# RANCANG BANGUN APLIKASI PENCOCOKAN DNA MANUSIA MENGGUNAKAN ALGORITMA *LEVENSHTEIN DISTANCE* (Studi Kasus: DNA Kanker Hati Manusia)

Zulmi Afriansyah[1], Diyah Puspitaningrum[2], Ernawati[3]

[1,2,3]Program Studi Teknik Infomatika, Fakultas Teknik, Universitas Bengkulu.
Jl. WR. Supratman Kandang Limun Bengkulu 38371A INDONESIA
(telp: 0736-341022; fax: 0736-341022)

[1]zulmi.afriansyah@outlook.com,
[2]diyahpuspitaningrum@gmail.com,
[3]w_ier_na@yahoo.com

*Abstrak*. Pencocokan *string* adalah suatu masalah yang hampir terjadi di seluruh bidang kehidupan. Pencocokan *string* bisa digunakan dalam penelitian tentang pencocokan sekuen DNA. DNA merupakan materi genetik yang terdapat dalam tubuh setiap orang yang diwarisi dari orang tua. Dari DNA yang bermutasi kita bisa mengetahui penyakit yang dialami oleh seseorang. Di Indonesia, kanker hati termasuk jenis penyakit yang banyak diderita. Kanker hati memiliki 4 jenis yaitu *Hepatocellular carcinoma (HCC), Cholangiocarcinoma, Hepatoblastoma* dan *Angiosarcoma* dan *Hemangiosarcoma*. Pada penelitian ini, digunakanlah suatu algoritma yang dapat menghitung jarak perbedaan antara sekuen-sekuen DNA kanker hati, yaitu algoritma *Levenshtein Distance*. Metode pengembangan sistem yang digunakan untuk membangun aplikasi ini adalah *model waterfall*. Sedangkan pada tahap analisa dan perancangan sistem dilakukan dengan pendekatan berorientasi objek menggunakan UML. Setelah mengetahui nilai jarak kita bisa mengetahui suatu DNA masuk ke dalam kelas mana, metode klasifikasi yang digunakan yaitu *k-fold cross validation*.

Kata Kunci : Pencocokan DNA, *Levensthein Distance*, *k-fold Cross Validation*

***Abstract:*** **String matching is a problem that almost occurred in all areas of life. String matching can be used in research on DNA sequence matching. DNA is the genetic material contained in every human being inherited from a parent. From mutated DNA we can find disease in a person's body. In Indonesia, the liver cancer is included that affects many types of diseases. Liver cancer has four types of Hepatocellular carcinoma (HCC), Cholangiocarcinoma, Hepatoblastoma and angiosarcoma and hemangiosarcoma. In this research, an algorithm is used to calculate the distance differences between DNA sequences of liver cancer, the Levenshtein Distance algorithm. System development method used to build this application is the waterfall model. Whereas at this stage of the analysis and design of the system is done with the object oriented approach using UML. After knowing the distance we can know the value of a DNA into**

**the class where the method of classification used is the k-fold cross validation.**
**Keywords: DNA Matching, Levensthein Distance, k-fold Cross Validation**

## I. PENDAHULUAN

*String matching* bisa digunakan dalam penelitian tentang pencocokan DNA. Permasalahan pencocokan *string* ini akan coba diterapkan pada *sequence* DNA (rangkaian DNA) dengan jenis DNA kenker hati untuk mengetahui berapa jarak perbedaan antara sekuen-sekuen DNA dalam pengklasifikasian DNA dengan menggunakan algoritma *string matching*, yaitu algoritma *Levenshtein Distance*. DNA (*Deoxyribonucleic Acid)* atau juga dapat diartikan dalam Bahasa Indonesia yaitu ADN (asam deoksiribonukleat). DNA atau ADN ini merupakan materi genetik yang terdapat dalam tubuh setiap orang yang diwarisi dari orang tua. DNA terdapat pada inti sel di dalam struktur kromosom dan pada mitokondria (organel terbesar kedua dalam sel). DNA dapat bermutasi dan dampak dari adanya mutasi DNA salah satunya dapat menyebabkan kanker pada manusia. Penyebab kanker biasanya tidak dapat diketahui secara pasti, karena merupakan gabungan dari sekumpulan faktor, genetik dan lingkungan.

## II. LANDASAN TEORI

### A. Kanker Hati

Kanker hati terjadi ketika sel DNA hati mengalami mutasi. Mutasi ini membuat sel tetap tumbuh dan berkembang, sementara sel normal lain memiliki siklus hidup dan mati. Kanker hati primer yang berasal dari sel hati terbagi dalam beberapa tipe, antara lain [2] :

1. *Hepatocellular carcinoma (HCC)*. Kanker hati yang paling umum terjadi pada anak-anak dan orang dewasa. Kanker ini dimulai dari *hepatocytes* yang merupakan tipe utama sel hati. HCC dapat memiliki pola pertumbuhan yang berbeda.
   a. Beberapa diawali sebagai sebuah tumor tunggal yang tumbuh lebih besar. Penyakit ini tidak menyebar ke bagian lain dari hati.
   b. Beberapa lainnya dimulai di titik-titik di seluruh hati, bukan sebagai tumor tunggal. Hal ini paling sering terlihat pada orang dengan kerusakan hati yang sedang berlangsung (sirosis) dan merupakan pola yang paling umum terlihat di Amerika Serikat
2. *Cholangiocarcinoma*. Kanker ini berasal dari saluran kantung empedu. Ada kanker langka yang dimulai pada pembuluh darah hati. Kanker ini tumbuh dengan cepat. Seringkali pada saat mereka ditemukan mereka terlalu luas untuk dihapus. Pengobatan dapat membantu memperlambat penyakit, tetapi kanker ini biasanya sangat sulit diobati.
3. *Hepatoblastoma*. Ini adalah tipe kanker langka yang menyerang anak-anak berusia 4 tahun ke bawah. Tipe kanker ini banyak yang berhasil disembuhkan. Sekitar 70% anak dengan penyakit ini memiliki hasil yang baik dengan operasi dan kemoterapi. Tingkat kelangsungan hidup lebih besar dari 90% untuk tahap awal penyakit.
4. *Angiosarcoma* dan *hemangiosarcoma*. Tipe kanker langka ini dimulai di pembuluh darah di hati dan tumbuh dengan sangat cepat. Sebagian besar ketika kanker ditemukan di hati tidak mulai dari sana, tetapi mulai di tempat lain (seperti, payudara, usus, atau paru-paru) dan menyebar ke hati. Ini disebut ***kanker metastatik***. Meskipun sel-sel kanker di hati,

mereka masih terlihat dan bertindak seperti sel-sel kanker dari bagian tubuh yang mereka berasal. Jika seseorang memiliki kanker paru-paru yang telah menyebar ke hati, sel-sel kanker di hati masih sel kanker paru-paru, sehingga orang tersebut akan dirawat karena kanker paru-paru metastatic [2].

B. K-fold Cross Validation

*Cross validation* digunakan dalam rangka menemukan parameter terbaik dari satu model. Ini dilakukan dengan cara menguji besarnya *error* pada *data testing*. Dalam *cross validation*, *data* dibagi ke dalam *k* sampel dengan ukuran yang sama. Dari *k* subset *data* yang digunakan akan dipakai $k-1$ sampel sebagai *data training* dan 1 sampel sisanya untuk *data testing*. Salah satu metode *cross-validation* yang populer adalah *K-Fold Cross Validation*. Dalam teknik ini dataset dibagi menjadi sejumlah K-buah partisi secara acak [4].

C. Levenshtein Distance

*Levenshtein Distance* dibuat oleh Vladimir Levenshtein pada tahun 1965. Perhitungan *edit distance* didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan *string* antara dua *string*. Perhitungan jarak antara dua *string* ini ditentukan dari jumlah minimum operasi perubahan untuk membuat *string* A menjadi *string* B. Ada 3 macam operasi utama yang dapat dilakukan oleh algoritma ini yaitu [1] :

1. Operasi Pengubahan Karakter
   Operasi pengubahan karakter merupakan operasi menukar sebuah karakter dengan karakter lain contohnya penulis menuliskan *string* ‘yamg‟ menjadi “yang‟. Dalam kasus ini karakter “m‟ diganti dengan huruf “n‟.
2. Operasi Penambahan Karakter
   Operasi penambahan karakter berarti menambahkan karakter ke dalam suatu *string*. Contohnya *string* “kepad‟ menjadi *string* “kepada‟, dilakukan penambahan karakter “a‟ di akhir *string*. Penambahan karakter tidak hanya dilakukan di akhir kata, namun bisa ditambahkan diawal maupun disisipkan di tengah *string.*
3. Operasi Penghapusan Karakter
   Operasi penghapusan karakter dilakukan untuk menghilangkan karakter dari suatu *string*. Contohnya *string* “barur‟ karakter terakhir dihilangkan sehingga menjadi *string* “baru‟. Pada operasi ini dilakukan penghapusan karakter “r‟.

$$\text{lev a,b(i,j)} = \begin{cases} \max(i,j) & if\ \min(i,j)=0 \\ \min\begin{cases} lev\ a,b(i-1,j)+1 \\ lev\ a,b(i,j-1)+1 \\ lev\ a,b(i-1,j-1)+1(ai \neq bj) \end{cases} & otherwise \end{cases} \quad (1)$$

Keterangan :

lev a,b adalah matriks lev a,b

i adalah baris matriks

j adalah kolom matriks

Sedangkan untuk menghitung nilai

$$Similarity = \{1 - \frac{edit\ distance}{maxLengt\ h(str1,str2)}\} \quad (2)$$

Keterangan :

***edit distance*** adalah hasil dari perbandingan yang telah dilakukan atau *Levenshtein distance*.

***maxLength*** adalah jumlah *string* dari kata yang terpanjang antara str1 dan str2

***Str1*** adalah panjang *string* pertama

***Str2*** adalah panjang *string* kedua

Untuk menghitung persentase adalah :

Persentase : *similarit*y x 100% (3)

Keterangan :

**Similarity** adalah nilai kesamaan antara kedua sekuen

## III. METODOLOGI

### A. Jenis dan Sumber Data

Jenis data yang digunakan dalam penelitian ini adalah data *primer* yaitu berupa 20 sekuen DNA yang terdiri dari 4 kelas dalam jenis penyakit kanker hati, disetiap kelas diambil masing-masing 5 sampel dengan panjang berbeda-beda antara 500 sampai dengan 800 karakter. Data diambil dari *website* NCBI yaitu salah satu *bank* data gen, protein dan *literature* khususnya di bidang kesehatan yang terlengkap dan di acu oleh para peneliti didunia.

### B. Metode Pengembangan Sistem

Sistem yang dikembangkan dalam penelitian ini menggunakan model pengembangan sistem *sekuensial linier* yang bersifat sistematis dan berurutan. Adapun penjelasan tahap-tahap model *sekuensial linier* dalam penelitian ini adalah sebagai berikut:

1. Analisis kebutuhan sistem

   Pada tahap ini peneliti akan melakukan analisis dan definisi kebutuhan sistem dengan teknik pengumpulan data menggunakan teknik studi pustaka yang bersumber dari literatur berupa buku-buku, laporan penelitian, karangan-karangan ilmiah dan lain sebagainya mengenai hal-hal yang dibutuhkan dan mendukung proses pembuatan aplikasi tes buta warna. Setelah itu, dilakukan analisis sistem yang akan dibangun. Hasil analisis ini akan dimodelkan yaitu dengan membuat UML.
2. Desain Sistem

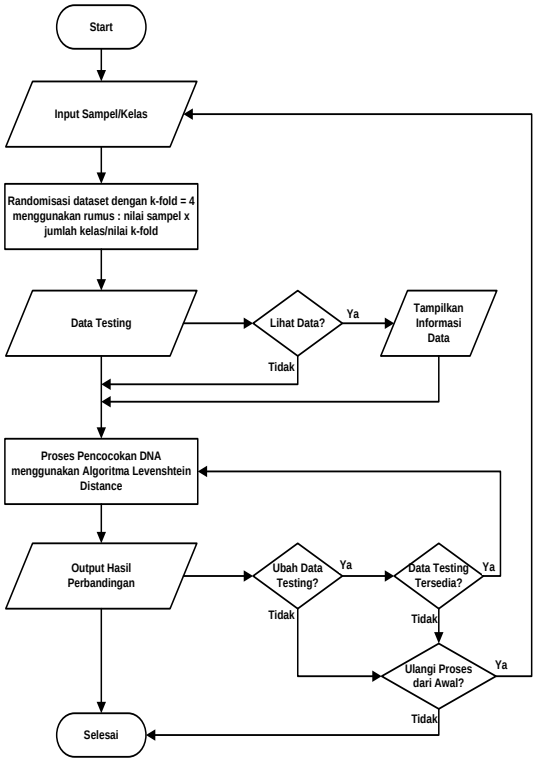
   Perancangan sistem dikerjakan setelah tahap analisis dan definisi kebutuhan selesai dikumpulkan secara lengkap. Kegiatan yang dilakukan di tahap ini adalah menerjemahkan analisis ke dalam bentuk rancangan antarmuka (*interface*), dan rancangan prosedur metode sebelum penulisan program (*coding*).
3. Generasi Kode

   Hasil perancangan sistem akan diubah menjadi bentuk yang dimengerti oleh mesin yaitu ke dalam bahasa pemrograman yang telah ditentukan melalui proses penulisan program (*coding*). Dalam penelitian ini, digunakan Netbeans 8.0.
4. Integrasi dan Pengujian Sistem

   Sistem yang sudah dibangun akan dilakukan pengujian untuk melihat apakah sistem tersebut sesuai dengan perencanaan dan perancangan. Pada penelitian ini akan dilakukan dengan menggunakan *Black-Box* dan *White-Box* sebagai metode pengujian sistem.
5. Operasi dan Pemeliharaan

   Tahap ini adalah tahap akhir pengembangan dan implementasi sistem yaitu pengoperasian sistem secara nyata. Namun dalam pengoperasiannya tetap dibutuhkan dukungan agar sistem dapat digunakan dalam jangka panjang dengan melakukan pemeliharaan sistem. Pemeliharaan sistem dilakukan bukan hanya sekedar proses memperbaiki kesalahan program tetapi proses yang memiliki karakteristik memperbaiki kesalahan yang tidak ditemukan pada tahapan sebelumnya atau menambahkan fungsi baru yang belum ada pada program tersebut.

## IV. ANALISIS DAN PERANCANGAN




Gambar 1 Diagram alir aplikasi pencocokan DNA kanker hati manusia

Berdasarkan gambar 1 diatas, maka aliran program akan berjalan dari awal hingga akhir yaitu *user* menekan tombol *input* jumlah sampel DNA yang digunakan untuk mencari DNA yang akan dibandingkan, sistem akan mengambil sejumlah *data* secara *random* sesuai dengan nilai *input*, dan *data* akan dibagi menjadi 4 *data set* sesuai dengan nilai k-fold (k=4) yang sudah ditentukan, setelah *data set* telah didapat, maka sistem menentukan *data testing* dan *data training* dari semua data set yang telah didapat, kemudian sistem akan melakukan perbandingan DNA yang di-*input* dengan DNA sampel yang ada pada *database* sistem.

Selama perhitungan berjalan, *user* bisa melihat *data* yang sedang diproses dengan menekan tombol Lihat Data, dan sistem akan menampilkan semua *data* yang sedang diolah. Setelah melakukan perbandingan, maka user bisa melihat hasil perbandingan yang sistem akan tampilkan, selanjutnya *user* bisa melihat tabel perbandingan DNA dengan menekan tombol tampilkan tabel, melakukan perhitungan selanjutnya dengan mengubah *data testing* atau bisa langsung menutup aplikasi, jika *user* ingin melakukan perhitungan selanjutnya, maka user harus mengganti *data testing* terlebih dahulu, apabila data testing masih tersedia, maka system bisa melakukan kembali proses perbandingan seperti sebelumnya, dan apabila semua perhitungan sudah dilakukan, *user* bisa melakukan perhitungan ulang dengan menekan tombol Ulang Proses dan sistem akan menampilkan *form* utama kembali seperti awal.

## V. PEMBAHASAN

Perhitungan Manual k-Fold Cross Validation dan Levenshtein Distance

### *A. K-Fold Cross Validation*

Dalam pendekatan *cross-validation*, setiap *record* digunakan beberapa kali dalam jumlah yang sama untuk *training* dan tepat sekali untuk *testing.* Untuk mengilustrasikan metode ini, kita mempartisi *data* ke dalam empat *subset* yang berukuran sama. Pertama, kita pilih satu dari keempat *subset* tersebut untuk *testing* dan sisa tiga *subset* untuk *training*. Kemudian dilakukan pertukaran fungsi dari *subset* sedemikian sehingga *subset* yang sebelumnya sebagai *training set* menjadi *test set* demikian sebaliknya. Pendekatan ini dinamakan *k-fold cross-validation*. Dalam contoh ini, setiap *record* digunakan tiga kali untuk *training* dan satu kali untuk *testing*. Metode *k-fold cross-validation* men-generalisasi pendekatan ini dengan mensegmentasi *data* ke dalam k partisi berukuran sama. Selama proses, salah satu dari

partisi dipilih untuk *testing*, sedangkan sisanya digunakan untuk *training*. Prosedur ini diulangi k kali sedemikian sehingga setiap partisi digunakan untuk *testing* tepat satu kali. Kasus khusus untuk metode *k-fold cross-validation* menetapkan k = N, yaitu ukuran dari *data set*. Pendekatan ini untuk mengulangi prosedur sebanyak N kali. Biasanya diterapkan jika sampel data berukuran terbatas (sedikit). Pada percobaab ini ditetapkan nilai k=4 dengan alasan jumlah sekuen ada 20 (kelipatan 4). Kita melakukan 4-*Fold Cross-Validation* maka desain data eksperimennya pada Tabel 1 :

Tabel 1. Pembagian *data training* dan *data testing*

| Perhitungan ke | *Data training* | *Data testing* |
|---|---|---|
| 1 | B, C, D | A |
| 2 | A, C, D | B |
| 3 | A, B, D | C |
| 4 | A, B, C | D |

*B. Levenshtein Distance*

Dalam algoritma ini, dilakukan penyeleksian panjang kedua *string* terlebih dahulu. Jika salah satu atau kedua *string* merupakan *string* kosong, jalannya algoritma ini berhenti dan memberikan hasil *edit distance* yang bernilai nol atau panjang *string* yang tidak kosong. Apabila kita melakukan perbandingan 2 string dengan panjang yang tidak bernilai 0, maka kita bisa melihat hasil perbandingan pada Tabel 2 berikut ini :

Tabel 2. Hasil perbandingan menggunakan Levenstein Distance

| | | T | G | G | A | C | C | T |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| A | 1 | 1 | 2 | 3 | 3 | 4 | 5 | 6 |
| T | 2 | 1 | 2 | 3 | 4 | 4 | 5 | 5 |
| G | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 |
| C | 4 | 3 | 2 | 2 | 3 | 3 | 4 | 5 |
| T | 5 | 4 | 3 | 3 | 3 | 4 | 4 | 4 |
| G | 6 | 5 | 4 | 3 | 4 | 4 | 5 | 5 |
| C | 7 | 6 | 5 | 4 | 4 | 4 | 4 | 5 |
| G | 8 | 7 | 6 | 5 | 5 | 5 | 5 | 5 |
| G | 9 | 8 | 7 | 6 | 6 | 6 | 6 | 6 |
| G | 10 | 9 | 8 | 7 | 7 | 7 | 7 | 7 |

Berdasarkan Tabel 2 kita telah melakukan perbandingan antara sekuen “ATGCTGCGGG” yang memiliki panjang 10 dengan sekuen “TGGACCT” yang memiliki panjang 7. Dari hasil perbandingan maka didapatlah nilai jarak perbedaan 7 yang bisa kita ambil dari nilai matriks pada ujung kanan bawah matriks berwarna merah.

Jika panjang *string* keduanya tidak nol, berarti setiap sekuen memiliki sebuah karakter, karakter pertama (c1) pada sekuen pertama yaitu karakter “A“ dan karakter kedua (c2) pada sekuen kedua yaitu karakter “T”. Setiap karakter memiliki nilai masing-masing, misalnya nilai karakter pertama (c1) pada sekuen pertama adalah 1 dan begitu juga nilai karakter kedua (c2) pada sekuen kedua yaitu 1, kemudian dapat dikatakan penghitungan yang dilakukan adalah cara mentransformasikan c1 menjadi c2. Jika c1 sama dengan c2, maka dapat diberikan nilai cost 0, sedangkan jika c1 berbeda dengan c2, maka nilai *cost*-nya 1 karena membutuhkan 1x operasi perubahan dari c1 menjadi c2. Akibatnya, nilai *edit distance*-nya dari pentransformasian sekuen pertama menjadi sekuen kedua ditambah 1, pada perbandingan karakter selanjutnya, nilai *cost* pertama mempengaruhi nilai *cost* selanjutnya. Begitu juga seterusnya hingga karakter terakhir pada sekuen pertama dibandingkan dengan karakter terakhir pada sekuen kedua. Dari semua perbandingan, didapatlah nilai akhir atau nilai *edit distance*-nya dari pentransformasian sekuen pertama menjadi sekuen kedua. Dari kemungkinan-kemungkinan tersebut, dicarilah nilai minimal yang dikenal sebagai nilai *edit distance*.

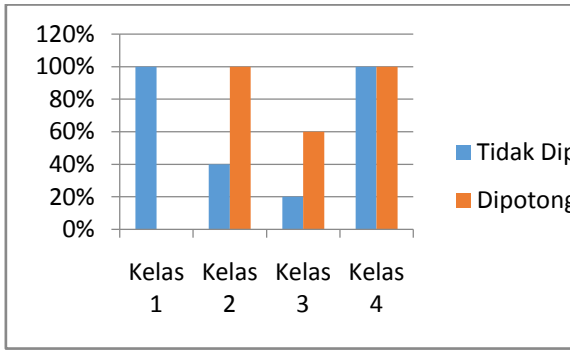
Percobaan dilakukan sebanyak 2 kali pada 20 sekuen DNA dari 4 kelas jenis penyakit kanker hati, percobaan pertama dilakukan dengan panjang sekuen beragam antara 500-800 karakter, dan percobaan kedua dilakukan dengan memotong sekuen DNA menjadi 500 karakter. Setelah kedua

percobaan perbandingan sekuen DNA dilakukan, maka dihasilkan tabel sebagai berikut :

Tabel 3. Hasil persentase kecocokan dengan memotong dan tanpa memotong 20 sekuen

| No | Kelas | Persentase kecocokan | |
|---|---|---|---|
| | | Tidak Dipotong | Dipotong |
| 1 | Kelas 1 (Hepatocellular Carcinoma) | 100 % | 0 % |
| 2 | Kelas 2 (Cholangio Carcinoma) | 40 % | 100 % |
| 3 | Kelas 3 (Hepatoblastoma) | 20 % | 60 % |
| 4 | Kelas 4 (Angiosarcoma) | 100 % | 100 % |

Apabila perbandingan kedua tabel tersebut dimasukkan kedalam grafik maka akan menghasilkan grafik sebagai berikut :




Gambar 2. Grafik persentase kebenaran kelas sebelum dan sesudah pemotongan sekuen

Dari Gambar 2 bisa kita lihat kenaikan dan penurunan persentase kebenaran setiap kelas sebelum dan setelah sekuen DNA mengalami pemotongan, dari grafik kelas 1 mengalami penurunan dari 100% menjadi 0%, ini bisa disimpulkan bahwa bagian sekuen DNA yang dipotong merupakan *feature* pada sekuen-sekuen yang berada pada kelas 1, sedangkan kelas 2 dan kelas 3 mengalami peningkatan dari 40% menjadi 100% dan 20% menjadi 60%, maka pada kedua kelas ini disimpulkan bahwa pada sekuen-sekuen dikelas ini mengalami pemotongan pada daerah yang bermutasi atau banyak mengalami penyisipan/penghapusan karakter DNA.

## VI. KESIMPULAN

1. Aplikasi pencocokan DNA manusia menggunakan algoritma *Levenshtein Distance* telah berhasil dibangun dengan menggunakan model perancangan sistem *UML* dan menggunakan bahasa pemrograman *Java* yang dijalankan di *desktop*.
2. Aplikasi yang dibangun dapat melakukan proses pencocokan DNA di komputer dengan menggunakan algoritma *Levenshtein Distance* dan melakukan klasifikasi dengan metode *k-fold Cross Validation.*
3. Kelemahan algoritma tidak bisa menemukan akurasi klasifikasi dengan nilai tinggi karena algoritma ini mengizikan penyisipan atau mutasi DNA tanpa ambang batas sehingga terlalu mudah ditoleransi. Nilai toleransinya yang terlalu besar akan mengakibatkan DNA yang semula di dalam kelas yang sama dapat saja dengan mutasi, misalnya menjadi lebih dekat dengan kelas lain.

## REFERENSI

[1] Adriyani, N. M., Santiyasa, I. W., & Muliantara, A. (2012). *Implementasi Algoritma Levenshtein Distance dan Metode Empiris untuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Dokumen Berbahasa Indonesia* (Skripsi). *Universitas Udayana*.

[2] Harjana, Dadan. (2013). *Gejala Kanker Hati, Penyebab dan Cara Pencegahan*. Dipetik Januari 7, 2015 dari http://gejalapenyakitmu.blogspot.com/2013 /08/gejala-kanker-hati-penyebab-dan-cara.html

[3] Raharjo. S,. Winarko. E. (2014). *Klasterisasi, Klasifikasi dan Peringkasan Teks Berbahasa Indonesia*. Yogyakarta: Fakultas Teknologi Industri, Institut Sains & Teknologi AKPRIND Universitas Gadjah Mada.