LAPORAN AKHIR

TUGAS BESAR MATA KULIAH STRATEGI ALGORITMA

PERBANDINGAN ALGORITMA BRUTE FORCE DAN ALGORITMA BACKTRACKING PADA PERMAINAN WORD SEARCH PUZZLE

Oleh :

Kelompok 3

Vianka Tetiana (1301184136)

M. Alif Faza (1301183449)

Daniel Septyadi (1301180009)

Muhammad Satria Pradananta (1301190243)

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

2021

ABSTRAK

Permainan word search puzzle adalah permainan mencari kata dalam kumpulan huruf. Permainan word search puzzle ini bisa diselesaikan dengan berbagai strategi algoritma, contohnya seperti algoritma brute force dan backtracking. Algoritma backtracking pada dasarnya adalah untuk mencari segala kemungkinan solusi, sama halnya dengan algoritma brute force, hanya saja algoritma backtracking merupakan perbaikan dari algoritma brute force itu sendiri. Makalah ini berisi pembahasan mengenai algoritma brute force dan backtracking serta penggunaannya dalam penyelesaian word search puzzle. Selain itu akan dibahas pula perbandingan algoritma brute force dan backtracking dalam menyelesaikan permainan word search puzzle berdasarkan analisis dari program yang dibuat penulis untuk mencari kata dalam matriks huruf.

Permainan ini merupakan permainan puzzle pencarian kata-kata tersembunyi yang disusun dalam bentuk array dua dimensi (matriks). Tujuan dari permainan ini adalah menemukan semua kata yang tersembunyi di matriks permainan. Kata-kata tersebut dapat disusun secara horizontal, vertikal maupun diagonal dan dapat ditulis dalam posisi terbalik. Hal yang menarik dari permainan ini adalah dibutuhkannya ketelitian dalam mencari kata-kata diantara serangkaian huruf yang membentuk matriks.

Salah satu cara untuk mencari kata dalam matriks huruf adalah dengan menggunakan algoritma brute force. Algoritma yang terbilang mudah ini seringkali digunakan orang untuk menyelesaikan suatu persoalan. Dalam penggunaan algoritma brute force dalam membuat permainan Word Search Puzzle ini Manusia biasanya secara alamiah akan menggunakan algoritma brute force untuk menyelesaikan persoalan baru yang ditemuinya dengan membandingkan banyak kemungkinan dari algoritma ini. Selain brute force, algoritma backtracking juga dapat menjadi pilihan untuk mencari kata dalam matriks huruf. Algoritma *backtracking* 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 menggunakan algoritma backtracking pada game word search puzzle dapat dijadikan sebagai jalan alternatif dan mempermudah dalam pencarian kata (Azanuddin, 2017).

BAB I
PENDAHULUAN

Permainan kata bukanlah hal yang sulit dijumpai saat ini. Permainan kata dapat ditemukan dalam berbagai bentuk yang menarik, seperti teka teki silang, word search puzzle, dan scrabble. Salah satu permainan kata yang cukup populer adalah word search puzzle. Word search puzzle dapat ditemukan di surat kabar atau majalah. Seiring dengan semakin banyaknya buku yang dibuat sebagai bacaan anak-anak, permainan ini juga dapat ditemukan di buku puzzle atau permainan untuk anak-anak.

Word search puzzle dimainkan dengan cara mencari kata tertentu dalam kumpulan huruf acak atau bisa disebut juga matriks huruf. Permainan kata seperti word search puzzle diminati tidak hanya oleh anak-anak, tapi juga segala usia. Permainan yang sederhana namun dapat mengasah otak membuat word search puzzle menjadi permainan yang menarik. Selain sebagai sarana hiburan, word search puzzle seringkali digunakan untuk tujuan edukasi, seperti membantu anak-anak dalam mengingat kata atau mengeja kata baru yang sedang dipelajari.

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.

BAB II
DASAR TEORI

Algoritma Brute force umumnya tidak "*cerdas*" dan tidak efektif, karena membutuhkan jumlah Langkah yang besar dalam penyelesaiannya. Kadang – kadang algoritma brute force disebut juga algoritma naif.

Algoritma brute force seringkali merupakan pilihan yang kurang disukai karena ketidak efektifannya, tetapi dengan mencari pola-pola yang mendasar, keteraturan, atau trik-trik khusus, biasanya akan membantu kita menemukan algoritma yang lebih cerdas dan lebih efektif. Algoritma brute force sering digunakan sebagai basis bila membandingkan beberapa algoritma yang efektif.

Backtracking adalah algoritma yang berbasis pada DFS (Depth First Search) untuk mencari solusi persoalan. Backtracking yang merupakan perbaikan algoritma brute force. Dengan metode ini, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan.

Untuk contoh kasus yang digunakan yaitu dalam pencarian kata "JUPITER". Dalam pencarian kata ini dapat dicari dengan algoritma Backtracking dan Brute Force. Untuk aturan algoritma Backtracking memiliki urutan sesuai dengan prioritasnya yaitu horizontal ke kanan, horizontal ke kiri, vertikal ke atas, vertikal ke bawah, diagonal ke kiri atas, diagonal ke kiri bawah diagonal ke kanan atas dan diagonal ke kanan bawah. Pencarian kata "JUPITER" diawali dengan pencarian huruf 'J' yang berada di baris pertama kolom pertama. Selanjutnya pencarian kata 'U' berada di arah diagonal kanan bawah dari kata 'J' dan dipilih karena untuk arah yang diprioritaskan sesuai dengan urutannya tidak terdapat kata 'U'. Setelah itu kata selanjutnya yang dicari adalah 'P' yang terletak di arah diagonal kanan bawah kanan. Lalu selanjutnya dicari kata 'I' yang berada di arah di arah diagonal kanan bawah kanan. Selanjutnya untuk huruf 'T' diambil di arah vertikal atas huruf 'I' untuk penggunaan algoritma Backtracking karena sesuai urutan prioritas arah yang diambil dan memiliki huruf yang sesuai dengan yang dicari adalah ke arah vertikal atas walaupun terdapat huruf 'T' ke arah diagonal bawah namun masih kalah prioritas untuk dipilih dibandingkan dengan arah vertikal atas namun jika menggunakan algoritma Brute Force untuk ke arah diagonal kanan bawah dapat dipilih karena masih memenuhi syarat. Selanjutnya untuk huruf 'E' secara Algoritma Backtracking melanjutkan sesuai dengan urutan huruf 'T' yang berada di baris ke-3 yang dipilih, yaitu ke arah diagonal kanan bawah dan untuk Algoritma Brute Force melanjutkan pemilihan kata dengan arah diagonal kanan bawah dengan huruf 'T' berada di baris ke-6 . Terakhir untuk huruf 'R' jika dengan Algoritma Backtracking melanjutkan sesuai dengan urutan huruf 'E' yang berada di baris ke-4 yang dipilih, yaitu ke arah diagonal kanan bawah dan untuk Algoritma Brute Force melanjutkan pemilihan kata dengan arah diagonal kanan bawah dengan huruf 'E' berada di baris ke-7 .

J	S	O	L	U	T	I	S
S	U	N	A	R	U	A	
N	E	P	T	U	N	E	T
S	O	N	I	E	I	S	U
R	C	E	V	I	R	E	R
A	H	T	R	A	E	S	N
M	M	E	R	C	U	R	Y

Kata yang dicari :

JUPITER

Hasil Backtracking :

1	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0
0	0	3	5	0	0	0	0
0	0	0	4	6	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Hasil Brute Force :

1	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0
0	0	3	0	0	0	0	0
0	0	0	4	0	0	0	0
0	0	0	0	5	0	0	0
0	0	0	0	0	6	0	0
0	0	0	0	0	0	0	0

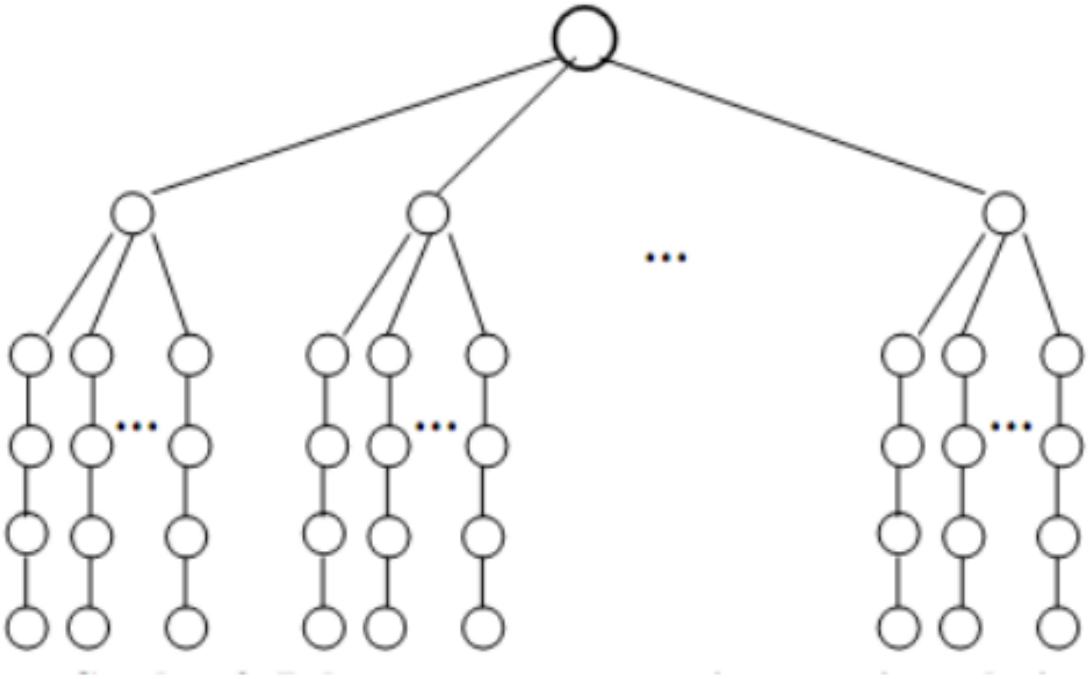
Solving Puzzle

BAB III
IMPLEMENTASI

Dalam implementasi Algoritma Backtracking di game Word Search Puzzle semua kemungkinan solusi dari persoalan disebut ruang solusi (solution space). Nama dari ruang solusi milik backtracking untuk variabel yang digunakan di program adalah solusi dan diorganisasikan ke dalam struktur pohon, tiap simpul pohon menyatakan status (state) persoalan. 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
g. diagonal ke kanan atas
h. diagonal ke kanan bawah

Sehingga dalam pencarian per huruf dari kata yang dicari, kita harus melihatnya dari path atau jalurnya. Solusi persoalan dari permainan word search puzzle adalah vektor N urutan dari huruf pada posisi tertentu. Ruang solusi dari persoalan ini adalah semua kemungkinan kata yang dapat diperoleh di tabel permainan dengan memeriksa path. Untuk ilustrasi Pohon Simpul untuk word search puzzle digambarkan seperti berikut:



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. 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

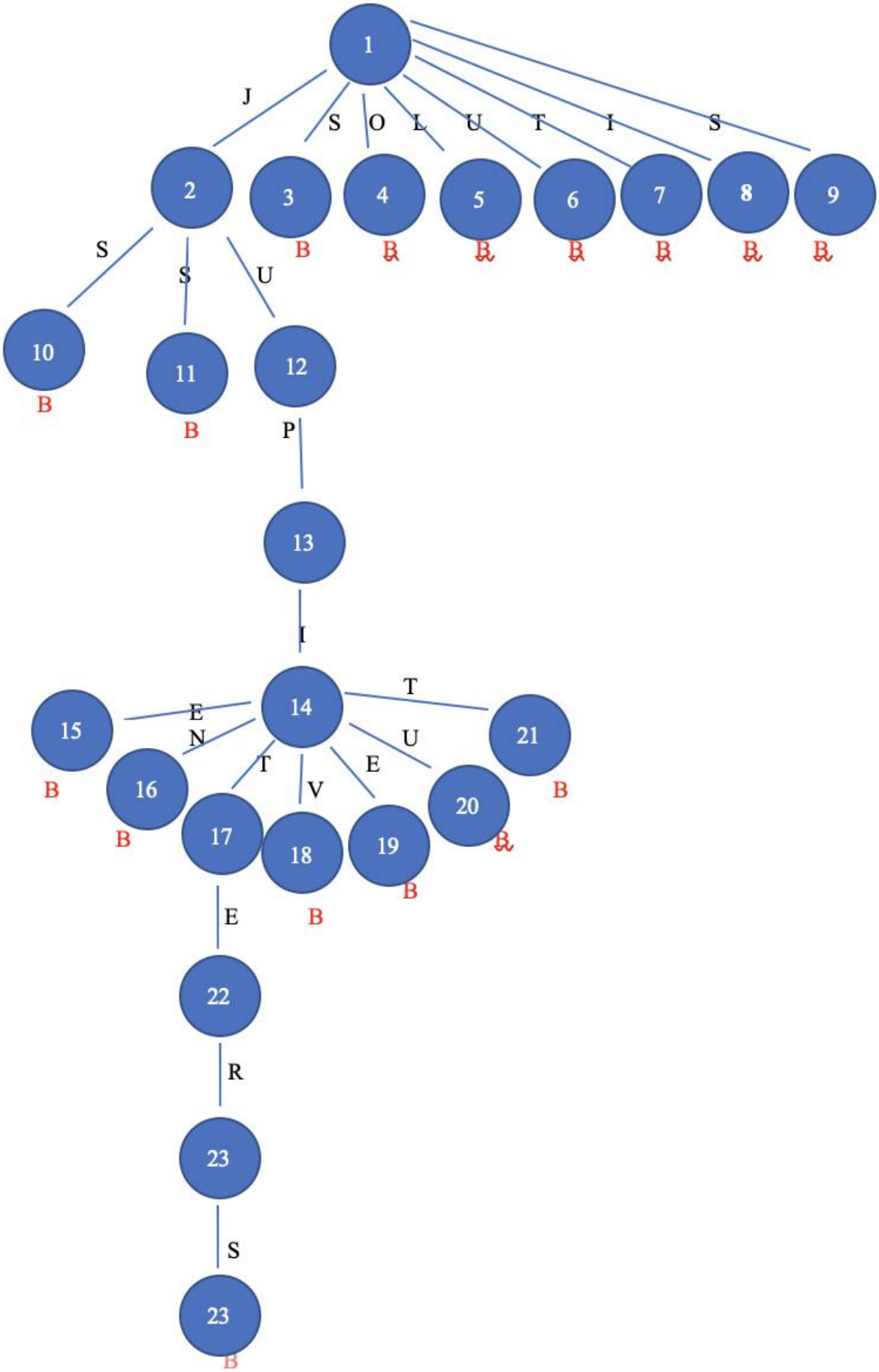
1. Simpul akar pada pohon ruang status merupakan inisialisasi dan menyatakan pencarian huruf awal dari kata yang ingin dicari di papan permainan. Pencarian posisi ini dilakukan dari bagian kiri atas hingga bagian kanan bawah papan permainan. Pencarian ini akan menghasilkan posisi-posisi pada papan permainan yang berisi huruf yang sama dengan huruf awal pada kata yang ingin dicari.
2. Jika pencarian untuk horizontal ke kanan tidak ditemukan satupun posisi pada tabel permainan yang berisi huruf awal dari kata yang dicari, berarti kata tersebut tidak ada di papan permainan.
3. Jika pencarian untuk horizontal ke kanan 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 untuk pencarian vertikal ke bawah 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 prioritas jalur yang telah disebutkan sebelumnya.
9. Jika simpul level 1 yang diperoleh dari pencarian huruf diagonal ke kanan bawah 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 pencarian horizontal ke kanan dengan mencari posisi lain dari huruf awal kata yang ingin dicari di papan permainan.

Dalam implementasi Algoritma Brute Force di game Word Search Puzzle yaitu dengan melakukan pencarian huruf pada matriks yang sesuai dengan huruf pertama kata yang dicari. Kemudian cari ‘tetangga’ huruf pada matriks yang sesuai dengan huruf kedua kata. Jika ada yang sama, tarik garis lurus dari huruf awal mengikuti mengikuti huruf kedua dan ulangi dari

pencocokan huruf pada matriks dengan huruf kata selanjutnya sesuai urutannya. Untuk pencarian hurufnya memiliki syarat tersendiri seperti:

1. Dalam pencarian kata ke arah horizontal kanan ke kiri memiliki syarat yaitu dari titik matriks ditemukan huruf pertama yang sesuai dengan huruf pertama kata yang dicari yaitu jika titik ditambah dengan panjang kata yang dicari tidak melebihi panjang kolom dan sesuai dengan ukuran tabel yang digunakan serta tidak terletak di kolom terakhir.
2. Dalam pencarian kata ke arah vertikal atas ke bawah memiliki syarat yaitu dari titik matriks ditemukan huruf pertama yang sesuai dengan huruf pertama kata yang dicari yaitu jika titik ditambah dengan panjang kata yang dicari tidak melebihi panjang baris dan sesuai dengan ukuran tabel yang digunakan serta titik tidak terletak di baris terakhir.
3. Dalam pencarian kata ke arah diagonal kanan atas ke kiri bawah memiliki syarat yaitu dari titik matriks ditemukan huruf pertama yang sesuai dengan huruf pertama kata yang dicari yaitu jika titik ditambah dengan panjang kata yang dicari tidak melebihi panjang kolom dan sesuai dengan ukuran tabel yang digunakan serta titik tidak terletak pada baris pertama.
4. Dalam pencarian kata ke arah diagonal kiri bawah ke kanan atas memiliki syarat yaitu dari titik matriks ditemukan huruf pertama yang sesuai dengan huruf pertama kata yang dicari yaitu jika titik ditambah dengan panjang kata yang dicari tidak melebihi panjang kolom dan sesuai dengan ukuran tabel yang digunakan serta titik tidak terletak di kolom terakhir dan baris terakhir.

Kemudian setelah melewati pengecekan sesuai dengan syarat di atas dengan menggunakan Algoritma Backtracking maupun Brute Force maka huruf akan diprint dengan diberi warna hijau dan jika ada salah satu huruf yang salah, hasil tidak akan diprint. Untuk mencari kata dalam papan permainan, pengguna hanya cukup menekan tombol solve maka semua solusi akan ditampilkan dan berikut tampilan papan dari matriks yang akan diselesaikan, sebagai contoh implementasi mencari kata “JUPITER” dengan Algoritma Backtracking dengan ukuran tabel 8x7 sesuai gambar di bawah ini:

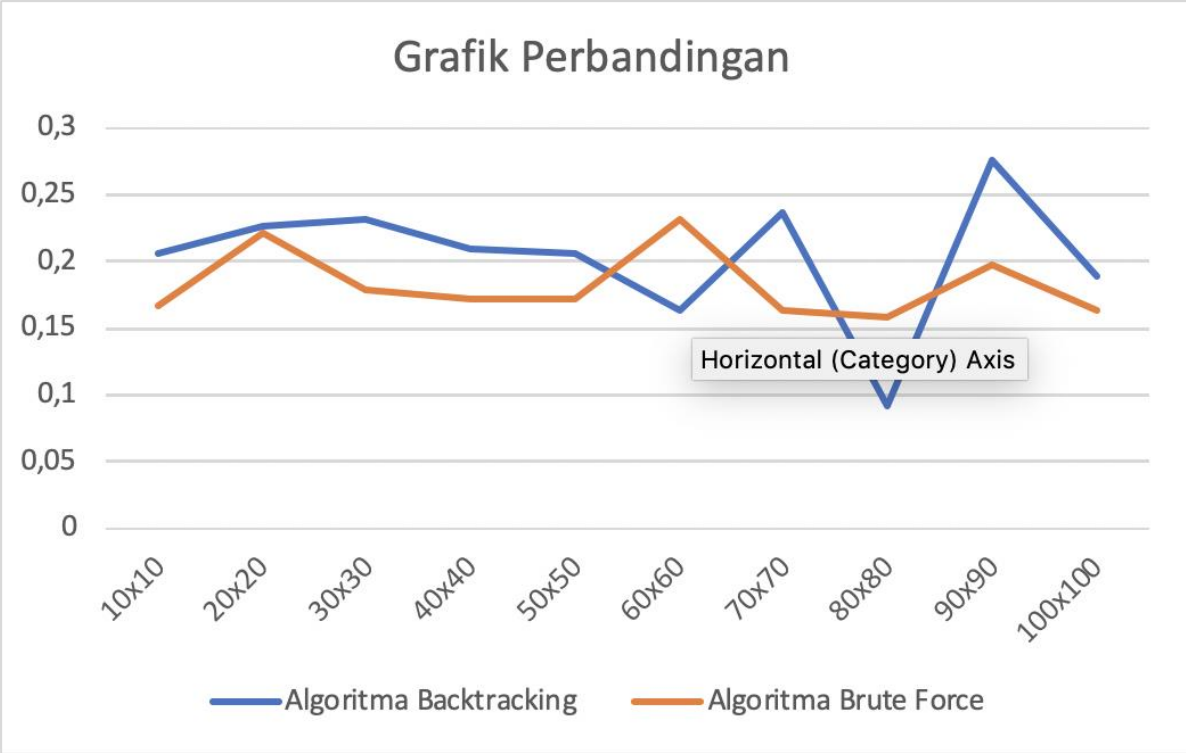

1
J
S
O
L
U
T
I
S
2
3
4
5
6
7
8
9
B
B
B
B
B
B
B
S
S
U
10
11
12
B
B
P
13
I
14
T
15
E
N
16
T
17
V
18
E
19
U
20
21
B
B
B
B
B
B
E
22
R
23
S
23
B

Untuk tanda ‘B’ berwarna merah untuk menunjukan bahwa simpul dibunuh jika pencarian kata yang dicari tidak ada.

BAB IV
ANALISIS

Perbandingan Running Time Algoritma Backtracking dan Brute Force

Ukuran Tabel	Running Time (s)	
	Backtracking	Brute Force
10x10	0.206	0.166
20x20	0.227	0.221
30x30	0.232	0.179
40x40	0.209	0.171
50x50	0.205	0.171
60x60	0.164	0.232
70x70	0.236	0.163
80x80	0.091	0.158
90x90	0.275	0.198
100x100	0.188	0.164



Backtracking

Kompleksitas algoritma yang diterapkan untuk pencarian solusi pada word search ini sebagai berikut:

1. Pilih kata pertama pada soal.
2. Cek pada sel tetangga, apakah sel sudah pernah dikunjungi. Gunanya untuk menentukan *N* sel berikutnya. Karena ini adalah sel pertama maka belum ada sel yang dikunjungi. Maka data sel disimpan di solusi
3. Cara pengecekan yaitu dengan Gerakan yang sudah ditentukan yaitu bawah: (row + 1) , atas: (row - 1), kanan: (col + 1), kiri: (col – 1), diagonal atas kanan: (row – 1, col + 1), diagonal atas kiri: (row -1 , col -1), diagonal bawah kiri: (row + 1, col -1), diagonal bawah kanan: (row + 1, col +1)
4. Proses ini dilakukan terus menerus sampai kondisi total sel = sel yang dikunjungi dan ptCurrent sel = ptEnd sel. Apabila ditemui sel yang sudah dikunjungi maka dilakukan proses backtracking yaitu dengan mengeluarkan data dari solusi yang mana data tersebut mengarah ke sel yang telah dikunjungi kembali ke data di mana sel masih memiliki kemungkinan pengecekan sel tetangga lainnya.

(**Pseudo Code)**

function solveBacktracking(input maze : matrix, input col : kolom, input row: baris, input index: index, input N: posisi) => Boolean
{*true jika solusi ditemukan, false jika tidak*}

Deklarasi

Arah : integer {bawah: row + 1, atas: row -1, kanan: col + 1, kiri: col – 1, diagonal atas kanan: (row – 1, col + 1), diagonal atas kiri: (row -1 , col -1), diagonal bawah kiri: (row + 1, col -1), diagonal bawah kanan: (row + 1, col +1) }

Brute Force

Algoritma brute force untuk pencocokan string adalah sebagai berikut:

(diasumsikan teks berada dalam array T [1..n] dan pattern berada dalam array P [1..m])

1. Mula-mula pattern P dibandingkan kecocokannya pada awal text T
2. Dengan bergerak dari kiri ke kanan, bandingkan setiap karakter di dalam pattern P dengan karakter yang berkesesuaian di dalam teks T sampai semua karakter yang dibandingkan cocok atau sama (pencarian berhasil), atau dijumpai sebuah ketidakcocokan karakter (pencarian belum berhasil)
3. Bila pattern P belum ditemukan kecocokannya dan teks T belum habis, geser pattern P satu karakter ke kanan dan ulangi langkah 2.

Contoh :

Teks : nobody noticed him

Pattern : not

Nobody **not**iced him

S = 0 not

S = 1 not

S = 2 not

S = 3 not

S = 4 not

S = 5 not

S = 6 not

S = 7 not

Pattern **not** ditemukan pada posisi index ke-8 dari awal teks.

Kompleksitas algoritma pencocokan string yang dihitung dari jumlah perbandingan yang dilakukan untuk kasus terbaik adalah O(*n*).

Kasus terbaik ini terjadi apabila karakter pertama pattern P tidak pernah sama dengan karakter pada teks T yang dicocokan. Pada kasus ini dilakukan paling banyak N buah operasi perbandingan. Untuk kasus terburuk, kompleksitas algoritma ini adalah O(*mn*) karena dibutuhkan m(n-m + 1) perbandingan.

(Pseudo Code)

```
function solveBruteForce(input m: integer, P: array[1..m] of char, output solusi: array[1..m]
of integer)
{menghitung nilai b[1..m] untuk pattern P [1..m] }

Deklarasi
    k, q : integer

Algoritma
    b[1] <- 0
```

```
q <- 2
k <- 0
for q <- 2 to m do
        while ((k > 0) and (P[q] ≠ P[k + 1])) do
                k <- b[k]
        endwhile
        if P[q] = P[k + 1] then
                k <- k +1
        endif
        b[q] = k
endfor
```

BAB V
KESIMPULAN

Kesimpulannya adalah 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, vertikal dan diagonal serta dengan Algoritma Backtracking tidak perlu memeriksa semua kemungkinan solusi yang ada hanya pencarian yang mengarah ke solusi yang akan dipertimbangkan dibandingkan dengan Algoritma Brute Force yang tidak efektif, karena membutuhkan jumlah Langkah yang besar dalam penyelesaiannya.

DAFTAR PUSTAKA

[1] Adli Abdillah Nababan. (2014). Retrieved from https://123dok.com/: https://123dok.com/document/q2krwvpq-implementasi-algoritma-brute-force-algoritma-morris-pencarian-suggestion.html

[2] Azanuddin, Iskandar Zulkarnaen, Purwadi. (2017, September). Retrieved from https://docplayer.info/: https://docplayer.info/79200936-Algoritma-backtracking-sebagai-solusi-game-word-search-puzzle-berbasis-java-mobile.html

[3] Cilvia Sianora Putri. (2013/2014). Retrieved from https://fdokumen.com/: https://fdokumen.com/document/algoritma-brute-force-dan-pencocokan-string-pada-game-word-rinaldimunirstmik2013-2014-genap.html

LAMPIRAN

Link Google Drive;

https://drive.google.com/file/d/1tZ_6YMgys8G7lWEAS_N4y7Ea3CBFBBIE/view?usp=sharing