LAPORAN HASIL TUGAS KECIL 2

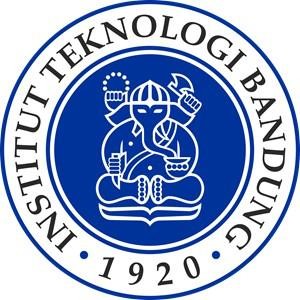
**Penyusunan Rencana Kuliah dengan *Topological Sort* “SUPREMO PLAN” dalam Bahasa *C++***

# DIAJUKAN UNTUK MEMENUHI TUGAS MATA KULIAH

**IF2211 – Strategi Algoritma SEMESTER II TAHUN 2020/2021**

**Disusun oleh : Karel Renaldi 13519180**

**K04**



**PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG 2021**

1. **Algoritma *Topological Sort***

## Buat representasi DAG(*Directed Acyclic Graph*) dan representasi untuk menyimpan nilai derajat masuk dari setiap simpul / *vertices*

* 1. Pilih semua simpul / *vertices* yang memiliki derajat masuk sebesar 0 lalu buang simpul-simpul tersebut dari graf.
  2. Lakukan perubahan derajat masuk kepada simpul / vertices dengan mengurangi derajat simpul-simpul tersebut yang berhubungan dengan simpul yang dibuang dari graf.
  3. Ulangi langkah (2) dan (3) hingga graf kosong, jika graf sudah kosong maka algoritma pengurutan sudah selesai.

Kaitan algoritma *topological sort* ini dengan pendekatan Decrease and Conquer adalah pada pengimplementasian *topological sort*, Algoritma yang saya buat ini adalah mereduksi tiap” simpul yang memiliki derajat masuk 0 di graf diulangi terus menerus sampai semua simpul sudah dikunjungi dan terurut.

1. ***Source Code* Program**

## Dibawah ini adalah kode program dari aplikasi “SUPREMO PLAN”:

*// Karel Renaldi / 13519180 / K-04 / Tucil Stima 2*

#include <bits/stdc++.h>

using namespace std;

*// Global variabel*

int semester = 0;

void split(*string* *const* *&*str, *const* char *\**delim, vector<*string*> *&*out)

{

    char \*token = strtok(const\_cast<char \*>(str.c\_str()), delim);

    while (token != nullptr)

    {

        out.push\_back(*string*(token));

        token = strtok(nullptr, delim);

    }

}

*string* convert\_roman\_numeral(int *&*number)

{

    int standard\_numeral[] = {1, 4, 5, 9, 10, 40, 50, 90, 100, 400, 500, 900, 1000};

*string* roman\_numeral[] = {"I", "IV", "V", "IX", "X", "XL", "L", "XC", "C", "CD", "D", "CM", "M"};

    int remainder = number, curr\_pointer = 12;*// roman\_numeral (arr size - 1)*

*string* res = "";

    while (remainder > 0)

    {

        int div\_result = remainder / standard\_numeral[curr\_pointer];

        while (div\_result--)

            res += roman\_numeral[curr\_pointer];

        remainder = remainder % standard\_numeral[curr\_pointer];

        curr\_pointer--;

    }

    return res;

}

void create\_graph(*string* filename, unordered\_map<*string*, int> *&*vertices\_degree\_in, unordered\_map<*string*, unordered\_set<*string*>> *&*graph)

{

*string* text;

*ifstream* file(filename);

    while (getline(file, text))

    {

        vector<*string*> parse\_text;

        unordered\_set<*string*> vertices;

        text.pop\_back();

        split(text, ", ", parse\_text);

        graph.insert({parse\_text[0], vertices});

        for (int i = 1; i < parse\_text.size(); i++)

            graph[parse\_text[0]].insert(parse\_text[i]);

    }

    for (auto adj : graph)

        vertices\_degree\_in[adj.first] = adj.second.size();

}

void topological\_sort(unordered\_map<*string*, int> *&*vertices\_degree\_in, unordered\_map<*string*, unordered\_set<*string*>> *&*graph)

{

    vector<*string*> erase\_key;

    semester++;

    cout << "Semester " << convert\_roman\_numeral(semester) << " : ";

    for (auto vertices = vertices\_degree\_in.begin(); vertices != vertices\_degree\_in.end(); vertices++)

    {

        if (vertices->second == 0)

        {

            erase\_key.push\_back(vertices->first);

            cout << vertices->first << ", ";

        }

    }

    cout << "\b\b"

         << " " << endl;

*// Decrease*

    for (*string* key : erase\_key)

    {

        vertices\_degree\_in.erase(key);

        for (auto adj : graph)

            if (adj.first != key && adj.second.count(key) != 0)

                vertices\_degree\_in[adj.first]--;

    }

*// Recursive*

    if (vertices\_degree\_in.size() != 0)

        topological\_sort(vertices\_degree\_in, graph);

}

int main()

{

    unordered\_map<*string*, int> vertices\_degree\_in;

    unordered\_map<*string*, unordered\_set<*string*>> graph;

*string* filename = "test1.txt", path = "../test/";

    cout << "============== WELCOME TO SUPREMO PLAN APPLICATION ==============" << endl

         << endl;

    cout << "Masukkan nama file: ";

    cin >> filename;

    cout << endl;

*// Parsing text to graph*

    create\_graph(path + filename, vertices\_degree\_in, graph);

*clock\_t* start = clock();

    double duration;

*// Implement topological sort algorithm*

    topological\_sort(vertices\_degree\_in, graph);

    duration = (clock() - start) / (double)CLOCKS\_PER\_SEC;

    cout << endl

         << "Time : " << duration << "s";

    return 0;

}

1. ***Screenshot Input* dan *Output***

|  |  |
| --- | --- |
| Input | Output |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Input | Output |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. **Alamat *Source Code* Program**

## Kode sumber program terdapat pada folder *src* dengan nama file tucil2.cpp. Terdapat file *executable code* pada folder bin. Untuk membuat kode program menjadi file *executable* gunakan perintah “g++ tucil2.cpp -o ../bin/tucil2”. Untuk menjalankan file *executable* tersebut perintah yang harus dijalankan adalah “cd ./src” lalu “./tucil2”.

Untuk *backup* terdapat pada link Github berikut ini:

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