LAPORAN TUGAS KECIL

**“Penyelesaian *Word Search Puzzle* dengan Algoritma Brute Force”**

Laporan Ini Dibuat untuk Memenuhi Tugas Perkuliahan

Mata Kuliah Strategi Algoritma (IF2211)

**KELAS 01**

**Dosen : Dr. Masayu Leylia Khodra, S.T., M.T.**

**Logo

Description automatically generated**

Rio Alexander Audino / 13520088

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**SEMESTER II TAHUN 2021/2022**

1. **ALGORITMA BRUTE FORCE**

Pada program yang saya buat, proses pengaplikasian Algoritma *Brute Force* terbagi ke dalam 4 langkah utama, yakni pembacaan file, validasi arah, validasi kata, dan terkahir penampilan jawaban.

1. Pembacaan File

Agar bisa menerima input file dari user, program harus bisa melakukan pembacaan dari sebuah file txt. File txt disesuaikan dengan standar pada spesifikasi tugas kecil. Proses pembacaan file akan menghasilkan dua buah matrix, matrix *random alphabet* (*puzzles*) dan matrix *searched words* (*words*). Proses pembacaan file di-*handle* library“fstream”.

1. Validasi Arah

Setelah berhasil melakukan pembacaan file, selanjutnya program akan masuk pada bagian algoritma *Brute Force*. Pada awalnya, akan dilakukan validasi arah terlebih dahulu. Proses validasi arah merupakan proses validasi arah-pengecekan. Proses validasi arah dilakukan untuk mengurangi jumlah komparasi huruf. Misal sekarang berada pada ujung kanan atas papan. Melalui proses validasi arah, hanya arah yang valid seperti kiri, kiri-bawah, atau bawah yang akan dicek, bukan arah atas, atas.

1. Validasi Kata

Setelah didapat arah-pengecekan yang valid, selanjutnya akan masuk pada proses pencocokan tiap huruf (validasi kata). Proses validasi kata mengambill porsi terbanyak dalam proses komputasi. Tiap huruf pertama dan arah yang telah valid, selanjutnya akan di-cek satu persatu sesuai dengan arah-pengecekannya. Sesuai dengan tema laporan kali ini – Algoritma *Brute Force* – tidak ada cara singkat dalam proses validasi kata. Tiap huruf harus dicocokan satu-persatu dari setiap posisi yang valid.

1. Penampilan Jawaban

Proses penampilan jawaban adalah akhir dari program ini. Program pada akhir prosesnya akan menampilkan kata-kata yang ditemukan dalam *puzzles* tadi. Penampilan jawaban disesuaikan dengan format pada spesifikasi tugas besar. Tiap kata akan ditampilkan satu-persatu pada layar.

1. Waktu Komputasi

Di luar 4 langkah utama ini, ada satu proses lagi untuk mengecek waktu komputasi. Waktu komputasi dihitung setelah proses pembacaan file, dari validasi arah sampai akhir program. Proses perhitungan waktu komputasi di-*handle* oleh library “chrono”

1. **KODE PROGRAM**

Pada program ini, saya menggunakan bahasa C++. Pemilihan bahasa C++ didasarkan oleh waktu komputasi bahasa C++ umumnya lebih cepat dibandingkan C maupun Java. Kode program akan saya tampilkan sesuai dengan pembagian pada bagian sebelumnya, yakni pembacaan file, validasi arah, validasi kata, penampilan jawaban, dan waktu komputasi.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Proses | Gambar | Deskripsi Kode/Fungsi |
| 1 | Pembacaan File |  | Membuka nama file sesuai input user |
|  | Membuat matriks puzzle huruf acak |
|  | Membuat matriks penyimpanan kata yang akan dicari |
| 2 | Validasi Arah |  | Memvalidasi arah-pengecekan. Fungsi akan mengembalikan true jika arah tersebut valid |
| 3 | Validasi Kata |  | Memastikan arah pengecekan. verMove dan horMove adalah arah pengecekan. |
|  | Melakukan pengecekan per karakter/huruf. |
| 4 | Penampilan Jawaban |  | Memastikan arah penampilan jawaban |
|  | Menampilkan kata yang dicari  Melakukan iterasi di tiap huruf puzzles. |
|  | Memastikan huruf yang ditampilkan masih dalam jangkauan panjang kata  Membagi ke beberapa kasus, untuk kanan-kiri, atas-bawah, dan diagonal. |
| 5 | Program Utama |  | Melakukan pembacaan file  Memulai waktu perhitungan |
|  | Melalukan iterasi dari setiap kata yang dicari  Melakukan iterasi ke seluruh huruf puzzle  Mengecek huruf pertama |
|  | Melakukan iterasi ke setiap arah  Melakukan validasi arah, validasi kata, dan penampilan jawaban |
|  | Mengakhiri waktu komputasi  Menampilkan statistik komputasi |

1. **DEMO PROGRAM**
   1. Medium 1 (1316ms / 5182 Comparison)

|  |  |  |
| --- | --- | --- |
| Input | Hasil | |
|  |  |  |

* 1. Medium 2 (1288ms / 6212 comparison)

|  |  |  |
| --- | --- | --- |
| Input | Hasil | |
|  |  |  |

* 1. Medium 3 (1307ms / 5802 Comparison)

|  |  |  |
| --- | --- | --- |
| Input | Hasil | |
|  |  |  |

* 1. Medium 4 (3557ms / 14626 Comparison)

|  |  |  |
| --- | --- | --- |
| Input | Hasil | |
|  |  |  |

* 1. Large 1 (8223ms / 34120 Comparison)

|  |  |  |
| --- | --- | --- |
| Input | Hasil | |
|  |  |  |

* 1. Large 2 (3223ms / 13785 Comparison)

|  |  |  |
| --- | --- | --- |
| Input | Hasil | |
|  |  |  |

* 1. Small 1 (954ms / 3996 Comparison)

|  |  |  |
| --- | --- | --- |
| Input | Hasil | |
|  |  | A screenshot of a computer  Description automatically generated with medium confidence |

* 1. Small 2 (298ms / 1332 Comparison)

|  |  |  |
| --- | --- | --- |
| Input | Hasil | |
| Text, arrow  Description automatically generated |  |  |

1. **SOURCE PROGRAM**

Link Github : https://github.com/Audino723/Tucil1\_13520088

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi tanpa kesalahan (no syntax error) | √ |  |
| 1. Program berhasil running | √ |  |
| 1. Program dapat membaca file masukan dan menuliskan luaran | √ |  |
| 1. Program berhasil menemukan semua kata di dalam puzzle | √ |  |