Penerapan Pohon Berakar dalam Pembentukan Folder pada Aplikasi *Desktop* Komputer

Rheno Manggala Budiasa - 13506119

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
e-mail: if16119@students.if.itb.ac.id

ABSTRAK

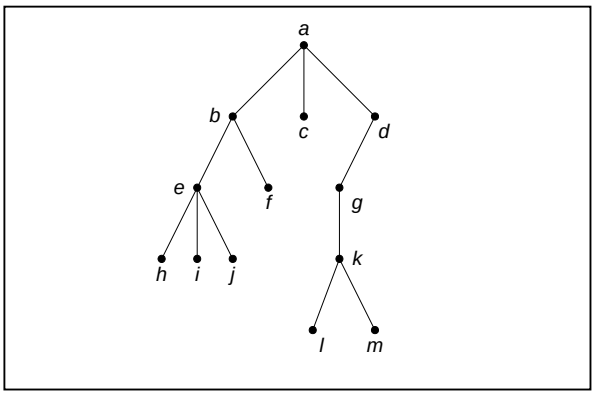
Makalah ini membahas tentang aplikasi pohon berakar dalam pembentukan folder, khususnya pada *desktop* komputer. Folder biasa kita temui pada media penyimpanan data seperti *HardDisk*, *Flash Disk*, *CD* dan lainnya. Beberapa sistem operasi seperti Linux (beserta varian distronya), Windows, Unix (Beserta variannya) dan lainnya, menerapkan struktur ini dalam pembuatan dan penamaan suatu folder. Pohon juga banyak digunakan oleh sistem operasi tertentu untuk keperluan lain seperti *File System*. Kebanyakan sistem operasi yang menggunakan *Desktop* berbasis GUI (*Graphic User Interface*) memudahkan pengguna (*User*) dalam membuat sekaligus memodifikasi foldernya. Khusus untuk aplikasi yang tidak berbasis GUI, user dapat membuat folder melalui perintah *command-line* yang tersedia pada hampir setiap sistem operasi. Pembuatan Kernel pada linux juga banyak menggunakan struktur data pohon. Karena strukturnya yang rekursif pohon juga banyak digunakan untuk proses komputasi pada sistem operasi tertentu. Jenis pohon yang biasa digunakan dalam membuat folder adalah N-Ary Tree. Hampir semua bahasa pemrograman tingkat tinggi maupun menengah sudah mengimplementasikan struktur data pohon.

Kata kunci: Pohon, Folder, *Hard Disk*, *Flash Disk*, CD, Desktop, Sistem Operasi, *File System*, GUI, pengguna, *Command-Line* Kernel, Komputasi, *N-Ary Tree*, Bahasa Pemrograman, Struktur Data.

I. PENDAHULUAN

Sebelum membahas lebih jauh mengenai topik di atas ada baiknya kita mengetahui dulu apa arti dari pohon itu sendiri. Pohon adalah graf tak-berarah terhubung dan tidak mengandung sirkuit (Pohon,Bahan Kuliah IF2091,Rinaldi Munir). Pohon juga dapat diartikan sebagai graf terhubung yang tidak membentuk suatu siklus (*Course Notes 4*, Professor Albert Meyer and Dr. Radhika Nagpal, MIT). Kedua Pengertian ini sama-sama menjelaskan bahwa pohon merupakan suatu graf berarah. Dalam matematika atau struktur diskrit istilah graf sendiri sering disebut sebagai kumpulan dari beberapa himpunan tidak kosong simpul-simpul yang dihubungkan oleh beberapa atau paling sedikit satu sisi (Graf (bagian 1),Bahan Kuliah IF2091,Rinaldi Munir).

Khusus untuk pembahasan kali ini kita hanya akan membahas Pohon Berakar. Secara umum struktur pohon berakar terbagi menjadi 2 bagian yaitu Akar dan Daun. Akar, yaitu simpul dari suatu pohon yang paling atas. Akar juga dapat dikatakan struktur pohon yang paling sederhana. Dikatakan pohon sederhana karena hanya terdiri dari satu simpul. Bagian lain dari pohon adalah Daun, yaitu kumpulan beberapa simpul yang berada di bawah akar dan tidak mempunyai anak.

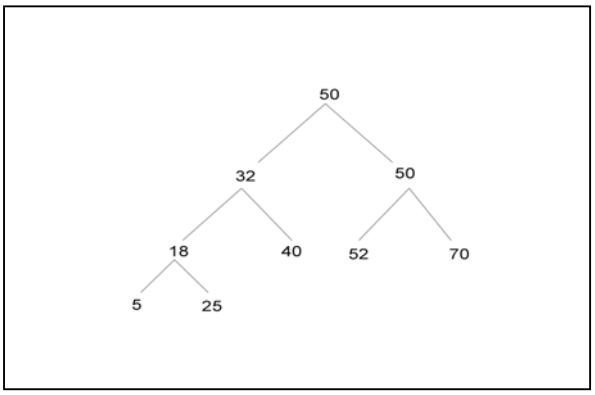


Gambar 1. Struktur Pohon Berakar

Gambar 1 menunjukkan sebuah pohon berakar dengan Akar (a) dan Daun (h,i,j,f,c,l,m).

Pohon beradasarkan jumlah anaknya terbagi menjadi 2 bagian yaitu *N-ary* dan *Binary*. Suatu pohon dikatakan *N-ary* jika setiap simpul cabangnya mempunyai paling banyak n-buah anak. Sedangkan suatu pohon dikatakan *Binary* jika setiap simpul cabangnya mempunyai paling banyak 2 buah anak. Khusus untuk Penamaan *folder*

pohon yang sesuai adalah pohon *N-ary*. Gambar 1 merupakan salah satu bentuk pohon *N-ary*.



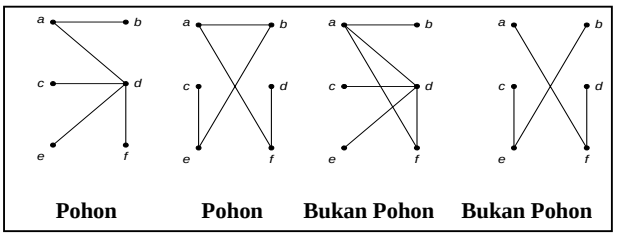
Gambar 2. Contoh Pohon *Binary*

II. METODE

Sesuai dengan pendahuluan sebelumnya bahwa proses pembuatan *folder* pada suatu sistem operasi (misal Macintosh) sangat banyak dipengaruhi oleh struktur pohon khususnya *N-ary* mari kita lihat implementasinya.

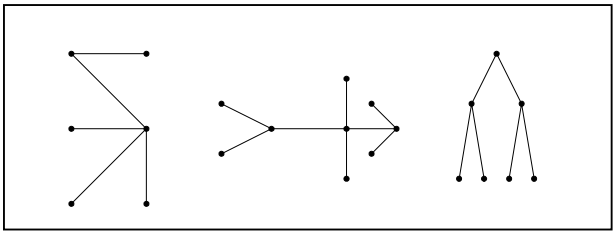
II.1 Pohon

Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit



Gambar 3 Pohon

Hutan (forest) adalah
- kumpulan pohon yang saling lepas, atau
- graf tidak terhubung yang tidak mengandung sirkuit. Setiap komponen di dalam graf terhubung tersebut adalah pohon.



Gambar 4 Fores

II.1.1 Sifat-Sifat Pohon

Misalkan $G = (V, E)$ adalah graf tak-berarah sederhana dan jumlah simpulnya n. Maka, semua pernyataan di bawah ini adalah ekivalen:
1. G adalah pohon.
2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
3. G terhubung dan memiliki $m = n - 1$ buah sisi.
4. G tidak mengandung sirkuit dan memiliki $m = n - 1$ buah sisi.
5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
6. G terhubung dan semua sisinya adalah jembatan.

II.2 Pohon Merentang (*SpanningTree*)

- Pohon merentang dari graf terhubung adalah upagraf merentang yang berupa pohon.
- Pohon merentang diperoleh dengan memutus sirkuit di dalam graf.
- Setiap graf terhubung mempunyai paling sedikit satu buah pohon merentang.
- Graf tak-terhubung dengan k komponen mempunyai k buah hutan merentang yang disebut hutan merentang (spanning forest).

II.3 Pohon Berakar

Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf dinamakan berarah dinamakan **pohon berakar** (*rooted tree)*.

II.3.1 Terminologi Pohon Berakar

Perhatikan gambar 1 pada bab pendahuluan :

1. Anak (*child* atau *children*) dan Orangtua (*parent*)

b, *c*, dan *d* adalah anak-anak simpul *a*,
a adalah orangtua dari anak-anak itu

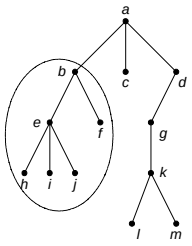
2. Lintasan (*path*)

Lintasan dari a ke j adalah a,b,e,j.
Panjang lintasan dari a ke j adalah 3

3. Saudara Kandung (*Sibling*)

f adalah saudara kandung *e*, tetapi, *g* bukan saudara kandung *e*, karena orangtua mereka berbeda.

4. Upapohon (*subtree*)



Gambar 5 Upapohon

5. Derajat (degree)

Derajat sebuah simpul adalah jumlah upapohon (atau jumlah anak) pada simpul tersebut.

Pada gambar 1 Derajat *a* adalah 3, derajat *b* adalah 2, Derajat *d* adalah satu dan derajat *c* adalah 0.

Jadi, derajat yang dimaksudkan di sini adalah derajat-keluar.

Derajat maksimum dari semua simpul merupakan derajat pohon itu sendiri. Pohon di atas berderajat 3

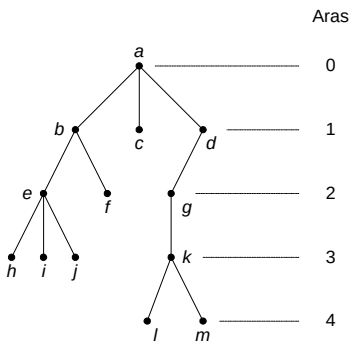
6. Daun (leaf)

Simpul yang berderajat nol (atau tidak mempunyai anak) disebut **daun**. Simpul *h, i, j, f, c, l,* dan *m* adalah daun.

7. Simpul Dalam (*internal nodes*)

Simpul yang mempunyai anak disebut simpul dalam. Simpul b, d, e, g, dan k (gambar1) adalah simpul dalam.

8. Aras (*level*) atau Tingkat



Gambar 6 Aras

9. Tinggi (*height*) atau Kedalaman (*depth*)

Aras maksimum dari suatu pohon disebut **tinggi** atau **kedalaman** pohon tersebut. Pohon di atas mempunyai tinggi 4.

II.4 Pohon *N-ary*

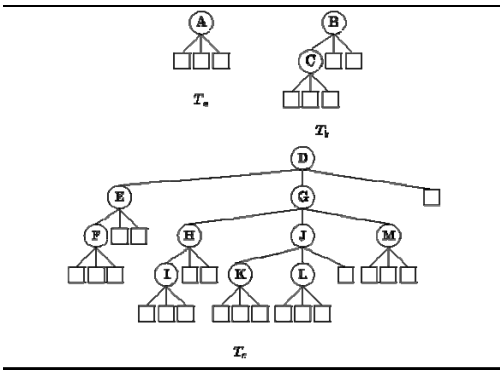
Seperti yang sudah dijelaskan sebelumnya suatu pohon dikatakan *N-ary* jika setiap simpul cabangnya mempunyai paling banyak n-buah anak. Karena itu pohon dengan hanya berupa sebuah simpul dapat kita sebut juga sebagai pohon *N-ary* namun tanpa anak. Kita dapat mengimplementasikan pohon dengan notasi himpunan sebagai berikut :

T_a = {A,Ø,Ø,Ø},
T_b = {B,{C,Ø,Ø,Ø},Ø,Ø},
T_c = {D,{E,{F,Ø,Ø,Ø},Ø,Ø},
{G,{H,{I,Ø,Ø,Ø},Ø,Ø},{J,{K,Ø,Ø,Ø},{L,Ø,Ø,Ø},Ø},
{M,Ø,Ø,Ø}},Ø}.

Ø = null

T_a menyatakan pohon dengan satu buah akar (A) dan simpul dengan 3 anak null. T_b menyatakan pohon dengan dua buah simpul (B dan C). T_c menyatakan pohon dengan 10 buah simpul. Nilai null menyatakan bahwa pohon tersebut kosong atau tidak mempunyai anak. Null juga dapat berarti daun.

Representasi pohon dari notasi himpunan tersebut adalah :



Gambar 7. Representasi pohon dari notasi Himpunan

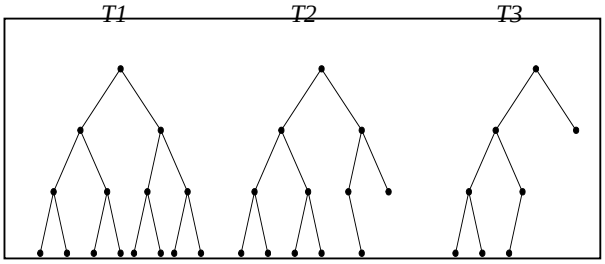
II.5 Pohon Biner (*Binary Tree*)

- Adalah pohon *n-ary* dengan *n* = 2.
- Pohon yang paling penting karena banyak aplikasinya.

- Setiap simpul di adlam pohon biner mempunyai paling banyak 2 buah anak.
- Dibedakan antara anak kiri (*left child*) dan anak kanan (*right child*). Karena ada perbedaan urutan anak, maka pohon biner adalah pohon terurut.

II.6 Pohon Biner Seimbang

Pada beberapa aplikasi, diinginkan tinggi upapohon kiri dan tinggi upapohon kanan yang seimbang, yaitu berbeda maksimal 1.



Gambar 8 T_1 dan T_2 adalah pohon seimbang, sedangkan T_3 bukan pohon seimbang.

III. Proses

Proses pembuatan *folder* pada aplikasi *desktop* dari suatu sistem operasi sangatlah mudah. Misalnya pada sistem operasi Macintosh, Pengguna tinggal mengklik kanan kemudian pilih *new folder*

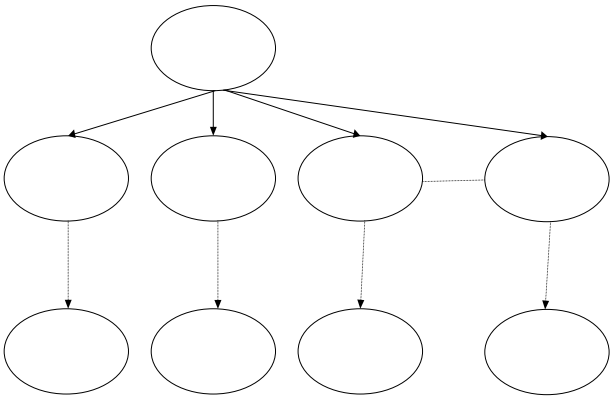
Setelah memilih *new folder* maka akan terbentuk *file* baru dengan nama *default* 'untitled folder'. Folder tersebut dapat kita beri nama sesuka hati. Kita juga dapat membuat *folder* baru di dalam *folder* yang telah kita buat.

Untuk sistem operasi lainnya seperti Windows (seluruh versi) proses pembuatan dan modifikasi *folder* juga sama seperti Macintosh.

Gambar 9. Proses pembentukan folder pada system operasi Macintosh

Seandainya nama *folder* yang sudah dinamakan tadi kita anggap sebagai Akar sekaligus Orangtua dari suatu pohon. Maka *folder-folder* lain yang ada di dalam akar kita anggap sebagai Sub-Pohon dari akar tersebut.

Cara kerja dari struktur pohon ini adalah Setiap pembuatan *folder* baru maka pada satu Akar (*folder1*) yang sama, akan membangkitkan atau membuat upapohon berikutnya (*folder2, folder3, folder4,…,folder-n*). Selanjutnya apabila upapohon tersebut bukan daun atau tidak memiliki anak lainya maka *folder* tadi tidak akan membangkitkan kembali anak-anaknya.



Gambar 10. Pohon pembentuk *folder*

Terlihat jelas bahwa setiap *folder* yang dibuat oleh *folder* sebelumnya akan membentuk Upapohon. Secara rekursif Upapohon akan terus terbentuk sebanyak *folder* yang akan dibuat. Kemampuan ini akan membuat pembuatan folder atau upapohon akan menjadi tidak terbatas. Pengguna dapat membuat *folder* berapa pun banyaknya dan dimana pun tempatnya.

Selain dengan cara di atas, beberapa sistem operasi lain ada yang menerapkan *command-line* dalam membuat *folder*-nya (misalnya minix). Secara konsep, aturan dan struktur yang digunakan sama dengan sebelumnya.

IV. KESIMPULAN

Beberapa kesimpulan yang dapat diambil adalah :

- Alikasi pohon pada bidang komputer sangat luas antara lain *Crawler* yaitu *agent* pencari situs yang bertugas untuk mencari alamat maupun content dari web.
- Beberapa bahasa pemrograman sudah mengimplementasikan pohon secara langsung seperti C++, java, C#.
- Struktur pohon sangat sesuai untuk pembuatan *folder* pada aplikasi desktop pada suatu sistem operasi karena strukturnya yang rekursif.
- Karena kemampuannya yang rekursif struktur ini jauh lebih optimal ketimbang iterasi biasa.
- Penerapan struktur pohon dalam penambahan dan modifikasi *folder* lebih cocok dari struktur lainnya.

REFERENSI

http://www.brpreiss.com/books/opus5/html/page257.html.

http://www.informatika.org/~rinaldi.

Rinaldi Mnir, Matematika Diskrit, 2006, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

MIT Lecture Notes 2009