BAB III

PENGEMBANGAN APLIKASI BERBASIS *WEB* UNTUK MEMBANGUN *PHYLOGENETIC TREE*

3.1 Pendahuluan

Seperti yang telah dijelaskan pada Bab I sebelumnya bahwa tujuan dari penulisan tugas akhir ini adalah membangun sebuah aplikasi berbasis *web* untuk membangun sebuah *phylogenetic tree* dari sebuah matriks jarak dengan bentuk umum sebagai berikut.

$$\begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1j} & \cdots & d_{1N} \\ d_{21} & d_{22} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ d_{i1} & \ddots & \ddots & d_{ij} & \ddots & d_{iN} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ d_{N1} & d_{N2} & \cdots & d_{Nj} & \cdots & d_{NN} \end{pmatrix}$$

$d_{ij} = 0,$ jika $i = j$ dan $d_{ij} > 0,$ jika $i \neq j$, untuk setiap $i = 1, 2, ..., N$ dan $j = 1, 2, ..., N$ dengan N adalah jumlah OTU yang terlibat.

Algoritma yang digunakan untuk membangun *phylogenetic tree* dari matriks jarak ini adalah *neighbor-joining*. Untuk menerapkan algoritma ini ke dalam aplikasi berbasis *web*, algoritma ini harus diterjemahkan terlebih dahulu ke dalam bahasa pemrograman yang digunakan untuk membuat aplikasi *web*, salah satunya yaitu bahasa pemrograman PHP.

Aplikasi yang dibuat pada tugas akhir ini memerlukan *input* berupa matriks jarak, matriks jarak ini akan diproses dengan melalui beberapa tahapan sehingga menghasilkan *output* berupa gambar *phylogenetic tree*. Tahapan-tahapan dan bagian-bagian apa saja yang ada pada aplikasi ini akan diperlihatkan oleh diagram *site map* berikut.

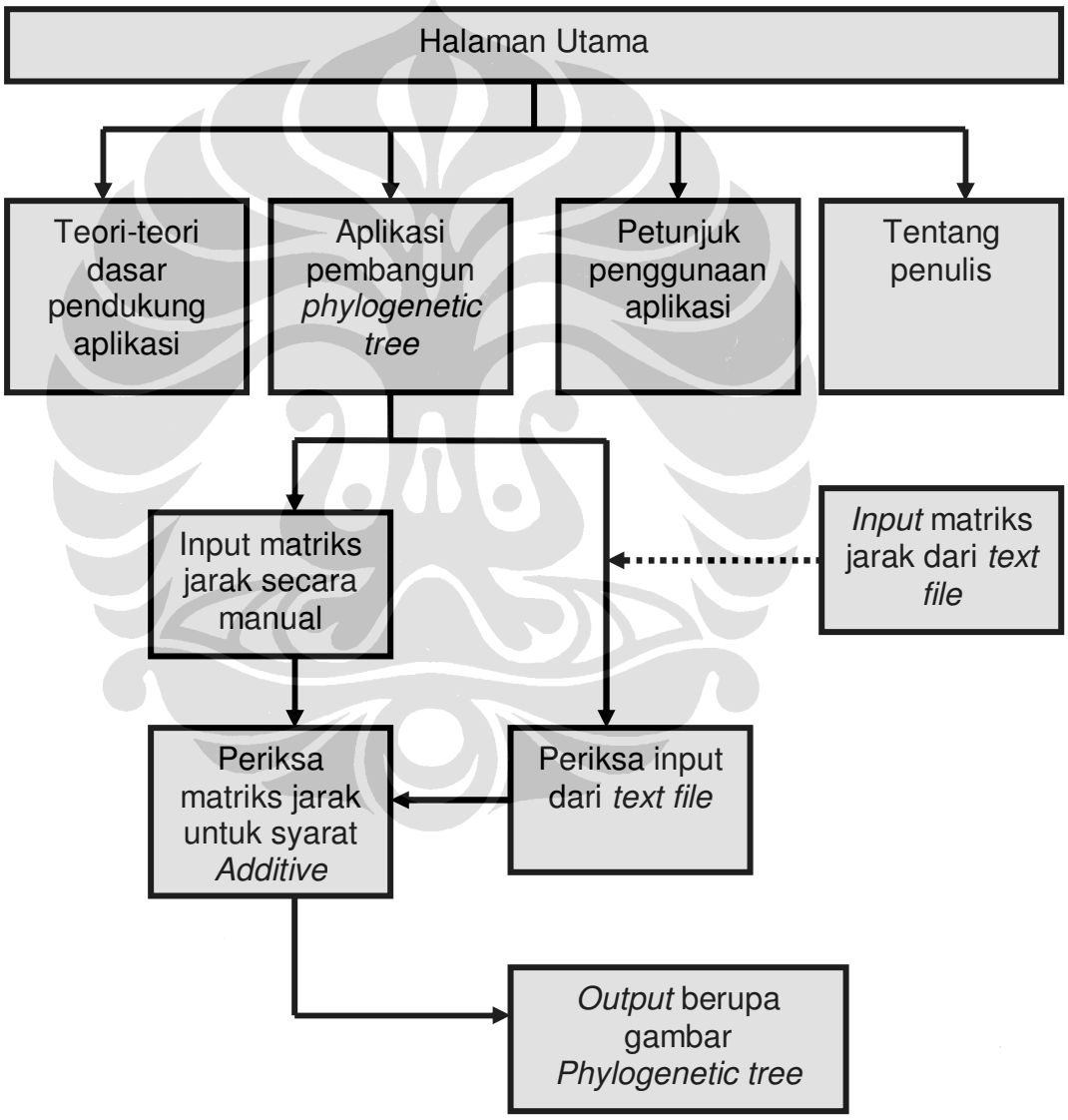


Diagram 2. *Site Map* aplikasi berbasis *web*

3.2 Pengembangan Aplikasi

Ilmu bioinformatika tidak mungkin dapat berkembang pesat tanpa adanya dukungan dari *software* atau aplikasi. Aplikasi yang terkait dengan ilmu tersebut sangat bervariasi, baik sifat maupun fungsinya. Aplikasi yang akan dibuat pada tugas akhir ini sifatnya *open source*, artinya pengguna dapat memakai aplikasi ini secara bebas dan gratis di manapun dan kapanpun, tanpa perlu menginstalnya ke komputer pribadi, dengan syarat komputer tersebut harus terhubung oleh sebuah jaringan baik *intranet* maupun *internet*.

Untuk membuat aplikasi dengan sifat seperti ini berbeda dengan aplikasi yang biasa. Selain memerlukan *software* pendukung, komputer juga harus terhubung ke sebuah komputer *server* pada jaringan. Komputer *server* adalah komputer yang menjadi pusat di suatu jaringan, fungsinya adalah melayani komputer-komputer pada jaringan tersebut. Namun, jika kita tidak memiliki jaringan dengan sebuah *server* bukan berarti kita tidak dapat membuat aplikasi seperti ini. Kita masih bisa membuat aplikasi ini dengan menggunakan *server* lokal, artinya kita harus menjadikan komputer pribadi kita sebagai *server* tanpa perlu membuat jaringan. *Software* yang dapat melakukan hal ini, antara lain *phpmyadmin, wamp*, dan *xampp*.

Software yang digunakan untuk membuat aplikasi pada tugas akhir ini adalah *wamp*. Bahasa pemrograman yang dipakai adalah PHP, PHP adalah bahasa pemrograman umum yang sering dipakai dalam membuat aplikasi berbasis *web*. Karena fungsi inilah bahasa pemrograman PHP hanya dapat

dieksekusi oleh komputer yang menjadi *server* atau yang terhubung ke *server*. Sebenarnya untuk membuat aplikasi dengan bahasa PHP ini, kita bisa menggunakan aplikasi **notepad** sebagai media penulisan *source code*. Namun, untuk mempermudah penulis memakai *software macromedia dreamweaver* sebagai *software* pendukung.

Pada tugas akhir ini, tujuan akhir dari aplikasi ini adalah membuat gambar *phylogenetic tree*. Untuk menggambar *tree* tersebut, penulis memakai paket yang tersedia pada PHP, yaitu gd *library* yang terdapat pada *file* **gd2.dll**. Umumnya paket ini sudah diaktifkan dan langsung dapat digunakan. Namun jika belum diaktifkan, maka harus diaktifkan terlebih dahulu. Caranya buka *file* **php.ini** kemudian hilangkan tanda ";" yang terletak di depan nama file **gd2.dll**. Simpan *file* **php.ini**, kemudian *restart server*.

Paket tersebut berisi perintah-perintah untuk menggambar pada *web*. Beberapa perintah yang ada pada paket ini dan digunakan pada aplikasi ini tertera pada tabel berikut ini.

Perintah	Fungsinya
imagecreate	Membuat gambar/image baru
imagecolorallocate	Memberi warna latar belakang gambar/image
imagestring	Memberi teks pada gambar/image
imagefilledellipse	Membuat sebuah elips pada gambar/image, elips dapat diisi oleh warna
imageline	Membuat garis pada gambar/image
imagefilledrectangle	Membuat sebuah segi empat pada gambar/image, segi empat dapat diisi oleh warna

imagefilledpolygon	Membuat sebuah poligon pada gambar/image, poligon dapat diisi oleh warna
imagepolygon	Membuat gambar/image berbentuk poligon
imagepng	Membentuk gambar/image dengan format PNG
imagedestroy	Menghapus gambar/image yang tersimpan pada memori

Tabel 1. Fungsi-fungsi pada PHP untuk menggambar [9]

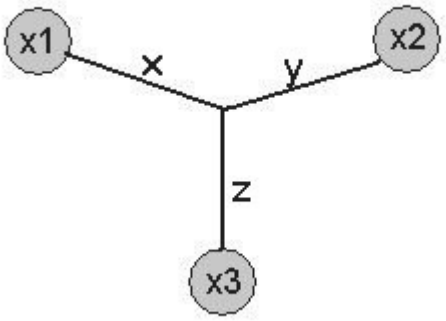
3.3 Algoritma *Neighbor-Joining*

Pada bagian sebelumnya telah diungkapkan bahwa pada penulisan tugas akhir ini, penulis memakai algoritma *neighbor-joining* dalam membangun suatu *phylogenetic tree*. Sebelumnya juga sudah sedikit dibahas mengenai algoritma *neighbor-joining*, dan pada bagian ini akan dibahas lebih lanjut mengenai algoritma *neighbor-joining*.

Untuk membangun suatu *phylogenetic tree* dengan menggunakan algoritma ini, diperlukan *input* berupa matriks jarak seperti pada Definisi 1.7. Sebelum membahas algoritma ini secara umum, perhatikan kasus khusus berikut. Asumsikan jumlah OTU yang dilibatkan adalah 3 (tiga) atau $N = 3$. Pada kasus ini, akan dicari tiga buah bilangan positif x, y, z sehingga memenuhi

$$\begin{aligned} x + y &= d_{12} \\ x + z &= d_{13} \\ y + z &= d_{23} \end{aligned} \tag{3.1}$$

dimana hanya ada satu *tree* yang bersesuaian, diperlihatkan pada Gambar 9.



Gambar 9. *Tree* untuk 3 spesies/OTU

dengan melakukan eleminasi dan subtitusi pada sistem persamaan (3.1), diperoleh solusi dari sistem persamaan tersebut, yaitu:

$$x = \frac{1}{2}\left(d_{12} + d_{13} - d_{23}\right)$$
$$y = \frac{1}{2}\left(d_{12} + d_{23} - d_{13}\right) \qquad (3.2)$$
$$z = \frac{1}{2}\left(d_{13} + d_{23} - d_{12}\right)$$

dari solusi tersebut akan diperoleh nilai x, y, z yang merepresentasikan panjang *edge* pada Gambar 9. Sehingga untuk kasus *N* = 3, tree yang akan terbentuk seperti Gambar 9 dengan panjang *edge* diperoleh dari persamaan (3.2). [1]

Sekarang akan dibahas algoritma *neighbor-joining* secara detail untuk jumlah spesies lebih dari 3 (tiga) atau $N > 3$. Misalkan diberikan sebuah himpunan A = $\{x1, x2, ..., xN\}$ yang merupakan himpunan terhingga

sekuens (OTU) yang akan dibentuk *phylogenetic tree*-nya, maka untuk setiap $i = 1, 2, 3, ..., N$, didefinisikan

$$r_i = \frac{1}{N-2} \sum_{k=1}^{N} d_{ik} \tag{3.3}$$

Selanjutnya, untuk setiap $i, j = 1, 2, 3, ..., N, \quad i < j$, didefinisikan

$$D_{ij} = d_{ij} - (r_i + r_j) \tag{3.4}$$

dan sangat tepat jika D_{ij} akan berbentuk matriks segitiga atas. Sekarang, ambil sepasang $1 \le i, j \le N$, dengan D_{ij} minimal. Selanjutnya pasangan OTU xi dan xj tersebut akan digabung dan akan digantikan dengan elemen tunggal yang baru yaitu OTU $x(N+1)$. OTU yang baru $x(N+1)$ merepresentasikan *internal node* yang nantinya akan dihubungkan oleh *edge* ke OTU xi dan OTU xj . Panjang masing-masing *edge* atau jaraknya ditentukan dengan rumus

$$\begin{aligned} d_{N+1\,i} &= \frac{1}{2}\left(d_{ij} + r_i - r_j\right) \\ d_{N+1\,j} &= \frac{1}{2}\left(d_{ij} + r_j - r_i\right) \end{aligned} \tag{3.5}$$

Sekarang akan didefinisikan jarak antara OTU $x(N+1)$ ke setiap OTU xm dengan $m \neq i, j$, sebagai berikut

$$d_{N+1\,m} = \frac{1}{2}\left(d_{im} + d_{jm} - d_{ij}\right) \tag{3.6}$$

Hingga tahap ini, akan diperoleh himpunan yang baru yaitu A' = $\{xm, x(N+1), m \neq i, j\}$ dengan jumlah sekuen $N-1$ OTU. Prosedur di atas

akan diterapkan kembali pada himpunan yang baru tersebut dan akan terus berulang. Algoritma akan melakukan iterasi sampai jumlah OTU sama dengan 3 (tiga), pada kasus ini hanya ada satu *tree* yang bersesuaian seperti pada kasus $N = 3$ sebelumnya, dengan panjang *edge* ditentukan dengan menggunakan rumus (3.2).

Agar lebih memudahkan, algoritma *neighbor-joining* dirangkum pada ilustrasi berikut.

Input: Matriks jarak $M_d = (d_{ij})$

Inialisasi:

1. Definisikan T sebagai himpunan *node* yang mewakili OTU
2. $L = T$

Iterasi:

1. $N =$ banyak anggota L.
2. $P =$ banyak anggota T.
3. Untuk setiap $i < j$ hitung $D_{ij} = d_{ij} - (r_i + r_j)$, dengan $r_i = \frac{1}{N-2}\sum_{k=1}^{N} d_{ik}$.
4. Ambil pasangan i, j yang meminimalkan D_{ij}.
5. Definisikan *node* baru $k = P + 1$
6. Masukan k ke T.
7. Hubungkan *node* k ke *node* i dan *node* k ke *node* j.
8. Hitung $d_{ki} = \frac{1}{2}(d_{ij} + r_i - r_j)$.
9. Hitung $d_{kj} = \frac{1}{2}(d_{ij} + r_j - r_i)$.
10. Hitung $d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$, $m \neq i, j$.
11. Keluarkan *node* i dan *node* j dari himpunan L dan masukan k.

Terminasi:

Iterasi berhenti ketika himpunan L hanya terdiri dari tiga *node* atau $N =$ 3. Selanjutnya gunakan rumus (3.2)
Hubungkan semua *node* yang terhubung.

Output : *Phylogenetic Tree*

Gambar 10. *Pseudo code* algoritma *neighbor-joining* [1]

3.4 Contoh Penerapan Algoritma *Neighbor-joining*

Misalkan diberikan himpunan $T = \{x1, x2, x3, x4, x5, x6\}$, jumlah OTU atau $N = 6$ dengan sebuah matriks jarak yang bersesuaian sebagai berikut.

Md	$x1$	$x2$	$x3$	$x4$	$x5$	$x6$
$x1$	0	8	3	14	10	12
$x2$	8	0	9	10	6	8
$x3$	3	9	0	15	11	13
$x4$	14	10	15	0	10	8
$x5$	10	6	11	10	0	8
$x6$	12	8	13	8	8	0

Dengan menggunakan persamaan (3.3), maka untuk setiap i diperoleh

$$r_1 = \frac{47}{4}, \quad r_2 = \frac{41}{4}, \quad r_3 = \frac{51}{4}, \quad r_4 = \frac{57}{4}, \quad r_5 = \frac{45}{4}, \quad r_6 = \frac{49}{4}.$$

Kemudian dengan menggunakan persamaan (3.4) untuk $i < j$ diperoleh matriks berikut.

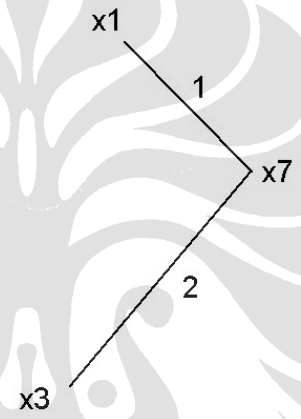
D	$x1$	$x2$	$x3$	$x4$	$x5$	$x6$
$x1$		-14	$-\frac{43}{2}$	-12	-13	-12
$x2$			-14	$-\frac{29}{2}$	$-\frac{31}{2}$	$-\frac{29}{2}$
$x3$				-12	-13	-12
$x4$					$-\frac{31}{2}$	$-\frac{37}{2}$
$x5$						$-\frac{31}{2}$
$x6$						

Pada matriks di atas nilai minimalnya adalah $D_{13} = -\frac{43}{2}$. Sekarang definisikan OTU baru yaitu $x7$ yang akan menggantikan pasangan $x1$ dan

$x3$. Selanjutnya hubungkan $x7$ dengan $x1$ dan $x7$ dengan $x3$, masing-masing memiliki panjang *edge* atau jarak yang ditentukan dari persamaan (3.5) yaitu sebagai berikut.

$$d_{71} = \frac{1}{2}\left(d_{31} + r_1 - r_3\right) = 1$$
$$d_{73} = \frac{1}{2}\left(d_{31} + r_3 - r_1\right) = 2$$

sehingga pada tahap ini tree yang dihasilkan seperti pada gambar berikut.



Gambar 11. Tree tahap 1

Sekarang hitung jarak antara $x7$ dengan setiap $x2, x4, x5, x6$ dengan menggunakan persamaan (3.6), yaitu:

$$d_{72} = \frac{1}{2}\left(d_{12} + d_{32} - d_{13}\right) = 7,$$
$$d_{74} = \frac{1}{2}\left(d_{14} + d_{34} - d_{13}\right) = 13,$$
$$d_{75} = \frac{1}{2}\left(d_{15} + d_{35} - d_{13}\right) = 9,$$
$$d_{76} = \frac{1}{2}\left(d_{16} + d_{36} - d_{13}\right) = 11,$$

sehingga menghasilkan sebuah matriks jarak baru untuk OTU

$x2, x4, x5, x6, x7$ atau $T = \{x2, x4, x5, x6, x7\}$ berikut.

Md	$x2$	$x4$	$x5$	$x6$	$x7$
$x2$	0	10	6	8	7
$x4$	10	0	10	8	13
$x5$	6	10	0	8	9
$x6$	8	8	8	0	11
$x7$	7	13	9	11	0

Untuk matriks yang baru ini, ulangi proses seperti pada tahap sebelumnya, diperoleh.

$$r_2 = \frac{31}{3}, \quad r_4 = \frac{41}{3}, \quad r_5 = 11, \quad r_6 = \frac{35}{3}, \quad r_7 = \frac{40}{3}.$$

yang memberikan matriks

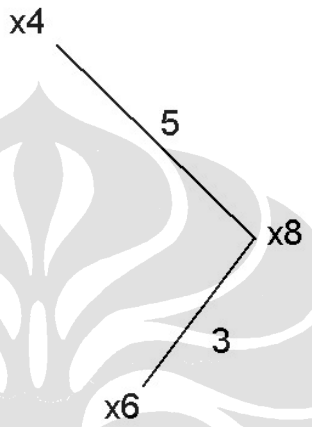
D	$x2$	$x4$	$x5$	$x6$	$x7$
$x2$		-14	$\frac{-46}{3}$	-14	$-\frac{50}{3}$
$x4$			$-\frac{44}{3}$	$-\frac{52}{3}$	-14
$x5$				$-\frac{44}{3}$	$\frac{-46}{3}$
$x6$					-14

karena $D_{46} = -\frac{52}{3}$ minimal, maka pasangan $x4$ dan $x6$ digantikan dengan OTU yang baru yaitu $x8$. Selanjutnya hubungkan $x8$ dengan $x4$ dan $x8$ dengan $x6$, masing-masing memiliki panjang *edge* atau jarak yang ditentukan dari persamaan (3.5) yaitu sebagai berikut.

$$d_{84} = \frac{1}{2}\left(d_{64} + r_4 - r_6\right) = 5$$

$$d_{86} = \frac{1}{2}\left(d_{64} + r_6 - r_4\right) = 3$$

sehingga pada tahap ini tree yang dihasilkan seperti pada gambar berikut.



Gambar 12. Tree tahap 2

Matriks jarak baru untuk OTU $x2, x5, x7, x8$ dengan $T = \{x2, x5, x7, x8\}$ adalah

Md	$x2$	$x5$	$x7$	$x8$
$x2$	0	6	7	5
$x5$	6	0	9	5
$x7$	7	9	0	8
$x8$	5	5	8	0

langkah berikutnya menghasilkan

$$r_2 = 9, \quad r_5 = 10, \quad r_7 = 12, \quad r_8 = 9.$$

dan

D	$x2$	$x5$	$x7$	$x8$
$x2$		-13	-14	-13
$x5$			-13	-14
$x7$				-13

pada matriks di atas ada dua nilai yang minimal yaitu D_{27} dan D_{58}. Pilih salah satunya, yaitu D_{58}. Pasangan $x5$ dan $x8$ digantikan dengan OTU yang baru yaitu $x9$. Selanjutnya hubungkan $x9$ dengan $x5$ dan $x9$ dengan $x8$, masing-masing memiliki panjang *edge* atau jarak yang ditentukan dari persamaan (3.5) yaitu sebagai berikut.

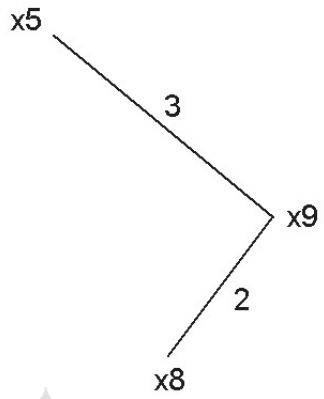
$$d_{95} = \frac{1}{2}\left(d_{85} + r_5 - r_8\right) = 3$$

$$d_{98} = \frac{1}{2}\left(d_{85} + r_8 - r_5\right) = 2$$

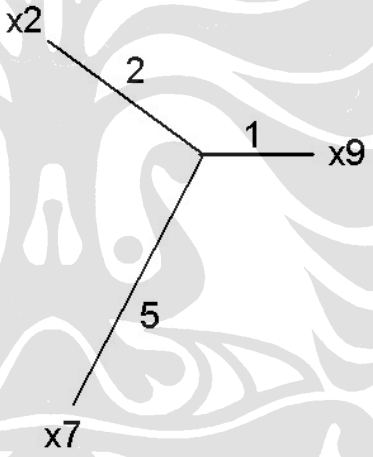
Hitung jarak antara $x9$ dengan $x2$ dan $x9$ dengan $x7$ yang akan menghasilkan *tree* seperti pada Gambar 13 dan memberikan matriks jarak untuk $T = \{x2, x7, x9\}$ berikut.

Md	$x2$	$x7$	$x9$
$x2$	0	7	3
$x7$	7	0	6
$x9$	3	6	0

Pada matriks jarak di atas jumlah N = 3. Untuk kasus ini, tree yang terbentuk seperti Gambar 14 dengan panjang *edge* diperoleh dari persamaan (3.2).

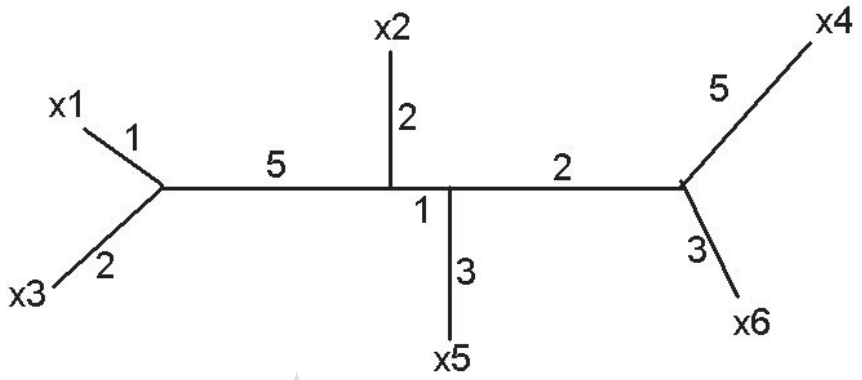


Gambar 13. Tree tahap 3



Gambar 14. Tree tahap 4

Tahap akhir dari algoritma *neighbor-joining* adalah menggabungkan semua *tree* yang diperoleh dari masing-masing tahap. *Tree* yang diperoleh pada contoh ini ditunjukan pada gambar berikut.

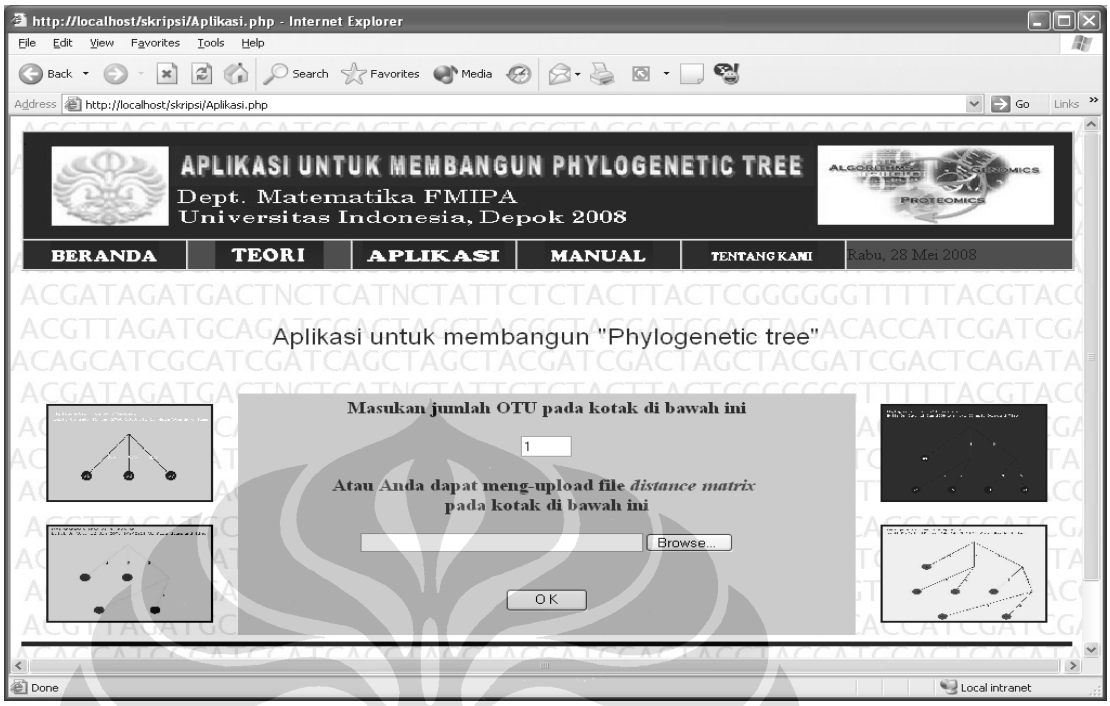


Gambar 15. *Phylogenetic tree* tahap akhir

3.5 Cara Penggunaan dan Alur Aplikasi

Alur aplikasi yang dibuat pada tugas akhir ini, secara umum telah diperlihatkan oleh diagram *site map* pada Diagram 2. Namun, untuk lebih memahami alur aplikasi dan cara penggunaannya penulis akan membahas hal tersebut secara lebih rinci pada bagian ini.

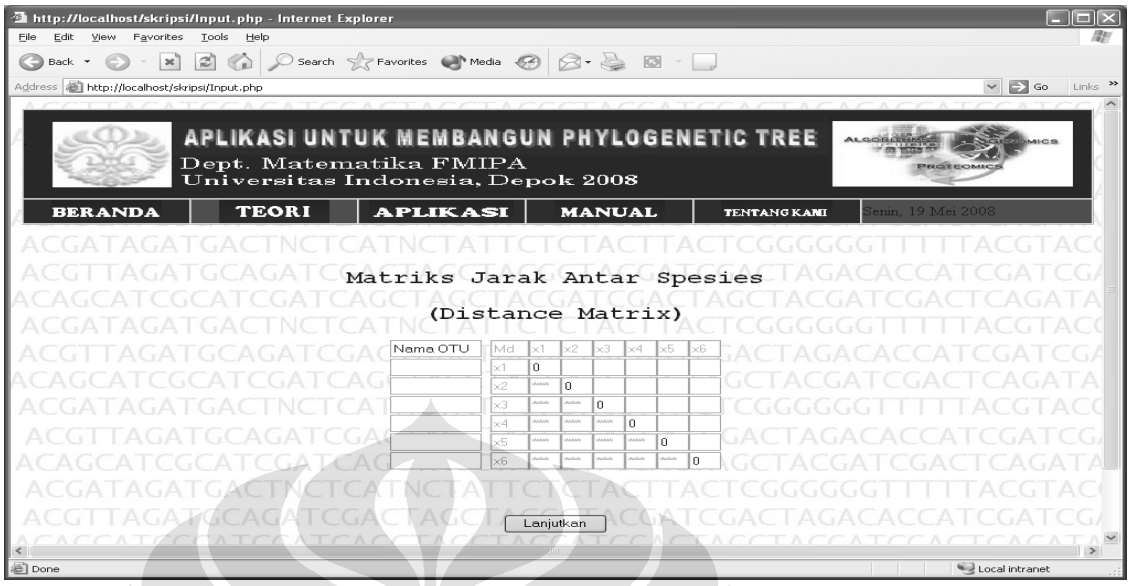
Aplikasi yang dibuat pada tugas akhir ini memiliki 5 (lima) menu pada halaman utama. Salah satu menu yang menjadi tujuan penulisan tugas akhir ini adalah menu Aplikasi yaitu menu aplikasi untuk membangun *phylogenetic tree* (*source code* terlampir pada *file* **Aplikasi.php**). Tampilan awal menu Aplikasi diperlihatkan pada Gambar 16.



Gambar 16. Tampilan awal menu Aplikasi

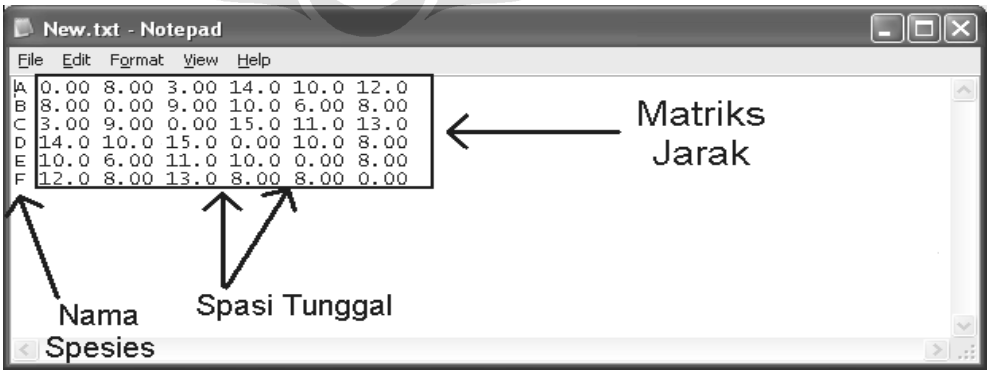
Pada Gambar 16, aplikasi meminta *input* dari *user* berupa matriks jarak. *Input* dapat dimasukan dengan 2 (dua) cara, yaitu:

1. *Input* secara manual, yaitu dengan cara mengisi *textbox* "Masukan jumlah OTU pada kotak di bawah ini". Kemudian tekan **OK** dan *user* akan menuju halaman **Input.php** (*source code* terlampir), minimal OTU yang dapat di-*input* untuk membuat *tree* adalah 3 (tiga). Pada halaman ini *user* akan diminta memasukkan elemen-elemen matriks jarak (*nonnegatif*) satu persatu pada kotak yang tersedia. Misalkan jumlah spesies atau OTU yang akan diteliti adalah 6, maka *input* matriks akan ditampilkan seperti gambar berikut.



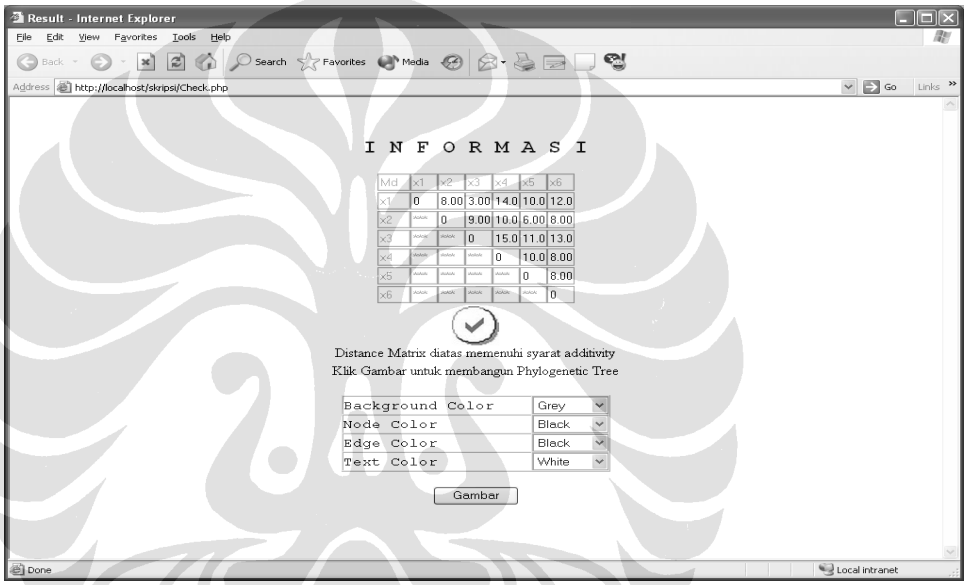
Gambar 17. *Form input* matriks jarak

2. *Input* menggunakan *file* (*upload file*), yaitu dengan meng-*upload textfile* yang berisi matriks jarak pada kotak *upload file.* Format penulisan matriks jarak diperlihatkan pada Gambar 17. Selanjutnya dengan menekan tombol **Lanjutkan**, *file* akan dicek apakah *file* yang di-*upload* sudah sesuai format. Pengecekan terjadi pada halaman **Input.php** (*source code* terlampir).



Gambar 18. Contoh format *textfile* matriks jarak

Setelah user memasukkan *input* berupa matriks jarak dengan salah satu cara yang disebutkan sebelumnya, matriks tersebut kemudian diperiksa apakah matriks tersebut memenuhi sifat *additive* atau tidak, tujuannya adalah jika matriks tidak bersifat *additive*, maka ada kemungkinan *tree* yang lain yang tidak terbentuk pada aplikasi ini. Proses pemeriksaan terjadi pada *file* **Check.php** (*souce code* terlampir). Perhatikan gambar berikut.



Gambar 19. Periksa matriks jarak untuk sifat *Additive*

Selanjutnya dengan menekan tombol **Gambar,** maka secara otomatis matriks jarak tersebut akan diproses untuk membentuk sebuah *phylogenetic tree*. *Source code* algoritma pembentukan *tree* dengan algoritma *neighbor-joining* terlampir pada *file* **Image.php**.

Output dari aplikasi ini adalah berupa gambar *phylogenetic tree* yang merepresentasikan matriks jarak yang di *input*.