APLIKASI PENYELESAIAN *BUILDING BLOCK PROBLEM* DENGAN MENGGUNAKAN ALGORITMA BFS (*BREADTH FIRST SEARCH*)

M. Irwan Ukkas[1)], Eka Arriyanti[2)], Tandy William[3)]

[1,3] Sistem Informasi, STMIK Widya Cipta Dharma
[2] Teknik Informatika, STMIK Widya Cipta Dharma
[1,2,3]Jl. Prof. M. Yamin No.25, Samarinda, 75123
E-mail : Irwan212@yahoo.com [1)], ivaneka@gmail.com [2)], mink.mink91@gmail.com [3)]

ABSTRAK

Aplikasi Algoritma BFS (*Breadth First Search*) dalam *Building Block Problem* merupakan aplikasi yang dibuat untuk mencari solusi dari permainan *Building Block* dengan menggunakan bantuan algoritma BFS. Permainan *Building Block* adalah permainan penyusunan balok dimana terdapat minimal 3 tumpukan dengan beberapa balok pada kondisi awal dapat di susun sedemikian sehingga menjadi tumpukan balok pada kondisi akhir yang di inginkan. Tujuan dari penelitian ini adalah untuk menghasilkan aplikasi algoritma BFS untuk mencari solusi dari permainan *building block* yang dimainkan dengan menggunakan bahasa pemrograman Visual Basic 6.0. Dalam penelitian ini, teknik pengumpulan data yang digunakan adalah studi pustaka dan dokumentasi, analisis data, analisis kebutuhan, dan analisis teknologi. Metode analisis dan design sistem yang digunakan dalam penelitian ini adalah *Flowchart* (Bagan Alir). Hasil dari penelitian ini adalah dibuatnya aplikasi BFS (*Breadth First Search*) untuk mencari solusi dari permainan *building block*. Pengguna dapat menginputkan keadaan awal dan keadaan akhir dari *building block*, kemudian aplikasi akan mencari solusi dengan algoritma BFS. Setelah solusi ditemukan, maka aplikasi akan menampilkan solusi tersebut langkah demi langkah. Aplikasi juga mencatat waktu pencarian solusi.

Kata Kunci : *Building Block Problem*, BFS

1. PENDAHULUAN

Dalam kehidupan sehari-hari banyak ditemui berbagai macam permainan yang mengandalkan logika. Salah satunya adalah permainan penyusunan balok atau yang lebih sering disebut *building block problem*. Permainan penyusunan balok memiliki keunikan tersendiri sehingga menarik untuk dimainkan dan sekaligus dapat digunakan sebagai media untuk melatih kecerdasan dan ketelitian.

Pada permainan penyusunan balok, pemain dapat menyusun balok yang ada pada kondisi awal sehingga menjadi balok pada kondisi akhir yang diinginkan. Permainan ini dapat diilustrasikan seperti berikut, terdapat 2 atau lebih kotak yang disusun menjadi 2 atau 3 tumpukan balok sebagai kondisi awal. Sasaran dari permainan ini adalah mendapatkan suatu tumpukan kotak sesuai dengan keinginan. Operasi yang diperbolehkan dalam proses penyelesaiannya adalah turunkan(x), yang berarti bahwa kotak x diturunkan dari suatu tumpukan tertentu dan letakkan (x, y), yang berarti bahwa kotak x diletakkan diatas kotak y, dengan persyaratan bahwa kotak y harus berada di urutan paling atas dari suatu tumpukan.

Banyak metode pencarian dalam kecerdasan buatan (*Artificial Intelegent*) atau (AI) yang dapat diterapkan untuk menyelesaikan problema tersebut. Salah satunya adalah dengan menggunakan bantuan pohon pelacakan. Kondisi-kondisi (*state-state*) yang mungkin digambarkan dalam suatu pohon pencarian dimulai dari mendeklarasikan kondisi awal (*start state*) sebagai akar dari pohon biner. Proses dilanjutkan dengan menggambarkan *state* berikutnya dari *state* tersebut hingga didapatkan kondisi yang diinginkan (*goal state*).

Dari latar belakang masalah tersebut, maka diharapkan aplikasi yang dibangun dengan menggunakan metode BFS (*Breadth First Search*) dapat membantu dalam mencari solusi terpendek dalam menyelesaikan permainan penyusunan balok dengan tepat dan waktu yang singkat.

2. RUANG LINGKUP PENELITIAN

Dalam penelitian ini permasalahan mencangkup:

1. ”Bagaimanakah membuat sebuah Aplikasi Penyelesaian *Building Block Problem* dengan menggunakan Algoritma BFS (*Breadth First Search*)?”.
2. Batasan Masalah
   a. Jumlah balok yaitu $3 \leq$ balok ≤ 10.
   b. Susunan tumpukan awal dan akhir di tentukan oleh user.
   c. Keadaan awal tumpukan dan keadaan akhir tumpukan pada satu susunan harus berbeda, jika tidak maka logika sistem menganggap tidak ada perpindahan.
   d. Waktu delay perpindahan dapat di atur oleh user. Dan total waktu yang digunakan untuk pencarian dapat di ketahui.

3. Tujuan Penelitian
   a. Menerapkan Algoritma BFS (*Breadth First Search*) dalam penyusunan balok (*building block problem*) sehingga dapat memberikan solusi terpendek penyelesaian penyusunan balok dengan tepat dan cepat.
   b. Sebagai bahan referensi bagi mahasiswa dan masyarakat umum mengenai pembuatan aplikasi algoritma BFS (*Breadth First Search*) dalam penyelesaian penyusunan balok. Dan dapat membantu mahasiswa dalam mengembangkan penyelesaian permainan dengan menggunakan metode-metode yang ada pada kecerdasan buatan.
   c. Aplikasi ini dapat digunakan sebagai fasilitas pendukung dalam proses belajar mengajar, terutama dalam mata kuliah Kecerdasan Buatan (*Artificial Intelligence*). Dan dalam studi kasus permainan building block problem dapat membantu mencari solusi terpendek.

3. BAHAN DAN METODE

Adapun bahan dan metode dalam metode aplikasi ini, yaitu :

3.1 Aplikasi

Menurut Dhanta (2009), Aplikasi (*application*) adalah *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya *Microsoft Word*, *Microsoft Excel*.

Menurut Jogiyanto (2008), Perangakat lunak aplikasi adalah program yang ditulis dan diterjemahkan oleh linguage software untuk menyelesaikan suatu aplikasi tertentu.

Menurut Febrian (2007), Aplikasi merupakan program siap pakai yang digunakan manusia dalam melakukan pekerjaan menggunakan komputer.

3.2 Algoritma

Menurut Kristanto (2010), Algoritma adalah suatu kumpulan terhingga (*finite set*) dari instruksi yang terdefinisi dengan baik (*well-defined instructions*) untuk menyelesaikan beberapa pekerjaan dimana diberikan *state* awal (*initial state*) dan akan dihentikan pada saat ditemukan *state* akhir (*end-state*) yang dikenal. Algoritma dapat diimplementasikan dalam pembuatan program komputer. Kesalahan dalam merancang algoritma untuk menyelesaikan suatu problema dapat menyebabkan program gagal dalam implementasinya.

Algoritma sering memiliki beberapa langkah perulangan (iterasi) atau memerlukan pengambilan keputusan seperti logika (*logic*) atau perbandingan (*comparison*) sampai pekerjaan diselesaikan. Menerapkan suatu algoritma secara benar belum tentu dapat menyelesaikan problema. Hal ini dikarenakan adanya kemungkinan algoritma tersebut rusak atau cacat, atau penerapannya tidak cocok (tidak tepat) untuk menyelesaikan problema.

Algoritma adalah hal yang mendasar untuk komputer dalam memproses informasi, karena sebuah program komputer adalah sebuah algoritma yang memberitahukan kepada komputer langkah-langkah spesifik yang akan dijalankan (dalam urutan spesifik) untuk melakukan pekerjaan tertentu, misalnya menghitung gaji karyawan atau mencetak rapor murid.

Suatu algoritma memiliki kemungkinan terbaik (*best case*) dan kemungkinan terburuk (*worst case*). *Best Case* maksudnya adalah waktu eksekusi tercepat dari algoritma, sedangkan *Worst Case* maksudnya adalah waktu eksekusi terlama dari algoritma. Waktu eksekusi dari algoritma ini biasanya disebut dengan kompleksitas algoritma.

3.3 Sejarah Kecerdasan Buatan

Menurut Kristanto (2010), secara singkat sejarah kecerdasan buatan dapat dijelaskan sebagai berikut:

1. Penelitian sebelum tahun 1950

   Pertama, ada pekerjaan ahli logika seperti Alonzo Church, Kurt Godel, Emil Past, dan Alan Turing. Mereka mengadakan kerja yang pertama oleh Whitehead dan Rusell, Tarski, dan Kleene. Pekerjaan ini dimulai dengan sungguh-sungguh sejak tahun 1920 dan tahun 1930. Kerja ini menolong memproduksi metode menyusun pemikiran, bentuk logis ini dikenal sebagai proporsional dan kalkulus predikat. Ini mendemonstrasikan bahwa fakta dan ide dari bahasa dapat secara formal dideskripsikan dan dimanipulasi secara mekanik dengan cara yang tepat. Turing yang kadang-kadang dianggap sebagai bapak kecerdasan buatan, juga mendemonstrasikan pada awal tahun 1936, bahwa *Simple Computer Processor* (nama terakir mesin Turing) dapat memanipulasi simbol-simbol seperti angka.

   Kedua, bidang baru sibernetika, sebuah nama yang diciptakan oleh Nobert Wiener diberikan bersamaan dengan banyaknya kesejajaran antara manusia dengan mesin. Sibernetika, pendidikan komunikasi manusia dan mesin, menjadi area *research* sejak tahun 1940 dan tahun 1950. Sibernetika mengkombinasikan konsep dari informasi teori, sistem kontrol pengaruh arus balik (baik secara biologis dan mesin) dan komputer elektronik.

   Ketiga, datang perkembangan baru yang dibuat dalam *grammar* yang formal. Pekerjaan ini adalah hasil pertumbuhan logis sejak awal tahun 1900. hal ini membantu menyediakan permulaan baru bagi teori bahasa dan bidang bahasa pada umumnya.

   Akhirnya, sejak tahun 1950, program komputer digital elektronik menjadi realita yang komersil.

2. Penelitian setelah tahun 1950

   Sejak tahun 1950 beberapa kejadian telah terjadi, yang ditandai dengan mulainya kecerdasan buatan. Ini adalah periode yang terkenal dengan program permainan catur yang dikembangkan oleh para peneliti seperti Claude Shannon di MIT dan Allen Newell di RAND corporation. Tipe permainan lainnya dan program-program simulasi juga dikembangkan sejak dini.

   Banyak usaha dikembangkan dengan program penerjemahan, dan ada banyak harapan bagi kesuksesan penerjemahan bahasa menggunakan komputer. Dirasakan bahwa penyimpanan sejumlah kamus dalam komputer secara mendasar semuanya diperlukan untuk memproduksi terjemahan yang

tepat dari satu bahasa ke bahasa lainnya, meskipun pendekatan ini terlalu sederhana, ini memakan waktu beberapa tahun sebelum usaha-usaha digagalkan.

Pertengahan tahun 1950, telah diakui secara resmi sebagai lahirnya kecerdasan buatan, disponsori oleh IBM yang diadakan di Universitas Dartmouth. Seminar pada bulan Juni 1956 dihadiri beberapa pionir muda dalam kecerdasan buatan. Sebagian besar diskusi pekerjaan yang melibatkan mereka sejak periode ini dinamakan percobaan Dalil Otomatis dan *Programming* bahasa. Antara tahun 1956-1957, teori logis, salah satu dari program pertama untuk percobaan dalil otomatis. Sebagai bagian dari perkembangan tersebut, daftar proses bahasa yang pertama disebut IPL (*Information Processing Language*) yang juga dilengkapi.

3. Tahun 1961-1965
   A.L. Samuel mengembangkan program yang mempelajari memainkan checkers pada level master.
4. Tahun 1965
   J.A. Robinson memperkenalkan resolusi sebagai metode kesimpulan dalam logika
5. Tahun 1968
   Membuat MACSYMA diprakarsai oleh Carl Engleman, William Martin, dan Joel Moses. MACSYMA adalah pekerjaan lanjutan SIN yaitu sebuah program pemecahan integrasi indefinite.

3.4 Pengertian Kecerdasan Buatan

Menurut Kristanto (2010), kecerdasan buatan merupakan bagian dari ilmu pengetahuan komputer yang khusus ditunjukan dalam perancangan otomatisasi tingkah laku cerdas dalam sistem kecerdasan komputer. Sistem memperlihatkan sifat-sifat khas yang dihubungkan dengan kecerdasan dalam kelakuan atau tindak-tanduk yang sepenuhnya bisa menirukan beberapa fungsi otak manusia, seperti pengertian bahasa, pengetahuan, pemikiran, pemecahan masalah dan lain sebagainya.

Ada beberapa definisi kecerdasan buatan, yaitu:

1. Kecerdasan buatan adalah cabang ilmu komputer yang berhubungan dengan studi dan kreasi sistem komputer yang mempertunjukkan beberapa bentuk kecerdasan.
2. Sistem yang mempelajari konsep-konsep baru dan tugas-tugas.
3. Sistem yang dapat berpikir dan menarik kesimpulan yang berguna bagi dunia sekitar kita.
4. Sistem yang dapat mengerti bahasa dan memahami pemandangan visual.

Sistem yang melakukan tipe-tipe yang lain seperti prestasi yang membutuhkan kecerdasan manusia.

3.5 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.

Algoritma *Breadth First Search* adalah sebagai berikut:

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

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. Lebih jauh karena mula-mula semua keadaan ditemukan melalui lintasan terpendek sehingga setiap keadaan yang ditemui pada kali kedua didapati pada sepanjang sebuah lintasan yang sama atau lebih panjang. Kemudian, jika tidak ada kesempatan ditemukannya keadaan yang identik pada sepanjang lintasan yang lebih baik maka algoritma akan menghapusnya, sehingga 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).

1. Sebagai contoh kasus penerapan algoritma BFS adalah pada penyelesaian permasalahan penyusunan balok (*Building Block Problem*). Problema ini dapat diilustrasikan seperti berikut, terdapat 3 atau lebih kotak yang disusun menjadi 3 tumpukan balok sebagai kondisi awal (*initial state*).

Sasaran (*goal*) dari problema ini adalah mendapatkan suatu tumpukan kotak sesuai dengan keinginan.

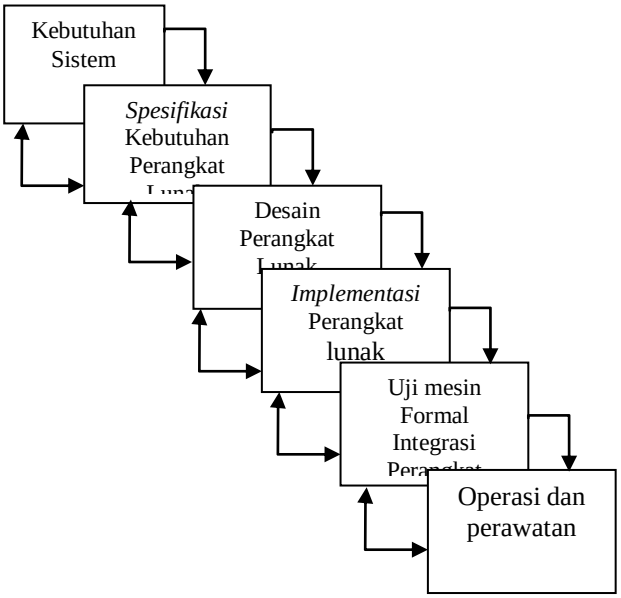
3.6. Masalah Ruang Keadaan

Pada aplikasi penyusunan balok ini menggunakan tiga buah tumpukan. Adapun aturan dan operator permainan penyusunan balok adalah sebagai berikut:

1. Tentukan keadaan awal (*initial state*) pada tumpukan balok.
2. Tentukan keadaan tujuan (*goal state*) pada tumpukan balok.
3. Operator perpindahan balok adalah:
   a. Pindahkan balok pada tumpukan 1 ke tumpukan 2
   b. Pindahkan balok pada tumpukan 1 ke tumpukan 3
   c. Pindahkan balok pada tumpukan 2 ke tumpukan 1
   d. Pindahkan balok pada tumpukan 2 ke tumpukan 3
   e. Pindahkan balok pada tumpukan 3 ke tumpukan 1
   f. Pindahkan balok pada tumpukan 3 ke tumpukan 2
4. Aturan perpindahan balok adalah:
   a. Hanya boleh satu balok yang dipindah
   b. Posisi balok yang dapat di pindah adalah posisi pada tumpukan paling atas
   c. Balok yang dipindah harus diletakkan di posisi paling atas pada tumpukan tujuan
   d. Setiap balok hanya dapat dipindahkan ke tumpukan lain yang sudah ada tanpa mengubah tumpukan awal yang ada pada tumpukan tujuan.

3.7 Model Air Terjun (*Waterfall*)

Menurut Simarmata (2010), Model Air Terjun (*Waterfall*) adalah untuk membantu mengatasi kerumitan yang terjadi akibat proyek-proyek pengembangan perangkat lunak. Sebuah model Air Terjun memacu tim pengembang untuk memerinci apa yang seharusnya perangkat lunak lakukan sebelum sistem tersebut dikembangkan.



Gambar 1. Model Pengembangan Air terjun

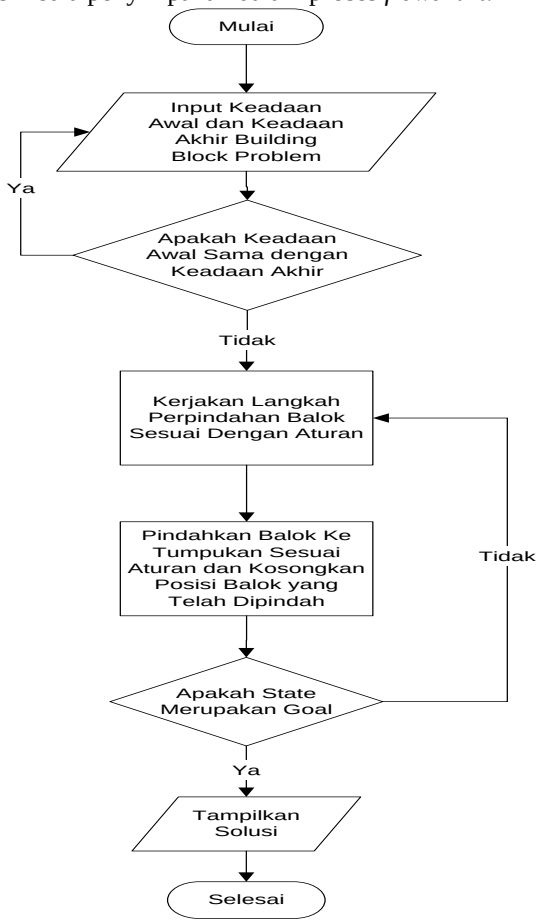
Kemudian model ini memungkinkan pemecahan misi pengembangan yang rumit menjadi beberapa langkah logis (desain, kode, pengujian, dan seterusnya) dengan beberapa langkah yang pada akhirnya akan menjadi produk akhir yang siap pakai.

Kelebihan model air terjun (*waterfall*) diantaranya :

1. Mudah diaplikasikan
2. Memberikan *template* tentang metode analisis, desain, pengkodean, pengujian, dan pemeliharaan.

4. RANCANGAN APLIKASI/ALGORITMA

Pada bagian ini memperlihatkan urutan proses dalam sistem yang menunjukkan alat media *input*, *output* serta jenis media penyimpanan dalam proses *flowchart.*

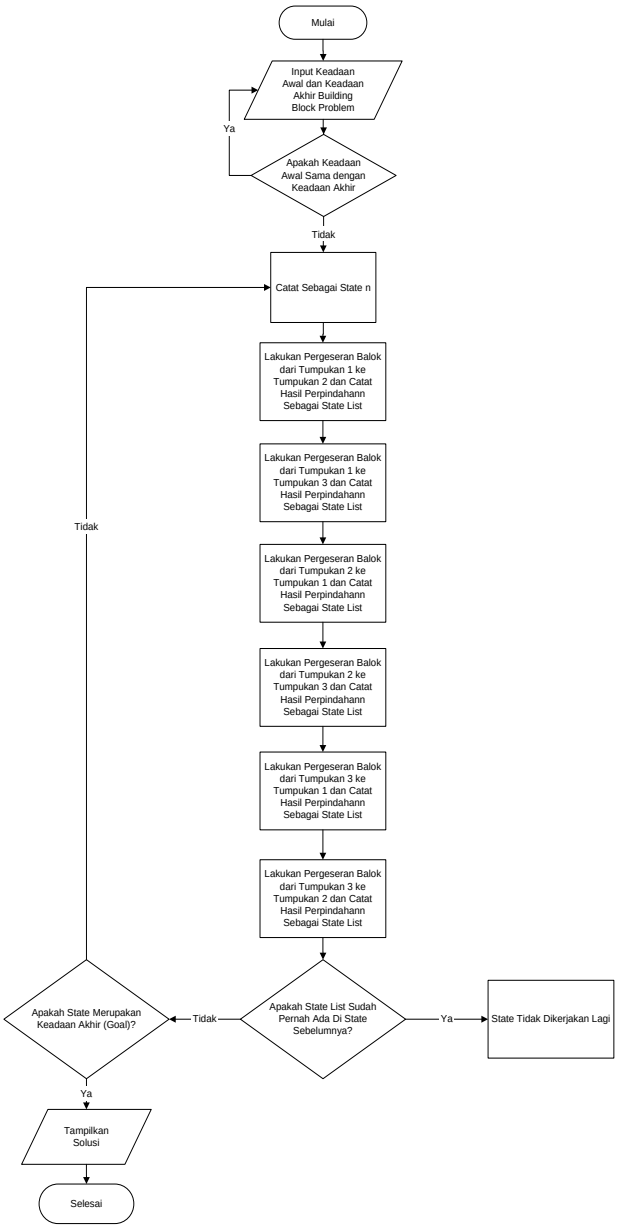


Gambar 2. *Flowchart Aplikasi*

Pada gambar 2 menjelaskan proses jalannya program. Dimulai dari *input* keadaan awal dan akhir lalu dilanjutkan pada proses penentuan dengan melihat keadaan awal dan akhir, kemudian akan menegerjakan langkah perpindahan balok sesuai dengan aturan dan jika sudah mencapai goal yang diinginkan makan akan menapilkan solusi.

Flowchart Proses Algoritma

Berikut adalah flowchart proses algoritma pada program ini :



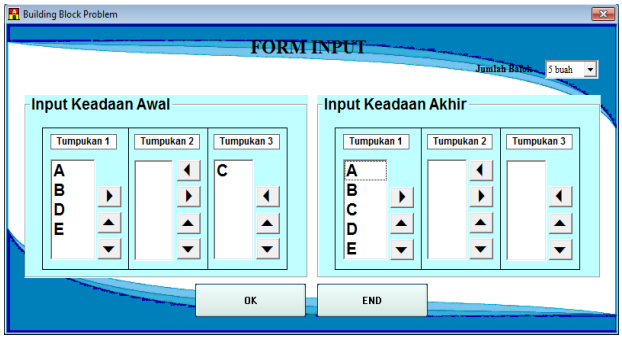
Gambar 3. *Flowchart* Proses Algoritma

5. IMPLEMENTASI

Berikut ini adalah beberapa tampilan form pada program.

5.1. Tampilan *Form Input* Keadaan Awal Dan Keadaan akhir

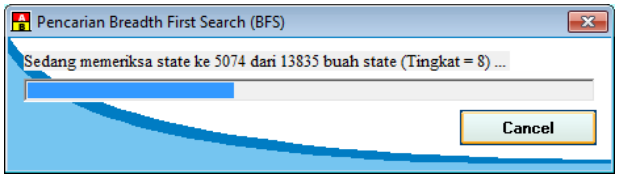
Pada form input keadaan awal dan keadaan akhir *Building Block Problem*, *user* dapat menentukan keadaan awal dan keadaan akhir *Building Block Problem* secara manual. Keadaan awal dan keadaan akhir tidak boleh sama. Jika keadaan awal dan akhir sama maka saat menekan tombol OK, akan diberikan peringatan. Kemudian jika keadaan awal dan akhir berbeda maka aplikasi akan menampilkan *form* pencarian solusi yang artinya pelacakan untuk mencari solusi terpendek sedang dikerjakan.



Gambar 3 Tampilan *Form Input* Keadaan Awal Dan Keadaan Akhir

5.2. Tampilan *Form* Pencarian Solusi

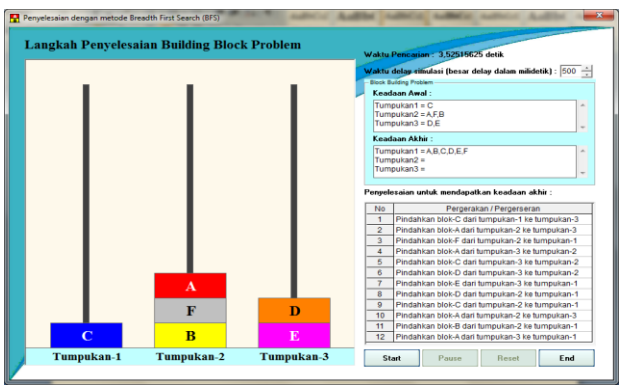
Form pencarian solusi digunakan untuk mencari solusi terpendek dengan menggunakan metode BFS (*Breadth First Search*). Pencarian solusi dilakukan berdasarkan input keadaan awal dan keadaan akhir dari user. Pada *form* pencarian solusi, *user* dapat mengetahui pada tingkat berapa pencarian yang sedang berlangsung. *User* juga dapat membatalkan pencarian dengan menekan tombol Cancel, maka pencarian akan dihentikan dan kembali pada *form* input keadaan awal dan keadaan akhir. Jika solusi ditemukan maka form solusi akan ditampilkan.



Gambar 4 Tampilan *Form* Pencarian Solusi

5.3. Tampilan *Form* Solusi

Form solusi *Building Block Problem* pada tiga tumpukan digunakan untuk menampilkan solusi dari hasil pencarian yang dilakukan oleh aplikasi, menampilkan waktu pencarian, dan panjang langkah solusi. Pada *form* solusi, *user* dapat memulai simulasi solusi dengan menekan tombol Start, simulasi dapat dihentikan dengan menekan tombol Pause, dan dapat diulangi dengan menekan tombol Reset. Kecepatan simulasi dapat ditentukan dengan memilih besarnya delay pergerakan yang telah tersedia. Langkah-langkah solusi juga ditampilkan pada tabel sehingga lebih memudahkan *user*.



Gambar 5 Tampilan *Form Solusi*

6. KESIMPULAN

Dengan adanya hasil penelitian yang dilaksanakan, maka peneliti menarik kesimpulan berdasarkan dari uraian yang telah dibahas pada bab-bab sebelumnya, yaitu:

1. Dengan adanya aplikasi ini, maka dapat membantu dalam pembelajaran pada mata kuliah Kecerdasan Buatan, terutama penggunaan metode BFS (*Breadth First Search*) dalam mencari solusi terpendek dari penyelesaian permainan *building block problem*.
2. Metode BFS (*Breadth First Search*) menjamin tidak menemui jalan buntu dalam proses pencarian solusi.
3. Solusi yang ditemukan dengan menggunakan metode BFS (*Breadth First Search*) adalah solusi terpendek (*shortest path*), sehingga permainan *building block problem* dapat diselesaikan dengan cepat.
4. Kelemahan dari metode BFS (*Breadth First Search*) adalah membutuhkan waktu pencarian yang cukup lama karena akan menguji semua *node* pada level ke-*n* untuk mendapatkan solusi pada level ke- (n + 1).
5. Semakin banyak jumlah balok atau semakin acak posisi awal terhadap posisi akhir pada *building block problem*, maka semakin lama waktu yang dibutuhkan untuk proses pencarian solusi.

7. SARAN

Adapun saran-saran yang dapat dikemukakan yaitu sebagai berikut:

1. Aplikasi dapat dikembangkan dengan menambahkan jumlah tumpukan sesuai dengan keinginan *user*, dengan kata lain jumlah tumpukan dapat di pilih oleh *user* saat akan *input* keadaan awal.
2. Aplikasi ini dapat dikembangkan dengan menambahkan metode pencarian lainnya, seperti: metode pencarian *Heuristic*, *Depth-First-Search*, *Hill-Climbing*, dan metode lainnya.
3. Aplikasi dapat dikembangkan dengan menambahkan penggambaran dan penjelasan pada pohon pelacakan yang digunakan untuk mencari solusi.
4. Aplikasi dapat lebih menarik jika dapat dijalankan pada mobile atau dengan menggunakan animasi.

8. DAFTAR PUSTAKA

Anita Desiani dan Muhammad Arhami, 2005, *Konsep Kecerdasan Buatan*. Yogyakarta : Andi.

Dhanta, Rizky, 2009. *Pengantar Ilmu Komputer*, Surabaya : Indah.

Febrian, Jack, 2007. *Kamus Komputer dan teknologi Informasi*, Bandung: Informatika.

Jogiyanto, HM, 2005, *Analisis & Desain Sistem Informasi*, Yogyakarta : Andi Offset.

Kristanto, Andi, 2010. *Kecerdasan Buatan*. Yogyakarta : Graha Ilmu.

Simarmata, 2010, *Rekayasa Perangkat Lunak*. Yogyakarta : Penerbit Andi.