Tugas Kecil IF2121 Strategi Algoritma

Penyusunan Rencana Kuliah dengan Topological Sort (Penerapan Decrease and Conquer)

Nama: Muhammad Zubair

NIM: 13519172

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2021

Algoritma topological sort:

Melakukan iterasi sampai graf kosong dengan mencari mata kuliah yang bisa diambil pada semester tersebut lalu menghilangkannya dari graf sehingga persoalan direduksi sebesar banyaknya mata kuliah yang tidak ada prequisitenya per iterasi sehingga variannya pada decrease and conquer adalah decrease by a variable size. Kemudian dilakukan terus sampai graf kosong.

Source Code:

#include <iostream>

#include <fstream>

#include <bits/stdc++.h>

class matkul;

class prerequisite;

// struktur data linked list bernama prequisite

class prerequisite {

private:

    matkul \*g;

    prerequisite \*next;

public:

    // class matkul dapat mengakses private variable dari class prequisite

    friend class matkul;

    prerequisite(matkul \*\_g): g(\_g), next(nullptr) {}

    prerequisite(matkul \*\_g, prerequisite \*\_next): g(\_g), next(\_next) {}

    ~prerequisite() {}

};

// struktur data linked list bernama matkul

// yang tersambung ke prequisite dari matkul tersebut dan matkul lain.

class matkul {

private:

    std::string name;   // nama matkul

    int count\_prereq;   // banyaknya prequisite

    prerequisite \*prereqs;  // menyambungkan ke prequsite dari matkul

    matkul \*next;   // menyambungkan ke matkul lain

public:

    // class graph dapat mengakses private variable dari class matkul

    friend class graph;

    matkul(std::string \_name) {

        name = \_name;

        count\_prereq = 0;

        prereqs = nullptr;

        next = nullptr;

    }

    // jika matkul didelete maka prequisitenya juga didelete

    ~matkul() {

        prerequisite \*node = prereqs;

        while (prereqs) {

            node = prereqs;

            prereqs = node->next;

            delete node;

        }

    }

    // menambahkan prequisite ke dalam matkul

    void add\_prereq(matkul\* g) {

        if (!prereqs) {

            prerequisite \*node = new prerequisite(g);

            prereqs = node;

        } else {

            prerequisite \*new\_neighbor = new prerequisite(g, prereqs);

            prereqs = new\_neighbor;

        }

        count\_prereq++;

    }

    // menghapus prequisite dari matkul jika ada

    // jika tidak ada prequisitenya maka tidak melakukan apa-apa

    bool del\_prereq(matkul\* g) {

        if (count\_prereq > 0) {

            // jika ada prequisitenya

            if (prereqs->g == g) {  // elmt >= 1, delete first

                prerequisite \*node = prereqs;

                prereqs = prereqs->next;

                delete node;

                count\_prereq--;

                return true;

            } else {

                prerequisite \*node = prereqs, \*prev = nullptr;

                while (node && node->g != g) {

                    prev = node;

                    node = node->next;

                }

                if (node) {

                    prev->next = node->next;

                    delete node;

                    count\_prereq--;

                    return true;

                }

            }

        }

        return false;

    }

    // mengakses matkul selanjutnya yang tersambung

    matkul \*Next() { return next; }

    std::string get\_name() { return name; }

    int get\_num\_prereq() { return count\_prereq; }

};

class graph {

private:

    std::string name;   // nama mahasiswa

    matkul \*first;

public:

    graph(std::string \_name): name(\_name), first(nullptr) {}

    ~graph() {

        matkul \*node = nullptr;

        while (first) {

            node = first;

            delete node;

            first = first->next;

        }

    }

    // jika matkul yang belum diambil sudah habis atau belum

    bool empty() { return (first) ? false : true; }

    // menambahkan matkul ke dalam graph jika belum ditambahkan

    // jika sudah ditambahkan maka tidak melakukan apa-apa

    // dan mengembalikan address dari matkul yang dicari atau ditambahkan tersebut

    matkul \*add\_matkul(std::string name\_matkul) {

        // cek apakah sudah ditambahkan

        matkul \*g = check\_matkul(name\_matkul);

        if (g == nullptr) {

            // jika belum ditambahkan

            g = new matkul(name\_matkul);

            if (!first) {

                first = g;

            } else {

                g->next = first;

                first = g;

            }

        }

        return g;

    }

    // menghapus matkul beserta prequisitenya

    matkul \*del\_matkul(matkul \*matkul\_a) {

        if (first == matkul\_a) {

            first = matkul\_a->next;

            matkul \*node = first;

            while (node) {

                // menghapus prequisitenya

                node->del\_prereq(matkul\_a);

                node = node->next;

            }

            delete matkul\_a;

            return first;

        } else {

            matkul \*node = first, \*prev = nullptr;

            while (node && node != matkul\_a) {

                prev = node;

                node = node->next;

            }

            if (node) {

                prev->next = node->next;

                matkul \*next\_after\_del\_node = prev->next;

                matkul \*del\_preq = first;

                while (del\_preq) {

                    // menghapus prequisitenya

                    del\_preq->del\_prereq(node);

                    del\_preq = del\_preq->next;

                }

                delete node;

                return prev->next;

            }

            return nullptr;

        }

    }

    // mengecek apakah matkul sudah ditambahkan ke dalam graph

    matkul \*check\_matkul(std::string name\_matkul) {

        matkul \*node = first;

        while (node && node->name != name\_matkul) {

            node = node->next;

        }

        return node;

    }

    // mendapatkan address pertama matkul dalam graph

    matkul \*get\_first() { return first; }

    // mendapatkan nama graph

    std::string get\_name() { return name; }

};

int main() {

    std::string file;

    std::cin >> file;

    graph G\_Matkul("Rencana Kuliah Mahasiswa A");

    {   // baca file dan masukkan ke dalam graph

        std::string r\_line;

        file = "../test/" + file;

        std::ifstream r\_file(file);

        std::string name\_matkul;

        // baca file per baris dan masukkan ke variabel r\_line

        while (getline(r\_file, r\_line)) {

            int idx = 0;

            // iterasi dan cek per karakter pada r\_line

            // jika ada ',' atau '.' maka masukkan ke dalam graf nama matkul

            while (r\_line[idx] != ',' && r\_line[idx] != '.') {

                char& c = r\_line[idx++];

                // mengumpulkan nama matkul

                if (c != ' ')

                    name\_matkul.push\_back(c);

            }

            // masukkan nama matkul ke dalam graf jika belum ada

            // lalu simpan addressnya pada variabel g

            matkul \*g = G\_Matkul.add\_matkul(name\_matkul);

            name\_matkul.clear();

            // jika sudah tanda '.', baca baris selanjutnya

            if (r\_line[idx] == '.') continue;

            // karena index pada r\_line berada pada tanda ',' maka index tambah 1

            idx++;

            // iterasi kembali per karakter pada r\_line

            while (idx < r\_line.length()) {

                char &c = r\_line[idx++];

                // jika ada spasi, dilewati

                if (c == ' ') continue;

                // jika ada ',' atau '.' maka nama matkul dimasukkan ke dalam graph jika belum ada

                // dan masukkan kedalam prequisite matkul g.

                if (c == ',' || c == '.') {

                    // tambahkan ke dalam graf jika belum ada pada graph

                    // lalu simpan prequisitenya kedalam graf

                    matkul \*prereq = G\_Matkul.add\_matkul(name\_matkul);

                    // masukkan prequisitenya ke dalam matkul g

                    g->add\_prereq(prereq);

                    name\_matkul.clear();

                } else {

                    name\_matkul.push\_back(c);

                }

            }

        }

    }

    std::vector <std::vector <std::string>> semester;   // menyimpan kumpulan matkul per semester

    // iterasi sampai tidak ada matkul yang bisa diambil

    while (!G\_Matkul.empty()) {

        // iterasi mulai dari matkul awal

        // untuk cek apakah ada matkul yg bisa diambil pada semester tsb

        matkul \*node = G\_Matkul.get\_first();

        std::vector <std::string> semester\_a;   // menyimpan nama matkul

        std::vector <matkul\*> addr\_semester\_a;  // menyimpan address matkul

        // iterasi

        while (node) {

            // jika ada matkul yg bisa diambil

            if (node->get\_num\_prereq() == 0) {

                // dimasukkan ke dalam semester\_a dan addr\_semester\_a

                semester\_a.push\_back(node->get\_name());

                addr\_semester\_a.push\_back(node);

            }

            node = node->Next();

        }

        // delete matkul yang sudah diambil di dalam graph

        for (auto matkul\_a = addr\_semester\_a.begin(); matkul\_a != addr\_semester\_a.end(); matkul\_a++) {

            G\_Matkul.del\_matkul(\*matkul\_a);

        }

        // menyimpan kumpulan matkul yang bisa diambil pada semester tersebut

        semester.push\_back(semester\_a);

    }

    // Mencetak matkul-matkul yang bisa diambil per semester

    std::cout << '\n';

    std::cout << G\_Matkul.get\_name() << ": \n";

    for (int i = 0; i < semester.size(); i++) {

        std::cout << "Semester " << i+1 << ": ";

        int j = 0;

        while (j < semester[i].size()-1)

            std:: cout << semester[i][j++] << ", ";

        std::cout << semester[i][j] << '\n';

    }

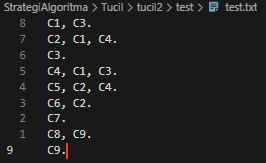
    std::cout << '\n';

    return 0;

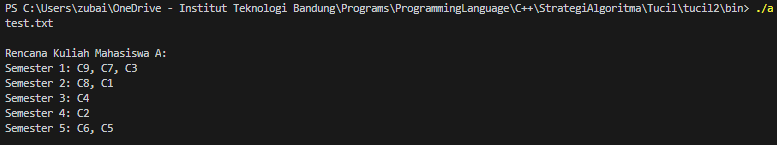
}

Screenshoot Input dan Output

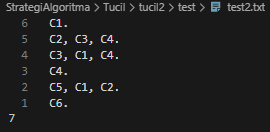
File test.txt



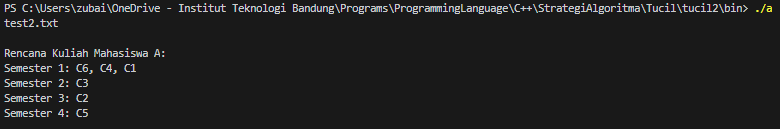
Hasil:



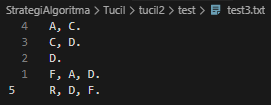
File test2.txt



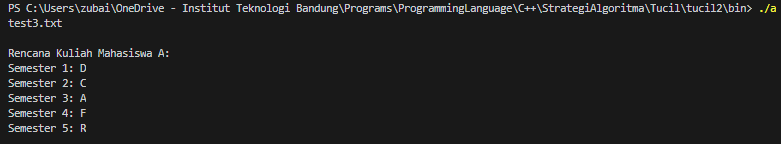
Hasil:



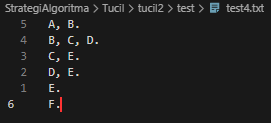
File test3.txt



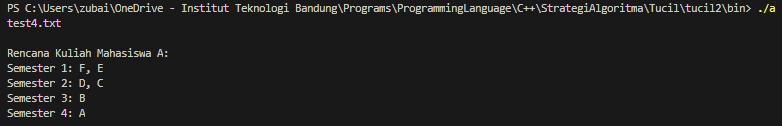
Hasil:



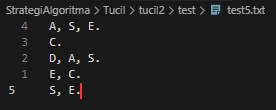
File test4.txt



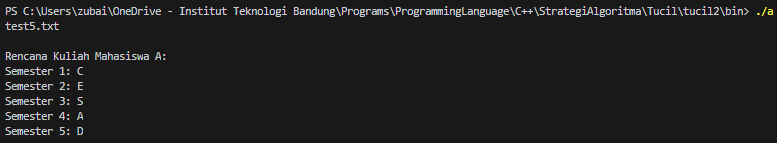
Hasil:



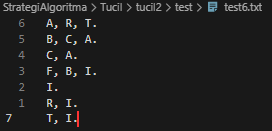
File test5.txt



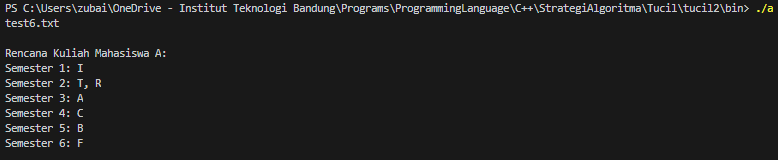
Hasil:



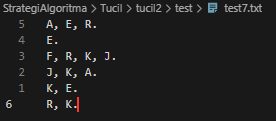
File test6.txt



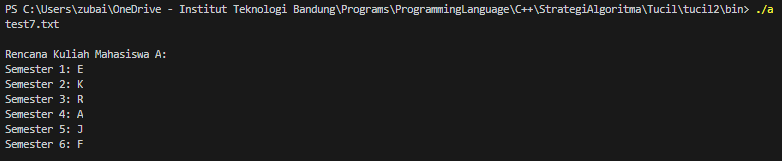
Hasil:



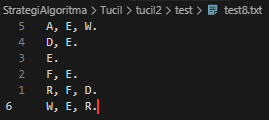
File test7.txt



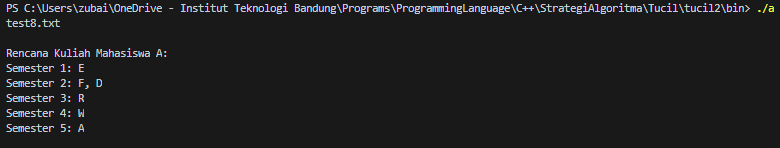
Hasil:



File test8.txt



Hasil:



Alamat tempat kode sumber program:

https://drive.google.com/drive/folders/1lBRpWLs2LzyMCCnikD3W1zpFIM\_Hiv\_K?usp=sharing

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi |  |  |
| 2. Program berhasil running |  |  |
| 3. Program dapat menerima berkas input dan menuliskan output. |  |  |
| 4. Luaran sudah benar untuk semua kasus input. |  |  |