Penerapan Konsep Lintasan Hamilton pada Graf Lengkap Berbobot dalam Pengurutan *Playlist* Musik

Kharris Khisunica - 13522051[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522051@std.stei.itb.ac.id

***Abstract*—Musik adalah salah satu bentuk seni yang dinikmati oleh Masyarakat. Perkembangan zaman membuat masyarkat terekspos dengan banyak jenis musik dari berbagai era dan dari artis mancanegara. Oleh sebab itu, dalam makalah ini akan dibahas salah satu metode untuk membentuk suatu *Playlist* (*Playlist*) yang mengurutkan lagu berdasarkan parameter yang ada dalam suatu musik. *Playlist* akan dibentuk dengan memanfaatkan konsep lintasan Hamilton dalam satu graf lengkap, yaitu graf yang berisi data musik-musik masukan dari pengguna menggunakan algoritma Mehendale.**

***Kata Kunci*—graf, lintasan Hamilton, Mehendale, *Playlist*, musik**

I. PENDAHULUAN

Musik dalam Kamus Besar Bahasa Indonesia diartikan sebagai: (1) Ilmu atau seni menyusun nada atau suara dalam urutan, kombinasi, dan hubungan temporal untuk menghasilkan komposisi (suara) yang mempunyai kesatuan dan kesinambungan; (2) Nada atau suara yang disusun demikian rupa sehingga mengandung irama, lagu, dan keharmonisan (terutama yang menggunakan alat-alat yang dapat menghasilkan bunyi-bunyi itu). [1]

Musik merupakan salah satu bentuk seni yang banyak dinikmati oleh masyarakat luas. Dengan perkembangan zaman dan globalisasi, musik juga mengalami perkembangan yang cukup pesat dimana musik dari berbagai era dan artis dengan lebih praktis.

Dengan bertambahnya variasi musik yang bisa didengarkan, muncul permasalahan dimana pengguna kebingungan bagaimana mengurutkan musik tersebut berdasarkan kategori-kategori yang diinginkan. Terdapat banyak kategori yang terhubung ke suatu musik tetapi dalam makalah ini, penulis mengambil beberapa kategori, yaitu Judul, Nama Artis, Nama Album, Tahun rilis, dan durasi.

Makalah ini membahas mengenai cara mengurutkan lagu menjadi suatu *Playlist* berdasarkan masukan lagu dan kategori yang dipilih. *Playlist* yang dibuat akan berisi seluruh lagu tepat satu kali dan dibentuk dengan mencari lintasan Hamilton paling pendek dalam suatu graf lengkap yang terbentuk dari lagu-lagu yang telah dimasukkan pengguna menggunakan algoritma Mehendale.

II. LANDASAN TEORI

A. Teori Graf [2]

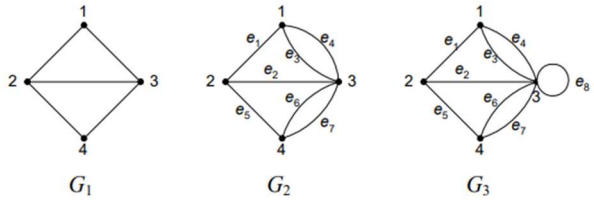
A.1 Definisi Graf

Graf adalah pasangan terurut dari pasangan G = (V,E) yang terdiri dari himpunan tidak-kosong V yang berisi *vertices*/simpul-simpul $\{v_1, v_2, \dots, v_n\}$, dan himpunan E berisi *edges*/sisi yang menghubungkan sepasang simpul $\{e_1, e_2, \dots, e_m\}$[1]. Hubungan antara sisi dan simpul dapat dinotasikan sebagai $e_p = (v_i, v_j)$ dimana $v_i, v_j \in V$ dan $e_p \in E$.

Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut.[3]

A.2 Jenis-Jenis Graf

A.2.1 Graf Sederhana dan Graf Tak-Sederhana



a) b) c)

Gambar 2.1 a) Ilustrasi Graf Sederhana, b) Ilustrasi Graf tak-sederhana jenis Graf Ganda, c) Ilustrasi Graf tak-sederhana jenis Graf semu
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 07/12/2023

Suatu graf dikatakan memiliki gelang jika dalam himpunan E terdapat e_i sedemikian hingga $e_i = (v_j, v_j)$ untuk $e_i, e_j \in E$. Secara visual, suatu sisi disebut gelang jika dia berawal dan berakhir pada simpul yang sama. Pada gambar 1.1, contoh gelang adalah e_8 di G3 karena $e_8 = (3,3)$.

Suatu graf dikatakan memiliki sisi ganda jika dalam himpunan E terdapat $e_i = e_j = (v_p, v_q), i \neq j$ untuk $v_p, v_q \in V$ dan $e_i, e_j \in E$ atau bisa dikatakan elemen himpunan E tidak unik. Secara visual, suatu graf disebut memiliki sisi ganda jika ada dua sisi yang berawal dari simpul yang sama dan berakhir

[ada simpul yang sama juga. Pada gambar 1.1, contoh sisi ganda adalah e_6 dan e_7 karena $e_6 = e_7 = (3,4)$ dan $6 \neq 7$.

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka graf digolongkan menjadi dua jenis:

1. Graf sederhana (*simple graph*).

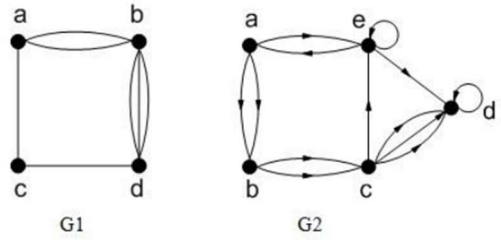
Suatu graf dikatakan sederhana jika graf tersebut tidak memiliki gelang dan tidak ada sisi ganda.

2. Graf tak-sederehana (*unsimple-graph*)

Suatu graf dikatakan tak-sederehana jika graf mengandung gelang dan/atau sisi ganda. Graf tak-sederhana bisa dibedakan menjadi dua jenis:

a. Graf ganda (*multi-graph*), yang memiliki sisi ganda.
b. Graf semu (*pseudo-graph*), yang memiliki sisi gelang.

A.2.2 Graf Tak-Berarah dan Graf Berarah



Gambar 2.2 a) Ilustrasi Graf Tak-Berarah, b) Ilustraksi Graf Berarah
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 07/12/2023

Berdasarkan orientasi arah pada sisi, graf dapat dibedakan menjadi 2 jenis:

1. Graf tak-berarah (*undirected graph*)

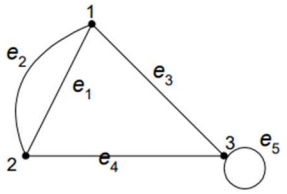
Suatu graf dikatakan tak berarah jika suatu graf tidak memiliki orientasi arah dan $(v_i, v_j) = (v_j, v_i)$ untuk $v_i, v_j \in V$. Secara visual, sisi dari graf tak-bearah hanya berupa garis yang menghubungkan dua simpul.

2. Graf berarah (*directed graph*)

Suatu graf dikatakan berarah jika setiap sisi pada graf tersebut memiliki orientasi arah. Hal ini berarti $(v_i, v_j) \neq (v_j, v_i)$ untuk $v_i, v_j \in V$. karena (v_i, v_j) menunjukan sisi yang berasal dari v_i dan menunjuk ke v_j dan (v_j, v_i) menunjukan sisi yang berasal dari v_j dan menunjuk ke v_i.

A.3 Teriminologi Graf

Beberapa terminologi graf yang akan dipakai di dalam makalah ini:



Gambar 2.3 Contoh Graf
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 07/12/2023

1. Ketetanggan (*Adjacent*)

Dua buah simpul dikatakan *bertetangga* bila keduanya terhubung secara langsung. Secara matematis, simpul $v_i, v_j \in V$ dikatakan bertetangga bila ditemukan $e_p \in E$ sedemikian hingga $e_p = (v_i, v_j)$. Secara visual, v_i dan v_j bertetangga bisa ada sisi yang menghubungkan kedua simpul. Contohnya pada gambar 1.3, simpul 1 dan 3 bertetangga karena ada simpul e_3 yang menghubungkan kedua simpul tersebut.

2. Bersisian (*Incidency*)

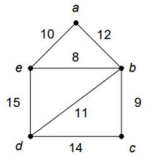
Sebuah simpul dikatakan *bersisian* dengan suatu sisi jika simpul tersebut adalah salah satu dari dua simpul pembentuk sisi tersebut. Secara matematis, simpul v_i dikatakan bersisian dengan e jika $e = (v_i, v)$ atau $e = (v, v_i)$ untuk suatu $v \in V$ dengan $v_i \in V$ dan $e \in E$. Secara visual, v_i bersisian dengan e jika v_i adalah salah satu titik pembentuk e_p untuk $v_i \in V$ dan $e \in E$. Contohnya pada gambar 1.3, simpul 1 dan sisi e_1 bersisian karena simpul 1 adalah salah satu pembentuk e_1.

3. Lintasan (*Path*)[5]

Lintasan pada graf G adalah barisan terbatas tak-kosong L = $v_0 e_1 v_1 e_2 v_2 \ldots e_k v_k$, dimana $v_i \in V, 0 \leq i \leq n$, dan $e_j \in E, 1 \leq j \leq m$. Dalam lintasan, simpul dan sisi disusun berselang-seling sedemikian hingga, untuk $1 \leq i \leq k$, $e_i = (v_{i-1}, v_i)$. Lintasan L bisa disebut sebagai lintasan dari v_0 ke v_k, dimana v_0 disebut simpul awal dan v_k disebut simpul tujuan dari L dan k adalah panjang dari lintasan L. Contoh lintasan pada gambar 1.3 adalah misal v_0 = simpul 1 dan $v_n =$ simpul 2. Maka L yang terbentuk bisa L = $1e_1 2$ dengan panjang 1 atau $L = 1e_3 3e_4 2$ dengan panjang 2.

4. Graf Berbobot (*Weighted Graph*)

Graf berbobot adalah graf yang setiap sisinya diberi harga (bobot).



Gambar 2.4 Graf Berbobot
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 07/12/2023

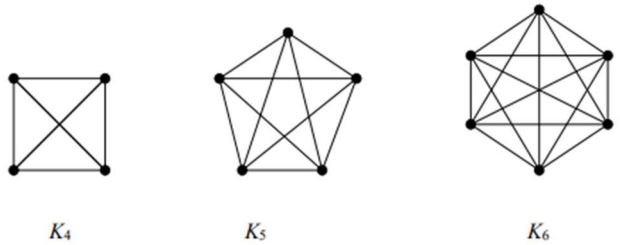
5. Derajat (*Degree*)

Derajat suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut. Derajat dari simpul v adalah $d(v)$. Derajat dari suatu simpul juga bisa didapat dengan

menghitung kemunculan simpul v dalam E. Contohnya terdapat 2 sisi yang salah satu simpul pembentuknya adalah v, maka $d(v) = 2$. Contoh visual pada gambar 2.4, bisa didapat derajat simpul $e = d(e) = 3$ karena ada tiga sisi yang bersisian dengan simpul e.

6. Graf Lengkap (*Complete Graph*)

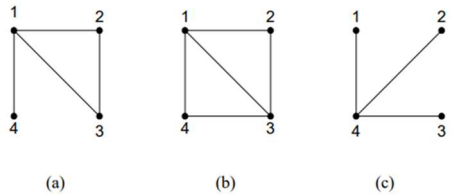
Graf lengkap adalah graf sederhana yang setiap simpulnya mempunyai sisi ke semua simpul lainnya. Jumlah sisi pada suatu graf lengkap yang terdiri dari n buah simpul adalah $\frac{n(n-1)}{2}$. Secara matematis, suatu graf G(V,E) disebut graf lengkap jika untuk semua $v_i \in V$ terdapat $e \in E$ unik sedemikian hingga $e = (v_i, v_j)$ atau $e = (v_j, v_i)$ untuk semua $v_j \in V, v_j \neq v_i$.



Gambar 2.5 Graf Lengkap
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 07/12/2023

B. Lintasan dan Sirkuit Hamilton

Lintasan Hamilton adalah lintasan yang melalui setiap simpul
di dalam graf tepat satu kali dan simpul awal dan akhirnya berbeda. Sirkuit Hamilton adalah lintasan yang melalui tiap simpul di dalam graf tepat satu kali, kecuali simpul asal yang sekaligus menjadi simpul akhir yang dilalui dua kali. Karena sirkuit Hamilton juga melalui graf tepat satu kali kecuali simpul awalnya, maka jika suatu graf memiliki sirkuit Hamilton, maka graf tersebut juga memiliki lintasan Hamilton. Dalam suatu graf lengkap dengan n buah simpul, terdapat $\frac{(n-1)!}{2}$ Sirkuit Hamilton. Karena dari tiap sirkuit Hamilton bisa dibentuk n buah lintasan Hamilton, maka untuk graf lengkap dengan n buah simpul, terdapat $\frac{n!}{2}$ lintasan Hamilton.



Gambar 2.6 Ilustrasi Lintasan dan Sirkuit Hamilton
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/21-Graf-Bagian3-2023.pdf, diakses pada 09/12/2023

Graf (a) memiliki lintasan Hamilton karena terdapat lintasan yang melalui seluruh simpul dengan jalur 2-3-1-4. Graf (b) memiliki sirkuit Hamilton karena terdapat jalur yang melalui seluruh simpul dan Kembali ke simpul awal, yaitu jalur 1-2-3-4-1. Graf (c) bukan lintasan dan sirkuit Hamilton karena tidak akan pernah ditemukan jalur yang bisa melalui setiap simpul tepat satu kali.

C. Algoritma Mehendale untuk Pencarian Lintasan Hamilton[4]

Misalkan G=(V,E) adalah graf berbobot lengkap dimana

$$V = \{v_1, v_2, \dots, v_p\} \text{ dan}$$
$$E = \{e_1, e_2, \dots, e_q\}.$$

Karena G adalah graf lengkap, berdasarkan teori dari graf lengkap, maka dapat diperoleh untuk p simpul, $q = \frac{p(p-1)}{2}$.

Misal dibentuk list ketetanggaan berbobot, *LKB(G)*, yang berisi seluruh elemen dari E dan disimpan dalam list berukuran $q \times 2$, dimana elemen di kolom ke-1 akan berisi bobot dari suatu sisi dan kolom ke-2 akan berisi simpul pembentuk sisi. List *LKB(G)* bisa diperoleh dari A_G dengan mengisi kolom ke-1 *LKB(G)* dengan elemen baris ke-i dan kolom ke-j dimana $1 \leq i < j \leq p$ secara transversal.

$$LKB(G) = \begin{bmatrix} w_{12} & (v_1, v_2) \\ w_{13} & (v_1, v_3) \\ w_{14} & (v_1, v_4) \\ \vdots & \vdots \\ w_{ij} & (v_i, v_j) \\ \vdots & \vdots \\ w_{(p-2)p} & (v_{p-2}, v_p) \\ w_{(p-1)p} & (v_{p-1}, v_p) \end{bmatrix}$$

Akan dibentuk lagi suatu list, yaitu list ketetanggan berbobot terurut, *LKBT(G)*. *LKBT(G)* adalah *LKB(G)* yang elemen bobotnya terurut tak mengecil, yakni $w_1 \leq w_2 \leq \cdots \leq w_{q-1} \leq w_q$, dimana q = jumlah sisi = $\frac{p(p-1)}{2}$.

$$LKBT(G) = \begin{bmatrix} w_1 & (v_{x_1}, v_{y_1}) \\ w_2 & (v_{x_2}, v_{y_2}) \\ w_3 & (v_{x_3}, v_{y_3}) \\ \vdots & \vdots \\ w_n & (v_{x_n}, v_{y_n}) \\ \vdots & \vdots \\ w_{q-1} & (v_{x_{q-1}}, v_{y_{q-1}}) \\ w_q & (v_{x_q}, v_{y_q}) \end{bmatrix}$$

Karena *LKBT(G)* adalah hasil pengurutan *MKB(G)* dan setiap simpul di kolom ke-2 dari *LKB(G)* elemen dari V, maka $v_{x_i}, v_{y_j} \in V, 1 \leq i, j \leq q$.

Beberapa definisi lagi yang perlu diketahui:

1. Sub-List ketetanggaan berbobot, *SubLKB(G)*, adalah sub-List yang terbentuk dengan mengambil secara acak kolom dari *LKB(G)*.
2. Sub-List ketetanggan berbobot terurut, *SubLKBT(G)*, adalah sub-list yang terbentuk dari mengambil bagian dari *LKBT(G)* dan elemen bobotnya tetap terurut tak membesar.
3. Besar dari *SubLKB(G)* atau *SubLKBT(G)* adalah jumlah baris dari sub-list tersebut.

4. Bobot dari *SubLKB(G)* atau *SubLKBT(G)* adalah jumlah elemen bobot di kolom ke-1 dari sub-list tersebut.

Lemma Lintasan Hamilton pada *SubLKBT(G)*:

Sebuah himpunan dari $(p-1)$ pasangan simpul pada kolom ke-2 dari *LKBT(G)* akan membentuk lintasan Hamilton jika (i) seluruh pasangan simpul mengandung seluruh simpul pada G, (ii) derajat dari $p-3$ simpul adalah 2, (iii) derajat dari 2 simpul lainnya adalah 1, dan (iv) seluruh pasangan simpul saling terhubung membentuk graf.

Pembuktian:

Pembuktian untuk (i) dan (iv) trivial karena definisi lintasan Hamilton adalah jalur yang melalui seluruh simpul pada graf G tepat satu kali dan simpul awal dan akhirnya berbeda.

(ii) dan (iii) akan dibuktikan dengan definisi lintasan. Ingat $L = v_1e_1v_2e_2v_3 \dots e_{p-1}v_{p-1}$ dimana L adalah lintasan Hamilton. Implikasinya adalah $v_i, \forall\ 1 \le i \le p-1$ unik dan untuk tiap $v_i, 2 \le i \le p-2$, sisi yang mengandung v_i hanya 2, yaitu $e_{i-1} = (v_{i-1}, v_i)$ dan $e_i = (v_i, v_{i+1})$. Dapat dilihat juga derajat v_1 dan v_{p-1} adalah 1 karena hanya terdapat 1 sisi yang mengandung v_1 dan v_{p-1}, yaitu $e_1 = (v_1, v_2)$ dan $e_{p-1} = (e_{p-2}, e_{p-1})$ berturut-turut. Bisa didapatkan banyak simpul yang derajatnya 2 adalah $(p-1)-2 = p-3$. (Terbukti)

Algoritma:

1. Bentuk list ketetanggaan berbobot, *LKB(G)* dari A_G lalu urutkan membentuk list ketetanggaan berbobot terurut, *LKBT(G)*.
2. Bentuk seluruh Sub-List ketetanggan berbobot terurut, *SubLKBT(G)*, masing-masing berukuran $(p-1)$.
3. Urutkan seluruh *SubLKBT(G)* yang terbentuk secara tak mengecil berdasarkan bobotnya.
4. Cek apakah *SubLKBT(G)* adalah lintasan Hamilton dengan menggunakan lemma Lintasan Hamilton pada *SubLKBT(G)* secara transversal dan berhenti saat lintasan Hamilton pertama kali ditemukan
5. Catat pasangan-pasangan simpul yang didapat serta bobot dari *SubLKBT(G)*. Dari pasangan-pasangan simpul, tentukan simpul awal dan akhir dan dapatkan jalur lintasan Hamilton menggunakan $L = v_1e_1v_2e_2v_3 \dots e_{p-1}v_{p-1}$. Maka didapat lintasan Hamilton terpendek.

III. PENERAPAN

A. Overview Program

Program ini membuat *Playlist* dengan menerima masukan data lagu dari pengguna. Disini pengguna diminta untuk memasukkan 5 kategori, yaitu judul lagu, nama artis, nama album, tahun rilis, dan durasi lagu. Setelah itu, program akan meminta pengguna untuk memasukkan kategori apa yang akan menjadi patokan lalu program akan mencari lintasan Hamilton paling pendek, yang bertepatan dengan urutan lagu yang diinginkan pengguna. Setelah itu, pengguna diminta untuk memasukkan apakah *Playlist* akan terurut menaik atau menurun, kemudian program akan mengembalikan *Playlist* lagu yang sudah terurut sesuai keinginan pengguna.

B. Deklarasi Kamus dan Fungsi/Prosedur Program

```
{*** Deklarasi Graf *** }
constant Kapasitas: integer = 100
constant Kapasitas_LKB: integer = Kapasitas*(Kapasitas-1)/2
type IdType: integer
type ElType: real
type Graf: < V: array [1..Kapasitas] of Vertices;
             E: array [1..Kapasitas_LKB] of Edges;
        NEffV: integer > 0;
        NEffE: integer = NEffV*(NEffV-1)/2 >
type Vertices: < Id:  IdType, {identitas Vertices}
                 ListNilai: List> {List menyimpan identitas
                                   lagu}

type PsgSimpul: (Awal:Vertices, Akhir:Vertices)
type Edges: < Pasangan:PsgSimpul,
        Bobot: ElType> { bobot dari sisi }
type List: < LSifat: array [1..5] of ElType {
        Index 1: ASCII_Judul: ElType, {Representasi ASCII
        dari judul}
        Index 2: ASCII_Artis: ElType,{Representasi ASCII
        dari Nama Artis}
        Index 3:ASCII_Album: ElType, {Representasi ASCII
        dari Nama Album}
        Index 4:  Tahun_Rilis: ElType, {Tahun rilis lagu},
        Index 5: Durasi: ElType, {Durasi lagu setelah diubah
        ke ElType},
        Nama_Judul: string, {Judul lagu}
        Nama_Artis: string, {Nama dari Artis}
        Nama_Album: string, {Nama dari Album}
>

{*** Selektor ***}
{Jika G adalah graf, V adalah Vertices, E adalah Edges, P
adalah PsgSimpul, dan L adalah List:
      Vertices(G)      untuk mengakses G.V;
      Edges(G)         untuk mengakses G.E;
      NEff_V(G)        untuk mengakses G.NEffV;
      NEff_E(G)        untuk mengakses G.NEffE;
      ID(V)            untuk mengakses V.Id;
      List(V)          untuk mengakses V.ListNilai;
      Awal(P)          untuk mengakses P.Awal;
      Akhir(P)         untuk mengkases P.Akhir;
      Pasangan(E)      untuk mengakses E.Pasangan;
      Bobot(E)         untuk mengakses E.Bobot;
      ASCII_Judul(L)   untuk mengakses L.LSifat[1];
      ASCII_Artis(L)   untuk mengakses L.LSifat[2];
      ASCII_Album(L)   untuk mengakses L.LSifat[3];
      Tahun(L)         untuk mengakses L.LSifat[4];
      Durasi(L)        untuk mengakses L.LSifat[5];
      Judul(L)         untuk mengakses L.Nama_Judul
      Artis(L)         untuk mengakses L.Nama_Artis;
      Album(L)         untuk mengakses L.Nama_Album;

}

{*** Deklarasi LKB(G) dan LKBT(G) ***}

type LKB: < Bobot: array [1..Kapasitas_LKB] of ElType,
            {Bobot}
```

```
        Pasangan: array [1..Kapasitas_LKB] of
        PsgSimpul {Pasangan simpul},
        Neff: integer  = G.NEffE  >

{*** Selektor ***}
{Jika LK adalah LKB:
    Bobot(LK)          untuk mendapatkan LK.Bobot;
    Pasangan(LK)       untuk mendapatkan LK.Pasangan;
}

{*** Deklarasi Path dan SubLKB ***}
type Path: < array [1..G.NEffV-1] of  array of [1..(G.NEffV-
1)] of PsgSimpul>
{*** Deklarasi fungsi/prosedur ***}

procedure MakeGraph (input nLagu: IdType, output G: Graf)
{Membentuk graf "kosong" dengan nLagu menjadi NEffV.
I.S. nLagu integer ≥ 0
F.S. Graf G terdefinisi dengan NEffV = nLagu, V terisi
[v1, v2, ... vNEffV] dimana ID(vi) = i dan E terisi
[e12, e13, ... , e(NEf      )NEffE] Dimana Pasangan(eij)= (vi, vj)
}

procedure InputDataLagu (output V:Vertices, output G: Graf)
{prosedur ini meminta user untuk menginput data-data relevan
dari suatu lagu, yaitu Tahun rilis, Durasi dalam format
MM:DD, Judul lagu, Nama Artis, Nama album, dan genre dari
lagu tersebut. lalu memproses data yang diinput user dan
dimasukkan ke dalam V dan dimasukkan dalam G}

function StrToASCIIAvg (str:string) → real
{fungsi ini menerima inputan string lalu mengubah setiap
karakter (tanpa memandang apakah karakter tersebut huruf
besar atau huruf kecil) menjadi nilai ASCII yang sesuai. Fungsi
mengembalikan hasil pembagian dari penjumlahan seluruh
nilai ASCII dari string tersebut dijumlahkan dengan panjang
string}

procedure InputBobot (input/output G:Graf, input
kriteria:IdType)
{prosedur ini menginputkan bobot antara vi dan vj pada eij
berdasarkan id kriteria yang diinput user, lalu memasukkan
nilai bobot ke dalam eij}

function MakeLKB (G:Graf, LK:LKB) → LKB
{Fungsi ini menerima input G dan LK dan mengisi LK dengan
data dari Edges(G) secara transversal.
I.S. Edges(G) sudah terisi penuh, yakni Pasangan(E) dan
Bobot(E) tidak kosong. }

procedure SortLKB (input/output LK:LKB)
{prosedur ini menghasilkan LKBT(G), LKB yang Bobot(LK)
terurut tidak mengecil.}

procedure MakeSubLKBT (input LK:LKB, input nLagu:
IdType, input/output SubLK: SubLKB)
{Prosedur ini membentuk SubLKBT dengan ukuran nLagu-1
dan mengecek apakah subLKBT tersebut memenuhi lemma
Lintasan Hamilton pada LKBT(G). Jika memenuhi, maka
subLKBT tersebut akan dimasukkan kedalam SubLK dimana
bobot SubLK terurut tak mengecil}

function IsHamiltonPath (pt: Path) → Boolean
{fungsi ini mengembalikan true jika suatu Path adalah
Hamilton Path berdasarkan lemma Lintasan Hamilton pada
LKBT(G).}

function SortSubLK (SubLK: SubLKB) → SubLKB
{fungsi ini mengembalikan anggota pertama SubLKB yang
memiliki bobot paling kecil.}

procedure PrintPlaylist(input/output Playlist: SubLKB, input
naik: Boolean)
{prosedur ini mengeluarkan playlist yang terurut naik atau
terurut menurun berdasarkan input pengguna. Playlist naik
akan memprint Path(SubLKB) dari indeks 1 – nLagu-1 dan
Playlist turun akan memprint Path(SubLKB) dari indeks
nLagu-1 - 1 }
```

C. Realisasi Fungsi/Prosedur

```
procedure MakeGraph (input nLagu: IdType, output G: Graf)
{Membentuk graf "kosong" dengan nLagu menjadi NEffV.
I.S. nLagu integer ≥ 0
F.S. Graf G terdefinisi dengan NEffV = nLagu, V terisi
[v1, v2, ... vNEffV] dimana ID(vi) = i dan E terisi
[e12, e13, ... , e(NEf      )NEffE] Dimana Pasangan(eij)= (vi, vj)
}
{Kamus Lokal}
i,j : IdType
{Algoritma}
{Isi V}
i transversal [1..nLagu]
        ID(Vertices(G))[i] ← i


{Isi E}
i transversal [1..nLagu-1]
  j transversal [i+1..nLagu]
    Awal(Pasangan(Edges(G))[i]) ← Vertices(G)[i]
    Akhir(Pasangan(Edges(G))[j]) ← Vertices(G)[j]

procedure InputDataLagu (input nLagu: IdType, output
V:Vertices, output G: Graf)
{prosedur ini meminta user untuk menginput data-data relevan
dari suatu lagu, yaitu Judul lagu, Nama Artis, Nama album,
Tahun rilis, dan Durasi dalam format MM:DD dari lagu
tersebut. lalu memproses data yang diinput user dan
dimasukkan ke dalam V dan dimasukkan dalam G. Asumsi
User menginput data valid.}

{Kamus Lokal}
Judul, Nama_Artis, Nama_Album: string
Tahun,Menit,Detik: ElType
i,j : IdType

{Algoritma}
i transversal [1..nLagu]
        Output("Masukkan Data Lagu ke-" ,i, "!")
```

```
        Output("1. Judul Lagu: ")
        Input(Judul)

        Output("2. Nama Artis: ")
        Input(Nama_Artis)

        Output("3. Nama Album")
        Input(Nama_Album)

        Output("4.Tahun Rilis Lagu: ")
        Input(Tahun)


        Output("5. Durasi Lagu: ")
        Output("Menit: ")
        Input(Menit)
        Output("Detik: ")
        Input(Detik)


{Masukkan data kedalam V}

        {Masukkan String}
        Judul(List(V)[i]) ← Judul
        Artis(List(V)[i]) ← Artis
        Album(List(V)[i]) ← Album

        {Masukkan hasil AvgASCII}
        ASCII_Judul(List(V)[i]) ← StrToASCIIAvg(Judul)
        ASCII_Artis(List(V)[i]) ← StrToASCIIAvg(Artis)
        ASCII_Album(List(V)[i]) ← StrToASCIIAvg(Album)

        {Masukkan Tahun dan Durasi}
        Tahun(List(V)[i]) ← Tahun
        Durasi(List(V)[i]) ← 60*Menit + Detik


function StrToASCIIAvg (str:string) → ElType
{fungsi ini menerima inputan string lalu mengubah setiap
karakter (tanpa memandang apakah karakter tersebut huruf
besar atau huruf kecil) menjadi nilai ASCII yang sesuai. Fungsi
mengembalikan hasil pembagian dari penjumlahan seluruh
nilai ASCII dari string tersebut dijumlahkan dengan panjang
string}
{Asumsi: ada fungsi yang mengubah setiap huruf menjadi
varian huruf kecilnya, yaitu CharKecil(cc: char) → char, dan
string adalah array of character dengan indeks pertama = 1.
Ada fungsi juga yang mendapatkan panjang string, yaitu
len(str: string) → IdType. Dan ada fungsi yang mengubah nilai
char menjadi nilai ASCII yang sesuai, yaitu CharToASCII
(cc:char) → integer}

{Kamus Lokal}
nStr: IdType
sum: real
cc: string
{Algoritma}
nStr ← len(str)
sum ← 0
k transversal [1..nStr]
  cc ← CharKecil(str[k])
  sum ← sum + CharToASCII(cc)

→ sum/nStr {Mengembalikan sum/nStr tanpa pembulatan}


procedure InputBobot (input/output G:Graf, input
kriteria:IdType)
{prosedur ini menginputkan bobot antara v_i dan v_j pada e_ij
berdasarkan id kriteria yang diinput user, lalu memasukkan
nilai bobot ke dalam e_ij}

{Kamus Lokal}

{Algoritma}
While (j ≤ NEff_E(G)) do
        Bobot(Edges(G))[j] ←
abs(Bobot(Awal(Pasangan(Edges(G)))[i]  -
Bobot(Akhir(Pasangan(Edges(G)))[k]  )


function MakeLKB (G:Graf, LK:LKB) → LKB
{Fungsi ini menerima input G dan LK dan mengisi LK dengan
data dari Edges(G) secara transversal.
I.S. Edges(G) sudah terisi penuh, yakni Pasangan(E) dan
Bobot(E) tidak kosong. }

{Kamus Lokal}
i : IdType
{Algoritma}
i transversal [1..NEff_E(G)]
        Bobot(LK)[i] ← Bobot(Edges(G)[i])
        Pasangan(LK)[i] ← Pasangan(Edges(G)[i])

{TIDAK SELESAI}
procedure SortLKB (input/output LK:LKB)
{prosedur ini menghasilkan LKBT(G), LKB yang Bobot(LK)
terurut tidak mengecil.}
{Kamus Lokal}

{Algoritma}
{Urutkan LKB dengan tak mengecil dengan bobot(LK)
menjadi patokan}

procedure MakeSubLKBT (input LK:LKB, input nLagu:
IdType, input/output SubLK: SubLKB)
{Prosedur ini membentuk SubLKBT dengan ukuran nLagu-1
dan mengecek apakah subLKBT tersebut memenuhi lemma
Lintasan Hamilton pada LKBT(G). Jika memenuhi, maka
subLKBT tersebut akan dimasukkan kedalam SubLK dimana
bobot SubLK terurut tak mengecil}
{Kamus Lokal}

{Algoritma}
```

```
function IsHamiltonPath (pt: Path) → Boolean
{fungsi ini mengembalikan true jika suatu Path adalah Hamilton Path berdasarkan lemma Lintasan Hamilton pada LKBT(G).}
{Kamus Lokal}

{Algoritma}
{Periksa apakah setiap Vertices ada dengan mengecek berapa elemen unik, dan apakah sama dengan NEFF_V(G)/nLagu. Periksa derajat dengan mengecek kemunculan dari tiap vertices. Periksa Apakah membentuk graf }


function SortSubLK (input SubLK: SubLKB) → SubLKB
{fungsi ini mengembalikan anggota pertama SubLKB yang memiliki bobot paling kecil.}
{Kamus Lokal}

{Algoritma}

procedure PrintPlaylist(input/output Playlist: SubLKB, input naik: Boolean)
{prosedur ini mengeluarkan playlist yang terurut naik atau terurut menurun berdasarkan input pengguna. Playlist naik akan memprint Path(SubLKB) dari indeks 1 – nLagu-1 dan Playlist turun akan memprint Path(SubLKB) dari indeks nLagu-1 - 1 }
{Kamus Lokal}

{Algoritma}
```

D. Program Utama

```
{Kamus}
G: Graf; V: Vertices; E: Edges; LK: LKB; SubLK: LKB;
nLagu, IdKriteria: IdType
i: integer

{Algoritma}
output("Masukkan jumlah lagu")
input(nLagu)

MakeGraph(nLagu, G)

i transversal [1..nLagu]
        output("Masukkan data lagu ke-"+i)
        InputDataLagu(V,G);
{V berhasil terbentuk}

{Tentukan kriteria apa yang mau dipilih user}
Output("Pilihan Parameter:
    1. Tahun Rilis
    2. Durasi
    3. Judul Lagu
    4. Nama Artis
    5. Nama Album ")
Output("Masukkan parameter pengurutan Playlist (1-6): ")
Input(IdKriteria)
{IdKriteria sudah valid}

InputBobot (G,IdKriteria)

{Masukkan data ke dalam LKB(G)}

LK ← MakeLKB (G,LK)

{Urutkan komponen bobot pada LK untuk menghasilkan LK terurut atau LKBT(G).}

SortLKB (LK)

{LK sudah terurut}

Buatlah Sub LKBT}

{Carilah SubLKBT yang memenuhi Lemma}

MakeSubLKBT (LK, nLagu, SubLK)

Playlist ← Path(ShortestHamiltonPath (SubLK))

Output("Playlist dibuat terurut menurun? (Y/N): ")
Input(Urutan)

If (Urutan = Y) then
        PrintPlaylist(Playlist, true)
Else
        PrintPlaylist(Playlist,false)
```

IV. KESIMPULAN

Dalam pembentukan *Playlist* lagu berdasarkan masukan lagu. *Playlist* dapat dibentuk dengan mengaplikasikan prinsip teori graf lengkap berbobot dan lintasan Hamilton dengan memanfaatkan atribut dari tiap lagu menjadi bobot dari graf.

Playlist terbentuk dengan mencari lintasan Hamilton dengan total bobot terkecil yang ditemukan menggunakan Algoritma Mehendale, dimana *Playlist* bisa diurutkan berdasarkan atribut yang dipilih oleh pengguna.

V. SARAN

Saran untuk peneliti selanjutnya adalah untuk mencoba algoritma-algoritma lain yang lebih efisien untuk membentuk *Playlist*. Hal ini dikarenakan metode ini masih menghasilkan $O(n!)$ yang tidak efisien jika n besar. Selain itu, peneliti selanjutnya bisa menambahkan atribut lagu sehingga *Playlist* bisa dibuat lebih beragam.

VI. UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan puji dan Syukur kepada Tuhan Yang Maha Esa karena dengan pertolonganNya, Makalah berjudul "Penerapan Konsep Lintasan Hamilton pada Graf Lengkap Berbobot dalam Pengurutan *Playlist* Musik" dapat terselesaikan walau ada kekurangan. Penulis juga ingin mengucapkan terimakasih kepada Dr. Nur Ulfa Maulidevi, S.T, M.Sc. yang telah mengajar mata kuliah IF 2120 Matematika Diskrit dan telah membimbing penulis. Penulis juga ingin berterimakasih kepada D.P. Mehendale yang

algoritma nya menjadi bagian pokok dari makalah penulis. Terakhir, penulis juga mengucapkan terimakasih kepada seluruh pihak lain yang tidak bisa sebutkan satu-satu untuk bantuannya.

REFERENCES

[1] "Hasil Pencarian - KBBI Daring," kbbi.kemdikbud.go.id. https://kbbi.kemdikbud.go.id/entri/musik. [Diakses 7 Desember 2023]

[2] R. Munir, "IF2120 Matematika Diskrit".. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/. [Diakses 7-9 Desember 2023]

[3] "Matematika Diskrit: Pengertian Graf," *lmsspada.kemdikbud.go.id*. https://lmsspada.kemdikbud.go.id/mod/resource/view.php?id=47638, [Diakses 7 Desember 2023]

[4] D. P. Mehendale, *Polynomial Algorithms for Shortest Hamiltonian Path and Circuit.* Pune, India: Vixra, 2013.

[5] J Adrian Bondy and U S R Murty, Graph theory with applications. New York ; Chichester: Wiley, 2002.

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 2023

Kharris Khisunica, 13522051