PEMBANGUNAN APLIKASI BANTU KECERDASAN BUATAN PEMBELAJARAN DAN PENYELESAIAN PADA PERMASALAHAN RUANG KEADAAN *(STATE AND SPACE)*

ARI WIBOWO 10107371



JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK DAN ILMU KOMPUTER
UNIVERSITAS KOMPUTER INDONESIA
BANDUNG
2012

LATAR BELAKANG MASALAH

- **Kecerdasan Buatan** merupakan salah satu bagian ilmu komputer yang mempelajari tentang bagaimana cara membuat agar komputer dapat melakukan pekerjaan seperti yang dilakukan oleh manusia.
- Salah satu teori kecerdasan buatan yang dibahas yaitu mengenai **permasalahan Ruang Keadaan (*State and Space)***
- ***State and space*** adalah semua keadaan yang mungkin, dan biasanya digambarkan sebagai jaringan dengan *verteks* merupakan *state* dan *edge* merupakan perubahan yang mungkin.



MAKSUD DAN TUJUAN

- Membangun suatu aplikasi bantu kecerdasan buatan pembelajaran pada permasalahan ruang keadaan.
- Membangun aplikasi untuk mensimulasikan langkah penyelesaian kasus-kasus yang menyerupai permasalahan dan aturan ruang keadaan.
- Menyediakan aplikasi bantu kecerdasan yang dinamis, dimana *user* dapat memasukkan kasus permasalahan ruang keadaan yang ingin diselesaikan.



BATASAN MASALAH

- **Membangun aplikasi** kecerdasan buatan sebagai media pembelajaran dan penyelesaian masalah ruang keadaan (*State and Space*) dengan menggunakan bahasa pemrograman *Visual Basic 6.0*.
- Aplikasi berbentuk **simulator** yang menghasilkan suatu bentuk pemahaman mengenai permasalahan ruang keadaan.
- Aplikasi menampilkan **teori dasar dan manual** mengenai permasalahan *State and Space.*
- yang akan digunakan untuk mensimulasikan permasalahan ruang keadaan yaitu permasala**Problem atau kasus** han tempat, kondisi, dan waktu. Seperti kasus yang sudah umum yaitu *Farmers Problem*, kemudian disertakan juga contoh kasus lainnya yang banyak disimulasikan dalam *Game logic* yaitu *Family Crossing Problem*, *Angry Father* dan juga permasalahan yang mirip yang telah dimodifikasi sendiri.
- Problem baru selain contoh diatas dapat diinputkan oleh *user* secara **dinamis** dengan memasukkan keadaan awal dari permasalahan, serta menginputkan variabel sendiri.



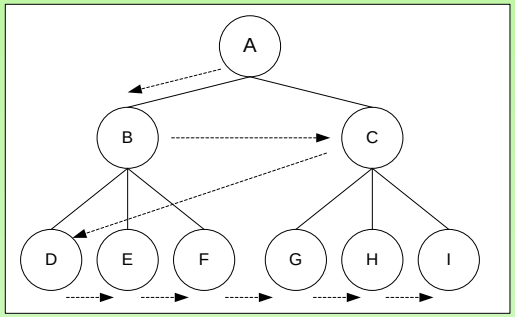
BATASAN MASALAH

- Aplikasi mencari dan menampilkan langkah penyelesaian terpendek ***(shortest path)*.**
- Metode penyelesaian menggunakan metode pencarian melebar pertama **(*Breadth-First Search*)** karena pencarian solusi ini tidak akan menemui jalan buntu, jika ada satu solusi, maka *breadth-first search* akan menemukannya. Dan jika ada lebih dari satu solusi, maka solusi minimum (*Shortest path)* akan ditemukan dan yang dipilih. Karena sesuai dengan batasan masalah pada poin sebelumnya yaitu aplikasi mencari dan menampilkan langkah penyelesaian terpendek.
- Jika tidak terdapat solusi, maka aplikasi akan menampilkan **pesan kesalahan (*error message*)** yang menandakan bahwa problem dan aturan yang dibuat oleh *user* melalui inputan tidak memenuhi format pada batasan masalah di poin 4.
- Langkah penyelesaian dapat dijalankan secara manual maupun secara otomatis oleh aplikasi.



PEMBAHASAN BREADTH-FIRST SEARCH

Pada metode *Breadth-First Search*, semua *node* pada level n akan dikunjungi terlebih dahulu sebelum mengunjungi *node-node* pada level n+1. Pencarian dimulai dari *node* akar terus ke level ke-1 dari kiri ke kanan, kemudian berpindah ke level berikutnya. Demikian seterusnya hingga ditemukannya solusi.



Algoritma :

- Buat suatu variabel Node_List dan tetapkan sebagai keadaan awal.
- Kerjakan langkah-langkah berikut ini sampai tujuan tercapai atau Node_List dalam keadaan kosong :
  - Hapus elemen pertama dari Node_List, sebut dengan nama D. Jika Node_List kosong, keluar.
  - Pada setiap langkah yang aturannya cocok dengan D, kerjakan :
    - Aplikasikan aturan tersebut untuk membentuk suatu keadaan baru
    - Jika keadaan awal adalah tujuan yang diharapkan, sukses dan keluar.
    - Jika tidak demikian, tambahkan keadaan awal yang baru tersebut pada akhir Node_List.



PEMBAHASAN BREADTH-FIRST SEARCH

Keuntungan dari metode ini:

- Tidak akan menemui jalan buntu.
- Jika ada satu solusi, maka *breadth-first search* akan menemukannya. Dan jika ada lebih dari satu solusi, maka solusi minimum akan ditemukan.

Kelemahan dari metode ini:

- Membutuhkan memori yang cukup banyak, karena menyimpan semua *node* dalam satu pohon.
- Membutuhkan waktu yang cukup lama, karena akan menguji n level untuk mendapatkan solusi pada level ke- (n + 1).



CONTOH KASUS PERMASALAHAN RUANG KEADAAN DAN PENYELESAIAN MENGGUNAKAN METODE BREADTH-FIRST SEARCH

Salah satu contoh permasalahan ruang keadaan yang dibahas adalah masalah petani, kambing, serigala dan sayur-sayuran yang sering juga disebut dengan *Farmer's Problem*. Ilustrasi dari permasalahan ini adalah sebagai berikut :

-seorang petani akan menyeberangkan seekor kambing, seekor serigala dan sayur-sayuran dengan sebuah rakit melalui sungai.

-- Rakit hanya bisa memuat petani dan satu penumpang yang lain (kambing, serigala atau sayur-sayuran).

- Jika ditinggalkan oleh petani tersebut, maka sayur-sayuran akan dimakan oleh kambing dan kambing akan dimakan oleh serigala



PENYELESAIAN :

•Definisikan ruang masalah, keadaan awal (*initial state)* dan keadaan tujuan (*goal state).*

Keadaan awal (*start state*):

Daerah kiri = (1,1,1,1).
Daerah kanan = (0,0,0,0)
Posisi rakit berada di kiri.

Artinya, terdapat kambing, petani, serigala dan sayur-sayuran di daerah kiri.

Keadaan tujuan (*goal state*):

Daerah kiri = (0,0,0,0).
Daerah kanan = (1,1,1,1)
Posisi rakit berada di kanan.

Artinya, terdapat kambing, petani, serigala dan sayur-sayuran di daerah kanan.

CONTOH KASUS PERMASALAHAN RUANG KEADAAN DAN PENYELESAIAN MENGGUNAKAN METODE BREADTH-FIRST SEARCH

PENYELESAIAN :

• Definisikan aturan produksi

mendefinisikan aturan produksi yang berlaku, mendefinisikan Variabel, Aturan (*rule*), dan strategi kontrol.

Variabel ID	Nama Variabel	Bisa mengemudikan rakit (Driver)
K	Kambing	Tidak
P	Petani	Ya
SR	Serigala	Tidak
SY	Sayur-sayuran	Tidak

No	Control
1	Kambing akan memakan sayur-sayuran apabila Petani tidak berada di tempat.
2	Serigala akan memakan Kambing apabila Petani tidak berada di tempat.



CONTOH KASUS PERMASALAHAN RUANG KEADAAN DAN PENYELESAIAN MENGGUNAKAN METODE BREADTH-FIRST SEARCH

PENYELESAIAN :

Aksi ke-	Aturan
1	Petani dan kambing menyeberang.
2	Petani dan serigala menyeberang.
3	Petani dan sayur-sayuran menyeberang.
4	Petani dan kambing kembali.
5	Petani dan serigala kembali.
6	Petani dan sayur-sayuran kembali.
7	Petani kembali.

- **Memilih metode pencarian yang tepat**

Metode pencarian yang dipergunakan yaitu metode pencarian buta (*Blind Search*) yaitu metode ***Breadth-First Search (BFS).***



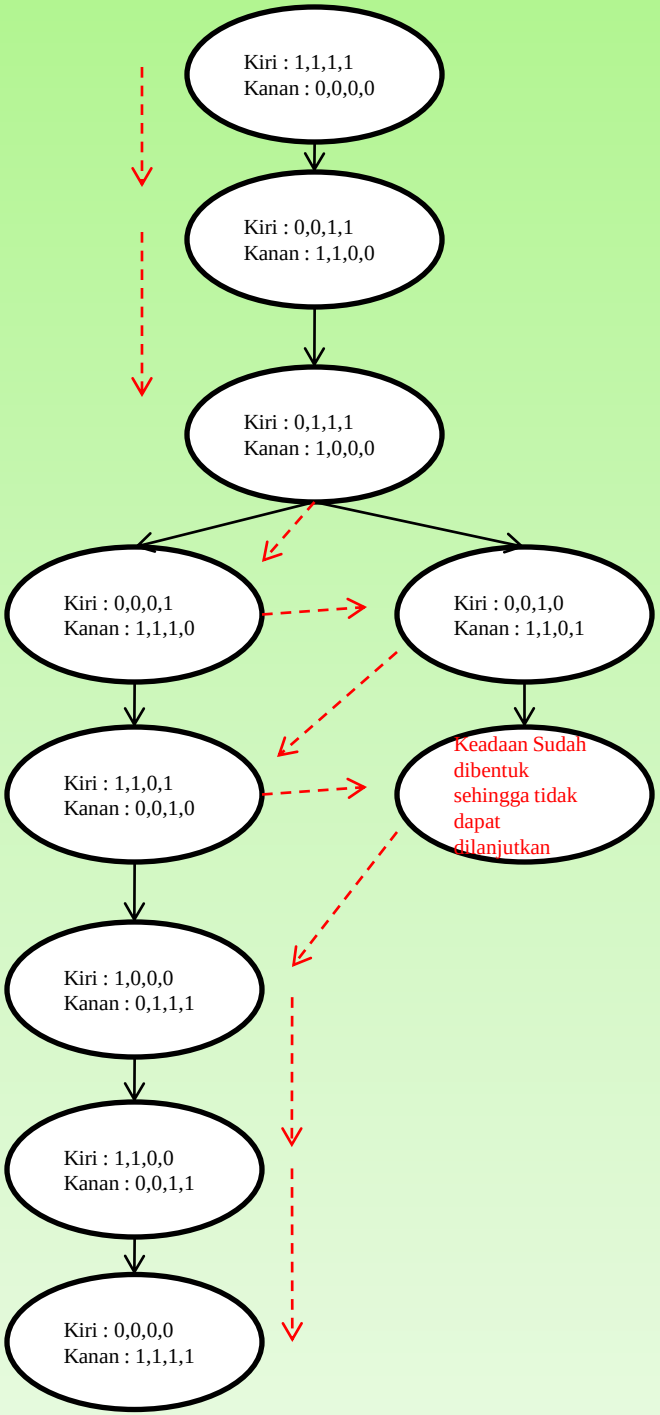
CONTOH KASUS PERMASALAHAN RUANG KEADAAN DAN PENYELESAIAN MENGGUNAKAN METODE BREADTH-FIRST SEARCH

PENYELESAIAN :

Berikut ini merupakan struktur pohon pelacakan untuk mencari solusi *farmer's problem* menggunakan *Breadth First Search* :



Pohon Pelacakan Kasus Farmer Problems



TABEL SOLUSI

Daerah Kiri	Aksi	Posisi Rakit	Daerah Kanan
(1,1,1,1)	-	KIRI	(0,0,0,0)
(0,0,1,1)	Aksi ke-1 (Petani dan kambing menyeberang)	KANAN	(1,1,0,0)
(0,1,1,1)	Aksi ke-7 (Petani kembali)	KIRI	(1,0,0,0)
(0,0,0,1)	Aksi ke-2 (Petani dan serigala menyeberang)	KANAN	(1,1,1,0)
(1,1,0,1)	Aksi ke-4 (Petani dan kambing kembali)	KIRI	(0,0,1,0)
(1,0,0,0)	Aksi ke-3 (Petani dan sayuran menyeberang)	KANAN	(0,1,1,1)
(1,1,0,0)	Aksi ke-7 (Petani kembali)	KIRI	(0,0,1,1)
(0,0,0,0)	Aksi ke-1 (Petani dan kambing menyeberang)	KANAN	(1,1,1,1)

ERD

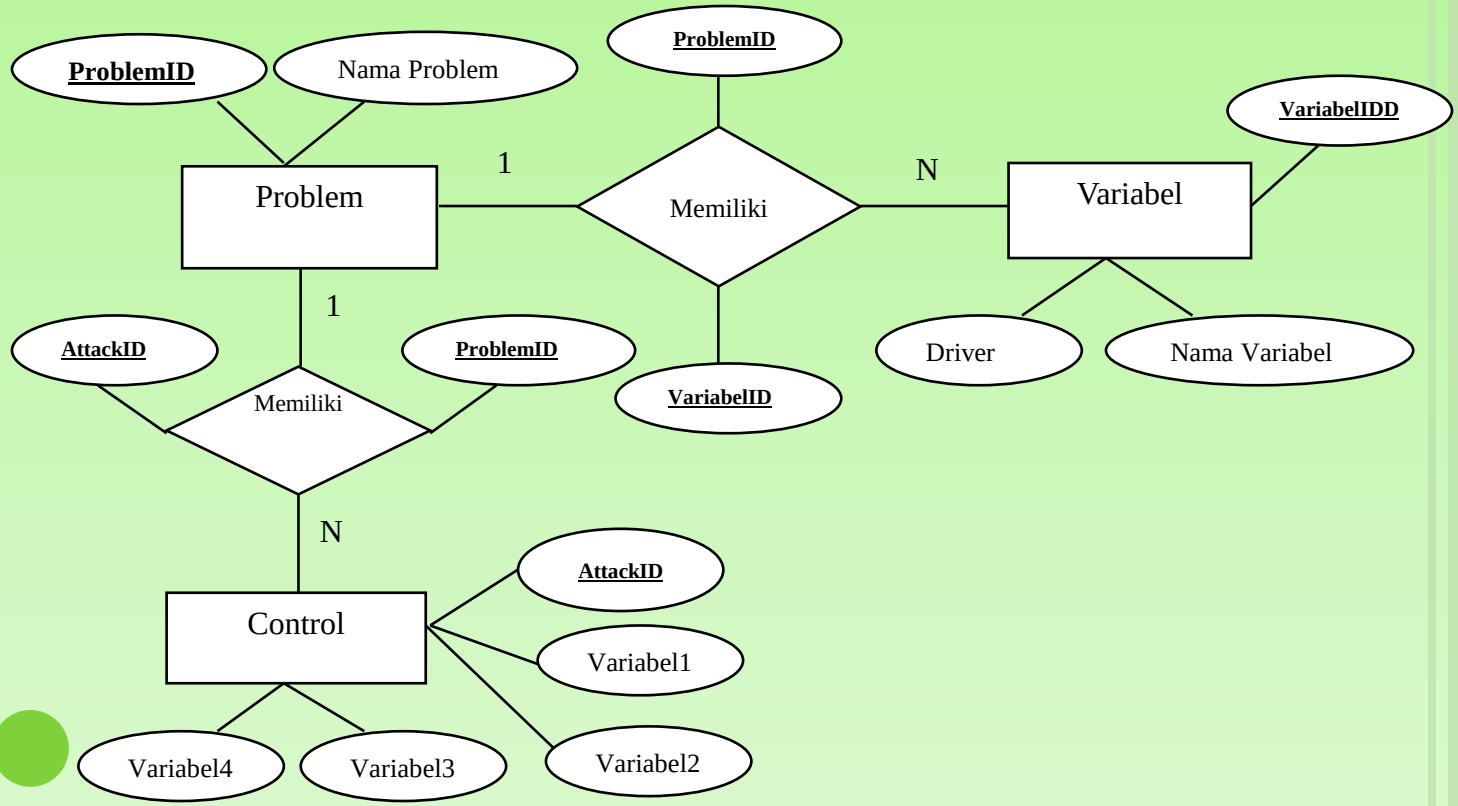
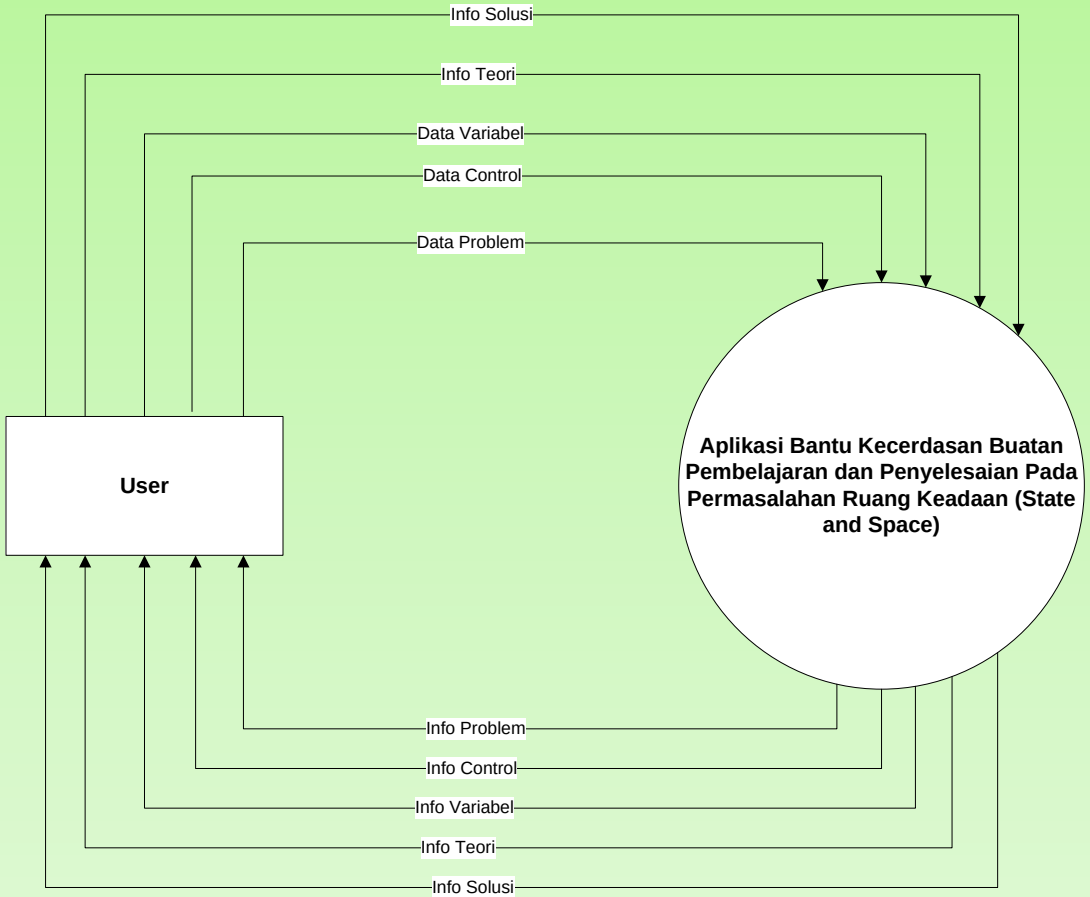

ProblemID
Nama Problem
ProblemID
VariabelIDD
Problem
1
Memiliki
N
Variabel
1
AttackID
ProblemID
Memiliki
VariabelID
Driver
Nama Variabel
N
AttackID
Control
Variabel1
Variabel4
Variabel3
Variabel2

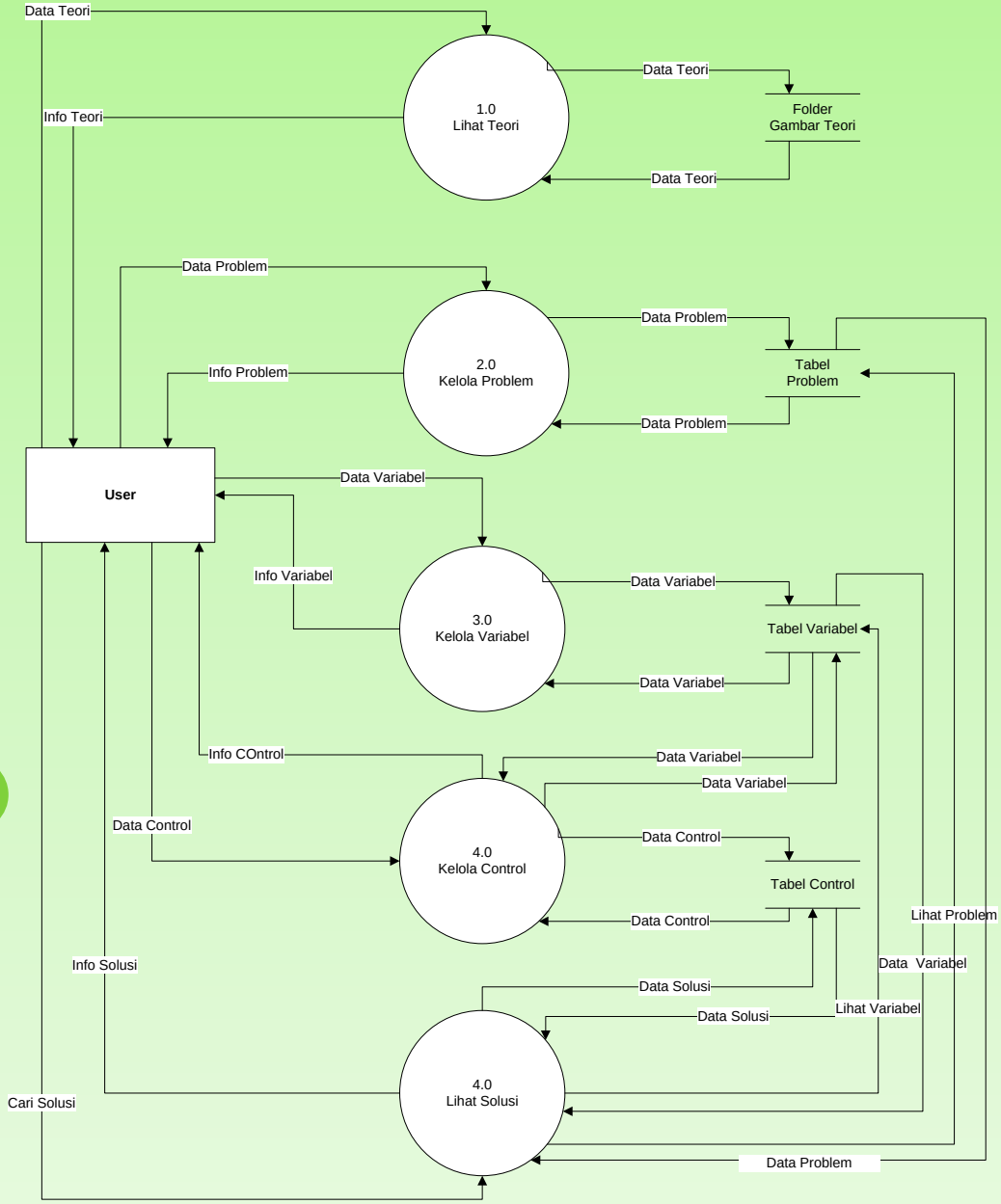


Diagram Konteks





DFD Level 1


Data Teori
1.0
Lihat Teori
Data Teori
Folder
Gambar Teori
Data Teori
Info Teori
Data Problem
2.0
Kelola Problem
Data Problem
Tabel
Problem
Data Problem
Info Problem
User
Data Variabel
Info Variabel
3.0
Kelola Variabel
Data Variabel
Tabel Variabel
Data Variabel
Info COntrol
Data Variabel
Data Variabel
Data Control
4.0
Kelola Control
Data Control
Tabel Control
Data Control
Lihat Problem
Info Solusi
Data Solusi
Data Variabel
Data Solusi
Lihat Variabel
4.0
Lihat Solusi
Cari Solusi
Data Problem



DEMO PROGRAM

