Aplikasi Struktur Diskrit dalam *Game*

Riefky Amarullah R - 13511038
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
riefkyamarullah@students.itb.ac.id

***Abstract*— *Game* atau permainan komputer merupakan suatu sarana hiburan yang digemari banyak orang pada masa kini. Banyaknya jenis dari permainan komputer tersebut menyebabkan banyaknya orang yang menjadikan permainan komputer sebagai sarana hiburan yang cocok untuk menghilangkan rasa jenuh dan bosan. Salah satu jenis permainan tersebut adalah jenis strategi, dimana dalam permainan tersebut dibutuhkan semacam taktik agar dapat memenangkan atau mendapatkan efisiensi maksimum dalam memainkannya. Untuk dapat memaksimumkan komponen yang ada dalam game tersebut dapat digunakan teori graf, dan kombinatorial. Di dalam game tersebut terdapat semacam *skill tree* atau pohon kemampuan dan juga kombinasi dari berbagai *skill* yang menghasilkan *skill* baru sehingga dibutuhkan ketepatan dalam menggunakannya, lebih tepatnya dengan menggunakan *topological sort* dan *decision tree*.**

***Index Terms*—game, topological sort, graph, kombinatorial**

I. PENDAHULUAN

Game atau permainan komputer merupakan suatu sarana hiburan yang digemari banyak orang pada masa kini. Banyaknya jenis dari permainan komputer tersebut menyebabkan banyaknya orang yang menjadikan permainan komputer sebagai sarana hiburan yang cocok untuk menghilangkan rasa jenuh dan bosan. Selain alasan tersebut, masih banyak lagi alasan lain sehingga orang-orang memainkan suatu *game*.

Salah satu jenis permainan tersebut adalah jenis strategi, dimana dalam permainan tersebut dibutuhkan semacam taktik agar dapat memenangkan atau mendapatkan efisiensi maksimum dalam memainkannya. Pada setiap *game* strategi pasti selalu ada unsur yang membedakan cara memainkannya, tetapi ada satu hal yang sama yaitu dalam setiap game tersebut pasti ada cara tertentu atau trik-trik khusus agar kita dapat memainkan *game* tersebut dengan baik dan maksimal. Semua game, dari *casual game* hingga *hardcore game* memerlukan strategi dalam memainkannya. Contohnya, lihat saja game yang sangat digemari sekarang, yaitu game sejenis *Angry Birds* dan *game* lain yang dibuat oleh developer *Angry Birds*. Meskipun *game* tersebut hanya membutuhkan satu jari untuk memainkannya, tetap saja harus menggunakan strategi agar dapat menyelesaikan level-levelnya.

Selain *game* tersebut, tentu saja *game* lain yang lebih kompleks juga membutuhkan strategi dalam memainkannya. Dalam makalah ini akan dibahas aspek-aspek yang umumnya ada pada *game* strategi dan khususnya akan diulas mengenai trik kombinatorial untuk *game* Dota 2. Aspek-aspek umum yang ada di game strategi adalah *skill tree* dan juga adanya persyaratan untuk mengambil *skill* tertentu dari *skill tree* tersebut, sehingga dapat digunakan *topological sort* untuk mendapatkan hasil yang maksimum.

Pada *game* Dota 2, terdapat karakter, biasa disebut dengan *hero*, yang bernama Invoker. *Hero* Invoker tersebut memiliki 3 buah *skill* dasar yang dapat digabungkan sehingga menjadi *skill* baru. Dengan kombinatorial kita dapat memetakan kombinasi 3 *skill* tersebut sehingga dapat menjadi 10 buah *skill* baru dan menjadikannya kedalam bentuk graf yang akan disort dengan *topological sort* sehingga mendapatkan hasil tercepat atau maksimum.

II. LANDASAN TEORI

A. Kombinatorial

Kombinatorial adalah cabang matematika untuk menghitung jumlah penyusunan objek-objke tanpa harus mengenumerasi semua kemungkinan susunannya. Cara menghitung dari kombinatorial ada 2 yaitu permutasi dan kombinasi.

Permutasi merupakan jumlah urutan berbeda dari pengaturan objek-objek. Permutasi r dari n elemen adalah jumlah kemungkinan urutan r buah elemen yang dipilih dari n buah elemen, dengan $r \leq n$, yang dalam hal ini, pada setiap kemungkinan urutan tidak ada elemen yang sama.

$$P(n,r) = n(n-1)(n-2)\ldots(n-(r-1)) = \frac{n!}{(n-r)!}$$

Sedangkan kombinasi adalah bentuk khusus dari permutasi, yaitu bentuk saat urutan kemunculan tidak diperhitungkan atau diabaikan

$$C(n,r) = \frac{n!}{r!(n-r)!}$$

Dalam masalah ini, yaitu *hero* Invoker, kita mempunyai 3 buah *skill* dasar dan 3 buah kotak *skill* dasar. 3 buah *skill* ini akan dimasukan dalam kotak dan digabungkan, dengan urutan *skill* dasar tidak berpengaruh dan 1 *skill* dasar dapat digunakan berulang kali sehingga memungkinkan 3 kotak tersebut berisi *skill* dasar yang sama. Didapat seluruh kombinasi yang mungkin adalah $3! + 4 = 10$, dimana 3! berasal dari jumlah *skill* dasar dan 4 berasal dari kombinasi sisa kombinasi, yaitu $2^2 = 4$.

Dalam masalah ini, suatu *skill* dasar yang akan dimasukan pasti akan masuk ke kotak pertama, menggeser *skill* dasar yang lain ke kotak kedua dan kotak kedua tergeser ke kotak ketiga dan menghapus yang ada di kotak ketiga sebelumnya. Kondisi ini dapat dijabarkan lagi kedalam bentuk graf, karena setelah mendapatkan 1 kombinasi, pemasukan *skill* dasar tetap diulangi, dan skill baru yang dimasukan ditaruh di kotak pertama, sehingga yang tadinya berada di kotak pertama digeser ke kotak kedua, dan kotak kedua digeser ke kotak ketiga dan yang tadinya berada di kotak ketiga dibuang. Dengan memetakannya menjadi graf dan menggunakan *topological sort* kita dapat mengetahui cara tercepat untuk menghasilkan kombinasi *skill* tertentu.

Adapun *skill* dasarnya ada 3 yaitu, Quas (Q), Wex (W), dan Exort(E). Jika dijabarkan maka daftar dari kombinasi beserta skill yang akan dikeluarkan sesuai dengan kombinasinya adalah sebagai berikut :

- *Cold Snap = Quas Quas Quas (QQQ)*
- *Ghost Walk = Quas Quas Wex (QQW)*
- *Ice Wall = Quas Quas Exort (QQE)*
- *EMP = Wex Wex Wex (WWW)*
- *Tornado = Wex Wex Quas (WWQ)*
- *Alacrity = Wex Wex Exort (WWE)*
- *Sun Strike = Exort Exort Exort (EEE)*
- *Forge Spirit = Exort Exort Quas (EEQ)*
- *Chaos Meteor = Exort Exort Wex (EEW)*
- *Defeaning Blast = Quas Wex Exort (QWE)*

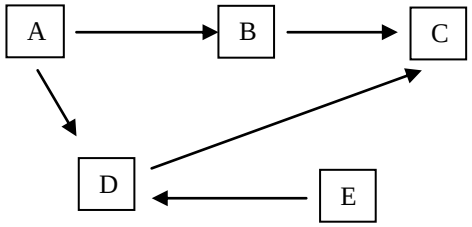
Atau jika dibuat dalam bentuk tabel akan seperti gambar 1 dibawah ini

Gambar 1. *List Skill* dari Invoker

Adapun, urutan dari pemasukan *skill* dasar tidak berpengaruh, contohnya *Tornado* bisa dibentuk dari kombinasi WWQ, WQW, ataupun QWW.

B. Graf dan *Topologic Sort*

Suatu graf digunakan untuk merepresentasikan objek-objek-objek disrit dan hubungan antara objek-objek tersebut. Suatu graf dapat direpresentasikan dengan menyatakan objek sebagai noktah, bulatan, atau titik, sedangkan hubungan antar objek dinyatakan dengan garis. Graf didefinisikan msebagai pasangan himpunan simpul yang tidak kosong dan himpunan sisi. Suatu graf dapat digunakan untuk menyelesaikan berbagai macam masalah. Pada makalah ini akan dibahas salah satu penerapan graf, khususnya *topological sort*, yaitu aplikasi teori graf dalam hal menentukan strategi ataupun taktik dalam *game*.



Graf 1 Contoh Graf Berarah

Jika diibaratkan bahwa graf diatas adalah *skill tree* maka A dan D merupakan syarat agar *skill* E dapat diambil dan seterusnya. Beberapa solusi agar semua skill dapat efektif terambil adalah A → E → D → B → C atau bisa juga E → A → B → D → C.

Topological sort merupakan suatu keterurutdari semua simpul dalam DAG (*directed acyclic graph)*. *Topological sort* ini berguna untuk mengurutkan sesuatu yang harus dikerjakan terlebih dahulu. Ide dari *tiopological sort* ini adalah membuat suatu list linier yang memiliki keterurutan dengan cara memilih salah satu simpul yang tidak memiliki pre-requisit atau dengan kata lain memilih simpul yang tidak memiliki predesesor. Jika tidak ada, maka graf tersebut tidak memiliki solusi. Selanjutnya penghapusan simpul yang tidak memiliki predesesor tersebut ke dalam suatu list, dan menghapus hubungan antara simpul tersebut dengan simpul yang lain. Dan proses tersebut terus berulang hingga semua simpul telah terhapus. Setelah semua elemen terhapus, urutan penghapusan dari elemen-elemennya akan menjadi suatu list linier yang memiliki keterurutan.

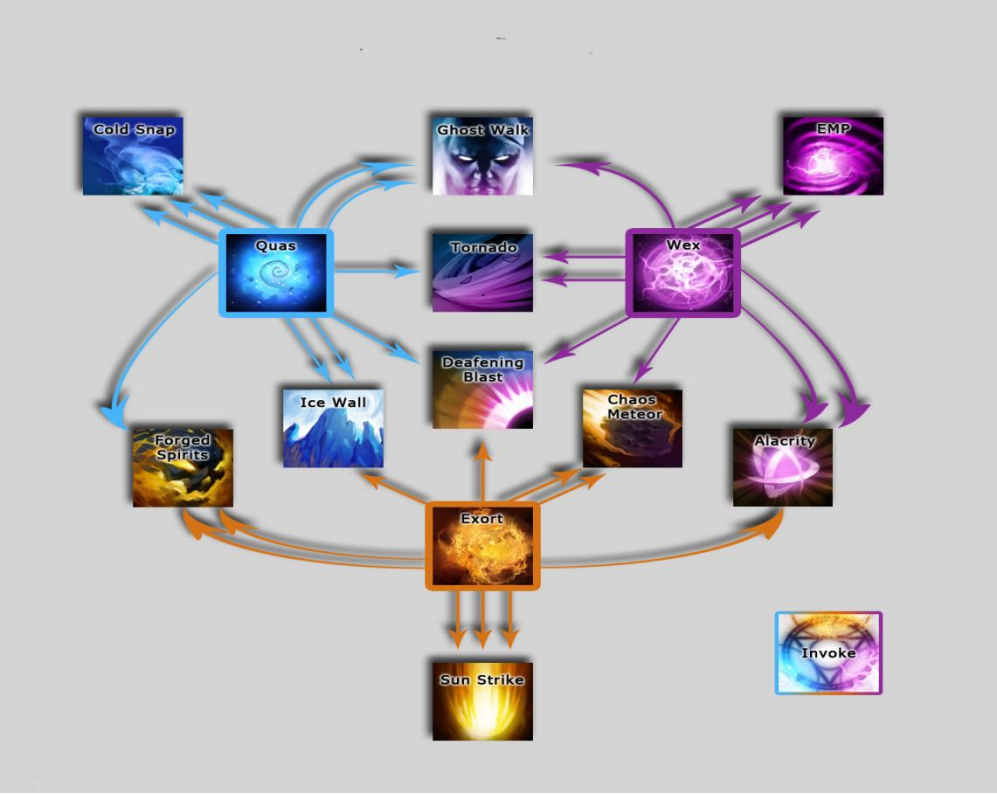
III. APLIKASI TEORI DALAM *GAME*

A. Invoker's Problem

Pada permasalahan Invoker ini sudah dapat dilihat semua kombinasi yang mungkin untuk dipakai. Dalam hal ini ketiga kotak sudah dipakai dan menghasilkan sebuah *skill*. Misalnya dalam ketiga kotak sudah terisi dengan Q, agar dapat mengeluarkan *Ice Wall*, kita tidak perlu lagi menekan seluruh kombinasi (QQE) tetapi cukup dengan menekan E saja, dengan demikian prequisite atau persyaratan dari *Ice Wall* adalah adanya salah satu aspek dari kombinasi QQE, jika tidak ada salah satu dari aspek tersebut (tersimpan kombinasi WWW), makanya keseluruhan kombinasi harus dimasukan kembali.

Contoh lain adalah, misalnya di kotak sudah tersimpan kombinasi untuk *Defeaning Blast* (EWQ), maka prequisite atau syarat yang sudah terpenuhi adalah EW, karena Q dimasukan paling awal, maka Q akan terhapus terlebih dahulu. Dengan adanya 2 *skill* dasar ini maka kita dapat membuat kombinasi dari skill lain yang membutuhkan WE. Contohnya dengan memasukan Wex (W) saja kita sudah dapat membuat kombinasi WEW, yaitu *Alacrity*. Contoh lain, dengan memasukan Exort (E) saja, kita sudah dapat membuat kombinasi EEW, yaitu *Chaos Meteor*.

Setelah dipetakan didapat graf dari semua kombinasi itu seperti pada graf 2 dibawah ini



Graf 2 Graf *Skill List* Invoker

Dalam graf tersebut dapat dilihat prequisite dari semua kombinasi *skill* invoker. Contohnya prequisite dari *Sun Strike* adalah 3 buah Exort (EEE). Dalam kotak *skill* dasar kita, prequisite yang kita punya akan selalu berubah sesuai dengan masukannya. Sehingga dengan cara seperti diatas kita dapat dengan mudah mengeluarkan kombinasi *skill* sesuai dengan yang diinginkan. Tetapi hal tersebut juga harus diimbangi dengan pola pikir yang cepat dan tepat sehingga tidak salah memetakan graf dan membuat kombinasi *skill* yang salah

B. Common Game Strategy's Problem

Pada umumnya, di suatu game terdapat trik khusus ataupun strategi agar dapat memenangkan ataupn memainkan game tersebut dengan maksimal. Biasanya pada suatu game strategi ataupun RPG terdapat *skill tree* yang memerlukan persyaratan agar kita dapat mengambil *skill* yang paling hebat atau yang tingkatnya paling tinggi. Persyaratan tersebut bermacam-macam, bisa dari harus mengambil *skill* sebelumnya ataupun memiliki point yang cukup pada bagian *skill* tertentu.

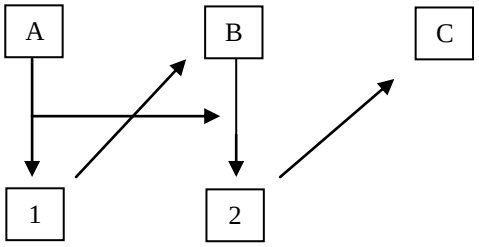
Untuk itu, *topological sort* merupakan salah satu cara yang efektif agar dapat menentukan *skill* apa yang harus dipelajari terlebih dahulu. Sebagai contohnya adalah *skill tree* yang ada pada *game Borderlands 2*. Pada *game* ini, cara pengambilan *skill* sangat sederhana, kita tidak membutuhkan prequisite atau persyaratan dari skill sebelumnya. Kita hanya membutuhkan point tertentu pada suatu *sub skill tree*. Gambarnya ada pada gambar 2 dibawah ini.

Dapat dilihat dari gambar tersebut bahwa tidak ada graf berarah yang menjadikan suatu *skill* merupakan prasyarat dari *skill* lain dibawahnya. Sehingga hanya perlu menginvestasikan 5 poin di sub *skill tree* tertentu untuk dapat mempelajari *skill* di baris 2 dan butuh 10 poin untuk dapat mempelajari skill di baris 3 dan seterusnya. Di setiap *skill* dapat diinvestasikan maksimal 5 buah poin, dengan setiap poin menambah efektifitas dari *skill* tersebut.

Misalnya kita ingin mengambil *skill* di baris ke-2 dalam sub *skill tree GUNLUST*, kita tidak perlu mengambil maksimal 5 poin pada 1 *skill* di atasnya, kita bisa membagi 5 poin tersebut ke dalam 2 buah *skill* di atasnya, misalnya dengan perbandingan 3-2.

Gambar 2 *Skill Tree Borderlands 2*

Dan juga untuk dapat mempelajari *skill* yang ada pada baris terbawah, yaitu baris 6, kita membutuhkan 25 poin pada *sub skill tree*. Dengan adanya 10 *skill* di atasnya, kita tidak bisa membagi dengan sesuka hati karena adanya prasyarat 5 poin tiap baris sehingga dengan *topological sort*, dapat disimpulkan bahwa poin terbanyak akan terbuang di baris-baris awal, yaitu baris 1 sampai 3. Dengan demikian dapat disederhanakan menjadi graf seperti pada Graf 3



Graf 3 *Simplified Borderlands 2 Skill Tree*

Setelah disederhanakan, *skill tree* dari *game Borderlands 2* dapat dibuat graf sederhana seperti pada Graf 3 dengan :

- A adalah skill baris 1
- B adalah skill baris 2
- C adalah skill baris 3
- 1 adalah 5 poin terpenuhi
- 2 adalah 10 poin terpenuhi

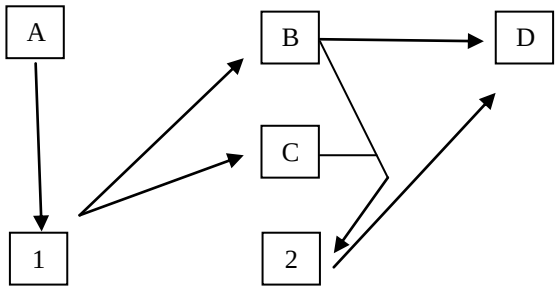
Skill B dapat diambil jika syarat 1 sudah terpenuhi, dengan syarat 1 terpenuhi jika sudah ada 5 poin pada skill A. Sedangkan *skill C* dapat diambil jika syarat 2 sudah terpenuhi, yaitu saat sudah ada total 10 poin pada *skill* A dan B. Sehingga solusi setelah digunakan *topological sort* adalah A → (1) → (2) → C atau A → (1) → B → (2) → C

Tipe dari *skill tree* di atas merupakan *skill tree* dengan graf berarah, tetapi sederhana. Jika *skill tree* menggunakan graf berarah, maka suatu skill sebelumnya akan menjadi prasyarat untuk mendapatkan *skill* selanjutnya sehingga semua *skill* yang menjadi prasyarat terpaksa diambil, tidak seperti *skill tree* diatas, ada *skill* yang boleh dilewati.

Contohnya adalah *skill tree* pada *mastery* dalam game *League of Legends. Skill tree* pada game ini menggunakan gabungan dari *skill tree* diatas dengan *skill* tertentu membutuhkan prasyarat *skill* sebelumnya. Contoh *skill tree* dapat dilihat pada gambar 3.

Gambar 3 *Skill Tree League of Legends*

Seperti yang dapat dilihat pada gambar 3, ada *skill* tertentu yang bergantung pada skill sebelumnya sehingga pada kasus *skill tree* ini harus mengambil *skill* sebelumnya untuk dapat *mengambil* skill dibawahnya. Selain itu, sama seperti sebelumnya, setiap baris membutuhkan 5 poin dalam *subtree* masing-masing.



Graf 4 *Simplified League of Legends Skill Tree*

Setelah disederhanakan, *skill tree* dari game *League of Legends* dapat dibuat graf planar seperti Graf 4 diatas, dimana misalnya :

- A adalah skill baris pertama
- B adalah skill baris kedua, prasyarat D
- C adalah skill baris kedua
- D adalah skill baris ketiga
- 1 adalah 5 poin terpenuhi
- 2 adalah 10 poin terpenuhi

Skill D dapat diambil jika prasyaratnya sudah terpenuhi yaitu sudah mengambil *skill B* dan sudah memenuhi syarat 2. Dengan demikian, dengan *topological sort*, kita dapat menentukan solusi yang harus diambil terlebih dahulu. Dalam kasus graf sederhana diatas kita dapat menentukan solusinya yaitu A → (1) → B → (2) → D → C. Jika kita mengambil C

terlebih dahulu maka *skill* D yang seharusnya sudah bisa diambil menjadi tidak bisa diambil karena prasyarat B tidak terpenuhi sehingga jalur solusi hanya 1.

C. Pros and Cons

Kelebihan menggunakan strategi ini adalah cocok jika dapat mengimbangin dengan pola pikir yang cepat dan tepat karena jika salah sedikit saja akan mempengaruhi kejadian selanjutnya, kecuali ada *game* yang memperbolehkan mengulang kembali atau me-*reset skill* ataupun *status* yang sudah diambil. Selain itu, strategi juga membuat orang tidak membuat langkah atau keputusan yang sia-sia, sehingga seluruh langkah yang dibuat berarti untuk kedepannya nanti. Mempercepat waktu permainan dalam game karena membuat langkah yang efektif setiap saat.

Kekurangannya yaitu, jika *skill tree* atau kombinasi dari suatu aspek dalam game tersebut terlalu kompleks, akan menimbulkan banyak solusi dalam *topological sort*, sehingga semua solusi dapat dikatakan efektif dan menjadikan pengguna pusing sendiri. Selain itu, agar dapat melakukan *planning* dan membuat graf seperti diatas, harus dilakukan *research* atau pencarian informasi terlebih dahulu sehingga tidak disarankan untuk *casual gamers*

D. Application in Informatics

Dalam bidang informatika, hal ini tentu saja digunakan oleh para perancang dan produsen *game* dalam membuat karyanya. Dengan menggunakan graf dan kombinatorial, pembuat *game* akan membuat *game* semenarik mungkin yang dapat dihasilkan tetapi juga tidak menyita banyak waktu jika yang memainkannya mengetahui trik-trik tertentu. Dengan cara *topological sort*, pembuat *game* dapat menyisipkan aspek-aspek yang dapat dilewati bersamaan, misalnya *quest* 1 dan 2 yang terletak di lokasi yang sama sehingga dapat menyelesaikan 2 *quest* dalam 1 waktu. Dengan demikian, pemain tidak akan merasa bosan untuk mengulangi 2 daerah yang sama sehingga memiliki nilai ketertarikan atau nilai jual yang lebih tinggi

Jika tidak dirancang dengan graf terlebih dahulu, maka suatu game akan kacau, tidak memiliki alur dan tujuan. Sehingga para pemain pun tidak suka memainkannya dan beralih ke *game* lain.

IV. KESIMPULAN

Pada pengerjaan makalah ini, penulis mendapatkan beberapa kesimpulan, diantaranya :

- Graf dapat digunakan dalam memainkan sebuah *game* untuk dapat menentukan strategi terbaik yang dapat digunakan
- Metode *topological sort* hanya berguna untuk game yang tidak terlalu kompleks
- Untuk dapat menggunakan *topological sort*, suatu graf dari sebuah *game* harus disederhanakan terlebih dahulu
- Kombinasi dari suatu aspek dalam game dapat disederhanakan juga dengan mengubahnya kedalam graf
- Dalam bidang informatika, graf dan kombinatorial berguna untuk para pembuat game merancang gamenya

REFERENCES

1. http://www.dotafire.com/dota-2/guide/invoker-blazing-across-a-black-sea-of-ignorance-868 diakses tanggal 18/12/2012
2. http://homepages.ius.edu/rwisman/C455/html/notes/Chapter22/TopSort.htm diakses tanggal 18/12/2012
3. http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2011-2012/Graf.ppt diakses tanggal 18/12/2012
4. http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2011-2012/Kombinatorial.ppt diakses tanggal 18/12/2012
5. Munir, Rinaldi. 2008. Struktur Diskrit. Bandung. Informatika
6. Gambar didapat dari screenshoot game

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, 18 Desember 2012

ttd

Riefky Amarullah R - 13511038