POHON BINER**Abi Wibowo****185100008**

Fakultas Komputer

abiwibowo.student@umitra.ac.id**Abstract**

Salah satu penerapan teori pohon yang paling berguna dan dipakai yaitu konsep binary search tree dimana konsep ini memberikan struktur data yang memudahkan operasi pencarian, penambahan, dan penghapusan terhadap data. Operasi tersebut lebih efisien dan jauh lebih baik pada konsep ini dibanding sequential search pada senarai berkait dalam waktu eksekusi / run-time. Dari konsep binary search tree ini dikembangkan lagi suatu struktur penyimpanan data yang merupakan modifikasi dari binary search tree tersebut yaitu AVL-Tree dan Splay Tree yang masing-masing mempunyai keunggulan pada kasus tertentu yang sekarang ini sering dijumpai. AVL-Tree merupakan modifikasi binary search tree yang tinggi setiap upapohon kiri dan upapohon kanan sama atau setidaknya selisih antara keduanya tidak lebih dari 1. Keunggulan dari AVL-Tree antara lain untuk mengoptimasi pencarian data terutama untuk kasus pohon yang condong ke kiri atau ke kanan sehingga pencarian akan jauh lebih mudah apabila pohon tersebut seimbang. Kasus pohon yang condong ke kiri atau kanan itu mungkin saja terjadi terutama apabila penambahan elemen dan penghapusan elemen dilakukan terus-menerus dan tidak dapat diketahui urutannya. Sedangkan Splay-Tree justru kebalikan dari AVL-Tree yang tidak mempermasalahkan kecondongan upapohonnya namun setiap kali data diakses maka simpul dari data yang diakses tersebut akan dinaikkan keatas mendekati akar pohon. Data yang sering diakses / aktif akan berada dekat pada akar pohon sehingga data tersebut mudah. Dengan demikian dapat disimpulkan bahwa penerapan teori pohon sangatlah bermanfaat dalam kajian struktur data.

Kata Kunci : Pohon Biner

A. PENDAHULUAN

Teori pohon merupakan salah satu teori yang cukup tua karena sudah dikenal sejak tahun 1857, dimana ketika itu matematikawan Inggris Arthur Cayley menggunakan teori pohon ini untuk menghitung jumlah senyawa kimia. Teori pohon ini sebenarnya adalah suatu mekanisme penyelesaian suatu masalah dengan menganalogikan permasalahan tersebut kedalam struktur pohon untuk memudahkan pencarian solusi masalah tersebut. Teori pohon ini juga merupakan salah satu penerapan konsep graf. Dimana pohon itu dapat didefinisikan sebagai graf tak-berarah terhubung yang tidak mengandung sirkuit. Kajian struktur data merupakan kajian yang sangat penting dalam bidang informatika. Dan di zaman sekarang ini yang teknologinya semakin berkembang, dibutuhkan struktur data yang efisien yang dapat meningkatkan kinerja program. Teori pohon ini merupakan teori yang sangat berguna dalam struktur data dimana aplikasiaplikasi dari teori pohon ini dapat dijadikan struktur penyimpanan data yang sangat baik dalam kasus tertentu yang mana kasus tersebut sudah umum ditemui sekarang ini. Oleh karena itu dalam makalah ini akan dijelaskan beberapa aplikasi teori pohon yang dipakai untuk

membentuk suatu struktur penyimpanan data yang efisien.

B. PEMBAHASAN / STUDI KASUS

- C. Dalam struktur data, pohon memegang peranan yang cukup penting. Struktur ini biasanya digunakan terutama untuk menyajikan data yang mengandung hubungan hirarkikal antara elemen-elemen mereka. Kita lihat misalnya, data pada *record*, keluarga dari pohon, ataupun isi dari tabel. Mereka mempunyai hubungan hirarkikal.
- D. Bentuk pohon berakar yang khusus, yang lebih mudah kita kelola dalam komputer adalah pohon binar (*binary tree*). Bentuk pohon berakar yang umum, kita kenal sebagai “pohon umum” atau “*general tree*.” Sebuah pohon binar T didefinisikan terdiri atas sebuah himpunan hingga elemen yang disebut simpul (*node*), sedemikian sehingga :
- E. (a) T adalah hampa (disebut pohon *null*) atau;
- F. (b) T mengandung simpul R yang dipilih (dibedakan dari yang lain), disebut “akar” atau “*root*” dari T , dan simpul sisanya membentuk 2 pohon binar (subpohon kiri dan subpohon kanan dari akar R) T_1 dan T_2 yang saling lepas.
- G. Perhatikan bahwa pendefinisian pohon binar di atas adalah rekursif. Jika T_1 tidak hampa, maka simpul akarnya disebut suksesor kiri

dari R. Hal serupa untuk akar dari T_2 (tidak hampa) disebut suksesor kanan dari R. Pohon binar acapkali disajikan dalam bentuk diagram. Perhatikan contoh Pohon binar pada Gambar 7.2 berikut.

H. Pohon binar tersebut mempunyai 11 simpul yang diberi label huruf A sampai L (tak termasuk I). Simpul akar adalah simpul yang digambar pada bagian paling atas. Untuk menggambarkan suksesor kiri serta suksesor kanan, dibuat garis ke kiri bawah dan ke kanan bawah. Perhatikan pada Gambar tersebut bahwa B adalah suksesor kiri dari A, sedangkan C adalah suksesor kanan dari A. Subpohon kiri dari A mengandung simpul-simpul B, D, E dan F, sedangkan subpohon kanannya mengandung simpul-simpul C, G, H, J, K dan L. Kita dapat melihat bahwa jika N adalah sebarang simpul dari pohon binar T, maka N mempunyai 0, 1 atau 2 buah suksesor. Suksesor kerap kali disebut anak atau anak lelaki (*child* atau *son*). Jadi simpul N tersebut boleh kita sebut ayah atau orang-tua (*father* atau *parent*) dari suksesornya. Pada contoh kita, Simpul A, B, C dan H mempunyai 2 anak, simpul E dan J mempunyai satu anak. Sementara itu simpul-simpul D, F, G, L dan K tidak mempunyai satu anakpun. Simpul yang tidak

mempunyai anak disebut “daun” atau “terminal.”

Gambar 7.2. Contoh pohon binar dengan 11 simpul

Dalam bab ini kita akan mempelajari satu bentuk struktur data tak linier yang mempunyai sifat-sifat khusus, yang dinamakan pohon (tree). Struktur ini biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada. Silsilah keluarga, hasil pertandingan yang berbentuk turnamen, atau struktur organisasi dari sebuah perusahaan adalah contoh dalam organisasi tree. Dalam pemrograman, sebuah pohon terdiri dari elemen-elemen yang dinamakan node(simpul) yang mana hubungan antar simpul bersifat hirarki. Simpul yang paling atas dari hirarki dinamakan root. Simpul yang berada di bawah root secara langsung, dinamakan anak dari root, yang mana biasanya juga mempunyai anak di bawahnya. Sehingga biasa

disimpulkan, kecuali root, masing-masing simpul dalam hirarki mempunyai satu induk (parent).

Jumlah anak sebuah simpul induk sangat bergantung pada jenis dari pohon. Jumlah anak dari simpul induk ini dinamakan faktor percabangan. Pada bab ini pembahasan difokuskan pada binary tree, yaitu pohon yang mempunyai factor percabangan 2.

1. Deskripsi dari Binary Tree

Sebuah binary tree adalah sebuah pengorganisasian secara hirarki dari beberapa buah simpul, dimana masing-masing simpul tidak mempunyai anak lebih dari 2. Simpul yang berada di bawah sebuah simpul dinamakan anak dari simpul tersebut. Simpul yang berada di atas sebuah simpul dinamakan induk dari simpul tersebut.

Masing-masing simpul dalam binary tree terdiri dari tiga bagian : sebuah data dan dua buah pointer yang dinamakan pointer kiri dan kanan. Dengan menggunakan tiga bagian ini, kita menampilkan sebuah binary tree dengan melakukan setting pada pointer kiri dan kanan untuk menghubungkan sebuah simpul dengan anak-anaknya. Jika sebuah simpul tidak mempunyai

anak pada pointer kiri atau kanan, kita melakukan setting pada pointer tersebut pada NULL, yang menunjukkan akhir dari percabangan adalah pada simpul tersebut.

Sebuah percabangan adalah kumpulan dari simpul-simpul yang dimulai dari sebuah root dan diakhiri dengan sebuah simpul terakhir. Simpul terakhir adalah simpul dari tree yang tidak mempunyai anak. Kadang-kadang ketika kita bekerja dengan beberapa pohon dalam satu waktu, maka pohon-pohon tersebut dinamakan sebuah hutan.

2. Istilah-Istilah Dasar

Simpul juga mempunyai sibling, descendants, dan ancestors. Sibling dari sebuah simpul adalah anak lain dari induk simpul tersebut. Descendants dari sebuah simpul adalah semua simpul-simpul merupakan cabang (berada di bawah) simpul tersebut.

Ancestors dari sebuah simpul adalah semua simpul yang berada di atas antara simpul tersebut dengan root. Penampilan dari sebuah tree akan ditampilkan dengan berat dari tree

tersebut, angka yang menunjukkan jumlah level yang ada di dalamnya.

Tingkat suatu simpul ditentukan dengan pertama kali menentukan akar sebagai bertingkat 1. Jika suatu simpul dinyatakan sebagai tingkat N, maka simpul-simpul yang merupakan anaknya akan berada pada tingkatan N + 1.

Tinggi atau kedalaman dari suatu pohon adalah tingkat maksimum dari simpul dalam pohon dikurangi dengan 1. Selain tingkat, juga dikenal istilah derajat (degree) dari suatu simpul. Derajat suatu simpul dinyatakan sebagai banyaknya anak atau turunan dari simpul tersebut.

Gambar: Ilustrasi Sebuah Pohon Biner dengan Kedalaman 3

3. Penyajian Pohon Biner

Implementasi pohon biner dalam pemrograman dapat dilakukan dengan beberapa cara. Cara pertama adalah dengan menggunakan link list dan cara lain adalah dengan menggunakan cara berurutan. Dalam bab ini kita lebih banyak mempelajari cara implementasi biner menggunakan link list.

4. Deklarasi Pohon

Jika kita memperhatikan setiap simpul dalam pohon biner, kita bisa menyusun struktur data yang tepat dari simpul-simpul tersebut. Kita dapat melihat bahwa dalam setiap simpul selalu berisi dua buah pointer untuk menunjuk ke cabang kiri dan cabang kanan, dan informasi yang akan disimpan dalam simpul tersebut. Dengan memperhatikan hal ini, simpul dalam pohon biner disajikan sebagai berikut:

Gambar Simpul Pada Pohon Biner

Sesuai dengan gambar tersebut, maka deklarasi list yang sesuai adalah:

```
typedef char TypeInfo;
typedef struct Simpul *Tree;
struct Simpul {
    TypeInfo Info;
    tree Kiri,      /* cabang kiri */
    Kanan; /* cabang kanan */
};
```

Program Deklarasi dari Tree

3.2. Implementasi Binary Tree dengan Linked List

Salah satu implementasi dari binary tree adalah Binary Search Tree (BST), dimana BST merupakan sekumpulan elemen yang bernilai unik, dan untuk setiap node X dalam struktur BST

Nilai elemen pada sub pohon kiri (left subtree) pasti memiliki nilai yang lebih kecil dari X dan nilai elemen pada sub pohon kanan (right sub tree) pasti memiliki nilai yang lebih besar.

BST dapat disajikan dengan beberapa cara. Dalam praktikum ini akan menggunakan linked list untuk implementasi Binary Search Tree. Linked list yang dipakai adalah double linked list non circular.

Gambar Penyajian Binary Tree dengan Double Linked List

Berikut penjelasan detail mengenai implementasi Binary Search Tree dengan linked list:

Terdapat setidaknya 8 operasi yakni, insert, findMin, findMax, find, remove, preOrder, inOrder, postOrder.

Proses insert dilakukan dengan mengikuti aturan sbb:

Setiap node baru akan menempati posisi sebagai daun (leaf).

Untuk insert kedalam BST dimulai dari root, jika data lebih kecil dari root, node baru harus di masukkan ke dalam left sub tree. Jika data pada node baru

lebih besar dari root, node baru harus di masukkan ke dalam right sub tree.

Gunakan rekursif.

Operasi findMin akan mencari nilai terkecil dari node yang ada di dalam BST dengan cara menelusuri left subtree sampai ke daun. (gunakan rekursif).

Operasi findMax akan mencari nilai terbesar dari node yang ada di dalam BST dengan cara menelusuri right subtree sampai ke daun. (gunakan rekursif).

Operasi find akan mencari nilai yang dicari dengan membandingkan dengan data pada root. Jika lebih kecil akan mencari di left sub tree, dan jika lebih besar cari di right sub tree. (gunakan rekursif).

Operasi remove dilakukan dengan mengikuti aturan sbb:

Jika node yang dihapus berposisi sebagai daun, dengan sederhana bisa dihapus.

Jika node memiliki satu anak, maka anak tersebut akan menggantikan posisi node yang dihapus.

Jika node memiliki 2 anak [pilih salah satu]:

§ Ganti node yang dihapus dengan node terkecil pada right sub tree.

§ Ganti node yang dihapus dengan node terbesar pada left sub tree.

Setiap elemen/node dari BST mempunyai 3 bagian yaitu bagian data yang bernilai dengan data, bagian left untuk menunjuk ke left subtree dan bagian right untuk menunjuk ke right sub tree. Dalam hal, ini setiap data dalam node bersifat unik/beda.

3.3. Inisialisasi Tree

Untuk pertama kali, saat kita akan membuat sebuah pohon biner, asumsi awal adalah pohon itu belum bertumbuh, belum memiliki node sama sekali, sehingga masih kosong. Oleh karena itu perlu kita tambahkan kode berikut pada baris awal fungsi Main:

Kita mendeklarasikan sebuah pointer yang akan menunjuk ke akar pohon yang kita buat, dengan nama *pohon. Pointer ini ditujukan untuk menunjuk struktur bertipe Node, yang telah dibuat pada bagian 1. Karena pohon tersebut sama sekali belum memiliki node, maka pointer *pohon ditunjukkan ke NULL.

3.4. Menambahkan Node Pada Tree

Karena pohon yang kita buat merupakan sebuah pohon biner, maka untuk menambahkan sebuah node, secara otomatis penambahan tersebut mengikuti aturan penambahan node pada pohon biner:

1. Jika pohon kosong, maka node baru ditempatkan sebagai akar pohon.
2. Jika pohon tidak kosong, maka dimulai dari node akar, dilakukan proses pengecekan berikut:

- a. Jika nilai node baru lebih kecil dari nilai node yang sedang dicek, maka lihat ke kiri node tersebut. Jika kiri node tersebut kosong (belum memiliki kiri), maka node baru menjadi kiri node yang sedang dicek. Seandainya kiri node sudah terisi, lakukan kembali pengecekan a dan b terhadap node kiri tersebut. Pengecekan ini dilakukan seterusnya hingga node baru dapat ditempatkan.

- b. Jika nilai node baru lebih besar dari nilai node yang sedang dicek, maka lihat ke kanan node tersebut. Jika kanan node tersebut kosong (belum memiliki kanan), maka node baru menjadi kanan node yang sedang dicek. Seandainya kanan node sudah terisi, lakukan kembali pengecekan a

dan b terhadap node kanan tersebut. Pengecekan ini dilakukan seterusnya hingga node baru dapat ditempatkan.

c.

3.5. Membaca dan Menampilkan Node Pada Tree

Untuk membaca dan menampilkan seluruh node yang terdapat pada pohon biner, terdapat 3 macam cara, atau yang biasa disebut kunjungan (visit). Semua kunjungan diawali dengan mengunjungi akar pohon. Karena proses kunjungan ini memerlukan perulangan proses yang sama namun untuk depth (kedalaman) yang berbeda, maka ketiganya diimplementasikan dengan fungsi rekursif.

a. Kunjungan Pre-Order.

Kunjungan pre-order dilakukan mulai dari akar pohon, dengan urutan:

1. Cetak isi (data) node yang sedang dikunjungi
2. Kunjungi kiri node tersebut,
 - Jika kiri bukan kosong (tidak NULL) mulai lagi dari langkah pertama, terapkan untuk kiri tersebut.
 - Jika kiri kosong (NULL), lanjut ke langkah ketiga.
3. Kunjungi kanan node tersebut,

- Jika kanan bukan kosong (tidak NULL) mulai lagi dari langkah pertama, terapkan untuk kanan tersebut.

- Jika kanan kosong (NULL), proses untuk node ini selesai, tuntaskan proses yang sama untuk node yang dikunjungi sebelumnya.

b. Kunjungan In-Order.

1. Kunjungi kiri node tersebut,

- Jika kiri bukan kosong (tidak NULL) mulai lagi dari langkah pertama, terapkan untuk kiri tersebut.

- Jika kiri kosong (NULL), lanjut ke langkah kedua.

2. Cetak isi (data) node yang sedang dikunjungi

3. Kunjungi kanan node tersebut,

- Jika kanan bukan kosong (tidak NULL) mulai lagi dari langkah pertama, terapkan untuk kanan tersebut.

- Jika kanan kosong (NULL), proses untuk node ini selesai, tuntaskan proses yang sama untuk node yang dikunjungi sebelumnya.

c. Kunjungan Post-Order.

1. Kunjungi kiri node tersebut,

- Jika kiri bukan kosong (tidak NULL) mulai lagi dari langkah pertama, terapkan untuk kiri tersebut.
 - Jika kiri kosong (NULL), lanjut ke langkah kedua.
2. Kunjungi kanan node tersebut,
- Jika kanan bukan kosong (tidak NULL) mulai lagi dari langkah pertama, terapkan untuk kanan tersebut.
 - Jika kanan kosong (NULL), lanjut ke langkah ketiga.

3.6. Membuat Pohon Biner

Tabel Program Deklarasi dan Tree memperlihatkan bagaimana proses pembentukan pohon biner dari sebuah persamaan.

Tabel Pembentukan Pohon Biner dari persamaan: $(A+B)*((B-C)+D)$

Jenis-Jenis Binary Tree

1. Complete Binary Tree

Suatu binary tree T akan disebut complete/lengkap jika semua levelnya memiliki child 2 buah kecuali untuk level paling akhir. Tetapi pada akhir level setiap leaf/daun muncul terurut dari sebelah kiri.

Gambar. Complete tree T_{12}

I. ID SECURITY

QWTD4452377-ASP-5244107

J. KESIMPULAN

K. Tree adalah sebuah struktur linier, biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada. Ada beberapa istilah dalam tree ini, yang mana masing-masing istilah mempunyai arti dalam kaitannya dengan hirarki antar elemen dalam tree tersebut, seperti sibling, descendant dsb. Dalam ilmu komputer, tree adalah sebuah struktur data yang secara bentuk menyerupai sebuah pohon, yang terdiri dari serangkaian node (simpul) yang saling berhubungan. Node-node tersebut dihubungkan oleh sebuah vektor. Setiap node dapat memiliki 0 atau lebih node anak (child). Sebuah node yang memiliki node anak disebut node induk (parent). Sebuah node anak hanya memiliki satu node induk. Sesuai konvensi ilmu komputer, tree bertumbuh ke bawah, tidak seperti pohon

di dunia nyata yang tumbuh ke atas. Dengan demikian node anak akan digambarkan berada di bawah node induknya.

L. DISKUSI

Saya bersama teman saya bernama Aria Sakti Nugraha mendiskusikan hasil materi ini
Saya : “Menurut anda bagaimana artikel ini?

Aria : “ Menurut saya artikel ini sangat bagus ,dan membantu mahasiswa untuk memahami tentang Pohon Biner .

M. REFERENCE

- [1] O. M. Febriani and A. S. Putra, “Sistem Informasi Monitoring Inventori Barang Pada Balai Riset Standardisasi Industri Bandar Lampung,” *J. Inform.*, vol. 13, no. 1, pp. 90–98, 2014.
- [2] A. S. Putra, “Paperplain: Execution Fundamental Create Application With Borland Delphi 7.0 University Of Mitra Indonesia,” 2018.
- [3] A. S. Putra, “2018 Artikel Struktur Data, Audit Dan Jaringan Komputer,” 2018.
- [4] A. S. Putra, “ALIAS MANAGER USED IN DATABASE DESKTOP STUDI CASE DB DEMOS.”
- [5] A. S. Putra, “COMPREHENSIVE SET OF PROFESSIONAL FOR DISTRIBUTE COMPUTING.”
- [6] A. S. Putra, “DATA ORIENTED RECOGNITION IN BORLAND DELPHI 7.0.”
- [7] A. S. Putra, “EMBARCADERO DELPHI XE 2 IN GPU-POWERED FIREMONKEY APPLICATION.”
- [8] A. S. Putra, “HAK ATAS KEKAYAAN INTELEKTUAL DALAM DUNIA TEKNOLOGY BERBASIS REVOLUSI INDUSTRI 4.0.”
- [9] A. S. Putra, “IMPLEMENTASI PERATURAN PERUNDANGAN UU. NO 31 TAHUN 2000 TENTANG DESAIN INDUSTRI BERBASIS INFORMATION TECHNOLOGY.”
- [10] A. S. Putra, “IMPLEMENTATION OF PARADOX DBASE.”
- [11] A. S. Putra, “IMPLEMENTATION OF TRADE SECRET CASE STUDY SAMSUNG MOBILE PHONE.”
- [12] A. S. Putra, “IMPLEMENTATION PATENT FOR APPLICATION WEB BASED CASE STUDI WWW. PUBLIKLAMPUNG.COM.”
- [13] A. S. Putra, “IMPLEMENTATION SYSTEM FIRST TO INVENT IN DIGITALLY INDUSTRY.”
- [14] A. S. Putra, “MANUAL REPORT & INTEGRATED DEVELOPMENT ENVIRONMENT BORLAND DELPHI 7.0.”
- [15] A. S. Putra, “PATENT AS RELEVAN SUPPORT RESEARCH.”
- [16] A. S. Putra, “PATENT FOR RESEARCH STUDY CASE OF APPLE. Inc.”

- [17] A. S. Putra, "PATENT PROTECTION FOR APPLICATION INVENT."
- [18] A. S. Putra, "QUICK REPORT IN PROPERTY PROGRAMMING."
- [19] A. S. Putra, "REVIEW CIRCUIT LAYOUT COMPONENT REQUIREMENT ON ASUS NOTEBOOK."
- [20] A. S. Putra, "REVIEW TRADEMARK PATENT FOR INDUSTRIAL TECHNOLOGY BASED 4.0."
- [21] A. S. Putra, "TOOLBAR COMPONENT PALLETTE IN OBJECT ORIENTED PROGRAMMING."
- [22] A. S. Putra, "WORKING DIRECTORY SET FOR PARADOX 7."
- [23] A. S. Putra, "ZQUERY CONNECTION IMPLEMENTED PROGRAMMING STUDI CASE PT. BANK BCA Tbk."
- [24] A. S. Putra, D. R. Aryanti, and I. Hartati, "Metode SAW (Simple Additive Weighting) sebagai Sistem Pendukung Keputusan Guru Berprestasi (Studi Kasus: SMK Global Surya)," in *Prosiding Seminar Nasional Darmajaya*, 2018, vol. 1, no. 1, pp. 85–97.
- [25] A. S. Putra and O. M. Febriani, "Knowledge Management Online Application in PDAM Lampung Province," in *Prosiding International conference on Information Technology and Business (ICITB)*, 2018, pp. 181–187.
- [26] A. S. Putra, O. M. Febriani, and B. Bachry, "Implementasi Genetic Fuzzy System Untuk Mengidentifikasi Hasil Curian Kendaraan Bermotor Di Polda Lampung," *SIMADA (Jurnal Sist. Inf. dan Manaj. Basis Data)*, vol. 1, no. 1, pp. 21–30, 2018.
- [27] A. S. Putra, H. Sukri, and K. Zuhri, "Sistem Monitoring Realtime Jaringan Irigasi Desa (JIDES) Dengan Konsep Jaringan Sensor Nirkabel," *IJEIS (Indonesian J. Electron. Instrum. Syst.)*, vol. 8, no. 2, pp. 221–232.
- [28] D. P. Sari, O. M. Febriani, and A. S. Putra, "Perancangan Sistem Informasi SDM Berprestasi pada SD Global Surya," in *Prosiding Seminar Nasional Darmajaya*, 2018, vol. 1, no. 1, pp. 289–294.