Otomasi Skala Prioritas Tugas Kuliah dengan Memanfaatkan Pohon Keputusan

Muhammad Naufal Izza Fikry 13519088
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13519088@std.stei.itb.ac.id

***Abstrak*—Mahasiswa seringkali dihadapkan dengan banyak tugas kuliah dalam satu waktu tertentu. Tak jarang, menentukan skala prioritas dari tugas-tugas yang ada menjadi cukup membingungkan karena ada beberapa faktor yang perlu dipertimbangkan. Pertimbangan ini bisa menjadi sulit dan melelahkan, sehingga pengerjan tugas-tugas yang ada pun dapat tertunda. Oleh karena itu, penulis mencoba membuat suatu program yang dapat mengotomasi penentuan prioritas dari tugas-tugas yang ada sehingga mahasiswa dapat lebih fokus dalam menyelesaikan tugas kuliah tanpa terlalu memikirkan urutan pengerjaan dari tugas-tugas tersebut.**

***Kata Kunci*—Otomasi, Tugas Kuliah, Skala Prioritas, Pohon Keputusan.**

I. PENDAHULUAN

Ketika dihadapkan dengan banyak pilihan, seseorang memiliki kecenderungan untuk tidak melakukan pekerjaan apapun. Seorang Psikologis, Barry Schwartz, menyebut fenomena ini *choice paralysis*. *Choice/Decision Paralysys* terjadi saat kita dihadapkan dengan pilihan yang sulit untuk dibandingkan. Banyaknya pilihan membuat otak bekerja lebih keras untuk memilih pilihan mana yang lebih baik dibanding pilihan yang lain. Hal ini dapat menghambat penyelesaian suatu pekerjaan karena seseorang malah lebih berfokus menentukan pekerjaan yang harus dikerjakan daripada memulai pekerjaan tersebut.

Sebagai mahasiswa, tak jarang kami dihadapkan dengan beberapa tugas sekaligus dengan berbagai tingkat kesulitan dan durasi pengerjaan dalam satu waktu tertentu. Banyaknya tugas dan pertimbangan membuat *Decision Paralysis* memiliki kecenderungan yang lebih tinggi untuk terjadi. Oleh karena itu, penulis berinisiatif untuk membuat suatu program yang dapat mengotomasi pengambilan keputusan untuk menentukan tugas mana yang memiliki prioritas lebih tinggi dan perlu dikerjakan terlebih dahulu. Program ini dapat meminimalkan proses berpikir dalam menentukan tugas apa yang perlu diselesaikan, sehingga pengguna program ini dapat lebih fokus menyelesaikan tugas yang urutannya sudah diotomasi oleh program. Program ini tidak hanya dapat mempermudah pengambilan keputusan, tetapi juga dapat memberi gambaran struktur yang lebih baik tentang urutan tugas yang harus diselesaikan.

II. TEORI DASAR

A. Struktur Data Graf

Graf adalah sebuah struktur data yang tersusun atas simpul dan sisi, dimana sebuah sisi menghubungkan dua buah simpul. Secara formal, Graf (G) didefiniskan sebagai

$$G = (V, E)$$

dengan V adalah himpunan tak kosong dari simpul-simpul (*Vertices*) dan himpunan dari sisi yang menghubungkan sepasang simpul (*Edge*).

$$V = \{v_1, v_2, v_3, \dots, v_n\}$$

$$E = \{e_1, e_2, e_3, \dots, e_m\}$$

Berdasar ada atau tidaknya gelang atau sisi ganda, Graf dibagi menadi 2, yaitu:

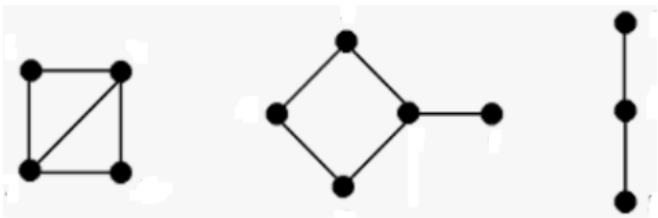
1. Graf sederhana
2. Graf tak-sederhana

Berdasar Orientasinya terhadap sisi, Graf juga dibagi menadi 2, yaitu:

1. Graf berarah
2. Graf tak-berarah

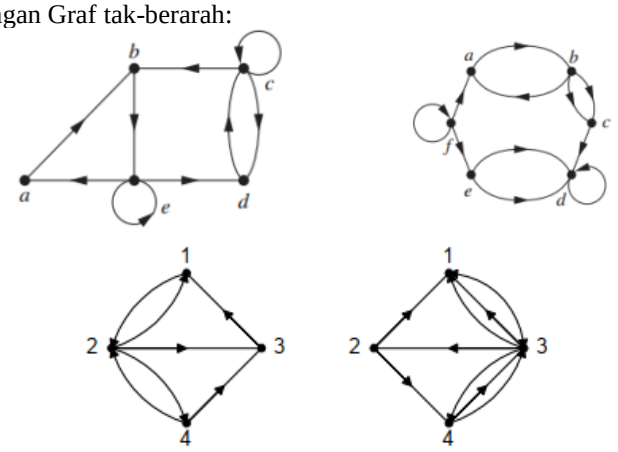
Pada makalah ini, penulis memanfaatkan struktur data pohon (akan dijelaskan kemudian) yang merupakan suatu Graf sederhana yang berarah.

Graf sederhana adalah Graf yang tidak mengandung gelang maupun sisi ganda. Dengan kata lain, hanya ada satu sisi yang menghubungkan sepasang simpul yang terhubung. Berikut adalah contoh graf sederhana:

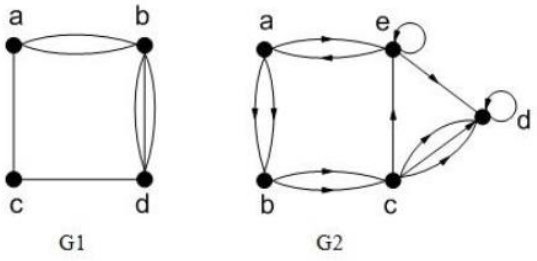


(Graf Sederhana - *sumber: Slide Kuliah Matematika Diskrit 2020/2021 oleh Dr. Rinaldi Munir)*

Graf berarah adalah graf yang sisinya memiliki orientasi arah. Berikut adalah contoh Graf berarah dan perbandingannya dengan Graf tak-berarah:



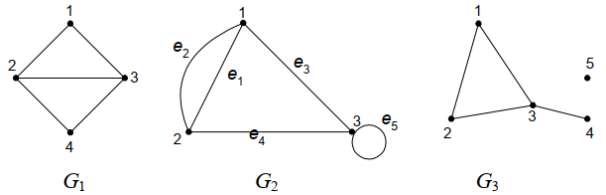
(Graf Berarah - *sumber: Slide Kuliah Matematika Diskrit 2020/2021 oleh Dr. Rinaldi Munir)*



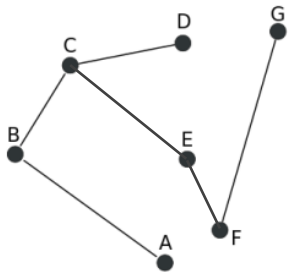
(*sumber: Slide Kuliah Matematika Diskrit 2020/2021 oleh Dr. Rinaldi Munir)*

B. Sirkuit dalam Graf

Sirkuit pada Graf adalah lintasan (rangkaian sisi) yang berawal dan berakhir dari simpul yang sama. Berikut adalah contoh perbandingan graf dengan dan tanpa sirkuit;



(Graf dengan sirkuit - *sumber: Slide Kuliah Matematika Diskrit 2020/2021 oleh Dr. Rinaldi Munir)*

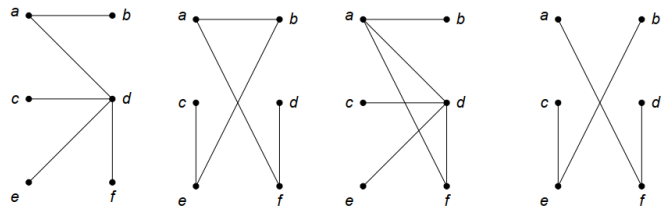


(Graf tanpa sirkuit - *sumber: Slide Kuliah Matematika Diskrit 2020/2021 oleh Dr. Rinaldi Munir, telah disesuaikan)*

Pohon merupakan salah contoh dari graf yang tidak memiliki sirkuit

C. Struktur Data Pohon

Pohon adalah Graf tak-berarah terhubung yang tidak mengandung sirkuit. Berikut adalah beberapa contoh dari Graf yang berupa pohon dan bukan berupa pohon:



(*sumber: Slide Kuliah Matematika Diskrit 2020/2021 oleh Dr. Rinaldi Munir*)

Karena pohon merupakan Graf, Pohon juga dapat didefinisikan sebagai

$$G = (V, E)$$

$$V = \{v_1, v_2, v_3, \dots, v_n\}$$

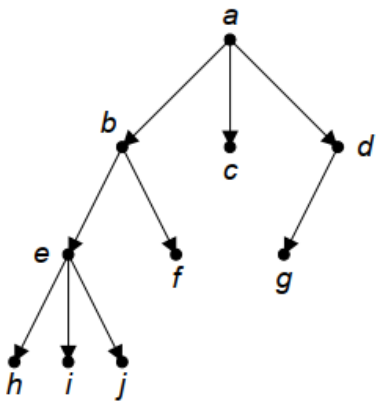
$$E = \{e_1, e_2, e_3, \dots, e_m\}$$

dengan $m = n - 1$.

Pohon dapat memiliki suatu akar, yakni simpul pada pohon yang menjadi awal percabagan simpul-simpul lainnya

D. Pohon Berakar

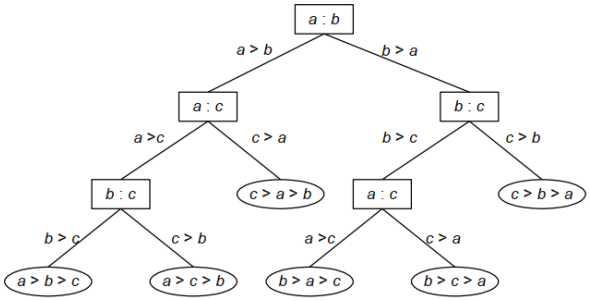
Pohon berakar adalah pohon yang salah satu simpulnya dijadikan akar dan sisi lainnya menjadi sisi berarah yang menjauhi akar. Berikut adalah contoh Pohon Berakar:



(Pohon berakar - *sumber: Slide Kuliah Matematika Diskrit 2020/2021 oleh Dr. Rinaldi Munir, telah disesuaikan)*

E. Pohon Keputusan

Pohon keputusan adalah suatu model berbentuk pohon yang digunakan untuk proses klasifikasi data-data masukan. Data yang telah diklasifikasi menjadi data yang terurut atau terkelompokkan berdasar spesifikasi Pohon Keputusan.



Gambar Pohon keputusan untuk mengurutkan 3 buah elemen

(*sumber: Slide Kuliah Matematika Diskrit 2020/2021 oleh Dr. Rinaldi Munir*)

Setiap simpul yang memiliki cabang dari pohon keputusan merupakan suatu hubungan fakta, dengan masing-masing cabangnya berupa kasus-kasus diterima dari hubungan tersebut.

III. Faktor-Faktor yang Menjadi Pertimbangan dalam Pembuatan Pohon Keputusan

Dalam menentukan faktor-faktor yang digunakan sebagai acuan pohon keputusan, penulis membuat program yang meminta data berupa tanggal-waktu tugas diberikan dan tanggal-waktu tenggat waktu pengumpulan. Penulis menggunakan asumsi bahwa semakin lama durasi pengerjaan suatu tugas kuliah, semakin cenderung tugas tersebut memerlukan waktu untuk memerlukan waktu yang lama dalam pengerjaannya. Durasi tugas inilah yang kemudian diolah menjadi parameter pertama yang penulis sebut dengan Skala Kepanikan.

Skala Kepanikan merupakan rentang nilai dari 1 – 4, dengan skala tertinggi adalah 1. Skala 1 adalah ketika waktu pengerjaan dari suatu tugas tersisa kurang dari 25%, serta Skala 2, 3, dan 4 berturut-turut terjadi ketika sisa waktu pengerjaan kurang dari 50%, kurang dari 75%, dan lebih dari atau sama dengan 75%. Pengklasifikasian berdasarkan waktu ini beralasan karena semakin sulit tugas kuliah, waktu yang diberikan untuk pengerjaannya cenderung lebih banyak.

Parameter kedua adalah tenggat waktu dari tugas tersebut. Parameter kedua digunakan apabila dalam pengurutan tugas menggunakan parameter pertama, nilai persentase yang dihasilkan sama. Tugas dengan tenggat waktu yang lebih awal akan mendapat prioritas yang lebih tinggi dalam pengurutan.

Perameter terakhir adalah id tugas yang selalu membesar setiap kali pengguna menambah data tugas baru. Tugas dengan Id lebih kecil (dibuat lebih dulu) akan mendapat prioritas yang lebih tinggi.

A. Skala Kepanikan

Skala kepanikan dihitung dengan membandingkan waktu yang tersisa dengan durasi pengerjaan tugas. Pengklasifikasian dilakukan dengan batas pada titik 25%, 50%, dan 75%. Berikut adalah potongan program yang menghasilkan nilai persentase dari skala kepanikan:

```
for tugas in List_tugas:
    # [Param1: panic rate(1-4, 1 is highest)]
    duration_in_sec = get_seconds_remaining(tugas.date_given, tugas.deadline)
    time_remaining_in_sec = get_seconds_remaining(dt_now, tugas.deadline)
    panic_rate = time_remaining_in_sec/duration_in_sec
```

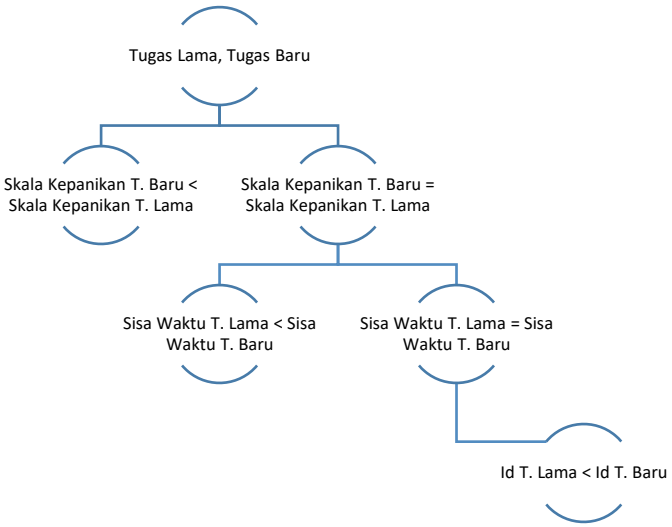
B. Sisa Waktu

Tugas dengan sisa waktu yang lebih banyak akan mendapatkan prioritas yang lebih tinggi. Parameter ini digunakan apabila perbandingan dengan parameter pertama mengasilkan nilai yang sama.

C. Id Tugas

Id Tugas selalu unik, dan selalu lebih besar dari tugas yang dibuat sebelumya. Dari fakta ini, nilai Id dapat digunakan untuk mengurutkan tugas berdasarkan waktu tugas tersebut ditambahkan ke dalam program.

Berikut adalah ilustrasi dari proses pengurutan tugas dengan menggunakan Pohon Keputusan.

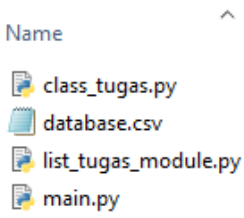


(*Ilustrasi Pohon Keputusan*)

Setelah melewati proses yang telah dirincikan dari Pohon Keputusan di atas terhadap setiap tugas, akan didapat suatu list yang berisi tugas yang telah diurutkan prioritasnya

IV. Organisasi Program

Program terdiri atas 4 bagian (file) yang meliputi main.py, class_tugas.py, list_tugas_module.py, dan database.csv. Keempat file tersebut terletak pada satu direktori yang sama.



(*sumber: Slide Kuliah Matematika Diskrit 2020/2021 oleh Dr. Rinaldi Munir*)

A. main.py

main.py adalah program utama. File ini tidak melakukan banyak hal selain melakukan load data, save data, dan memanggil fungsi main. File ini dibuat untuk memodularkan program sehingga mempermudah proses pengerjaan dan meningkatkan kualitas kode.

```python
from list_tugas_module import *

if __name__ == "__main__":
    load_tugas()
    main()
    print()
    save_tugas()
```

B. class_tugas.py

Berisi Class Tugas yang mendefinisikan Objek Tugas. Suatu Objek Tugas memiliki beberapa parameter, yaitu id, namaTugas, tanggalDiberikan, tenggatWaktu, isComplete, dan deskripsi. Setiap parameter itu akan diolah menjadi atribut Objek 'Tugas' yang merepresentasikan data tugas yang dibaca oleh program.

```python
from datetime import datetime

class Tugas():
    def __init__(self, id, namaTugas, tanggalDiberikan, tenggatWaktu, isComplete, deskripsi):
        self.id = int(id)                                   #int
        self.name = namaTugas                               #str
        self.date_given = datetime.strptime(str(tanggalDiberikan), "%A - %b %d %Y at %H:%M:%S")  #datetime
        self.deadline = datetime.strptime(str(tenggatWaktu), "%A - %b %d %Y at %H:%M:%S")        #datetime
        if(isComplete == "True"):
            self.completed = True                           #boolean
        else:
            self.completed = False                          #boolean
        self.description = deskripsi                        #str
```

C. list_tugas_module.py

Berisi fungsi-fungsi yang dibutuhkan oleh program, seperti save-load, sort, dsb. File ini memuat fungsi utama yang membuat program bekerja. Berikut adalah potongan fungsi sort_list_tugas() yang merupakan fungsi pengurutan utama dalam program ini.

```python
def sort_list_tugas():
    global List_tugas
    global List_tugas_sorted
    List_tugas_sorted = []
    dt_now = datetime.now()
    for tugas in List_tugas:
        # [Param1: panic rate(1-4, 1 is highest)]
        duration_in_sec = get_seconds_remaining(tugas.date_given, tugas.deadline)
        time_remaining_in_sec = get_seconds_remaining(dt_now, tugas.deadline)
        panic_rate = time_remaining_in_sec/duration_in_sec

        # [Param 2: time remaining]
        # |- already satified

        # [Param 3: id]
        # |- already satified
```

D. database.csv

Merupakan file basis data yang menyimpan informasi terhadap tugas-tugas yang diolah oleh program. Informasi yang disimpan adalah detail masing-masing tugas dengan baris pertama dari file merupakan keterangan untuk tiap-tiap kolomnya. Data yang disimpan meliputi id, namaTugas, tanggalDiberikan, tenggatWaktu, completed, dan deskripsi.

```
id, nama_tugas, waktu_diberikan, tenggat_waktu_pengumpulan, completed, deskripsi_tugas
1, Tugas Makalah Matdis, Wednesday - Dec 09 2020 at 09:02:00, Friday - Dec 11 2020 at 23:59:59, False, Membuat Makalah dengan memanfaatkan materi yang telah diberikan di kelas
2, tugas 2, Wednesday - Dec 09 2020 at 09:02:00, Friday - Dec 11 2020 at 23:59:59, False, None
```

V. CARA KERJA PROGRAM

Program memiliki 2 variabel global utama, yakni list of tugas yang terurut berdasar id, atau dengan kata lain, belum terurut berdasarkan prioritasnya, dan list of tugas yang terurut berdasarkan skala prioritas.

Mula-mula, program melakukan pemuatan data dari file 'database.csv'. data yang dimuat adalah Id Tugas, Nama Tugas, Waktu Tugas Diberikan, Tenggat Waktu Tugas, Indikator apakah Tugas Sudah Selesai atau Belum, dan Deskripsi Tugas.

Untuk setiap data tugas yang telah dibaca, dibuat suatu Objek 'Tugas' yang memuat keseluruhan data tersebut. Kemudian, Objek Tugas tersebut ditambahkan sebagai elemen baru 'list_tugas'.

Setelah data dimuat, program utama dijalankan. Tujuan utama dari program utama adalah membuat/mengisi 'list_tugas_sorted' yang merupakan list of Tugas yang elemennya sudah terurut berdasarkan skala prioritas.

Data yang sudah terurut ini kemudian ditampilkan dengan info tambahan lainnya.

```
PS E:\_data\listTugas> python3 main.py
SUCCESS! 'database.csv' HAS BEEN READ
FILE CLOSED
__ID
|-1   time passed 99.16% : Tugas Makalah Matdis
|-2   time passed 99.16% : tugas 2
```

VI. FUGNSI-FUNGSI

Berikut adalah fungsi-fungsi yang digunakan dalam program ini.

```
def get_date(DateTime): #(datetime) -> str
    return DateTime.strftime("%b %d %Y")

def get_day(DateTime): #(datetime) -> str
    return DateTime.strftime("%A")

def get_time_to_minute(DateTime):#(datetime) -> str
    h = str(DateTime.strftime("%H"))
    m = str(DateTime.strftime("%M"))
    return (h+":"+m)

def get_time_to_second(DateTime):#(datetime) -> str
    h = str(DateTime.strftime("%H"))
    m = str(DateTime.strftime("%M"))
    s = str(DateTime.strftime("%S"))
    return (h+":"+m+":"+s)


def get_seconds_remaining(dt_init, dt_final): #(datetime, datetime) -> int{time_difference}
    delta_dt = dt_final - dt_init
    delta_sec = delta_dt.total_seconds()
    return int(delta_sec)


def second_to_str_dd_hh_mm(time_in_sec): #(int{second}) -> str{`${dd} Day(s), ${hh} Hour(s), ${mm} Minute(s)`}
    remainder = int(time_in_sec)

    day = remainder//86400
    remainder %= 86400

    hour = remainder//3600
    remainder %= 3600

    minute = remainder//60

    return ("%d Day(s), %d Hour(s), %d Minute(s)" % (day, hour, minute))
```

```
def get_duration_in_dd_hh_mm(dt_init, dt_final): #(datetime,datetime) -> str
    delta_dt = dt_final - dt_init
    duration_in_sec = delta_dt.total_seconds()
    return second_to_str_dd_hh_mm(duration_in_sec)
```

```
def load_tugas():
    global List_tugas
    global File_directory
    global Filename
    path = File_directory + Filename

    f = False
    try:
        f = open(path, mode='r', encoding='utf-8')
        print("SUCCESS! '%s' HAS BEEN READ" % (Filename))
    except:
        print("|
        ========================================\n|-
        ERROR, Cannot read '%s'\n|-   try interpret using
        python3\n|
        ========================================\n" %
        (path))
        print("FILE HASN'T BEEN READ")


    if (f): #read success
        eof = False #eof: End of File

        #reading first line (data_info)
        s = f.readline()
        if (s==""):
            eof = True
            print("\n|- Uh-oh, the file '%s' is empty. -|
            \n" % (Filename))

        #reading the next line(s)
        while(not eof):
            s = f.readline()
            if (s==""):
                eof = True

            if (not eof):
                s = s[:-1].split(", ")

                T = Tugas(s[0], s[1], s[2], s[3], s[4], s
                [5])
                List_tugas.append(T)

        #closing file
        f.close()
        print("FILE CLOSED")
```

```python
def save_tugas():
    global List_tugas
    global File_directory
    global Filename
    path = File_directory + Filename
    with open(path, mode="w", encoding="utf-8") as f:
        #write description header
        f.write("id, nama_tugas, waktu_diberikan,
        tenggat_waktu_pengumpulan, completed,
        deskripsi_tugas\n")

        for tugas in List_tugas:
            fdate_given = get_day(tugas.date_given)+" - "
            +get_date(tugas.date_given)+" at "
            +get_time_to_second(tugas.date_given)
            fdeadline = get_day(tugas.deadline)+" - "
            +get_date(tugas.deadline)+" at "
            +get_time_to_second(tugas.deadline)
            if(tugas.completed):
                str_completed = "True"
            else:
                str_completed = "False"
            str_line =  ("%d, %s, %s, %s, %s, %s\n" %
            (tugas.id, tugas.name, fdate_given,
            fdeadline, tugas.completed, tugas.description)
            )
            f.write(str_line)
```

```python
def new_tugas():
    global List_tugas

    print("="*23 + "\n|- Tambah Tugas Baru -|\n" + "="*23)
    nama_tugas = str(input("Nama Tugas\t\t: "))
    waktu_diberikan = str(input("waktu_diberikan, harus
    sesuai contoh di bawah\n|- Wednesday - Dec 09 2020 at
    09:02:00\t: "))
    waktu_pengumpulan = str(input("waktu pengumpulan,
    harus sesuai contoh di bawah\n|- Friday - Dec 11 2020
    at 13:00:00\t: "))
    selesai = "False"
    deskripsi_tugas = str(input("deskripsi tugas\t\t: "))

    if (len(List_tugas) != 0):
        new_id = List_tugas[-1].id + 1
    else:
        new_id = 1
    T = Tugas(new_id, nama_tugas, waktu_diberikan,
    waktu_pengumpulan, selesai, deskripsi_tugas)
    List_tugas.append(T)

```

```python
def sort_list_tugas():
    global List_tugas
    global List_tugas_sorted
    List_tugas_sorted = []
    dt_now = datetime.now()
    for tugas in List_tugas:
        # [Param1: panic rate(1-4, 1 is highest)]
        duration_in_sec = get_seconds_remaining(tugas.
        date_given, tugas.deadline)
        time_remaining_in_sec = get_seconds_remaining
        (dt_now, tugas.deadline)
        panic_rate = time_remaining_in_sec/duration_in_sec

        # [Param 2: time remaining]
        # |- already satified

        # [Param 3: id]
        # |- already satified


        '''
        Hence, all requirement already satisfied.
        Next: using decision tree to fill
        List_tugas_sorted
        '''

        current_tuple_tugas = {"data":tugas,
        "panic_rate": panic_rate, "time_remaining" :
        time_remaining_in_sec, "id":tugas.id}
        if (len(List_tugas_sorted)==0):
            List_tugas_sorted.append(current_tuple_tugas)
        else:

            for i in range(len(List_tugas_sorted)):
                if(List_tugas_sorted[i]["panic_rate"] >
                current_tuple_tugas["panic_rate"]):
                    List_tugas_sorted.insert(i,
                    current_tuple_tugas)
                elif(List_tugas_sorted[i]["panic_rate"]
                == current_tuple_tugas["panic_rate"]):

                    if(List_tugas_sorted[i]
                    ["time_remaining"] >
                    current_tuple_tugas["time_remaining"])
                    :
                        List_tugas_sorted.insert(i,
                        current_tuple_tugas)
                    elif(List_tugas_sorted[i]
                    ["time_remaining"] ==
                    current_tuple_tugas["time_remaining"])
                    :

                        if(List_tugas_sorted[i]["id"] >
                        current_tuple_tugas["id"]):
                            List_tugas_sorted.insert(i,
                            current_tuple_tugas)

                if(i == len(List_tugas_sorted)-1): #jika
                sampai elemen terakhir
                    List_tugas_sorted.insert(i+1,
                    current_tuple_tugas)

```

```
def show_tugas_by_id():
    global List_tugas
    dt_now = datetime.now()
    for tugas in List_tugas:
        #recurrence setup
        str_date_given = get_day(tugas.date_given)+", "
        +get_date(tugas.date_given)
        str_deadline = get_day(tugas.deadline)+", "
        +get_date(tugas.deadline)+" at "
        +get_time_to_minute(tugas.deadline)
        waktu_pengerjaan = get_duration_in_dd_hh_mm(tugas.
        date_given,tugas.deadline)
        waktu_tersisa = get_duration_in_dd_hh_mm(dt_now,
        tugas.deadline)
        if (tugas.completed):
            isComplete = "True"
        else:
            isComplete = "False"

        #printing info
        print("|-\nid\t\t\t: %d\nname\t\t\t: %s\ndate
        given\t\t: %s\ndeadline\t\t: %s\nWAKTU
        PENGERJAAN\t: %s\nWAKTU TERSISA\t\t: %s\nSelesai?
        \t\t: %s\nDeskripsi Tugas\t\t: %s\n" % (tugas.id,
        tugas.name, str_date_given, str_deadline,
        waktu_pengerjaan, waktu_tersisa, isComplete,
        tugas.description))

def show_sorted_tugas():
    global List_tugas_sorted
    if (len(List_tugas_sorted) == 0):
        sort_list_tugas()
    num = 0
    first = True
    for sorted_tugas in List_tugas_sorted:
        if (first):
            print("__ID")
            first = False
        num += 1
        print("|-%d   time passed %.2f%% : %s" %
        (sorted_tugas["id"], ((1-sorted_tugas
        ["panic_rate"])*100),sorted_tugas["data"].name))

def main():
    global List_tugas
    show_sorted_tugas()
```

VII. KESIMPULAN

Banyaknya tugas yang diterima mahasiswa pada satu waktu tertentu dapat mengakibatkan fenomena *Decision Paralysys*. Penulis mencoba mengurangi dampak masalah ini dengan membuat suatu program yang memanfaatkan materi pada mata kuliah Matematika Diskrit, yakni Pohon Keputusan, untuk mengotomasi proses penentuan skala prioritas dari tugas-tugas kuliah mahasiswa. Program ini dapat membantu pengguna dalam menentukan urutan tugas kuliah, juga untuk memberi struktur yang lebih jelas terhadap linimasa tugas yang perlu dikerjakan. Harapannya, program ini dapat mempermudah pengguna dalam mengerjakan tugas kuliah.

VIII. UCAPAN TERIMAKASIH

Puji Syukur kepada Allah S.W.T. yang atas rahmatnya, penulis dapat menyelesaikan tugas makalah dengan judul "Otomasi Skala Prioritas Tugas Kuliah dengan memanfaatkan Pohon Keputusan" ini. Tak lupa juga ucapan terimakasih penulis sampaikan kepada bapak ibu dosen yang telah membantu penulis memahami materi perkuliahan sehingga tugas makalah ini dapat dibuat. Juga pada pihak-pihak lain yang baik secara langsung maupun tidak langsung membantu penulis memahami berbagai persoalan dan menyelesaikan tugas ini.

REFERENSI

[1] https://www.cs.yale.edu/homes/aspnes/pinewiki/GraphTheory.html diakses pada 11 Desember 2020, Pukul 20.36 WIB.
[2] https://www.thehedgescompany.com/the-surprising-poverty-of-too-many-choices/ diakses pada 11 Desember 2020, Pukul 20.07 WIB.
[3] https://procrastination.com/blog/9/decision-paralysis-overthinking-choices diakses pada 11 Desember 2020, Pukul 20.07 WIB.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Desember 2020

Ttd (scan atau foto ttd)

Muhammad Naufal Izza Fikry (13519088)