PENENTUAN KOMPOSISI MENU MAKANAN UNTUK PENDERITA DIABETES MELLITUS MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

Disusun oleh :

RESTHY KUSHARDIANA

NIM. 0810960060

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

PENENTUAN KOMPOSISI MENU MAKANAN UNTUK PENDERITA DIABETES MELLITUS MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

Sebagai salah satu syarat untuk memperoleh
Gelar Sarjana dalam bidang Ilmu Komputer

Disusun oleh :
RESTHY KUSHARDIANA
NIM. 0810960060

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2013

PENENTUAN KOMPOSISI MENU MAKANAN UNTUK PENDERITA DIABETES MELLITUS MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

Sebagai salah satu syarat untuk memperoleh
Gelar Sarjana dalam bidang Ilmu Komputer

Disusun oleh :
RESTHY KUSHARDIANA
NIM. 0810960060

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I,	Dosen Pembimbing II,
Drs. Muh.Arif Rahman, M.Kom.	**Drs. Achmad Ridok, M.Kom.**
NIP. 19660423 199111 1 001	**NIP. 19680825 199403 1 002**

LEMBAR PENGESAHAN SKRIPSI

PENENTUAN KOMPOSISI MENU MAKANAN UNTUK PENDERITA DIABETES MELLITUS MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

Sebagai salah satu syarat untuk memperoleh
Gelar Sarjana dalam bidang Ilmu Komputer

Disusun oleh:

RESTHY KUSHARDIANA

NIM. 0810960060

Setelah dipertahankan di depan Majelis Penguji

pada tanggal 22 Januari 2013

dan dinyatakan memenuhi syarat untuk memperoleh

gelar Sarjana dalam bidang Ilmu Komputer

Penguji,	**Penguji,**
Dian Eka Ratnawati, S.Si., M.Kom.	**Suprapto, S.T., M.T.**
NIP. 19710727 199603 1 001	**NIP. 19710727 199603 1 001**

Penguji,

Ahmad Afif Supianto, S.Si., M.Kom.
NIP.

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Januari 2013

Mahasiswa,

Resthy Kushardiana

NIM. 0810960060

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yesus Kristus, karena hanya dengan berkat dan anugerah serta bimbingan-Nya penulis dapat menyesaikan tugas akhir yang berjudul **"Penentuan Komposisi Menu Makanan untuk Penderita *Diabetes Mellitus* Menggunakan Algoritma Genetika"**. Selama penyelesaian tugas akhir, tak sedikit penulis mendapatkan bantuan petunjuk, bimbingan, do'a, dukungan serta saran dari berbagai pihak hingga penulis dapat menyelesaikan tugas akhir ini. Oleh karena itu segala rasa terima kasih penulis sampaikan kepada :

1. Drs. Muh. Arif Rahman, M.Kom. selaku dosen pembimbing utama, atas kesediaan menjadi pembimbing, ilmu yang berharga, segala masukan, kesabaran dalam membimbing, nasehat dan motivasi yang telah diberikan. Serta Drs. Achmad Ridok, M.Kom. selaku dosen pembimbing pendamping, atas segala bimbingan, ilmu yang berharga, nasehat dan motivasi yang telah diberikan.
2. Muhammad Tanzil Furqon, S.Kom., selaku dosen Penasehat Akademik atas bimbingan dan saran yang diberikan selama ini.
3. Kedua orang tua penulis dan adik tercinta atas segala kasih sayang, do'a yang tak henti, nasehat, pelajaran hidup yang berharga dan segala keikhlasan yang diberikan.
4. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Eddy Santoso, S.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
5. Bapak Drs. Marji, M.T. dan Bapak Issa Arwani, S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Informatika / Ilmu Komputer Universitas Brawijaya.
6. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer/Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.

7. Dafid Eko, Fahrur, Tirana, Nisa, Nike, dan teman-teman Ilkom B 2008 lainnya beserta rekan – rekan di Program Studi Ilmu Komputer Universitas Brawijaya yang telah banyak memberikan bantuan demi kelancaran tugas akhir ini.
8. Seluruh Civitas Akademi Informatika / Ilmu Komputer Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Informatika / Ilmu Komputer Universitas Brawijaya dan selama penyelesaian skripsi ini.
9. Dan semua pihak yang telah terlibat baik secara langsung maupun tidak langsung yang tidak dapat penulis sebutkan satu per satu terima kasih atas semua bantuan yang telah diberikan.

Penulis berharap agar apa yang telah penulis buat dapat bermanfaat nantinya. Dan besar harapan penulis semoga tugas ini dapat bermanfaat bagi semua pihak yang berkepentingan.

Penulis menyadari bahwa masih ada ketidaksempurnaan selama penyelesaian tugas akhir ini. Pada kesempatan ini pula penulis memohon maaf atas segala ketidaksempurnaan yang ada. Kritik dan saran dari berbagai pihak, yang bersifat membangun juga sangat penulis harapkan untuk bahan perbaikan di masa yang akan datang.

Malang, Januari 2013

Penulis

ABSTRAK

Resthy Kushardiana. 2013. : Penentuan Komposisi Menu Makanan untuk Penderita *Diabetes Mellitus* Menggunakan Algoritma Genetika.
Dosen Pembimbing : Drs. Muh. Arif Rahman, M.Kom. dan Drs. Achmad Ridok, M.Kom.

Indonesia merupakan salah satu negara dengan jumlah penderita diabetes terbanyak di dunia. Sampai dengan tahun 2008, Departemen Kesehatan Republik Indonesia mencatat Indonesia sebagai peringkat ke-4 di dunia yaitu 5,6%. Pada penyakit *Diabetes Mellitus*, diet merupakan pengobatan paling mendasar melalui kombinasi dalam penyusunan menu makanannya. Penyusunan komposisi menu makanan termasuk dalam suatu permasalahan kombinatorik dan dapat diselesaikan dengan algoritma genetika. Sesuai struktur umum dari algoritma genetika, data bahan makanan dikombinasikan menjadi individu yang merupakan 1 kesatuan menu dalam sehari dan akan dievaluasi menggunakan sebuah fungsi evaluasi (fungsi *fitness*). Dari hasil pengujian yang dilakukan, didapatkan bahwa nilai probabilitas *crossover* yang terlalu rendah atau terlalu tinggi mengakibatkan *fitness* yang buruk. Sedangkan jumlah populasi dan generasi yang semakin tinggi, cenderung menghasilkan *fitness* yang baik. Dari beberapa hasil pengujian, dapat dihasilkan solusi lokal optimum dengan nilai *fitness* maksimum yaitu 1,0 sehingga sistem dikatakan dapat diimplementasikan untuk menghasilkan kombinasi menu dan berat makanan dalam sehari sesuai dengan kebutuhan tubuh penderita *Diabetes Mellitus*.

Kata kunci : Algoritma Genetika, *Diabetes Mellitus*, menu makanan

ABSTRACT

Resthy Kushardiana. 2013. : ***Food Menu Composition Determination for Diabetes Mellitus Patient Using Genetic Algorithm***

Advisor : **Drs. Muh. Arif Rahman, M.Kom. and Drs. Achmad Ridok, M.Kom.**

Indonesia is one of the countries which have the highest number of diabetics in the world. In 2008, the Ministry of Health of Republic of Indonesia noted that Indonesia was ranked the fourth in the world approximately 5.6%. In diabetes mellitus, diet is a fundamental treatment through healthy food combination. Composing the menu is one of the combinatory problems and can be solved using genetic algorithm. According to the common structure of genetic algorithm, foodstuff data is compounded as individual which formed one menu combination in a day and will be evaluated using fitness function formula. The testing result show that a too high or low crossover probability will bring out the bad fitness. Meanwhile, the higher the value of population and generation amount tends to give the better fitness. From several testing, the optimum local is reached with the maximum value of fitness 1.0, thus this system can be implemented to result the menu and weight food combination in one day according to diabetic body requirement.

Keyword *: Genetic Algorithm, Diabetes Mellitus, food composition*

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *SOURCE CODE*

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia merupakan salah satu negara dengan jumlah penderita diabetes terbanyak di dunia. Sampai dengan tahun 2008, Departemen Kesehatan Republik Indonesia mencatat Indonesia sebagai peringkat ke-4 di dunia yaitu 5.6%, itupun hanya jumlah data yang tercatat, belum termasuk data yang dimungkinkan tidak tercatat. Badan Kesehatan Dunia (*World Health Organization/WHO*) memperkirakan jumlah penderita *Diabetes Mellitus* di Indonesia akan meningkat tiga kali lipat dan pada tahun 2030 akan mencapai 21,3 juta orang [ANO[1]-12].

Diabetes merupakan salah satu jenis penyakit seumur hidup di mana tubuh seseorang tidak memproduksi insulin (hormon pengendali glukosa) dalam jumlah yang cukup atau tidak dapat menggunakan insulin yang diproduksi dengan baik. Jika insulin tersebut tidak ada, maka akan mengakibatkan semua glukosa yang berasal dari makanan yang dikonsumsi akan mengendap dalam darah sehingga kadar gula darah meningkat. Penyakit diabetes sendiri merupakan penyakit kronis yang merugikan dari sisi produktivitas dan biaya penanganannya [MAR-98].

Diabetes dapat dikendalikan dengan cara menurunkan kadar gula darah, salah satu caranya adalah melakukan diet yang merupakan dasar utama pengobatan dengan cara menekan total kalori yang dimakan. Diet untuk penderita diabetes pun harus dikelola secermat mungkin. Prinsip terpenting adalah jumlah karbohidrat dan kombinasi pemenuhan gizi lain yang dimakan harus stabil [MAR-98].

Diet yang dilakukan melalui penyusunan komposisi menu makanan termasuk dalam suatu permasalahan kombinatorik. Kombinatorik merupakan salah satu cabang matematika yang menggunakan prinsip pencacahan objek-objek yang berada dalam suatu himpunan secara cepat dengan teknik-teknik tertentu. Jika anggota dalam himpunan tersebut sedikit, maka mudah untuk menyusun alternatif solusinya, tetapi kombinatorik ini sendiri akan lebih terlihat fungsinya ketika kombinasi yang dihasilkan bisa menjadi ratusan ribu. Dalam penyelesaian

persoalan kombinatorik ini, perlu diketahui terlebih dahulu prinsip dasar apa yang nantinya akan diterapkan dalam metode kombinatorik [DAN-12].

Kombinatorik dapat diselesaikan dengan bermacam cara, salah satunya adalah dengan algoritma genetika. Algoritma genetika pertama kali ditemukan oleh John Holland. Algoritma Genetika secara khusus dapat diterapkan untuk memecahkan suatu permasalahan optimasi yang kompleks karena algoritma genetika mampu menghasilkan *performance* yang lebih optimal daripada algoritma klasik [SAD-09] selain itu algoritma genetika baik untuk aplikasi yang memerlukan strategi pemecahan masalah secara adaptif [GEN-00]. Algoritma genetika juga merupakan teknik pencarian stokastik, yaitu pencarian berdasarkan populasi dan algoritma yang mengadopsi paradigma evolusi evolusi [SUY-10].

Pada algoritma genetika terdapat beberapa istilah evolusi seperti individu dalam sebuah populasi untuk beberapa generasi. Setiap individu itu sendiri merepresentasikan satu solusi yang nantinya akan di evaluasi dengan suatu fungsi yang disebut fungsi *fitness*. Fungsi *fitness* dicari secara matematis untuk mencapai solusi yang optimal. Dalam algoritma genetika juga terdapat 2 jenis *transformasi* yaitu mutasi, yang menghasilkan individu baru dari perubahan satu individu saja dan *crossover*, yang akan menghasilkan individu baru dengan mengkombinasikan dua individu [GEN-00].

Penentuan komposisi dalam menu makanan penderita *Diabetes Mellitus* merupakan salah satu permasalahan kombinatorik yang solusinya bisa dicari menggunakan algoritma genetika. Secara garis besar, permasalahan kombinatorik ini terlihat dalam penyusunan gen dalam kromosom untuk mencari sebuah solusi yaitu suatu kesatuan menu makanan yang optimal sesuai dengan kebutuhan tubuh.

Sebelumnya, sudah pernah juga dilakukan penelitian tentang penyusunan menu makanan dengan algoritma genetika, antara lain pemecahan masalah kombinatorik dengan menggunakan algoritma genetika untuk menghitung komposisi bahan pangan dalam rangka penentuan diet bagi penderita gangguan ginjal dengan menggunakan 400 data bahan pangan [UYU-11]. Selain dengan menggunakan algoritma genetika, juga pernah dilakukan penelitian dengan menggunakan *Fuzzy Mathematical Programming* untuk menyusun menu makanan yang baik dalam pola makan sehari-hari penduduk Jepang [TOM-08].

Berdasarkan latar belakang di atas, pada skripsi ini akan coba dilakukan penelitian terhadap Penentuan Komposisi Menu Makanan untuk Penderita *Diabetes Mellitus* menggunakan Algoritma Genetika.

1.2 Rumusan Masalah

Rumusan masalah dalam penulisan skripsi ini adalah :

1. Bagaimana mengaplikasikan algoritma genetika dalam menentukan komposisi menu makanan dalam sehari untuk penderita *Diabetes Mellitus* ?
2. Bagaimanakah pengaruh nilai probabilitas *crossover* dan nilai probabilitas mutasi yang digunakan pada penentuan komposisi menu makanan untuk penderita *Diabetes Mellitus* ?

1.3 Batasan Masalah

Batasan masalah dalam penulisan skripsi ini adalah :

1. Data yang digunakan dalam penelitian ini adalah data pasien penderita penyakit *Diabetes Mellitus* II dan tanpa komplikasi penyakit yang didapatkan dari Rumah Sakit Baptis Kediri dalam periode Januari-Maret 2012.
2. Parameter utama yang digunakan dalam mendapatkan komposisi menu makanan adalah nilai kalori yang sudah dihitung dari tiap pasien.
3. Kekurangan atau kelebihan nutrisi dari kebutuhan seimbang dalam makanan, dianggap sama-sama buruk.
4. Tidak dilakukan evaluasi kesamaan menu dalam sehari (untuk tiap individu).
5. Tidak dilakukan perbandingan dengan metode lain.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penulisan skripsi ini adalah sebagai berikut :

1. Mengimplementasikan algoritma genetika dalam penentuan komposisi menu makanan dalam sehari untuk penderita *Diabetes Mellitus*.
2. Mengukur dan mengetahui pengaruh pengaruh nilai probabilitas *crossover* dan nilai probabilitas mutasi pada penentuan komposisi menu makanan untuk penderita *Diabetes Mellitus*.

1.5 Manfaat Penelitian

Manfaat yang diperoleh dari penulisan skripsi ini adalah tersedianya perangkat lunak untuk menentukan menu makanan yang menerapkan prinsip algoritma genetika dan pada akhirnya diharapkan bisa dijadikan bahan pertimbangan yang bisa membantu tim ahli gizi dalam menyusun menu makanan untuk penderita *Diabetes Mellitus* sesuai dengan kebutuhan tubuh mereka.

1.6 Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan skripsi ini adalah sebagai berikut :

1. Studi Literatur

   Mempelajari dan mengkaji beberapa literatur (jurnal, buku, dan artikel dari website) mengenai Diet Makanan *Diabetes Mellitus* dan Algoritma Genetika. Serta melakukan konsultasi dengan tim ahli dalam bidang yang mencakup penyusunan skripsi ini.
2. Pendefinisian dan Analisis Masalah

   Mengkaji permasalahan sebagai hasil dari studi pustaka dan menganalisis apa saja yang dibutuhkan dalam pemecahan masalah.
3. Perancangan dan Implementasi Sistem

   Mengimplementasikan prinsip dalam algoritma genetika dengan merancang dan membangun sebuah perangkat lunak untuk membuat sistem penentuan komposisi menu makanan untuk penderita *Diabetes Mellitus*.
4. Uji coba dan analisa hasil implementasi

   Menguji coba sistem dan mengetahui akurasinya dengan cara mencocokkan dengan perhitungan manual tim ahli gizi.
5. Pengambilan Kesimpulan

   Mengambil kesimpulan dari *output* program yang dihasilkan

1.7 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. **BAB I PENDAHULUAN**

   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi pemecahan masalah serta sistematika penulisan skripsi.

2. **BAB II TINJAUAN PUSTAKA**

   Menguraikan teori – teori yang berhubungan dengan *Diabetes Mellitus* dan algoritma genetika serta teori-teori yang berhubungan dengan penggunaan metode algoritma genetika dalam mencari solusi dalam permasalahan ini.

3. **BAB III METODOLOGI DAN PERANCANGAN SISTEM**

   Pada bab ini akan dijelaskan mengenai metode-metode yang digunakan dalam membangun sistem yang mampu memecahkan permasalahan pada penentuan komposisi nutrisi dalam menu makanan untuk penderita *Diabetes Mellitus*.

4. **BAB IV HASIL DAN PEMBAHASAN**

   Dalam bab ini akan dijelaskan mengenai implementasi program, pengujian dan analisa hasil penelitian.

5. **BAB V KESIMPULAN DAN SARAN**

   Bab ini berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan pengembangan.

BAB II
TINJAUAN PUSTAKA

2.1 *Diabetes Mellitus*

2.1.1 Gambaran Umum

Diabetes Mellitus adalah suatu penyakit yang terjadi bila tubuh seseorang tidak dapat menangani kadar glukosa (gula) dalam darah. Penderita *Diabetes Mellitus* tidak memproduksi insulin (hormon pengendali glukosa) dalam jumlah yang cukup atau tidak dapat menggunakan insulin yang diproduksi dengan baik (Marilyn, 1998). Ada dua jenis *Diabetes Mellitus* yaitu Tipe I - IDDM *(Insulin Dependent Diabetes Mellitus*) dan Tipe II - NIDDM (*Non-Insulin Dependent Diabetes Mellitus*). Pada tipe I, merupakan *Diabetes Mellitus* yang terjadi sejak anak-anak atau merupakan faktor keturunan, sedangkan pada tipe II merupakan *Diabetes Mellitus* yang terjadi ketika usia dewasa dan terjadi karena pola makan yang kurang atau bahkan tidak baik [DIE-90].

Menurut Marylin, *Diabetes Mellitus* dapat dikendalikan dengan cara menurunkan kadar gula darah, salah satu caranya adalah dengan melakukan diet yang merupakan dasar utama pengobatan dengan cara menekan total kalori yang dimakan. Diet untuk penderita *Diabetes Mellitus* pun harus dikelola secermat mungkin. Prinsip terpenting adalah jumlah karbohidrat dan kombinasi pemenuhan gizi lain yang dimakan harus stabil [MAR-98],

2.1.2 Angka Metabolisme Basal

Angka Metabolisme Basal (AMB) atau *Basal Metabolic Rate* (BMR) adalah kebutuhan energi minimal yang dibutuhkan tubuh untuk menjalankan proses tubuh yang vital. Kebutuhan energi metabolisme basal termasuk jumlah energi yang diperlukan untuk pernapasan, peredaran darah, pekerjaan ginjal, pankreas, dan lain-lain serta untuk proses metabolisme di dalam sel-sel dan untuk mempertahankan suhu tubuh. Angka metabolisme basal dinyatakan dalam kilo kalori berat badan perjam dan dalam setiap orang, nilai AMB berbeda-beda. Dari banyak penelitian, ternyata indeks yang paling berpengaruh terhadap AMB adalah

berat badan menurut umur. Sehingga *FAO/WHO/UNU*/1985 telah mengeluarkan rumus untuk menaksir AMB dari berat badan seperti pada tabel 2.1 berikut [ALM- 03].

Tabel 2. 1 Rumus Regresi Linier, FAO/WHO/UNU/1985

Kelompok Umur (tahun)	AMB (Kkal/hari)	
	Laki-laki	Perempuan
0-3	60.9 B - 54	61 B - 51
3-10	22.7 B + 495	22.5 B + 499
10-15	17.5 B + 651	12.2 B + 746
18-30	15.3 B + 679	14.7 B + 496
30-60	11.6 B + 879	8.7 B + 829
>60	13.5 B + 487	10.5 B + 596

* Sumber: *FAO/ WHO/USU, Energi and Protein Requirements*, 1985

Keterangan : B = berat badan dalam kilogram

2.1.3 Diet *Diabetes Mellitus*

Diet merupakan dasar utama pengobatan *Diabetes Mellitus*. Tujuan dari diet ini adalah untuk memperbaiki kebiasaan makan dan olahraga untuk mendapatkan kontrol metabolik yang lebih baik dengan cara mempertahankan kadar glukosa dalam darah. Diet yang digunakan, akan dikontrol berdasarkan kandungan energi, protein, lemak, dan karbohidrat. Dalam diet penyakit *Diabetes Mellitus*, makanan dibagi ke dalam 3 porsi besar, yaitu makan pagi 20%, siang 30%, malam 25% serta 2-3 porsi kecil untuk makanan selingan (masing-masing 10-15%) [ALM-04]. Menurut Almatsier, terdapat 8 jenis diet *Diabetes Mellitus* seperti yang dicantumkan dalam Tabel 2.2 berikut [ALM-08] :

Tabel 2. 2 Jenis Diet *Diabetes Mellitus* menurut kandungan energi, protein, lemak, dan karbohidrat

Jenis Diet	Energi (kkal)	Protein (gr)	Lemak (gr)	Karbohidrat (gr)
I	1100	43	30	172
II	1300	45	35	192
III	1500	51,5	36,5	235
IV	1700	55,5	36,5	275
V	1900	60	48	299
VI	2100	62	53	319
VII	2300	73	59	369
VIII	2500	80	62	396

Dari kebutuhan seimbang pada Tabel 2.2, dapat dibuat batasan *minimum* dan maksimum yang masih diperbolehkan oleh ahli gizi pihak rumah sakit seperti dalam tabel 2.3 (Wawancara ahli gizi, 2012).

Tabel 2. 3 Batasan *minimum* dan maksimum yang masih diperbolehkan

Jenis Diet	Energi (kkal)	Protein (gr)	Lemak (gr)	Karbohidrat (gr)
I	**1100**	**41,5-44,5**	**28,5-31,5**	**170-180**
II	**1300**	**45-47**	**32-34**	**181-192**
III	**1500**	**48-51,5**	**36-39**	**200-235**
IV	**1700**	**52-55,5**	**36-39**	**240-275**
V	**1900**	**56-60**	**40-47**	**280-299**
VI	**2100**	**61-64**	**48-53**	**300-329**
VII	**2300**	**65-73**	**54-59**	**330-369**
VIII	**2500**	**74-80**	**60-63**	**370-396**

Sesuai kebutuhan dalam sehari, maka masing-masing komponen *makronutrein* tersebut akan dibagi lagi menjadi kebutuhan untuk makan pagi, siang, malam dan pelengkap (Wawancara ahli gizi, 2012).

Tabel 2. 4 Pembagian jumlah kebutuhan tiap komponen karbohidrat, lemak dan protein dalam sehari sesuai tipe diet

Jenis Diet	Energi (kkal)	Kebutuhan	Protein (gr)	Lemak (gr)	Karbohidrat (gr)
I	1100	Pagi	9-10	6-8	34-36
		Siang	13-14	9-11	50-52
		Malam	10-12	7,5-9	42-44
		Pelengkap	10-12	7,5-9	42-44
II	1300	Pagi	11-12	8-10	37-39
		Siang	13-15	10,5-12	56-58
		Malam	10-13	8-10	46-49
		Pelengkap	10-13	8-10	46-49
III	1500	Pagi	9-11	6-8	45-49
		Siang	15-16	10-12	69-72
		Malam	12-14	9,125	57-60
		Pelengkap	12-14	9,125	57-60
IV	1700	Pagi	10-12	6-8	50-55
		Siang	15-17	9-11	80-84
		Malam	12-14	8-10	67-70
		Pelengkap	12-14	8-10	67-70
V	1900	Pagi	10-13	8-10	56-61
		Siang	17-19	13-15	80-93
		Malam	14-16	11-14	70-80
		Pelengkap	14-16	11-14	70-80
VI	2100	Pagi	11-13,5	9-11	61,5-68
		Siang	17-19	14-17	90-100
		Malam	14-16	12-15	73-85
		Pelengkap	14-16	12-15	73-85
VII	2300	Pagi	13-15	10-13	70-80
		Siang	19-22	16-18	100-110
		Malam	17-19	14-16	89-100
		Pelengkap	17-19	14-16	89-100
VIII	2500	Pagi	15,5-16,5	11-13	70-81
		Siang	22-26	18-19,5	110-121
		Malam	18-21	14-16	90-100
		Pelengkap	18-21	14-16	90-100

2.2 *Makronutrein*

Makanan merupakan bahan nabati dan hewani yang layak masuk ke dalam tubuh. Dalam makanan tersebut mengandung zat gizi yang diperlukan oleh tubuh. Berdasarkan jumlah yang diperlukan oleh tubuh, zat gizi terbagi menjadi dua,

yaitu zat gizi makro/*makronutrein* dan zat gizi mikro/*mikronutrein*. *Makronutrein* terdiri dari karbohidrat, lemak dan protein. Dalam tubuh, sebagian besar energi diperoleh dari *makronutrein* ini. Menurut PUGS (Panduan Umum Gizi Seimbang), dianjurkan perbandingan jumlah kebutuhan protein adalah 10-15% dari energi total, kebutuhan lemak adalah sebesar 20-25% dari kebutuhan energi total dan sisanya sebesar 60-70% dari kebutuhan energi total adalah untuk karbohidrat [ALM-04]. Berikut ini akan dijelaskan masing-masing dari *makronutrein* menurut Pudjiadi [PUD-09].

2.2.1 Karbohidrat

Karbohidrat merupakan salah satu sumber nutrisi dalam tubuh manusia. Dalam satu gram karbohidrat, terdapat nilai energi sebesar 4 kkal. Fungsi dari karbohidrat adalah sebagai sumber energi utama, pengatur metabolisme lemak, penghemat fungsi protein, sumber energi utama bagi otak dan syaraf. Agar dapat memenuhi fungsinya, maka karbohidrat harus dipenuhi dalam menu sehari-hari.

2.2.2 Protein

Protein juga merupakan komponen penting atau komponen utama pada sel hewan atau manusia. Protein yang terdapat dalam makanan berfungsi sebagai zat utama dalam pembentukan dan pertumbuhan tubuh, pemeliharaan jaringan tubuh, sebagai pengatur proses dalam tubuh, sebagai sumber energi, dan sebagai pertahanan tubuh. Dalam satu gram protein, dihasilkan energi sebesar 4 kkal .

2.2.3 Lemak

Lemak merupakan sumber cadangan energi selain karbohidrat dan protein. Dalam satu gram lemak terdapat 9 kkal energi. Peranan lemak dalam tubuh, di antaranya adalah sebagai sumber penghasil energi, dan zat pembangun/ pembentuk struktur tubuh, dan pengatur suhu tubuh. Kelebihan lemak pada tubuh seseorang, bisa menimbulkan resiko penyakit jantung koroner dan *stroke*, sebaliknya jika kekurangan lemak, dapat mengganggu metabolisme pada tubuh.

2.3 Algoritma Genetika

Algoritma genetika pertama kali diperkenalkan oleh John Holland dari Universitas Michigan pada tahun 1975. Algoritma genetika adalah cabang dari algoritma evolusi yang merupakan metode *adaptive* dan bisa digunakan untuk memecahkan suatu pencarian nilai dalam masalah optimasi. Algoritma ini mengadaptasi istilah dalam proses genetik yang ada dalam makhluk hidup yaitu seleksi alam [GEN-00].

Algoritma genetika mempunyai 5 komponen penting, yaitu [MIC-96] :

1. Representasi genetik sebagai solusi dari sebuah masalah
2. Cara untuk membangkitkan populasi awal
3. Fungsi untuk mengevaluasi solusi dengan nilai *fitness* yang dimiliki masing-masing individu
4. Beberapa operator genetika yang menghasilkan keturunan/*offspring*
5. Nilai untuk beberapa parameter yang digunakan dalam algoritma genetika (ukuran populasi, nilai probabilitas yang digunakan dalam operator genetika, dll).

2.3.1 Struktur Algoritma Genetika

Pada algoritma genetika, teknik pencarian dilakukan sekaligus atas sejumlah kemungkinan solusi yang dikenal dengan istilah populasi. Solusi yang terdapat dalam satu populasi disebut juga dengan kromosom atau individu. Populasi awal dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evolusi kromosom-kromosom melalui iterasi yang disebut dengan generasi. Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang disebut dengan fungsi *fitness*. Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut [KUS-03].

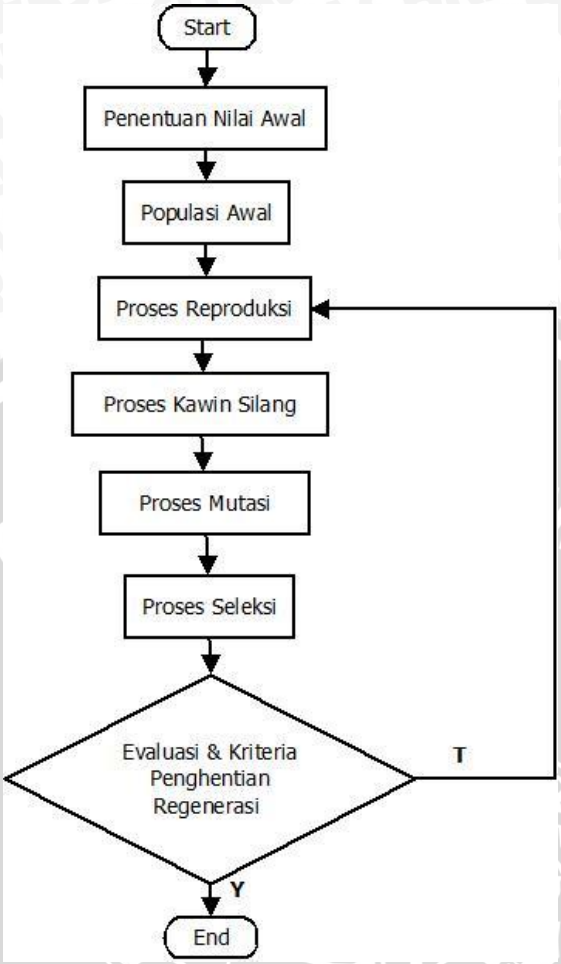
Individu berikutnya dalam sebuah populasi dikenal dengan istilah anak (*offspring*) yang terbentuk dari proses *crossover*. Dalam operator *crossover* dibutuhkan dua individu sebagai *parent*, sedangkan dalam proses mutasi hanya diperlukan satu individu untuk menghasilkan individu baru melalui modifikasi kromosomnya [MIC-96]. Dalam populasi, generasi baru dibentuk dengan cara

menyeleksi nilai *fitness* dari semua individu, termasuk individu hasil *crossover* dan mutasi yang dianggap bagus, serta menolak individu lainnya sehingga ukuran populasi selalu konstan [KUS-03].

Pencarian solusi dalam algoritma genetika, mempunyai karakteristik sebagai berikut [ZUL-04] :

1. Algoritma genetik bekerja berdasar pada metode pengkodean untuk sekumpulan parameter yang berkaitan
2. Algoritma genetik melakukan pencarian dari sejumlah titik yang tergabung dalam satuan populasi, bukan pada titik pencarian tunggal
3. Algoritma genetik menggunakan fungsi obyektif (fungsi *fitness*)
4. Algoritma genetik menggunakan aturan transisi dengan probabilitas, bukan dengan aturan deterministik.

Hal yang perlu diingat dalam algoritma genetika adalah algoritma genetika merupakan algoritma yang dikembangkan dari proses pencarian solusi menggunakan pencarian acak, hal ini terlihat pada proses pembangkitan populasi awal yang menyatakan sekumpulan solusi yang bisa dipilih secara acak. Selanjutnya pencarian dilakukan berdasarkan proses-proses teori genetika yang memperhatikan pemikiran bagaimana memperoleh individu yang lebih baik, sehingga dalam proses evolusi diharapkan diperoleh individu yang terbaik [BAS-03]. Secara umum, blok diagram dari mekanisme kerja algoritma genetika dapat dilihat dalam gambar 2.1 [HAN-02].



Gambar 2. 1 Mekanisme Kerja Algoritma Genetika

2.3.2 Representasi Kromosom

Hal paling penting pada penggunaan algoritma genetika untuk menyelesaikan suatu permasalah adalah bagaimana mengkodekan permasalahan ke dalam kromosom [GEN-00]. Hal ini disebabkan karena representasi kromosom dalam setiap permasalahan berbeda-beda dan tidak semua model representasi cocok untuk setiap permasalahan. Representasi kromosom sendiri adalah suatu teknik penyandian gen dalam kromosom. Representasi gen ini sendiri bisa dalam bentuk *string bit*, pohon, *array*, bilangan *real*, dan lain-lain [KUS-03].

Menurut Guz Eiben dan Jim Smith, jenis representasi kromosom adalah sebagai berikut [GUZ-03] :

A. Representasi Biner

Representasi ini merupakan bentuk paling sederhana dan umum digunakan untuk penyelesaian masalah yang tidak terlalu kompleks. Dalam representasi ini, gen hanya bernilai 1 atau 0 sehingga operator *crossover* dan mutasi yang digunakan pun akan sederhana.

Kromosom X	10110001110001110001 10
Kromosom Y	00110101011111000000 0

Gambar 2. 2 Representasi biner pada kromosom

Dari Gambar 2.2, dapat dijelaskan bahwa terdapat dua kromosom yaitu Kromosom X dan Kromosom Y dimana gen nya bernilai 1 atau 0 .

B. Representasi *Integer*

Tidak semua permasalahan dapat dikodekan dengan bentuk biner, misalnya masalah yang variabel penyusunnya adalah *integer*. Pada representasi ini, setiap gen bernilai bulat (*integer*) yang merepresentasikan jenis atau kuantitas objek. Dalam Gambar 2.3 dijelaskan bahwa terdapat dua kromosom yaitu Kromosom X dan Kromosom Y dimana gen-gen nya bernilai bulat.

Kromosom X	3 2 4 7 5 11 9 27 1
Kromosom Y	5 9 1 4 27 6 7 14 2

Gambar 2. 3 Representasi *integer* pada kromosom

C. Representasi *Real*

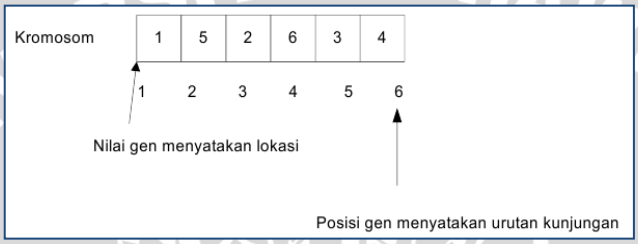
Representasi *real* bisa digunakan ketika representasi biner dan *integer* tidak bisa mencapai ketelitian yang diinginkan. Dalam representasi *real*, setiap gen bisa bernilai *real* dalam interval [0,1]. Penggunaan representasi *real* ditunjukkan pada Gambar 2.4 berikut

Kromosom X	0,234	0,456	1,0	0,541
Kromosom Y	0,165	0,269	0,876	1,0

Gambar 2. 4 Representasi *real* pada kromosom

D. Representasi Permutasi

Dalam permasalahan tertentu, seperti TSP (*Travelling Salesman Problem*) misalnya, representasi biner, *integer* maupun *real* tidak bisa digunakan karena dalam solusi permasalahan ini yang dicari adalah bagaimana menemukan kunjungan lokasi dan yang total nilainya paling optimal. "Nilai" di sini bisa berupa jarak, kenyamanan, biaya dan sebagainya. Dalam membangun representasi permutasi ini, ada hal-hal yang harus diperhatikan, yaitu "satu kromosom harus menyatakan satu solusi", selain itu juga "posisi" dan "nilai" gen. Posisi gen (indeks pada kromosom) bisa digunakan untuk menyatakan kunjungan lokasi.



Gambar 2. 5 Representasi permutasi pada kromosom

Untuk permasalahan kombinasi menu makanan ini akan dicari bobot yang optimal dalam gen. Bobot pada tiap jenis makanan bernilai *real*, sehingga representasi kromosom yang digunakan adalah representasi *real* untuk mengkodekan susunan bobot ke dalam kromosom.

2.4 Mendefinisikan Nilai *Fitness*

Nilai *fitness* adalah nilai yang menyatakan baik atau tidaknya suatu solusi (individu). Nilai inilah yang dijadikan acuan untuk mencapai nilai optimal dalam algoritma genetika. Dalam *Travelling Salesman Problem* (TSP) misalnya, TSP bertujuan untuk meminimalkan jarak, maka nilai *fitness*nya adalah inversi dari jarak [BAS-03].

Menurut Suyanto, fungsi *fitness* (f) bisa berupa fungsi yang memaksimalkan dan meminimalkan fungsi h, dimana keduanya sama dan bersifat kebalikan. Dalam permasalahan yang memaksimalkan fungsi f (maka akan meminimalkan fungsi h), fungsi ini tidak dapat langsung dibentuk sebagai fungsi *fitness* karena fungsi ini memiliki arti bahwa individu yang memiliki *fitness* yang

tinggi adalah individu yang mampu bertahan hidup dalam sebuah populasi. Secara matematis, fungsi maksimasi tersebut dituliskan dalam persamaan 2.1 berikut :

$$f = \frac{1}{(h+a)} \qquad (2.1)$$

Dengan a adalah bilangan kecil yang bervariasi sesuai dengan permasalahan yang akan diselesaikan [SUY-05],.

2.5 Pembangkitan Populasi Awal

Pembangkitan populasi awal merupakan suatu proses membangkitkan sejumlah individu secara acak atau menggunakan metode-metode tertentu dalam sebuah populasi. Dalam penentuan ukuran populasi juga tergantung dari masalah dan operator genetika yang akan digunakan. Berikut ini akan dijelaskan beberapa teknik pembangkitan populasi awal [ANO^2-12].

2.5.1 *Random Generator*

Dalam teknik pengkodean ini, nilai gen akan dibangkitkan secara *random* sesuai dengan representasi kromosom yang digunakan.

2.5.2 Pendekatan Tertentu

Cara ini adalah dengan memasukkan nilai tertentu sesuai dengan permasalahan yang akan diselesaikan ke dalam gen dari populasi awal yang dibentuk.

2.5.3 Permutasi Gen

Salah satu cara permutasi gen dalam pembangkitan populasi awal adalah penggunaan permutasi *Josephus* dalam permasalahan kombinatorial seperti TSP. Sebagai contoh permutasi Jose*p*hus, misalkan ada 9 kota, permutasi lintasan dapat dilakukan dengan menetukan titik awal dan selang. Jika titik awal 6 dan selang adalah 5, maka lintasan berangkat dari kota 6, selang 5 dari kota 6 adalah kota 2 (asumsi kota 1 – 9 berupa *circular List*). Selanjutnya kota 2 akan dilewati untuk menghitung selang 5 yang kedua, dan ditemukan kota 7. Proses ini diulang hingga ada satu lintasan dalam *List*. Hasil permutasi ini adalah 2-7-3-8-4-9-5-1-6.

2.6 Perkawinan Silang/*Crossover*

Proses perkawinan silang (*crossover*) bertujuan menghasilkan anak/keturunan. Dalam *crossover*, diperlukan dua kromosom terpilih sebagai induk atau *parent*. Dalam proses *crossover* ini, akan dihasilkan dua kromosom anak. Kromosom anak yang dihasilkan merupakan kombinasi gen dari dua kromosom induk. Dalam *crossover*, terdapat nilai probabilitas *crossover* (Pc) yang akan menentukan frekuensi dari *crossover* yang terjadi dalam sebuah populasi. Semakin tinggi nilai Pc maka akan semakin besar kemungkinan dilakukan kawin silang [SET-03].

Dalam proses *crossover* jumlah populasi sangat mempengaruhi hasil dari proses *crossover*. Jumlah populasi yang sangat kecil akan berakibat buruk, hal itu akan menyebabkan suatu kromosom dengan gen – gen yang mengarah pada solusi akan sangat cepat menyebar ke kromosom–kromosom yang lain. Untuk mengatasi masalah tersebut digunakan suatu aturan bahwa proses *crossover* hanya dapat dilakukan dengan suatu probabilitas tertentu, artinya *crossover* hanya dapat dilakukan dengan membangkitkan bilangan *random* [0..1] dan nilai bangkitan *random* adalah kurang dari nilai Pc yang telah ditentukan. Pada umumnya, nilai Pc adalah mendekati 1 [SUY-05].

Secara umum, mekanisme *crossover* adalah sebagai berikut [SET-03] :
Memilih dua buah kromosom sebagai induk

1. Memilih secara acak posisi dalam kromosom, biasa disebut *crossover point*, sehingga masing-masing kormosom induk terbagi menjadi dua segmen
2. Lakukan pertukaran antar bagian kromosom induk untuk menghasilkan kromosom anak.

Dalam penggunaannya, teknik *crossover* akan disesuaikan dengan teknik pengkodean kromosom. Metode *crossover* pada *floating point representation* di antaranya adalah *simple arithmetic crossover*, *single arithmetical crossover*, dan *arithmetical crossover*. Berikut akan dijelaskan mengenai *simple arithmetic crossover* [GEN-00].

2.6.1 *Simple Arithmetic Crossover*

Crossover dua buah *parent* yang dilakukan dengan menukarkan sebesar k-bagian satu sama lain untuk menghasilkan anak (*child*), dimana sebelumnya akan ditentukan titik potong ke-k.

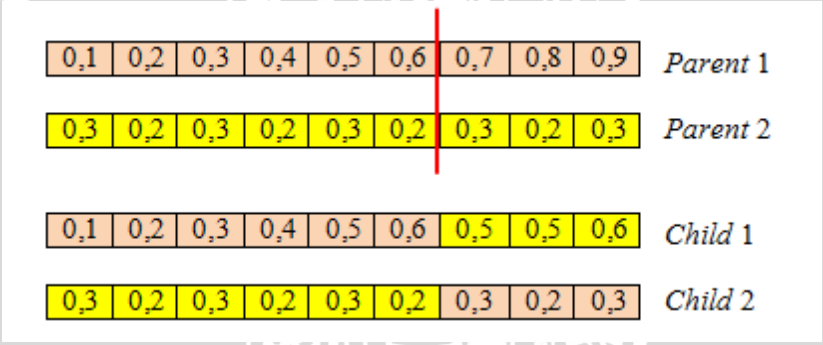
$$x'_1 = \langle x_1, \dots, x_k, y_{k+1} \cdot a + x_{k+1} \cdot (1-a), \dots, y_q \cdot a + x_q \cdot (1-a) \rangle \quad (2.2)$$

$$x'_2 = \langle y_1, \dots, y_k, x_{k+1} \cdot a + y_{k+1} \cdot (1-a), \dots, x_q \cdot a + y_q \cdot (1-a) \rangle \quad (2.3)$$

Dimana :

X = *parent* 1 ; Y= *parent* 2 ; X'1 = *child* 1 ; X'2 = *child* 2 ;

α = nilai *random* antara 0 .. 1



Gambar 2. 6 *Simple arithmetic crossover* dengan α = 0,5

2.7 Mutasi

Mutasi merupakan bentuk operator genetika yang menukar nilai gen dengan nilai gen yang lain, dalam mutasi biner misalnya, gen yang bernilai 0 menjadi bernilai 1. Proses ini dilakukan secara acak pada posisi gen tertentu pada individu-individu yang terpilih untuk dimutasikan. Selain mutasi yang bernilai biner, terdapat juga mutasi yang bernilai *real*. Banyaknya individu yang mengalami mutasi ditentukan oleh besarnya probabilitas mutasi [BAS-03] .

Dalam representasi *integer*, ada beberapa teknik mutasi, diantaranya adalah *random resetting*. Dalam *random resetting*, akan dibangkitkan nilai Pm

secara *random* dalam tiap gen sepanjang kromosom, dimana sebelumnya sudah ditentukan Pm, yang menjadi batasan, lalu nilai Pm yang kurang dari Pm yang ditentukan tadi akan dimutasi. Dalam representasi *real* atau *floating point* terdapat metode mutasi *uniform mutation*, teknik ini dianalogikan sama dengan *random resetting* seperti pada reperesentasi *integer* [GEN-00].

Proses mutasi pada penelitian ini menggunakan metode mutasi *random* dengan memodifikasi dari penelitian yang dilakukan oleh Lamhot Pranata S. Adapun langkah-langkah dari metode mutasi dalam penelitian sebelumnya adalah sebagai berikut [PRA-11] :

1. Membangkitkan bilangan *random* r [0..1] sebanyak jumlah individu.
2. Apabila bilangan *random* r $<$ *pm*, maka individu tersebut terpilih untuk dikenai mutasi.
3. Memilih gen yang akan dimutasi (k) secara *random*.
4. Membangkitkan bilangan *random* antara 1 hingga jumlah bahan pakan sebanyak 1 bilangan (m) dengan ketentuan bilangan (m) merupakan bilangan yang belum pernah terpakai pada populasi awal.
5. Mengganti nilai pada gen k dengan nilai m.

2.8 Seleksi

Proses seleksi akan melakukan pemilihan terhadap individu yang akan diikutkan dalam proses reproduksi. Pada proses seleksi, keanekaragaman populasi memegang peranan penting dalam proses seleksi yaitu daerah *sampling*, probabilitas seleksi, dan mekanisme seleksi [SET-03].

Perlu diingat, sebelum dilakukan seleksi jumlah anggota populasi ditambah dengan hasil *offspring* dari proses operasi genetik *crossover* dan mutasi. Hasil operasi genetik dan populasi semula selanjutnya di seleksi dengan metode tertentu untuk diambil sejumlah n anggota populasi yang terbaik sesuai dengan jumlah populasi awal [WID-07] .

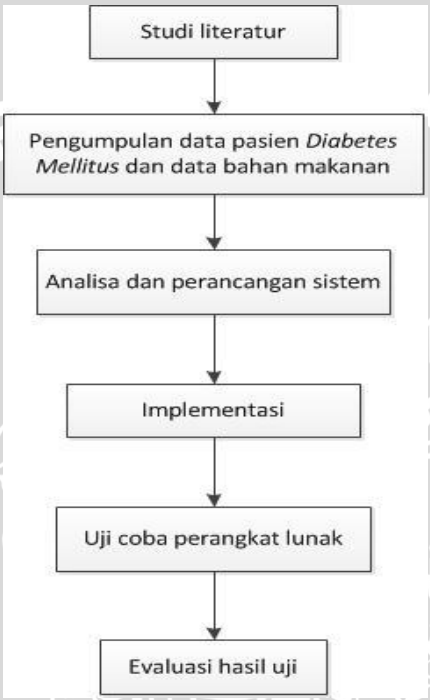
Beberapa metode dalam seleksi adalah *Tournament Selection, Roulette Wheel Selection, Rank Based Fitness Selection, Truncation Selection, Seleksi Lokal* dan *Stocastic Universal Sampling*. Berikut akan dijelaskan tentang *Rank Based Fitness Selection* [KUS-03].

2.8.1 *Rank Based Fitness Selection*

Dalam metode seleksi ini, individu dalam populasi akan diurutkan menurut nilai objektifnya. Pada proses seleksi ini digunakan suatu parameter yang disebut fungsi *fitness*. Fungsi *fitness* digunakan untuk menentukan seberapa baik kromosom akan bertahan hidup. Pada *Rank Based Fitness Selection*, hanya kromosom yang mempunyai nilai *fitness* tinggi yang dapat bertahan pada generasi berikutnya sebaliknya yang mempunyai nilai *fitness* rendah akan hilang pada generasi berikutnya.

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab metodologi dan perancangan ini akan dibahas metode, rancangan yang digunakan dan langkah-langkah yang dilakukan dalam pembuatan sistem penentuan komposisi menu makanan untuk penderita *Diabetes Mellitus* menggunakan algoritma genetika.



Gambar 3. 1 Langkah-langkah Penelitian

Berdasarkan gambar 3.1, langkah-langkah yang dilakukan dalam penelitian ini adalah dengan melakukan :

1. Studi literatur mengenai algoritma genetika serta literatur lain yang berkaitan seperti yang telah dijelaskan pada bab 2.
2. Pengumpulan data-data yang diperlukan dalam penelitian, yaitu data pasien *Diabetes Mellitus* dan data bahan makanan.
3. Analisa dan perancangan sistem.
4. Implementasi hasil analisis dan perancangan yang telah dilakukan dalam bentuk perangkat lunak.
5. Uji coba terhadap perangkat lunak.
6. Evaluasi hasil yang diperoleh dari uji coba perangkat lunak.

3.1 Analisa Sistem

3.1.1 Deskripsi umum sistem

Sistem yang akan dimodelkan ini merupakan sistem yang mengimplementasikan algoritma genetika dalam penyusunan kombinasi menu makanan untuk penderita *Diabetes Mellitus* II dan tanpa komplikasi penyakit sesuai dengan jumlah kalori yang sudah dihitung dari tiap penderita. Dalam sistem ini, pengguna akan memasukkan data berupa umur, jenis kelamin dan berat badan yang akan digunakan untuk penghitungan kalori. *Output* yang dihasilkan berupa menu makanan dengan pembobotan dan kandungan gizi sesuai dengan kebutuhan tubuh pasien/penderita. Tahapan sistem yang akan dibangun dengan menggunakan algoritma genetika ini adalah sebagai berikut :

1. Representasi kromosom
2. *Crossover* untuk mendapatkan individu baru
3. Mutasi untuk membuat variasi dalam populasi
4. Evaluasi dari semua individu dengan menggunakan fungsi *fitness*
5. Seleksi untuk membentuk populasi baru

3.1.2 Data Penelitian

Data yang akan digunakan dalam skripsi ini adalah:

1. Data pasien penderita *Diabetes Mellitus* tipe II di Rumah Sakit Baptis Kediri selama periode Januari-April 2012, di mana data ini akan digunakan untuk menghitung kalori yang dibutuhkan tiap penderita melalui penghitungan dengan rumus dalam Tabel 2.1. Atribut dalam data penderita *Diabetes Mellitus*, terdiri dari nama, umur, berat badan dan jenis kelamin.
2. Data bahan pangan yang digunakan untuk menyusun menu makanan yang didapatkan dari Tabel Komposisi Makanan Indonesia serta makalah hasil penelitian oleh Pusat Penelitian dan Pengembangan Gizi dan Makanan Departemen Kesehatan RI. Komponen yang akan digunakan dalam data pangan tersebut adalah Karbohidrat, Protein dan Lemak, karena dalam studi penelitian ini, permasalahan komposisi akan dipertimbangkan dari komponen tersebut sebagai penyusun zat makro dalam makanan.

3. Data Aturan, merupakan data kriteria kuantitatif untuk menentukan aturan berat komposisi penyusun menu makanan berdasarkan kalori yang dibutuhkan.
4. Data Algoritma Genetika, terdiri dari:
   a. Jumlah generasi
   b. Ukuran populasi (*popsize*)
   c. Probabilitas *crossover* (Pc)
   d. Probabilitas mutasi (Pm)
   e. Nilai *alpha*

3.2 Perancangan Sistem

Pada subbab ini akan dijelaskan proses-proses yang terjadi pada sistem.

3.2.1 Proses penyelesaian penentuan kombinasi menu makanan untuk *Diabetes Mellitus* menggunakan aloritma genetika

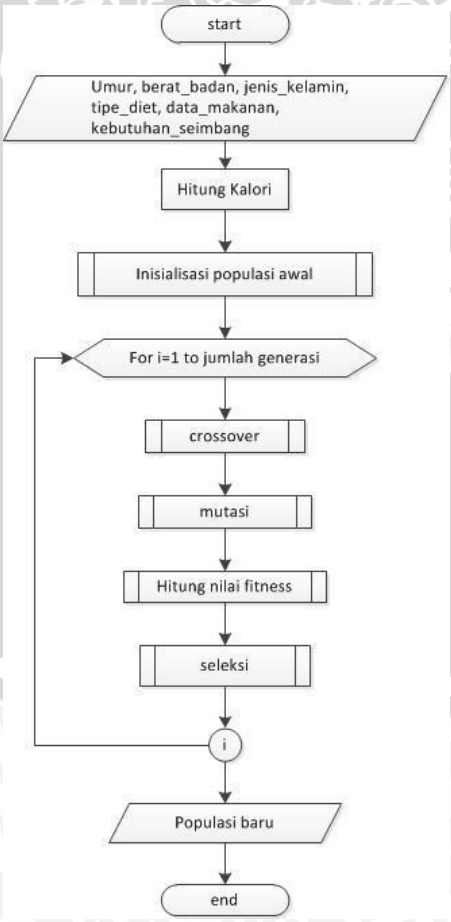
1. Proses penghitungan kalori tiap pasien

Dalam proses penghitungan ini, akan digunakan rumus regresi linier dari *FAO/WHO/UNU*/1985 yang menekankan pada faktor berat badan berdasarkan umur seperti pada Tabel 2.1.

2. Menentukan semua parameter awal, yaitu :
   - Data Pasien
   - Peluang *crossover* (Pc)
   - Nilai *alpha*
   - Peluang mutasi (Pm)
   - Jumlah populasi
   - Jumlah generasi
3. Membangkitkan populasi awal secara *random* sebanyak ukuran populasi yang ditentukan

4. Membentuk populasi baru dengan langkah-langkah sebagai berikut :
   - Melakukan *crossover* dengan metode simple arithmetic *crossover* pada orang tua (*parent*) yang telah terpilih sehingga mendapatkan anakan (*offspring*)
   - Melakukan mutasi pada individu terpilih dengan metode *random mutation*
   - Melakukan evaluasi pada semua individu, termasuk individu hasil *crossover* dan mutasi dengan menggunakan fungsi *fitness*
5. Melakukan seleksi pada semua individu untuk mendapatkan populasi baru dengan metode *rank based fitness selection*
6. Memakai populasi baru untuk iterasi pada langkah 4
7. Melakukan proses tersebut sesuai dengan jumlah generasi (iterasi) yang diinginkan

Diagram alir sistem dari langkah-langkah penyusunan kombinasi menu makanan dengan algoritma genetika di atas akan ditunjukkan pada Gambar 3.2.



Gambar 3. 2 Diagram Alir Sistem untuk Algoritma Genetika

3.2.2 Representasi individu

Individu terdiri dari gen-gen yang merupakan suatu pengkodean yang digunakan untuk merepresentasikan masalah kombinasi menu makanan untuk penderita *Diabetes Mellitus*. Dalam hal ini, individu adalah menu makanan dalam sehari sesuai kebutuhan gizi seseorang dari kalori yang sudah dihitung.

Teknik *encoding* yang akan digunakan dalam penelitian ini adalah representasi *real* atau *floating point*. Digunakan teknik representasi ini, karena setiap gen yang terdapat dalam sebuah individu berisi berat makanan yang diskalakan dalam rentang nol sampai dengan satu. Satu individu terdiri dari tiga kromosom yang memiliki 15 gen penyusun dengan urutan yang sudah ditentukan dan dikodekan. Misalnya suatu makanan pokok akan dikodekan dengan indeks 1 maka gen tersebut akan merujuk pada data dengan indeks 1 pada Tabel 3.1 yaitu nasi (untuk bahan makanan lain, akan dilampirkan pada Lampiran 1). Dalam setiap gen masih terdapat komponen penyusunnya yaitu karbohidrat, lemak dan protein, dimana nilainya bergantung dari bobot makanan dan nantinya akan digunakan untuk menghitung nilai *fitness*.

Tabel 3. 1 Daftar makanan pokok per 100gr bahan makanan

Indeks	Nama Makanan	Energi (kkal)	Protein (gr)	Lemak (gr)	Karbohidrat (gr)
0	Nasi tim	120	2,4	0,4	26
1	Nasi	180	3	0,3	39,8
2	Bihun	348	4,7	0,1	82,1
3	Nasi beras merah	149	2,8	0,4	32,5
4	Kentang	62	2,1	0,2	13,5
5	Mie basah	88	0,6	3,3	14
6	Ketupat	109	2,2	5,2	13,4

Makan Pagi				Makan Siang				Makan Malam				Pelengkap		
S	N	H	Y	S	N	H	Y	S	N	H	Y	P1	P2	P3

Gambar 3. 3 Representasi Kromosom untuk 1 Individu

Pagi , Siang, Malam

S			N			H			Y			Pelengkap		
K1	L1	R1	K2	L2	R2	K3	L3	R3	K4	L4	R4	K5	L5	R5

Gambar 3. 4 Representasi Komponen dalam tiap bahan makanan

Ket :

S = Makanan Pokok H = Lauk Hewani

N = Lauk Nabati Y = Sayur

P1,P2,P3 = Pelengkap

K1,K2,K3,K4,K5 = berat komponen karbohidrat

L1,L2.L3,L4,L5 = berat komponen lemak

R1,R2,R3,R4,R5 = berat komponen protein

3.2.3 Pembangkitan populasi awal

Populasi merupakan kumpulan dari beberapa individu. Langkah pertama yang dilakukan adalah membangkitkan individu dalam sebuah populasi. Setiap gen dalam kromosom akan dibangkitkan secara acak, yaitu pada bahan dan bobot makanan sesuai dengan indeks yang sudah ditentukan dalam tabel. Misalkan, jumlah individu dalam populasi awal adalah 5, maka populasinya adalah sebagai berikut :

Individu 1 : 2313 1482 1421 281

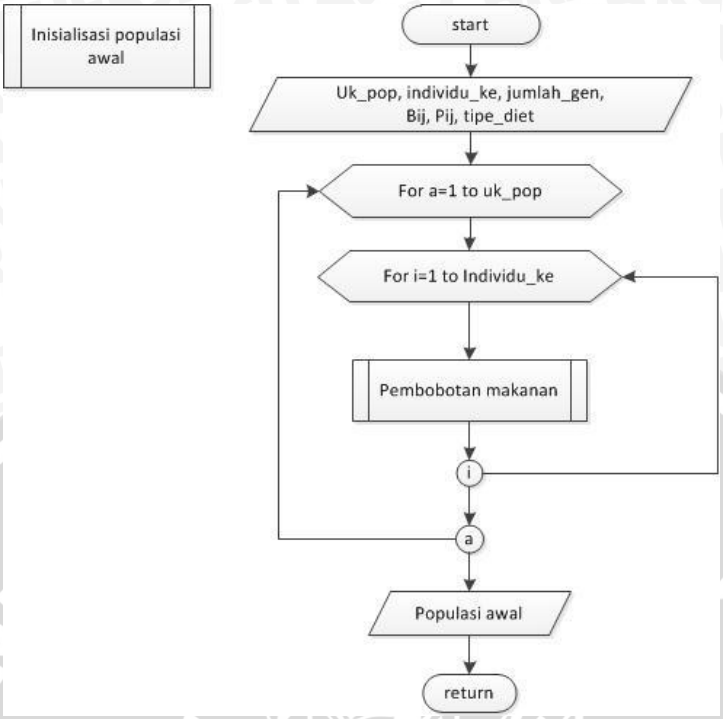
Individu 2 : 1243 3221 4123 305

Individu 3 : 1924 7179 1529 187

Individu 4 : 4411 1413 1621 298

Individu 5 : 1392 5992 2823 650

Dari individu 1 maka menu makanan yang dihasilkan untuk makan pagi= bihun, tempe murni, ayam, buncis, makan siang = nasi, kacang merah, bandeng presto, sawi, makan malam = nasi, kacang merah, daging sapi, bayam, pelengkap = apel, alpukat, manggis. *Flowchart* dari pembangkitan populasi awal akan ditunjukkan pada Gambar 3.5 berikut ini :



Gambar 3. 5 *Flowchart* pembangkitan populasi awal secara *random*

3.2.4 Pembobotan makanan

Sesuai dengan pengkodean kromosom yang digunakan, maka pembobotan makanan akan dilakukan secara *random* dari skala nol sampai dengan satu. Skala ini menunjukkan rentang maksimum berat makanan yang dapat dikonsumsi oleh seseorang dalam sekali waktu makan. Dari hasil wawancara dengan ahli gizi, rentang untuk makanan pokok adalah 0-180 gr, sumber lauk nabati 0-50 gr, sumber lauk hewani 0-80 gr, sayur 0-200 gr dan makanan pelengkap 0-150 gr .Langkah-langkah pembobotan makanan adalah sebagai berikut :

1. Untuk masing-masing gen dalam individu, bangkitkan jenis makanan secara acak.
2. Bangkitkan bilangan acak [0 .. 1] dalam tiap gen tersebut sebagai inisialisasi bobot makanan
3. Hitung bobot sebenarnya dengan aturan rentang untuk makanan pokok adalah 0-180 gr, untuk sumber lauk nabati 0-50 gr, sumber lauk hewani 0-80 gr, sayur 0-200 gr dan makanan pelengkap 0-150 gr

4. Hitung berat masing-masing komponen penyusun (karbohidrat, lemak dan protein) makanan sesuai dengan berat sebenarnya

Misalkan pada Individu 1 :

Makan Pagi				Makan Siang				Makan Malam				Pelengkap		
S	N	H	Y	S	N	H	Y	S	N	H	Y	P1	P2	P3
2	3	1	3	1	4	8	2	1	4	2	1	2	8	1
0,4	0,5	0,1	0,2	0,8	0,1	0,6	0,5	0,5	0,4	0,3	0,5	0,3	0,6	0,7

Gambar 3. 6 Representasi Individu 1

Maka bobot makanan adalah sebagai berikut :

Untuk makan pagi =

S → 0,4 * 180 = 72 gr

N → 0,5 * 50 = 25 gr

H → 0,1 * 80 = 8 gr

Y → 0,2 * 200 = 40 gr

Untuk makan siang =

S → 0,8 * 180 = 144 gr

N → 0,1 * 50 = 5 gr

H → 0,6 * 80 = 48 gr

Y → 0,5 * 200 = 40 gr

Untuk makan malam =

S → 0,5 * 180 = 90 gr

N → 0,4 * 50 = 20 gr

H → 0,3 * 80 = 24 gr

Y → 0,5 * 200 = 40 gr

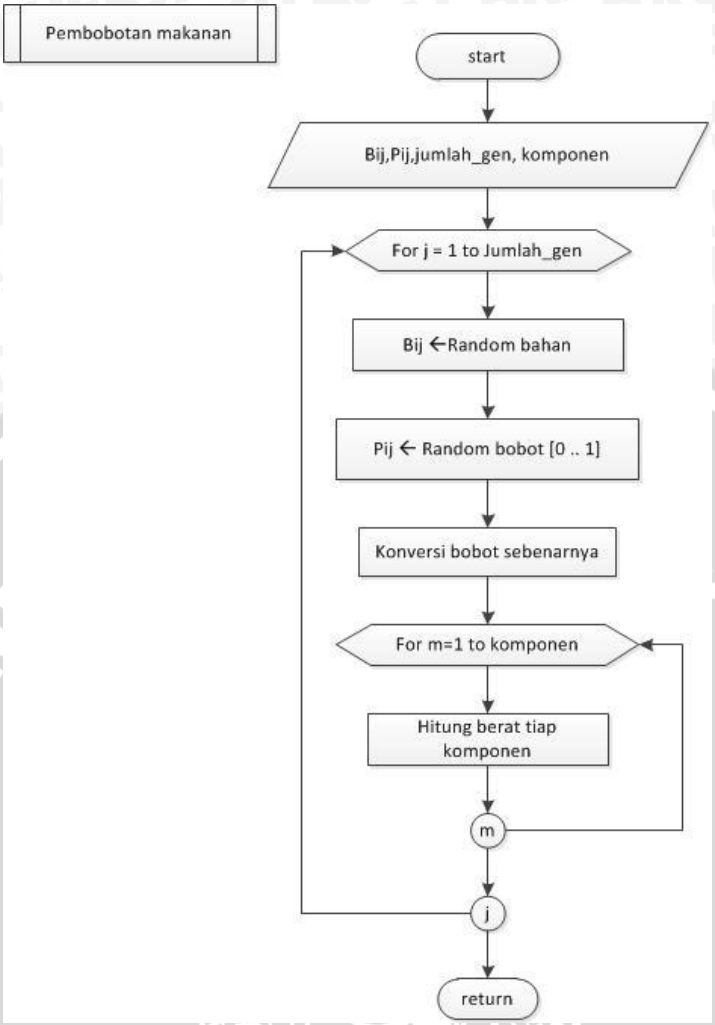
Untuk pelengkap =

P1→ 0,3 * 150 = 45 gr

P2→ 0,6 * 150 = 90 gr

P3→ 0,7 * 150 = 105 gr

Flowchart dari pembobotan makanan yang akan ditunjukkan pada Gambar 3.7 berikut ini :

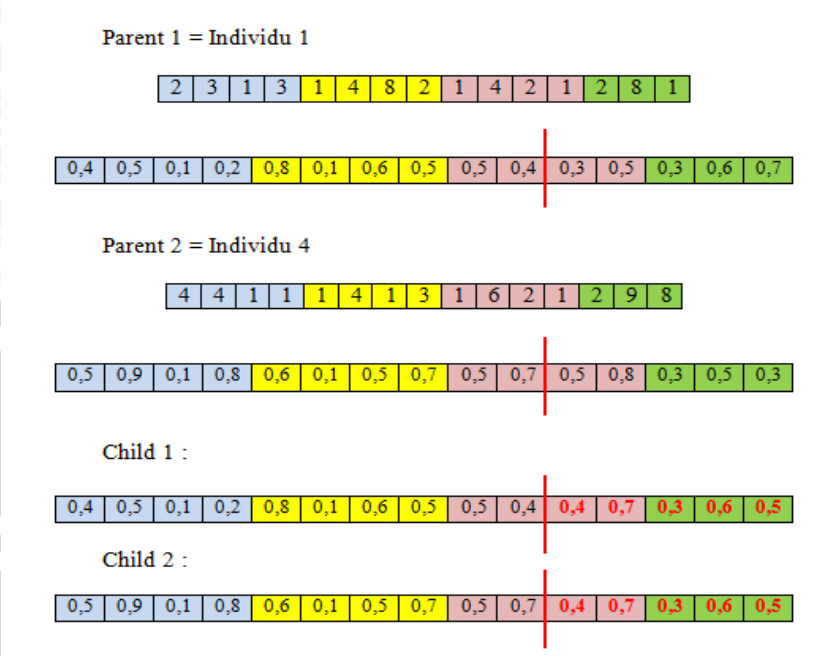


Gambar 3. 7 *Flowchart* pembobotan makanan

3.2.5 *Crossover*

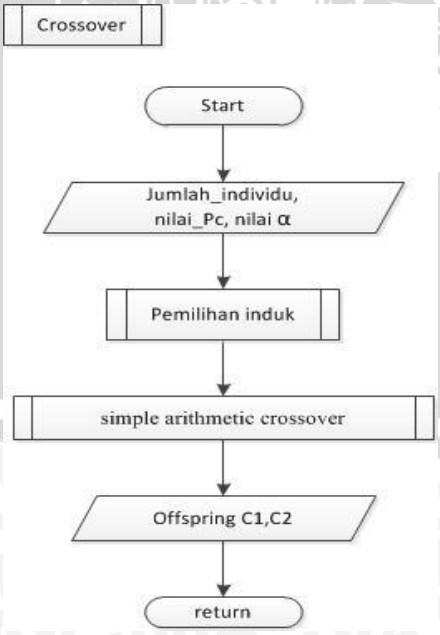
Dalam operator *crossover*, dipilih metode *simple arithmetic crossover* yaitu metode persilangan dimana keturunan (*offspring*) didapat dengan melakukan operasi pada nilai gen dari *parent* pada titik tertentu sesuai dengan persamaan 2.2 dan 2.3. Untuk pemilihan *parent* dalam *crossover* digunakan nilai Probabilitas *Crossover* (Pc). Pertama akan dibangkitkan nilai Pc secara *random* untuk setiap individu, lalu akan diberikan nilai *input* Pc misal 0,8 berikutnya akan dilakukan pengecekan individu mana yang memiliki nilai Pc < 0.8 . Individu-individu itulah yang menjadi *parent* lalu dikombinasikan 2 pada proses *crossover*. Dalam metode ini akan juga digunakan nilai α (nilai α berkisar antara 0 sampai dengan 1) dan

akan dicari titik potong dalam kromosom dengan cara *random* lalu akan dihitung sesuai dengan persamaan 2.2 dan 2.3. Ilustrasinya akan ditunjukkan dalam gambar 3.8 berikut :

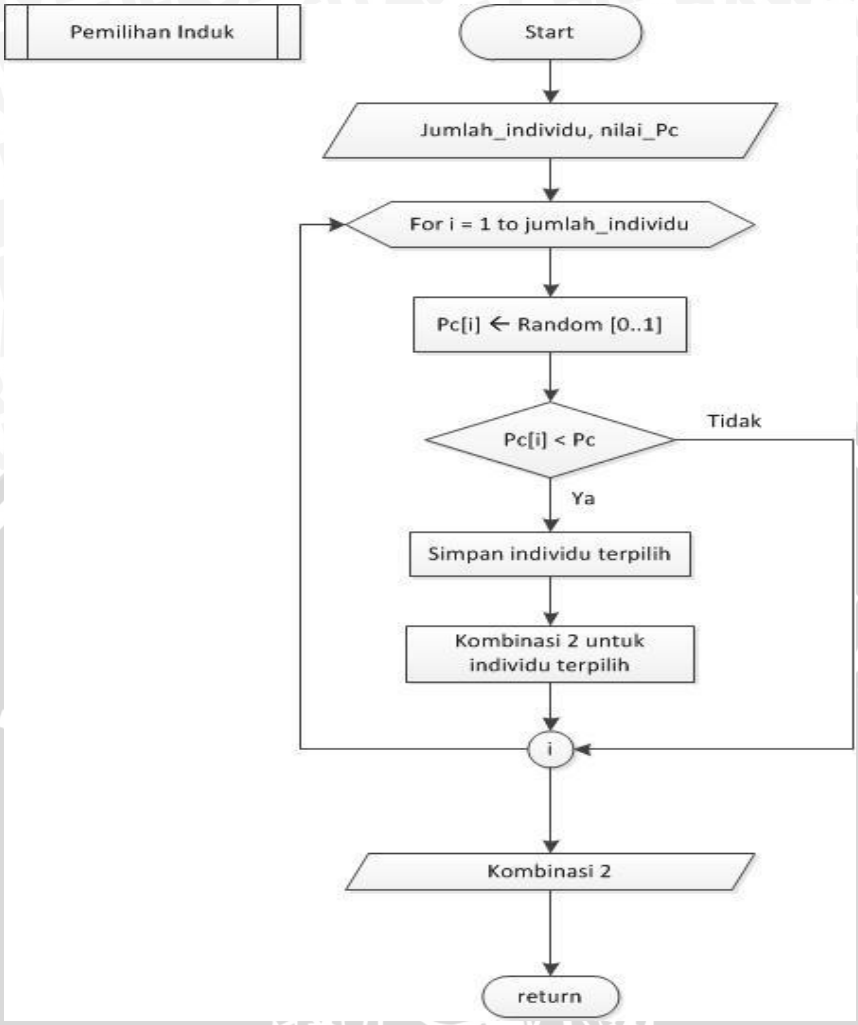


Gambar 3. 8 *Crossover* dengan metode *simple arithmetic crossover* dimana nilai α=0.5

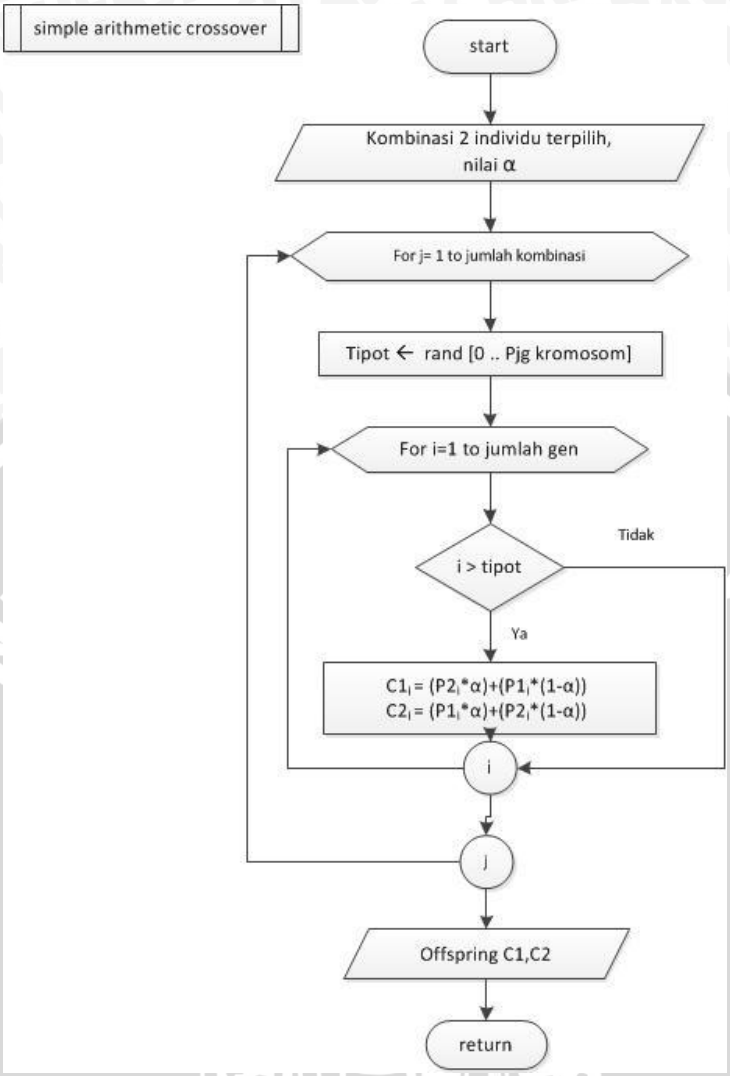
Flowchart untuk proses *crossover* akan ditunjukkan dalam Gambar 3.9 dan *Flowchart* untuk metode *simple arithmetic crossover* pada gambar 3.10



Gambar 3. 9 *Flowchart* untuk proses *crossover*



Gambar 3. 10 *Flowchart* untuk proses pemilihan induk



Gambar 3. 11 *Flowchart* untuk proses *simple arithmetic crossover*

Penjelasan langkah-langkah proses *simple arithmetic crossover* yang digunakan adalah sebagai berikut:

1. Masukkan dua *parent* yang akan di-*crossover*.
2. Tentukan bilangan *random* sebagai titik potong antara 0 sampai sepanjang kromosom pada masing-masing *parent*.
3. Untuk setiap gen pada kromosom *parent*, lakukan:
   a. Jika indeks kromosom terletak sebelum titik potong, salin nilai gen *parent* untuk diberikan pada anakan (*offspring*).
   b. Jika indeks kromosom berada pada titik setelah titik potong, lakukan persamaan 2.2 dan 2.3 untuk mendapatkan nilai baru yang akan diberikan kepada anakan (*offspring*).

3.2.6 Mutasi

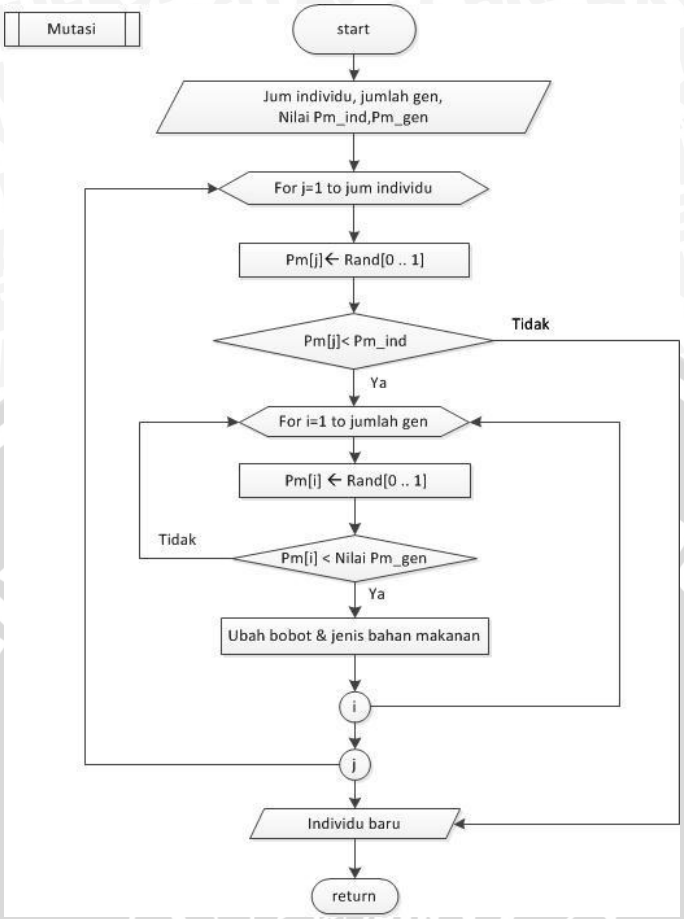
Dalam sebuah populasi semua individu bisa mengalami mutasi. Tingginya peluang mutasi akan ditentukan oleh nilai Peluang Mutasi (Pm) yang bernilai nol sampai dengan satu. Nilai Pm terdapat dalam tiap gen dan akan dibangkitkan secara *random*. Jika nilai Pm yang dibangkitkan dalam tiap individu bernilai kurang dari Pm individu yang ditentukan, maka gen yang berada dalam individu tersebut juga akan mengalami mutasi. Dalam penelitian ini akan digunakan metode *random mutation* terhadap jenis dan bobot makanan yang dihasilkan. Langkah-langkah dari proses mutasi yang digunakan adalah sebagai berikut :

1. Tentukan nilai Pm untuk individu
2. Generate bilangan *random* r [0..1] sebagai Pm sebanyak jumlah individu dalam popolusi awal ditambah *child* atau *offspring* untuk menentukan individu yang akan dimutasi.
3. Bandingkan hasil pembangkitan Pm masing-masing individu dengan nilai Pm yang ditentukan. Apabila hasil r < Pm yang ditentukan, maka individu tersebut merupakan kandidat terkena mutasi.

Untuk individu yang mengalami mutasi, lakukan :

1. Tentukan nilai Pm untuk setiap gen
2. *Generate* bilangan *random* r1 [0..1] sebagai Pm sebanyak jumlah gen dalam individu yang mengalami mutasi tadi untuk menentukan gen mana yang mengalami mutasi.
3. Bandingkan hasil pembangkitan Pm masing-masing individu dengan nilai Pm yang diinginkan. Apabila hasil r1 < Pm yang ditentukan, maka gen tersebut terkena mutasi.
4. Melakukan perhitungan nilai *fitness* pada individu hasil mutasi

Flowchart untuk metode ini akan ditunjukkan dalam gambar 3.12 .



Gambar 3. 12 Proses Mutasi dengan *random mutation*

3.2.7 Fungsi *fitness*

Fungsi *fitness* merupakan fungsi yang digunakan untuk mengevaluasi setiap individu yang terdapat dalam sebuah populasi. Fungsi *fitness* dalam studi kasus ini merupakan suatu fungsi maksimasi, sehingga semakin besar nilai *fitness*, maka individu akan semakin baik. Nilai *fitness* tersebut juga akan dihitung dengan menggunakan penalti, dimana penalti merupakan suatu cara untuk memberikan nilai terhadap pelanggaran yang dilakukan oleh setiap individu. Rumusan fungsi *fitness* yang digunakan akan ditunjukkan dalam persamaan 3.1.

$$\boldsymbol{fitness} = \frac{\mathbf{1}}{\mathbf{1} + \sum_{\mathbf{i=1}}^{\mathbf{3}} \mathbf{P_i}} \qquad (3.1)$$

Dengan : Pi = penalti ke-i

3.2.8 Pendefinisian Penalti

Nilai *fitness* didapat dari total penalti yang bergantung pada selisih tiap komponen sesuai dengan persamaan 3.2. Dari nilai selisih inilah bisa ditentukan penalti mana yang akan mengenai individu tersebut. Penalti di sini akan dihitung masing-masing dari jumlah penyusun komponen dalam makan pagi, makan siang, makan malam dan pelengkap kemudian akan dijumlahkan. Dalam pemberian skor penalti ini, karbohidrat akan diberikan skor yang besar jika dilakukan pelanggaran karena karbohidrat merupakan unsur utama yang diperhatikan. Dalam kasus ini, digunakan nilai *absolute* untuk hasil perhitungan selisihnya karena baik kekurangan atau kelebihan nutrisi akan dianggap sama-sama buruk.

$$selisih\ x_i y_j = |kebutuhan\ seimbang\ x_i y_j - nilai x_i y_j| \quad (3.2)$$

Dengan :

x_i = nilai komponen penyusun (karbohidrat, lemak, protein)

i = 0, 1 ,2

y_j= posisi ketika komponen tersebut diberikan (pagi, siang, malam, pelengkap)

j = 0, 1 ,2, 3

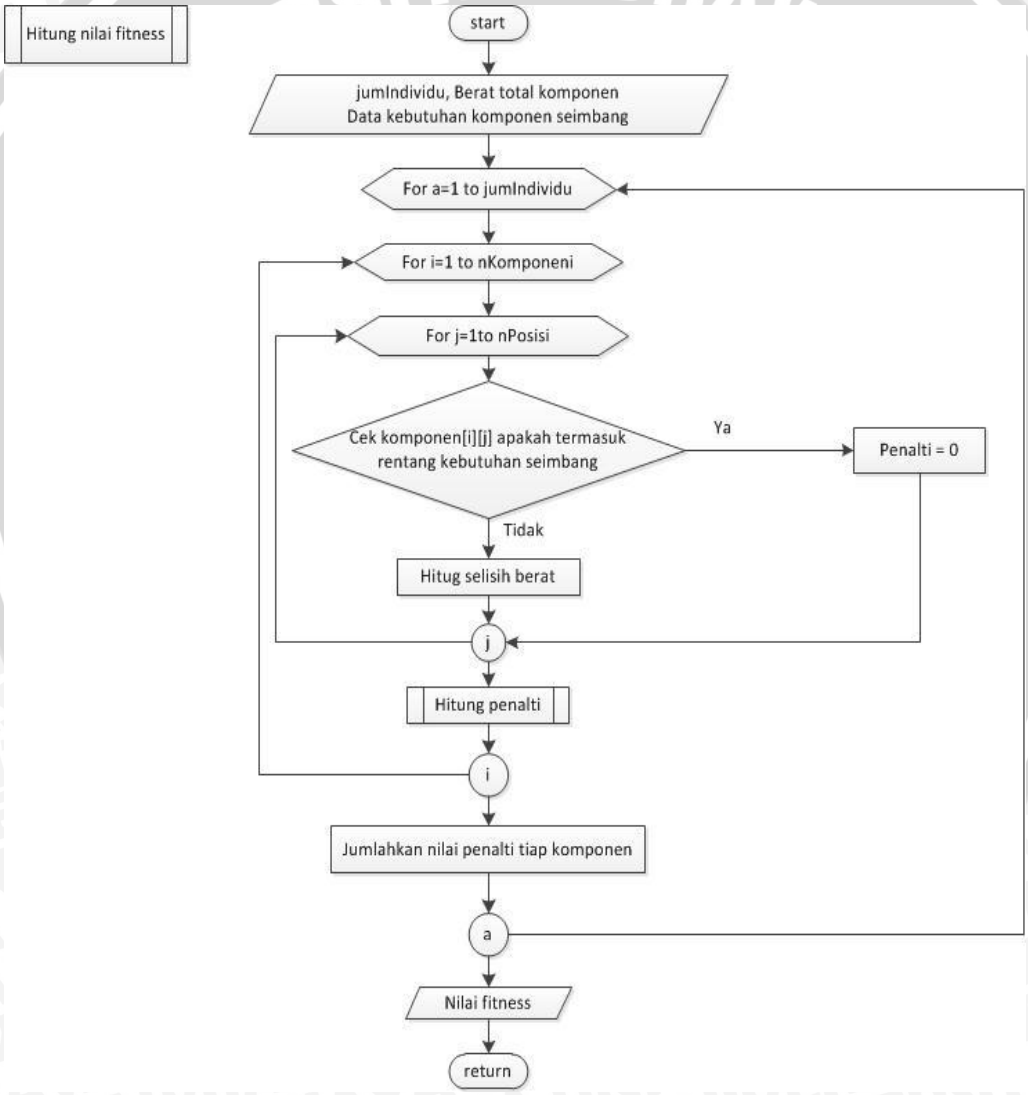
Tabel 3. 2 Skor untuk pelanggaran aturan

No	Aturan	Penalti
1	Jika kekurangan atau kelebihan karbohidrat dari jumlah yang diperbolehkan pada makanan pagi, siang, malam, pelengkap	$\sum_{j=0}^{3} selisih\ x_{karbohidrat} y_j * 5$
2	Jika kekurangan atau kelebihan protein dari jumlah yang diperbolehkan pada makanan pagi, siang, malam, pelengkap	$\sum_{j=0}^{3} selisih\ x_{protein} y_j$
3	Jika kekurangan atau kelebihan lemak dari jumlah yang diperbolehkan pada makanan pagi, siang, malam, pelengkap	$\sum_{j=0}^{3} selisih\ x_{lemak} y_j$

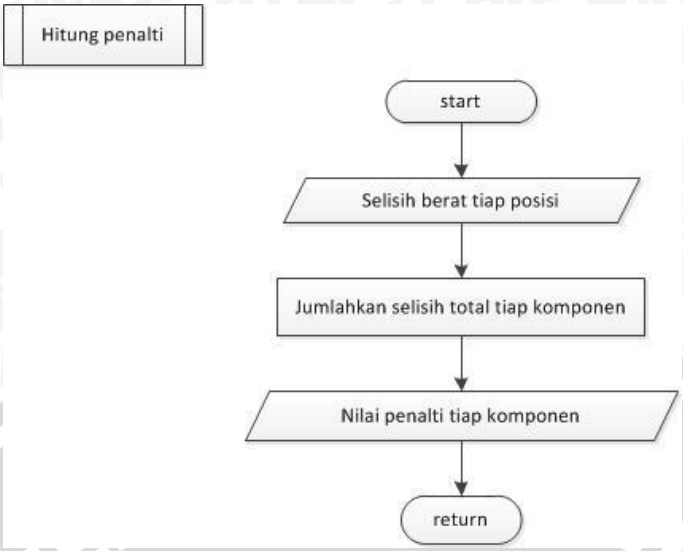
Berikut adalah langkah-langkah untuk menghitung nilai *fitness* pada penelitian ini

1. Masukkan semua individu termasuk *offspring* dan individu hasil mutasi
2. Hitung selisih antara berat seimbang dengan berat komponen hasil penghitungan sesuai persamaan 3.2
3. Bandingkan hasilnya dengan berat seimbang
4. Hitung skor penalti yang didapat
5. Cari nilai *fitness* tiap individu sesuai dengan persamaan 3.1

Flowchart untuk menghitung nilai *fitness* ini akan ditunjukkan dalam gambar 3.13



Gambar 3. 13 *Flowchart* untuk menghitung nilai *fitness*



Gambar 3. 14 *Flowchart* untuk menghitung penalti

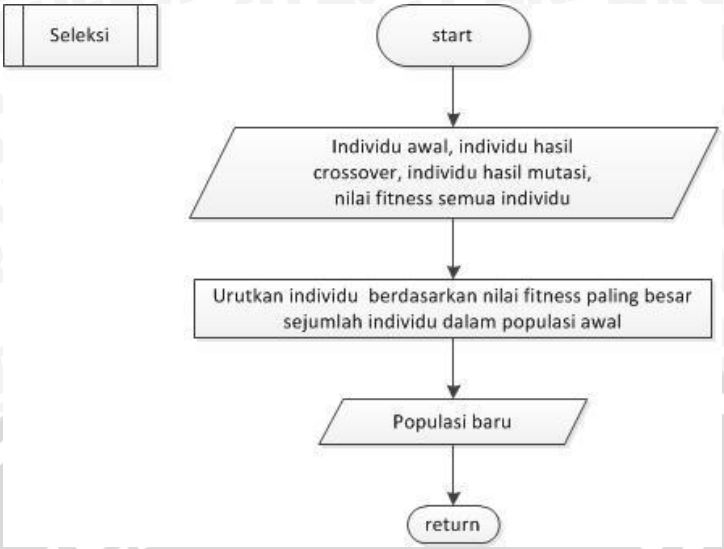
3.2.9 Seleksi

Proses seleksi digunakan untuk memilih individu-individu yang akan digunakan dalam iterasi atau generasi berikutnya. Proses *crossover* dan mutasi yang telah dilakukan sebelumnya menghasilkan keturunan (*offspring*) sebagai individu baru. Ukuran satu populasi tiap generasi selalu sama, oleh karena itu hanya individu yang terbaiklah yang mampu bertahan dalam populasi atau keturunan terbaiklah yang akan masuk ke dalam populasi untuk membentuk generasi berikutnya yang lebih baik. Dengan demikian harus dilakukan proses seleksi.

Metode seleksi yang digunakan adalah *rank based fitness selection.* Adapun langkah-langkah dalam metode seleksi ini adalah sebagai berikut :

1. Masukkan semua individu yaitu individu awal, *offspring*, dan individu hasil mutasi
2. Urutkan berdasarkan nilai *fitness* yang tertinggi
3. Ambil individu berdasarkan urutan *fitness* maksimum sejumlah individu dalam populasi awal untuk digunakan sebagai populasi baru
4. Gunakan populasi baru untuk penghitungan pada generasi berikutnya sampai ditemukan generasi yang optimum

Flowchart untuk proses seleksi akan akan ditunjukkan dalam gambar 3.15 berikut .



Gambar 3. 15 *Flowchart* dari *rank based fitness selection*

3.3 Perhitungan manual

Berikut adalah contoh perhitungan manual untuk 1 generasi untuk pasien dengan data contoh sebagai berikut :

Berat badan : 54 kg

Usia : 72 tahun

Jenis Kelamin : L

Kalori : 1714,2 kkal ≈ 1700 kkal

Untuk jumlah seimbang karbohidrat, lemak dan protein yang dibutuhkan pasien dengan 1700 kalori dapat dilihat dalam tabel 2.4.

3.3.1 Pembangkitan populasi awal

Individu 1 : 2313 1482 1421 281

Individu 2 : 1243 3221 4123 305

Individu 3 : 1924 7179 1529 187

Individu 4 : 4411 1413 1621 298

Individu 5 : 1392 5992 2823 650

3.3.2 Pembobotan Makanan

Pembobotan makanan akan dilakukan secara *random* dengan skala nol sampai dengan satu yang menunjukkan perbandingan dalam skala tertentu. Untuk jumlah seimbang karbohidrat, lemak dan protein yang dibutuhkan pasien dengan 1700 kalori dapat dilihat dalam tabel 2.4.

2	3	1	3	1	4	8	2	1	4	2	1	2	8	1
0,4	0,5	0,1	0,2	0,8	0,1	0,6	0,5	0,5	0,4	0,3	0,5	0,3	0,6	0,7

Gambar 3. 16 Representasi kromosom pada Individu 1

Untuk makan pagi =

S → 0,4 * 180 = 72 gr

N → 0,5 * 50 = 25 gr

H → 0,1 * 80 = 8 gr

Y → 0,2 * 200 = 40 gr

Untuk makan siang =

S → 0,8 * 180 = 144 gr

N → 0,1 * 50 = 5 gr

H → 0,6 * 80 = 48 gr

Y → 0,5 * 200 = 40 gr

Untuk makan malam =

S → 0,5 * 180 = 90 gr

N → 0,4 * 50 = 20 gr

H → 0,3 * 80 = 24 gr

Y → 0,5 * 200 = 40 gr

Untuk pelengkap =

P1→ 0,3 * 150 = 45 gr

P2→ 0,6 * 150 = 90 gr

P3→ 0,7 * 150 = 105 gr

Berat komponen penyusun Karbohidrat, Lemak, Protein untuk makan pagi :

Untuk 72 gr bihun mengandung

karbohidrat = $\frac{72}{100}$ x 82,1 = 59,11 gr

lemak = $\frac{72}{100}$ x 0,1 = 0,07 gr

protein = $\frac{72}{100}$ x 4,7 = 3,38 gr

Untuk 25 gr tempe murni mengandung

karbohidrat = $\frac{25}{100}$ x 13,5 = 3,38 gr

lemak = $\frac{25}{100}$ x 8,8 = 2,2 gr

protein = $\frac{25}{100}$ x 20,8 = 5,2 gr

Untuk 8 gr ayam mengandung

karbohidrat = $\frac{8}{100}$ x 0 = 0 gr

lemak = $\frac{8}{100}$ x 25 = 2 gr

protein = $\frac{8}{100}$ x 18,2 = 1,46 gr

Untuk 40 gr buncis mengandung

karbohidrat = $\frac{40}{100}$ x 7,2 = 2,88 gr

lemak = $\frac{40}{100}$ x 0,3 = 0,12 gr

protein = $\frac{40}{100}$ x 2,4 = 0,96 gr

Karbohidrat	Lemak	Protein

bihun			Tempe murni			ayam			Buncis		
59,1	0,07	3,38	3,38	2,2	5,2	0	2	1,46	2,88	0,12	0,96

Berat komponen penyusun Karbohidrat, Lemak, Protein untuk makan siang :

Untuk 144 gr nasi mengandung

karbohidrat = $\frac{144}{100}$ x 39,8 = 57,31 gr

lemak = $\frac{144}{100}$ x 0,3 = 0,432gr

protein = $\frac{144}{100}$ x 3 = 4,32 gr

Untuk 5 gr kacang merah mengandung

karbohidrat = $\frac{5}{100}$ x 28 = 1,4 gr

lemak = $\frac{5}{100}$ x 2,2 = 0,11 gr

protein = $\frac{5}{100}$ x 11 = 0,55 gr

Untuk 48 gr bandeng presto mengandung

karbohidrat = $\frac{48}{100}$ x 11,3 = 5,42 gr

lemak = $\frac{48}{100}$ x 20,3 = 9,74 gr

protein = $\frac{48}{100}$ x 17,1 = 8,21 gr

Untuk 40 gr sawi mengandung

karbohidrat = $\frac{40}{100}$ x 4 = 1,6 gr

lemak = $\frac{40}{100}$ x 0,3 = 0,12 gr

protein = $\frac{40}{100}$ x 2,3 = 0,92 gr

nasi			Kacang merah			Bandeng presto			sawi		
57,31	0,43	4,32	1,4	0,11	0,55	5,42	9,74	8,21	1,6	0,12	0,92

Berat komponen penyusun Karbohidrat, Lemak, Protein untuk makan malam :

Untuk 90 gr nasi mengandung

karbohidrat = $\frac{90}{100}$ x 39,8 = 35,82 gr

lemak = $\frac{90}{100}$ x 0,3 = 0,27 gr

protein = $\frac{90}{100}$ x 3 = 2,7 gr

Untuk 20 gr kacang merah mengandung

karbohidrat = $\frac{20}{100}$ x 28 = 5,6 gr

lemak = $\frac{20}{100}$ x 2,2 = 0,44 gr

protein = $\frac{20}{100}$ x 11 = 2,2 gr

Untuk 24 gr daging sapi mengandung

karbohidrat = $\frac{24}{100}$ x 0 = 0 gr

lemak = $\frac{24}{100}$ x 14 = 3,36 gr

protein = $\frac{24}{100}$ x 18,8 = 4,51 gr

Untuk 40 gr bayam mengandung

karbohidrat = $\frac{40}{100}$ x 2,9 = 1,16 gr

lemak = $\frac{40}{100}$ x 0,4 = 0,16 gr

protein = $\frac{40}{100}$ x 0,9 = 0,36 gr

nasi			Kacang merah			Daging sapi			Bayam		
35,82	0,27	2,7	5,6	0,44	2,2	0	3,36	4,51	1,16	0,16	0,36

Berat komponen penyusun Karbohidrat, Lemak, Protein untuk pelengkap :

Untuk 45 gr apel mengandung

karbohidrat = $\frac{45}{100}$ x 14,9 = 6,71gr

lemak = $\frac{45}{100}$ x 0,4 = 0,18 gr

protein = $\frac{45}{100}$ x 0,3 = 0,14 gr

Untuk 90 gr alpukat mengandung

karbohidrat = $\frac{90}{100}$ x 7,7 = 6,93 gr

lemak = $\frac{90}{100}$ x 6,5 = 5,85 gr

protein = $\frac{90}{100}$ x 0,9 = 0,81 gr

Untuk 105 gr manggis mengandung

karbohidrat = $\frac{105}{100}$ x 15,6 = 16,38 gr

lemak = $\frac{105}{100}$ x 0,6 = 0,63 gr

protein = $\frac{105}{100}$ x 0,6 = 0,63gr

apel			Alpukat			manggis		
6,71	0,18	0,14	6,93	5,85	0,81	16,38	0,63	0,63

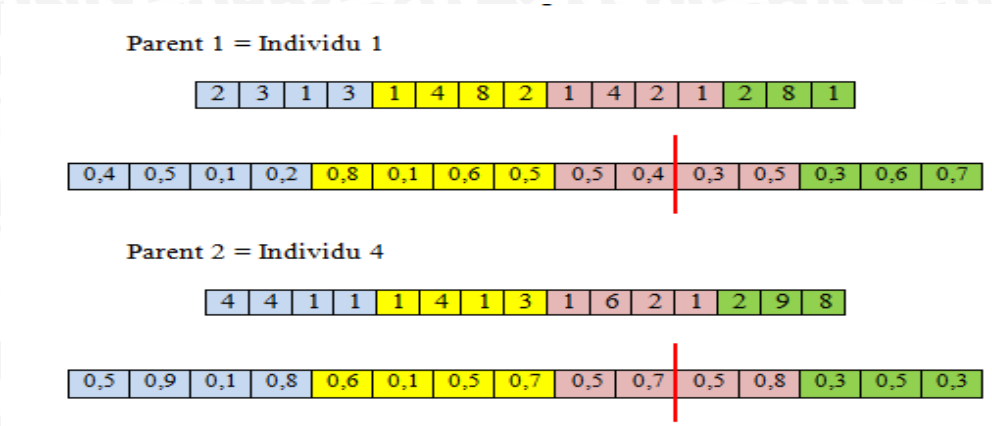
Tabel 3. 3 Hasil penghitungan bobot dalam Individu 1

Individu ke-	Nama makanan	Berat (gr)	Karbohidrat (gr)	Lemak (gr)	Protein (gr)
Individu 1					
Makan pagi	bihun	72	59,112	0,072	3,384
	tempe murni	25	3,375	2,2	5,2
	ayam	8	0	2	1,456
	buncis	40	2,88	0,12	0,96
	Total		65,367	4,392	11
Makan siang	nasi	144	57,312	0,432	4,32
	kacang merah	5	1,4	0,11	0,55
	bandeng presto	48	5,424	9,744	8,208
	sawi	40	1,6	0,12	0,92
	Total		65,736	10,406	13,998
Makan malam	nasi	90	35,82	0,27	2,7
	kacang merah	20	5,6	0,44	2,2
	daging sapi	24	0	3,36	4,512
	bayam	40	1,16	0,16	0,36
	Total		42,58	4,23	9,772
pelengkap	apel	45	6,705	0,18	0,135
	alpukat	90	6,93	5,85	0,81
	manggis	105	16,38	0,63	0,63
	Total		30,015	6,66	1,575

Dengan cara yang sama, maka untuk perhitungan individu yang lain akan ditunjukkan dalam Lampiran 2.

3.3.3 *Crossover*

Untuk *crossover*, individu-individu terpilih yang menjadi *parent* akan dikombinasikan 2. Dalam hitungan ini misalnya yang dikombinasikan adalah individu 1 dan individu 4.



Gambar 3. 17 Ilustrasi proses *crossover* dengan α = 0.5

Sesuai persamaan 2.2, maka dapat dihitung :

Child 1 gen ke 1 (setelah *cut point*)

C_1= (P_2**alpha*)+(P_1*(1-*alpha*)) = (0.5*0.5)+ (0.3*0.5)= 0.25+0.15=0.4

Child 1 gen ke 2 (setelah *cut point*)

C_1= (P_2**alpha*)+(P_1*(1-*alpha*)) =(0.8*0.5)+(0.5*0.5)= 0.4+0.25=0.65

Child 1 gen ke 3 (setelah *cut point*)

C_1= (P_2**alpha*)+(P_1*(1-*alpha*)) = (0.3*0.5)+(0.3*0.5)= 0.15+0.15=0.3

Child 1 gen ke 4 (setelah *cut point*)

C_1= (P_2**alpha*)+(P_1*(1-*alpha*)) = (0.5*0.5)+(0.6*0.5)= 0.25+0.3=0.55

Child 1 gen ke 5 (setelah *cut point*)

C_1= (P_2**alpha*)+(P_1*(1-*alpha*)) = (0.3*0.5)+(0.7*0.5)= 0.15+0.35=0.5

Sesuai persamaan 2.3, maka dapat dihitung :

Child 2 gen ke 1 (setelah *cut point*)

C_2= (P_1**alpha*)+(P_2*(1-*alpha*)) =(0.3*0.5)+(0.5*0.5)= 0.15+0.25=0.4

Child 2 gen ke 2 (setelah *cut point*)

C_2= (P_1**alpha*)+(P_2*(1-*alpha*)) =(0.5*0.5)+(0.8*0.5)= 0.25+0.4=0.65

Child 2 gen ke 3 (setelah *cut point*)

C_2= (P_1**alpha*)+(P_2*(1-*alpha*)) =(0.3*0.5)+(0.3*0.5)= 0.15+0.15=0.3

Child 2 gen ke 4 (setelah *cut point*)

C_2= (P_1**alpha*)+(P_2*(1-*alpha*)) =(0.6*0.5)+(0.5*0.5)= 0.3+0.25=0.55

Child 2 gen ke 5 (setelah *cut point*)

C_2= (P_1**alpha*)+(P_2*(1-*alpha*)) =(0.7*0.5)+(0.3*0.5)= 0.35+0.15=0.5

3.3.4 Mutasi

Dalam mutasi, akan dipilih individu-individu yang akan mengalami mutasi melalui nilai probabilitas mutasi (Pm). Berikut adalah nilai Pm dari setiap individu.

Nilai Pm input = 0,4

Tabel 3. 4 Nilai Pm dari tiap individu

No	Inidvidu	Pm
1	Individu 1	0,5
2	Individu 2	0,6
3	Individu 3	0,9
4	Individu 4	0,6
5	Individu 5	0,7
6	*Child* 1	0,2
7	*Child* 2	0,5

Dari tabel 3.4, individu yang mengalami mutasi adalah *child* 1, maka akan dibangkitkan nilai Pm dari setiap gen dalam *Child* 1.

Nilai Pm gen = 0,7

A = Jenis makanan

B = Pembobotan makanan

C = Nilai Pm

Child 1 :

A	2	3	1	3	1	4	8	2	1	4	2	1	2	8	1
B	0,4	0,5	0,1	0,2	0,8	0,1	0,6	0,5	0,5	0,4	0,4	0,7	0,3	0,6	0,5
C	0,4	0,8	0,1	0,6	0,5	0,8	0,3	0,6	0,5	0,3	0,1	0,7	0,3	0,8	0,8

Child 1'

A	1	3	4	2	5	4	4	1	5	2	6	1	5	8	1
B	0.5	0.5	0.7	0.3	0.5	0.1	0.6	0.4	0.8	0.5	0.2	0.7	0.4	0.6	0.5

Gambar 3. 18 Ilustrasi proses mutasi dengan *metode random mutation*

Dengan cara yang sama akan dilakukan penghitungan *fitness* seperti individu yang lain, sehingga didapatkan nilai *fitness* yang baru.

3.3.5 Nilai *Fitness*

Nilai *fitness* yang dicari dari total penalti bergantung dari selisih kebutuhan seimbang dengan hasil perhitungan sesuai dengan persamaan 3.2. Dari nilai selisih inilah bisa ditentukan penalti mana yang akan mengenai individu tersebut.

Tabel 3. 5 Perhitungan nilai *fitness* dari individu 1

individu 1	komponen	energi	berat	selisih	penalti	*fitness*
pagi	karbohidrat	261,468	65,367	10,367	51,835	
	lemak	39,528	4,392	1,608	1,608	
	protein	44	11	0	0	
	total	344,996				
siang	karbohidrat	262,944	65,736	14,264	71,32	
	lemak	93,654	10,406	0	0	
	protein	55,992	13,998	1,002	1,002	
	total	412,59				
malam	karbohidrat	170,32	42,58	24,42	122,1	
	lemak	38,07	4,23	3,77	3,77	
	protein	39,088	9,772	2,228	2,228	
	total	247,478				
pelengkap	karbohidrat	120,06	30,015	36,985	184,925	
	lemak	59,94	6,66	1,34	1,34	
	protein	6,3	1,575	10,425	10,425	
	total	186,3				
kalori dalam sehari		1191,364			450,553	0,002219

$$fitness = \frac{1}{1 + \sum_{1=1}^{6} P_i}$$

$$= \frac{1}{1 + (51{,}835 + 1{,}608 + 0 + 71{,}32 + 0 + 1{,}002 + 122{,}1 + 3{,}77 + 2{,}228 + 184{,}925 + 1{,}34 + 10{,}425)}$$

$$= \frac{1}{450{,}553}$$

$$= 0{,}002219$$

Setelah dihitung dengan persamaan 3.1 dan 3.2, maka didapatkan nilai *fitness* dari semua individu seperti pada Tabel 3.6 berikut

Tabel 3. 6 Nilai *fitness* hasil *crossover* dan mutasi

No	Individu	*Fitness*
1	Individu ke 1	0,002219
2	Individu ke 2	0,001315
3	Individu ke 3	0,001566
4	Individu ke 4	0,00158
5	Individu ke 5	0,001569
6	*Child* 1	0,002192
7	*Child* 2	0,001619
8	*Child* 1’	0,002256262

3.3.6 Seleksi

Untuk proses seleksi, maka semua individu akan diurutkan berdasarkan *fitness*-nya dan diambil individu sesuai dengan jumlah individu pada populasi awal.

Tabel 3. 7 Populasi dengan *fitness* dari terbesar ke terkecil

No	Individu	*Fitness*
1	*Child* 1’	0,002256262
2	Individu ke 1	0,002219
3	*Child* 1	0,002192
4	*Child* 2	0,001619
5	Individu ke 4	0,00158
6	Individu ke 5	0,001569
7	Individu 3	0,001566
8	Individu 2	0,001315

Berdasarkan jumlah populasi awal, maka hanya terdapat 5 individu baru yang akan digunakan dalam generasi berikutnya sehingga 5 individu dengan nilai *fitness* teratas akan dipilih sebagai individu dalam populasi berikutnya seperti dalam tabel 3.8.

Tabel 3. 8 Populasi Baru

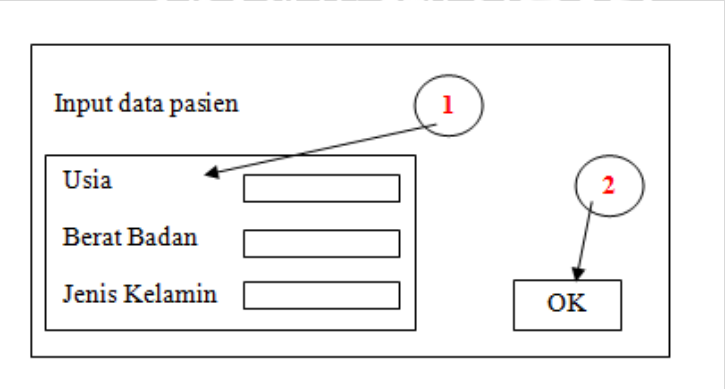
No	Individu	*Fitness*
1	*Child* 1’	0,002256262
2	Individu ke 1	0,002219
3	*Child* 2	0,001619
4	Individu ke 4	0,00158
5	Individu ke 5	0,001569

3.4 Rancangan antarmuka

Dalam perancangan uji coba, terdapat 2 bagian utama yang akan digunakan oleh pengguna, yaitu *input* data dan *output* data.

3.4.1 *Form* Proses *Input* Data

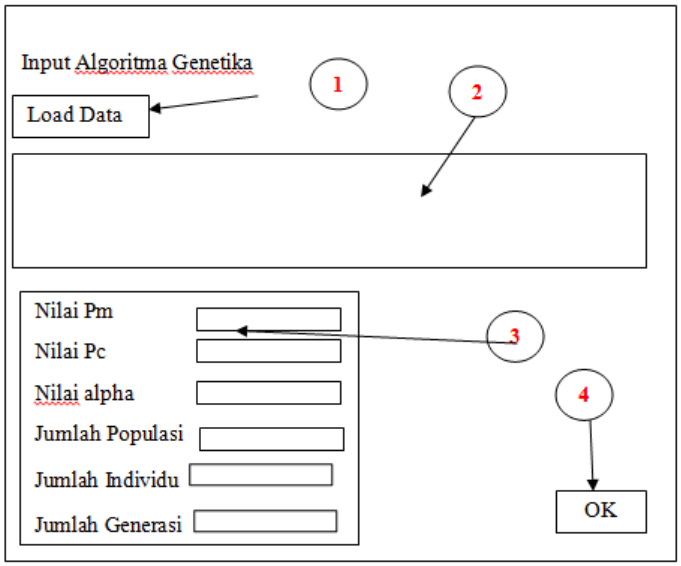
Dalam *form* ini, pengguna akan melakukan proses *input* data seperti ditunjukkan dalam Gambar 3.19



Gambar 3. 19 Rancangan *User Interface* untuk *input* data pasien

Penjelasan :

1. Bagian no.1 merupakan *field* untuk mengisikan data-data yang berkaitan dengan pasien
2. Bagian no.2 merupakan tombol yang akan melakukan proses penghitungan untuk kalori berdasarkan umur dan berat badan pasien, sekaligus akan mengarahkan ke bagian proses algoritma genetika.



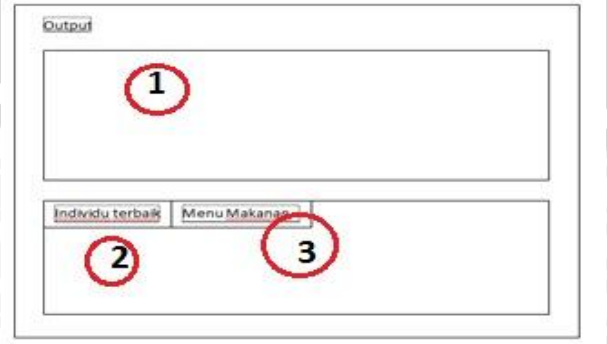
Gambar 3. 20 Rancangan *User Interface* untuk *input* proses genetika

Penjelasan :

1. Bagian no.1 merupakan tombol untuk menampilkan data bahan makanan
2. Bagian no. 2 akan berisi *output* data bahan makanan
3. Bagian no. 3 merupakan *field* untuk mengisikan parameter genetika yang akan digunakan pada proses pencarian kombinasi menu makanan dengan menggunakan algoritma genetika.
4. Bagian no. 4 merupakan tombol untuk mulai melakukan proses algoritma genetika

3.4.2 *Form* Proses *Output* Data

Dalam *form* ini, sistem akan menampilkan *output* data yang akan ditunjukkan dalam Gambar 3.21



Gambar 3. 21 *Rancangan User Interface* untuk *output* sistem

Penjelasan :

1. Bagian no.1 akan menampilkan semua proses dan penghitungan dengan algoritma genetika dari semua individu dalam setiap generasi
2. Bagian no.2 akan menampilkan individu terbaik yang merupakan solusi
3. Bagian no.3 akan menampilkan menu makanan sesuai dengan individu terbaik

3.5 Skenario uji coba

Uji coba pada penelitian kombinasi menu makanan untuk penderita *Diabetes Mellitus* menggunakan Algoritma genetika ini dilakukan dengan cara menghitung setiap bobot makanan yang menjadi suatu kesatuan solusi untuk kemudian diproses atau dihitung dengan menggunakan operator genetika. Tujuan dari uji coba ini adalah untuk mendapatkan kombinasi jenis dan bobot makanan yang sesuai dengan kebutuhan pasien.

Parameter yang akan diujikan adalah nilai Probabilitas *Crossover* (Pc) dan nilai Probabilitas Mutasi (Pm) , jumlah generasi dan ukuran populasi. Setiap pengujian akan dilakukan untuk mengetahui pengaruh perubahan nilai Pc dan Pm, jumlah generasi dan ukuran populasi terhadap nilai *fitness*. Tabel rancangan pengujian Pc dan Pm dapat dilihat pada Tabel 3.9, Tabel rancangan pengujian jumlah generasi dapat dilihat pada Tabel 3.10, dan Tabel rancangan pengujian ikuran populasi dapat dilihat pada tabel 3.11 .

Tabel 3. 9 Rancangan Pengujian Pengaruh Pc dan Pm

Nilai *alpha* = … , jumlah individu= …, jumlah populasi = …						
	Pm					
Pc	0.1	0.2	…	….	…	0.9
0.1						
0.2						
….						
….						
0.9						

Tabel 3. 10 Rancangan Pengujian Pengaruh Jumlah Generasi

No	Jumlah generasi	Nilai pc= … , Nilai pm=… ,*alpha*= …, jumlah populasi= …					
		Percobaan ke-					*Fitness* rata-rata
		1	2	3	...	10	
1	10						
2	30						
3	50						
4	80						
5	100						
6	150						

Tabel 3. 11 Rancangan Pengujian Pengaruh Jumlah Populasi

No	Jumlah populasi	Nilai pc= … , Nilai pm=… , alpha= …, jumlah generasi= …					
		Percobaan ke-					*Fitness* rata-rata
		1	2	3	...	10	
1	10						
2	20						
3	30						
4	40						
5	50						

BAB IV
ANALISA DAN PEMBAHASAN

Untuk melakukan implementasi sistem, maka perlu dipersiapkan lingkungan implementasi untuk memenuhi kebutuhan program dalam mengimplementasikan sistem.

4.1 Lingkungan Implementasi

Proses implementasi merupakan tahapan penerapan rancangan pada bahasa pemrograman yang dapat dimengerti oleh komputer. Lingkungan implementasi yang akan dijelaskan pada bab ini meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan sistem ini adalah sebagai berikut :

1. *Processor Intel(R) Core(TM) i3 2,53 GHz*
2. *RAM 1,00 GB*
3. *Harddisk 500 GB*
4. *Monitor 14'*
5. *Keyboard*
6. *Mouse*

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang diperlukan dalam pengembangan dan pengujian sistem ini adalah sebagai berikut :

1. *Sistem Operasi Windows 7 Ultimate 32 bit*
2. *Netbeans 6.9.1*
3. *JDK Version 7 Update 3*

4.2 Implementasi Program

Berdasarkan perancangan perangkat lunak yang telah diuraikan pada bab 3, maka akan dibahas mengenai implementasi program sesuai dengan rancangan. Sistem diimplementasikan pada kelas utama yang menampung seluruh proses. Selanjutnya tiap-tiap proses dilakukan oleh kelas lain sesuai dengan fungsi dan pembagiannya. Kelas-kelas yang dibangun akan dijelaskan dalam tabel 4.1

Tabel 4. 1 Kelas-kelas yang dibangun

No	Nama	Keterangan
1	`Amb.java`	Kelas untuk menyimpan fungsi regresi linier AMB (Angka Metabolisme Basal)
2	`Diet.java`	Kelas untuk mengatur jenis diet dan kebutuhan diet
3	`Gizi.java`	Kelas yang mengolah nilai gizi dalam setiap data makanan
4	`Individu.java`	Kelas yang mendefinisikan individu dari populasi
5	`KebutuhanDiet.java`	Kelas yang menyimpan kebutuhan makanan yang diperlukan oleh masing masing tipe diet
6	`JAlgen.java`	Kelas yang menyimpan fungsi fungsi utama dalam proses algoritma genetika, seperti *crossover* / mutasi

Untuk penjelasan *method* yang terdapat pada masing-masing kelas akan diuraikan dalam tabel 4.2, tabel 4.3 , tabel 4.4, tabel 4.5, tabel 4.6 , dan tabel 4.7 berikut.

Tabel 4. 2 *Method -method* dalam *class* `Amb.java`

No	Nama	Keterangan
1	`usiaIsInRange(double usia)`	*Method* untuk melakukan penegecekan apakah usia berada dalam range AMB
2	`getKalori(double beratBadan, Constant jenisKelamin)`	*Method* untuk mendapatkan kalori dengan rumus AMB (Pengali * BeratBadan + Penambah)
3	`getUmurMin()`	*Method* untuk mendapatkan umur min
4	`getUmurMax()`	*Method* untuk mendapatkan umur max
5	`getLMultiplier()`	*Method* untuk mendapatkan umur bilangan pengali berat badan Laki Laki
6	`getPMultiplier()`	*Method* untuk mendapatkan umur bilangan pengali berat badan Perempuan
7	`getLIncrement()`	*Method* untuk mendapatkan bilangan penambah kalori laki-laki
8	`getPIncrement()`	*Method* untuk mendapatkan bilangan penambah kalori Perempuan

Tabel 4. 3 *Method -method* dalam *class* KebutuhanDiet.java

No	Nama	Keterangan
1	`getMinProtein()`	*Method* dengan tipe data *double* untuk mengambil data nilai *minimum* Protein pada diet
2	`setMinProtein(double minProtein)`	*Method* untuk memasukkan nilai *minimum* Protein yang dimiliki makanan
3	`getMaxProtein()`	*Method* dengan tipe data *double* untuk mengambil data nilai maksimum Protein pada diet
4	`setMaxProtein(double maxProtein)`	*Method* untuk memasukkan nilai maksimum Protein yang dimiliki makanan
5	`getMinLemak()`	*Method* dengan tipe data *double* untuk mengambil data nilai *minimum* Lemak pada diet
6	`setMinLemak(double minLemak)`	*Method* untuk memasukkan nilai *minimum* Lemak yang dimiliki makanan
7	`getMaxLemak()`	*Method* dengan tipe data *double* untuk mengambil data nilai maksimum Lemak pada diet
8	`setMaxLemak(double maxLemak)`	*Method* untuk memasukkan nilai maksimum Lemak yang dimiliki makanan
9	`getMinKarbohidrat()`	*Method* dengan tipe data *double* untuk mengambil data nilai *minimum* Karbohidrat pada diet
10	`setMinKarbohidrat(double minKarbohidrat)`	*Method* untuk memasukkan nilai *minimum* Karbohidrat yang dimiliki

		makanan
11	`getMaxKarbohidrat()`	*Method* dengan tipe data *double* untuk mengambil data nilai maksimum Karbohidrat pada diet
12	`setMaxKarbohidrat(double maxKarbohidrat)`	*Method* untuk memasukkan nilai maksimum Lemak yang dimiliki makanan

Tabel 4. 4 *Method -method* dalam *class* diet.java

No	Nama	Keterangan
1	`getName()`	*Method* dengan tipe data String untuk mendapatkan Nama / tipe Diet
2	`setName(java.lang.String.name)`	*Method* untuk memasukkan nama/tipe Diet
3	`getEnergi()`	*Method* untuk mengambil nilai energi yang dibutuhkan pada tipe diet
4	`setEnergi(double energi)`	*Method* untuk memasukkan nilai energi
5	`setKebutuhanPagi(KebutuhanDiet kebutuhanPagi)`	*Method* dengan variabel KebutuhanDiet, untuk memasukkan nilai Kebutuhan yang dibutuhkan oleh tipe Diet pada waktu Pagi
6	`setKebutuhanSiang(KebutuhanDiet kebutuhanSiang)`	*Method* dengan variabel KebutuhanDiet, untuk memasukkan nilai Kebutuhan yang dibutuhkan oleh tipe Diet pada waktu Siang
7	`setKebutuhanMalam(KebutuhanDiet kebutuhanMalam)`	*Method* dengan variabel KebutuhanDiet, untuk memasukkan nilai Kebutuhan yang dibutuhkan oleh tipe Diet pada waktu Malam
8	`setKebutuhanPelengkap(Kebu`	*Method* dengan variabel

	`tuhanDiet kebutuhanPelengkap)`	KebutuhanDiet, untuk memasukkan nilai Kebutuhan yang dibutuhkan oleh tipe Diet untuk Pelengkap
9	`getKebutuhanMalam()`	*Method* untuk mengambil nilai kebutuhan malam pada diet
10	`getKebutuhanPagi()`	*Method* untuk mengambil nilai kebutuhan pagi pada diet
11	`getKebutuhanSiang()`	*Method* untuk mengambil nilai kebutuhan siang pada diet
12	`getKebutuhanPelengkap()`	*Method* untuk mengambil nilai kebutuhan pelengkap pada diet

Tabel 4. 5 *Method -method* dalam *class* `gizi.java`

No	Nama	Keterangan
1	`getNamaMakanan()`	*Method* dengan tipe data String untuk mengambil data nama makanan
2	`setNamaMakanan(java.lang.String namaMakanan)`	*Method* dengan tipe data String untuk memasukkan nama makanan
3	`getEnergi()`	*Method* untuk mendapatkan nilai energi yang dimiliki makanan
4	`setEnergi(double energi)`	*Method* untuk memasukkan nilai energi yang dimiliki makanan dengan tipe *double*
5	`getProtein()`	*Method* untuk mendapatkan nilai Protein yang dimiliki makanan
6	`setProtein(double protein)`	*Method* untuk memasukkan nilai Protein yang dimiliki makanan
7	`getLemak()`	*Method* untuk mendapatkan nilai Lemak yang dimiliki makanan
8	`setLemak(double lemak)`	*Method* untuk memasukkan nilai Lemak yang dimiliki makanan
9	`getKarbohidrat()`	*Method* untuk mendapatkan nilai Karbohidrat yang dimiliki makanan
10	`setKarbohidrat(double karbohidrat)`	*Method* untuk memasukkan nilai Karbohidrat yang dimiliki makanan

Tabel 4. 6 *Method -method* dalam *class* `individu.java`

No	Nama	Keterangan
1	`setJenisMakanan(java.util.ArrayList <java.lang.Integer> jenisMakanan)`	*Method* untuk memasukkan jenisMakanan S N H Y | S N H Y | S N H Y|P1P2P3 ke dalam *ArrayList* jenisMakanan
2	`setGrMakanan(java.util.ArrayList <java.lang.Double> grMakanan)`	*Method* untuk memasukkkan bobot makanan dalam *ArrayList* grMakanan
3	`setBobotMakanan(java.util.ArrayList <java.lang.Double> bobotMakanan)`	*Method* untuk memasukkkan bobot makanan sebenarnya dari hasil perhitungan dalam *ArrayList* bobotMakanan
4	`getBobotMakanan()`	*Method* untuk mengambil nilai bobot makanan sebenarnya
5	`getCrossoverProb()`	*Method* untuk mengambil nilai Probabilitas *Crossover*
6	`getFitness`	*Method* untuk mengambil nilai *fitness*
7	`getGrMakanan`	*Method* untuk mengambil nilai gram makanan
8	`getJenisMakanan`	*Method* untuk mengambil jenis makanan
9	`getMutationProb`	*Method* untuk mengambil nilai Probabilitas Mutasi
10	`setCrossoverProb(double crossoverProb)`	*Method* untuk memasukkan nilai Probabilitas *crossover*
11	`setFitness(double fitness)`	*Method* untuk memasukkan nilai *fitness*
12	`setMutationProb(double mutationProb)`	*Method* untuk memasukkan nilai Probabilitas mutasi

Tabel 4. 7*Method -method* dalam *class* `JAlgen.java`

No	Nama	Keterangan
1	`countFitness(java.util.ArrayList<Individu> oldPopulasi, java.util.ArrayList<Gizi> pokok, java.util.ArrayList<Gizi> nabati, java.util.ArrayList<Gizi> hewani, java.util.ArrayList<Gizi> sayuran, java.util.ArrayList<Gizi> pelengkap, Diet diet)`	*Method* untuk menghitung nilai *fitness* dari tiap populasi
2	`countGramMakananPopulasi(java.util.ArrayList<Individu> oldPopulasi, java.util.HashMap<java.lang.String,java.lang.Double> maxGramMakanan, boolean doRandomBobot)`	*Method* untuk menghitung Gram makanan sebenarnya dari tiap populasi
3	`crossover(java.util.ArrayList<Individu> oldPopulasi, double crossoverProb, double alpha)`	*Method* untuk melakukan proses *crossover* dengan proses seleksi *parent* dan metode simple arithmetic
4	`generateRandomDoubleint jumlah, int start, int end)`	*Method* untuk mendapatkan bilangan *random* antara 0 sampai dengan 1
5	`getAmbFromSheet(jxl.Sheet ambSheet)`	*Method* untuk mendapatkan data dari *sheet* AMB
6	`getGiziMakanan(java.lang.String dir, java.lang.String filename)`	*Method* untuk mendapatkan nilai gizi makanan yang sudah dibaca dari sheet
7	`getGiziMakananFromSheet(jxl.Sheet sheet)`	*Method* untuk melakukan pembacaan dan pengambilan data giziMakanan dari *sheet*
8	`getTipeDiet(java.lang.String baseDir)`	*Method* untuk mendapatkan tipe diet yang sudah dibaca dari *sheet*

9	`getTipeDietFromSheet(jxl.Workbook workbook, jxl.Sheet sheet)`	*Method* untuk melakukan pembacaan dan pengambilan tipe diet dari *sheet*
10	`initPopulasi(java.util.ArrayList<Gizi> pokok, java.util.ArrayList<Gizi> nabati, java.util.ArrayList<Gizi> hewani, java.util.ArrayList<Gizi> sayuran, java.util.ArrayList<Gizi> pelengkap, int jumlah)`	*Method* untuk inisialisasi populasi awal yang disimpan dalam bentuk *ArrayList*
11	`mutasi(java.util.ArrayList<Individu> oldPopulasi, double mutationProb, java.util.HashMap<java.lang.String,java.lang.Double> maxGramMakanan)`	*Method* untuk melakukan proses mutasi
12	`seleksi(java.util.ArrayList<Individu> oldPopulasi, int jumlah)`	*Method* untuk melakukan proses seleksi

4.2.1 Inisialisasi Populasi Awal

Pada proses inisialisasi populasi awal, terdapat 2 proses yaitu inisialisasi individu atau representasi kromosom dan penghitungan bobot makanan.

4.2.1.1 Inisialisasi individu

Tahapan pertama dalam proses ini adalah pembangkitan individu sesuai dengan *flowchart* pada gambar 3.5. Individu diinisialisasikan dengan tipe data *ArrayList* (baris 4). Pada proses ini, akan dilakukan inisialisasi variabel (baris 7-9). Inisialisasi secara *random* akan diterapkan pada tiap menu makan pagi (baris 14-17), siang, malam dan pelengkap dengan cara yang sama. Setelah individu dibangkitkan, maka akan disimpan di populasi (baris 20). Proses inisialisasi individu ini dapat dilihat pada *Source code* 4.1 berikut

```
1  public static ArrayList<Individu> initPopulasi(
2  ArrayList<Gizi> pokok, ArrayList<Gizi> nabati,
   ArrayList<Gizi> hewani, ArrayList<Gizi> sayuran,
   ArrayList<Gizi> pelengkap, int jumlah)
3  {
4  ArrayList<Individu> populasi = new ArrayList<Individu>();
5
```

```
6   //random generator + seed
7   Random random = new Random();
8
9   for (int i = 0; i < jumlah; i++) {
10
11  ArrayList<Integer> jenisMakanan = new ArrayList<Integer>();
12
13  //pagi
14  jenisMakanan.add(random.nextInt(pokok.size()));
15  jenisMakanan.add(random.nextInt(nabati.size()));
16  jenisMakanan.add(random.nextInt(hewani.size()));
17  jenisMakanan.add(random.nextInt(sayuran.size()));
18
19  //add to populasi
20  populasi.add(new Individu(jenisMakanan));
21  }
22  return populasi;
23  }
```

***Source code* 4. 1** Proses inisialisasi individu

4.2.2 Pembobotan makanan

Langkah berikutnya setelah dilakukan inisialisasi jenis makanan pada tiap individu maka akan dilakukan pembobotan makanan dalam ukuran sebenarnya untuk tiap jenis makanan pada pagi, siang, malam dan pelengkap (baris 3-6) . Sebelumnya akan dilakukan proses *random* bobot untuk populasi pada generasi pertama. Disini dilakukan *generate random* dengan menggunakan *method generateRandomDouble* (baris 12) seperti pada *Source code 4.3*. Pertama-tama dilakukan pengecekan jika *Random* Bobot bernilai *true*, maka akan dilakukan penghitungan bobot sebenarnya, jika *false* maka akan dilakukan proses *Random* Bobot (baris 17-20). Jika sudah melakukan *random* bobot (pada generasi berikutnya) maka bobot akan diset pada individu tersebut, atau tidak dilakukan proses *random* bobot (baris 23-24). Proses dapat dilihat dalam *Source code* 4.2 berikut

```
1   public static ArrayList<Individu> countGramMakananPopulasi(ArrayList<Individu> oldPopulasi, HashMap<String, Double> maxGramMakanan, boolean doRandomBobot) {
2
3   ArrayList<Individu> populasi = new ArrayList<Individu>();
4   populasi.addAll(oldPopulasi);
5
```

```
6   for (Individu individu : populasi) {
7
8   ArrayList<Double> randomBobot = new ArrayList<Double>();
9   if (doRandomBobot)
10  randomBobot =
11
12  generateRandomDouble(individu.getJenisMakanan().size(),    0,
    1);
13
14  ArrayList<Double> gramMakanan = new ArrayList<Double>();
15
16  //pagi
17  gramMakanan.add((doRandomBobot    ?    randomBobot.get(0)    :
    individu.getBobotMakanan().get(0))                          *
    maxGramMakanan.get("pokok"));
18  gramMakanan.add((doRandomBobot    ?    randomBobot.get(1)    :
    individu.getBobotMakanan().get(1))
    *maxGramMakanan.get("nabati"));
19  gramMakanan.add((doRandomBobot    ?    randomBobot.get(2)    :
    individu.getBobotMakanan().get(2))
    *maxGramMakanan.get("hewani"));
20  gramMakanan.add((doRandomBobot    ?    randomBobot.get(3)    :
    individu.getBobotMakanan().get(3))                          *
    maxGramMakanan.get("sayur"));
21  individu.setGrMakanan(gramMakanan);
22
23  if (doRandomBobot)
24  individu.setBobotMakanan(randomBobot);
25  }
26  }
27
28  return populasi;
29      }
```

***Source code* 4. 2** Proses pembobotan makanan

Sesuai dengan *flowchart* pada gambar 3.7 maka akan dilakukan proses mendapatkan bobot makanan dalam ukuran sementara dengan cara *random* dalam *range* 0-1. Sama seperti proses inisialisasi jenis makanan, maka proses pembobotan akan dilakukan untuk tiap gen pagi, siang, malam dan pelengkap. Prosedur pembobotan makanan sementara secara *random* dilakukan pada *method generateRandomDouble* yang dapat dilihat dalam *Source code* 4.2 berikut

```
1   public static ArrayList<Double> generateRandomDouble(int
    jumlah, int start, int end) {
2         ArrayList<Double>      randomDouble      =      new
    ArrayList<Double>();
3
4         Random random = new Random();
5
6    for (int i = 0; i < jumlah; i++) {
7    double  randomD = (random.nextDouble() * (end - start)) +
    start;
```

```
8   randomDouble.add(Double.parseDouble(String.format("%.1f",
    randomD)));
9           }
10
11          return randomDouble;
12      }
```

***Source code* 4. 3** Proses pembobotan berat sementara makanan

4.2.3 Proses *Crossover*

Dalam proses *crossover* terdapat proses pemilihan induk untuk proses *crossover* ini akan diambil dengan membandingkan nilai Pc yang di*input*kan dengan nilai Pc yang di*random* untuk tiap individu. Individu dengan nilai Pc < Pc *input* akan dipilih sebagai *parent*. Setelah didapatkan *parent*, maka akan dilakukan proses *crossover* pada tiap individu terpilih dengan metode *simple arithmetic crossover*.

4.2.3.1 Proses pemilihan induk

Sesuai dengan *flowchart* pada gambar 3.10, dalam proses pemilihan induk atau *parent*, diinisialisasikan variabel populasi (baris 1), inisialisasi variabel *random, parent, child, indexParent* (7-10). Lalu akan dilakukan proses pembangkitan nilai Pc secara acak (baris 5) dan dibandingkan dengan nilai *input* Pc, jika nilainya kurang dari Pc *input*, maka akan disimpan sebagai *parent* dalam *ArrayList parent* (baris 14-18). Proses pemilihan *parent* ini dapat dilihat pada *Source code* 4.4

```
1   ArrayList<Individu> populasi = new
    ArrayList<Individu>();
2
3
4   //random crossover prob untuk tiap individu jika
    perlu random
5   ArrayList<Double> randomCrossoverProb =
    JAlgen.generateRandomDouble(populasi.size(), 0, 1);
6
7   Random random = new Random();
8   ArrayList<Individu>parent = new
    ArrayList<Individu>();
9   ArrayList<Individu>child = new ArrayList<Individu>();
10  ArrayList<Integer> indexParent = new
    ArrayList<Integer>();
11
12  int i = 0;
```

```
13  for (double crossoverProbIndividu : randomCrossoverProb) {
14
15      if (crossoverProbIndividu <crossoverProb) {
16
17  parent.add(populasi.get(i));
18      }
19      i++;
20  }
```

***Source code* 4. 4** Proses pemilihan induk

4.2.3.2 *Simple arithmetic crossover*

Setelah didapatkan *parent* dari proses sebelumnya, maka akan dilakukan proses *crossover*. Sesuai dengan *flowchart* pada gambar 3.11, dari semua *parent* akan dikombinasikan 2 *crossover* untuk setiap pasangan *parent* dan akan dicari *cut point* secara *random* sebagai titik persilangan atau titik tukar gen (baris 5). Untuk hasil anakan, pada masing-masing *parent* akan diambil nilai dari setiap gen setelah titik potong tersebut (baris 9-22). Berikutnya akan diinisialisasikan tempat penyimpanan berupa *ArrayList* bobot pada masing-masing *child* (baris 25-28). Setelah dilakukan proses penghitungan bobot baru pada *child* (baris 32-34) maka akan dilakukan penyimpanan bobot baru pada masing-masing *child* (baris 34-37). *Source code* dari proses tersebut dapat dilihat pada *Source code* 4.5

```
1   Individu child1 = parent.get(x);
2   Individu child2 = parent.get(y);
3
4   //random cut point 0 sampai panjang kromosom
5   int cutPoint = random.nextInt(parent.get(0).getBobotMakanan().size());
6
7   getLogger().info("Cut Point terpilih : " + cutPoint);
8
9   Log.getLogger().info("Parent 1 : ");
10  String print = "";
11  for (double bobotMakanan : parent.get(x).getBobotMakanan()) {
12      print += ((String.format("%1$,.2f", bobotMakanan)) + " | ");
13  }
14      Log.getLogger().info(print);
15
16      Log.getLogger().info("Parent 2 : ");
17      print = "";
18  for (double bobotMakanan : parent.get(y).getBobotMakanan())
```

```
    {
19      print += ((String.format("%1$,.2f", bobotMakanan)) + " | ");
20                  }
21      Log.getLogger().info(print);
22      Log.getLogger().info("");
23
24
25      ArrayList<Double>child1BobotMakanan = new ArrayList<Double>()
26      child1BobotMakanan.addAll(child1.getBobotMakanan());
27      ArrayList<Double>child2BobotMakanan = new ArrayList<Double>();
28      child2BobotMakanan.addAll(child2.getBobotMakanan());
29
30      //child 1
31      //substitute from index cut point to end
32      for (i = cutPoint; i <child1BobotMakanan.size(); i++) {
33          double newBobotMakanan = (alpha * parent.get(y).getBobotMakanan().get(i)) + ((1 - alpha) * parent.get(x).getBobotMakanan().get(i));
34          child1BobotMakanan.set(i, newBobotMakanan);
35      }
36      child.add(child1);
37      child.add(child2);
38    }
39  }
40
41
42
43  if (child.size() > 0) {
44      populasi.addAll(child);
45  }
```

***Source code* 4. 5** Proses *simple arithmetic crossover*

4.2.4 Mutasi

Dalam proses mutasi, juga terdiri dari 2 prosedur. Prosedur yang pertama adalah memilih individu mana yang akan mengalami mutasi. Prosedur yang kedua adalah proses mutasi itu sendiri .

4.2.4.1 Prosedur pilih individu yang mengalami mutasi

Sesuai dengan *flowchart* pada gambar 3.12, dalam pemilihan individu yang mengalami mutasi ini akan dilakukan secara *random* berdasarkan nilai probabilitas mutasi, jadi akan dibangkitkan nilai Pm tiap individu secara acak (baris 4) dan nilai Pm masing-masing individu dibandingkan dengan nilai Pm yang sudah ditentukan (baris 8-9), jika nilai Pm *random* kurang dari nilai Pm

yang ditentukan, maka individu akan mengalami mutasi (baris 9-12). Proses tersebut dapat dilihat pada *Source code* 4.6

```
1  ArrayList<Individu> populasi = new ArrayList<Individu>();
2  populasi.addAll(oldPopulasi);
3
4  ArrayList<Double> randomMutationProb =
   JAlgen.generateRandomDouble(populasi.size(), 0, 1);
5
6  ArrayList<Integer> indexToMutate = new ArrayList<Integer>();
7
8  int i = 0;
9  for (double individuMutationProb : randomMutationProb) {
10
11 if (individuMutationProb < mutationProb) {
12 indexToMutate.add(i);
13 }
14 i++;
15      }
```

***Source code* 4. 6** Proses pemilihan individu yang akan mengalami mutasi

4.2.4.2 Proses mutasi

Proses berikutnya setelah didapatkan individu mana yang akan mengalami mutasi, maka akan dilakukan mutasi terhadap bobot makanan pada gen terpilih (baris 6-9). Jadi dalam proses ini juga akan dilakukan pembangkitan nilai Pm *random* dan dibandingkan (baris 16), jika gen tersebut tidak memenuhi syarat, maka akan dilakukan mutasi terhadap bobot makanannya (baris 17-19). Setelah didapatkan gen mana yang terpilih untuk dimutasi maka akan dilakukan perubahan bobot (baris 22), juga dilakukan pengecekan apakah nilai Pm acak tersebut sama dengan nilai Pm sebelum diacak (baris 25-27). *MutatedBobotMakananan* ke i diganti dengan hasil *random* di *newRandom BobotMakanan* (baris 28). Jika syarat terpenuhi, maka *mutatedBobotMakanan* ditambahkan di *mutated* (baris 32). Proses mutasi tersebut dapat dilihat pada *Source code* 4.7

```
1  double genMutationProb = random.nextDouble();
2
3  ArrayList<Double> randomGenMutationProb =
   JAlgen.generateRandomDouble(populasi.get(indexMutate).getBob
   otMakanan().size(), 0, 1);
4
5  //inisialisasi individu termutasi
```

```
6   Individu mutated = new Individu(populasi.get(indexMutate).getJenisMakanan());
7   ArrayList<Double> mutatedBobotMakanan = new ArrayList<Double>();
8   mutatedBobotMakanan.addAll(populasi.get(indexMutate).getBobotMakanan());
9   mutated.setBobotMakanan(mutatedBobotMakanan);
10  ArrayList<Double> mutatedGrMakanan = new ArrayList<Double>();
11  mutatedGrMakanan.addAll(populasi.get(indexMutate).getGrMakanan());
12  mutated.setGrMakanan(mutatedGrMakanan);
13  mutated.setFitness(populasi.get(indexMutate).getFitness());
14
15  i = 0;
16  for (double individuGenMutationProb : RandomGenMutationProb) {
17  if (individuGenMutationProb < genMutationProb) {
18
19  Log.getLogger().info("Melakukan perubahan pada individu " + (indexMutate + 1) + " Gen ke " + (i+1));
20
21  //mutasi gen !
22  double  newRandomBobotMakanan = random.nextDouble();
23
24  //jika hasil random gen masih sama (lakukan random ulang)
25  while (newRandomBobotMakanan == populasi.get(indexMutate).getBobotMakanan().get(i)) {
26  newRandomBobotMakanan = random.nextDouble();
27  }
28  mutatedBobotMakanan.set(i, newRandomBobotMakanan);
29  }
30  i++;
31  }
32  mutated.setBobotMakanan(mutatedBobotMakanan);
33  }
34  double  genMutationProb = random.nextDouble();
35
36  ArrayList<Double> randomGenMutationProb = JAlgen.generateRandomDouble(populasi.get(indexMutate).getBobotMakanan().size(), 0, 1);
37
38  //inisialisasi individu termutasi
39  Individu mutated = new Individu(populasi.get(indexMutate).getJenisMakanan());
```

***Source code* 4. 7** Proses mutasi pada individu terpilih

4.2.5 Hitung *Fitness*

Sesuai dengan *flowchart* pada gambar 3.13, proses penghitungan *fitness* dilakukan pada tiap populasi sesuai dengan perancangan pada bab sebelumnya. Pertama akan dilakukan proses hitung karbohidrat, lemak dan protein pada

makanan pokok di pagi hari (5-9). Selanjutnya dilakukan inisialisasi penalti dan dilakukan pengecekan apakah jumlah tiap komponen melakukan pelanggaran atau tidak, kemudian akan dilakukan penghitungan total pelanggaran yang dilakukan oleh individu (baris 13-37). Untuk *fitness* sendiri didapatkan dari rumus (1/penalti) (baris 39). Proses hitung *fitness* dapat dilihat dalam *Source code* 4.8

```
1  ArrayList<Individu> populasi = new ArrayList<Individu>();
2
3  //pagi
4  //pokok
5  grKarbohidrat = (individu.getGrMakanan().get(0) / 100) *
   pokok.get(individu.getJenisMakanan().get(0)).getKarbohidrat
   ();
6
7  grLemak = (individu.getGrMakanan().get(0) / 100) *
   pokok.get(individu.getJenisMakanan().get(0)).getLemak();
8
9  grProtein = (individu.getGrMakanan().get(0) / 100) *
   pokok.get(individu.getJenisMakanan().get(0)).getProtein();
10
11 //penalti
12
13 if (diet.getKebutuhanPagi().getMinKarbohidrat() >
   grKarbohidrat) {
14
15 totalPenalti +=
   Math.abs(diet.getKebutuhanPagi().getMinKarbohidrat() -
   grKarbohidrat) * 5;
16 }
17 else if (diet.getKebutuhanPagi().getMaxKarbohidrat() <
   grKarbohidrat) {
18
19 totalPenalti +=
   Math.abs(diet.getKebutuhanPagi().getMaxKarbohidrat() -
   grKarbohidrat) * 5;
20 }
21 if (diet.getKebutuhanPagi().getMinLemak() > grLemak) {
22
23 totalPenalti +=
   Math.abs(diet.getKebutuhanPagi().getMinLemak() - grLemak);
24
25 }
26 else if (diet.getKebutuhanPagi().getMaxLemak() < grLemak) {
27
28 totalPenalti +=
   Math.abs(diet.getKebutuhanPagi().getMaxLemak() - grLemak);
29 }
30
31 if (diet.getKebutuhanPagi().getMinProtein() > grProtein) {
32
33 totalPenalti +=
   Math.abs(diet.getKebutuhanPagi().getMinProtein() -
```

```
        grProtein);
34      }
35      else if (diet.getKebutuhanPagi().getMaxProtein() < grProtein) {
36
37      totalPenalti += Math.abs(diet.getKebutuhanPagi().getMaxProtein() - grProtein);
38      }
39      individu.setFitness(1 / totalPenalti);
40      }
41      return populasi;
42      }
```

***Source code* 4. 8** Proses hitung *fitness*

4.2.6 Seleksi

Proses seleksi dilakukan dengan metode *Rank Based Fitness*, dimana semakin besar nilai *fitness* suatu individu, maka semakin besar kesempatan terpilih menjadi individu dalam populasi di generasi berikutnya. Sesuai dengan *flowchart* pada gambar 3.15, seleksi ini dimulai dengan mengurutkan nilai *fitness* dari tiap individu yang sudah dihitung dari posisi tertinggi ke posisi terendah (baris 8-20). Dalam proses ini akan diambil individu sejumlah populasi awal, sehingga individu dengan nilai *fitness* terendah akan ditukar (baris 30-33). Hasil pengurutan ditambahkan di populasi *selected* (baris 39). Proses seleksi ini dapat dilihat pada *Source code* 4.9 berikut

```
1   ArrayList<Individu> populasi = new ArrayList<Individu>();
2   populasi.addAll(oldPopulasi);
3   ArrayList<Individu> populasiSelected = new ArrayList<Individu>();
4
5   //sorting by fitness
6   ArrayList<Individu> populasiSorted = populasi;
7
8
9   for (int i = 0; i < populasiSorted.size(); i++) {
10  //find max, put on i
11  double  maxFitness = Double.MIN_VALUE;
12  int toSwap = i;
13  for (int j = i; j < populasiSorted.size(); j++) {
14
15  if (populasi.get(j).getFitness() > maxFitness) {
16
17  maxFitness = populasi.get(j).getFitness();
18  toSwap = j;
```

```
19  }
20  }
21
22  //swap
23  if (toSwap != i) {
24  Individu oldIndividu = populasi.get(i);
25  populasi.set(i, populasi.get(toSwap));
26  populasi.set(toSwap, oldIndividu);
27  }
28
29  }
30  if (toSwap != i) {
31  Individu oldIndividu = populasi.get(i);
32  populasi.set(i, populasi.get(toSwap));
33  populasi.set(toSwap, oldIndividu);
34  }
35
36  }
37
38  for (int i = 0; i < jumlah ; i++) {
39              populasiSelected.add(populasi.get(i));
40          }
41
42  return populasiSelected;
43  }
```

***Source code* 4. 9** Proses seleksi individu untuk generasi berikutnya

4.3 Implementasi Antarmuka Penentuan Komposisi Menu Makanan untuk Penderita *Diabetes Mellitus* menggunakan Algoritma Genetika

Aplikasi ini diterapkan dengan memasukkan data-data yang diperlukan, yaitu data makanan seperti pada lampiran 1, data pasien seperti pada tabel 4.8 dan parameter genetika seperti pada tabel 4.9.

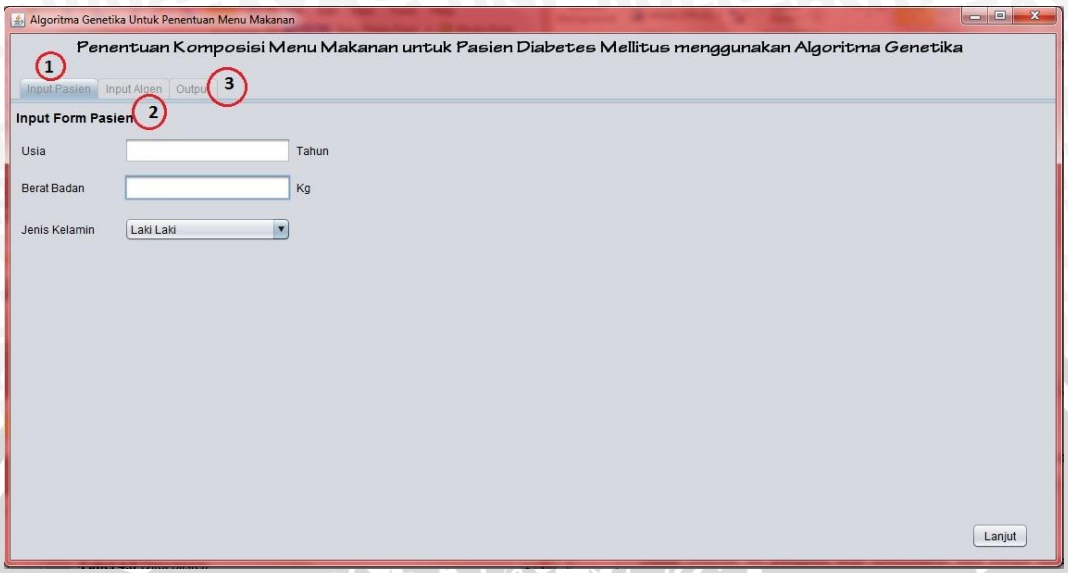
Tabel 4. 8 Data pasien

Umur	45
Berat	72
Jenis Kelamin	Laki-laki

Tabel 4. 9 Data *input* algoritma genetika

Nilai Pm	0.8
Nilai Pc	0.8
Nilai *alpha*	0.6
Jumlah Populasi	30
Jumlah Generasi	100

Tampilan utama dari program Penentuan Komposisi Menu Makanan untuk Penderita *Diabetes Mellitus* menggunakan Algoritma Genetika dapat dilihat pada gambar 4.1.

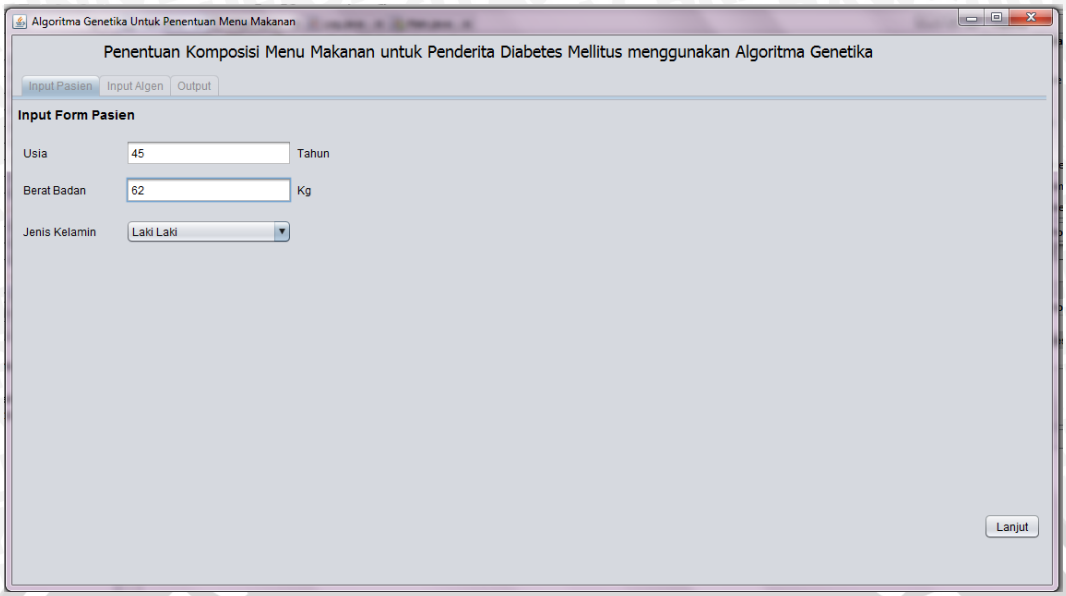


Gambar 4. 1 Tampilan utama program

Program ini terdiri dari 3 bagian utama yaitu data *input* pasien, data *input* algoritma genetika dan *output*. Masing-masing bagian akan dijelaskan sebagai berikut :

1. *Input* Data Pasien

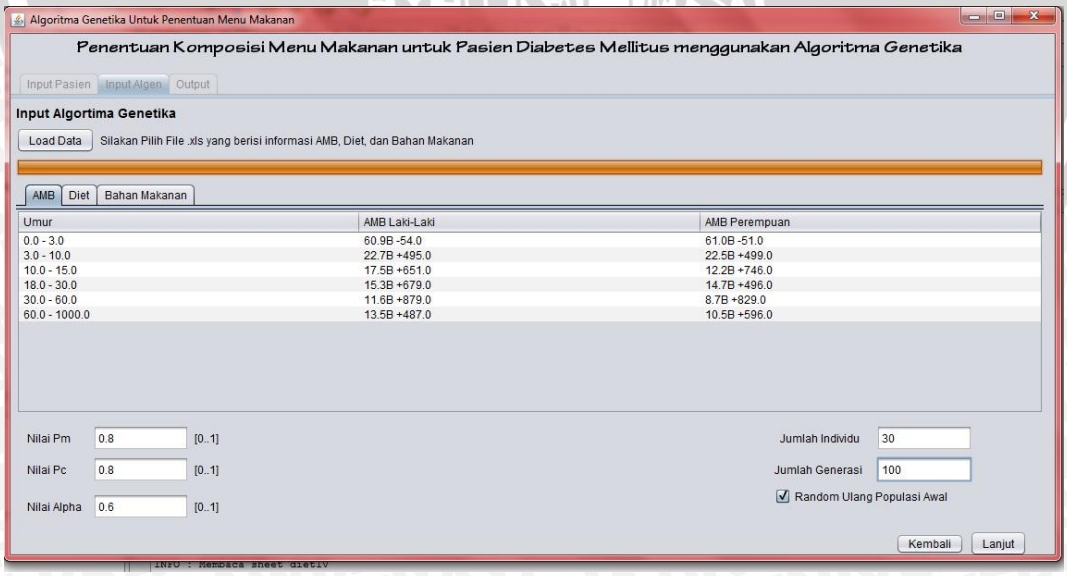
Dalam tampilan ini, pengguna akan memasukkan data berupa umur, berat badan dan jenis kelamin. Lalu klik tombol lanjut sehingga masuk ke tabsheet selanjutnya yaitu menu *input* algoritma genetika. Tampilan form *input* data pasien terdapat pada gambar 4.2 berikut



Gambar 4. 2 Implementasi program

2. *Input* Algoritma Genetika

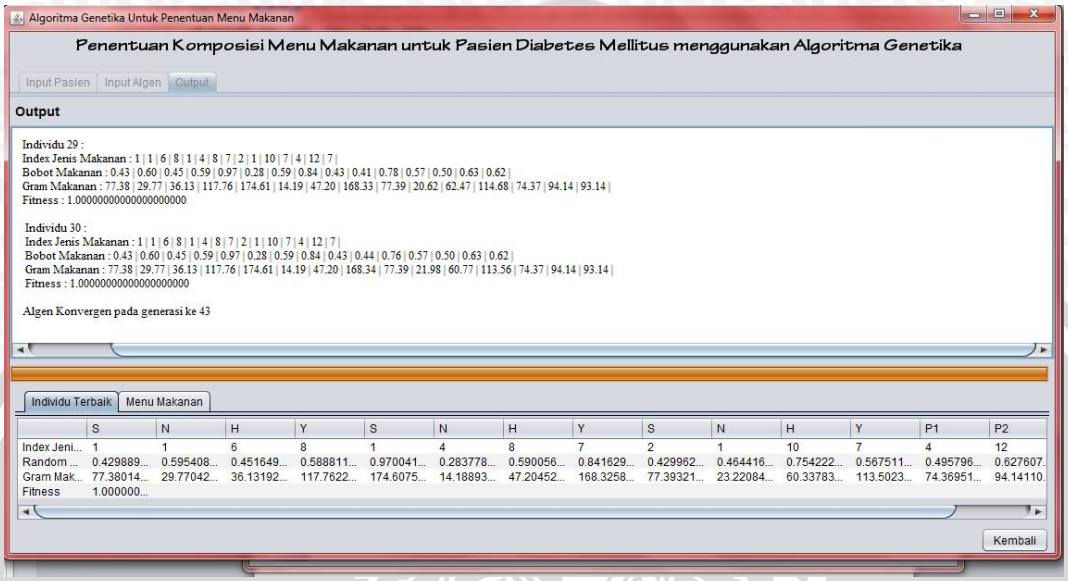
Dalam tampilan ini, pengguna akan melakukan *load* / pembacaan data dari excel dengan melakukan klik pada button *Load* Data, selanjutnya jika data sudah berhasil dibaca maka akan dilakukan proses *input* data algoritma genetika berupa nilai Pm, nilai Pc, nilai alpha, jumlah populasi dan jumlah generasi. Lalu klik tombol lanjut sehingga program akan melakukan proses algoritma genetika. Tampilan *form input* data algen terdapat pada gambar 4.3 berikut



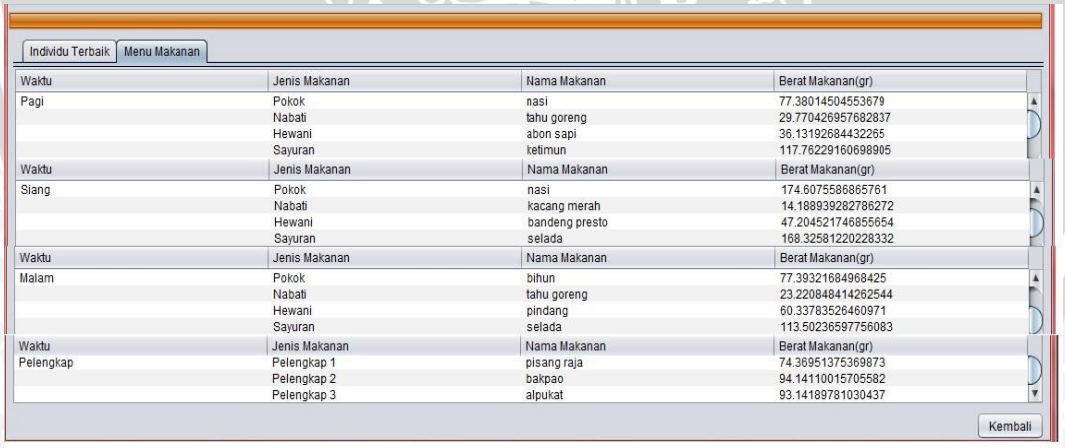
Gambar 4. 3 Implementasi program untuk *input* algoritma genetika

3. Menu *output*

Menu *output* untuk menampilkan proses penghitungan dengan algoritma genetika dan juga untuk menampilkan individu terbaik berupa menu makanan dalam sehari. Tampilan menu *output* seperti pada gambar 4.4.



Gambar 4. 4 Implementasi program bagian *output*



Gambar 4. 5 *Output* berupa Menu Makanan

Setelah dilakukan proses algoritma genetika, diperoleh individu terbaik yaitu individu pada generasi ke 40 dengan nilai *fitness* sebesar 1. Dari individu ini didapatkan menu makanan dan bobotnya dalam tabel 4.10

Tabel 4. 10 Menu makanan dan bobot yang dihasilkan dari implementasi program

Waktu	Makanan	Berat (gr)
Pagi	Nasi	77,38
	Tahu goreng	29,77
	Abon sapi	36
	Ketimun	117,76
Siang	Nasi	174
	Kacang merah	14
	Bandeng presto	47
	Selada	168
Malam	Bihun	77
	Tahu goreng	23
	Pindang	60
	Selada	113,5
Pelengkap	Pisang raja	74,36
	Bakpao	94
	Alpukat	93

4.4 Sistematika Pengujian

Pada sub bab ini akan dijelaskan mengenai sistematika pengujian. Sesuai dengan perancangan pada bab 3, terdapat 3 macam pengujian yang akan dilakukan yaitu uji probabilitas *crossover*-mutasi, uji jumlah generasi dan uji jumlah populasi dimana masing-masing pengujian tersebut dibandingkan terhadap nilai *fitness*.

4.4.1 Sistematika uji probabilitas *crossover* dan probabilitas mutasi terhadap nilai *fitness*

Pengujian yang pertama adalah mengetahui pengaruh perubahan probabilitas *crossover* dan probabilitas mutasi terhadap nilai *fitness* yang dihasilkan dari proses penghitungan menggunakan algoritma genetika. Tujuan dari pengujian ini adalah mengetahui apakah perubahan nilai probabilitas *crossover* dan probabilitas mutasi berpengaruh terhadap perubahan nilai *fitness* serta mengetahui berapa besar nilai peluang *crossover* dan peluang mutasi terbaik. Dalam pengujian ini digunakan parameter jumlah populasi 20, jumlah generasi 100, nilai *alpha* 0.7, usia pasien 47 tahun dengan berat badan 72 kg dan jenis kelamin laki-laki. Selanjutnya, dalam proses pengujian akan dilakukan terhadap

kombinasi dari 9 nilai probabilitas *crossover* dan 9 nilai probabilitas mutasi dengan nilai antara 0.1 – 0.9 . Dalam setiap kali kombinasi, akan dilakukan 5 kali percobaan lalu diambil nilai rata-rata *fitness*nya. Nilai *fitness* inilah yang akan dibandingkan pada tiap kombinasi probabilitas *crossover* dan probabilitas mutasi. Dalam pengujian ini, waktu tidak dipertimbangkan sebagai nilai yang akan dianalisis.

4.4.2 Sistematika uji jumlah generasi terhadap nilai *fitness*

Pengujian yang kedua adalah mengetahui pengaruh perubahan jumlah generasi terhadap besar nilai *fitness* maksimum yang dihasilkan. Dalam pengujian ini, digunakan parameter nilai probabilitas *crossover* 0.7, probabilitas mutasi 0.9, *alpha* 0.6, dan jumlah individu 50. Proses pengujian dilakukan dengan mengubah nilai generasi 10, 30, 50, 80, 100 dan 150 generasi. Pada tiap pengujian 1 jumlah generasi, dilakukan 10 kali percobaan lalu diambil nilai rata-rata *fitness* yang kemudian akan dibandingkan dengan setiap nilai generasi yang lain untuk dianalisa. Dalam pengujian ini, waktu juga tidak dipertimbangkan sebagai nilai yang akan dianalisis.

4.4.3 Sistematika uji ukuran populasi terhadap nilai *fitness*

Pengujian yang ketiga adalah mengetahui pengaruh perubahan ukuran populasi terhadap besar nilai *fitness* maksimum yang dihasilkan. Dalam pengujian ini, digunakan parameter nilai probabilitas *crossover* 0.7, probabilitas mutasi 0.9, *alpha* 0.6, dan jumlah generasi 100. Proses pengujian dilakukan dengan mengubah ukuran populasi 10, 20, 30, 40 dan 50 populasi. Pada tiap pengujian populasi, dilakukan 10 kali percobaan lalu diambil nilai rata-rata *fitness* yang kemudian akan dibandingkan dengan setiap nilai *fitness* populasi yang lain untuk dianalisa. Dalam pengujian ini, waktu juga tidak dipertimbangkan sebagai nilai yang akan dianalisis.

4.5 Implementasi Uji Coba dan Analisa Hasil

Pada sub bab ini akan dijelaskan mengenai hasil uji coba yang telah dilakukan. Sesuai dengan sistematika yang telah dijelaskan sebelumnya.

4.5.1 Pengujian Probabilitas *Crossover* dan Probabilitas Mutasi

Berdasarkan sistematika uji coba yang telah dijelaskan sebelumnya, pengujian probabilitas *crossover* dan probabilitas mutasi dilakukan pada rentang 0.1-0.9. Setiap kombinasi nilai probabilitas *crossover* dan probabilitas mutasi dilakukan uji coba selama 5 kali dan diambil nilai rata-ratanya. Hasil uji coba dapat dilihat pada tabel 4.11.

Tabel 4. 11 Pengujian probabilitas *crossover* 0.1 – 0.9 dan probabilitas mutasi 0.1 – 0.4

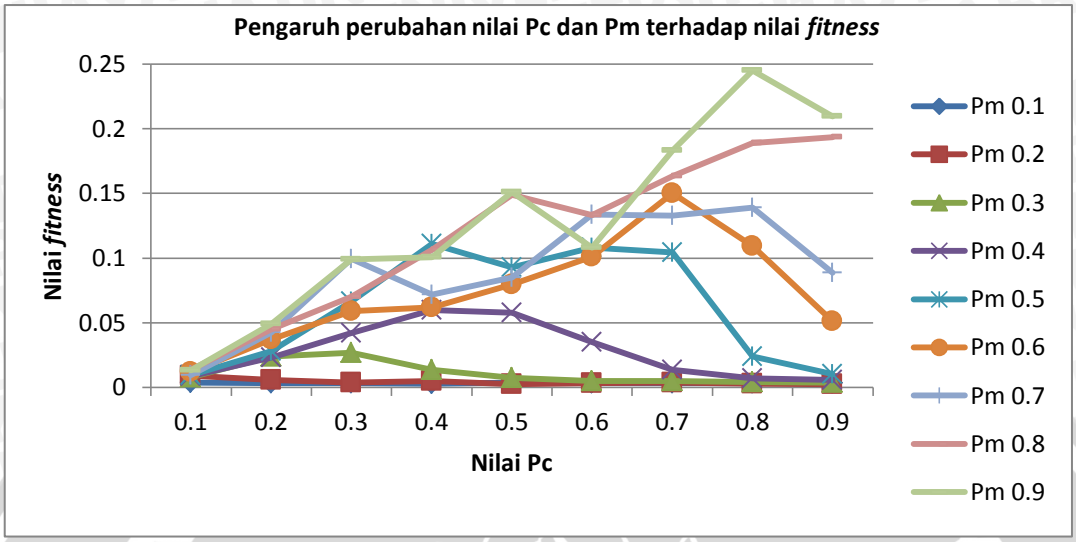
Nilai Pc	Nilai Pm			
	0.1	0.2	0.3	0.4
0.1	0.003584	0.0032216	0.0032902	0.0029114
0.2	0.0091054	0.0056954	0.0038782	0.0051146
0.3	0.0079972	0.02389328	0.0268912	0.013584
0.4	0.0086086	0.0232106	0.042123	0.059679
0.5	0.0097188	0.0282494	0.0663162	0.11087
0.6	0.012024	0.0370048	0.0591122	0.0619686
0.7	0.0096572	0.0425552	0.0992936	0.071781
0.8	0.013468	0.0450154	0.0703034	0.1063344
0.9	0.0131096	0.0491542	0.0990862	0.1008602

Tabel 4. 12 Pengujian probabilitas *crossover* 0.1 – 0.9 dan probabilitas mutasi 0.5 – 0.9

Nilai Pc	Nilai Pm				
	0.5	0.6	0.7	0.8	0.9
0.1	0.003636	0.0034718	0.0032384	0.003231	0.003132
0.2	0.0026594	0.003499	0.0036688	0.0030116	0.0025738
0.3	0.0073952	0.0048698	0.0051346	0.0042494	0.0037898
0.4	0.0579674	0.0349562	0.013495	0.0072196	0.0056784
0.5	0.0929644	0.1082426	0.1042414	0.0240236	0.0105234
0.6	0.0794706	0.101031	0.1503062	0.1095258	0.0510568
0.7	0.0849246	0.1337482	0.1327678	0.13914254	0.0886582
0.8	0.1488158	0.1334508	0.1632812	0.1888312	0.193354489
0.9	0.1509602	0.1081422	0.1831924	0.2451526	0.209472

Dari tabel 4.12 dapat dilihat bahwa nilai *fitness* rata-rata terbesar adalah 0,2451526. Nilai ini terdapat pada saat nilai Pc 0.9 dan nilai Pm 0.8 . Adapun

grafik pengaruh probabilitas *crossover* dan probabilitas mutasi terhadap nilai *fitness* rata-rata dapat dilihat pada gambar 4.6



Gambar 4. 6 Grafik pengujian probabilitas *crossover* dan probabilitas mutasi

Berdasarkan grafik perbandingan nilai Pc dan Pm dengan nilai *fitness* rata-rata dari gambar 4.6, dapat dilihat seiring perubahan nilai probabilitas *crossover* dan probabilitas mutasi maka nilai *fitness* rata-rata yang didapat fluktuatif. Nilai *fitness* rata-rata seperti ini, dimungkinkan terjadi karena proses *random* yang dilakukan pada saat proses *crossover* dan mutasi. Untuk kombinasi nilai Pc dan Pm yang seimbang atau berdekatan, dihasilkan nilai *fitness* rata-rata puncak untuk setiap Pm, hal ini terlihat misalnya pada kombinasi Pm 0.9 dan Pc 0.8, lalu pada Pm 0.6 dan Pc 0.7, dan seterusnya.

Nilai *fitness* rata-rata yang relatif tinggi tersebut dapat disebabkan karena semakin seringnya proses *crossover* terjadi sehingga dimungkinkan diperoleh variasi individu yang semakin banyak. Semakin besar nilai probabilitas *crossover* maka semakin banyak kemungkinan individu yang disilangkan (*crossover*), tentunya akan semakin banyak individu baru yang diciptakan. Dengan demikian peluang untuk didapatkan individu yang memiliki nilai *fitness* yang tinggi akan semakin besar.

Untuk nilai Pm dan Pc yang rendah, dihasilkan nilai *fitness* rata-rata yang buruk. Hal ini disebabkan sedikitnya variasi individu yang terbentuk, sehingga nilai *fitness* yang dihasilkan cenderung seragam. Sedangkan terjadi penurunan

nilai *fitness* rata-rata saat nilai Pm tinggi, bisa disebabkan oleh individu yang baik ikut terkena mutasi dan setelah di mutasi tidak menjadi solusi yang lebih baik dari keadaan sebelum di mutasi. Selain itu, dapat di lihat pada grafik di atas bahwa pada perubahan nilai peluang mutasi cenderung seragam meskipun mengalami kenaikan. Dengan metode mutasi pada jenis dan bobot makanan cukup berpengaruh pada nilai *fitness* sehingga dimungkinkan bisa menjadi lebih baik.

4.5.2 Pengujian Jumlah Generasi

Berdasarkan sistematika uji coba yang telah dijelaskan sebelumnya, proses pengujian dilakukan dengan mengubah nilai generasi dari 10, 30, 50, 80, 100 dan 150 generasi. Pada setiap pengujian jumlah generasi, dilakukan 10 kali percobaan untuk kemudian diambil nilai rata-rata *fitness*nya. Hasil uji coba dapat dilihat pada tabel 4.13.

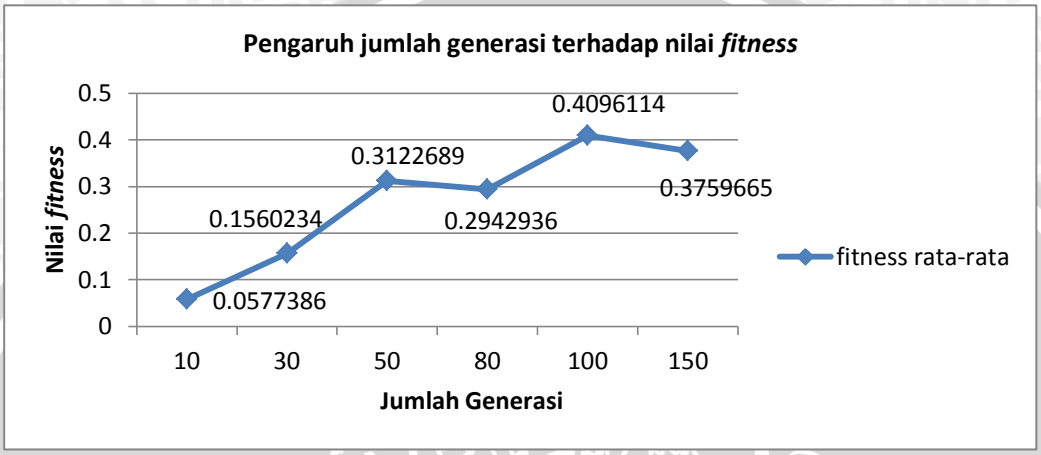
Tabel 4. 13 Hasil Uji Coba ke 1-6 untuk Perubahan Jumlah generasi terhadap nilai *fitness*

Jumlah Generasi	Percobaan ke-					
	1	2	3	4	5	6
10	0.062306	0.05677	0.046251	0.046665	0.040983	0.059543
30	0.081519	0.109061	0.437137	0.104892	0.173316	0.155173
50	0.279168	0.422694	0.163739	0.176056	0.113429	0.398042
80	0.342825	0.845447	0.074185	0.140411	0.178057	0.24474
100	0.74625	0.127689	0.179218	0.082062	0.17007	0.142731
150	1	0.135192	0.178958	0.22394	0.158922	0.615554

Tabel 4. 14 Hasil Uji Coba ke 7-10 untuk Perubahan Jumlah generasi terhadap nilai *fitness*

Jumlah Generasi	Percobaan ke-				
	7	8	9	10	*Fitness* rata-rata
10	0.083564	0.053918	0.043962	0.083424	0.0577386
30	0.103411	0.160184	0.090759	0.144782	0.1560234
50	1	0.262537	0.149753	0.157271	0.3122689
80	0.167649	0.158119	0.167366	0.624137	0.2942936
100	1	1	0.153174	0.49492	0.4096114
150	0.127477	0.141716	0.177906	1	0.3759665

Dari tabel 4.14 ditunjukkan bahwa nilai *fitness* rata-rata paling besar yang dihasilkan adalah 0.4096114 pada saat jumlah generasi sebesar 100. Sedangkan nilai *fitness* rata-rata paling kecil adalah 0.0577386 pada saat jumlah generasi sebesar 10. Pergerakan nilai *fitness* terhadap jumlah generasi digambarkan pada grafik 4.6 .



Gambar 4. 7 Grafik pengujian jumlah generasi terhadap fungsi *fitness*

Pada gambar 4.7 dapat dilihat bahwa perbandingan jumlah generasi dan nilai *fitness* rata-rata yang fluktuatif. Dalam grafik terlihat nilai *fitness* rata-rata turun pada saat generasi 80 kemudian mengalami kenaikan kembali dan turun pada saat jumlah generasi 150. Nilai yang fluktuatif ini bisa disebabkan oleh *starting point* dalam proses algoritma genetika. Pada setiap satu generasi, akan dilakukan beberapa kali proses *crossover* dan mutasi, dimana pada proses ini individu baru akan dihasilkan. Semakin sering proses *crossover* dilakukan maka individu baru yang terbentuk akan semakin banyak dan variasi *fitness* akan semakin beragam.

Jumlah generasi yang tinggi akan mengakibatkan proses evolusi semakin sering dilakukan sehingga kesempatan untuk terjadinya proses pembentukan individu-individu barupun semakin besar pula. Semakin banyak individu yang dihasilkan, memungkinkan semakin banyak variasi nilai *fitness* nya, sehingga peluang untuk mendapatkan nilai *fitness* yang tinggi akan semakin besar.

4.5.3 Pengujian Ukuran Populasi

Pengujian yang terakhir adalah uji ukuran populasi. Digunakan 5(lima) macam ukuran populasi yaitu 10, 20, 30, 40 dan 50 populasi. Pada tiap pengujian populasi, dilakukan 10 kali percobaan lalu diambil nilai rata-rata *fitness* yang kemudian akan dibandingkan dengan setiap nilai *fitness* populasi yang lain. Hasil pengujian akan ditunjukkan dalam tabel 4.13 berikut.

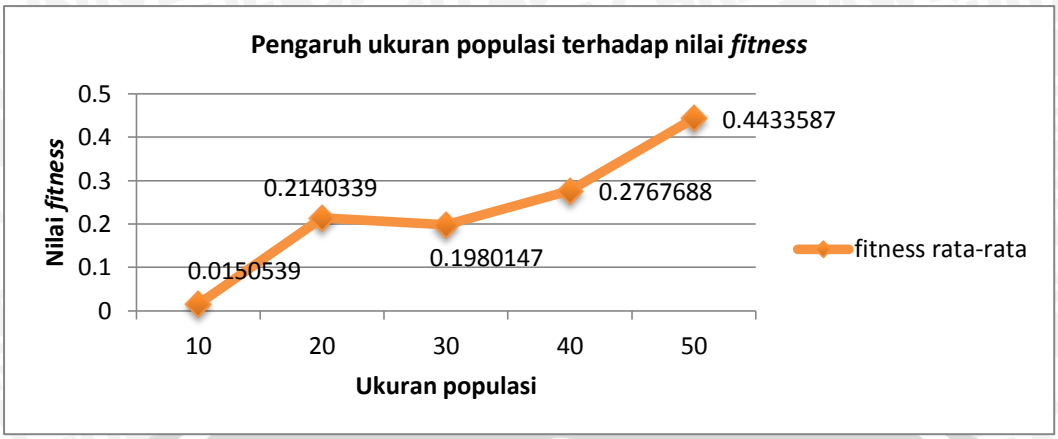
Tabel 4. 15 Hasil Uji Coba ke 1- 6 untuk pengaruh perubahan ukuran populasi terhadap nilai *fitness*

Jumlah Populasi	Percobaan ke-					
	1	2	3	4	5	6
10	0.01225	0.016083	0.0237	0.012996	0.010271	0.009681
20	0.08497	0.152621	0.099825	1	0.106529	0.123517
30	0.207804	0.12064	0.105899	0.119563	0.114307	0.140354
40	0.100627	0.194559	0.143898	0.139801	0.227892	0.814921
50	1	0.359383	0.168304	0.164055	0.457145	0.722304

Tabel 4. 16 Hasil Uji Coba ke 7- 10 untuk pengaruh perubahan ukuran populasi terhadap nilai *fitness*

Jumlah Populasi	Percobaan ke-				
	7	8	9	10	*Fitness* rata-rata
10	0.007572	0.029036	0.020059	0.008891	0.0150539
20	0.115062	0.249644	0.094411	0.11376	0.2140339
30	0.139951	0.738035	0.182587	0.111007	0.1980147
40	0.128687	0.152053	0.738406	0.126844	0.2767688
50	0.156256	0.217603	0.210204	0.978333	0.4433587

Dari tabel 4.13 ditunjukkan bahwa nilai *fitness* rata-rata terbesar yang dihasilkan dari pengujian ini adalah 0.4433587 pada saat ukuran populasi sebesar 50. Sedangkan nilai *fitness* rata-rata paling kecil adalah 0.0150539 pada saat ukuran populasi sebesar 10. Pergerakan nilai *fitness* terhadap ukuran populasi digambarkan pada gambar 4.8 sebagai berikut.



Gambar 4. 8 Grafik pengujian jumlah populasi terhadap fungsi *fitness*

Dari gambar 4.8, dapat dilihat bahwa perbandingan jumlah populasi terhadap nilai *fitness* rata-rata cenderung naik, yaitu setiap pertambahan jumlah populasi akan diikuti pula bertambahnya nilai *fitness* rata-rata. Dalam grafik terlihat nilai *fitness* rata-rata turun pada saat generasi 30 tapi pada akhirnya grafik akan naik kembali. Banyaknya ukuran populasi mempengaruhi jumlah individu baru yang dihasilkan. Dengan adanya proses *crossover* dan proses mutasi dalam ukuran populasi yang besar, memungkinkan jumlah individu baru yang dihasilkan juga semakin besar dan beragam. Tentunya hal ini akan berpengaruh pula terhadap variasi nilai *fitness* yang dihasilkan oleh individu–individu baru. Sehingga peluang untuk mendapatkan nilai *fitness* yang tinggi semakin besar.

Dari hasil pengujian parameter pengaruh nilai Pc dan Pm, pengujian jumlah generasi dan pengujian ukuran populasi, didapatkan nilai maksimum *fitness* rata-rata < 0.5, hal ini disebabkan karena pada saat pembangkitan individu atau *starting point* dilakukan secara *random.*

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, dapat diambil kesimpulan sebagai berikut :

1. Algoritma Genetika dapat digunakan untuk permasalahan penyusunan menu makanan untuk penderita *Diabetes Mellitus* dengan menerapkan pengkodean *floating point, simple arithmetic crossover, random mutation* dan *rank based fitness selection*.
2. Penyelesaian permasalahan penyusunan menu makanan untuk penderita *Diabetes Mellitus* menggunakan algoritma genetika dipengaruhi oleh beberapa parameter algoritma genetika. Pengaruh parameter probabilitas *crossover* (Pc) menunjukkan bahwa semakin besar nilai Pc dan diikuti dengan nilai probabilitas mutasi yang seimbang, maka nilai *fitness* cenderung semakin tinggi (baik). Hal tersebut juga terjadi pada jumlah populasi dan jumlah generasi, seiring dengan bertambahnya jumlah populasi dan/atau jumlah generasi akan cenderung menaikkan nilai *fitness* .

5.2 Saran

Saran yang dapat diberikan untuk pengembangan lebih lanjut adalah sebagai berikut :

1. Ditambahkan komponen kandungan nutrisi bahan yang lain, tidak hanya protein, lemak dan karbohidrat.
2. Ditambahkan untuk pasien *Diabetes Mellitus* dengan komplikasi penyakit lain.
3. Ditambahkan pengujian untuk parameter ukuran populasi dan jumlah generasi sehingga diharapkan bisa menghasilkan nilai *fitness* yang lebih tinggi sesuai dengan teori Algoritma Genetika.

DAFTAR PUSTAKA

[ALM-03] Almatsier,Sunita. 2003. *Prinsip Dasar Ilmu Gizi*. Jakarta:Gramedia Pustaka Utama.

[ALM-08] Almatsier, Sunita. 2008. *Penuntun Diet*. Jakarta: Gramedia Pustaka Utama.

[ALM-04] Almatsier, Sunita. 2004. *Penuntun Diet edisi baru*. Jakarta: Gramedia Pustaka Utama.

[ANO[1]-12] Anonim1. 2012. http://nationalgeographic.co.id. Diakses pada tanggal 16 Maret 2012 pukul 09.18 WIB.

[ANO[2]-12] Anonim2. 2012. *Bab 7 Algoritma Genetika*. http://lecturer.eepis-its.edu. Diakses pada tanggal 1 April 2012 pukul 11.55 WIB.

[BAS-03] Basuki, Achmad. 2003. *ALGORITMA GENETIKA Suatu Alternatif Penyelesaian Permasalahan Searching, Optimasi dan Machine Learning*. http://budi.blog.undip.ac.id. Diakses pada tanggal 1 April 2012 pukul 11.45 WIB.

[DAN-12] Danas. 2012. *Kombinatorik*. http://mti.ugm.ac.id/~adji/courses/resources/lectures/DiscMath/tugas.doc. Diakses pada tanggal 1 April 2012 pukul 11.35 WIB.

[DIE-92] Diehl, Hans. 1992. *Waspadai Diabetes, Kolesterol, Hipertensi*. Bandung: Indonesia Publishing House.

[GEN-00] Gen, Mitsuo and Cheng, Runwei. 2000. *Genetic Algorithms & Engineering Optimization*. Canada : John Willey & Sons, Inc.

[HAN-02] Hannawati, Anies dan Thiang, Eleazer. 2002. *Pencarian Rute Optimum Menggunakan Algoritma Genetika.* Jurnal Teknik Elektro Vol 2 no 2. September 2007.

[KUS-03] Kusumadewi, Sri. 2003. *Artificial Intelligence*. Yogyakarta : Graha Ilmu.

[MAR-98] Marilyn, Johnson. 1998. *Diabetes ; Terapi dan Pencegahannya*. Bandung : Indonesia Publishing House .

[MIC-96] Michalewicz, Zbigniew. 1996. *Genetic Algorithms+Data Structures = Evolution Programs*. Germany : Springer.

[PRA-11] Pranata, Lamhot. 2011. *Penerapan Algoritma Genetika pada Penentuan Komposisi Pakan Ikan*. SKRIPSI. Fakultas MIPA. Universitas Brawijaya.

[SAD-09] Sadeghzadeh. 2009. *Task Schedullingin Distributed Environment Using Genetic Algorithm*. Proceeding of the 9th WSEAS International Conference on Applied Informatics and Communications.

[SET-03] Setiawan, Kuswara. 2003. *Paradigma Sistem Cerdas*. Malang : Bayumedia Publishing .

[SUY-05] Suyanto. 2005. *Algoritma Genetika dalam Matlab*. Yogyakarta : Andi.

[SUY-10] Suyanto. 2010. *Algoritma Optimasi atau Probabilitik*. Yogyakarta: Graha Ilmu .

[TOM-10] Tomoko, Kashima et al. 2008 . *Well-Balanced menu Planning with Fuzzy Weight* .

[UYU-11] Uyun, Shofwatun dan Hartati, Sri. 2011. http://journal.uii.ac.id/index.php/Snati/article/viewFile/2196/2019. *Penentuan Komposisi Bahan Pangan untuk Diet Penyakit Ginjal dan Saluran kemih dengan Algoritma Genetika.* Seminar Nasional Aplikasi Teknologi Informasi 2011 (SNATI, 2011). Yogyakarta, 17-18 Juni 2011.

[ZUL-04] Zuliansyah, Mochammad. 2004. *Alternatif Pemecahan Masalah Open Shop Schedulling dengan Pendekatan Algoritma Genetik dan Heuristik.* Seminar Nasional Aplikasi Teknologi Informasi 2004.

Lampiran 1

Daftar Bahan Makanan per 100 gram dan Kandungan Zat Gizinya

Makanan Pokok

Indeks	Nama	Energi (gr)	Karbohidrat (gr)	Lemak (gr)	Protein (gr)
0	nasi tim	120	26	0,4	2,4
1	nasi	180	39,8	0,3	3
2	bihun	348	82,1	0,1	4,7
3	nasi beras merah	149	32,5	0,4	2,8
4	kentang	62	13,5	0,2	2,1
5	mi basah	88	14	3,3	0,6
6	ketupat	109	13,4	5,2	2,2

Sumber Protein Nabati

Indeks	Nama	Energi (gr)	Karbohidrat (gr)	Lemak (gr)	Protein (gr)
0	kembang tahu	380	23,3	13,8	48,9
1	tahu goreng	115	2,5	8,5	9,7
2	tahu	80	0,8	4,7	10,9
3	tempe murni	201	13,5	8,8	20,8
4	kacang merah	171	28	2,2	11
5	oncom	187	22,6	6	13
6	jamur kuping segar	21	0,9	0,6	3,8

Sumber Protein Hewani

Indeks	Nama	Energi (gr)	Karbohidrat (gr)	Lemak (gr)	Protein (gr)
0	kakap	92	0	0,7	20
1	ayam	298	0	25	18,2
2	daging sapi	201	0	14	18,8
3	hati sapi	132	6	3,2	19,7
4	udang	91	0,1	0,2	21
5	ikan segar	113	0	4,5	17
6	abon sapi	212	59,3	10,6	18
7	telur puyuh	116	1,6	7	10,7
8	bandeng presto	296	11,3	20,3	17,1
9	telur ayam kampung	174	1,2	14	10,8
10	pindang	124	0	9,6	9,5
11	mujair	89	0	1	18,7

Sayuran

Indeks	Nama	Energi (gr)	Karbohidrat (gr)	Lemak (gr)	Protein (gr)
0	kangkung	28	3,9	0,7	3,4
1	bayam	16	2,9	0,4	0,9
2	sawi	28	4	0,3	2,3
3	buncis	34	7,2	0,3	2,4
4	wortel	36	7,9	0,6	1
5	tauge	34	4,3	1,2	3,7
6	terong	28	5,5	0,2	1,1
7	selada	18	2,9	0,2	1,2
8	ketimun	8	1,4	0,2	0,2
9	kembang kol	25	4,9	0,2	2,4
10	kapri muda	45	9	0,2	3,3
11	jantung pisang	32	7,1	0,3	1,2
12	gambas	19	4,1	0,2	0,8

Pelengkap

Indeks	Nama	Energi (gr)	Karbohidrat (gr)	Lemak (gr)	Protein (gr)
0					
1	manggis	63	15,6	0,6	0,6
2	apel	58	14,9	0,4	0,3
3	bengkoang	55	12,8	0,2	1,4
4	semangka	28	6,9	0,2	0,5
5	pisang raja	120	31,8	0,2	1,2
6	pepaya	46	12,2	0	0,5
7	salak	77	20,9	0	0,4
8	alpukat	85	7,7	6,5	0,9
9	belimbing	36	8,8	0,4	0,4
10	anggur	50	12,8	0,2	0,5
11	pisang ambon	99	25,8	0,2	1,2
12	roti kukus	249	52,5	2,1	5,1
13	bakpao	239	41,6	2,6	12,2
14	pastel	208	31,4	15,4	5,2
15	risoles	134	28,2	1,4	2,1
16	kelepon	215	41,8	3,7	3,7
17	kue bakpia	272	44,1	6,7	3,7

Lampiran 2

Perhitungan Komponen Penyusun Makanan Tiap Individu

Individu ke-	Nama makanan	Berat (gr)	Karbohidrat (gr)	Lemak (gr)	Protein (gr)
Individu 1					
Makan pagi	bihun	72	59,112	0,072	3,384
	tempe murni	25	3,375	2,2	5,2
	ayam	8	0	2	1,456
	buncis	40	2,88	0,12	0,96
	Total		65,367	4,392	11
Makan siang	nasi	144	57,312	0,432	4,32
	kacang merah	5	1,4	0,11	0,55
	bandeng presto	48	5,424	9,744	8,208
	sawi	40	1,6	0,12	0,92
	Total		65,736	10,406	13,998
Makan malam	nasi	90	35,82	0,27	2,7
	kacang merah	20	5,6	0,44	2,2
	daging sapi	24	0	3,36	4,512
	bayam	40	1,16	0,16	0,36
	Total		42,58	4,23	9,772
pelengkap	apel	45	6,705	0,18	0,135
	alpukat	90	6,93	5,85	0,81
	manggis	105	16,38	0,63	0,63
	Total		30,015	6,66	1,575
Individu 2					
Makan pagi	Nasi	54	21,492	0,162	1,296
	Tahu	25	0,25	1,175	2,725
	Udang	16	0,016	0,032	3,36
	buncis	40	2,88	0,12	0,96
	Total		24,638	1,489	8,341
Makan siang	nasi beras merah	126	40,95	0,504	3,528
	tahu	40	0,32	1,88	4,36
	daging sapi	20	0	2,8	3,76
	bayam	40	1,16	0,16	0,36
	Total		42,43	5,344	12,008
Makan malam	kentang	36	4,86	0,072	0,756
	tahu goreng	40	1	3,4	3,88
	daging sapi	32	0	4,48	6,016

	buncis	180	12,96	0,54	4,32
	Total		18,82	8,492	14,972
pelengkap	bengkoang	30	3,84	0,06	0,42
	-	0	0	0	0
	pisang raja	90	28,62	0,18	1,08
	Total		32,46	0,24	1,5
Individu 3					
Makan pagi	Nasi	126	50,148	0,378	3,78
	Tahu	25	0,2	1,175	2,725
	daging sapi	24	0	3,36	4,512
	Wortel	80	6,32	0,48	0,8
	Total		56,668	5,393	11,817
Makan siang	nasi beras merah	90	29,25	0,36	2,52
	tahu goreng	15	0,375	1,275	1,455
	telur puyuh	56	0,896	3,92	5,992
	Kembang kol	40	1,96	0,08	0,96
	Total		32,481	5,635	10,927
Makan malam	Nasi	72	28,656	0,216	2,16
	Oncom	10	2,26	0,6	1,3
	daging sapi	64	0	8,96	12,032
	kembang kol	40	1,96	0,08	0,96
	Total		32,876	9,856	16,452
pelengkap	Manggis	120	18,72	0,72	0,72
	Alpukat	90	6,93	5,85	0,81
	-	0	0	0	0
	Total		25,65	6,57	1,53
Individu 4					
Makan pagi	Kentang	90	12,15	0,18	1,89
	kacang merah	45	12,6	0,99	4,95
	ayam	8	0	2	1,456
	bayam	160	4,64	0,64	1,44
	Total		29,39	3,81	9,736
Makan siang	Nasi	108	42,984	0,324	3,24
	kacang merah	5	1,4	0,11	0,55
	Ayam	40	0	10	7,28
	Buncis	140	10,08	0,42	3,36
	Total		54,464	10,854	14,43
Makan malam	Nasi	90	35,82	0,27	2,7
	jamur kuping	35	0,315	0,21	1,33

	segar				
	daging sapi	40	0	5,6	7,52
	Bayam	160	4,64	0,64	1,44
	Total		40,775	6,72	12,99
pelengkap	Apel	45	6,705	0,18	0,135
	belimbing	75	6,6	0,3	0,3
	alpukat	45	3,465	2,925	0,405
	Total		16,77	3,405	0,84
Individu 5					
Makan pagi	Nasi	72	28,656	0,216	2,16
	tempe murni	10	1,35	0,88	2,08
	telur ayam kampung	64	0,768	8,96	6,912
	Sawi	40	1,6	0,12	0,92
	Total		32,374	10,176	12,072
Makan siang	mi basah	108	15,12	3,564	0,648
	jamur kuping segar	20	0,18	0,12	0,76
	telur ayam kampung	24	0,288	3,36	2,592
	Sawi	140	5,6	0,42	3,22
	Total		21,188	7,464	7,22
Makan malam	bihun	90	73,89	0,09	4,23
	kembang tahu	15	3,495	2,07	7,335
	daging sapi	56	0	7,84	10,528
	Buncis	40	2,88	0,12	0,96
	Total		80,265	10,12	23,053
pelengkap	Pepaya	45	5,49	0	0,225
	Pisang raja	90	28,62	0,18	1,08
	-	0	0	0	0
	Total		34,11	0,18	1,305

Individu ke-	Nama makanan	Berat (gr)	Karbohidrat (gr)	Lemak (gr)	Protein (gr)
Child 1					
Makan pagi	bihun	72	59,112	0,072	3,384
	tempe murni	25	3,375	2,2	5,2
	ayam	8	0	2	1,456
	buncis	40	2,88	0,12	0,96

	Total		65,367	4,392	11
Makan siang	nasi	144	57,312	0,432	4,32
	kacang merah	5	1,4	0,11	0,55
	bandeng presto	48	5,424	9,744	8,208
	sawi	40	1,6	0,12	0,92
	Total		65,736	10,406	13,998
Makan malam	nasi	90	35,82	0,27	2,7
	kacang merah	20	5,6	0,44	2,2
	daging sapi	32	0	4,48	6,016
	bayam	140	4,06	0,56	1,26
	Total		45,48	5,75	12,176
pelengkap	apel	45	6,705	0,18	0,135
	alpukat	90	6,93	5,85	0,81
	manggis	75	11,7	0,45	0,45
	Total		25,335	6,48	1,395

Individu ke-	Nama makanan	Berat (gr)	Karbohidrat (gr)	Lemak (gr)	Protein (gr)
Child 2					
Makan pagi	Kentang	90	12,15	0,18	1,89
	kacang merah	45	12,6	0,99	4,95
	ayam	8	0	2	1,456
	bayam	160	4,64	0,64	1,44
	Total		29,39	3,81	9,736
Makan siang	Nasi	108	42,984	0,324	3,24
	kacang merah	5	1,4	0,11	0,55
	Ayam	40	0	10	7,28
	Buncis	140	10,08	0,42	3,36
	Total		54,464	10,854	14,43
Makan malam	Nasi	90	35,82	0,27	2,7
	jamur kuping segar	35	0,315	0,21	1,33
	daging sapi	32	0	4,48	6,016
	Bayam	140	4,06	0,56	1,26
	Total		40,195	5,52	11,306
pelengkap	Apel	45	6,705	0,18	F61
	belimbing	90	7,92	0,36	0,36
	alpukat	75	5,775	4,875	0,675
	Total		20,4	5,415	1,035

Individu ke-	Nama makanan	Berat (gr)	Karbohidrat (gr)	Lemak (gr)	Protein (gr)
Child 1'					
Makan pagi	nasi	90	35.82	0.27	2.7
	tempe murni	25	3.375	2.2	5.2
	udang	56	0.056	0.112	11.76
	buncis	60	4.32	0.18	1.44
	Total		43.571	2.762	21.1
Makan siang	mi basah	90	35.82	0.27	2.7
	kacang merah	5	1.4	0.11	0.55
	udang	48	0.048	0.096	10.08
	bayam	80	2.32	0.32	0.72
	Total		39.588	0.796	14.05
Makan malam	mi basah	144	57.312	0.432	4.32
	tahu	25	0.2	1.175	2.725
	abon sapi	16	9.488	1.696	2.88
	bayam	140	4.06	0.56	1.26
	Total		71.06	3.863	11.185
pelengkap	pisang raja	60	19.08	0.12	0.72
	alpukat	90	6.93	5.85	0.81
	manggis	75	11.7	0.45	0.45
	Total		37.71	6.42	1.98