

Penerapan Algoritma Backtracking Dengan Metode Rekursif Pada Game Word Search

Application of Backtracking Algorithm with Recursive Method in Word Search Game

M Iqbal Anshori¹, Ilham Fahri H², Moch Rafly Pratama³, Asti Herliana⁴

^{1,2,3}Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Universitas Adhirajasa Reswara Sanjaya

⁴Dosen Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Universitas Adhirajasa Reswara Sanjaya
Email: iqbalanshr@gmail.com¹, ilhamfahri397@gmail.com², mochraflypratama05@gmail.com³, asti@ars.ac.id⁴

ABSTRAK

Game word search merupakan game teka-teki untuk pencarian kata di dalam matrik, kata tersebut bisa tersusun secara horizontal, vertical maupun diagonal, proses penyelesaian dalam game word search ini harus menemukan semua kata yang tersembunyi. Algoritma backtracking rekursif dapat menyelesaikan game tersebut dengan lebih efektif dan efisien karena strategi pemecahan masalahnya hanya fokus pada solusi. Metode yang digunakan untuk pengumpulan data ialah studi kepustakaan untuk mengumpulkan informasi terkait yang relevan, analisis game word search yang menjelaskan aturan-aturan dan jalur yang mungkin dilalui, serta Algoritma backtracking rekursif yang menggambarkan langkah-langkah dan skema umum algoritma tersebut. Dalam penelitian ini menjelaskan solusi langkah-langkah penerapan algoritma backtracking rekursif, dengan merujuk pada penelitian terdahulu yang menggunakan algoritma backtracking versi iteratif dan membandingkan penelitian tersebut. Algoritma backtracking versi rekursif dan iteratif memiliki perbedaan dalam penulisan kode dan pengalokasian memori, serta kecepatan waktu eksekusi yang berbeda tergantung kompleksitas kata yang dicari dan ukuran matrik pada game word search.

Kata Kunci: Word search, game, Backtracking, Rekursif

ABSTRACT

The word search game is a puzzle game for finding words in the matrix, the words can be arranged horizontally, vertically or diagonally, the solving process in this word search game must find all the hidden words. Recursive backtracking algorithm can solve the game more effectively and efficiently because the problem-solving strategy only focuses on the solution. The methods used for data collection are library studies to collect relevant related information, word search game analysis that explains the rules and possible paths, and recursive backtracking algorithm that describes the steps and general scheme of the algorithm. This study explains the solution steps of applying the recursive backtracking algorithm, by referring to previous studies that used the iterative version of the backtracking algorithm and comparing these studies. The recursive and iterative versions of the backtracking algorithm have differences in code writing and memory allocation, as well as different execution time speeds depending on the complexity of the word being searched and the size of the matrix in the word search game.

Keywords: Word search, game, Backtracking, Rekursif

Disubmit: 17 Juli

Info Artikel :
Direview: 29 Agustus

Diterima : 24 September

Copyright © 2022 - Journal UPU. All rights reserved.

1. PENDAHULUAN

Di zaman yang serba modern ini kita tidak dapat terlepas dari kata game, game adalah salah satu sarana untuk menghibur diri atau mengisi waktu yang luang, game puzzle merupakan salah satu game yang paling digemari oleh semua kalangan dan salah satu game puzzle yang ada sejak dulu ialah game word search.

Word search puzzle atau game pencarian kata adalah sebuah game teka-teki di mana tujuan utamanya adalah mencari kata-kata yang tersembunyi di dalam sebuah matriks. Kata-kata tersebut dapat berada secara horizontal, vertikal, atau diagonal dengan menggunakan jalur yang terhubung secara horizontal dan vertikal. Dalam proses menyelesaikan permainan ini, kita harus menemukan semua kata yang tersembunyi di dalam sebuah matriks. [1]

Untuk menyelesaikan masalah dalam game word search dengan efektif dan efisien kita dapat mengimplementasikan algoritma backtracking. Algoritma backtracking atau yang juga dikenal sebagai metode runut balik, adalah salah satu strategi pencarian solusi dalam pemecahan masalah. Dalam algoritma ini, tidak perlu memeriksa semua kemungkinan yang ada, melainkan hanya fokus pada jalur yang berpotensi mengarah ke solusi. Algoritma backtracking bekerja secara rekursif, mengadopsi pendekatan Depth-First Search (DFS) untuk melakukan pencarian solusi dengan sistematis pada semua kemungkinan yang ada. Pencarian solusi dilakukan dengan menelusuri struktur pohon berakar. [2]

Algoritma backtracking akan memeriksa huruf awal yang sesuai dengan kata yang dicari. jika sesuai, algoritma akan melanjutkan pencarian huruf ke tahap selanjutnya di sekitar posisi dalam matrix secara rekursif sampai semua huruf dalam kata yang dicari ditemukan, dan jika huruf tidak sesuai algoritma akan melakukan backtrack untuk mencari alternatif lainnya.

Oleh sebab itu, artikel ini akan membahas bagaimana penerapan algoritma backtracking dan cara kerja algoritma tersebut dalam game word search yang akan membantu mencari kata dengan lebih efektif dan efisien.

2. METODE

A. Metode Pengumpulan data

Metode pengumpulan data yang dipakai dalam penelitian ini yaitu dengan studi kepustakaan, dengan mengumpulkan dokumen-dokumen yang relevan seperti jurnal, artikel, makalah dan buku sebagai bahan referensi.

B. Analisis Game Word Search

Dalam game Word Search kita harus mencari kata-kata yang tersembunyi di dalam sebuah matriks yang dapat tersusun secara horizontal, vertical atau diagonal. Untuk lebih jelasnya lagi dalam menganalisis masalah, berikut tampilan dalam game word search.

d	y	o	i	o	m
f	i	y	y	a	d
v	c	a	t	z	o
g	v	o	e	r	g
u	v	b	z	t	q
x	b	a	s	u	s

Gambar 1. Matrik Game Word Search

Dari gambar diatas ada 8 jalur yang mungkin untuk dilalui untuk mencari kata tersembunyi diantaranya:

- 1) Vertical ke atas
- 2) Vertical ke bawah
- 3) Horizontal ke kanan
- 4) Horizontal ke kiri
- 5) Diagonal ke atas kanan
- 6) Diagonal ke atas kiri
- 7) Diagonal ke bawah kanan
- 8) Diagonal ke bawah kiri

Dengan memeriksa jalur-jalur tersebut, kita dapat menemukan kata yang tersembunyi di dalam matrik game word search.

C. Algoritma Backtracking Rekursif

Algoritma Backtracking Rekursif adalah sebuah pengembangan dari metode Depth First Search (DFS). Metode DFS digunakan untuk mencari solusi dari akar ke daun dalam sebuah pohon ruang solusi dengan melakukan pencarian secara mendalam. Prinsip dasar dari algoritma backtracking adalah mencoba semua kemungkinan solusi yang ada. Algoritma Backtracking Rekursif bekerja dengan membangun pohon solusi (tree) yang merepresentasikan semua kemungkinan solusi. Pohon ini kemudian ditelusuri menggunakan metode Depth First Search (DFS) untuk menemukan solusi terbaik yang diinginkan. [3]

Berikut adalah langkah-langkah dalam pencarian algoritma backtracking rekursif:

- 1) Mencari solusi dengan menghubungkan akar ke daun melalui jalur yang terbentuk. Simpul yang sudah dikunjungi disebut simpul hidup, dan simpul hidup yang diperluas untuk melanjutkan pencarian disebut simpul-E (Expand-node).
- 2) Simpul akan dibunuh dan menjadi simpul mati jika jalur yang berasal dari simpul E tidak mengarah ke solusi yang diinginkan.
- 3) Jika terdapat simpul mati, pencarian dilanjutkan dengan mencoba membangkitkan simpul anak (child) lainnya. Jika tidak ada simpul anak, dilakukan backtracking kembali ke simpul induk.
- 4) Pencarian akan dihentikan saat menemukan solusi yang diinginkan atau simpul hidup yang dapat diperluas sudah tidak ada lagi.

Dengan menerapkan metode algoritma backtracking rekursif, kita dapat menemukan solusi pada permasalahan seperti game word search dengan cara yang efektif. Algoritma ini memungkinkan kita untuk menelusuri semua kemungkinan solusi dengan cara yang sistematis dan efisien, sehingga memastikan bahwa tidak ada solusi yang terlewatkan.

Berikut skema umum algoritma backtracking rekursif:

```

procedure RunutBalikR(input  $k$  : integer)
  {Mencari semua solusi persoalan dengan metode runut-balik; skema rekursif}
  Masukan:  $k$ , yaitu indeks komponen vektor solusi,  $x[k]$ . Diasumsikan  $x[1], x[2], \dots, x[k-1]$  sudah
  ditentukan nilainya.
  Luaran: semua solusi  $x = (x[1], x[2], \dots, x[n])$ 
}
Algoritma:
for setiap  $x[k] \in T(x[1], x[2], \dots, x[k-1])$  do
  if  $B(x[1], x[2], \dots, x[k]) = \text{true}$  then
    if  $(x[1], x[2], \dots, x[k])$  adalah lintasan dari akar ke simpul solusi then
      write( $x[1], x[2], \dots, x[k]$ )  { cetak solusi }
    endif
    if  $k < n$  then
      RunutBalikR( $k+1$ )  { tentukan nilai untuk  $x[k+1]$  }
    endif
  endif
endfor

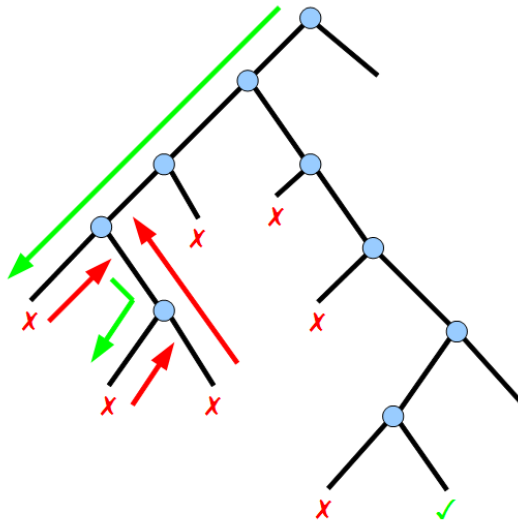
```

Pemanggilan pertama kali: RunutBalikR(1)

Gambar 2. Skema umum algoritma backtracking

(Sumber: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf>)

Gambaran ruang status dari algoritma backtracking dan visualisasi bagaimana algoritma bekerja



Gambar 3. Pohon pencarian algoritma backtracking

(Sumber: <https://www.w3.org/2011/Talks/01-14-steven-phenotype/>)

Tanda silang merah pada simpul menandakan simpul tersebut dibunuh dan menjadi simpul mati atau dead node, tanda ceklis hijau merupakan goal node dari lintasan solusi yang dilalui. Setiap lintasan yang menemukan simpul mati akan backtrack ke simpul sebelumnya untuk mencari lintasan solusi yang lain, begitu seterusnya hingga menemukan goal node.

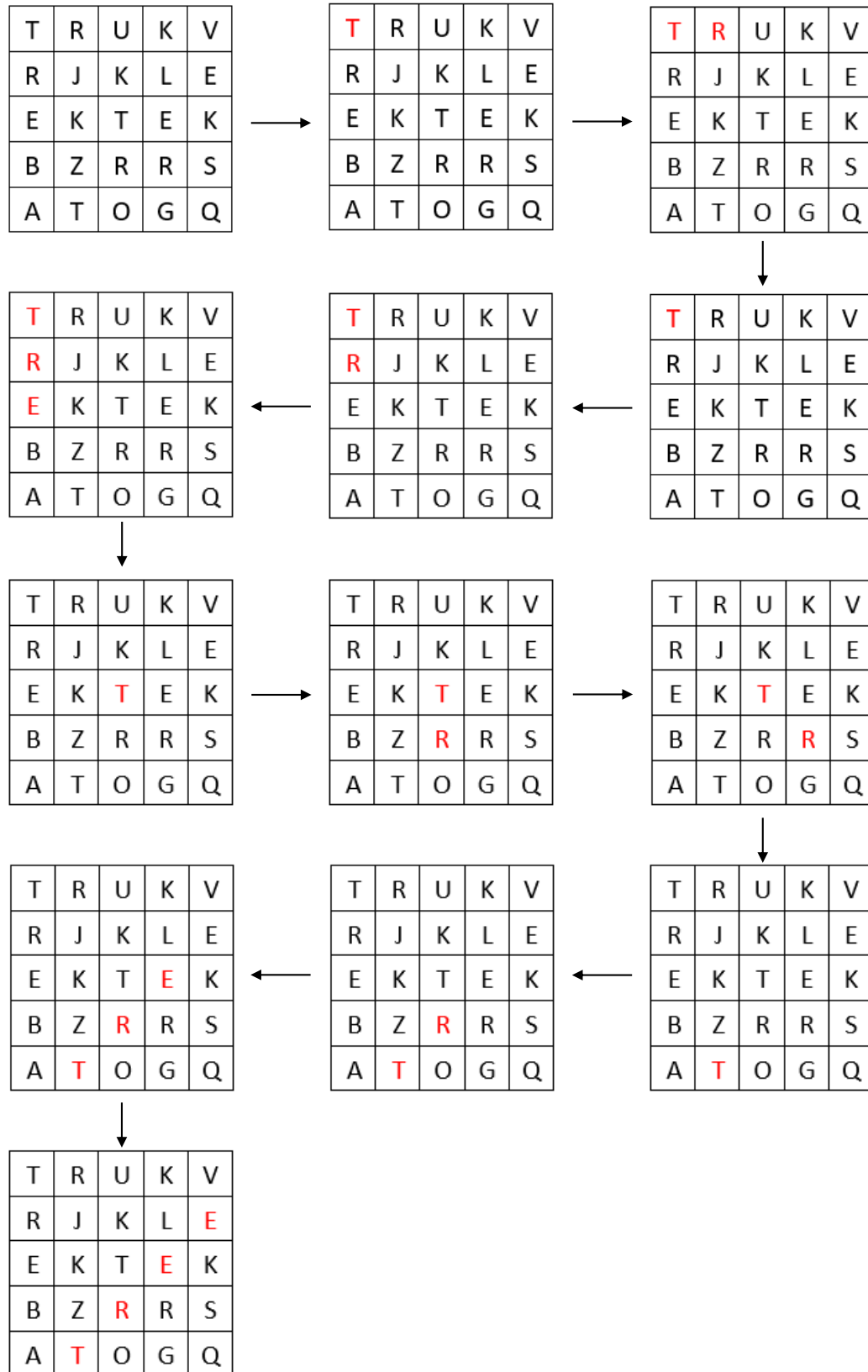
3. HASIL DAN PEMBAHASAN

A. Solusi Penerapan Algoritma

Berikut adalah langkah-langkah pencarian solusi pada game word search menggunakan algoritma backtracking rekursif:

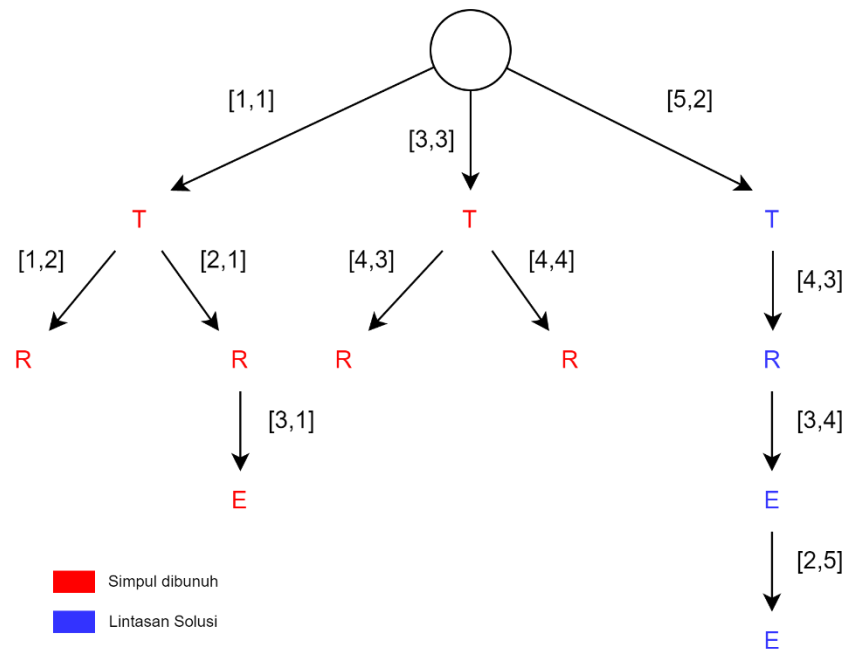
- 1) Pertama, kita harus mengidentifikasi huruf awal dari kata yang ingin ditemukan pada matrik di dalam game word search. Selanjutnya lakukan iterasi tiap sel dalam matrik dari index terkecil hingga terbesar untuk mencari huruf yang cocok dengan huruf awal dari kata yang dicari.
- 2) Jika pencarian huruf awal pada tahap (1) tidak ditemukan, maka kata tersebut tidak ada dalam matrix game word search. Pencarian dihentikan dan program mengembalikan hasil “tidak ditemukan”.
- 3) Jika pencarian huruf awal pada tahap (1) ditemukan, pencarian huruf selanjutnya akan dilakukan pada posisi matrix tersebut. Pencarian huruf selanjutnya dapat berupa sel ke atas, bawah, kanan, kiri dan diagonal seperti yang sudah dijelaskan diatas, pada analisis game word search ada 8 jalur yang mungkin dilalui.
- 4) Simpul-simpul dari posisi matrix tersebut dibangkitkan secara rekursif dengan memanggil kembali algoritma pencarian dengan posisi yang berbeda. Contoh untuk melanjutkan posisi ke kanan kita dapat menambahkan index kolom+1 atau ke kiri kolom-1.
- 5) Pada setiap langkah pembangkitan simpul berikutnya dan untuk mempercepat pencarian solusi ada beberapa fungsi pembatas yang bisa diterapkan:
 - a. Indeks kolom dan baris tidak melebihi batasan matriks pada game word search
 - b. Pembatasan panjang kata, jika pencarian index ke-n melebihi index panjang kata yang dicari maka pencarian dihentikan.
 - c. Pencocokan Huruf, jika huruf pada posisi saat ini tidak sesuai dengan index huruf kata yang dicari maka pencarian akan dihentikan
- 6) Jika pembangkitan simpul berhasil menemukan kata yang dicari fungsi pembatas akan mengembalikan nilai true, pencarian solusi berhasil dan program mengembalikan hasil "Kata ditemukan"
- 7) Fungsi pembatas akan mengembalikan nilai false jika jalur yang terbentuk tidak menuju ke solusi yang diinginkan, sehingga simpul tersebut tidak akan diperluas atau disebut simpul mati.
- 8) Jika pembentukan lintasan tidak diperluas atau berakhir, maka akan dilakukan backtrack ke simpul orangtua untuk melakukan proses pencarian selanjutnya. Pencarian solusi akan kembali ke simpul pada tahap (4) atau simpul orangtua level 1 dalam pohon ruang status. Selanjutnya, simpul tersebut akan menjadi simpul yang diperluas dan akan dilakukan pencarian pada simpul anak berikutnya sesuai jalur yang ditentukan.
- 9) Jika pada tahap ke (8) sudah tidak simpul yang dapat diperluas lagi atau seluruh jalur telah diperiksa dan tidak ditemukan solusi, maka pencarian selanjutnya kembali ke simpul akar pada tahap (1) mencari huruf awal yang sesuai dengan kata yang dicari.

Untuk menggambarkan penyelesaian algoritma backtracking rekursif pada game word search, berikut gambaran tahapan pencarian kata TREE:



Gambar 4. Tahapan Penyelesaian game word search.

Perlu di garis bawah urutan jalur yang diambil terlebih dahulu ditentukan oleh kita saat penerapan solusi pada algoritma. Selanjutnya penggambaran ruang status untuk melihat semua kemungkinan posisi dan arah kata-kata yang dicari dalam matrik, kita asumsikan indek matrix dimulai dari [1,1]



Gambar 5. Ruang status game word search

Dapat dilihat dari gambar ruang status di atas algoritma backtracking hanya membangkitkan simpul yang mengarah pada solusi dengan begitu pencarian kata pada game word search menjadi lebih efisien dengan menghindari semua pemeriksaan kemungkinan solusi.

B. Penelitian Terkait

Makalah penelitian dari Alvin Andhika Zulen (2009) dengan judul Penerapan Algoritma Backtracking Pada Permainan Word Search Puzzle. Pada makalah ini, Peneliti menggunakan algoritma backtracking versi iteratif dan menjelaskan tentang properti umum algoritma backtracking, pengorganisasian solusi dalam pohon ruang status, prinsip pencarian solusi, serta skema umum algoritma backtracking versi iteratif. Selain itu, penulis menganalisa penerapan algoritma backtracking dalam game Word Search Puzzle dan menampilkan tampilan program pencarian solusi.[4]

C. Perbandingan dengan Penelitian terkait

Perbedaan algoritma backtracking versi rekursif dan iteratif secara umum yaitu penulisan kode fungsi rekursif lebih mudah dipahami dan lebih ringkas sedangkan iteratif lebih panjang, pada beberapa kasus menerapkan solusi secara iteratif bisa menjadi lebih sulit untuk. Fungsi rekursif biasanya memerlukan penggunaan memori yang besar tidak cocok digunakan untuk keperluan dengan kinerja yang tinggi karena akan terjadi overhead pemanggilan fungsi yang banyak, penggunaan memori fungsi iteratif relatif lebih kecil dan fungsi iteratif cocok diterapkan saat kinerja aplikasi menjadi faktor penting karena hanya terjadi satu kali pemanggilan fungsi. [5]

Berikutnya kita akan membandingkan kecepatan waktu eksekusi algoritma backtracking versi rekursi dan iteratif, sebagai contoh kita akan membuat kasus game word search dengan matrix 5x5 seperti berikut:

D	C	A	E	W
S	F	R	S	T
A	C	R	T	A
L	D	A	E	H
A	D	Y	T	W

Gambar 6. Kasus Game Word search

Untuk kompleksitas waktu algoritma backtracking bergantung dengan ukuran matrik dan panjang kata yang dicari. Selain itu faktor optimalisasi algoritma dan struktur data yang efisien mempengaruhi kecepatan waktu eksekusi. Informasi waktu eksekusi diukur menggunakan fungsi “high_resolution_clock” dari bahasa pemrograman C++.

Table 1. Waktu Eksekusi Game word search

<i>Kata</i>	<i>Ditemukan</i>	<i>Waktu Eksekusi (Rekursif)</i>	<i>Waktu Eksekusi (Iteratif)</i>
CAT	YA	5/1000 Detik	17/1000 Detik
ARRAY	YA	4/1000 Detik	12/1000 Detik
WHAT	YA	5/1000 Detik	18/1000 Detik
DOG	TIDAK	5/1000 Detik	18/1000 Detik
DFS	TIDAK	4/1000 Detik	9/1000 Detik

Selanjutnya kita akan mencoba kasus game word search dengan matrix 10x10

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

Gambar 7. Kasus-2 Game Word search

Table 2. Waktu Eksekusi Game word search kasus 2

<i>Kata</i>	<i>Ditemukan</i>	<i>Waktu Eksekusi (Rekursif)</i>	<i>Waktu Eksekusi (Iteratif)</i>
MOUSE	YA	3/1000 Detik	10/1000 Detik
WHY	YA	5/1000 Detik	31/1000 Detik
UNIVERSE	YA	6/1000 Detik	58/1000 Detik
HORSE	TIDAK	9/1000 Detik	63/1000 Detik
ITERATIF	TIDAK	7/1000 Detik	76/1000 Detik

4. KESIMPULAN

Penerapan algoritma backtracking rekursif untuk game word search sangat efisien karena hanya membangkitkan simpul pada solusi, serta penulisan kode yang lebih ringkas, mudah dibaca dan dipahami. langkah-langkah penerapan versi rekursif mencakup inisialisasi huruf pertama, pencarian huruf selanjutnya dengan membangkitkan simpul-simpul menggunakan fungsi pembatas, serta proses backtracking rekursif.

Perbandingan antara algoritma backtracking versi rekursif dan iteratif yaitu ada apa cara penulisan serta cara pemanggilan fungsi. Penulisan kode iteratif akan lebih sulit untuk diterapkan untuk beberapa kasus akan tetapi fungsi iteratif lebih memerlukan alokasi memori yang lebih sedikit karena hanya melakukan pemanggilan fungsi sebanyak satu kali. Berbeda dengan rekursif yang cenderung lebih memerlukan alokasi memori yang cukup besar karena pemanggilan fungsi yang berkali-kali dan dapat mengalami overhead fungsi.

Dalam pengujian 2 kasus diatas waktu eksekusi algoritma backtracking versi rekursif lebih cepat di bandingkan dengan iteratif, kompleksitas kata yang dicari dan ukuran matriks memengaruhi waktu eksekusi. Akan tetapi pada kasus lain algoritma backtracking versi iteratif dapat lebih cepat dibandingkan dengan rekursif. Kebutuhan dan karakteristik permasalahan yang akan di selesaikan perlu disesuaikan agar waktu eksekusi bisa lebih optimal, oleh karena itu pemilihan algoritma backtracking versi rekursif dan iteratif sangat penting.

Saran untuk penelitian selanjutnya, optimasi algoritma backtracking dengan meningkatkan kecepatan waktu eksekusi dan efisiensi dalam game word search, serta membandingkan algoritma backtracking versi rekursif dan iteratif dengan beberapa kasus yang memiliki kompleksitas dan kata yang berbeda.

UCAPAN TERIMA KASIH

Ucapan terimakasih kepada dosen kita bu asti yang selalu berkenan menjawab semua pertanyaan, dan mensupport kita, kepada diri saya sendiri tentunya sudah mau berjuang sampai sejauh ini dan semua orang yang telah membantu menyelesaikan penelitian ini dan universitas ARS (Adhirajasa Reswara Sanjaya) tempat saya menempu pendidikan saya ucapkan terimakasih yang sebesar-besarnya.

REFERENSI

- [1] Purnomo, A. J., & Hermawan, G. (2016). Implementasi Algoritma Simplified Memory Bounded A* Untuk Pencarian Kata Pada Permainan Word Search Puzzle. *Jurnal Ilmiah Komputer Dan Informatika*, 5(1), 1–10. <https://doi.org/10.34010/komputa.v5i1.2432>
- [2] Rahman, F. A., & Anubhakti, D. (2020). Implementasi Algoritma Backtracking Pada Permainan Sudoku. *MEANS (Media Informasi Analisa Dan Sistem)*, 5(1), 67–71. <https://doi.org/10.54367/means.v5i1.711>
- [3] Krisdiawan, R. A., Fitriani, A., & Budianto, H. (2022). Penerapan Algoritma Recursive Backtracking Sebagai Maze Generator Pada Game Labirin Aksara Sunda. *Media Jurnal Informatika*, 14(1), 31. <https://doi.org/10.35194/mji.v14i1.2326>
- [4] Zulen, A. (2009). Penerapan Algoritma Backtracking Pada Permainan Word Search Puzzle. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2009-2010/Makalah2009/MakalahIF3051-2009-037.pdf>
- [5] Herlambang, S. (2008). Implementasi Fungsi Rekursif Dalam Algoritma dan Perbandingannya dengan Fungsi Iteratif. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2008-2009/Makalah2008/Makalah0809-079.pdf>
- [6] Munir, R. (2020). Algoritma Runut-balik (Backtracking) Bahan Kuliah IF2211 Strategi Algoritma. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf>