KELOMPOK 3

M. Alif Faza (1301183449)

Daniel Sepiyadi (1301180009)

Vianka Tetiana (1301184136)

Muhammad Satria P. (1301190243)

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

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.**

PENDAHULUAN

Permainan kata dapat ditemukan dalam berbagai bentuk yang menarik, seperti t**eka 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.**

.

DASAR TEORI

Algoritma Brute force umumnya tidak “cerdas” dan tidak efektif, karena membutuhkan jumlah dan langkah yang besar dalam penyelesaiannya. Algoritma brute force disebut juga algoritma naif. **Algoritma Backtracking** adalah algoritma yang berbasis pada DFS (Depth First Search) untuk mencari solusi persoalan. Backtracking yang merupakan perbaikan algoritma brute force.

IMPLEMENTASI

```
const warna = '#00FF00';
const soal = ['EARTH', 'JUPITER', 'MARS', 'MERCURY'];
const matrix = [
    [],
    ['J', 'S', 'O', 'L', 'U', 'T', 'I', 'S'],
    ['S', 'U', 'N', 'A', 'R', 'U', 'U', 'A'],
    ['N', 'E', 'P', 'T', 'U', 'N', 'E', 'T'],
    ['S', 'O', 'N', 'I', 'E', 'I', 'S', 'U'],
    ['R', 'C', 'E', 'V', 'T', 'R', 'E', 'R'],
    ['A', 'H', 'T', 'R', 'A', 'E', 'S', 'N'],
    ['M', 'M', 'E', 'R', 'C', 'U', 'R', 'Y'],
];
var solusi = [];
var solusi2 = [];
var visited = [];
let path = 1;
let path2 = 1;
const boardbody = $('#board tbody');
const soalbody = $('#soal tbody');
```

IMPLEMENTASI

WordMairix & WordMairixBruie

```
function wordMatrix(N) {
    // creating two dimensional array
    for (let i = 0; i < N; i++) {
        for (let j = 0; j < N; j++) {
            solusi[i] = [];
        }
    }

    for (let i = 0; i < N; i++) {
        for (let j = 0; j < N; j++) {
            solusi[i][j] = 0;
        }
    }
}

function wordMatrixBrute(N) {
    // creating two dimensional array
    for (let i = 0; i < N; i++) {
        for (let j = 0; j < N; j++) {
            solusi2[i] = [];
        }
    }

    for (let i = 0; i < N; i++) {
        for (let j = 0; j < N; j++) {
            solusi2[i][j] = 0;
        }
    }
}
```

IMPLEMENTASI

SEARCHWORD()

```
function searchWord(matrix, soal) {
    let N = matrix.length;
    for (let i = 0; i < N; i++) {
        for (let j = 0; j < N; j++) {
            if (solveBacktracking(matrix, soal, i, j, 0, N)) {
                return true;
            }
        }
    }
    return false;
}

function searchWordBrute(matrix, soal) {
    let N = matrix.length;
    for (let i = 0; i < N; i++) {
        for (let j = 0; j < N; j++) {
            if (solveBruteForce(matrix, soal, i, j, 0, N)) {
                return true;
            }
        }
    }
    return false;
}
```

IMPLEMENTASI

SOLVEBACKTRACKING()

```
function solveBacktracking(matrix, soal, row, col, index, N) {

    if (solusi[row][col] !== 0 || soal.charAt(index) !== matrix[row][col]) {
        return false;
    }

    if (index === soal.length - 1) {
        matrix[row][col] = path++;
        $(`table#board tr#r${row} td#c${col + 1}`).css('background-color', warna).attr("rel", "X");
        return true;
    }

    solusi[row][col] = path++;

    if (row + 1 < N && solveBacktracking(matrix, soal, row + 1, col, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('background-color', warna).attr("rel", "X");
        return true;
    }

    if (row - 1 >= 0 && solveBacktracking(matrix, soal, row - 1, col, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('background-color', warna).attr("rel", "X");
        return true;
    }

    if (col + 1 < N && solveBacktracking(matrix, soal, row, col + 1, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('background-color', warna).attr("rel", "X");
        return true;
    }

    if (col - 1 >= 0 && solveBacktracking(matrix, soal, row, col - 1, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('background-color', warna).attr("rel", "X");
        return true;
    }

    if (row - 1 >= 0 && col + 1 < N && solveBacktracking(matrix, soal, row - 1, col + 1, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('background-color', warna).attr("rel", "X");
        return true;
    }

    if (row - 1 >= 0 && col - 1 >= 0 && solveBacktracking(matrix, soal, row - 1, col - 1, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('background-color', warna).attr("rel", "X");
        return true;
    }

    if (row + 1 < N && col - 1 >= 0 && solveBacktracking(matrix, soal, row + 1, col - 1, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('background-color', warna).attr("rel", "X");
        return true;
    }

     if (row + 1 < N && col + 1 < N && solveBacktracking(matrix, soal, row + 1, col + 1, index + 1, N)) {
         $(`table#board tr#r${row} td#c${col + 1}`).css('background-color', warna).attr("rel", "X");
         return true;
     }

     solusi[row][col] = 0;
     path--;
     return false;

 }
```

IMPLEMENTASI

SOLVEBRUTEFORCE()

```
function solveBruteForce(matrix, soal, row, col, index, N) {
    if (solusi2[row][col] !== 0 || soal.charAt(index) !== matrix[row][col]) {
        return false;
    }

    if (index === soal.length - 1) {
        matrix[row][col] = path2++;
        $(`table#board tr#r${row} td#c${col + 1}`).css('text-decoration', 'line-through').attr("rel", "X");
        return true;
    }

    solusi2[row][col] = path2++;

    if (row + 1 < N && solveBruteForce(matrix, soal, row + 1, col, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('text-decoration', 'line-through').attr("rel", "X");
        return true;
    }

    if (row - 1 >= 0 && solveBruteForce(matrix, soal, row - 1, col, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('text-decoration', 'line-through').attr("rel", "X");
        return true;
    }

    if (col + 1 < N && solveBruteForce(matrix, soal, row, col + 1, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('text-decoration', 'line-through').attr("rel", "X");
        return true;
    }

    if (col - 1 >= 0 && solveBruteForce(matrix, soal, row, col - 1, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('text-decoration', 'line-through').attr("rel", "X");
        return true;
    }

    if (row - 1 >= 0 && col + 1 < N && solveBruteForce(matrix, soal, row - 1, col + 1, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('text-decoration', 'line-through').attr("rel", "X");
        return true;
    }

    if (row - 1 >= 0 && col - 1 >= 0 && solveBruteForce(matrix, soal, row - 1, col - 1, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('text-decoration', 'line-through').attr("rel", "X");
        return true;
    }

    if (row + 1 < N && col - 1 >= 0 && solveBruteForce(matrix, soal, row + 1, col - 1, index + 1, N)) {
        $(`table#board tr#r${row} td#c${col + 1}`).css('text-decoration', 'line-through').attr("rel", "X");
        return true;
    }

   if (row + 1 < N && col + 1 < N && solveBruteForce(matrix, soal, row + 1, col + 1, index + 1, N)) {
       $(`table#board tr#r${row} td#c${col + 1}`).css('text-decoration', 'line-through').attr("rel", "X");
       return true;
   }

   solusi2[row][col] = 0;
   path2--;
   return false;
}
```

IMPLEMENTASI

MAIN()

```
$(document).ready(function() {
    var soal = "JUPITER";
    drawWordMap();
    // drawSoal(soal);
    wordMatrix(matrix.length);
    wordMatrixBrute(matrix.length);

    $('#soal tr').click(function() {
        $(this).find('td input:radio').prop('checked', true);
    });
    // ketika tombol solve diklik
    $('#solve').click(function() {
        var nama = $('input[type=radio]');
        var solusi = searchWord(matrix, soal);
        var solusi2 = searchWordBrute(matrix, soal);
        console.log(solusi2);
        if (solusi) {
            print();
            print2();
            print3();
        } else {
            alert("Solusi tidak ditemukan")

        }


    });
```

ANALISIS Backiracking

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)**
sampai kondisi **total sel = sel** yang dikunjungi dan **ptCurrent sel = ptEnd sel.**

ANALISIS
Brute Force

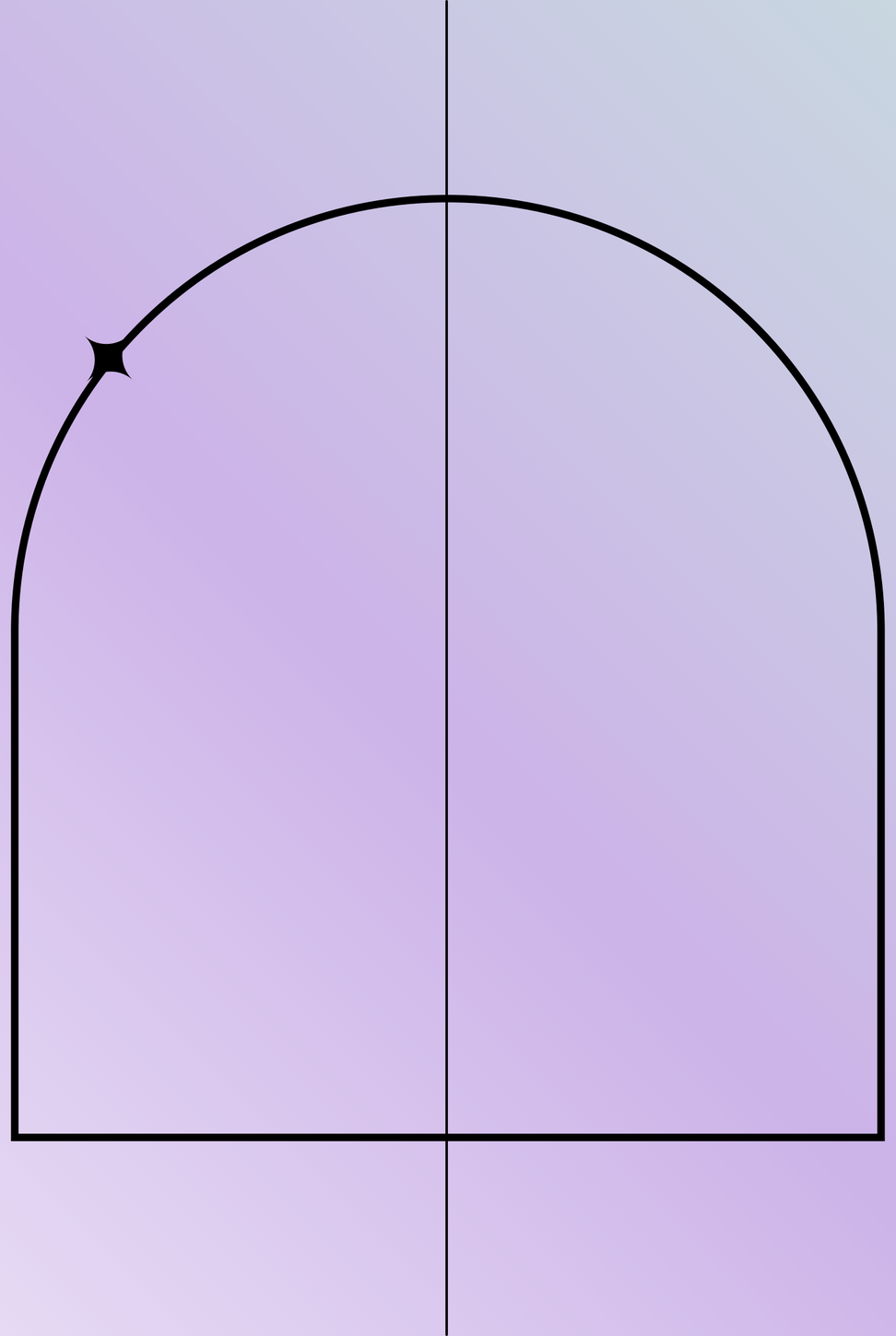
Dengan bergerak dari **kiri ke kanan**, bandingkan setiap karakter di dalam **pattern P** dengan karakter yang **berkesesuaian** di dalam **teks T** sampai semua karakter yang dibandingkan **cocok atau sama** (pencarian berhasil), atau dijumpai sebuah **ketidakcocokan karakter** (pencarian belum berhasil).

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

Untuk kasus **terburuk** adalah **O(mn)** karena diburuhkan m(n-m + 1) perbandingan.

KESIMPULAN

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.



TERIMAKASIH