Aplikasi Algoritma *Brute Force* dan *Backtracking* pada Permainan *Slitherlink*

Kevin Chandra Irwanto 13508063
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18063@students.if.itb.ac.id

ABSTRAK

Slitherlink merupakan suatu permainan logika yang bermain-main dengan titik, garis, dan angka. Slitherlink ini di-develop oleh Nikoli Co., Ltd. (perusahaan game di Jepang yang juga men-develop Sudoku).
Cara bermainnya cukup mudah hanya dengan menaruh garis-garis sebanyak angka, dan yang terpenting adalah garis-garis tersebut harus membentuk suatu loop yang tidak boleh putus.
Pada Slitherlink tidak mempunyai solusi ganda, yakni hanya 1 solusi mutlak untuk setiap puzzle-nya. Angka-angka yang digunakan hanya 0, 1, 2, dan 3.
Pada makalah ini akan dibahas penyelesaian solusi untuk permainan Slitherlink ini dengan algoritma Brute Force dan Backtracking. Pembahasan juga disertai dengan peraturan-peraturan dalam permainan Slitherlink ini.

Kata Kunci : *Slitherlink*, *Fences*, *Loop the Loop*, *Ouroboros*, *Dotty Dilemma*, *Brute Force*, *Backtracking*,

1. PENDAHULUAN

Slitherlink adalah suatu permainan logika yang hanya menggunakan angka, garis, dan titik dalam permainannya. Permainan ini di-*develop* oleh Nikoli Co., Ltd., perusahaan yang juga men-*develop* permainan Sudoku.

Awal mula penulis bermain **Slitherlink** lewat suatu aplikasi pada situs jejaring yang sudah tidak asing lagi, *Facebook*. Lalu akhirnya tercetus ide untuk mengambil tema makalah tentang permainan ini.

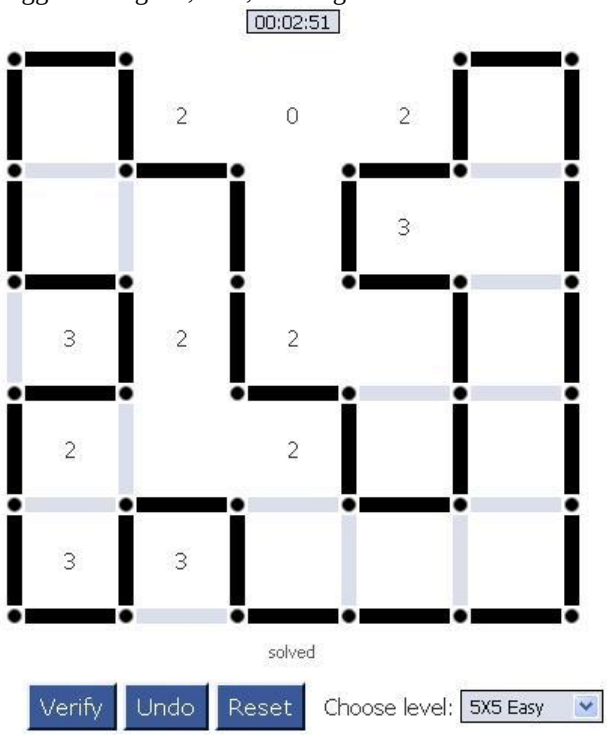
Jadi, pada makalah ini akan dibahas tentang bagaimana permainan **Slitherlink** itu sendiri, peraturan-peraturan serta bagaimana penerapan algoritma *Brute Force* dan *Backtracking* dalam permainan ini.

2. DASAR TEORI

Pada bagian ini akan dijelaskan secara rinci apa itu permainan **Slitherlink** maupun algoritma yang dipakai, dalam hal ini algoritma yang dipakai adalah algoritma *Brute Force* dan *Backtracking*.

2.1. Slitherlink

Seperti yang telah dijelaskan sebelumnya, permainan ini adalah permainan sederhana yang hanya menggunakan garis, titik, dan angka.



Gambar diatas memperlihatkan suatu solusi untuk penempatan angka seperti pada gambar.

Berikut peraturan-peraturan permainan **Slitherlink** ini, antara lain :

- Angka di dalam suatu kotak kecil menandakan banyaknya garis yang harus dibuat diantara 4 garis yang tersedia,
- Garis-garis yang dibuat harus membentuk suatu *loop* (seperti pada contoh gambar di atas, garis hitam tebal membentuk suatu *loop* besar yang mengikuti aturan sebelumnya), karena itu tidak boleh ada garis yang bercabang,
- Hanya ada satu loop besar, tidak boleh ada 2 atau lebih loop dalam satu *problem*, dan
- Hanya ada satu solusi mutlak (tidak mengenal solusi ganda).

Bentuk dasar pada permainan ini tidak hanya terpaku pada bujur sangkar saja, namun bisa juga menggunakan suatu *hexagon*, *pentagon*, atau bentuk dasar lainnya.

Pada **Slitherlink** terdapat kotak kosong yang tidak jelas berapa garis yang harus dibuat. Kotak-kotak inilah yang sebenarnya membuat permainan menjadi sulit, karena kotak dengan angka di dalamnya lebih mudah diisi daripada kotak yang tidak ada. Namun tingkat kesulitannya bukan berarti terletak pada banyaknya kotak yang tidak berangka, tetapi terletak pada kombinasi kotak-kotak itu sendiri.

2.2 Algoritma *Brute Force*

Brute Force adalah sebuah pendekatan yang lempang(*straight-forward*) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah(*problem statement*) dan definisi konsep yang dilibatkan. Algoritma ini memecahkan masalah dengan sangat sederhana, langsung, dan dengan cara yang jelas(*obvious way*).

Algoritma ini umumnya tidak "cerdas" dan tidak mangkus, karena ia membutuhkan jumlah langkah(*step*) yang besar dalam penyelesaiannya, terutama bila masalah yang dipecahkan berukuran besar. Walaupun bukan merupakan teknik pemecahan masalah yang mangkus, namun algoritma ini dapat diterapkan hampir untuk seluruh masalah yang ada. Bahkan ada beberapa masalah yang hanya bisa dipecahkan secara *Brute Force*.

Ketika menggunakan *Brute Force*, biasanya membantu kita dalam menemukan algoritma yang lebih mangkus. Selain itu, implementasinya juga lebih sederhana dibanding algoritma-algoritma yang lebih canggih.

2.3 Algoritma *Backtracking*

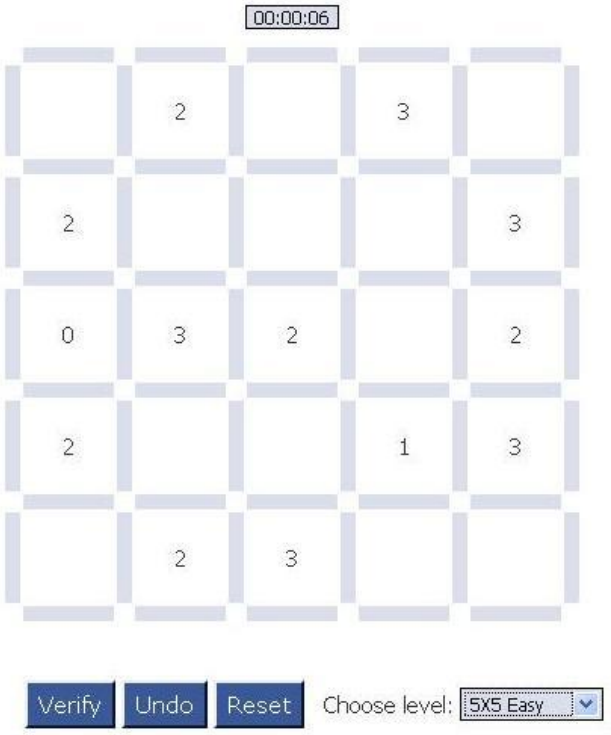
Backtrack atau juga disebut Runut-Balik adalah algoritma yang berbasis pada DFS(*Depth First Search*) untuk mencari suatu solusi persoalan secara lebih mangkus. Algoritma ini sebenarnya merupakan perbaikan dari algoritma *Brute Force*. Kadang pula disebutkan bahwa runut-balik merupakan bentuk tipikal dari algoritma rekursif(berulang).

Istilah runut-balik pertama kali diperkenalkan oleh D.H. Lehmer tahun 1950. Algoritma ini banyak digunakan pada permainan logika seperti *tic-tac-toe*, catur, menemukan jalan keluar dari labirin, dan masalah-masalah pada AI(*Artificial Intelligence*).

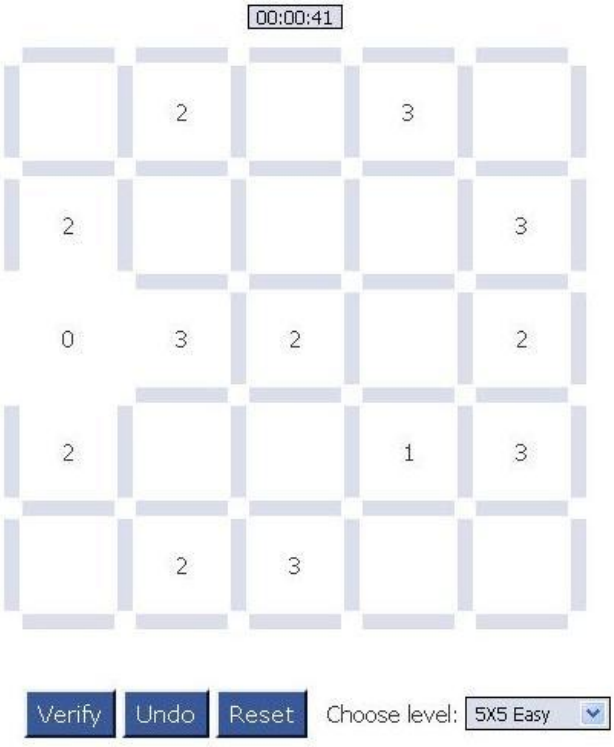
3. METODE

Metode yang digunakan adalah dengan algoritma *Brute Force*.

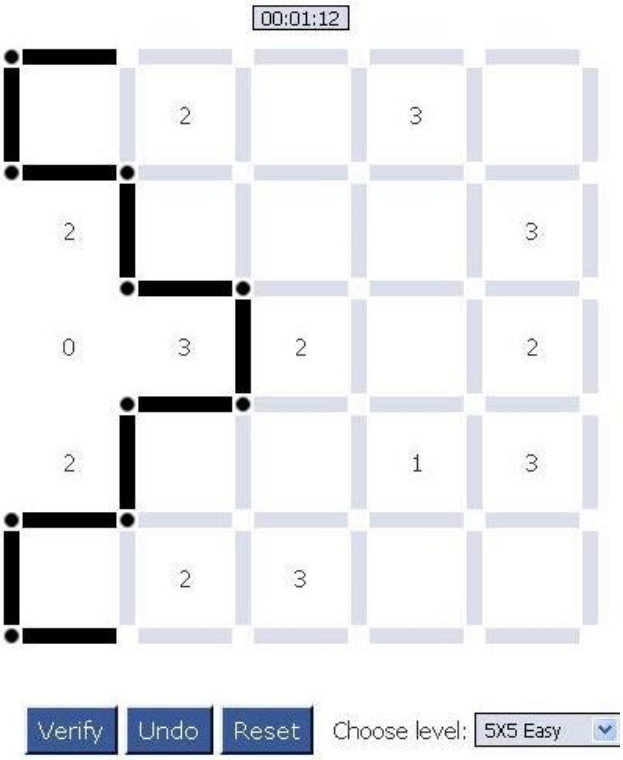
Salah satu contoh *problem* yang termasuk kategori mudah adalah *problem* yang didalamnya terdapat angka 0.



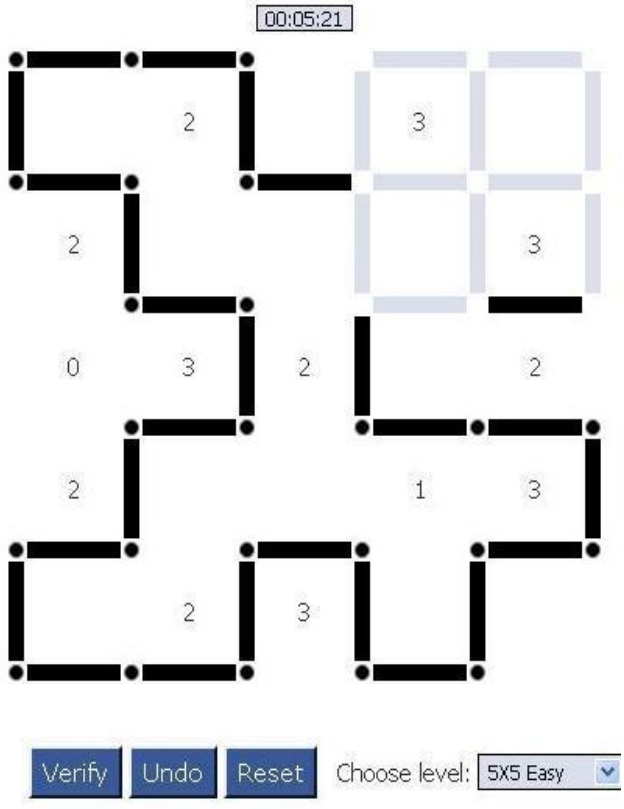
Langkah pertama adalah menghilangkan semua kemungkinan di sekitar angka 0.



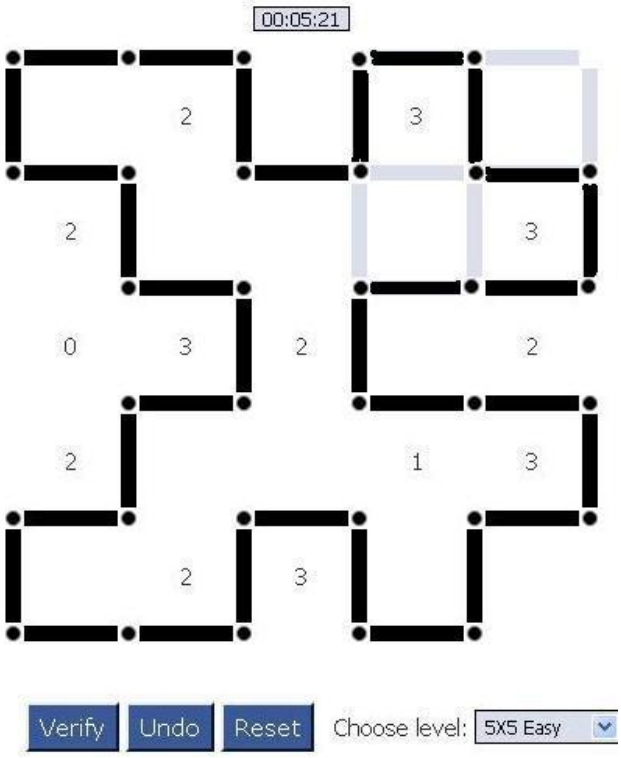
Langkah kedua, adalah menghilangkan kemungkinan yang menyebabkan tidak terjadinya suatu *loop*. Ketika menghilangkan kemungkinan tersebut, maka 'kotak 2' di atas dan di bawah 'kotak 0' harus diisi. Karena hanya tersisa 2 garis saja. Kemudian teruskan sampai ada kemungkinan garis tersebut bercabang.



Langkah ketiga adalah menghilangkan kemungkinan-kemungkinan garis bercabang, karena garis-garis tersebut harus membentuk suatu *loop* yang besar.

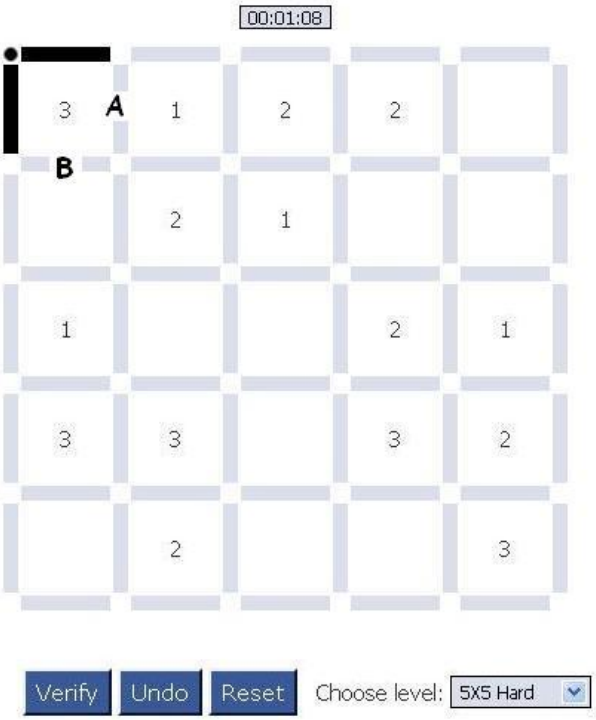


Langkah kedua dan ketiga diulangi terus sampai selesai.



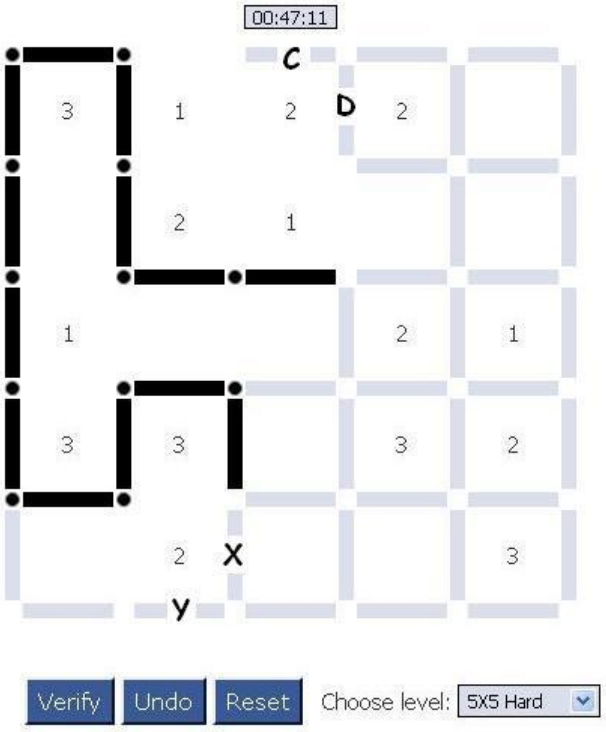
Untuk kasus '*easy problem'* dengan metode tersebut dapat terselesaikan dengan mudah. Dikarenakan ada angka 0 dan susunan angka serta kotak yang masih relatif mudah untuk dilanjutkan dengan mengulangi kedua langkah diatas. Untuk selanjutnya akan dicoba dengan menggunakan contoh '*hard problem'*.

Berikut salah satu contoh '*hard problem'*.

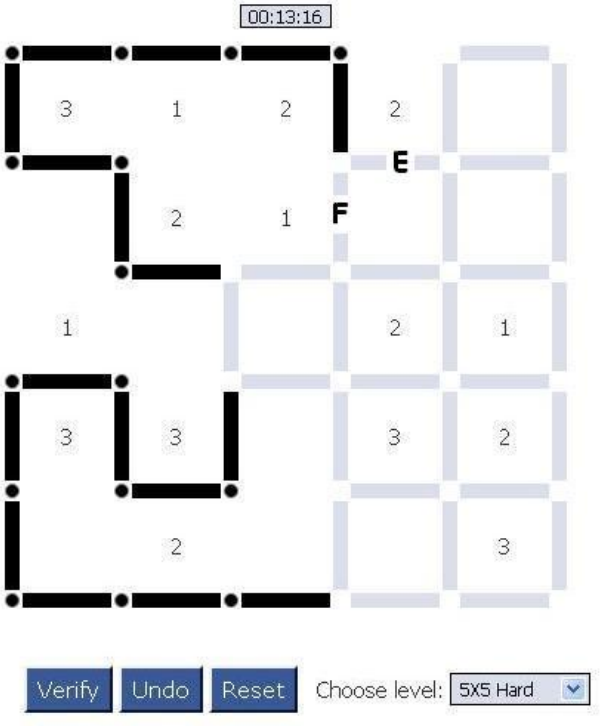


Karena tidak ada kotak 0, jadi algoritma dimulai dari kotak paling kiri atas. Misal algoritma *Brute Force* memilih A terlebih dahulu. Menggunakan kedua langkah

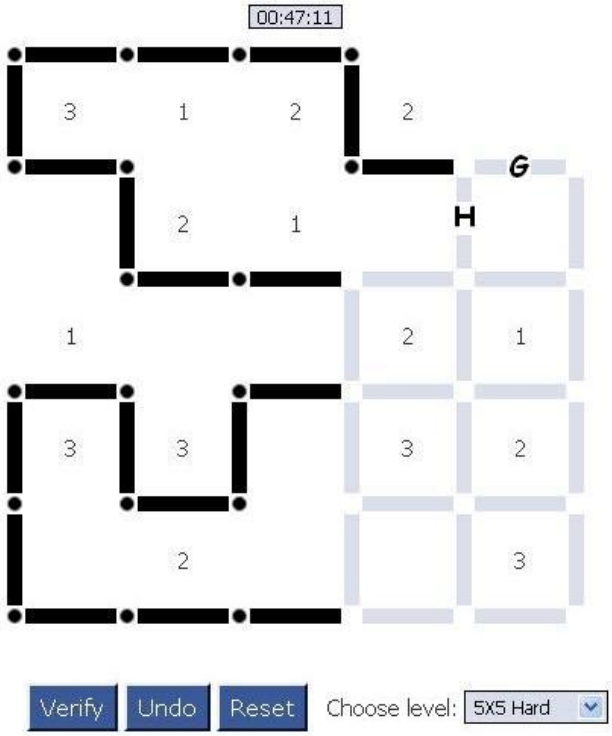
metode sebelumnya yang diulang terus-menerus akan didapat :



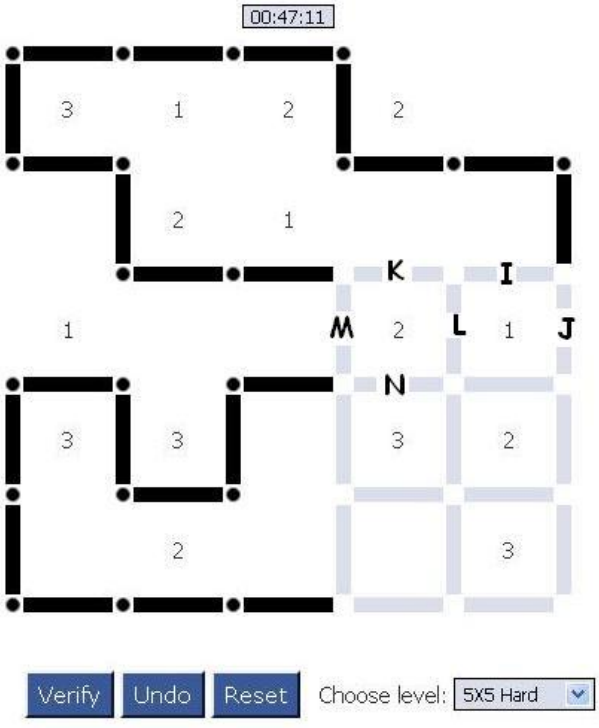
Garis X dan Y harus diambil, tetapi hal ini mengakibatkan tidak dapat membuat suatu *loop* dengan baik. Sama halnya dengan garis C dan D. Karena *stuck*, maka disinilah algoritma *Backtracking* diperlukan. Kondisi dirunut-balik sampai dengan kondisi pemilihan garis A pada gambar sebelumnya. Selanjutnya, akan dicoba dengan memilih garis B sebagai solusi.



Permasalahan lagi muncul memilih antara 2 pilihan yaitu garis E atau garis F yang akan dipilih sebagai solusi. Sesuai *Brute Force*, kita akan mengambil garis E terlebih dahulu sebagai solusi *problem* tersebut. Jika diteruskan maka akan terlihat seperti gambar dibawah.



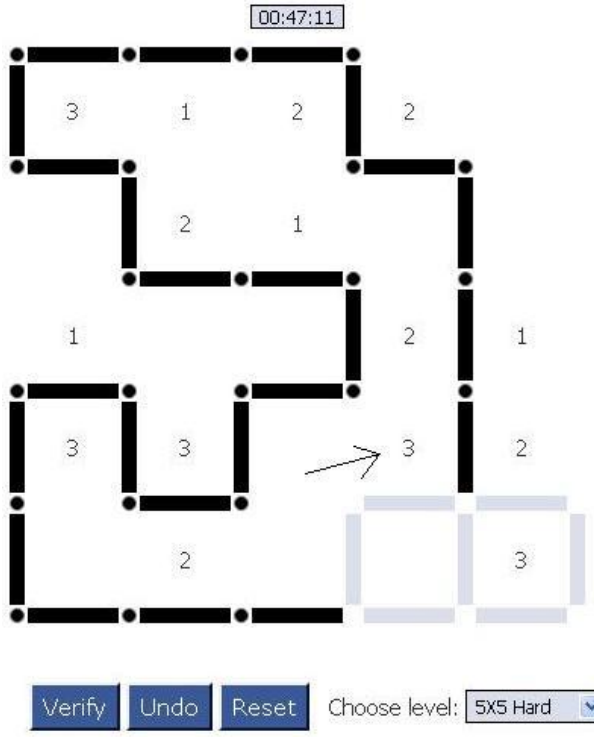
Kembali muncul masalah antara dua pilihan. Misal kita pilih garis G sebagai salah satu solusi.



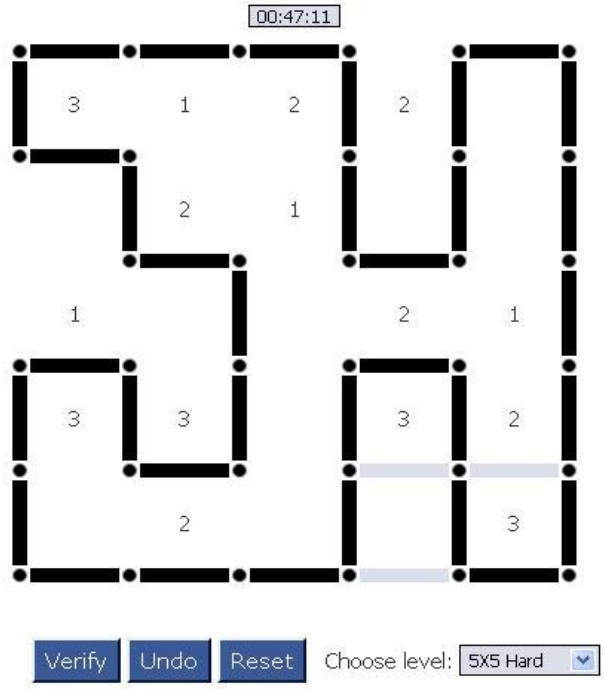
Disini jika kita memilih I sebagai solusi, maka mau tidak mau kita harus mengambil garis K juga. Sehingga berakibat terjadinya 2 buah *loop*. Padahal dalam peraturan disebutkan tidak boleh ada 2 atau lebih *loop*. Maka dari itu kita *Backtrack* pengambilan garis I, lalu memilih garis J.

Namun, pemilihan garis J juga mengakibatkan garis I, L, dan K menjadi tidak boleh diambil. Sehingga pada

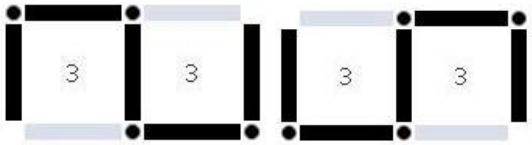
kotak 2 yang dikelilingi garis KLMN, hanya tersisa garis M dan N saja. Hal ini jelas bertentangan karena garis tidak boleh bercabang. Lakukan *Backtrack* lagi sebelum kita memilih garis G sebagai solusi, kita ganti dengan memilih garis H sebagai solusi.



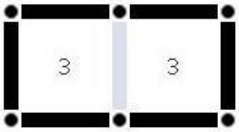
Pemilihan garis H mengakibatkan kotak 3 yang ditunjuk tidak dapat membuat tepat 3 buah garis disekitar angka tersebut. Sehingga harus di-*backtrack* lagi sampai dengan sebelum memilih garis E sebagai solusi, dan menggantinya dengan garis F.



Akhirnya selesai juga salah satu dari '*hard problem'*. Sebenarnya, dalam pengerjaan suatu *problem* dalam **Slitherlink** ini ada beberapa trik. Misalnya jika ada kotak 3 bertetanggaan satu sama lainnya, harus dibuat sedemikian rupa agar membentuk angka 2 atau angka 5 tiduran.



Dari contoh di atas, dapat ditarik kesimpulan jika ada 2 buah kotak 3 berdekatan, pasti garis-garis sejajarnya merupakan solusi. Karena jika tidak maka akan membentuk suatu *loop* kecil yang menyalahi aturan.



Namun tidak akan dibahas semua trik-trik yang ada, karena pada dasarnya algoritma *Brute Force* adalah algoritma sederhana yang memerlukan waktu dan proses yang cukup lama tetapi pasti menghasilkan suatu *result*.

4. KESIMPULAN

- Hampir seluruh permainan logika dapat diselesaikan dengan algoritma *Brute Force* + *Backtracking*.
- Algoritma *Brute Force* sebenarnya dapat dilengkapi dengan beberapa trik lain yang dapat membantu penyelesaian menjadi lebih cepat, namun hal tersebut pasti menambah *cost* dari proses pencarian solusi.

5. DAFTAR REFERENSI

[1] http://www.nikoli.co.jp/en/puzzles/slitherlink/ terakhir diakses 8 Desember 2010,

[2] http://www.facebook.com/looppuzzle terakhir diakses 8 Desember 2010,

[3] http://en.wikipedia.org/wiki/Slitherlink terakhir diakses 8 Desember 2010,

[4] Ir. Rinaldi Munir, M.T. , Diktat Kuliah IF3051 Strategi Algoritma, STEI – ITB, 2009.

6. PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 29 April 2010
Ttd

Kevin Chandra Irwanto
13508063