Penerapan Topological Sort pada Perencanaan Pembuatan Program Berskala Besar

Lionnarta Savirandy 13518128
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518128@std.stei.itb.ac.id

***Abstrak*—Salah satu aktifitas seorang programmer adalah membuat program. Program-program kecil umumnya dapat diselesaikan dengan mudah oleh seorang programmer handal, Namun seorang programmer handal juga harus dapat bekerja dalam suatu tim untuk menyelesaikan program-program skala besar. Salah satu aspek penting agar kerja sama dapat berjalan dengan baik adalah memiliki pembagian kerja yang jelas dalam tim tersebut. Pembagian kerja suatu tim dapat dijabarkan dengan menggunakan algoritma *Topological Sort,* dimana antara programmer satu dengan yang lain dapat mengetahui hal apa saja yang diperlukan untuk menyelesaikan program dengan baik.**

Keywords*—Program, Programmer, Pembagian kerja, *Topological Sort.

I. PENDAHULUAN

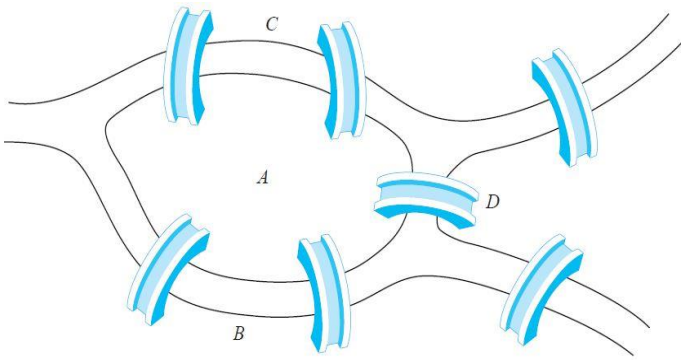
Dalam membuat suatu program yang memiliki skala besar, tentunya seorang programmer dituntut untuk bekerja sama dengan programmer lainnya supaya program dapat diselesaikan lebih cepat. Untuk mempermudah pekerjaan, para programmer setidaknya harus membuat suatu perencanaan yang baik supaya program dapat diselesaikan dengan cepat. Namun pada saat pembagian kerja, programmer umumnya membagi sistem kerja mereka secara garis besar saja tanpa mengetahui keterkaitan antara program yang akan mereka buat nantinya. Pada saat programmer mulai bekerja, mereka baru menyadari bahwa di dalam program yang mereka kerjakan membutuhkan program lainnya yang ternyata merupakan bagian kerja dari anggota timnya yang lain. Hal ini menyebabkan programmer saling menunggu satu sama lain dan rencana kerja yang telah mereka buat menjadi berantakan.

Masalah tersebut merupakan salah satu masalah yang umum terjadi dalam lingkungan kerja programmer, terutama bagi programmer baru. Hal ini dapat diselesaikan dengan berbagai macam solusi seperti menciptakan lingkungan kerja yang nyaman, meningkatkan komunikasi antara programmer, dan sebagainya. Namun cara tersebut tidak selalu dapat dilakukan sehingga diperlukan suatu cara agar programmer dapat menyusun rencana kerja mereka secara efektif.

Topological Sort merupakan salah satu jenis graf yang mana elemennya memiliki urutan *partial*. Dengan adanya hal tersebut, graf yang dibuat dapat membentuk suatu *list* yang terurut secara linier sehingga elemen graf memiliki ketergantungan dengan elemen lainnya. Dalam algoritma ini, beberapa elemen graf memiliki prasyarat agar dapat diakses yaitu dengan mengakses elemen lainnya terlebih dahulu. Dengan menerapkan algoritma ini, programmer diharapkan dapat dengan mudah menyusun rencana kerja suatu program dengan lebih efektif dan efisien.

II. TEORI DASAR

A. Graf



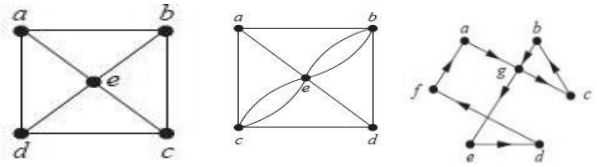
Gambar 1. Jembatan Konigsberg
Sumber : Kenneth H. Rosen, "Discrete Mathematics and Its Applications", 7th edition

Graf merupakan kumpulan dari simpul-simpul yang memiliki keterhubungan antara satu dengan yang lainnya. Suatu graf setidaknya memiliki sebuah simpul. Graf memiliki dua komponen, yaitu simpul yang dinotasikan dengan V dan sisi yang dapat dinotasikan dengan E. Sebuah sisi menghubungkan satu atau dua simpul pada graf.

Jumlah simpul dalam graf dapat berjumlah *infinite*. Graf ini disebut dengan *infinite graph*. Sebaliknya, graf dengan jumlah simpul berhingga disebut dengan *finite graph*.

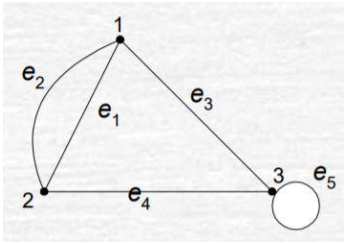
Graf dapat dikelompokkan dalam berbagai jenis. Berdasarkan adanya gelang atau sisi ganda, graf dapat dibagi menjadi dua. Pertama, graf sederhana yaitu graf yang tidak mengandung gelang maupun sisi ganda dan yang kedua adalah graf tak sederhana yaitu graf yang mengandung sis ganda atau gelang.

Berdasarkan arahnya graf dapat dibedakan menjadi dua yaitu graf berarah dan graf tak berarah. Graf berarah merupakan graf yang sisinya memiliki orientasi, sedangkan graf tak berarah merupakan graf yang sisinya tidak memiliki orientasi.



Gambar 2. Graf sederhana, graf tak sederhana, graf berarah
Sumber : Kenneth H. Rosen, "Discrete Mathematics and Its Applications", 7th edition

B. Terminologi Graf



Gambar 3. Contoh graf
Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf diakses pada 6 Desember 2019

1) Ketetanggaan (*adjacent*)

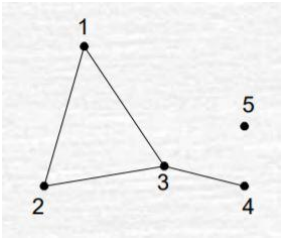
Kedua simpul disebut bertetangga jika kedua simpul tersebut terhubung langsung. Pada gambar 3, simpul 1 bertetangga dengan simpul 2 dan 3.

2) Bersisian (*incidency*)

Untuk sembarang sisi e = (v_j, v_k), maka dapat dikatakan e bersisian dengan simpul v_j, atau e bersisian dengan simpul v_k. Pada gambar 3, sisi e_1 bersisian dengan simpul 1 dan 2.

3) Simpul terpencil (*isolated vortex*)

Simpul terpencil adalah simpul yang tidak memiliki sisi yang bersisian dengannya.

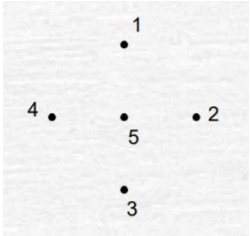


Gambar 4. Contoh graf
Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf diakses pada 6 Desember 2019

Pada gambar 4, simpul 5 disebut dengan simpul terpencil.

4) Graf kosong (*null graph* atau *empty graph*)

Graf kosong adalah graf yang semua sisinya merupakan himpunan kosong atau tidak memiliki sisi.



Gambar 5. Graf kosong
Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf diakses pada 6 Desember 2019

5) Derajat (*degree*)

Derajat sebuah simpul merupakan jumlah sisi yang bersisian dengan simpul tersebut. Pada gambar 4, derajat simpul 5 adalah satu, derajat simpul 3 adalah tiga.

6) Lintasan (*path*)

Lintasan merukan sisi yang dilalui dari simpul asal ke simpul tujuan. Panjang lintasan merupakan jumlah sisi yang dilalui. Pada gambar 4, lintasan 1,3,4 adalah lintasan dengan barisan sisi (1,3), (3,4). Panjangnya lintasan 1,3,4 adalah 2.

7) Siklus (*cycle*) atau sirkuit (*circuit*)

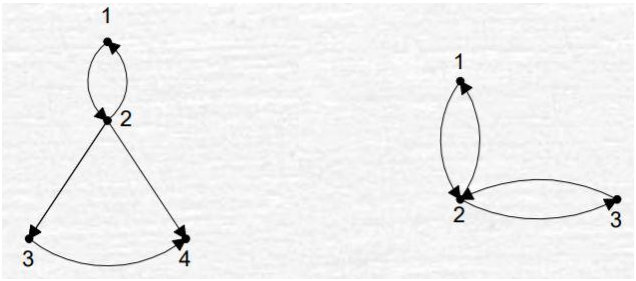
Sirkuit atau siklus merupakan lintasan yang berawal dan berakhir pada simpul yang sama. Pada gambar 4, 1,3,2,1 merupakan sebuah sirkuit. Panjang dari sirkuit tersebut adalah 3.

8) Terhubung (*connected*)

Dua buah simpul v_1 dan v_2 dikatakan terhubung jika terdapat suatu lintasan dari v_1 ke v_2. Graf yang setiap simpulnya memiliki lintasan disebut graf terhubung. Sebaliknya jika ada dua buah simpul yang tidak memiliki lintasan, maka disebut graf tak terhubung.

Dua buah simpul u dan v dikatakan terhubung kuat jika pada graf berarah terdapat lintasan berarah dari u ke v dan lintasan berarah dari v ke u. Jika setiap simpul terhubung kuat, maka disebu graf berarah terhubung kuat.

Dua buah simpul u dan v dikatakan terhubung lemah jika pada graf tidak berarahnya terhubung namun tidak terhubung kuat. Jika terdapat simpul terhubung lemah, maka graf disebut graf berarah terhubung lemah.



Gambar 6. Graf berarah terhubung lemah dan graf berarah terhubung kuat
Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf diakses pada 6 Desember 2019

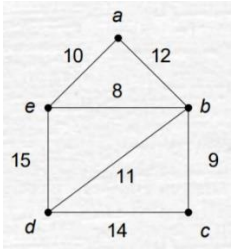
9) Upagraf dan komplemen graf

Misalkan graf G = (V, E), graf G_1 = (V_1, E_1) sedemikian sehingga $V_1 \subseteq V$ dan $E_1 \subseteq E$, maka G_1 disebut upagraf dari G. Komplemen dari upagraf G_1 adalah G_2 = (V2, E2) sedemikian

sehingga $E_2 = E - E_1$ dan V_2 adalah himpunan simpul yang anggota-anggota E_2 bersisian dengannya.

10) Graf berbobot

Graf berbobot merupkan graf yang setiap sisinya diberikan suatu nilai/harga.



Gambar 7. Graf berbobot

Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf diakses pada 6 Desember 2019

11) Upagraf rentang

Graf $G_1 = (V_1,E_1)$ adalah upagraf rentang dari graf $G = (V,E)$ jika setiap simpul V_1 merupakan anggota dari V atau $V_1 = V$.

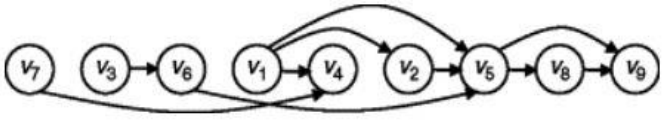
12) *Cut-Set*

Cut-set dari graf terhubung adalah himpunan sisi sedemikian sehingga jika himpunan tersebut dibuang, maka graf menjadi tidak terhubung.

C. Topological Sort

Topological sort merupakan algoritma untuk mengurutkan simpul-simpul pada graf berarah, sehingga membentuk keterurutan linier di mana suatu simpul tidak akan pernah dapat diakses jika elemen sebelum simpul tersebut belum diakses. Misalkan terapat simpul a ke b dan simpul b ke c, maka simpul c tidak dapat diakses jika simpul b belum diakses, demikian juga simpul b tidak dapat diakses jika simpul a belum diakses. Dengan demikian pengaksesan simpul membentuk keterurutan yaitu a→b→c. Dapat juga dinyatakan dengan a<b dan b<c.

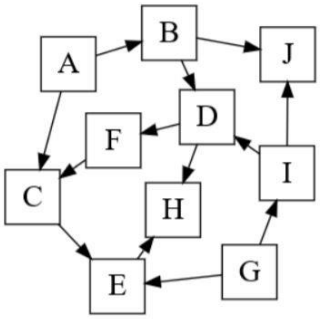
Topological sort dapat direpresentasikan dalam bentuk *directed acyclic graph*. Penggambaran dari *topological sort* dapat dinyatakan sebagai simpul yang terurut sehingga membentuk sebuah garis horizontal.



Gambar 8. *Topological sort* dalam bentuk simpul terurut

Sumber : https://www.sciencedirect.com/topics/computer-science/topological-sort diakses pada 6 Desember 2019

Selain itu, *topological sort* juga dapat digambarkan secara tidak beraturan seperti pada gambar 9 dibawah ini.



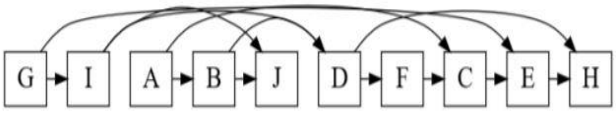
Gambar 9. *Topological sort*

Sumber : https://stei.kuliah.itb.ac.id/pluginfile.php/38937/mod_resource/content/1/W15_A3_SK6_TopologicalSort.pdf diakses pada 6 Desember 2019

Directed acyclic graph (DAG) memiliki berbagai pengaplikasian selain pada *topological sort* DAG juga digunakan pada industri *electronic design automation* (EDA) dikarenakan mampu untuk memodelkan relasi masukan dan keluaran dari rangkaian sirkuit. Selain itu DAG juga dapat digunakan dalam rangkaian acara jika pada rangkaian tersebut terdapat depedensi dengan rangkaian lainnya.

Algoritma sederhana dalam *topological sort* adalah sebagai berikut :

1) Pilih salah satu elemen yang tidak mempunyai elemen sebelumnya (predesesor), misalkan A. Pada suatu graf berarah, minimal terdapat satu elemen seperti ini agar dapat dibuat *topological sort*-nya. Jika tidak, maka akan terjadi *looping*.
2) Hapus A dari himpunan simpul beserta keterhubungannya dengan simpul lainnya dan masukkan ke dalam *list* yang baru.
3) Sisa dari himpunan simpul yang masih terurut *partial* akan diproses lagi melalui langkah 1 dan 2 hingga himpunan simpul habis dan terbentuk *list* yang merupakan *topological sort*.



Gambar 10. Pemecahan *topological sort* pada gambar 9

sumber : https://stei.kuliah.itb.ac.id/pluginfile.php/38937/mod_resource/content/1/W15_A3_SK6_TopologicalSort.pdf diakses pada 6 Desember 2019

III. MEKANISME PERENCANAAN DENGAN TOPOLOGICAL SORT

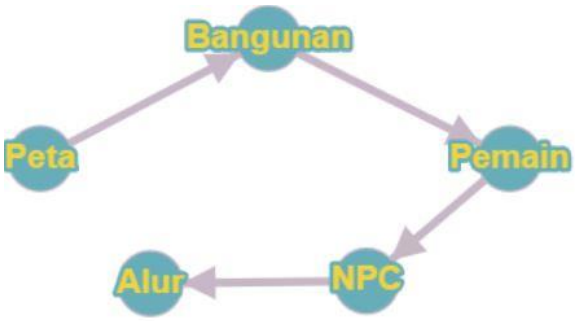
Pada pembuatan program skala besar, perencanaan yang baik diperlukan supaya program dapat dibuat dengan efektif dan efisien baik diri segi waktu maupun program itu sendiri. Dengan menggunakan *topological sort* programmer dapat membuat skema kerja di mana depedensi setiap program dapat diketahui dengan jelas dan programmer dapat mengerjakan program tersebut secara teratur sesuai dengan skema yang terbuat. Untuk memperjelas akan diberikan suatu contoh perencanaan pembuatan program permainan simulasi kehidupan dalam permainan peran (*role playing life simulation game*). Permainan ini terinspirasi dari *fantasy life* yaitu game yang dirilis oleh Nintendo.

Permainan ini menempatkan pemain sebagai seseorang yang

akan memilih suatu pekerjaan dari sekian pekerjaan, di mana pekerjaan tersebut memiliki depedensi dengan pekerjaan lainnya dan pemain dapat berpindah pekerjaan tanpa meninggalkan kemampuan yang diperoleh dari pekerjaan sebelumnya. Dalam merancang permainan ini, secara sederhana programmer dapat membagi kerjanya sebagai berikut:

1) Membuat peta permainan
2) Membuat bangunan permainan
3) Membuat pemain dan pekerjaannya
4) *Non Player Character* (NPC)
5) Alur cerita dari permainan

Pada rancangan sederhana di atas, pemrogram dapat memutuskan bahwa peta tidak memiliki depedensi dengan bagian lainnya. Sebaliknya bangunan memiliki depedensi terhadap peta, pemain memiliki depedensi terhadap bangunan dan peta, NPC memiliki depedensi terhadap peta, bangunan, dan pemain, dan alur cerita yang memiliki depedensi terhadap semuanya.



Gambar 11. *Topological sort* pertama
Sumber : dokumen penulis

Kemudian akan lebih didetailkan satu persatu setiap bagian yang ada. Pertama peta memiliki komponen seperti berikut:

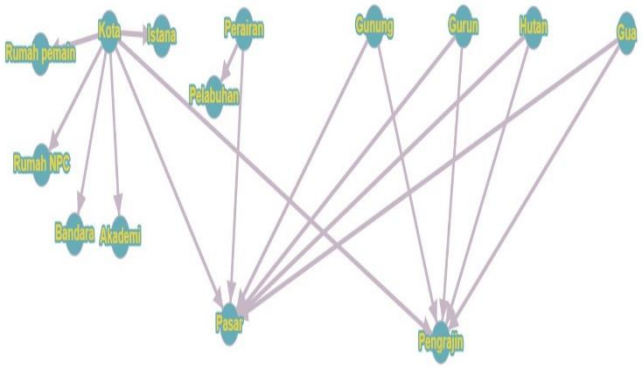
1) Kota
2) Sungai, Danau, dan Pantai (perairan)
3) Gunung
4) Gurun
5) Hutan
6) Gua

Semua bagian ini tidak memiliki depedensi satu dengan yang lainnya sehingga dapat langsung dibuat oleh programmer.

Kemudian terdapat bangunan. Bangunan memiliki komponen sebagai berikut:

1) Rumah pemain
2) Rumah NPC
3) Pasar
4) Bandara
5) Pelabuhan
6) Istana
7) Akademi
8) Tempat Pengrajin

Pada bagian ini tidak memiliki depedensi dengan komponen bangunan lainnya, namun memiliki depedensi dengan komponen peta sehingga dapat digambarkan sebagai berikut:



Gambar 12. *Topological sort* kedua
Sumber : dokumen penulis

Dapat dilihat bahwa untuk membuat program Rumah pemain, rumah NPC, Bandara, Akademi, dan Istana setidaknya program kota telah ada. Sedangkan untuk pelabuhan membutuhkan program perarian. Untuk Pasar membutuhkan semua komponen peta karena pasar pada umumnya menjual produk dari berbagai tempat. Tempat pengrajin membutuhkan material dari berbagai tempat terkecuali perairan. Pasar dan tempat pengrajin setidaknya membutuhkan program kota di mana bangunan tersebut diletakkan.

Kemudian pemain yang memiliki komponen sebagai berikut:

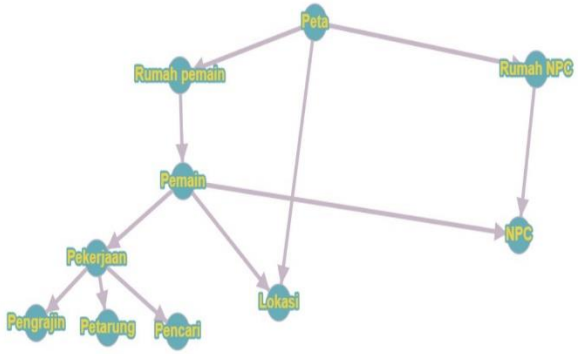
1) Pekerjaan
2) Rumah
3) Lokasi

Dalam permainan peran, pada umumnya pemain memiliki tiga jenis pekerjaan yaitu:

1) Petarung
2) Pengrajin
3) Pencari material

Komponen rumah menjelaskan di mana pemain tinggal dan komponen lokasi menunjukkan di mana pemain berada dan bagian mana saja yang dapat diakses pemain.

Berikutnya terdapat *non-player character* (NPC) dimana NPC yang memberikan tugas kepada pemain dan pemain dapat meningkatkan relasinya dengan NPC.



Gambar 13. *Topological sort* ketiga

Sumber : dokumen penulis

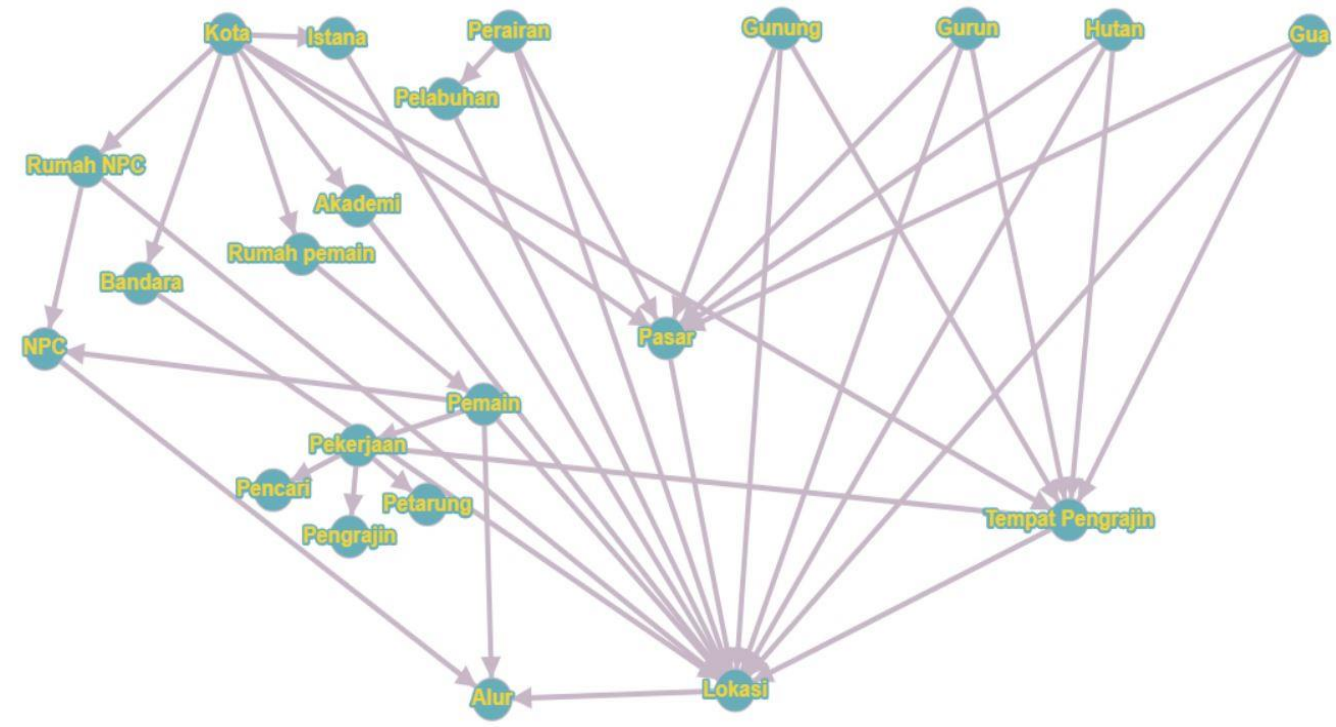
Setelah semua komponen selesai maka yang terakhir adalah membuat alur cerita. Alur cerita penting agar program permainan yang dibuat menarik. Selain itu alur cerita dapat dibuat terlebih dahulu agar tidak membingungkan ketika program lainnya sudah selesai. Depedensi alur cerita pada gambar 10 adalah memasukkan urutan cerita ketika permainan sudah selesai. Alur cerita ini pada gambar tersebut dapat juga disebut sebagai program utama.

Peta	Kota
	Perairan
	Gunung
	Gurun
	Hutan
	Gua
Bangunan	Rumah pemain
	Rumah NPC
	Pasar
	Bandara
	Pelabuhan
	Istana
	Akademi
	Tempat pengrajin
Pemain	Pekerjaan: 1. Petarung 2. Pengrajin 3. Pencari
	Rumah
	Lokasi
NPC	Relasi dengan pemain
Alur	Program utama

Dengan komponen yang telah dijelaskan maka gambaran akhir dari perencanaan program dapat dilihat pada gambar 14. Semakin detail komponen yang dijabarkan, maka akan semakin banyak depedensi yang ada. Untuk mempermudah programmer, maka *topological sort* dapat dibagi menjadi beberapa graf sehingga tidak membingungkan programmer.

Pada gambar 14 dapat dilihat bahwa suatu elemen yang memilki depedensi tidak dapat dibuat secara langsung oleh programmer jika program depedensi nya tidak dibuat terlebih dahulu. Jika programmer membuat tanpa depedensi nya, maka program akan kacau karena kekurangan suatu bagian yang mengakibatkan programmer yang satu dengan yang lain hanya saling menunggu sehingga efektifitas pengerjaan menjadi berkurang.

.



Gambar 14. *Topological sort* perencanaan program pembuatan permainan
Sumber : dokumen penulis

IV. KESIMPULAN

Pada masa ini teknologi semakin maju, permintaan terhadap teknologi meningkat. Oleh karena itu diperlukan suatu cara bagi programmer agar dapat bekerja dengan efektif dan efisien. *Topological* sort membantu programmer dalam menyusun rencana dan pembagian kerja dengan lebih mudah, meskipun tim nya terdiri dari programmer baru.

Topological sort yang mana menerapkan sistem depedensi sehingga programmer dapat membuat program dengan mengikuti keterurutan linier yang ada. Semakin detail komponen program dijelaskan, maka akan semakin kompleks graf yang dibuat, namun hal itu juga mempermudah jalur kerja programmer.

Adapun kekurangan dari pengaplikasian *topological sort* adalah semakin kompleks program maka pembuatan graf nya

juga semakin sulit terutama untuk program skala besar. Namun dibandingkan dengan hasil yang diperoleh tentu akan lebih efektif menggunakan *topological sort* dikarenakan programmer hanya akan kesulitan disaat membuat perencanaan dibandingkan ketika pembuatan program sudah berjalan dan tiba-tiba kekurangan komponen lainnya sehingga pekerjaan menjadi kacau.

Melalui penerapan algoritma *topological sort*, programmer diharapkan mampu membuat rencana program skala lebih besar dengan lebih efektif dan efisien sehingga pengerjaan program menjadi lebih mudah dan terarah.

UCAPAN TERIMA KASIH

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa atas segala berkat dan pertolongannya sehingga penulis dapat menyelesaikan makalah ini. Selain itu penulis juga mengucapkan syukur kepada :

1) Ibu Dra. Harlili M.Sc., Bapak Dr. Ir. Rinaldi Munir, MT. dan Ibu Fariska Zakhralativa Ruskanda, S.T., M.T., atas bimbingannya terkhusus Ibu Harlili selaku dosen pengajar di kelas K-02 yang telah memberikan ilmunya kepada kami sehingga makalah ini dapat diselesaikan.
2) Keluarga dan teman-teman yang selalu mendukung dan menolong penulis.

REFERENSI

[1] Kenneth H. Rosen, “Discrete Mathematics and Its Applications”, 7th ed, New York : The McGraw-Hill Companies, 2012.
[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf diakses pada 6 Desember 2019.
[3] https://www.sciencedirect.com/topics/computer-science/topological-sort diakses pada 6 Desember 2019.
[4] https://stei.kuliah.itb.ac.id/pluginfile.php/38937/mod_resource/content/1/W15_A3_SK6_TopologicalSort.pdf diakses pada 6 Desember 2019.

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, 6 Desember 2019

Lionnarta Savirandy 13518128