Implementasi Algoritma Breadth First Search (BFS) Pada Masalah Penyusunan Blok

Jufri
STMIK Dipanegara Makassar
Jl. P.Kemerdekaan KM. 9, Makassar, HP: +628124211274
Jufri.ldp@gmail.com

Abstrak

Penyusunan blok bisa juga dikatakan sebagai Blocks World Architecture, yaitu sebuah permainan yang mengajak pemainnya untuk berfikir secara logika dalam menyelesaikan permainan ini, misalnya pemain diharuskan untuk menyusun blok angka atau huruf sampai terurut berdasarkan susunan yang telah diacak sebelumnya. Permainan ini bisa dikategorikan sebagai permainan puzzle, yang mana pada keadaan awal blok huruf disusun secara acak untuk beberapa huruf misalnya 6 huruf (A,F,C,E,B,D) pada tiga tempat, kemudian blok huruf tersebut harus disusun kembali dengan menggunakan tiga tempat yang ada sampai didapatkan susunan blok huruf yang tersusun dari huruf A sampai F pada sebuah tempat.
Untuk menyelesaikan masalah penyusunan blok diperlukan suatu metode atau algoritma pencarian solusi. Salah satunya algoritma pencarian yang dapat digunakan adalah algoritma Breadth First Search, karena algoritma ini dapat melakukan pencarian solusi dari keadaan awal yang disediakan berdasarkan parameter-parameter yang ditentukan dari masalah penyusunan blok.

Kata kunci :Block Word, Breadth First Search.

1. Pendahuluan

Penyusunan blok bisa juga dikatakan sebagai *Blocks World Architecture*, yaitu sebuah permainan yang mengajak pemainnya untuk berfikir secara logika dalam menyelesaikan permainan ini, misalnya pemain diharuskan untuk menyusun blok angka atau huruf sampai terurut berdasarkan susunan yang telah diacak sebelumnya. Permainan ini bisa dikategorikan sebagai permainan puzzle, yang mana pada keadaan awal blok angka disusun secara acak untuk beberapa angka misalnya 6 angka (2,7,3,5,6,4) pada tiga tempat, kemudian blok angka tersebut harus disusun kembali dengan menggunakan tiga tempat yang ada sampai didapatkan susunan blok angka yang tersusun dari angka 2 sampai 7 pada sebuah tempat.

Untuk menyelesaikan masalah penyusunan blok diperlukan suatu metode atau algoritma pencarian solusi. Salah satunya algoritma pencarian yang dapat digunakan adalah algoritma *Breadth First Search*, karena algoritma ini dapat melakukan pencarian solusi dari keadaan awal yang disediakan berdasarkan parameter-parameter yang ditentukan dari masalah penyusunan blok. Keadaan awal dari masalah penyusunan blok adalah terdapat blok angka yang acak sehingga susunan blok angka menjadi tidak terurut.

Berdasarkan uraian di atas, maka penulis merasa perlu untuk merancang sebuah aplikasi yang di dalamnya terimplementasikan algoritma *Breadth First Search*, untuk menyelesaikan masalah penyusunan blok.

2. Metode Penelitian

2.1 Dasar Pencarian

Pencarian atau pelacakan merupakan salah satu teknik untuk menyelesaikan permasalahan kecerdasan buatan.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 kecerdasan buatan[5].

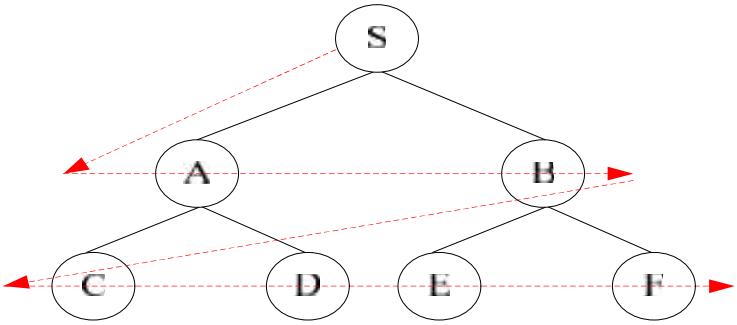
Konsep pencarian untuk suatu solusi dalam ruang keadaan (*state space*) merupakan pusat AI yang menjadikan AI lebih unggul dalam bidang ilmu komputer dibandingkan dengan yang lainnya.dan prinsip kontribusi AI untuk ilmu pengetahuan dari pencarian ini merupakan konsep basis pengetahuan (*knowledge based*)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

2.2 Pencarian Mendalam

Pencarian boleh jadi merupakan hasil dari suatu solusi atau ruang keadaan yang mungkin telah terkunjungi semua, tetapi tanpa penyelesaian.Pencarian yang mendalam (*Exhaustive Search Strategy*) dapat dilakukan dengan menggunakan algoritma *Breadth First Search* atau *Depth First Search*, dan kedua algoritma pencarian ini merupakan pencarian buta (*blind search*).

2.3 Breadth First Search

Breadth First Search (BFS) merupakan algoritma pencarian yang dilakukan dengan mengunjungi tiap-tiap *node* secara sistematis pada setiap level hingga keadaan tujuan (*goal state*) ditemukan[1]. Untuk lebih jelasnya, perhatikan illustrasi dari algoritma pencarian ini pada gambar berikut :



Gambar 1. *Breadth First Search*

Ada beberapa keuntungan menggunakan algoritma ini, di antaranya adalah tidak akan menemui jalan buntu dan jika ada satu solusi maka algoritma BFS akan menemukannya, dan jika ada lebih dari satu solusi maka solusi minimum akan ditemukan. Namun demikian, ada empat kelemahan dari algoritma BFS 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. Membutuhkan waktu yang cukup lama.

2.4 Masalah Penyusunan Blok

Penyusunan blok bisa juga dikatakan sebagai *Blocks World Architecture*, yaitu sebuah permainan yang mengajak pemainnya untuk berfikir secara logika dalam menyelesaikan permainan ini.*Blocks World Architecture* merupakan salah satu bagian dari *Artificial Intelligence* (AI).*Blocks World Architecture* dapat diilustrasikan seperti berikut, terdapat 2 atau lebih kotak yang disusun menjadi 2 atau 3 tumpukan balok sebagai kondisi awal (*initial state*) dan sasaran (*goal*) dari masalah ini adalah mendapatkan suatu tumpukan kotak sesuai dengan keinginan, dimana operasi yang diperbolehkan dalam proses penyelesaiannya yaitu turunkan(x), yang berarti bahwa kotak x diturunkan dari suatu tumpukan tertentu dan letakkan (x, y), yang berarti bahwa kotak x diletakkan di atas kotak y, dengan persyaratan bahwa kotak x dan kotak y harus berada di urutan paling atas dari suatu tumpukan.

Secara umum, *pseudocode* prosedur pencarian pada problema *blocks world architecture* dapat dituliskan seperti berikut ini:

```
Function GSearch(Problem, QueuingFn) : Solution | Failure;
Var nodes:structure;
Begin
    While
        Begin
Node := remove_front_node(nodes);
          IF Goal_test(problem, STATE(node)) succeeds then
Solution := Node
                    Else
```

```
Nodes := QueuingFn(node,OPERATOR(problem));
            End;
    EndWhile
End;
```

Sebagai contoh, perhatikan kasus sederhana berikut:

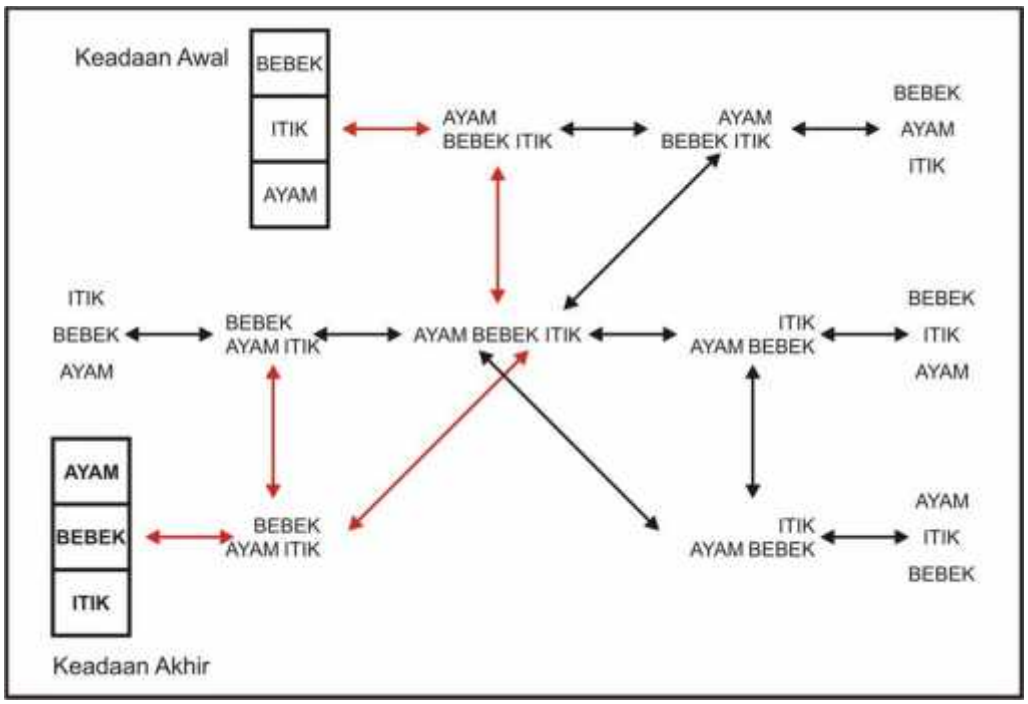
a. Keadaan awal:

Gambar 2. Keadaan awal

b. Keadaan akhir (*goal state* atau *final state*) adalah:

Gambar 3. Keadaan akhir (*goal state*)

Dengan demikian, ruang keadaannya (*state space*) adalah sebagai berikut:



Gambar 4. Ruang keadaan pada penyusunan blok

Dari gambar 4. di atas, dapat dilihat bahwa ruang keadaan tersebut mempunyai 13 elemen atau node, dan penyelesaian untuk permasalahan ini adalah anggota dari kumpulan semua lintasan dari keadaan awal hingga tujuan yang lintasannya ditandai dengan garis berwarna merah.

3. Hasil dan Analisis

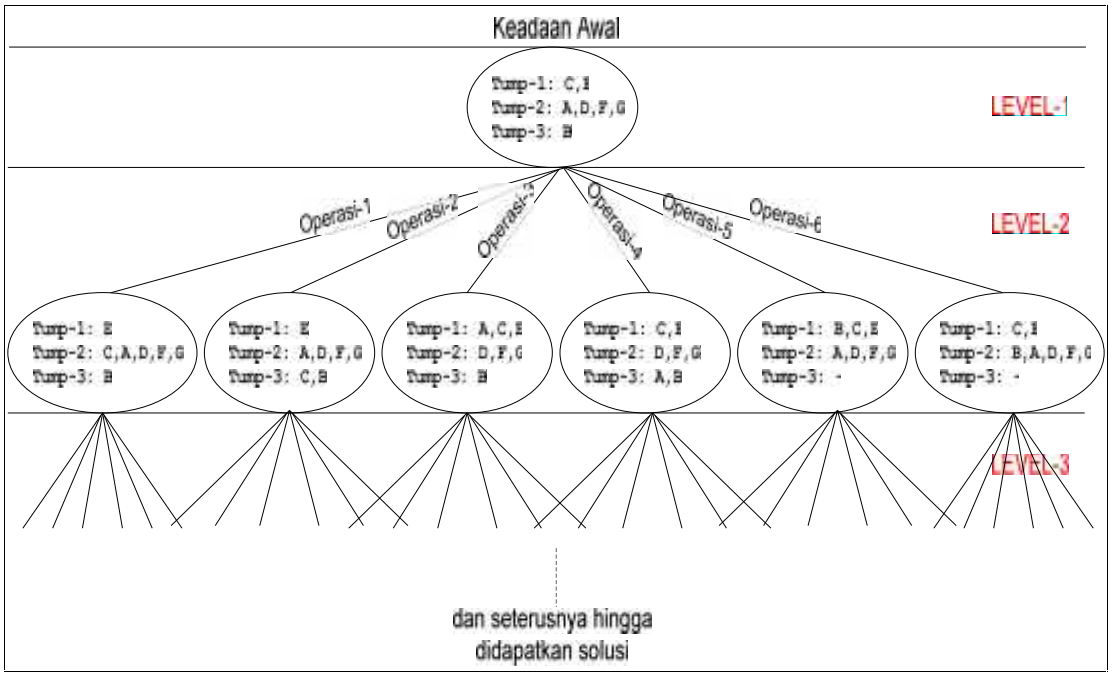
3.1 Algoritma *Breadth First Search*

Proses pencarian dengan menggunakan algoritma *Breadth First Search* merupakan pencarian yang dilakukan dengan mengunjungi tiap-tiap *node* secara sistematis pada setiap level hingga mencapai tujuan. Atau dengan kata lain, penelusuran dilakukan dengan mengunjungi *node-node* per level hingga ditemukan tujuan pencaraian. Pencarian *Breadth First Search* menjamin ditemukannya solusi dengan lintasan terpendek (*shortest path*). Karena proses BFS 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. Sebagai contoh implementasi algoritma *Breadth First Search* adalah sebagai berikut :

a. Keadaan awal :
   1. Tumpukan-1: C, E. (Balok C berada di atas balok E).
   2. Tumpukan-2: A, D, F dan G.
   3. Tumpukan-3: B.

b. Keadaan akhir :
   Tumpukan : A, B, C, D, E, F, G, H.

Pencarian solusi yang dilakukan algoritma *Breadth First Search* lebih jelasnya dapat dilihat pada gambar 5.



Gambar 5. Pencarian solusi dengan algoritma *breadth first search*

Pada gambar 5, dapat dilihat bahwa pencarian dimulai dari *node* paling atas (*root*) yang berisi masalah (level 1).Pada *node* root dilakukan beberapa operasi sehingga didapatkan keadaan-keadaan baru (level 2).Keadaan-keadaan baru ini direpresentasikan sebagai *node* anak dari *node* akar. Selanjutnya hal yang sama dilakukan untuk *node-node* yang baru terbentuk hingga ditemukan solusi atau tidak ada *node* yang dapat dikembangkan lagi (ini berarti tidak terdapat solusi – level n). Penyelesaian dari contoh di atas dengan menggunakan algoritma *Breadth First Search* dapat dilihat pada tabel 1.

Tabel 1. Urutan langkah-langkah penyelesaian susunan blok

No.	Langkah	Tump-1	Tump-2	Tump-3
0.	Keadaan Awal	C E	A D F G	B
1.	Pindahkan blok-C dari tump-1 ke tump-3	E	A D F G	B C
2.	Pindahkan blok-A dari tump-2 ke tump-3	E	D F G	B C A
3.	Pindahkan blok-D dari tump-2 ke tump-3	E	F G	B C A D
4.	Pindahkan blok-E dari tump-1 ke tump-3	-	F G	B C A D E
5.	Pindahkan blok-F dari tump-2 ke tump-3	-	G	B C A D E F
6.	Pindahkan blok-G dari tump-2 ke tump-1	G	-	B C A D E F
7.	Pindahkan blok-F dari tump-3 ke tump-1	F G	-	B C A D E
8.	Pindahkan blok-E dari tump-3 ke tump-1	E F G	-	B C A D
9.	Pindahkan blok-D dari tump-3 ke tump-1	D E F G	-	B C A
10.	Pindahkan blok-A dari tump-3 ke tump-2	D E F G	A	B C
11.	Pindahkan blok-C dari tump-3 ke tump-1	C D E F G	A	B
12.	Pindahkan blok-B dari tump-3 ke tump-1	B C D E F G	A	-
13.	Pindahkan blok-A dari tump-2 ke tump-1	A B C D E F G	-	-

Dari tabel di atas dapat dilihat bahwa pencarian solusi dengan algoritma *Breadth First Search* membutuhkan 13 langkah untuk mendapatkan solusi.

3.2 Spesifikasi Kebutuhan Perangkat Lunak

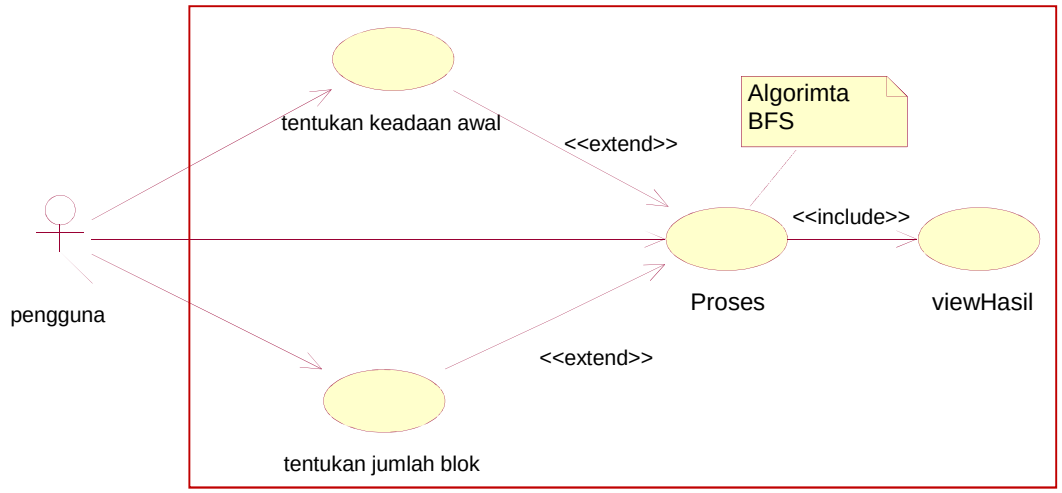
Terdapat sebuahspesifikasi kebutuhan pada kondisi awal, yaitu kebutuhan fungsional yang harus dipenuhi oleh perangkat lunak,

1. Pengguna dapat mengatur keadaan awal dari susunan blok
2. Aplikasi bisa menampilkan keadaan awal secara acak
3. Pengguna dapat memilih jumlah blok yang akan digunakan
4. Pengguna dapat melakukan proses BFS
5. Aplikasi dapat memproses algoritma BFS dan menampilkan hasilnya.

3.3 Rancangan Sistem

3.3.1 *Use Case* Diagram

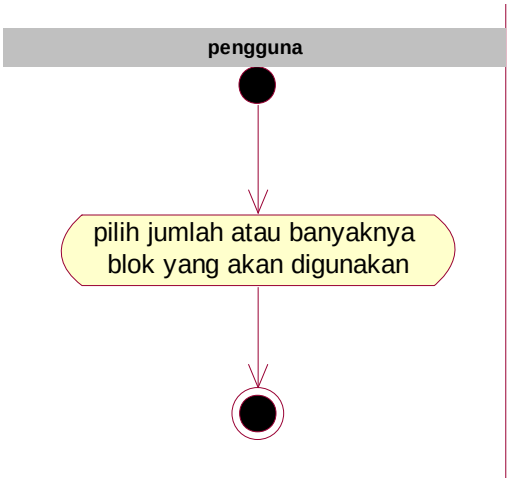
Dari *use case* diagram pada gambar 4.2 dapat dilihat fungsional dari setiap aktor pada sistem.



Gambar 6. *Use Case* Diagram

3.3.2 *Activity* Diagram Tentukan Jumlah Blok

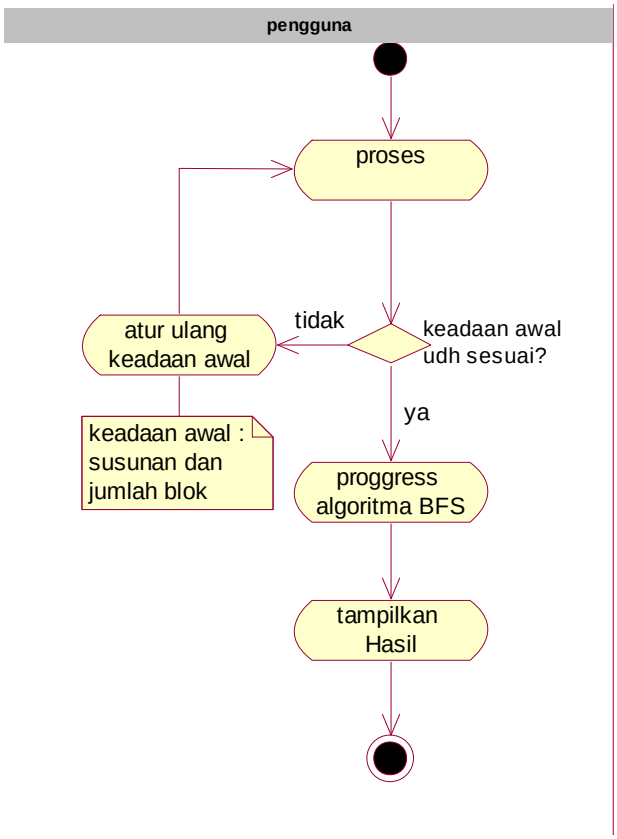
Pada gambar 7. menunjukkan *activity diagram* tentukan jumlah blok yang dilakukan oleh pengguna.



Gambar 7. *Activity* Diagram Tentukan jumlah blok

3.3.3 *Activity* Diagram Proses

Pada gambar 8. menunjukkan *activity diagram* proses yang dilakukan oleh pengguna.



Gambar 8. *Activity* Diagram Proses

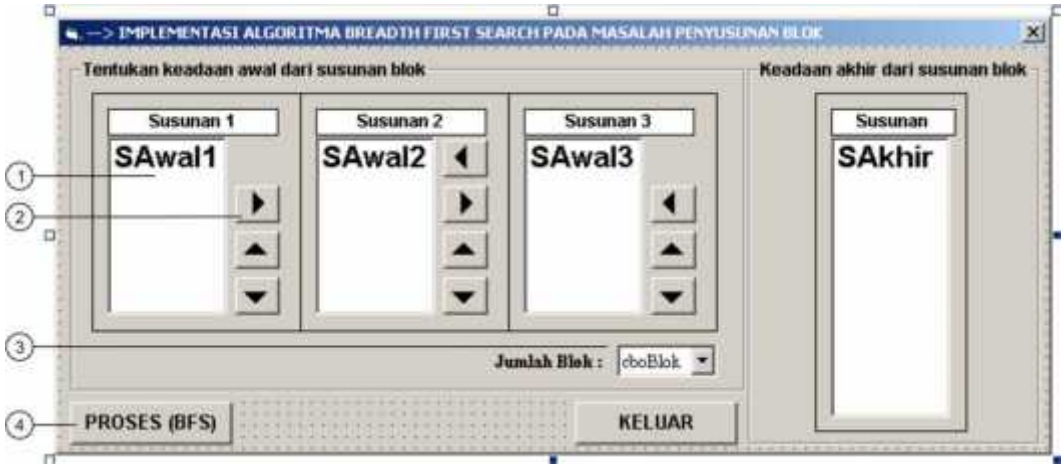
3.4 Rancangan Antarmuka (*Interface*)

Antarmuka perangkat lunak ini dirancang dengan menggunakan bahasa pemrograman *Microsoft Visual Basic 6.0* dengan menggunakan komponen standar. Antarmuka perangkat lunak ini memiliki beberapa tampilan *form*, seperti:

1. *Form* Awal
2. *Form* BFS
3. *Form* Hasil

3.4.1 Rancangan Antarmuka *Form* Awal

Form Awal berfungsi sebagai *form* utama perangkat lunak yang memiliki beberapa komponen, yaitu : komponen '*ListBox*' yang digunakan untuk melakukan menampung daftar susunan blok (awal atau akhir), komponen 'ComboBox' yang digunakan untuk menampung daftar jumlah blok yang akan digunakan (3 -10). Selain menu, *form* ini juga menyediakan tombol "ProsesBFS" yang berfungsi untuk memproses semua keadaan dengan algoritma BFS. Untuk lebih jelasnya perhatikan gambar 9. di bawah ini.

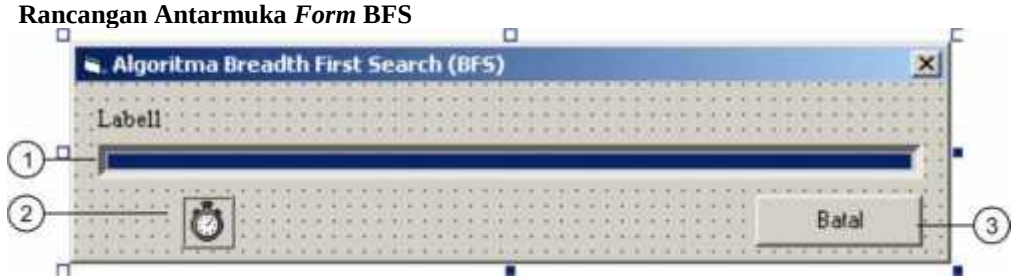


Gambar 9. *form*Awal

Keterangan :

1 Komponen ListBox digunakan untuk menampung daftar susunan blok Awal dan Akhir.
2 Tombol arah yang digunakan untuk mengatur susunan blok pada keadaan awal.
3 Komponen ComboBox digunakan untuk menentukan jumlah blok yang akan digunakan.
4 Tombol ProsesBFS digunakan untuk menjalankan algoritma BFS.

3.4.2 Rancangan Antarmuka *Form* BFS

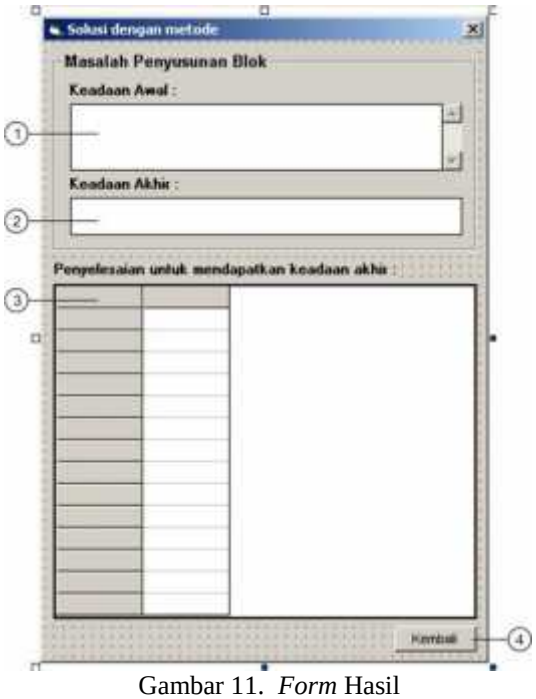


Gambar 10. *form*BFS

Keterangan :

1 ProggressBar digunakan untuk melakukan progress kemajuan dari pencarian algoritma BFS.
2 Timer digunakan untuk mengontrol waktu dari pencarian.
3 Tombol Batal digunakan untuk membatalkan pencarian dan kembali ke *form*Awal .

Rancangan Antarmuka *Form* Hasil



Gambar 11. *Form* Hasil

Keterangan :
1 TextBox digunakan untuk menampilkan keaadaan awal dari susunan blok.
2 TextBox digunakan untuk menampilkan keaadaan akhir dari susunan blok.
3 Grid digunakan untuk menapilkan hasil pencarian dengan menggunakan algoritma BFS.
4 Tombol Kembali digunakan untuk kembali ke *form* Awal.

4. Penutup

4.1 Kesimpulan

Berdasarkan perancangan, pembahsan dan pengujian yang telah dilakukan di bab-bab sebelumnya, maka dapat disimpulkan bahwa :

1. Untuk menentukan pencarian solusi dalam penyelesaian masalah penyusunan blok dapat menggunakan algoritma *Breadth First Search*.
2. Dari hasil pengujian yang disesuaikan dengan spesifikasi kebutuhan secara fungsional didapatkan bahwa semua fungsi yang terdapat di dalam perangkat lunak berjalan dengan baik.

4.2 Saran

Beberapa saran yang mungkin dapat membantu dalam pengembangan perangkat lunak ini, yaitu :

1. Perangkat lunak harus bisa menentukan keadaaan akhir dari susunan blok.
2. Diadakan studi kasus perbandingan antara metode BFS dengan metode pencarian solusi lainnya.

Referens:

[1] Alvin Hapendi, Tjatur Kandaga, 2010. "*Evaluasi dan Usaha Optimalisasi Algoritma Depth First Search dan Breadth First Search dengan Penerapan pada Aplikasi Rat Race dan Web Peta*". Fakultas Teknologi informasi, Universitas Kristen Maranatha.
[2] Himmatul Azizah,2011,"*Ekstraksi Metada Dokumen Software Requierement Specification*(SRS)", Bandung, (diakses dari http://digilib.its.ac.id/public/ITS-Undergraduate-16779-Paper-pdf.pdf pada 18 Januari 2014)
[3] Rosa, Shalahuddin M., 2011, "*Rekayasa Perangkat Lunak* (*Terstruktur dan* berorientasi *objek*)". Modula, Bandung.
[4] Simarmata Janner, 2010,"*Rekayasa Perangkat Lunak*", Andi, Yogyakarta.
[5] Setiawan Sandi, 2010, "*Artificial Intelegencia*", Andi Offset, Yogyakarta