# 6. Struktur Tree

**Oleh : Ade Nurhopipah**

**Pokok Bahasan :**

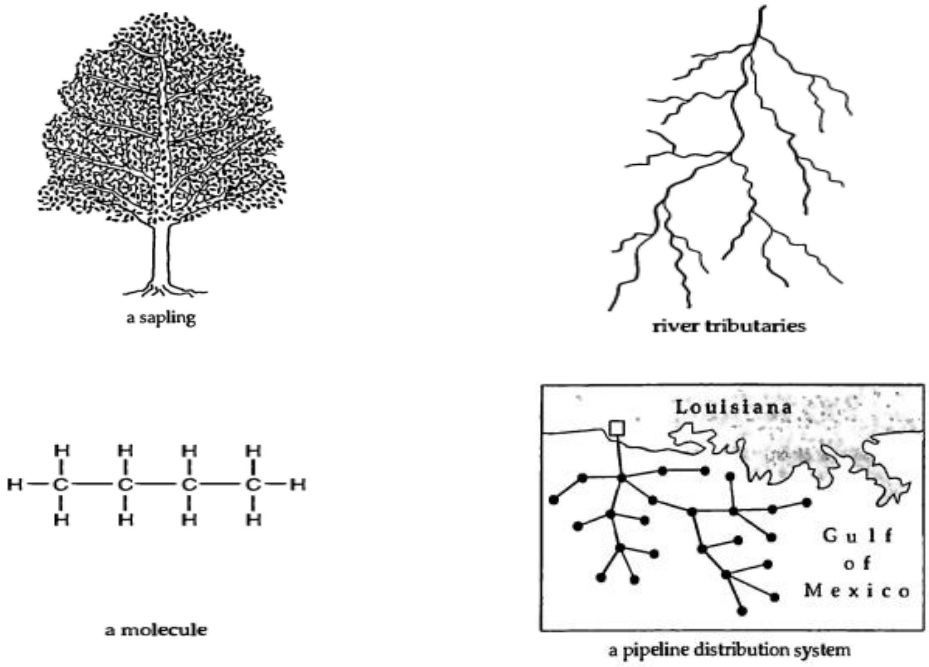
1. Properties of Trees
2. Spanning Trees
3. Rooted Trees

**Sumber :**

Aldous, Joan M. ,Wilson, Robin J. 2004. *Graph and Applications*. Springer: UK.

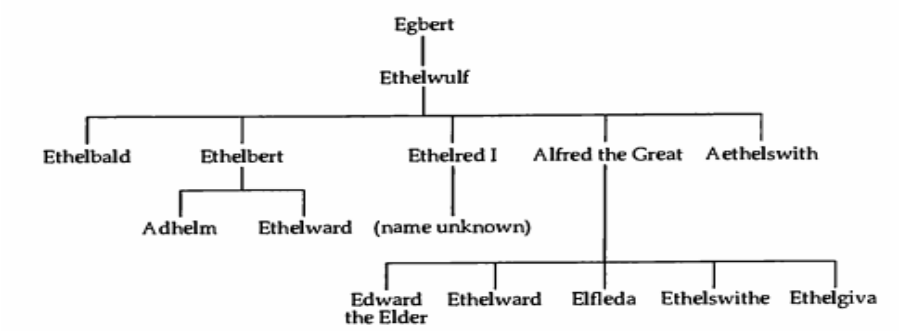
Tree (pohon) merupakan suatu graph yang memiliki struktur yang sederhana, namun merupakan dasar dari teknik penting yang banyak dipakai untuk membangun model atau desain pada berbagai sistem. Sekarang ini tree banyak digunakan dalam bidang ilmu komputer, pengambilan keputusan, pemrosesan bahasa, desain sistem saluran pipa gas, desain molekul kimia, dan lain-lain.

Tree digunakan untuk menggambarkan suatu situasi baik itu secara fisik maupun secara konsep yang mirip dengan struktur pohon. Contoh kondisi yang secara fisik/ alami memiliki struktur seperti tree adalah bentuk anak sungai dan bentuk struktur kimia pada molekul organik. Adapula contoh dari struktur tree buatan misalnya tree untuk sistem distribusi pipa minyak atau gas. Contoh struktur tree ditunjukan pada Gambar 6.1

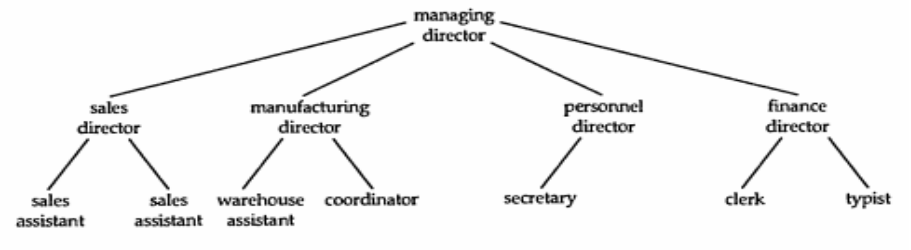



**Gambar 6.1 Contoh Struktur Yang Menyerupai Pohon Secara Fisik**

Banyak tree yang tidak memiliki definisi struktur fisik, numun memiliki bentuk secara konseptual. Misalnya saja tree sebagai pohon keluarga dan pohon hirarki. Pohon keluarga menggambarkan hubungan nenek moyang dengan keturunannya, sedangkan pohon hirarki, misalnya pada suatu perusahaan menggambarkan garis pertanggungjawaban.




(a) Pohon keluarga



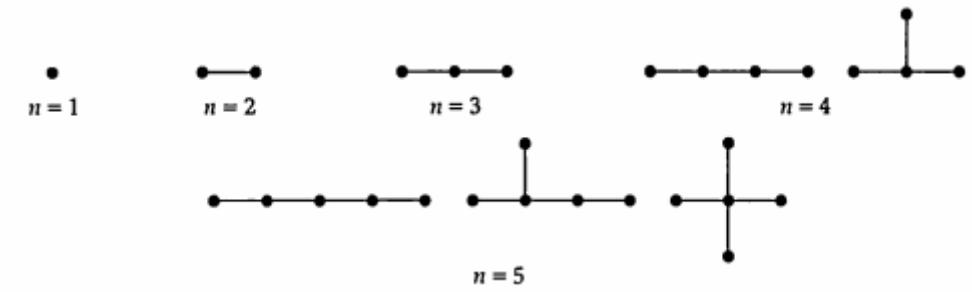

(b) Pohon hirarki

**Gambar 6.2 Contoh Struktur Yang Membentuk Tree Secara konseptual**

**Definisi 6.1**

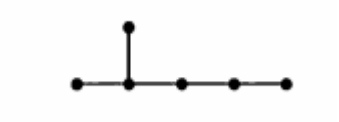
| |
|---|
| Sebuah Tree adalah graph terhubung yang tidak memiliki cycle. |

Diagram berikut menggambarkan tree tidak berlalbel.

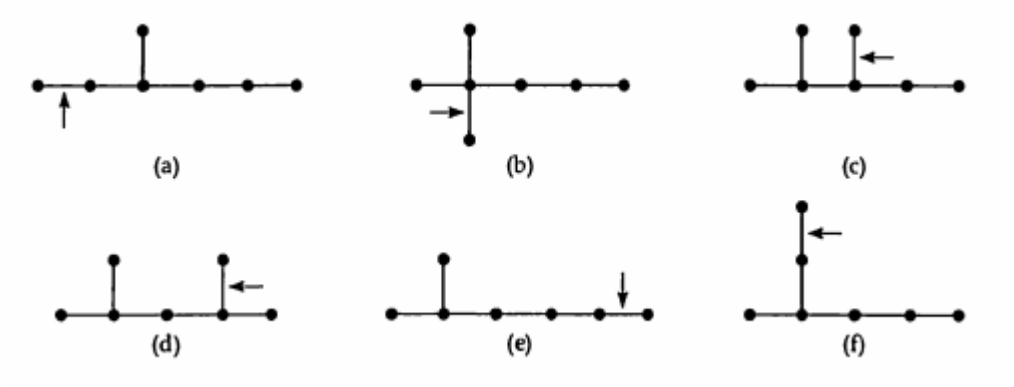



**Gambar 6.3 Contoh Tree Tidak Berlabel**

Suatu tree tidak berlabel yang memiliki *n* titik, dapat diperoleh dari tree tak berlabel yang memiliki *n-1* titik, dengan menambahkan sebuah sisi yang menghubungkan suatu titik baru ke sebuah titik yang telah ada. Ini adalah prosedur umum untuk menambah ukuran sebuah tree tanpa menciptakan sebuah cycle. Misalnya kita memiliki tree dengan 6 titik berikut.



Jika kita menghubungkan sebuah titik baru dengan sebuah titik yang sudah ada pada tree melalui sebuah sisi, kita dapat memperoleh tree dengan 7 buah titik berikut.




**Gambar 6.4 Membangun Tree dengan *n*=7 dari Tree dengan *n*=6**

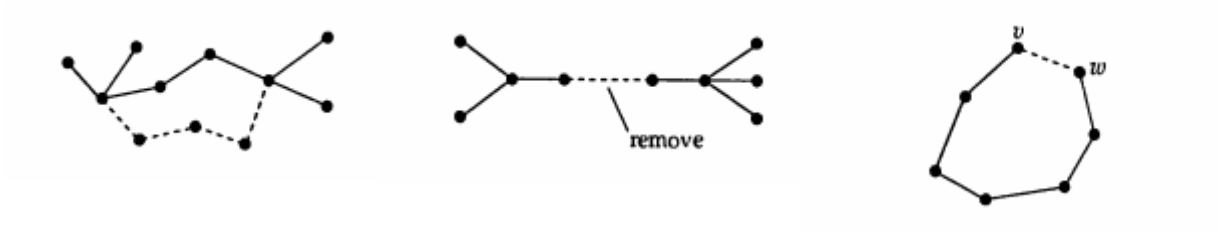
Kesulitan dalam menghasilkan tree dengan cara ini adalah dalam hal mengenali duplikat. Dapat kita perhatikan bahwa tree pada bagian (f) ternyata isomorphic dengan tree pada bagian (a). Sehingga kita memperoleh 5 tree dengan 7 titik.

**Latihan 6.1**

1. Buatlah sebuah tree dengan 5 titik.
2. Dengan menambahkan sebuah sisi pada titik yang sudah ada dalam graph yang anda buat, buatlah 5 buah tree dengan 6 titik. Tentukan graph mana yang isomorphic.

Jika kita memiliki tree dengan sebuah titik, kita dapat membangun banyak tree yang kita inginkan dengan berturut-turut menambah sebuah sisi dan sebuah titik baru. Pada setiap tahap, jumlah titik lebih satu dari jumlah sisinya, sehingga pada setiap tree dengan *n* titik, akan memiliki *n-1* sisi.

Pada setiap tahap, tree tersebut tetap terhubung namun tidak ada cycle yang tercipta. Sehingga setiap dua titik dalam sebuah tree terhubug oleh tepat sebuah path. Oleh karena itu pula, jika kita menghilangkan sebuah sisi dari sebuah tree, maka hal itu menyebabkan tree tersebut menjadi tidak terhubung. Sebaliknya, jika kita menghubungkan dua titik pada tree, maka hal itu akan mengakibatkan terbentuknya cycle.




**Gambar 6.5 Menambah dan Menghapus Sisi Pada Tree**

Dari sifat-sifat tersebut kita memperoleh beberapa definisi yang ekuivalen tentang tree sebagai berikut.

**Teorema 6.1**
**Definisi Ekuivalen Pada Tree**
Misalkan *T* adalah suatu graph dengan *n* titik. Maka pernyataan-pernyataan berikut adalah ekuivalen.

1. *T* terhubung dan tidak memiliki cycle
2. *T* memiliki *n*-1 sisi dan tidak memiliki cycle
3. *T* terhubung dan memiliki *n*-1 sisi
4. *T* terhubung dan penghapusan suatu sisi menyebabkan *T* tidak terhubung
5. Setiap dua titik pada *T* dihubungkan dengan tepat satu path
6. *T* tidak memiliki cycle, namun penambahan sebuah sisi menciptakan sebuah cycle.

**Latihan 6.2**

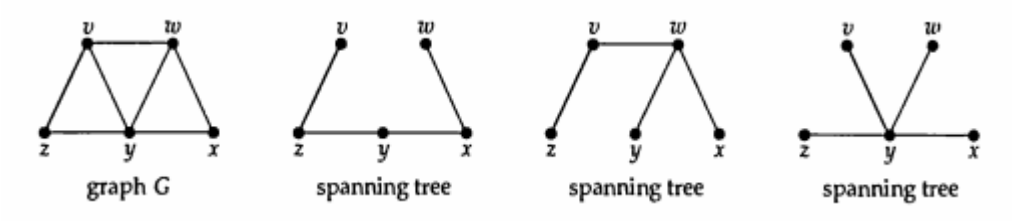
1. Berikan contoh sebuah tree dengan 9 titik dan
   a. Tepat memiliki 2 titik yang berderajat 1
   b. Tepat memiliki 4 titik yang berderajat 1
   c. Tepat memiliki 8 titik berderajat 1
2. Gunakan *handshaking lemma* untuk membuktikan bahwa setiap tree dengan n titik, dimana $n \geq 2$, memiliki setidaknya 2 titik berderajat 1.

## Spanning Tree

### Definisi 6.2

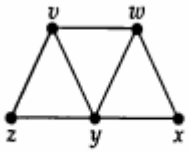
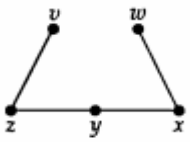
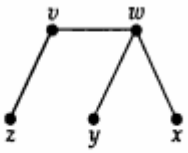
Misalnya G adalah graph terhubung. Maka spanning tree pada G adalah sebuah subgraph dari G yang memuat semua titik dan juga merupakan sebuah tree.

Sebagai contoh, diagram berikut menunjukan sebuah graph dan 3 buah spanning tree-nya.



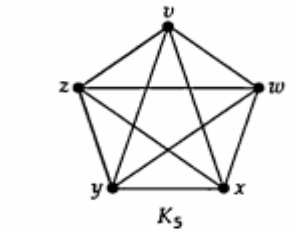

**Gambar 6.6 Membuat Spanning Tree dari Sebuah Graph**

Jumlah spanning tree pada sebuah graph bisa sangat banyak. Sebagai contoh graph Patersen memiliki 2000 spaning tree berlabel. Kita dapat membangun spanning tree dari sebuah graph terhubung dengan dua cara yaitu metode *Building-up* dan metode *Cutting down*. Berikut ilustrasi kedua metode tersebut.

<table>
<tr><th>Metode Buliding-up</th><th>Metode Cutting-down</th></tr>
<tr><td>Cara :<br>a. Pilih satu sisi dari graph sedemikian rupa sehingga tidak ada cycle yang tercipta.<br>b. Ulangi prosedur ini sampai semua titik termuat.</td><td>Cara :<br>a. Pilih suatu cycle dan buang satu sisinya.<br>b. Ulangi prosedur ini sampai tidak ada cycle tersisa.</td></tr>
<tr><td colspan="2">Contoh : Membuat spanning tree dari graph G berikut.<br>  <br>graph G</td></tr>
<tr><td>Pada graph G tersebut, kita memilih sisi vz, wx, xy, yz, sehingga tidak ada cycle yang yang tercipta.<br>Kita memperoleh spanning tree sebagai berikut.<br>  <br>spanning tree</td><td>Dari graph G, kita menghapus sisi :<br>vy ( menghancurkan cycle vwyv)<br>yz (menghancurkan cycle vwyzv)<br>xy (menghancurkan cycle wxyw)<br>Kita menghasilkan spanning tree berikut.<br>  <br>spanning tree</td></tr>
</table>

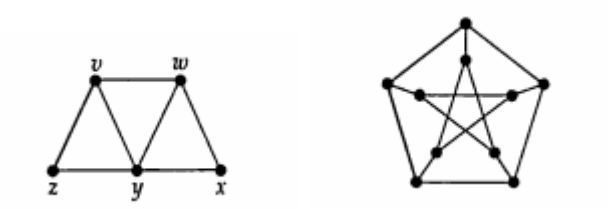
**Latihan 6.3**

1. Pakai setiap metode untuk menghasilkan spanning tree dari graph $K_5$




$K_5$

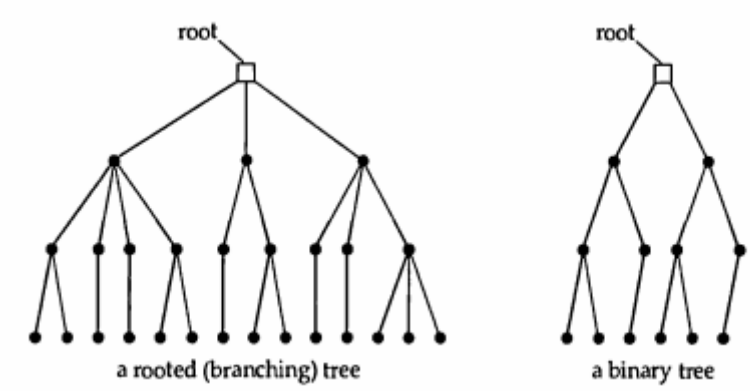
2. Temukan sebanyak mungkin spanning tree dari graph berikut.




## Rooted Trees

Diantara contoh-contoh struktur tree, terdapat suatu type tree yang yang sering muncul yaitu struktur hirarki. Pada struktur ini, satu titik dikhususkan sebagai titik awal, dan cabang-cabangnya keluar dari titik ini. Kita menyebut struktur tree seperti ini sebagai rooted tree (pohon berakar), dengan titik awal sebagai akar (root). Sebagai contoh adalah suatu tree yang merepresentasikan garis pertanggungjawaban pada sebuah perusahaan.

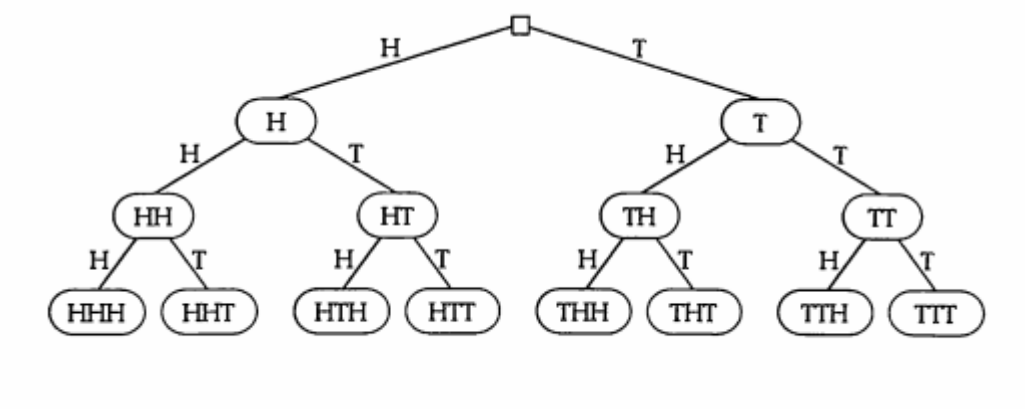
Sebuah rooted tree sering digambarkan dengan akar yang diindikasikan dengan segiempat kecil di atas dan cabang-cabang turun darinya. Walaupun tersirat suatu arah dari atas-bawah, kita biasanya menggambar sebuah rooted tree sebagai graph tak berarah tidak dengan busur dengan arah ke bawah. Sebuah rooted tree di mana ia memiliki maksimal dua cabang menurun pada setiap titiknya disebut binary tree (pohon biner). Tree seperti ini sering disebut branching tree (Pohon bercabang).




**Gambar 6.7 Rooted Tree dan Binary Tree**

## Hasil Percobaan

Jika kita melempar sebuah koin beberapa kali, maka hasil yang mungkin muncul dapat direpresentasikan ke dalam branching tree. Pada kasus pelemparan sebuah koin, setiap hasil memiliki kemungkinan dua sisi yang muncul yaitu head (H) atau tail (T). Oleh karena itu kita memperoleh pohon biner. Sebagai contoh, jika kita melempar sebuah koin tiga kali, maka ada delapan hasil yang mungkin diperoleh, dan kita mendapatkan branching tree berikut.

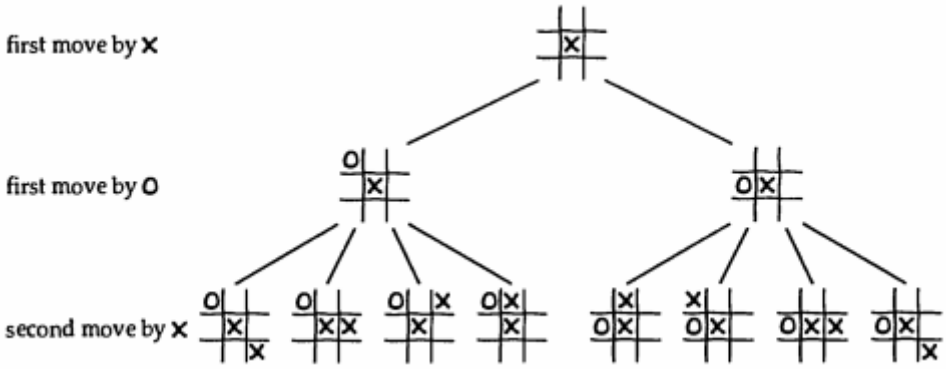



**Gambar 6.8 Branching Tree dari Hasil Percobaan**

**Latihan 6.4**
Gambarkan sebuah branching tree yang merepresentasikan hasil dari pelemparan dua buah dadu (6 sisi).

## Permainan Stategi

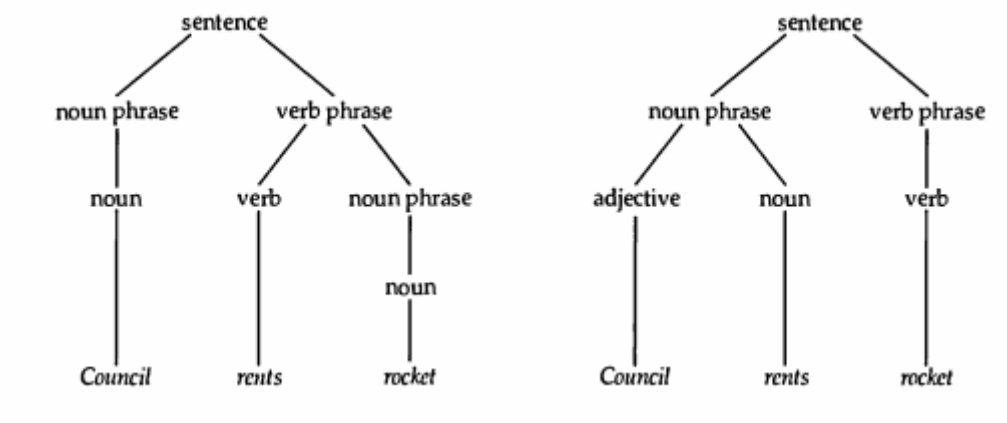
Branching tree muncul dalam analisis permainan, terutama permainan stategi seperti catur dan tic-tac-toe. Pada representasi ini ditunjukan barisan langkah dari setiap posisi ke posisi berikutnya. Diagram berikut menggambarkan brancing tree yang merepresentasikan tiga langkah pertama dalam permainan tic-tac-toe.




**Gambar 6.8 Branching Tree dari Permainan Tic-Tac-Toe**

## Grammatical Trees

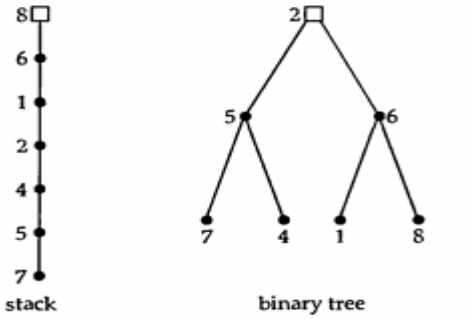
Branching tree juga dipakai dalam parsing pada kalimat dalam pemrosesan bahasa alami. Tree merepresentasikan keterkaitan di antara kata-kata dan prase pada suatu kalimat. Suatu branching tree diperoleh dengan memisahkan kalimat ke dalam frase, kemudian memisahkan frase ini kedalam kata benda, kata kerja, kata sifat dan seterusnya. Jika kalimat tersebut ambigu , kita dapat memakai branching tree untuk membedakan konstruksi kalimat yang berbeda. Sebagai contoh kalimat "*Council rents rocket*" dapat diinterpretasikan dalam dua tree berikut.




**Gambar 6.9 Branching Tree pada Parsing Kalimat**

## Ilmu Komputer

Struktur rooted tree muncul dalam ilmu komputer, di mana rooted tree digunakan untuk memodelkan dan menjelaskan prosedur percabangan dalam Bahasa pemrograman. Rooted tree dipakai khususnya untuk menyimpan data pada memori komputer dalam berbagai cara yang berbeda. Misalnya, terdapat daftar angka berikut : 7,5,4,2,1,6,8. Tree pada Gambar 6.10 menunjukan cara penyimpanan angka tersebut dalam memori (sebagai sebuah stack dan sebagai binary tree). Setiap representasi memiliki keuntungan, bergantung pada bagaimana data dimanipulasi, namun pada kedua representasi tersebut, sangat penting untuk membedakan dimana data dimulai, sehingga tree yang dibuat merupakan sebuah rooted tree.

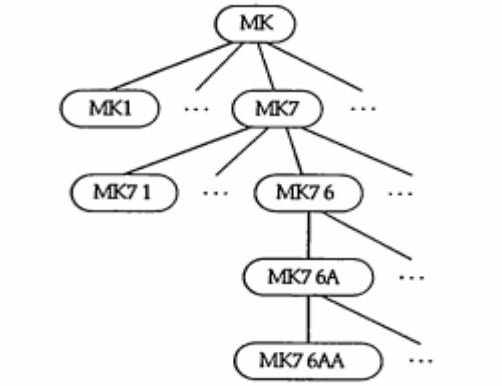



**Gambar 6.10 Rooted Tree untuk Penyimpanan Data pada Memori Komputer**

Kita memperoleh tree dengan menulis angka sebagai string 7542168. Kemudian kita mempromosikan setiap angka kedua (5,2,6) dan selanjutnya mempromosikan angka kedua yang baru (2).

## Sorting Trees

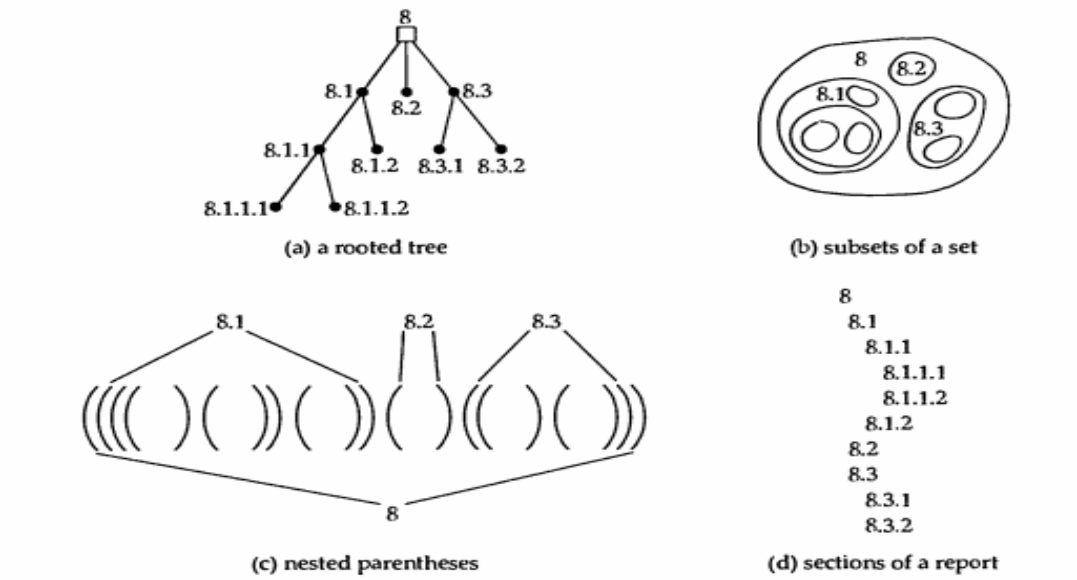
Sorting tree (pohon terurut) adalah branching tree yang muncul ketika kita menginginkan sebuah pilihan yang terurut, setiap pilihan bergantung pada pilihan sebelumnya. Sebagai contoh, sorting tree dapat dipakai untuk merepresentasikan pengurutan surat berdasarkan pada kode post.




**Gambar 6.11 Sorting Tree untuk Pengurutan Surat**

### Bentuk Ekuivalen

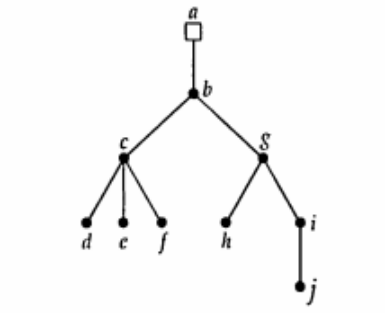
Rooted tree dapat direpresentasikan ke dalam cara-cara yang berbeda. Diagram berikut menunjukan empat cara yang ekuivalen untuk merepresentasikan rooted tree yang sama.




**Gambar 6.12 Bentuk Ekuivalen Rooted Tree**

1. Rooted Tree pada diagram (a) memiliki penampakan konvensional dari rooted tree.
2. Subsets of a set pada diagram (b) merepresentasikan tree sebagai sebuah sistem sub himpunan dari himpunan.
3. Nested parantheses pada diagram (c) merepresentasikan tree dengan tanda kurung bersarang. Bentuk ini biasa dipakai dalam persamaan matematika.
4. Section of a report pada diagram (d), setiap level pada setiap bagian diindikasikan dengan lekukan dan panjang angka desimal.

**Latihan 6.5**
Gambarkan tree berikut dalam bentuk subset of a set, nested parantheses dan section of a report.



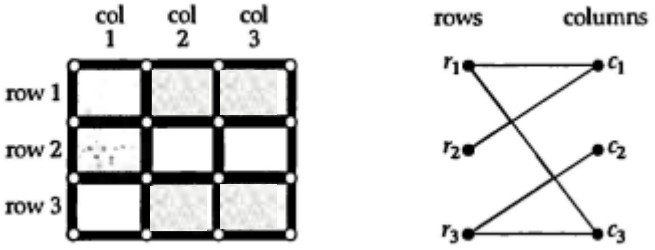

Keuntungan dari stuktur tree adalah kemudahannya untuk diubah atau diperbaharui. Ini penting untuk aplikasi komputer, di mana kita dapat melakukan *insert* dan *delete* suatu cabang tanpa mengubah keseluruhan sistem. Pada sisi lain, kekurangan besar pada struktur tree adalah kerentanan pada kesalahan. Penghapusan sebuah titik atau sisi cukup untuk menjadikannya tak terhubung dan dapat menghancurkan sistem.

## Studi Kasus

**Braced Rectangular Framework**

Banyak bangunan yang memakai kerangka baja berupa balok-balok segiempat yang sendinya terpaku. Agar kerangka tetap kokoh/kaku diperlukan suatu penyokong/penahan yang dipasang pada kerangka tersebut. Adapun kriteria yang dipakai agar penambahan penyokong membuat kerangka tetap rigid (kaku) adalah sebagai berikut.

Kasus ini dapat direpresentasikan sebagai suatu graph bipartit di mana titik pada baris dihubungkan dengan titik pada kolom jika ia diperkuat suatu penyokong. Contoh representasi kerangka kedalam graph bipartit ditunjukan pada Gambar 6.13. Setelah suatu kerangka direpresentasikan ke dalam graph bipartit, maka kita akan dapat melihat sifat graph yang rigid atau tidak dari keterhubungan graph tersebut. Suatu kerangka segiempat yang diperkuat adalah rigid jika hanya jika graph bipartite-nya merupakan graph yang terhubung. Graph yang terhubung berarti untuk setiap dua titik pada graph tersebut selalu memiliki path.




**Gambar 6.13 Representasi Braced Rectangular Framework ke dalam Graph Bipartit**

## Minimum Bracings

Graph bipartit untuk suatu kerangka yang kuat pasti terhubung. Jika pada graph tersebut terdapat cycle, lalu kita menghapus suatu sisi pada cycle tersebut, maka graph tersebut masih terhubung. Jika kita mengulangi prosedur tersebut sehingga menghancurkan semua cycle pada graph, maka kita akan mendapatkan suatu spanning tree. Dengan kata lain suatu graph yang merupakan spanning tree adalah suatu graph untuk kerangka dengan penyokong yang minimal. Graph untuk kerangka dengan penyokong minimal ini tercipta jika graph dengan *n* titik memiliki tepat *n-1* sisi atau tidak memiliki cycle.

**Latihan 6.6**

1. Dengan membuat graph bipartit pada setiap kerangka dibawah, tentukan apakah kerangka tersebut rigid atau tidak.
2. Jika terdapat kerangka yang rigid tentukan apakah kerangka tersebut memiliki penyokong yang minimal? Jelaskan!