Aplikasi Topological Sort pada Kompilasi Makefile

Kevin Yudi Utama(13512010)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512010@std.stei.itb.ac.id

***Abstrak*—Topological sort merupakan salah satu aplikasi BFS dan DFS yang banyak digunakan. Salah satu kegunaan topological sort adalah dalam menentukan urutan kompilasi pada Makefile. Makefile merupakan salah satu tools kompilasi yang banyak digunakan oleh developer C dan developer C++.**

***Index Terms*—BFS , DFS , Teknik kompilasi, Topological sort.**

I. PENDAHULUAN

Topological sort merupakan salah satu permasalahan Graf yang banyak digunakan . Salah satunya digunakan oleh Makefile untuk menentukan urutan kompilasi program. Makefile merupakan salah satu aplikasi open source yang sering digunakan untuk membantu kompilasi projek yang besar. Makefile merupakan tools yang akrab oleh para developer terutama developer C dan developer C++.

Karena adanya depedensi antar objek, maka pada saat kompilasi program Makefile harus menetukan urutan kompilasi yang benar. Urutan kompilasi program merupakan hal yang vital. Jika program dikompilasi dengan urutan yang salah maka akan menyebabkan kesalahan pada kompilasi dan hasil kompilasi tidak dapat dihasilkan.

Penulis memilih topik ini berawal dari rasa ingin tahu bagaimana cara kerja kompilasi mengunakan Makefile. Penulis ingin menganalisis cara kerja kompilasi Makefile serta algoritma apa yang digunakan.

Pada makalah ini penulis akan membahas algoritma topological sort dengan menggunakan DFS. Penulis juga akan menjabarkan alternatif cara lain yang mungkin dapat digunakan untuk mengimplemtasikan topological sort.

Penulis berharap hasil makalah ini dapat dijadikan bahan referensi kepada mahasiswa lain dan membantu mahasiswa lain mempelajari DFS serta topological sort.

II. TEORI DASAR

A. Graf

Graf adalah sebuah struktur yang terdiri dari kumpulan simpul – simpul serta ketetrhubunganya satu sama lain. Setiap graf paling sedikit memiliki satu simpul. Graf dapat direpresentasikan dengan menggunakan himpunan simpul atau V dan himpunan sisi atau E.

Graf terdiri dari berbagai jenis. Berdasarkan ada tidaknya gelang dan sisi ganda, graf dapat dikelompokkan menjadi graf sederhana dan graf tidak sederhana. Graf sederhana adalah graf yang tidak memiliki dua buah sisi yang menghubungkan dua buah simpul yang sama dan tidak terdapat gelang. Gelang adalah sebuah sisi yang menghubungkan sebuah simpul ke simpul itu sendiri

Berdasarkan ada tidaknya orientasi atau arah pada simpul-simpul graf,graf dapat dibagi menjadi graf berarah dan graf tidak berarah. Graf berarah adalah graf yang setiap sisinya mempunyai arah dari sebuah simpul ke simpul lainya. Sedangkan graf tidak berarah adalah graf yang setiap sisinya tidak memiliki arah

Berdasarkan ada tidaknya bobot pada simpul – simpul graf. Graf dapat dibagi menjadi graf berbobot dan graf tidak berbobot. Sebuah graf dikatakan berbobot apabila setiap simpulnya diberikan sebuah nilai. Sebuah graf dikatakan tidak berbobot apabila setiap simpulnya tidak memiliki nilai.

Selain jenis-jenis graf diatas masih terdapat jenis-jenis graf lainya seperti graf asiklik tidak berarah, graf turnamen, dan graf –graf lainya. Selain itu, sifat – sifat dari graf dapat dikombinasikan seperti graf sederhana tidak berbobot, yaitu graf yang tidak memiliki gelang dan simpul ganda serta tidak memiliki arah atau orientasi pada simpul-simpulnya.

Selain terminologi – terminologi di atas terdapat beberapa terminologi lain yang harus dipahami.

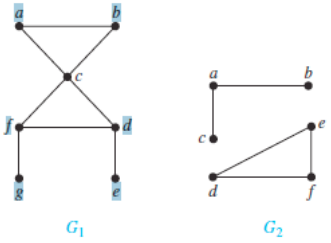
1. Lintasan dan Sirkuit

Lintasan dengan panjang n dari u ke v adalah barisan n buah sisi $e_1,e_2,\ldots,e_{n-1},e_n$ dimana terdapat barisan simpul $v_0=u,v_1,v_2,\ldots..,v_n=v$, demikian sehingga $e_1=(v_0,v_1)$, $e_2=(v_1,v_2)$ …., $e_n=(v_{n-1},v_n)$. Apabila u=v maka lintasan tersebut juga dapat dikatakan sebagai

sirkuit. Sebuah sirkuit dan lintasan dikatakan sederhana apabila sirkuit dan lintasan tersebut tidak melewati dua buah simpul yang sama

2. Terhubung

Dua buah simpul a dan b dikatakan terhubung apabila terdapat lintasan dari a ke b, untuk semua pasangan a dan b dimana a dan b merupakan anggota V.



GAMBAR 1. GRAF TERHUBUNG G1 DAN GRAF TIDAK TERHUBUNG G2(ROSEN, 2013)

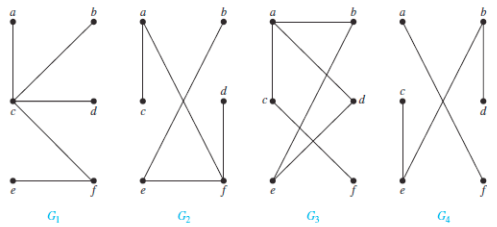
Sebuah graf tidak berarah (V,E) dikatakan terhubung apabila terdapat lintasan untuk setiap pasangan vertex a dan b dimana a dan b merupakan anggota V.

Sebuah graf berarah (V,E) dikatakan terhubung kuat apabila terdapat lintasan untuk setiap pasangan simpul a dan b dimana a dan b merupakan anggota himpunan V.

Sebuah graf berarah(V,E) dikatakan terhubung lemah apabila jika kita mengabaikan arahnya terdapat lintasan untuk setiap pasangan simpul a dan b dimana a dan b merupakan anggota himpunan V.

3. Pohon

Pohon adalah graf tidak berarah yang tidak memiliki sirkuit sederhana. Sebuah graf tidak berarah merupakan pohon jika dan hanya jika terdapat lintasan sederhana unik untuk semua pasangan sisi a dan b.



GAMBAR 2. GRAF G1 DAN G2 MERUPAKAN POHON SEDANGKAN G3 DAN G4 BUKAN MERUPAKAN POHON

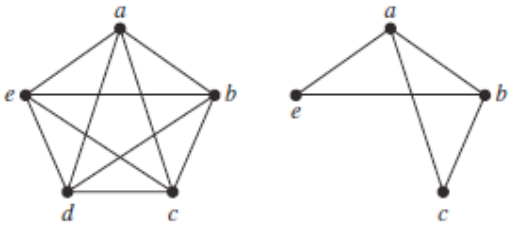
(Rosen ,2013)

Pohon merentang dari sebuah graf sederhana adalah upagraf yang merupakan pohon yang mengandung semua simpul pada graf tersebut.

Pohon merentang minimum dari sebuah graf berbobot sederhana adalah pohon merentang yang memiliki jumlah bobot paling sedikit.

4. Upagraf

Upagraf dari graf(V,E) adalah graf(V',E') dimana V merupakan himpunan bagian tidak kosong dari V dan E' adalah himpunan bagian dari E.



GAMBAR 3. UPAGRAF DARI GRAF K5 (ROSEN, 2013)

B. DFS

DFS merupakan salah satu algoritma traversal graf. Kepanjangan dari DFS adalah Depth First Search. Pseudocode dari algoritma DFS adalah sebagai berikut :

```
Input : Graf G dan vertex v
Output : Semua vertex yang dapat dicapai dari v
Prosedur DFS (G,v)
    Hasil.add(v)
    Tandai v sudah dikunjungi
    for all semua sisi dari v ke w
        if jika w belum ditandai then
            DFS(G,w)
```

C. Topological sort

Topological sort adalah algoritma untuk mengurukan simpul – simpul pada graf berarah asiklik dimana untuk setiap sisi dari a ke b maka a berada sebelum b pada

urutan. Topological sort dapat diimplementasikan dengan menggunakan DFS. Topological sort memilki beberapa aplikasi seperti penjadwalan kerja jika diberikan depedensi pada pekerjaan tersebut. Topological sort juga digunakan pada cell evaluation, logic synthesis . dan menentukan urutan kompilasi serta mengatasi depedinsi symbol pada linker. Berikut adalah pseudocode dari Topological sort.

```
L <- List kosong yang akan mengandung elemen
yang diurutkan
AdjList <- AdjacencyList dari Graph G

Prosedur dfs2(int u)
    Tandai u sudah dikunjungi
    for j<-0 , j<AdjList[u].size();j+=1
        v<-AdjList[u][j]
        if (v belum ditandai)
            dfs2(v)
    L.add(u);

Prosedur topologicalSort
    For i<-0 , i< jumlah simpul , i++
        dfs2(i)
    i<-L.size()
    while  i>=0
        print L.get(i-1)
```

III. MAKEFILE

A.Definisi dan arti

Makefile adalah sebuah file khusus yang berisi shell command dan kita namai dengan makefile atau Makefile. Untuk menjalankan perintah yang ada di dalam Makefile kita dapat mengetikkan make pada command line. Make ada utilitas UNIX yang membaca Makefile dan mengeksekusinya .

Make mengingat konfigurasi files terakhir kali dan hanya melakukan kompilasi pada bagian file yang berubah untuk menjaga proyek tetap yang terbaru. Jika kita memiliki program yang besa yang terdiri dari banyak file , ketika kita mengubah sutau file yang merupakan depedensi dari file lain kita harus melakukan kompilasi ulang. Tanpa bantuan makefile ini merupakan tugas yang memakan waktu yang besar.

B. Sejarah Makefile

Makefile pertama kali dikembangkan oleh Sturart Feldman pada bulan April tahun 1997 di Ball Labs. Pada tahun 2003 Dr.Feldman mendapatkan ACM Software System Award dalam mengembangkan aplikasi yang banyak digunakan ini

C. Struktur Makefile

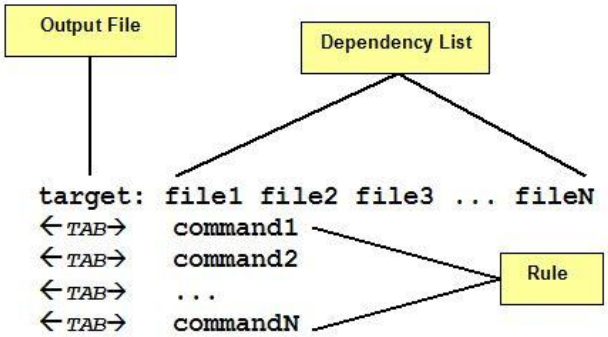
Makefile berisi perintah – peritah yang merupakan kelas bahasa deklaratif. Setiap makefile tersusun dari berbagai komponen – komponen. Beberapa komponen tersebut adalah :

1. Rules

   Setiap makefile tersusun dari rules- rules yang terdiri dari 2 baris. Baris pertama adalah Depedency Line dan baris kedua adalah command.

   Setiap depedensi line dibagi menjadi 2 bagian. Bagian pertama adalah target files dan bagian kedua adalah source files. Apabila terjadi perubaha pada source files maka make akan membangun target files dengan menggunakan command yang disediakan pada baris kedua. Makefile menentukan urutan kompilasi dengan menggunakan topological sort .

   Skema dari Makefile dapat dilihat dibawah



GAMBAR 4. SKEMA MAKEFILE PADA UNIX

(www.codeproject.com)

2. Macros

   Macros adalah sejenis variable yang digunakan didalam Makefile. Cara melakukan definisi pada macro adalah dengan sintaks
   MACRO = DEFINITION

   Jika kita ingin menggunakan macro, maka kita tutup nama macro tersebut dengan $(), misalnya

   g++ -o $(OBJ).o $(CC).o

3. Suffix Rules

   Suffix Rules merupakan cara lama dalam mendifiniskan rules yang implisit pada make. Suffix Rules didukung oleh GNU untuk mempertahankan backward compatibility dengan makefile yang lama. Suffix Rules terdiri dari dua jenis yaitu Double suffix dan single suffix.

   Double suffix rule didefinisikan dengan sepasang buah suffix. Suffix target dan suffix source.

   Single suffix rule didefiniskan dengan sebuah suffix yaitu suffix dari source

4. Pattern Rules

   Pattern rules terlihat seperti rules biasa , tetapi pattern rules memiliki target dengan tepat satu buah

karakter ‘%’. ‘%’ tersebut digunakan sebagai pattern matching pada file

IV. PENGGUNAAN TOPOLOGICAL SORT PADA KOMPILASI MAKEFILE

A. Representasi MakeFile dalam Bentuk Graf

Pada makalah ini kita akan menggunakan depedensi Makefile berikut sebagai bahan analisa :

Tabel 1 Tabel Depedensi

Objek	Depedensi
DateTime.o	Date.o
DateTime.o	Time.o
DateTimeEvent.o	DateTime.o
DateTime.o	Event.o
DateTimeEvent.o	Event.o
Queue.o	Element.o
ArrayQueue.o	Queue.o
ArrayQueue.o	Element.o
System.o	ArrayQueue.o
System.o	DateTime.o

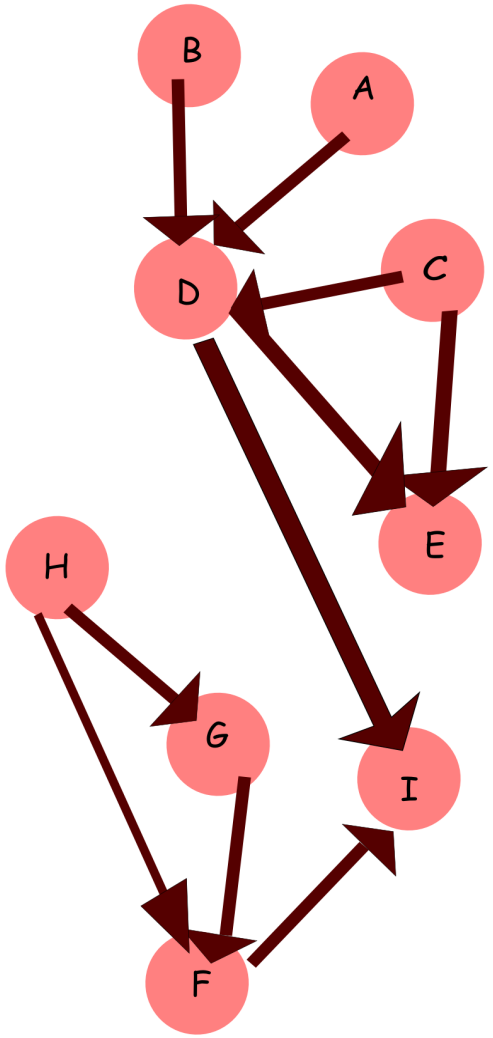
Objek – objek pada MakeFile direpresentasikan sebagai simpul pada graf. Setiap depedensi digambarkan sebagai sisi pada graf. Jika objek A merupakan salah satu depedensi dari objek B maka terdapat sisi yang berarah dari simpul A ke simpul B. Graf yang didapatkan adalah graf tidak berbobot , berarah dan asiklik. Graf tersebut mungkin saja tidak terhubung. Jika graf yang didapatkan bukan merupaka graf DAG berarti tidak topological order pada graf atau depedensi tersebut. Hal ini terjadi mungkin karena adanya kesalahan pengguna dalam mendefinisikan depedensi antar file.

Hasil pemetaan Objek menjadi simpul – simpul

Tabel 2 Tabel pemetaan Objek menjadi simpul simpul

Objek	Simpul
Date.o	A
Time.o	B
Event.o	C
DateTime.o	D
DateTimeEvent.o	E
ArrayQueue.o	F
Queue.o	G
Element.o	H
System.o	I

Hasil representasi Graf



GAMBAR 5. REPRESENTASI GRAF DAG UNTUK TABEL DEPEDENSI .

B. Pembangunan Urutan Kompilasi Objek

Tahapan selanjutnya setelah merepresentasikan MakeFile sebagai graf adalah membangun urutan kompilasi objek. Ada berbagai cara yang dapat digunakan dalam membangun algoritma topological sort. Cara tersebut memiliki kompleksitas yang bervariasi satu sama lain.

Cara yang pertama adalah dengan menggunakan brute force. Pertama – tama kita enumerasi semua kemungkinan urutan. Kemudian kita periksa satu – persatu urutan tersebut. Urutan yang valid adalah apabila untuk semua sisi yang mnghubungkan simpul a dan simpul b maka simpul a terdapat sebelum simpul b.

Cara yang kedua adalah dengan DFS. Pertama kita menadai setiap simpul sebagai simpul yang belum dikunjungi. Kemudian kita lakukan traversal secara DFS.

Untuk setiap simpul yang sudah tidak bertetangga dengan simpul lain yang belum dikunjungi, kita tambahkan simpul tersebut ke dalam list. Hasil akhirnya adalah list yang berisi urutan dalam keadaan terbalik.

Cara yang ketiga adalah dengan menggunakan BFS. Pertama kita tandai semua simpul sebagai simpul yang belum dikunjungi. Kemudian kita lakukan traversal secara BFS. Untuk setiap simpul yang sudah tidak bertetangga dengan simpul lain yang belum dikunjungi maka kita tambahkan ke dalam list. Sama seperti DFS, hasil akhirnya adalah list yang berisi urutan dalam keadaan terbalik.

C. Pemilihan Algoritma

Baik BFS maupun DFS memiliki kompleksitas waktu yang sama yaitu O(V+E) dimana E adalah jumlah sisi pada graf dan V adalah jumah sipul pada graf. Sedangkan pada cara brute force diperlukan kompleksitas waktu eksponensial.

Dengan menggunakan BFS dan DFS hanya didapatkan satu buah urutan yang valid. Namun dengan menggunakan brute force dapat didapatkan semua urutan yang valid. Pada aplikasinya MakeFile hanya membutuhkan satu buah urutan yang valid jadi pada kasus ini BFS dan DFS jauh lebih baik dibandingkan cara brute force.

Pada makalah ini kita akan menggunakan algoritma DFS karena keuntunganya dibandingkan algoritma brute force. Sedangkan algoritma BFS juga dapat dijadikan alternatif lain dengan kompleksitas waktu yang sama dengan DFS.

D. Pengaplikasian Algoritma pada Graf MakeFile

Dengan menggunakan Algoritma DFS didapatkan hasil seperti berikut :

Tabel 1 Tabel Traversal DFS

Simpul Awal	Lintasan traversal	List Akhir
A	A - D - E - I	E-I-D-A
B	B	E-I-D-A-B
C	C	E-I-D-A-B-C
F	F - I	E-I-D-A-B-C-F
G	G	E-I-D-A-B-C-F-G
H	H	E-I-D-A-B-C-F-G-H

Dari table di atas didapatkan urutan kompilasi adalah H-G-F-C-B-A-D-I-E atau Element.o - Queue.o - ArrayQueue.o – Event.o – Time.o – Date.o – DateTime.o – System.o – DateTimeEvent.o.

Kita dapat melakukan verifikasi terhapad hasil yang diberikan dengan mengecek setiap depedensi Objek. Jika terdapat depedensi dimana Objek A merupakan salah satu depedensi Objek B dan Objek A berada setelah Objek B maka Algorima tersebut gagal. Setalah melakukan verifikasi didapatkan bahwa hasil topological sort di atas benar.

Selain solusi yang diberikan diatas, ada juga solusi lain pada graf tersebut misalnya H-G-F-C-A-B-D-I-E . Solusi ini bias didapatkan dengan Algoritma brute force atau dengan cara mengubah urutan traversal untuk simpul – simpul yang setingkat

V. KESIMPULAN

Algoritma DFS cukup mangkus dan efektif dalam melakukan topological sort . Urutan yang didapatkan hamya satu buah namun merupakan solusi optimal. Algoritma ini memiliki kompleksitas waktu O(V+E) dengan V adalah jumlah vertex dan E adalah jumlah sisi pada graf. Algoritma ini juga hanya bekerja pada DAG karena hany graf DAG yang memiliki topological order.

Selain DFS topological sort juga dapat dilakukan dengan cara brute force dan BFS. BFS memiliki kompleksitas yang sama dengan DFS dan dapat dipakai sebagai solusi alternatif untuk topological sort. Selain memiliki kompleksitas waktu yang sama BFS juga hanya menghasilkan satu buah solusi. Brute force dapat memberikan lebih dari satu buah solusi namun memiliki kompleksitas waktu eksponensial.

Algoritma ini dapat dimodifikasi dan disesuaikan dengan fitur-fitur permainan yang ada. Selain permainan *maze* algoritma ini juga dapat digunakan untuk permainan yang memiliki titik mulai dan tujuan serta jalan terpendek menuju tujuan merupakan hal yang penting untuk dipertimbangkan.

Selain untuk mengurutkan urutan kompilasi topological sort juga digunakan dalam cell evaluation, logic synthesis serta mengatasi depedinsi symbol pada linker dan memiliki banyak aplikasi pada database modeling.

VI. PENGHARGAAN

Penulis bersyukur kepada Tuhan yang telah memberikan kemampuan kepada penulis untuk dapat menyelesaikan makalah ini. Terima kasih juga kepada orang tua yang telah membesarkan penulis hingga dapat berkarya saat ini. Penulis juga berterima kasih kepada dosen mata kuliah Strategi Algoritma semester genap 2013/2014, Dr. Ir. Rinaldi Munir, MT. dan Dr. Masayu Leylia Khodra, ST., MT. yang telah memberikan bimbingan. Tidak lupa juga penulis berterima kasih kepada teman-teman yang telah memberikan inspirasi untuk makalah ini.

REFERENSI

[1] Munir, Rinaldi, *Strategi Algoritma*, 4th, Teknik Informatika ITB, 2006.

[2] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford, *Introduction to Algorithms*, 3rd, MIT Press and McGraw-Hill, 2009.

[3] Munir, Rinaldi, *Matematika Diskrit*, 4th, Teknik Informatika ITB, 2006.

[4] Rosen, Kenneth H., *Discrete Mathematics and Its Applications*, 7th, McGraw-Hill, 2012.

[5] http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/topoSort.htm

[6] https://www.gnu.org/software/make/manual/html_node/Suffix-Rules.html

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, 19 Mei 2014

Kevin Yudi Utama
13512010