

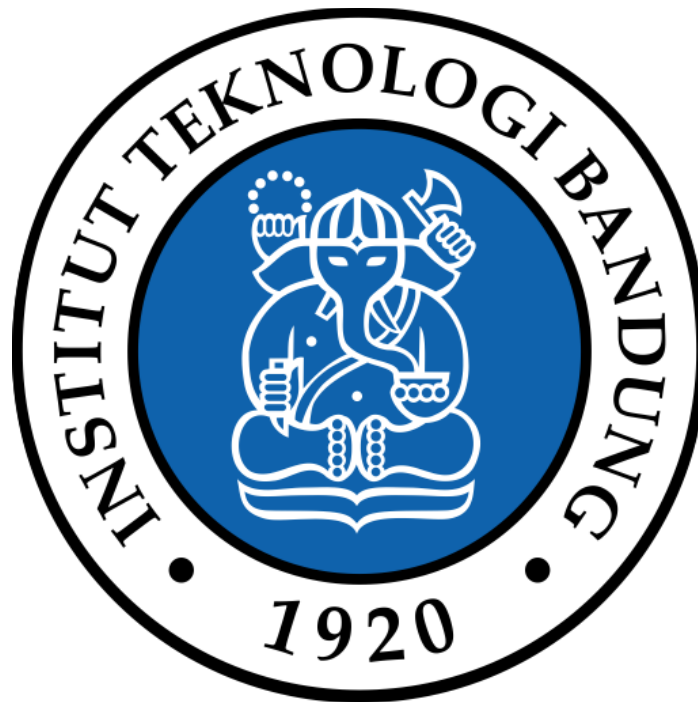
Tugas Kecil Strategi Algoritma

**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

A. 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 search puzzle* 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.

Selanjutnya akan dijelaskan langkah-langkah pada algoritma yang penulis buat. Pertama-tama, ADT matriks dan *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 melewati 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*.

B. Source Program Dalam Bahasa Java

1. Inisialisasi Awal

```

1 static ArrayList<ArrayList<Character>> matrix = new ArrayList<ArrayList<Character>>();
2 static Path curPath = Paths.get(System.getProperty("user.dir"));
3 static LinkedList<String> words = new LinkedList<String>();
4 static String[] ans;
5 static int[][] ansCoord;
6 static int ansIdx = 0;
7 static int comparisonCount = 0;

```

2. File Reader

```

1 public static int readFile(String fileName) {
2     Path filePath = Paths.get("test", fileName);
3     File file = new File(filePath.toString());
4     String fileLine;
5     boolean readPuzzle = true;
6     try (Scanner reader = new Scanner(file)) {
7         while(reader.hasNextLine()) {
8             fileLine = reader.nextLine();
9             if (fileLine.equals("")) {
10                 readPuzzle = false;
11                 fileLine = reader.nextLine();
12             }
13
14             if (readPuzzle) {
15                 ArrayList<Character> lineArray = new ArrayList<Character>();
16                 for (String alphabet: fileLine.split(" ")) {
17                     lineArray.add(alphabet.charAt(0));
18                 }
19                 matrix.add(lineArray);
20             } else {
21                 words.add(fileLine);
22             }
23         }
24         return words.size();
25     } catch (FileNotFoundException e) {
26         System.out.println("File tidak ditemukan!");
27         return -1;
28     }
29 }

```

3. Prosedur Penyelesaian *Puzzle*

```

1 public static void solvePuzzle() {
2     int i, j;
3     System.out.println("\nSolving...\n");
4     Instant start = Instant.now();
5
6     for (i = 0; i < matrix.size() && !words.isEmpty(); i++) {
7         for (j = 0; j < matrix.get(0).size() && !words.isEmpty(); j++) {
8             solveUp(i, j);
9             solveUpRight(i, j);
10            solveRight(i, j);
11            solveBottomRight(i, j);
12            solveBottom(i, j);
13            solveBottomLeft(i, j);
14            solveLeft(i, j);
15            solveUpLeft(i, j);
16        }
17    }
18
19    Instant end = Instant.now();
20    Duration timePassed = Duration.between(start, end);
21    System.out.println("Waktu eksekusi program: " + timePassed.toMillis() + " ms\n");
22 }

```

4. Prosedur Pencetakan (Output) Solusi Puzzle

```

1 public static void solvePuzzle() {
2     int i, j;
3     System.out.println("\nSolving...\n");
4     Instant start = Instant.now();
5
6     for (i = 0; i < matrix.size() && !words.isEmpty(); i++) {
7         for (j = 0; j < matrix.get(0).size() && !words.isEmpty(); j++) {
8             solveUp(i, j);
9             solveUpRight(i, j);
10            solveRight(i, j);
11            solveBottomRight(i, j);
12            solveBottom(i, j);
13            solveBottomLeft(i, j);
14            solveLeft(i, j);
15            solveUpLeft(i, j);
16        }
17    }
18
19    Instant end = Instant.now();
20    Duration timePassed = Duration.between(start, end);
21    System.out.println("Waktu eksekusi program: " + timePassed.toMillis() + " ms\n");
22 }

```

5. Prosedur Penyelesaian *Puzzle* dan Pencetakan (*Output*) *Puzzle* Dalam Delapan Arah

a. Arah Atas

```
1 public static void solveUp(int i, int j) {
2     if (!words.isEmpty()) {
3         ListIterator<String> iterator = words.listIterator(0);
4         while (iterator.hasNext()) {
5             String word = iterator.next();
6             if (i >= word.length() - 1) {
7                 int k = 0;
8                 int idx_i = i;
9                 boolean equal = true;
10                while (equal && idx_i >= 0 && k < word.length()) {
11                    equal = matrix.get(idx_i).get(j) == word.charAt(k);
12                    comparisonCount++;
13                    idx_i--;
14                    k++;
15                }
16                if (equal && k == word.length()) {
17                    ans[ansIdx] = word;
18                    ansCoord[ansIdx][0] = i;
19                    ansCoord[ansIdx][1] = j;
20                    ansCoord[ansIdx][2] = 1;
21                    ansIdx++;
22                    iterator.remove();
23                }
24            }
25        }
26    }
27 }
28
29 public static void printUp(int idx) {
30     String word = ans[idx];
31     int k = word.length() - 1;
32     for (int idx_i = 0; idx_i < matrix.size(); idx_i++) {
33         for (int idx_j = 0; idx_j < matrix.get(0).size(); idx_j++) {
34             if (k >= 0 && idx_i > ansCoord[idx][0] - word.length() && idx_i <= ansCoord[idx][0] && idx_j == ansCoord[idx][1]) {
35                 System.out.print(word.charAt(k));
36                 k--;
37             }
38             else {
39                 System.out.print("-");
40             }
41             System.out.print(" ");
42         }
43         System.out.println();
44     }
45     System.out.println();
46 }
```

b. Arah Atas Kanan

```
1 public static void solveUpRight(int i, int j) {
2     if (!words.isEmpty()) {
3         ListIterator<String> iterator = words.listIterator(0);
4         while (iterator.hasNext()) {
5             String word = iterator.next();
6             if (i >= word.length() - 1 && j + word.length() - 1 <= matrix.get(0).size()) {
7                 int k = 0;
8                 int idx_i = i, idx_j = j;
9                 boolean equal = true;
10                while (equal && idx_i >= 0 && idx_j < matrix.get(0).size() && k < word.length()) {
11                    equal = matrix.get(idx_i).get(idx_j) == word.charAt(k);
12                    comparisonCount++;
13                    idx_i--;
14                    idx_j++;
15                    k++;
16                }
17                if (equal && k == word.length()) {
18                    ans[ansIdx] = word;
19                    ansCoord[ansIdx][0] = i;
20                    ansCoord[ansIdx][1] = j;
21                    ansCoord[ansIdx][2] = 2;
22                    ansIdx++;
23                    iterator.remove();
24                }
25            }
26        }
27    }
28 }
29
30 public static void printUpRight(int idx) {
31     String word = ans[idx];
32     int col = ansCoord[idx][1] + word.length() - 1;
33     int k = word.length() - 1;
34     for (int idx_i = 0; idx_i < matrix.size(); idx_i++) {
35         for (int idx_j = 0; idx_j < matrix.get(0).size(); idx_j++) {
36             if (k >= 0 && idx_i > ansCoord[idx][0] - word.length() && idx_i <= ansCoord[idx][0] && idx_j == col) {
37                 System.out.print(word.charAt(k));
38                 k--;
39             }
40             else {
41                 System.out.print("-");
42             }
43             System.out.print(" ");
44         }
45         System.out.println();
46     }
47     System.out.println();
48 }
49 }
```

c. Arah Kanan

```
1 public static void solveRight(int i, int j) {
2     if (!words.isEmpty()) {
3         ListIterator<String> iterator = words.listIterator(0);
4         while (iterator.hasNext()) {
5             String word = iterator.next();
6             if (j + word.length() - 1 <= matrix.get(0).size()) {
7                 int k = 0;
8                 int idx_j = j;
9                 boolean equal = true;
10                while (equal && idx_j < matrix.get(0).size() && k < word.length()) {
11                    equal = matrix.get(1).get(idx_j) == word.charAt(k);
12                    comparisonCount++;
13                    idx_j++;
14                    k++;
15                }
16                if (equal && k == word.length()) {
17                    ans[ansIdx] = word;
18                    ansCoord[ansIdx][0] = i;
19                    ansCoord[ansIdx][1] = j;
20                    ansCoord[ansIdx][2] = 3;
21                    ansIdx++;
22                    iterator.remove();
23                }
24            }
25        }
26    }
27 }
28
29 public static void printRight(int idx) {
30     String word = ans[idx];
31     int k = 0;
32     for (int idx_i = 0; idx_i < matrix.size(); idx_i++) {
33         for (int idx_j = 0; idx_j < matrix.get(0).size(); idx_j++) {
34             if (k < word.length() && idx_j < ansCoord[idx][1] + word.length() && idx_j >= ansCoord[idx][1] && idx_i == ansCoord[idx][0]) {
35                 System.out.print(word.charAt(k));
36                 k++;
37             } else {
38                 System.out.print("-");
39             }
40             System.out.print(" ");
41         }
42         System.out.println();
43     }
44     System.out.println();
45 }
```

d. Arah Bawah Kanan

```
1 public static void solveBottomRight(int i, int j) {
2     if (!words.isEmpty()) {
3         ListIterator<String> iterator = words.listIterator(0);
4         while (iterator.hasNext()) {
5             String word = iterator.next();
6             if (j + word.length() - 1 <= matrix.get(0).size() && i + word.length() - 1 <= matrix.size()) {
7                 int k = 0;
8                 int idx_i = i, idx_j = j;
9                 boolean equal = true;
10                while (equal && idx_i < matrix.size() && idx_j < matrix.get(0).size() && k < word.length()) {
11                    equal = matrix.get(idx_i).get(idx_j) == word.charAt(k);
12                    comparisonCount++;
13                    idx_i++;
14                    idx_j++;
15                    k++;
16                }
17                if (equal && k == word.length()) {
18                    ans[ansIdx] = word;
19                    ansCoord[ansIdx][0] = i;
20                    ansCoord[ansIdx][1] = j;
21                    ansCoord[ansIdx][2] = 4;
22                    ansIdx++;
23                    iterator.remove();
24                }
25            }
26        }
27    }
28 }
29
30 public static void printBottomRight(int idx) {
31     String word = ans[idx];
32     int row = ansCoord[idx][0];
33     int col = ansCoord[idx][1];
34     int k = 0;
35     for (int idx_i = 0; idx_i < matrix.size(); idx_i++) {
36         for (int idx_j = 0; idx_j < matrix.get(0).size(); idx_j++) {
37             if (k < word.length() && idx_j == col && idx_i == row) {
38                 System.out.print(word.charAt(k));
39                 k++;
40                 col++;
41                 row++;
42             } else {
43                 System.out.print("-");
44             }
45             System.out.print(" ");
46         }
47         System.out.println();
48     }
49     System.out.println();
50 }
```


e. Arah Bawah

```
1 public static void solveBottom(int i, int j) {
2     if (!words.isEmpty()) {
3         ListIterator<String> iterator = words.listIterator(0);
4         while (iterator.hasNext()) {
5             String word = iterator.next();
6             if (i + word.length() - 1 <= matrix.size()) {
7                 int k = 0;
8                 int idx_i = i;
9                 boolean equal = true;
10                while (equal && idx_i < matrix.size() && k < word.length()) {
11                    equal = matrix.get(idx_i).get(k) == word.charAt(k);
12                    comparisonCount++;
13                    idx_i++;
14                    k++;
15                }
16                if (equal && k == word.length()) {
17                    ans[ansIdx] = word;
18                    ansCoord[ansIdx][0] = i;
19                    ansCoord[ansIdx][1] = j;
20                    ansCoord[ansIdx][2] = 6;
21                    ansIdx++;
22                    iterator.remove();
23                }
24            }
25        }
26    }
27 }
28
29 public static void printBottom(int idx) {
30     String word = ans[idx];
31     int k = 0;
32     for (int idx_i = 0; idx_i < matrix.size(); idx_i++) {
33         for (int idx_j = 0; idx_j < matrix.get(0).size(); idx_j++) {
34             if (k < word.length() && idx_i < ansCoord[idx][0] + word.length() && idx_i == ansCoord[idx][0] && idx_j == ansCoord[idx][1]) {
35                 System.out.print(word.charAt(k));
36                 k++;
37             } else {
38                 System.out.print("-");
39             }
40             System.out.print(" ");
41         }
42         System.out.println();
43     }
44     System.out.println();
45 }
```

f. Arah Bawah Kiri

```
1 public static void solveBottomLeft(int i, int j) {
2     if (!words.isEmpty()) {
3         ListIterator<String> iterator = words.listIterator(0);
4         while (iterator.hasNext()) {
5             String word = iterator.next();
6             if (i + word.length() - 1 <= matrix.size() && j >= word.length() - 1) {
7                 int k = 0;
8                 int idx_i = i, idx_j = j;
9                 boolean equal = true;
10                while (equal && idx_i < matrix.size() && idx_j >= 0 && k < word.length()) {
11                    equal = matrix.get(idx_i).get(idx_j) == word.charAt(k);
12                    comparisonCount++;
13                    idx_i++;
14                    idx_j--;
15                    k++;
16                }
17                if (equal && k == word.length()) {
18                    ans[ansIdx] = word;
19                    ansCoord[ansIdx][0] = i;
20                    ansCoord[ansIdx][1] = j;
21                    ansCoord[ansIdx][2] = 6;
22                    ansIdx++;
23                    iterator.remove();
24                }
25            }
26        }
27    }
28 }
29
30 public static void printBottomLeft(int idx) {
31     String word = ans[idx];
32     int row = ansCoord[idx][0];
33     int col = ansCoord[idx][1];
34     int k = 0;
35     for (int idx_i = 0; idx_i < matrix.size(); idx_i++) {
36         for (int idx_j = 0; idx_j < matrix.get(0).size(); idx_j++) {
37             if (k < word.length() && idx_j == col && idx_i == row) {
38                 System.out.print(word.charAt(k));
39                 k++;
40                 col--;
41                 row++;
42             } else {
43                 System.out.print("-");
44             }
45             System.out.print(" ");
46         }
47         System.out.println();
48     }
49     System.out.println();
50 }
```

g. Arah Kiri

```
1 public static void solveLeft(int i, int j) {
2     if (!words.isEmpty()) {
3         ListIterator<String> iterator = words.listIterator(0);
4         while (iterator.hasNext()) {
5             String word = iterator.next();
6             if (j >= word.length() - 1) {
7                 int k = 0;
8                 int idx_j = j;
9                 boolean equal = true;
10                while (equal && idx_j >= 0 && k < word.length()) {
11                    equal = matrix.get(1).get(idx_j) == word.charAt(k);
12                    comparisonCount++;
13                    idx_j--;
14                    k++;
15                }
16                if (equal && k == word.length()) {
17                    ans[ansIdx] = word;
18                    ansCoord[ansIdx][0] = 1;
19                    ansCoord[ansIdx][1] = j;
20                    ansCoord[ansIdx][2] = 7;
21                    ansIdx++;
22                    iterator.remove();
23                }
24            }
25        }
26    }
27 }
28
29 public static void printLeft(int idx) {
30     String word = ans[idx];
31     int k = word.length() - 1;
32     for (int idx_i = 0; idx_i < matrix.size(); idx_i++) {
33         for (int idx_j = 0; idx_j < matrix.get(0).size(); idx_j++) {
34             if (k >= 0 && idx_j > ansCoord[idx][1] - word.length() && idx_j <= ansCoord[idx][1] && idx_i == ansCoord[idx][0]) {
35                 System.out.print(word.charAt(k));
36                 k--;
37             } else {
38                 System.out.print("-");
39             }
40             System.out.print(" ");
41         }
42         System.out.println();
43     }
44     System.out.println();
45 }
```

h. Arah Atas Kiri

```
1 public static void solveUpLeft(int i, int j) {
2     if (!words.isEmpty()) {
3         ListIterator<String> iterator = words.listIterator(0);
4         while (iterator.hasNext()) {
5             String word = iterator.next();
6             if (j >= word.length() - 1 && i >= word.length() - 1) {
7                 int k = 0;
8                 int idx_i = i, idx_j = j;
9                 boolean equal = true;
10                while (equal && idx_i >= 0 && idx_j >= 0 && k < word.length()) {
11                    equal = matrix.get(idx_i).get(idx_j) == word.charAt(k);
12                    comparisonCount++;
13                    idx_i--;
14                    idx_j--;
15                    k++;
16                }
17                if (equal && k == word.length()) {
18                    ans[ansIdx] = word;
19                    ansCoord[ansIdx][0] = i;
20                    ansCoord[ansIdx][1] = j;
21                    ansCoord[ansIdx][2] = 8;
22                    ansIdx++;
23                    iterator.remove();
24                }
25            }
26        }
27    }
28 }
29
30 public static void printUpLeft(int idx) {
31     String word = ans[idx];
32     int row = ansCoord[idx][0] - word.length() + 1;
33     int col = ansCoord[idx][1] - word.length() + 1;
34     int k = word.length() - 1;
35     for (int idx_i = 0; idx_i < matrix.size(); idx_i++) {
36         for (int idx_j = 0; idx_j < matrix.get(0).size(); idx_j++) {
37             if (k >= 0 && idx_j == col && idx_i == row) {
38                 System.out.print(word.charAt(k));
39                 k--;
40                 col++;
41                 row++;
42             } else {
43                 System.out.print("-");
44             }
45             System.out.print(" ");
46         }
47         System.out.println();
48     }
49     System.out.println();
50 }
```

C. Screenshot dari *Input* dan *Output* Program

a. Small

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

ETARETILLITC
SGWKSYOJCGFE
TEURUEZRZJLI
HSITLLODIEKXI
JTEZYLTUATQK
ESNLERLYRICK
PSEGLGEUFTGR
OTCNGURHJOBL
LRNAONFCCDHO
IEKFPPBIEJTEM
TBVKPINTLFUY
ILHPDOSITIOZB
CETKSAVTHADS
SSVTYLD RUSBA

ABSURDLY
BAITING
BUTCHERY
DOLEFULLEST
DOLTISH
ESCAPIST
ILLITERATE
OBTUDES
POLITICS
SEIZE
SHINBONES
TAUTLY
TORE
TREBLES

- **Output 1**

Masukkan nama file yang berada di folder test: small1.txt

Solving...

Waktu eksekusi program: 5 ms

Jumlah perbandingan huruf: 3859

[illegible]

- **Input 2 (small2.txt)**

ADVISORS
AFRAID
AUDITS
CONGEAL
DETERRENT
DONNED
HABITABILITY
NUTRIENT
OVERHEAD
SCUMMING
SHORTING
STROKE
SULKIER
SWEETIE
THROW
WETNESS

• Output 2

```
Masukkan nama file yang berada di folder test: small2.txt
Solving...
Waktu eksekusi program: 8 ms
Jumlah perbandingan huruf: 5125
```

NUTRIENT

N U T R I E N T

SWEETIE

S W E E T I E

SHORTING

S H O R T I N G

DETERRENT

T N E R R E T E D

WETNESS

W E T N E S S

CONGEAL

C O N G E A L

DONNED

D O N N E D

SCUMMING

S C U M M I N G

STROKE

E K O R T S

THRONG

T H R O N G

HABITABILITY

H A B I T A B I L I T Y

AFRAID

D I A R F A

ADVISORS

S R O S I V D A

OVERHEAD

D A E H R E V O

AUDITS

S T I D U A

SULKIER

S U L K I E R

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

PERPETUALLYDYNACX
QUYXZHGUVNUXIDMRJ
LDLWGHSMRQJKTQBM
LZXIRBZIKKVJKEZH
JRSGLBANOSAESNU
AZSREPPACIDNAHTU
YLNRETTALSFVGVWCU
ZDIQSSVUVSXEEJXXA
FRETPENYZDWHRYQV
MIJVNZOZEROTICISM
VCDLIGJIPAMRSOOI
DETUXAUISSWNEHQS
MUQVPBWKRAINUNEL
EYLRAEPLUNHDAVZSD
UVILAKLAWMUPDRZK
PALLADIUMGIIMUYJ
GLOWERBWT CERXEIP
YSTRANDSSUSCFUJI

ALKALI
CANDY
DISPENSERS
DULLARD
EMPHASIZES
EROTICISM
GLOWER
HANDICAPPERS
PALLADIUM
PEARLY
PERPETUALLY
REFINISH
SLATTERNLY
STRANDS
SWISHED
UNSEASONABLE
WAIVED
WARY

- **Output 3**

PERPETUALLY P E R P E T U A L L Y	UNSEASONABLE E L B A N O S A E S N U	SLATTERNLY Y L N R E T T A L S	REFINISH H S T I F E R
CANDY - - - - Y D N A C	HANDICAPPERS S R E P P A C I D N A H	SNITCHED S W I T C H E D	EROTICTISM E R O T I C I S M

WAIVED

D E V I A W

PEARLY

Y L R A E P

EMPHASIZES

S E Z I S A H P M E

STRANDS

S T R A N D S

DISPENSERS

S R E S N E P S I D

ALKALI

I L A K L A

PALLADIUM

P A L L A D I U M

GLOWER

G L O W E R

DULLARD

D U L L A R D

WARY

Y R A W

b. Medium

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

```
1 LCSPQVERIFICATIONP
2 VAXEZHUGSJTBUEGMC
3 QBYYTHFDSBNXNCLLL
4 PNRKYHPEENLTUORYAO
5 MHJGWPPENHOBVNZBDR
6 KDBBAYRADASIBCAIEL
7 OPRERARIRIWNTRBELCM
8 RRRFVNQACGHTOCGGUT
9 HXSTOIGNCEOENRUSEI
10 NKDLYNRBUSYINGADOJ
11 DXKEKFLNIBBBIU8EP
12 LSQSTQLTCDDGTOAVDR
13 FEAXONTWNTFAGOTEDD
14 SCNSTELLATIONSUUV
15 ROMNCLASSYVEKFXNAL
16 MNMPKOWSAPIRHSJCB
17 TDXOFBADAERLUVRAY
18 PLWZVVGOFUVVLIUNT
19 DYFHLNCRZYZFBADID
20 GFZEREVOCATIONIGPZ
21
22
23 ASSENTED
24 AUTOBIOGRAPHIES
25 BARONS
26 BUSTLES
27 BUSYING
28 CLASSY
29 CONJELLATIONS
30 DIURETIC
31 FANGOTED
32 GALLIVANTING
33 HEEHAWING
34 IMPEDIMENTA
35 PATCFY
36 READABLE
37 REDUCTIONS
38 REVOCATION
39 ROUTINE
40 SCARY
41 SECONDLY
42 VERIFICATION
```

- **Output 1**

Masukkan nama file yang berada di folder test: medium1.txt

Solving...

Waktu eksekusi program: 9 ms

Jumlah perbandingan huruf: 16176

```
READABLE
-----
- E L B A D A E R -
-----
DIURETIC
-----
- C -
- I -
- T -
- E -
- R -
- U -
- I -
- D -

REVOCATION
-----
- R E V O C A T I O N -
-----
GALLIVANTING
-----
- G -
- N -
- I -
- T -
- N -
- A -
- V -
- I -
- L -
- A -
- G -
```


VERIFICATION
- - - - - V E R I F I C A T I O N -

HEEHAWING

PRICEY

BUSYING

BUSYING

IMPEDIMENTA

ROUTINE
- - - - -
- - - - -
- - - - -
- - - ENITUOR - -

BEETLES

SCARY

Y
R
A
C
S

BARONS

REDUCTIONS

S
N
O
I
T
C
U
D
E
R

CONSTITUTIONS

Search (CUI + SMt + F)

CONSTITUTIONS

AUTOGRAPHIES

S
E
I
H
P
A
R
G
O
I
B
O
T
U

SECONDLY

FAGOTED

CLASSY

CLASSY

ASSENTED

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

```
1 ZZYLCVFFLEABHOJAEQUA
2 XYLBLSNDHFCQXKHNFPPI
3 TQQSIAARBNUZAFHACUSB
4 XUHOBMDIYLSIKTEEONBQ
5 FSNMLYPNRTTUSQ CZUNMF
6 WICEGQZEEVRDLIHRRTOS
7 YYBDQVBQCDOPUDGALFM
8 CHXAJUBYBUDYLYXNGLDN
9 EYSEBRIVPJNAAIVTIILP
10 IZRUSIKVNLKIDNBANRMS
11 XGBLWORLDOPFOUTEGGGL
12 MIRXYXICGCPAAULQLPSH
13 MCDEBTRCSNARTZSTYLVN
14 KMWPCETVEHLYONESOED
15 TEODSOYNQHDAMPNESSPR
16 GNBUSSEKWFRTXSDDOUN
17 MDVNXMMQPTSTRBPFEDJ
18 USSSDNSLDAHOPNIHXRRC
19 UDPYMMZDHDSEWUEABGC
20 IOCIPTSCITNAMESDRXDX
21 KLYLSUOEIXONBOVPKOPIK
22 LCDGHBDENOSAESCIFPOD
23
24 ADDENDA
25 ADULTS
26 ANTHEMS
27 CATCHALLS
28 CLAIRVOYANT
29 CREEDENZA
30 DAMPRESS
31 DOSES
32 ENCOURAGINGLY
33 FINESTOZING
34 FUNNELTING
35 HALIER
36 IMPECUNIOUS
37 INDESCRIBABLE
38 LIBELLER
39 OBNOXIOUSLY
40 POMADES
41 SEASONED
42 SEMANTICS
43 SKEMER
44 TRANSCRIPTED
45 UNEQUIVOCAL
```

- **Output 2**

```
Masukkan nama file yang berada di folder test: medium2.txt

Solving...

Waktu eksekusi program: 23 ms

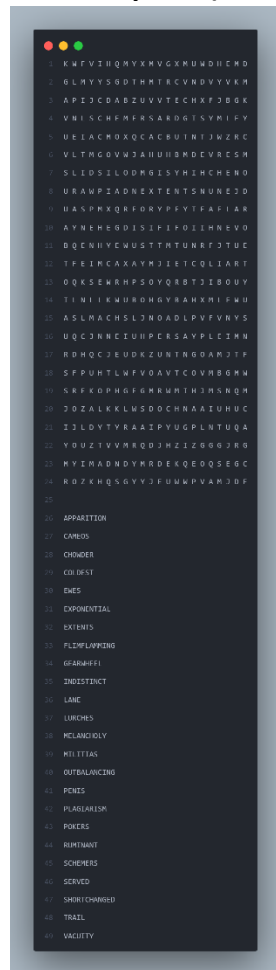
Jumlah perbandingan huruf: 23853
```

CLAIRVOYANT C A I R V O A N T	CATCHALLS C A T H A L L S	UNEQUIVOCAL U N E Q U I V O C A L	ADDENDA A D D E N D A	DAMPNESS D A M P N E S S	DOSES D O S E S	SEMANTICS S E M A N T I C S
INVESTING I N V E S T I N G	PANELLING P A N E L L I N G	LIBELLER L I B E L L E R	TRANSCIBED T R A N S C R I B E D	SEMER S E M E R	POPPIES P O P P I E S	OBNOXIOLUSLY O B N O X I O L U S L Y
ENCOURAGINGLY E N C O U R A G I N G L Y	SPECULOUS S P E C U L O U S	ADULTS A D U L T S	HAIRIER H A I R I E R	UNDESCRIBABLE U N D E S C R I B A B L E	CREDENZA C R E D E N Z A	ANTHERS A N T H E R S

SEASONED

-----DENOSAES-----

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



```
1 KMFVTHQRYXKVGXHMUDHCND
2 GLMYYSBDTHMTRCVNDVYVKH
3 APIJCDABZUVVTECHXFJBGK
4 VNIKCHMERSARDDTSYMIKY
5 UETACKOXCACRUTNTJWZRC
6 VLTNGOVWJAHUUSNDEVRECH
7 SLIDSILODRBISTYHCHERO
8 URAMPIDNREXTENTSNUNEJD
9 UASPMXKRFORYPVYTFAFIAR
10 AYNEHEBDSIPFPOIHNREVO
11 BQENHYCWUSTTRTUNRTJTUC
12 TFEIMCAXXYPJERTCOLIART
13 OQKSEWRHPSOYQRBTTIBOUY
14 TIMTIEWDRONGYRAHMTFWU
15 ASLMACHSLJNDADLPVFNYS
16 UQCJHNCZUHPCESAYPLCIHM
17 RDNQCJEUOKZUNTHOAKJTF
18 SFPUNTLWVVOAVTCOVNBGMH
19 SKFKOPHIGAKKUMTHJMNQHM
20 DZALKKLSDDOCHNAAIUHUC
21 IZLDYTYRAAIPYUGPLNTUQA
22 YOUTZVVRKQDJHZIZGGGJRG
23 MYINADNDYMRDEKQEQQSEGC
24 ROZKHQSGVYJFUMWPMVAMJDF
25
26 APPARITION
27 CARLOS
28 CHUNDER
29 COLEST
30 DUES
31 EXPONENTIAL
32 EXTENTS
33 FLIMFLAMING
34 GRABOFFI
35 INDISTINCT
36 LAME
37 LURCHES
38 MELANCHOLY
39 MELITIAS
40 OUTBALANCING
41 PENIS
42 PLAGIARISM
43 POICES
44 RHYTHMANT
45 SCHEMERS
46 SERVED
47 SHORTCHAINED
48 TRAIL
49 VACUITY
```

- **Output 3**

Masukkan nama file yang berada di folder test: medium3.txt

Solving...

Waktu eksekusi program: 31 ms

Jumlah perbandingan huruf: 22414

[illegible]

c. Large

- Input 1 (large1.txt)

```
1 0NITAREVESSAHMCOLANDERSCUSGPNR
2 DCODEBAUCHESISSUDYTHIVNTZIAO3H
3 SEICTEPHLLZZKJLIRELHASILECHLWI
4 PMNHAYTIRCOIDEMVDTHOTOSOTISM7TZ
5 IAMOPREDIUGSIMPQADACNL6ROAISTO
6 KLDHDAFACKERKKKXETAJHALIDNNHYM
7 ELQSYIRDEHQUULPUQZACLIHSJETIPE
8 TEACWUSGRFFORMULATEWUEHPRHYRDEJ
9 STUDDMEOGNIREVILSZMIVDSMDASFI
10 SFTONTSPSFNCGATHAVOVNLEKTHUUAO
11 BFCNCQQUATARMALWPSNR7FDZMTTXCB
12 TKMTDVRBYGTFH7ZVZKPIKOTH7TRFX
13 EPIIVRVSKN3LCJGCTNGTXYVNSFV7TJ
14 BCIAVRZDDBVFCDDUFECCJORDRHCNSRK
15 SUKQTFYLOXQENWCNRRJAFPTVYSNEVG
16 KVMGGPBCDXQBWEDCDLQKLCIYZHLDZ
17 ENNHRMXZLZBRICQEQXYBZKXQSZTFV
18 QNEDDHMPSNUTYLSERARMYKOKHRUMRAM
19 QXZUHFYQQPCMDXKNNRGSYXDTUTOVYI
20 LGGQMRZKXNDFXXQKLSHYKZYFUDXEN
21 NEKUVRVMARXIZAYBFUKPRCOEXHMKIK
22 POVRVMWAYEANDIJKCHGGLCOLDVLQNA
23 YVKNFXXHAFPCSPJSPAOPQLCARRYAZZ
24 PHZDUMFYHQXVIXSORPSNWCXKQILKHM
25 MQYLYLZLHFRBDGKXLPNOCLARGUOPHH
26 XDRNIFRBASATHSSVTAIQZFTCRVXUSQ
27 RSHDNVVKSNWQQGGNDVFXLRPVZCFDHIV
28 YTG3HQNVOPGIMPCLONNHSTDWNDZERP
29 TUUFHBCVOCHXEMYPHXJYFCDYLRNKVQ
30 AFDKXRMAIRDUTYTLJTXIMQVMMQDPT
31 BYYNUPKMMXDNRYUUXLNGGNZZBTJNY
32 XHNUZDBZAK3BNQPQAXIXNZRQQZNCJH
33
34 ASSEVERATING
35 ATAVIZH
36 BACON
37 CADOSH
38 COLANDERS
39 CURTALLED
40 DEBAUCHES
41 DISORDERS
42 FESTOONED
43 GARNISH
44 TIGERMAN
45 I SWET
46 MEDIOCRITY
47 MISGUIDE
48 NOTTIVE
49 OCEANOGRAPHIC
50 PACKER
51 FLOURISH
52 PRESETS
53 REFORMULATE
54 RHIZOME
55 SALT
56 SLIVERING
57 SNEERCHO
58 SPINDRIPS
59 SPIRITS
60 TRIUMPHANTLY
61 TROLLS
62 TYPEFACE
63 VANTAGES
64 MALLET
65 MELDING
```

- Output 1

```
Masukkan nama file yang berada di folder test: large1.txt

Solving...

Waktu eksekusi program: 27 ms

Jumlah perbandingan huruf: 22633
```

ASSIMILATING
GNITAREVSSA

COLANDERS
COLANDERS

CURTALLED
C
U
R
T
A
I
L
E
D

DISCORDERS
D
I
S
C
O
R
D
E
R
S

SALT
S
A
L
T

CAPMENT
G
A
R
R
M
E
N
T

PHIZCHE
R
H
I
Z
O
M
E

DEBAUCHES
D
E
B
A
U
C
H
E
S

WOLLS
T
R
O
L
L
S

SPIRITS
S
P
I
R
I
T
S

LIMPET
L
I
M
P
E
T

WILLET
W
I
L
L
E
T

REDUCEDLY
V
I
R
C
O
I
D
E
M

SACON
N
O
C
A
B

REDUCEDLY
E
D
U
C
T
I
O
N

TYPEFACE
T
Y
P
E
F
A
C
E

PACKER

PERFECTIVE

POLISHED

ATAVISM

PANDISH

PEZOMAN

PRESLTS

REFORMULATE

WELDING

OLYMERING

STORCHED

SPINOFFS

FESTCONED

WINTAGES

TRELPHANTLY

OCTAOCOGNAPHIC

- **Input 2 (large2.txt)**

```
1  QSD EHSUBCGBSSEBIRCSMUCRICSCEXEJJ
2  TNLFIQCHHOAHRRTNERSARRADHEKVFYA
3  DSTWOGDOABSRTEFFVITLNETSAHYGNATI
4  REITAABLBTESUUDMSINAKRATINAMUHL
5  ESTNAJMATLAHPPOGNSOHLASSOEDMUS
6  AJTAOILREERSTBRMAAKESDHMLITIAGN
7  LGJRRITCSIRDSKDVUNETCTCHTELPPTRJN
8  JVQHAETQSSIULMRNLOLSIMANNSESUOSG
9  SLDPONCRSRZREOLLLSTEYPOEWETZHBCQ
10 TMLXHGADSEESKHAUQSTBBHSHTGNERTS
11 IQPAHEBIMTASWIMMERSAODKTCQTPOPXH
12 CQFDPLOTENKFPQILSDDVPJCMLOONJQFPF
13 ABAJVVUERBBOQJSOBMVIMWJGFOPUZIAP
14 LWPZZCPLVCJACVBMYSTZUIOUCSTEBVB
15 LIEGGQRDHTDCSHUFUMNSBTJOLNEOURPA
16 YJZLMHMLKFMZDZBMXCZJMTLCLVFGEXX
17 WURHRMISYXMUJLYNJTHEDHSFMMPKOBBP
18 CJGCKFXHVWZTUOBLSTSUGFMBHLHLYB
19 CBHMQFSYFMDVNGFYPTIVQWJCMXTEFYV
20 LDJBDPGGDEKRNKQXHEFPWRMTUJFUBJD
21 YUUAHVYIXXDJCKPDNQNRGTGNWPIYIAUG
22 KMXNTXRCFCRSDDRUCUASFJUXPNBNLUR
23 PMPUKSXVMNJLHOGAHLALEIVMFSAVDJJZ
24 XLYZZIVKZKIAUTVUPQZJVSUKKEWAZQFF
25 FYACOSFITIREQSTFMDGBEUUFFGXSAZUT
26 DMRHAYEHHPWUJMNCCWDCSXJIVFPLJQJK
27 SQRFWRVYVWVYTFCPGVVZBGQQTZDHCWGTZ
28 UELAVDNHNPJYUEJJKUAPPDQVNGYNSTJA
29 MUZLSDWNHSPPENQFOYEFFZINDHVKWFE
30 IGQXSUVVLKTVZFQBFISPGXTWHENUCZF
31 CVNPKFZNCCJVVJSEKPCGCFUDAWRXBN
32 HZDNVODNVMBANOMWLDGOTXCQBZKAZI
33 DRQVLIUOKLEJGHHFUCQBYTYSJIBTKUY
34 MSXEGYWDOSQDIUXVFWTECHTRANSZSLPRO
35
36 ASSOCIATING
37 AWLS
38 BASTARDIZE
39 BELAGUERS
40 BUSHED
41 BYSTANDERS
42 CHATTEL
43 CIRCUMSCRIBES
44 CONTORTIONIST
45 COTTOMMOUTH
46 EMBARRASSMENT
47 ENTRALS
48 EXECES
49 FOAM
50 GOBLERS
51 HASTEN
52 HUMANITARIANISM
53 JAILS
54 LASSOED
55 MACERATED
56 MILLITIA
57 MOLAR
58 REALISTICALLY
59 REASSURES
60 RIPPLE
61 SLURPS
62 SOURCES
63 SQUAKES
64 STETHOSCOPE
65 STRANGLER
66 STRENGTH
67 SUIZMEL
68 TANGY
69 TOOTHPICKS
```

- **Output 2**

```
Masukkan nama file yang berada di folder test: large2.txt

Solving...

Waktu eksekusi program: 37 ms

Jumlah perbandingan huruf: 29381
```


- Input 3 (large3.txt)

```
1 1SN5NR005DSJUMDIABETESRE5Y0RHPQ0H
2 SVCDAESNNRRLKVEHEFORTUNESBNW0ALURE
3 YE1BIL0N11E5EMFYI1LK6RMOEAKK1SAFAS
4 5N1TATMDARMTCUOKSP1HSDN1E1F1UCYFD1
5 5E11ASAD0H110R6GN0R6F1R1QXNRNNUR
6 SEHTITDLNCTTTERGGDEZIRUETSAPDS0TAA
7 KRRCULIELEFASFFBSEILL00WCLXLATONTI
8 NNETNHERTELGITIOR0K0YCAWQ83MVASMG1N
9 5NAAHUVDDITLHVLCBRVEKLFH2CYBCS10F
10 AKJBRPPLTHASATERYTUCRYC3PQNLRS0NG
11 QNNTMKPCRPTRNEFLVANLKKRFXPMYFSESJ
12 JAVNLKRAZLNUI0Z1K0VUAKRZ11CSH0HFI
13 Q3PRFXENSSCIASCQAKRCOTFUZ3USMGUJ20
14 M+VPHLYK6LZWEGZUQ0ACEJ1U0DXULVK
15 XVUHG8B1Z3UREZATTHFSYUHGSL52SGG0W
16 OKH1SK01CH1JNZHJ1W1UZYMHVVRKZVHQ
17 L1BH1WGLGSKPD1N0DZCULYBQCQ03WKKVZY
18 UYK1JVU1XX1KRWZ0RMB1M0WV1NS11ZXK1K
19 5UAVXNTTCYHVN1RUC0VPPATZHIW1JVB1LG
20 MPUP1PL0EPXV1NS1P1NH1KCHU1B1JTF1DDK
21 NKART1BK1FDCC0B3Q0SGN1UC3W0FLW1MRV
22 HZHTQ0YDEH1R1H1CE1F1R1ED1QBMV1ABG0H0P
23 C0KUE1333CTEYV1HH1CLAGDDBR25A0LMTAXYK
24 YK5S0FKQ0G6YV1XD1R1J1KKUGVY0CBUNHTL6S
25 HP1XNM1DR1UQ1L0MEY1YD1R1MS1ZSHS1E1U1P
26 TR5XK1FK1EY1V0H0E1N1HXQ0K0X30N2FXFD
27 1R1PL1F1N1LL1M0R0R0H0R1FL1U0L1R1Y0R
28 BRK1CRQ33VZEZ1P1V1X1Y1CEV1CXD0C3H1N2H
29 11K1K1H1P1F1N1L1J1N1X1S1Q1K1J1R1V1O1S1R1N1S1C
30 XCZYH1XT0P1S1XV1H1N1CES1T001Z3D1O1GHJ
31 RZ1N1G1CUGY1Y1PDW1L1CFU11J1K1N1D1V1D1N1K1H1D1P
32 1T1M1F1N1H1X1L1M1F1Q1Q01N01W1V1E1M1D1G1E1Q1P1G1I1R1E
33 N1D1X1M1S1N1X1X1G1C1N1M1P1F1Y1G1J1X1H1Y1D1P1O1Q
34 CS1R1PGY1J3VY1R1B1Y1L1K1K1X1A1H1B1R1H1Y1B1K1W1MA
35 LQCE1L1J1N1A1P1S1U1Q1W1X1M1N1A1U1T1U1H1H1Y1V1F1B1S
36 XN1S1VY1M1J1T1CX1M1P1Q1B1W1EY1SD1R1U1L01W1D1H1E1Q1H1V1S
37
38 AUTH0R1T1A1T1V1S
39 BABES
40 BH1T
41 BL1ST1ER1NG
42 CO0G1R
43 COFF1NG
44 CON1ST1ELL1AT1ON
45 COUN1T1R1E1T1ERS
46 D1H0R1M
47 D1AB1ET1ES
48 D1SC0M1B1UL1ATE
49 ESC1AL1AT1ONS
50 FASC1STS
51 FORT1UN1ES
52 FRI1END1SH1PS
53 GAY1SR
54 GRADU1AT1ONS
55 GR1TT1Y
56 GRUEL1S
57 1N1F1H1T1T1HS
58 LAD1S
59 1F1Y1A1TH1ANS
60 PONG1R
61 PUT1Z1Y
62 PAST1C1R1L1Z1ED
63 PLAY1R00M1S
64 PUN1C1ES
65 QUE1EN1NG
66 R1F1ST1R1A1N
67 R1P1ER
68 ROT1UND1A
69 SAL1M0N1ELL1AE
70 S1P1W1N1G1S
71 S1P1E1AR
72 STR1AT1IF1D
73 M00LL1ES
```

- Output 3

```
Masukkan nama file yang berada di folder test: large3.txt

Solving...

Waktu eksekusi program: 25 ms

Jumlah perbandingan huruf: 30581
```


D. Alamat Drive

a. **Google Drive**

https://drive.google.com/drive/folders/1bAtV2jtMvsL_3yAKZbmqho4huTozp8GX

b. **Github Repository**

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

E. Tabel Ceklist

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