Aplikasi Pohon Berakar dalam Pembuatan Kisah Interaktif

Stefanus Jeremy Aslan - 13519175[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13519175@std.stei.itb.ac.id

***Abstract*—Untuk perencanaan dan penggalian ide, bantuan visualisasi dapat mempermudah individu dalam proses tersebut. Salah satu bantuan visualisasi yang umum digunakan yaitu dengan menggunakan graf.**

Dengan menggunakan graf, individu menjadi lebih mudah dalam mengaitkan skenario yang satu dengan skenario yang lain serta menyimpan skenario-skenario tersebut dalam bentuk tertulis. Dalam aktivitas seperti pembuatan cerita yang memerlukan perencanaan dan penggalian ide yang luas dan mendalam, bantuan graf menjadi penting untuk hasil yang lebih baik dan efisien, terutama kisah interaktif yang benar-benar harus menelusuri segala kemungkinan yang akan ditulis. Akan tetapi, tidak semua graf dapat membantu penulis dengan efisien dalam dalam pembuatan cerita, beberapa bahkan memperumit penulis. Salah satu graf yang ideal dalam membantu penulisan kisah adalah graf pohon berakar karena strukturnya dari atas ke bawah, awal mula yang berasal dari satu tempat, dan elemen yang tersusun rapih sehingga cocok untuk menyusun skenario.

Dalam makalah ini, peran pohon berakar dalam perencanaan dan penggalian ide akan ditelusuri lebih lanjut dengan mengaplikasikan penggunaan graf dalam pembuatan cerita interaktif sederhana menggunakan bahasa pemrograman.

***Keywords*—graf, pohon berakar, skenario, kisah, perencanaan.**

I. PENDAHULUAN

Dalam kehidupan sehari-hari, menyusun rencana merupakan tindakan vital. Dengan rencana, individu dapat mengarahkan dirinya ke keadaan atau skenario yang dianggap ideal. Akan tetapi, tidak semua rencana dapat berjalan dengan lancar yang menyebabkan individu mengalami kesulitan dalam menghadapi situasi yang tidak terduga. Dalam menghadapi situasi tersebut, individu dapat menyusun rencana cadangan dengan basis berupa prediksi terhadap berbagai skenario lain yang mungkin terjadi mempertimbangkan *cause and effect* skenario, dan bagaimana cara mendapatkan atau menghindari skenario tersebut.

Tidak hanya untuk menyusun rencana cadangan dalam kehidupan nyata, pola pikir yang mempertimbangkan berbagai ragam skenario juga menjadi komponen penting dalam menyusun suatu kisah fiktif, terutama kisah interaktif yang alurnya bercabang tergantung dari keputusan pengguna. Dalam menyusun kisah interaktif, kisah yang alurnya berdasarkan keputusan pembaca, sang penulis didorong untuk berimajinasi luas, memikirkan segala macam skenario yang dapat dilalui oleh tokoh cerita yang merepresentasikan pembaca, dan mengembangkan skenario tersebut lebih lanjut menjadi skenario-skenario lain hingga mencapai suatu *ending*. Semakin panjang cerita, semakin banyak jumlah skenario yang harus dirancang oleh penulis yang menyebabkan tingginya kebutuhan akan metode untuk mengorganisir skenario-skenario tersebut sehingga lebih mudah untuk dikembangkan lebih lanjut.

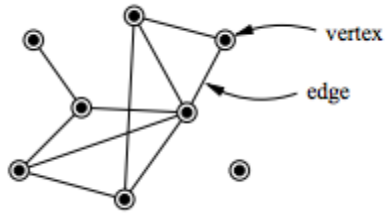
Dalam menghadapi permasalahan, dipilih solusi berupa graf pohon berakar. Graf pohon berakar dapat dengan akurat menggambarkan hubungan skenario-skenario dalam suatu kisah interaktif dengan akar sebagai skenario awal (*starting scenario*) dan skenario-skenario lain yang terhubung dengannya sebagai skenario penerus atau successor. Dengan alasan tersebut, pohon berakar dapat menjadi solusi yang ideal dalam menghadapi permasalahan.

Untuk menerapkan graf lebih lanjut dalam penyelesaian permasalahan, akan digunakan bahasa pemrograman C sebagai bahasa untuk menciptakan program yang menggambarkan alur dari kisah interaktif.

II. LANDASAN TEORI

A. Graf

Graf merupakan metode pemetaan objek-objek diskrit dengan satu sama lain melalui garis hubung. Dalam graf, objek diskrit disebut dengan simpul (*vertex*) dan garis hubung yang menggambarkan relasi objek diskrit disebut sisi (*edge*).



Gambar 2.1 Graf, simpul (vertex), dan sisi (edge)

Sumber: https://en.wikipedia.org/wiki/Vertex_(graph_theory)#/media/File:Small_Network.png ; diakses pada 9 Desember 2020, pukul 16.44 WIB

Graf dapat umumnya didefinisikan menggunakan tupel dua buah elemen:

$$G = (V, E)$$

Dengan V merupakan himpunan simpul dalam graf G dan E merupakan himpunan sisi dalam graf G.

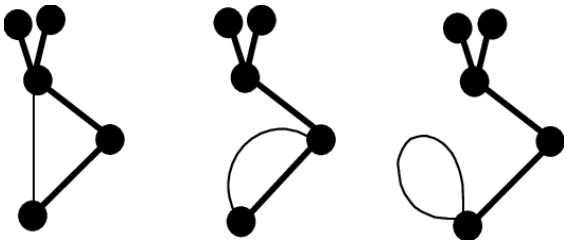
Untuk sisi dalam graf, terdapat istilah sisi gelang dan sisi ganda. Sisi gelang merupakan sisi yang mengubungkan simpul dengan simpul itu sendiri sehingga membentuk cincin atau gelang sedangkan sisi ganda merupakan sebutan untuk sisi-sisi yang menghubungkan 2 buah simpul yang sama.

Secara umum, graf diklasifikasikan berdasarkan dua kriteria, yaitu:

1. Eksistensi sisi gelang dan sisi ganda
2. Orientasi arah pada sisi.

Berdasarkan eksistensi sisi gelang dan sisi ganda, graf dibedakan menjadi dua:

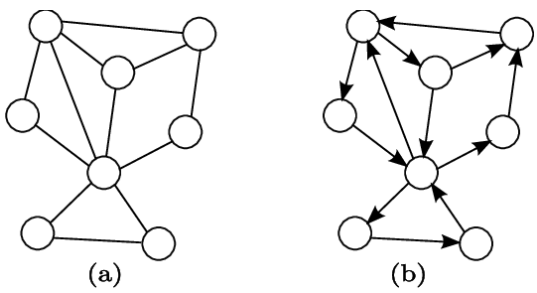
1. Graf sederhana
   Graf yang tidak mengandung baik sisi gelang maupun sisi ganda sebagai sisi penyusunnya.
2. Graf tak-sederhana
   Graf yang mengandung sisi gelang atau sisi ganda sebagai sisi penyusunnya. Graf tak-sederhana dibagi menjadi dua: graf ganda dan graf semu. Graf ganda (*multigraf*) merupakan graf yang memiliki sisi ganda, tetapi tidak memiliki sisi gelang; graf semu (pseudo graph)merupakan graf yang dapat memiliki sisi ganda dan sisi gelang.



Gambar 2.2 Dari kiri ke kanan: graf sederhana, graf ganda, graf semu
Sumber: https://www.researchgate.net/figure/Examples-of-graph-A-multigraph-B-and-pseudograph-C-with-cyclomatic-number-C-1_fig3_209529467 ; diakses pada 9 Desember 2020, pukul 15.13 WIB

Berdasarkan orientasi arah pada sisi, graf dibedakan menjadi dua, yaitu:

1. Graf berarah
   Graf yang setiap sisinya memiliki arah yang umumnya secara visual digambarkan dengan kepala panah.
2. Graf tak berarah
   Graf yang setiap sisinya tidak memiliki arah.



Gambar 2.3 Graf berarah dan graf tidak berarah
Sumber: https://www.researchgate.net/figure/a-An-example-of-undirected-graph-and-b-an-example-of-directed-graph_fig3_50591619 ; diakses pada 9 Desember 2020, pukul 16.20 WIB

Dalam mengidentifikasi graf lebih lanjut, graf memiliki terminologi-terminologi sebagai berikut.

1. Ketetanggaan (*adjacent*)
   Ketetanggaan atau *adjacent* merupakan relasi antara dua buah simpul yang berdasarkan keterhubungan dua simpul dengan sisi.
2. Bersisian (*incidency*)
   Bersisian atau *incidency* merupakan relasi sisi dengan simpul. Sebuah sisi dikatakan bersisian dengan sebuah simpul apabila sisi tersebut menghubungkan simpul yang dimaksud dengan simpul lain atau dengan simpul itu sendiri
3. Simpul terpencil (*isolated vertex*)
   Simpul terpencil atau *isolated vertex* adalah sebutan bagi simpul yang sama sekali tidak memiliki sisi penghubung.
4. Graf kosong (*empty graph*)
   Graf kosong atau *empty graph* merupakan istilah untuk graf dengan himpunan sisi berupa himpunan kosong (E = {}), sehingga setiap simpul dalam graf dapat dikatakan sebagai simpul terpencil.
5. Derajat (*degree*)
   Derajat atau *degree* merupakan jumlah sisi yang terhubung dengan suatu simpul. Derajat suatu simpul dinotasikan sebagai: d(v), dengan v merupakan nama suatu simpul tersebut. Untuk graf berarah, derajat dibagi menjadi dua, yaitu derajat masuk dan derajat keluar. Derajat masuk merupakan jumlah sisi yang mengarah ke dalam simpul, sedangkan derajat keluar merupakan jumlah sisi yang mengarah ke luar simpul.
6. Lintasan (*path*)
   Lintasan atau *path* merupakan rantaian simpul yang saling terkoneksi secara urut melalui sisi. Lintasan dapat dikemukakan menggunakan tupel berelemen simpul yang dinyatakan secara terurut dengan simpul yang bersampingan dihubungkan melalui sisi. Lintasan juga dapat dikemukakan menggunakan tupel berelemen simpul dan sisi yang sama dengan tupel berelemen simpul, tetapi di antara setiap simpul dilengkapi dengan sisi yang menghubungkan simpul yang mengapit. Panjang suatu lintasan merupakan jumlah total sisi dalam lintasan.
7. Sirkuit (*circuit*)
   Sirkuit atau *circuit* merupakan lintasan yang memiliki simpul awal dan simpul akhir yang sama.
8. Keterhubungan (*connected*)
   Keterhubungan merupakan sebutan relasi untuk dua buah simpul yang terhubung melalui suatu lintasan dalam graf. Suatu graf disebut sebagai graf terhubung apabila setiap simpul dalam graf terhubung dengan semua simpul dalam graf
9. Upagraf (*subgraph*)
   Upagraf atau *subgraph* graf yang merupakan bagian dari graf lain. Suatu graf $G_1 = (V_1,E_1)$ merupakan upagraf dari graf G = (V,E) apabila V_1 merupakan himpunan bagian dari V dan E_1 merupakan himpunan bagian dari E.
10. Komplemen Upagraf
    Komplemen upagraf merupakan graf pelengkap dari graf yang merupakan upagraf dari graf G apabila gabungan himpunan simpul dan himpunan sisi milik upagraf

tersebut dengan komplemen upagraf tersebut menghasilkan graf G.

11. Komponen
    Komponen merupakan jumlah maksimum upagraf terhubung dari suatu graf.
12. *Cut set*
    Cut set merupakan himpunan sisi graf dengan jumlah minimum yang akan menyebabkan graf terhubung menjadi graf tidak terhubung apabila sisi yang terdapat dalam himpunan tersebut dihilangkan dari graf.
13. Graf berbobot (*weighted graph*)
    Graf berbobot atau *weighted graph* merupakan graf yang setiap sisinya diberikan nilai atau bobot tertentu.

Selain graf-graf tersebut, terdapat beberapa sebutan khusus untuk graf dengan sifat tertentu, yaitu:

1. Graf Lengkap
   Graf lengkap merupakan graf sederhana yang setiap simpulnya memiliki sisi-sisi yang menghubungkan simpul dengan semua simpul graf lainnya.
2. Graf Lingkaran
   Graf lingkaran merupakan graf yang setiap simpulnya memiliki derajat dua.
3. Graf Teratur
   Graf teratur merupakan graf yang setiap simpulnya memiliki derajat yang sama dan memenuhi persamaan berikut:

$$Jumlah\ Sisi = \frac{Jumlah\ Simpul\ \times Derajat\ Simpul}{2}$$

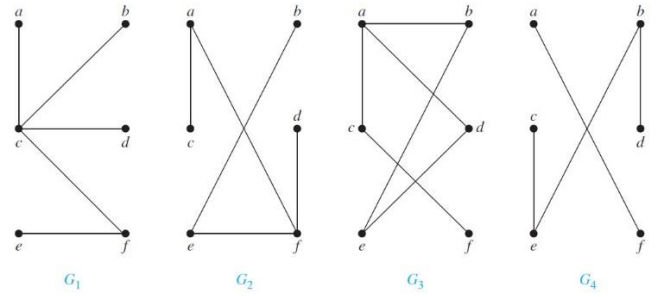
B. Pohon

Pohon atau *Tree* merupakan graf sederhana terhubung yang tidak berarah serta tidak memiliki sirkuit. Maka dari itu, setiap pohon tidak memiliki sisi ganda, tidak memiliki sisi gelang, tidak memiliki komponen lebih dari satu, dan tidak memiliki upagraf yang berupa graf lingkaran.

Berdasarkan fakta tersebut, suatu graf dapat dikatakan memenuhi syarat sebagai pohon apabila memenuhi semua syarat berikut:

1. Setiap simpul terhubung dengan simpul lain melalui satu lintasan saja.
2. Jumlah sisi ekivalen dengan jumlah simpul dikurang satu.
3. Berupa graf terhubung
4. Tidak mengandung sirkuit
5. Penambahan satu sisi pada pohon hanya membuat satu sirkuit

Sebagai contoh, dapat dilihat empat buah graf sebagai pada Gambar 1.1 sebagai berikut



Gambar 2.4 Contoh Pohon dan Bukan Pohon

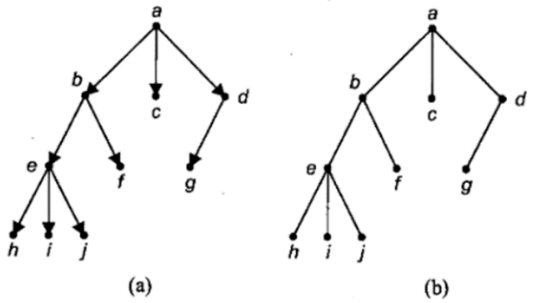
Sumber : https://www.haimatematika.com/2018/12/graf-pohon-teori-graf.html ; diakses pada 9 Desember 2020, pukul 14.46 WIB.

Pada gambar, graf paling kiri, graf yang dinamakan G1, memenuhi syarat sebagai pohon, sama halnya dengan Graf kedua dari paling kiri yang dinamakan G2. Untuk graf kedua dari paling kanan yang dinamakan G3, graf tersebut tidak memenuhi syarat sebagai pohon karena mengandung sirkuit: a,b,c,d,a. Pada graf paling kanan yang dinamakan G4, graf tersebut merupakan graf tidak terhubung yang memiliki dua komponen, yaitu komponen yang memiliki himpunan simpul terhubung {a,f} dan komponen yang memiliki himpunan simpul terhubung {c,e,b,d}.

Dalam praktiknya, pohon dibagi menjadi beberapa tipe. Salah satu tipe yang umum digunakan yaitu pohon berakar.

C. Pohon Berakar

Dasarnya, pohon berakar merupakan graf berarah dengan karakteristik semua simpul memiliki derajat masuk bernilai 1 kecuali satu simpul yang berperan sebagai akar, simpul yang memiliki derajat masuk bernilai 0. Karena pohon berakar umumnya digambarkan mengarah dari akar ke simpul di bawahnya dan memiliki sifat sama dengan graf pohon kecuali sisi yang berarah, pohon berakar dapat dinyatakan menggunakan sisi tanpa arah dengan perjanjian semua sisi secara kaidahnya mengarah ke bawah. Oleh karena itu, pohon berakar dapat dinyatakan sebagai graf pohon atau tree.



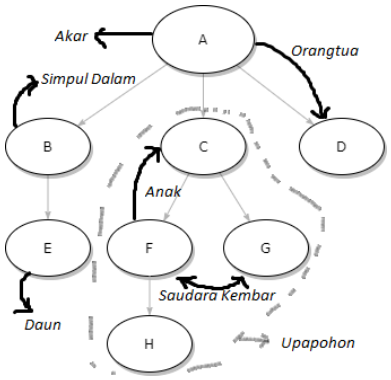
Gambar 2.5 (a) Pohon berakar (b) Pohon berakar tanpa sisi yang diberikan arah
Sumber: http://poetra70.blogspot.com/2015/09/pohon-matematika-diskrit.html ; diakses pada 9 Desember 2020, pukul 16.04 WIB

Pada pohon berakar, terdapat terminologi-terminologi sebagai berikut:

1. Orangtua (*parent*)
   Orangtua atau *parent* dalam pohon berakar merupakan sebutan untuk relasi ketetanggan antara simpul dengan simpul yang secara visual di tempatkan di bawah simpul tersebut. Sebagai contoh, pada gambar 2.5 (b), simpul a merupakan simpul yang bertetangga dengan simpul yang secara visual ditempatkan di bawahnya, yaitu simpul b, c, dan d. Maka dari itu, simpul a orangtua simpul b, c, dan d.
2. Anak (*child*)
   Anak atau *child* dalam pohon berakar merupakan sebutan untuk relasi ketetanggan antara simpul dengan simpul yang secara visual di tempatkan di atas simpul tersebut.

Sebagai contoh, pada gambar 2.5 (b), simpul b bertetangga dengan simpul yang secara visual ditempatkan di atasnya, yaitu simpul a. Maka dari itu, simpul b anak simpul a.

3. Saudara kandung (*sibling*)
   Saudara kandung atau *sibling* dalam pohon berakar merupakan relasi antara simpul dengan orangtua yang sama. Sebagai contoh, pada gambar 2.5 (b), simpul b dan simpul d memiliki orangtua yang sama, yaitu simpul a. Maka dari itu, b dan d merupakan saudara kandung.
4. Upapohon (*subtree*)
   Upapohon atau *subtree* merupakan upagraf dari pohon berakar. Upapohon juga merupakan pohon berakar, sehingga terminologi pohon berakar berlaku untuk upapohon.
5. Derajat (*degree*)
   Degree, dalam pohon berakar, merupakan jumlah anak dari suatu simpul.
6. Daun (leaf)
   Daun atau leaf dalam pohon berakar merupakan sebutan untuk simpul yang memiliki tidak memiliki relasi orangtua dengan simpul apapun (simpul orang tua suatu simpul yang tidak ada).
7. Simpul Dalam (*internal node*)
   Simpul dalam pada pohon berakar merupakan sebutan untuk simpul yang berupa orangtua dari simpul lain (simpul orangtua dari suatu simpul lain).
8. Tingkat (*level*)
   Tingkat dalam pohon berakar panjang lintasan dari simpul akar ke suatu simpul.



Gambar 2.6 Pohon berakar, terminologi simpul dan terminologi relasi antarsimpul
Sumber*:* Penulis

D. Pohon n-ary

Pohon n-ary merupakan sebutan untuk pohon berakar yang setiap simpulnya memiliki paling banyak n buah anak. Dengan menggunakan pohon n-ary, terdapat batasan yang pasti terhadap jumlah anak yang dimiliki oleh setiap simpul pohon. Dengan adanya pohon n-ary, terdapat dasar yang jelas dalam penyusunan abstrak struktur data untuk merepresentasikan properti dari pohon berakar.

E. Bahasa Pemrograman C

Bahasa Pemrograman C merupakan bahasa pemrograman yang dirintis oleh Dennis MacAlistair Ritchie di Bell AT&T Laboratories pada awal tahun 1970. Dengan semakin populernya Unix, *operating system* yang juga dikembangkan oleh oleh Bell AT&T Laboratories, bahasa C menjadi semakin populer karena statusnya sebagai bahasa pemrograman standar pada *operating system* tersebut. Karena fleksibilitas, efisiensi, dan kecocokan yang tinggi dengan berbagai macam *operating system*, bahasa pemrograman C menjadi bahasa yang populer hingga saat ini.

F. Variabel

Dalam bahasa pemrograman, variabel merupakan wadah yang digunakan untuk menyimpan informasi dengan tipe data tertentu. Melalui variabel, program dapat menyimpan informasi dari pihak eksternal maupun informasi yang diciptakan sendiri berdasarkan instruksi-instruksi *source code*, algoritma yang disusun oleh pencipta program agar program dapat menjalankan fungsi tertentu.

G. Pernyataan if-else dan switch-case

Program diciptakan untuk menjalankan aksi secara sistematis berdasarkan apa yang diberikan pembuatnya dalam rupa instruksi yang disebut *source code.* Dalam praktiknya, belum tentu semua instruksi yang diberikan pada *source code* dijalankan, melainkan hanya serangkaian instruksi saja berdasarkan data atau informasi yang diinput pengguna. Maka dari itu, program membutuhkan suatu metode untuk mengolah informasi dan membandingkannya dengan syarat suatu instruksi dijalankan. Salah satu caranya yaitu menggunakan *if-else statement* atau *switch-case statement.* Dengan menggunakan salah satu *statement* atau pernyataan tersebut, pembuat program dapat menentukan jalan atau tidaknya suatu instruksi berdasarkan kondisi yang diinginkan. Salah satu perbedaan mendasar dari *if-else statement* dan *switch-case statement* terletak pada jumlah kondisi macam kondisi.

If-else statement dapat menampung beberapa dengan tipe yang bermacam-macam. Sebagai contoh, *if-else statement* dapat memproses kondisi yang membandingkan argumen berupa data bertipe bilangan bulat (*integer*) dengan suatu nilai bilangan bulat dengan berbagai operator perbandingan seperti operator lebih kecil dari (<), lebih besar dari (>), sama dengan (=), dan sebagainya. Berbeda halnya dengan *switch-case statement* yang hanya memiliki satu kondisi saja setiap instruksi tertentu dengan hanya perbandingan sama dengan.

Dalam pemrograman menggunakan bahasa C, *if-else statement* dinyatakan menggunakan format sebagai berikut:

Untuk *switch-case statement* dalam bahasa C, dapat dilihat contoh deklarasinya sebagai berikut:

```
switch (<argumen>) {
    case <nilai_argumen1>:
        //Kode yang akan dieksekusi
        //jika <argumen> = <nilai_argumen1>
        break
    case <nilai_argumen2>:
        //Kode yang akan dieksekusi
        //jika <argumen> = <nilai_argumen2>
        break
    case <nilai_argumen3>:
        //Kode yang akan dieksekusi
        //jika <argumen> = <nilai_argumen3>
        break
    default:
        //Code yang akan dieksekusi
        //Jika tidak ada semua kondisi case
        //tidak memenuhi.
}
```

Gambar 2.7 Switch-case statement
Sumber: Penulis

H. Tipe Data Abstrak

Tipe data abstrak (TDA), atau *Abstract Data Type* (ADT), merupakan lingkup yang digunakan untuk memenuhi sebagai dasar dari instruksi pengguna. Melalui ADT, pencipta program dapat mendefinisikan istilah untuk sebutan variabel, mendeklarasikan struktur data yang lebih cocok untuk keperluan program, dan fungsi serta prosedur sebagai berbagai aksi yang dapat dipanggil oleh program.

III. Aplikasi Pohon Berakar dalam Pembuatan Kisah Interaktif

A. Pembuatan Tipe Data Abstrak

Untuk mengaplikasikan graf pohon berakar dalam pembuatan kisah interaktif, telah diciptakan program berbahasa C beserta ADT yang mencakupi fungsi, prosedur, serta struktur data pohon berakar untuk menyimpan informasi yang digunakan. Pohon berakar yang dimaksud pohon 3-ary, atau pohon terner. Berikut gambar ADT yang telah diciptakan penulis untuk program kisah interaktif.

.

```
// ADT STORY TELLING
// Mendeklarasikan struktur data, selektor, fungsi, dan prosedur
// by Stefanus Jeremy Aslan 13519175

#include <stdio.h>
#include <stdlib.h>

#ifndef STORY_H
#define STORY_H
typedef int Skenario;

typedef struct nVertex *Address;
typedef struct nVertex{
    Skenario ID;
    Address Left;
    Address Middle;
    Address Right;
} Vertex;

typedef Address Tree;

#define ID(P) (P)->ID
#define Left(P) (P)->Left
#define Middle(P) (P)->Middle
#define Right(P) (P)->Right
#define Nil NULL

Address InitTree();
Tree Alokasi(Skenario ID, Address Left, Address Middle, Address Right);
Skenario choice(Skenario X);

void StartStory(Tree DataPohon);

void BridgeScene(Address *P);
    void KnifeFightScene(Address *P);
    void JungleScene(Address *P);
        void RunScene(Address *P);
        void MauledScene(Address *P);
        void VictScene(Address *P);
    void JailScene(Address *P);
        void ExecScene(Address *P);
        void EscapeScene(Address *P);
#endif
```

Gambar 3.1 ADT Story Telling
Sumber: Penulis

Variabel dengan tipe data Vertex merepresentasikan simpul pada pohon terner. Tipe data Vertex memiliki data penyusun berupa ID, data bertipe integer yang dipanggil Skenario, dan tiga alamat memori: Left, Middle, Right, yang digunakan untuk menyimpan tiga alamat anak dari simpul.

Fungsi dan prosedur yang dideklarasikan ADT antara lain:

1. InitTree()
   Fungsi yang menciptakan pohon terner lengkap dengan semua simpul yang merepresentasikan skenario yang dirancang kemudian mengembalikan akar dari pohon terner tersebut.
2. Alokasi(P: Address)
   Fungsi Alokasi digunakan untuk menciptakan simpul dan mengirimkan alamatnya untuk pemakaian.
3. choice(X: integer)
   Fungsi choice digunakan untuk menerima input dari pengguna yang merepresentasikan keputusan yang diambil pengguna dalam skenario kisah. Memiliki satu parameter: integer X, yang digunakan untuk memblok pilihan tidak valid jika pilihan lebih sedikit dari tiga.
4. StartStory(DataPohon: Tree)
   Prosedur yang dipanggil program utama untuk memulai kisah untuk dinikmati pengguna. Membutuhkan alamat root yang diciptakan InitTree.
5. [Name]Scenario(P: Address)
   Prosedur yang memainkan skenario tertentu. Mengubah

address P menjadi anak dari P, Left, Middle, atau Right, tergantung dengan keputusan yang diambil player jika skenario meminta keputusan.

B. Implementasi Fungsi dan Prosedur ADT

Setelah dideklarasikan di dalam ADT, fungsi dan prosedur yang di dalamnya diimplementasikan. Berikut merupakan fungsi utama program, fungsi main.

```
int main() {
    Address DataPohon;
    DataPohon = InitTree();
    StartStory(DataPohon);
    return 0;
}
```

Gambar 3.2 Fungsi main
Sumber: Penulis

Pertama, fungsi main mendeklarasikan variabel alamat dengan tipe Address yang kemudian diisi dengan alamat akar dari pohon terner melalui fungsi InitTree.

```
Address InitTree() {

    Address PohonStory;
        Address KnifeFightScene;
        Address JungleScene;
            Address RunScene;
            Address MauledScene;
            Address VictScene;
        Address JailScene;
            Address ExecScene;
            Address EscapeScene;

    KnifeFightScene = Alokasi(11, Nil, Nil, Nil);

    RunScene = Alokasi(121, Nil, Nil, Nil);
    MauledScene = Alokasi(122, Nil, Nil, Nil);
    VictScene = Alokasi(123, Nil, Nil, Nil);
    JungleScene = Alokasi(12, RunScene, MauledScene, VictScene);

    ExecScene = Alokasi(131, Nil, Nil, Nil);
    EscapeScene = Alokasi(132, Nil, Nil, Nil);
    JailScene = Alokasi(13, ExecScene, EscapeScene, Nil);

    PohonStory = Alokasi(1,KnifeFightScene,JungleScene,JailScene);

    return PohonStory;
}
```

Gambar 3.3 Fungsi InitTree
Sumber: Penulis

Mula-mula, fungsi InitTree akan mendeklarasikan variabel-variabel alamat bertipe Address yang digunakan untuk menampung informasi. Setelah itu, semua variabel alamat simpul diberikan ID nilai bertipe integer untuk mengakses fungsi skenario yang sesuai, beserta anak simpul yang disimpan pada, Left, Middle, dan Right.

Fungsi utama kemudian memanggil fungsi StartStory dengan parameter diisi oleh variabel DataPohon yang menyimpan alamat akar pohon terner.

```
void StartStory(Tree DataPohon) {
    Address P;
    int End;

    P = DataPohon;
    End = 0;

    printf("\nSTORY START!\n");
    printf("\n------------------------------------------------------------\n\n");
    do {
        switch (ID(P)) {
            case 1:
                BridgeScene(&P);
                break;
            case 11:
                KnifeFightScene(&P);
                End = 1;
                break;
            case 12:
                JungleScene(&P);
                break;
            case 121 :
                RunScene(&P);
                End = 1;
                break;
            case 122 :
                MauledScene(&P);
                End = 1;
                break;
            case 123 :
                VictScene(&P);
                End = 1;
                break;
            case 13 :
                JailScene(&P);
                break;
            case 131 :
                ExecScene(&P);
                End = 1;
                break;
            case 132 :
                EscapeScene(&P);
                End = 1;
                break;
        }
    } while(End == 0);
}
```

Gambar 3.4 Prosedur StartStory
Sumber: Penulis

Prosedur StartStory pertama melakukan deklarasi terhadap variabel bertipe integer bernama End dan variabel alamat bertipe Address bernama P. Variabel End merupakan variabel yang digunakan untuk menentukan sudah berakhirnya kisah atau belum. End diberikan nilai 0, yang berarti false, dan bernilai 1 bila cerita berakhir. Variabel P diinisialisasi dengan alamat akar dari pohon terner, yang kemudian digunakan untuk menelusuri alamat-alamat skenario lain untuk berpindah skenario.

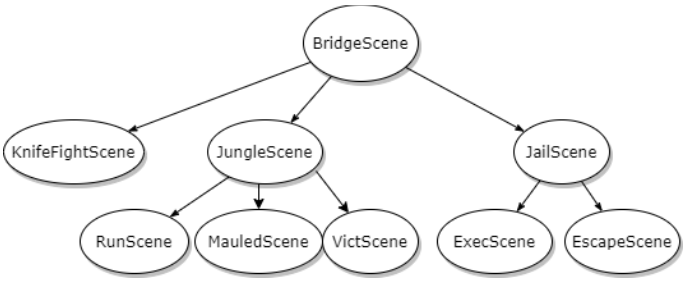
Selanjutnya, fungsi akan memainkan skenario berdasarkan ID dari variabel P hingga cerita berakhir. Setiap skenario akan mengubah nilai P menjadi salah satu dari anak P kecuali skenario ending, yang ditandai assignment End = 1 setelah fungsi skenario dimainkan.

Pada skenario memilih yang dimainkan dalam prosedur BridgeScene, JungleScene, dan JailScene, digunakan fungsi choice untuk menerima keputusan pengguna.

Pada fungsi Pemain akan diminta untuk menginput dengan *range* [1,3]. Jika choice dipanggil parameter integer dengan nilai pada *range* [1..3], pilihan dengan nilai yang sama dengan X menjadi tidak valid. Apabila input tidak valid, pengguna diminta untuk menginput ulang hingga input valid.

C. Demo Program

Fungsi InitTree akan menciptakan pohon terner yang jika digambarkan secara visual adalah sebagai berikut.



Gambar 3.5 Pohon Terner Kisah Interaktif
Sumber: Penulis

Berikut hasil demo program dengan rute BridgeScene → JungleScene → VictScene.

```
STORY START!

------------------------------------------------------------

"Berhenti!" ucap Jenderal Sukijah, pedang tajam di tangan. Dengan rasa takut Anda menatapnya kembali.

Mata sang Jenderal tajam, tapi ada rasa iba di dalamnya. Anda sudah tidak bisa kabur lagi.

"Kembalikan cincin istriku, dan aku akan membiarkanmu pergi," Ucap Jenderal dengan halus.

Memang ada benar katanya. Jembatan sungai tempat anda berpijak sudah dikepung dari dua ujung.

Anda menatap cincin besar dengan kepala berlian yang mengkilau. Apa yang akan Anda lakukan?

1. Hunuskan pisau dapur di kantung dan tantang Jenderal Sukijah dalam pertarungan 1 lawan 1.
2. Lompat ke sungai di bawah jembatan.
3. Kembalikan cincin curian.
2

------------------------------------------------------------

Ketika Anda sadarkan diri, Anda berada di pinggir sungai entah berantah

Lapar, Anda pergi ke dalam hutan untuk mencari makanan.

HROOORGHF!

Beruang liar telah muncul. Apa yang akan kau lakukan?

1. LARI!
2. Hunuskan pisau dapur di kantung dan tantang beruang liar dalam pertarungan 1 lawan 1.
3. Pura-pura mati.
3

------------------------------------------------------------

Anehnya, beruang itu hanya mengendus punggungmu sebelum pergi setengah jam kemudian.

Anda berdiri dari tempat anda merebahkan diri, cincin di kantung dan nyawa di badan.

"Waktunya menjadi kaya raya!"

TAMAT
```

Gambar 3.6 Rute BridgeScene → JungleScene → VictScene.
Sumber: Penulis

Berikut hasil demo program dengan rute BridgeScene → JailScene → ExecScene.

```
STORY START!

------------------------------------------------------------

"Berhenti!" ucap Jenderal Sukijah, pedang tajam di tangan. Dengan rasa takut Anda menatapnya kembali

Mata sang Jenderal tajam, tapi ada rasa iba di dalamnya. Anda sudah tidak bisa kabur lagi.

"Kembalikan cincin istriku, dan aku akan membiarkanmu pergi," Ucap Jenderal dengan halus.

Memang ada benar katanya. Jembatan sungai tempat anda berpijak sudah dikepung dari dua ujung.

Anda menatap cincin besar dengan kepala berlian yang mengkilau. Apa yang akan Anda lakukan?

1. Hunuskan pisau dapur di kantung dan tantang Jenderal Sukijah dalam pertarungan 1 lawan 1.
2. Lompat ke sungai di bawah jembatan.
3. Kembalikan cincin curian.
3

------------------------------------------------------------

Jenderal Sukijah berbohong dan Anda berakhir di balik batang-batang besi.

Anda mendengar perguncangan para penjaga mengenai si pencuri dengan rekor tangkap-eksekusi tercepat.

Kemungkinan besar si pencuri yang dibicarakan ialah Anda. Apa yang akan Anda lakukan?
1. Tidak melakukan apa-apa. Anda bisa memikirkan sebuah rencana besok pagi.
2. Mencoba kabur lewat sela batang besi yang terlalu lebar.
1

------------------------------------------------------------

Anda terlalu positif. Tidur anda tidak melebihi tengah malam sebelum anda di seret keluar sel.

Di sana ia berdiri, seorang perwira lengkap dengan topi perwira. Satu langkah disusul yang lain,
"Ada kata-kata terakhir?" Jenderal Sukijah berdiri di depan dirimu yang terkekang di guillotine

Dengan muka masam, Anda menatapnya.

"Tidak."

Sang Jenderal hanya mengangguk. Dengan tarikan tali, Anda kehilangan kesadaran Anda selamanya.

TAMAT
```

Gambar 3.7 Rute BridgeScene → JailScene → ExecScene.
Sumber: Penulis

Berikut hasil demo program dengan rute BridgeScene → KnifeFightScene.

```
STORY START!

------------------------------------------------------------

"Berhenti!" ucap Jenderal Sukijah, pedang tajam di tangan. Dengan rasa takut Anda menatapnya kembali.

Mata sang Jenderal tajam, tapi ada rasa iba di dalamnya. Anda sudah tidak bisa kabur lagi.

"Kembalikan cincin istriku, dan aku akan membiarkanmu pergi," Ucap Jenderal dengan halus.

Memang ada benar katanya. Jembatan sungai tempat anda berpijak sudah dikepung dari dua ujung.

Anda menatap cincin besar dengan kepala berlian yang mengkilau. Apa yang akan Anda lakukan?

1. Hunuskan pisau dapur di kantung dan tantang Jenderal Sukijah dalam pertarungan 1 lawan 1.
2. Lompat ke sungai di bawah jembatan.
3. Kembalikan cincin curian.
1

------------------------------------------------------------

Seperti banteng, Anda berlari menuju Jenderal, pisau di atas kepala.

"Penuh celah." gumam Jenderal Sukijah.

SPLAAAT!

Segalanya berubah menjadi gelap sebelum Anda sadar apa yang telah terjadi.

TAMAT.
```

Gambar 3.8 Rute BridgeScene → KnifeFightScene.
Sumber: Penulis

V. KESIMPULAN

Berdasarkan teori dan praktik aplikasi graf pohon berakar menggunakan program berbahasa C, didapatkan bahwa graf terbukti efektif dalam pembuatan kisah interaktif dengan berbagai rute yang menjalar keluar dari awal cerita yang digambarkan oleh akar pohon graf. Tidak hanya mendekati permasalahan dengan efektif dan efisien, sifat graf pohon berakar yang mudah divisualisasikan membuat alur penciptaan kisah dan program menjadi lebih ringan.

VI. UCAPAN SYUKUR

Puji dan syukur penulis ucapkan kepada Tuhan Yang Maha Esa karena atas penyertaan dan bimbingan-Nya, penulis dapat menyelesaikan makalah ini tanpa ada kendala. Penulis juga mengucapkan terima kasih kepada Ibu Fariska Zakhralativa Ruskanda selaku dosen pengajar penulis di mata kuliah Matematika Diskrit kelas K-3 dan kepada seluruh dosen pengajar mata kuliah Matematika Diskrit yang berperan aktif dalam pembuatan konten pembelajaran mahasiswa selama pandemi karena tanpa ilmu pengetahuan yang dilimpahkan Bapak dan Ibu dosen sekalian, penulis tidak mungkin dapat membuat makalah ini.

REFERENSI

[1] Kochan, Stephen G, "Programming in C, Third Edition," Indianapolis, Indiana: Sams Publishing, 2005, pp. 1-2.
[2] https://www.tutorialspoint.com/computer_programming/computer_programming_operators.htm; diakses pada 10 Desember 2020, pukul 21.32 WIB.
[3] https://www.geeksforgeeks.org/abstract-data-types/; diakses pada 11 Desember 2020, pukul 19.21.
[4] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf; diakses pada 8 Desember 2020, pukul 16.53.
[5] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf; diakses pada 8 Desember 2020, pukul 17.12.
[6] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag2.pdf; diakses pada 8 Desember 2020, pukul 17.12.

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.

Jakarta, 11 Desember 2020

Stefanus Jeremy Aslan
13519175