Penerapan Pohon Pencarian Biner Seimbang dalam Aplikasi NIM Finder Sederhana

Christopher Billy Setiawan / 13517050
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517050@std.stei.itb.ac.id

***Abstract*—Pohon Pencarian Biner merupakan pohon Biner yang setiap simpul Sub Pohon dikirinya memiliki nilai lebih kecil daripada simpul tersebut. Dan setiap Simpul Sub Pohon di kanannya memiliki nilai lebih besar atau sama daripada simpul tersebut. Tapi dalam kasus ini, pohon pencarian yang dibuat nilai datanya unik hanya satu setiap data.**

***Keywords*—Pohon Pencarian Biner, Simpul, Tree, Binary Search**

I. PENDAHULUAN

Pada Jaman ini, pemrograman merupakan hal yang sudah biasa. Apalagi untuk setiap mahasiswa Teknik Informatika, setiap mahasiswa Teknik Informatika harus sudah terlatih dalam pengembangan *software* di Dunia yang sudah memasukki jaman revolusi industri ke 4 dimana teknologi sudah jauh berkembang daripada tahun-tahun silam. Salah satu Algoritma yang paling sering digunakan dalam pemrograman adalah algoritma pencarian.

Dalam dunia pemrograman ada beberapa jenis algoritma pencarian seperti, *Sequential Search*, *Binary Search*, dan *Interpolation Search.*

Sequential Search atau Pencarian Sekuensial merupakan metode pencarian yang paling mudah. Metode ini adalah suatu Teknik pencarian data dalam array 1 dimensi yang akan menelusuri semua elemen-elemen *array* satu demi satu dari awal sampai akhir. Dimana data tidak perlu diurutkan terlebih dahulu. Proses pencarian elemen ini tidak efektif jika elemen dari *array* tersebut terlampau banyak dan elemen berada di indeks terakhir. Oleh karena itu ditemukanlah algoritma pencarian yang baru.

Binary Search atau Pencarian Biner merupakan metode pencarian yang hanya bisa dilakukan pada kumpulan data yang sudah diurutkan terlebih dahulu. Metode ini digunakan untuk kebutuhan pencarian dengan waktu yang cepat dan jumlah data yang banyak. Jika terdapat N buah data yang akan diolah, program akan mengecek elemen ke N/2. Jika elemen tidak sesuai dan lebih besar maka dia akan mengecek setengah elemen di kanannya lalu akan mencari elemen tengah dari bagian kanan array lalu program akan melakukan pengecekan lagi. Proses dilakukan sampai elemen yang dicari ditemukan atau tidak ditemukan.

Interpolation Search atau Pencarian Interpolasi merupakan metode yang hamper mirip dengan metode Pencarian Biner dimana pencarian dilakukan pada kumpulan data yang sudah terurut. Akan tetapi jika pada Pencarian Biner kita membagi data menjadi 2 bagian tiap prosesnya, pada Pencarian Interpolasi kita akan membagi data menurut rumus sebagai berikut :

Posisi = (kunci–data[low] / data[high]–data[low])*(high-low) + low

Singkatnya proses Algoritma Pencarian Interpolasi hampir mirip dengan proses pencarian kata dikamus, yaitu kita mencari data yang dimaksud dengan cara memperkirakan letak data.

Dalam Makalah ini, Akan dipaparkan mengenai Pencarian Biner di Pohon *(Binary Search Tree)* dalam Aplikasi NIM Finder. Dengan menggunakan Pencarian Biner, Proses pencarian akan menjadi jauh lebih cepat dibandingkan dengan Metode Pencarian Sekuensial dan Pencarian Interpolasi. Apalagi dengan data yang sangat banyak seperti data nomor induk mahasiswa (NIM) di ITB.

II. LANDASAN TEORI

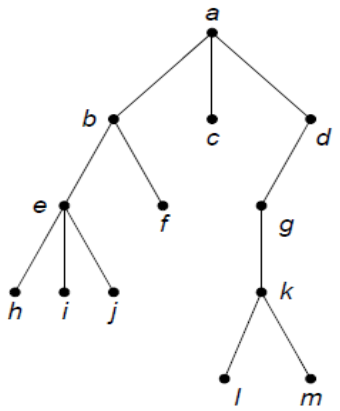
2.1 Pohon

2.1.1 Definisi

Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Ia merupakan salah satu bentuk struktur data tak linear yang menggambarkan hubungan yang bersifat hierarkis (hubungan one to many) antara elemen-elemen. Salah satu elemen tree disebut akar dan elemen yang lain disebut simpul.

2.1.2 Pohon Berakar

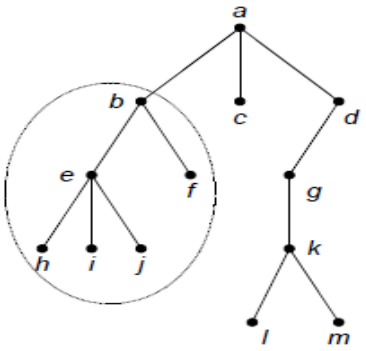
Pohon Berakar adalah pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah. Sebagai perjanjian, tanda anah pada sisi dapat dibuang.



Gambar 2.1 Pohon Berakar

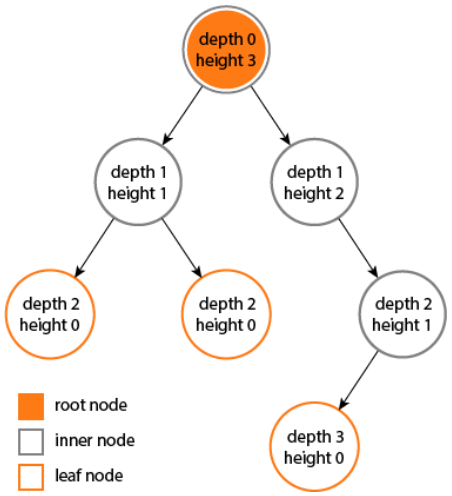
Terminologi pada pohon berakar :

1. Orang tua : simpul yang berada diatas simpul tertentu.
2. Anak : simpul yang berada di bawah simpul tertentu.
3. Lintasan : Lintasan dari a ke j adalah a,b,e,j. Panjang lintasan dari a ke j adalah 3.
4. Saudara kandung : f adalah saudara kandung e, tetapi g bukan saudara kandung e, karena orang tua berbeda.
5. Upapohon : Berikut adalah contoh sebuah upapohon.



Gambar 1.2 Upapohon

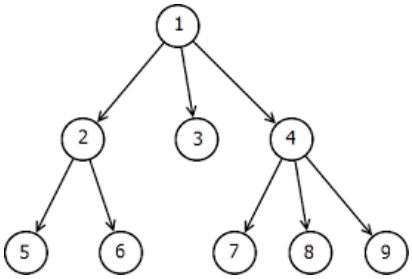
6. Derajat : Jumlah anak atau upapohon dari simpul tersebut. Derajat yang dimaksud adalah derajat keluar. Contoh : *a* memiliki derajat 2, *b* memiliki derajat 3.
7. Daun (*leaf*) : Simpul berderajat 0. Contoh daun : simpul *d, i, j ,k, f,* dan *h*.
8. Simpul dalam (*internal simpuls*) : Simpul yang memiliki anak disebut simpul dalam. Contoh : simpul *b, e, g, c*.
9. Aras *(level)*, tingkat, atau kedalaman *(depth)*, dan tinggi*(height) :*



Gambar 2.3 Level and Depth

2.1.3 Pohon *n-ary*

Pohon *n-ary* adalah pohon berakar yang setiap simpul cabangnya mempunyai paling banyak n buah anak. Pohon *n-ary* dikatakan teratur atau penuh(full) jika setiap cabangnya mempunyai tepat n-anak. Contoh Pohon *n-ary* :

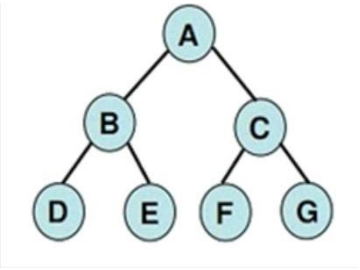


Gambar 2.4 Pohon 3-ary

Tetapi dalam pembahasan ini, kita akan menggunakan pohon biner (2-ary).

2.1.4 Pohon Biner

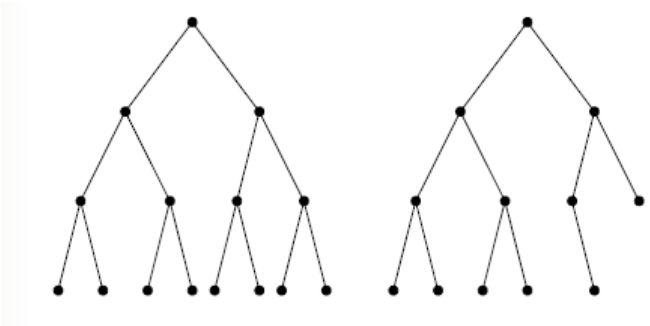
Pohon biner merupakan pohon *n-ary* dengan *n=2*. Pohon biner adalah pohon yang paling penting karena banyak aplikasinya. Setiap simpul di dalam pohon biner mempunyai paling banyak 2 buah anak. Kedua anak tersebut dibedakan antara anak kiri *(Left Child)* dan anak kanan *(Right Child)*. Karena ada perbedaan urutan anak, maka pohon biner adalah pohon terurut.



Gambar 2.5 Pohon Biner

2.1.5 Pohon Biner Seimbang

Pohon biner seimbang adalah pohon dengan perbedaan tinggi subpohon kiri dan subpohon kanan maksimum satu. Lalu perbedaan banyak simpul subpohon kiri dan subpohon kanan maksimum satu. Subpohon kiri dan subpohon kanan adalah pohon seimbang.



Gambar 2 6 Pohon Biner Seimbang

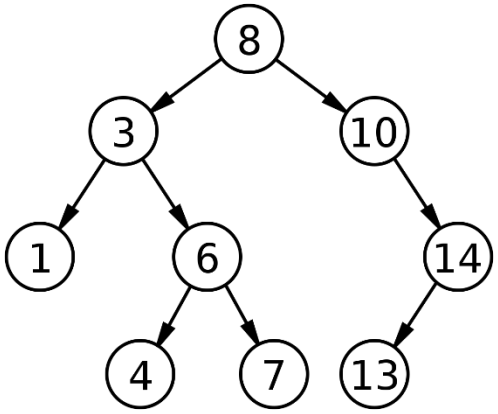
2.2 Pohon Pencarian Biner

2.2.1 Definisi

Pohon Pencarian Biner merupakan pohon Biner yang semua *Key* simpul Sub Pohon dikirinya memiliki nilai lebih kecil daripada *Key* simpul tersebut. Lalu *Key* simpul Sub Pohon di kanannya memiliki nilai lebih besar atau sama daripada *Key* simpul tersebut.

$$Key(Left(Simpul)) \leq Key(Simpul) \leq Key(Right(Simpul))$$

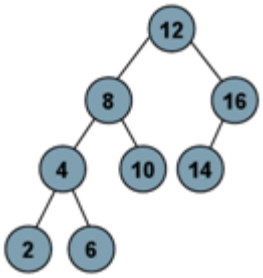
Pohon pencarian biner sengaja disusun terurut seperti itu untuk membantu dalam proses pencarian. Pohon ini memang dibuat untuk tujuan mempermudah algoritma pencarian *Binary Search* untuk kasus pohon, sehingga pencarian dilakukan dengan mencari key dari nilai data yang diinginkan, hal ini memanfaatkan sifat dari *Binary Search* dengan kompleksitas : $\Theta(\log n)$.



Gambar 2.7 Pohon Pencarian Biner

2.2.2 Pohon Pencarian Biner Seimbang (AVL Tree)

AVL Tree adalah Pohon pencarian biner yang memiliki perbedaan tinggi/ level maksimal 1 antara subpohon kiri dan subpohon kanan. AVL Tree muncul untuk menyeimbangkan pohon pencarian biner. Dengan AVL Tree, waktu pencarian dan bentuk pohon dapat dipersingkat dan disederhanakan.

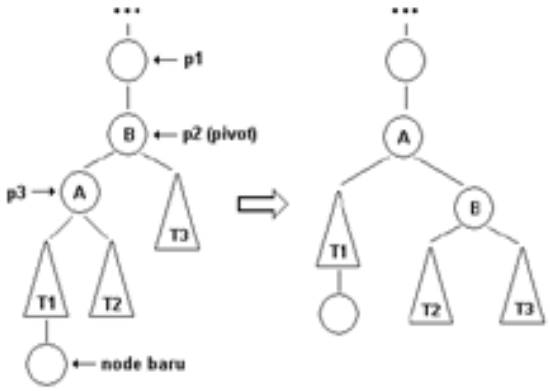


Gambar 2.9 AVL Tree Awal

Untuk menjaga pohon tetap imbang, setelah penyisipan sebuah simpul, dilakukan pemeriksaan dari simpul baru (root). Simpul pertama yang memiliki |*balance factor*| > 1 diseimbangkan. Proses penyeimbangan dilakukan dengan : *Single Rotation* dan *Double Rotation.*

2.2.2.1 Single Rotation

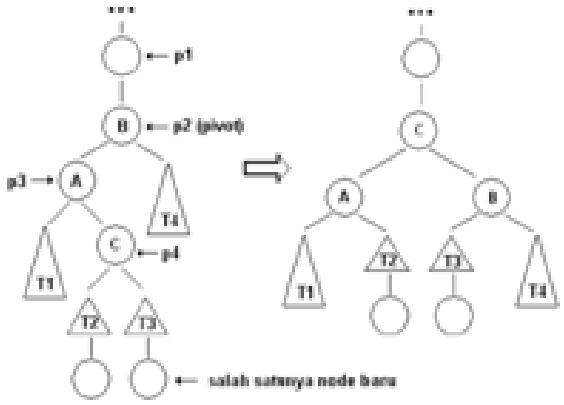
Single rotation dilakukan bila kondisi AVL tree waktu akan ditambahkan simpul baru dan posisi simpul baru seperti pada gambar 2. T1, T2, dan T3 adalah subtree yang urutannya harus seperti demikian serta height- nya harus sama (≥ 0). Hal ini juga berlaku untuk AVL tree yang merupakan citra cermin (mirror image) gambar 10.



Gambar 2.10 Single Rotation

2.2.2.2 Double Rotation

Double rotation dilakukan bila kondisi AVL tree waktu akan ditambahkan simpul baru dan posisi simpul baru seperti pada gambar 3. T1, T2, T3, dan T4 adalah subtree yang urutannya harus seperti demikian. Tinggi subtree T1 harus sama dengan T4 (≥ 0), tinggi subtree T2 harus sama dengan T3 (≥ 0). Simpul yang ditambahkan akan menjadi child dari subtree T2 atau T3. Hal ini juga berlaku untuk AVL tree yang merupakan citra cermin (mirror image) gambar 11.



Gambar 2.11 Double Rotation

2.2.2.3 Menghapus Simpul di AVL Tree

Proses menghapus sebuah simpul di AVL tree hampir sama dengan BST. Penghapusan sebuah simpul dapat menyebabkan tree tidak imbang Setelah menghapus sebuah simpul, lakukan pengecekan dari simpul yang dihapus → root. Gunakan single atau double rotation untuk menyeimbangkan simpul yang tidak imbang. Pencarian simpul yang imbalance diteruskan sampai root.

2.3 Pencarian Biner (*Binary Search)* di Pohon

Algoritma Pencarian Biner bekerja dengan kompleksitas algoritma sebesar $O(^2\log_n)$ untuk kasus terburuk maupun kasus terbaik. Untuk cara kerjanya, pertama program akan membandingkan *Key* dari Akar node paling atas dengan Key yang dicari.

Misalkan *Key* yang dicari adalah *a.* Jika *a* lebih besar besar *Key* dari Node tersebut, maka *pointer* dari node akan pindah ke Sub Pohon Kanan *Node* tersebut. Akan tetapi, jika *a* lebih kecil dari *Key* Node yang sedang di *compare,* maka *pointer node* akan berpindah ke Sub Pohon kiri *node* tersebut. Proses ini akan terus berulang hingga *Key* dari Node yang dibandingkan sama dengan *a* atau *pointer* sudah mencapai daun dari pohon, dan nilai *a* tidak ditemukan.

```
procedure PencarianBiner(input a1, a2, ..., an : integer, x : integer,
                         output idx : integer)
Deklarasi
   i, j, mid : integer
   ketemu : boolean

Algoritma
   i←1
   j←n
   ketemu←false
   while (not ketemu) and ( i ≤ j) do
      mid ← (i+j) div 2
      if amid = x then
        ketemu ← true
      else
         if amid < x then    { cari di belahan kanan }
          i←mid + 1
         else                { cari di belahan kiri }
          j←mid - 1;
         endif
      endif
   endwhile
   {ketemu or i > j }

   if ketemu then
      idx←mid
   else
      idx←0
   endif
```

Gambar 2.12 Contoh Pencarian Biner dalam Notasi Algoritmik

N	2^N
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384

Tabel 2.1 Perbandingan banyak pencarian dengan banyak elemen

Dengan Data dari Tabel 2.1 diatas, dapat kita ketahui bahwa jika banyak elemen lebih kecil sama dengan dari 2^N dan lebih besar dari 2^{N-1} , maka batas maksimal dari banyak pencarian dalam pohon akan sebanyak N.

III. PENERAPAN POHON PENCARIAN BINER SEIMBANG DALAM ALGORITMA PENCARIAN NOMOR INDUK MAHASISWA

Pada bagian pendahuluan, telah dipaparkan bahwa algoritma

pencarian biner merupakan algoritma *searching* yang paling efisien dibandingkan dengan algoritma *searching* yang lain. Maka dalam Bab ini akan dibahas bagaimana menerapkan algoritma tersebut dalam Aplikasi NIM Finder sederhana dengan menambahkan beberapa modifikasi di ADT Tree.

Dengan menggunakan pohon pencarian biner seimbang, kompleksitas algoritma searching akan jauh lebih efisien. Dengan $O(^2\log_n)$.

Untuk mengimplementasikan pohon pencarian biner seimbang, kita harus membuat ADT Binary Tree yang sudah di modifikasi terlebih dahulu agar sesuaikan dengan ADT Binary Search Tree Seimbang (Pohon Pencarian Biner Seimbang).

Dalam pohon pencarian biner seimbang, terdapat beberapa ADT fungsi prosedur, beberapa diantaranya yaitu fungsi untuk mencari *Key* dalam Pohon yaitu *SearchBST* yang mengembalikan *Infotype* pohon. Lalu *InsertKey* untuk memasukkan elemen baru ke dalam pohon secara terurut. *DeleteKey* untuk menghapus suatu elemen di dalam pohon secara terurut.

Dalam Aplikasi NIM Finder sederhana ini. Akan ada 2 tipe Pohon Pencarian Biner:

1. Pohon Pencarian Biner dengan Key Nama Lengkap.
2. Pohon Pencarian Biner dengan Key NIM.

InfoType dari Pohon Pencarian Biner berdasarkan Nama Lengkap dibuat *struct* dengan isi sebagai berikut:

1. Nama Lengkap
2. Fakultas/Jurusan
3. Tahun Masuk
4. NIM TPB
5. NIM Jurusan (Jika masih TPB, akan diisi '-')

InfoType dari Pohon Pencarian Biner berdasarkan NIM dibuat *struct* dengan isi sebagai berikut:

1. NIM
2. Nama Lengkap
3. Fakultas/Jurusan
4. Tahun Masuk

Uuntuk PPB berdasarkan NIM, NIM Jurusan dengan NIM TPB adalah 2 elemen yang berbeda. Tapi isi dari variable nama lengkap, Fakultas/Jurusan, dan Tahun Masuk sama.

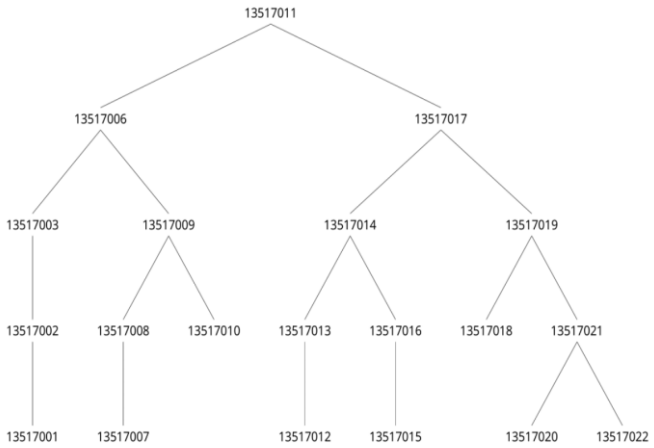
Sedangkan untuk PPB berdasarkan Nama, PPB akan mengurutkan Nama berdasarkan jumlah Ascii dari character pada Nama di pohon.

Saat *User* memasukkan inputan *string*, program akan mengecek apakah *string*[0]nya pertamanya angka atau bukan. Jika angka, maka akan program akan masuk ke Pohon Pencarian Biner berdasarkan NIM. Tetapi jika *string[0]*nya merupakan sebuah huruf, maka program akan masuk ke Pohon Pencarian Biner berdasarkan nama. Dimana untuk pohon nama, tidak ada penyortiran seperti pohon NIM (hanya 1 pohon besar).

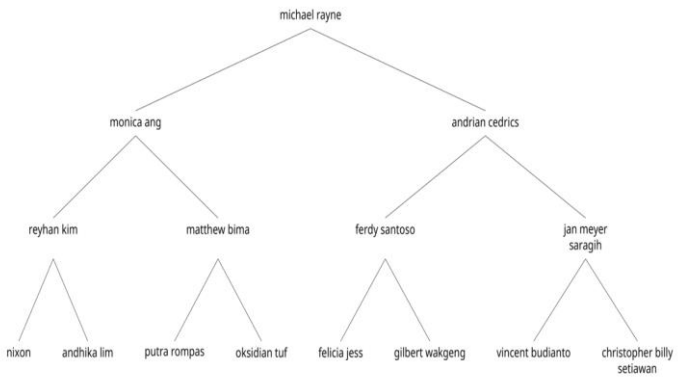
Setelah mengecek string ke-0, untuk tipe Pohon NIM, program akan mengecek string ke 0,1, dan 2. Berdasarkan susunan ketiga char itu, program akan memilih Jurusan atau TPB. Jika NIM jurusan, maka program akan langsung mencari pohon jurusan yang tepat. Misalkan 135 berarti program akan lompat ke pohon jurusan Teknik Informatika. Apabila NIM yang dicari adalah NIM TPB, maka program akan meloncat ke pohon NIM TPB sesuai dengan fakultasnya masing-masing.

Kemudian Program akan langsung mencocokkan string inputan user dengan pohon biner yang sudah ditetapkan di awal dan memulai algoritma Pencarian Biner (*Binary Search)* di dalam Pohon. Perlu diingat Aplikasi NIM Finder sederhana ini hanya bisa mencari Nama Lengkap dan NIM yang benar-benar sesuai atau sama dengan *Key* dari pohon.

Ketika match ditemukan, maka program akan menampilkan Informasi dari Mahasiswa tersebut (struct) yang sudah dipaparkan diatas. Akan tetapi jika match tidak ditemukan, maka program akan mengeluarkan pesan "No Match Found".



Gambar 3.1 Contoh Pohon NIM Jurusan



Gambar 3.2 Contoh Pohon Nama

IV. KESIMPULAN

Pohon adalah struktur data yang tercipta dari pengembangan graf. Pohon merupakan salah satu struktur data yang terbilang banyak aplikasinya di dunia pemrograman. Salah satu kegunaannya adalah untuk mempermudah kerja algoritma pencarian biner. Salah satu contoh sederhana dari pengimplementasian pohon pencarian biner adalah membuat aplikasi NIM Finder. Tetapi harus diakui masih banyak kekurang dalam pengimplementasiannya. Seperti saat awal memasukkan data ke dalam pohon tersebut.

V. UCAPAN TERIMA KASIH

Pertama-tama penulis ingin mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena dengan rahmat dan karunia-Nya penulis dapat menyelesaikan makalah dengan judul "Penerapan Pohon Pencarian Biner Seimbang dalam Aplikasi NIM Finder Sederhana" ini dengan baik. Penulis juga berterima kasih kepada dosen yang memberikan tugas ini, Dr. Ir. Rinaldi Munir, M.T., dan kepada dosen pengajar, Dra. Harlili S., M.Sc., atas bimbingan beliau selama ini dalam mengajar dan memberikan ilmu dalam mata kuliah matematika diskrit sehingga penulis mampu membuat makalah ini. Selain itu, penulis juga berterima kasih kepada rekan-rekan yang telah memberikan semangat dan dorongan kepada penulis sehingga makalah ini dapat diselesaikan.

REFERENCES

[1] Rinaldi Munir, Matematika Diskrit. Bandung : Penerbit Informatika, Palasari.
[2] http://student.blog.dinus.ac.id/fredoandrean/2017/08/08/jenis-jenis-searching-dalam-algoritma-pemrograman/ diakses pada 7 Desember 2018 pukul 18.37
[3] https://www.geeksforgeeks.org/avl-tree-set-1-insertion/ diakses pada 8 Desember 2018 pukul 15.46
[4] http://dinda-dinho.blogspot.com/2013/06/pengertian-dan-konsep-avl-tree.html diakses pada 8 Desember 2018 pukul 17.58
[5] https://www.tutorialspoint.com/data_structures_algorithms/binary_search_tree diakses pada 8 Desember 2018 pukul 18.32
[6] https://www.geeksforgeeks.org/binary-search/ diakses pada 8 Desember 2018 pada pukul 19.25

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, 9 Desember 2017

Christopher Billy Setiawan
13517050