Tugas Kecil Strategi Algoritma

**Laporan Penyelesaian *Word Search Puzzle* dengan Algoritma *Brute Force***

Oleh:

David Karel Halomoan

13520154



PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

1. **Algoritma *Brute Force***

Algoritma *brute force* adalah penyelesaian suatu persoalan atau masalah dengan pendekatan yang lempang (*straightforward*). Algoritma ini disebut juga dengan algoritma naif (*naïve algorithm*) karena algoritma ini biasanya didasarkan langsung pada pernyataan pada persoalan (*problem statement*) dan definisi/konsep yang dilibatkan.

Pada algoritma *brute force*, pencarian solusi suatu persoalan dapat dilakukan dengan membangkitkan (*generate*) semua kemungkinan solusi dan memeriksa semua kemungkinan tersebut untuk menemukan yang terbaik, teknik ini disebut dengan *exhaustive search*. Pencarian solusi dengan teknik ini membutuhkan waktu dan sumber daya yang sangat besar.

Algoritma *exhaustive search* dapat diperbaiki kinerjanya dengan mengurangi kemungkinan solusi yang perlu diperiksa. Salah satu teknik untuk mengurangi kemungkinan solusi yang perlu dicari adalah teknik heuristik. Teknik heuristic tidak memerlukan pembuktian yang benar secara matematis, teknik ini menggunakan pendekatan yang tidak formal misalnya terkaan, intuisi, atau akal sehat. Penulis menggunakan beberapa teknik heuristik dalam penyelesaian *word search puzzle*.

Penulis menggunakan ADT (*Abstract Data Type*) berupa *dynamic list* dua dimensi (berbentuk matriks, selanjutnya ADT ini akan disebut matriks) untuk merepresentasikan *word search puzzle*, pemilihan penggunaan *dynamic list* digunakan untuk mengubah ukuran matriks saat membaca karakter-karakter pada *word search puzzle* sebab ukuran *word searchpuzzle* tidak tertulis pada *file* masukan. Pengguna juga menggunakan ADT *linked list* untuk menyimpan daftar kata yang akan dicari pada *word search puzzle*. ADT ini memiliki kemudahan menambahkan elemen baru pada akhir *list* yang memudahkan pembacaan *file* masukan yang tidak memiliki jumlah daftar kata yang akan dicari. ADT ini juga dipilih karena ADT ini memudahkan penghapusan elemen yang sedang diiterasi pada *list* tersebut (O(1)), hal ini dibutuhkan untuk melakukan pengimplementasian salah satu teknik heuristik pada algoritma penulis. Penulis juga menggunakan *array* biasa untuk mencetak hasil ke layar.

Selanjtnya akan dijelaskan langkah-langkah pada algoritma yang penulis buat. Pertama-tama, ADT matriksdan *linked list* diinisiasi terlebih dahulu, lalu *file* masukan akan dibaca dan matriks akan diisi dengan karakter-karakter yang ada pada *word search puzzle* dan *linked list* akan diisi dengan dengan daftar kata yang akan dicari pada *word search puzzle*. Selanjutnya, akan diinisiasi dua buah *array* berukuran jumlah kata yang akan dicari, yaitu *array* berisi *string*, yang selanjutnya akan disebut *ans*, yang digunakan untuk menyimpan kata yang ditemukan pada *word search puzzle* dan *array* yang setiap elemennya adalah *array* berukuran tiga, yang selanjutnya akan disebut *ansCoord*, elemen pertama pada *array* ini adalah indeks baris karakter pertama kata pada *array ans*, elemen kedua adalah indeks kolom karakter pertama kata pada *array ans*, dan elemen ketiga adalah arah penemuan kata tersebut. Elemen ketiga ini dapat diisi dengan 8 buah bilangan bulat (*integer*) dari 1 sampai dengan 8, bilangan-bilangan ini melambangkan:

* 1 melambangkan kata ditemukan pada arah atas
* 2 melambangkan kata ditemukan pada arah atas kanan (miring)
* 3 melambangkan kata ditemukan pada arah kanan
* 4 melambangkan kata ditemukan pada arah bawah kanan (miring)
* 5 melambangkan kata ditemukan pada arah bawah
* 6 melambangkan kata ditemukan pada arah bawah kiri (miring)
* 7 melambangkan kata ditemukan pada arah kiri
* 8 melambangkan kata ditemukan pada arah atas kiri (miring)

Bilangan-bilangan ini akan digunakan untuk menentukan arah pencetakan (*output*) kata yang ditemukan. *Array ans* dan *ansCoord* saling berkorespondensi, elemen ke-i pada *ansCoord* mengandung informasi ditemukannya kata pada elemen ke-i pada *array ans*. Selanjutnya, algoritma akan mengiterasi setiap elemen (karakter) pada matriks (*exhaustive search*), lalu algoritma akan melakukan iterasi setiap elemen (kata) pada *linked list* dan melakukan pemeriksaan delapan arah (atas, atas kanan, kanan, bawah kanan, bawah, bawah kiri, kiri, atas kiri) jarak dari karakter matriks yang sedang diiterasi ke batas-batas (pinggir) matriks. Jika jarak yang ditemukan kurang dari panjang kata yang sedang diiterasi, algoritma akan melewatkan pemeriksaan pada arah tersebut, sedangkan jika jarak yang ditemukan sama atau lebih dari panjang kata yang sedang diiterasi, algoritma akan melakukan pencocokkan tiap karakter dari matriks pada arah tersebut dengan tiap karakter dari kata pada *linked list* yang sedang diiterasi. Pengecekan panjang kata dan jarak ini merupakan salah satu teknik heuristik yang digunakan oleh penulis untuk meningkatkan efisiensi algoritma. Jika kata pada arah tersebut ditemukan, kata akan disimpan pada *array ans* dan baris karakter pertama, kolom karakter pertama, dan bilangan yang melambangkan arah ditemukannya kata tersebut (8 bilangan bulat yang sudah disinggung sebelumnya) akan disimpan pada *array* *ansCoord*. Penyimpanan pada *ans* dan *ansCoord* akan dilakukan pada indeks yang sama. Saat kata ditemukan, kata tersebut juga akan dihapus dari *linked list* untuk mengurangi pencarian yang perlu dilakukan. Hal inilah yang membuat penulis memilih *linked list* sebagai ADT penyimpanan daftar kata yang akan dicari. Penghapusan suatu kata dapat dilakukan selagi kata tersebut sedang diiterasi dengan efisiensi O(1). Penulis juga memilih ADT tersebut karena algoritma akan selalu melakukan iterasi setiap daftar kata yang ada pada *linked list*, sehingga efisiensi pengambilan elemen dengan indeks tertentu tidak dibutuhkan. Penghapusan kata ini juga merupakan salah satu teknik heuristik yang digunakan oleh penulis. Program akan terus melakukan langkah-langkah di atas sampai *linked list* kosong (semua kata yang dicari ditemukan) atau elemen terakhir matriks (elemen terbawah dan terkanan) sudah diperiksa. Pada laporan ini, diasumsikan *word search puzzle* selalu mengandung setiap kata yang dicari sehingga algoritma akan berhenti karena *linked list* telah kosong. Algoritma lalu melakukan iterasi *array ans* dan *ansCoord* dan mencetak (*output*) kata pada *ans* sesuai informasi yang terdapat pada *ansCoord*.

1. **Source Program Dalam Bahasa Java**
2. **Inisialisasi Awal**

**Text

Description automatically generated**

1. **File Reader**

**Text

Description automatically generated**

1. **Prosedur Penyelesaian *Puzzle***

Text

Description automatically generated

1. **Prosedur Pencetakan (*Output*) Solusi *Puzzle***

**Text

Description automatically generated**

1. **Prosedur Penyelesaian *Puzzle* dan Pencetakan (*Output*) *Puzzle* Dalam Delapan Arah**
2. **Arah Atas**

**Text

Description automatically generated**

1. **Arah Atas Kanan**

**Text

Description automatically generated**

1. **Arah Kanan**

**Text

Description automatically generated**

1. **Arah Bawah Kanan**

**Text

Description automatically generated**

1. **Arah Bawah**

**Text

Description automatically generated**

1. **Arah Bawah Kiri**

**Text

Description automatically generated**

1. **Arah Kiri**

**Text

Description automatically generated**

1. **Arah Atas Kiri**

**Text

Description automatically generated**

1. ***Screenshot* dari *Input* dan *Output* Program**
2. **Small**

* **Input 1 (small1.txt)**

**A screenshot of a computer

Description automatically generated with low confidence**

* **Output 1**

Background pattern

Description automatically generatedText

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

* **Background pattern

  Description automatically generatedInput 2 (small2.txt)**

**Text

Description automatically generated with medium confidence**

* **Output 2**

Text

Description automatically generated

**Background pattern

Description automatically generated**

**Background pattern

Description automatically generated**

* **Input 3 (small3.txt)**

**Text

Description automatically generated**

* **Output 3**

**Background pattern

Description automatically generated**

**Background pattern

Description automatically generated**

**Background pattern

Description automatically generated with medium confidence**

1. **Medium**

* **Input 1 (medium1.txt)**

**Text

Description automatically generated**

* **Output 1**

**Text

Description automatically generated**A picture containing background pattern

Description automatically generated

Background pattern

Description automatically generated**Background pattern

Description automatically generated**

* **Input 2 (medium2.txt)**

**Text

Description automatically generated**

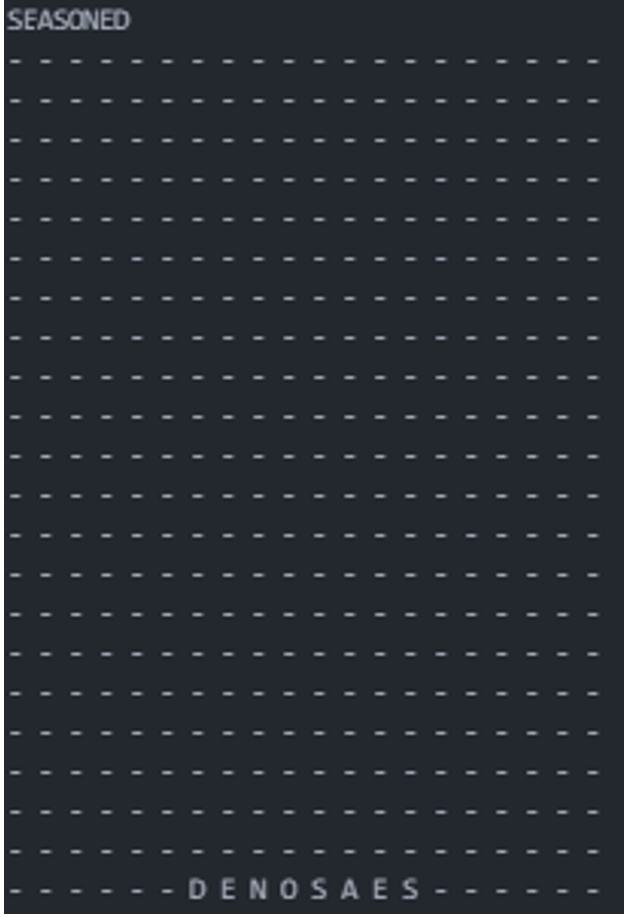
* **Output 2**

Text

Description automatically generated

**Timeline

Description automatically generated with medium confidence**

****

* **Input 3 (medium3.txt)**

**Text

Description automatically generated**

* Text

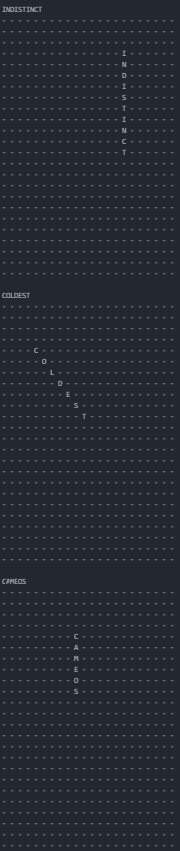
  Description automatically generated**Output 3**

Background pattern

Description automatically generatedBackground pattern

Description automatically generatedBackground pattern

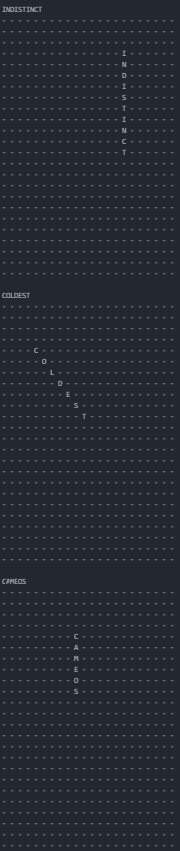
Description automatically generatedBackground pattern

Description automatically generatedBackground pattern

Description automatically generated

**Background pattern

Description automatically generated**Background pattern

Description automatically generatedBackground pattern

Description automatically generatedBackground pattern

Description automatically generatedBackground pattern

Description automatically generatedBackground pattern

Description automatically generated

1. **Large**

* **Input 1 (large1.txt)**

**Text

Description automatically generated**

* **Output 1**

Text

Description automatically generated

* Background pattern

  Description automatically generated with low confidenceA picture containing background pattern

  Description automatically generatedA picture containing background pattern

  Description automatically generatedA picture containing background pattern

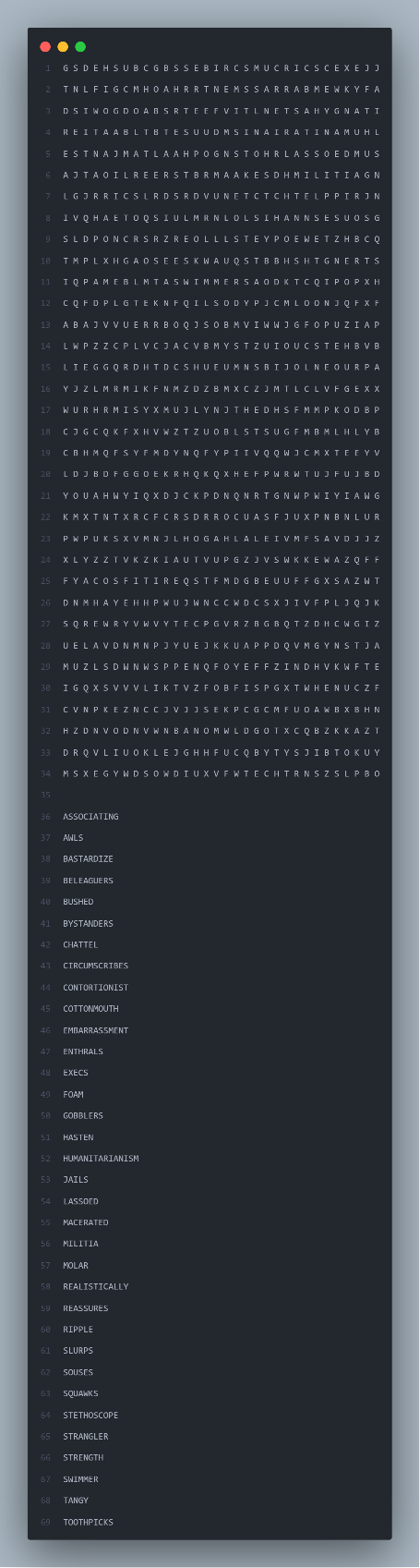
  Description automatically generated
* A picture containing background pattern

  Description automatically generatedBackground pattern

  Description automatically generatedBackground pattern

  Description automatically generated with low confidenceA picture containing background pattern

  Description automatically generated
* **Input 2 (large2.txt)**

****

* **Output 2**

Text

Description automatically generated

A picture containing background pattern

Description automatically generatedA picture containing table

Description automatically generatedA picture containing text

Description automatically generatedA picture containing background pattern

Description automatically generatedA picture containing text

Description automatically generated

A picture containing graphical user interface

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceA picture containing text

Description automatically generatedA picture containing table

Description automatically generated

* **Input 3 (large3.txt)**

**Text

Description automatically generated**

* **Output 3**

Text

Description automatically generated

A picture containing text

Description automatically generatedGraphical user interface, text

Description automatically generatedA picture containing text

Description automatically generatedA picture containing table

Description automatically generatedGraphical user interface, text

Description automatically generated

Text

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA picture containing text

Description automatically generated

1. **Alamat Drive**
2. **Google Drive**

https://drive.google.com/drive/folders/1bAtV2jtMvsL\_3yAKZbmqho4huTozp8GX

1. **Github *Repository***

https://github.com/davidkarelh/Tugas-Kecil-1-IF2211-Strategi-Algoritma-Penyelesaian-Word-Search-Puzzle-dengan-Algoritma-Brute-Force

1. **Tabel Ceklist**

|  |  |  |
| --- | --- | --- |
| **Poin** | **Ya** | **Tidak** |
| Program berhasil dikompilasi tanpa kesalahan (no syntax error) | √ |  |
| Program berhasil *running* | √ |  |
| Program dapat membaca file masukan dan menuliskan luaran | √ |  |
| Program berhasil menemukan semua kata di dalam puzzle. | √ |  |