LAPORAN PRAKTIKUM ALGORITMA DAN PEMOGRAMAN

(Dosen pengampu : Sherly Gina Supratman, M.Kom.)

Nama : Muhammad Rizky
NIM : 20240810023
Kelas : TINFC-2024-02

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS KUNINGAN

2024

A. Latar Belakang

Dalam Perkembangan Teknologi yang berkembang sangat cepat ini, algoritma dan pemograman merupakan salah dua penopang perkembangan teknologi yang begitu cepat, dengan demikian jika kita ingin ikut berkontribusi dalam perkembangan teknologi, kita perlu memahami apa itu algorima dan programan, dan untuk memahaminya kita bisa mulai dari mempelajari programan dasar yang di implementasikan melalui salah satu matakuliah di semester satu ini.

B. Tujuan

Tujuan di laksanakanya praktikum ini, sebagian sebagai penerapan dari modul 10.

C. Landasan Teori

Menurut sebuah artikel dari kazokku, Proyek IT merujuk pada upaya pengembangan, implementasi, atau pemeliharaan sistem informasi atau teknologi di sebuah perusahaan. Dalam dunia bisnis yang semakin kompleks dan berubah dengan cepat, proyek IT menjadi penting untuk menjaga daya saing dan memenuhi kebutuhan pasar yang terus berubah. Namun, tidak semua perusahaan memiliki sumber daya internal yang cukup untuk menangani proyek IT secara efektif. Di sinilah layanan Talenta IT Outsourcing KAZOKKU hadir menawarkan beragam IT manpower yang bisa Anda manfaatkan.

D. Alat, Bahan dan Perangkat

Alat, bahan, prangkat kersa, dan perangkat lunak yang digunakan:

1. Laptop
2. Visual Studio Code

E. Prosuder Kerja

1. Tugas anda adalah penerapan algoritma genetika berdasarkan masalah yang akan diselesaikan.

F. Hasil

tenth > tugas.cpp > initializeSchedule(int, int, const vector<vector<int>>&)

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <algorithm>

using namespace std;

struct Schedule {
    vector<int> slots; // Slot waktu untuk setiap mata kuliah
    int conflicts;     // Jumlah konflik
};

// Fungsi untuk menghitung konflik dalam jadwal
int calculateConflicts(const vector<int>& slots, const vector<vector<int>>& studentCourses) {
    int conflicts = 0;
    for (const auto& student : studentCourses) {
        for (size_t i = 0; i < student.size(); ++i) {
            for (size_t j = i + 1; j < student.size(); ++j) {
                if (slots[student[i]] == slots[student[j]]) {
                    conflicts++;
                }
            }
        }
    }
    return conflicts;
}
```

```cpp

// Inisialisasi jadwal acak
Schedule initializeSchedule(int numCourses, int numSlots, const vector<vector<int>>& studentCourses) {
    Schedule schedule;
    schedule.slots.resize(numCourses);
    for (int i = 0; i < numCourses; ++i) {
        schedule.slots[i] = rand() % numSlots;
    }
    schedule.conflicts = calculateConflicts(schedule.slots, studentCourses);
    return schedule;
}

// Mutasi jadwal
void mutate(Schedule& schedule, int numSlots, const vector<vector<int>>& studentCourses) {
    if ((double)rand() / RAND_MAX < 0.2) { // 20% peluang mutasi
        int course = rand() % schedule.slots.size();
        schedule.slots[course] = rand() % numSlots;
        schedule.conflicts = calculateConflicts(schedule.slots, studentCourses);
    }
}

// Crossover dua jadwal
Schedule crossover(const Schedule& parent1, const Schedule& parent2, const vector<vector<int>>& studentCourses) {
    Schedule child;
    child.slots.resize(parent1.slots.size());
    for (size_t i = 0; i < parent1.slots.size(); ++i) {
        child.slots[i] = (rand() % 2 == 0) ? parent1.slots[i] : parent2.slots[i];
    }
    child.conflicts = calculateConflicts(child.slots, studentCourses);
    return child;
}

```

```cpp
int main() {
    srand(time(0));

    // Input data dinamis
    int numCourses, numStudents, numSlots;
    cout << "Masukkan jumlah mata kuliah: ";
    cin >> numCourses;

    cout << "Masukkan jumlah mahasiswa: ";
    cin >> numStudents;

    cout << "Masukkan jumlah slot waktu: ";
    cin >> numSlots;

    // Input daftar mata kuliah yang diambil oleh setiap mahasiswa
    vector<vector<int>> studentCourses(numStudents);
    for (int i = 0; i < numStudents; ++i) {
        int numTaken;
        cout << "Masukkan jumlah mata kuliah yang diambil oleh mahasiswa ke-" << i + 1 << ": ";
        cin >> numTaken;

        cout << "Masukkan indeks mata kuliah (0-" << numCourses - 1 << ") yang diambil:\n";
        for (int j = 0; j < numTaken; ++j) {
            int course;
            cin >> course;
            studentCourses[i].push_back(course);
        }
    }

    // Parameter Algoritma Genetika
    const int POP_SIZE = 20; // Ukuran populasi
    const int MAX_GEN = 100; // Generasi maksimum

    // Inisialisasi populasi
    vector<Schedule> population(POP_SIZE);
    for (int i = 0; i < POP_SIZE; ++i) {
        population[i] = initializeSchedule(numCourses, numSlots, studentCourses);
    }
```

```
    // Proses evolusi
    for (int gen = 0; gen < MAX_GEN; ++gen) {
        // Sortir populasi berdasarkan jumlah konflik
        sort(population.begin(), population.end(), [](const Schedule& a, const Schedule& b) {
            return a.conflicts < b.conflicts;
        });

        // Cetak konflik terbaik
        cout << "Generasi " << gen + 1 << ": Konflik terbaik = " << population[0].conflicts << endl;

        // Jika tidak ada konflik, berhenti
        if (population[0].conflicts == 0) {
            cout << "Solusi optimal ditemukan pada generasi " << gen + 1 << endl;
            break;
        }

        // Seleksi dan crossover
        vector<Schedule> newPopulation;
        for (int i = 0; i < POP_SIZE / 2; ++i) {
            Schedule child = crossover(population[i], population[i + 1], studentCourses);
            mutate(child, numSlots, studentCourses);
            newPopulation.push_back(child);
        }

        // Tambahkan individu terbaik dari generasi sebelumnya
        while (newPopulation.size() < POP_SIZE) {
            newPopulation.push_back(population[rand() % (POP_SIZE / 2)]);
        }
        population = newPopulation;
    }

    // Cetak jadwal terbaik
    cout << "Jadwal terbaik:" << endl;
    for (int i = 0; i < numCourses; ++i) {
        cout << "Mata Kuliah " << i << ": Slot " << population[0].slots[i] << endl;
    }

```

Adapun cara penggunaan diatas, kita perlu memasukan jumlah mata kuliah, jumlah mahasiswa, dan juga kita perlu memasukan berapa jumlah waktu yang akan kita gunakan, lalu setelah keti inputan tadi di isi, kita akan diminta untuk memasukan mata kuliah yang akan di ambil oleh satu maha siswa, dan itu berurut dari index 0 nya mahasiswa, setelah menginputkannya kita akan mendaoatkan hasil yang memnungkinkan agar setiapo mahasiswa memiliki waktu yang terjawal unruk setiao matkulnya

```
--interpreter=mi' ;1a9deb46-27d7-4b23-819f-3d9473750f57Masukkan jumlah mata kuliah: 3
Masukkan jumlah mahasiswa: 3
Masukkan jumlah slot waktu: 3
Masukkan jumlah mata kuliah yang diambil oleh mahasiswa ke-1: 2
Masukkan indeks mata kuliah (0-2) yang diambil:
0
1
Masukkan jumlah mata kuliah yang diambil oleh mahasiswa ke-2: 1
Masukkan indeks mata kuliah (0-2) yang diambil:
1
Masukkan jumlah mata kuliah yang diambil oleh mahasiswa ke-3: 3
Masukkan indeks mata kuliah (0-2) yang diambil:
2
3
1
Generasi 1: Konflik terbaik = 0
Solusi optimal ditemukan pada generasi 1
Jadwal terbaik:
Mata Kuliah 0: Slot 0
Mata Kuliah 1: Slot 1
Mata Kuliah 2: Slot 2
```

G. Kesimpulan

Tugas diatas kita dapat mengambil kesimpulan, bahawasanya membuat algoritma dapat sangan membantu kita untuk menyelesaikan sebuah kasus yang akan rumit apabila kita membuatnbya secara asal asalan

H. Daftar Pustaka

- https://kazokku.com/blog/2024/06/11/proyek-it-hkn/#:~:text=Jun%2011%2C%202024-,Proyek%20IT%20yang%20Sebaiknya%20Menggunakan%20Solusi%20IT%20Manpower%20KAZOKKU,manpower%20yang%20bisa%20Anda%20manfaatkan.