IMPLEMENTASI PENYUSUNAN KATA MENGGUNAKAN ALGORITMA GENETIKA DALAM PERMAINAN *SCRABBLE*

LAPORAN TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
untuk Memperoleh Gelar Sarjana Teknik Informatika

Disusun Oleh:

Nama : Dinda Eling Kartikaning Sasmito
NIM : 07523164

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2011

HALAMAN JUDUL

IMPLEMENTASI PENYUSUNAN KATA MENGGUNAKAN ALGORITMA GENETIKA DALAM PERMAINAN *SCRABBLE*

LAPORAN TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
untuk Memperoleh Gelar Sarjana Teknik Informatika

Disusun Oleh:

Nama : Dinda Eling Kartikaning Sasmito
NIM : 07523164

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2011

LEMBAR PENGESAHAN PEMBIMBING

IMPLEMENTASI PENYUSUNAN KATA MENGGUNAKAN ALGORITMA GENETIKA DALAM PERMAINAN *SCRABBLE*

LAPORAN TUGAS AKHIR

Disusun Oleh:

Nama : Dinda Eling Kartikaning Sasmito
NIM : 07523164

Yogyakarta, Agustus 2011

Telah Diterima dan Disetujui dengan Baik Oleh:

Dosen Pembimbing

Zainudin Zukhri, S.T, M.IT

LEMBAR PENGESAHAN PENGUJI

IMPLEMENTASI PENYUSUNAN KATA MENGGUNAKAN ALGORITMA GENETIKA DALAM PERMAINAN *SCRABBLE*

TUGAS AKHIR

Disusun Oleh:

Nama : Dinda Eling Kartikaning Sasmito

NIM : 07523164

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Informatika
Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, Agustus 2011

Tim Penguji

Zainudin Zukhri, S.T., M.I.T
Ketua

Izzati Muhimmah, S.T., M.Sc., Ph.D
Anggota I

Syarif Hidayat, S.Kom, M.I.T
Anggota II

Mengetahui,
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

Yudi Prayudi, S.Si, M.Kom

LEMBAR PERNYATAAN KEASLIAN
HASIL TUGAS AKHIR

Saya yang bertanda tangan di bawah ini,
Nama : Dinda Eling Kartikaning Sasmito
NIM : 07523164

Menyatakan bahwa seluruh komponen dan isi dalam laporan Tugas Akhir ini adalah hasil karya sendiri. Apabila dikemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, maka saya siap menanggung resiko dan konsekuensi apapun.

Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, Agustus 2011

Dinda Eling Kartikaning Sasmito

HALAMAN PERSEMBAHAN

Karya sederhana ini kupersembahkan untuk
keluarga kecilku tercinta,
Papa Slamet Sasmito,
Mama Lesly Andalusia,
Mbakku, Endah Sasmitohening Stephanie,
Kembaranku, Dindi Eneng Chandraning Sasmito,

HALAMAN MOTTO

Harapan adalah doa dalam tindakan .
doa anda meningkat kelasnya menjadi harapan bila anda membuktikan
kesungguhan permintaan didalam upaya yang nyata.

Keberhasilan sering berasal dari langkah yang salah ,
tetapi yang ternyata menuju arah yang tepat,
maka jangan terlalu menyesalkan kesalahan .
Bersikaplah lebih positif.

Apapun masalah anda
jangan turun,
jangan tergesa-gesa
dan jangan berhenti.

(Mario Teguh)

KATA PENGANTAR

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Assalamu'alaikum Wr. Wb

Alhamdulillah, segala puji dan syukur kepada Allah, atas rahmat, hidayah dan karuniaNya, sehingga penulisan laporan Tugas Akhir yang berjudul **Implementasi Penyusunan Kata Menggunakan Algoritma Genetika dalam Permainan *Scrabble*** ini dapat penulis selesaikan dengan baik. Shalawat dan salam semoga senantiasa tercurah kepada Nabi Muhammad SAW, para kerabat serta pengikutnya hingga hari akhir nanti.

Laporan Tugas Akhir ini dibuat sebagai salah satu syarat yang harus dipenuhi untuk memperoleh gelar sarjana di Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia. Laporan Tugas Akhir juga sebagai sarana untuk mempraktekkan secara langsung ilmu dan teori yang telah didapatkan selama penulis menjalani studi di Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

Melalui Tugas Akhir ini penulis dapat mempelajari pembuatan implementasi teori Algoritma Genetika dengan baik beserta dokumentasinya, sehingga dapat dikembangkan lebih lanjut. Selama membuat implementasi ini penulis mengalami sedikit kesulitan dalam hal merepresentasikan teori yang ada menjadi bentuk implementasi utuh. Namun kesulitan tersebut dapat teratasi berkat dukungan dari beberapa pihak. Untuk itu pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT, atas segala rahmat dan karuniaNya, sehingga penulis diberikan kesehatan dalam menyelesaikan Tugas Akhir ini, dan dipertemukan dengan orang-orang yang membantu penulis baik secara langsung maupun secara moral.
2. Orang tua penulis, Mama Lesly Andalusia dan Papa Slamet Sasmito, atas semua do'a, dukungan, nasehat, didikan dan segala yang diberikan.

3. Bapak Ir. Gumbolo Hadi Susanto, M.Sc., selaku dekan Fakultas Teknologi Industri Universitas Islam Indonesia dan seluruh jajaran Dekanat Fakultas Teknologi Industri.
4. Bapak Yudi Prayudi, S.Si., M.Kom., selaku Ketua Jurusan Teknik Informatika.
5. Bapak Zainudin Zukhri, S.T., M.IT., selaku pembimbing yang telah memberikan pengarahan, bimbingan, saran serta semangat selama pengerjaan Tugas Akhir dan laporan ini.
6. Dosen-dosen Jurusan Teknik Informatika Universitas Islam Indonesia dan seluruh jajaran staf jurusan. Terima kasih atas ilmu pengetahuan, saran dan motivasi serta bantuannya.
7. Saudaraku, Mbak Endah dan Dindi, yang selalu memberikan do'a, dukungan, semangat dan motivasi dalam segala hal.
8. Muhammad Arief Widyananda, terima kasih atas do'a, dukungan, semangat dan bantuannya.
9. Orang-orang terdekatku, Dian Afriani, Mbak Niddya Ayu, Debby Patricia, Vira Haqni, "Lulut" Amalia, Anggit Prabangkara yang menghibur dan mengisi waktu jenuh selama pengerjaan Tugas Akhir dan laporan ini dengan keceriaan, serta selalu memberikan dukungan dan semangat.
10. Teman-teman Laboratorium Pemrograman dan Informatika Teori (PIT). Mas Indrato, Mas Dayu, Mas Andi, Mbak Fafa, Mas Indra Diday, Mas Wira, Mas Rakhmat, Mbak Vivi, Ronny, Fikar, Puguh, Hada dan seluruh asisten PIT, terima kasih atas kerja keras, pelajaran dan kekeluargaan kalian.
11. Teman-teman Laboratorium Terpadu Informatika, Mas Bowo, Mas Hendra, Mbak Liza, Mbak Kisty, Mas Ucup, Mbak Lutfia dan seluruh teman-teman asisten serta laboran Terpadu Informatika, terima kasih atas ilmunya.
12. Teman-teman seperjuangan, Linda, Disty, Yunita, Dina dan B *class*, terima kasih atas persahabatan yang diberikan.

13. Teman-teman KKN Jrakah Antar Waktu 2010, khususnya Unit-58 Gesikan, seluruh warga Gesikan dan Dosen Pembimbing yang telah memberikan warna baru dalam hidup penulis.
14. Teman-teman Informatika 2005-2011, khususnya teman-teman 2007 yang sama-sama berjuang dalam Tugas Akhir.
15. Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu sehingga laporan tugas akhir terselesaikan.

Terima kasih kepada semua pihak yang telah membantu terselesaikannya penulisan laporan Tugas Akhir ini. Semoga Allah SWT melimpahkan rahmat dan hidayahNya serta membalas semua kebaikan yang telah diberikan.

Penulis menyadari bahwa dalam penyusunan laporan Tugas Akhir ini masih banyak terdapat kekeliruan dan kekurangan. Untuk itu penulis menyampaikan permohonan maaf dan diharapkan kritik dan saran yang membangun untuk penyempurnaan di kesempatan berikutnya.

Akhir kata semoga laporan ini dapat bermanfaat bagi kita semua. Amin.

Wassalamu'alaikum Wr. Wb

Yogyakarta, Agustus 2011

Dinda Eling Kartikaning Sasmito

SARI

Metode belajar sambil bermain diakui lebih menarik dibandingkan dengan metode belajar secara monoton. Hal ini mempengaruhi sarana belajar sambil bermain yang semakin berkembang dari waktu ke waktu, diantaranya permainan *scrabble* sebagai sarana belajar bahasa inggris. Permainan *scrabble* pun telah semakin modern, disajikan menggunakan perangkat lunak (*software*). Metode kecerdasasan buatan dimanfaatkan untuk pengembangan perangkat lunak permainan *scrabble* tersebut. Salah satu metode kecerdasan buatan yang dapat digunakan untuk mengembangkan perangkat lunak permainan *scrabble* adalah Algoritma Genetika. Menggunakan algoritma ini, komputer akan mencari kata optimum dalam permainan *scrabble* sehingga komputer dapat menjadi lawan bermain. Tugas Akhir ini merupakan implementasi dari teori Algoritma Genetika menjadi sebuah perangkat lunak permainan *scrabble*. Untuk membangun implementasi tersebut, dilakukan analisis kebutuhan perangkat lunak dan perancangan perangkat lunak. Analisis kebutuhan dilakukan menggunakan metode terstruktur dan perancangan dilakukan mengunakan *flowchart*. Setelah dibuat implementasi kemudian dilakukan analisis kinerja perangkat lunak dengan pengujian normal dan tidak normal serta analisis waktu kompilasi perangkat lunak. Hasil penelitian Tugas Akhir ini adalah Algoritma Genetika mampu diterapkan dalam permainan *scrabble* dengan waktu kompilasi yang singkat. Akan tetapi, untuk implementasi yang lebih optimum, dibutuhkan analisis representasi kromosom yang tepat.

Kata Kunci:
Algoritma Genetika, *Scrabble*, optimasi kata, permainan kecerdasan buatan.

TAKARIR

alelle	nilai dari gen
blank	huruf kosong
crossover	penyilangan dua buah kromosom untuk mendapatkan kromosom baru dengan sifat-sifat indukannya
edit distance	jumlah minimum penyisipan huruf, penghapusan huruf dan pergantian huruf yang dibutuhkan untuk mengubah satu kata menjadi kata yang lain
fitness	nilai kecocokan
genotype	sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom
list	daftar, rangkaian objek
object oriented	metode analisis yang berorientasi pada objek
pad score	pencatat nilai / skor
random-walk	pengacakan secara berkala
scrabble	permainan menyusun kata dari huruf-huruf acak yang dimiliki dengan poin masing-masing pada setiap huruf
soft computing	metode pengolahan data dengan aturan yang disesuaikan, tidak bersifat pasti
vocabulary	kosa kata

DAFTAR ISI

HALAMAN JUDUL ..... **i**
LEMBAR PENGESAHAN PEMBIMBING ..... **ii**
LEMBAR PENGESAHAN PENGUJI ..... **iii**
LEMBAR PERNYATAAN KEASLIAN TUGAS AKHIR ..... **iv**
HALAMAN PERSEMBAHAN ..... **v**
HALAMAN MOTTO ..... **vi**
KATA PENGANTAR ..... **vii**
SARI ..... **x**
TAKARIR ..... **xi**
DAFTAR ISI ..... **xii**
DAFTAR TABEL ..... **xv**
DAFTAR GAMBAR ..... **xvi**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan Penelitian 2
1.5 Manfaat Penelitian 2
1.6 Metodologi Penelitian 2
1.6.1 Analisis Kebutuhan Perangkat Lunak 2
1.6.2 Perancangan Perangkat Lunak 3
1.6.3 Implementasi perangkat Lunak 3
1.6.4 Analisis Kinerja Perangkat Lunak 3
1.7 Sistematika Penulisan 3
BAB II LANDASAN TEORI **5**
2.1 *Scrabble* 5
2.1.1 Sejarah *Scrabble* 5
2.1.2 Aturan Permainan *Scrabble* 5

2.1.3 Perhitungan Poin 8
2.1.4 Tahap Permainan 9
2.1.5 Skenario pada Permainan *Scrabble* 10
2.2 Algoritma Genetika 12
2.2.1 Dasar-dasar Algoritma Genetika 12
2.2.2 Definisi Penting dalam Algoritma Genetika 14
2.2.3 Siklus Algoritma Genetika 15
2.3 Algoritma *Edit Distance* 16
BAB III ANALISIS KEBUTUHAN DAN PERANCANGAN PERANGKAT LUNAK 22
3.1 Analisis Kebutuhan Perangkat Lunak 22
3.1.1 Metode Analisis 22
3.1.2 Hasil Analisis 22
3.2 Perancangan Perangkat Lunak 26
3.2.1 Metode Perancangan 26
3.2.2 Hasil Perancangan 26
3.2.3 Perancangan Pemodelan Algoritma Genetika dalam Permainan *Scrabble* 33
BAB IV IMPLEMENTASI DAN ANALISIS KINERJA PERANGKAT LUNAK 45
4.1 Implementasi Perangkat Lunak 45
4.1.1 Batasan Implementasi Perangkat Lunak 45
4.1.2 Tampilan Menu Awal 46
4.1.3 Tampilan Cara Bermain 46
4.1.4 Tampilan Awal Permainan 47
4.1.5 Tampilan Akhir Permainan 48
4.2 Analisis Kinerja Perangkat Lunak 49
4.2.1 Pengujian Normal 49
4.2.2 Pengujian Tidak Normal 51
4.3 Algoritma Genetika dalam Perangkat Lunak 53
4.3.1 Alur Proses Genetika 53

4.3.2 Waktu Kompilasi 68
4.4 Kelebihan Perangkat Lunak 70
4.5 Kekurangan Perangkat Lunak 70
BAB V KESIMPULAN DAN SARAN 71
5.1 Kesimpulan 71
5.2 Saran 71
DAFTAR PUSTAKA 73

DAFTAR TABEL

Tabel 2.1	Distribusi dan Poin Huruf *Scrabble*	7
Tabel 4.1	Nilai *Fitness*, Probabilitas Seleksi, Probabilitas Kumulatif dan Nilai Random pada Proses Seleksi dengan Roda rolet	61
Tabel 4.2	Probabilitas *Crossover* Kromosom	62
Tabel 4.3	Perbandingan Waktu Kompilasi Berdasarkan Parameter Algoritma	68

DAFTAR GAMBAR

Gambar 2.1	Papan *Scrabble*	6
Gambar 2.2	Contoh Perhitungan Poin	9
Gambar 2.3	Skenario Permainan *Scrabble* Meletakkan Huruf Berdempet sehingga Membentuk Dua Buah Kata atau Lebih	11
Gambar 2.4	Skenario Permainan *Scrabble* Menggunakan Lebih dari Satu Huruf Papan	11
Gambar 2.5	Skenario Permainan *Scrabble* Menyambung Kata yang Sudah Ada di Papan Permainan	12
Gambar 2.6	*Flowchart* Algoritma Genetika	14
Gambar 2.7	Ilustrasi Representasi Penyelesaian Permasalahan dalam Algoritma Genetika	15
Gambar 2.8	Siklus Algoritma Genetika oleh David Goldberg	15
Gambar 2.9	Contoh Pencarian *Edit Distance* Dua Buah *Srting*	16
Gambar 2.10	Contoh Tabel *Edit Distance*	20
Gambar 2.11	Pola Urutan Perubahan *String*	21
Gambar 3.1	*Flowchart* Perangkat Lunak Permainan *Scrabble*	27
Gambar 3.2	*Flowchart* Permainan	28
Gambar 3.3	*Flowchart* Permainan *Player*	29
Gambar 3.4	*Flowchart* Permainan Komputer	29
Gambar 3.5	*Flowchart* Pencarian Kata Terbaik	30
Gambar 3.6	Rancangan Tampilan Menu Awal	31
Gambar 3.7	Rancangan Tampilan Saat Permainan	32
Gambar 3.8	Rancangan Tampilan Cara Bermain	33
Gambar 3.9	Contoh Kondisi Papan Permainan	34
Gambar 3.10	Gambaran Representasi Kromosom	35
Gambar 3.11	*Flowchart* Representasi Kromosom	36
Gambar 3.12	*Flowchart* Inisialisasi	37
Gambar 3.13	*Flowchart* Fungsi Objektif	38

Gambar 3.14 *Flowchart* Fungsi *Fitness* 39
Gambar 3.15 *Flowchart* Proses Seleksi 40
Gambar 3.16 Ilustrasi Metode Penyilangan Berbasis Posisi 41
Gambar 3.17 *Flowchart* Proses Penyilangan 42
Gambar 3.18 Ilustrasi Proses Mutasi 43
Gambar 3.19 *Flowchart* Proses Mutasi 43
Gambar 4.1 Tampilan Menu Awal 46
Gambar 4.2 Tampilan Cara Bermain 47
Gambar 4.3 Tampilan Awal Permainan 48
Gambar 4.4 Tampilan Akhir Permainan Jika Menang 48
Gambar 4.5 Tampilan Akhir Permainan Jika Kalah 48
Gambar 4.6 Pengujian Membentuk Sebuah Kata Inggris 49
Gambar 4.7 Pengujian Membatalkan Peletakan Huruf 50
Gambar 4.8 Konfirmasi Pengujian Menu Lewat 50
Gambar 4.9 Pengujian Huruf Tidak Teratur atau Kata Tidak Memiliki Arti 51
Gambar 4.10 Pengujian Memilih Tombol OK Sebelum Meletakkan Huruf ke Papan 52
Gambar 4.11 Pengujian Kata Pertama pada Permainan 52
Gambar 4.12 Huruf Acak Pemain pada Simulasi Permainan 54
Gambar 4.13 Proses Langkah Pemain pada Simulasi Permainan 55
Gambar 4.14 Tampilan Permainan Setelah Langkah Pemain pada Simulasi Permainan 56
Gambar 4.15 Proses Penyimpanan Huruf Papan Dapat Disambung 57
Gambar 4.16 Kode Program Mengacak Panjang Kromosom 57
Gambar 4.17 Kode Program Proses Genetika 57
Gambar 4.18 Kode Program Pengambilan Huruf Papan 58
Gambar 4.19 Hasil Proses Pengambilan Huruf Papan 58
Gambar 4.20 Kromosom Awal 59
Gambar 4.21 Nilai *Fitness* Kromosom pada Populasi Awal 60
Gambar 4.22 Hasil Kromosom Setelah Elitisme 60

Gambar 4.23 Proses Seleksi dengan Metode Roda rolet 62
Gambar 4.24 Proses *Crossover* 64
Gambar 4.25 Probabilitas Mutasi Gen 66
Gambar 4.26 Hasil Kromosom Setelah Mutasi 67
Gambar 4.27 Nilai *Fitness* Baru 67
Gambar 4.28 Tampilan Setelah Komputer Main 68

BAB I
PENDAHULUAN

1.1 Latar Belakang

Scrabble adalah suatu permainan menyusun huruf-huruf menjadi sebuah kata yang mempunyai arti. *Scrabble* dimainkan pada sebuah papan dengan kotak-kotak kecil berukuran 15x15. Menjadi sarana hiburan adalah manfaat utama sebuah permainan. Selain menjadi hiburan, *scrabble* yang dimainkan dalam bahasa asing bermanfaat sebagai sarana belajar. Melalui *scrabble*, anak-anak maupun orang dewasa dapat lebih mudah mempelajari kata asing beserta artinya. Memainkan *scrabble* dalam bahasa asing (biasanya bahasa Inggris) akan memberi tambahan kosa kata (*vocabulary*) baru untuk lebih mudah diingat.

Permainan *scrabble* berupa *game* PC saat ini sudah banyak dibuat oleh beberapa perusahaan *game* di dunia. Menggunakan komputer, scrabble dapat dimainkan oleh satu orang melawan komputer atau dengan banyak orang (paling banyak 4 orang). *Scrabble* yang disajikan dalam *game* PC melibatkan "otak" pada komputer sebagai lawan bermain. Sebagai lawan bermain, komputer harus memiliki pengetahuan yang setara dengan manusia agar menjadi lawan yang sepadan dan permainan tidak membosankan. Untuk itulah, "otak" komputer dibuat menggunakan kecerdasan buatan. Algoritma optimasi seperti Algoritma Genetika dapat digunakan untuk meningkatkan kemampuan "otak" komputer ini. Algoritma Genetika akan berperan dalam optimasi pencarian kata yang dapat terbentuk dalam permainan *scrabble*.

Algoritma Genetika merupakan algoritma pencarian yang berdasar pada seleksi alam dan genetika alam. Pertama kali Algoritma Genetika diperkenalkan oleh John Holland pada tahun 1975. Dalam beberapa tahun terakhir, komunitas Algoritma Genetika telah banyak digunakan dalam bidang lain seperti pada permasalahan industri yang kemudian memperluas bidang studi sebelumnya. Dalam penelitian tugas akhir ini, Algoritma Genetika digunakan dalam optimasi pencarian kata pada permainan *scrabble* dimaksudkan sebagai pembelajaran implementasi Algoritma Genetika secara nyata.

1.2 Rumusan Masalah

Bagaimana memberikan gambaran pembelajaran Algoritma Genetika dalam permasalahan pembentukan kata pada permainan *scrabble.*

1.3 Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah:

1 Permainan *scrabble* dilakukan oleh 1 pemain melawan komputer
2 Aturan permainan sama dengan permainan *scrabble* pada umumnya
3 Bahasa yang digunakan dalam permainan *scrabble* adalah Bahasa Inggris
4 Kamus pada permainan *scrabble* bersifat statis
5 Perangkat lunak yang dibuat tidak menyimpan nilai tertinggi pemain
6 Perangkat lunak yang dibuat tidak memiliki tingkatan level permainan.

1.4 Tujuan Penelitian

Tujuan dari penelitian tugas akhir ini adalah sebagai implementasi nyata pemanfaatan Algoritma Genetika dalam permainan *scrabble*.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Memberikan gambaran penerapan Algoritma Genetika sebagai salah satu algoritma optimasi.
2. Menjadi sarana bermain anak maupun orang dewasa sehingga dapat menjadi hiburan sekaligus menjadi sarana belajar Bahasa Inggris untuk mengingat kosa kata (*vocabulary*).

1.6 Metodologi Penelitian

Metodologi penelitian yang digunakan dalam tugas akhir ini meliputi:.

1.6.1 Analisis Kebutuhan Perangkat Lunak

Analisis ini berupa metode yang digunakan beserta hasil analisis masukan (*input*), proses, keluaran (*output*) serta antarmuka yang diinginkan.

1.6.2 Perancangan Perangkat Lunak

Rancangan perangkat lunak dibuat berupa diagram alir serta rancangan antarmuka perangkat lunak.

1.6.3 Implementasi Perangkat Lunak

Rancangan perangkat lunak yang telah dibuat kemudian diimplementasikan menjadi sebuah perangkat lunak yang utuh.

1.6.4 Analisis Kinerja Perangkat Lunak

Hasil implementasi dilakukan pengujian untuk mengetahui kinerja perangkat lunak yang dibuat.

1.7 Sistematika Penulisan

Dalam penyusunan tugas akhir ini, sistematika penulisan dibagi menjadi beberapa bab, yaitu:

Bab 1 membahas masalah umum mengenai optimasi Algoritma Genetika dalam permainan *Scrabble* meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

Bab 2 membahas landasan teori sebagai sumber atau alat dalam memahami permasalahan yang berhubungan dengan tugas akhir seperti *scrabble* dan Algoritma Genetika.

Bab 3 membahas analisis kebutuhan perangkat lunak yang akan dibuat, meliputi metode analisis dan hasil analisis kebutuhan perangkat lunak permainan *scrabble* dengan optimasi Algoritma Genetika menggunakan pendekatan terstruktur. Pada bab ini juga dibahas perancangan perangkat lunak meliputi metode perancangan dan hasil perancangan perangkat lunak. Metode perancangan yang dibuat menggunakan alat pengembangan sistem berupa diagram alir (*flowchart*).

Bab 4 membahas implementasi perangkat lunak, pengujian yang dilakukan untuk mengetahui kinerja perangkat lunak yang dibuat, alur algoritma genetika

dalam perangkat lunak yang dibangun, waktu kompilasi perangkat lunak, kelebihan dan kekurangan perangkat lunak yang dibangun. Pengujian perangkat lunak berupa simulasi permainan yang dilakukan pemain melawan komputer.

Bab 5 membahas kesimpulan berupa rangkuman dari hasil dan pembahasan pada bab sebelumnya serta saran yang yang perlu diperhatikan berdasarkan keterbatasan yang ditemukan dan asumsi-asumsi yang ada selama pengerjaan perangkat lunak untuk perbaikan dan pengembangan berikutnya.

BAB II
LANDASAN TEORI

2.1 *Scrabble*

Scrabble merupakan salah satu permainan di dunia yang menggunakan huruf-huruf dalam memainkannya dan menyusunnya dari atas ke bawah (vertikal) atau kiri ke kanan (horizontal). Sebuah permainan yang menggunakan referensi kamus untuk pengecekan keabsahan barisan huruf yang membentuk sebuah kata tersebut (Virniawati, 2007).

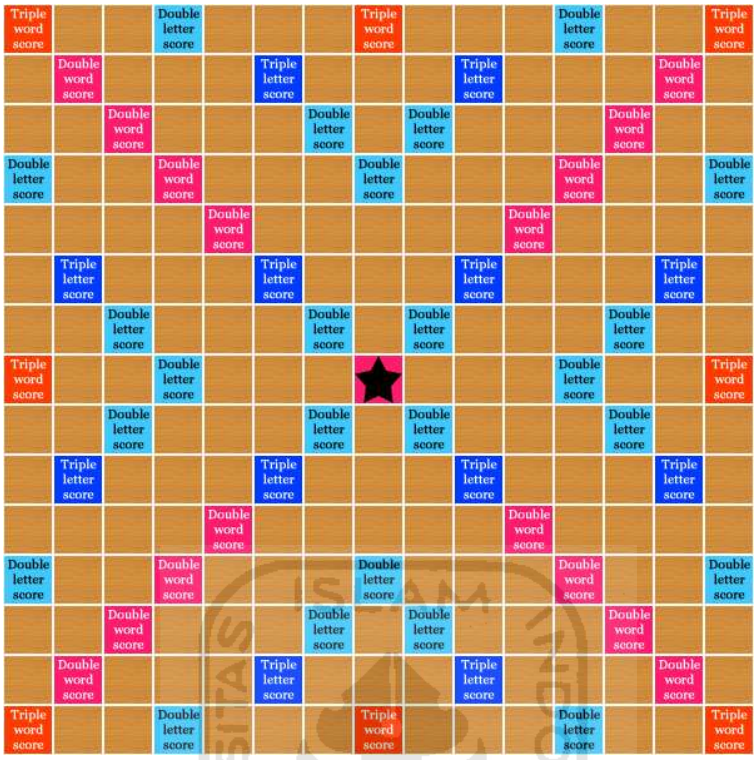
2.1.1 Sejarah *Scrabble*

Alfred Mosher Butts, seorang arsitek asal New York menciptakan permainan LEXIKO yang merupakan gabungan dari anagram dan teka-teki silang pada tahun 1931. Permainan tersebut kemudian dikembangkan, dimainkan pada sebuah papan dan dikenal dengan nama Criss-Crossword pada tahun 1938. Seorang pengusaha permainan, James Brunot menawarkan beberapa perbaikan konsep aturan dan desain pada tahun 1942. Permainan *scrabble* pun mulai diproduksi pada tahun 1948 dan mulai dikenal luas pada 1952. Hingga saat ini, permainan *scrabble* telah dijual di ratusan negara dengan puluhan ragam bahasa.

2.1.2 Aturan Umum Permainan

Aturan dalam permainan scrabble pada umumnya:

1. Permainan dilakukan oleh 2 hingga 4 orang pemain
2. Permainan dilakukan pada papan berukuran 15x15
3. Setiap kotak pada papan diisi dengan satu huruf
4. Papan permainan memiliki kotak-kotak bonus *double letter*, *triple letter*, *double word* dan *triple word* seperti pada **Gambar 2.1**.



Gambar 2.1 Papan *Scrabble*

a. Kotak *double letter* akan menggandakan poin huruf yang diletakkan pada kotak tersebut
b. Kotak *double word* akan menggandakan total poin untuk kata yang dibentuk jika salah satu huruf pada kata yang dibentuk terletak pada kotak *double word*
c. Kotak *triple letter* akan melipat tigakan poin huruf yang diletakkan pada kotak tersebut
d. Kotak *triple word* akan melipat tigakan total poin untuk kata yang dibentuk jika salah satu huruf pada kata yang dibentuk terletak pada kotak *triple word*

5. Distribusi huruf dalam satu kali permainan serta poin/nilai untuk setiap hurufnya (dalam versi Bahasa Inggris) dapat dilihat pada **Tabel 2.1**.

Tabel 2.1 Distribusi dan Poin Huruf *Scrabble*

No.	Huruf	Banyak	Poin
1.	A	9	1
2.	B	2	3
3.	C	2	3
4.	D	4	2
5.	E	12	1
6.	F	2	4
7.	G	3	2
8.	H	2	4
9.	I	9	1
10.	J	1	8
11.	K	1	5
12.	L	4	1
13.	M	2	3
14.	N	6	1
15.	O	8	1
16.	P	2	3
17.	Q	1	10
18.	R	6	1
19.	S	4	1
20.	T	6	1
21.	U	4	1
22.	V	2	4
23.	W	2	4
24.	X	1	8
25.	Y	2	4
26.	Z	1	10
27.	*Blank*	2	0

6. Setiap pemain memiliki 7 huruf random untuk disusun menjadi kata yang memiliki arti pada setiap gilirannya
7. Huruf *Blank* dapat diganti dengan huruf apapun pada saat pertama kali dikeluarkan dan tidak dapat diubah setelah digunakan.
8. Setiap pemain menyusun huruf yang dimiliki untuk dijadikan sebuah kata yang memiliki arti, diletakkan pada papan permainan dengan posisi menyamping atau menurun seperti pada Teka Teki Silang

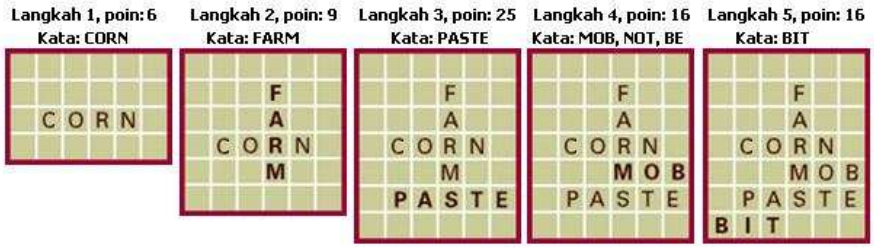
2.1.3 Perhitungan Poin

Perhitungan poin/nilai pada permainan *scrabble* berdasarkan situs resmi perusahaan permainan Hasbro (www.hasbro.com):

1. Gunakan *pad score* atau kertas untuk mencatat skor masing-masing pemain setiap gilirannya. Nilai untuk setiap huruf sesuai dengan angka di seiap huruf (seperti pada tabel 2.1)
2. Nilai untuk tiap giliran adalah jumlah dari nilai setiap huruf pada kata yang dibentuk atau diubah, disesuaikan dengan poin/nilai tambahan sesuai posisi huruf pada papan
3. Kotak penambahan nilai huruf, berwarna biru muda untuk menggandakan nilai setiap huruf pada kotak tersebut serta biru tua untuk nilai tiga kali lipat setiap huruf yang diletakkan pada kotak tersebut.
4. Kotak penambahan nilai kata, berwarna merah muda untuk menggandakan total nilai dari kata yang terbentuk serta merah tua untuk nilai tiga kali lipat dari total nilai kata yang terbentuk. Apabila terdapat kotak penambahan nilai huruf secara bersamaan, maka jumlahkan nilai untuk huruf tersebut terlebih dahulu dilanjutkan dengan menghitung total nilai kata yang dibentuk
5. Penambahan nilai huruf atau kata dari kotak bonus hanya berlaku pertama kali kotak digunakan. Untuk pemakaian selanjutnya, huruf dinilai sesuai dengan poin/nilai hurufnya saja (seperti pada tabel 2.1)
6. Jika huruf *blank* dimainkan, nilai tetap digandakan atau menjadi tiga kali lipat meskipun *blank* tidak memiliki nilai (nol)
7. Boleh terdapat dua atau lebih kata yang sama dalam satu permainan. Setiap kata dinilai penuh seperti aturan nomor 2, 3, 4 dan 5 pada cara perhitungan poin
8. ”Bingo”, jika pemain dapat menyusun 7 huruf yang dimiliki menjadi satu kata yang memiliki arti dalam satu kali giliran. Nilai tambahan 50 poin akan diberikan kepada pemain yang memperoleh ”Bingo”.

9. Ketika permainan berakhir, skor pemain yang masih memiliki huruf akan dikurangi dengan jumlah poin huruf yang dimiliki. Poin tersebut akan diberikan kepada pemain yang menyelesaikan permainan.
10. Pemain dengan total skor poin/nilai tertinggi adalah pemenang dalam permainan. Dalam beberapa kasus, pemenang ditentukan sebelum penambahan dan pengurangan terhadap huruf yang belum dimainkan seperti aturan nomor 9 pada perhitungan poin.

Contoh permainan scrabble serta perhitungan poin nya dapat dilihat pada **Gambar 2.2**. Pada gambar, permainan dimulai dengan kata *corn* dengan meletakkan huruf R pada tengah papan (kotak bintang) sehingga poin yang dihasilkan adalah 6. Giliran berikutnya terbentuk kata *farm* dengan meletakkan kata pada posisi huruf R pada kata sebelumnya sehingga poin yang dihasilkan adalah 9. Kata selanjutnya diletakkan di bawah kata *farm* sehingga membentuk kata *farms* dan *paste*. Jika dilihat pada papan, huruf P dan E diletakkan pada kotak *triple letter* sehingga poin yang dihasilkan adalah 25. Pemain selanjutnya meletakkan huruf O dan B sehingga membentuk kata *mob, not* dan *be*. Huruf O terletak pada kotak *double letter* sehingga poin yang dihasilkan adalah 16. Terakhir, huruf yang duletakkan adalah B, I dan T sehingga menghasilkan kata *bit, pi* dan *at* dengan posisi B pada kotak *double word* sehingga poin yang dihasilkan adalah 16.



Gambar 2.2 Contoh Perhitungan Poin

2.1.4 Tahap Permainan

Setelah mengetahui aturan umum permainan *scrabble*, berikut tahapan dalam permainan scrabble secara umum berdasarkan situs resmi Hasbro:

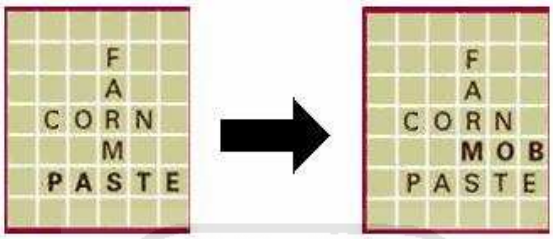
1. Pemain pertama menyusun dua atau lebih huruf yang dimiliki menjadi satu kata berarti dan menempatkannya pada papan permainan secara mendatar atau menurun dengan posisi salah satu huruf diletakkan di tengah papan (biasanya ditandai dengan kotak bergambar bintang)
2. Hitung poin/nilai yang didapat oleh pemain pertama. Kembali tambahkan huruf untuk pemain pertama hingga deret huruf yang dimiliki menjadi 7 huruf
3. Pemain kedua dan berikutnya pada setiap giliran menambahkan satu atau lebih huruf pada kata yang sudah dimainkan untuk membentuk kata baru. Semua huruf yang dimainkan di setiap giliran harus ditempatkan dalam satu baris atau satu kolom untuk membentuk setidaknya satu kata yang memiliki arti. Jika pada saat bersamaan huruf yang diletakkan menyentuh huruf lainnya di baris yang berdekatan, maka huruf-huruf tersebut juga harus memiliki arti. Setiap pemain mendapatkan nilai penuh untuk setiap kata yang terbentuk
4. Tidak ada huruf yang bergeser atau diganti jika sudah dimainkan di papan permainan dan dinilai
5. Saat memainkan *blank*, pemain harus menetapkan huruf yang diwakili. *Blank* akan tetap sebagai huruf tersebut hingga akhir permainan
6. Pemain dapat menukar beberapa, seluruh atau tidak sama sekali huruf yang dimiliki dengan stok huruf cadangan pada huruf yang ada. Setelah melakukan pertukaran, giliran pemain akan berakhir dan dilanjutkan oleh pemain berikutnya.

Permainan akan selesai jika salah satu pemain telah menghabiskan semua huruf yang dimiliki dan stok huruf cadangan sudah tidak ada atau seluruh pemain tidak dapat membuat kata yang berarti.

2.1.5 Skenario pada Permainan *Scrabble*

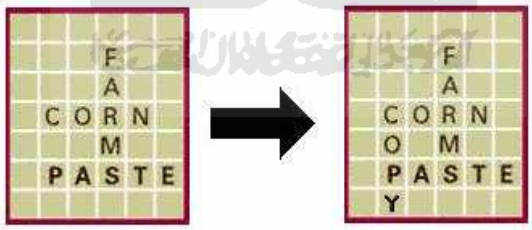
Dalam permainan *scrabble*, terdapat beberapa kemungkinan kasus yang terjadi. Berikut ini adalah kasus-kasus yang mungkin terjadi dalam permainan *scrabble*:

1. Meletakkan huruf berdempet sehingga membentuk dua buah kata atau lebih.
   Huruf yang diletakkan pada satu kali giliran bermain dapat membentuk sekaligus dua buah kata atau lebih seperti pada **Gambar 2.3**. Huruf yang diletakkan boleh berdempet dengan dua atau lebih kata pada papan, namun seluruh susunan huruf baik secara vertikal maupun horizontal harus memiliki arti.



Gambar 2.3 Skenario Permainan *Scrabble* Meletakkan Huruf Berdempet sehingga Membentuk Dua Buah Kata atau Lebih

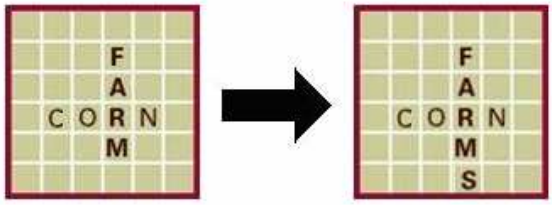
2. Membentuk kata menggunakan lebih dari satu huruf pada papan.
   Pemain menyusun huruf-huruf yang dimiliki ke papan permainan sehingga membentuk kata menggunakan dua atau lebih huruf yang sudah ada pada papan. Skenario kasus seperti ini dapat dilihat pada **Gambar 2.4**.



Gambar 2.4 Skenario Permainan *Scrabble* Menggunakan Lebih dari Satu Huruf Papan

3. Menyambung kata yang sudah ada di papan permainan.
   Dalam giliran bermain, pemain menyambungkan kata yang sudah ada di papan dengan menambahkan satu atau lebih huruf pada kata yang sudah ada di papan permainan. Huruf tambahan yang sering ditambahkan biasanya seperti “S”, “ES”, “ED”, “LY”, dan lain-lain sehingga dengan menyambung huruf-huruf

tersebut pada kata yang sudah ada di papan menyebabkan kata tersebut tetap memiliki arti. Skenario seperti ini dapat diperjelas dengan **Gambar 2.5**.



Gambar 2.5 Skenario Permainan *Scrabble* Menyambung Kata yang Sudah Ada di Papan Permainan

2.2 Algoritma Genetika

Peletak prinsip dasar sekaligus pencipta algoritma genetika adalah John Holland. Algoritma genetika menggunakan analogi secara langsung dari kebiasaan alami yaitu seleksi alam (Satriyanto, 2009). Algoritma genetika akan digunakan sebagai algoritma pencarian kata optimal untuk permainan *scrabble*.

2.2.1 Dasar Algoritma Genetika

Algoritma genetik merupakan teknik *search stochastic* (pencarian dengan peluang) yang berdasarkan mekanisme seleksi alam dan genetika natural. Yang membedakan algoritma genetik dengan berbagai algoritma konvensional lainnya adalah bahwa algoritma genetik memulai dengan suatu himpunan penyelesaian acak awal yang disebut populasi. Masing-masing individu di dalam populasi disebut kromosom, yang merepresentasikan suatu penyelesaian terhadap masalah yang ditangani (Fadlisyah, 2009).

Kromosom terbentuk setiap generasi dan kemudian dievaluasi menggunakan beberapa ukuran *fitness*. Untuk mendapatkan generasi baru, berbagai kromosom baru yang disebut *offspring*, dibentuk dengan salah satu cara sebagai berikut (Fadlisyah, 2009):

a. Memadukan dua kromosom dari generasi terakhir menggunakan operator *crossover*
b. Dengan memodifikasi sebuah kromosom menggunakan operator mutasi

Sedangkan generasi yang baru dapat dihasilkan dengan cara:

a. Penseleksian, berdasarkan nilai fitness, berbagai parent, dan offspring
b. Menolak yang lainnya, untuk menjaga ukuran populasi agar tetap konstan

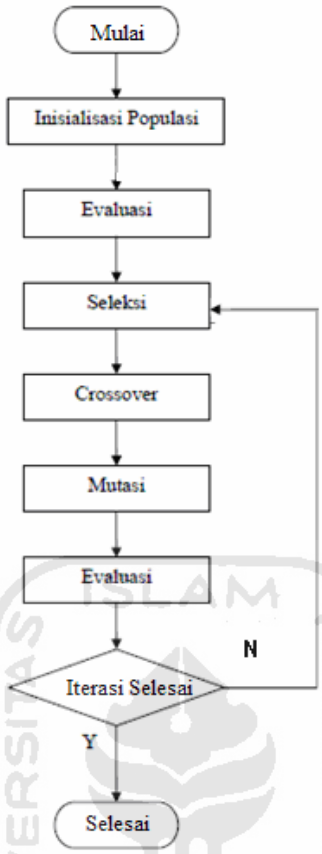
Beberapa hal yang harus dilakukan dalam algoritma genetika adalah (Satriyanto, 2009):

a. Mendefinisikan individu, dimana individu menyatakan salah satu solusi (penyelesaian) yang mungkin dari permasalahan yang diangkat
b. Mendefinisikan nilai *fitness*, yang merupakan ukuran baik tidaknya sebuah individu atau baik tidaknya solusi yang didapatkan
c. Menentukan proses pembangkitan populasi awal. Hal ini biasanya dilakukan dengan pembangkitan acak seperti *random-walk*
d. Menentukan proses seleksi yang digunakan
e. Menentukan proses perkawinan silang (*crossover*) dan mutasi gen yang akan digunakan

Algoritma genetika merupakan algoritma *softcomputing*, dimana urutan atau proses pengerjaannya tidak harus selalu sama, namun tetap memiliki komponen-komponen yang adal dalam algoritma genetika Secara umum, algoritma genetika adalah sebagai berikut (Kusumadewi dan Hari, 2005):

1. Generasi = 0
2. Inisialisasi Populasi Awal, P(generasi), secara acak
3. Evaluasi nilai fitness pada setiap individu dalam P(generasi)
4. Kerajakan langkah-langkah berikut hingga generasi mencapai maksimum generasi:
   a. Generasi = generasi+1
   b. Seleksi populasi untuk mendapatkan kandidat induk, P'(generasi)
   c. Lakukan *crossover* pada P'(generasi)
   d. Lakukan mutasi pada P'(generasi) // optional
   e. Lakukan evaluasi *fitness* setiap individu pada P'(generasi)
   f. Bentuk populasi baru P(generasi) = {P(generasi-1) yang bertahan, P'Generasi}

Untuk mempermudah memahami algoritma genetika, *flowchart* proses algoritma genetika dapat dilihat pada **Gambar 2.6**.



Gambar 2.6 *Flowchart* Algoritma Genetika

2.2.2 Definisi Penting dalam Algoritma Genetika

Beberapa definisi penting yang perlu diperhatikan untuk membangun oermasalahan dengan algoritma genetika adalah sebagai berikut (Satriyanto, 2009):

a. *Genotype* (Gen) : Sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom. Dalam algoritma genetika, gen ini bias berupa nilai *biner*, *float*, *integer* maupun karakter atau kombinatorial

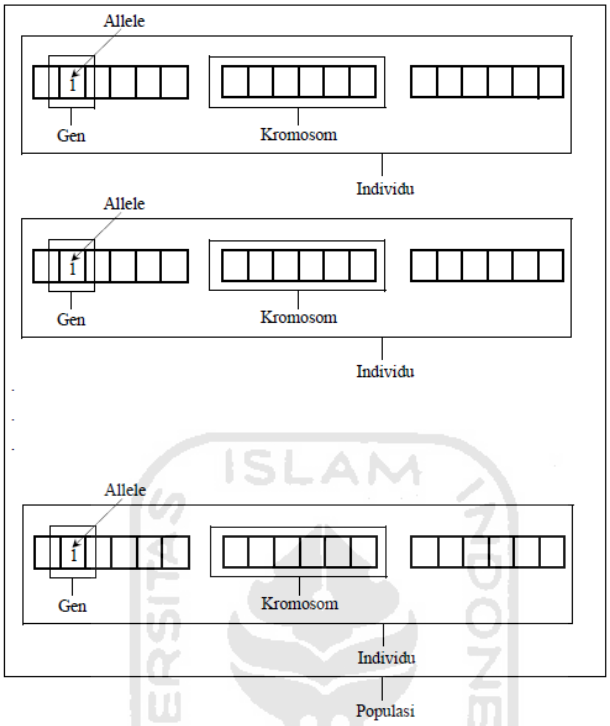
b. *Allele* : Nilai dari gen

c. Kromosom : Gabungan gen-gen yang membentuk suatu nilai tertentu

d. Individu : menyatakan satu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat

e. Populasi : merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evolusi

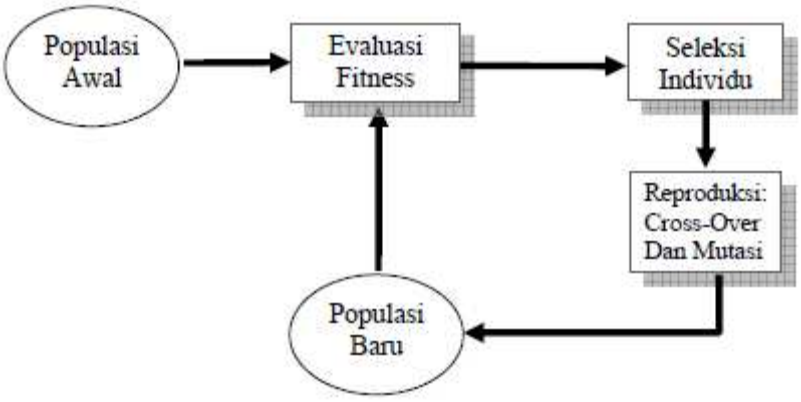
f. Generasi : menyatakan satu siklus proses evolusi atau nilai iterasi di dalam algoritma genetika



Gambar 2.7 Ilustrasi Representasi Penyelesaian Permasalahan dalam Algoritma Genetika

2.2.3 Siklus Algoritma Genetika

Siklus dari algoritma genetika pertama kali dikenalkan oleh David Goldberg, dimana gambaran siklus tersebut dapat dilihat pada **Gambar 2.8**. (Satriyanto, 2009):



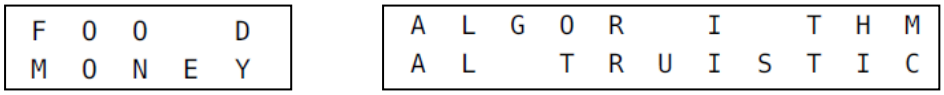
Gambar 2.8 Siklus Algoritma Genetika oleh David Goldberg

2.3 Algoritma *Edit Distance*

The edit distance between two words—sometimes also called the Levenshtein distance—is the minimum number of letter insertions, letter deletions, and letter substitutions required to transform one word into another (Jeff Ericson, 2011). *Edit distance* antara dua kata—kadang disebut sebagai jarak Levenshtein—adalah jumlah minimum penyisipan huruf, penghapusan huruf dan pergantian huruf yang dibutuhkan untuk mengubah satu kata menjadi kata yang lain (Jeff Ericson, 2011).

Edit distance dapat diterapkan dalam beberapa masalah, seperti pengecekan kata, mengoreksi kesalahan pengejaan dan pendeteksi praktik plagiat. Contoh aplikasi penerapan *edit distance* terdapat pada *email client* Yahoo Mail, pengecekan ejaan pada Google *search engine* dan pengecekan ejaan pada Microsoft Word. Dalam tugas akhir ini, *edit distance* digunakan untuk membandingkan individu yang terbentuk dari sebuah kromosom dengan kata-kata pada kamus.

Cara terbaik mengetahui perbedaan dua buah kata (*string*) adalah dengan meletakkan dua buah *string* yang dibandingkan dengan posisi atas bawah. Untuk proses penyisipan huruf, maka berikan spasi pada kata pertama. Berikan spasi pada kata kedua untuk proses penghapusan huruf. Kolom dengan dua karakter yang **berbeda** pada kedua kata menunjukkan proses perubahan huruf. Untuk lebih jelasnya dapat dilihat dua contoh pada **Gambar 2.9**.



Gambar 2.9 Contoh Pencarian *Edit Distance* Dua Buah *String*

Contoh pertama merupakan perbedaan antara *string* FOOD dan MONEY. Sekilas terlihat bahwa perbedaan (*edit distance*) pada dua *string* tersebut adalah 4. Artinya, untuk mengubah *string* FOOD menjadi MONEY dibutuhkan 4 operasi, yaitu:

a. Mengganti huruf F dengan huruf M

F O O D → **M** O O D

b. Mengganti huruf O dengan huruf N

M O **O** D → M O **N** D

c. Menyisipkan huruf E

M O N D → M O N **E** D

d. Mengganti huruf D dengan huruf Y

M O N E **D** → M O N E **Y**

Contoh kedua merupakan perbedaan *string* ALGORITHM dan ALTRUISTIC. Banyaknya langkah yang dilakukan untuk mengubah *string* ALGORITHM menjadi ALTRUISTIC adalah 6 langkah, yaitu:

a. Menghapus huruf G

A L **G** O R I T H M → A L O R I T H M

b. Mengubah huruf O Menjadi huruf T

A L **O** R I T H M → A L **T** R I T H M

c. Menyisipkan huruf U

A L T R I T H M → A L T R **U** I T H M

d. Menyisipkan huruf S

A L T R U I T H M → A L T R U I **S** T H M

e. Mengubah huruf H menjadi huruf I

A L T R U I S T **H** M → A L T R U I S T **I** M

f. Mengubah huruf M menjadi huruf C

A L T R U I S T I **M** → A L T R U I S T I **C**

Sehingga *edit distance* untuk kedua *string* di atas adalah 6.

Untuk mendapatkan *edit distance* antara dua buah *string*, dibutuhkan sebuah fungsi rekursif yang mendefinisikan *edit distance* antara dua buah *string* A[1..m] dan B[1..n] yang ditunjukkan oleh Edit(A[1..m], B[1..n]). Jika *string* tidak kosong, ada tiga kemungkinan untuk kolom terakhir dalam urutan perubahan terpendek (Jeff Ericson, 2011):

a. Penyisipan: Masukan terakhir pada baris bawah (*string* ke dua) kosong. Dalam hal ini, *edit distance* sama dengan Edit(A[1..m-1], B[1..n])+1. Penambahan dengan 1 merupakan biaya penyisipan terakhir, dan ekspresi rekursif memberikan biaya minimum untuk kolom lainnya.
b. Penghapusan: Masukan terakhir pada baris atas (*string* pertama) kosong. Dalam hal ini, *edit distance* sama dengan Edit(A[1..m], B[1..n-1])+1. Penambahan dengan 1 merupakan biaya penghapusan terakhir, dan ekspresi rekursif memberikan biaya minimum untuk kolom lainnya.
c. Pengubahan: Kedua baris (*string*) memiliki karakter pada kolom terakhir. Jika karakter pada kedua *string* sama, biaya pengubahan sama dengan 0 atau gratis, sehingga *edit distance* sama dengan Edit(A[1..m-1], B[1..n-1]). Jika karakter berbeda, maka *edit distance* sama dengan Edit(A[1..m-1], B[1..n-1])+1. Penambahan dengan 1 merupakan biaya pengubahan terakhir, dan ekspresi rekursif memberikan biaya minimum untuk kolom lainnya.

Edit distance antara dua buah *string* adalah nilai minimum dari tiga kemungkinan di atas. Sehingga didapatkan persamaan sebagai berikut:

$$Edit(A[1..m], B[1..n]) = \min \begin{Bmatrix} Edit(A[1..m-1], B[1..n]) + 1 \\ Edit(A[1..m], B[1..n-1]) + 1 \\ Edit(A[1..m-1], B[1..n-1]) + [A[m] \neq B[n]] \end{Bmatrix} \quad \text{.....(2.1)}$$

Perulangan tersebut memiliki dua basis kasus. Cara untuk mengkonversi *string* kosong menjadi *string* dengan n karakter adalah dengan menyisipkan huruf sebanyak n. Sedangkan cara untuk mengkonversi *string* dengan m karakter menjadi *string* kosong adalah dengan menghapus huruf sebanyak m. Jadi, jika ε menunjukkan *string* kosong, maka didapatkan:

$$Edit(A[1..m], \varepsilon) = m, \qquad Edit(\varepsilon, B[1..n]) = n \quad \text{.....(2.2)}$$

Kedua ekspresi di atas menyiratkan kasus dasar Edit(ε,ε) = 0. Berdasarkan persamaan 2.1 dan 2.2 di atas, dapat dibentuk persamaan baru dengan menyederhanakan notasi Edit(A[1..m], B[1..n]) atau dapat ditulis dengan

Edit(A[1..i], B[1..j]) menjadi Edit(i, j) dengan persamaan *edit distance* sebagai berikut (Jeff Ericson, 2011):

$$Edit(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \begin{cases} Edit(i-1,j)+1, \\ Edit(i,j-1)+1, \\ Edit(i-1,j-1)+[A[i] \neq B[j]] \end{cases} & \text{otherwise} \end{cases} \quad \text{.....(2.3)}$$

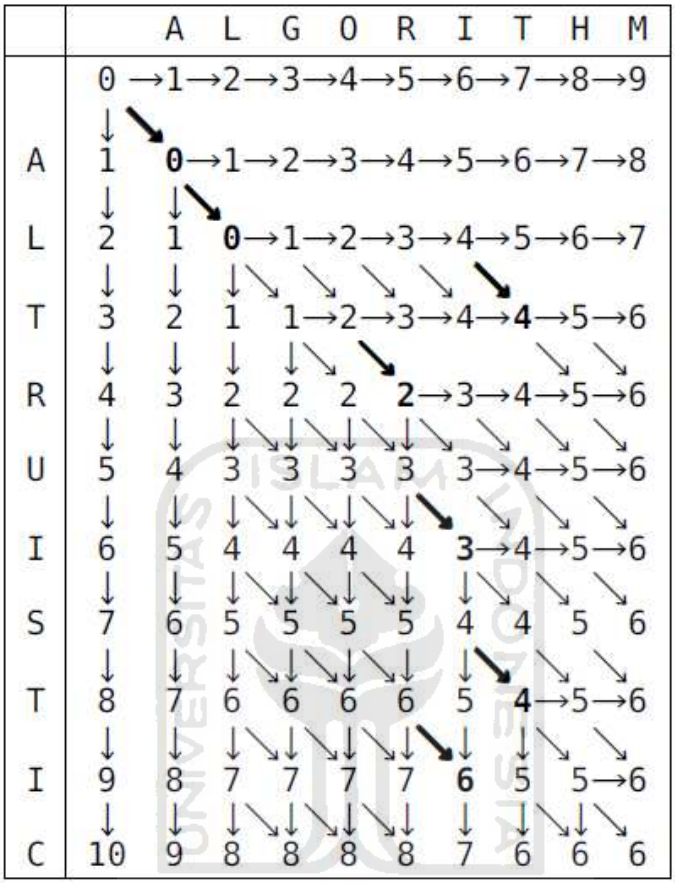
Algoritma *edit distance* yang didapatkan dari persamaan di atas adalah sebagai berikut (Jeff Ericson, 2011):

```
EditDistance(A[1..m], B[1..n]):
  for j ← 1 to n
      Edit[0, j] ← j
  for i ← 1 to m
      Edit[i, 0] ← i
      for j ← 1 to n
          if A[i] = B[j]
              Edit[i, j] ← min{Edit[i − 1, j] + 1, Edit[i, j − 1] + 1, Edit[i − 1, j − 1]}
          else
              Edit[i, j] ← min{Edit[i − 1, j] + 1, Edit[i, j − 1] + 1, Edit[i − 1, j − 1] + 1}
  return Edit[m, n]
```

Tabel *edit distance* untuk dua buah *string* akan terbentuk dari algoritma di atas. **Gambar 2.10** merupakan contoh tabel *edit distance* yang terbentuk dari perbandingan *string* ALGORITHM dan ALTRUISTIC (Jeff Ericson, 2011):

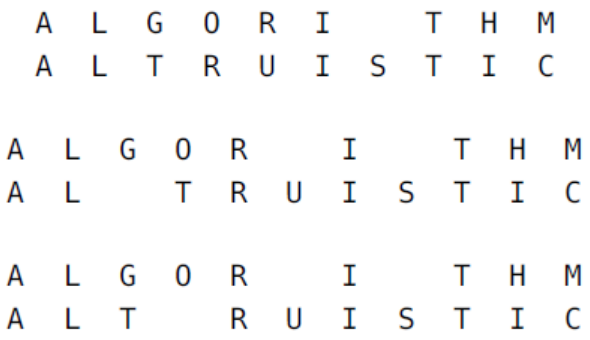
Angka yang dicetak tebal pada **Gambar 2.10** menunjukkan bahwa karakter pada kedua *string* sama. Panah menunjukkan pendahulunya yang mendefinisikan masing-masing masukan. Setiap arah panah sesuai dengan operasi pengubahan yang berbeda: horizontal menunjukkan proses penghapusan, vertikal menunjukkan proses penyisipan dan diagonal menunjukkan proses pengubahan (subtitusi). Panah yang di cetak tebal menunjukkan huruf tersebut bebas digantikan dengan huruf itu sendiri. Setiap arah panah dari sudut kiri atas ke sudut kanan bawah tabel mewakili *edit distance* optimal antara kedua *string*. Angka

terakhir yang terdapat pada sudut kanan bawah merupakan hasil akhir *edit distance* dua buah *string.*



Gambar 2.10 Contoh Tabel *Edit Distance*

Dalam hal ini bisa terdapat banyak pola *edit distance*. Untuk contoh *edit distance* pada ALGORITHM dan ALTRUISTIC di atas memiliki *edit distance* 6 dan dapat memiliki 3 pola urutan yang optimum untuk mengubah *string* ALGORITHM menjadi ALTRUISTIC. Pola tersebut dapat dilihat pada **Gambar 2.11**. Ketiga pola pada **Gambar 2.11** dapat dibuat urutan prosesnya seperti penjelasan urutan proses **Gambar 2.9**. Ketiga pola yang dihasilkan memiliki *edit distance* yang sama yaitu 6.



Gambar 2.11 Pola Urutan Perubahan *String*

BAB III
ANALISIS KEBUTUHAN DAN PERANCANGAN PERANGKAT LUNAK

3.1 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan merupakan langkah awal untuk menentukan perangkat lunak yang akan dibuat. Beberapa metode analisis dapat dilakukan, diantaranya metode analisis dengan pendekatan terstruktur, metode analisis dengan pendekatan sistematis dan metode analisis dengan pendekatan *Object Oriented*.

3.1.1 Metode Analisis

Metode analisis yang digunakan dalam pengembangan perangkat lunak permainan *scrabble* dengan algortima genetika ini adalah metode analisis dengan pendekatan terstruktur. Metode ini menganalisis masukan, proses, keluaran dan antarmuka yang dibutuhkan dalam pengembangan perangkat lunak yang dibuat dengan teratur sehingga perangkat lunak yang akan dibuat dapat didefinisikan dengan baik dan jelas.

3.1.2 Hasil Analisis

Hasil analisis kebutuhan untuk perangkat lunak permainan *scrabble* dengan algoritma genetika adalah sebagai berikut:

a. Analisis Kebutuhan Masukan

Berdasarkan analisis yang dilakukan, masukan yang dibutuhkan oleh perangkat lunak permainan *scrabble* adalah sebagai berikut:

1. Susunan huruf-huruf pada papan

   Pengguna atau pemain menyusun huruf random yang didapat menjadi kata yang memiliki arti dan meletakkan susunan huruf-huruf tersebut pada papan permainan untuk setiap gilirannya.

2. Pilihan bermain

   Pengguna atau pemain dapat menggunakan pilihan OK, Batal, Bantuan atau Lewat pada setiap gilirannya. Pilihan OK digunakan jika huruf telah disusun pada papan dan dirasa membentuk kata yang memiliki arti untuk dihitung poin/nilainya. Pilihan Batal digunakan untuk mengembalikan huruf-huruf yang telah diletakkan di papan ke rak. Pilihan ini digunakan untuk menyusun ulang huruf-huruf yang dimiliki. Pilihan Bantuan digunakan jika pemain tidak dapat membentuk kata dengan huruf-huruf yang ada sehingga meminta bantuan dari komputer untuk memberikan pilihan kata yang dapat dimainkan pada giliran tersebut. Pilihan Lewat digunakan jika pemain memilih untuk tidak melakukan apapun pada gilirannya dan permainan dilanjutkan dengan giliran komputer.

b. Analisis Kebutuhan Proses

Proses-proses yang dilakukan oleh perangkat lunak permainan *scrabble* yang dibuat adalah sebagai berikut:

1. Inisialisasi awal, yaitu proses pemberian nilai awal pada saat permainan dimulai. Proses ini memiliki sub proses yaitu:
   a. Inisialisasi papan, yaitu proses penyiapan papan *scrabble.* Pada proses ini, setiap kotak pada papan *scrabble* diberi nilai awal.
   b. Inisialisasi balok huruf, yaitu proses penyiapan balok-balok huruf *scrabble.* Pada proses ini, setiap balok huruf diberi nilai awal berupa huruf, poin, jumlah dan gambar.
2. Menampilkan tampilan, yaitu proses menampilkan tampilan pada saat permainan berlangsung di setiap gilirannya. Pada proses ini terdapat sub proses berupa:
   a. Gambar Papan, yaitu proses menampilkan keadaan papan pada saat permainan berlangsung di setiap gilirannya.
   b. Gambar Rak, yaitu proses menampilkan keadaan rak pemain pada saat permainan berlangsung di setiap gilirannya.

3. Random huruf, yaitu proses mengacak huruf yang diberikan untuk setiap pemain. Setiap pemain akan mendapatkan tujuh huruf acak untuk kemudian disusun membentuk kata pada papan. Huruf yang diperoleh diambil dari balok-balok huruf yang ada dengan distribusi jumlah huruf masing-masing.
4. Ambil dari rak, yaitu proses mengambil huruf yang ada dari rak pemain untuk kemudian diletakkan di papan permainan.
5. Letakkan huruf ke papan, yaitu proses meletakkan huruf yang telah diambil dari rak ke papan permainan.
6. Kembalikan huruf ke rak, yaitu proses untuk mengembalikan huruf yang sudah diletakkan ke papan kembali ke rak pemain.
7. Membaca papan, yaitu proses membaca keadaan papan pada saat tertentu. Keadaan papan yang dilihat adalah huruf yang dimiliki masing-masing kotak atau kotak yang masih kosong. Proses membaca papan ini digunakan untuk mencari kemungkinan peletakan huruf ke papan permainan untuk membentuk kata.
8. Cek kamus, yaitu proses untuk mengecek bahwa huruf yang baru disusun pada papan membentuk kata yang memiliki arti dan sesuai dengan kamus yang ada. Jika tidak membentuk kata yang sesuai, pemain diminta untuk menyusun kembali atau mengambil pilihan lainnya yaitu Lewat.
9. Menghitung poin, yaitu proses untuk menjumlahkan poin pemain pada setiap giliran.
10. Cari kata terbaik, yaitu proses untuk mencari kemungkinan kata terbaik yang dapat dibentuk dari huruf yang dimiliki dan huruf-huruf pada papan yang ada pada giliran tertentu. Proses ini memiliki sub-proses seperti membaca papan dan merandom huruf pada papan yang akan digunakan.
11. Proses Genetika, yaitu proses dilakukannya pencarian kata optimum yang dapat dibentuk menggunakan algoritma genetika. Proses ini memiliki beberapa subproses utama yaitu set populasi awal, evaluasi (menghitung nilai *fitness*), seleksi, *crossover*, mutasi, cetak kromosom dan set populasi baru.

a. Set populasi awal, merupakan proses inisialisasi populasi awal yang merupakan sejumlah kromosom dengan nilai random antara 0 hingga panjang kromosom.
b. Evaluasi, merupakan proses menghitung nilai *fitness* kromosom.
c. Seleksi, merupakan proses memilih kromosom yang akan dilakukan proses *crossover*.
d. *Crossover*, merupakan proses penyilangan yang dilakukan untuk menghasilkan kromosom baru dengan sifat induknya.
e. Mutasi, merupakan proses mengubah nilai gen pada sebuah kromosom menjadi nilai lainnya.
f. Cetak kromosom, merupakan proses mengganti kromosom dari indeks (angka) menjadi huruf yang membentuk kata
g. Set populasi baru, merupakan proses mengisikan populasi baru dengan populasi lama yang terpilih dengan populasi hasil *crossover* dan mutasi.

12. Hitung *Edit Distance*, yaitu proses menghitung *edit distance* (selisih atau perbedaan) antara kata yang dimainkan dengan kata-kata di kamus.
13. Cek selesai, yaitu proses untuk mengecek selesainya permainan. Permainan akan dinyatakan selesai jika kedua pemain tidak dapat membentuk kata yang memiliki arti dari huruf-huruf random yang dimiliki atau jumlah stok huruf yang ada pada permainan telah habis.
14. Ganti pemain, yaitu proses untuk mengganti giliran pemain satu dengan pemain lainnya.

c. Analisis Kebutuhan Keluaran

Keluaran yang dihasilkan oleh sistem adalah tampilan permainan *scrabble* berupa representasi keadaan permainan *scrabble* pada setiap langkah (giliran) serta hasil akhir permainan.

d. Kebutuhan Antarmuka

Antarmuka perangkat lunak permainan *scrabble* dibuat menyesuaikan dengan pola permainan *scrabble* pada umumnya sehingga mudah untuk digunakan. Antarmuka dibuat sederhana dengan tujuan agar pemain tidak merasa terganggu dengan desain yang berlebihan selama permainan berlangsung. Antarmuka yang dibuat pada perangkat lunak permainan *scrabble* ini berupa tampilan menu awal, tampilan saat permainan serta tampilan cara bermain.

3.2 Perancangan Perangkat Lunak

Hasil analisis yang didapatkan kemudian dituangkan dalam rancangan perangkat lunak. Perancangan dibuat untuk mempermudah membangun perangkat lunak yang dibuat.

3.2.1 Metode Perancangan

Metode perancangan yang digunakan dalam perancangan perangkat lunak permainan *scrabble* dengan optimasi algoritma genetika ini adalah perancangan sistem berupa metode perancangan dengan menggunakan *flowchart* (diagram alir). *Flowchart* adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program.

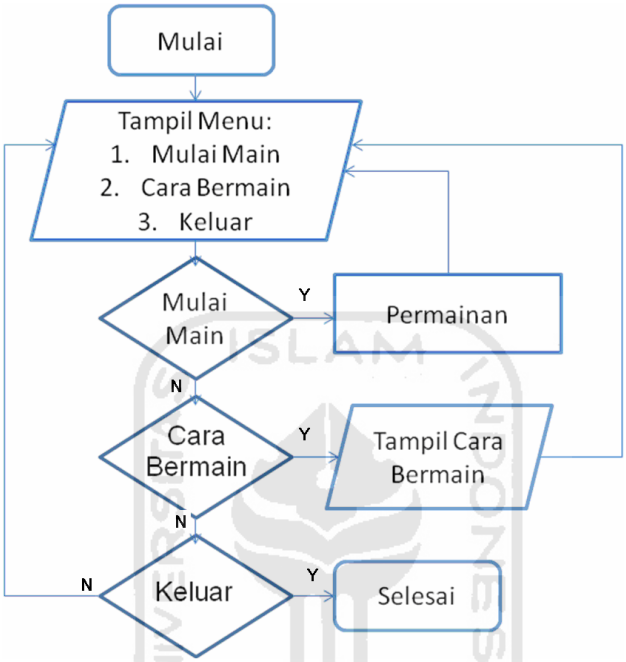
3.2.2 Hasil Perancangan

Hasil perancangan perangkat lunak permainan *scrabble* terdiri dari *Flowchart* (diagram alir) dan Rancangan Antarmuka. Penjelasan dari masing-masing hasil rancangan dapat dilihat sebagai berikut:

a. *Flowchart*

Flowchart menolong analis dan *programmer* untuk memecahkan masalah ke dalam segmen-segmen yang lebih kecil dan menolong daam menganalisis alternatif-alternatif lain dalam pengoperasian. *Flowchart* biasanya mempermudah penyelesaian

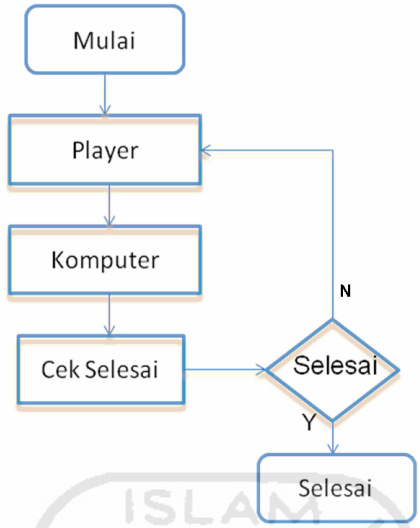
suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut. *Flowchart* permainan *scrabble* pada perangkat lunak dapat dilihat pada **Gambar 3.1**.



Gambar 3.1 *Flowchart* Perangkat Lunak Permainan *Scrabble*

Perangkat lunak yang akan dibuat diawali dari menampilkan menu permainan berupa mulai main, cara bermain dan pilihan keluar. Jika pengguna / pemain memilih menu mulai main, maka proses berikutnya yang dijalankan adalah proses permainan. Jika menu yang dilpilih adalah cara bermain maka perangkat lunak akan menampilkan cara bermain *scrabble* dengan perangkat lunak tersebut dan menu keluar digunakan untuk keluar dari perangkat lunak permainan *scrabble.*

Pada proses permainan, dibuat *flowchart* yang berbeda untuk lebih memperjelas proses permainan yang belangsung. *Flowchart* permainan tersebut dapat dilihat pada **Gambar 3.2**.

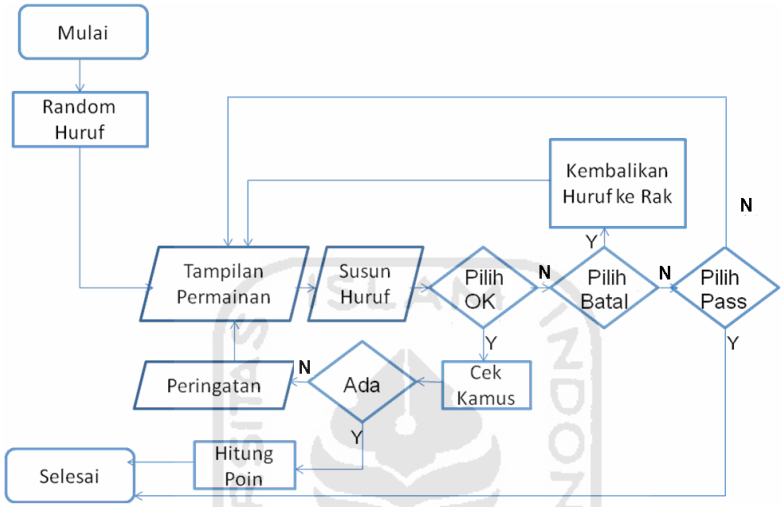


Gambar 3.2 *Flowchart* Permainan

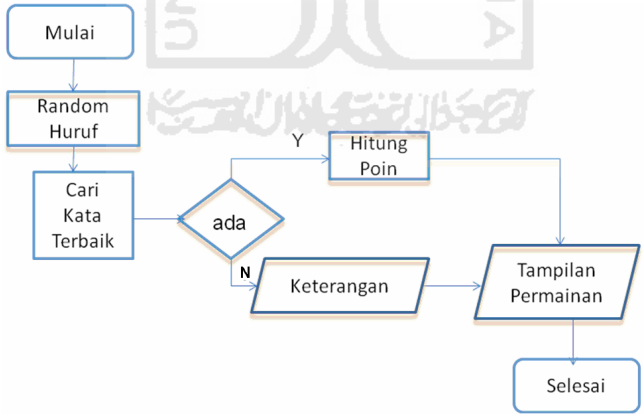
Permainan dimulai oleh *player* (*user*) terlebih dahulu. Setelah *player* menyelesaikan gilirannya, dilanjutkan oleh komputer untuk bermain. Proses berikutnya adalah cek selesai, yaitu proses pengecekan selesainya permainan oleh perangkat lunak. Permainan akan dinyatakan selesai jika kedua pemain tidak dapat membentuk kata yang memiliki arti dari huruf-huruf random yang dimiliki atau jumlah stok huruf yang ada pada permainan telah habis. Jika permainan belum selesai maka proses permainan akan diulang kembali oleh giliran bermain *player*. Proses bermain pada giliran *player* dilihat melalui *flowchart* pada **Gambar 3.3** sedangkan giliran komputer dapat dilihat melalui *flowchart* pada **Gambar 3.4**.

Giliran bermain *player* (pemain) diawali dengan proses random huruf. Proses ini merupakan proses merandom huruf yang belum dimainkan untuk diletakkan di rak pemain. Setelah didapatkan huruf-huruf random dapat dimainkan oleh pemain. Pemain dapat meletakkan huruf-huruf ke papan membentuk sebuah kata yang memiliki arti atau memilih menu-menu lain pada setiap permainan. Jika pemain memilih untuk membentuk kata, pemain dapat memilih tombol OK untuk mengecek huruf-huruf yang dimainkan. Jika kata yang dibentuk tidak terdapat di kamus, akan

muncul peringatan dan pemain mengulang gilirannya. Jika kata terdapat di kamus, dilakukan perhitungan poin dan giliran bermain *player* (pemain) selesai. Pemain dapat memilih menu batal untuk mengembalikan huruf yang sudah diletakkan ke papan kembali ke rak jika tidak jadi menggunakan huruf yang sudah diletakkan.



Gambar 3.3 *Flowchart* Permainan *Player*



Gambar 3.4 *Flowchart* Permainan Komputer

Giliran bermain komputer diawali dengan merandom huruf-huruf pada rak komputer. Proses berikutnya, komputer mencari kata terbaik yang bisa dibentuk dari huruf-huruf acak tersebut. Jika pada proses tersebut dapat dibentuk sebuah kata, maka

kata yang didapatkan dari proses pencarian akan dimainkan dan dilanjutkan dengan proses menghitung poin. Jika proses pencarian tidak dapat membentuk kata, maka giliran akan ada keterangan bahwa komputer tidak memainkan huruf-huruf pada raknya. Selanjutnya perangkat lunak akan menampilkan kondisi permainan dan giliran bermain komputer selesai.

Proses pencarian kata terbaik merupakan inti dari penggunaan algoritma genetika dalam perangkat lunak permainan *scrabble* yang dibuat. *Flowchart* proses pencarian kata terbaik dapat dilihat pada **Gambar 3.5**.

Gambar 3.5 *Flowchart* Pencarian Kata Terbaik

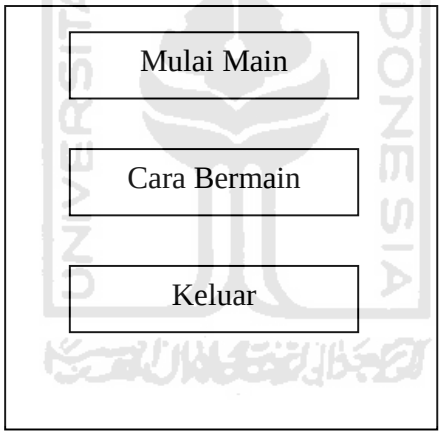
Pencarian kata terbaik diawali dengan membaca kondisi papan permainan. Perangkat lunak akan menyimpan *list* huruf-huruf yang dapat digunakan untuk membuat kata. Huruf-huruf yang dapat digunakan pada papan hanya huruf yang memiliki kotak kosong di sekitarnya. Huruf di papan tersebut kemudian digabungkan dengan huruf rak pemain yang mencari kata terbaik pada gilirannya. Dari huruf-huruf tersebutlah dilakukan proses algoritma genetika pada perangkat lunak. Setelah proses algoritma genetika dilakukan, akan didapatkan hasil pencarian. Hasil pencarian dapat berupa sebuah kata atau menyatakan tidak dapat membentuk kata pada giliran tersebut. Proses algoritma genetika pada perangkat lunak permainan *scrabble* akan dijelaskan bada subsubbab berikutnya.

b. Rancangan Antarmuka

Rancangan antarmuka dari perangkat lunak permainan *scrabble* dibuat sederhana agar pemain tidak terganggu dengan desain perangkat lunak yang berlebihan selama bermain. Rancangan antarmuka perangkat lunak permainan *scrabble* adalah sebagai berikut:

1. Tampilan Menu Awal

   Tampilan menu awal merupakan tampilan pertama kali perangkat lunak permainan *scrabble* dijalankan. Pada tampilan ini terdapat tiga pilihan menu yaitu mulai main, cara bermain dan keluar. Rancangan tampilan menu awal dapat dilihat pada **Gambar 3.6**.



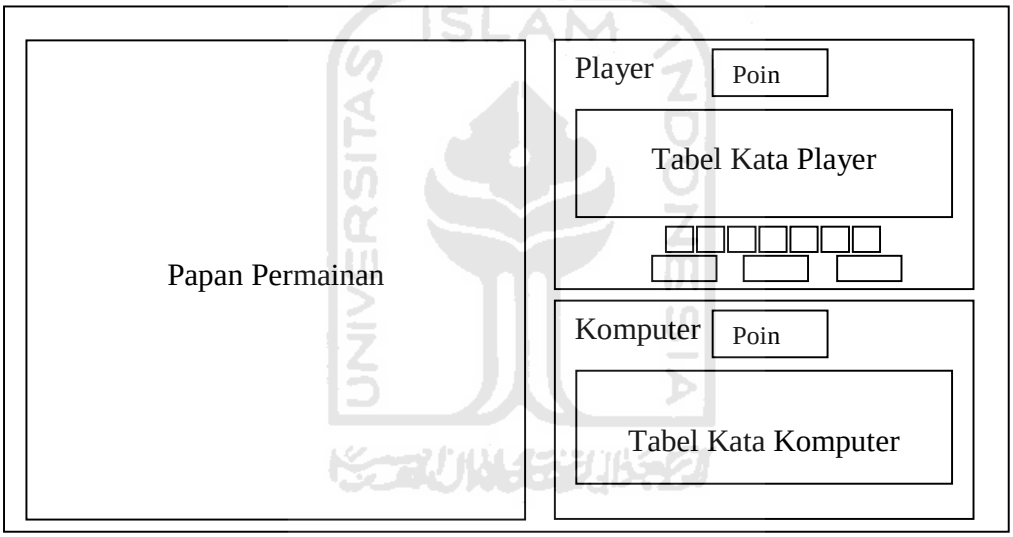
Gambar 3.6 Rancangan Tampilan Menu Awal

2. Tampilan Saat Permainan

   Tampilan saat permainan akan ditampilkan jika pada menu awal pemain / pengguna memilih menu mulai main. Tampilan ini akan terus ditampilkan selama permainan masih berlangsung. Rancangan tampilan saat permainan dapat dilihat pada **Gambar 3.7**.

   Tampilan saat permainan di sebelah kiri berisi papan permainan dengan balok-balok huruf di atas papan sesuai dengan kondisi saat permainan berlangsung.

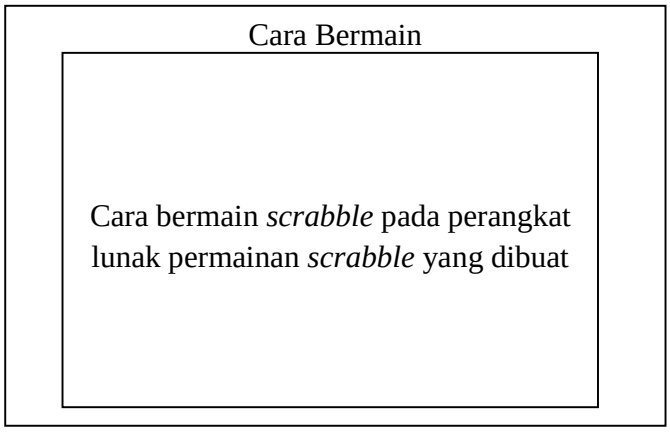
Untuk bagian kanan tampilan, dibagi menjadi dua, bagian atas untuk menyimpan data-data *player* dan bagian bawah untuk menyimpan data-data komputer selama permainan berlangsung. Masing-masing pemain memiliki *field* yang menyimpan total poin yang dimiliki dan tabel kata. Pada tabel kata akan disimpan kata-kata yang dimainkan oleh masing-masing pemain selama permainan berlangsung. Untuk bagian *player* terdapat susunan huruf random yang dimiliki pada setiap giliran. Selain itu juga terdapat tombol-tombol untuk menentukan langkah *player* di setiap gilirannya.



Gambar 3.7 Rancangan Tampilan Saat Permainan

3. Tampilan Cara Bermain

   Tampilan ini akan ditampilkan jika pemain / pengguna memilih menu cara bermain pada menu awal. Tampilan ini berisi cara bermain *scrabble* pada perangkat lunak yang dibuat. Rancangan tampilan cara bermain dapat dilihat pada **Gambar 3.8**.



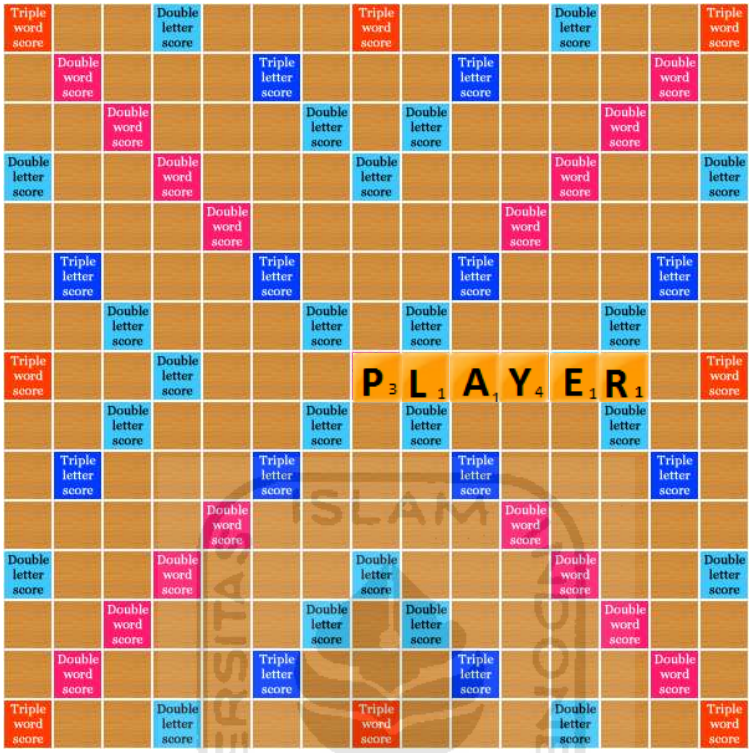
Gambar 3.8 Rancangan Tampilan Cara Bermain

3.2.3 Perancangan Pemodelan Algoritma Genetika dalam Permainan *Scrabble*

Algoritma Genetika dapat digunakan dalam permainan *scrabble*. Penggunaan algoritma genetika dalam permainan *scrabble* terdapat pada proses optimasi pencarian kata yang dapat dimainkan. Berikut ini adalah pemodelan Algoritma Genetika dalam permainan *scrabble*:

a. Representasi Kromosom

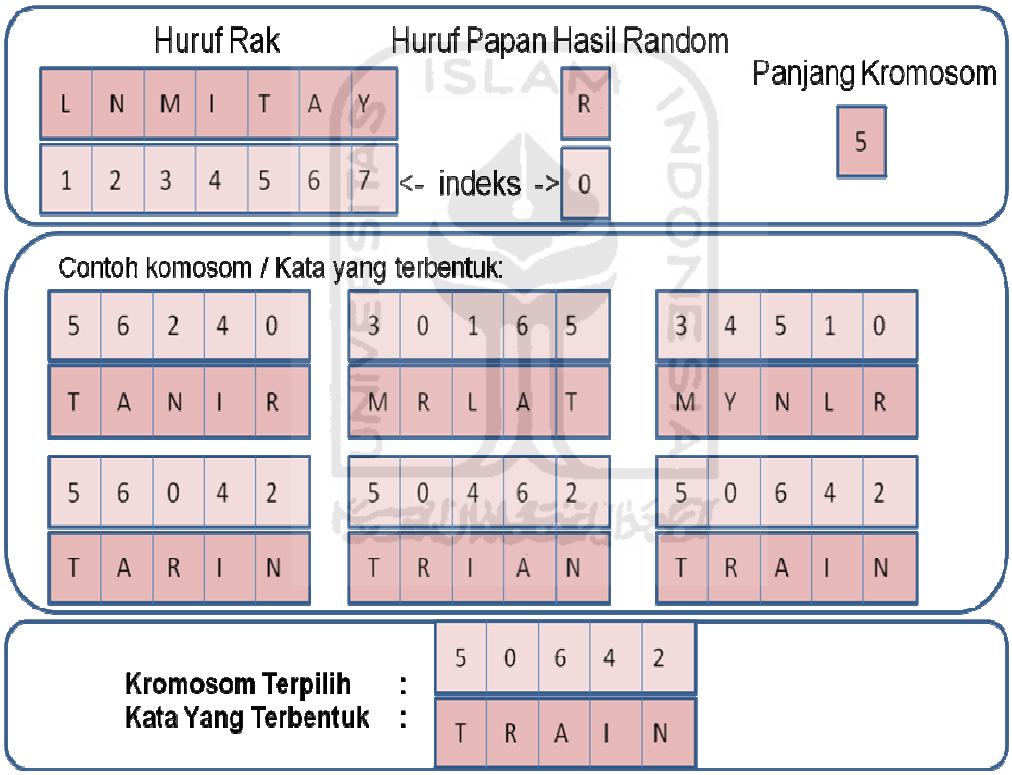
Pada komponen representasi kromosom, kromosom terbentuk dari gabungan huruf-huruf random yang dimiliki pemain dengan huruf yang ada pada papan permainan. Sebagai contoh, kondisi pada papan permainan saat pencarian kata terbaik dilakukan adalah seperti pada **Gambar 3.9**.



Gambar 3.9 Contoh Kondisi Papan Permainan

Pada saat kondisi papan permainan seperti gambar di atas, dimisalkan huruf random yang dimiliki komputer adalah L, N, M, I, T, A dan Y. Representasi kromosom yang terbentuk adalah gabungan dari indeks huruf-huruf random yang dimiliki komputer dengan indeks huruf pada papan permainan yang dapat dimainkan. Huruf pada papan permainan yang dapat dimainkan adalah huruf yang memiliki kotak-kotak kosong di sekitarnya sehingga dapat membentuk sebuah kata dengan cara menyusun huruf-huruf lain yang ada pada rak ke papan permainan tersebut. Panjang kata yang terbentuk didapatkan dari statistik tertinggi panjang kata yang terbanyak terdapat pada kamus. Statistik panjang kata terbanyak adalah kata dengan banyak huruf 7. Jika pada panjang kata terbanyak komputer tidak dapat membentuk kata maka dilakukan pencarian ulang pada statistik panjang kata terbanyak berikutnya yaitu banyak huruf 6, banyak huruf 5,

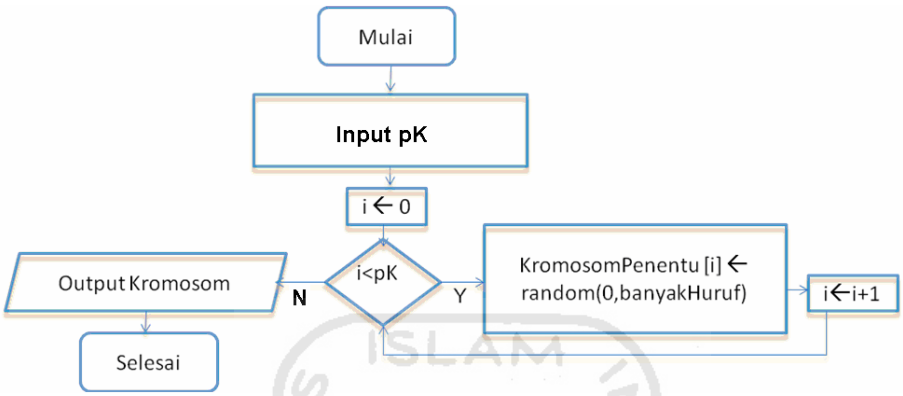
banyak huruf 4 dan terakhir banyak huruf 3. Sehingga pencarian dimulai dengan panjang kromosom 7, jika tidak ditemukan hingga panjang kromosom 3 maka komputer tidak dapat menemukan kata yang dapat dibentuk pada giliran tersebut. Sebagai contoh, huruf pada papan yang dapat dimainkan adalah P, L, A, Y, E dan R. Hingga panjang kromosom 6, komputer tidak dapat menemukan kata yang terbentuk. Selanjutnya pada saat panjang kromosom adalah 5, komputer dapat menemukan kata yang dapat dibentuk. Gambaran proses representasi kromosom pada saat panjang kromosom 5 dapat dilihat pada **Gambar 3.10**.



Gambar 3.10 Gambaran Representasi Kromosom

Contoh representasi kromosom di atas merupakan gabungan indeks huruf random pada rak komputer dengan huruf R yang terdapat pada papan permainan (diberikan indeks 0) dengan panjang kromosom 5. Pada contoh representasi kromosom di atas, individu (kata) yang terbentuk adalah TANIR, MRLAT,

MYNLR, TARIN, TRIAN dan TRAIN. *Flowchart* representasi kromosom dapat dilihat pada **Gambar 3.11**.

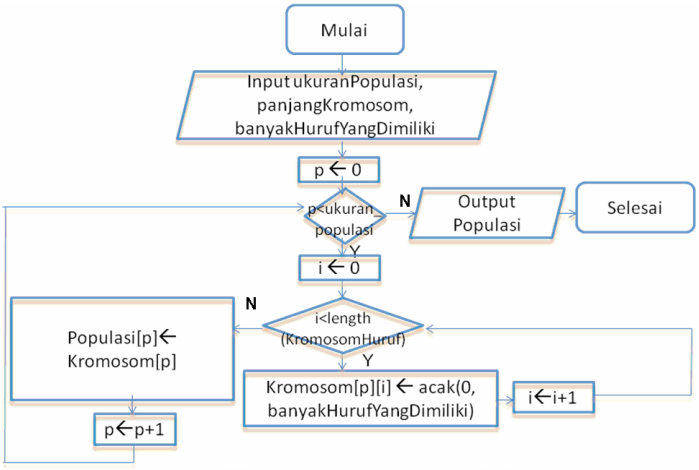


Gambar 3.11 *Flowchart* Representasi Kromosom

Kromosom berupa nilai random berkisar antara 0 hingga banyak huruf yang dimiliki komputer dengan panjang kromosom didapatkan dari statistik panjang kata terbanyak pada kamus. Indeks 0 merupakan indeks untuk huruf pada papan yang dimainkan, sedangkan indeks 1 dan seterusnya merupakan indeks yang dimiliki oleh huruf pada rak.

b. Inisialisasi

Proses inisialisasi populasi pada perangkat lunak permainan *scrabble* membentuk sejumlah kromosom pada satu populasi merupakan rangkaian angka acak dengan panjang tertentu. Panjang kromosom ditentukan berdasarkan statistik panjang kata terbanyak pada kamus. Setelah didapatkan panjang kromosom dan ukuran populasi (jumlah kromosom pada satu populasi), dilakukan pengacakan nilai 0-banyak huruf yang dimiliki sebanyak ukuran populasi. *Flowchart* untuk tahap inisialisasi dapat dilihat pada **Gambar 3.12**.



Gambar 3.12 *Flowchart* Inisialisasi

c. Evaluasi

Tahap evaluasi pada algoritma genetika dilakukan terhadap fungsi objektif (fungsi tujuan) dan konversi fungsi objektif kedalam fungsi *fitness*. Pada permainan *scrabble*, fungsi objektif ditentukan melalui tingkat kemiripan kata yang terbentuk dengan setiap kata pada kamus. Tingkat kemiripan didapatkan dari nilai *edit distance* (selisih kata) dari kata yang terbentuk dengan setiap kata pada kamus dibagi panjang kata dalam persen. Tingkat kemiripan kata yang lebih tinggi memiliki nilai objektif yang lebih tinggi. Fungsi objektif pada permainan *scrabble* dapat dilihat pada persamaan 3.1.

$$h(x) = \max\left(\frac{ED_i(kataTerbentuk, kataKamus_i)}{panjangKata_i * 100} \right) \mid i=1..n_{kamus} \quad \text{..... (3.1)}$$

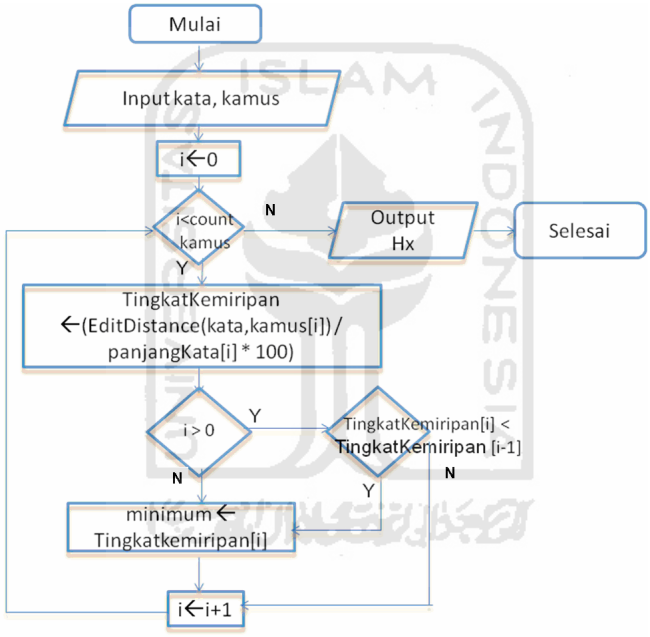
dimana i adalah indeks kata pada kamus, n_{kamus} adalah banyaknya kata pada kamus dan $panjangKata_i$ adalah maksimum anatara panjang kata yang terbentuk dan panjang kata kamus.

Pada proses seleksi algoritma genetika menggunakan roda *roulette* bersifat memaksimalkan optimasi dan tujuan yang dicapai sama, yaitu memaksimalkan

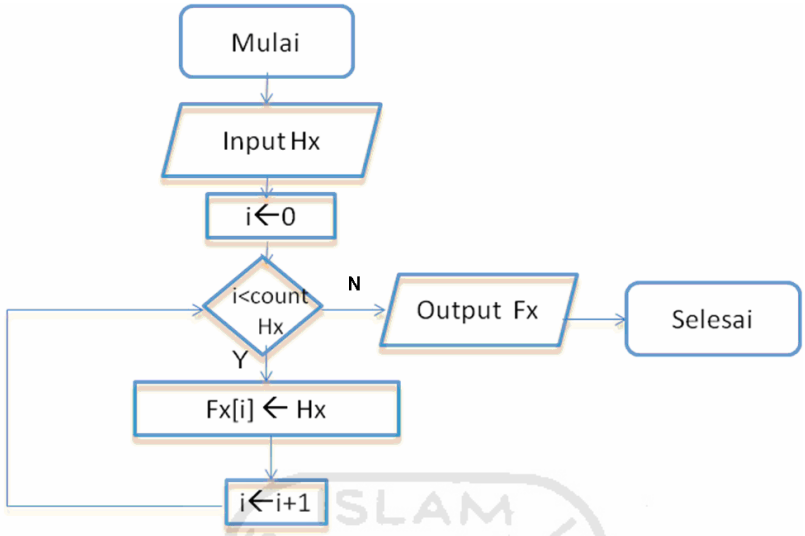
tingkat kemiripan kata yang terbentuk dengan kata pada kamus. Sehingga fungsi *fitness* yang ada sesuai dengan fungsi objektif seperti yang terlihat pada persamaan 3.2.

$$f(x) = h(x) \qquad \text{.....(3.2)}$$

Flowchart untuk fungsi objektif dan fungsi *fitness* dapat dilihat pada **Gambar 3.13** dan **Gambar 3.14**.



Gambar 3.13 *Flowchart* Fungsi Objektif

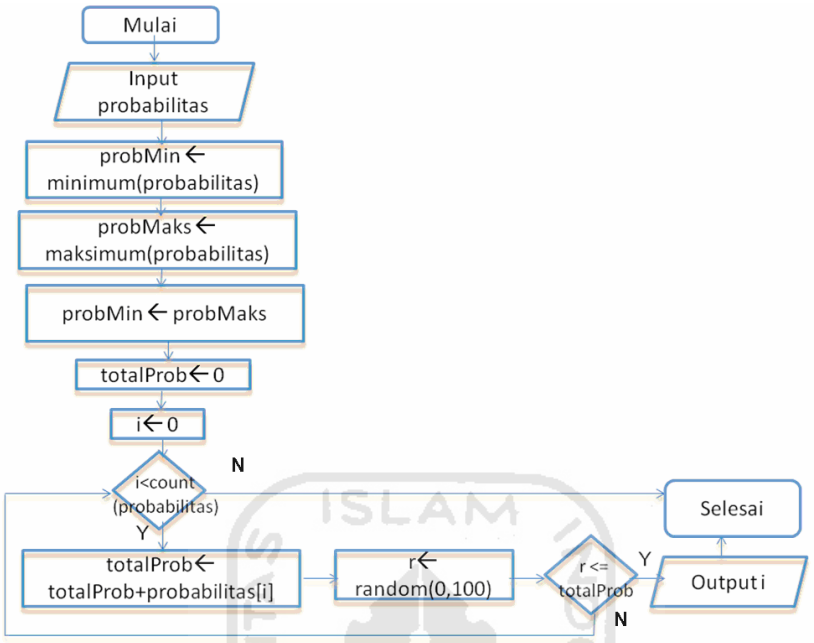


Gambar 3.14 *Flowchart* Fungsi *Fitness*

d. Seleksi

Tahap seleksi digunakan untuk memilih kromosom yang baik dengan nilai *fitness* tinggi dan mengeliminasi kromosom-kromosom yang jelek. Pada perangkat lunak permainan *scrabble*, metode seleksi yang digunakan adalah kombinasi metode elitis dengan metode roda *roulette*. Metode elitis akan menggantikan kromosom dengan nilai *fitness* terjelek dari sebuah populasi dengan kromosom yang memiliki nilai *fitness* terbaik pada generasi tersebut. Sedangkan metode roda *roulette* yang digunakan akan memilih kromosom dengan *fitness* yang tinggi.

Pada metode roda *roulette*, masing-masing kromosom memiliki nilai probabilitas yang didapat dari perbandingan nilai *fitness* masing-masing kromosom dengan total nilai *fitness* seluruh kromosom pada satu populasi. Selanjutnya dilakukan random nilai untuk memilih kromosom, sehingga terpilih kromosom-kromosom dengan nilai *fitness* yang tinggi. *Flowchart* proses seleksi dapat dilihat pada **Gambar 3.15**.



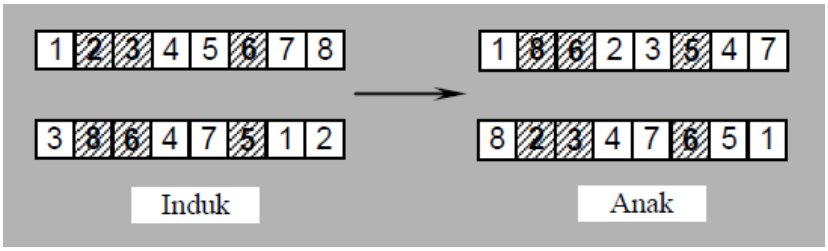
Gambar 3.15 *Flowchart* Proses Seleksi

e. Penyilangan (*Crossover*)

Penyilangan dalam algoritma genetika bertujuan untuk melahirkan kromosom baru yang mewarisi sifat-sifat induknya sebagaimana proses reproduksi yang terjadi dalam kehidupan alam (Zukhri, 2009). Proses penyilangan pada permainan *scrabble* menggunakan metode penyilangan berbasis posisi (*Position Based Crossover*).

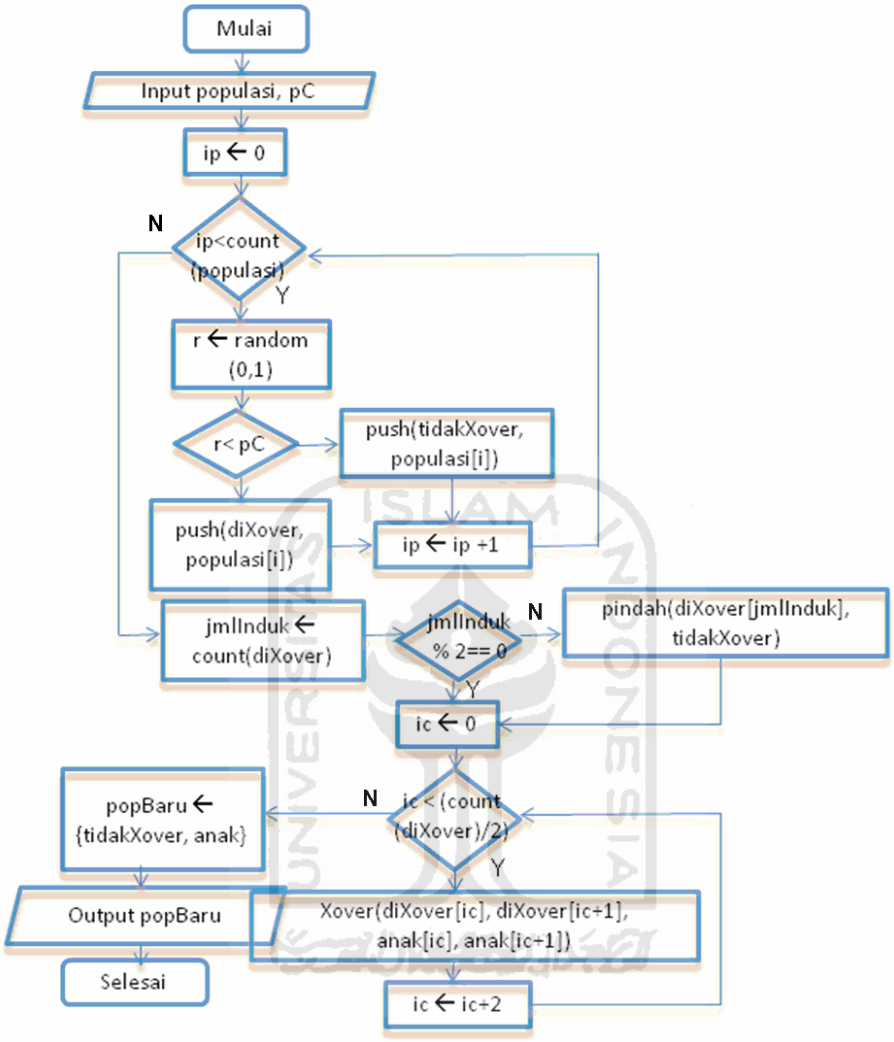
Dalam metode ini dipilih sejumlah posisi gen secara acak (pada perangkat lunak permainan *scrabble* dibatasi hanya memilih dua posisi gen secara acak), kemudian gen-gen pada posisi terpilih pada induk yang pertama diwariskan kepada kromosom anak yang kedua, sedangkan gen-gen lainnya dari kromosom anak yang kedua diambil daripada gen-gen induk yang kedua dengan urutan yang sama. Demikian juga sebaliknya, gen-gen pada posisi terpilih pada induk yang kedua diwariskan kepada kromosom anak yang pertama, sedangkan gen-gen lainnya dari kromosom anak yang pertama diambil daripada gen-gen induk yang

pertama dengan urutan yang sama. Untuk lebih jelasnya dapat dilihat ilustrasi metode penyilangan dalam **Gambar 3.16**.



Gambar 3.16 Ilustrasi Metode Penyilangan Berbasis Posisi

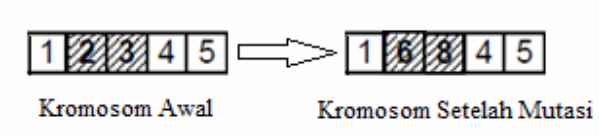
Flowchart proses penyilangan (*crossover*) pada perangkat lunak permainan scrabble dapat dilihat pada **Gambar 3.17**. Kromosom yang dijadikan induk merupakan kromosom-kromosom yang memiliki probabilitas lebih kecil dari probabilitas *crossover* yang ditetapkan sebagai parameter algoritma genetika. Jika induk yang terpilih berjumlah ganjil, maka induk kromosom yang terakhir dihapus, atau tidak dilakukan proses penyilangan, sehingga jumlah kromosom yang tidak dilakukan penyilangan dan anak kromosom yang dihasilkan sama dengan jumlah ukuran populasi di awal proses genetika.



Gambar 3.17 *Flowchart* Proses Penyilangan

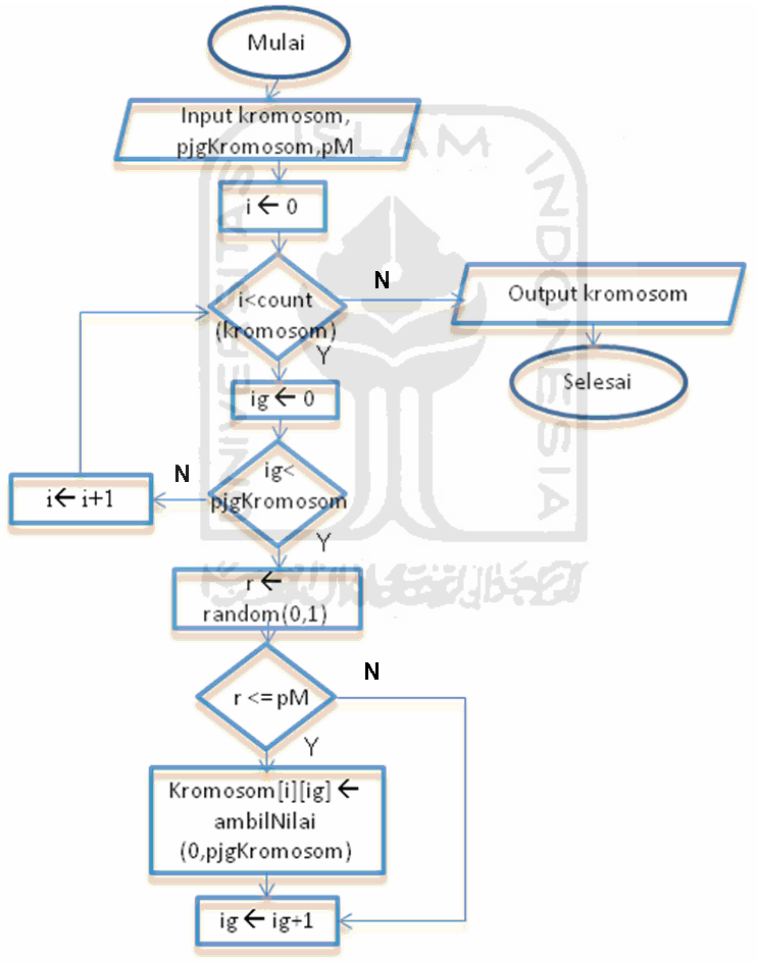
f. Mutasi

Mutasi dalam algoritma genetika bertujuan untuk mengubah gen-gen tertentu dari sebuah kromosom. Mutasi dalam perangkat lunak permainan *scrabble* akan mengubah nilai suatu gen terpilih menjadi nilai gen lain yang tidak ada dalam satu kromosom. Untuk lebih jelasnya, dapat dilihat ilustrasi metode mutasi dalam **Gambar 3.18**.



Gambar 3.18 Ilustrasi Proses Mutasi

Flowchart proses mutasi pada perangkat lunak permainan *scrabble* dapat dilihat pada **Gambar 3.19**.



Gambar 3.19 *Flowchart* Proses Mutasi

Bab selanjutnya akan membahas mengenai implementasi dan analisis kinerja perangkat lunak serta simulasi permainan *scrabble* pada perangkat lunak. Pada bab tersebut dilakukan pengujian normal dan pengujian tidak normal. Pembahasan meliputi batasan implementasi perangkat lunak, pengujian perangkat lunak, proses Algoritma Genetika pada perangkat lunak, waktu kompilasi perangkat lunak yang dibangun, kekurangan dan kelebihan perangkat lunak yang dibangun.

BAB IV
IMPLEMENTASI DAN ANALISIS KINERJA PERANGKAT LUNAK

4.1 Implementasi Perangkat Lunak

Implementasi perangkat lunak bertujuan untuk memastikan perangkat lunak yang dibuat dapat bekerja dan sesuai dengan tujuan dibangun. Perangkat lunak dinilai baik jika bebas dari kesalahan. Kesalahan yang mungkin terjadi antara lain kesalahan tampilan ataupun kesalahan proses. Untuk mengetahui kesalahan yang mungkin terdapat pada perangkat lunak, dilakukan pengujian dengan cara mengoperasikan perangkat lunak.

4.1.1 Batasan Implementasi Perangkat Lunak

Perangkat lunak yang dibangun memiliki beberapa batasan implementasi, yaitu:

1. Parameter algorimta genetika dalam perangkat lunak masih bersifat statis.
2. Kamus yang digunakan dalam permainan *scrabble* masih bersifat statis.
3. Perangkat lunak yang dibangun tidak mengimplementasikan beberapa elemen dalam permainan *scrabble.* Elemen yang tidak diimplementasikan terebut adalah:
    a. Huruf *blank* pada permainan *scrabble* pada umumnya dapat digunakan dan diubah menjadi huruf apa saja ketika dimainkan dengan poin 0. Pada perangkat lunak yang dibangun tidak memiliki huruf *blank*.
    b. Pemain dapat melakukan tukar huruf sebanyak satu kali pada setiap gilirannya. Pada perangkat lunak yang dibangun, tidak terdapat fitur mengganti huruf oleh setiap pemain pada setiap gilirannya.
    c. Perangkat lunak yang dibangun tidak dapat mengimplementasikan kasus permainan *scrabble* seperti skenario yang dibahas pada bab 2 pada **Gambar 2.3**, **Gambar 2.4** dan **Gambar 2.5**.

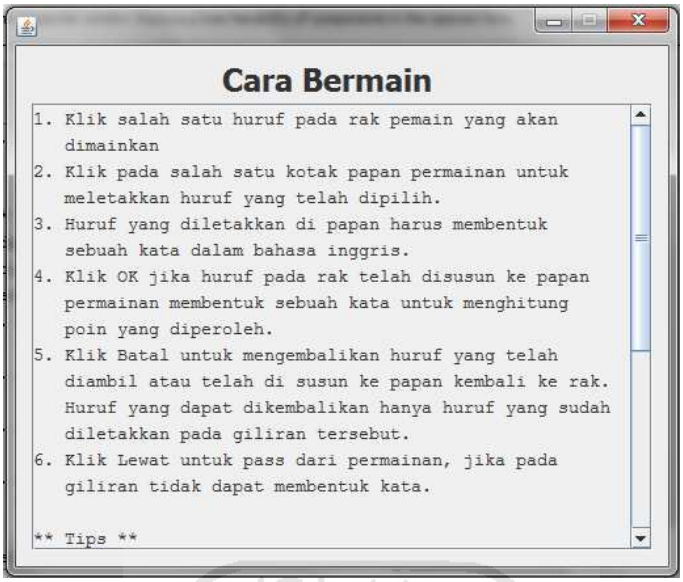
4.1.2 Tampilan Menu Awal

Tampilan ini merupakan tampilan yang muncul pertama kali perangkat lunak dijalankan. Seperti pada rancangan, pada tampilan menu awal ini terdapat menu mulai bermain, melihat cara bermain dan keluar dari permainan. Pada **Gambar 4.1** dapat dilihat tampilan awal pada perangkat lunak permainan *scrabble*.

Gambar 4.1 Tampilan Menu Awal

4.1.3 Tampilan Cara Bermain

Setelah menu awal permainan muncul, pemain dapat memilih menu Cara Bermain untuk membaca petunjuk permainan *scrabble* pada perangkat lunak yang dibangun. Tampilan cara bermain dapat dilihat pada **Gambar 4.2**.



Gambar 4.2 Tampilan Cara Bermain

4.1.4 Tampilan Awal Permainan

Selain menu Cara Bermain, pada tampilan menu awal terdapat pilihan Mulai Main untuk memulai permainan. Tampilan Awal Permainan yang akan muncul pertama setelah menu Mulai Main dipilih dapat dilihat pada **Gambar 4.3**. Tampilan Awal Permainan akan langsung menampilan huruf-huruf acak yang dimiliki oleh pemain. Permainan dimulai oleh pemain untuk membentuk sebuah kata dalam Bahasa Inggris menggunakan huruf-huruf acak yang terdapat pada rak.

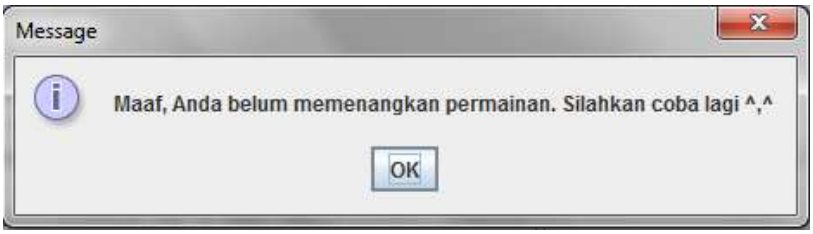
Pada Tampilan Awal Permainan, warna kolom milik *player* akan lebih terang, sedangkan warna kolom milik komputer akan lebih gelap. Namun setelah dilakukan pergantian pemain, maka warna kolom akan saling berganti dimana warna kolom yang lebih cerah menandakan pemain tersebut pada posisi aktif dalam giliran bermain.

Gambar 4.3 Tampilan Awal Permainan

4.1.5 Tampilan Akhir Permainan

Di akhir permainan, perangkat lunak akan menampilkan keterangan pemenang dalam permainan. Tampilan akhir permainan *scrabble* pada perangkat lunak yang dibuat dapat dilihat pada **Gambar 4.4** dan **Gambar 4.5**.

Gambar 4.4 Tampilan Akhir Permainan Jika Menang



Gambar 4.5 Tampilan Akhir Permainan Jika Kalah

4.2 Analisis Kinerja Perangkat Lunak

Analisis kinerja perangkat lunak merupakan proses dimana perangkat lunak yang dibuat dioperasikan pada tahap yang sebenarnya, sehingga dapat diketahui apakah perangkat lunak yang dibuat sesuai dengan yang direncanakan. Pada tahap ini juga dilakukan beberapa pengujian untuk penanganan kesalahan yang terdapat dalam perangkat lunak.

4.2.1 Pengujian Normal

Pengujian normal dilakukan dengan menjalankan perangkat lunak sesuai dengan ketentuan, sehingga dapat dilihat apakah perangkat lunak berjalan dengan baik sesuai dengan yang diharapkan.

a. Pengujian membentuk sebuah kata inggris

Dalam permainan, setelah pemain menyususn huruf-huruf acak yang dimiliki ke papan permainan, perangkat lunak akan membaca kata dan mencocokkan kata yang terbentuk dengan kata-kata yang ada pada kamus. Jika kata yang terbentuk terdapat pada kamus, maka proses selanjutnya adalah menghitung poin yang dihasilkan dari kata yang terbentuk tadi. Tampilan jika pemain berhasil menyusun huruf-huruf acak yang dimiliki menjadi sebuah kata dalam bahasa Inggris yang terdapat pada kamus dapat dilihat pada **Gambar 4.6**.

Gambar 4.6 Pengujian Membentuk Sebuah Kata Inggris

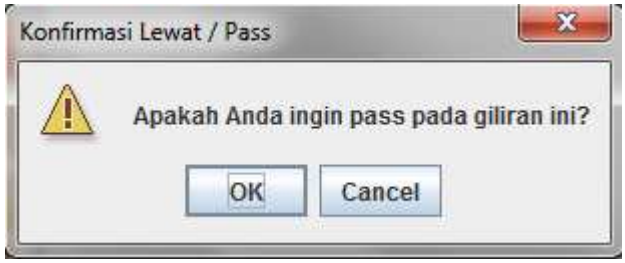
b. Pengujian membatalkan peletakan huruf

Dalam permainan, jika pemain telah melektakkan huruf-huruf pada papan, namun ingin mengembalikan huruf-huruf yang telah disusun ke rak, maka pemain cukup memilih tombol Batal di bawah rak pemain. Jika tombol Batal dipilih, maka huruf-huruf yang telah disusun akan kembali ke rak. Tampilan jika pemain membatalkan peletakan huruf dapat dilihat pada **Gambar 4.7**.

Gambar 4.7 Pengujian Membatalkan Peletakan Huruf

c. Pengujian memilih menu lewat

Dalam permainan, jika pemain tidak dapat membenuk kata menggunakan huruf-huruf acak yang dimiliki, maka pemain dapat memilih Lewat. Pilihan ini berarti pemain tidak memainkan kata pada gilirannya dan giliran permainan selanjutnya untuk bermain adalah komputer. Tampilan jika pemain memilih untuk Lewat pada gilirannya dapat dilihat pada **Gmbar 4.8**.



Gambar 4.8 Konfirmasi Pengujian Menu Lewat

4.2.2 Pengujian Tidak Normal

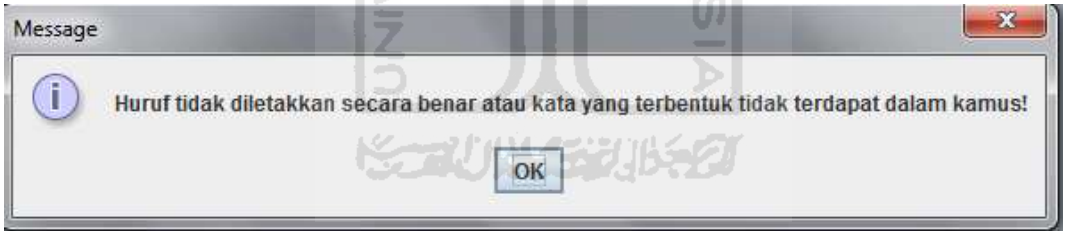
Pengujian tidak normal dilakukan dengan menjalankan perangkat lunak tidak sesuai dengan ketentuan, sehingga akan memunculkan pesan kesalahan yang memberitahukan kepada pengguna agar menjalankan perangkat lunak sesuai dengan ketentuan yang ada.

a. Pengujian meletakkan huruf tidak teratur

Dalam permainan, pemain diharuskan meletakkan huruf acak pada papan secara sejajar sama baris atau sama kolom untuk membentuk sebuah kata. Jika dalam permainan pemain menyusun huruf tidak sama baris atau sama kolom, maka akan muncul peringatan seperti pada **Gambar 4.9**.

b. Pengujian meletakkan huruf yang tidak memiliki arti

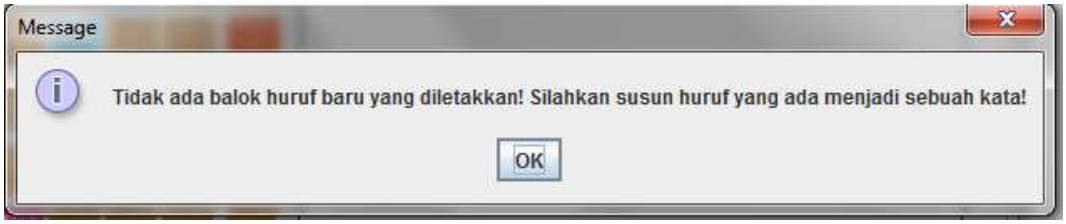
Huruf-huruf acak yang dimiliki pemain harus disusun membentuk sebuah kata dalam bahasa inggris. Jika huruf yang dimiliki disusun namun tidak memiliki arti (bukan sebuah kata dalam bahasa Inggris), maka akan muncul peringatan seperti pada **Gambar 4.9**.



Gambar 4.9 Pengujian Huruf Tidak Teratur atau Kata Tidak Memiliki Arti

c. Pengujian Memilih tombol OK sebelum meletakkan huruf ke papan

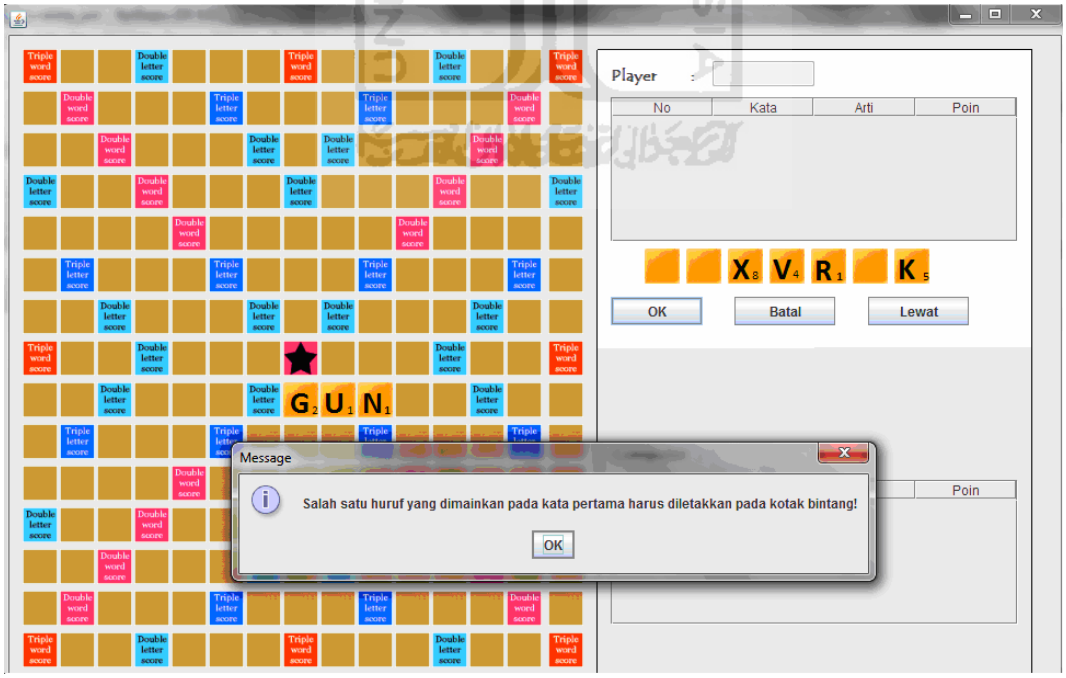
Jika pemain memilih tombol OK sebelum meletakkan huruf ke papan, maka tidak ada poin yang akan dihitung dan pemain akan diminta kembali untuk menyusun kata. Tampilan jika pemain memilih tombol OK sebelum meletakkan huruf ke papan dapat dilihat pada **Gambar 4.10**.



Gambar 4.10 Pengujian Memilih Tombol OK Sebelum Meletakkan Huruf ke Papan

d. Pengujian kata pertama yang dimainkan tidak diletakkan pada kotak bintang

Dalam permainan *scrabble*, untuk kata pertama yang dimainkan, salah satu huruf harus diletakkan di kotak bintang pada papan permainan. Kotak bintang adalah kotak paling tengah pada papan permainan. Kata pertama yang dimainkan akan mendapatkan tambahan poin 50 untuk pemain yang memainkannya. Tampilan jika pada kata pertama yang dimainkan, tidak ada huruf yang diletakkan pada kotak bintang papan permainan dapat dilihat pada **Gambar 4.11**.



Gambar 4.11 Pengujian Kata Pertama pada Permainan

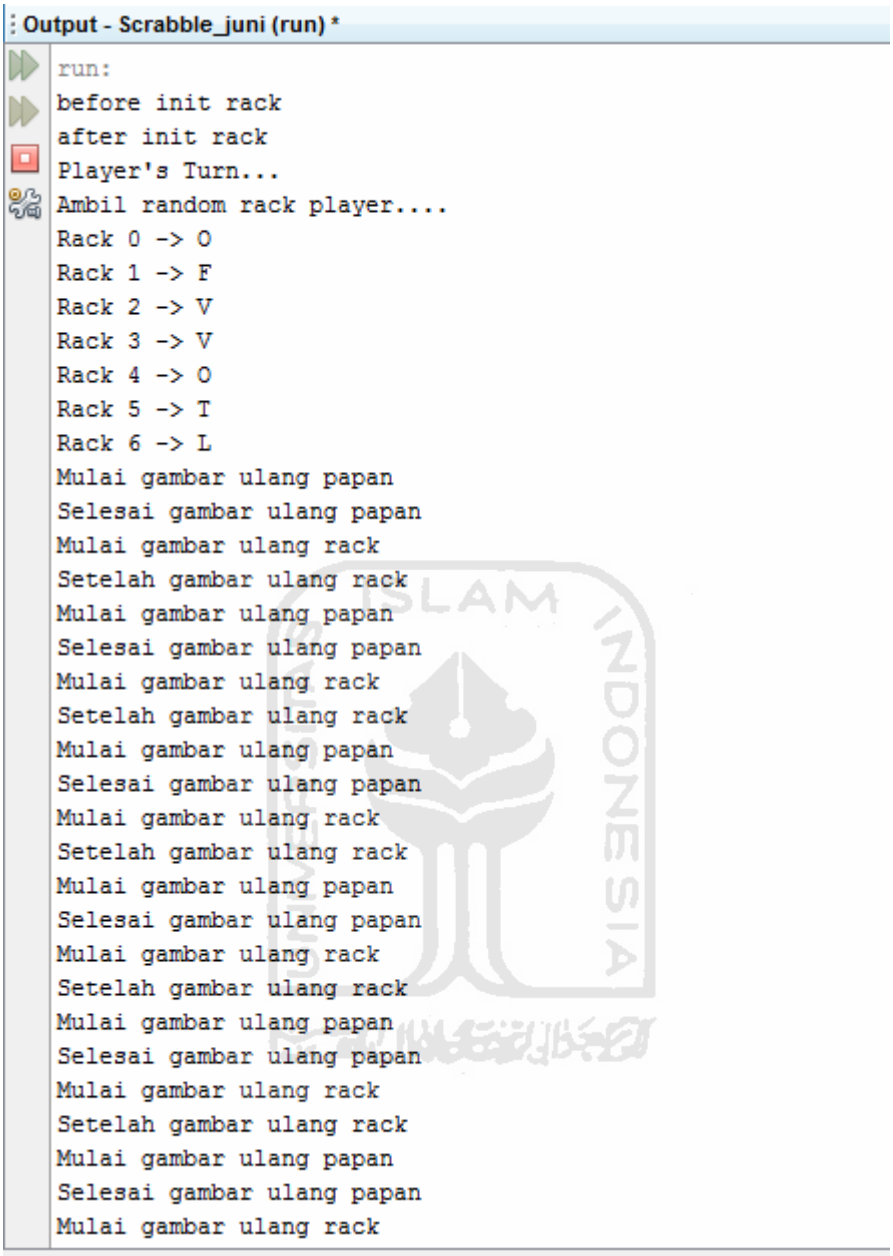
4.3 Algoritma Genetika dalam Perangkat Lunak

Pada subbab ini akan dibahas mengenai algoritma genetika yang terdapat dalam perangkat lunak yang dibuat. Perangkat lunak yang dibangun menggunakan beberapa metode pada operasi genetika. Untuk operasi seleksi, perangkat lunak menggunakan metode roda rolet dan elitis. Penyiangan (*crossover*) dilakukan dengan metode penyilangan berbasis posisi, dan mutasi menggunakan penggantian nilai gen dengan nilai yang belum ada.

4.3.1 Alur Proses Algoritma Genetika

Sebelum dilakukan proses genetika seperti pada umumnya, perangkat lunak melakukan beberapa proses untuk menyesuaikan kromosom yang akan dihasilkan dengan solusi yang tepat dengan keadaan permainan. Agar penjelasan alur proses algoritma lebih jelas, penjelasan akan disertakan dengan tampilan program berdasarkan simulasi permainan.

Dalam perangkat lunak permainan *scrabble* yang dibuat, *player* mendapatkan giliran pertama bermain. Masing-masing pemain mendapatkan tujuh huruf acak yang akan disusun membentuk sebuah kata pada kamus. Sebagai simulasi permainan, *player* pertama kali mendapatkan beberapa huruf acak seperti pada **Gambar 4.12**. *Player* kemudian mencoba membentuk kata dari huruf acak yang dimiliki dan melakukan pemindahan beberapa huruf dari rak ke papan permainan. Setiap pemindahan huruf dari rak ke papan permainan akan dilakukan proses menggambar ulang papan permainan sehingga pada jendela permainan terlihat huruf-huruf yang dipilih oleh *player* berpindah ke kotak papan permainan yang diinginkan. Proses di atas dapat dilihat pada **Gambar 4.12**.



Gambar 4.12 Huruf Acak Pemain pada Simulasi Permainan

Langkah yang diambil *player* pada gilirannya adalah menyusun huruf acaknya menjadi sebuah kata (VOLT) dengan proses seperti pada **Gambar 4.13**.

Gambar 4.13 Proses Langkah Pemain pada Simulasi Permainan

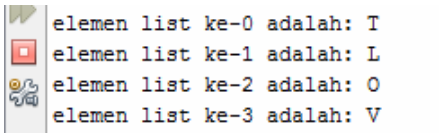
Berdasarkan proses seperti pada gambar di atas, maka tampilan permainan yang dihasilkan setelah *player* menyelesaikan giliran bermainnya akan terlihat seperti pada **Gambar 4.14**.

Gambar 4.14 Tampilan Permainan Setelah Langkah Pemain pada Simulasi Permainan

Selanjutnya dimulailah rangkaian proses yang dilakukan oleh perangkat lunak sebagai giliran main komputer. Proses pertama yang dilakukan adalah membaca papan permainan. Hal ini dilakukan untuk mengetahui huruf-huruf apa saja yang ada pada papan permainan serta huruf-huruf apa saja yang dapat digunakan untuk menjadi bagian dari kromosom solusi. Penyimpanan huruf-huruf papan yang dapat disambung ini dibutuhkan karena dalam permainan *scrabble*, kata baru yang diletakkan harus menempel dengan salah satu huruf pada kata yang sudah ada sebelumnya. Proses tersebut dapat dilihat seperti **Gambar 4.15**.

Proses selanjutnya adalah memberikan nilai panjang kromosom. Nilai awal panjang kromosom yang diberikan adalah 7, dimana 7 didapatkan dari statistik panjang kata terbanyak yang terdapat pada kamus. Jika pada panjang kromosom 7 tidak dapat ditemukan, maka proses genetika akan diulang dengan panjang kromosom diberikan nilai statistik panjang kata terbanyak berikutnya yaitu 6 dan seterusnya jika masih belum ditemukan hingga panjang kromosom 3. Jika hingga panjang kromosom 3 komputer tidak dapat membentuk kata, maka

proses pencarian dihentikan dengan hasil tidak dapat membentuk kata. Kode program untuk proses pemberian nilai panjang kromosom dan proses genetika dapat dilihat pada **Gambar 4.16** dan **Gambar 4.17**.



Gambar 4.15 Proses Penyimpanan Huruf Papan yang Dapat Disambung

```
panjangKromosom=7;
while(panjangKromosom>2 && !ketemu)
{
    ketemu=genetika();
    if(!ketemu)
    {
        panjangKromosom--;
    }
}
```

Gambar 4.16 Kode Program Mengacak Panjang Kromosom

```
public boolean genetika()
{
    boolean ketemu=false;
    int iterasi=0;
    float fitness=0;
    //set populasi awal
    setPopulasiAwal();
    fitness=nilaiFitness(populasiKromosom);
    if(fitness!=1)
    {
        iterasi++;
        while(!ketemu && iterasi<jumlahGenerasi)
        {
            iterasi++;
            crossOver();
            seleksiMutasi();
            setPopulasiBaru();
            fitness=nilaiFitness(populasiBaru);
            if(maksimumFitnessGenerasi==1)
            {
                ketemu=true;
                CEK("kromosom terpilih: "+cetakKromosom(kromosomOptimum));
            }
        }
    }
    else
    {
        kromosomOptimum=cetakBalokKromosom(populasiKromosom[indeksMaksimumFitness]);
        CEK("kromosom terpilih: "+cetakKromosom(kromosomOptimum));
        ketemu=true;
    }

    return ketemu;
}
```

Gambar 4.17 Kode Program Proses Genetika

Perangkat lunak kemudian mengambil secara acak huruf papan yang akan digunakan yang telah disimpan pada proses pertama. Jika pada proses pertama tadi tidak terdapat satu pun huruf papan yang dapat disambung, maka proses mengambil huruf papan secara acak ini akan dilewatkan dan dilanjutkan ke proses selanjutnya. Kode program dan hasil proses pengambilan huruf papan ini dapat dilihat pada **Gambar 4.18** dan **Gambar 4.19**.

```
public BalokHuruf ambilBalokDiPapan()
{
    BalokHuruf balokHuruf = null;
    int panjangList = listHurufPapanDapatDisambung.countElement();
    if(panjangList !=0 )
    {
        int random = new Random().nextInt(panjangList);
        int index=0;
        Elemen elemen = listHurufPapanDapatDisambung.getFirst();

        while (elemen != null)
        {
            if(index == random)
            {
                balokYangDiambilDariPapan = (BalokDariPapan) elemen.getElemen();
                balokHuruf=balokYangDiambilDariPapan.getBalokHuruf();
            }
            elemen = elemen.getBerikut();
            index++;
        }
    }
    CEK("balok huruf dari papan yang digunakan: "+balokYangDiambilDariPapan.getBalokHuruf().getHuruf());
    return balokHuruf;
}
```

Gambar 4.18 Kode Program Pengambilan Huruf Papan

```
balok huruf dari papan yang digunakan: L
```

Gambar 4.19 Hasil Proses Pengambilan Huruf Papan

Setelah proses penyesuaian kromosom dilakukan, operasi genetika dapat dilakukan. Pada perangkat lunak ini, terdapat beberapa parameter algoritma genetika statis yang digunakan yaitu:

a. Jumlah Generasi : 10 generasi
b. Ukuran Populasi : 10 kromosom
c. Probabilitas *Crossover* : 0.6
d. Probabilitas Mutasi : 0.05

Representasi kromosom awal untuk perangkat lunak ini berupa angka acak 0 hingga banyak huruf yang dimiliki sebanyak panjang kromosom. Pada simulasi permainan ini dimisalkan panjang kromosom yang diberikan adalah 4. Angka-

angka tersebut merupakan indeks huruf pada rak komputer. Jika pada proses pengambilan huruf papan terdapat huruf yang dihasilkan, maka 0 adalah indeks untuk huruf tersebut. 1 merupakan indeks untuk huruf pada rak pertama, 2 merupakan indeks untuk huruf pada rak kedua dan seterusnya. Sedangkan jika pada proses pengambilan huruf papan tidak terdapat huruf yang dihasilkan, maka 0 merupakan indeks untuk huruf pada rak pertama, 1 merupakan indeks untuk huruf pada rak kedua dan seterusnya. Sebagai simulasi permainan, hasil proses pengacakan kromosom pada populasi awal terlihat seperti **Gambar 4.20**.

```
kromosom ke-0: 0213  LSIT
kromosom ke-1: 0213  LSIT
kromosom ke-2: 2310  STIL
kromosom ke-3: 2130  SITL
kromosom ke-4: 0231  LSTI
kromosom ke-5: 0132  LITS
kromosom ke-6: 1203  ISLT
kromosom ke-7: 3210  TSIL
kromosom ke-8: 0132  LITS
kromosom ke-9: 2301  STLI
```

Gambar 4.20 Kromosom Awal

Setiap kromosom pada populasi awal dihitung nilai *fitness*nya berdasarkan persamaan (3.1) dan persamaan (3.2) pada bab sebelumnya. Hasil nilai *fitness* untuk masing-masing kromosom pada populasi awal dapat dilihat pada **Gambar 4.21**.

```
NILAI FITNESS KROMOSOM KE-1: 0.5
==================================================
NILAI FITNESS KROMOSOM KE-2: 0.5
==================================================
NILAI FITNESS KROMOSOM KE-3: 0.75
==================================================
NILAI FITNESS KROMOSOM KE-4: 0.75
==================================================
NILAI FITNESS KROMOSOM KE-5: 0.5
==================================================
NILAI FITNESS KROMOSOM KE-6: 0.75
==================================================
NILAI FITNESS KROMOSOM KE-7: 0.75
==================================================
NILAI FITNESS KROMOSOM KE-8: 0.75
==================================================
NILAI FITNESS KROMOSOM KE-9: 0.75
==================================================
```

Gambar 4.21 Nilai *Fitness* Kromosom pada Populasi Awal

Proses algoritma genetika berikutnya adalah proses seleksi, dimana pada perangkat lunak ini proses seleksi menggunakan metode elitisme dan roda rolet. Pada metode elitisme, kromosom dengan nilai *fitness* terjelek dari populasi akan digantikan oleh kromosom dengan nilai *fitness* terbaik. Untuk kasus di atas, kromosom dengan nilai terjelek adalah kromosom ke-0 (0213 – LSIT) dengan nilai *fitness* 0.5 dan digantikan dengan kromosom ke-2 (2310 – STIL) dengan nilai *fitness* 0.75. Sehingga kromosom yang terbentuk setelah proses elitisme adalah seperti terlihat pada **Gambar 4.22**.

```
kromosom ke-0: 2310  STIL
kromosom ke-1: 0213  LSIT
kromosom ke-2: 2310  STIL
kromosom ke-3: 2130  SITL
kromosom ke-4: 0231  LSTI
kromosom ke-5: 0132  LITS
kromosom ke-6: 1203  ISLT
kromosom ke-7: 3210  TSIL
kromosom ke-8: 0132  LITS
kromosom ke-9: 2301  STLI
```

Gambar 4.22 Hasil Kromosom Setelah Elitisme

Proses seleksi dilanjutkan dengan seleksi menggunakan metode roda rolet. Pada proses ini, masing-masing kromosom memiliki probabilitas yang didapatkan dari nilai *fitness* masing-masing kromosom dibagi dengan total nilai *fitness* pada generasi tersebut. Masing-masing kromosom juga memiliki probabilitas kumulatif yang didapatkan dari menjumlah probabilitas kromosom tersebut dengan kromosom sebelumnya, kemudian dilakukan pengacakan nilai *random* sebanyak jumlah kromosom pada satu populasi. Untuk kromosom yang memiliki nilai *random* lebih kecil daripada probabilitas kumulatif akan terpilih untuk dioperasikan pada tahap selanjutnya. Nilai *fitness*, nliai probabilitas, kumulatif probabilitas dan nilai *random* untuk masing-masing kromosom dapat dilihat pada **Tabel 4.1**.

Tabel 4.1 Nilai *Fitness*, Probabilitas Seleksi, Probabilitas Kumulatif dan Nilai Random pada Proses Seleksi dengan Roda rolet

Kromosom	**Nilai *Fitness***	**Probabilitas**	**Probabilitas Kumulatif**	**Nilai Random**
1.	0.75	0.11	0.11	0.043357786
2.	0.5	0.74	0.184	0.5030361
3.	0.75	0.11	0.294	0.26361219
4.	0.75	0.11	0.404	0.40227098
5.	0.5	0.074	0.478	0.7382916
6.	0.75	0.11	0.588	0.8794345
7.	0.75	0.11	0.698	0.6222697
8.	0.75	0.11	0.808	0.24557735
9.	0.75	0.11	0.918	.9706708
10.	0.5	0.074	0.992	0.9382545
Jumlah	6.75	0.992		

Berdasarkan **Tabel 4.1**, maka kromosom-kromosom yang terpilih menggunakan roda rolet adalah kromosom 1, 6, 3, 5, 8, 9, 7, 3, 10 dan 10. Proses seleksi menggunakan metode roda rolet ini dapat dilihat pada **Gambar 4.23**.

Kromosom-kromosom yang terpilih dari proses seleksi kemudian menjadi calon induk pada proses *crossover*. Tidak semua kromosom yang telah terseleksi dilakukan operasi *cossover*, namun hanya kromosom dengan probabilitas *crossover* yang lebih kecil dari parameter probabilitas *crossover* yang telah ditetapkan. Pada perangkat lunak permainan *scrabble* ini, ketetapan parameter *crossover* adalah 0.6. Untuk probabilitas *crossover* masing-masing calon induk kromosom, diperoleh dari nilai acak antara 0 hingga 1. **Tabel 4.2** merupakan probabilitas masing-masing kromosom pada simulasi permainan.

```
random: 0.043357786 dan probabilitas kumulatif: 0.11
calon induk ke-1 dari kromosom ke-1: LSIT: 0 2 1 3
random: 0.5030361 dan probabilitas kumulatif: 0.588
calon induk ke-2 dari kromosom ke-6: LITS: 0 1 3 2
random: 0.26361219 dan probabilitas kumulatif: 0.294
calon induk ke-3 dari kromosom ke-3: STIL: 2 3 1 0
random: 0.40227098 dan probabilitas kumulatif: 0.404
calon induk ke-4 dari kromosom ke-5: LSTI: 0 2 3 1
random: 0.7382916 dan probabilitas kumulatif: 0.808
calon induk ke-5 dari kromosom ke-8: TSIL: 3 2 1 0
random: 0.8794345 dan probabilitas kumulatif: 0.918
calon induk ke-6 dari kromosom ke-9: LITS: 0 1 3 2
random: 0.6222697 dan probabilitas kumulatif: 0.698
calon induk ke-7 dari kromosom ke-7: ISLT: 1 2 0 3
random: 0.24557735 dan probabilitas kumulatif: 0.294
calon induk ke-8 dari kromosom ke-3: STIL: 2 3 1 0
random: 0.9706708 dan probabilitas kumulatif: 0.992
calon induk ke-9 dari kromosom ke-10: STLI: 2 3 0 1
random: 0.9382545 dan probabilitas kumulatif: 0.992
calon induk ke-10 dari kromosom ke-10: STLI: 2 3 0 1
```

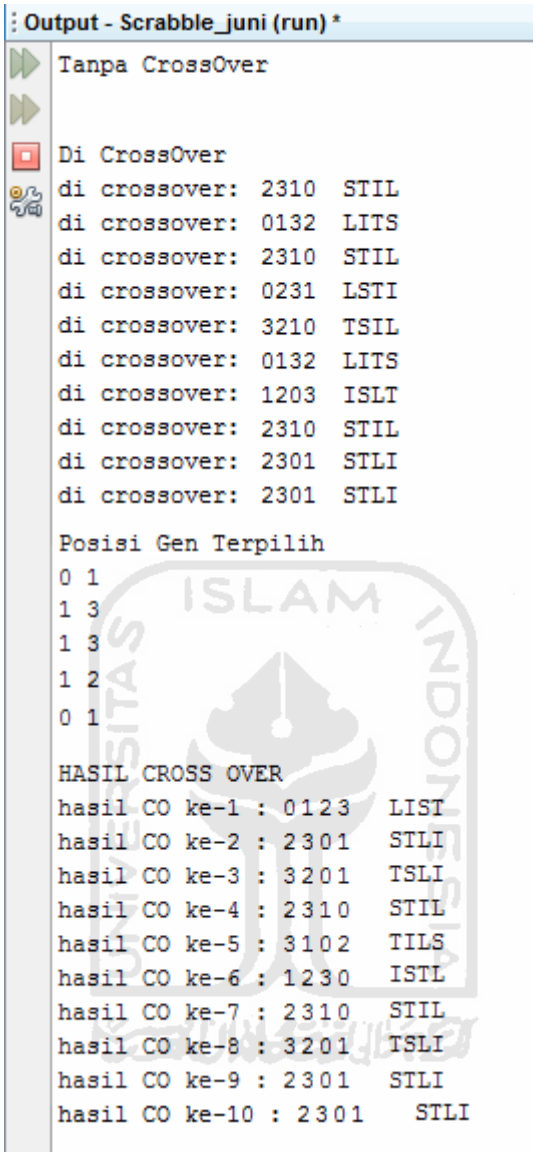
Gambar 4.23 Proses Seleksi dengan Metode Roda rolet

Tabel 4.2 Probabilitas *Crossover* Kromosom

Kromosom	**Probabilitas *Crossover***	**Dilakukan *Crossover***	**Tanpa *Crossover***
1	0.55456	Ya	Tidak
2	0.32412	Ya	Tidak
3	0.18929	Ya	Tidak

Kromosom	**Probabilitas *Crossover***	**Dilakukan *Crossover***	**Tanpa *Crossover***
4	0.18293	Ya	Tidak
5	0.29385	Ya	Tidak
6	0.38640	Ya	Tidak
7	0.12495	Ya	Tidak
8	0.27485	Ya	Tidak
9	0.13955	Ya	Tidak
10	0.43567	Ya	Tidak

Berdasarkan **Tabel 4.2**, diketahui semua kromosom memiliki probabilitas yang lebih kecil daripada probabilitas *crossover* yang ditetapkan. Oleh karena itu, seluruh kromosom akan dilakukan proses *crossover*. Untuk proses *crossover* yang dilakukan menggunakan metode penyilangan berbasis posisi dengan dua titik posisi gen. Pada proses *corssover*, dua kromosom induk akan menghasilkan dua kromosom anak. Jika jumlah kromosom yang akan dilakukan *crossover* berjumlah ganjil, maka kromosom terakhir yang dipilih tidak akan dilakukan *crossover*. Pada kasus simulasi ini, 10 kromosom induk akan menghasilkan 10 kromosom anak. Proses *crossover* simulasi permainan dapat dilihat pada **Gambar 4.24**. Proses diawali dengan memisahkan kromosom yang dilakukan *crossover* dan kromosom yang tidak dilakukan *crossover*. Kemudian akan dilakukan pengacakan posisi gen mana yang akan ditetapkan sebagai titik perpindahan gen induk kromosom pertama kepada anak kromosom kedua dan gen induk kedua kepada anak pertama. Alur proses metode penyilangan berbasis posisi yang digunakan dapat dilihat pada bab sebelumnya.



Gambar 4.24 Proses *Crossover*

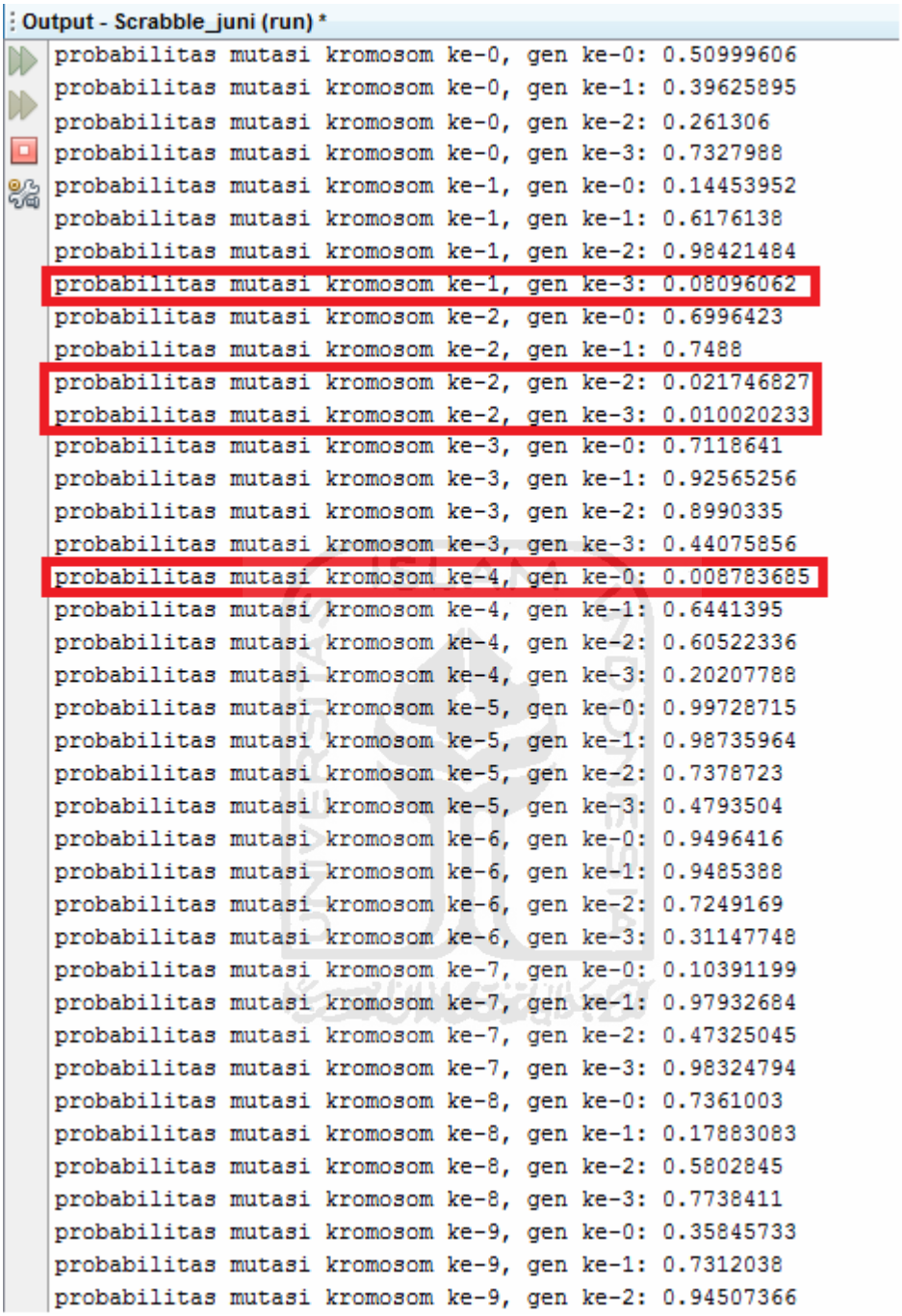
Setelah dilakukan proses *crossover*, selanjutnya dilakukan proses mutasi. Proses mutasi tidak dilakukan untuk setiap kromosom, melainkan dilakukan terhadap gen pada sebuah kromosom. Gen yang memiliki probabilitas lebih kecil daripada parameter mutasi yang diberikan (0.05) akan dimutasi, digantikan dengan nilai yang tidak ada pada kromosom dimana gen tersebut terdapat. Mutasi tidak dapat terjadi pada gen dengan nilai indeks huruf papan karena huruf papan harus selalu ada pada kromosom. Probabilitas mutasi untuk setiap gen pada masing-masing kromosom dapat terlihat pada **Gambar 4.25**, sehingga terlihat gen

mana yang akan dimutasi. **Gambar 4.26** menunjukkan probabilitas mutasi setiap gen dan gen-gen yang akan dimutasi nilainya pada masing-masing kromosom.

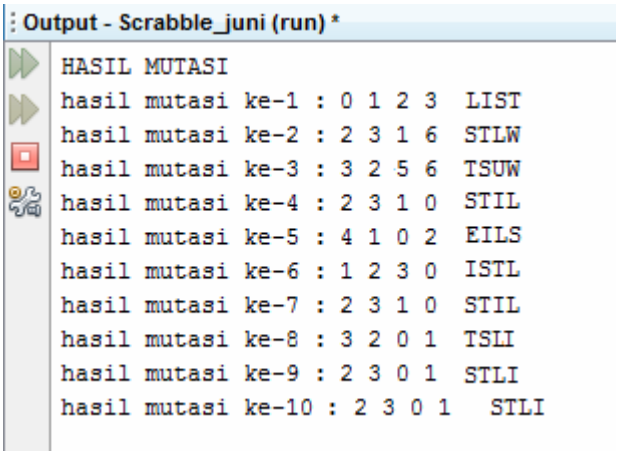
Berdasarkan probabilitas setiap gen seperti **Gambar 4.25**, maka gen yang akan dimutasi adalah gen ke-2 pada kromosom pertama, gen ke-3 pada kromosom ke dua, gen ke-2 pada kromosom ke tiga, gen ke-3 pada kromosom ke tiga dan gen ke-0 pada kromosom ke lima. Nilai gen pengganti adalah angka acak berupa indeks yang tidak terdapat pada kromosom dimana gen tersebut dimutasi. Hasil kromosom setelah mutasi dapat dilihat pada **Gambar 4.26**.

Populasi baru didapatkan dari kromosom-kromosom yang terseleksi (kromosom calon induk yang tidak dilakukan proses penyilangan atau *crossover*) dan hasil akhir genetika sehingga jumlah kromosom pada satu populasi tetap sama dengan jumlah kromosom pada populasi awal. Hasil populasi baru pada simulasi permainan dapat dilihat pada **Gambar 4.26**.

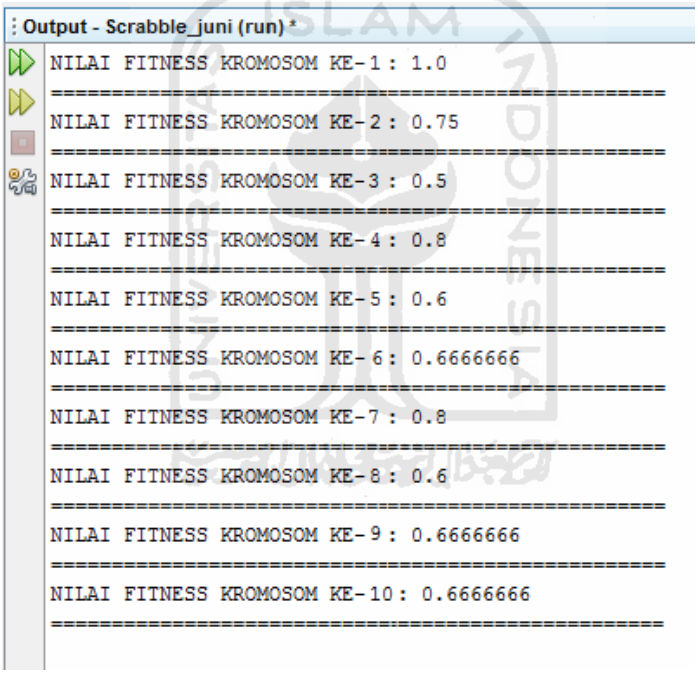
Proses berikutnya yaitu pengecekan kembali nilai *fitness* masing-masing kromosom atau disebut proses evaluasi. Pengecekan nlai *fitness* masing-masing kromosom pada populasi baru dapat dilihat pada **Gambar 4.27**. Jika pada populasi baru ini telah didapatkan kromosom yang diharapkan (kromosom dengan nilai *fitness* 1 yang artinya kata yang terbentuk telah sama dengan salah satu kata pada kamus), maka proses genetika dihentikan. Namun jika belum maka proses-proses genetika akan terus diulang hingga memenuhi jumlah generasi maksimal. Jika pada generasi terakhir belum didapatkan kromosom dengan nilai *fitness* 1, maka komputer tidak dapat membentuk kata dengan panjang kromosom saat itu dan dilakukan pencarian ulang pada panjang kromosom yang lain. Pada kasus simulasi ini, kromosom pertama pada populasi baru dengan panjang kromosom 4 telah memenuhi syarat berhentinya proses genetika sehingga proses genetika dihentikan. Hasil yang diperoleh dari proses genetika untuk kasus di atas, kromosom optimumnya yaitu pada kromosom pertama: 0 1 2 3 yang membentuk kata LIST.



Gambar 4.25 Probabilitas Mutasi Gen



Gambar 4.26 Hasil Kromosom Setelah Mutasi



Gambar 2.27 Nilai *Fitness* Baru

Jika setelah proses genetika dilakukan sebanyak generasi maksimal namun belum ditemukan kromosom yang memiliki nilai *fitness* 1, maka algoritma genetika tidak menemukan solusi dan komputer tidak memainkan hurufnya pada giliran permainan. Sebaliknya, jika genetika mampu memperoleh kata yang terdapat pada kamus, maka proses selanjutnya adalah memindahkan huruf yang didapatkan ke papan permainan seperti pada **Gambar 4.28**.

Gambar 4.28 Tampilan Setelah Komputer Main

4.3.2 Waktu Kompilasi

Pengujian perangkat lunak dilakukan dengan membedakan beberapa parameter algoritma. Perbandingan waktu kompilasi jalannya perangkat lunak dengan parameter algoritma yang berbeda dapat dilihat pada **Tabel 4.3**.

Tabel 4.3 Perbandingan Waktu Kompilasi Berdasarkan Parameter Algoritma

No	Jumlah Generasi	Ukuran Populasi	PC	PM	Waktu Kompilasi	Solusi	Panjang Kromosom
1	10	30	0.75	0.75	2m 50d	Ada	3
2	10	30	0.75	0.5	2m 10d	Ada	5
3	10	30	0.75	0.5	2m 18d	Ada	4
4	10	30	0.75	0.3	3m 22d	Tidak	3
5	10	30	0.75	0.1	2m 38d	Ada	4
6	10	30	0.5	0.75	2m 52d	Ada	3
7	10	30	0.5	0.5	3m 03d	Ada	3

No	Jumlah Generasi	Ukuran Populasi	PC	PM	Waktu Kompilasi	Solusi	Panjang Kromosom
8	10	30	0.5	0.3	2m 53d	Ada	3
9	10	30	0.5	0.1	2m 30d	Ada	4
10	10	30	0.3	0.75	2m 40d	Ada	4
11	10	30	0.3	0.5	1m 59d	Ada	5
12	10	30	0.3	0.3	3m 24d	Tidak	3
13	10	30	0.3	0.1	3m 13d	Ada	3
14	10	30	0.1	0.75	2m 56d	Ada	3
15	10	30	0.1	0.5	2m 55d	Ada	3
16	20	30	0.75	0.5	3m 35d	Ada	5
17	30	30	0.75	0.5	2m 53d	Ada	6
18	10	35	0.75	0.5	2m 03d	Ada	5
19	10	40	0.75	0.5	3m 02d	Ada	4
20	10	45	0.75	0.5	4m 28d	Ada	3

Berdasarkan **Tabel 4.3** terlihat bahwa parameter yang sangat mempengaruhi waktu kompilasi adalah ukuran populasi. Semakin banyak ukuran populasi maka semakin lama waktu kompilasi. Hal ini dikarenakan semakin banyak pula kemungkinan kromosom dilakukan *crossover* dan gen yang dilakukan mutasi. Untuk parameter probabilitas *crossover* dan probabilitas mutasi tidak terlalu mempengaruhi waktu kompilasi, namun parameter ini mempengaruhi tingkat varian kromosom. Artinya, semakin kecil probabilitas *crossover* dan probabilitas mutasi maka kromosom-kromosom yang dihasilkan lebih bervariasi. Untuk panjang kromosom juga mempengaruhi waktu kompilasi. Namun hal ini dikarenakan pencarian dengan algortima genetika diawali dengan pencarian pada panjang kromosom 7 (statistik panjang kata terbanyak pada kamus). Jika pencarian pada panjang kromosom 7 ini tidak didapatkan kata terbentuk, maka dilakukan pencarian ulang dengan panjang kromosom 6 dan seterusnya. Sehingga semakin sedikit panjang kromosom di akhir pencarian maka waktu kromosom semakin lama. Parameter jumlah generasi juga tidak terlalu mempengaruhi waktu kompilasi karena jika pada suatu proses telah ditemukan kromosom sebagai solusi

kata yang terbentuk maka proses pencarian langsung dihentikan. Melalui **Tabel 4.3** juga dapat terlihat bahwa perangkat lunak yang dibuat kemungkinan besar menemukan solusi kata yang dapat dimainkan sehingga Algoritma Genetika cukup mangkus untuk digunakan untuk permasalahan pencarian kata pada permainan *scrabble*. Namun sangat disayangkan bahwa waktu kompilasi rata-rata perangkat lunak yang dibuat cukup lambat. Hal ini dapat dikarenakan penulisan kode program yang belum cukup efisien.

4.4 Kelebihan Perangkat Lunak

Perangkat lunak permainan *scrabble* yang dibuat memiliki kelebihan, yaitu:

1. Perangkat lunak yang dibangun menggunakan Algoritma Genetika mampu memberikan solusi permainan sehingga menjadikan komputer sebagai lawan main manusia.
2. Representasi kromosom Algoritma Genetika yang dibangun dapat membentuk solusi permainan.

4.5 Kekurangan Perangkat Lunak

Selain kelebihan yang disebutkan pada subbab sebelumnya, perangkat lunak yang dibangun juga memiliki beberapa kekurangan, yaitu:

1. Perangkat lunak yang dibuat cukup lambat untuk menemukan solusi kata yang terbentuk dikarenakan penulisan kode program yang belum efisien.
2. Desain dan animasi perangkat lunak yang dibangun terlalu sederhana sehingga kurang menarik.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah dilakukan serangkaian pengujian pada perangkat lunak permainan *scrabble* yang dibangun dalam Tugas Akhir ini, dapat disimpulkan beberapa hal, yaitu:

1. Algoritma genetika dapat digunakan untuk optimasi pencarian kata pada permainan *scrabble.*
2. Parameter algoritma yang mempengaruhi waktu kompilasi pada perangkat lunak yang dibangun adalah ukuran populasi.
3. Parameter algoritma yang cukup mempengaruhi waktu kompilasi pada perangkat lunak yang dibangun adalah jumlah generasi.
4. Parameter algoritma yang tidak terlalu mempengaruhi waktu kompilasi pada perangkat lunak yang dibangun adalah probabilitas kromosom dan probabilitas mutasi.
5. Hasil kinerja algoritma genetika dengan representasi kromosom yang dibuat kurang efisien sehingga cukup lambat dalam menemukan solusi.

5.2 Saran

Hasil dari penelitian dan pengujian pada perangkat lunak yang dibangun terdapat beberapa keterbatasan, untuk itu pengembangan penelitian di masa yang akan datang dapat ditambahkan beberapa hal, yaitu:

1. Diperlukan analisis algoritma lebih lanjut mengenai penulisan kode program yang dibuat agar perangkat lunak lebih efisien menemukan solusi kata yang terbentuk.
2. Pengembangan perangkat lunak dengan desain dan animasi yang lebih menarik, namun tetap tidak mengganggu permainan dan mudah untuk dioperasikan.

Pengembangan perangkat lunak juga dapat melibatkan hal-hal yang masih dibatasi pada perangkat lunak yang telah dibangun ini, yaitu:

1. Membangun perangkat lunak permainan *scrabble* dengan algoritma genetika yang parameter algoritma genetikanya bersifat dinamis.
2. Membangun perangkat lunak permainan *scrabble* dengan kamus yang bersifat dinamis.
3. Membangun perangkat lunak permainan *scrabble* yang mencakup elemen permainan *scrabble* secara lengkap, menambahkan huruf *blank* pada permainan, dapat menyusun huruf berdempetan dengan catatan setiap baris dan setiap kolom kata yang terbentuk harus memiliki arti, menambahkan fitur ganti huruf untuk masing-masing pemain pada setiap giliran bermain dan dapat menambahkan huruf pada kata yang sudah terbentuk di papan sehingga kata baru yang terbentuk setelah disambung juga memiliki karti.
4. Membangun perangkat lunak permainan *scrabble* dengan tingkatan level permainan dan penyimpanan nilai tertinggi.

DAFTAR PUSTAKA

Erickson, Jeff. 2011. *Algorithms*. Diakses dari http://compgeom.cs.uiuc. edu/~jeffe/teaching/algorithms/notes/all-notes.pdf, pada tanggal 29 April 2011.

Hasbro. 2010. *Hasbro*. Diakses dari http://www.hasbro.com/scrabble/en_ US/gamePlayRules.cfm, pada tanggal 5 Januari 2011.

Hasbro. 2010. *Hasbro*. Diakses dari http://www.hasbro.com/scrabble/en_ US/ scoringRules.cfm, pada tanggal 5 Januari 2011.

Sri Kusumadewi, dan Hari Purnomo 2005. Penyelesaian Masalah Optimasi Menggunakan Teknik-teknik Heuristik. Yogyakarta: Graha Ilmu.

Satriyanto, Edi. 2009. *Bab 7 Algoritma Genetika.* Surabaya : Institut Teknologi Sepuluh November. (Online) (http://lecturer.eepis-its.edu/~kangedi/materi% 20kuliah/Kecerdasan%20Buatan/Bab%207%20Algoritma%20Genetika.pdf. Diakses 19 Januari 2011).

Virniawati, Anatariani. 2007. *Pemanfaatan Algoritma Backtracking dalam Prpgram Permainan Scrabble*. Makalah. Bandung. (Online) (http://www.informatika.org/~rinaldi/Stmik/2006-2007/Makalah_2007/MakalahSTMIK2007-119.pdf. Diakses 12 Desember 2010).

Zukhri, Zainudin. 2010. *Optimasi dengan Algoritma Genetika*. Modul Kuliah.