ISSN : 1978-6603

JURNAL Ilmiah SAINTIKOM Sains dan Komputer

ALGORITMA BACKTRACKING SEBAGAI SOLUSI GAME WORD SEARCH PUZZLE BERBASIS JAVA MOBILE

***Azanuddin[#1], Iskandar Zulkarnaen[#2],Purwadi[#3]**
[#1,2,3] Program Studi Sistem Informasi, STMIK Triguna Dharma
E-Mail : [#1] azdin.bpc@gmail.com

Abstrak

Game Word Search Puzzle merupakan *game* berbasis *puzzle* untuk mencari kata yang disusun dalam bentuk *array* dua dimensi atau yang lebih dikenal dengan matriks. Kata-kata tersebut dapat disusun secara *horizontal*, *vertikal* maupun *diagonal* dan dapat ditulis dalam posisi terbalik maupun tidak. *Game Word Search Puzzle Mobile* dibuat dengan menggunakan bahasa JAVA™, yaitu J2ME, dan akan berjalan pada *ponsel* berbasis JAVA™ MIDP 2.0.
Algoritma *backtracking* adalah algoritma yang berbasis pada algoritma DFS untuk mencari solusi persoalan secara lebih tepat. Algoritma ini merupakan perbaikan dari algoritma *brute force* yang memeriksa semua kemungkinan solusi yang ada. Dengan algoritma *backtracking*, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi yang akan dipertimbangkan. Simpul-simpul yang tidak mengarah ke solusi akan dipangkas. Sehingga waktu pencarian solusi dapat dihemat.

Kata Kunci *: Backtracking, Simpul dan game.*

Abstract

Game Word Search Puzzle is a puzzle-based game to search for words arranged in the form of two-dimensional array or better known as matrix. The words can be arranged horizontally, vertically or diagonally and can be written in reverse or not. Word Search Puzzle Mobile games are built using JAVA™ language and will run on JAVA™ MIDP 2.0 based phones.
Backtracking algorithm is an algorithm based on DFS algorithm to find the solution of the problem more precisely. This algorithm is an improvement of the brute force algorithm that examines all possible solutions. With backtracking algorithm, we do not need to check all possible solutions. Only searches that lead to the solution will be considered. Nodes that do not lead to the solution will be trimmed. So the search time of the solution can be saved.

Keywords*: Backtracking, nodes dan games.*

I. PENDAHULUAN

1. Latar Belakang

Bermain *game* merupakan salah satu aktifitas yang sangat disukai oleh sebagian besar masyarakat. Alasan mereka bermain *game* tentunya berbeda-beda, ada yang untuk melepas lelah, ada juga yang memang suka atau hobi bermain *game*. *Game* bukan hanya sebagai pengisi waktu senggang, tetapi juga sebagai sarana hiburan bagi sebagian besar orang yang memiliki banyak kesibukan. Salah satu *game* yang banyak digemari saat ini adalah *game* pada *mobile*, diantaranya adalah *Game Word Search Puzzle* (pencarian kata). Sebagian besar orang memainkan *game* ini dengan hanya mengandalkan penglihatan dan intuisi untuk menemukan kata-kata yang tersembunyi pada matriks permainan.

Word Search Puzzle (pencarian kata) adalah salah satu jenis *game* yang sangat popular. Objektifnya *game* ini menebak kata dengan karakter-karakter yang telah diacak, dan di bagian lain terdapat kata-kata yang harus dicari sesuai dengan karakter-karakter yang telah diacak. Untuk mencari keseluruhan kata yang ada memang tidak mudah, karena pemain harus menemukan semua kata yang tersembunyi di matriks permainan. Kata-kata tersebut dapat disusun secara *horizontal, vertical* maupun *diagonal* dan dapat ditulis dalam posisi terbalik maupun tidak. Permainan ini cukup menarik karena pemain diajak untuk teliti dalam mencari kata-kata diantara serangkaian huruf yang membentuk matriks. Oleh karena itu aplikasi *game* ini menyediakan sarana pencarian solusi secara otomatis.

Game mobile saat ini telah menjadi bagian yang tidak terpisahkan dari hampir setiap peralatan bergerak. Menurut perusahaan *Sun Microsystem*, *game mobile* memiliki peminat yang tertinggi diantara layanan-layanan lainnya. Sehingga, *game mobile* dapat menjadi pangsa pasar yang menjanjikan.

Algoritma *backtracking* (runut-balik) adalah algoritma yang berbasis pada algoritma DFS (*Depth First Search*). Algoritma ini merupakan perbaikan dari algoritma *brute force* yang memeriksa semua kemungkinan solusi yang ada. Dengan algoritma *backtracking,* tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi yang akan dipertimbangkan. Simpul-simpul yang tidak mengarah ke solusi akan dipangkas. akibatnya, waktu pencarian solusi dapat dihemat.

Oleh karena itu dengan di terapkannya algoritma *backtracking* pada *mobile* yang akan dipasangkan pada perangkat *game*, dapat dijadikan sebagai jalan alternatif dan mempermudah pencarian kata dalam menyelesaikan *game word search puzzle*.

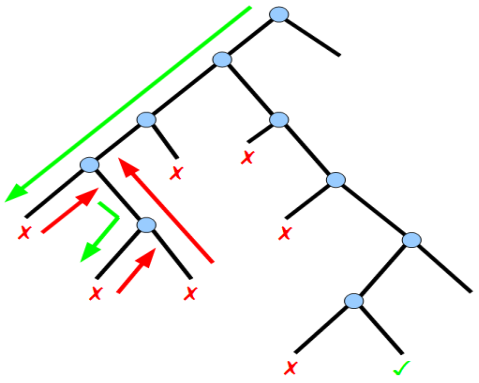
II. KAJIAN PUSTAKA

2.1 Algoritma Backtracking

Algoritma *Backtracking* pertama kali diperkenalkan oleh D.H Lehmer pada tahun 1950. Dalam perkembanganya beberapa ahli seperti RJ Walker, Golomb dan Baumert menyajikan uraian umum tentang *Backtracking* dan penerapannya dalam berbagai persoalan dan aplikasi.

Algoritma runut-balik (*Bactracking Algorithm)* adalah algoritma yang berbasis pada DFS (*Depth First Search*) yaitu semua solusi dibuat dalam bentuk pohon solusi

(*tree*), dan kemudian pohon tersebut akan ditelusuri secara DFS untuk mencari persoalan secara lebih mangkus. Runut-balik yang merupakan perbaikan dari algoritma *brute force*, secara sistematis mencari solusi persoalan diantara kesemua kemungkinan solusi yang ada. Dengan menggunakan metode runut-balik tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan. Akibatnya, waktu pencarian dapat dihemat. Runut-balik merupakan bentuk tipikal dari algoritma rekursif.



Gambar 2.1 Contoh Nodes Backtracking

Saat ini algoritma runut-balik banyak diterapakan untuk program *games* (seperti permainan *tic-tac-toe,* merupakan jalan keluar dalam sebuah labirin, catur, dll) dan masalah-masalah pada bidang kecerdasan buatan (*artificial intelligence*). (Rinaldi Munir, 2007)

Algoritma *backtracking* memiliki tiga properti utama dalam penerapannya yaitu :

1. Solusi Persoalan
   Solusi dinyatakan sebagai *vector* dengan *n-tuple*. $X=(x_{1,}x_{2,\ldots,}x_n)$, x_i himpunan berhingga S_i
2. Fungsi pembangkit
   Fungsi pembangkit digunakan untuk membangkitkan nilai X_k yang merupakan komponen vektor solusi. Fungsi pembangkit dinyatakan sebagai : T(k)
3. Fungsi pembatas
   Fungsi pembatas digunakan untuk menentukan apakah (x1, x2, …, xk) mengarah ke solusi atau tidak. Jika ya, pembangkitan nilai dilanjutkan. Jika tidak, (x1, x2, …, xk) dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi. Fungsi pembangkit dinyatakan sebagai : $B(x_{1,}x_{2,\ldots,}x_k)$

Prinsip Pencarian Solusi Dengan Algoritma Backtracking :

1. Solusi dicari dengan membentuk lintasan dari akar kedaun, aturan pembentukan yang dipakai adalah mengikuti aturan pencarian mendalam (DFS). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (*live node*), simpul hidup yang sedang diperluas dinamakan simpul E (*expand node*).
2. Tiap kali simpul E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibangun tidak mengarah kesolusi maka simpul E tersebut dibunuh, sehingga menjadi simpul mati (*dead node*). Fungsi yang digunakan untuk membunuh simpul E adalah dengan menerapakan fungsi pembatas (*bounding finction)* simpul yang sudah mati tidak akan diperluas kembali.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan dengan simpul anak lainnya. Bila tidak ada simpul anak yang bisa

dibangkitkan, maka pencarian solusi dilanjutkan dengan melakukan runut-balik ke simpul hidup terdekat (simpul orang tua), selanjutya simpul ini akan menjadi simpul yang baru, pencarian dihentikan bila telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut-balik.

4. Pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut-balik.

2.2 Game Word Search Puzzle

Game Word Search Puzzle telah ada sejak dahulu dan semakin berkembang. Karena *game* ini mudah dimainkan tetapi juga cukup menantang, maka *game* ini menjadi cukup populer di kalangan para pengguna *mobile atau handphone*. Tujuan *game* ini adalah mencari kata-kata pada sebuah matriks *array* dua dimensi. Kata-kata yang dicari adalah kata-kata dalam bahasa Inggris. Oleh karena itu, *game* ini juga bermanfaat menambah perbendaharaan kata dalam bahasa Inggris. Kata-kata yang akan dicari telah ditentukan sebelumnya pada seranai kata yang tertulis di luar matriks. Huruf-huruf di dalam matriks dapat tersusun sehingga menjadi sebuah kata secara *horisontal*, *vertikal*, maupun *diagonal* dengan arah kiri ke kanan, kanan ke kiri, atas ke bawah, bawah ke atas, kiri bawah ke kanan atas, kanan bawah ke kiri atas, kiri atas ke kanan bawah, serta kanan bawah ke kiri atas.

Pemain dapat memainkan permainan ini dengan membuat suatu garis melewati huruf-huruf pada matriks. Garis dapat diletakkan di huruf awal sampai ke huruf akhir dengan arah *vertikal*, *horisontal*, maupun *diagonal*. Huruf-huruf yang dilewati garis tersebut harus membentuk kata yang sesuai dengan salah satu kata pada kata di sebelah kanan matriks. Apabila benar, kata yang bersesuaian pada kata di sebelah kanan martiks akan ditandai dengan garis. Pemain dinyatakan menang apabila pemain dapat menemukan seluruh kata yang terdapat di matriks sebelah kanan.

2.3 Mobile Device

Mobile Device merupakan suatu alat yang digunakan oleh *pemakai* untuk meminta informasi yang dibutuhkan, dimana informasi dapat diberikan dalam bentuk suara, gambar, dan *text* dimana informasi yang diinginkan dapat dicari melalui fasilitas untuk mengakses *internet* seperti GPRS atau *wireless*.

Pada umumnya perangkat *mobile* atau *Mobile device* lebih praktis karena bersifat mudah dibawa *(portable)* jika dibandingkan dengan perangkat-perangkat teknologi yang lainnya.

Meningkatnya pemakaian peranti *mobile* (*mobile* device) telah merevolusi kegiatan-kegiatan yang bersifat tradisional menjadi lebih sederhana dan mudah dengan penggunaan perangkat *mobile*. Mobilitas yang tinggi tidak menjadi penghalang lagi, karena saat ini peranti *mobile* sudah dapat mengakses *sever* di pusat data. Peranti mobile sekarang tidak hanya berfungsi sebagai pencatat jadwal dan buku alamat. Fungsi peranti *mobile* sudah berkembang pesat dan idealnya siap mengganti dokumen berbasis kertas. (Tri Mardiono, 2006, chap.2)

Vendor-vendor peranti *mobile* telah menanam teknologi penangkap data *(data acquisition)* ke dalam produknya. Hal ini menciptakan

peningkatan produktivitas, proses pelaporan yang lebih cepat, dan pengurangan biaya operasional. Secara umum aplikasi peranti *mobile* terbagi atas:

1. *Personal Information Managenent* (PIM)
   Menyediakan fungsi kalender, buku alamat, jadwal, memo, pengirim *email*, dan tugas yang harus dilakukan.
2. Dokumen
   Menyediakan fungsi *word processing* dan *spread sheet.* Fungsi yang lebih maju antara lain : akses *web*, *e-book*, multimedia, dan presentasi.
3. Aplikasi M*obile Business (mBusiness)*
   Menyediakan fungsi penangkapan, pemrosesan, dan pengiriman data. Komunikasi yang umum antara server dan aplikasi adalah melalui sistem *messaging* seperti SMS dan sebagainya. Namun sekarang aplikasi *business* sudah dapat terhubung langsung dengan *server* melalui HTTP secara *prorietary* atau melalui *web service*. Dengan J2ME peranti *mobile* yang menggunakan CDC bisa melakukan koneksi berbobot melalui RME, JDBC dan sebagainya.
4. Game dan hiburan
   Menyediakan fungsi permainan, musik, video dan sebagainya. J2ME menyediakan *library-library* untuk membangun aplikasi *games* dan hiburan.(Tri Mardiono, 2006, chap.2)

III. METODE PENELITIAN

Didalam penelitian ini terdapat beberapa metode yang digunakan dalam penyelesaian penelitian ini diantaranya :

1. Studi literatur
   Pada studi literatur dicari bahan-bahan untuk referensi yang sejalan dengan penelitian diantaranya menggunakna buku, jurnal ilmiah nasional maupun internasional serta bebrapa sumber referensi dari internet untuk memperkuat teori dasar serta hasil yang optimal dalam penelitian unutk untuk kasus penerapan backtracking untuk game berbasis mobile.
2. Observasi
   Pada metode observasi ini dilakukan tahapan menganalisa game word search puzzle yang ada dengan mempelajari rancang bangun serta algoritma pemrograman yang ada, hal ini digunakan agar dapat memastikan kebenaran dalam proses *solve* (memecahkan / solusi) pengacakan kata yang dicari pada *game word search puzzle* dengan memahami aturan / *rule* yang ada.
3. *Testing* dan *Implementation*
   Pada tahap testing (pengujian) dan implementasi ini dilakukan tahapan penerapan backtracking pada salah satu button atau tombol kemudaian pada tahap pengujian dilakukan penekanan button atau tombol untuk mengetahui proses pencarian kata apakah sudah sesuai dengan soal yang akan diselesaikan.

IV. HASIL DAN PEMBAHASAN

4.1 Analisa Permasalahan

Game Word Search Puzzle telah ada sejak dahulu dan semakin berkembang. Karena *game* ini mudah dimainkan tetapi juga cukup menantang, maka *game* ini menjadi cukup populer di kalangan para

pengguna *mobile*. Tujuan *game* ini adalah mencari kata-kata pada sebuah matriks *array* dua dimensi pada papan permainan. Kata-kata yang dicari adalah kata-kata dalam bahasa Inggris. Oleh karena itu, *game* ini juga bermanfaat menambah perbendaharaan kata dalam bahasa Inggris. Kata-kata yang akan dicari telah ditentukan sebelumnya pada serangkai kata yang tertulis di luar matriks.

Aturan dalam permainan ini adalah pemain dapat memainkan permainan ini dengan membuat suatu garis melewati huruf-huruf pada matriks. Garis dapat diletakkan di huruf awal sampai ke huruf akhir dengan arah *vertikal*, *horisontal*, maupun *diagonal*. Huruf- huruf yang dilewati garis tersebut harus membentuk kata yang sesuai dengan salah satu kata pada kata di sebelah kanan matriks. Apabila benar, kata yang bersesuaian pada kata di sebelah kanan martiks akan ditandai dengan garis. Pemain dinyatakan menang apabila pemain dapat menemukan seluruh kata yang terdapat di matriks sebelah kanan.

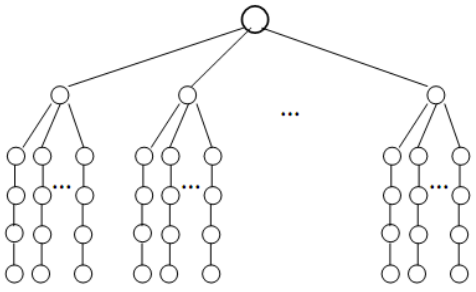
Dalam algoritma *backtracking* semua kemungkinan solusi dari persoalan disebut ruang solusi (*solution space*). Ruang solusi diorganisasikan ke dalam struktur pohon, tiap simpul pohon menyatakan status (*state*) persoalan, sedangkan sisi (cabang) dilabeli dengan nilai-nilai x_i. Lintasan dari akar ke daun menyatakan solusi yang mungkin, seluruh lintasan dari akar ke daun membentuk ruang solusi, pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status (*state space tree*).

Dalam melakukan pencarian satu kata, pemain harus mencari melalui delapan jalur yang mungkin, yaitu:

a. *horizontal* ke kanan
b. horizontal ke kiri
c. vertikal ke atas
d. *vertikal* ke bawah
e. *diagonal* ke kiri atas
f. *diagonal* ke kiri bawah
h. *diagonal* ke kanan atas
i. *diagonal* ke kanan bawah

Jadi untuk pencarian huruf per huruf dari kata yang dicari, kita harus melihatnya dari delapan jalur tersebut.

Solusi persoalan dari permainan *word search puzzle* adalah *vektor* n-*tuple* dimana tiap komponennya adalah huruf pada posisi tertentu pada papan permainan. Ruang solusi dari persoalan ini adalah semua kemungkinan kata yang dapat diperoleh di papan permainan dengan memeriksa ke delapan jalur yang telah disebutkan diatas. Dengan kata lain, untuk matriks berukuran NxN, terdapat maksimal $8N^2$ kemungkinan solusi persoalan. Disebut maksimal karena tidak semua posisi pada papan permainan memiliki tepat delapan jalur pencarian, seperti huruf-huruf pada pinggir papan permainan. Pohon ruang status statis pada program pencarian solusi ini dapat dilihat pada gambar berikut



Gambar 4.1 Pohon Simpul untuk word search puzzle

Tiap simpul pohon (*state*) menyatakan kumpulan huruf telah ditemukan yang berada pada posisi-posisi tertentu. Sisi (cabang) menyatakan huruf selanjutnya pada posisi tertentu yang sedang diperiksa. Lintasan dari akar ke daun

menyatakan solusi yang mungkin. Solusi dicari dengan membentuk lintasan dari akar ke daun dengan mengikuti algoritma DFS. Pada program pencarian solusi yang dibuat, pencarian dimulai dari huruf pada bagian kiri atas papan permainan dan simpul selanjutnya dibangkitkan sesuai algoritma DFS untuk kemudian diperiksa kecocokannya.

Langkah-langkah pencarian solusi pada program ini adalah sebagai berikut

1. Simpul akar pada pohon ruang status merupakan inisialisasi dan menyatakan pencarian huruf awal dari kata yang ingin dicari di papan permainan. Pencarian posisi-posisi ini dilakukan dari bagian kiri atas hingga bagian kanan bawah papan permainan. Pencarian ini akan menghasilkan posisi-posisi pada papan permainan yang berisi huruf yang sama dengan huruf awal pada kata yang ingin dicari.
2. Jika pencarian pada langkah (a) tidak menemukan satupun posisi pada papan permainan yang berisi huruf awal dari kata yang dicari, berarti kata tersebut tidak ada di papan permainan.
3. Jika pencarian pada langkah (a) menemukan posisi yang tepat, pencarian huruf selanjutnya akan dimulai dari posisi tersebut.
4. Pembangkitan simpul-simpul dari posisi tersebut dilakukan dengan mengikuti algoritma DFS. Simpul dibangkitkan dengan urutan jalur dari atas, kanan-atas, kanan, kanan-bawah, bawah, kiri-bawah, kiri, kiri-atas.
5. Pembangkitan simpul selanjutnya dilakukan dengan berdasarkan pada fungsi pembatas. Terdapat tiga fungsi pembatas pada program ini, yaitu :
   a. Jika pembangkitan suatu simpul telah memilih suatu jalur, pembangkitan simpul selanjutnya untuk lintasan tersebut dilakukan menurut jalur tersebut seperti dapat dilihat pada pohon ruang status. Sebagai contoh, jika telah memilih jalur atas, maka pembangkitan simpul selanjutnya untuk lintasan tersebut hanya untuk jalur atas saja.
   b. Jumlah huruf yang terdapat dalam kata yang ingin dicari. Dengan kata lain, jika ingin memeriksa suatu jalur, terlebih dahulu dilakukan pengecekan apakah pada jalur tersebut dapat diperoleh jumlah huruf yang sama dengan jumlah huruf pada kata yang ingin dicari. Sebagai contoh, pada program yang dibuat, jika sedang berada di posisi kiri-atas pada papan permainan, jalur yang bisa dibangkitkan pada posisi ini hanya jalur kanan, kanan-bawah, dan bawah saja.
   c. Kecocokan huruf pada suatu posisi di papan permainan dengan huruf pada kata yang ingin dicari.
6. Pencarian berhasil jika menemukan kata yang dicari pada pembangkitan simpul-simpul pada proses pencarian melalui suatu jalur (fungsi pembatas selalu mengembalikan nilai *true* pada pembangkitan simpul),

7. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi (fungsi pembatas mengembalikan nilai *false*), simpul tersebut "dibunuh" sehingga menjadi simpul mati (*dead node*) dan tidak akan diperluas lagi.
8. Jika pembentukan lintasan berakhir dengan simpul mati, proses pencarian solusi dilanjutkan dengan melakukan *backtracking* ke simpul orangtua. Dalam hal ini, *backtracking* dilakukan hingga kembali ke simpul pada langkah (d) atau dengan kata lain simpul orangtua *level* 1 pada pohon ruang status (asumsi : simpul akar memiliki *level* 0). Selanjutnya simpul ini menjadi simpul-E yang baru dan simpul anak berikutnya dibangkitkan sesuai urutan priotitas jalur yang telah disebutkan sebelumnya.
9. Jika simpul *level* 1 yang diperoleh dari langkah (h) tidak bisa diekspansi lagi (semua jalur telah diperiksa dan tidak ada yang memenuhi), pencarian solusi dilakukan dengan *backtracking* ke simpul akar. Dengan kata lain, proses dimulai kembali dari langkah (a) dengan mencari posisi lain dari huruf awal kata yang ingin dicari di papan permainan.

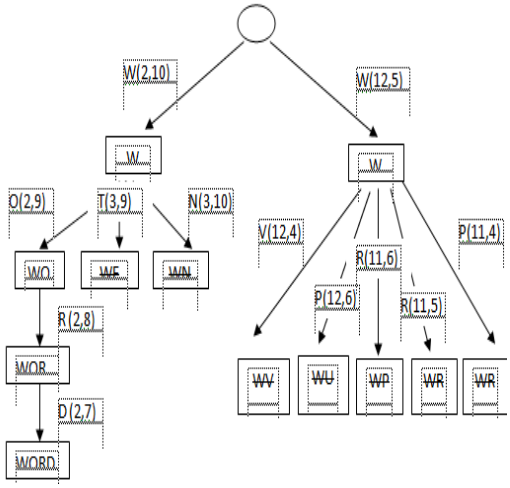
Untuk melakukan analisis terhadap penerapan algoritma *backtracking* dalam game *word search puzzle*, penulis merancang aplikasi dengan menggunakan matriks 13 x 11, dimana terdapat 143 huruf dalam matriks tersebut. Untuk mencari kata dalam papan permainan, pengguna hanya cukup menekan tombol *solve* maka semua solusi akan ditampilkan.

Berikut tampilan papan dari matriks yang akan diselesaikan, sebagai contoh penulis akan mencari kata "**WORD**"

Tabel 4.1 Matriks Soal Word Search

A	O	H	A	I	X	K	N	X	C	D	O	F	
R	B	O	J	N	P	E	A	N	U	T	K	J	
A	T	A	A	Q	P	L	S	R	D	Q	N	H	
N	C	F	O	P	C	H	E	V	Y	R	V	N	
I	J	G	Y	O	O	X	H	N	T	R	W	X	**W**
H	S	S	Q	K	A	H	A	P	A	P	U	L	**O**
C	D	E	N	D	S	E	S	T	I	K	J	R	**R**
C	R	I	E	M	M	R	H	I	X	K	A	P	**D**
A	O	F	E	F	J	E	V	A	B	O	P	A	
C	W	N	A	K	R	P	F	Q	G	G	A	G	
J	K	R	T	S	J	C	Q	Q	T	X	N	E	

Maka pohon ruang status dinamisnya sebagai berikut :



Gambar 4.2 Pohon Statis Pencarian

4.2 Algoritma Backtracking

Algoritma dapat didefenisikan sebagai urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis. Langkah-langkah tersebut harus logis, berarti nilai kebenarannya harus dapat ditentukan benar atau salahnya. Langkah-langkah yang tidak benar dapat memberikan hasil yang salah. Berikut Algoritma Backtracking secara rekursif :

INPUT : tupple/vector = (k)
OUTPUT : solusi = x(1:n)
PROSES :
k = 1
SELAMA k > 0 **LAKUKAN PERULANGAN**
JALUR (Part)
JIKA ada x(k) yang belum dicoba sedemikian sehingga
x(k) elemen T(x(1),……,x(k-1))
DAN
Bk (x(1),….x(k)) = **TRUE**
MAKA
JIKA (x(1),……x(k)) = adalah sebuah jalur (path) yang merupakan solusi
MAKA
CETAK (x(1),……,x(k))
AKHIR JIKA
k = k +1
SELAINNYA
k = k – 1
AKHIR JIKA
AKHIR PERULANGAN

4.3 Hasil

Pada Hasil ditampilkan dari proses pencarian dengan Backtracking. Pada game word search puzzle disediakan tombol solve yang telah diterapkan algoritma backtracing seperti gambar dibawah ini

V. KESIMPULAN

Sebagai penutup sajian pembahasan dalam penulisan dapat diambil kesimpulan – kesimpulan sekaligus memberikan saran untuk memajukan sistem yang dibuat, dengan adanya kesimpulan dan saran ini dapatlah diambil suatu perbandingan yang akhirnya dapat memberikan perbaikan - perbaikan pada masa yang akan datang. Adapun kesimpulan yaitu sebagai berikut:

1. Proses dan aturan pada *game word search puzzle* adalah kata yang tertulis diluar matriks kemudian huruf di dalam matriks dapat tersusun sehingga menjadi sebuah kata secara *horizontal*,

vertikal, *diagonal* dengan arah dari kiri ke kanan, kanan ke kiri, atas ke bawah, bawah ke atas, kiri bawah ke kanan atas, kanan bawah ke kiri atas, kiri atas ke kanan bawah serta kanan bawah ke kiri atas.

2. Spesifkasi *handphone* yang dapat menjalankan *game word search puzzle* adalah *handphone* yang berbasis java™ yang mendukung MIDP 2.0
3. Algoritma *backtracking* dapat diterapkan dalam *game word search puzzle*, sehingga pemain dapat dengan mudah menemukan kata yang dicari pada luar matriks, algoritma *backtracking* juga dapat mencari kata secara *horizontal*, *vertical* dan *diagonal* serta dengan Algoritma *backtracking* tidak perlu memeriksa semua kemungkinan solusi yang ada hanya pencarian yang mengarah ke solusi yang akan dipertimbangkan.
4. Untuk selanjutnya diharapkan aplikasi seperti ini dapat dikembangkan tidak hanya pada *mobile device* berbasis JAVA™ , tetapi dengan *mobile device* berbasis apapun seperti ANDROID.

DAFTAR PUSTAKA

Adi Nugroho. (2010). Rekayasa Perangkat Lunak Berorientasi Objek dengan Metode USDP (Unified Software Development Process), Yogyakarta : Penerbit C.V Andi Offset.

Alvin Andhika Zulen. (2009) Penerapan Algoritma Backtracking Pada Permainan Word Search Puzzle, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung,

Jogiyanto, H.M. (2001). Analisa dan Desain Sistem Informasi, Yogyakarta : Penerbit Andi Offset.

M Shalahuddin & Rossa.(2008). Pemrograman J2ME, Bandung : Penerbit Informatika.

Tri Mardiono. (2006). Membangun Solusi Mobile Business dengan Java, Jakarta : Penerbit PT. Elex Media Komputindo.

Yuniar Supardi. (2008). Pemrograman Handphone dengan J2ME, Jakarta : Penerbit PT. Elex Media Komputindo.

Munir Rinaldi "Diktat Kuliah IF3051 Strategi Algoritma" Institut Teknologi Bandung, 2009

Gunawan, Agustinus Tri 2010, Penerapan Algoritma Backtracking dan Elimination untuk Membangun Generator dan Solver dalam Menyelesaikan Permainan Sudoku, Sekolah Tinggi Manajemen Informatika dan Komputer AMIKOM, Yogyakarta.

Morenvino, M. Ray, I. dan Anton, S. 2006. Penerapan Algoritma RunutBalik Untuk Penyelesaian Teka-Teki Sudoku, Lab Ilmu dan Rekayasa Komputasi Depertemen Teknik Informatika Institut Teknologi Bandung, Bandug