MEMECAHKAN PERMAINAN *FREECELL* DENGAN ALGORITMA *BACKTRACKING*

Hengky Budiman
13505122
Program StudiTeknik Informatika
STEI ITB

ABSTRAK

Makalah ini membahas mengenai cara penyelesaian permainan *FreeCell* dengan menggunakan metode *backtracking* atau runut balik. *FreeCell* merupakan sebuah permainan *puzzle* kartu yang sangat terkenal. Permainan ini merupakan permainan yang dikembangkan dari permainan terdahulunya seperti *Klondike* dan *Eight Off*. Permainan *FreeCell* merupakan permainan yang lebih mengandalkan kemampuan dan strategi daripada keberuntungan. Permainan ini berkembang dengan sangat pesat sehingga *Windows* mengimplementasikannya dalam *entertainment pack*nya sehingga semua komputer yang memiliki OS *Windows* (kecuali *Windows Vista*) biasanya memiliki permainan *FreeCell*. Selain itu terdapat juga *website* – *website* yang menyediakan permainan *FreeCell* dengan fasilitas – fasilitas seperti turnamen antar pemain dan sebagainya.

Algoritma *backtracking* (runut balik) merupakan algoritma yang sangat terkenal dan banyak digunakan dalam memecahkan berbagai jenis permainan. Algoritma ini merupakan algortima yang berdasarkan pada DFS (*Deep First Search*) untuk mencari solusi suatu masalah. Runut balik merupakan algoritma yang lebih mangkus daripada algoritma *Brute Force* karena algoritma runut balik hanya memeriksa semua kemungkinan yang mengarah ke solusi saja. [4]

Kata Kunci : *games*/permainan, *FreeCell*, solver, algoritma *backtracking* / runut balik, kompleksitas

1. PENDAHULUAN

Permainan FreeCell telah berkembang sejak 1945. Saat itu nama permainan ini adalah *Napoleon at St. Helena*. [2] Permainan ini meraih kepopuleran di dunia berkat Jim Horne, yang membuat versi game ini dengan grafis berwarna dan mengimplementasikannya pada *Windows*. Permainan ini merupakan *best of Microsoft entertainment pack* pada saat itu.

[1]Saat itu, game ini memiliki 32.000 set (kombinasi kartu) yang dapat dimainkan. Tidak jelas apakah semua set tersebut dapat diselesaikan. Karena itu *Microsoft* menuliskan kalimat berikut pada bagian *help*nya:

> *It is believed (although not proven) that every game is winnable.*

Yang artinya: Dipercaya bahwa (meskipun tidak terbukti) semua game dapat diselesaikan. Untuk memperjelas hal ini, Dave Ring memulai suatu proyek bernama *The Internet FreeCell Project.* Proyek ini diselesaikan oleh orang – orang di seluruh dunia saat itu. Mereka diberikan beberapa game untuk diselesaikan, hasilnya kemudian dikumpulkan untuk membuktikan bahwa semua game dapat diselesaikan. Proyek ini selesai pada bulan Oktober 1995, dan ternyata terdapat sebuah permainan yang tidak bisa diselesaikan oleh semua pemain, dan juga semua program *solver* yang ada, yaitu game nomor #11982. Namun pada saat itu, *solver* yang digunakan masih diragukan kebenaranya. Game nomor #11982 ini akhirnya berhasil dibuktikan tidak dapat diselesaikan oleh seorang matematikawan.

[5]Meskipun terdapat 52!/8! atau sekitar 2×10^{63} kemungkinan urutan kartu atau set yang ada, tetapi beberapa memiliki kemiripan seperti kartu yang hitam dapat digantikan antara tanda sekop dan keriting. Karena itulah *Windows* pertama kali hanya memiliki 32.000 set permainan. Pada implementasinya kemudian, *Windows* menyediakan 1.000.000 set permainan. Dari set itu, beberapa set telah dibuktikan tidak dapat diselesaikan, yaitu nomor 11982, 146692, 186216, 455889, 495505, 512118, 517776, dan 781948

Seperti *Minesweeper, FreeCell* adalah permainan yang memiliki kompleksitas *NP-complete* untuk penyelesaiannya. Hasil ini terbukti pada tahun 2000, dan pertama kali diplubikasikan pada tahun 2001.

2. PERATURAN *FREECELL*

Berikut ini adalah perataruran dan cara bermain Freecell. [2]

1. Kocok kartu (joker tidak diikutsertakan). Kemudian bagi kartu tersebut ke dalam 8 kolom (terletak di bawah pada gambar 1). Akan terdapat 4 buah kolom yang memiliki 7 buah kartu dan 4 buah kololm yang memiliki 6 buah kartu. Pada

setiap kolom, kartu kedua diletakkan di bawah kartu pertama, kartu ketiga di bawah kartu kedua, dan seterusnya. Peletakkan kartu di bawah kartu yang lainnya hanya bermaksud bahwa hanya kartu terbawah yang boleh diambil. Apabila kita memainkan permainan ini dari program yang telah disediakan Windows, maka langkah ini tidak perlu dilakukan karena telah ditangani oleh program. Selain itu akan terdapat 4 buah kotak kosong (*free cells*) di kiri atas dan 4 buah tumpukan fondasi di kanan atas.

Gambar 1. Tampilan *FreeCell* di *Windows*

2. Objektif atau tujuan dari permainan ini adalah memindahkan semua kartu ke tumpukan fondasi.
3. Setiap kartu yang tunggal bisa dipindahkan apabila memenuhi kriteria berikut:
   a. Dari kolom yang satu ke kolom yang lain yang tidak kosong, apabila kolom yang dituju memiliki kartu terakhir memiliki indeks yang lebih rendah dengan perbedaan 1 dan berbeda warna. Indeks tertinggi adalah *King* dan indeks terendah adalah *As*. Urutan indeks sama dengan urutan yang digunakan dalam permainan kartu pada umumnya. Setiap kartu yang dipindahkan akan ditempatkan ke bawah kartu terkahir dari kolom yang dituju
   b. Dari kolom ke kolom lain yang masih kosong.
   c. Dari kolom ke *free cell*, apabila kotak *free cell* yang dituju masih kosong.
   d. Dari *free cell* ke kolom yang masih kosong.
   e. Dari *free cell* ke kolom yang tidak kosong, memiliki syarat yang sama dengan pemindahan kartu dari kolom ke kolom lain yang tidak kosong.
   f. Dari kolom ke tumpukan fondasi yang masih kosong, apabila kartu yang dipindahkan adalah As.
   g. Dari kolom ke tumpukan fondasi yang telah terisi, apabila kartu yang dipindahkan memiliki lambang yang sama dengan kartu terakhir tumpukan fondasi dan indeksnya lebih tinggi dengan perbedaan satu
   h. Dari *free cell* ke tumpukan fondasi yang masih kosong, apabila kartu yang dipindahkan adalah As.
   i. Dari *free cell* ke tumpukan fondasi yang tidak kosong, dengan syarat yang sama dengan pemindahkan kartu dari kolom ke tumpukan fondasi yang tidak kosong.

Penjelasan di atas akan lebih mudah dimengerti apabila langsung diterapkan di dalam permainan. Program FreeCell di Windows menyediakan layanan bantuan / *Help Content* dan akan menampilkan pesan kesalahan apabila kita melakukan pemindahan kartu yang melanggar batasan – batasan di atas.

3. METODE PENYELESAIAN PERMAINAN FREECELL

Pada bagian ini, akan dijelaskan algoritma backtracking untuk menyelesaikan permainan FreeCell.

1. Apabila status sekarang telah merupakan status *final* atau status selesai, permainan telah berhasil diselesaikan, return true. Kalau tidak lanjut ke langkah 2.
2. Kita daftarkan semua langkah yang mungkin dilakukan dari status sekarang.
3. Apabila tidak ada langkah yang bisa dilakukan maka berarti status sekarang tidak dapat melahirkan penyelesaian. *Backtracking* ke status sebelumnya. Apabila tidak ada lagi status yang dapat di*backtrack* maka berarti tidak ada solusi, *return false*. Apabila masih ada langkah yang bisa dilakukan, lanjut ke langkah 4.
4. Kita ambil satu langkah yang pertama dari *list*, hapus langkah yang diambil tersebut dari *list* dan kita lakukan langkah tersebut sehingga membentuk status yang baru.
5. Apabila status yang baru tersebut sudah pernah terjadi, kembali ke langkah ke 3. Apabila belum, kita ulangi langkah 1 hingga 5 dengan status yang baru terbentuk.

Algoritma utama untuk itu akan dijelaskan dengan menggunakan *pseudocode* berikut: [3]

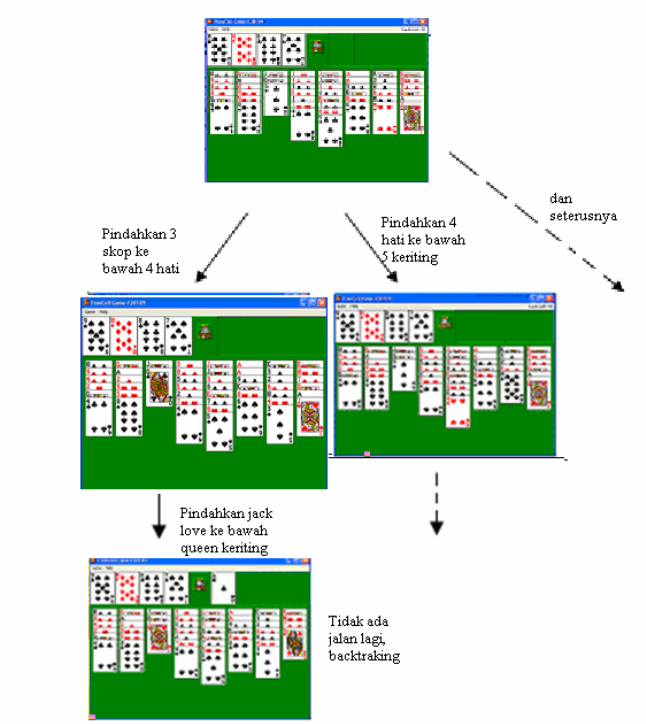
```
Boolean Solve(status_sekarang,
status_sudah_pernah, urutan_langkah)
Begin
  If (status_sekarang == kosong)
  Begin
    Push(urutan_langkah, status_sekarang);
    Return true;
  End
  Else
  Begin
    For (setiap langkah yang bisa dilakukan)
    Begin
      status_baru = status_sekarang;
      pindahkan(status_baru, langkah);
      if(status_baru belum ada pada
      status_sudah_pernah)
      Begin
        tambah(status_baru, status_sudah_pernah);
        if (Solve(status_baru,status_sudah_pernah,
        urutan_langkah))
        Begin
            Push(urutan_langkah,status_sekarang);
            Return true;
        End
      End
    End
    Return false;
  End
End

FreeCell-Solver (State_Init)
Begin
  If(Solve(State_init,Null,urutan_langkah))
  Begin
    Write("solusi ditemukan");
    While(status = Pop(urutan_langkah))
    Begin
        Write(state);
    End
  End
  Else
  Begin
    Write("solusi tidak ditemukan");
  End
End
```

Catatan:

Variabel dengan nama depan status atau *state* adalah variabel yang menggambarkan status kartu permainan FreeCell. Variabel ini bisa direpresentasikan dengan menggunakan representasi data internal yang berbeda – beda sesuai dengan keiinginan kita. Pada bagian selanjutnya, penulis akan menjelaskan representasi data ini dengan menggunakan data integer di dalam program yang dibuatnya.

Contoh ilustrasi dari algoritma di atas adalah sebagai berikut:



Gambar2. Ilustrasi Algoritma Runut Balik

4. KOMPLEKSITAS

Kompleksitas dari algoritma ini seperti yang telah disebutkan di atas, sangatlah besar dan merupakan masalah *NP-complete*. Pada kasus terburuk, terdapat 52+4= 56 sel atau tempat yang dapat diisi. 52 pada kolom utama, dan 4 pada sel kosong. Selain itu juga terdapat 13*13*13*13 kemungkinan nilai dari tumpukan fondasi. Karena itu, jumlah semua kemungkinan yang ada ialah 56 P (permutasi) 52. Dengan 52 adalah jumlah kartu yang ada. Hasil tersebut kemudian dibagi dengan 8! dan 4!. Dibagi 8! karena urutan kedelapan kolom dapat diacak, sedangkan dibagi 4! karena urutan keempat sel kosong juga dapat diacak. Hasilnya kemudian dikalikan dengan 13^4 (^ = pangkat) berdasarkan kombinasi tumpukan fondasi yang mungkin. Perhitungan di atas dilakukan apabila masukan awal adalah permainan yang baru (belum ada kartu pada tumpukan fondasi). Apabila masukan pertama telah memiliki kartu pada tumpukan fondasi sebanyak n1,n2,n3,n4, dengan n1 adalah nilai pada tumpukan kartu tanda sekop, n2 tanda hati, n3 tanda keriting dan n4 tanda wajik, maka jumlah semua kemungkinan yang ada ialah:

$$T(n1,n2,n3,n4) = \frac{((56-n1-n2-n3-n4)\,P\,(52-n1-n2-n3-n4))}{8!\,4!} \cdot (13-n1)(13-n2)(13-n3)(13-n4) \quad (1)$$

Terlihat bahwa kompleksitas algoritma di atas adalah O(p(n) n!), dengan p(n) adalah waktu komputasi setiap kemungkinan state kartu.

5. PENGGUNAAN METODE HEURISTIK

Salah satu cara yang digunakan untuk meningkatkan efektifitas algoritma adalah dengan menggunakan metode heuristik. Pada bagian ini, penulis akan mencoba untuk memberikan beberapa ide dan gagasannya mengenai metode heuristik yang dapat membantu.

Dari semua list langkah yang ada, akan sangat membantu kalau kita bisa mengambil langkah yang secara pasti akan menuju ke solusi (bila memang ada solusi). Beberapa di antaranya adalah:

1. Bila terdapat langkah yang menempatkan kartu pada tumpukan fondasi, dan angka pada kartu tersebut memiliki nilai indeks yang minimal sama dengan nilai indeks tumpukan fondasi lain yang berbeda warna, maka kartu tersebut pasti akan aman untuk ditempatkan di fondasi. Seperti kita ketahui, kartu pada tumpukan fondasi tidak dapat diambil lagi, hal ini menyebabkan terkadang terjadi kesalahan seperti terlalu cepat menaruh kartu pada tumpukan fondasi padahal kartu tersebut masih diperlukan untuk menyangga kartu – kartu lain yang berbeda warna dan memiliki angka lebih kecil dari dirinya. Tetapi dengan cara di atas, kesalahan tersebut tidak akan terjadi karena kartu lain yang berbeda warna dan memiliki angka lebih kecil sudah dimasukkan ke dalam tumpukan fondasi.
2. Bila terdapat langkah yang menempatkan kartu pada tumpukan fondasi, dan kartu tersebut memiliki angka yang lebih tinggi dari angka kartu pada tumpukan fondasi lain yang berbeda warna sebesar n, maka kartu itu akan aman dipindahkan apabila tumpukan fondasi lain yang memiliki warna yang sama dengan kartu tersebut memiliki angka yang lebih kecil sebesar 2n. Hal ini dikarenakan meskipun kita kehilangan kemungkinan untuk menyangga kartu – kartu lain dengan memasukkan kartu tersebut, tetapi masih terdapat kartu lain yang memiliki warna yang sama dengan kartu yang kita masukkan yang mampu menggantikan peran kartu yang kita masukkan tersebut. Tentu saja metode ini belum tentu benar karena bisa saja kartu yang kita harapkan dapat mengganti peran kartu lain yang kita masukkan ternyata terdapat di kolom yang paling bawah. Namun demikian, biasanya metode heuristik ini berhasil.
3. Pada Permainan *FreeCell*, terdapat gerakan yang disebut *super move* [6], yaitu gerakan yang memindahkan 2 atau lebih kartu sekaligus dari kolom yang satu ke kolom yang lain. Hal ini bisa dilakukan apabila terdapat ruang lain untuk menyimpan kartu sementara. Misalnya kita memiliki sebuah kolom dengan kartu teratas adalah *jack* hitam dan 10 merah, selain itu terdapat kolom lain dengan kartu teratas adalah *queen* merah. Kita dapat memindahkan *jack* dan 10 sekaligus ke atas *queen* apabila terdapat sebuah ruang kosong seperti sebuah sel kosong. Sebenarnya yang terjadi adalah pertama – tama 10 merah dipindahkan terlebih dahulu ke sel kosong tersebut, lalu *jack* dipindahkan ke atas *queen*, dan baru kemudian 10 dipindahkan ke atas *jack*. *Super move* akan sangat membantu mempercepat kerja program apabila diimplementasikan, meskipun demikian, super move juga tidak selalu menuntun kita menuju penyelesaian.
4. Urutan – urutan kartu yang dipindahkan terlebih dahulu cukup penting untuk meningkatkan kemangkusan program. Pemindahan kartu yang salah dari awal akan membuat program menjadi lambat karena program akan memeriksa semua kemungkinan hingga ditemukan jalan buntu sebelum melakukan runut balik ke awal. Dari penelitian yang telah dilakukan penulis, urutan pemindahan kartu dari yang sebaiknya dilakukan terlebih dahulu adalah sebagai berikut:
   - Pemindahan kartu ke tumpukan fondasi.
   - *Super move*.
   - Pemindahan kartu dari kolom ke sel kosong.
   - Pemindahaan kartu dari sel kosong ke kolom atau dari kolom ke kolom lain.

6. IMPLEMENTASI ALGORITMA

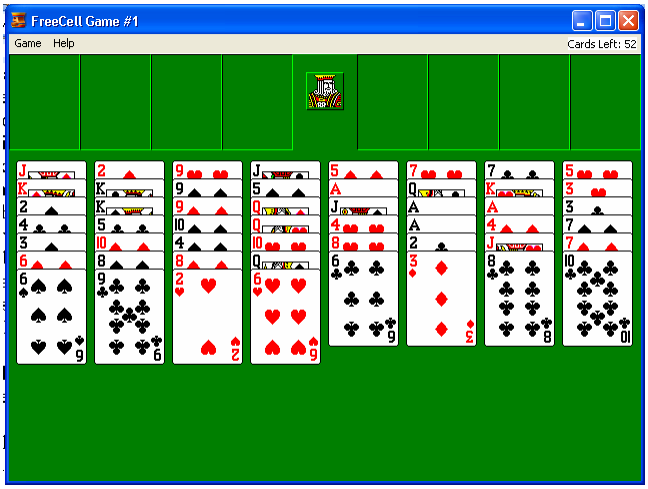
Dalam mengimplementasikan algoritma runut balik ini ke dalam program, penulis telah menyiapkan sebuah program dalam bahasa Java. Program dan *source code*nya dapat diunduh dari alamat berikut :

students.if.itb.ac.id/~if15122/FreeCellSolver.class (program)
students.if.itb.ac.id/~if15122/FreeCellSolver.java (source code)

Program ini membutuhkan JVM untuk menjalankannya. Program ini dibuat dan ditulis sendiri oleh penulis, dan merupakan program yang *open source.* Saran dan kritik untuk perbaikan program tersebut sangat penulis harapkan.

Berikut ini akan diberikan contoh hasil penggunaan program tersebut, permainan yang diambil adalah permainan nomor#1. Untuk memulai permainan ini, kita

cukup memasukkan angka 1 pada menu *game* > *select game* di Microsoft FreeCell Game. Setelah itu akan muncul tampilan seperti berikut :



Gambar 3. FreeCell Set #1

Gambar ini adalah permainan yang harus kita selesaikan. Pertama – tama kita harus membuat sebuah file yang berisi data yang merepresentasikan urutan kartu pada gambar tersebut. Tipe data yang digunakan adalah *integer*, karena itu kita perlu membuat suatu kesepakatan dalam menamai tanda dan angka pada kartu dengan nilai *integer* yang sesuai. Kesepakatan yang digunakan dalam program ini adalah sebagai berikut:

- Untuk angka, *As* diberi nomor 1, 2 diberi nomor 2, dan seterusnya hingga *King* diberi nomor 13.
- Untuk tanda, sekop diberi nomor 1, hati diberi nomor 2, keriting diberi nomor 3, dan wajik diberi nomor 4.

Sebagai contoh, gambar di atas jika direpresentasikan dalam file akan menjadi seperti berikut:

```
0 0 0 0 0 0 0 0
0 0 0 0
7 11 4 13 4 2 1 4 3 3 1 6 4 6 1
7 2 4 13 3 13 1 5 3 10 4 8 1 9 3
7 9 2 9 1 9 4 10 1 4 1 8 4 2 2
7 11 3 5 1 12 4 12 2 10 2 12 1 6 2
6 5 4 1 4 11 1 4 2 8 2 6 3
6 7 2 12 3 1 1 1 3 2 3 3 4
6 7 3 13 2 1 2 4 4 11 2 8 3
6 5 2 3 2 3 3 7 1 7 4 10 3
```

Baris pertama pada file tersebut berisi data dari sel kosong (*freecell*). Dua angka pertama merepresentasikan sel pertama, dua angka berikutnya merepresentasikan sel kedua, dan seterusnya. Untuk setiap sel, angka pertama menyatakan angka pada kartu dan angka kedua menyatakan tanda pada kartu. Karena belum ada kartu yang terisi pada sel – sel kosong ini, maka nilainya semua diberi nilai 0.

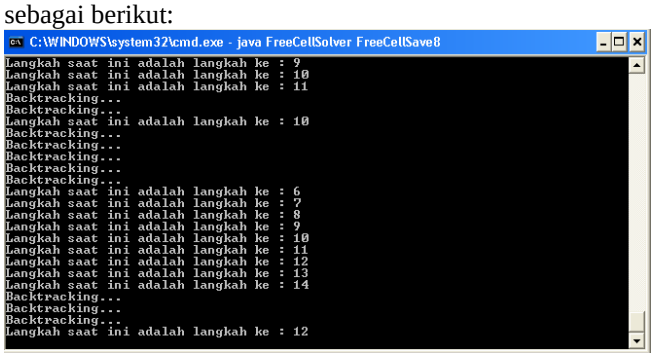
Baris kedua pada file tersebut menyatakan nilai dari tumpukan fondasi. Angka pertama untuk tanda sekop, kedua untuk tanda hati, ketiga untuk tanda keriting, dan keempat untuk tanda wajik. Karena tanda dari kartu sudah dinyatakan secara implisit dari urutan penulisannya, maka tanda dari kartu tidak perlu ditulis ulang, hal ini tidak seperti baris pertama yang menyatakan sel kosong. Misalnya jika kita ingin mencari solusi permainan di tengah permainan yang tumpukan fondasinya sudah terisi, maka bila tumpukan fondasi untuk tanda keriting sudah terisi hingga 5 keriting, kita menyatakannya dengan angka 5 pada posisi angka ketiga. Karena pada game ini, belum ada kartu pada tumpukan fondasi, maka semua isi dari baris kedua diberi nilai 0.

Baris ketiga hingga terakhir menyatakan nilai dari kolom utama. Baris ketiga untuk kolom pertama dari kiri, baris keempat untuk kolom kedua dari kiri, dan seterusnya. Dari setiap baris, diberikan isi yang menyatakan jumlah kartu pada kolom tersebut dan isi kartunya. Pada baris ketiga, angka pertama yaitu 7 menyatakan bahwa jumlah kartu pada kolom pertama adalah 7. Dua angka berikutnya 11 dan 4 menyatakan bahwa kartu pertama (teratas) dari kolom pertama adalah 11 wajik. Dua angka berikutnya, 13 4, menyatakan kartu berikutnya adalah *King* wajik, dan seterusnya.

Kemudian kita menjalankan program tersebut dengan cara:

- Buka Command Prompt
- Masuk ke direktori tempat menyimpan file FreeCellSolver.class
- Ketikkan java FreeCellSolver <namafileinput>, hilangkan tanda ”< >”
- Setelah selesai, hasil dari program yang bernama HasilFreeCellSolver.txt akan dibuat. Pada file ini terdapat langkah – langkah penyelesaian permainan.

Ilustrasi yang muncul ketika program dijalankan adalah sebagai berikut:



Gambar 4. Ilustrasi Program Yang Sedang Berjalan

Untuk file input permainan dengan nomor #1 di atas, penulis telah menyiapkannya. File tersebut dapat diunduh di students.if.itb.ac.id/~if15122/FreeCellSolverSave

Hasil dari file output untuk permainan di atas berisi langkah – langkah yang jumlahnya 387 langkah. Langkah – langkah tersebut tidak dapat ditulis di sini karena terlalu banyak. Berikut ini adalah potongan hasil file output:

```
Pindahkan : dari kolom urutan ke 7 ke freecell
urutan ke 2
Pindahkan : dari kolom urutan ke 7 ke kolom urutan
ke 5
Pindahkan : dari freecell urutan ke 2 ke kolom
urutan ke 5
Pindahkan : dari kolom urutan ke 7 ke freecell
urutan ke 2
Pindahkan : dari kolom urutan ke 7 ke tumpukan
fondasi urutan ke 2
```

7. KESIMPULAN

Algoritma runut balik merupakan algoritma yang lebih mangkus daripada algoritma *brute force*. Meskipun demikian, untuk kasus yang sangat besar, algoritma ini tetap memiliki komplesitas algoritma yang tinggi.

Penggunaan metode heuristik akan sangat membantu dalam meningkatkan kemangkusan algoritma ini. Meskipun tidak selalu benar, tetapi metode ini dapat menuntun algoritma runut balik agar memilih langkah yang sekiranya paling tepat sehingga kemungkinan program melakukan runut balik karena pemilihan yang salah akan berkurang.

Penyelesaian permainan seperti *FreeCell* dan permainan lainnya yang merupakan masalah *NP – complete* mungkin akan lebih cepat diselesaikan dengan cara manual daripada dengan menggunakan program. Tetapi untuk beberapa kasus, seperti membuktikan suatu set permainan tidak dapat diselesaikan, penggunaan program akan banyak membantu.

8. REFERENSI

[2] FreeCell FAQ And Links, http://www.solitairelaboratory.com/fcfaq.html. Diakses pada tanggal 19 Mei 2007, Pukul 12.00 WIB.

[2] FreeCell Solver – lecture , http://www.shlomifish.org/lecture/Freecell-Solver/fcs-lecture-points-take3.txt. Diakses pada tanggal 19 Mei 2007, Pukul 12.30 WIB.

[3] FreeCell Solver – lecture , http://www.shlomifish.org/lecture/Freecell-Solver/ Summary-deprecated.txt. Diakses pada tanggal 19 Mei 2007, Pukul 12.30 WIB.

[4] Munir, Rinaldi, ”Diktat Kuliah IF2251 Strategi Algoritmik”, Program Studi Teknik Informatika STEI, ITB, 2007.

[5] Wikipedia – The Free Encyclopedia, http://en.wikipedia.org/wiki/FreeCell. Diakses pada tanggal 16 Mei 2007, Pukul 16.00 WIB.

[6] Wikipedia – The Free Encyclopedia, http://en.wikipedia.org/wiki/Solitaire_terminology. Diakses pada tanggal 16 Mei 2007, Pukul 16.00 WIB.