Aplikasi Penentuan Graf Bipatrit

Menggunakan Algoritma *Breadth First Search*

Syarifuddin Fakhri Al Husaini 13518095
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
Email : 13518095@std.stei.itb.ac.id

Abstraksi—Dalam kehidupan sehari-hari, kita sering menjumpai graf. Contoh yang paling sering dilihat adalah bagan sebuah organisasi atau lembaga. Teori tentang graf sudah muncul sejak abad 18 dan telah diterapkan di berbagai bidang dalam kehidupan ini. Kegunaan dari graf sendiri adalah untuk mempresentasikan objek-objek diskrit dan hubungan antar objek itu sendiri. Salah satu jenis dari graf adalah graf bipatrit. Graf bipatrit termasuk graf yang unik karena memiliki sifat-sifat khusus tertentu. Dalam makalah ini akan dijelaskan penggunaan algoritma *Breadth First Search* dalam menentukan apakah sebuah graf termasuk dalam graf bipatrit. Algoritma tersebut merupakan sebuah algoritma penelusuran graf dan cara kerjanya dengan cara menandai simpul-simpul pada graf dengan suatu penanda yang berbeda, misal 1/0 atau biru/kuning dan lainnya. Dengan penggunaan algoritma *Breadth First Search* tersebut diharapkan dapat menyelesaikan persoalan ini secara mangkus atau efisien.

Kata kunci—graf, bipatrit, algoritma, BFS, *Breadth First Search*

I. PENDAHULUAN

1.1 Latar Belakang

Banyak sekali pengaplikasian graf dalam kehidupan sehari-hari. Mulai dari yang sederhana seperti bagan kepengurusan, pemetaan wilayah, hingga diterapkan dalam membangun sebuah jaringan LAN komputer. Umumnya graf digunakan untuk memetakan atau menentukan alur sebuah objek ke objek lainnya.

Salah satu jenis graf adalah graf bipatrit. Graf tersebut mempunyai keunikan tersendiri diantara graf yang lainnya. Kita sebenarnya dapat menentukan apakah sebuah graf bipatrit atau tidak dengan mata telanjang. Akan tetapi semakin banyak simpul pada graf akan membuat semakin sulit untuk menentukan apakah graf tersebut bipatrit atau tidak. Makalah ini akan dijelaskan sebuah algoritma yang digunakan untuk menentukan apakah sebuah graf bipatrit atau tidak secara mangkus atau efisien.

1.2 Tujuan dan Manfaat

Tujuan dari pembuatan makalah ini adalah sebagai berikut:

a. Memenuhi tugas perkuliahan mata kuliah strategi algoritma teknik informatika semester 4 sebagai salah satu indikator dalam penilaian;
b. Mengetahui cara kerja algoritma *Breadth First Search* dalam menentukan apakah sebuah graf tergolong dalam graf bipatrit.

Manfaat dari pembuatan makalah ini adalah sebagai berikut:

a. Meningkatkan kesadaran mahasiswa pentingnya membaca;
b. Meningkatkan kemampuan mahasiswa teknik informatika semester 4 dalam hal menganalisis sebuah permasalahan;
c. Meningkatkan kemampuan mahasiswa teknik informatika semester 4 dalam memecahkan permasalahan dengan menggunakan pengetahuan strategi algoritma yang telah diajarkan dalam perkuliahan;
d. Mengasah kemampuan mahasiswa teknik informatika semester 4 dalam membuat sebuah karya ilmiah ataupun makalah dengan waktu yang telah ditentukan;
e. Sebagai media penyebaran informasi hasil pemikiran dan penelitian. Makalah yang dibuat oleh mahasiswa teknik informatika semester 4 nantinya akan dimuat dalam *website* dosen sehingga siapa pun dapat membaca karya ilmiah yang dibuat oleh mahasiswa tersebut.

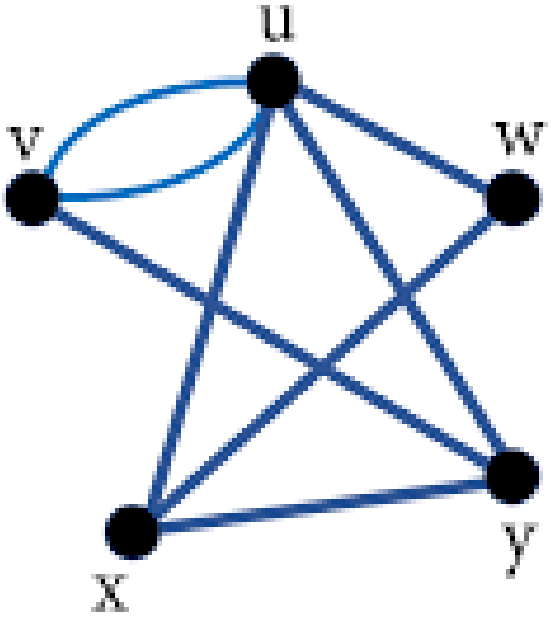
II. DASAR TEORI

2.1 Graf

Teori graf merupakan salah satu cabang matematika yang sudah ada sejak hampir 300 tahun silam. Jurnal pertama yang berkaitan tentang graf dibuat oleh Euler, seorang matematikawan terkenal asal Swiss, pada tahun 1736. Secara sederhana graf adalah kumpulan titik-titik yang disebut dengan simpul yang antar titik-titiknya dihubungkan dengan garis atau disebut dengan sisi sehingga membentuk sebuah keterhubungan antar satu sama lain.

Sebuah graf G berisikan dua himpunan yaitu himpunan berhingga tak kosong V(G) dari objek-objek

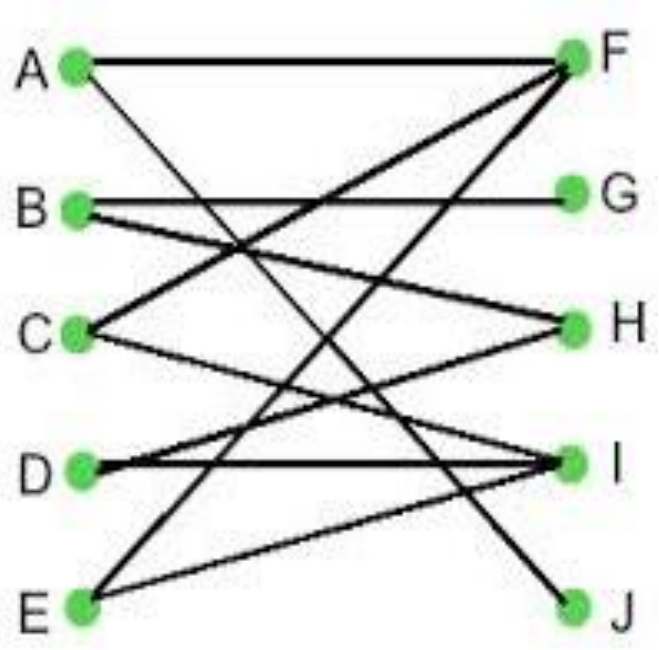
yang disebut titik dan himpunan berhingga (mungkin kosong) E(G) yang elemen-elemennya disebut sisi sedemikian hingga setiap elemen e dalam E(G) merupakan pasangan tak berurutan dari titik-titik di V(G) disebut himpunan titik G (Ketut, 2007: 1-2). Menurut Munir (2005: 356), graf didefinisikan sebagai pasangan himpunan (V,E), ditulis dengan notasi G = (V,E), yang dalam hal ini adalah himpunan tidak kosong dari titik-titik (vertices atau node) dan adalah himpunan sisi (edges atau arcs) yang menghubungkan sepasang titik, E boleh kosong. Contoh graf dapat dilihat pada gambar 2.1.



Gambar 2.1 Contoh graf

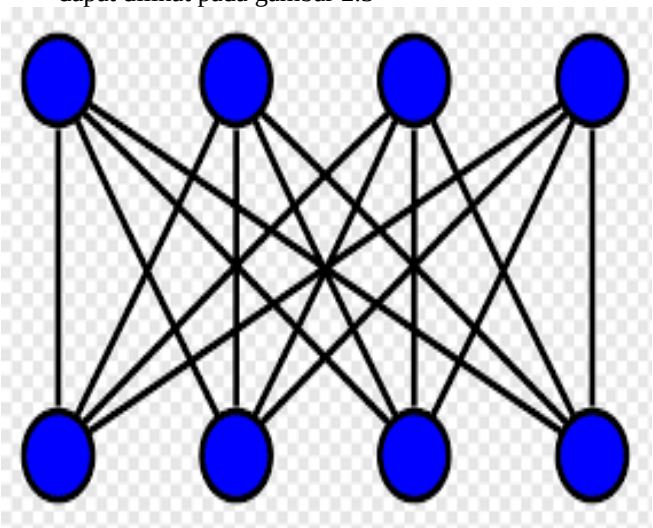
2.2 Graf Bipatrit

Graf bipartit adalah sebuah graf G yang simpulnya dapat dikelompokkan menjadi tepat dua himpunan H1 dan H2 sedemikian hingga tiap anggota H1 terhubung semua dengan tiap anggota H2. Graf tersebut juga dapat dikatakan bahwa tiap simpul yang terdapat di anggota H1 tidak bertetangga, begitu pula simpul yang terdapat di anggota H2. Graf bipatrit tersebut dapat dinyatakan sebagai G(H1, H2). Contoh graf bipatrit terdapat pada gambar 2.2



Gambar 2.2 Graf bipatrit

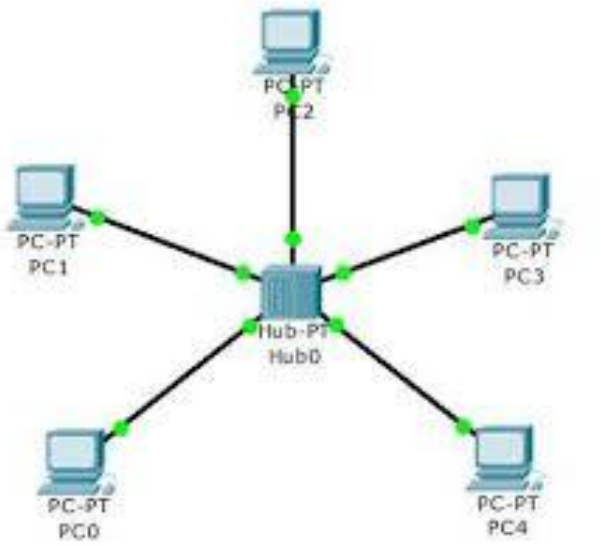
Apabila setiap simpul yang terdapat di anggota H1 bertetangga dengan setiap simpul yang terdapat di anggota H2 begitu pula sebaliknya, maka G(H1, H2) disebut dengan graf bipatrit lengkap. Jika H1 mempunyai jumlah anggota M dan H2 mempunyai jumlah anggota N, maka jumlah sisi graf bipatrit lengkap adalah M×N. Contoh graf bipatrit lengkap dapat dilihat pada gambar 2.3



Gambar 2.3 Graf bipatrit lengkap

Salah satu pemanfaatan graf bipatrit adalah persoalan utilitas. Misalnya ada ada empat rumah R1, R2, R3, dan R4. Masing-masing rumah dihubungkan dengan 4 utilitas yang meliputi listrik (L), air (A), gas (G), dan saluran pembuangan (P). Graf yang mempresentasikan hubungan di atas adalah graf bipatrit. Graf yang terbentuk tersebut sama halnya dengan graf pada gambar 2.3. Contoh lain graf bipatrit adalah topologi bintang (star topologi) pada pembentukan

jaringan komputer. Pada gambar 2.4, H1 dapat direpresentasikan oleh Hub0 dan H2 dapat direpresentasikan oleh PC0 hingga PC4.



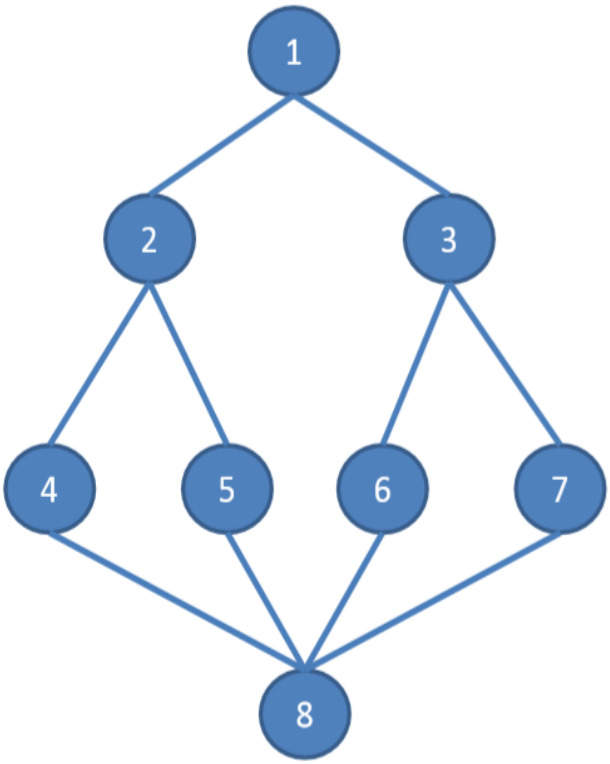
Gambar 2.4 Topologi bintang pada jaringan komputer

2.3 Algoritma

Algoritma adalah suatu langkah atau urutan untuk menghitung atau menyelesaikan suatu masalah secara terstruktur dan sistematis secara berurutan dengan cara sebuah masukan menjadi sebuah keluaran. Algoritma umumnya digunakan dalam pemecahan persoalan dalam matematika atau ilmu komputer. Algoritma muncul karena dalam penyelesaian persoalan dengan instansiasi yang besar akan menjadi lebih sulit dalam menemukan solusinya. Dengan adanya algoritma kita dapat mengukur kinerjanya apakah sebuah program mangkus (efisien atau cepat) dalam menyelesaikan sebuah persoalan. Alat ukur yang digunakan dalam mengukur tingkat kemangkusan algoritma adalah dengan kompleksitas waktu (T(n)) dan kompleksitas ruang (S(n)) dengan n adalah ukuran masukan yang diproses oleh algoritma. Beberapa algoritma yang ada adalah sebagai berikut: algoritma *brute force*, algoritma *greedy*, algoritma *divide and conquer*, algoritma *decrease and conquer*, algoritma *backtracking*, algoritma *branch and bound*, *dan* algoritma *dynamic programming*. Menurut Levitin, 2003, dengan mempelajari strategi algoritma dapat memberikan panduan untuk merancang algortima untuk persoalan baru dan dapat mengklasifikasikan algoritma berdasarkan gagasan perancang yang mendasarinya.

2.4 Algoritma *Breadth First Search*

Algoritma *Breadth First Search* atau BFS adalah algoritma transversal graf yang mengunjungi simpul dengan cara pencarian melebar yang sistematik. Algoritma *Breadth First Search* merupakan sebuah algoritma pencarian tanpa menggunakan informasi tambahan. Representasi algoritma *Breadth First Search* ini dapat dinyatakan dalam graf statis yaitu graf yang sudah terbentuk sebelum proses pencarian dilakukan dengan graf direpresentasikan sebagai sebuah struktur data ataupun graf dinamis yaitu graf yang terbentuk saat proses pencarian dilakukan (graf tidak tersedia sebelum pencarian dilakukan). Skema pencarian melebar adalah transversal dimulai dari simpul v, lalu mengunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu. Kemudian kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang telah dikunjungi, dan demikian seterusnya. Untuk lebih jelasnya, dapat dilihat pada gambar 2.5 dan 2.6



Gambar 2.5 Contoh Graf untuk penyelesaian BFS

Iterasi	V	Q	dikunjungi							
			1	2	3	4	5	6	7	8
Inisialisasi	1	{1}	T	F	F	F	F	F	F	F
Iterasi 1	1	{2,3}	T	T	T	F	F	F	F	F
Iterasi 2	2	{3,4,5}	T	T	T	T	T	F	F	F
Iterasi 3	3	{4,5,6,7}	T	T	T	T	T	T	T	F
Iterasi 4	4	{5,6,7,8}	T	T	T	T	T	T	T	T
Iterasi 5	5	{6,7,8}	T	T	T	T	T	T	T	T
Iterasi 6	6	{7,8}	T	T	T	T	T	T	T	T
Iterasi 7	7	{8}	T	T	T	T	T	T	T	T
Iterasi 8	8	{}	T	T	T	T	T	T	T	T

Urutan simpul2 yang dikunjungi: 1, 2, 3, 4, 5, 6, 7, 8

Gambar 2.5 Iterasai penggunaan algoritma BFS pada gambar 2.6

Kompleksitas waktu dan ruang untuk algoritma Breadth First Search adalah O(bd)dengan keterangan b adalah branching factor yaitu maksimum percabangan yang mungkin dari suatu simpul d adalah depth yaitu kedalaman dari solusi terbaik.

III. PEMBAHASAN

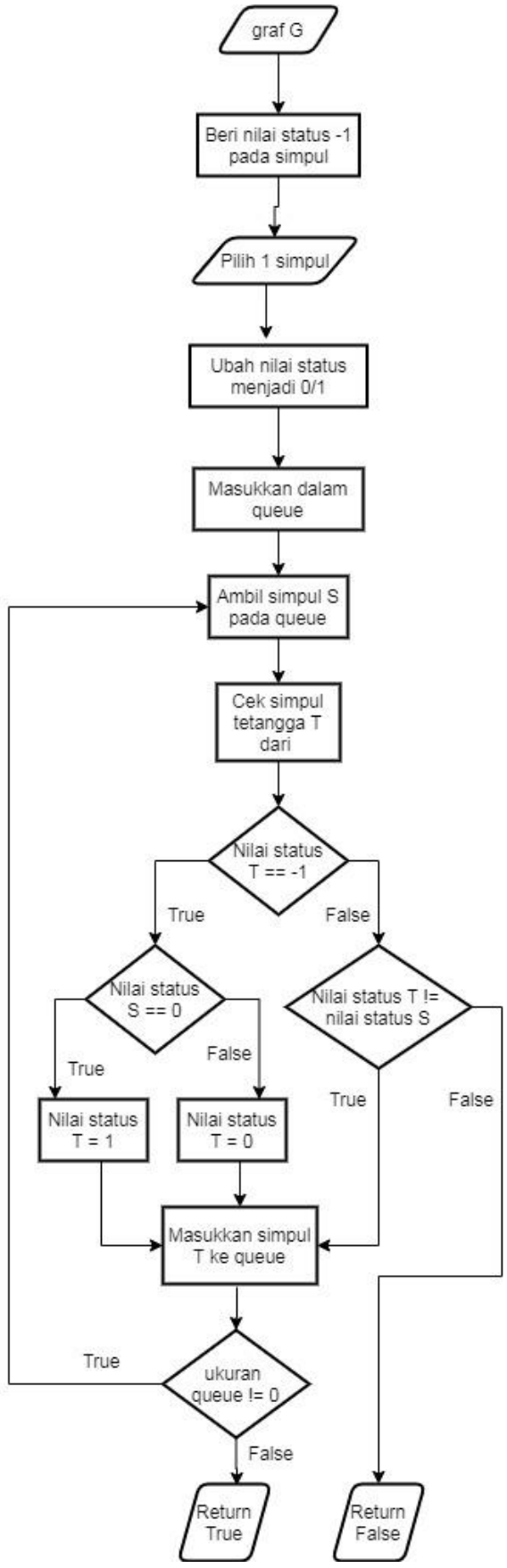
Dalam menentukan apakah sebuah graf termasuk graf bipatrit atau tidak dapat dilakukan menggunakan algoritma *Breadth First Search*. Sebelum kita mengecek menggunakan algoritma *Breadth First Search*, dapat diperhatikan syarat graf bipatrit berikut:

1. Sebuah graf adalah bipatrit jika dan hanya jika dapat direpresentasikan menjadi 2 hal berbeda (misal 0 /1 atau warna merah/biru);
2. Sebuah graf adalah bipatrit jika dan hanya jika tidak memiliki jumlah simpul ganjil yang membentuk graf lingkaran.

Berikut algoritma *Breadth First Search* yang digunkan unutk menentukan apakah sebuah graf adalah sebuah graf bipatrit:

1. Berikan nilai status -1 untuk semua simpul yang ada di graf G;
2. Ambil secara sembarang satu buah simpul dari graf G dan ubah nilai statusnya menjadi 0 atau 1 (pilih salah satu);
3. Masukkan simpul yang diambil secara acak ke dalam *queue;*
4. Selama *queue* tidak kosong, ambil sebuah simpul dari *queue*. Misalkan simpul yang telah diambil adalah simpul S;
5. Cek pada seluruh simpul tetangga dari simpul S
   a. Apabila simpul tetangga tersebut memiliki nilai status -1
      i. Apabila simpul S memiliki nilai status 0, maka nilai status simpul tetangga tersebut adalah 1 dan simpul tetangga tersebut dimasukkan ke dalam *queue*;
      ii. Apabila simpul S memiliki nilai status 1, maka nilai status simpul tetangga tersebut adalah 0 dan simpul tetangga tersebut dimasukkan ke dalam *queue*;
   b. Apabila simpul tetangga tersebut memiliki nilai status 0 atau 1
      i. Apabila nilai status simpul S berbeda dengan nilai status simpul tetangga, maka simpul tetangga tersebut dimasukkan ke dalam *queue* dan ulangi langkah 4 dan langkah 5 hingga *queue* kosong atau sudah menghasilkan nilai False
      ii. Apabila nilai status simpul S sama dengan nilai status simpul tetangga, maka akan mengembalikan nilai *False* atau graf G adalah bukan graf biparit;
6. Apabila semua simpul pada graf G tidak ada yang mempunyai nilai status -1, maka akan mengembalikan nilai *True* atau graf G adalah sebuah graf bipatrit.

Dengan penjabaran algoritma di atas, keluaran yang akan dihasilkan cukup mangkus dan akurat akan tetapi apabila terdapat komponen graf yang tidak terhubung maka tidak dapat melakukan pengecekan pada komponen yang tidak terhubung tersebut.



Gambar 3.1 Diagram alir proses BFS untuk penentuan graf bipatrit

Berikut adalah penguraian algoritma *Breadth First Search* dalam notasi algoritmik :

```
Procedure BipatriteCheck (input G : graf,
output bipatrit : boolean)
Deklarasi
   GM : map of graf {mapping simpul dengan
   nilai status}
   Q : queue
   S : simpul {simpul-S}
   T : simpul {simpul tetangga}
Algoritma
   for i←1 to (jumlah simpul G)
       GM←GM+(G[i],-1)
   GM[0]←(GM[0].tupple1, 0)
   Q←Q+GM[0]
   While(Q.size!=Null)
      S←GM[0]
      While(masih ada simpul tetangga yang
   belum dipilih dari simpul S dan bipatrit
   = false)
         Q←Q-S
         T←(simpul yang belum pernah dipilih
   dari simpul S)
         If(T.tupple2=-1):
            If(S.tupple2=0):
               GM.tupple2[T]←1
            Else:
               GM.tupple2[T]←0
            Q←Q+T
         Else:
            If(T.tupple2!=S.tupples2):
               Q←Q+T
               Break
            Else:
               Return false
   Return true
```

Dengan adanya notasi algoritmik di atas dapat dilihat bahwa terjadi pengulangan sebanyak dua kali. Akan tetapi apabila

diteliti lebih lanjut, pengulangan ini dilakukan bertahap dan hanya memilih simpul yang belum pernah dikunjungi itu. Oleh sebab itu setiap sisiakan ditelusuri tepat hanya satu kali. Sedangkan kompleksitas dari penggunaan algoritma *Breadth First Search* dalam penentuan graf bipatrit adalah O(m) dengan m adalah jumlah sisi pada graf tersebut.

IV. Ucapan Terima Kasih

Puji syukur kepada Allah SWT. karena penulis dapat menyelesaikan tugas makalah matematika diskrit ini. Terima kasih kepada seluruh dosen pengampu strategi algoritma atas ilmu yang telah diberikan. Terima kasih juga kepada keluarga dan teman-teman yang selalu meng-support penulis dalam menjalani perkuliahan pada semester ini.

V. Tautan Video pada Youtube

Video untuk pemaparan makalah ini dapat diakses pada link youtube sebagai berikut.

https://youtu.be/lm9uVoS5dU4

Video tersebut berjudul "Aplikasi Penentuan Graf Bipatrit Menggunakan Algoritma BFS"

VI. Daftar Pustaka

[1] Budayasa, I.K. 2007. Teori graf dan Aplikaisnya. Surabaya :Unesa University Press.
[2] Munir, R. 2005. Matematika Diskrit. Bandung: Informatika.
[3] http://informatika.stei.itb.ac.id/~rinaldi.munir/, diakses pada 3 Mei 2020
[4] https://algorithms.tutorialhorizon.com/check-if-graph-is-bipartite-adjacency-list-using-breadth-first-searchbfs/, diakses pada 3 Mei 2020
[5] https://www.techiedelight.com/bipartite-graph/, diakses pada 3 Mei 2020

VII. 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.

Surakarta, 3 Mei 2020

Syarifuddin F A
13518095