Pengaplikasian Pohon dalam Sistem Repository Ubuntu Linux

Ricardo Pramana Suranta / 13509014[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13509014@std.stei.itb.ac.id

Sistem *Repository* adalah salah satu sistem instalasi program yang tersedia dalam Ubuntu Linux. Sistem *repository* menjamin setiap software yang terdapat didalamnya sesuai dengan versi dengan Ubuntu yang digunakan, oleh karena telah diuji coba oleh pihak penyedia *repository* tersebut. Pengaplikasian pohon dalam sistem *indexing* dari *repository* dapat membantu efisiensi dalam pengunduhan sebuah *software dari repository,* dan pendekatan pohon juga dapat digunakan untuk memeriksa *dependency* suatu *software,* beserta tahap – tahap instalasi *software* tersebut kedalam komputer.

Kata kunci : repository, dependency, pohon, binary search tree

I. PENDAHULUAN

Ubuntu Linux adalah salah satu distribusi (atau yang seringkali disebut *distro)* Linux yang cukup populer akhir – akhir ini. Dari sekian banyak distribusi Linux yang beredar di pasaran, seperti OpenSUSE, Mandriva, Fedora, dan banyak lainnya, Ubuntu banyak dipilih oleh karena gratis (beberapa perusahaan seperti Novell yang menjual SUSE seharga sekitar $50 USD dan RedHat Enterprise yang menjual RedHat dengan harga $110 USD), diperbaharui (*update*) secara berkala, yakni enam bulan sekali, memiliki banyak dukungan teknis (*technical support*), baik dari pihak Canonical, yakni pengembang Ubuntu yang dikepalai oleh Mark Shuttleworth, maupun oleh sesama *user* di internet, menyediakan berbagai program untuk segala jenis bidang, baik permainan dan hiburan, multimedia, edukasi, pemrograman, dan lainnya secara gratis, menyediakan *source - code* (kode sumber yang dapat di*compile* kedalam bahasa mesin) dari sistem operasi Ubuntu sendiri maupun program – program yang disediakan didalamnya (sebagian besar, beberapa tidak mau membagikan *source – code* programnya), sehingga dapat dipelajari dan dikembangkan oleh banyak orang, serta banyak lainnya. Oleh karena popularitasnya, sistem operasi ini sering disandingkan dengan dua sistem operasi berbayar yang paling terkenal di dunia hingga saat ini, yakni Microsoft Windows dan Apple OSX. Tidak jarang, Ubuntu juga dimodifikasi oleh orang – orang menjadi sebuah *distro* yang membawa program – program tambahan serta *skin* atau penampilan yang berbeda dari yang disediakan oleh Ubuntu pada dasarnya, seperti LinuxMint, Ubuntu Studio, dan OSGX yang merupakan salah satu karya mahasiswa Teknik Informatika ITB. Modifikasi Ubuntu ini juga diterima dengan baik di pasaran, terutama kepada mereka yang memiliki koneksi internet yang "minim" untuk memodifikasi Ubuntu mereka untuk keperluan sehari – hari.

Sistem yang dimiliki Ubuntu sendiri cukup berbeda dengan Windows yang kerap kali dipakai oleh khayalak ramai. Salah satu perbedaan yang signifikan dari keduanya adalah sistem *repository* yang dimiliki oleh Ubuntu. Berbeda dengan Windows, dimana kebanyakan program yang digunakan oleh *user* namun tidak berasal dari Windows, seperti Adobe Photoshop dan CorelDraw, harus diinstalasi kedalam mesin melalui *installer* yang dimiliki (kerap kali dalam format *.exe atau *.msi) oleh *user* sendiri, Ubuntu menyediakan sebuah sistem dimana *user* cukup mencari program yang diinginkan untuk diinstall kedalam komputer yang terdapat didalam daftar program yang disediakan, lalu memerintahkan komputer untuk melakukan instalasi, baik dengan memasukkan sebuah *command line* pada aplikasi Terminal maupun melalui sebuah klik dari mouse melalui Ubuntu Software Center. Sistem *repository* ini merupakan salah satu fungsi dasar yang dimiliki Debian Linux, yang menjadi dasar dari Ubuntu.

Dalam makalah ini, penulis hendak menyorot sistem *repository* yang dimiliki oleh Ubuntu melalui pendekatan pohon.

II. DASAR TEORI

2.1 Pohon

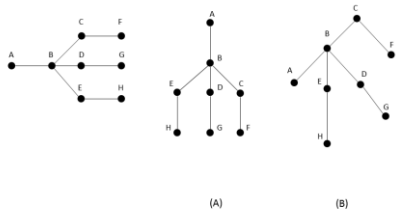
Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Sebuah pohon dibentuk paling sedikit oleh sebuah simpul tanpa sisi. Berdasarkan perlakuan terhadap simpul yang menyusun sebuah pohon, pohon dapat dibagi menjadi dua jenis, antara lain :

A. Pohon Bebas

Pohon bebas adalah sebuah pohon dimana tiap simpulnya diperlakukan sama, tidak ada yang dianggap sebagai akar, daun, maupun lainnya.

B. Pohon Berakar

Pohon berakar adalah sebuah pohon dimana salah satu simpulnya dianggap sebagai sebuah akar, dan simpul – simpul lainnya dapat dicapai melalui akar, dengan memberikan arah pada sisi – sisi pohon yang mengikutinya. Simpul yang terhubung dengan sebuah akar disebut sebagai daun, dan bila simpul tersebut terhubung dengan simpul lainnya (selain simpul yang menjadi akar dari simpul tersebut), maka simpul tersebut adalah akar terhadap simpul lainnya itu. Pengambilan simpul yang digunakan sebagai akar dari sebuah pohon bebas akan menghasilkan pohon yang berbeda terhadap simpul yang lainnya. Sebuah pohon yang memiliki jumlah daun yang konstan untuk setiap akar yang dimilikinya disebut pohon *n-ary*, dengan n sebagai jumlah daun yang dimiliki tiap akar dari pohon tersebut.



Gambar 1 : Sebuah pohon berakar yang diambil dari pohon bebas. gambar (A) menunjukkan pengambilan simpul A sebagai akar utama, dan gambar (B) dengan simpul C sebagai simpul utama

Terdapat beberapa terminologi pohon berakar yang digunakan secara umum, antara lain:

a. Upapohon
Upapohon adalah sebuah simpul yang menjadi "anak" dari simpul akar (terhubung dengan simpul akar), berserta simpul – simpul lain yang menjadi "anak" dari simpul tersebut. Pada gambar 1.B, kita dapat menyatakan bahwa simpul B beserta "anak - anak"nya adalah upapohon dari pohon tersebut. Sebuah pohon dapat terdiri dari banyak upapohon.

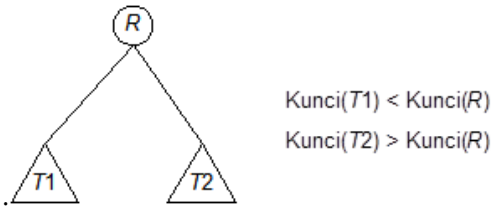
b. Derajat(*degree)*
Derajat sebuah simpul pada pohon berakar adalah jumlah upapohon pada simpul tersebut. Pada gambar 1.A, kita dapat melihat bahwa simpul B memiliki derajat sebesar 3.

c. Daun
Daun adalah sebuah simpul berderajat nol. Pada gambar 1.B, kita dapat menyatakan bahwa simpul H dan G adalah daun dari pohon tersebut.

d. Aras (level) atau Tingkat
Akar memiliki aras = 0, sedangkan aras simpul lainnya = 1 + panjang lintasan dari akar ke simpul tersebut. Pada gambar 1.A, simpul D memiliki aras sebesar 2, dan simpul G memiliki aras sebesar 3.

e. Tinggi (height) atau Kedalaman (depth)
Aras maksimum dari suatu pohon disebut **tinggi** atau **kedalaman** pohon tersebut. Pohon pada gambar 1.B memiliki tinggi 3.

Terdapat beberapa pohon berakar khusus dalam aplikasinya, salah satunya adalah *binary search tree* (pohon pencarian biner), yang disingkat sebagai BST. BST adalah pohon yang sering digunakan untuk persoalan yang banyak melakukan operasi pencarian, penyisipan, atau penghapusan elemen. Simpul dalam pohon ini dapat berupa sebuah *field* (kunci) pada data *record* atau data itu sendiri, dengan catatan setiap data haruslah unik. Penyusunan kunci dalam pohon biner disusun dalam suatu urutan tertentu. Jika R adalah akar dari sebuah BST, maka semua simpul pada upapohon kiri memiliki kunci yang nilainya lebih kecil dari kunci simpul R, dan semua simpul pada upapohon kanan memiliki kunci yang nilainya lebih besar dari kunci simpul R



Gambar 2: Sebuah binary search tree (BST)

2.2 Repository System

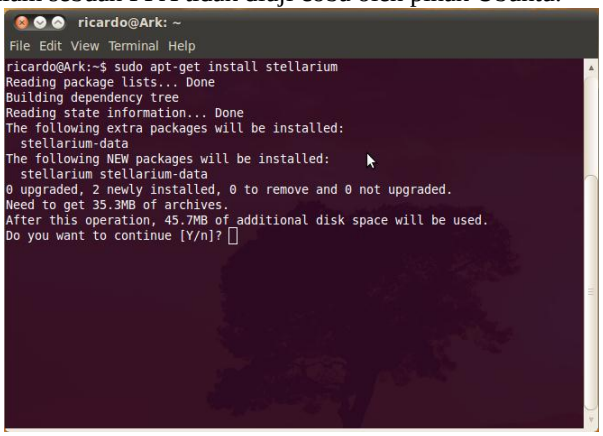
Repository system adalah suatu sistem dimana perangkat lunak *(software)* yang diperlukan disimpan didalam suatu arsip yang dinamakan *repository*. Setiap *software* maupun *library* yang terdapat dalam *repository* disebut sebagai *package*, dan berbentuk *.deb, *.tar.gz, maupun format kompresi lainnya yang sering digunakan. Dalam penerapannya oleh Ubuntu, tiap *software* yang disimpan dalam *repository* diuji coba terlebih dahulu oleh pihak Ubuntu, sehingga keamanan serta kompabilitas *software* tersebut terjamin. Ketika *user* menginginkan sebuah perangkat lunak untuk diinstall kedalam komputernya, *user* tersebut cukup memerintahkan komputer untuk mengunduh program tersebut dari *repository* yang disediakan, beserta *library* maupun program lainnya yang menjadi prasyarat dari instalasi program yang diminta tersebut, yang disebut sebagai *dependency*. Oleh pihak Ubuntu, *repository* yang disediakan dibagi berdasarkan versi *release* dari Ubuntu yang dipakai. *Repository* sendiri dibagi menjadi beberapa golongan, yakni :

- **Main** : *software* yang didukung secara resmi.
- **Restricted** : *software* yang didukung, namun tidak sepenuhnya berada dibawah

lisensi bebas, oleh karena ada beberapa daerah yang mengharuskan membayar hak paten untuk penggunaannya. (contoh : MP3 codec)

- **Universe** : *software* yang tidak didukung secara resmi, namun pemeliharaannya dilakukan oleh komunitas.
- **Multiverse** : *software* yang tidak benar – benar bebas digunakan, oleh Karena tidak diizinkan untuk digunakan oleh jurisdiksi dari beberapa daerah.

Beberapa pihak maupun perseorangan selain Ubuntu juga menyediakan *repository* milik mereka sendiri untuk digunakan oleh orang lain, yang seringkali disebut sebagai Personal Package Archives. Pihak tersebut menyediakan alamat dari *repository* yang dimilikinya, dan *user* yang menginginkan *software* yang terdapat dalam *repository* tersebut cukup memasukkan alamat tersebut kedalam daftar *repository* yang dimiliki Ubuntu. Dianjurkan untuk menggunakan PPA yang sudah terjamin keamanannya, dan banyak digunakan oleh orang lain (seperti milik Google atau Opera), oleh karena perangkat lunak yang disediakan dalam sebuah PPA tidak diuji coba oleh pihak Ubuntu.



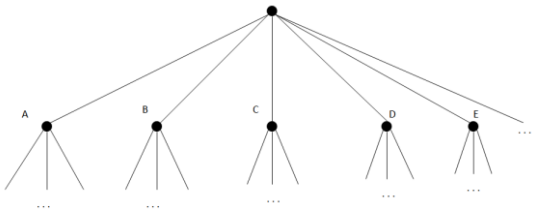
Gambar 3 dan 4 : Instalasi melalui command line (atas) dan melalui GUI (bawah) dari sebuah program di Ubuntu

III. PENGAPLIKASIAN POHON DALAM SISTEM REPOSITORY

Dalam sistem *repository* Ubuntu, pohon dapat digunakan sedikitnya dalam dua hal, antara lain *indexing* dan pengecekan *dependency* suatu *software* serta instalasi *software* tersebut kedalam komputer.

A.Indexing

Ada banyak bentuk pohon yang dapat diaplikasikan untuk *indexing* dalam sebuah *repository*. Salah satu bentuk yang saya ajukan ialah sebuah pohon, dengan satu buah akar, dan daun pada aras pertama merepresentasikan abjad yang menjadi huruf awal dari sebuah *package* (A - Z), dan daun dari tiap – tiap abjad tersebut adalah alamat dari *package* yang kita inginkan, sehingga dapat diunduh. Sebagai contoh, daun dari simpul abjad A adalah *software* Akonadi, Amarok, dll. , daun dari simpul abjad O adalah OpenOffice Word Processor, OpenOffice Spreadsheet, dll., dan begitu seterusnya.

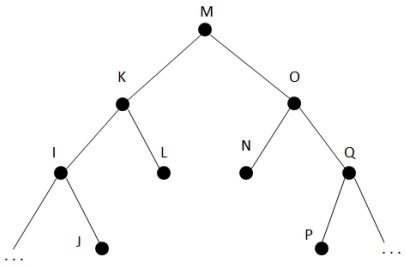


Gambar 5 : Pohon dengan akar pada aras pertama berupa abjad (A-Z) dan aras kedua berupa alamat dari suatu package

Bentuk pohon ini dapat "mempersingkat" pencarian suatu *package*, daripada hanya sekedar "menyisir" semua *package* yang ada di *repository* hingga package yang diinginkan ditemukan. Sebagai contoh, bila kita ingin mengunduh sebuah *package* yang memilliki huruf awal Z, kita cukup menelusuri daun pada aras pertama hingga mencapai abjad Z, lalu menelusuri seluruh *package* yang terdapat dalam daun – daun yang dimiliki oleh simpul Z. Penelusuran dengan pohon ini lebih baik daripada harus menelusuri seluruh *package* yang ada, sebab dalam kebanyakan kasus, proses yang dilakukan untuk mencari *package* tersebut (atau yang disebut sebagai kompleksitas waktu) melalui pohon akan jauh lebih sedikit daripada "menyisir" semua *package* yang ada di *repository*. Asumsikan kita memiliki 1024 *package* dalam *repository*, apabila *package* yang inginkan tidak berada pada *repository* tersebut, komputer akan melakukan 1024 kali pembandingan (untuk membandingkan *package* yang terdapat saat itu dengan yang diinginkan *user*), sedangkan dengan menggunakan pohon, komputer hanya akan melakukan proses sebanyak 26 + X, dimana proses sebanyak 26 kali adalah proses pencarian daun Z dalam pohon utama, dan X adalah jumlah *package* yang diawali dengan abjad Z dalam *repository* tersebut. Kasus terburuk dari penggunaan pohon ini adalah apabila seluruh *package* dalam *repository* tersebut diawali oleh abjad yang sama. Dari contoh diatas, apabila seluruh 1024 *package* yang terdapat dalam *repository* berawal dengan huruf Z, maka proses yang dilakukan untuk mencari, apabila *package* yang kita inginkan tidak terdapat dalam *repository* tersebut

adalah 26 + 1024 = 1050 kali proses pembandingan, yang berarti lebih banyak daripada "sekedar menyusuri" ke-1024 *package* dalam *repository*. Namun, kejadian seperti ini jarang terjadi, oleh karena pada umumnya, sebuah *repository* terdiri dari berbagai macam *package*, yang diawali oleh abjad yang berbeda – beda pula.

Bentuk pohon lain yang dapat digunakan untuk *indexing* adalah *binary search tree*. Abjad yang berada di tengah – tengah susunan abjad, yakni M, diambil sebagai akar, dan daun – daunnya disusun sesuai urutan, dimana abjad yang berada dua urutan sebelum M, yakni abjad K, diletakkan pada daun sebelah kiri dari akar M, sedangkan abjad L, yang berada diantara K dan M diletakkan sebagai daun sebelah kanan dari K. Daun kiri dari abjad K juga diisi oleh abjad yang berada dua urutan sebelum K, yakni I, dan abjad J yang berada di tengah – tengah mereka menjadi daun sebelah kanan dari abjad I, begitu seterusnya. Hal yang sama juga dilakukan untuk abjad yang berada setelah abjad M, dimana abjad O yang berada dua urutan setelah M diletakkan pada daun sebelah kanan M, dan abjad N yang berada diantara M dan O diletakkan sebagai daun sebelah kiri abjad O, dst. Lalu, masing – masing abjad yang terdapat pohon tersebut menyimpan sebuah *address* yang menunjuk kepada sebuah *binary search tree* lainnya, yang terdiri atas seluruh *package* yang terdapat dalam abjad tersebut. Ada dua opsi yang dapat digunakan dalam membentuk *binary search tree* ini, sehingga dapat mempersingkat proses pencarian. Pertama, kita dapat mengurutkan seluruh *package* yang ada dalam abjad tersebut, dan mengambil *package* yang berada di tengah – tengah urutan tersebut sebagai akar dari *binary search tree* yang dibentuk. Kedua, kita dapat menggunakan *package* yang paling sering diakses sebagai akar, lalu, *package* yang paling sering diakses kedua dan ketiga sebagai daun dari akar tersebut (peletakan sebagai daun kiri atau kanan disesuaikan dengan algoritma pembentukan *binary search tree*), dan begitu seterusnya.



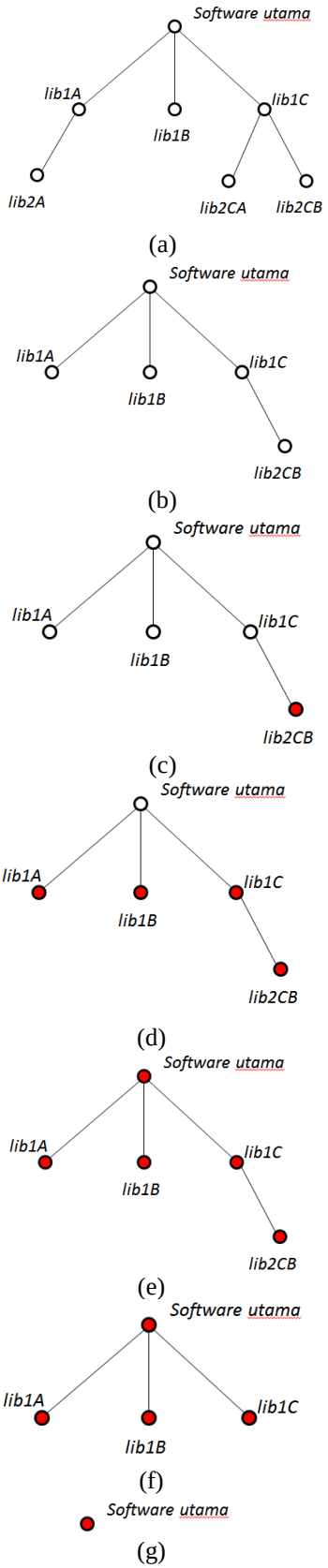
Gambar 6: Binary Search Tree untuk indexing abjad

B.Pembentukan Pohon Dependency dan Instalasi Software

Seperti yang diutarakan sebelumnya, dalam instalasi sebuah *software*, adalah hal yang wajar apabila *software* tersebut meminta prasyarat beberapa *package* tertentu harus diinstall terlebih dahulu dalam computer kita. Gambar 2 menunjukkan proses instalasi sebuah aplikasi bernama Stellarium, yang meminta agar *package* stellarium-data diinstall terlebih dahulu kedalam komputer. Dalam hal ini, *package* yang menjadi *dependency* dari *software* Stellarium adalah stellarium-data. Untuk memeriksa apakah suatu komputer sudah menginstall *package* yang menjadi *dependency* suatu software, kita dapat menggunakan pohon biasa, dimana kita menggunakan *package* dari *software* utama sebagai akar, dan meletakkan *package* yang menjadi *dependency* dari *software* tersebut sebagai daunnya. Tiap simpul yang menjadi daun harus memiliki sebuah variabel, yang digunakan untuk memeriksa apakah *package* tersebut telah diunduh atau belum. Contohnya, apabila variabel yang digunakan bertipe *Boolean*, maka kita dapat mengisi variabel tersebut dengan nilai *true*, dengan perjanjian bahwa nilai *true* menandakan bahwa *package* tersebut telah diunduh. Lalu, setiap *package* yang terdapat dalam daun tersebut diperiksa, berikut adalah langkah – langkahnya:

1. Bila *package* tersebut telah terinstall didalam komputer, maka daun tersebut dihapus dari pohon yang ada.
2. Bila *package* yang terdapat dalam daun tersebut belum terdapat dalam komputer, maka kita harus memeriksa seluruh *dependency* dari *package* tersebut, lalu menempatkannya sebagai daun dari simpul *package* tersebut, setelah itu setiap *package* yang terdapat pada daun tersebut kembali diperiksa. Jika *package* tersebut tidak memiliki *dependency* atau semua *dependency* dari *package* tersebut telah terpenuhi, maka *package* tersebut dapat diunduh, dan variabel penanda yang terdapat pada simpul dapat diisi dengan nilai *true*.
3. Bila *package* tersebut ada di dalam komputer namun belum diinstall (biasanya terjadi apabila koneksi internet terputus pada proses pengunduhan *package*), maka variabel penanda dari simpul tersebut cukup diisi dengan nilai *true*.

Proses ini terus diulang hingga setiap *package* yang menjadi *dependency* telah diunduh. Proses selanjutnya adalah pengunduhan *package* dari software utama yang diminta, setelah itu instalasi setiap *package* yang menjadi tersebut, dimulai dari *package* yang terdapat pada daun dengan aras tertinggi. Seusai instalasi dari *package* tersebut, maka simpul yang menjadi penanda dihapus. Proses ini terus diulang hingga hanya tersisa satu simpul yang terdapat dari pohon tersebut, yakni simpul yang menjadi penanda atas *package* dari *software* utama, yang berarti menandakan seluruh *dependency* telah diinstall kedalam komputer. Setelah itu, *package* dari *software* utama dapat diinstall kedalam komputer, dan pohon tersebut dapat dihapus.



Gambar 7 : Proses instalasi sebuah software dengan dependency tree, dengan tahapan sebagai berikut (a) seluruh dependency dari software utama dimasukkan sebagai daun dari simpul dari program utama, (b) pengecekan bagian yang sudah terinstall, (c) – (d) pengunduhan package, (e) – (g) instalasi program.

IV. KESIMPULAN

Pendekatan pohon dapat digunakan untuk sistem *repository* dari Ubuntu Linux, sedikitnya dalam *indexing* dan pengecekan *dependency* dari suatu *software* yang hendak diinstall. Pengaplikasian pohon untuk *indexing* dapat meningkatkan efisiensi dari pencarian suatu *package* dalam sebuah *repository* untuk sebagian besar kasus, dan pengaplikasian pohon untuk pemeriksaan *dependency* dan instalasi sebuah *software* memungkinkan proses instalasi yang terstruktur atas sebuah *software* kedalam komputer.

REFERENCES

[1] Munir, Rinaldi, *Diktat Kuliah IF2091 Struktur Diskrit*, Penerbit Informatika : Bandung, 2008

[2] http://www.easy-ubuntu-linux.com/other-ubuntus.html, diakses pada tanggal 16 Desember 2010, pukul 19.00

[3] https://help.ubuntu.com/community/Repositories/Ubuntu, diakses pada tanggal 16 Desember 2010, pukul 19.00

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, 17 Desember 2010

Ricardo Pramana Suranta / 13509014