BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi

Implementasi adalah penerapan hasil perancangan yang telah dilakukan pada tahap analisis dan perancangan sistem. Implementasi yang dilakukan antara lain menerapkan hasil analisis dan perancangan perangkat keras, perangkat lunak dan antarmuka.

4.1.1 Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam membangun sistem untuk mengimplementasikan algoritma genetika untuk menyelesaikan permainan *puzzle* kakuro terdapat pada tabel 4.1.

Tabel 4.1 Implementasi Perangkat Keras

No	Perangkat Keras	Spesifikasi
1	Prosesor	Intel Core i3 1,8 Ghz
2	Monitor	Monitor 14”
3	Memori	4096 GB
4	VGA	1024 GB

4.1.2 Implementasi Perangkat Lunak

Perangkat keras yang digunakan dalam membangun sistem untuk mengimplementasikan algoritma genetika untuk menyelesaikan permainan *puzzle* kakuro terdapat pada tabel 4.2.

Tabel 4.2 Implementasi Perangkat Lunak

No	Perangkat Lunak	Spesifikasi
1	Sistem Operasi	Windows 7 Ultimate 64 Bit
2	Pembuat Aplikasi	Microsoft Visual Studio 2010

4.1.3 Implementasi *Class Diagram*

Implementasi class merupakan implementasi dari perancangan class diagram ke dalam bentuk fisik file-file dengan ekstensi .cs yang digunakan dalam pembuatan dan jalannya proses dalam sistem. Berikut ini beberapa implementasi class yang telah terdefinisi, sedangkan untuk *source code* dari *class* tersebut terdapat pada lampiran B.

Tabel 4.3 Implementasi *Class*

No	Nama Kelas	Nama Fisik Kelas
1	Genetika	Genetika.cs
2	*Puzzle*_Kakuro	*Puzzle*_Kakuro.cs

4.1.4 Implementasi Antarmuka

Berikut ini adalah penjelasan implementasi antarmuka pada sistem untuk mengimplementasikan algoritma genetika untuk menyelesaikan permainan *puzzle* kakuro.

Tabel 4.4 Implementasi Antarmuka

No	Nama Antarmuka	Deskripsi
1	*Puzzle* Kakuro	Digunakan untuk melakukan proses pembangkitan nilai soal, proses implementasi algoritma genetika untuk menyelesaikan permainan *puzzle* kakuro dan menampilkan hasil dari proses algoritma genetika setiap generasinya.

4.2 Pengujian Sistem

Pengujian sistem dilakukan untuk menemukan kesalahan dan kekurangan yang tidak ditemukan sebelumnya, pengujian ini dilakukan khususnya pada

pengujian fungsionalitas sistem serta kinerja sistem yang telah dibangun. Proses penguijan ini dilakukan dengan metode *White Box* serta pengujian *Black Box.*

4.2.1 Rencana Pengujian

Pada bagian rencana pengujian, digunakan rencana pengujian fungsionalitas yaitu dengan menggunakan metode *black box* yang berfokus pada persyaratan fungsional perangkat lunak serta *white box* yang berfokus untuk mengetahui kinerja logika telah bekerja dengan baik atau tidak . Berikut adalah rencana pengujian yang dilakukan pada tabel 4.5 :

Tabel 4.5 Rencana Pengujian

Item Uji	Detail Pengujian	Jenis
Pembangkitan nilai soal	Membangkitkan nilai soal permainan *puzzle* kakuro.	*Black Box*
Penyelesaian *Puzzle* Kakuro	Menguji algoritma genetika untuk menyelesaikan permainan *puzzle* kakuro.	*Black Box dan White Box*
Hasil Algoritma Genetika	Menampilkan hasil algoritma genetika setiap generasinya	*Black Box*

4.2.2 Hasil Pengujian

Hasil pengujian didapat dari pengujian *white box* dan *black box*. Kesesuaian item mana saja yang akan menggunakan pengujian *white box* dan *black box* dilakukan sesuai dengan rencana pengujian.

4.2.2.1 Hasil Pengujian *Black Box*

Teknik pengujian *Black Box* yang akan dilakukan adalah *Equivalence Partitioning.* Berdasarkan rencana pengujian, maka dapat dilakukan pengujian fungsionalitas pada implementasi algoritma genetika untuk menyelesaikan permainan *puzzle* kakuro sebagai berikut :

1. Pengujian Pembangkitan Nilai Soal

   Hasil dari pengujian untuk membangkitkan nilai soal *puzzle* kakuro dapat dilihat pada tabel 4.6 :

Tabel 4.6 Pengujian Pembangkitan Nilai Soal

Kasus dan Data Hasil Uji (Data Valid)			
Aktifitas yang dilakukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Menekan tombol Bangkitkan.	Menghasilkan nilai soal untuk setiap lajur *puzzle* kakuro dan muncul pesan bahwa nilai soal telah berhasil dibangkitkan.	Nilai soal untuk setiap lajur *puzzle* kakuro berhasil dibangkitkan dan muncul pesan nilai soal telah berhasil dibangkitkan.	[√] Diterima [ ] Ditolak

2. Pengujian Penyelesaian *Puzzle* Kakuro

   Hasil dari pengujian untuk melihat hasil dari algoritma genetika dapat dilihat pada tabel 4.7 :

Tabel 4.7 Pengujian Penyelesaian *Puzzle* Kakuro

Kasus dan Data Hasil Uji (Data Valid)			
Aktifitas yang dilakukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Mengisi waktu percobaan kemudian menekan tombol Selesaikan.	Proses akan berjalan sesuai dengan waktu yang dimasukkan, jika proses selesai akan keluar pesan *puzzle* berhasil terselesaikan atau tidak berhasil diselesaikan.	Proses berjalan sesuai dengan waktu yang dimasukkan dan keluar pesan bahwa *puzzle* tidak berhasil diselesaikan.	[√] Diterima [ ] Ditolak

3. Pengujian Hasil Algoritma Genetika

   Hasil dari pengujian untuk melihat hasil dari algoritma genetika dapat dilihat pada tabel 4.8 :

Tabel 4.8 Pengujian Hasil Algoritma Genetika

Kasus dan Data Hasil Uji (Data Valid)			
Aktifitas yang dilakukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Memilih *Radio Button* Tampilkan, kemudian menekan tombol Selesaikan.	Hasil dari algoritma genetika akan ditampilkan setiap generasinya.	Hasil algoritma genetika ditampilkan.	[√] Diterima [ ] Ditolak

4. Kesimpulan Pengujian *Black Box*

   Berdasarkan hasil pengujian *black box* dapat ditarik kesimpulan bahwa seluruh fungsionalitas dapat berfungsi sebagaimana mestinya.

4.2.2.2 Hasil Pengujian White Box

Pengujian *white box* bertujuan untuk mengetahui kinerja logika yang dibuat pada sebuah perangkat lunak apakah berjalan dengan baik atau tidak. Pengujian *white box* akan digunakan pada algoritma genetika, untuk mengukur kinerja logika yang telah dibuat. Berikut hasil pengujian untuk white box pada algoritma genetika untuk menyelesaikan permainan *puzzle* kakuro :

1. Perubahan Kode Program ke Flowgraph

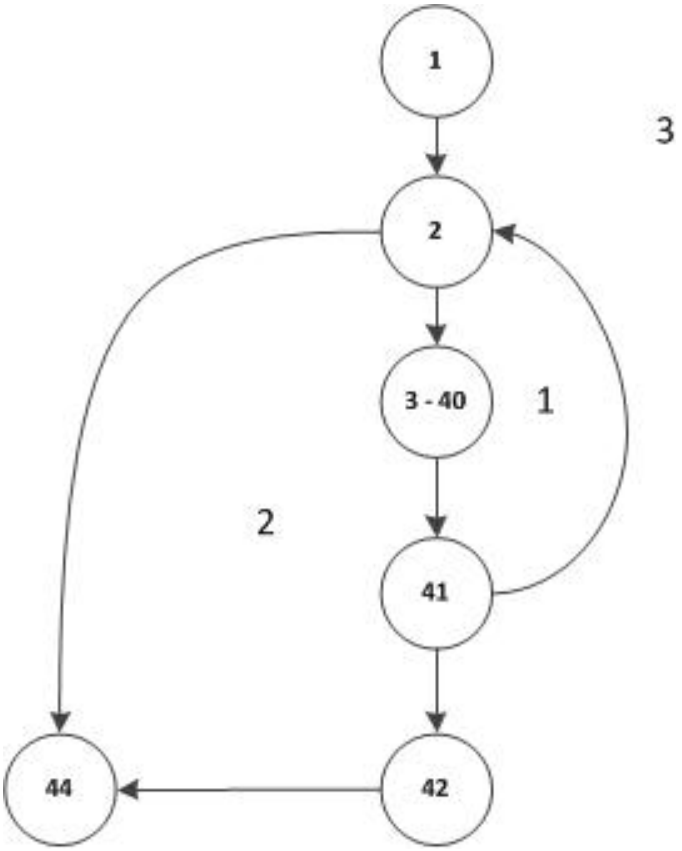
   Pada tabel 4.9 berisikan kode program yang digunakan untuk melakukan proses penyelesaian *puzzle* kakuro dengan mengimplementasikan algoritma genetika.

Tabel 4.9 Kode Program

No	Kode Program
1	genetika.PopulasiAwal14x14();
2	while (solve != true) {
3	genetika.HitungFitness14x14_1();
4	CekJawaban14x14();
5	genetika.HitungFitness14x14_2();
6	CekJawaban14x14();
7	genetika.HitungFitness14x14_3();
8	CekJawaban14x14();
9	genetika.HitungFitness14x14_4();
10	CekJawaban14x14();
11	genetika.HitungFitness14x14_5();
12	CekJawaban14x14();
13	genetika.HitungFitness14x14_6();
14	CekJawaban14x14();
15	genetika.HitungFitness14x14_7();
16	CekJawaban14x14();
17	genetika.HitungFitness14x14_8();
18	CekJawaban14x14();
19	genetika.HitungFitness14x14_9();
20	CekJawaban14x14();
21	genetika.HitungFitness14x14_10();
22	CekJawaban14x14();
23	genetika.Crossover14x14();
24	genetika.Mutasi14x14();
25	genetika.HitungFitnessAnak14x14_1();
26	CekJawaban14x14();
27	genetika.HitungFitnessAnak14x14_2();
28	CekJawaban14x14();
29	genetika.HitungFitnessAnak14x14_3();
30	CekJawaban14x14();
31	genetika.HitungFitnessAnak14x14_4();

```
32          CekJawaban14x14();
33          genetika.HitungFitnessAnak14x14_5();
34          CekJawaban14x14();
35          genetika.HitungFitnessAnak14x14_6();
36          CekJawaban14x14();
37          genetika.HitungFitnessAnak14x14_7();
38          CekJawaban14x14();
39          genetika.Seleksi14x14();
40          genetika.PopulasiBaru14x14();
41          if (s1.Elapsed.Minutes.ToString()==textWaktu.Text.Trim())
            {
42              break;
            }
44      }
```

Kode program tersebut kemudian akan diubah ke dalam bentuk flowgraph. Gambar 4.1 berikut ini menggambarkan flowgraph yang merupakan hasil terjemahan dari kode program proses penyelesaian *puzzle* kakuro dengan mengimplementasikan algoritma genetika.



Gambar 4. 1 Flowgraph Proses Penyelesaian *Puzzle* Kakuro Dengan Mengimplementasikan Algoritma Genetika

Keterangan :

 : Menggambarkan simpul atau *node*.

 : Menggambarkan busur atau *edge*.

2. Perhitungan Cyclomatic Complexity

   Perhitungan Cyclomatic Complexity menggunakan rumus sebagai berikut :

   V(G) = E – N + 2

   dengan, E adalah *edge* dan N adalah *node.*

   Maka, diperoleh :

   V(G) = 7 – 6 + 2 = 3

3. Perhitungan Region Testing

   Perhitungan Region Testing menggunakan rumus sebagai berikut :

   Jumlah Region = jumlah kurva tertutup + 1

   Maka, diperoleh :

   Jumlah Region = 2 + 1 = 3

4. Perhitungan Independent Path

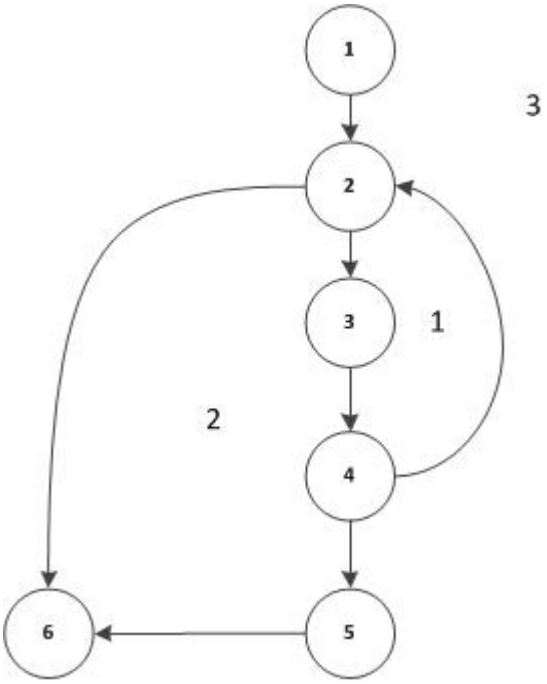
   Berikut ini adalah *independent path* untuk flowgraph proses penyelesaian *puzzle* kakuro dengan mengimplementasikan algoritma genetika yang terdapat pada tabel 4.10 :

Tabel 4.10 Independent Path

No	***Path***
1	1-2-44
2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-2-44
3	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-44

5. Penomoran Ulang Flowgraph

   Berikut adalah hasil penomoran ulang dari flowgraph proses penyelesaian *puzzle* kakuro dengan mengimplementasikan algoritma genetika yang terdapat pada gambar 4.2 :



Gambar 4.2 Penomoran Ulang Flowgraph

6. Pembentukan Matriks dan Perhitungan V(G)

   Untuk pembentukan matriks dan perhitungan V(G) dapat dilihat pada tabel 4.11 :

Tabel 4.11 Pembentukan Matriks dan Perhitungan V(G)

Node	1	2	3	4	5	6	Jumlah
1		1					0
2			1			1	1
3				1			0
4		1		1			1
5						1	0
6							0
	Jumlah						2

Perhitungan V(G) dapat dilakukan dengan rumus sebagai berikut :

V(G) = x + 1

V(G) = 2 + 1 = 3

7. *Test Case Independent Path*

Tabel 4.12 *Test Case Independent Path*

No	Kasus Uji	Hasil Yang Diharapkan	Hasil Pengamatan	Keterangan
1	Kondisi *solve* bernilai *true*. (*puzzle* terselesaikan) (*Path 1*)	Sistem keluar dari kondisi *while* dan mengeluarkan status *puzzle* berhasil diselesaikan.	Jika *puzzle* berhasil diselesaikan, sistem keluar dari kondisi *while* dan mengeluarkan status *puzzle* berhasil diselesaikan.	[√] Alur Terlewati [ ] Alur Tidak Terlewati
2	Kondisi sistem berhasil menyelesaikan *puzzle* setelah melakukan looping sebanyak minimal 1 kali. (*Path 2*)	Sistem keluar dari kondisi *while* dan mengeluarkan status *puzzle* berhasil diselesaikan.	Jika *puzzle* berhasil diselesaikan, sistem keluar dari kondisi *while* dan mengeluarkan status *puzzle* berhasil diselesaikan.	[√] Alur Terlewati [ ] Alur Tidak Terlewati
3	Kondisi sistem tidak berhasil menyelesaikan *puzzle* dalam batas waktu yang ditentukan. (*Path 3*)	Sistem keluar dari kondisi *while* dan mengeluarkan status *puzzle* berhasil tidak berhasil diselesaikan.	Sistem keluar dari kondisi *while* dan mengeluarkan status *puzzle* berhasil tidak berhasil diselesaikan.	[√] Alur Terlewati [ ] Alur Tidak Terlewati

8. Kesimpulan

   Berdasarkan pengujian *white box* dengan menggunakan perhitungan *Cyclomatic Complexcity*, *Region Testing*, *Independent Path* dan *Graph Matrics Testing* menghasilkan nilai 3, serta hasil dari test case yang menunjukan bahwa seluruh alur dapat terlewati, maka dapat disimpulkan bahwa pengujian *white box* pada algoritma genetika berjalan dengan baik.

4.2.2.3 Hasil Pengujian Implementasi Algoritma Genetika

Pengujian implementasi algoritma genetika dilakukan untuk menguji algoritma genetika yang diimplementasikan untuk menyelesaikan permainan *puzzle* kakuro. Percobaan akan dilakukan pada 3 batasan waktu yaitu 10 menit, 20 menit dan 30 menit. Berikut adalah hasil dari pengujian tersebut :

1. Pengujian Dengan Waktu 10 Menit

   Pada pengujian ini, didapatkan 34.046 generasi, dengan jumlah lajur yang benar sebanyak 54 dari total 87 lajur dan memiliki nilai f_{total} terbesar 1,62. Berikut adalah hasil dari algoritma genetika pada generasi ke 34.046 yang terdapat pada tabel 4.12 :

Tabel 4.13 Hasil Algoritma Genetika Generasi Ke 34.046

Kromosom	Fitness	Lajur Benar
1	1,62	54
2	1,62	54
3	1,62	54
4	1,62	54
5	1,62	54
6	1,62	54
7	1,62	54
8	1,62	54
9	1,62	54

10	1,62	54
11	1,62	54
12	1,62	54
13	1,62	54
14	1,62	54
15	1,62	54
16	1,62	54
17	0,96	0

2. Pengujian Dengan Waktu 20 Menit

   Pada pengujian ini, didapatkan 68.075 generasi, dengan jumlah lajur yang benar sebanyak 52 dari total 87 lajur dan memiliki nilai f_{total} terbesar 1,60. Berikut adalah hasil dari algoritma genetika pada generasi ke 68.075 yang terdapat pada tabel 4.13 :

Tabel 4.14 Hasil Algoritma Genetika Generasi Ke 68.075

Kromosom	Fitness	Lajur Benar
1	1,60	52
2	1,60	52
3	1,60	52
4	1,60	52
5	1,60	52
6	1,60	52
7	1,60	52
8	1,60	52
9	1,60	52
10	1,60	52
11	1,60	52
12	1,60	52
13	1,60	52
14	1,60	52

15	1,60	52
16	1,60	52
17	0,98	0

3. Pengujian Dengan Waktu 30 Menit

Pada pengujian ini, didapatkan 102.337 generasi, dengan jumlah lajur yang benar sebanyak 60 dari total 87 lajur dan memiliki nilai f_{total} terbesar 1,69. Berikut adalah hasil dari algoritma genetika pada generasi ke 102.337 yang terdapat pada tabel 4.14 :

Tabel 4.15 Hasil Algoritma Genetika Generasi Ke 102.337

Kromosom	Fitness	Lajur Benar
1	1,69	60
2	1,69	60
3	1,69	60
4	1,69	60
5	1,69	60
6	1,69	60
7	1,69	60
8	1,69	60
9	1,69	60
10	1,69	60
11	1,69	60
12	1,69	60
13	1,69	60
14	1,69	60
15	1,69	60
16	1,69	60
17	0,98	0

Untuk nilai soal serta hasil nilai jawaban dari pengujian dapat dilihat pada lampiran D.

4.2.2.3.1 Kesimpulan Pengujian Implementasi Algoritma Genetika

Hasil yang diperoleh pada pengujian algoritma genetika untuk menyelesaikan *puzzle* kakuro adalah algoritma genetika tidak dapat diimplementasikan untuk menyelesaikan *puzzle* kakuro, nilai jawaban yang didapatkan dari hasil algoritma genetika tidak dapat memenuhi aturan yang mengharuskan akumulasi jumlah angka yang terdapat pada setiap lajur harus sama dengan nilai soalnya. Hal ini ditunjukan dengan nilai f_2 dari hasil pengujian yang hanya maksimal bernilai 0,69 dari maksimal bernilai 1.