Pembuatan Timeline Penayangan Film Marvel Cinematic Universe dengan Menggunakan Algoritma Breadth-First Search (BFS)

Muhammad Aufa Helfiandri-13516008[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13516008@std.stei.itb.ac.id

***Abstract*— Pada zaman sekarang ini Cinematic Univere merupakan sesuatu yang sering digunakan oleh perusahaan perusahaan Enterainment Dunia. Bermula dari *Marvel Cinematic Universe* (MCU) yang merupakan *Cinematic Universe* pertama yang mengadaptasi komik dengan judul yang sama, *Cinematic Universe* pada pengertiannya merupakan film film bioskop yang saling berhubungan antar filmnya. Dengan setiap bagian filmnya yang berhubungan tentunya pembuatan timeline penayangan untuk Film yang berbasis *Cinematic Universe* akan lebih sulit dibandingkan dengan film biasa. Setiap Film harus ditayangkan setelah semua plotnya dijelaskan oleh film lain jika ada film sebelumnya yang harus dimengerti, sementara itu juga harus dipastikan tidak ada film yang berpotensi memberikan spoiler terhadap film lain yang belum dikeluarkan. Makalah ini akan membahas bagaimana menyusun Timeline pada *Marvel Cinematic Universe* menggunakan Algoritma Breadth-First Search (BFS)**

***Keywords*— *Marvel Cinematic Universe, Cinematic Universe,* Timeline, Spoiler**

I. PENDAHULUAN

Perusahaan Perusahaan Entertaiment besar seperti Disney atau Pixar sering kali menghubungkan tiap filmnya satu sama lain dan Film Film tersebut menjadi sebuah seri besar yang tiap serinya memiliki plot-plot tersendiri yang independen dari plot lainnya. Seri Film ini sering kali disebut sebagai *Cinematic Universe*, yaitu film film berbeda yang ditempatkan dalam 1 semesta yang sama. Karena itu pada *Cinematic Universe* kontinuitas antar film tersebut merupakan sesuatu yang sangat penting. Setiap Film dan Scene Film harus ditempatkan sedemikian rupa sehingga tidak ada Film yang memberikan spoiler untuk film lain yang keluar setelahnya ataupun tidak dapat dimengerti dengan baik karena ada beberapa bagian dari film atau scene film tersebut yang belum dikenalkan pada film film sebelumnya.

Salah satu *Cinematic Universe* yang cukup terkenal adalah *Marvel Cinematic Universe* (MCU) yang berdasarkan kepada Komik Marvel. Dimulai dari Film *Iron Man*, yang dikeluarkan pada tahun 2008 hingga sekarang ini *Marvel Cinematic Universe* telah memiliki 3 Phase film yang terdiri dari 19 Film dan 244 episode seri TV dengan 17 Season. Film terakhir dari *Marvel Cinematic Universe* yang baru baru ini keluar adalah *Avengers : Infinity War* (2018) yang menandai awal dari pengakhiran Phase 3 *Marvel Cinematic Universe* dan generasi film Avengers awal yang bermula dari film *Iron Man* di Phase 1. Marvel sendiri telah memiliki rencana terhadap beberapa film yang ada di Phase 4 yang akan dimulai setelah *Avenger 4* selesai ditayangkan.

Untuk mengatur Semesta Film yang sangat besar seperti *Marvel Cinematic Universe* banyak faktor yang perlu diperhatikan dan tentunya memerlukan teknik khusus untuk mengaturnya. Salah satu cara untuk mengatur Film Film tersebut adalah dengan menggunakan salah satu materi yang diajarkan pada Mata Kuliah Strategi Algoritma, yaitu dengan menggunakan Algoritma Breadth-First Search (BFS).

II. TEORI DASAR

A. *Cinematic Universe*

Source : www.forbes.com
Gambar 2.1 : Poster Film *Avengers Infinity War*

Cinematic Universe pada pengertiannya merupakan Semesta Fiksi yang berisikan Film Film Bioskop yang berhubungan dan ditempatkan di Semesta Film yang sama. *Cinematic Universe* berasal dari Ide fantastis bahwa setiap karakter dari semua film favorit kita bisa saja berada pada Semesta dan Realitas yang sama dan bisa saja pada suatu waktu berinteraksi satu sama lain.

Pada Film-Film yang berada pada sebuah Cinematic Universe, Crossover antar Film merupakan hal yang biasa, sebuah karakter bisa saja menjadi Protagonis pada sebuah film

dan menjadi peran pembantu atau bahkan Antagonis dari film lainnya. Cinematic Univese juga dapat memiliki seri film lagi di dalamnya yang memiliki satu plot lengkap namun tetap berhubungan dengan film film dan seri lain yang berada di Cinematic Universe yang sama.

Marvel Cinematic Universe merupakan Cinematic Universe yang cukup sukses hingga saat ini, Setiap film yang dibuat dalam *Marvel Cinematic Universe* merupakan *Blockbuster* yang menghasilkan Jutaan Dollar dengan sendirinya. *Marvel Cinematic Universe* membagi film filmnya yang dibuatnya menjadi tahapan tahapan yang disebut dengan Phase. Phase 1 bermula dengan Film *Iron Man* (2008) dan berakhir dengan Film *The Avengers* (2012), di dalamnya juga terdapat beberapa film lainnya yaitu *The Incredible Hulk* (2008), *Iron Man 2* (2010), *Thor* (2011), dan *Captain America: The First Avenger* (2011), Phase 1 memiliki plot awal yang berisi pengenalan tiap karakter yang ada, kisah awal mereka, dan motivasi mereka menjadi Superhero.

Phase 2 meliputi 6 Film yang memiliki urutan *Iron Man 3* (2013), *Thor The Dark World* (2013), *Captain America: The Winter Soldier* (2014), *Guardians of The Galaxy* (2014), *Avengers Age of Ultron* (2015) dan *Ant Man* (2015). Phase 2 memiliki Plot yang menjelaskan mengenai musuh musuh yang harus mereka lawan dan Bahaya yang berada di dunia luar.

Phase 3 masih berlangsung hingga sekarang dan memiliki 7 Film yang telah rilis dan 3 lain yang belum. Film film yang sudah rilis adalah *Captain America: Civil War* (2016), *Doctor Strange* (2016), *Guardians of The Galaxy Vol.2* (2017), *Spiderman: Homecoming* (2017), *Thor: Ragnarok* (2017), *Black Panther* (2018) dan *Avengers: Infinity War* (2018). Sementara itu 3 Film yang belum keluar adalah *Ant-Man And The Wasp* (2018), *Captain Marvel* (2019), dan *Avengers 4* (2019) yang judulnya masih belum keluar secara *official*. Phase 3 merupakan kulminasi yang berawal dari Phase 1, pengenalan objek berkekuatan yang disebut sebagai *Infinity Stone* dan munculnya *Thanos*, musuh utama dari *Marvel Cinematic Universe* yang telah dikenalkan sejak *The Avengers*.

Marvel Cinematic Universe juga memiliki seri di dalamanya, beberapa diantaranya yaitu Trilogi *Iron Man* yang terdiri dari *Iron Man, Iron Man 2,* dan *Iron Man 3*. Trilogi *Captain America* yang terdiri dari *Captain America: The First Avenger*, *Captain America: The Winter Soldier,* dan *Captain America Civil War*. Trilogi *Thor* yang terdiri dari *Thor*, *Thor The Dark World,* dan *Thor Ragnarok*. Selain 3 itu juga terdapat seri *Guardians of The Galaxy* yang baru memiliki 2 Film dan Trilogi Avengers yang berfungsi sebagai *Crossover* antar film yang ada pada MCU dan masih berlanjut hingga sekarang.

Hampir setiap karakter pada MCU memiliki cerita mereka sendiri sendiri namun terlepas dari itu Semesta *Marvel Cinematic Universe* juga memiliki cerita lebih besar yang mereka ingin ceritakan. Dengan proyek yang sangat besar seperti itu tentunya MCU memerlukan pengaturan tiap film yang bagus agar walau setiap filmnya independen 1 sama lain dan tidak menimbulkan *Cliffhanger* namun film-film itu juga harus berhubungan dengan satu sama lain

B. Graf dan Pohon

Graf merupakan suatu struktur yang menggambarkan objek objek diskrit dan hubungan antar mereka. Graf terdiri dari 2 unsur penting, yaitu *Vertex* (simpul) dan *Edge* (Sisi). Sebuah Graf G dengan Himpunan Simpul V dan Himpunan Sisi E dapat dituliskan sebagai G = (V, E). [1]

Teori Graf yang dikenal sekarang ini berasal dari seorang matematikawan asal Swiss, *Leonhard Euler* saat ia sedang berada di Konigsberg (Sekarang Kaliningrad) dan menyelesaikan sebuah masalah disitu yang dikenal dengan nama Masalah 7 Jembatan Konigsberg. Masalah tersebut menanyakan apakan seseorang dapat melewati ketujuh Jembatan itu masing masing tepat 1 kali, kemudian kembli ke tempat semula. Pada masa itu setiap orang yang ingin membuktikannya harus langsung mencoba berjalan pada jembatan tersebut.

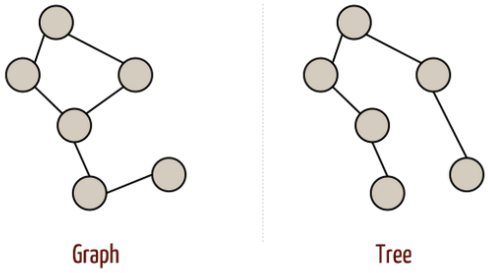
Namun pada tahun 1736, Euler berhasil memodelkan Masalah tersebut ke dalam bentuk graf dan kemudian menemukan sebuah rumus yang dinamakan rumus Euler yang memiliki notasi matematika :

$$V - E + F = 2$$

Dengan V adalah Vertex/Simpul, E adalah Edge/Sisi dan F adalah Face/Muka.

Tree atau Pohon merupakan Bentuk Khusus dari Graf berarah yang sisi berarahnya tidak boleh membentuk loop/kalang baik secara langsung maupun tidak langsung. Simpul pada Tree terbagi menjadi 2 yaitu Parent yang merupakan Simpul yang sedang dikunjungi dan Children yaitu List Simpul simpul yang ditunjuk oleh Parent, Satuan Simpul yang ditunjuk oleh Parent disebut dengan nama Child.

Pada Pohon juga dikenal istilah Root (Akar) yang artinya adalah Simpul di Tree yang tidak memiliki Parent. Pada normalnya Tree hanya memiliki 1 Buah Root namun Bukan tidak boleh sebuah Tree memiliki lebih dari 1 buah Root.



Source : www.quora.com
Gambar 2.2 Perbandingan Graf dengan Pohon

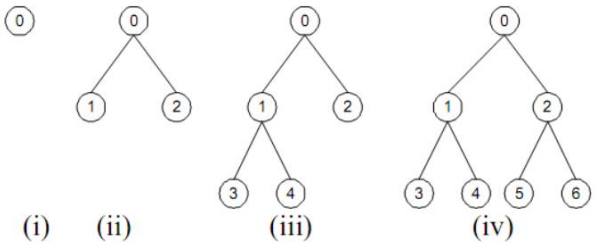
Pada Gambar 2.2 dapat terlihat bahwa pada Gambar Graf Biasa terdapat bagian sisi dan simpul yang memiliki *Loop* sementara pada Tree setiap bagiannya tidak memiliki Loop Sama sekali dan dari Parent kita hanya dapat mengunjungi Child yang berhubungan dengan Parent tersebut.

C. Breadth-First Search (BFS)

BFS (Breadth First Search) merupakan salah satu bentuk dari

Algoritma Traversal Graf, yaitu Algoritma yang bekerja dengan mengunjungi simpul simpul dengan cara yang sistematik dimana tiap persoalan yang akan diselesaikan dimodelkan sebagai Graf yang memiliki Simpul (Node) dan Sisi (Edge). Pada Algoritma Traversal Graf menggunakan metode BFS simpul yang akan dibangkitkan adalah Simpul simpul yang memiliki Level yang sama dan dapat dicapai dari Simpul yang sedang dikunjungi.

BFS melakukan penyelesaian masalah dengan melakukan Pembentukan Ruang Status yang dilakukan secara melebar. Dan dapat digunakan untuk menyelesaikan Graf Statis, yaitu Graf yang sudah terbentuk sebelum proses pencarian dan Graf dinamis, yaitu Graf yang belum terbentuk saat proses pencarian dilakukan.



Source :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/
Gambar 2.3 Pembentukan Ruang Status menggunakan Metode Breadth First Search

Pada Gambar 2.2 dapat terlihat fungsi Pembentukan pohon ruang status dengan Algoritma BFS. Simpul pertama yang dikunjungi adalah simpul 0 yang kemudian akan digunakanuntuk membangkitkan Simpul 1 dan Simpul 2 yang dapat diturunkan dari Simpul 0 dan memiliki level yang sama. Setelahnya Simpul akan membangkitkan simpul 3 dan 4 yang dapat dicapai dari Simpul 1 dan kemudian setelahnya membangkitkan Simpul 5 dan 6 yang dapat dibangkitkan dari Simpul 2.

Algoritma umum untuk pencarian BFS pada Graf Statis adalah sebagai berikut (Traversal bermula dari simpul x) :

1. Mulai kunjungi Graf dari Simpul x
2. Kunjungi semua simpul yang bertetangga dengan simpul 0 terlebih dahulu
3. Kunjungi Simpul yang belum dikunjungi dan bertetangga dengan simpul simpul yang telah dikunjungi sebelumnya

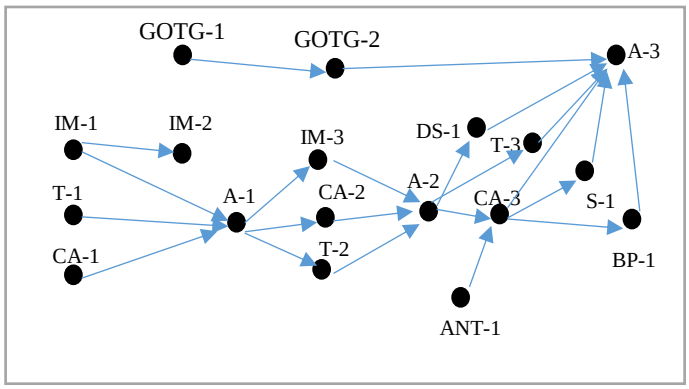
III. PEMBAHASAN TEORI

Sebagai Perusahaan yang mengeluarkan Banyak Film setiap Tahunnya, Marvel Studio pastinya perlu untuk mengatur Film Filmnya agar memiliki *Timeline* yang pas dan tidak membingungkan, mereka juga perlu untuk mengatur film filmnya agar tidak ada film yang terlalu cepat keluar (Memberikan *Spoiler* bagi film yang keluar setelahnya) ataupun membuat filmnya tidak dapat dimengerti karena membutuhkan penjelasan dari Film yang seharusnya ditayangkan sebelumnya.

Source: http://www.slashfilm.com/marvel-movie-plans/
Gambar 3.1 Timeline *Marvel Cinematic Universe* Dari Awal Phase 3 hingga permulaan Phase 4

Untuk menemukan timeline yang Cocok dalam menentukan ururtan Film dalam *Marvel Cinematic Universe* dapat dilakukan beberapa cara, salah satunya adalah dengan memanfaatkan Graf dan menggunakan Algoritma Breadth-First Search (BFS).

Hal Pertama yang perlu dilakukan adalah dengan memodelkan persoalan timeline MCU menjadi sebuah Graf berarah dimana simpul menunjukkan Film yang ingin ditampilkan dan sisi berarah menunjukkan ketergantungan sebuah Film dari Film-Film lainnya.



Gambar 3.3 Illustrasi Graf yang merepresentasikan keterhubungan Film Film pada Marvel Cinematic Universe

Code	Movie Name	Code	Movie Name
IM-1	Iron Man	GOTG-2	Guardians of The Galaxy Vol.2
T-1	Thor	A-2	Avengers Age Of Ultron
CA-1	Captain America: The First Avenger	DS-1	Doctor Strange
IM-2	Iron Man 2	CA-3	Captain America: Civil War
A-1	The Avengers	ANT-1	AntMan
IM-3	Iron Man 3	T-3	Thor Ragnarok
CA-2	Captain America: The Winter Soldier	S-1	Spiderman Homecoming
T-2	Thor The Dark World	BP-1	Black Panther
GOTG-1	Guardians Of The Galaxy	A-3	Avengers Infinity War

Tabel 3.1 Nilai Kode yang tertera pada Graf di Gambar 3.3

Pada Gambar 3.3 telah terbentuk graf yang memodelkan hubungan antara film film yang ada di Marvel Cinematic Universe, setiap sisi yang ada merepresentasikan bahwa simpul

yang ditunjuk tidak boleh ditampilkan sebelum sisi yang menunjuk telah ditayangkan. Sebagai Contoh pada Graf di Gambar 3.3 Film The Avengers tidak boleh ditampilkan sebelum Film Iron Man, Thor, dan Captain America sudah ditayangkan terlebih dahulu namun Film Guardians of The Galaxy boleh saja ditayangkan sebelum Film The Avengers karena tidak adanya sisi yang menghubungkan antara kedua Film.

Berikut ini adalah beberapa Struktur data berupa kelas yang akan digunakan untuk menyelesaikan permasalahan dengan Algoritma BFS dalam Bahasa pemrograman Java dan beberapa method spesial setiap kelas yang akan digunakan nantinya :

```
//Graph.java Package : GraphPack
public class Graph{
    String value;
    ListGraph connections;
    ....
    //Constructor, Getter Setter
    ....
    public boolean isNoPrerequisite(ListGraph){
       .....
    }
}

//ListGraph.java Package : GraphPack
public class ListGraph extends LinkedList<Graph>{
    ...
    //Function to Copy ListGraph And Return
    public ListGraph clone(){
        ....
    }
    //Function to find graphs with no connections
    public ListGraph findNoConnection(){
        ListGraph LG = new ListGraph();
        for (Graph G : this){
           if (!G.isPreRequisite(this)){
              LG.add(G);
           }
        }
        return(LG);
    }
    public ListGraph without(Graph G){
        ListGraph LG = this.clone();
        LG.remove(G);
        return(LG);
    }
}

//ListString.java
public class ListString extends LinkedList<String>{
    ...
    //Function to Clone ListString Ana Return
    public ListString clone(){
       ....
    }
}

//Tree.java
public class Tree<T>{
   T value;
   List<Tree<T>> children
   ....
   //Constructor, Getter Setter
   ....
}
```

Pada Algoritma Java yang saya buat, Terdapat 4 Kelas yang digunakan untuk menyelesaikan permasalahan Timeline Marvel Cinematic Universe. Kelas pertama adalah Kelas Graph, yang berfungsi sebagai representasi Graf. Kelas Graph memiliki 2 buah atribut, yaitu String yang merupakan nilai dari Simpul Graf tersebut dan ListGraph yang merupakan List dari Simpul Simpul yang berkoneksi dengan Simpul Graf tersebut. Kelas Graph ini memiliki 2 buah konstruktor, yaitu konstruktor default yang tidak memiliki parameter dan konstruktor dengan parameter String yang merupakan value dari Graf tersebut. Pada kelas ini juga terdapat getter untuk mengambil nilai dari Graf dan sebuah method isNoPrerequisite yang memiliki sebuah parameter ListGraph dan mengembalikan boolean false jika Prerequisite dari Graf tersebut ditemukan di ListGraph dan boolean true jika tidak ada Graf yang menjadi prerequisite di ListGraph yang menjadi parameter.

Kelas kedua adalah ListGraph yang merupakan turunan dari kelas generic LinkedList yang berada di library java.util.Collections dengan parameter Graph. Kelas ini tidak memiliki Atribut baru selain dari Atribut yang ia dapatkan dari parent LinkedListnya. Extensi di Kelas ListGraph dari kelas LinkedList adalah dengan adanya method clone yang berfungsi untuk mengembalikan ListGraph baru yang merupakan kopian dari semua elemen yang berada pada this (ListGraph lama). Selain itu terdapat 2 buah method lainnya yaitu method findNoConnection dan method without yang dimana method findNoConnection akan mengembalikan ListGraph yang berisi semua Graf di ListGraph yang tidak memiliki prerequisite dengan memanfaatkan method isNoPrerequisite yang berada pada kelas Graf. Method Without akan mengembalikan ListGraph yang dikurangi dengan Graf yang menjadi parameter method.

Kelas ketiga adalah kelas ListString yang merupakan turunan dari kelas Generic LinkedList dengan parameter String dan memiliki method clone yang berfungsi untuk mengkopi objek ListString menjadi objek ListString baru.

Kelas Terakhir adalah kelas Tree Generic yang memiliki 2 buah parameter, yaitu parameter generik value yang merupakan nilai dari parent Tree yang sedang dikunjungi dan parameter Children yang merupakan list child yang dimiliki oleh parent. Pada kelas ini terdapat method toList yang akan mengubah Tree menjadi List kemungkinan dari setiap value yang disimpan pada Tree.

Dapat terlihat pada Gambar 3.3 Graf yang terbentuk sudah juga terbentuk sebelum Algoritma digunakan sehingga yang kita lakukan selanjutnya adalah menggunakan Algoritma BFS untuk Graf Statis. Karena itu Langkah-Langkah menentukan urutan Film dengan Algoritma BFS adalah sebagai berikut :

1. Jika sisa di Queue Graph kosong maka kembalikan Tree yang berisi nama Film yang tersisa tersebut
2. Selainnya maka lakukan :
    a. Cari semua Graph di List Graph yang merupakan Root (Tidak memiliki Prerequisite yang ada di ListGraph) dan assign ke sebuah variabel misalnya LG
    b. Untuk semua Graph yang berada di dalam List Graph maka lakukan :

i. Clone list String yang merupakan parameter dari fungsi pembuatan pohon status
ii. Panggil kembali fungsi Pembuat Pohon secara rekursif dengan parameter List Graph yang tidak mengandung Graph yang sedang diperiksa
iii. Masukkan hasil yang didapatkan dari fungsi rekursif ke pohon penyelesaian

c. Kembalikan pohon yang dibentuk

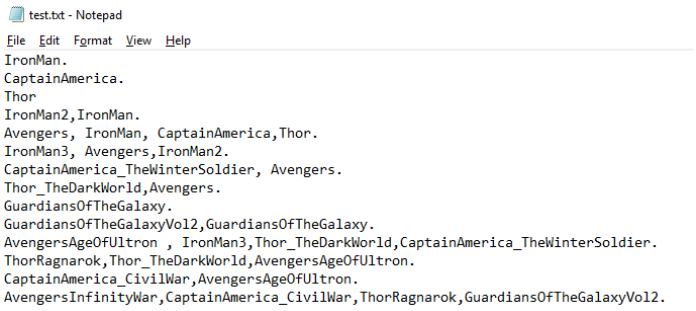
3. Simpan semua bagian pohon yang terbentuk di Layar dalam List Urutan dan kembalikan List kemungkinan urutan
4. Print hasil List Urutan tersebut

Sebagai contoh implementasi, saya membuat Algoritma penyelesaian masalah timeline yang sudah dimodelkan dalam Graf tersebut dalam Bahasa Pemrograman Java yang dapat dilihat dibawah ini

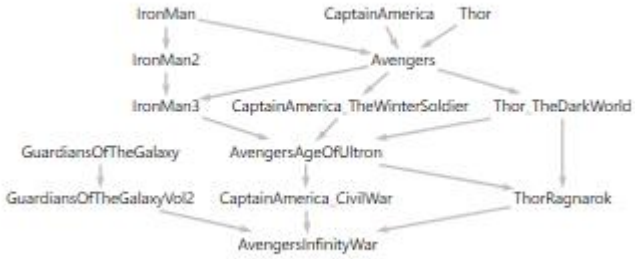
```
//Drive.java
import GraphPack.Graph;
import GraphPack.ListGraph;

public class Drive{
...
   public static Tree<ListString> BFSTree(ListGraph LG,
ListString LS){
     if (LG == null){
        return(new Tree<ListString>(LS));
     }
     else if (LG.size() == 0){
        return(new Tree<ListString>(LS));
     }
     else{
        ListGraph LGNC = LG.findNoConnection();
        Tree<ListString> TRS;
        TRS = new Tree<ListString>(LS);
        ListString LSTemp = new ListString();
        for (Graph G : LGNC){
            LSTemp = LS.clone();
            LSTemp.add(G.getValue();
            TRS.addChild(BFSTree(LG.without(G),LSTemp));
        }
        return(TRS);
     }
   }
   .... //Fungsi Main
}
```

Pada program yang saya buat pada awalnya program akan mengecek apakah ListGraph yang menjadi parameter kosong, jika iya maka program akan langsung mengembalikan Tree baru yang berisikan ListString yang menjadi parameter fungsi. Jika tidak maka Program akan menginisiasi variabel LGNC dengan semua Graf di ListGraph LG yang tidak memiliki prerequisite dengan memanggil method findNoConnection yang dimiliki oleh kelas ListGraph.

Setelah itu program akan membuat Tree baru dengan memanggil konstruktor dengan parameter yang berisikan ListString dari parameter fungsi kemudian mengassignnya ke variabel TRS. Selanjutnya program akan masuk ke dalam loop untuk semua Graf yang terdapat pada ListGraph LGNC dimana untuk setiap Graf yang berada pada ListGraph LGNC akan dibuat Tree baru sebagai anak dari TRS yang merupakan hasil dari pemanggilan fungsi secara rekursif dengan parameter ListGraph LG tanpa Graf yang sedang diperiksa dan LSTemp yang merupakan kopian dari ListString parameter yang ditambah dengan value dari Graf yang sedang diperiksa.



Gambar 3.4 Sample Input



Gambar 3.5 Graf yang dibuat oleh Input pada Gambar 3.4

Menggunakan program yang saya buat dengan menggunakan input pada gambar 3.4 output yang saya terima adalah sebagai berikut :

[IronMan , CaptainAmerica , Thor , IronMan2 , Avengers , IronMan3 , CaptainAmerica_TheWinterSoldier , Thor_TheDarkWorld , GuardiansOfTheGalaxy , GuardiansOfTheGalaxyVol2 , AvengersAgeOfUltron , ThorRagnarok , CaptainAmerica_CivilWar , AvengersInfinityWar]

[IronMan , CaptainAmerica , Thor , IronMan2 , Avengers , IronMan3 , CaptainAmerica_TheWinterSoldier , Thor_TheDarkWorld , GuardiansOfTheGalaxy , GuardiansOfTheGalaxyVol2 , AvengersAgeOfUltron , CaptainAmerica_CivilWar , ThorRagnarok , AvengersInfinityWar]

[IronMan , CaptainAmerica , Thor , IronMan2 , Avengers , IronMan3 , CaptainAmerica_TheWinterSoldier , Thor_TheDarkWorld , GuardiansOfTheGalaxy , AvengersAgeOfUltron , GuardiansOfTheGalaxyVol2 , ThorRagnarok , CaptainAmerica_CivilWar , AvengersInfinityWar]

[IronMan , CaptainAmerica , Thor , IronMan2 , Avengers , IronMan3 , CaptainAmerica_TheWinterSoldier , Thor_TheDarkWorld , GuardiansOfTheGalaxy , AvengersAgeOfUltron , GuardiansOfTheGalaxyVol2 , CaptainAmerica_CivilWar , ThorRagnarok , AvengersInfinityWar]
... dan beberapa ribu output lainnya.

V. SIMPULAN

Pada perusahaan yang membuat film bertipe Cinematic Universe, salah satunya adalah *Marvel Studios* kontinuitas dari timeline merupakan salah satu hal yang cukup penting dan harus dijaga agar film mereka dapat laku di pasaran. Mereka harus menjaga agar film yang dirilis tepat waktu dirilisnya tidak terlalu cepat dan tidak terlalu lambat.

Penentuan Timeline perilisan film Cinematic Universe dapat dilakukan dengan menggunakan Algoritma BFS dengan masukan ketergantungan antar film kepada film yang lainnya. Tentunya penggunaan Algoritma ini tidak sepenuhnya sempurna, masih banyak bagian pertimbangan bisnis yang tidak diperhitungkan Algoritma BFS tersebut, namun Algoritma ini dapat digunakan sebagai pengontrol timeline agar tidak ada film yang *loss*. Yang tentunya merupakan satu hal penting bagi perusahaan entertaiment raksasa seperti Disney dan Marvel.

VII. UCAPAN TERIMA KASIH

Pertama tama Penulis berterima kasih kepada Allah Subhanahu Wa Ta'ala yang atas Rahmat dan Hidayahnya Penulis dapat memulai dan menyelesaikan Makalah ini. Selanjutnya kepada Tim Dosen Strategi Algoritma, Bapak Rinaldi, Ibu Ulfa, dan Ibu Masayu yang telah mengajarkan materi materi mengenai Strategi Algoritma terutama mengenai Algoritma BFS yang digunakan dalam Makalah ini. Terakhir Penulis juga mengucapkan terimakasih kepada Orang tua Penulis yang memberi dukungan terhadap Penulis dalam menjalani Proses Perkuliahan dan Proses Penyelesaian Makalah ini.

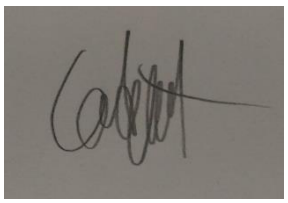
REFERENCES

[1] Munir, Rinaldi (2003) "Matematika Diskrit Revisi Keenam"
[2] Munir, Rinaldi (2009) "Diktat Kuliah IF2211 Strategi Algoritma. Bandung :Institut Teknologi Bandung
[3] Marvel.com
[4] https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
[5] https://medium.com/basecs/going-broad-in-a-graph-bfs-traversal-959bd1a09255

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, 3 Desember 2017



Muhammad Aufa Helfiandri - 13516008