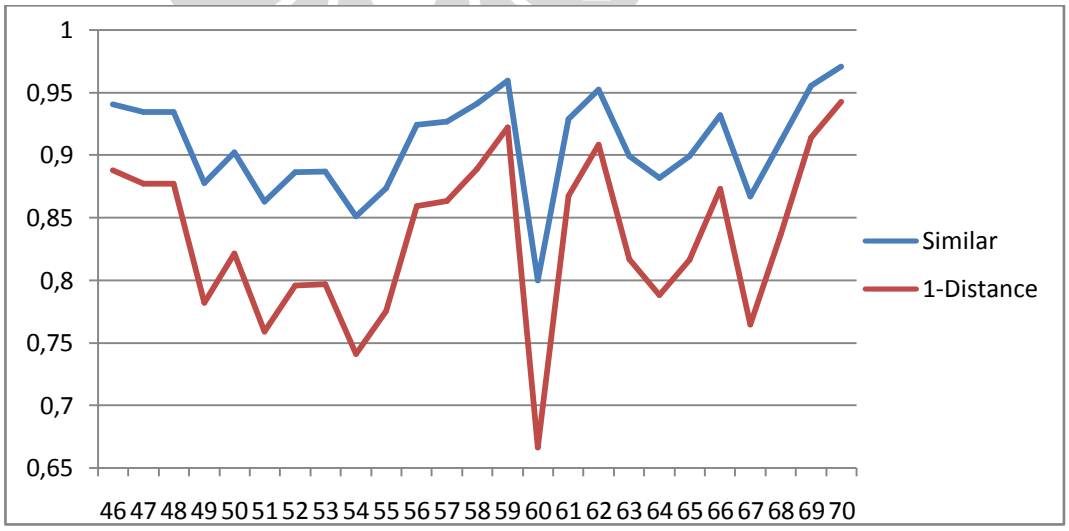
Hasil yang diperoleh dari pengukuran rantai DNA virus H5N1 dengan DNA Virus Influenza lainnya adalah sebagai berikut. Data lebih lengkap dapat dilihat di lampiran.



Gambar 12.Grafik Hasil Pengukuran Virus Influenza H5N1 dengan Virus Influenza Lainnya.

IV.3.3. Pengukuran DNA Virus Influenza Dengan Virus Lain

Hasil pengukuran dari rantai DNA virus influenza dengan virus jenis lainnya adalah sebagai berikut. Data lebih lengkap dapat dilihat di lampiran.



Gambar 13. Grafik Hasil Pengukuran Virus Influenza H5N1 dengan Virus Lainnya.

IV.4. Analisis Hasil Uji Coba

Analisis dari hasil uji coba dibahas pada bagian ini. Beberapa temuan juga dibahas pada bagian analisis uji coba ini.

IV.4.1. Pengukuran DNA Virus Influenza Bertipe Sama

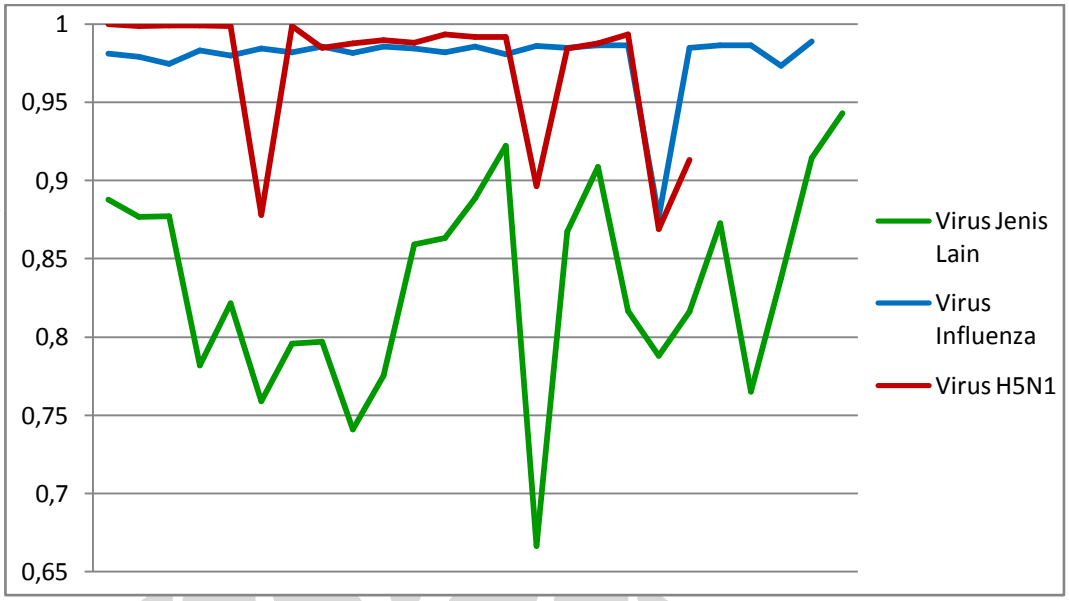
Hasil pengukuran kemiripan yang dilakukan menghasilkan nilai yang sangat tinggi. Dari 20 data, rata-rata kemiripan yang diperoleh dengan rumus Sadegh – Zadeh adalah 0.985394 dan untuk rumus Nieto Tores adalah 0.972154. Hal ini sesuai harapan yang diinginkan, virus sejenis seharusnya memiliki struktur yang tidak jauh berbeda. Terdapat beberapa data yang memiliki nilai berbeda dengan kebabanyakan data. Hal ini menunjukkan tingkat variasi yang cukup tinggi juga dimiliki oleh virus influenza H5N1 .

IV.4.2. Pengukuran DNA Virus Influenza Berbeda Tipe

Hasil pengukuran pada virus influenza yang berbeda tipe ternyata tidak sesuai harapan. Berdasarkan sampel yang diambil sebanyak 24 data, rantai DNA virus influenza H5N1 dengan virus influenza lainnya ternyata memiliki struktur yang tidak jauh berbeda. Rata-rata nilai kemiripan adalah 0.989108 dan 0.978698. Hasil yang Metode jarak ini sulit digunakan untuk membedakan antara satu virus dengan virus lainnya karena jarak yang begitu dekat.

IV.4.3. Pengukuran DNA Virus Influenza Dengan Virus Lain

Pengukuran pada skenario ketiga dilakukan antar rantai DNA virus influenza yang sama yang digunakan pada skenario sebelumnya dengan virus lainnya. Pada skenario ini terlihat variasi nilai yang lebih besar dibandingkan dengan skenario sebelumnya. Nilai kemiripan relatif lebih rendah dibandingkan dengan skenario sebelumnya. Nilai kemiripan yang cukup rendah ini dapat digunakan untuk membedakan jenis virus. Hal ini sesuai harapan, namun ada beberapa jenis virus yang ternyata memiliki nilai yang cukup tinggi juga.



Gambar 14. Grafik Perbandingan Tiga Skenario Hasil Metode Tores.

Berdasarkan hasil percobaan juga terlihat bahwa panjang rantai DNA dapat berbeda cukup besar bila virus berbeda.