Penerapan *Critical Path Method* (CPM) Menggunakan Graf Berarah untuk Optimalisasi Rencana Belajar Mandiri Berdasarkan Skala Prioritas dan *Deadline* Tugas

Implementation of Critical Path Method (CPM) Using Directed Graphs to Optimize Self-Directed Learning Plans Based on Priority Scales and Task Deadlines

Vinsensius Juan Setiady - 13525093
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: v.juansetiady@gmail.com , 13525093@std.stei.itb.ac.id

***Abstract*—Di tengah jadwal perkuliahan yang padat, manajemen waktu menjadi kemampuan yang penting untuk dikuasai. Metode *Critical Path Method* berbagis Python dapat diterapkan untuk memodelkan penjadwalan pengerjaan tugas mahasiswa. Pengujian dilakukan dengan dua skenario. Skenario pertama menerapkan variabel *release time* dan *deadline* sebagai pembatas, sedangkan skenario kedua menerapkan model murni tanpa kedua batasan eksternal tersebut. Hasil penelitian menunjukkan bahwa skenario kedua menghasilkan jalur kritis pengerjaan tugas yang lebih panjang dibandingkan skenario pertama yang hanya mengandung satu tugas dalam jalur kritis. Secara praktis, skenario kedua lebih aplikatif karena data tugas dapat diperbarui secara dinamis (*dynamic scheduling*) sehingga sistem penjadwalan dapat berubah secara fleksibel setiap kali tugas baru diinput dan akibatnya lebih mudah untuk diterapkan dalam kehidupan sehari-hari.**

Kata kunci—Critical Path Method, Graf Berarah

I. PENDAHULUAN

Di dalam lingkungan perguruan tinggi, mahasiswa dihadapkan dengan tantangan akademis yang berat, mulai dari beban tugas yang kontinu, jadwal ujian yang padat, kegiatan organisasi, dan lain-lain. Mahasiswa dituntut untuk beradaptasi dengan cepat, mandiri dalam manajemen waktu untuk menyelesaikan berbagai tugas pribadi maupun kelompok. Namun, mahasiswa yang tidak dibekali dengan kemampuan manajemen waktu yang baik berpotensi menimbulkan masalah akademik yang berkelanjutan dan bahkan dapat memengaruhi kehidupan mahasiswa secara langsung, seperti fenomena *procrastination* (penundaan pengerjaan).

Metode penjadwalan yang dilakukan mahasiswa secara mandiri acap kali menimbulkan pengaturan jadwal yang tidak efisien karena tidak melibatkan analisis sistematis untuk menentukan prioritas tugas-tugas tertentu, sehingga penjadwalan dilakukan secara subjektif. Akibatnya, mahasiswa sering kali salah dalam menentukan apa yang harus didahulukan untuk menyelesaikan semua tugas yang perlu dikerjakan.

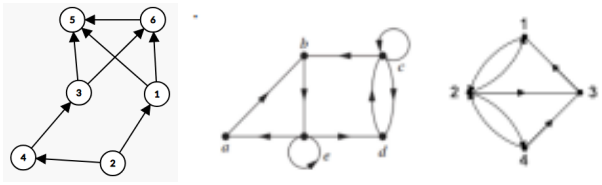
Untuk menghadapi masalah tersebut, diperlukan sebuah solusi yang objektif dan sistematis. Permasalahan ini dapat diselesaikan dengan Teori Graf pada Matematika Diskrit, khususnya Graf Berarah. Dalam studi ini, setiap jaringan tugas dimodelkan dengan Graf Berarah Berbobot dan manajemen waktu dilakukan dengan metode *Critical Path Method.* Penelitian ini berguna untuk mencari Jalur Kritis (*Critical Path*) serta nilai slack dari setiap tugas akademik pada graf untuk mencari skala prioritas yang paling efisien agar setiap tugas dapat diselesaikan dengan baik dan tentunya sesuai dengan tenggat waktu.

II. LANDASAN TEORI

A. Graf

Graf adalah sebuah struktur data yang merepresentasikan hubungan antarobjek diskrit. Graf, secara matematis, dilambangkan dalam bentuk G = (V,E), di mana V adalah sebuah himpunan *vertex* atau simpul dan E adalah sebuah himpunan *edge* atau sisi. Setiap sisi menghubungkan satu simpul (jika membentuk gelang) atau dua simpul, kemudian dapat ditulis dalam bentuk $e_1 = (v_1, v_2)$ di mana v_1 dan v_2 merupakan anggota V.

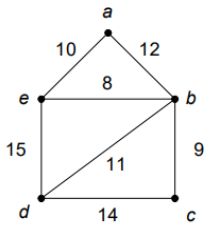
Penelitian ini secara khusus menggunakan Graf Berarah (*Directed Graph/Digraph*) yang memiliki arah pada setiap sisinya sehingga hubungan kedua simpul hanya berlaku satu arah sesuai dengan arah panah yang terdapat pada sisi tersebut.



(a) (b)

Gambar 2.1. Graf Berarah (*Directed Graph/Digraph*)
Sumber: (b) Graf Bagian I (oleh Prof. Dr. Ir. Rinaldi, M.T.)

Graf tersebut kemudian diberi bobot pada setiap sisi atau *edge* sehingga menjadi Graf Berbobot.



Gambar 2.2. Graf Berbobot (*Weighted Graph*)
Sumber: Graf Bagian I (oleh Prof. Dr. Ir. Rinaldi, M.T.)

Terdapat beberapa terminologi dalam Teori Graf yang mewakili identitas dari suatu graf, yaitu:

1. Bersisian (*Incidency*)

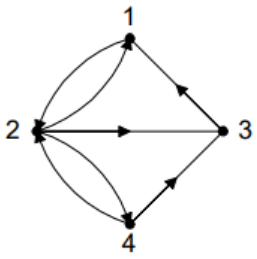
   Jika kedua simpul sembarang v_1 dan v_2 terhubung melalui sisi sembarang e, dapat dikatakan bahwa sisi e bersisian dengan simpul v_1 atau sisi e bersisian dengan simpul v_2.

2. Ketetanggaan (*Adjacent*)

   Jika kedua simpul sembarang v_1 dan v_2 terhubung melalui sisi sembarang e, dapat dikatakan bahwa simpul v_1 bertetangga dengan simpul v_2.

3. Derajat (*Degree*)

   Suatu simpul memiliki derajat yang menunjukkan jumlah sisi yang bersisian dengan simpul tersebut, disimbolkan dengan d(v), di mana v adalah *vertex* atau simpul. Pada Graf Berarah, derajat dibagi menjadi derajat masuk (*in-degree,* disimbolkan dengan $d_{in}(v)$) dan derajat keluar (*out-degree*, disimbolkan dengan $d_{out}(v)$).



Gambar 2.3. Graf Berarah
Sumber: Graf Bagian I (oleh Prof. Dr. Ir. Rinaldi, M.T.)

Berdasarkan gambar graf tersebut, dapat diketahui bahwa:

$d_{in}(1) = 2$; $d_{out}(1) = 1$

$d_{in}(2) = 2$; $d_{out}(2) = 3$

$d_{in}(3) = 2$; $d_{out}(3) = 1$

$d_{in}(4) = 1$; $d_{out}(4) = 2$

4. Lintasan (*Path*)

   Secara matematis, lintasan pada graf sembarang G adalah sebuah barisan berselang-seling simpul dan sisi yang panjangnya n dari simpul awal v_0 hingga simpul akhir v_n berbentuk v_0, e_1, v_1, e_2, …, v_{n-1}, e_n, v_n sedemikian sehingga $e_1 = (v_0,v_1)$, $e_1 = (v_1,v_2)$, …, $e_n = (v_{n-1},v_n)$ adalah sisi-sisi dari G. Secara praktis, lintasan pada sebuah graf adalah barisan simpul yang sekuensial di mana setiap simpul yang berdampingan dihubungkan oleh sebuah sisi, dengan syarat tidak ada simpul atau sisi yang didatangi lebih dari satu kali.

5. Sirkuit (*circuit*)

   Sirkuit pada graf adalah lintasan yang memiliki simpul pertama dan simpul terakhir yang sama, sehingga akan menghasilkan sebuah *loop*.

Graf yang digunakan pada metode *Critical Path Method* wajib memenuhi karakteristik Graf Berarah Asiklik (*Acyclic Directed Graph*). Karakteristik "berarah" karena setiap tugas memiliki hubungan ketergantungan dengan tugas lainnya, di mana suatu tugas hanya dapat dikerjakan jika tugas prasyaratnya telah selesai dikerjakan. Sementara itu, karakteristik "asiklik" karena untuk mencegah terjadinya kondisi *infinite loop* dalam implementasi *forward pass* dan *backward pass* sehingga menyebabkan total durasi pengerjaan tugas tidak dapat didefinisikan.

B. Pengurutan Topologis (Topological Sort)

Topological Sort adalah pengurutan linear dari setiap simpul pada graf bertipe Graf Berarah Asiklik (*Directed Acyclic Graph*) untuk menentukan urutan simpul pada suatu graf. Jika terdapat sebuah sisi berarah e yang menghubungkan simpul u ke simpul v (u -> v), maka simpul u harus muncul terlebih dahulu sebelum simpul v.

Implementasi *topological sort* dapat dilakukan dengan beberapa algoritma yang relevan, pada umumnya adalah algoritma *Breadth First Search* (BFS) dan *Depth First Search* (DFS). Perbedaan mendasar antara keduanya adalah arah eksplorasi simpul. BFS mengecek semua tetangga terdekat sebelum menelusuri level berikutnya, sedangkan DFS menelusuri simpul hingga level terdalam sebelum melakukan proses *backtracking* untuk memeriksa cabang lain.

Pada penelitian ini, algoritma yang digunakan untuk implementasi *topological sort* adalah *Depth First Search* Pemilihan tersebut didasarkan pada struktur algoritma yang lebih intuitif dengan memanfaatkan fungsi rekursif dan kemudahan implementasi pada kode program komputer

dengan menggunakan struktur data *stack* untuk menghasilkan urutan linear secara langsung.

C. Algoritma Critical Path Method (CPM)

Critical Path Method adalah sebuah algoritma untuk mencari waktu penyelesaian (diambil dari durasi pengerjaan tugas) terpanjang dari sebuah rangkaian graf. Beberapa komponen dari algoritma CPM adalah:

1. Durasi: jumlah waktu yang diperlukan untuk mengerjakan sebuah tugas
2. *Early Start (ES)*: waktu tercepat suatu tugas dapat mulai dikerjakan.
3. *Early Finish (EF)*: waktu tercepat suatu tugas dapat selesai dikerjakan, dihitung dengan rumus EF = ES + Durasi
4. *Late Finish (LF)*: waktu paling lambat suatu tugas dapat selesai dikerjakan
5. *Late Start (LS)*: waktu paling lambat suatu tugas dapat mulai dikerjakan, dihitung dengan rumus LS = LF - Durasi
6. *Slack Time*: batas toleransi waktu penundaan dalam mengerjakan suatu tugas. Slack Time dihitung dengan cara mengurangi LS dengan ES atau LF dengan EF
7. *Jalur Kritis (Critical Path)*: lintasan sekuensial dari simpul awal hingga simpul akhir yang terdiri dari rangkaian tugas dengan nilai Slack sama dengan nol (Slack Time = 0). Hal ini mengindikasikan bahwa tugas yang berada di dalam jalur kritis harus dikerjakan tepat waktu tanpa ada toleransi penundaan agar semua tugas dapat selesai tepat waktu.

Untuk menentukan ES dan LF, perlu digunakan aturan *forward pass* (penyisiran maju) dan *backward pass* (penyisiran mundur) dengan aturan dan modifikasi sebagai berikut:

1. Untuk ES (*forward pass*): jika satu tugas memiliki lebih dari satu tugas prasyarat, nilai ES adalah nilai maksimum dari nilai EF seluruh tugas prasyarat. Secara matematis dapat ditulis:

$$ES = max(max(EF_{prasyarat}), release\ time) \quad (1)$$

2. Untuk LF (*backward pass*): jika satu tugas menjadi tugas prasyarat dari beberapa tugas, nilai LF adalah nilai minimum dari nilai LS seluruh tugas setelahnya. Secara matematis dapat ditulis:

$$LF = min(min(LS_{tugas\ setelahnya}), deadline) \quad (2)$$

Rumus matematis untuk menentukan ES dan LF terdapat tambahan variabel *release_time* dan *deadline* untuk membuat model menjadi lebih realistis dengan jadwal yang sebenarnya terjadi di dalam perkuliahan. Variabel *release_time* adalah waktu tugas dirilis, relatif dari hari ke-0, serta variabel *deadline* adalah batas waktu pengerjaan tugas, relatif dari hari ke-0.

Dalam implementasi graf, terdapat dua pendekatan dalam merepresentasikan sebuah tugas dan bobotnya, yaitu *Activity-on-Node* (AoN) dan *Activity-on-Edge* (AoE). Perbedaan inti dari kedua pendekatan tersebut adalah tempat meletakkan tugas dan bobotnya secara visual, meskipun kedua pendekatan tersebut menghasilkan output *critical path* dan total durasi yang sama. Penelitian ini menggunakan pendekatan AoN karena merupakan representasi murni dari Graf Berarah Asiklik dan memiliki kemudahan serta efisiensi untuk diimplementasikan ke dalam kode program komputer.

Langkah-langkah algoritma *Critical Path Method*:

1. Buat sebuah graf dengan simpul dari graf tersebut adalah tugas yang menyimpan variabel identitas: Nama, Durasi, ES, EF, LS, LF, dan Slack
2. Hubungkan setiap tugas tersebut dengan sisi berarah sesuai dengan urutan tugas prasyarat dan tugas lanjutan
3. Lakukan pengurutan topologis pada seluruh simpul untuk memastikan bahwa tidak ada tugas lanjutan yang diproses sebelum tugas prasyarat selesai diproses.
4. Baca simpul dari simpul awal setelah dilakukan pengurutan topologis hingga ke simpul terakhir dengan metode *forward pass*.
5. Untuk setiap simpul yang diperiksa, tentukan nilai ES dengan aturan:
    a. Jika simpul tersebut merupakan simpul awal, nilai ES = 0
    b. Jika simpul memiliki satu atau lebih simpul prasyarat, nilai ES simpul tersebut diambil dari nilai EF maksimum dari seluruh simpul prasyarat.
6. Hitung nilai EF dari simpul aktif saat ini dengan rumus:

$$EF = ES + Durasi \quad (3)$$

7. Ulangi langkah 5 dan 6 hingga simpul terakhir dari graf selesai diproses.
8. Nilai durasi total pengerjaan tugas didapat dari nilai EF pada simpul terakhir
9. Untuk mencari nilai LS dan LF, baca simpul dari simpul terakhir hingga simpul awal dengan metode *backward pass*
10. Untuk setiap simpul yang diperiksa, tentukan nilai LF dengan aturan:
    a. Jika simpul tersebut merupakan simpul akhir, nilai LF = nilai durasi total
    b. Jika simpul tersebut menjadi prasyarat dari simpul-simpul setelahnya, nilai LF adalah nilai minimum dari seluruh nilai LS simpul setelahnya
11. Hitung nilai LS dari simpul aktif saat ini dengan rumus:

$$LS = LF - Durasi \quad (4)$$

12. Ulangi langkah 10 dan 11 hingga simpul awal selesai diproses
13. Hitung nilai *Slack Time* dari seluruh simpul dengan rumus:

$$Slack = LS - ES\ atau\ Slack = LF - EF \quad (5)$$

14. Tandai simpul dengan nilai Slack = 0 untuk dimasukkan ke dalam *critical path*.

15. *Critical Path* dari graf adalah lintasan pada graf dengan nilai Slack pada simpul sama dengan nol.

III. METODE PENELITIAN

A. *Pengumpulan Data Tugas*

Data seluruh tugas akademik semester II disimpan di dalam satu file berekstensi *comma separated value* (.csv). Penggunaan file ini bertujuan untuk efisiensi skalabilitas sistem agar data dapat dengan mudah diubah, dikurangi, maupun ditambah tanpa memengaruhi kode program utama. Pada fase ini, ditambahkan *dummy nodes* berupa START dan FINISH dengan durasi 0 hari sebagai simpul awal dan akhir pada graf.

	A	B	C	D	E	F	G
1	id	mata_kuliah	nama_tugas	durasi_jam	prasyarat	deadline_hari	release_time_hari
2	START	Umum	Simpul Awal	0		0	0
3	T01	Alpro	Alpro Minggu 1	4	START	4	0
4	T02	Alpro	Alpro Minggu 2	4	T01	7	0
5	T03	Alpro	Kuis Alpro	2	T02	8	7
6	T04	Lokom	Logika Proposisi	2	START	9	8
7	T05	Alpro	Skema Pengulangan	3	T02;T03	11	8
8	T06	Lidia	Kuis 2	1	START	22	10
9	T07	Matdis	Kuis 3 bab	2	START	13	11
10	T08	Orkom	Latihan Assembly	3	START	17	12
11	T09	Orkom	Praktikum	5	T02;T08	20	13
12	T10	Matdis	Kuis Induksi, Boolean	2	T07	24	21
13	T11	Matdis	Makalah Bab 1	5	T10	30	25
14	T12	Matdis	Makalah Bab 2	5	T11	35	30
15	T13	Alpro	Tugas Besar 1	30	T05;T12	39	35
16	T14	Alpro	Tugas Besar 2	40	T13	48	45
17	T15	Lokom	Praktikum	20	T04;T13;T14	51	50
18	FINISH	Umum	Simpul Akhir	0	T06;T09;T12;T14;	53	0

Gambar 3.1. Data Tugas Setiap Mata Kuliah

Penjelasan variabel di dalam data tersebut adalah:

1. id: menunjukkan identitas tugas secara singkat
2. mata_kuliah: menunjukkan nama mata kuliah
3. nama_tugas: menunjukkan nama tugas sesuai dengan mata kuliah
4. durasi_jam: durasi pengerjaan tugas dalam satuan jam
5. prasyarat: tugas prasyarat yang harus dikerjakan untuk bisa mengerjakan tugas tertentu
6. deadline_hari: tanggal tenggat waktu tugas, dihitung dari hari ke-0 pada simpul START dalam satuan hari
7. release_time_hari: tanggal tugas tersebut diberikan, dihitung dari hari ke-0 pada simpul START dalam satuan hari

B. *Prapemrosesan Data*

Agar setiap data mentah dapat diproses dengan baik, terdapat proses *preprocessing* menggunakan fungsi *loadAndPrepare()* dengan parameter lokasi file data mentah. Data mentah tersebut kemudian diolah menjadi data yang siap untuk diproses oleh kode program utama. Proses pada tahap *preprocessing* adalah menangani data kosong pada simpul START di kolom *prasyarat* serta mengubah satuan kolom *deadline_hari* dan *release_time_hari* menjadi dalam satuan jam.

```python
def loadAndPrepare(filepath):
    df = pd.read_csv(filepath)
    df['prasyarat'] = df['prasyarat'].fillna('')
    df['durasi_jam'] = df['durasi_jam'].astype(float)
    df['deadline_jam'] = df['deadline_hari'].astype(float)*24
    df['release_time_jam'] = df['release_time_hari'].astype(float)*24

    df['ES'] = 0.0
    df['EF'] = 0.0
    df['LS'] = 0.0
    df['LF'] = 0.0

    return df
```

Gambar 3.2 Potongan Kode Program untuk Proses *Preprocessing* Data

C. *Pembentukan Graf*

Setelah data mentah tugas kuliah selesai diolah menjadi data yang siap digunakan, setiap tugas akan diurutkan dengan *topological sort* pada fungsi *topologicalSort()* menggunakan metode *Depth First Search* sehingga tugas prasyarat dan tugas setelahnya berada pada urutan yang benar dalam graf.

```python
def topologicalSort(df):
    adjList = {}
    for _, row in df.iterrows():
        preds = [p.strip() for p in str(row['prasyarat']).split(';') if p.strip()]
        adjList[row['id']] = preds

    visited = set()
    stack = []

    def dfs(node):
        visited.add(node)

        successors = df[df['prasyarat'].str.contains(r'(^|;)' + node + r'(;|$)', regex=True, na=False)]['id'].tolist()
        for successor in successors:
            if successor not in visited:
                dfs(successor)
        stack.insert(0, node)

    dfs('START')
    return stack
```

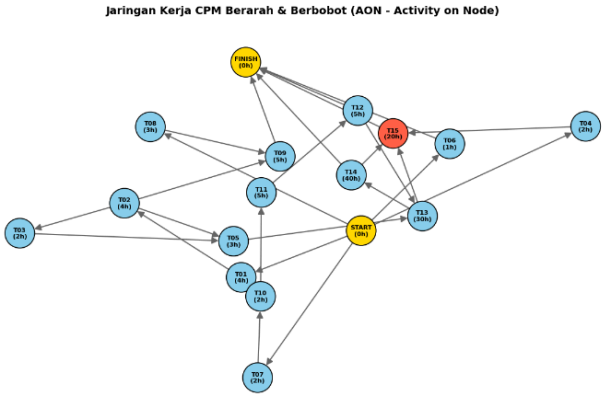
Gambar 3.3. Potongan Kode Program untuk *Topological Sort*

Hasil dari *topological sort* adalah sebuah list berbentuk *stack* yang mengandung urutan logis dari tugas prasyarat dan tugas *successor*. Simpul-simpul dalam *stack* tersebut divisualisasikan ke dalam sebuah Graf Berarah Asiklik dengan kode sebagai berikut:

```python
import networkx as nx
import matplotlib.pyplot as plt

def visualisasikan_graf(df):
    # 1. Inisialisasi Graf Berarah (DiGraph)
    G = nx.DiGraph()

    color_map = []
    labels = {}

    # 2. Iterasi untuk membuat Simpul (Nodes) beserta Bobot & Labelnya
    for _, row in df.iterrows():
        node_id = row['id']
        G.add_node(node_id)

        # ROBOT DI DALAM SIMPUL: Gabungkan ID Tugas dan Durasi Jam (h = hours)
        if node_id in ['START', 'FINISH']:
            labels[node_id] = f"{node_id}\n(0h)"
        else:
            labels[node_id] = f"{node_id}\n({int(row['durasi_jam'])}h)"

        if node_id in ['START', 'FINISH']:
            color_map.append('#FFD700')
        elif row['status'] == 'Kritis':
            color_map.append('#FF6347')
        elif row['status'] == 'Jadwal Mustahil (Minus)':
            color_map.append('#DC143C')
        else:
            color_map.append('#87CEEB')

    # 3. Iterasi untuk membuat Sisi/Garis Berarah (Edges)
    for _, row in df.iterrows():
        node_id = row['id']
        preds = [p.strip() for p in str(row['prasyarat']).split(';') if p.strip()]
        for pred in preds:
            # Menggambar arah dari Prasyarat -> Tugas Selanjutnya
            G.add_edge(pred, node_id)

    # 4. Pengaturan Layout Tampilan Graf
    plt.figure(figsize=(16, 9))

    # Menggunakan kamus posisi (layout) agar graf menyebar secara proporsional
    pos = nx.spring_layout(G, seed=42, k=0.6)

    # 5. Proses Menggambar Elemen Graf Berarah
    nx.draw_networkx_nodes(G, pos, node_color=color_map, node_size=1500, edgecolors='black', linewidths=1.2)

    # Menggambar RANAH BERARAH (Dipertegas dengan arrowstyle dan arrowsize)
    nx.draw_networkx_edges(
        G, pos,
        arrowstyle='-|>',
        arrowsize=18,
        edge_color='#666666',
        width=1.5,
        node_size=1500
    )

    nx.draw_networkx_labels(G, pos, labels=labels, font_size=8, font_weight='bold', font_family='sans-serif')

    # 6. Finalisasi Tampilan
    plt.title("Jaringan Kerja CPM Berarah & Berbobot (AON - Activity on Node)", fontsize=14, fontweight='bold', pad=20)
    plt.axis('off')
    plt.show()
```

Gambar 3.4. Potongan Kode Program untuk Visualisasi Graf

Dari kode program tersebut, dihasilkan sebuah Graf Berarah Asiklik sebagai berikut:



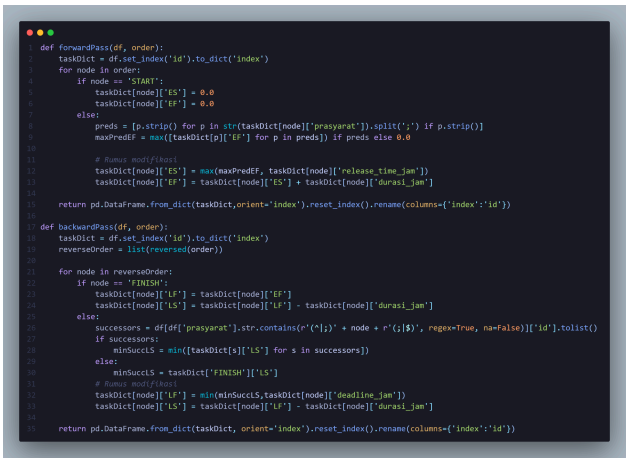
Gambar 3.5. Contoh Visualisasi Jaringan Tugas dengan Graf Menggunakan Bahasa Pemrograman Python

Visualisasi graf tersebut menggunakan *library NetworkX* dan *Matplotlib* menggunakan model *Activity on Node* (AoN). Interpretasi elemen graf pada Gambar 3.3 adalah sebagai berikut:

1. Simpul (*Node*): merepresentasikan tugas kuliah dengan identitas ID Tugas dan bobot durasi pengerjaan tugas.
2. Sisi Berarah (*Directed Edge*): merepresentasikan hubungan alur pengerjaan tugas. Simpul yang berada di pangkal panah adalah simpul prasyarat, sedangkan simpul yang berada di ujung panah adalah simpul *successor*.
3. Terdapat pewarnaan simpul dengan kriteria sebagai berikut:
    a. Kuning: menandai simpul START dan FINISH
    b. Biru: menandai simpul yang tidak termasuk *critical path*
    c. Merah: menandakan simpul yang termasuk *critical path*

D. *Implementasi Algoritma Critical Path Method*

Setelah data melewati proses *preprocessing* dan *topological sort*, dilakukan proses *forward pass* pada fungsi *forwardPass()* untuk menghitung nilai ES dan EF untuk mengetahui batas waktu tercepat setiap tugas selesai dikerjakan, serta proses *backward pass* pada fungsi *backwardPass()* untuk menghitung nilai LS dan LF berdasarkan batas waktu terakhir pengerjaan keseluruhan tugas secara paralel dengan modifikasi menggunakan variabel *release_time* dan *deadline*. Hasil nilai ES, EF, LS, dan LF dari setiap simpul disimpan dalam bentuk *pandas DataFrame*.



```python
def forwardPass(df, order):
    taskDict = df.set_index('id').to_dict('index')
    for node in order:
        if node == 'START':
            taskDict[node]['ES'] = 0.0
            taskDict[node]['EF'] = 0.0
        else:
            preds = [p.strip() for p in str(taskDict[node]['prasyarat']).split(';') if p.strip()]
            maxPredEF = max([taskDict[p]['EF'] for p in preds]) if preds else 0.0

            # Rumus Modifikasi
            taskDict[node]['ES'] = max(maxPredEF, taskDict[node]['release_time_jam'])
            taskDict[node]['EF'] = taskDict[node]['ES'] + taskDict[node]['durasi_jam']

    return pd.DataFrame.from_dict(taskDict,orient='index').reset_index().rename(columns={'index':'id'})

def backwardPass(df, order):
    taskDict = df.set_index('id').to_dict('index')
    reverseOrder = list(reversed(order))

    for node in reverseOrder:
        if node == 'FINISH':
            taskDict[node]['LF'] = taskDict[node]['EF']
            taskDict[node]['LS'] = taskDict[node]['LF'] - taskDict[node]['durasi_jam']
        else:
            successors = df[df['prasyarat'].str.contains(r'(^|;)' + node + r'(;|$)', regex=True, na=False)]['id'].tolist()
            if successors:
                minSuccLS = min([taskDict[s]['LS'] for s in successors])
            else:
                minSuccLS = taskDict['FINISH']['LS']
            # Rumus Modifikasi
            taskDict[node]['LF'] = min(minSuccLS,taskDict[node]['deadline_jam'])
            taskDict[node]['LS'] = taskDict[node]['LF'] - taskDict[node]['durasi_jam']

    return pd.DataFrame.from_dict(taskDict, orient='index').reset_index().rename(columns={'index':'id'})
```

Gambar 3.6. Potongan Kode Program untuk *Forward Pass* dan *Backward Pass*

E. *Perhitungan Nilai Slack dan Status*

Dari hasil ES, EF, LS, dan LF yang telah didapatkan sebelumnya, langkah terakhir adalah menghitung nilai *Slack* untuk setiap simpul tugas dengan rumus yang sudah dijabarkan pada Bab II bagian C. Status simpul kemudian diberikan berdasarkan hasil nilai *slack* tersebut, jika nilai *slack* kurang dari nol, status simpul tersebut adalah ‘Jadwal Mustahil (Minus)’. Jika nilai slack sama dengan nol, status simpul tersebut adalah ‘Kritis’. Jika nilai slack lebih dari nol, status simpul tersebut adalah ‘Santai’.

```python
def calculateSlackAndStatus(df):
    df['slack'] = df['LS'] - df['ES']

    def calculateSlack(row):
        if row['slack'] < 0:
            status = 'Jadwal Mustahil (Minus)'
        elif row['slack'] == 0:
            status = 'Kritis'
        else:
            status = 'Santai'
        return status
    df['status'] = df.apply(calculateSlack, axis = 1)
    return df
```

Gambar 3.7. Potongan Kode Program untuk Perhitungan Nilai *Slack* dan Status

F. Implementasi Kode utama

Blok program utama (if __name__ == “__main__”) berfungsi untuk mengatur alur data yang menggabungkan seluruh fungsi pemrosesan secara berurutan. Aliran data dari data mentah hingga menghasilkan *critical path* dirancang sebagai berikut:

1. Inisialisasi dan pemuatan data: dimulai dengan menentukan lokasi *file* berekstensi .csv pada variabel *filename. File* ini kemudian diteruskan kepada fungsi *loadAndPrepare()* untuk dilakukan proses *preprocessing* data, menghasilkan struktur data awal dalam variabel *dfInit*.
2. Pengurutan logis: variabel *dfInit* kemudian diteruskan kepada fungsi *topologicalSort()* untuk dilakukan proses penyusunan pengurutan logis sesuai dengan urutan antartugas dan menghasilkan sebuah urutan logis yang disusun di dalam list *urutanGraf*.
3. Proses *forward pass*: variabel *dfInit* dan *urutanGraf* diinput ke dalam fungsi *forwardPass()* untuk kalkulasi nilai ES dan EF, menghasilkan struktur data bertipe *pandas DataFrame* dengan nama variabel *dfForward*.
4. Proses *backward pass*: variabel *dfForward* dan *urutanGraf* diinput ke dalam fungsi *backwardPass()* untuk kalkulasi nilai LS dan LF hingga kembali ke simpul awal, menghasilkan *DataFrame* baru dengan nama variabel *dfBackward*.
5. Kalkulasi *slack* dan status: variabel *dfBackward* diinput ke dalam fungsi *calculateSlackAndStatus()* untuk kalkulasi nilai *slack* dan status. Data final yang siap disajikan disimpan dalam variabel *dfFinal*.

```python
if __name__ == "__main__":
    filename = '/kaggle/input/datasets/vinsju/data-tugas-6/data tugas 6.csv'

    dfInit = loadAndPrepare(filename)
    urutanGraf = topologicalSort(dfInit)
    dfForward = forwardPass(dfInit, urutanGraf)
    dfBackward = backwardPass(dfForward,urutanGraf)
    dfFinal = calculateSlackAndStatus(dfBackward)


    kolom_tampil = ['id', 'mata_kuliah', 'nama_tugas', 'durasi_jam', 'ES', 'EF', 'LS', 'LF', 'slack', 'status']
    print("\n" + "="*40 + " HASIL KALKULASI CPM AKADEMIK " + "="*40)
    print(dfFinal[kolom_tampil].to_string(index=False))
    print("="*110)

    # Cetak rangkuman tugas kritis
    tugas_kritis = dfFinal[dfFinal['status'] == 'Kritis']['id'].tolist()
    tugas_minus = dfFinal[dfFinal['status'] == 'Jadwal Mustahil (Minus)']['id'].tolist()
    print(f"\n Tugas di Jalur Kritis (Toleransi 0 Jam): {', '.join(tugas_kritis)}")
    print(f"Tugas Berstatus Mustahil/Overdue: {', '.join(tugas_minus) if tugas_minus else 'Tidak ada'}")
    visualisasikan_graf(dfFinal)
```

Gambar 3.8. Potongan Kode Program Utama

Output dari kode utama adalah sebuah tabel yang berisi *id, mata_kuliah, nama_tugas, durasi_jam, ES, EF, LS, LF, slack,* dan *status*. Di bagian bawah tabel terdapat informasi simpul atau tugas yang termasuk ke dalam *critical path* dan simpul atau tugas yang berstatus mustahil (nilai *slack* < 0).

```
========================================= HASIL KALKULASI CPM AKADEMIK =========================================
    id mata_kuliah                      nama_tugas  durasi_jam     ES     EF     LS     LF  slack status
 START        Umum                     Simpul Awal         0.0    0.0    0.0    0.0    0.0    0.0 Kritis
   T01       Alpro                 Alpro Minggu 1         4.0    0.0    4.0   92.0   96.0   92.0 Santai
   T02       Alpro                 Alpro Minggu 2         4.0    4.0    8.0  164.0  168.0  160.0 Santai
   T03       Alpro                     Kuis Alpro         2.0  168.0  170.0  190.0  192.0   22.0 Santai
   T04       Lokom              Logika Proposisi          2.0  192.0  194.0  214.0  216.0   22.0 Santai
   T05       Alpro            Skema Pengulangan          3.0  192.0  195.0  261.0  264.0   69.0 Santai
   T06       Lidia                       Kuis 2          1.0  240.0  241.0  527.0  528.0  287.0 Santai
   T07      Matdis                  Kuis 3 bab           2.0  264.0  266.0  310.0  312.0   46.0 Santai
   T08       Orkom          Latihan Assembly             3.0  288.0  291.0  405.0  408.0  117.0 Santai
   T09       Orkom                  Praktikum            5.0  312.0  317.0  475.0  480.0  163.0 Santai
   T10      Matdis Kuis Induksi, Boolean, Barisan        2.0  504.0  506.0  574.0  576.0   70.0 Santai
   T11      Matdis                Makalah Bab 1          5.0  600.0  605.0  715.0  720.0  115.0 Santai
   T12      Matdis                Makalah Bab 2          5.0  720.0  725.0  835.0  840.0  115.0 Santai
   T13       Alpro                Tugas Besar 1         30.0  840.0  870.0  906.0  936.0   66.0 Santai
   T14       Alpro                Tugas Besar 2         40.0 1080.0 1120.0 1112.0 1152.0   32.0 Santai
   T15       Lokom                   Praktikum          20.0 1200.0 1220.0 1200.0 1220.0    0.0 Kritis
FINISH        Umum                  Simpul Akhir          0.0 1220.0 1220.0 1220.0 1220.0    0.0 Kritis
================================================================================================================

Tugas di Jalur Kritis (Toleransi 0 Jam): START, T15, FINISH
Tugas Berstatus Mustahil/Overdue: Tidak ada
```

Gambar 3.9. Contoh Output Hasil Kalkulasi *Critical Path Method*

IV. HASIL DAN PEMBAHASAN

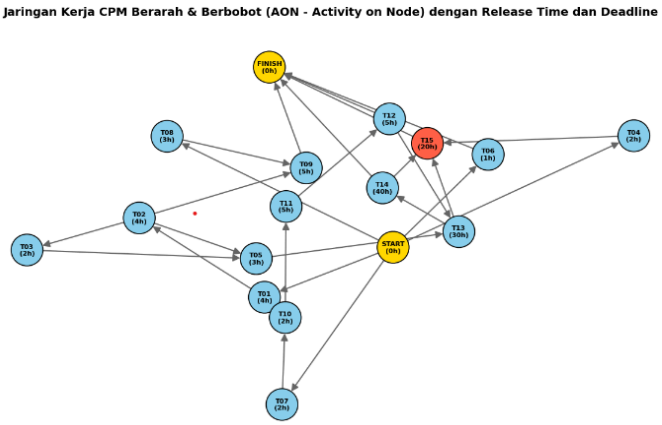
Untuk mendapatkan hasil *critical path* yang berbeda, akan dilakukan dua pengujian dengan kasus yang berbeda.

Studi kasus pertama, menggunakan parameter *release_time* (waktu rilis tugas) dan *deadline* (tenggat waktu tugas) sebagai batasan pada tugas untuk mencerminkan situasi nyata dalam perkuliahan. Hasil yang didapatkan adalah sebagai berikut:

```
========================================= HASIL KALKULASI CPM AKADEMIK =========================================
    id mata_kuliah                      nama_tugas  durasi_jam     ES     EF     LS     LF  slack status
 START        Umum                     Simpul Awal         0.0    0.0    0.0    0.0    0.0    0.0 Kritis
   T01       Alpro                 Alpro Minggu 1         4.0    0.0    4.0   92.0   96.0   92.0 Santai
   T02       Alpro                 Alpro Minggu 2         4.0    4.0    8.0  164.0  168.0  160.0 Santai
   T03       Alpro                     Kuis Alpro         2.0  168.0  170.0  190.0  192.0   22.0 Santai
   T04       Lokom              Logika Proposisi          2.0  192.0  194.0  214.0  216.0   22.0 Santai
   T05       Alpro            Skema Pengulangan          3.0  192.0  195.0  261.0  264.0   69.0 Santai
   T06       Lidia                       Kuis 2          1.0  240.0  241.0  527.0  528.0  287.0 Santai
   T07      Matdis                  Kuis 3 bab           2.0  264.0  266.0  310.0  312.0   46.0 Santai
   T08       Orkom          Latihan Assembly             3.0  288.0  291.0  405.0  408.0  117.0 Santai
   T09       Orkom                  Praktikum            5.0  312.0  317.0  475.0  480.0  163.0 Santai
   T10      Matdis Kuis Induksi, Boolean, Barisan        2.0  504.0  506.0  574.0  576.0   70.0 Santai
   T11      Matdis                Makalah Bab 1          5.0  600.0  605.0  715.0  720.0  115.0 Santai
   T12      Matdis                Makalah Bab 2          5.0  720.0  725.0  835.0  840.0  115.0 Santai
   T13       Alpro                Tugas Besar 1         30.0  840.0  870.0  906.0  936.0   66.0 Santai
   T14       Alpro                Tugas Besar 2         40.0 1080.0 1120.0 1112.0 1152.0   32.0 Santai
   T15       Lokom                   Praktikum          20.0 1200.0 1220.0 1200.0 1220.0    0.0 Kritis
FINISH        Umum                  Simpul Akhir          0.0 1220.0 1220.0 1220.0 1220.0    0.0 Kritis
================================================================================================================

Tugas di Jalur Kritis (Toleransi 0 Jam): START, T15, FINISH
Tugas Berstatus Mustahil/Overdue: Tidak ada
```

Gambar 4.1. Hasil Analisis dengan Memasukkan Parameter Batasan *release_time* dan *deadline*

Graf yang dihasilkan adalah sebagai berikut:



Gambar 4.2. Graf Berarah Berbobot dengan Variabel *release_time* dan *deadline*

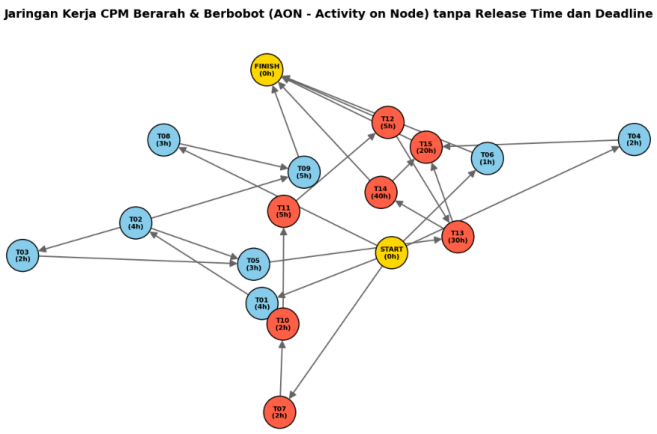
Pada Gambar 4.1 didapatkan hasil bahwa simpul yang memiliki nilai *slack* = 0 dan termasuk jalur kritis hanya pada simpul tugas T15, direpresentasikan pada Gambar 4.2 dengan simpul berwarna merah. Hal tersebut disebabkan karena tugas T15 memiliki waktu rilis pada jam ke-1200 serta tenggat waktu pada jam ke-1220, sesuai dengan durasi pengerjaan tugas (20 jam). Selain itu, untuk tugas lainnya yang tidak termasuk jalur kritis disebabkan karena rentang waktu yang besar (*time window*) yang berasal dari variabel *release_time* dan *deadline*, disertai dengan durasi pengerjaan tugas yang lebih singkat daripada rentang waktu yang ada, menyebabkan nilai *slack* yang relatif besar. Sebagai contoh, tugas T01 memiliki nilai *slack* sebesar 92 dan tugas T02 memiliki nilai slack sebesar 160. Akibatnya, mahasiswa memiliki waktu luang yang lebih banyak tanpa beresiko untuk menunda pengerjaan tugas tersebut hingga mengganggu pengerjaan tugas berikutnya.

Studi kasus kedua dilakukan dengan menghilangkan parameter batasan *release_time* dan *deadline*, sehingga perhitungan nilai *slack* hanya berdasarkan variabel durasi. Hal tersebut mengakibatkan sebuah asumsi bahwa setiap tugas tidak memiliki waktu rilis dan tenggat waktu sehingga dapat langsung dikerjakan setelah tugas sebelumnya selesai. Hasil yang didapatkan adalah sebagai berikut:

```
========================================= HASIL KALKULASI CPM AKADEMIK =========================================
    id mata_kuliah                nama_tugas  durasi_jam    ES    EF    LS    LF  slack status
 START       Umum               Simpul Awal         0.0   0.0   0.0   0.0   0.0    0.0 Kritis
   T01      Alpro            Alpro Minggu 1         4.0   0.0   4.0   1.0   5.0    1.0 Santai
   T02      Alpro            Alpro Minggu 2         4.0   4.0   8.0   5.0   9.0    1.0 Santai
   T03      Alpro               Kuis Alpro         2.0   8.0  10.0   9.0  11.0    1.0 Santai
   T04      Lokom          Logika Proposisi         2.0   0.0   2.0  82.0  84.0   82.0 Santai
   T05      Alpro        Skema Pengulangan         3.0  10.0  13.0  11.0  14.0    1.0 Santai
   T06      Lidia                  Kuis 2         1.0   0.0   1.0 103.0 104.0  103.0 Santai
   T07    Matdis              Kuis 3 bab         2.0   0.0   2.0   0.0   2.0    0.0 Kritis
   T08      Orkom        Latihan Assembly         3.0   0.0   3.0  96.0  99.0   96.0 Santai
   T09      Orkom               Praktikum         5.0   8.0  13.0  99.0 104.0   91.0 Santai
   T10    Matdis Kuis Induksi, Boolean, Barisan         2.0   2.0   4.0   2.0   4.0    0.0 Kritis
   T11    Matdis              Makalah Bab 1         5.0   4.0   9.0   4.0   9.0    0.0 Kritis
   T12    Matdis              Makalah Bab 2         5.0   9.0  14.0   9.0  14.0    0.0 Kritis
   T13      Alpro            Tugas Besar 1        30.0  14.0  44.0  14.0  44.0    0.0 Kritis
   T14      Alpro            Tugas Besar 2        40.0  44.0  84.0  44.0  84.0    0.0 Kritis
   T15      Lokom               Praktikum        20.0  84.0 104.0  84.0 104.0    0.0 Kritis
FINISH       Umum             Simpul Akhir         0.0 104.0 104.0 104.0 104.0    0.0 Kritis
================================================================================================================

Tugas di Jalur Kritis (Toleransi 0 Jam): START, T07, T10, T11, T12, T13, T14, T15, FINISH
Tugas Berstatus Mustahil/Overdue: Tidak ada
```

Gambar 4.3. Hasil Analisis Tanpa Menyertai Parameter Batasan *release_time* dan *deadline*

Graf yang dihasilkan berdasarkan data hasil analisis *Critical Path Method* adalah sebagai berikut:



Gambar 4.4. Visualisasi Graf Berarah Berbobot tanpa Variabel *release_time* dan *deadline*

Pada Gambar 4.3 didapatkan hasil bahwa simpul yang memiliki nilai *slack* = 0 dan termasuk ke dalam jalur kritis adalah T07, T10, T11, T12, T13, T14, dan T15. Hal tersebut disebabkan karena ketiadaan batasan sehingga proses *backward pass* dapat mengalirkan nilai *slack* = 0 secara kontinu, seperti yang dapat dilihat pada graf yang tercantum di Gambar 4.4. Studi kasus kedua yang berbasis *dynamic scheduling* memberikan fleksibilitas dan penjadwalan yang lebih realistis bagi mahasiswa, sehingga mahasiswa dapat memperbarui daftar tugas yang dimiliki setiap mendapatkan tugas baru.

V. KESIMPULAN

Algoritma *Critical Path Method* berhasil diimplementasikan dalam penjadwalan pengerjaan tugas mahasiswa di dalam perkuliahan. Hasil penelitian menunjukkan bahwa terdapat perbedaan signifikan antara dua skenario, yaitu skenario pertama yang menggunakan batasan waktu rilis dan deadline, serta skenario kedua yang tidak menggunakan kedua batasan tersebut. Skenario pertama menunjukkan jalur kritis yang lebih pendek, dengan hanya melibatkan satu simpul tugas (T15), serta simpul lain yang memiliki nilai slack cukup besar. Sedangkan, skenario kedua menunjukkan jalur kritis yang lebih panjang dan lebih relevan untuk diterapkan dalam kehidupan nyata, karena tidak terbatas pada waktu rilis tugas dan deadline yang kaku.

GITHUB LINK AT GITHUB

https://github.com/Vinsju/Discrete-Mathematics-Critical-Path-Method

ACKNOWLEDGMENT

Saya mengucapkan syukur kepada Tuhan Yang Maha Esa karena berkat-Nya, penulisan makalah ini yang berjudul " Penerapan Critical Path Method (CPM) Menggunakan Graf Berarah untuk Optimalisasi Rencana Belajar Mandiri Berdasarkan Skala Prioritas dan Deadline Tugas" dapat selesai dengan baik dan maksimal. Saya mengucapkan terima kasih kepada dosen pengampu mata kuliah Matematika Diskrit,

Bapak Rinaldi Munir, yang telah memberikan ilmu dengan sebaik-baiknya selama perkuliahan.

References

[1] Department of Industrial and Manufacturing Systems Engineering, Iowa State University, “The Critical Path,” Aug. 2015. [Online]. Available: https://www.imse.iastate.edu/files/2015/08/Critical-Path.pdf. [Accessed: Jun. 17, 2026].

[2] College of Engineering and Computer Science, California State University, Sacramento, “The Critical Path Method,” n.d. [Online]. Available: https://athena.ecs.csus.edu/~buckley/CSc233/The%20Critical%20Path%20Method.pdf. [Accessed: Jun. 17, 2026].

[3] M. M. Priya, “Topological Sorting,” Department of Computer Science, Illinois Institute of Technology, 2012. [Online]. Available: https://www.cs.iit.edu/~cs560/fall_2012/Research_Paper_Topological_sorting/Topological%20sorting.pdf. [Accessed: Jun. 17, 2026].

[4] R. Munir, “Graf Bagian I (Update 2026),” IF1220 Matematika Diskrit, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/20-Graf-Bagian1-2026.pdf. [Accessed: Jun. 17, 2026].

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, 19 Juni 2026

Vinsensius Juan Setiady - 13525093