TUGAS KECIL 1 IF 2211

STRATEGI ALGORITMA

APLIKASI ALGORITMA BRUTE FORCE UNTUK MEMBUAT SUATU PROGRAM SOLVER UNTUK PERMAINAN *SEARCH WORD PUZZLE*



DIBUAT OLEH

13520028 Timothy Stanley Setiawan

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG

Semester II Tahun 2021/2022

BAB 1: Deskripsi Masalah

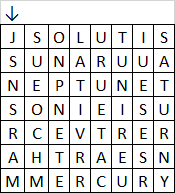
Brute force adalah sebuah pendekatan *straightforward* untuk memecahkan suatu masalah, biasa didasarkan pada pernyataan masalah dan definisi konsep yang dilibatkan. Algoritma brute force adalah penyelesaian masalah dengan pendekatan secara langsung, sederhana, dan jelas. Algoritma brute force ini hampir bisa menyelesaikan semua permasalahan. Namun, memiliki kelemahan memerlukan *resources* yang tinggi untuk menyelesaikannya, sedangkan *resources* itu sendiri terbatas. Akan tetapi, algoritma brute force ini masih cukup revelan untuk beberapa permasalahan dan terkadang ada beberapa permasalahan yang hanya bisa diselesaikan dengan menggunakan algoritma brute force ini dan atau paling optimal menggunakan brute force. Salah satu dari hal tersebut adalah algoritma untuk menyelesaikan *word search puzzle*, suduko, tsp, knapsack 0/1, dan lain sebagainya. Pada tucil 1 mata kuliah strategi algoritma IF2211 ini, dibuat suatu program yang memanfaatkan algoritma brute force untuk menyelesaikan *word search puzzle*. Pada kesempatan kali ini, kompleksitas algoritma yang berupa perbandingan jumlah huruf dan waktu eksekusi program diperhitungkan untuk mengetahui seberapa efektif program yang telah dibuat.

BAB 2: Algoritma Brute Force

Pada tucil 1 ini, penulis memanfaatkan dua jenis algoritma brute force untuk membuat program penyelesaian puzzle *word search puzzle*. Pada algoritma pertama, program akan mengecek secara *naif* satu per satu huruf pada yang tersedia pada *puzzle* ke 8 arah (kanan, kanan-bawah, bawah, kiri-bawah, kiri, kiri-atas, atas, kanan-atas). Untuk algoritma kedua, akan memanfaatkan konsep heuristik yang akan melewati pengecekan setiap kali sisa panjang pengecekan *puzzle* < panjang huruf. Berikut ini penjelasan dari kedua algoritma tersebut

* 1. Algoritma 1

1. Pertama-pertama, letakan suatu penanda (*pointer*) pada matrix *puzzle* dibagian kiri atas dari puzzle (index *puzzle* i = 0, j = 0)



*Gambar 1 Pointer berada di kiri atas* puzzle

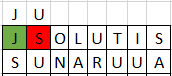
1. Kemudian, pada *pointer* tersebut, bandingkan kata yang akan dicari dengan huruf-huruf yang tersedia di matrix *puzzle* dalam delapan arah pengecekan.
2. Pada pengecekan :
3. ke arah kanan



*Gambar 2 Pengecekan ke arah kanan*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kiri kanan. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* y + i >= panjang kolom matriks
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda



*Gambar 3 Pengecekan huruf yang sesuai (J) dan tidak sesuai (U)*

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah kanan-bawah



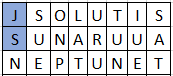
*Gambar 4 Pengecekan ke arah kanan-bawah*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kiri-atas ke kanan-bawah. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* y + i >= panjang kolom matriks
* x + 1 >= panjang baris matriks
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah bawah



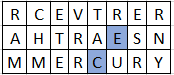
*Gambar 5 Pengecekan ke arah bawah*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari atas ke bawah. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* x + 1 >= panjang baris matriks
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah kiri-bawah



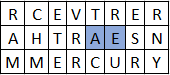
*Gambar 6 Pengecekan ke arah kiri-bawah*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kanan-atas ke kiri-bawah. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* y - i < 0
* x + 1 >= panjang baris matriks
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah kiri



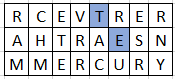
*Gambar 7 Pengecekan ke arah kiri*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kiri ke kanan. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* y - i < 0
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah kiri-atas



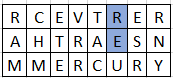
*Gambar 8 Pengecekan ke arah kiri-atas*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kanan-bawah ke kiri-atas. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* y - i < 0
* x - 1 < 0
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah atas



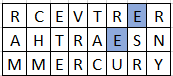
*Gambar 9 Pengecekan ke arah atas*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari bawah ke atas. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* x - 1 < 0
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah kanan-atas

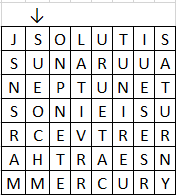


*Gambar 10 Pengecekan ke arah kanan-atas*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kiri-bawah ke kanan-atas. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* y + i >= panjang kolom matriks
* x - 1 < 0
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.



*Gambar 11 Menggeser Pointer*

1. Jika dari hasil pengecekan belum menemui susunan huruf yang sesuai, geser *pointer* itu menunju huruf selanjutnya pada matrix *puzzle*, kemudian ulangi langkah ke dua.
2. Jika dari hasil pengecekan ditemukan susunan huruf yang sesuai dengan huruf yang di cari, tampilkan total waktu eksekusi program dan jumlah perbandingan huruf yang dilakukan serta dengan arah pencarian mana susunan huruf tersebut ditemukan.



*Gambar 12* Output *ketika kata berhasil ditemukan*

1. Jika hingga pointer menunjukkan index terakhir matrix dan kata yang dicari tidak ada, tampilkan pesan “kata tidak ditemukan dalam puzzle”.

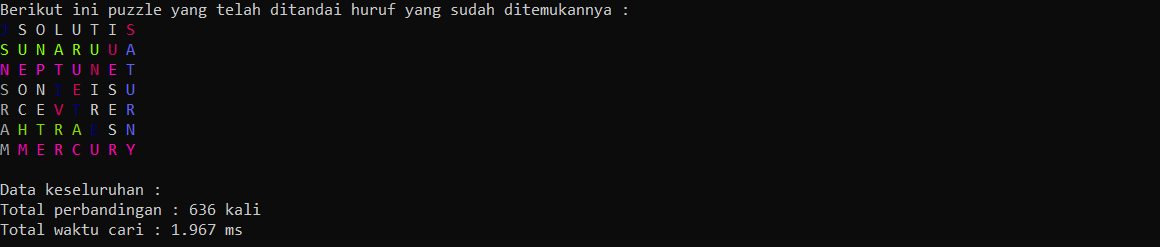


*Gambar 13 Pointer sudah berada diujung matriks*



*Gambar 14* Output *ketika kata gagal ditemukan*

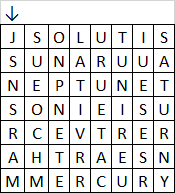
1. Ulangi langkah ke satu hingga seluruh kata yang dicari sudah dicek keberadaannya di dalam *puzzle*.
2. Tampilan matriks *puzzle* hasil pencarian dengan memberi warna berbeda pada kat-kata yang berhasil ditemukan dan tunjukan waktu ekseskusi dan juga jumlah perbandingan yang dilakukan.



*Gambar 15* Output *akhir*

* 1. Algoritma 2

1. Pertama-pertama, letakan suatu penanda (*pointer*) pada matrix *puzzle* dibagian kiri atas dari puzzle (index *puzzle* i = 0, j = 0)



*Gambar 16 Pointer berada di kiri atas* puzzle

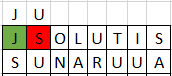
1. Cek kondisi arah pengecekan sehingga ketika arah pengecekan dirasakan pasti tidak akan menemukan jawaban (out of bound), arah pengecekan tersebut akan dilewati. x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi yang akan membuatkan pengecekan pada itu dilewati adalah
2. Arah kanan : y > panjang kolom matriks – panjang kata yang dicari
3. Arah kanan-bawah : y > panjang kolom matriks – panjang kata yang dicari dan x > panjang baris matriks – panjang kata yang dicari
4. Arah bawah : x > panjang baris matriks – panjang kata yang dicari
5. Arah kiri-bawah : y - panjang kata yang dicari +1 < 0 dan x > panjang baris matriks – panjang kata yang dicari
6. Arah kiri: y - panjang kata yang dicari +1 < 0
7. Arah kiri-atas : y - panjang kata yang dicari +1 < 0 dan x – panjang kata yang dicari + 1 < 0
8. Arah atas : x – panjang kata yang dicari + 1 < 0
9. Arah kanan-atas : y > panjang kolom matriks – panjang kata yang dicari dan x – panjang kata yang dicari + 1 < 0
10. Kemudian, pada *pointer* tersebut, bandingkan kata yang akan dicari dengan huruf-huruf yang tersedia di matrix *puzzle* dalam arah pengecekan yang memenuhi syarat.
11. Pada pengecekan :
12. ke arah kanan



*Gambar 17 Pengecekan ke arah kanan*

Ketika kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kiri kanan. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda



*Gambar 18 Pengecekan huruf yang sesuai (J) dan tidak sesuai (U)*

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah kanan-bawah



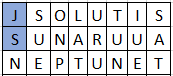
*Gambar 19 Pengecekan ke arah kanan-bawah*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kiri-atas ke kanan-bawah. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah bawah



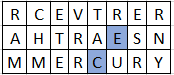
*Gambar20 Pengecekan ke arah bawah*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari atas ke bawah. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah kiri-bawah



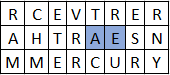
*Gambar 21 Pengecekan ke arah kiri-bawah*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kanan-atas ke kiri-bawah. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah kiri



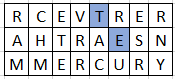
*Gambar 22 Pengecekan ke arah kiri*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kiri ke kanan. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah kiri-atas



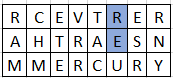
*Gambar 23 Pengecekan ke arah kiri-atas*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kanan-bawah ke kiri-atas. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah atas



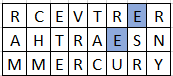
*Gambar 24 Pengecekan ke arah atas*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari bawah ke atas. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.

1. ke arah kanan-atas

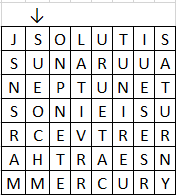


*Gambar 25 Pengecekan ke arah kanan-atas*

kata yang dicari dengan huruf-huruf pada puzzle dibandingkan satu per satu dari kiri-bawah ke kanan-atas. Jika salah satu dari kondisi ini terpenuhi maka pengecekan akan dihentikan. Misalkan i adalah indeks pencarian kata, x adalah indeks baris puzzle dan y adalah indeks kolom puzzle, kondisi-kondisi tersebut adalah

* i sudah sama dengan panjang kata yang dicari
* pada waktu pengecekan antara huruf pada kata dan huruf pada matrix sudah berbeda

Jika i sama dengan panjang kata yang dicari, langsung lompat ke langkah empat. Jika tidak, lakukan pengecekan kembali ke arah selanjutnya.



*Gambar 26 Menggeser Pointer*

1. Jika dari hasil pengecekan belum menemui susunan huruf yang sesuai, geser *pointer* itu menunju huruf selanjutnya pada matrix *puzzle*, kemudian ulangi langkah ke dua.
2. Jika dari hasil pengecekan ditemukan susunan huruf yang sesuai dengan huruf yang di cari, tampilkan total waktu eksekusi program dan jumlah perbandingan huruf yang dilakukan serta dengan arah pencarian mana susunan huruf tersebut ditemukan.



*Gambar 12* Output *ketika kata berhasil ditemukan*

1. Jika hingga pointer menunjukkan index terakhir matrix dan kata yang dicari tidak ada, tampilkan pesan “kata tidak ditemukan dalam puzzle”.

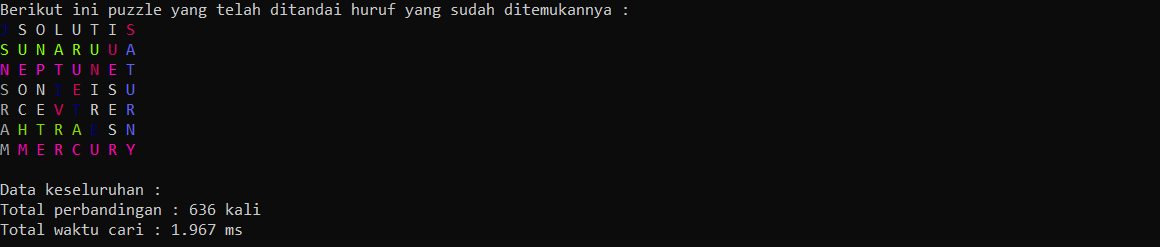


*Gambar 27 Pointer sudah berada diujung matriks*



*Gambar 28* Output *ketika kata gagal ditemukan*

1. Ulangi langkah ke satu hingga seluruh kata yang dicari sudah dicek keberadaannya di dalam *puzzle*.
2. Tampilan matriks *puzzle* hasil pencarian dengan memberi warna berbeda pada kat-kata yang berhasil ditemukan dan tunjukan waktu ekseskusi dan juga jumlah perbandingan yang dilakukan.



*Gambar 29* Output *akhir*

BAB 3: *Source Code* Program

* 1. SearchWord.java

|  |
| --- |
| import java.util.Scanner;  public class SearchWord {      public static void main(String[] args) {          Scanner key = new Scanner(System.in);          System.out.print("Masukan nama file (lengkap dengan .txt nya) : ");          String path = key.nextLine();          solve(path);          key.close();      }        public static void solve(String path) {          Query filQuery = Query.readPuzzle(path);            char[][] puzzle = filQuery.puzzle;          String[] word = filQuery.word;          int[][] puzzleStat = filQuery.puzzleStat;          System.out.printf("Masukan versi bruteforce yang diinginkan (1/2) : %n1. Dengan Heuristik %n2. Tanpa Heuristik %n>> ");          Scanner op = new Scanner(System.in);          int ver = op.nextInt();          if (ver == 1) {              BruteForceAlgorithm.searchingword(puzzle, word, puzzleStat);          } else {              BruteForceAlgorithm.searchingwordv2(puzzle, word, puzzleStat);          }          op.close();      }  } |

* 1. Query.java

|  |
| --- |
| import java.util.ArrayList;  import java.util.Scanner;  import java.io.File;  import java.io.FileNotFoundException;  public class Query {      public char[][] puzzle;      public int[][] puzzleStat;      public String[] word;      public Query(char[][] \_puzzle, int[][] \_puzzleStat, String[] \_word) {          puzzle = \_puzzle;          puzzleStat = \_puzzleStat;          word = \_word;      }      public static Query readPuzzle(String path) {          ArrayList<ArrayList<String>> temp = new ArrayList<ArrayList<String>> ();          ArrayList<String> dump = new ArrayList<String>();          char[][] puzzlefail = new char[0][0];          int[][] puzzleStatfail = new int[0][0];          String[] wordfail = new String[0];          int i = 0, j;          String filePath = "../test/" + path;          File dataFile = new File(filePath);          Boolean scanPuzzle = true;            Scanner dataScanner;          try {              dataScanner= new Scanner(dataFile);                while (dataScanner.hasNextLine() && scanPuzzle) {                  String line = dataScanner.nextLine();                  if (line == "") {                      scanPuzzle = false;                  } else {                      String[] rowPuzzle = line.split(" ");                        ArrayList<String> tempList = new ArrayList<String>();                      for (i = 0; i < rowPuzzle.length; i++) {                          tempList.add(rowPuzzle[i]);                        }                      temp.add(tempList);                  }              }              while (dataScanner.hasNextLine()) {                  String line = dataScanner.nextLine();                  dump.add(line);              }              dataScanner.close();              char[][] puzzle = new char[temp.size()][temp.get(0).size()];              int[][] puzzleStat = new int[temp.size()][temp.get(0).size()];              String[] word = new String[dump.size()];              for (i = 0; i < temp.size(); i++) {                  for (j = 0; j < temp.get(i).size(); j++) {                      puzzle[i][j] = temp.get(i).get(j).charAt(0);                      puzzleStat[i][j] = -1;                  }              }              for (i = 0; i < dump.size(); i++) {                  word[i] = dump.get(i);              }              Query filQuery = new Query(puzzle, puzzleStat, word);              return filQuery;          } catch (FileNotFoundException e) {              System.out.println("File tidak ditemukan");              System.exit(0);          }          Query filQuery = new Query(puzzlefail, puzzleStatfail, wordfail);          return filQuery;      }  } |

* 1. Helper.java

|  |
| --- |
| public class Helper {      public static String coloring(int n) {          if (n == -1) {              return "\033[0m";          } else {              return "\033[38;5;" + Integer.toString(n) + "m";          }      }  } |

* 1. Direction.java

|  |
| --- |
| public class Direction {      public static int checkRight(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i+y < puzzle[x].length && i < arrword.length && puzzle[x][i+y] == arrword[i] ) {              i++;          }          return (i+1);      }      public static int checkLeft(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (y-i >= 0 && i < arrword.length && puzzle[x][y-i] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkUp(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (x-i >= 0 && i < arrword.length && puzzle[x-i][y] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkDown(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i >= 0 && i+x < puzzle.length && i < arrword.length && puzzle[i+x][y] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkUpRight(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i+y < puzzle[x].length && x-i >= 0 && i < arrword.length && puzzle[x-i][i+y] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkDownRight(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i+y < puzzle[x].length && i+x < puzzle.length && i < arrword.length && puzzle[i+x][i+y] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkUpLeft(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (y-i >= 0 && x-i >= 0 && i < arrword.length && puzzle[x-i][y-i] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkDownLeft(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (y-i >= 0 && i+x < puzzle.length && i < arrword.length && puzzle[i+x][y-i] == arrword[i]) {              i++;          }          return (i+1);      }  } |

* 1. DirectionHeuristic.java

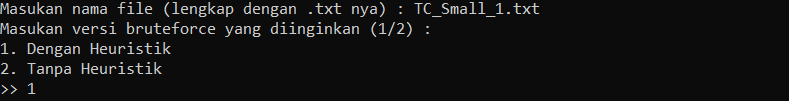
|  |
| --- |
| public class DirectionHeuristic {      public static int checkRight(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i < arrword.length && puzzle[x][i+y] == arrword[i] ) {              i++;          }          return (i+1);      }      public static int checkLeft(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i < arrword.length && puzzle[x][y-i] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkUp(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i < arrword.length && puzzle[x-i][y] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkDown(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i < arrword.length && puzzle[i+x][y] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkUpRight(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i < arrword.length && puzzle[x-i][i+y] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkDownRight(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i < arrword.length && puzzle[i+x][i+y] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkUpLeft(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i < arrword.length && puzzle[x-i][y-i] == arrword[i]) {              i++;          }          return (i+1);      }      public static int checkDownLeft(char[][] puzzle, char[] arrword, int x, int y) {          int i = 0;          while (i < arrword.length && puzzle[i+x][y-i] == arrword[i]) {              i++;          }          return (i+1);      }  } |

* 1. BruteForceAlgorithm.java

|  |
| --- |
| public class BruteForceAlgorithm {      public static void searchingword(char[][] puzzle, String[] word, int[][] puzzleStat) {          int i, j, k, n, sum, sumFind, sumAll = 0;            Long totalWaktu = (long) 0;          for (n = 0; n < word.length; n++) {              char[] arrword = word[n].toCharArray();              i = 0;              boolean found = false;              sumFind = 0;              Long startTimeTemp = System.nanoTime();              while ( i < puzzle.length && !found) {                  j = 0;                  while (j < puzzle[i].length && !found) {                      if (j <= puzzle[i].length - arrword.length && !found) {                          sum = DirectionHeuristic.checkRight(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i][j+k] = (n%25)+196;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (i <= puzzle.length - arrword.length && j <= puzzle[i].length - arrword.length && !found) {                          sum = DirectionHeuristic.checkDownRight(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i+k][j+k] = (n%25)+17;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan atas dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (i <= puzzle.length - arrword.length && !found) {                          sum = DirectionHeuristic.checkDown(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i+k][j] = (n%25)+58;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah bawah dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (i <= puzzle.length - arrword.length && j - arrword.length + 1 >= 0 && !found) {                          sum = DirectionHeuristic.checkDownLeft(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i+k][j-k] = (n%25)+214;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kiri atas dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (j - arrword.length + 1 >= 0 && !found) {                          sum = DirectionHeuristic.checkLeft(puzzle, arrword, i, j);                              if (sum == arrword.length+1) {                                  found = true;                                  for (k = 0; k < arrword.length; k++) {                                      puzzleStat[i][j-k] = (n%25)+112;                                  }                                  System.out.printf("%s ditemukan dalam pencarian ke arah kiri dengan ", word[n]);                                  sumFind += sum - 1;                              } else {                                  sumFind += sum;                              }                      }                      if (i - arrword.length + 1 >= 0 && j - arrword.length + 1 >= 0 && !found) {                          sum = DirectionHeuristic.checkUpLeft(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i-k][j-k] = (n%25)+178;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kiri atas dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (i - arrword.length + 1 >= 0 && j <= puzzle[i].length - arrword.length && !found) {                          sum = DirectionHeuristic.checkUpRight(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i-k][j+k] = (n%25)+154;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan bawah dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (i - arrword.length + 1 >= 0 && !found) {                          sum = DirectionHeuristic.checkUp(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i-k][j] = (n%25)+246;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah atas dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                        j++;                  }                  i++;              }              Long endTimeTemp = System.nanoTime();              sumAll += sumFind;              Long totalWaktuTemp = endTimeTemp - startTimeTemp;              totalWaktu += totalWaktuTemp;              Double time = (double) totalWaktuTemp/1000000;              if (found) {                  System.out.printf("total perbandingan : %d kali dan total waktu cari : %.3f ms\n", sumFind, time);                  System.out.println();              } else {                  System.out.printf("%s tidak berada didalam puzzle", word[n]);              }          }          System.out.println("Berikut ini puzzle yang telah ditandai huruf yang sudah ditemukannya :");          for (i = 0; i < puzzle.length; i++) {              for (j = 0; j < puzzle[i].length; j++) {                  System.out.print(Helper.coloring(puzzleStat[i][j]) + puzzle[i][j] + " ");              }              System.out.println();          }          System.out.println(Helper.coloring(-1));          Double time = (double) totalWaktu/1000000;          System.out.printf("Data keseluruhan : %nTotal perbandingan : %d kali %nTotal waktu cari : %.3f ms\n", sumAll, time);      }      public static void searchingwordv2(char[][] puzzle, String[] word, int[][] puzzleStat) {          int i, j, k, n, sum, sumFind, sumAll = 0;            Long totalWaktu = (long) 0;          for (n = 0; n < word.length; n++) {              char[] arrword = word[n].toCharArray();              i = 0;              boolean found = false;              sumFind = 0;              Long startTimeTemp = System.nanoTime();                while ( i < puzzle.length && !found) {                  j = 0;                  while (j < puzzle[i].length && !found) {                      if (!found) {                          sum = Direction.checkRight(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i][j+k] = (n%25)+196;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (!found) {                          sum = Direction.checkDownRight(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i+k][j+k] = (n%25)+17;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (!found) {                          sum = Direction.checkDown(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i+k][j] = (n%25)+58;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                          sum = Direction.checkDownLeft(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i+k][j-k] = (n%25)+214;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (!found) {                          sum = Direction.checkLeft(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i][j-k] = (n%25)+112;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                          sum = Direction.checkUpLeft(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i-k][j-k] = (n%25)+178;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (!found) {                          sum = Direction.checkUp(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i-k][j] = (n%25)+246;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      if (!found) {                          sum = Direction.checkUpRight(puzzle, arrword, i, j);                          if (sum == arrword.length+1) {                              found = true;                              for (k = 0; k < arrword.length; k++) {                                  puzzleStat[i-k][j+k] = (n%25)+154;                              }                              System.out.printf("%s ditemukan dalam pencarian ke arah kanan dengan ", word[n]);                              sumFind += sum - 1;                          } else {                              sumFind += sum;                          }                      }                      j++;                  }                  i++;              }              Long endTimeTemp = System.nanoTime();              sumAll += sumFind;              Long totalWaktuTemp = endTimeTemp - startTimeTemp;              totalWaktu += totalWaktuTemp;              Double time = (double) totalWaktuTemp/1000000;              if (found) {                  System.out.printf("total perbandingan : %d kali dan total waktu cari : %.3f ms\n", sumFind, time);                  System.out.println();              } else {                  System.out.printf("%s tidak berada didalam puzzle", word[n]);              }          }            System.out.println("Berikut ini puzzle yang telah ditandai huruf yang sudah ditemukannya :");          for (i = 0; i < puzzle.length; i++) {              for (j = 0; j < puzzle[i].length; j++) {                  System.out.print(Helper.coloring(puzzleStat[i][j]) + puzzle[i][j] + " ");              }              System.out.println();          }          System.out.println(Helper.coloring(-1));          Double time = (double) totalWaktu/1000000;          System.out.printf("Data keseluruhan : %nTotal perbandingan : %d kali %nTotal waktu cari : %.3f ms\n", sumAll, time);      }  } |

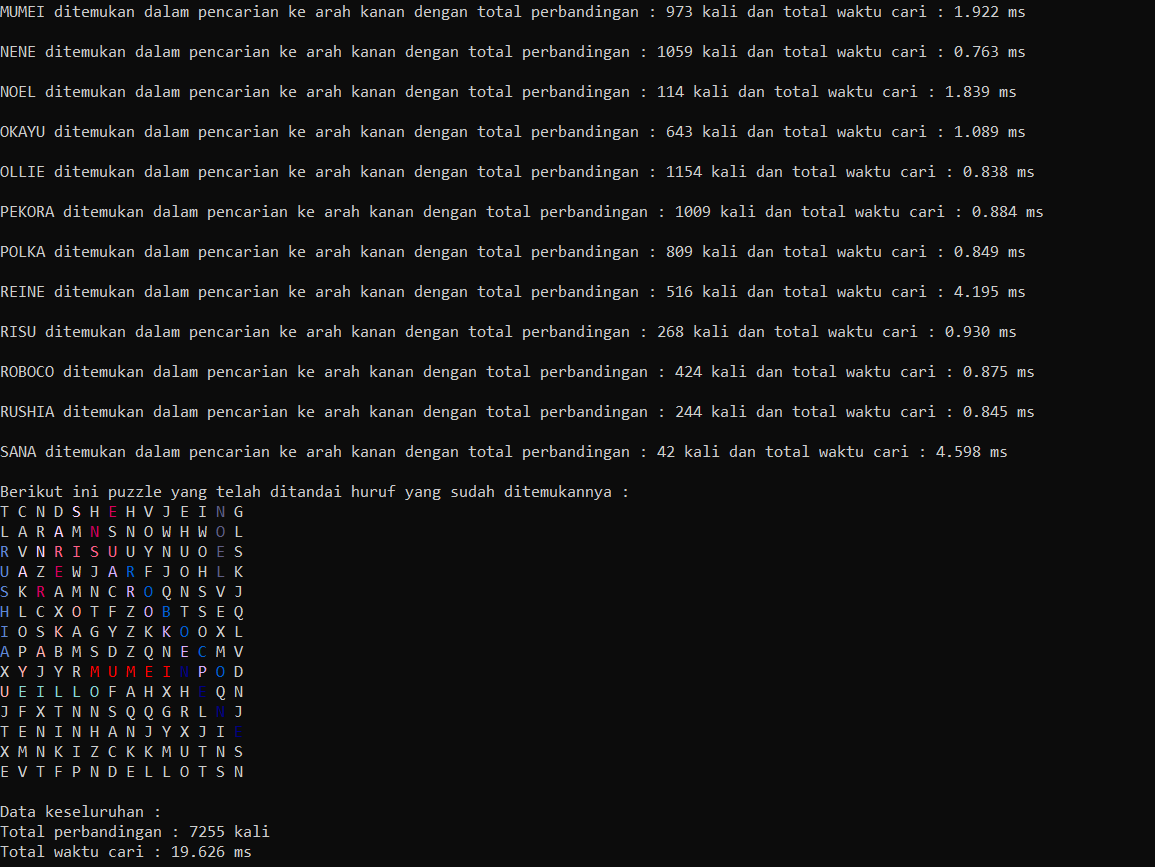
BAB 4: *Screen Shot Input Output* Program

* 1. TC Small 1 (14 x 14 ukuran puzzle dan 12 kata yang dicari)

*Gambar 30 Input TC Small 1*

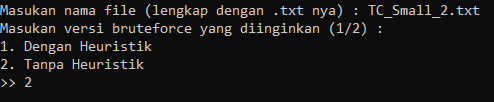


*Gambar 31 Ouput TC Small 1 dengan heuristik*

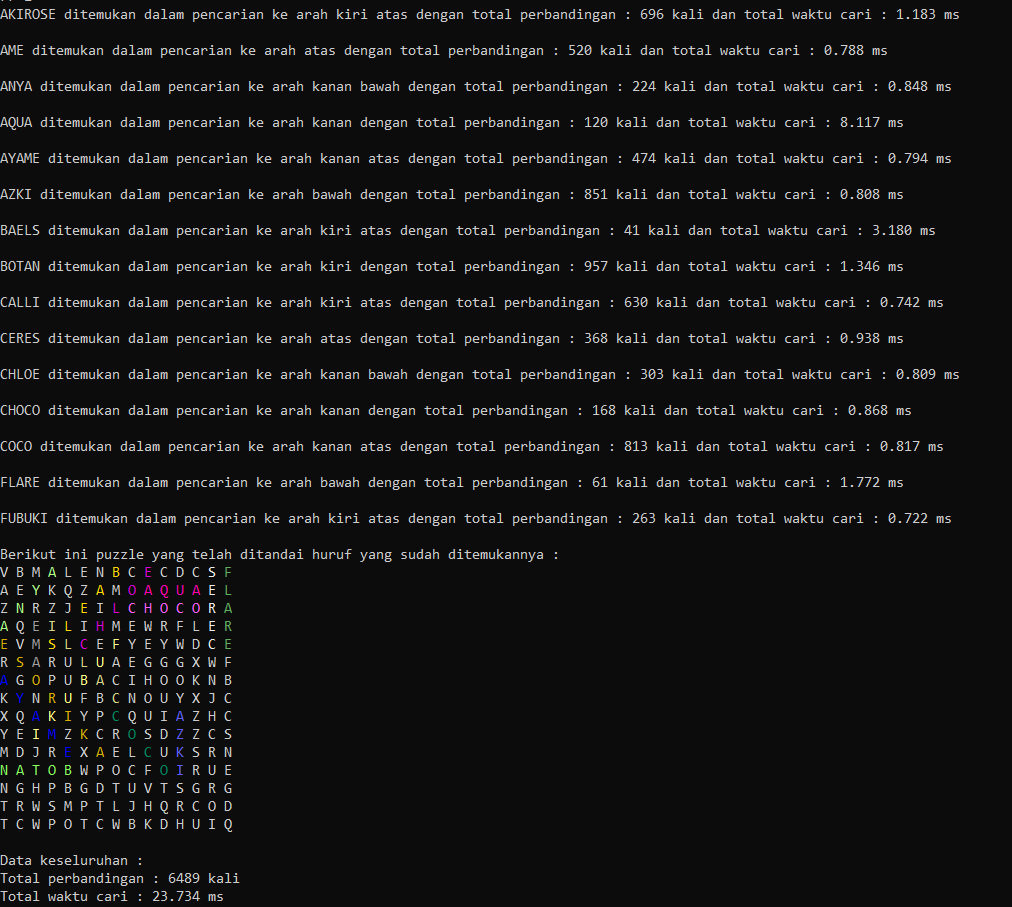


*Gambar 32 Output TC Small 1 tanpa heuristik*

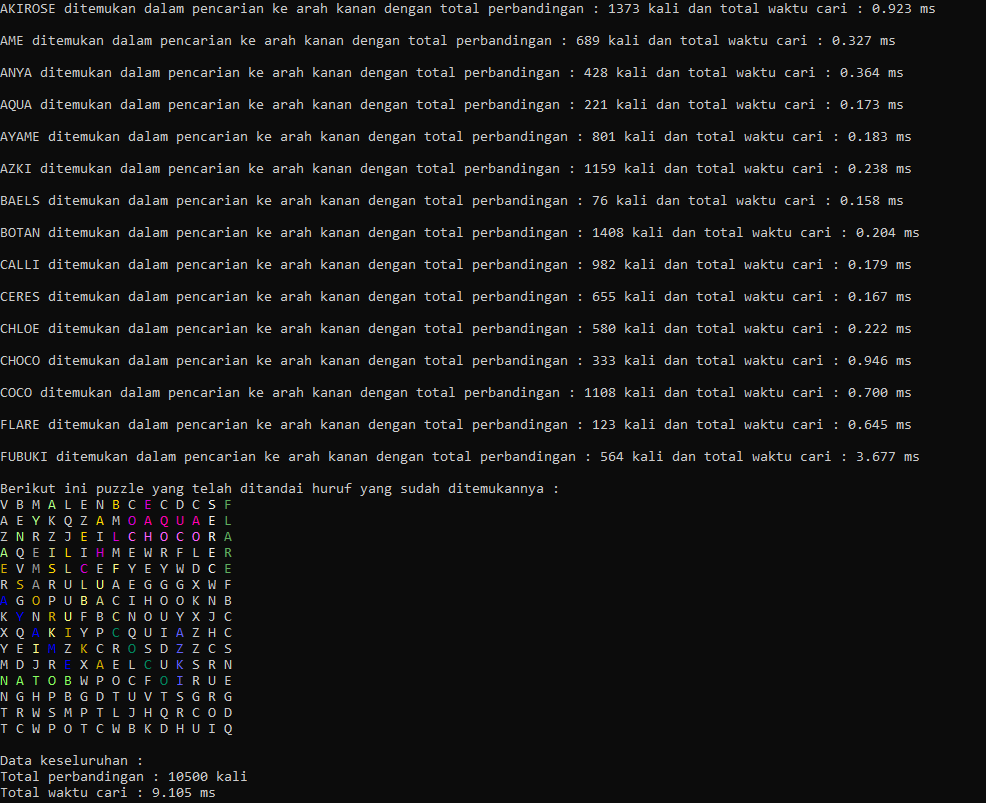
* 1. TC Small 2 (15 x 15 ukuran puzzle dan 15 kata yang dicari)



*Gambar 33 Input TC Small 2*

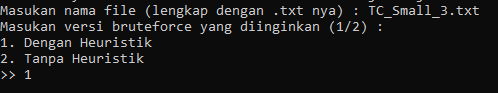


*Gambar 34 Output TC Small 2 dengan heuristik*

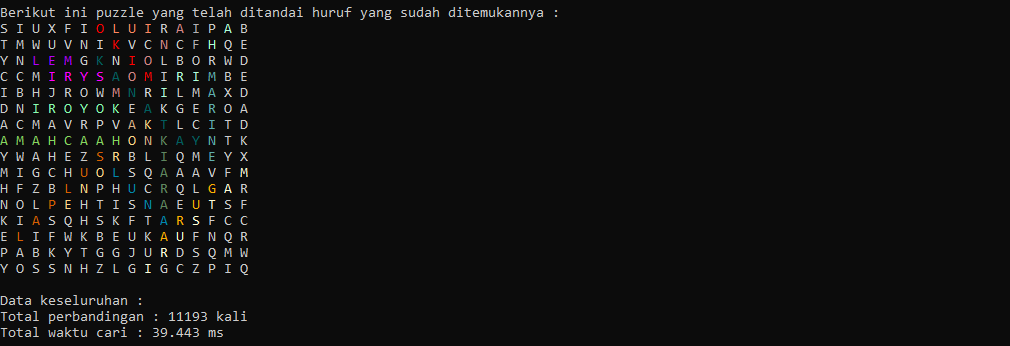
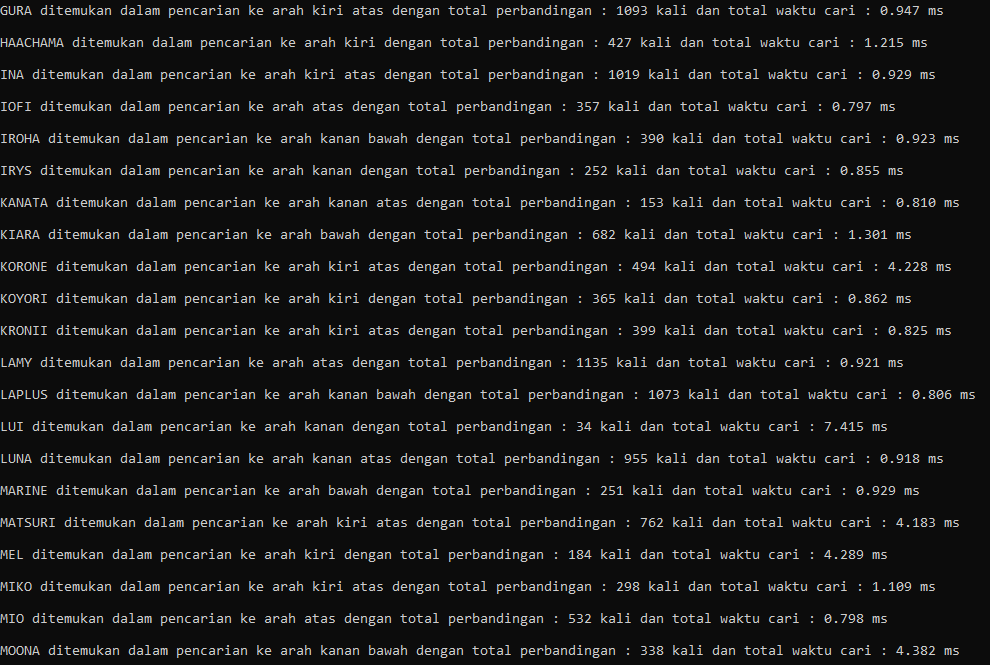


*Gambar 35 Output TC Small 2 tanpa heuristik*

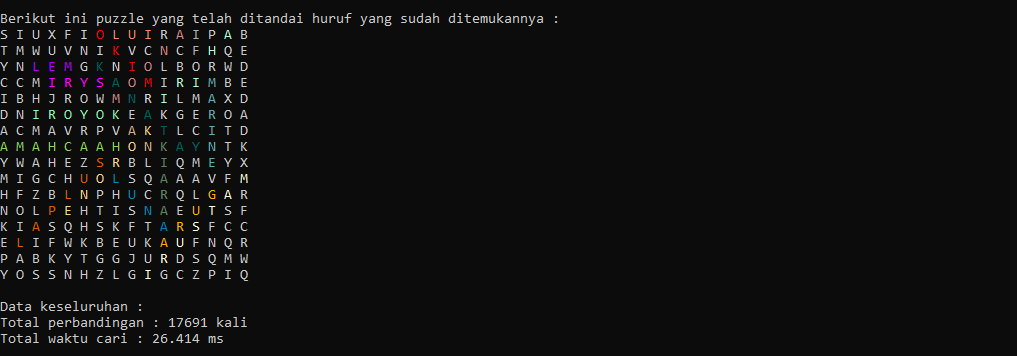
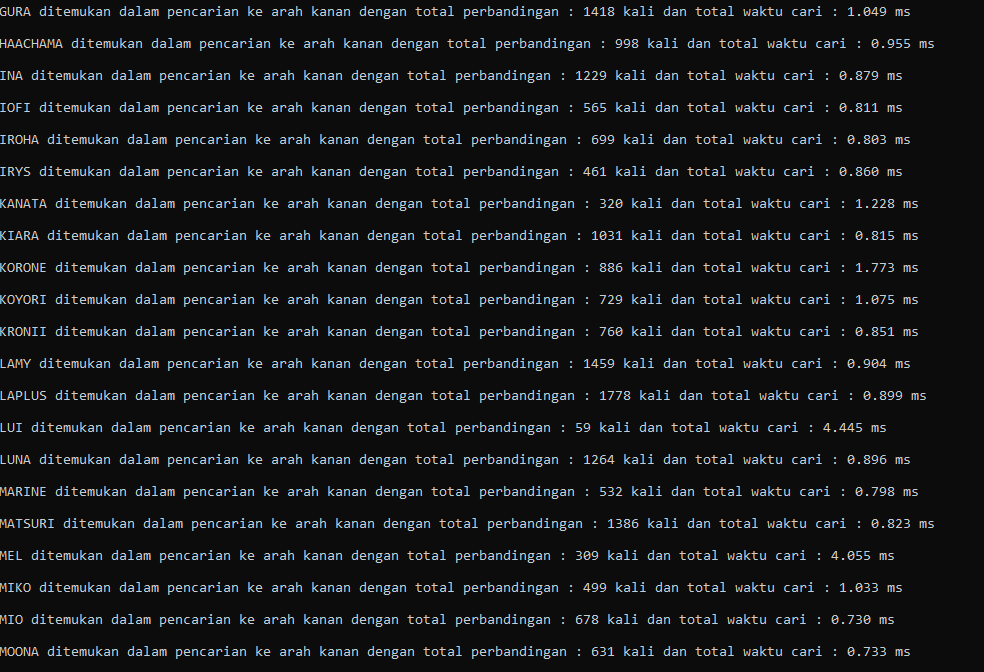
* 1. TC Small 3 (16 x 16 ukuran puzzle dengan 21 kata yang dicari)



*Gambar 36 Input TC Small 3*

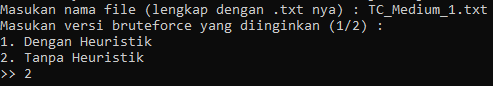


*Gambar 37 Output TC Small 3 dengan heuristik*

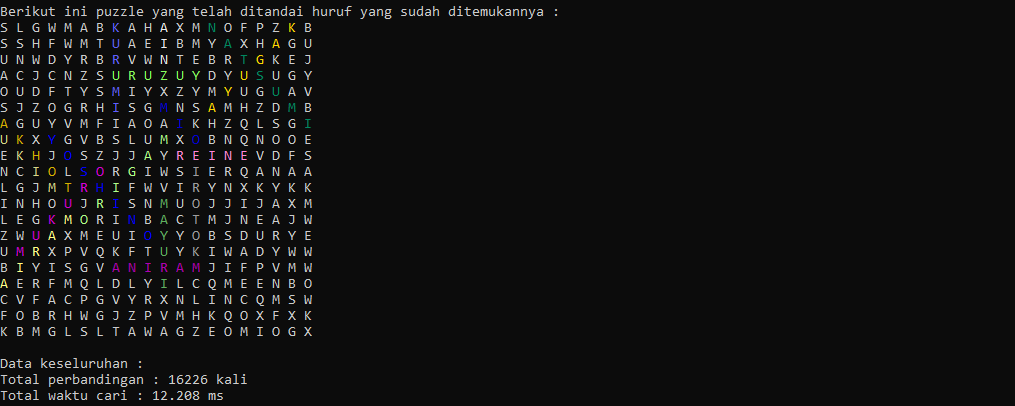
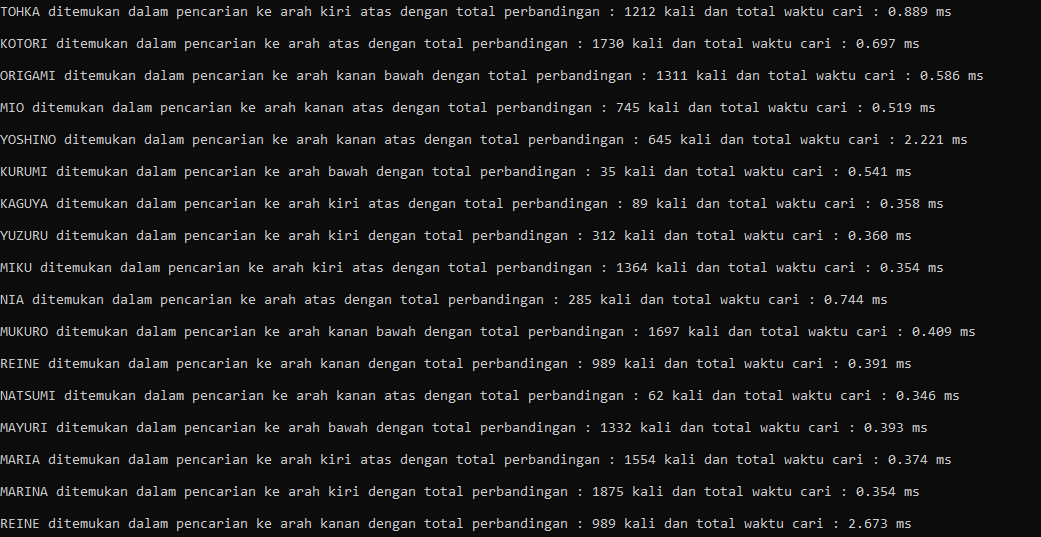


*Gambar 38 Output TC Small 3 tanpa heuristik*

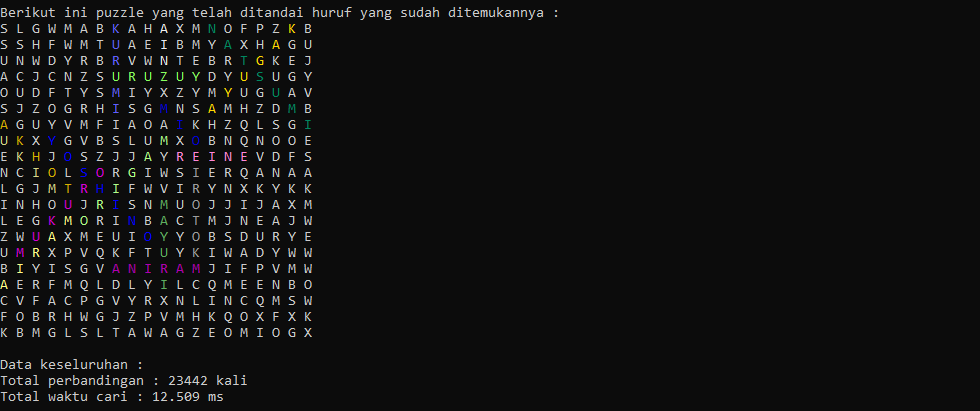
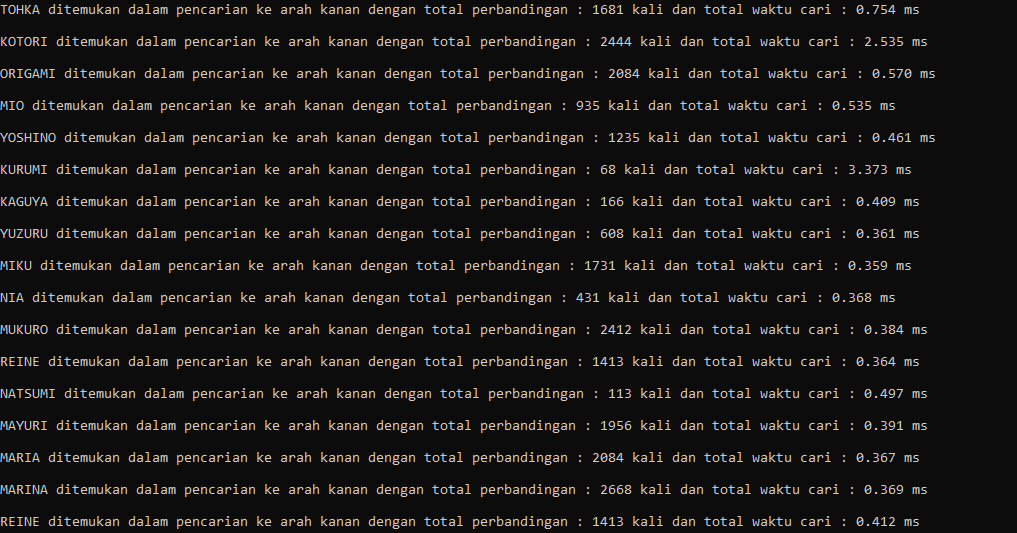
* 1. TC Medium 1 (20 x 20 ukuran puzzle dengan 17 kata yang dicari)



*Gambar 39 Input TC Medium 1*

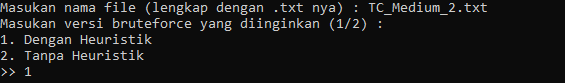


*Gambar 40 Output TC Medium 1 dengan heuristik*

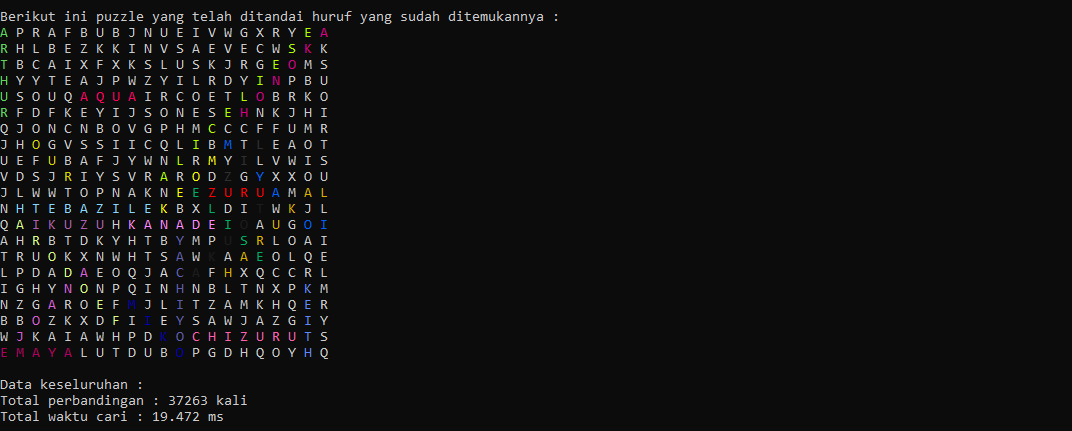
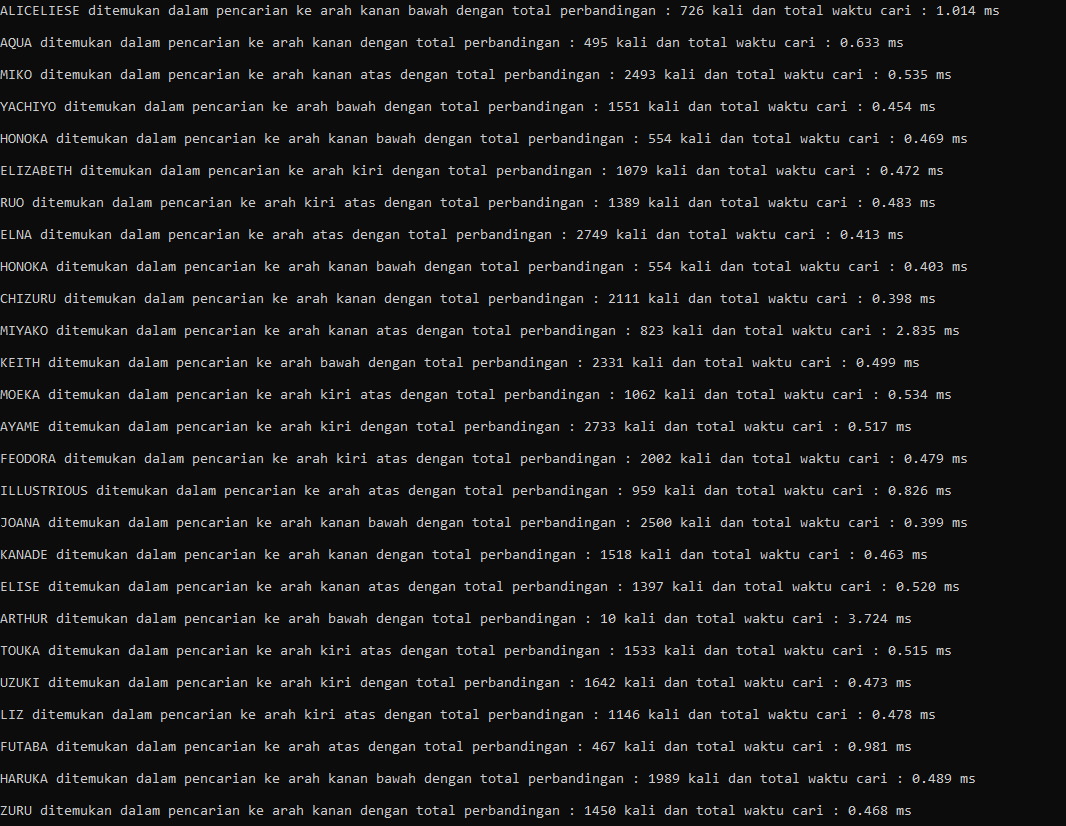


*Gambar 41 Output TC Medium 1 tanpa heuristik*

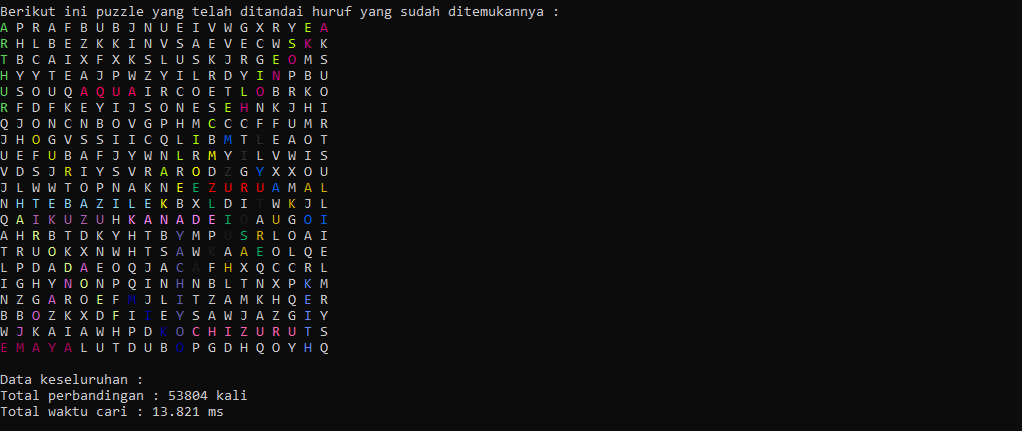
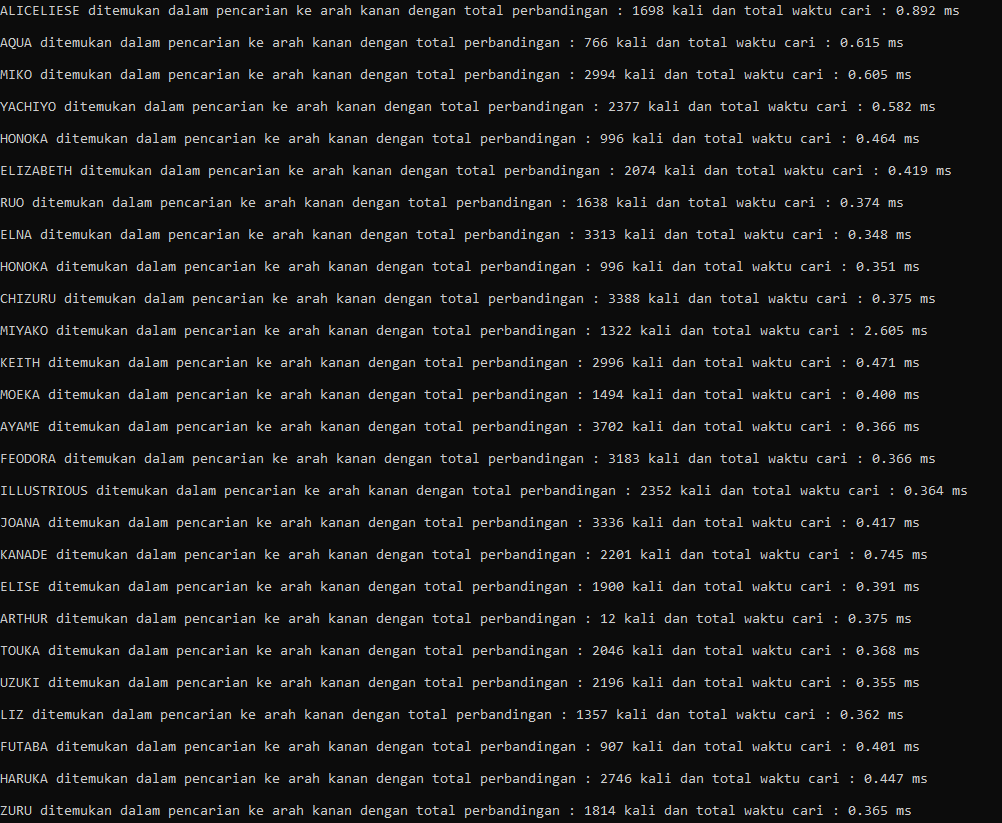
* 1. TC Medium 2 (21 x 21 ukuran puzzle dengan 26 kata yang dicari)



*Gambar 42 Input TC Medium 2*

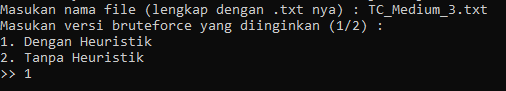


*Gambar 43 Output TC Medium 2 dengan heuristik*

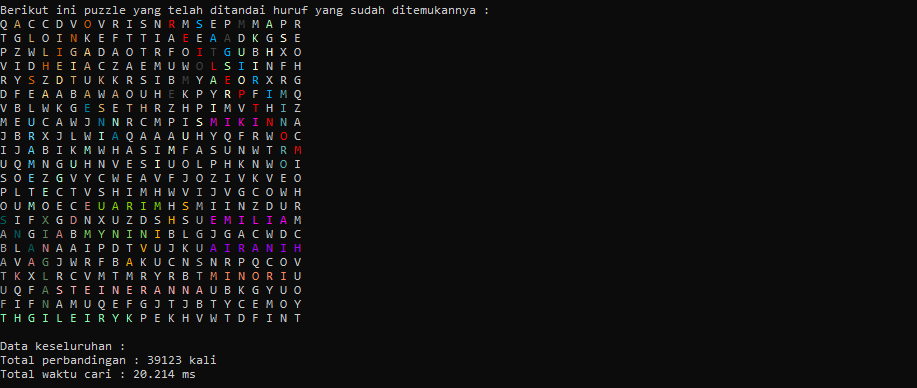
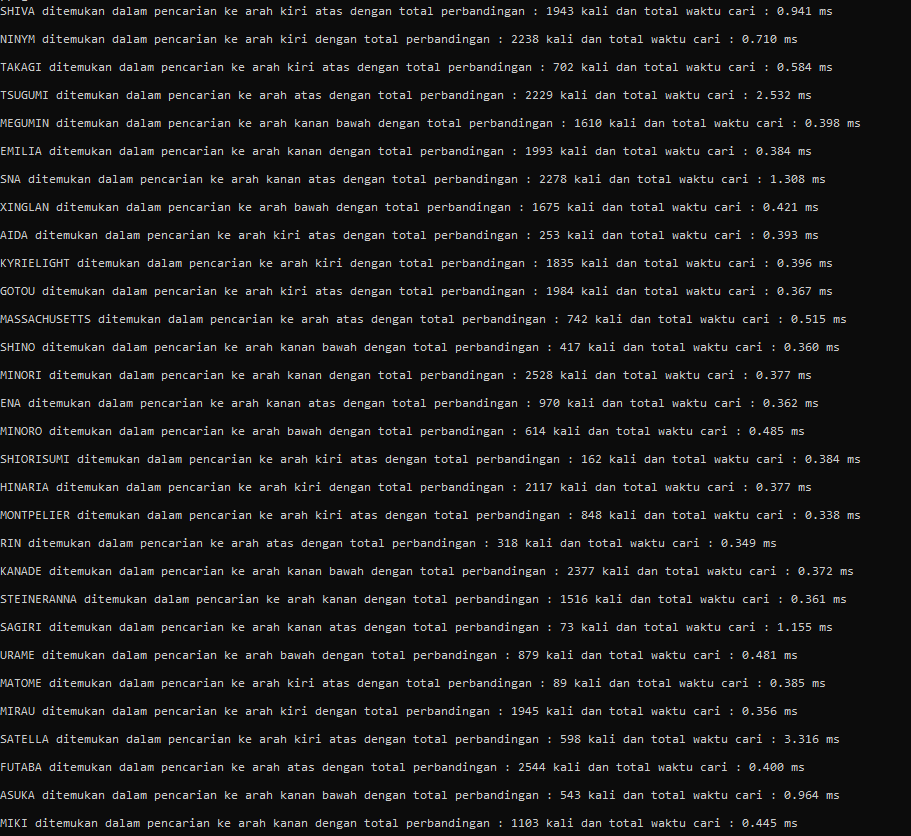


*Gambar 44 Output TC Medium 2 tanpa heuristic*

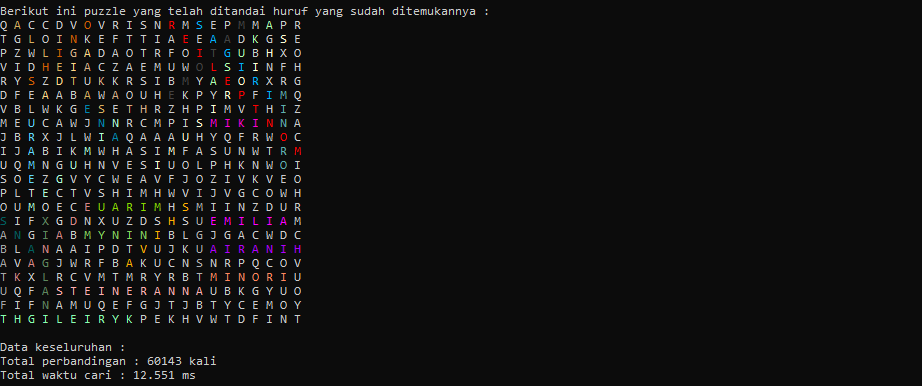
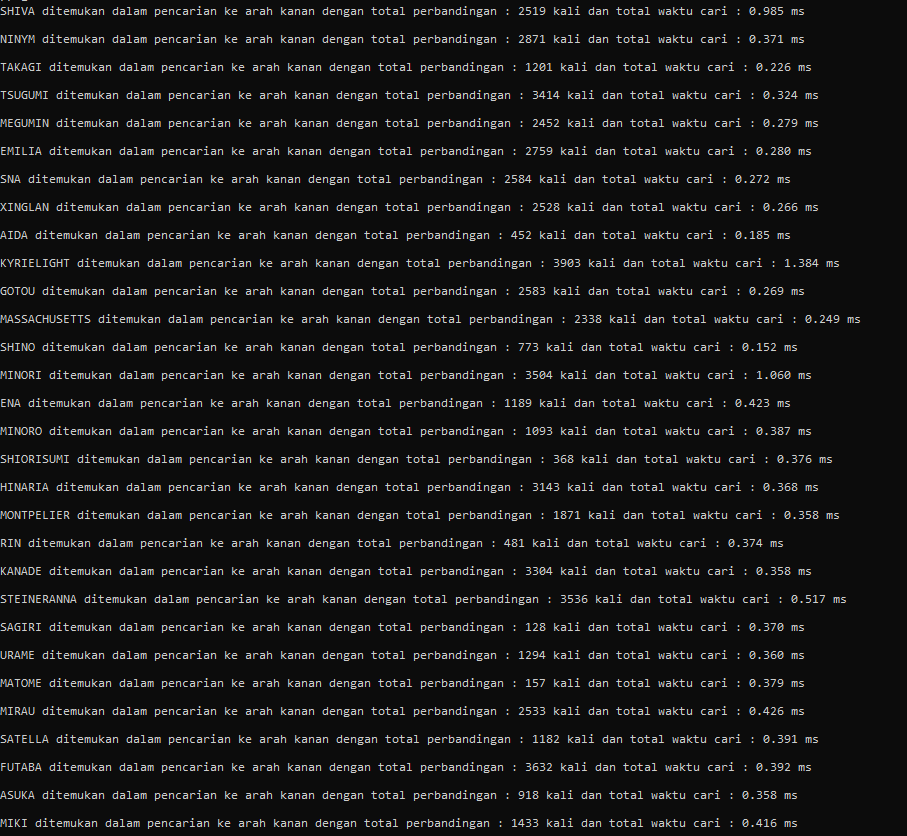
* 1. TC Medium 3 (22 x 22 ukuran puzzle dengan 30 kata yang dicari)



*Gambar 44 Input TC Medium 3*

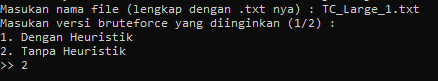


*Gambar 45 Output TC Medium 3 dengan heuristik*

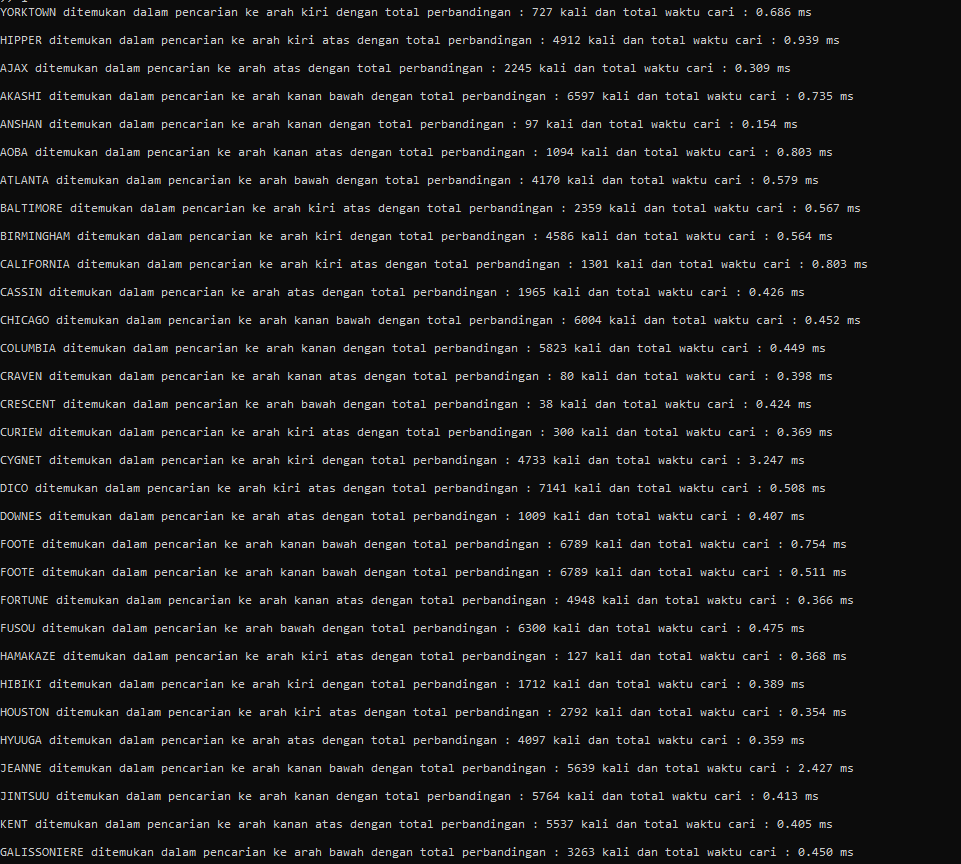


*Gambar 46 Output TC Medium 3 tanpa heuristic*

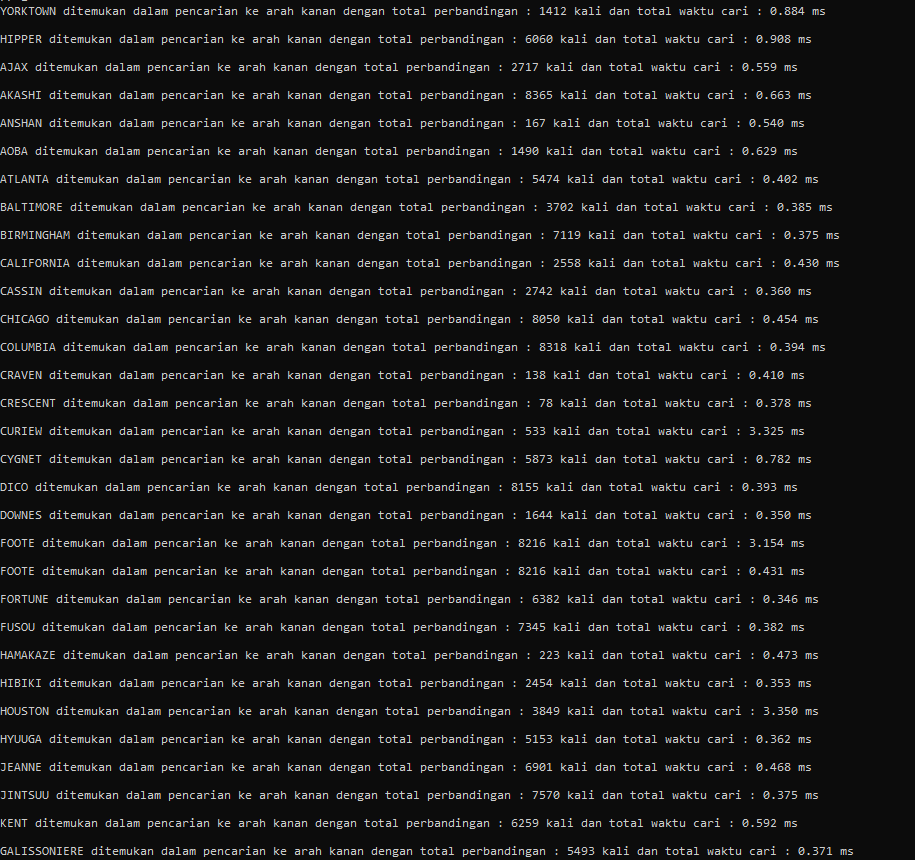
* 1. TC Large 1 (32 x 32 ukuran puzzle dengan 41 kata yang dicari)



*Gambar 47 Input TC Large 1*

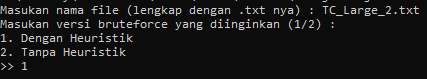


*Gambar 48 Output TC Large 1 dengan heuristik*

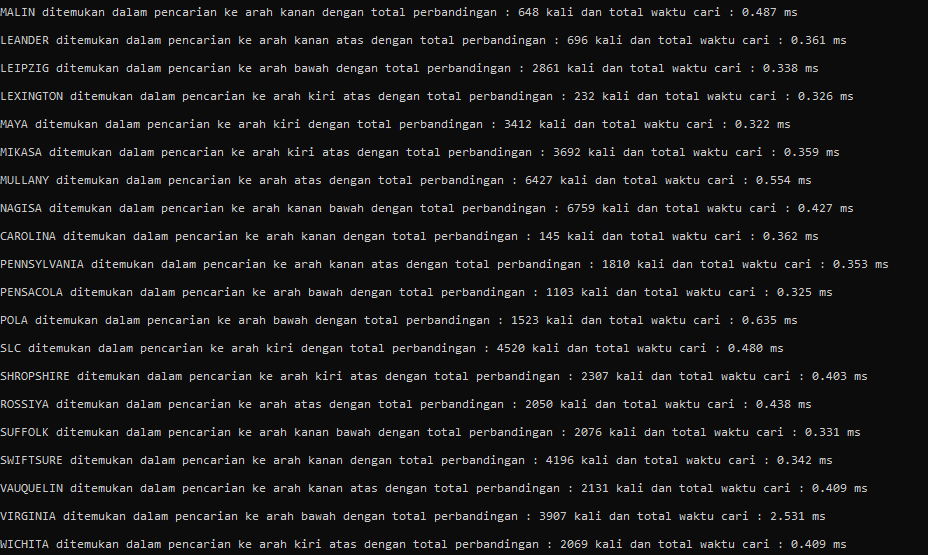
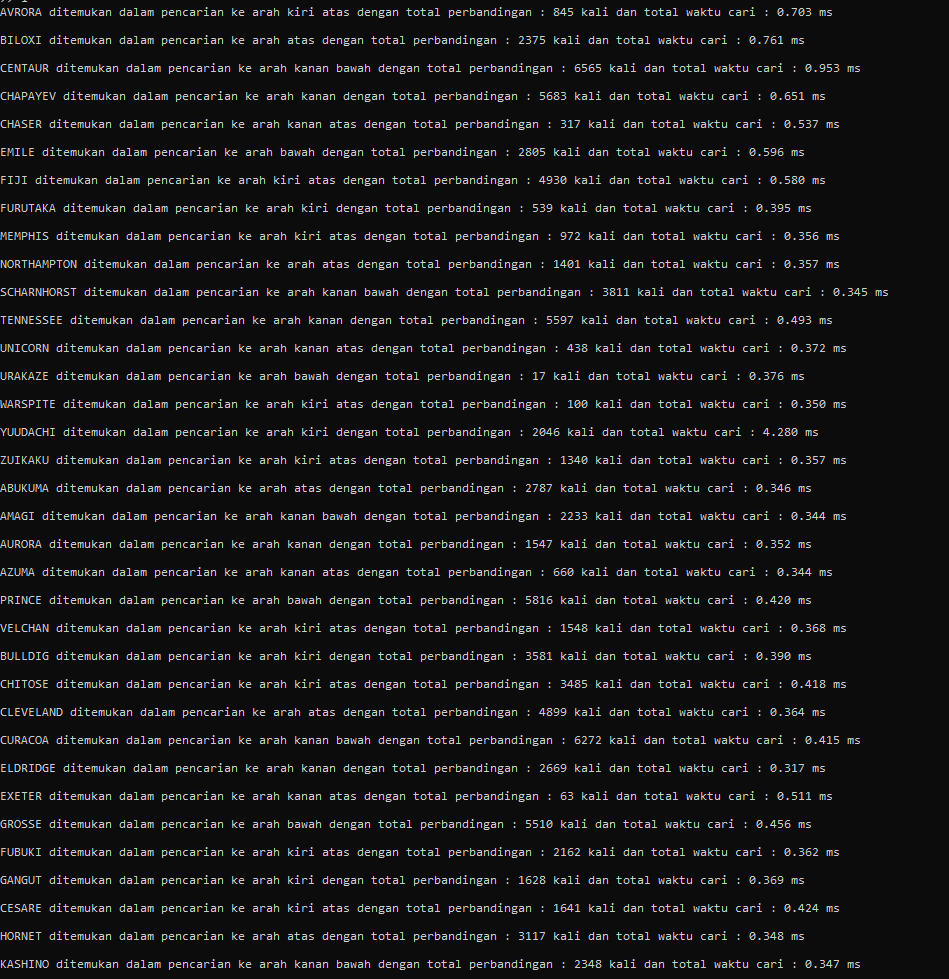


*Gambar 49 Output TC Large 1 tanpa heuristik*

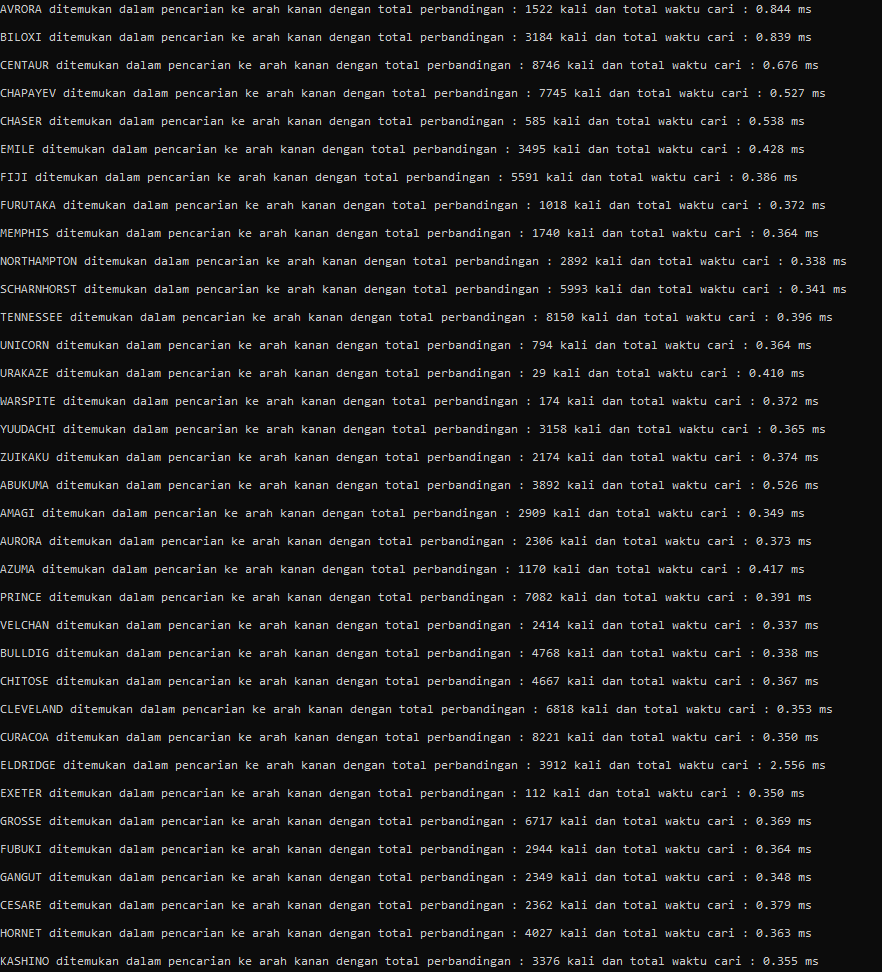
* 1. TC Large 2 (33 x 33 ukuran puzzle dengan 55 kata yang dicari)

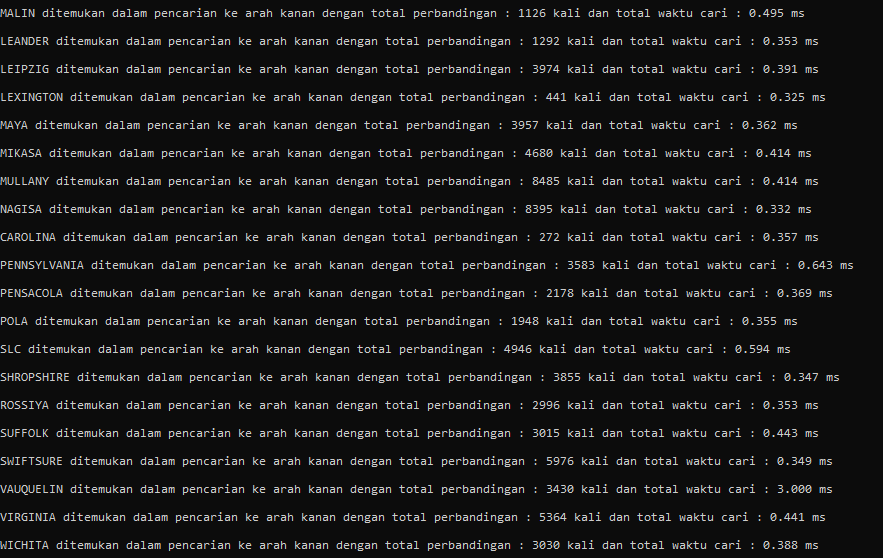


*Gambar 50 Input TC Large 2*



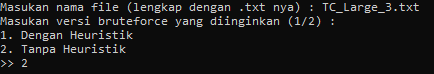
*Gambar 51 Output TC Large 2 dengan heuristik*



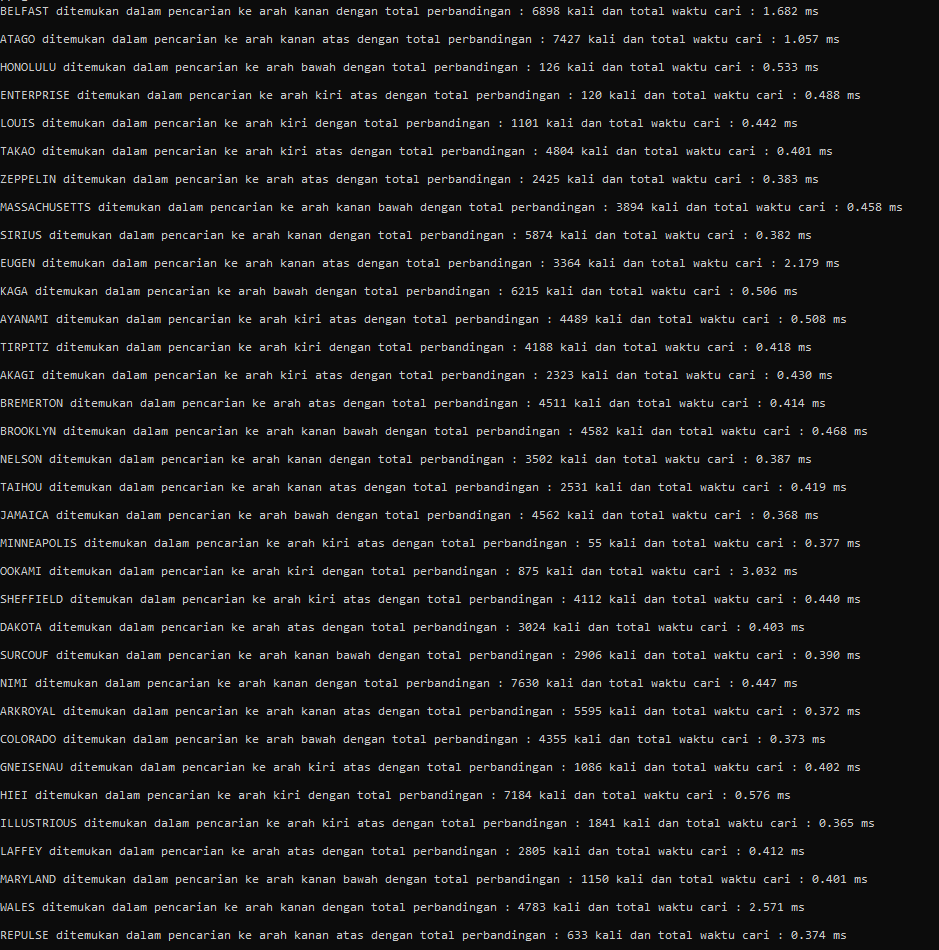


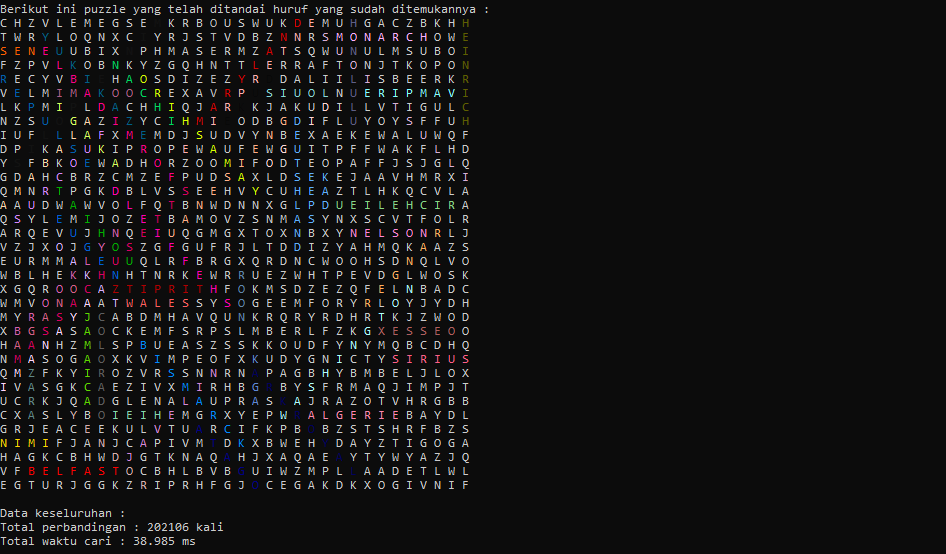
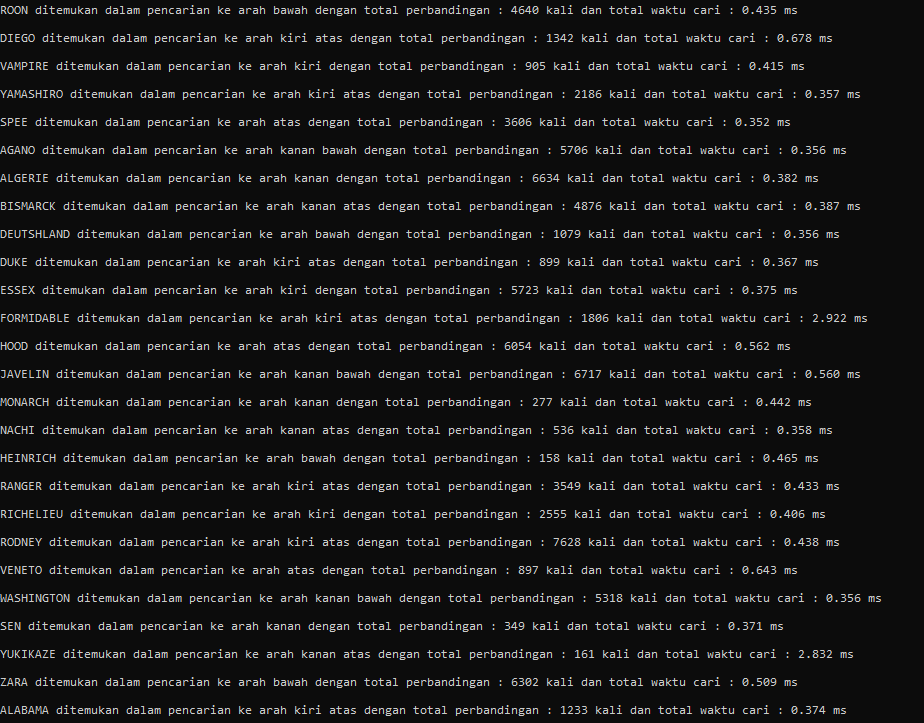
*Gambar 52 Output TC Large 2 tanpa heuristic*

* 1. TC Large 3 (34 x 34 ukuran puzzle dengan 60 kata yang dicari)

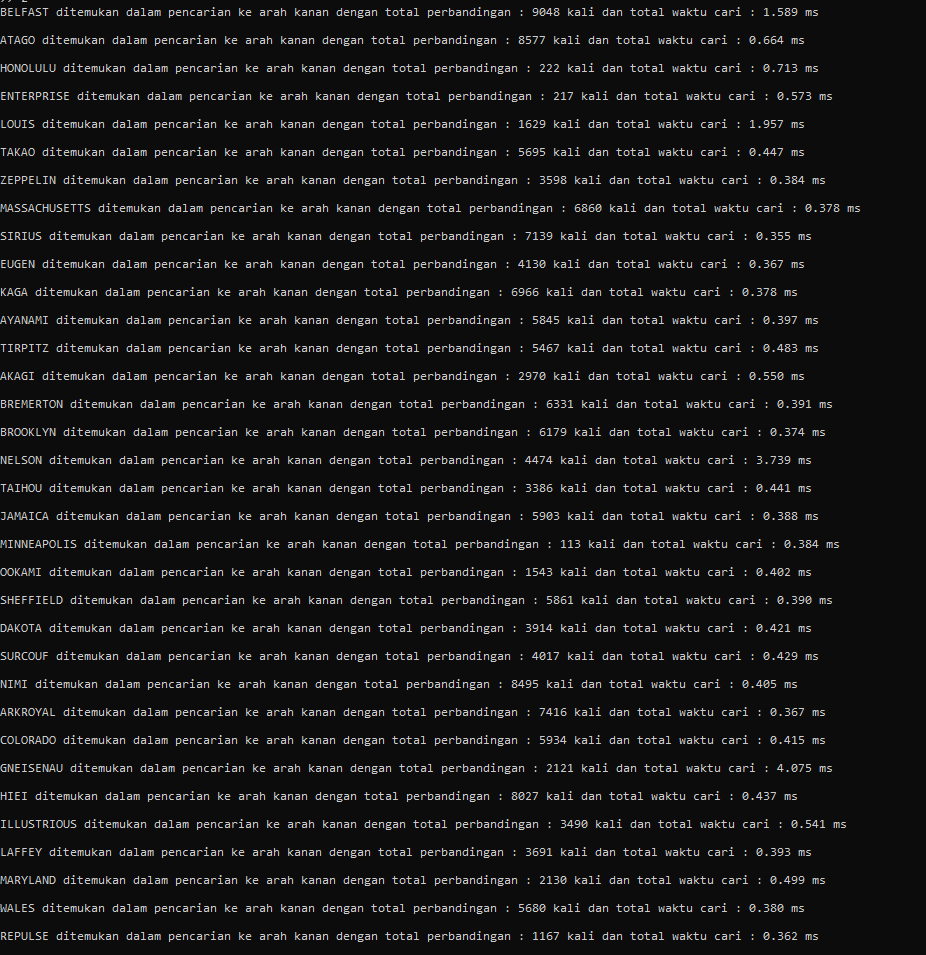


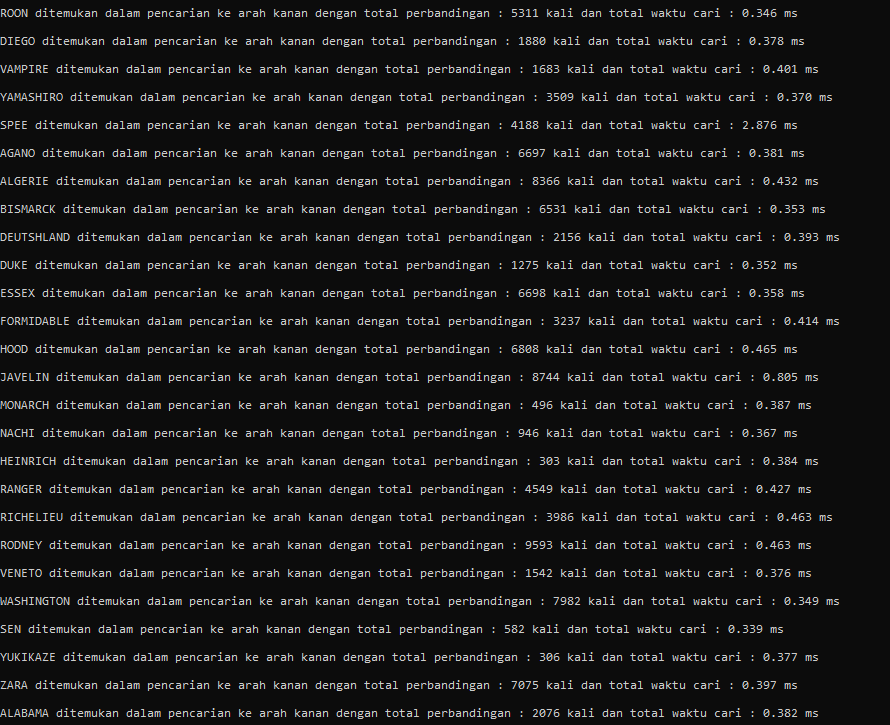
Gambar 53 Input TC Large 3





*Gambar 53 Output TC Large 3 dengan heuristik*





*Gambar 54 Output TC Large 3 tanpa heuristik*

BAB 5: Link *G-Drive*

|  |  |  |
| --- | --- | --- |
| Poin. | Ya | Tidak |
| 1. Program berhasil dikompilasi tanpa kesalahan (no syntax error) |  |  |
| 2. Program berhasil running |  |  |
| 3. Program dapat membaca file masukan dan menuliskan luaran. |  |  |
| 4. Program berhasil menemukan semua kata di dalam puzzle |  |  |