Praktikum Sistem Cerdas

NRP	:	3223600019
Nama	:	Muhammad Bimo Fachrizky
Materi	:	Algoritma Genetika
Tanggal	:	Senin, 26 Mei 2025

Praktikum 11

Algoritma Genetika

I. Tujuan Pembelajaran

- Mahasiswa dapat memahami dan menjelaskan konsep Algoritma Genetika
- Mahasiswa dapat menjelaskan model Algoritma Genetika
- Mahasiswa dapat membuat aplikasi Algoritma Genetika

Software yang di perlukan

- Microsoft Visual C++
- PyCharm

II. Langkah percobaan

1. Algoritma Genetika.cpp

```
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>

#define POP_SIZE 4
#define SIZE 5

int individu[POP_SIZE][SIZE];
int individuBaru[POP_SIZE][SIZE];
int fitnessValue[POP_SIZE];
int totalFitness, bestFitness;
double percentValue[POP_SIZE];

void init_population() {
   for (int i = 0; i < POP_SIZE; i++) {
      for (int j = 0; j < SIZE; j++) {
         individu[i][j] = rand() % 2;
      }
   }
}

int fitness() {
   int jum;
   totalFitness = 0;

   for (int i = 0; i < POP_SIZE; i++) {
      jum = 0;
```

```
        for (int j = 0; j < SIZE; j++) {
            jum += individu[i][j];
        }
        fitnessValue[i] = jum;
        totalFitness += jum;
    }

    bestFitness = fitnessValue[0];
    for (int i = 1; i < POP_SIZE; i++) {
        if (fitnessValue[i] > bestFitness)
            bestFitness = fitnessValue[i];
    }

    for (int i = 0; i < POP_SIZE; i++) {
        printf("Individu[%d]: ", i + 1);
        for (int j = 0; j < SIZE; j++) {
            printf("%d", individu[i][j]);
        }
        printf(" NF: %d\n", fitnessValue[i]);
    }

    printf("Total Fitness: %d Best Fitness: %d\n", totalFitness, bestFitness);
    return bestFitness;
}

void rouletteSelection() {
    double totalPercentFitness = 0.0;

    for (int j = 0; j < POP_SIZE; j++) {
        percentValue[j] = (fitnessValue[j] * 100.0) / totalFitness;
        printf("Percent individu[%d]: %.2f\n", j + 1, percentValue[j]);
        totalPercentFitness += percentValue[j];
    }

    for (int i = 0; i < POP_SIZE; i++) {
        double daerahPersen = 0.0;
        double random = rand() % 100;
        int k;

        for (k = 0; k < POP_SIZE; k++) {
            daerahPersen += percentValue[k];
            if (random <= daerahPersen) break;
        }
```

```
      for (int j = 0; j < SIZE; j++) {
         individuBaru[i][j] = individu[k][j];
      }
   }

   for (int i = 0; i < POP_SIZE; i++) {
      printf("Individu[%d]: ", i + 1);
      for (int j = 0; j < SIZE; j++) {
         individu[i][j] = individuBaru[i][j];
         printf("%d", individu[i][j]);
      }
      printf("\n");
   }
}

void crossOver() {
   for (int i = 0; i < POP_SIZE; i += 2) {
      int r = rand() % SIZE;
      printf("Crossover point (random): %d\n", r);

      for (int j = 0; j < SIZE; j++) {
         if (j < r) {
            individuBaru[i][j]     = individu[i][j];
            individuBaru[i + 1][j] = individu[i + 1][j];
         } else {
            individuBaru[i][j]     = individu[i + 1][j];
            individuBaru[i + 1][j] = individu[i][j];
         }
      }
   }

   for (int i = 0; i < POP_SIZE; i++) {
      printf("Individu[%d]: ", i + 1);
      for (int j = 0; j < SIZE; j++) {
         individu[i][j] = individuBaru[i][j];
         printf("%d", individu[i][j]);
      }
      printf("\n");
   }
}

void mutasi() {
```

```
    for (int i = 0; i < POP_SIZE; i += 2) {
        int r = rand() % SIZE;
        printf("Mutasi index (random): %d\n", r);

        for (int j = 0; j < SIZE; j++) {
            individuBaru[i][j] = individu[i][j];
        }

        individuBaru[i][r] = 1 - individu[i][r];  // Flip bit
    }

    for (int i = 0; i < POP_SIZE; i++) {
        printf("Individu[%d]: ", i + 1);
        for (int j = 0; j < SIZE; j++) {
            individu[i][j] = individuBaru[i][j];
            printf("%d", individu[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int best;

    init_population();

    for (int iter = 0; iter < 100; iter++) {
        best = fitness();
        if (best == SIZE) break;

        rouletteSelection();
        crossOver();
        mutasi();
    }

    getch();
    return 0;
}
```

2. Algoritma Genetika Untuk Mencari Nama Secara Acak

```
#include <iostream>
#include <vector>
```

```
#include <string>
#include <cstdlib>
#include <ctime>
#include <algorithm>

using namespace std;

const string target = "BASUKI";
const string karakter = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
const int jumlah_populasi = 100;
const float mutation_rate = 0.2;

struct Individu {
    string angka;
    int fitness;

    Individu(string s) {
        angka = s;
        fitness = hitungFitness(s);
    }

    static int hitungFitness(const string& s) {
        int f = 0;
        for (int i = 0; i < target.size(); ++i) {
            if (s[i] == target[i]) {
                f += 26;
            } else {
                f += 26 - abs(s[i] - target[i]);
            }
        }
        return f;
    }
};

bool bandingFitness(const Individu& a, const Individu& b) {
    return a.fitness > b.fitness;
}

string randomString(int length) {
    string result = "";
    for (int i = 0; i < length; ++i) {
        result += karakter[rand() % karakter.length()];
    }
```

```
    return result;
}

Individu crossover(const Individu& parent1, const Individu& parent2) {
    string child = "";
    for (int i = 0; i < parent1.angka.size(); ++i) {
        if ((rand() % 2) == 0)
            child += parent1.angka[i];
        else
            child += parent2.angka[i];
    }

    for (int i = 0; i < child.size(); ++i) {
        if (((float)rand() / RAND_MAX) < mutation_rate) {
            child[i] = karakter[rand() % karakter.length()];
        }
    }

    return Individu(child);
}

int main() {
    srand(time(0));
    vector<Individu> populasi;

    for (int i = 0; i < jumlah_populasi; ++i) {
        populasi.push_back(Individu(randomString(target.size())));
    }

    int generasi = 1;

    while (true) {
        for (auto& individu : populasi) {
            individu.fitness = Individu::hitungFitness(individu.angka);
        }

        sort(populasi.begin(), populasi.end(), bandingFitness);

        cout << generasi << " "
             << populasi[0].angka
             << " >> Fitness = " << populasi[0].fitness << endl;

        if (populasi[0].angka == target) {
```

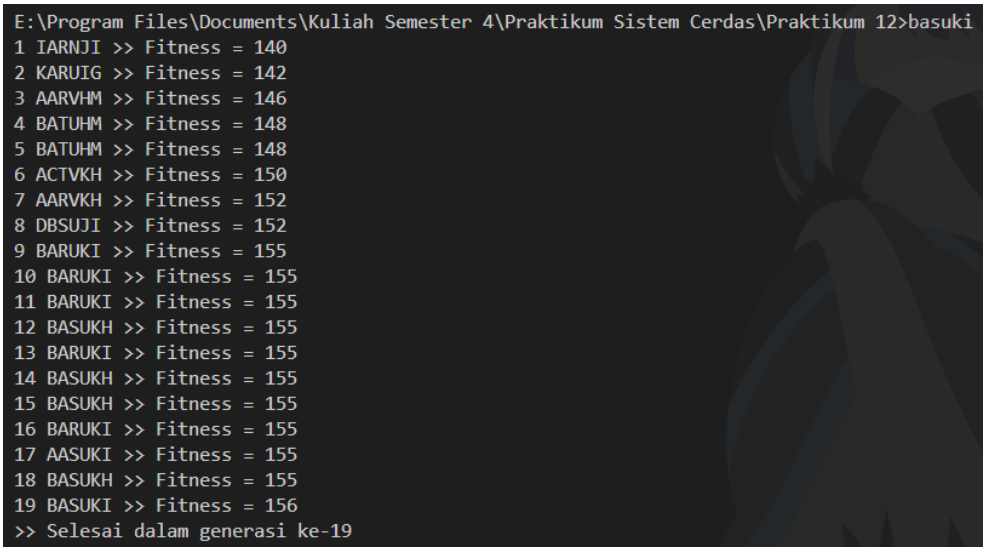
```
            cout << ">> Selesai dalam generasi ke-" << generasi << endl;
            break;
        }

        vector<Individu> new_populasi;

        for (int i = 0; i < 10; ++i) {
            new_populasi.push_back(populasi[i]);
        }

        while (new_populasi.size() < jumlah_populasi) {
            int idx1 = rand() % 50;
            int idx2 = rand() % 50;
            Individu anak = crossover(populasi[idx1], populasi[idx2]);
            new_populasi.push_back(anak);
        }

        populasi = new_populasi;
        generasi++;
    }

    return 0;
}
```

III. Hasil Percobaan

1. Algoritma Genetika.cpp

```
E:\Program Files\Documents\Kuliah Semester 4\Praktikum Sistem Cerdas\Praktikum 12>algoritma
Individu[1]: 11001 NF: 3
Individu[2]: 00000 NF: 0
Individu[3]: 11111 NF: 5
Individu[4]: 11010 NF: 3
Total Fitness: 11 Best Fitness: 5
```

2. Algoritma Genetika untuk mencari nama secara acak



IV. Analisa

Praktikum ini mencakup implementasi dasar dari algoritma genetika (genetic algorithm) yang digunakan untuk mencari solusi terbaik berdasarkan nilai fitness maksimum dari kumpulan individu biner. Algoritma ini terdiri dari beberapa tahapan, yaitu inisialisasi populasi, evaluasi fitness, seleksi, crossover, dan mutasi, yang dilakukan secara iteratif. Populasi awal diinisialisasi secara acak dengan kombinasi bilangan biner (0 dan 1) sepanjang 5 bit untuk setiap individu, dan terdapat 4 individu dalam satu generasi (POP_SIZE = 4, SIZE = 5).

Setiap individu dievaluasi menggunakan fungsi fitness() yang menghitung jumlah bit 1 dalam representasi individu sebagai nilai fitness. Total dan nilai fitness terbaik dari semua individu dicatat, dan informasi ini ditampilkan ke terminal. Kemudian, proses seleksi dilakukan menggunakan metode roulette wheel selection, yaitu dengan mengalokasikan peluang pemilihan berdasarkan proporsi fitness terhadap total fitness. Individu dengan nilai fitness lebih tinggi memiliki peluang lebih besar untuk dipilih sebagai induk generasi berikutnya.

Setelah seleksi, fungsi crossOver() akan melakukan pertukaran gen (bit) antara pasangan individu pada titik acak untuk menghasilkan variasi genetik baru. Selanjutnya, fungsi mutasi() melakukan perubahan nilai gen (bit flip dari 0 ke 1 atau sebaliknya) secara acak pada sebagian individu untuk menjaga keberagaman populasi dan menghindari konvergensi dini. Proses ini dilakukan berulang hingga ditemukan individu dengan fitness sempurna (jumlah bit 1 sebanyak 5), atau hingga iterasi mencapai 100 kali.

Hasil pada gambar menunjukkan proses evaluasi fitness dari populasi awal. Terdapat 4 individu dengan representasi biner sepanjang 5 bit. Nilai fitness dihitung berdasarkan jumlah bit 1 dalam setiap individu. Total fitness seluruh individu adalah 11, dan individu terbaik memiliki fitness 5 (maksimum, karena SIZE = 5).

Ini berarti ada satu solusi optimal yang sudah ditemukan, yaitu 11111, dan iterasi akan berhenti setelah ini.

Program kedua merupakan implementasi sederhana dari algoritma genetika untuk mencari string target "BASUKI" dengan menggunakan populasi string acak yang berevolusi tiap generasi. Program memulai dengan membuat sejumlah individu (string acak) sebagai populasi awal, kemudian setiap individu dihitung nilai fitness-nya berdasarkan kemiripan dengan target. Proses evolusi berlangsung dengan memilih 10 individu terbaik, lalu menghasilkan keturunan baru melalui operasi crossover dan mutasi dari individu-individu teratas tersebut, menggantikan populasi lama. Iterasi ini diulang hingga ditemukan individu yang persis sama dengan string target. Program menampilkan generasi ke berapa dan string dengan fitness tertinggi setiap iterasi, serta menghentikan proses saat target berhasil ditemukan.

V. Kesimpulan

Praktikum ini mengimplementasikan algoritma genetika secara dasar untuk mencari solusi optimal secara iteratif melalui proses seleksi, crossover, dan mutasi pada populasi individu. Program pertama menggunakan representasi biner sederhana dengan evaluasi fitness berdasarkan jumlah bit 1, sedangkan program kedua menggunakan string karakter dan menghitung fitness berdasarkan kemiripan dengan string target.