PERANGKAT LUNAK PENCARIAN SOLUSI PERMASALAHAN TEKO AIR (*WATERJUG PROBLEM*) MENGGUNAKAN ALGORITMA *BREADTH FIRST SEARCH* (*BFS*)

Timbo Faritcan Parlaungan S [*1], Imin Tugimin[#2]

Program Studi Teknik Informatika, STMIK Subang
Jl. Marsinu No. 5 - Subang, Tlp. 0206-417853 Fax. 0206-411873
*E-mail: timbo_faritcan_parlaungan@yahoo.com[*1], imintugimin@yahoo.co.id[#2]*

ABSTRAKSI

Permasalahan teko air merupakan suatu permasalahan klasik dalam bidang ilmu *Artificial Intelligence* (AI). Permasalahan ini dapat diilustrasikan seperti berikut, terdapat 2 buah teko air masing-masing memiliki kapasitas *x* dan *y* liter. Permasalahannya adalah bagaimana mendapatkan air sebanyak *n* liter dengan menggunakan bantuan kedua teko air tersebut dan mengambil asumsi bahwa sumber air tidak terbatas. Aksi-aksi yang dapat dilakukan, antara lain mengisi teko air, mengosongkan teko air dan menuangkan isi teko air ke teko air lain.

Permasalahan ini dapat diselesaikan dengan menerapkan konsep AI yaitu dengan bantuan pohon pelacakan dan menerapkan metode pencarian melebar pertama (*breadth-first search* / BFS). Pencarian solusi dimulai dari kondisi dimana kedua teko kosong (*node* akar dari pohon pelacakan). Proses dilanjutkan dengan menggambarkan kondisi (*state*) berikutnya (dengan melakukan aksi terhadap *state* sebelumnya) hingga semua *state* diperiksa dan mendapatkan tujuan (*goal state*).

Perangkat lunak dapat digunakan sebagai fasilitas pendukung dalam proses belajar mengajar khususnya mata pelajaran Algoritma, juga dapat menambah daya tarik kreativitas peserta didik. Akhir kata, Penulis menyadari bahwa Penulisan Ilmiah ini masih terdapat banyak kekurangan, oleh karena itu dengan senang hati Penulis menerima kritikan dan saran yang dapat menunjang Penulisan Ilmiah ini. Namun besar harapan Penulis semoga Penulisan Ilmiah ini dapat berguna dan bermanfaat bagi almamater tercinta Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK Subang).

Kata Kunci: ***Artificial Intelligenc,* Algoritma *Breadth First Search* (*BFS*)**

1. Pendahuluan

1.1. Latar Belakang

Permasalahan teko air merupakan suatu permasalahan klasik dalam bidang ilmu *Artificial Intelligence* (AI). Permasalahan ini dapat diilustrasikan seperti berikut, terdapat 2 buah teko air masing-masing memiliki kapasitas *x* dan *y* liter. Permasalahannya adalah bagaimana mendapatkan air sebanyak *n* liter dengan menggunakan bantuan kedua teko air tersebut dan mengambil asumsi bahwa sumber air tidak terbatas. Aksi-aksi yang dapat dilakukan, antara lain mengisi teko air, mengosongkan teko air dan menuangkan isi teko air ke teko air lain.

Permasalahan ini dapat diselesaikan dengan menerapkan konsep AI yaitu dengan bantuan pohon pelacakan dan menerapkan metode pencarian melebar pertama (*breadth-first search* / BFS). Pencarian solusi dimulai dari kondisi dimana kedua teko kosong (*node* akar dari pohon pelacakan). Proses dilanjutkan dengan menggambarkan kondisi (*state*) berikutnya (dengan melakukan aksi terhadap *state* sebelumnya) hingga semua *state* diperiksa dan mendapatkan tujuan (*goal state*).

1.2. Identifikasi Masalah

Sesuai dengan uraian dalam latar belakang di atas, yang menjadi permasalahan dalam penelitian ini adalah bagaimana merancang perangkat lunak untuk mencari solusi permasalahan kendi air..

1.3. Tujuan

Tujuan yang diperoleh dari penelitian ini adalah merancang dan membuat perangkat lunak pencarian solusi permasalahan teko air (*waterjug problem*) menggunakan algoritma *breadth first*

search (*bfs*).

1.4. Manfaat

Manfaat yang ingin dicapai adalah:

- Membantu pemahaman permasalahan teko air.
- Mencari solusi untuk menyelesaikan permasalahan teko air dengan menggunakan metode pencarian melebar pertama (*breadth-first search*).
- Perangkat lunak dapat digunakan sebagai fasilitas pendukung dalam proses belajar mengajar.

1.5. Metodologi Penelitian

Metode penelitian yang akan digunakan dalam pembuatan sistem penentu keputusan ini adalah metode prancangan perangkat lunak *Waterfall*. Pengembangan metode *Waterfall* sendiri melalui beberapa tahapan yaitu

- Penelitian Lapangan (*Field Research*), Penelitian dilakukan secara langsung turun kelapangan tempat penelitian.
- Penelitian Kepustakaan (*Library Research*), Penelitian ini bertujuan untuk mendapatkan data yang bersifat teori seperti mengumpulkan buku-buku atau bahan lainnya.
- Observasi, Observasi yang dilakukan penulis adalah mengamati secara langsung data yang diperoleh.
- Analisis Perangkat Lunak, Kegiatan analisis perangkat lunak meliputi analisis spesifikasi perangkat lunak yang akan digunakan sebagai alat bantu penelitian.
- Perancangan Perangkat Lunak, Perancangan perangkat lunak meliputi perancangan keras dan perancangann antarmuka dari hasil analisis.
- Implementasi Perangkat Lunak, Implementasi dari hasil analisis dan perancangan perangkat lunak.
- Pengujian Perangkat Lunak, Pengujian terhadap perangkat lunak yang telah diimplementasikan.

2. Tinjauan Pustaka

2.1. Metode Pencarian dan Pelacakan

Konsep utama dalam menentukan keberhasilan sistem yang berlandaskan AI adalah kesuksesan dalam melakukan dan mengembangkan pencarian. Pencarian atau pelacakan merupakan salah satu teknik untuk menyelesaikan permasalahan AI. Keberhasilan suatu sistem, salah satunya ditentukan oleh kesuksesan dalam pencarian dan pencocokan. Teknik dasar pencarian memberikan suatu kunci bagi banyak sejarah penyelesaian yang penting dalam bidang AI. Ada beberapa aplikasi yang menggunakan teknik pencarian ini, yaitu:

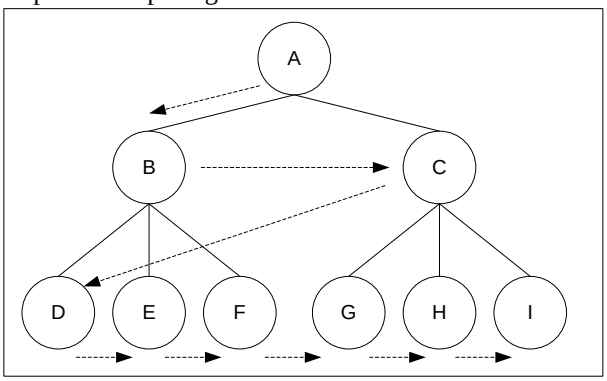
a. Papan *game* dan *puzzle* (tic-tac-toe, catur)
b. Penjadwalan dan masalah *routing* (*travelling salesman problem*)
c. *Parsing* bahasa dan interpretasinya (pencarian struktur dan arti)
d. Logika pemograman (pencarian fakta dan implikasinya)
e. *Computer vision* dan pengenalan pola

Konsep pencarian untuk suatu solusi dalam ruang keadaan (*state space*) merupakan pusat AI yang menjadikan AI lebih unggul dalam bidang ilmu komputer dibandingkan dengan yang lainnya, dan prinsip kontribusi AI untuk ilmu pengetahuan dari pencarian ini merupakan konsep basis pengetahuan (*knowledge based*) heuristik untuk pembatasan dan pencarian berarah (*directing search*). Pada dasarnya, ada dua teknik pencarian dan pelacakan yaitu:

a. Pencarian buta (*blind search*), terdiri atas:
   (i) Pencarian melebar pertama (*Breadth-First Search*)
   (ii) Pencarian mendalam pertama (*Depth-First Search*)
b. Pencarian heuristik (*heuristic search*).

2.2. Pencarian Melebar Pertama (*Breadth-First Search*)

Pada metode pencarian ini, 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. Gambaran pencarian BFS dapat dilihat pada gambar 1.



Gambar 1 Pencarian melebar pertama (*Breadth-First Search*)

Pada gambar 1, terlihat bahwa pencarian dimulai dari keadaan awal (*node* A). Dari *node* A dikembangkan dua *node* baru yang menjadi anaknya, yaitu *node* B dan *node* C. Penelusuran dilanjutkan ke *node* B (menghasilkan *node* D, E, F) dan *node* C (menghasilkan *node* G, H, I). Demikian seterusnya hingga ditemukan solusi.

Karena proses *breadth-first search* mengamati setiap *node* di setiap level graf sebelum bergerak menuju ruang yang lebih dalam, maka mula-mula semua keadaan akan dicapai lewat lintasan yang terpendek dari keadaan awal. Oleh sebab itu, proses ini menjamin ditemukannya lintasan terpendek dari keadaan awal ke keadaan tujuan. Keuntungan dari metode pencarian ini adalah:

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

Namun demikian, ada empat persoalan utama berkenaan dengan metode pencarian ini, yaitu:

1. Membutuhkan memori yang besar, karena menyimpan semua *node* dalam satu pohon. Jumlah *node* di setiap tingkat dari pohon bertambah secara eksponensial terhadap jumlah tingkat, dan semuanya ini harus disimpan sekaligus.
2. Membutuhkan sejumlah besar pekerjaan, khususnya jika lintasan solusi terpendek cukup panjang, karena jumlah *node* yang perlu diperiksa bertambah secara eksponensial terhadap panjang lintasan.
3. Tidak relevannya operator akan menambah jumlah *node* yang harus diperiksa sangat besar.
4. Relatif membutuhkan waktu yang cukup lama, karena akan menguji semua *node* pada level ke-*n* untuk mendapatkan solusi pada level ke- (*n* + 1).

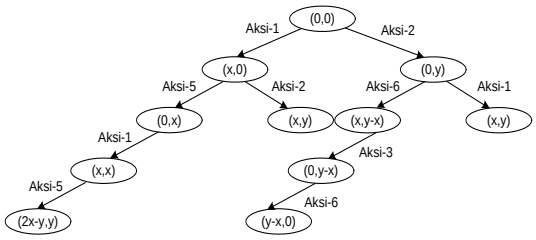
3. Analisa

3.1 Analisa Permasalah Teko Air

Permasalahan Teko Air merupakan salah satu permasalahan klasik dalam *Artificial Intelligence* (AI). Masalah ini dapat diselesaikan dengan menggunakan bantuan struktur pohon biner. Satu keadaan (*state*) mewakili satu keadaan.

Prosedur kerja untuk mencari solusi yang mungkin dari suatu keadaan dengan menggunakan bantuan struktur pohon biner adalah sebagai berikut :

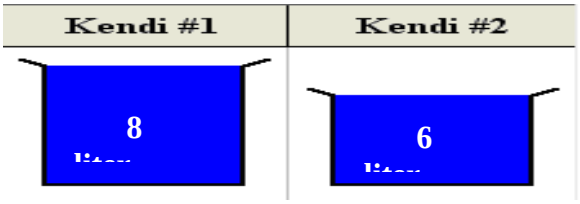
Misalkan besar kendi-1 sebesar x liter dan kendi-2 sebesar y liter, dengan nilai x lebih kecil daripada nilai y ($x < y$), maka bentuk struktur pohon biner untuk mencari solusinya adalah seperti terlihat pada gambar 2



Gambar 2 Struktur pohon biner untuk mencari solusi

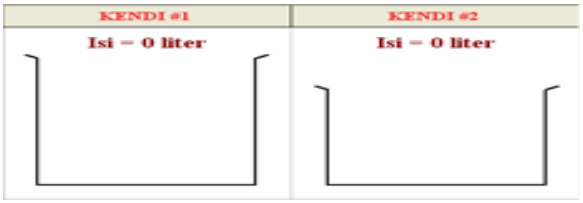
Struktur pohon di atas akan dilanjutkan terus hingga nilai yang terdapat pada daun sama dengan besar volume yang diinginkan. Untuk lebih jelas mengetahui mengenai proses pencarian solusi dari permasalahan teko air ini, simaklah contoh berikut ini :

Misalkan kapasitas kendi 1 = 8 liter, kendi 2 = 6 liter dan besar volume air yang diinginkan 4 liter seperti ditunjukkan gambar 3 berikut ini.



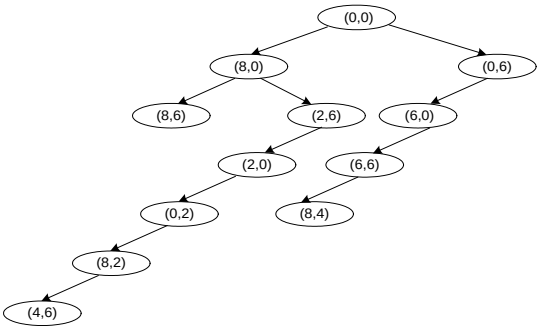
Gambar 3 Contoh Kasus Permasalahan Teko Air

Kondisi awal dari problem di atas adalah kedua kendi dalam keadaan kosong seperti ditunjukkan oleh gambar 4 berikut ini.



Gambar 4 Keadaan Awal Permasalahan Teko Air

Problem di atas dapat diselesaikan dengan menggunakan bantuan struktur pohon biner pada halaman berikut ini :

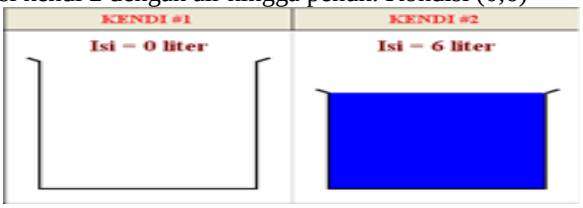


Gambar 5 Struktur pohon pencarian solusi

Sesuai dengan struktur pohon biner di atas, maka terdapat dua buah solusi, yaitu :

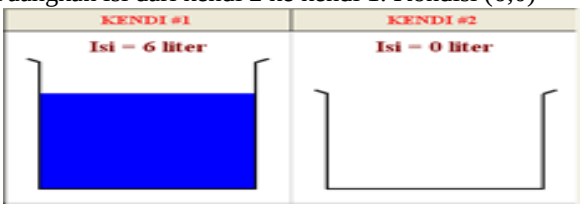
1. Solusi pertama sebanyak 4 langkah, dirincikan sebagai berikut :

a. Langkah 1 : Isi kendi 2 dengan air hingga penuh. Kondisi (0,6)



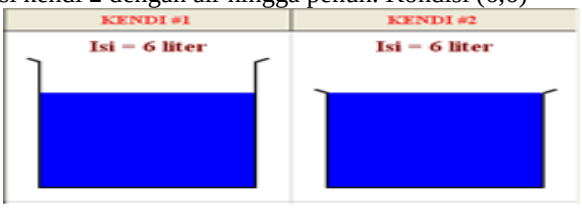
Gambar 6 Langkah-1 pada Solusi-1

b. Langkah 2 : Tuangkan isi dari kendi 2 ke kendi 1. Kondisi (6,0)



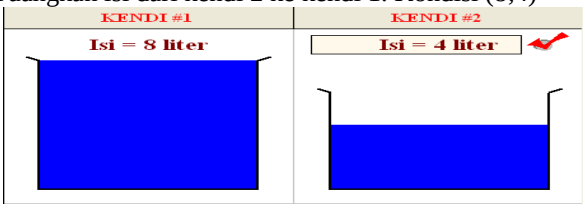
Gambar 7 Langkah-2 pada Solusi-1

c. Langkah 3 : Isi kendi 2 dengan air hingga penuh. Kondisi (6,6)



Gambar 8 Langkah-3 pada Solusi-1

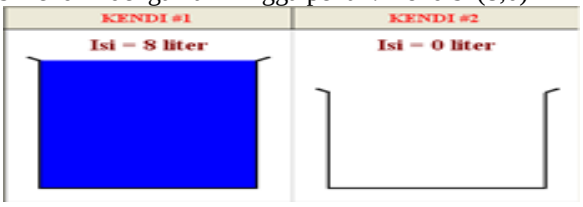
d. Langkah 4 : Tuangkan isi dari kendi 2 ke kendi 1. Kondisi (8,4)



Gambar 9 Langkah-4 pada Solusi-1

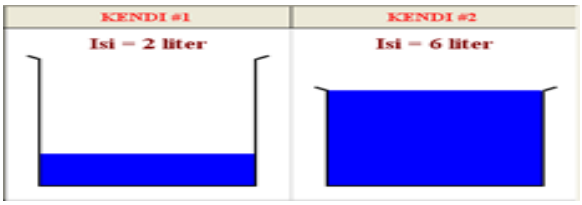
2. Solusi kedua sebanyak 6 langkah, dirincikan sebagai berikut :

a. Langkah 1 : Isi kendi 1 dengan air hingga penuh. Kondisi (8,0)



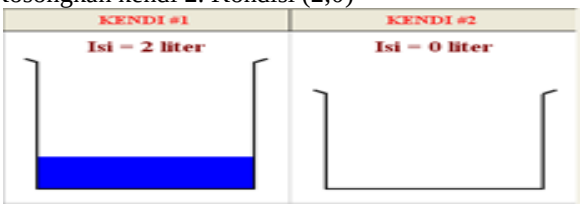
Gambar 10 Langkah-1 pada Solusi-2

b. Langkah 2 : Tuangkan isi dari kendi 1 ke kendi 2. Kondisi (2,6)



Gambar 11 Langkah-2 pada Solusi-2

c. Langkah 3 : Kosongkan kendi 2. Kondisi (2,0)

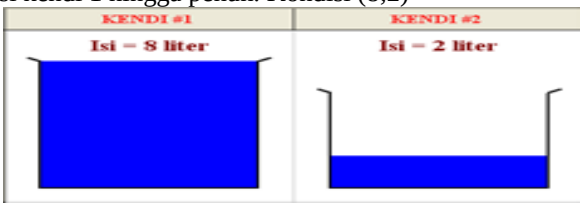


Gambar 12 Langkah-3 pada Solusi-2

d. Langkah 4 : Tuangkan isi dari kendi 1 ke kendi 2. Kondisi (0,2)

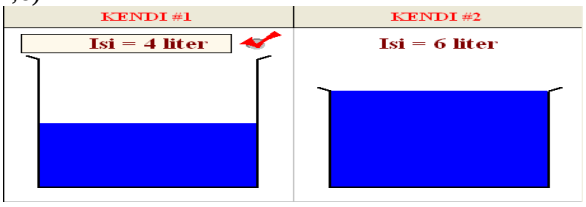
Gambar 13 Langkah-4 pada Solusi-2

e. Langkah 5 : Isi kendi 1 hingga penuh. Kondisi (8,2)



Gambar 14 Langkah-5 pada Solusi-2

f. Langkah 6 : Tuangkan isi dari kendi 1 ke kendi 2. Isi dari kendi 1 merupakan jawaban dari problem ini. Kondisi (4,6)



Gambar 15 Langkah-6 pada Solusi-2

Aturan yang ada pada setiap keadaan dari permasalahan teko air, yaitu :

1. Keadaan kedua kendi kosong, aturan yang berlaku berupa :
   a. Isi kendi 1 dengan air.
   b. Isi kendi 2 dengan air.
2. Keadaan kedua kendi terisi penuh dengan air, aturan yang berlaku berupa :
   a. Kosongkan kendi 1.
   b. Kosongkan kendi 2.
3. Keadaan kendi 1 terisi penuh dengan air dan kendi 2 kosong, aturan yang berlaku berupa :
   a. Kosongkan kendi 1.
   b. Tuangkan isi kendi 1 ke kendi 2.
   c. Isi kendi 2 dengan air.
4. Keadaan kendi 2 terisi penuh dengan air dan kendi 1 kosong, aturan yang berlaku berupa :
   a. Kosongkan kendi 2.
   b. Tuangkan isi kendi 2 ke kendi 1.
   c. Isi kendi 1 dengan air.
5. Keadaan kendi 1 terisi penuh dengan air dan kendi 2 juga terisi tetapi tidak penuh, aturan yang berlaku berupa :
   a. Kosongkan kendi 1.
   b. Kosongkan kendi 2.
   c. Tuangkan isi kendi 1 ke kendi 2.
   d. Isi kendi 2 dengan air.
6. Keadaan kendi 2 terisi penuh dengan air dan kendi 1 juga terisi tetapi tidak penuh, aturan yang berlaku berupa :
   a. Kosongkan kendi 1.
   b. Kosongkan kendi 2.
   c. Tuangkan isi kendi 2 ke kendi 1.
   d. Isi kendi 1 dengan air.
7. Keadaan kedua kendi terisi air tetapi tidak penuh, aturan yang berlaku berupa:
   a. Kosongkan kendi 1.
   b. Kosongkan kendi 2.
   c. Isi kendi 1 dengan air.
   d. Isi kendi 2 dengan air.
   e. Tuangkan isi kendi 1 ke kendi 2.
   f. Tuangkan isi kendi 2 ke kendi 1.
8. Keadaan kendi 1 terisi air tetapi tidak penuh dan kendi 2 kosong, aturan yang berlaku berupa :
   a. Kosongkan kendi 1.
   b. Isi kendi 1 dengan air.
   c. Isi kendi 2 dengan air.
   d. Tuangkan isi kendi 1 ke kendi 2.
9. Keadaan kendi 2 terisi air tetapi tidak penuh dan kendi 1 kosong, aturan yang berlaku berupa :
   a. Kosongkan kendi 2.
   b. Isi kendi 1 dengan air.
   c. Isi kendi 2 dengan air.
   d. Tuangkan isi kendi 2 ke kendi 1.

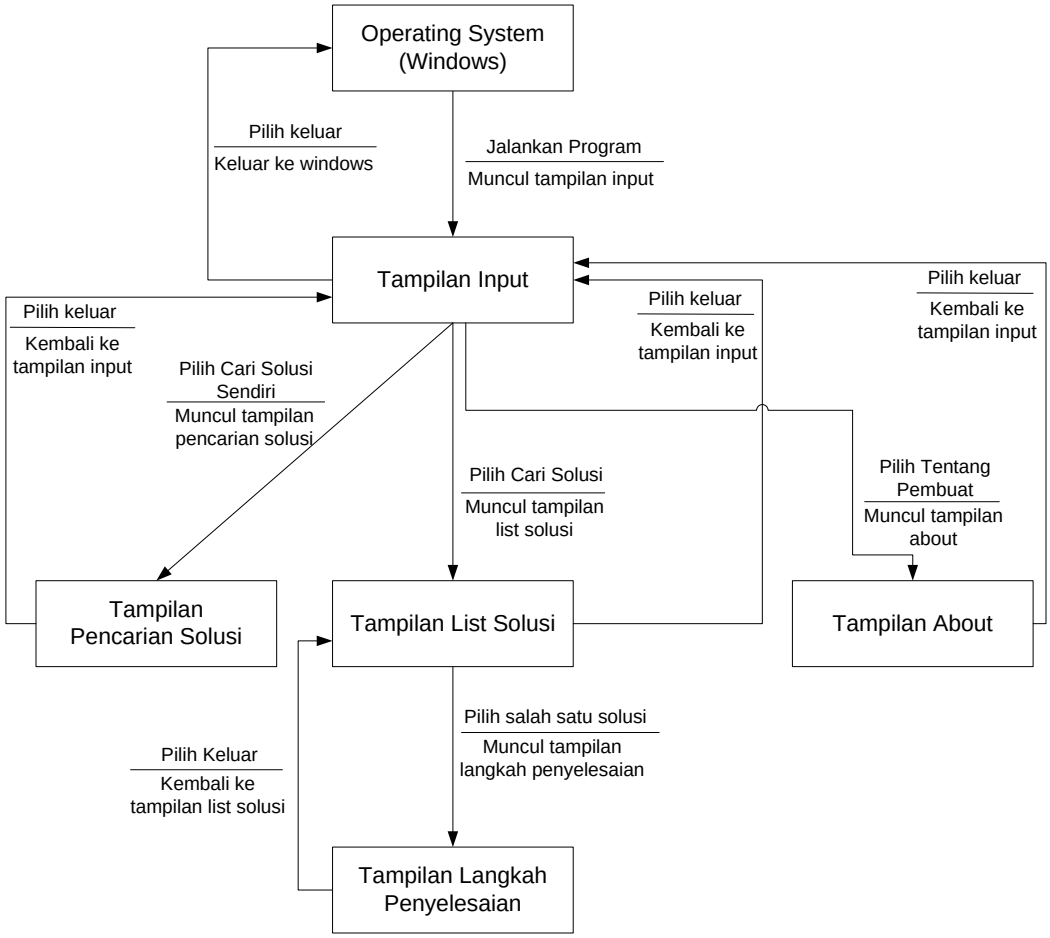
3.2 Analisis Perangkat Lunak

Perangkat lunak pencarian solusi permasalahan teko air memiliki lima tampilan, yaitu: tampilan *input*, tampilan *list* solusi, tampilan langkah penyelesaian, tampilan pencarian solusi dan tampilan *about*. Fungsi dari masing-masing tampilan adalah sebagai berikut:

1. Tampilan *input*, berfungsi untuk memasukkan besar kapasitas kendi1, kendi2 dan volume air yang diinginkan. Pada tampilan ini, *user* dapat memanggil tampilan *list* solusi, tampilan pencarian solusi dan tampilan *about*.

2. Tampilan *list* solusi, berfungsi untuk menampilkan solusi-solusi menuju *goal state* yang didapatkan dari pencarian. Apabila salah satu solusi dipilih maka tampilan langkah penyelesaian akan keluar untuk menunjukkan langkah-langkah penyelesaian dari solusi yang dipilih.
3. Tampilan langkah penyelesaian, berfungsi untuk menampilkan langkah-langkah penyelsaian dari solusi.
4. Tampilan pencarian solusi, berfungsi untuk mencari solusi secara manual.

Struktur rancangan *state transition diagram* dapat dilihat pada gambar 16 berikut.



Gambar 17 Struktur State Transition Diagram (STD)

3.3 Algoritma Pencarian Solusi

Algoritma ini melakukan pencarian solusi permasalahan teko air dengan menggunakan pohon pelacakan. Disebut pohon pelacakan, karena struktur pencarian solusi berbentuk sebuah pohon.

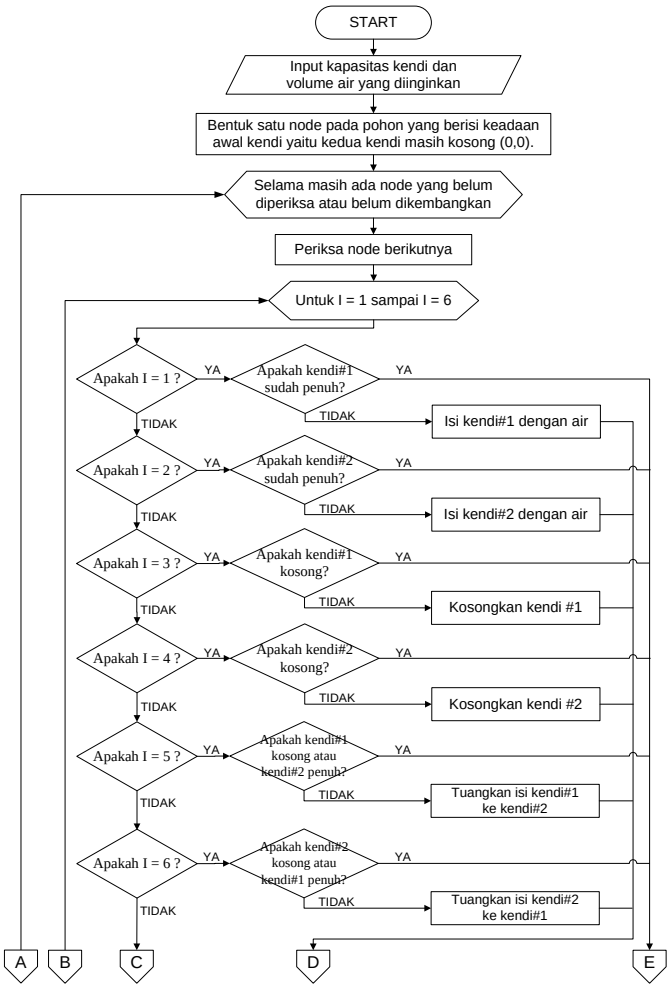
Pencarian solusi dimulai dari kondisi dimana kedua bejana kosong (*node* akar dari pohon pelacakan), dilanjutkan dengan mencari kemungkinan aksi – aksi yang dapat dilakukan pada kondisi tersebut. Ada 6(enam) aksi yang dapat dilakukan untuk setiap kondisi, yaitu:

1. Isi kendi-1 dengan air.
2. Isi kendi-2 dengan air.
3. Kosongkan kendi-1.
4. Kosongkan kendi-2.

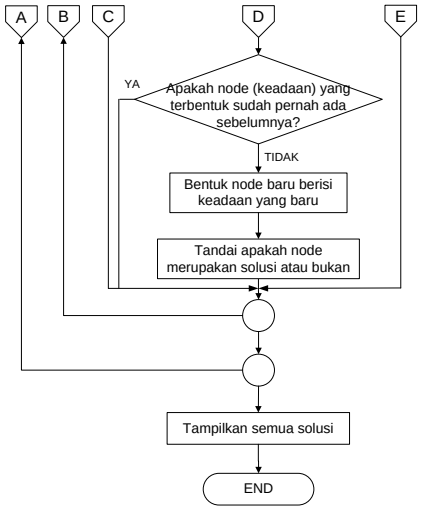
5. Tuang isi kendi-1 ke kendi-2.
6. Tuang isi kendi-2 ke kendi-1.

Kondisi baru yang terbentuk dari suatu aksi terhadap kondisi sebelumnya merupakan *node* anak dari kondisi tersebut. Apabila dihasilkan kondisi (*node*) baru yang sama dengan kondisi (*node*) yang sudah pernah ada sebelumnya, maka kondisi baru diabaikan dan tidak dicatat sebagai *node* dalam pohon. Pencarian dilakukan untuk semua node hingga semua *node* pada pohon diperiksa dan didapatkan solusi dari permasalahan teko air.

Algoritma pada perangkat lunak ini dapat digambarkan dalam bentuk *flowchart*, seperti terlihat pada gambar 18 dan 19 berikut.



Gambar 18 Flowchart Algoritma Pencarian Solusi-1



Gambar 19 Flowchart Algoritma Pencarian Solusi-2

Berikut merupakan algoritma pencarian solusi permasalahan teko air,

1. Setting variabel keadaan awal.
   a. Set properti variabel Langkah(1) sebagai berikut:
      i. Set .Kendi1 = 0,.Kendi2 = 0,.Atas = 0,.Aksi = 0.
      ii. Set .Keterangan = "Kondisi awal (kendi masih kosong)."
      iii. Set .Level = 1, .Cek = *True*.
2. Set nLangkah = 1.
3. Set Jalan = *True*.
4. Selama Jalan = True, maka lakukan prosedur berikut:
   a. Untuk nAksi = 1 Sampai 6, lakukan prosedur berikut:
      i. Jika nAksi = 1, maka set nK1 = MaxKendi1, nK2 = Langkah(nLangkah).Kendi2, strKet = Aksi(1).
      ii. Jika nAksi = 2, maka set nK1 = Langkah(nLangkah).Kendi1, nK2 = MaxKendi2, strKet = Aksi(2).
      iii. Jika nAksi = 3, maka set nK1 = 0, nK2 = Langkah(nLangkah).Kendi2, strKet = Aksi(3).
      iv. Jika nAksi = 4, maka set nK1 = Langkah(nLangkah).Kendi1, nK2 = 0, strKet = Aksi(4).
      v. Jika nAksi = 5, maka lakukan prosedur berikut:
         1) Jika Langkah(nLangkah).Kendi1 + Langkah(nLangkah). Kendi2 > MaxKendi2, maka set nK1 = Langkah(nLangkah).Kendi1 - (MaxKendi2 - Langkah(nLangkah).Kendi2), nK2 = MaxKendi2.
         2) Jika tidak, maka set nK1 = 0, nK2 = Langkah(nLangkah).Kendi1+Langkah(nLangkah).Kendi2.
         3) Set strKet = Aksi(5).
      vi. Jika nAksi = 6, maka lakukan prosedur berikut:
         1) Jika Langkah(nLangkah).Kendi1 + Langkah(nLangkah). Kendi2 > MaxKendi1, maka set nK1 = MaxKendi1, nK2 = Langkah(nLangkah).Kendi2 - (MaxKendi1 - Langkah(nLangkah).Kendi1).
         2) Jika tidak, maka set nK1 = Langkah(nLangkah).Kendi1+Langkah(nLangkah).Kendi2, nK2 = 0.
         3) Set strKet = Aksi(6).
   b. Jika Kondisi(nK1, nK2) belum pernah ada, maka bentuk *node* baru dengan melakukan prosedur berikut:
      i. Set Temp = Ubound(Langkah) + 1.

ii. Bentuk ulang *array* Langkah dengan dimensi sebesar Temp.
iii. Set *node* baru dengan properti sebagai berikut:
1) Set .Kendi1 = nK1,.Kendi2 = nK2,.nAnak = 0, .Atas = nLangkah.
2) Set .Keterangan = strKet
3) Set .Level = Langkah(nLangkah).Level + 1, .Cek = False.
iv. *Update array* induk dengan melakukan prosedur sebagai berikut:
1) Set .nAnak = .nAnak + 1.
2) Bentuk ulang dimensi array Langkah(nLangkah). Anak(nAnak).
3) Set .Anak(.nAnak) = Temp.
c. Set Langkah(nLangkah).Cek = *True*.
d. Cari node berikutnya dengan melakukan prosedur berikut:
i. Untuk Temp = 1 sampai batas atas array Langkah,
1) Jika Langkah(Temp).Cek = *False* maka set Jalan = *True*, nLangkah = Temp dan keluar dari *looping*.

5. Periksa pohon pelacakan yang terbentuk dan simpan setiap *node* atau kondisi yang memiliki jumlah volume air sama dengan volume air yang dicari dengan melakukan prosedur berikut:
a. Untuk Temp=1 sampai batas atas *array* Langkah, maka:
i. Jika Langkah(Temp).Kendi1 = nTarget atau Langkah(Temp).Kendi2 = nTarget maka bentuk ulang dimensi array Hasil(ubound(Hasil) + 1), Hasil(ubound(Hasil)).Index = Temp.
6. Simpan jalur dari semua node solusi ke keadaan awal. Untuk Temp = 1 sampai batas atas array Hasil, lakukan prosedur berikut:
a. Set bTarget = False, nAksi = Langkah(Hasil(Temp).Index).Atas.
b. Selama nAksi > 1 dan bTarget = False,
i. Jika Langkah(nAksi).Kendi1 = nTarget atau Langkah(nAksi).Kendi2 = nTarget maka set bTarget = *True*.
ii. Set nAksi = Langkah(nAksi).Atas.
c. Jika bTarget = *False*, maka tambahkan array pada HasilT dengan nilai Hasil(Temp).
7. Set Hasil = HasilT.
8. Balikkan jalur yang dihasilkan pada langkah ke-6. Untuk Temp=1 sampai Ubound(Hasil), lakukan prosedur berikut:
a. Set nAksi = Hasil(Temp).Index.
b. Selama nAksi<>0, maka:
i. Tambah array baru pada variabel HasilT dengan nilai nAksi.
ii. Set nAksi = Langkah (nAksi).Atas.
c. Untuk nAksi = batas *array* variabel HasilT sampai 1 dengan setiap pengurangan nilai 1, tambah *array* pada variabel .Langkah dan set hasilnya dengan HasilT(nAksi). Index.
9. Cari jalur terpendek (*shortest path*).
a. Set nShortest = 0, nLangkah = 9999.
b. Untuk Temp = 1 sampai batas *array* variabel Hasil,
i. Jika nLangkah > Langkah(Hasil(Temp).Index).Level maka set nShortest = Temp, nLangkah = Langkah(Hasil(Temp).Index).Level

3.4 Algoritma Menggambar Kendi dan Air dalam Kendi

Algoritma ini mengatur tampilan visualisasi dari kendi dan air di dalam kendi, sehingga tinggi kendi dan isi air yang ditampilkan memiliki ukuran yang proporsional sesuai dengan isi air dan kapasitas masing – masing kendi.

Algoritma menerima *input* berupa besar kapasitas kendi dan kondisi air pada setiap kendi. Tinggi kendi didapat dengan membandingkan kapasitas kendi-1 dengan kapasitas kendi-2. Kendi yang memiliki kapasitas terbesar memiliki tinggi yang paling maksimal pada daerah visualisasi, sedangkan tinggi kendi dengan kapasitas lebih kecil dihitung dengan rumus : (kapasitas kendi sedikit / kapasitas

kendi lebih besar) * tinggi maksimal pada daerah visualisasi. Tinggi air di dalam suatu kendi dapat dihitung dengan rumus (volume air / kapasitas kendi) * tinggi kendi.

Algoritma untuk menggambar kendi adalah sebagai berikut,

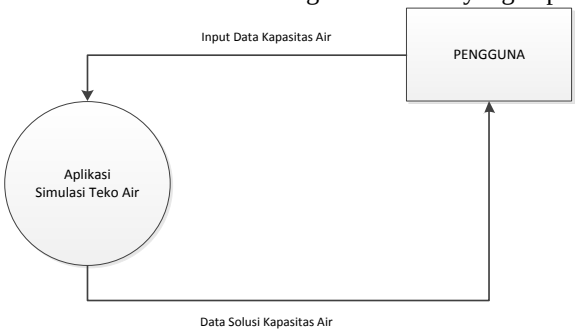
1. Ambil jumlah kapasitas kendi paling besar. Jika MaxKendi1 > MaxKendi2, maka set Max = MaxKendi1. Jika tidak maka set Max = MaxKendi2.
2. Gambar Kendi #1.
   a. Set Temp1 = Int(Val(MaxKendi1) * (3240 - 630) / Max).
   b. Set LineK1.Y1 = LineK1.Y2 - Temp1, LineK2.Y1 = LineK2.Y2 - Temp1.
   c. Set LineM1.Y1 = LineK1.Y1 – 5, LineM1.Y2 = LineM1.Y1 – 75, LineM2.Y1 = LineK2.Y1 – 5, LineM2.Y2 = LineM2.Y1 – 75.
3. Gambar Kendi #2.
   a. Set Temp1 = Int(Val(MaxKendi2) * (3240 - 630) / Max).
   b. Set LineK3.Y1 = LineK3.Y2 - Temp1, LineK4.Y1 = LineK4.Y2 - Temp1.
   c. Set LineM3.Y1 = LineK3.Y1 – 5, LineM3.Y2 = LineM2.Y1 – 75, LineM4.Y1 = LineK4.Y1 – 5, LineM4.Y2 = LineM4.Y1 – 75.

Algoritma untuk menggambar air di dalam kendi adalah sebagai berikut,

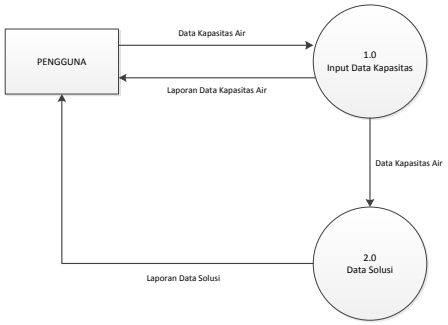
1. Gambar Isi Kendi#1.
   a. Set Temp1 = Int(Val(K1) * 2595 / Max) + 15.
   b. Set shpIsi1.Height = Temp1, shpIsi1.Top = LineB1.Y1 – shpIsi1.Height.
2. Gambar Isi Kendi#2.
   a. Set Temp1 = Int(Val(K2) * 2595 / Max) + 15.
   b. Set shpIsi2.Height = Temp1, shpIsi2.Top = LineB2.Y1 – shpIsi2.Height.

3.5 Model Proses

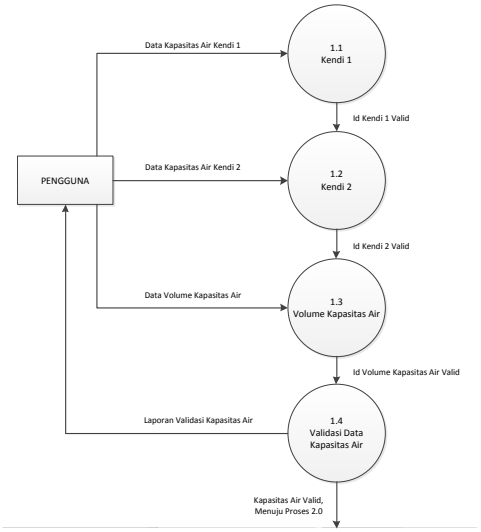
Untuk pemodelan proses sistem dibuat sebuah Diagram Konteks yang dapat dilihat pada Gambar



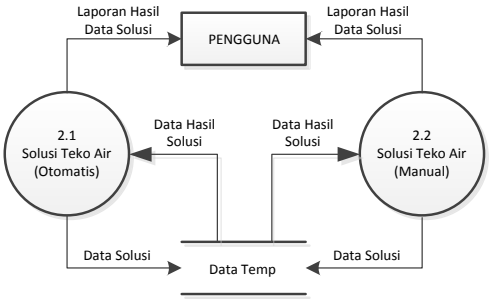
Gambar 20 Diagram Konteks



Gambar 21 DFD Level 1



Gambar 22 DFD Level 2 Proses 1.0



Gambar 23 DFD Level 2 Proses 2.0

Gambar 24 DFD Level 3 Proses 2.1

4. Hasil dan Pembahasan

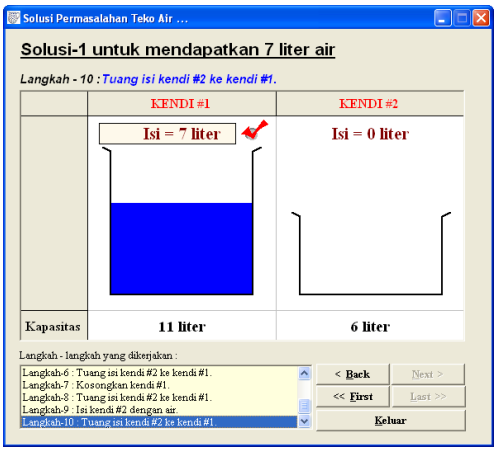
4.1 Implementasi

Sebagai contoh, *input* data sebagai berikut.

Contoh – 1 :

- Kapasitas kendi-1 : 11 liter
- Kapasitas kendi-2 : 6 liter
- Volume air yang diinginkan : 7 liter
- Solusi permasalahan teko air yang didapatkan :

Gambar 25 List Solusi permasalahan teko air dengan kapasitas kendi-1 = 11 liter, kapasitas kendi-2 = 6 liter dan volume air yang diinginkan = 7 liter



Gambar 26 Solusi permasalahan teko air dengan kapasitas kendi-1 =11 liter, kapasitas kendi-2 = 6 liter dan volume air yang diinginkan = 7 liter

Langkah – langkah yang dikerjakan pada solusi:

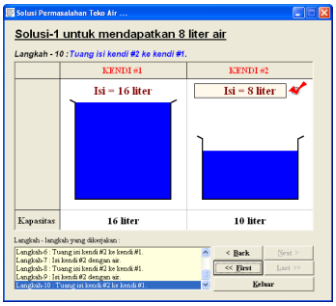
Langkah-0 : Kondisi Awal (Kendi kosong). *(Kendi-1 = 0 liter, Kendi-2 = 0 liter)*
Langkah-1 : Isi kendi #2 dengan air. *(Kendi-1 = 0 liter, Kendi-2 = 6 liter)*
Langkah-2 : Tuang isi kendi #2 ke kendi #1. *(Kendi-1 = 6 liter, Kendi-2 = 0 liter)*
Langkah-3 : Isi kendi #2 dengan air. *(Kendi-1 = 6 liter, Kendi-2 = 6 liter)*
Langkah-4 : Tuang isi kendi #2 ke kendi #1. *(Kendi-1 = 11 liter, Kendi-2 = 1 liter)*
Langkah-5 : Kosongkan kendi #1. *(Kendi-1 = 0 liter, Kendi-2 = 1 liter)*
Langkah-6 : Tuang isi kendi #2 ke kendi #1. *(Kendi-1 = 1 liter, Kendi-2 = 0 liter)*
Langkah-7 : Isi kendi #2 dengan air. *(Kendi-1 = 1 liter, Kendi-2 = 6 liter)*
Langkah-8 : Tuang isi kendi #2 ke kendi #1. *(Kendi-1 = 7 liter, Kendi-2 = 0 liter)*

Contoh – 2 :

- Kapasitas kendi-1 : 16 liter
- Kapasitas kendi-2 : 10 liter
- Volume air yang diinginkan : 8 liter
- Solusi permasalahan teko air yang didapatkan:

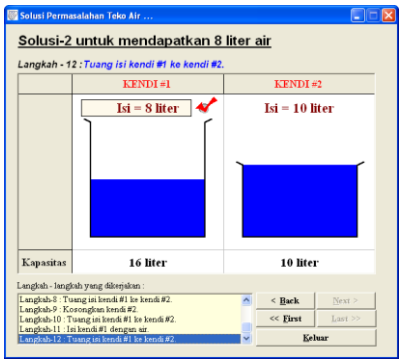
 ISSN: 2252-4517

Gambar 27 List Solusi permasalahan teko air dengan kapasitas kendi-1 =16 liter, kapasitas kendi-2 = 10 liter dan volume air yang diinginkan = 8 liter



Gambar 28 Solusi-1 permasalahan teko air dengan kapasitas kendi-1 =16 liter, kapasitas kendi-2 = 10 liter dan volume air yang diinginkan = 8 liter

Langkah – langkah yang dikerjakan pada solusi-1:
Langkah-0 : Kondisi Awal (Kendi kosong). *(Kendi-1 = 0 liter, Kendi-2 = 0 liter)*
Langkah-1 : Isi kendi #2 dengan air. *(Kendi-1 = 0 liter, Kendi-2 = 10 liter)*
Langkah-2 : Tuang isi kendi #2 ke kendi #1. *(Kendi-1 = 10 liter, Kendi-2 = 0 liter)*
Langkah-3 : Isi kendi #2 dengan air. *(Kendi-1 = 10 liter, Kendi-2 = 10 liter)*
Langkah-4 : Tuang isi kendi #2 ke kendi #1. *(Kendi-1 = 16 liter, Kendi-2 = 4 liter)*
Langkah-5 : Kosongkan kendi #1. *(Kendi-1 = 0 liter, Kendi-2 = 4 liter)*
Langkah-6 : Tuang isi kendi #2 ke kendi #1. *(Kendi-1 = 4 liter, Kendi-2 = 0 liter)*
Langkah-7 : Isi kendi #2 dengan air. *(Kendi-1 = 4 liter, Kendi-2 = 10 liter)*
Langkah-8 : Tuang isi kendi #2 ke kendi #1. *(Kendi-1 = 14 liter, Kendi-2 = 0 liter)*
Langkah-9 : Isi kendi #2 dengan air. *(Kendi-1 = 14 liter, Kendi-2 = 10 liter)*
Langkah-10 : Tuang isi kendi #2 ke kendi #1. *(Kendi-1 = 16 liter, Kendi-2 = 8 liter)*



Gambar 29 Solusi-2 permasalahan teko air dengan kapasitas kendi-1 =16 liter, kapasitas kendi-2 = 10 liter dan volume air yang diinginkan = 8 liter

Langkah – langkah yang dikerjakan pada solusi-2:
Langkah-0 : Kondisi Awal (Kendi kosong). *(Kendi-1 = 0 liter, Kendi-2 = 0 liter)*
Langkah-1 : Isi kendi #1 dengan air. *(Kendi-1 = 16 liter, Kendi-2 = 0 liter)*
Langkah-2 : Tuang isi kendi #1 ke kendi #2. *(Kendi-1 = 6 liter, Kendi-2 = 10 liter)*
Langkah-3 : Kosongkan kendi #2. *(Kendi-1 = 6 liter, Kendi-2 = 0 liter)*
Langkah-4 : Tuang isi kendi #1 ke kendi #2. *(Kendi-1 = 0 liter, Kendi-2 = 6 liter)*
Langkah-5 : Isi kendi #1 dengan air.. *(Kendi-1 = 16 liter, Kendi-2 = 6 liter)*
Langkah-6 : Tuang isi kendi #1 ke kendi #2. *(Kendi-1 = 12 liter, Kendi-2 = 10 liter)*
Langkah-7 : Kosongkan kendi #2. *(Kendi-1 = 12 liter, Kendi-2 = 0 liter)*
Langkah-8 : Tuang isi kendi #1 ke kendi #2. *(Kendi-1 = 2 liter, Kendi-2 = 10 liter)*
Langkah-9 : Kosongkan kendi #2. *(Kendi-1 = 2 liter, Kendi-2 = 0 liter)*
Langkah-10 : Tuang isi kendi #1 ke kendi #2. *(Kendi-1 = 0 liter, Kendi-2 = 2 liter)*
Langkah-11 : Isi kendi #1 dengan air. *(Kendi-1 = 16 liter, Kendi-2 = 2 liter)*
Langkah-12 : Tuang isi kendi #1 ke kendi #2. *(Kendi-1 = 8 liter, Kendi-2 = 10 liter)*

5. Simpulan

Setelah menyelesaikan perangkat lunak pencarian solusi permasalahan teko air, dapat ditarik kesimpulan sebagai berikut,

1. Perangkat lunak ini memiliki kemampuan untuk mencari solusi permasalahan teko air dengan 2 buah kendi dan menampilkan semua langkah – langkah yang dapat diambil untuk mendapatkan solusi atau volume air yang diinginkan.
2. Perangkat lunak dapat mencari solusi terpendek dari permasalahan teko air karena menggunakan metode pencarian melebar pertama (*breadth-first search*).
3. Pencarian tidak selalu menghasilkan solusi walaupun setelah semua *node* pada pohon pelacakan diperiksa dan dikembangkan.

Pustaka

Arhami Muhammad, *Artificial Intelligence* (Kecerdasan Buatan), Penerbit Andi Yogyakarta, 2006.

Leman, Metodologi Pengembangan Sistem Informasi, Penerbit PT Elex Media Komputindo, Jakarta 1998

Munir Rinaldi, Algoritma dan Pemograman, Penerbit Informatika Bandung, 2007

Pramono Djoko, Mudah menguasai *Visual Basic 6*, PT. Elex Media Komputindo, 2002.

Suryokusumo Ario, *Microsoft Visual Basic 6.0*, PT. Elex Media Komputindo, 2001.

Suyatno, *Artificial Intelligence*, Penerbit Informatika Bandung, 2007