

PENERAPAN ALGORITMA BACKTRACKING PADA PERMAINAN PEG SOLITAIRE

Alim Astiri Kirani
10111438

Yulia Anggrawati
10111451

Ezra Janitra R
10111440

Abstrak

Abstrak—*Peg Solitaire* merupakan salah satu *board game* populer di dunia yang dapat dimainkan oleh satu orang pemain. *Peg Solitaire* dimainkan dengan menggunakan beberapa set kelereng dan papan yang berlubang. Objektif dari permainan ini adalah memindahkan *peg* dengan melangkahi suatu *peg* di depannya. *Peg* dapat dipindahkan ke lubang yang kosong, dan *peg* yang dilangkahi dapat dihilangkan. Tujuan akhir permainan ini adalah menghabiskan seluruh *peg* yang ada dengan melangkahi *peg-peg* tersebut hingga tersisa satu *peg*. Meskipun terdengar sederhana, namun untuk dapat mencapai kondisi seperti itu sangat sulit untuk dicapai. Dibutuhkan pemikiran yang matang dalam setiap langkah yang diambil. Karena jika salah, maka di akhir akan ada lebih dari satu *peg* yang tersisa. Dalam makalah ini, penulis mencoba untuk mencari solusi untuk menyelesaikan permainan ini dengan menerapkan algoritma *backtracking*. Algoritma *backtracking* adalah salah satu algoritma yang digunakan untuk mencari solusi dari permasalahan komputasional. Ada banyak cara yang dapat digunakan untuk mencari solusi, dan algoritma *backtracking* bukanlah solusi terbaik setiap saat. Namun, algoritma ini dapat digunakan untuk mencapai kondisi akhir yang ideal dari permainan *peg solitaire*.

Kata Kunci: Algoritma *backtracking*, *peg solitaire*, *peg*, *hash-table*.

1. PENDAHULUAN

Peg Solitaire adalah salah satu permainan computer dan cukup populer sejak dulu. Menurut beberapa orang, permainan ini pertama kali diciptakan oleh seorang aristokrat Perancis pada abad ke 17. Namun, hal ini masih menjadi sebuah perbedaan di tahun 1800-an. Salah satu orang yang mencari pembuktian ini bernama John Beasley. John Beasley mengemukakan bahwa permainan ini bukanlah berasal dari Perancis, melainkan salah satu warisan turun-temurun dari suku asli Amerika. Walaupun pernyataannya ini juga belum memiliki bukti yang cukup kuat. Permainan ini terkenal di beberapa negara dan memiliki nama yang berbeda-beda pula tergantung tempat permainan ini dimainkan.

Ada dua jenis papan yang lazim digunakan pada permainan ini, jenis papan Inggris dan jenis papan yang digunakan Negara Eropa lain seperti gambar berikut :



Gambar 1. Jenis papan *Peg Solitaire* Inggris



Kesalahan sangat mungkin dilakukan dan menyebabkan adanya dua atau tiga *peg* yang letaknya saling berjauhan. Hal ini menyebabkan banyak orang tidak bisa menyelesaikan permainan ini.

Ada banyak solusi untuk permasalahan yang sering ditemui, dan satu cara yang bisa dilakukan yaitu memberi huruf pada tiap lubang yang ada :

| Inggris | | | | | | | | | | | | | Eropa | | | | | | | | | | | | |
|---------------|--|--|--|--|--|--|--|--|--|--|--|--|---------------|--|--|--|--|--|--|--|--|--|--|--|--|
| a b c | | | | | | | | | | | | | a b c | | | | | | | | | | | | |
| d e f | | | | | | | | | | | | | y d e f z | | | | | | | | | | | | |
| g h i j k l m | | | | | | | | | | | | | g h i j k l m | | | | | | | | | | | | |
| n o p x P O N | | | | | | | | | | | | | n o p x P O N | | | | | | | | | | | | |
| M L K J I H G | | | | | | | | | | | | | M L K J I H G | | | | | | | | | | | | |
| F E D | | | | | | | | | | | | | Z F E D Y | | | | | | | | | | | | |
| C B A | | | | | | | | | | | | | C B A | | | | | | | | | | | | |

Notasi gambar cermin seperti ini digunakan karena pada papan Eropa, satu set alternative dari permainan ini adalah dimulai dengan lubang di beberapa posisi dan diakhiri dengan satu *Peg* diposisi cerminnya. Dalam papan Inggris set alternative yang serupa dimulai dengan lubang dan diakhiri dengan *Peg* diposisi yang sama.

Tidak ada solusi untuk papan Eropa yang memiliki lubang ditengah, jika hanya pergerakan orthogonal yang diizinkan. Hal ini sangat mudah kita lihat sebagai berikut, dengan argument dari seseorang bernama Hans Zantema. Bagi posisi papan ke dalam posisi A, B, dan C seperti berikut :

```

      A B C
    A B C A B
  A B C A B C A
B C A B C A B
C A B C A B C
  B C A B C
    A B C

```

Pada awalnya dengan hanya posisi tengah papan permainan yang kosong, jumlah posisi yang dimiliki oleh A adalah 12, posisi B 12, juga posisi C 12. Setelah setiap perpindahan, jumlah posisi A naik atau turun sejumlah satu, dan hal yang sama juga berlaku pada B dan C. Sehingga, setelah perpindahan sebanyak n kali dengan n adalah bilangan genap pula, dan setelah perpindahan sebanyak m kali dengan m adalah bilangan ganjil, ketiga angka (untuk A, B, dan C) akan menjadi ganjil pula. Sehingga posisi akhir dengan hanya 1 (ganjil), sedangkan yang dua lainnya adalah 0 (genap).

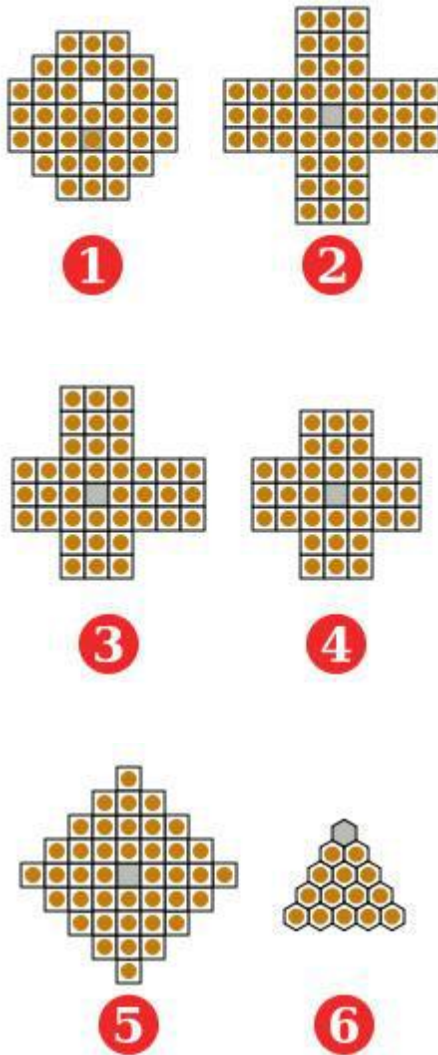
Namun, terdapat cara lain dimana sebuah lubang yang ada dapat disederhanakan menjadi hanya satu *Peg* yang tersisa. Taktik yang digunakan adalah dengan membagi papan permainan menjadi paket-paket terdiri dari tiga *Peg* tersebut dengan menggunakan sebuah *Peg* tersebut dengan menggunakan sebuah *Peg* ekstra, sebagai katalis, yang digunakan untuk lompat keluar dan kemudian lompat ke dalam lagi. Contohnya dapat dilihat digambar berikut, dimana * adalah katalisnya :



Teknik ini dapat digunakan dengan baris yang terdiri dari 3 *Peg*, blok yang terdiri dari 2×3 *Peg*, dan 6 *Peg* berbentuk L dengan dasar yang panjangnya 3 *Peg* dan tinggi yang panjangnya 4 *Peg*.

Selain kedua jenis papan permainan yang populer tersebut, ada juga jenis dan ukuran papan

yang lainnya, termasuk juga papan permainan berbentuk segitiga, dengan *peg* yang digunakan boleh melompat ke tiga arah.



Gambar 4. Macam-macam papan Peg Solitaire lain.

- 1) Tipe Perancis, 37 lubang, digunakan pada abad 17
- 2) J.C.Wiegand (Jerman), 45 lubang, 1779
- 3) 3-3-2-2 Asimetris oleh George Bell, pada abad 20
- 4) Tipe Inggris (standar), 33 lubang
- 5) Wajik, 41 lubang
- 6) Segitiga, 15 lubang

B. Algoritma Backtracking

Terdapat beberapa algoritma yang digunakan secara global untuk menemukan seluruh atau sebagian permasalahan komputasi, dua diantaranya adalah Brute Force dan Backtracking. Kedua algoritma ini sama-sama mencoba untuk mencari solusi dari permasalahan dengan membuat kandidat-kandidat solusi secara gradual. Bagaimanapun, backtracking lebih efisien dibandingkan dengan algoritma brute force. Ini dikarenakan, pada algoritma brute force, sebuah program akan membuat setiap pemecahan atau solusi yang mungkin. Untuk setiap solusi, program akan kemudian mengecek apakah solusi tersebut telah memenuhi spesifikasi atau syarat solusi final yang sebenarnya. Sementara itu, algoritma backtracking akan berhenti memproses kandidat solusi jika telah terbukti tidak mengarah pada solusi final yang sebenarnya. Misalnya, sebuah program diminta untuk menyusun sebuah kata yang terdiri dari 5 huruf, dengan syarat terdiri dari maksimal tiga huruf konsonan. Saat program menemukan huruf konsonan ke empat dalam prosesnya mencari solusi yang sesuai, maka kandidat tersebut dan yang selanjutnya yang mengikuti akan dibuang. Karena sudah tidak mungkin lagi untuk memenuhi syarat solusi yang diminta.

Namun, algoritma backtracking hanya dapat diaplikasikan pada beberapa permasalahan tertentu. Algoritma ini hanya dapat digunakan pada solusi atau pemecahan permasalahan yang dapat dicari secara sistematis dan bertahap. Algoritma backtracking hanya dapat diterapkan pada permasalahan yang memiliki konsep kandidat solusi parsial, dan dapat dicek secara cepat apakah kandidat tersebut mengarah pada solusi yang valid.

Algoritma backtracking didasarkan pada DFS (Depth First Search). Mekanisme backtracking menggunakan prinsip rekursif. Untuk menemukan solusi dari keseluruhan permasalahan, diperlukan solusi dari sub

permasalahannya. Setelah itu, solusi ini akan digunakan untuk memecahkan subpermasalahan lainnya secara rekursif. Saat kandidat yang saat ini sedang diperiksa gagal, atau jika seluruh solusi yang mungkin harus ditemukan, maka program akan melakukan backtrack ke node sebelumnya dan mengecek solusi berikutnya yang mungkin. Proses backtracking akan berhenti saat sudah tidak ada lagi solusi yang mungkin. Salah satu karakteristik yang paling signifikan dari algoritma backtracking adalah fungsi “pemotongan”. Misalnya tahapan dari pencarian solusi tersebut direpresentasikan dalam bentuk tree, pemotongan akan dilakukan untuk node-node yang tampaknya tidak akan membentuk solusi yang valid. Saat sebuah node telah dipotong, anak node tersebut secara otomatis akan dibuang dari proses, karena memotong sebuah node sama artinya dengan membuang seluruh path yang mengikutinya.

Algoritma backtracking digunakan secara luas dalam pembuatan game PC seperti tic-tac-toe, labirin dan catur. Algoritma ini biasanya digunakan untuk membangun Artificial Intelligence (AI) untuk game. Di samping itu, algoritma ini juga merupakan cara paling efisien untuk parsing dan permasalahan kombinatorial lainnya.

3. PENGAPLIKASIAN ALGORITMA BACKTRACKING

Algoritma backtracking atau runut balik dapat diaplikasikan untuk mencari solusi dari permainan peg solitaire. Contoh algoritma untuk menyelesaikan permainan peg solitaire adalah sebagai berikut :

```
Boolean move (board x, moveSequence mseq) {
  If (solved(x)) return true;
  curMoves = currentMoves (x) ; // himpunan
  semua pergerakan
  for (every m in curMoves) {
    y = makeMove(x, m);
    if (move (y, mseq)) {
      mseq = push (m, mseq);
```

```
      return true;
    }
  }
  Return false ;
}
```

Algoritma backtracking seperti yang ditulis diatas akan menyelesaikan permainan dengan jumlah peg dengan jumlah sekitar 20. Tetapi untuk papan dengan jumlah peg yang lebih banyak, algoritma backtracking akan memakan waktu beberapa menit hingga beberapa jam, bahkan bisa sehari-hari. Tetapi dengan menambahkan Hash Table dapat meningkatkan performansi dari algoritma secara signifikan.

Ide utamanya adalah sebagai berikut. Misalkan X adalah posisi awal. Dan kemudian banyak urutan pergerakan yang akan membawa X ke posisi Y. Algoritma backtracking akan mengikuti jalur yang sesuai dengan salah satu urutan hingga sampai ke posisi Y. Pada tahap ini, rekursif akan terjadi dengan Y sebagai input. Misalkan Y tidak menunjukkan solusi yang diinginkan, maka panggilan ini akan mengeluarkan false dan pencarian akan dilanjutkan. Pada suatu saat, algoritma backtracking akan mengikuti urutan pergerakan yang lain dan akan menuju posisi Y. Pada tahap ini, pemanggilan rekursif akan terjadi lagi dengan Y dan dapat terjadi berulang kali. Pengulangan dapat dihindari dengan memperhatikan semua papan dimana pemanggilan rekursif telah gagal sejauh ini dalam sebuah struktur data. Operasi ini akan lebih efisien dengan didukung oleh struktur data dengan metode search and insert. Dengan begitu, hash table digunakan untuk menyimpan papan-papannya. Berikut ini adalah versi modifikasi dari algoritma backtracking dengan menggunakan hash table.


```

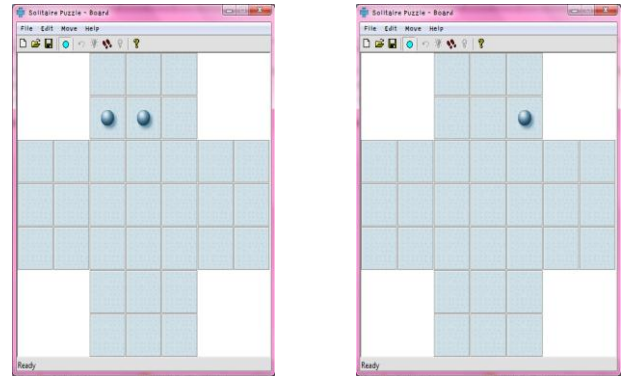
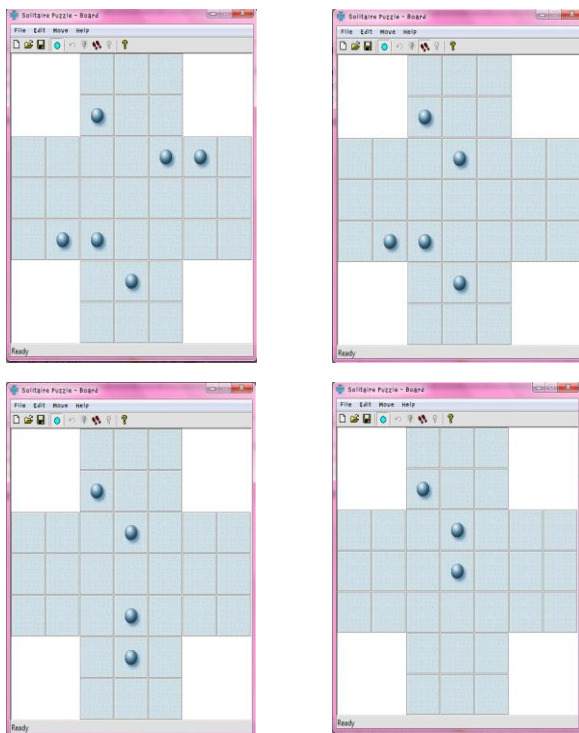
Boolean move ( board x , moveList mseq ,
Hashtable H) {
If (solved ( x )) return ;
curMoves = currentMoves ( x ) ; // semua
pergerakan yang mungkin di x
for ( every m in curMoves ) {
    y = makeMove ( x , m); // y adalah hasil
    pembuatan m di x
    if ( y is not in H) {
        if move ( y , mseq , y );
        mseq = push (mseq , y);
        return true;
    } // end inner if
} Else
    Insert y into H ;
} // end outer if
} // end for
Return false;
} // end move

```

Dapat dilihat bahwa perubahan yang dibuat yaitu :

1. Sebelum pemanggilan rekursif, dilakukan pencarian untuk papan Y di hashtable untuk menghindari pemanggilan berulang.
2. Saat pemanggilan mengembalikan false, disisipkan papan ke hash-table.

Contoh implementasi solusi :



Gambar 5. Implementasi algoritma Backtracking

4. Kesimpulan

Algoritma backtracking biasa digunakan untuk menyelesaikan masalah komputasional. Penggunaan algoritma ini cukup efektif saat diaplikasikan ke permasalahan yang sesuai. Dengan beberapa perbaikan, seperti penambahan hash-table, algoritma ini dapat menyelesaikan permainan peg solitaire dengan cukup baik. Tetapi, solusi menggunakan algoritma backtracking mungkin bukan solusi yang terbaik untuk permainan ini.

REFERENSI

- [1] Munir, Rinaldi, "Strategi Algoritmik", Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2007, 125-149
- [2] http://en.wikipedia.org/wiki/Peg_solitaire
- [3] <http://en.wikipedia.org/wiki/Backtracking>
- [4] <http://www.codeproject.com/Articles/11595/Solitaire-Puzzle-with-Backtracking>