**TUGAS KECIL**

**Implementasi Algoritma Brute Force dalam Penyelesaian Word Search Puzzle**

**LAPORAN**

**Diajukan sebagai salah satu tugas mata kuliah IF2211 Strategi Algoritma pada**

**Semester II**

**Tahun Akademik 2021-2022**

**oleh**

**Owen Christian Wijaya 13520124**

**A picture containing text

Description automatically generated**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2022**

DAFTAR ISI

[BAB I. ALGORITMA *BRUTE FORCE* 3](#_Toc93830136)

[BAB II. *SOURCE PROGRAM* DALAM BAHASA JAVA 4](#_Toc93830137)

[2.1 CharObj,java 4](#_Toc93830138)

[2.2 Exec.java 5](#_Toc93830139)

[2.3 MainProgram.java 5](#_Toc93830140)

[2.4 Matcher.java 5](#_Toc93830141)

[2.5 Matrix.java 6](#_Toc93830142)

[2.6 SearchWord.java 6](#_Toc93830143)

# BAB I. ALGORITMA *BRUTE FORCE*

Algoritma *brute force* adalah algoritma yang menggunakan metode-metode yang diimplementasikan secara langsung tanpa mempertimbangkan kapabilitas komputasi atau memori komputer. Algoritma *brute force* diimplementasikan dengan menganalisa deskripsi masalah dan mencoba setiap kemungkinan yang dapat dilakukan hingga ditemukan solusi dari permasalahan yang diinginkan. Algoritma *brute force* mengandalkan pendekatan yang terus terang adanya dan cenderung mengabaikan teknik-teknik lanjutan yang meningkatkan efisiensi program.

Kelebihan dari algoritma *brute force* adalah kemampuannya untuk dapat menyelesaikan hampir semua persoalan yang ada (*widely applicable*). Meskipun tidak seefisien algoritma-algoritma lainnya, algoritma *brute force* cenderung lebih sederhana untuk diimplementasikan, dan karena itu, algoritma *brute force* sering digunakan sebagai basis pembanding dengan algoritma-algoritma lain yang lebih efisien. Algoritma *brute force* menghasilkan algoritma-algoritma yang layak untuk masalah-masalah penting seperti pencarian atau pencocokan *string* dan operasi matriks. Akan tetapi, algoritma *brute force* tidak dapat digunakan di sistem dengan sumber daya terbatas dan mengambil waktu yang lama untuk memproses masukan dengan ukuran besar.

Dalam tugas kecil ini, algoritma *brute force* diimplementasikan dalam pencarian kata-kata yang ada dalam permainan *word search puzzle*. *Puzzle* disimpan dalam bentuk matriks karakter, sementara *keywords* disimpan ke dalam sebuah *list*. Algoritma *brute force* akan mencari keberadaan sebuah kata dalam *puzzle* dengan pencocokan karakter pertama dari *keyword* yang hendak dicari dengan karakter yang ada di dalam *puzzle* secara sekuensial. Apabila ditemukan karakter yang sesuai dengan karakter pertama *keyword*, maka pencarian dilanjutkan dengan mencocokkan karakter-karakter lain dengan karakter di *keyword*.

Pencocokan dilakukan terhadap delapan arah: ke arah kanan, ke arah kiri, ke arah atas, ke arah bawah, ke arah diagonal atas kiri, ke arah diagonal atas kanan, ke arah diagonal bawah kiri, dan ke arah diagonal bawah kanan. Sebelum melakukan pencocokan, program akan melakukan pengecekan apakah panjang *keyword* melebihi dimensi matriks *puzzle*. Apabila panjang *keyword* yang hendak dicek sesuai arah melebihi dimensi matriks *puzzle* (akan menyebabkan *out of bounds*), maka pengecekan ke arah tersebut tidak akan dilakukan dan program akan melakukan pengecekan ke arah lainnya. Sebaliknya, apabila panjang *keyword* tidak menyebabkan *out of bounds*, maka pengecekan akan dilakukan. Pengecekan akan dihentikan apabila ditemukan satu karakter di arah tersebut yang tidak cocok. Jika pengecekan yang dilakukan berhasil, maka pencarian akan dihentikan dan program akan memberikan keluaran berupa arah yang ditemukan, jumlah perbandingan yang dilakukan, dan waktu yang dibutuhkan untuk melakukan perbandingan. Setelah semua *keyword* dicocokkan, program akan memberikan keluaran berupa matriks awal dengan *keyword* yang di-*highlight* untuk menandai posisi *keyword* di dalam *puzzle*.

# BAB II. *SOURCE PROGRAM* DALAM BAHASA JAVA

*Source program* dibuat dalam bahasa pemrograman *Java* versi "17" 2021-09-14 LTS dan *Java(TM) SE Runtime Environment (build 17+35-LTS-2724*). *Source program* dibagi ke tujuh file:

* CharObj.java
* Exec.java
* MainProgram.java
* Matcher.java
* Matrix.java
* ReadFile.java
* SearchWord.java

## 2.1 CharObj,java

File CharObj.java adalah *source code* berisi *abstract datatype* yang akan digunakan sebagai elemen dari matriks *puzzle*. ADT CharObj terdiri atas dua variabel:

* oriChar: String untuk menyimpan karakter awal
* colorChar: String untuk menyimpan karakter yang telah di-*highlight*

Pada waktu program membaca file masukan, setiap karakter akan di simpan ke sel matriks yang berisi ADT CharObj, di mana karakter akan disimpan di oriChar dan colorChar. Saat memperbarui karakter matriks untuk diberikan pewarnaan, variabel colorChar yang digunakan untuk menyimpan karakter yang telah diwarnai menggunakan pewarnaan ANSI. Setelah itu, untuk menampilkan matriks akhir, variabel colorChar yang akan dijadikan keluaran.

## 2.2 Exec.java

File Exec.java adalah *source code* berisi *abstract datatype* untuk menyimpan hasil eksekusi dari fungsi. ADT Exec terdiri atas dua variabel:

* flag: boolean untuk menandai apakah pencocokan berhasil atau tidak
* compCount: integer untuk menyimpan jumlah perbandingan

ADT Exec adalah datatype yang dikembalikan oleh fungsi-fungsi pencocokan. Apabila nilai variabel flag adalah true, maka proses pencarian akan dihentikan, namun jika nilainya false, maka proses pencarian akan terus dilanjutkan. Nilai perbandingan total kata akan dijumlahkan dengan variabel compCount.

## 2.3 MainProgram.java

File MainProgram.java berisi class yang dijalankan sebagai program utama. File MainProgram mengatur *workflow* program dari pemanggilan fungsi ReadFile, fungsi SearchWord, dan keluaran matriks akhir dan penghitungan waktu dan perbandingan.

## 2.4 Matcher.java

File Matcher.java berisi fungsi-fungsi yang akan dipanggil untuk melakukan pencocokan kata. Ada delapan fungsi yang digunakan untuk melakukan pencocokan:

* checkHL: mengecek kata searah horizontal kiri *(horizontally left / HL)*
* checkHR: mengecek kata searah horizontal kanan *(horizontally right / HR)*
* checkVU: mengecek kata searah vertikal atas *(vertically upper / VU)*
* checkVL: mengecek kata searah vertikal bawah *(vertically lower / VL)*
* checkDLU: mengecek kata searah diagonal kiri atas *(diagonally left upper / DLU)*
* checkDLL: mengecek kata searah diagonal kiri bawah *(diagonally left lower / DLL)*
* checkDRU: mengecek kata searah diagonal kanan atas *(diagonally right upper / DRU)*
* checkDRL: mengecek kata searah diagonal kanan bawah *(diagonally right lower / DRL)*

Secara umum, fungsi-fungsi tersebut bekerja dengan algoritma yang sama. Apabila ditemukan karakter di matriks yang sama dengan karakter pertama *keyword* yang hendak dicari, fungsi-fungsi tersebut akan dipanggil. Fungsi menerima argumen berupa matriks *puzzle*, *keyword* yang telah dipisah dalam bentuk *array*, dan koordinat dimana karakter pertama ditemukan. Setiap fungsi akan melakukan pengecekan sesuai arah. Apabila panjang *keyword* yang dicocokkan di luar dimensi matriks (*out of bounds*), maka pengecekan tidak akan dilakukan dan fungsi akan mengembalikan Exec dengan flag bernilai false. Sebaliknya, fungsi melakukan pencocokan *array keyword* dengan karakter-karakter di koordinat sekitar.

Dengan menggunakan skema pencarian menggunakan *boolean*, fungsi akan melakukan pencocokan per karakter. Apabila ada satu karakter yang tidak cocok, pencarian akan berhenti dan fungsi akan mengembalikan Exec dengan flag bernilai false. Sebaliknya, apabil pencocokan berhasil secara keseluruhan, maka program akan melakukan perubahan terhadap elemen matriks yang dicek. Nilai variabel colorChar dalam elemen charObj dalam matriks akan diubah menjadi karakter berwarna sesuai dengan nilai koordinat dan panjang kata yang dicek menggunakan kode ANSI. Pada akhir program, matriks yang akan ditampilkan adalah matriks dengan variabel colorChar yang telah berwarna.

## 2.5 Matrix.java

File Matrix.java berisi *abstract datatype* berupa matriks yang digunakan untuk menyimpan *puzzle* dalam bentuk matriks. Matriks yang disusun terdiri atas elemen-elemen bentukan CharObj. File ini juga mengandung fungsi-fungsi yang diperlukan untuk mencetak matriks di awal (dalam bentuk karakter asli) dan matriks dengan karakter yang telah diwarnai.

## 2.6 SearchWord.java

File SearchWord.java berisi implementasi dari fungsi-fungsi yang ada dalam file Matcher.java. File ini mengandung fungsi yang diperlukan untuk melakukan semua pencocokan secara sekuensial. Fungsi search menerima argumen berupa matriks *puzzle* dan *keyword* yang harus dicari. Fungsi akan melakukan pencocokan karakter pertama dari *keyword* terhadap elemen-elemen di matriks secara sekuensial. Apabila ditemukan karakter yang sesuai, fungsi akan melakukan pencocokan. Jika pencocokan berhasil, pencarian akan dihentikan, fungsi memberikan keluaran berupa informasi eksekusi, dan fungsi mengembalikan daftar berupa waktu dan banyaknya perbandingan eksekusi.

**2.7 ReadFile.java**

File ReadFile.java berisi fungsi yang melakukan *parsing* terhadap file masukan. Pengguna dapat memasukkan nama file tanpa .txt yang berisi matriks dan daftar *keyword* sesuai ketentuan.

Text

Description automatically generated

Setelah itu, program akan membaca isi file dan memperbarui matriks kosong dengan karakter-karakter dalam *puzzle* dan memperbarui daftar dengan daftar *keywords* yang harus dicari.